Mobile Platform For Wireless Grid Measurements

Master Thesis
Mobile Communications
Department of Electronics Systems
Aalborg University, 2009

Emanuele Eugenio Scuderi

Rocco Emilio Parrinello

David Lopez Izal

Aalborg University
Department of Electronic Systems
Fredrik Bajers Vej 7
9220 Aalborg Ø
http://es.aau.dk

Title:
Mobile Platform
for Wireless Grid Measurements

Theme:
Mobile Communications

Project Period:
10th smester, spring 2009

Project group:
Group 1116

Members of the group:
Emanuele Scuderi
Rocco Parrinello
David Izal

Supervisor:
Frank H.P. Fitzek

Number of copies: 6

Number of pages: 83

Attachments: 1 CD-ROM

Completed: June 1st 2009

Synopsis:

Nowadays, Wireless Networks can be found everywhere. In order to create new applications and make services better, it is essential to improve the performance of such networks. One way of doing this is using Principles of Cooperation. In fact, Cooperative Wireless Networks have experienced a remarkable development in the last years. In order to evaluate the performance of these networks it is necessary to measure the throughput, the delay of the communication, the battery consumption of the devices, etc.
However, in many measurement campaigns mobile devices used to be in a fixed position without moving around, therefore, the topology of the network does not change during the measurements, consequently, the results differ from reality.
Another approach is to make simulations, but in some cases they can only offer a good approximation of the simulated environment.
In this project we present the mobile platform we have designed to optimize measurements for mobile wireless networks with dynamic topology.

Preface

This report has been carried out by group 1116 during the 10th semester in Mobile Communication at the Department of Electronic Systems, Aalborg University, Denmark.

The aim of this thesis is to develop a mobile platform designed to optimize measurements for mobile wireless networks with dynamic topology.

The report consists of 6 main chapters and 4 appendixes. The contents, list of figures, tables and abbreviations are presented in order for the sake of reading convenience. The references are placed at the end of the report.

All the source code for programming the Opensensor and the Remote control application (GUI) for the PC are inserted in a the digital version of the report.

A CD containing the following can be found in the back of the report:

- The report in PDF form.
- The source code for programming the Opensensor.
- The Graphical User Interface (GUI).
- The Python codes to install on the Nokia N95 phones.
- The Nokia Energy Profiler application.
- Three demo videos showing the examples in Chapter 4.

Finally, we would like to express our gratitude to our supervisors Frank H.P. Fitzek and Gian Paolo Perrucci as well as Ben Krøyer, Anders Grauballe and Achuthan Paramanathan, who has helped with the Opensensors and the motion parts, and for their guidance to the successful completion of this project.

Aalborg University, June 1st 2009

Emanuele Scuderi	Rocco Parrinello	David Izal

Contents

Chapter 1

Introduction

1.1 Introduction and Motivation

Wireless Networks have experienced an extraordinary growth in the last years to become one of the most important segments of the telecommunications industry, partly due to a better technology and lower prices. Furthermore, the use of mobile phones and domestic wireless networks has also helped this development. Due to the increasing demand of new applications, better and cheaper services, etc, the field of wireless networks has become a very profitable field for the mentioned industry.

According to [24] in a few years more than half of the world population will have a mobile phone, and together with other wireless devices will make it possible to create a global network with tens of billions of additional wireless nodes [9].

The way to organize such a network can be complicated, for that reason, the wireless mesh networks have emerged as a good technology. Authors in [12] describe these networks as a communications network made up of radio nodes organized in a mesh topology. These kinds of networks are self-configured and dynamically self-organized [3], offering lower costs and lower power consumption, giving great possibilities for a future market where everything will be connected wirelessly [9].

Several research studies in recent years show that the performance of these kinds of wireless network can be improved by designing new protocols and services, and by using principles of cooperation [9] This makes it possible to save energy and improving network parameters such as throughput or transmission delay. Cooperative wireless networks attracts the interest from many researchers in the scientific and industry communities [4], [18].

Some examples of this can be found in [7] and [4]. In [7] using short range technology, mobile devices create cooperative clusters with neighboring devices and each one contributes to the cooperative cluster by sharing its cellular communication link. As a result, can be achieved a better use of the resources. In [4] is explained how the users, in a wireless communication, can increase their effective quality of service using cooperation.

Performance can be improved, for instance, in terms of energy consumption, datarate and Quality of Service [17], [23], [10]. A common way to quickly estimate the performance of a wireless network is to measure the throughput and delay of the communication, and also the power consumption of the devices. Some examples of this will be shown in subsequent chapters.

1.2 Problem Definition

In a mesh network the nodes are used to be static and do not experience mobility. In a wireless mesh network the nodes, i.e., mobile devices, can move around creating a dynamic topology.

Taking measurements for a wireless network with the devices in a fixed position, the results will differ from reality [11], because in the real world the mobile devices can change the position over the time.

To solve this problem, two solutions are possible. The first one is to use some sort of software to simulate cooperative wireless networks. In fact, the majority of the studies related to networks are based on simulations, that can only offer a good approximation of the simulated environment [9].

An example of these simulation tools is Netlogo [8], an integrated modeling environment (IME) which is free of charge.

Another solution is to make experimental measurements in a real environment. This is not easy to implement, but it could give more precise results.

1.3 Project Purposes

The main goal of this project is to develop a mobile platform, that will be able to optimize measurements for mobile wireless networks with dynamic topology. This platform has to be able to move around in a real scenario.

A success key for this project is to make public and open all the information for building, programming and using the developed platform to the scientific community.

For better understanding, we establish the grounds that will help to describe the platform in later chapters. From now on we will call the whole platform Robot.

1.4 Project Timeline

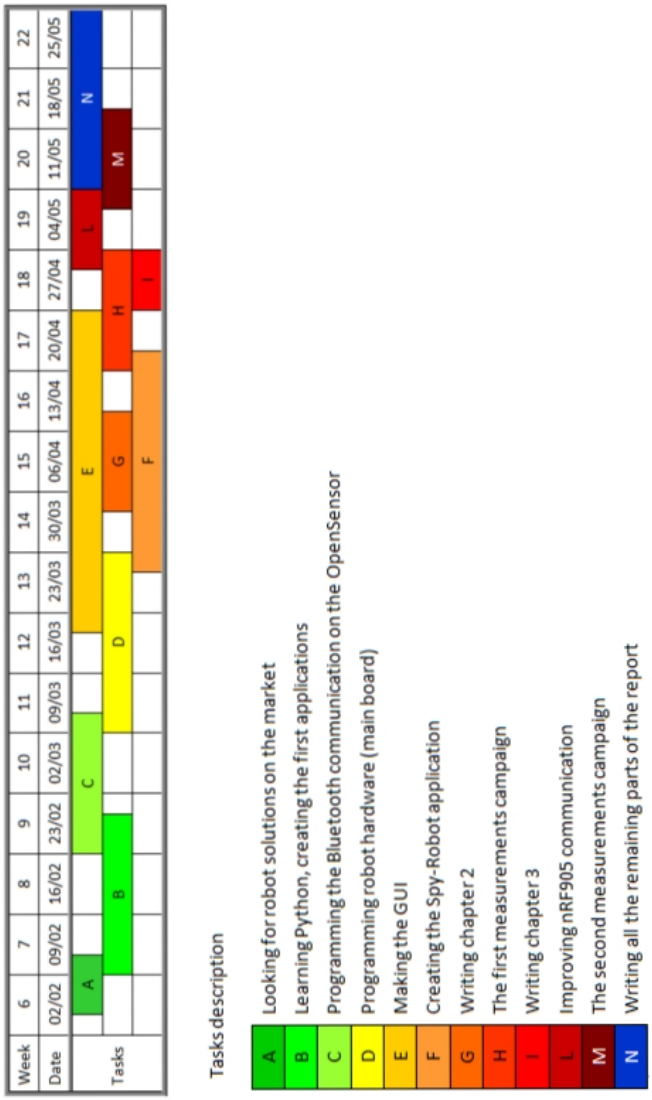

Figure 1.1: Timeline and Tasks description

Chapter 2

Analisys

In this chapter the necessary requirements that we need to carry out this project satisfactorily are specified. Furthermore, possible solutions that can fulfill these requirements will be presented.

This Robot is composed by two different parts: the Mechanical Block and the Controller Block. The first one will control the mobile parts of the Robot which consists some wheels, engines, a battery pack and a body to host the Controller Block. The latter will be the core of the Robot and it will be able to communicate with other devices.

2.1 Requirements Specification

In order to design the platform a series of requirements specifications must be specified. In the following subsections the requirements are divided into three main groups: Hardware, Software and Functional Requirements, which are also divided into subgroups in order to assure a correct description of all the desired requirements. The final platform should comply with all of them.

2.1.1 Hardware Requirements

Mechanical Block

- MB1: The mechanical block must have wheels
- MB2: It must have as many engines as many wheels
- MB3: The wheels must work on plain surfaces
- MB4: The mechanical block must be easy to use
  - MB5: It must be as small as possible
  - MB6: It must be as light as possible

- MB7: The mechanical block must have a battery pack
  - MB8: The battery pack must feed the engines as well as the controller block
  - MB9: The battery pack must be as small as possible
  - MB10: The battery pack must be possible to connect and disconnect the battery
- MB11: It must have a platform to host the controller block, sensors or any needed devices

Controller Block

- CB1: It must be able to control the engines
- CB2: It must be able to establish a communication with other devices
- CB3: It must have a LED to show the status

2.1.2 Software Requirements

- GUI1: The GUI must show IP Addresses, MAC Addresses and the names of the robots that are connected
- GUI2: The GUI must show which robots are connected and which ones are disconnected
- GUI3: The User must be able to control one or more robots
- GUI4: The User must be able to send a single command or a script with a list of commands
- GUI5: The User must be able to receive a video stream from the Robot
- GUI6: The User must be able to change the speed and the direction of the robots
- GUI7: The GUI must show an error message if one or more Robots are disconnected

2.1.3 Fuctional requirements

- FR1: The Platform must be able to communicate over Bluetooth with other devices
- FR2: The Platform must be able to communicate over nRF905 with other devices
- FR3: The Mechanical Block must be able to move forward, backward, left and right

2.2 Possible Solutions Available on the Market

This section will describe both Hardware and Software possibilities that are on the market. We can find a large number of components, but we will base our search on the ones that are within our acceptable budget, even though other options could offer more benefits.

By giving a short description of advantages and drawbacks, we will be able to take the choice that better fits with the requirements we described in the previous section. At first we detail the selections related to the Hardware, both Mechanical and Controller Blocks. After that we will present the ones referred to the Software.

2.2.1 Hardware

Mechanical Block

Nowadays it is quite easy to find hobby and professional robots, robots kits, components, etc, at any electronic shop.

Another component needed is the battery. This one is certainly one of the most important parts in any mobile device. The battery should feed all the circuits and engines. We can find a lot of different brands and models in the market; however, we need to choose a battery with enough lifetime, and with an appropriate size that will depend on the platform. The platform will be at first glance easy to get.

Controller Block

In this part we will review the different boards that we can find in the market to control the engines and to communicate with other devices over Bluetooth, nRF, etc. This is really important because it is the core of the Controller Block.

Arduino. Arduino, which is shown in the figure 2.1, is a physically open source computer platform based on a microcontroller board. It offers an environment which is developed for creating software for the board. Arduino platform can be a standalone device or it can be integrated with software in the computer [21].

Arduino can be used, for instance, to develop interactive objects or for taking information from a large variety of sensors. Also, it can control engines, sensors or any kind of devices that can be connected to the arduino board through a special interface [21].

Mainly the advantages of Arduino are: the hardware can be assembled by handwork or purchased preassembled; this means that Arduino Platform is very simple to build, and therefore the Arduino boards are relatively inexpensive compared to the other microcontroller platforms. The preassembled version of Arduino costs around $50.

Arduino has two different types of boards: Arduino Mini that adds a Mini USB interface, and Arduino BT that adds a Bluetooth interface. In case you need both interfaces, you have to buy both boards and this can result expensive.

Another drawback is that Arduino uses a Wiring implementation as a programming language. This is based on C and C++ languages but is still in development and needs a specific hardware to program the Arduino platform [21].

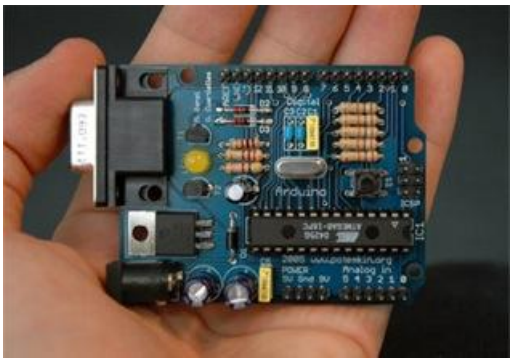

Figure 2.1: Arduino Board [21]

Basic Stamp 2. Basic Stamp 2, which is shown in the figure 2.2, is a microcontroller board developed by Palallax Inc. The name "Stamp "is because it is close to the size of a postage stamp [15].

There are eight different versions of the Basic Stamp module, but probably the Basic Stamp 2 is the most interesting model in a trade-off between performance and price. The price of this version is $49 and is cheap compared to the other microcontrollers.

Each BASIC Stamp comes with a BASIC Interpreter chip, internal memory (RAM and EEPROM), a 5 Volt regulator, a number of general-purpose I/O pins, a set of built-in commands and I/O pin operations. BASIC Stamp modules are capable of running a few thousand instructions per second and are programmed with a form of the BASIC programming language, called PBASIC [15].

The main advantage of Basic Stamp 2 is the use of EEPROM memory, which can be used for data storage and secure the program on the microcontroller even if the the alimentation is disconnect.

This microcontroller has a lack of wireless technology, and the Bluetooth and nRF905 modules are not embedded on the board. For that reason this board is not good for our project.

Figure 2.2: Basic Stamp 2 Board [14]

BL2100 Single Board Computer. Rabbit Inc. is one of the most important providers of 8 and 16 bits microprocessors, which have embedded control communications such as Ethernet connectivity. We will only analyze one of their microcontroller boards: the BL2100 [16].

The BL2100 Single Board, which is shown in the figure 2.3, is an integrated platform that gives the designers the possibility of controlling many devices over different interfaces.

The main advantage of this microcontroller is the 10Base-T Ethernet interface. This one uses a RJ-45 connector and almost 40 of sinking/sourcing digital I/O.

This board is one of the most completed boards in the market, but unfortunately it is very expensive; The price is around 280 Euro and there is a possibility by ordering it online [16].

Another important disadvantage is the lack of wireless technology which is extremely necessary for the development of our project.

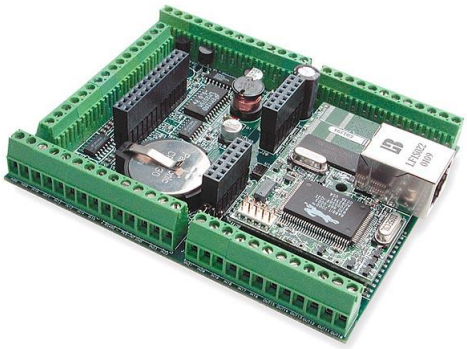

Figure 2.3: BL2100 Single Board Computer [16]

MINI-MAX/908-C. MINI-MAX/908-C, which is shown in the figure 2.4, is a microcontroller board developed by BiPOM Electronics [6].

The microcontroller is integrated with 512 Bytes of RAM, 33 general purpose I/O pins, two channel timer interface modules of 16 bits, 512 bytes serial EEPROM, 32 Kilobytes of In-System Re-programmable Downloadable Flash Memory and a RS232 Serial Port to program it.

In this microcontroller the lack of wireless technology is yet another problem. The price is high because the wireless device is not integrated in the board. It costs around $69.

Figure 2.4: MINI-MAX/908-C Board [6]

In order to summarize the feautures of the previous boards, a comparison requierements table is shown in figure 2.5

Requirements	Arduino	Basic Stamp 2	BL2100 SBC	MINI-MAX/908-C
CB1	✓	✓	✓	✓
CB2	✓	✓	✓	✓
CB3	✓			✓
FR1	✓			
FR2				
FR3	✓	✓	✓	✓

Figure 2.5: Comparison requirements table

2.2.2 Software

In the Market we can find several software programs to control different kinds of Robots both amateur and professional. We can describe some of them. In [13], a robot can be remote controlled from a computer to guard an area, e.g., a house or a shop. Besides, there are a lot of hobby robots that can be controlled with a very easy Graphical User Interfaces, but just with pre-programmed movements. An example of this are [5] and [22]

The main problem is that programs usually come with a specific hardware and cannot be reprogrammed to reproduce new movements. Another problem is that new features such as Bluetooth, wireless connection, etc, cannot be added. Some of them can be developed but always within the range that the manufacturer determines, e.g.,[22], which is based on Python, which permits to experiment with new possibilities.

For that reason, we consider to create a new Graphical User Interface customized for our hardware, with the consequent degree of freedom that this entails.

Chapter 3

Proposed Solutions

In this chapter we describe the solutions we consider are the best for our project. As we mentioned before, we are aiming to find the best quality-price ratio, but we also have to understand that time is an important aspect. To design and build a prototype of the robot is not an easy task, and the project has a limited time, for that reason, we weighed the pros and cons up carefully before deciding.

3.1 Hardware

3.1.1 Mechanical Block

After studying in close collaboration with the departments of Navigation & Communication and Electronic systems [25] the different possibilities are in the market, in this chapter the proposed solutions for the hardware part are presented .

Taking into account that the robot is going to be used in indoor environments, we do not need big and special wheels or very powerful engines. Thus, a micro robot may be more than enough for our needs. Therefore, the requirements from MB1 to MB6 are fulfilled.

Another point to discuss is the number of engines we need, and consequently the number of wheels. A robot can be built with only two wheels and two supporting ones. In that case we get the balance the robot needs and we only use two engines.

We have chosen is a micro robot chassis called Airat2 Micromouse Body Set [2], composed of two rubber wheels, two engines, two stabilizers and an aluminium body as shown in the figure 3.1

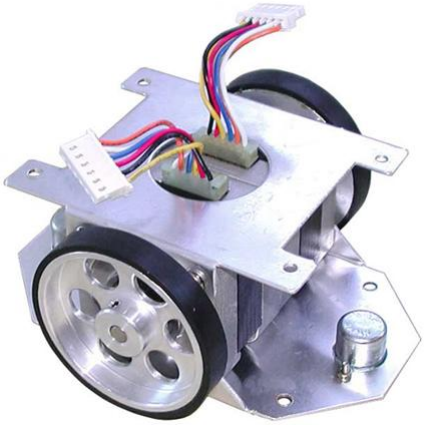

Figure 3.1: Airat2 Micromouse Body Set [2]

After choosing the platform that will determine the size of the Robot, we can focus on the choice of the battery. There is a trade-off between size and battery lifetime. Therefore, we use a 9 Volt battery, pack of six pieces of 1.5 volts, that anyone can buy at any shop, which allows us to use the robot with the necessary time to carry out the desired experiments. If more battery lifetime is needed, the battery can be replaced per another one. The battery pack used in this project is shown in figure 3.2.

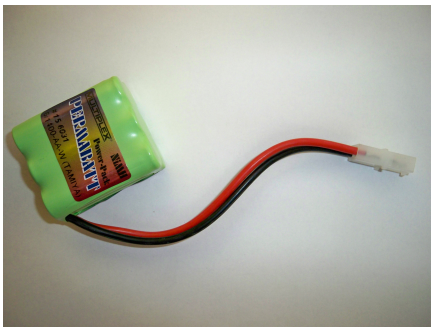

Figure 3.2: Battery Pack

A customized metal platform is difficult to find, so we decided to ask the Metal office Staff at Aalborg University to build one designed by us. The metal platform, shown in figure 3.3, is designed to host the controller block and is connected to the aluminium body with some screws over the engines of the Robot.

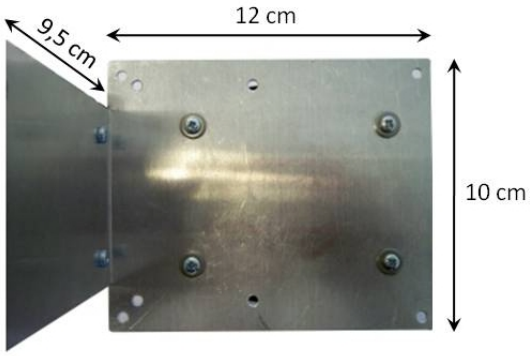

Figure 3.3: Metal Platform

3.1.2 Controller Block

As mentioned before, there are already many programmable circuit boards on the market. Nevertheless, we consider to use the Opensensor[19] for many reasons. This board has been designed by Mobile Devices Group at Aalborg University [26] and it is shown in figure 3.4.

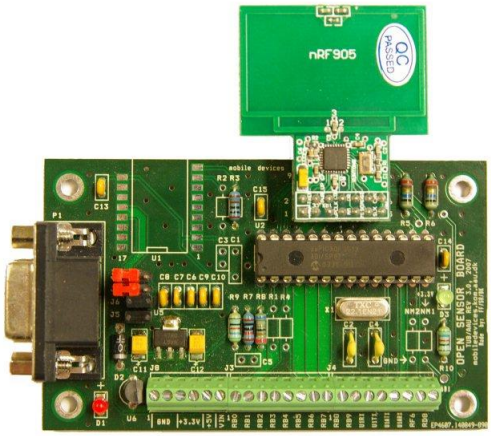

Figure 3.4: Opensensor [19]

On the other hand, we do not have to order them, we already have Opensensors in our department.

The Opensensor [19] is an open source sensor hardware platform, shown in the figure 3.4, and is the core of the Robot. It consists a main board with a processor and a transceiver board. The board has several interfaces to program the opensensor and to communicate with other devices. Some of these interfaces are the Bluetooth

interface, the RS232 connector and the nRF antenna, which are plugged on the main board.It is also possible to plug different sensor extension modules on the same hardware [1].

The movement of the Robot is controlled by a Microprocessor from *Microchip* which is a 16-bit Digital Signal Controller (DSC).

The most important features of this Microprocessor are:

- 256 KB Flash Memory
- 30 KB RAM
- 85 programmable digital I/O pins, 100 pins in total
- Two (Universal Asynchronous Receiver Transmitter) UARTs
- Two (Serial Peripheral Interface) SPIs
- Nine 16-bit timers
- C compiler optimized instruction set, 83 instructions

The Opensensor is directly connected to the main board (figure 3.5, which control the engines of the robot over some pins. The pins from RB1 to RB4 allow the movements of the left engine, and the pins from RB5 to RB8 allow the movements of the right one. The pin RB0 is used to manage the Green LED on the Opensensor which flickers when the robot is moving or when it is receiving some kind of data.

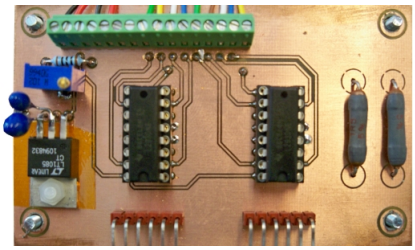

Figure 3.5: Main Board

The Opensensor needs a 9 Volt battery to work as the engines, and it has a series of pin where some external devices can be connected, e.g., a light or a camera. It is a highly integrated device that can start to operate when the power is supplied. Finally the processor is very easy to use for the developer. For instance this can be an in-circuit serial programming feature (ICSP) that allows the programmer to control the program on the processor in operation. The programming language used to program the Opensensor is C.

3.2 Software

3.2.1 Programming Languages

In this project we have to program both the opensensor and create applications for the PC. In the following, two programming languages are shortly described in order to justify our choice.

Python

Python is the most suitable language to program the applications we need. The main advantage of Python is that its scripts need just an interpreter to run. This programming language allows to create modular applications which can be run in many devices. Furthermore, we can create easily powerful applications with a friendly user approach.

Besides, it has a lot of libraries which allow the user to face a large variety of problems, creating a good application for each different scenario; even it is possible to build, additional needed libraries using C/C++; Besides, the standard module, which contains many standard libraries ,e.g., time, thread, socket, is needed for all the basic operations, e.g., threading, establishing a connection. Some extra modules can be really useful for the developer. One of them is the Python Imaging Library (PIL), which allows the user to program graphical environments.

C Language

C is the most common choice when a hardware has to be programmed, because it uses concepts and objects close to the hardware; for example some instructions can be translated directly into the assembly language by using a single command (like the increment operator). In any case, it has the logical structures of a high-level programming language; although the syntax sometimes is a bit complex and hard, the C is one of the most powerful languages.

This programming language compiles the scripts for a specific Operative System that gives a great effieciency to the applications. Furthermore, it fulfills the requirements FR1, FR2 and FR3.

3.2.2 Modules

In the following the modules related to the PC are explained. A representation of these modules are shown in the figure 3.6.

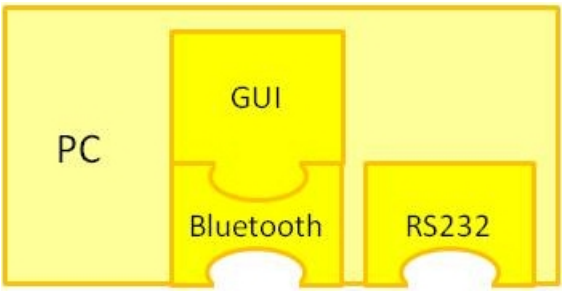

Figure 3.6: Modules of the PC

Graphical User Interface of the PC

The final purpose of this Graphical User Interface, written in Python 2.6, is to manage all the robots at the same time from a single PC. Every robot is described by its Name, its IP address and the MAC address of the opensensor. A screenshot of the GUI is shown in the figure 3.7.

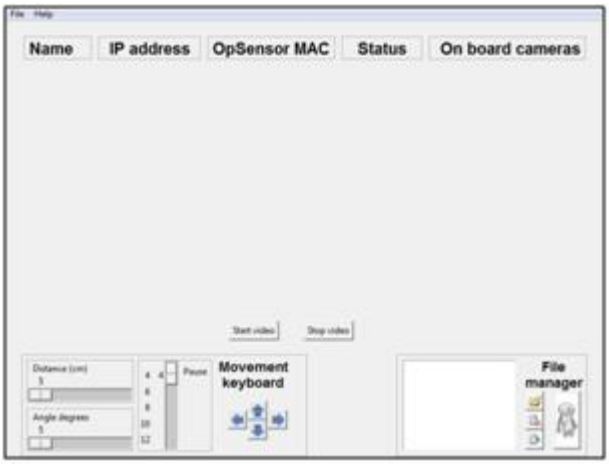

Figure 3.7: Graphical User Interface

In this interface a process is always running. This process listens to new possible incoming clients. When a new robot is connected, the process receives all its data (Name, IP address and opensensor MAC address); once this handshaking phase is completed, the robot‘s data displays on the GUI, so the requirement GUI1

is fulfilled, and at the same time a new element, containing all these data, is created in the Tuple called ROBOTS. A zoom of the GUI when a robot is connected is shown in figure 3.8

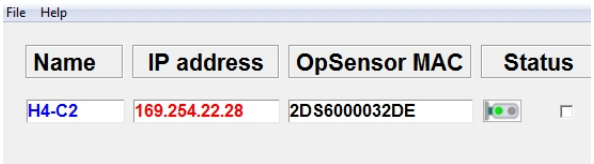

Figure 3.8: Zoom of the GUI when a robot is connected

Each element of this Tuple in addition to Name, IP and MAC addresses, also contains the value of the related checkbox, the position in the GUI. Therefore, the Tuple‘ s structure can be represented as in figure 3.9

Figure 3.9: Logical Tuple structure

In this way, a list with all the connected robots are displayed on the GUI. The user can send moving commands to each one of the robots, assigning the commands by checking the related box in the *Status column*, fulfilling the requirement GUI3. Thanks to the ROBOTS tuple the software can easily find the IP addresses related to the *checked boxes*, just making the loop cycle.

```
def get_ip(self, command):
 i=0
 counter=0
 while(i<len(ROBOTS)):
  if((ROBOTS[i][3].get())==1):
   counter=counter+1
   if(online_list[i]==0):
    self.offline_robot()
    break
   outsock.sendto(command, (ROBOTS[i][1], 8100))
   print(command+" sent to "+ROBOTS[i][0])
  i=i+1
 if(counter==0):
  self.no_robots()
```

When the function seen above is invoked, it receives a string containing the moving commands; in that moment, it starts to examine every *checked box*. Once an enabled *checked box* is found, the integer counter increases itself, and the string

command is sent through the output socket. This function also provides some exception control; if no *checked box* is found, the integer counter remains equal to zero, and the `no_robots()` function is called to show an error message as shown in the figure 3.10.

During the loop, the `online_list` shows the connections of each checked robot. If a selected robot is disconnected, the `offline_robot()` function is called, and as a result, an error message appears on the screen; in that case the requirement GUI7 is fulfilled.

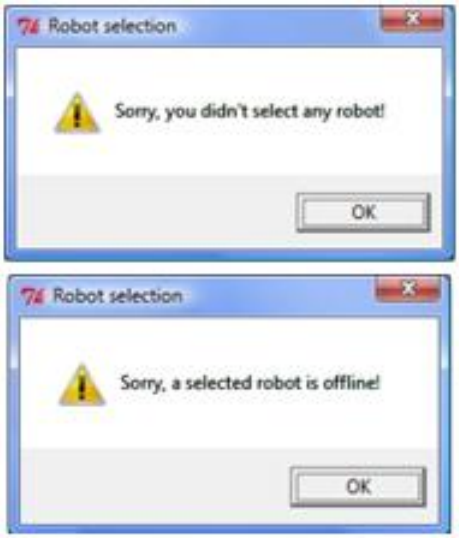

Figure 3.10: Error Messages

When a button in the *Movement keyboard* is pressed, the robot has to execute the moving commands contained in the string command. Every time one of the arrow keys is pressed, the related moving function is called, e.g., if the forward key is pressed, the Robot will move in the same direction as the key indicates.

```
def move_f(self):
 cmd=("Move#F#"+pause+"#"+dist+"#")
 self.get_ip(cmd)
```

After that, the `get_ip` function is called and it sends a command string, as shown in the code above, containing all the needed parameters: direction, pause, distance and degrees (this one only in rotation movements). These parameters are taken from the current values in the related scales, which allow the user to send the needed command. The requirement GUI6 is fulfilled. This can be done using the movement keyboard shown in figure 3.11.

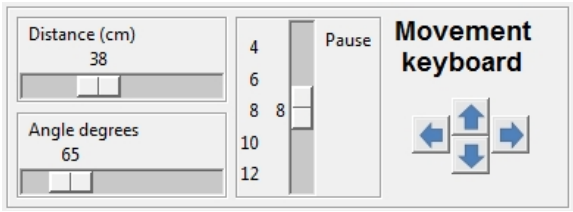

Figure 3.11: Movement keyboard with the scales parameters

Furthermore, the GUI provides the possibility of sending the robots a list containing multiple commands; this fullfils the requirement GUI4. This can be done by using the *File manager* box, which is composed by a small window text and four buttons. A screenshot of this is shown in the figure 3.12.

The first one opens a standard dialog window that allows the user to choose a txt file that contains the commands list. Only the robots that have enabled check boxes will receive them, and their names will appear together with the txt file on the *text window* as a visual feedback. This txt file can be found in the list `command_files`.

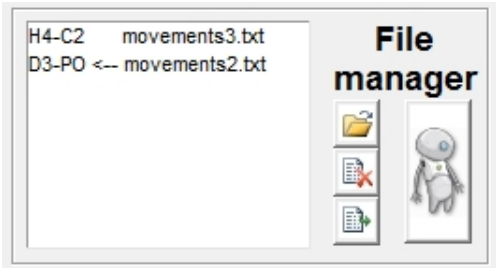

Figure 3.12: File manager screenshot. The file movements3.txt is associated with a robot called H4-C2, while movements2.txt is associated to D3-PO

The second one makes it able to delete possible unwanted commands.

Pressing another button the function `sendfile()` sends the commands list to the right IP address; and a small arrow is added in the feedback window to confirm the sending.

Part of the `send_file` function is shown below.

```
def sendfile(self):
 .....
 .....
 swap=filename.rsplit("/")
 string = ROBOTS[i][0] + " <-- " + swap[len(swap)-1]
 print(string)
 m=0
 while(m<len(view_list)):
  if(view_list[m]==ROBOTS[i][0]):
   self.file_listbox.delete(m)
   self.file_listbox.insert(m, string)
   break
  m=m+1
 outsock.sendto("<FILE>", (ROBOTS[i][1], 8100))
 p=0
 while(p<len(command_files[i])):
  string = string + command_files[i][p]
  p=p+1
 outsock.sendto(string, (ROBOTS[i][1], 8100))
```

Before doing this, the *while cycle* is looking for the robots which have to receive the command lists. When a robot is found, a small arrow is added in the feedback window. After that, this function sends the keyword `<FILE>` to inform the robot that a multiple commands list is incoming.

When the last button is pressed,a function is called. This one sends the keyword `<GO>` to every robot. Therefore, the execution of the robots start at the same time.

The GUI checks every 20 seconds if all the robots are connected. At the beginning of the execution, a thread is created as a time counter, which generates a process that invokes an request of the `online() function` per each established connection.

```
def online(self, position):
 outsock.sendto("<HowRU>", (ROBOTS[position][1], 8100))
 while(1):
  try:
   swap1 = self.insock_online.recv(1024)
   r=0
   while(r<len(ROBOTS)):
    if(ROBOTS[r][0]==swap1):
     online_list[r]=1
     break
    r=r+1
  except:
   print("Error in acknowledgement receiving process")
```

This function, which is shown above, checks the connection with the identified robot using the position integer. This is the last information of every element in the ROBOTS tuple; therefore after the keyword `<HowRU>` is sent, the process waits for the acknowledgement from the robot, and when it arrives, the associated flag is set on.

In this way, every 20 seconds the main thread checks all the flags. If a robot does not answer, it means that the connection is down, and therefore a red light appears in the *status window*. When the connection is established again, instead of a red light a green light appears, which result that the requirement GUI2 will be fulfilled.

In figure 3.13 the main keywords supported by the GUI in its I/O communications are shown.

Keyword - Message	Input	Output	Logical port	Short description
`<FILE>`		✓	8100	It begins a commands list transmission
`<GO>`		✓	8100	It starts the commands list execution of all the robots; its answer is the "*RobotName* has done" message
`<HowRU>`		✓	8100	It is the connection checking message; sent every 20 seconds, its answer is *RobotName*
`Measure` + parameters		✓	8100	It's the measurement command; its answer is *RobotName* + "T_put1" + *measure_results*
`Move` + parameters		✓	8100	It's a single movement command; it is created by the "Movement keyboard"
`RobotName`	✓		8500	It's the acknowledgement for the connection checking process
`RobotName` + `has + done`	✓		8100	This message advices the GUI that the robot finished the command list execution
`RobotName` + `"T_put1"` + *`measure_results`*	✓		8100	This message contains the results after a measurement session
`RobotName` + *`IP_address`* + *`MAC_address`*	✓		8100	This is the new client incoming message; at the end of the handshake all the robot's data are shown on the GUI

Figure 3.13: Main commands in the I/O communication of the GUI

In figure 3.14, the three main processes have been represented. On the right there is a checking connection process. After sending the `<HowRU>` package, it waits for the reply of the robot. If there is the expected acknowledgement, the process sends another package. If there is no reply, the status of the robot changes, and the GUI will not be able to connect to the robot.

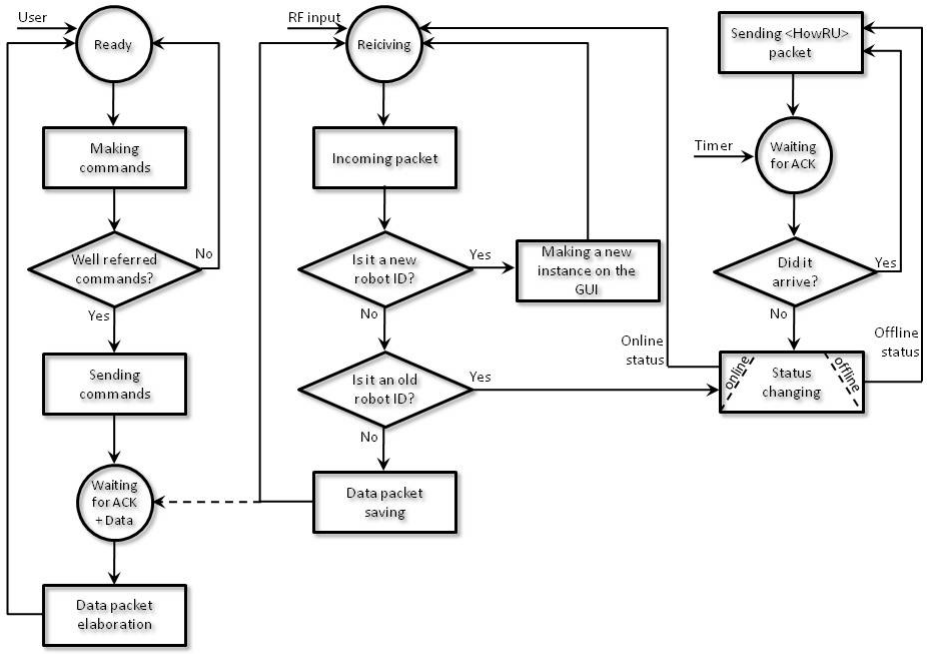

Figure 3.14: Flow diagram of the GUI

Pressing the *key boards*, the users can send a single command or a script by using the *File Manager*.

Before sending a new command, the process checks if the robot is busy or ready to receive more commands. If the robot is busy, the process displays an error message on the *Ready Status* screen.
Otherwise if the robot is ready, it receives a single command or a script. In that case the process waits for an acknowledgement from the robot and listens to new possible incoming messages.

Once the package has been received satisfactorily, the process detects if the package is a handshake message from a new client, a message from an old client, or a data packet from a measurement session. In the first case, the GUI creates a new request for managing the new robot; in the second case the GUI changes the status of the robot from offline to online. In the last case the data is stored and sent to the process, which is waiting for it.

Finally, thanks to the all data from the robot, the user is able to extract the necessary information that is needed to draw a conclusion.

Bluetooth Interface of the PC

The Bluetooth module in the PC is used to test the hardware, i.e., opensensor, engines, etc. For this purpose, we connect the hyperterminal (simple communication software) to the opensensor Bluetooth module. Thus, we can check if every sent command works; We can also use the Bluetooth module to connect the PC directly to the opensensor, and in this way the PC can control the robot. For this application an additional Python module is needed, called PyblueZ, which provides commands and tools to establish a Bluetooth communication by a PC. A representation of this module is shown in the figure 3.15.

Figure 3.15: Representation of the Bluetooth Interface of the PC

RS232 Interface of the PC

The serial connection is only used for testing purposes. As mentioned in the previous paragraph, the Bluetooth communication from the PC is useful to test the Opensensor, but the serial connection is needed because it offers us a channel feedback. In this way, the PC receives variable values and very useful status information in the testing phase. Thanks to the channel feedback, possible errors can be fixed. A representation of this module is shown in the figure 3.16.

Figure 3.16: Representation of the RS232 Interface of the PC

Once the modules of the PC are explained, we describe the modules of the Opensensor. These modules are shown in the figure 3.17.

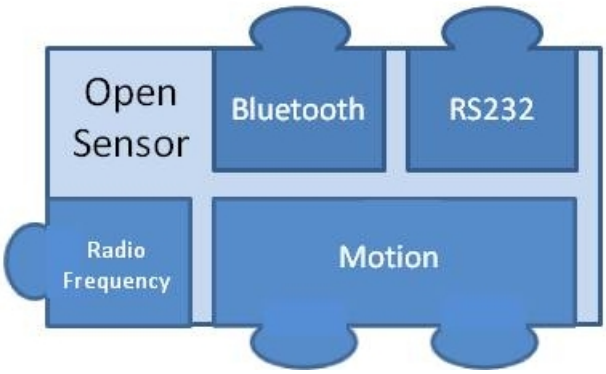

Figure 3.17: Modules of the Opensensor

Bluetooth Interface of the Opensensor

The Bluetooth module is used on the opensensor for two main purposes: hardware code testing and robot managing from a mobile device which supports a Bluetooth communication. A representation of this module is shown in the figure 3.18. As written before, we can send the commands to test directly the Bluetooth interface of the opensensor from the PC. Furthermore, this module allows us to establish a communication with a mobile device. In this case, the robot can be controlled by sending movement commands.

Figure 3.18: Representation of the Bluetooth Interface of the Opensensor

nRF905 Interface of the Opensensor

The Radio Frequency module is useful when other interfaces cannot be used. In that case the robot can be controlled by a radio communication device at the frequencies of 433, 868 and 915 MHz. A representation of this module is shown in the figure 3.19.

Figure 3.19: Representation of the nRF905 Interface of the Opensensor

RS232 Interface of the Opensensor

The RS232 module on the opensensor is used to receive a communication feedback, which allows the opensensor to send the PC all the information about the execution of the code. In this way, the execution status is known, and it leads us to understand how to modify the code when an error occurs. A representation of this module is shown in the figure 3.20.

Figure 3.20: Representation of the RS232 Interface of the Opensensor

Motion Interface on the Opensensor

The Motion module is the software interface between the Opensensor and the main board; therefore, it has to translate the high-level movement commands into current pulses.

The rotation of the wheels is realized by several tics. When they are executed in sequence, the wheel turns. To complete a cycle, every engine needs to execute four tics. Consequently, four output pins are needed.

In the code the value of the pins is assigned by setting the variables related to each of them. We call rb1, rb2, rb3 and rb4 the variables that control the four pins

of the left engine; and rb5, rb6, rb7 and rb8 the variables for the pins of the right one. The pins can be set to 1 or 0, i.e., ON or OFF. To show how this works an example is described.

Firstly, the variable rb1 is set to 1. Consequently, the related tic is executed and the wheel turns. Then the variable rb2 has to be set to 1 and rb1 has to set to 0. After that, the wheel turns again. This process has to be repeated for all the pins. Once the last variable has been set to 1, the cycle has to be reseted, and the variable rb1 is set to 1. This has to be repeated as many times as the user requires.

To illustrate an example of this, a sequence referred to the formward movement is shown in the code below.

```
rb1=1;rb2=0;rb3=0;rb4=0;
rb5=1;rb6=0;rb7=0;rb8=0;
Pause(speed);

rb1=0;rb2=1;rb3=0;rb4=0;
rb5=0;rb6=1;rb7=0;rb8=0;
Pause(speed);

rb1=0;rb2=0;rb3=1;rb4=0;
rb5=0;rb6=0;rb7=1;rb8=0;
Pause(speed);

rb1=0;rb2=0;rb3=0;rb4=1;
rb5=0;rb6=0;rb7=0;rb8=1;
Pause(speed);
```

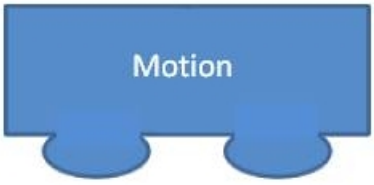

Figure 3.21: Representation of the Motion Interface of the Opensensor

3.2.3 Protocol Stack

After a short description of each module, we are able to put all of them together. For this purpose, a protocol stack of the robot, with all the communications among the modules is described.

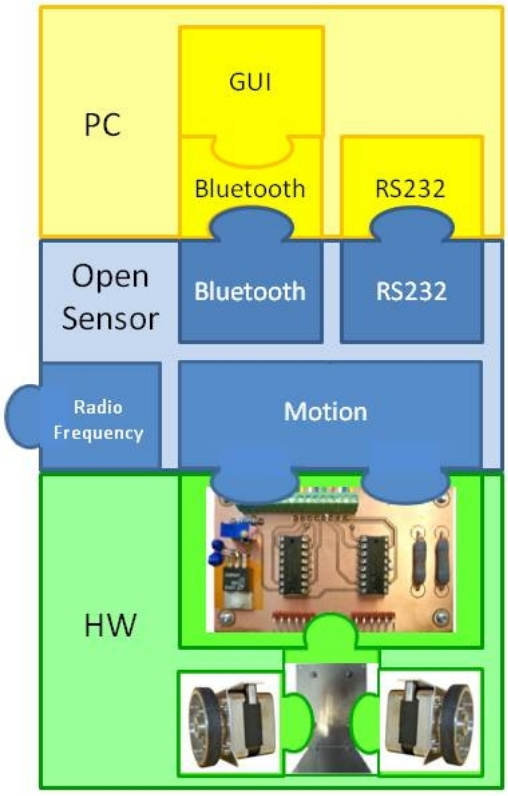

Figure 3.22: Protocol Stack

As we can see in the figure 3.22, the green part is the lowest layer and it holds all the mechanical parts: the wheels, the engines and the metal platform that hots the other parts. This part is directly connected to the blue part, through the Motion section. This layer, the middle one, holds all the possible connection interfaces describe before: the nRF905 Antenna, the Bluetooth and RS232 serial connection.

In this protocol stack, the robot can be connected to a PC over two interfaces as shown in the figure above: the bluetooth and the serial ones. A Graphical User Interface, which allows the user to control the Robots, is built on the Bluetooth module.

In figure 3.23, a table summarizes the requirements which are fulfilled by the modules described in this chapter.

Requirements	GUI PC	Main board	Opensensor			Mechanical parts			
			BT	nRF905	Motion	Wheels	Engines	Platform	Battery
CB1		✓							
CB2			✓	✓					
CB3					✓				
GUI1	✓								
GUI2	✓								
GUI3	✓								
GUI4	✓								
GUI5	✓								
GUI6	✓								
GUI7	✓								
FR1			✓						
FR2				✓					
FR3					✓				
MB1						✓			
MB2							✓		
MB3						✓			
MB4						✓	✓	✓	✓
MB5						✓	✓	✓	✓
MB6						✓	✓	✓	✓
MB7									✓
MB8									✓
MB9									✓
MB10									✓
MB11								✓	

Figure 3.23: Requirements table

Chapter 4

Examples

In this chapter we introduce some examples showing how we carried out some energy and throughput measurements in order to show how the robot works. We will focus on mobile wireless networks with dynamic topology.

Due to the use of mobile phones in these experiments, we need to introduce new requirements in order to carry out the measurement campaign satisfactorily. We will show two different approaches. All the measurements were taken with the mobile phones placed on the platforms of the robots.

Before starting to take the measurements, it is important to understand that the main goal of this project is to offer the scientific community a new tool able to optimize measurements in wireless networks, and not to give some final conclusions about the measurements taken.

Figure 4.1: The Robot

All the measurements have been taken at Aalborg University, Denmark. The first measurement location is a big corridor of approximately 30 meters long in the basement of the Electronics System Department; the second one is a corridor with a corner which is located in the second floor of the same building.

4.1 New Requirements

- MR1: The device must be able to take measurements of throughput
- MR2: The device must be able to take measurements of signal strength
- MR3: The device must be able to take measurements of energy consumption
- MR4: The device must be able to communicate over Bluetooth and have simultaneous connections
- MR5: The device must run Python 1.9.2 or a newer version to make it possible to use the needed applications
- MR6: The device must have a colour screen to show a video stream
- MR7: The device must be able to remote control the robot

With the purpose of fulfilling the new requirements, a Nokia N95 8GB is chosen. The Nokia N95 8GB is a commercial mobile phone available in the market equipped with WLAN IEEE802.11b/g, a Bluetooth module and many internal sensors.

An example of internal sensor is the light sensor which is used to adjust the level of brightness of the display on the phone. This can be useful, for instance, to save energy or permitting the users to make use of the phone even in the night. This device fulfills all the requirements listed before. Some additional programs are needed to take measurements and they will be described in the following sections.

In order to prove the functionality of the Robot, two approaches can be considered: Passive and Active.

4.2 Passive Scenario

In this scenario the phone is placed on the metal platform of the Robot as shown in the figure 4.2, but it does not have any kind of connection with the motion part of the Robot. It is important to emphasize that in this scenario the phone acts like a passenger, and it is free to take measurements in the environment with its internal sensors.

In order to make the measurement campaign possible, we use the Nokia N95 8GB mobile phone because fulfills the requirements MR4, MR5 and MR6 described in the previous section.

Figure 4.2: A snapshot of the measurements campaign

Furthermore, the phones will run two different tools which are described at the end of the chapter in order to fulfill the requirements MR1, MR2 and MR3 for taking the measurements.

The phones are connected in Ad-Hoc Mode (See Appendix C to set up a connection among the phones).

According to the Passive Scenario, two different examples can be considered.

Example 1

As mentioned before in this chapter, the first example of the measurements are taken in a corridor in the basement of the university. To carry out this measurement campaign two Nokia N95 8GB mobile phones are needed - Phone 1 works as a sender and the Phone 2 as a receiver. As shown in the figure 4.3, the sender is in a fixed position at the beginning of the corridor. The receiver is placed on the metal platform of the Robot. The Robot is moving in a straight line from the sender to the end of the corridor with a low speed.

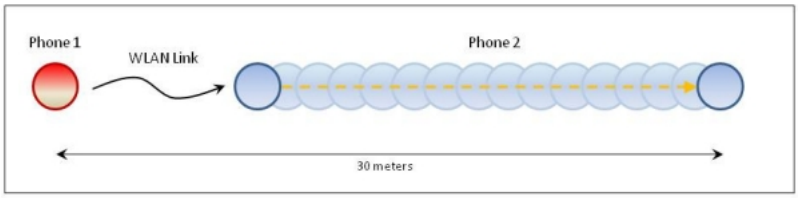

Figure 4.3: Plan of the example 1

In figure 4.4, the throughput of the transmission is shown. The sender is forwarding continually packets of 1 Kilobytes to the receiver. The receiver on the Robot is measuring the throughput every two seconds. As we can notice the throughput decreases with long distances.

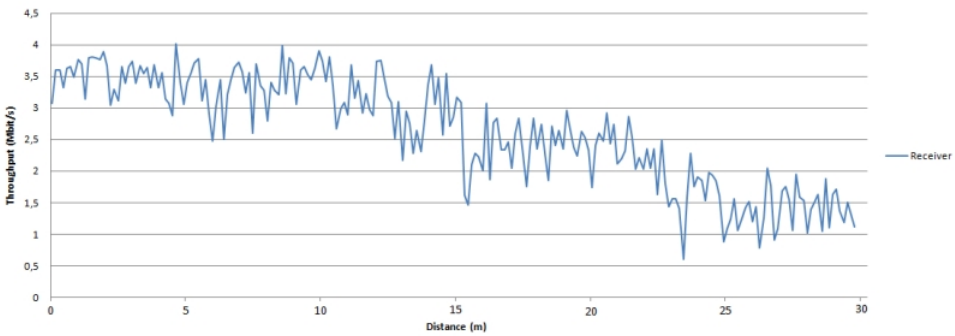

Figure 4.4: Throughput on the Receiver in example 1 of the Passive Scenario

Example 2

Another example has been carried out to prove the functionality of the Robot. In this second example a sender and a receiver are not in the line-of-sight. They are located at the beginning, and at the end of a 30 meters L-shaped corridor. The corridor is divided in two halfs of 15 meters. The plan of the testing environment is shown in the figure 4.5.

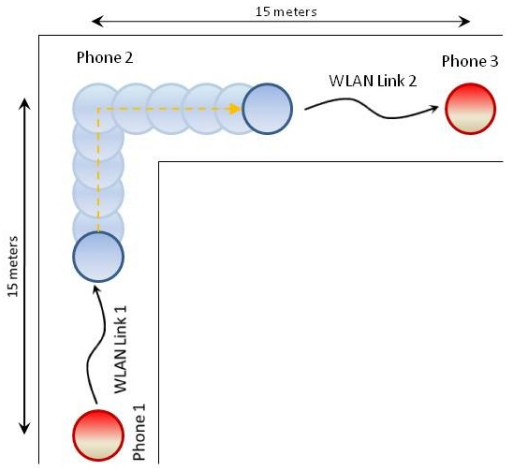

Figure 4.5: Plan of the example 2

The Relay on the Robot is moving from the sender to the receiver with low speed. The sender is transmitting continuously packets of 1.5 Kilobytes and at the same time, the relay and the receiver are taking measurements, every two seconds, of the throughput of the communication. In figure 4.6 a snapshot of this example is shown.

Figure 4.6: A snapshot of the measurements campaign in the example 2

In this example the direct transmission is not possible, because the sender and the receiver are not in the range of connection. The task of the relay becomes strictly necessary to assure a transmission from the sender to the receiver. After setting up an Ad-Hoc network among the three phones, the sender starts the transmission of the packets to the relay. The relay stores the packets, estimates the throughput and forwards them to the receiver. When the final phone has received all the packets, it estimates the throughput of the communication as shown in the figure 4.7.

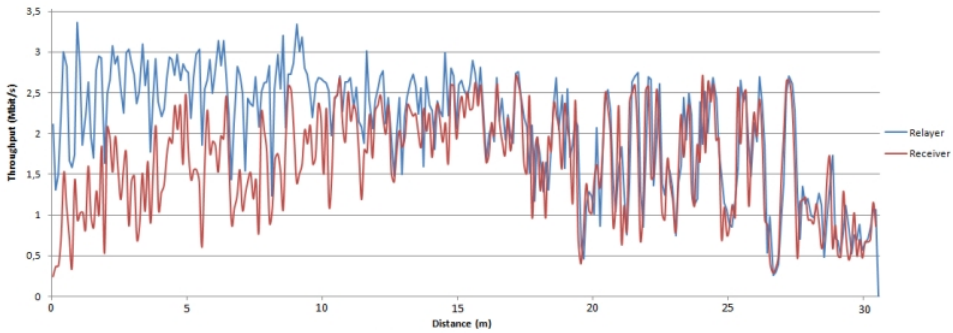

Figure 4.7: Throughput on the Relay and the Receiver in the example 2 of the Passive Scenario

A trend line has been plotted to understand what happens with the throughput of this transmission. We use the Moving Average trend line, which smoothes out the fluctuations of the obtained data to show a clearer pattern. The Moving Average calculates the average of the last values in the sequence. As an example of this if we choose p values to calculate it, the n-th average value is:

$$a_{n,p} = \frac{a_{n-p} + a_{n-p+1} + \cdots + a_{n-1}}{p} \text{ ,where} p, n \in \mathbf{N} \tag{4.1}$$

In formula 4.1, p is the number of previous values used to calculate the n-th avarage value.

In figure 4.8, we use a parameter p equal to 10, therefore, we can observe a clearer pattern of the throughput. But in the other side, if we use a p equal to 20, as shown in the figure 4.9, we have something better, enough to understand what happens during the transmission.

As we can notice from the plot, the throughput of the Receiver has a parabolic trajectory. The maximum value of the throughput is 2.6 Mbit/s and is given at 15 meters, when the relay reaches the corner of the corridor. Instead of this, the throughput of the relay decreases with long distances and reaches its maximum value of 3.4 Mbit/s when the robot is very close to the sender.

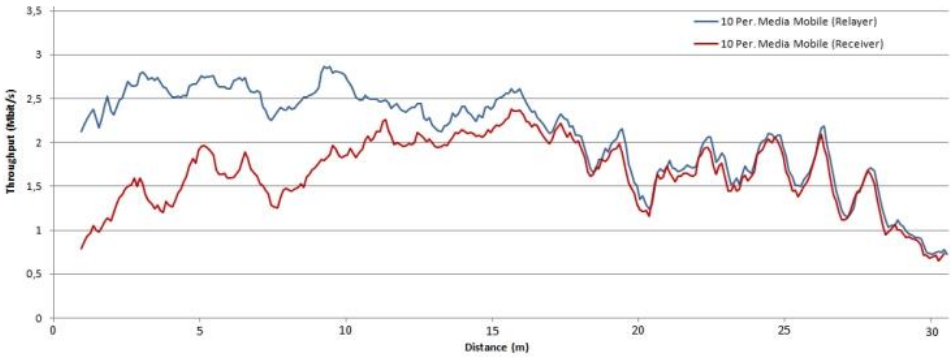

Figure 4.8: Moving Average throughputs when p=10 is chosen

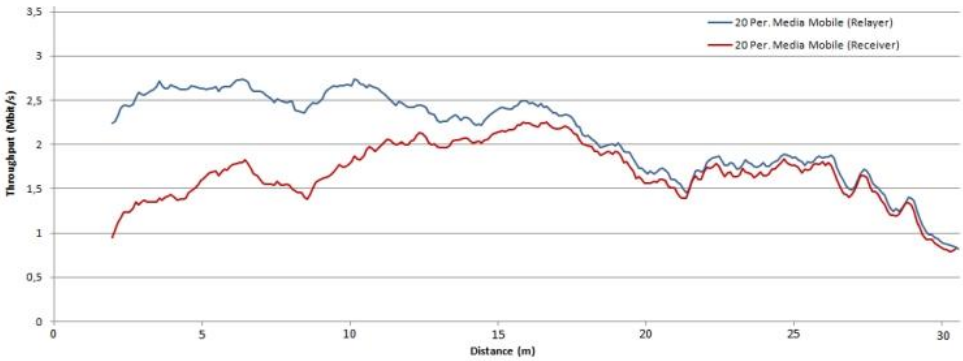

Figure 4.9: Moving Average throughputs when p=20 is chosen

According to the figure 4.10, we can also plot the signal strength on the receiver. Therefore, the signal strength increases when the robot is approaching the receiver.

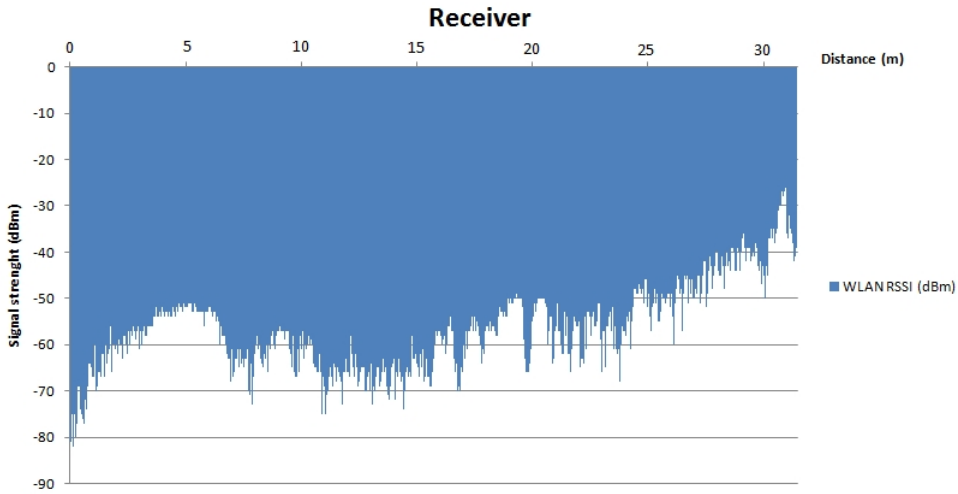

Figure 4.10: Signal Strength of the Receiver in the Passive Scenario

4.3 Active Scenario: Spy Robot, a new Application

The Active Scenario is the second experiment where the Robot is used. As we referred, there are many possibilities how devices can communicate with the Robot. In this scenario, a Nokia N95 mobile phone has been chosen to be connected with the Robot via Bluetooth. In this way the phone is not just a simple passenger of the robot anymore. In this example we could control the Robot with a remote controller connected to the phone on the mobile platform. The plan of the testing environment can be seen in the figure 4.11.

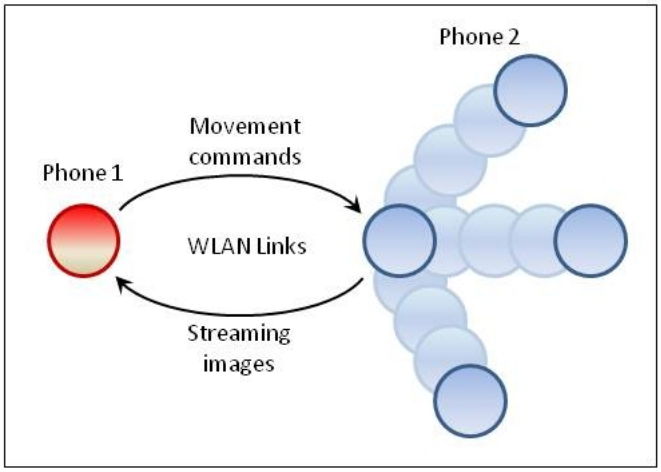

Figure 4.11: Plan of the active scenario

Our robot is very versatile; any device with a Bluetooth or a nRF905 module can be able to control it. To give a simple example on how the active scenario works, we decided to develop an application called *Spy Robot*.

Spy Robot is an application developed for phones supporting Python 3rd Edition. Using this application we are able to control the robot to patrol the environment.

Two N95 8GB mobile phones and the Robot Platform are needed to develop this application. The Phone 1 acts as the receiver and the Phone 2 as the sender.

The sender is connected to the opensensor of the Robot over Bluetooth and is placed on the robot. It forwards video stream to the Phone 1 by using its camera module. The receiver is connected to the sender on the Robot in AdHoc mode.

When the program begins working, it asks the user to select the Ad-Hoc network that both phones have to use for their communication; this network has to be set up as a wireless ad-hoc network before we run the program.

The parameters of this network are shown in the figure 4.12.

Figure 4.12: Set up of the network on the two phones

A Graphical User Interface for this application is developed using Python 1.9.2. When the user starts the application an image with the background of the program is shown on the screen. After few seconds, on the screen of the phone placed on the robot appears a menu to select the video resolution between 160X120, 240X180, 240X240 or Fullscreen. After this step a new screen appears, which gives the possibility to start or stop to send video from the phone on the robot to the phone of the user. All these steps are shown in the figure 4.13.

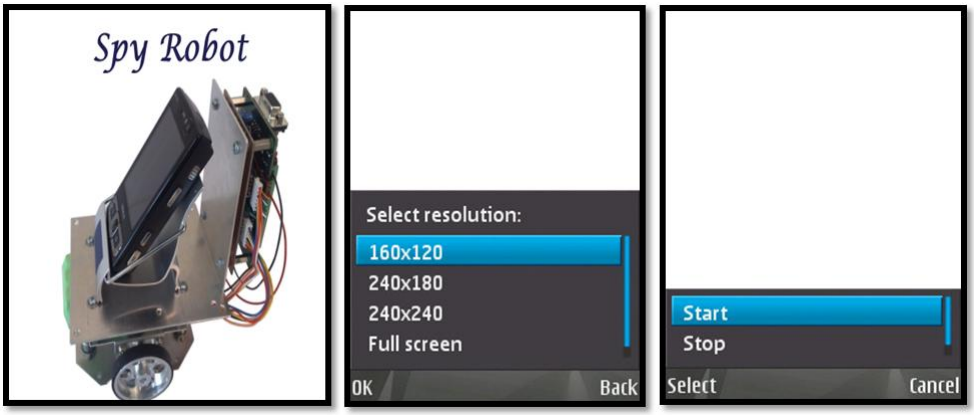

Figure 4.13: From the left to the right: Application Background, Resolution Menu and Start Stop Menu

A screenshot of the video that the user can see from the robot is shown in the figure 4.14.

Figure 4.14: Image from the camera of the phone on the robot

The Phone 1, at the same time is receiving video stream from the Phone 2, can control the robot movements by using the keyboard as a remote controller. To make this possible, the keyboard of the receiver is bounded with the movement commands of the robot. The commands are sent over WLAN from the receiver to the sender by using the arrow keys, which are showed in the figure 4.15. When the Phone 2 receives the commands, it forwards them to the Robot over Bluetooth, and the Robot moves. This feature of the application fulfills the requirement MR7.

Figure 4.15: Arrow keys which make it possible to remote control the robot

As we mentioned before, any device with a Bluetooth module can be integrated with this application. For this reason, we can extend the same application for the GUI on the PC. To make this possible we need to improve the PC GUI described in the section 3.2.2 by adding a canvas in order to receive video stream from the Phone 2.

Furthermore, to make the application works, it is necessary to connect to the previous access point on the PC which was created to control the robots.

At the end, the new protocol stack is showed in the figure 4.16. As we can see the phone replaces the PC on the top of the stack but in this case we do not have any RS232 connection.

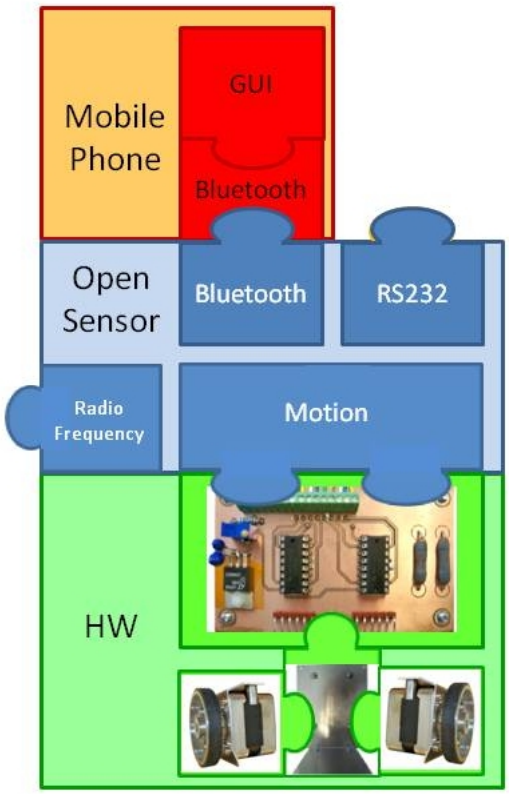

Figure 4.16: New Protocol Stack

4.4 Tools

In this section we explain the tools we used to take the measurements in the previous scenarios.

4.4.1 Python Measurement Application

This application was developed to measure the throughput and the delivery packets on the phones during the measurement campaign. The code of the relay is listed below as an example. We can notice the Ad-Hoc Class which is useful to connect three phones in Ad-Hoc Mode. The *Connect function* is needed to connect the phone, to the NokiaNet Access Point. Furthermore, two files named *throughput.txt* and *packets.txt* will be created to save the results. In the *Connect function* an input socket is opened to receive the packets from the sender.

The *Test function* opens an output socket to send packets to the last phone in the path. Furthermore, the function is calculating the throughput of the previous transmission every two seconds. At the end of the process of the application the results are saved in txt files.

```
import time,appuifw,e32
import appuifw
DATA = str().zfill(1000)
class Adhoc:
    def connect(self):
        self.f=open('e:\\python\\throughput.txt', 'w')
        self.f2=open('e:\\python\\packets.txt', 'w')
        self.api = select_access_point()
        self.apo = access_point(self.api)
        set_default_access_point(self.apo)
        self.apo.start()
        print "IP address =",self.apo.ip()
        self.soc = socket(AF_INET, SOCK_DGRAM)
        self.soc.bind(("0.0.0.0", 8100))
        while 1:
            self.test()
    def test(self):
        self.out_soc = socket(AF_INET, SOCK_DGRAM)
        num_pack=(" ---- Packets number: ")
        through=(" - Throughtput: ")
        timing=(" - Time: ")
        comma=(", ")
        i = 0
        l = 0
        t1 = 0
        t2 = 0
        print("\nReceiving started..")
        while (((t2-t1))<2):
            data = self.soc.recv(1000)
```

```
            self.out_soc.sendto(data,('172.16.0.1', 8100))
            if(t1 == 0):
                t1 = time.clock()
            i = i + 1
            l=l + len(data)
            t2 = time.clock()
        self.f2.write(str(i))
        self.f2.write(comma)
        self.f.write(str(l / 1024.0 / 128.0 / (t2-t1)))
        self.f.write(comma)
        self.out_soc.close()
```

4.4.2 Nokia Energy Profiler

The Nokia Energy Profiler 1.2 is a test and measurement application developed by Nokia. It can be used for S60 3rd Edition, Feature Pack 1 and later devices. This application allows developers to monitorize and test the energy consumption of the applications in real time which are installed on the phones, used in the measurement campaign.

This application can measure the power consumption of a device during a measurement period. There is a possibility to switch between average and instant power.

This application also measures current consumption using the battery power level. Voltage goes down and current increases when battery discharges, therefore power consumption keeps the same level, meanwhile current consumption depends on the battery-charge level.

CPU measurements can also be made by using Nokia Energy Profiler 1.2. It shows the load processor during the measurement period in percentage out of total potential of CPU-processing. The results can be between 0% and 100 %, where 0% is no processing data and 100% is the highest processing result.

Download and upload speeds can also be measured, which is called Network measurements.

Received-signal strength can be measured as well, when the phone is connected to a WLAN base station, the WLAN measurements are given in decibels referred to 1milliwatt (dBm) and usually varies from -40 to -90 dBm. Those values closer to -40 show best signal conditions.

Signal Levels measurements reflect the cellular signal levels as Transmission and Reception. The Reception level reflects the power of the received signal while the Transmision level corresponds to the power of transmission from the cellular

radio. Transmission levels appear during periods when the transmission is active, Reception levels are approachable each time the phone is connected to a cellular network. These measures are in dBm.

Voltage measurements show the levels of battery-voltage, for example when battery discharges, the voltage goes down until the phone switches off by itself [20].

4.5 Scenarios

Once we have described how the robot works, all the possible scenarios that can be implemented with the robot. For each scenario, different topologies and a description about how they work are presented.

4.5.1 Scenario 1

Topology

One-to-One Two mobile phones and one robot are needed.

One-to-Many Several mobile phones and robots are needed.

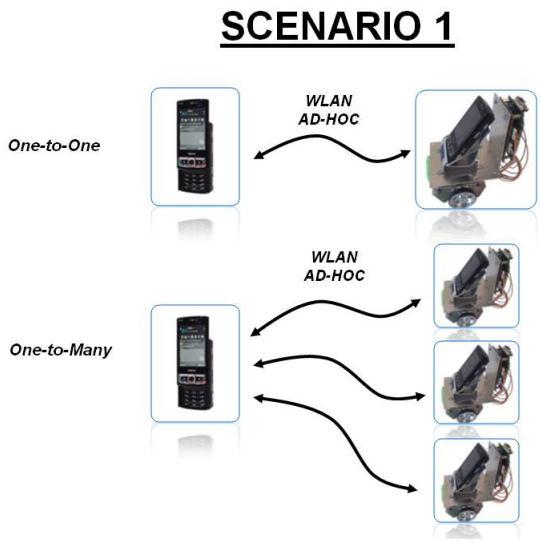

Figure 4.17: Topology of the Scenario 1

How it works

An AdHoc network using local addresses has to be created. After that, the Python application on the sender connects to the AdHoc network and it waits for incoming clients. The phones in the receiver part, connect over Bluetooth with the Opensensor. At this time, the connection between the two parts is established, and the sender can start to forward commands to the receiver. This one translates these commands into a code known by the Opensensor, which sends the right pulses to the engines after a decoding process. An example of this code is `id_command#direction#velocity# distance/angle`.

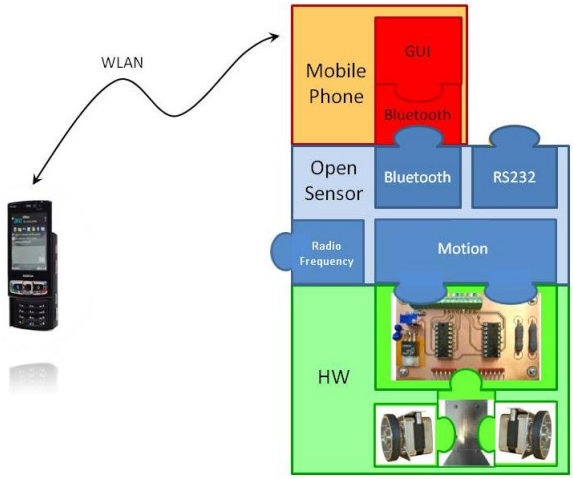

Figure 4.18: Protocol Stack of the Scenario 1

4.5.2 Scenario 2

Topology

One-to-One One PC with the Graphical User Interface (GUI) we have designed, and one robot are needed.

One-to-Many One PC with the Graphical User Interface (GUI) we have designed and several mobile phones and robots are needed.

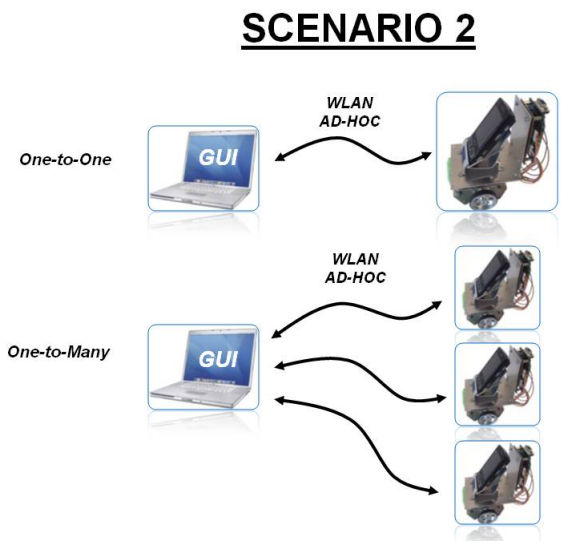

Figure 4.19: Topology of the Scenario 2

How it works

In this scenario, once the wireless connection is established, the laptop works as a server. When the GUI is inizialized, it waits for possible incoming clients, then application is on the phones asks the user to connect to the Opensensor over the Bluetooth device. After that, it looks for the Ad-Hoc network to use.

When each robot is connected to the Ad-Hoc network, they send their data (name of the Robot, IP address and MAC address of the Opensensor) to the laptop. In this moment is possible to control the robots by sending them moving commands. Besides, video stream from the robot can be sent to the PC.

The phones receive commands from the laptop and translate them into the code `id_command#direction#velocity#distance/angle` that the Opensensor is able to understand. Then, this code is sent over the Bluetooth interface, and after a decoding process, the Opensensor sends the right pulses to the engines.

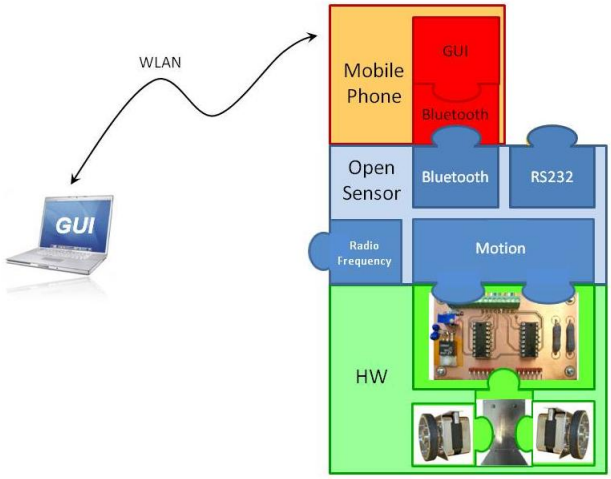

Figure 4.20: Protocol Stack of the Scenario 2

4.5.3 Scenario 3

Topology

One-to-One A mobile phone and one robot are needed.

One-to-Many A mobile phone, one opensensor and several robots are needed

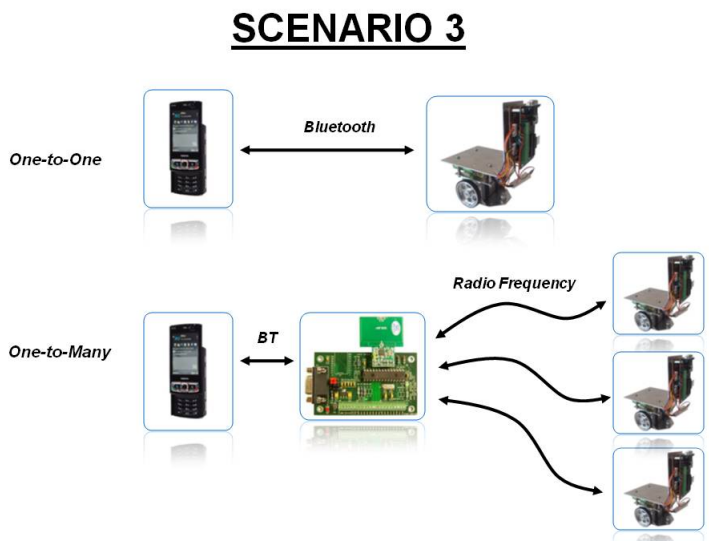

Figure 4.21: Topology of the Scenario 3

How it works

In this scenario a mobile phone is allowed to control directly the robot by using the Bluetooth module.

The number of platforms that one phone is able to control dependens on the nRF communication between the first opensensor and the others placed on the robots.

As we mentioned before, the first communication between a mobile phone and an opensensor is established over Bluetooth and the opensensor is directly connected to the engines of the robot.

Thanks to the GUI developed on the phone, the user can selects in which way the commands are sent to the opensensor. The direction, the speed and the distance can be changed from the menu on the phone. Furthermore, the robot can be remote controlled by using the keyboard of the phone.

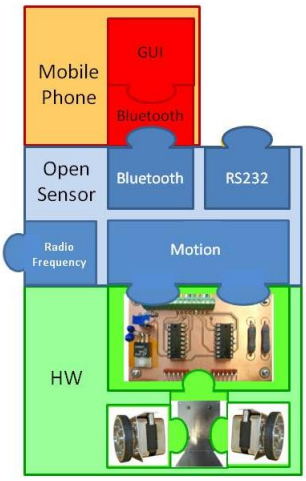

Figure 4.22: Protocol Stack of the Scenario 3 for the One-to-One topology

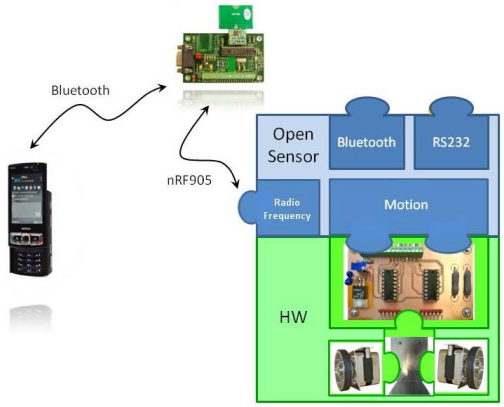

Figure 4.23: Protocol Stack of the Scenario 3 for the One-to-Many topology

4.5.4 Scenario 4

Topology

One-to-One One PC with the Graphical User Interface (GUI) we have designed and one robot are needed.

One-to-Many One PC with the Graphical User Interface (GUI) we have designed, an opensensor and several robots are needed.

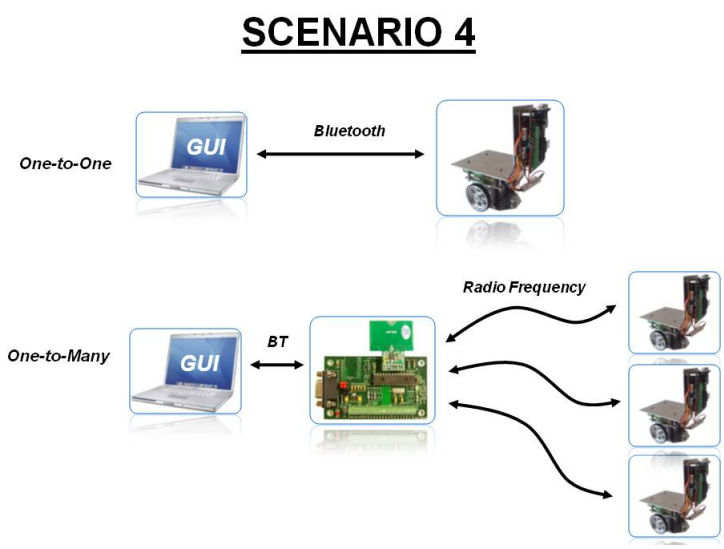

Figure 4.24: Topology of the Scenario 4

How it works

In this scenario a PC (GUI) is allowed to control directly the robot by using the Bluetooth module

The number of platforms that a phone can control it depends on the nRF communication between the first opensensor and the others located on the platforms.

The communication between the computer and the opensensor is established by using the Bluetooth module. the opensensor is directly connected to the engines of the robot. Thanks to the GUI developed on the PC, the user can selects in which way the commands are sent to the opensensor.

In this scenario the Mobile Phone placed on the robot can be used as a passenger, and it is not connected to the Opensensor.

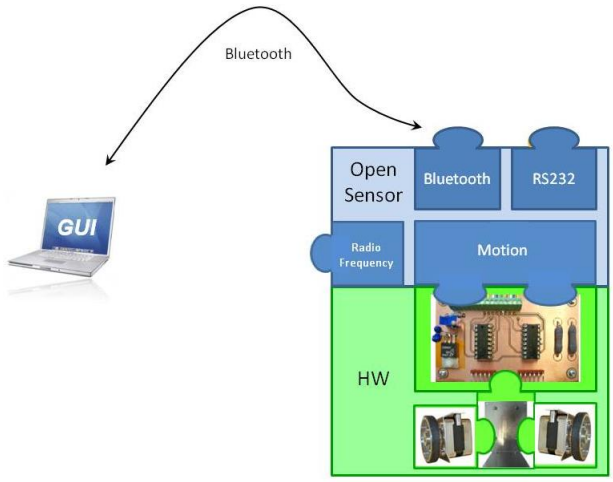

Figure 4.25: Protocol Stack of the Scenario 4 for the One-to-One topology

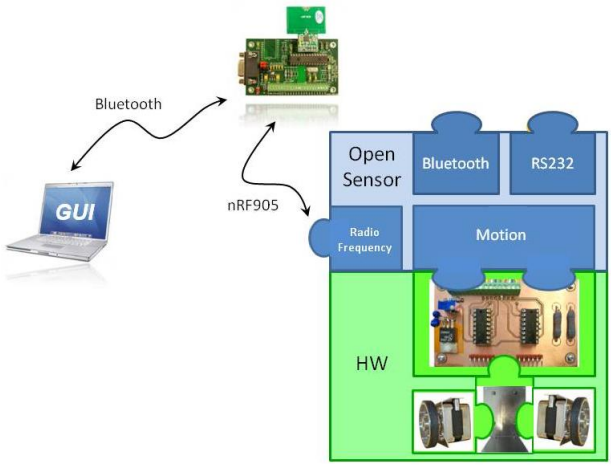

Figure 4.26: Protocol Stack of the Scenario 4

In the figure 4.27 all the scenarios are summarized to give a overall view of them.

Scenario		Sender			Receiver		
		Mobile Phone	Opensensor	Laptop	Mobile Phone	Opensensor	Robot Platform
COMM1	One-to-One	✓			✓	✓	✓
	One-to-Many	✓			✓	✓	✓
COMM2	One-to-One			✓	✓	✓	✓
	One-to-Many			✓	✓	✓	✓
COMM3	One-to-One	✓				✓	✓
	One-to-Many	✓	✓			✓	✓
COMM4	One-to-One			✓		✓	✓
	One-to-Many		✓	✓		✓	✓

Figure 4.27: Comparison table of the Scenarios

Chapter 5

Tests

In this chapter some tests are performed in order to validate the requirements describe in previous chapters. For each requirement a test has been carried out. At the end of each one, the results are presented.

5.1 Software Requirements: Testing the GUI

GUI1: The GUI must show IP Addresses, MAC Addresses and names of the robots

Test Purpose: To test if the GUI receives the correct IP Addresses, MAC Addresses and name of the Robot which are online.

Starting Condition: The Robot must have a mobile phone with a Bluetooth module enabled.

Test Sequence: The Ad-Hoc receiver application on the phone which is placed on the robot begins running. Then, the application asks the user to select the Bluetooth device to establish the communication by selecting the MAC address of the opensensor. When the phone is connected to the opensensor located on the robot, it sends over Bluetooth the name, the IP and MAC Addresses of the opensensor.

Success Criteria: The GUI plots the data on the screen of the PC. On the phone appears the message *receiving started*.

Results: The test has been carried out successfully.

GUI2: The GUI must show which Robots are connected and which ones are not

Test Purpose: Test if all the connected Robots are displayed in the right way on the screen of the GUI.

Starting Condition: A traffic light diagram on the GUI is needed

Test Sequence: When a Robot is connected to the GUI the traffic light becomes green. When the robot is disconnected it becomes red.

Success Criteria: On the screen of the phone, the message *Connected* appears.

Results: The test has been carried out successfully.

GUI3: The User must be able to select one or more robots to be controlled

Test Purpose: To test the possibility of selecting one or more robot using the GUI.

Starting Condition: A checkbox on the GUI is needed.

Test Sequence: By using the mouse pointer the Robots connected on the GUI can be selected or not .

Success Criteria: The user can control one or more robots by selecting them in the checkbox.

Results: The test has been carried out successfully.

GUI4: The User must be able to send a single command or script with a list of commands

Test Purpose: To test if the user can move the robot with the keyboard or sending several commands.

Starting Condition: We need a keyboard with four arrows and an interface on the GUI for sending txt files to the phones placed on the robots. Furthermore, we need a window called *execution log* to show the content of the files.

Test Sequence: By pressing the arrow keys, a command is sent over Bluetooth to the phone and it is displayed on the *execution log* When the phone receives the command, a string is printed, e.g., `Move#F#10#10`, and is forwarded to the opensensor. This one receives it and forwards it to the motion part.

Success Criteria: : When the commands are sent to the phones, a list is displayed on the screen.

Results: The test has been carried out successfully.

GUI5: The User must be able to send and receive video from the Robot

Test Purpose: Test if it is possible to receive a video stream from the camera of the phone placed on the robot.

Starting Condition: A robot connected over Bluetooth or nRF905 to the GUI and a new canvas for the video stream on the GUI are needed

Test Sequence: This test is referred to the application Spy Robot. As we mentioned in the Chapter 4, the GUI is equipped with a new canvas that shows the video stream from the camera of the phone placed on the Robot. There are two buttons to start and stop the video.

Success Criteria: Video stream from the camera of the phone blinks on the PC when it is sent to the GUI.

Results: The test has been partially carried out.

GUI6: The User must be able to change the speed and the direction of the robots

Test Purpose: We are testing if there is the possibility of changing the direction and the speed of the Robots.

Starting Condition: The GUI must be equipped with more elements to allow the changing of path and speed and the robots must be connected to the GUI over Bluetooth or nRF905.

Test Sequence: Several tests have been carried out, for distance, speed and angles, to create three checkboxes on the GUI.

Success Criteria: The user can change parameters such as speed, distance, etc, on the GUI by using the mouse. The GUI creates different strings with different values of speed and distance. The strings have to be sent to the phone on the Robot.

Results: The test has been carried out successfully.

GUI7: The GUI must show an error message if one or more Robots are disconnected

Test Purpose: Test if an error message appears on the screen when the connection with one Robot is down.

Starting Condition: An *execution log* window on the GUI to print error messages is needed.

Test Sequence: The GUI on the PC shows an error message on the execution log when the connection is down.

Success Criteria: The message *name of the robot* is offline is displayed on the *execution log* .

Results: The test has been carried out successfully.

5.2 Functional Requirements

FR1: The Platform must be able to communicate over Bluetooth with other devices

Test Purpose: Test the Bluetooth modules.

Starting Condition: We need a PC with the *Hyperterminal* and a Bluetooth module. Also an opensensor equipped with a Bluetooth module is needed.

Test Sequence: The user runs the *Hyperterminal* on the PC and send movement commands to the Bluetooth module of the opensensor.

Success Criteria: The Robot moves in all the directions.

Results: The test has been carried out successfully.

FR2: The Platform must be able to communicate over nRF905 with other devices.

Test Purpose: Test the nR905 modules.

Starting Condition: We need a PC with the *Hyperterminal* and an opensensor equipped with a nRF905 module.

Test Sequence: The nRF905 module of the opensensor uses some pins which are already used for the engines of the two motors. The pins are RB9 which is used for the carrier detect, RB4 for the Address Match and RB5 for receive and transmit data. We are not allowed to change pins for this tasks. For these reasons, when the communication starts, the opensensor is not able to move the wheels of the Robot using nRF905 even if it can receive commands from the GUI on the PC or from another nRF device. Furthermore the microprocessors on the motion board overheat. A possible solution of this problem could be supplementing the opensensor with more pins and using them only for the nRF905 communications. Another solution is the development of a Switch which allows us to bypass the pins of the engines when we are not making a movement.

Results: The test failed.

5.3 Requirements for the examples

MR1: The device must be able to measure the throughput

Test Purpose: Test the application to take measurements of throughput.

Starting Condition: We need at least two mobile phones with WLAN and Bluetooth modules and Python for S60 3rd edition installed.

Test Sequence: An application for the mobile phones is developed to measure the throughput of the communication. For this purpose, two mobile phones are needed. The receiver runs the application *Receiver(noBT).py*. and the message *Receiving Started* is shown on the screen. At the same time the sender is forwarding packets to the receiver by using the application *Sender.py*. (For further information see Chapter 4).

Success Criteria: The measurement results have to be saved in a txt file.

Results: The test has been carried out successfully.

MR2: The device must be able to take measurements of the signal strength of the communication and the energy consumption of the devices

Test Purpose: Test if the phones can take measurements of the signal strength and the energy consumption.

Starting Condition: Two phones and the application Nokia Energy Profiler v1.2. are needed.

Test Sequence: The application is running on the phones during the measurement campaign.

Success Criteria: Once the measurements have been taken, all data is saved on the phones and can be exported to the PC.

Results: The test has been carried out successfully.

MR4: The device must be able to communicate over Bluetooth and other interfaces at the same time.

Test Purpose: Test if the phones can find devices. In case a device is found, we test if the phone is able to send and receive data over Bluetooth.

Starting Condition: One Bluetooth enabled phones and the Robot.

Test Sequence: The GUI on the phone asks the user which Bluetooth device can be selected. Once a device is selected, the robot can be moved by sending strings.

Success Criteria: The string which is sent is displayed on the screen of the phone and the Robot starts the movement,

Results: The test has been carried out successfully.

MR5: The device must be able to move the robot as a remote controller.

Test Purpose: Test if we can control the robot from the phone as we did with the application Spy Robot (Chapter 4)

Starting Condition: Two phones and the application Spy Robot are needed.

Test Sequence: A video stream from the camera of the robot can be received by running the application *SpyRobotOnBoardPhone.py*. The phone placed on the robot and the phone of the user, which receives the video stream, are connected in WLAN adHoc Mode.

Success Criteria: The Robot can be moved by using the keyboard of the phone.

Results: The test has been carried out successfully.

Chapter 6

Conclusion

6.1 Conclusion

The main goal of this project has been to develop a mobile platform designed to optimize measurements for mobile wireless networks, especially those with dynamic topology.

Firstly, to build the mobile platform, which is composed by the mechanical and the controller block, we made an analysis of the needed software and hardware requirements. All the software modules have been designed using a modular structure, so the robot can be interfaced with devices that support communication over Bluetooth and over nRF905. Furthermore, we created a Graphical User Interface, both for the PC and the Nokia N95 8GB mobile phone, that make it easy to control the robot. This modular structure allows handling different topologies, which were divided into four scenarios. Each scenario is characterized by different communication links and devices.

Secondly, we wrote all the necessary code for the mobile phones in order to carry out measurements in two different scenarios. In this part the phones were used together with the robots to create a wireless network with dynamic topology.

Thirdly, we evaluated the performance of such networks by measuring the throughput of the communication. To do this we considered two different approaches. In the passive scenario, two examples have been carried out. In the first one, the sender was in a fixed position and the Robot was moving in a straight line from the sender to the end of the corridor. In the second example, the sender and the receiver were not in the range of the communication, each mobile phone was located on each side of an L-shaped corridor and the Relay was moving from the sender to the receiver. After all of this we demonstrate that the robot offers more realistic measurements in real environments than simulations.

Finally, the purpose of this project is not to offer good results about the wireless networks with dynamic topology, but to offer to the scientific community a new interesting tool that is freely available and can be easily configured that permits the researchers to improve their research studies about wireless networks.

6.2 Future Works

In this project we measured the throughput of the communication. But the robot can be used to evaluate other kinds of parameters thanks to the multiple sensors that can be plugged. For instance, evaluations of temperature, brightness, etc.
Some others examples are:

The robot can be used to create a mobile access point in disaster areas. In that case the robot should be improved in order to be able to move on rough surfaces.

Another possible application that could be developed is an Auto Charger for mobile phones by using photovoltaic module embedded on the Robot,

The Radio Frequency Identification is another possible application where our Robot could be used. BRFID gives us the possibility of identifying and tracking an object incorporated into a product by using radio waves. For this reason one possible solution could be to integrate the RFID into our Robot. This new platform could be used in a large variety of application such as Location-based services where GPS would not work.

Furthermore, the robot could be used to fix broken communications,i.e., bridging, as a direct application of the work we carried out.

Appendix A

Opensensor Programming Environment

In this appendix, we explain how to install the MPLAB Integrated Development Environment under Microsoft Windows. Nevertheless, in [19] the Linux version can also be found. Once the program is successfully installed, we will shortly explain how to use it to program the Opensensor.

A.1 Installation

The intallation of a software program can depend on the operating system. As we mentioned before,we will use Microsoft Windows for this purpose. The installation process of the MPLAB is quite easy; we just have to follow the instructions that will appear on the screens.

First, we have to click twice on the exe file that can be found at [19]. In the second step, we press the *Next* button on each screen, until the *Welcome menu* appears; in that case, we have to choose the *Modify* option where the features to install can be checked. An important aspect is to make sure that the *Third party applications* are NOT selected.

After these actions, we have to keep on pressing the *Next* button on each screen, until a warning window appears; informing us that the PC must be rebooted. Some of the installation screens can be seen in figure A.1.

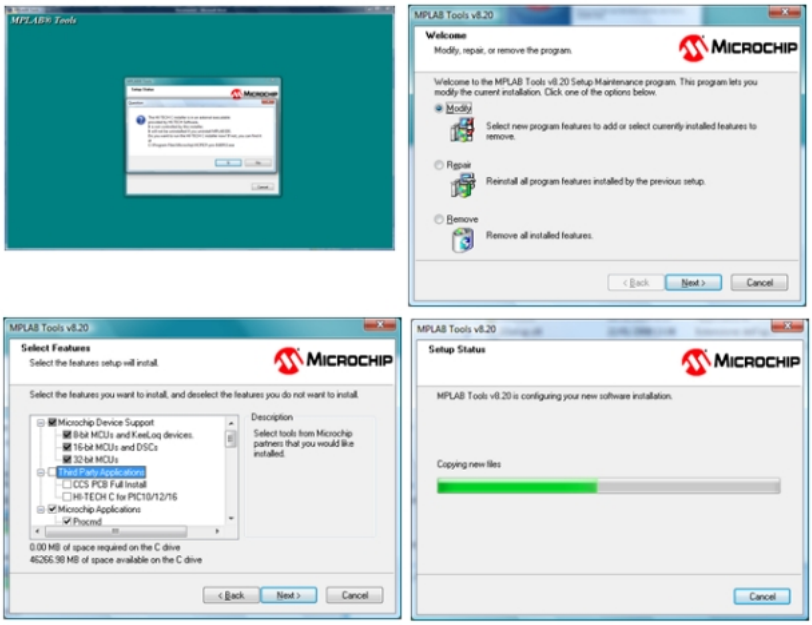

Figure A.1: Phases of the installation process

A.2 Programming Environment

MPLAB is a programming environment that supports C and Assembler languages. Also, it has a Graphical User Interface, making the programming task easy [1]

Now we describe shortly the User Interface. On the left there is the Workspace window. In this window, we link the library files *libp30F3013-coff.a* and *libp30F3013-elf.a* to the current project. We also do the same with the Linker *script (p30f3013.gld*. We can link the source and the header files.

An *Output window* can be found in this Windows version, where the information about the building process (success, or any possible errors) and the status of the PICkit2 device appear.

The PICkit is the hardware connector we use to program the Opensensor.

The last two windows show the linked file source that allows the user to edit the code files, using a well visible structure with highlighted keywords.
An MPLAB IDE screenshot is shown in figure A.2.

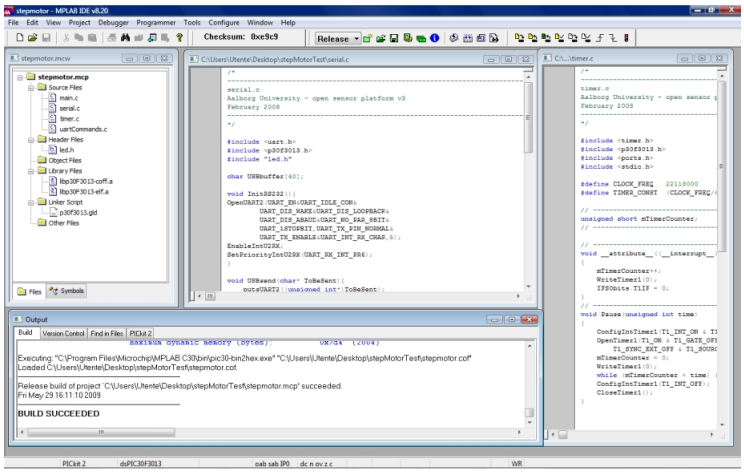

Figure A.2: User Layout Interface of the MPLab software

Appendix B

Setting up an Ad-Hoc network

In this appendix how to setup an Ad-Hoc network with Nokia N95 mobile phones is explained step by step.

B.1 Setting up an Ad-Hoc network with Nokia N95 Mobile Phones

Step 1

Two Access Points have to be setup on the two phones.

	Phone 1	Phone 2
Acces Point Name	NokiaNet	nokiaNet2

Table B.1: Acces Points Names

After that, the *settings* window will appear. In this screen, we have to select the *phone settings* option, as shown in the screeshot B.1.

In the *Connection* window, the *Access Points* option has to be selected.

For creating a new *Access Point*, select *Options*. Later the *New Access Point* option

In the table B.2, we can find the settings we need to fullfil.

After the previous step, we have to select *Options* and inmeadiately *Advanced Settings* to set up the IP Address on the two phones.

Now, the *IPV4 Settings* have to be fulfilled with the following parameters:

Step 2

After setting up the two *Access Points* on the two Phones we have to implement the Python codes for sending and receiving the packets; the Phone1 will be the receiver and the Phone2 the sender.

Figure B.1: Settings Screen on a N95 Mobile Phone

Figure B.2: Connection Screen on a N95 Mobile Phone

In this way the Phone2 will send the messages *Text1*, *Text2*, etc , and the Phone1 will receive them.

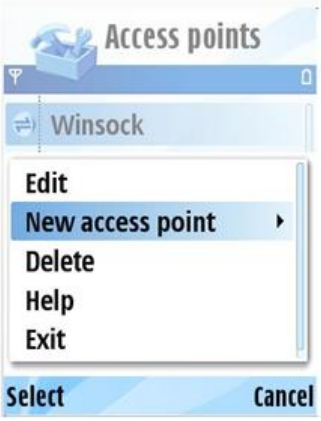

Figure B.3: Access Points Screen on a N95 Mobile Phone

	Phone 1	Phone 2
Connection Name	NokiaNet	NokiaNet2
Data Bearer	Wireless LAN	Wireless LAN
WLAN Network Name	NokiaNet	NokiaNet
Network Status	Public	Public
WLAN Network Mode	Ad-hoc	Ad-hoc
WLAN Security Mode	Open Network	Open Network
WLAN Security Settings	Empty	Empty
Homepage	None	None

Table B.2: Settings for the two access points on the phones

Figure B.4: NokiaNet Screen on a N95 Mobile Phone

	Phone 1	Phone 2
Phone IP Address	172.16.0.2	172.16.0.1
Subnet Mask	255.255.255.0	255.255.255.0
Default Gateway	172.16.0.2 (optional)	172.16.0.1 (optional)
DNS Address	Automatic	Automatic

Table B.3: IPV4 Settings

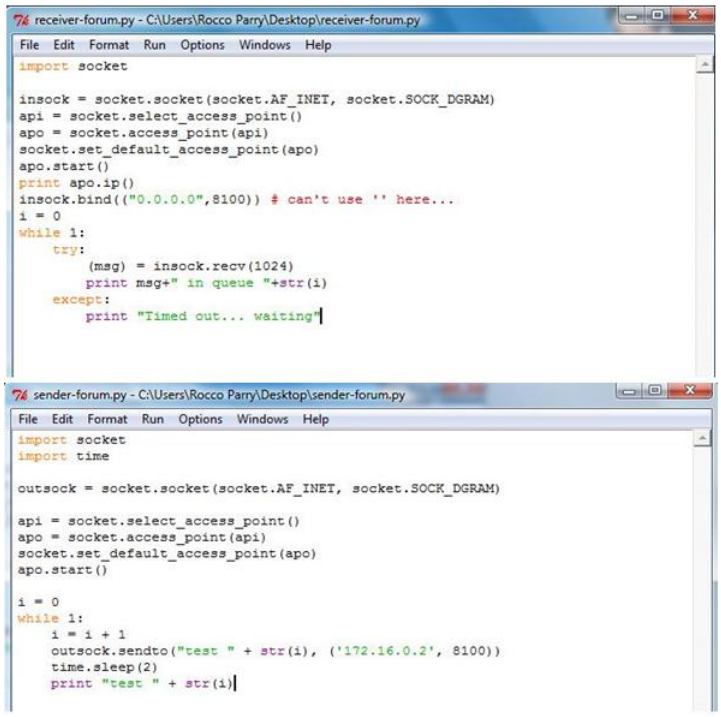

Figure B.5: From top to bottom: Codes on the phone1, receiver, and on the phone2, sender

Appendix C

Setting up a communication between the Robots and the Graphical User Interface

In this appendix we explain, step by step, how to connect the robots (with Nokia N95 mobile phones on board) with the PC by using the Graphical User Interface (GUI)

Step 1

An ad-hoc network on the PC has to be created by giving a name to this network. Furthermore, it is important to select the *no authentication method* in the settings screen, as shown in the screenshot C.1.

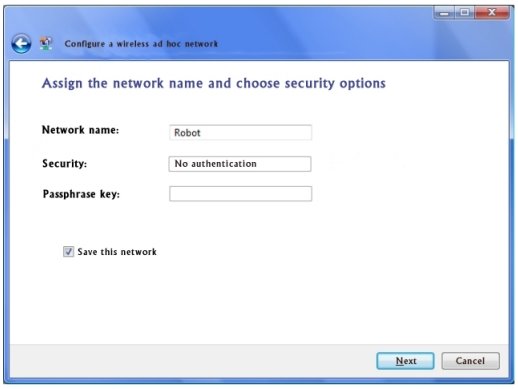

Figure C.1: Window: Configuring a new ad hoc network

Once the network is created, the PC has to be connected to it. After this, in the wireless network menu screen will appear *Waiting for new users* screen appersin the .

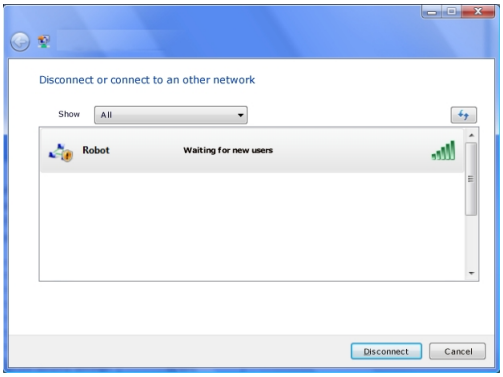

Figure C.2: Window: Wireless network status

Step 2

Now we have to set up an ad hoc network on the mobile phone with similar steps of previous paragraph, using the following parameters for the mobile phone

Connection Name	R2D2
Data Bearer	Wireless LAN
WLAN Network Name	Robot
Network Status	Public
WLAN Network Mode	Ad-hoc
WLAN Security Mode	Open Network
WLAN Security Settings	Empty
Homepage	None

Table C.1: Ad hoc network parameters on the mobile phone

The connection name is the local identifier of the connection, so it is not a nodal point in the settings; instead the WLAN Network Name has to be obviously the chosen name when the wireless network is created on the PC. Then, in the Advanced settings menu, we modify the IPv4 settings on the mobile phonewith the following parameters

Phone IP Address	169.254.22.26
Subnet Mask	255.255.255.0
Default Gateway	169.254.221.54
DNS Address	Automatic

Table C.2: Advanced settings on the mobile phone

Now the network is created for both of the sides (PC and mobile phone), and the robots can be connected to the GUI.

Step 3

The GUI has to be executed before the scripts on the phones, and so it starts to listen to any possible incoming robot; while, on the phone, when the script is executed a setting form appears on its screen; if the script runs for the first time in that mobile, the form is empty and it has to be filled in all of its fields with robot`s data (Name, IP, and MAC) and with the IP address of the PC. If this one is not known, we put a temporary address for this time; obviously the GUI cannot work, but anyway this procedure is needed to know the IP address of the PC.

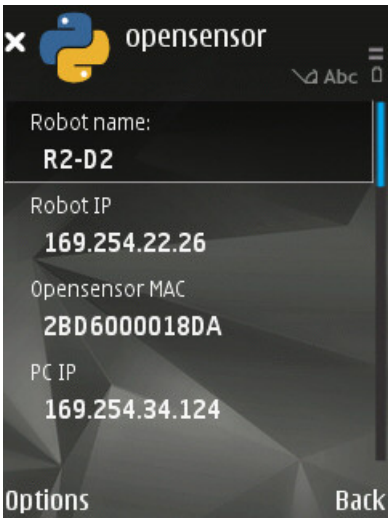

Figure C.3: Python Screen, Settings Parameters on the Robot

After that, a list of possible Bluetooth devices is shown, and we have to select the OpenSensor which is in the robot with that phone; now, there is the selection of the Bluetooth communication port (usually the *COM1 port*), and then we have to select the wireless network to connect the mobile to the PC, and the Connection Name (in the Mobile Phone parameters table) has to be chosen.

Now the phone is connected to the wireless network and now the status on the PC changes in *Connected*.

So in the Status menu of the connection we can open the window *Details*, finding the IP address of the PC.

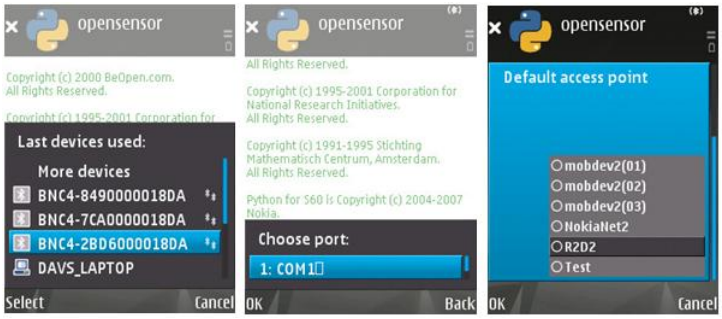

Figure C.4: From the left to the right: Device Selection Screen, Port Selection Screen and wireless network Selection Screen

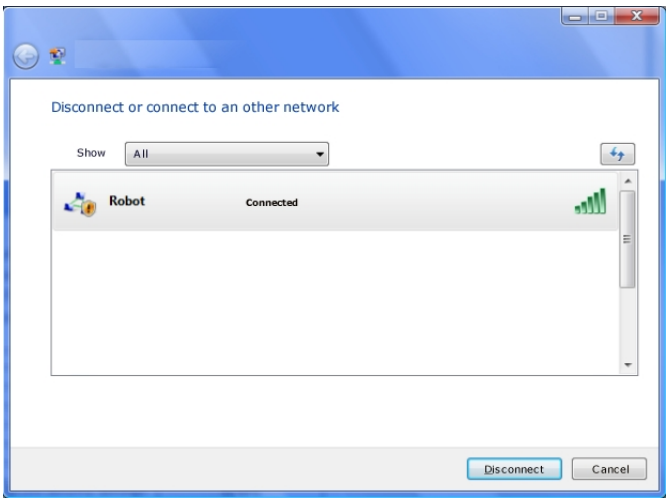

Figure C.5: Status of the Robot connection

Step 4

Now we can execute again the script on the phone, and this time the Robot data fields are already filled, so we have to fill the IP PC field with the address just obtained (when the script will be executed again, all the fields will be filled as default with the last data); after selecting the Bluetooth device, the BT communication port and the wireless network, the mobile phone sends all its information to the GUI and the connection is established.

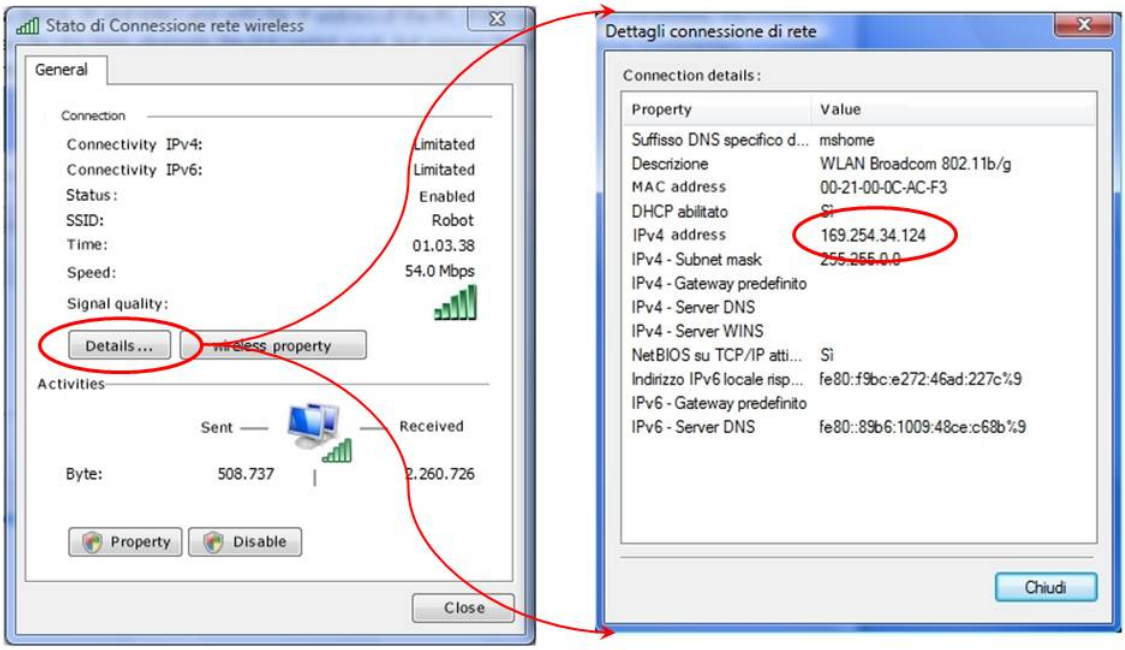

Figure C.6: Wireless Connection Properties

Bibliography

[1] G. Perrucci A. Grauballe and F. Fitzek. Opensensor - an open wireless sensor platform. July 2008.

[2] Robotics Active Robots and Electronics technology. Airat2 micromouse body set. Web site, May 2009. http://www.active-robots.com/products/platforms/microrobot-chassis.shtml.

[3] Ian F. Akyildiz and Xudong Wang. A survey on wireless mesh networks. *Communications Magazine, IEEE*, 43:S23– S30, September 2005.

[4] Todd E. Hunt Aria Nosratinia and Ahmadreza Hedayat. Cooperative communication in wireless networks. *Communications Magazine, IEEE*, 42:74–80, October 2004.

[5] Robot Dreams. Robonova-1 robot intuitive gui control interface. Web site, May 2009. http://www.robots-dreams.com/2006/03/robonova1_robot.html.

[6] BIPOM Electronics. Mini-max/908-c single board computer technical manual. Web site, May 2009. http://www.bipom.com/documents/boards/minimax908c.pdf.

[7] M. Katz F. Fitzek and Q. Zhang. Cellular controlled short-range communication for cooperative p2p networkin. 2006.

[8] netlogo community model Federico Albiero. Wireless-coop-mobile, netlogo. Web site, May 2009. http://ccl.northwestern.edu/netlogo/models/community/Wireless-Coop-Mobile.

[9] Frank H.P. Fitzek and Marcos D. Katz. *Cooperation in Wireless Networks: Principles and Applications - Real Egoistic Behavior is to Cooperate!* Springer, April 2006.

[10] F. Fitzek G. Perrucci and M. Petersen. *Heterogeneous Wireless Access Networks: Architectures and Protocols*. Springer, 2008.

[11] F. Fitzek G. S. B. Pietrarca, G. Perrucci and M. Katz. Measurement campaign on connectivity of mesh networks formed by mobile devices. 2007.

[12] Xudong Wang Ian F. Akyildiz and Weilin Wang. *Wireless mesh networks: a survey*. Computer Networks and ISDN Systemsr, March 2005.

[13] Mobile Robbots Inc. Mobileeyes tm, robot control gui. Web site, May 2009. http://www.mobilerobots.com/MobileEyes.html.

[14] Parallax INC. Basic stamp figure. Web site, May 2009. http://www.parallax.com/Store/Microcontrollers/BASICStampModules/tabid/134/CategoryID/9/List/0/Level/a/ProductID/1/Default.aspx?SortField=ISBN%2cISBN.

[15] Parallax INC. Basic stamp syntax and reference manual version 2.1. Web site, May 2009. http://www.parallax.com/dl/docs/prod/stamps/BasicStampMan.pdf.

[16] Rabbit Inc. Bl2100 single-board computer. Web site, May 2009. http://www.rabbit.com/products/bl2100/.

[17] A. Iera L. Militano, F. Fitzek and A. Molinaro. On the beneficial effects of cooperative wireless peer to peer networking. September 2007.

[18] A. Molinaro L. Militano, A. Iera and F. H. P. Fitzek. Wireless peer-to-peer cooperation: when is it worth adopting this paradigm? September 2008.

[19] Aalborg University Mobile Devices Group. Opensensor. Web site, May 2009. http://mobiledevices.kom.aau.dk/projects/00/.

[20] Forum Nokia. Nokia energy profiler quick start. Web site, May 2009. http://www.forum.nokia.com/Technology_Topics/Application_Quality/Power_Management/Nokia_Energy_Profiler_Quick_Start.xhtml.

[21] The Arduino project. Arduino. Web site, May 2009. http://arduino.cc/.

[22] Python Robotics Pyro. Pyro, robotics platforms. Web site, May 2009. http://pyrorobotics.org/?page=PyroHardware.

[23] F.H.P. Fitzek Q. Z. M. K., G.P. Perrucci. Cooperative mobile web browsing. *EURASIP Journal on Wireless Communications and Networking*, 2009.

[24] International Telecommunication Union. Worldwide mobile cellular subscribers. Web site, June 2009. http://www.itu.int/newsroom/press_releases/2008/29.html.

[25] Aalborg University. Department of electronic systems. Web site, May 2009. http://vbn.aau.dk/research/(312)?language=sec&AnonymousLoginFilter_language=sec.

[26] Aalborg University. Mobile devices group. Web site, May 2009. http://mobiledevices.kom.aau.dk/.

List of Figures

List of Tables