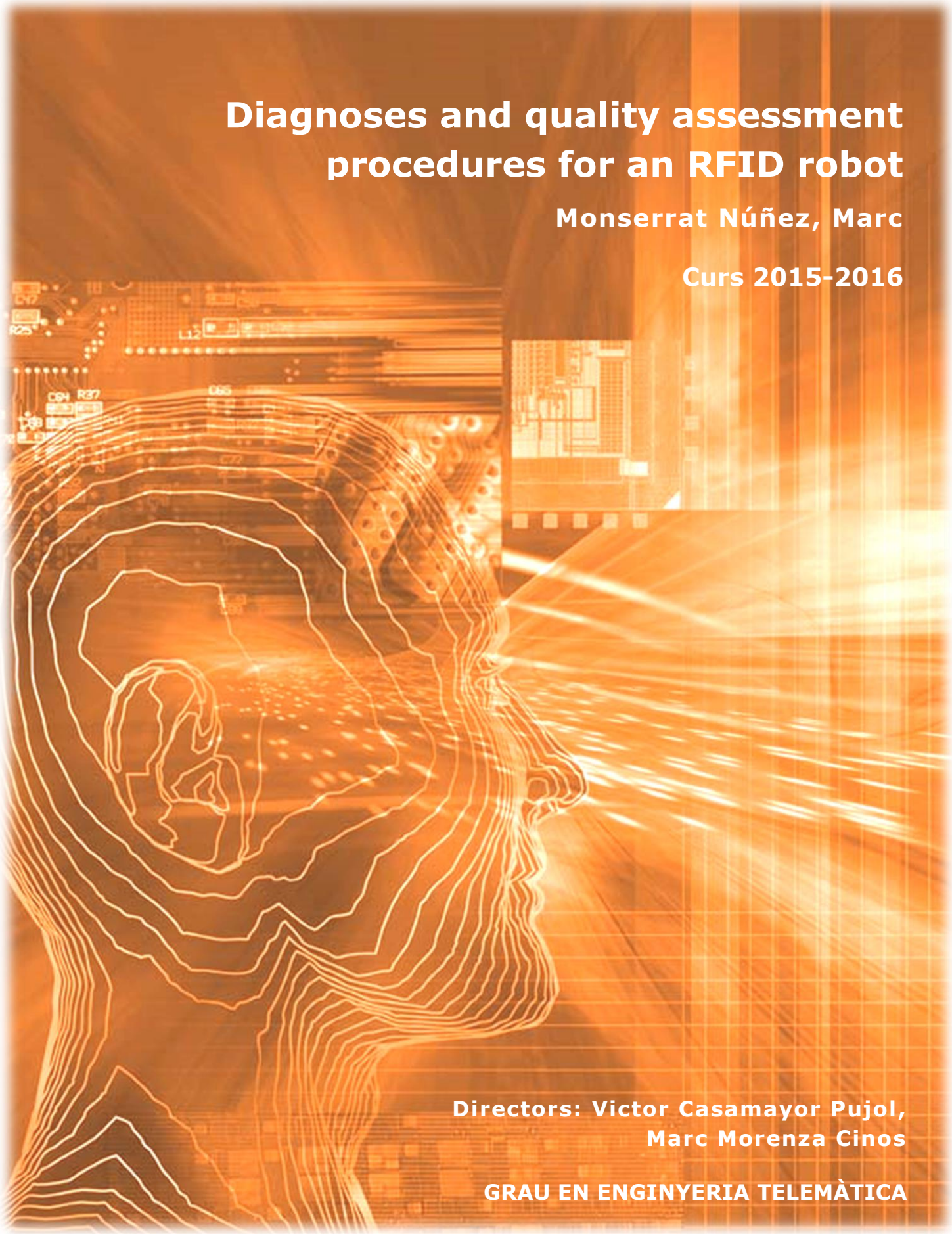
Diagnoses and quality assessment procedures for an RFID robot

Monserrat Núñez, Marc

Curs 2015-2016



Directors: Victor Casamayor Pujol, Marc Morenza Cinos

GRAU EN ENGINYERIA TELEMÀTICA

upf. Universitat Pompeu Fabra Barcelona
Escola Superior Politècnica

Treball de Fi de Grau

Diagnoses and quality assessment procedures for an RFID robot

Marc Monserrat Núñez

TREBALL FI DE GRAU
GRAU EN ENGINYERIA TELEMÀTICA
ESCOLA SUPERIOR POLITÈCNICA UPF
2016

DIRECTORS:
Victor Casamayor Pujol
Marc Morenza Cinos

Acknowledgements

I would like to thank Rafael for giving me the opportunity to do this dissertation in Keonn and to Victor and Marc for helping me on the day-a-day.

Abstract

A company must verify its products' quality before serving them as clients will expect a correct performance. This is crucial when the product is a robot which has a lot of components that influence its behaviour.

This dissertation defines a set of procedures for the diagnosis and quality assessment of a specific RFID robot which lacks a methodology to check if it satisfies the design requirements.

A diagnoses mechanism is developed to find out if any of the robot's components are inoperative. A list of processes is then created so that the person assigned to the robot follows them in order to test the quality of the robot and its components before being dispatched fulfilling the specifications. Finally, an interface improves the usability of both diagnoses and quality assessments.

These procedures will not only help to test the robot but also analyse its weaknesses and correcting them in future models.

Resum

Quan un producte és fabricat es comproba la seva qualitat abans de ser venut esperant un rendiment establert.

Aquest Treball de Fi de Grau es centra en definir uns procediments per realitzar diagnòstics i evaluacions de qualitat a un robot RFID el qual manca d'una metodologia per saber si compleix els requeriments de disseny.

S'han desenvolupat uns mecanismes per trobar si algun dels components del robot falla. Després s'ha creat una llista de procediments perquè siguin seguits pel tècnic encarregat del robot per tal evaluar el comportament del robot abans de ser entregat al client. Finalment s'ha creat una interficie per millorar la usabilitat tant dels diagnòstics com de les evaluacions de qualitat.

Aquests procediments no només ajudaran a examinar al robot, també serviran per analitzar els seus punts febles i poder-los corregir en futurs models.

Index

1. INTRODUCTION

The content of this dissertation focuses on a robot called AdvanRobot. It turns the effort and time on doing an inventory in retail shops in an easy task thanks to the Radio Frequency Identification (RFID) technology. As many concepts about AdvanRobot's behaviour, RFID technology and different software used are mentioned throughout this report, a basic knowledge about these concepts is needed to understand it.

1.1. AdvanRobot

AdvanRobot moves around retail shops autonomously reading RFID labels found on products. A label is actually a tag wrapped under layers of fabric, paper or other materials. By doing this, the robot can take an inventory of the shop in a matter of minutes. This is something that done manually could take a lot of days and employees to be completed and besides the robot's precision is much higher than a human's.

Thanks to making inventories regularly it provides the ability to easily control the shop's stock and replenish it. It is also capable to do 3D maps of the shops and divide the shop in different areas. Thanks to dividing the shop in areas it can be ordered to take an inventory just of a concrete area and having the inventories of different areas separately. This could be analysed after and identify which areas of the shop have better sales.

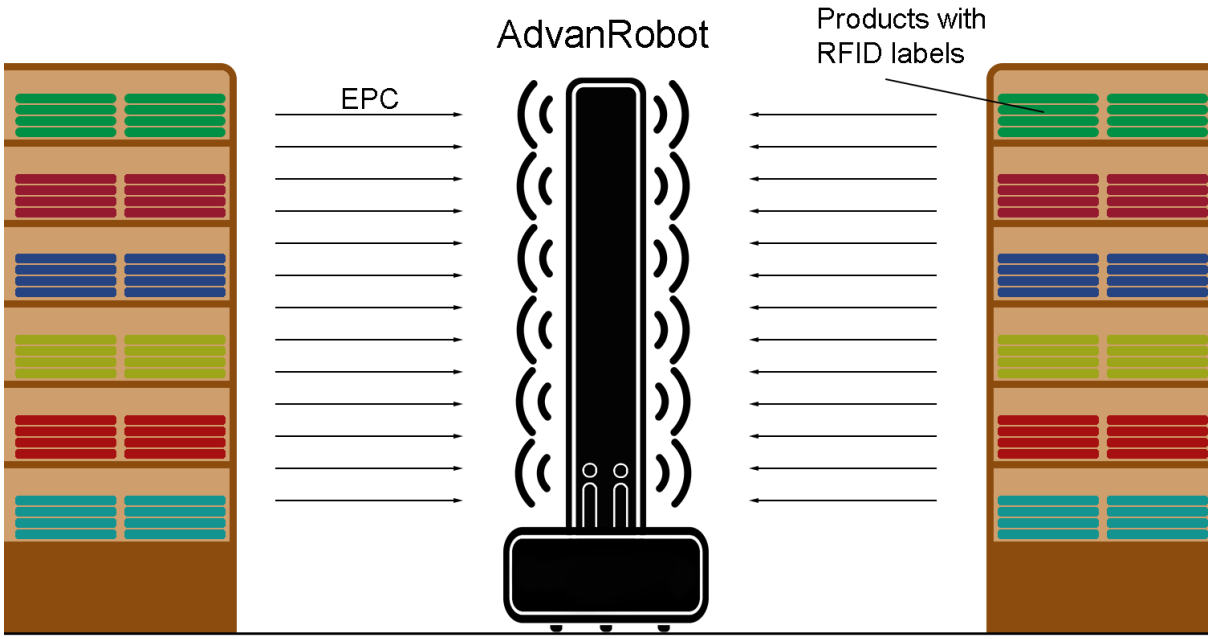


Figure 1.1. AdvanRobot reading tags from shelfs

AdvanRobot can be divided in two sub-parts:

a) Robotic base

It contains almost every robotic component of the whole robot. The robotic base is a metallic cylinder of 25 cm of radium and 23 cm of height. The most remarkable things it contains are:

1- A computer which has the needed software to operate the robot.

2- Wheels with traction power that enables the robot to move.

3- Omni-directional wheels allow it to turn.

4- Motors give traction to the wheels.

5- A WiFi router to connect the robot to networks.

6- A switch that distributes the signal between the readers and the computer.

7- A laser sensor which detects walls and objects in front of it. The object has to be placed 20 cm from the floor as much to could be seen by the laser.

8- An Inertial Measurement Unit (IMU) sensor. An IMU is used by the AdvanRobot to know its actual rotation and angular rate.

9- A security red button that engages and disengages the wheels from the motor. This is useful to avoid accidents or just to push the robot when is desired.

b) Payload

The payload is placed on the robotic base. It is a 185 cm height prismatic column with a rectangular base. Inside it are 12 antennas situated along both sides which are in charge of communicating with the tags in the shop. The payload's height allows the robot to read the tags placed on various shelf levels within the shop. There are also three RFID readers. In each reader a group of 4 antennas is connected.

The payload is split into two twin parts separated by hinges which allows it to be folded in order to move it around easily, for example, being able to enter a lift. A mechanism is designed to keep the robot upright when unfolded.

At the top of the payload holds a 3D camera which allows the robot to see objects as it moves therefore being able to react accordingly to unexpected obstacles on its path. It also allows to create 3D maps.

1.1.1. Main AdvanRobot's operations

Two of AdvanRobot's main operations are: mapping the shop and performing inventories. It has actually an app that allows the operator to manage these operations. When the robot is not moving autonomously the operator can move it thanks to a remote controller or just pushing it when it is disengaged by pulling the security red button.

These operations also need a previous setting on the shop. It must be equipped with at least two Quick Response (QR) codes located around the shop's walls at approximately 1.5 meters from the floor. These QR codes are used to divide the shop in sections to perform inventories separately as it has been mentioned. One of the codes marks the start of one of the sections and another code marks the end. At the same time, the end one acts also as the start of the next section.

a) Mapping

Once AdvanRobot arrives into a new shop it has to map it to memorize its surroundings. To carry out this, the operator places the robot in front of the QR code corresponding to the desired section that wants to be mapped.

The robot reads it thanks to its 3D camera and recognizes the section which is shown on the app. The operator indicates on the app that it is going to start the mapping and afterwards controls the robot through the shop using the remote controller. The robot must do the route desired to be performed on the inventory operation and finish on the next QR code. Once it reads the code it processes the map and saves it.

a) Inventory

Having a map saved, the operator can place again the robot at the start QR code and begin an inventory on the app. The robot reads the code like the mapping process and starts traveling autonomously through the route that was saved finishing on the next code. On its travel, AdvanRobot reads the tags that founds on its route and saves the list of EPCs into a file.

1.2. Context

The most important subjects related to AdvanRobot on this dissertation are RFID, OS and Django:

1.2.1. RFID

The RFID technology is capable to identify objects from a distance thanks to radio frequency signals emitted from an antenna connected to a microchip. This combination of chip and antenna is more commonly known as a tag and it is found on the object which is desired to be identified. Following the scheme on the Figure 1.2. the tag communicates with an antenna connected to a reader which has the role of receiving the signal broadcasted by the tag and processing the information saved on it.

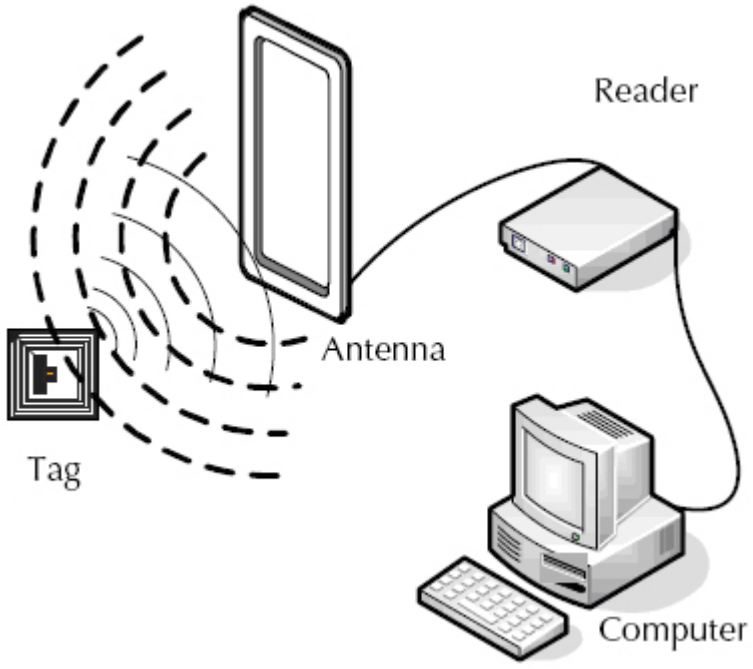


Figure 1.2. Tag and reader communication [3]

RFID follows the concept of barcode identification systems but it differs from them. The RFID tags are identified by means of a unique code called the Electronic Product Code (EPC) which is saved into the chip. The EPC codes typically have 96 bits normally shown in hexadecimals, which gives the possibility to have more or less $7.9 \cdot 10^{28}$ different codes. This way it can be guaranteed that every single product has its own distinct EPC, for example a

million units of the same type of shirt in a range of colours will have the same barcode, but each single shirt will have their individual EPC.

There are different RFID systems depending on the communication between the reader and the tag: active systems, passive systems and battery assisted passive systems.

AdvanRobot is a passive system where tags work without a battery. To communicate the reader sends a signal and the tag uses the signal's power to send the energy back to the reader. This reflected signal is called backscatter. As long as the tags do not need a battery to work the tags are smaller, but due to the limited power of the backscatter they are also limited in range. It depends on the frequency band it works but typically the range is lower than 10 meters.[4][5] AdvanRobot's system operates on Ultra-High Frequency (UHF) at around 865 MHz.[6]

A passive system is the best solution for its purpose because it can move so it does not need a large signal to reach everywhere. Moreover, tags have an almost unlimited life because of their energy autonomy.

1.2.2. ROS

The computer that AdvanRobot carries inside is what connects the components together. It works with Ubuntu operating system which is open source.[7]

But developing a robot from scratch is a really difficult job. A lot of software libraries are needed before starting to developing the robot itself. Some middlewares and frameworks provide developing libraries to make this job easier.

"The Robot Operating System (ROS) is a flexible framework for writing robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms." [8]

AdvanRobot uses ROS on its implementation so some basic concepts about it are needed.

A node is a process on the ROS network that performs computation. The IMU's node is for example the /mavros.

There are two types of nodes, publishers and subscribers. A publisher node generates messages and posts them on ROS topics. On the other hand, a subscriber node reads the messages published on the topics. A node can be publisher and subscriber at the same time.

ROS topics are where the messages are published and the way ROS provides to communicate between nodes. The /mavros node for example publishes messages into the /mavros/imu/data_raw which is subscribed by another node.

The Figure 1.3. shows a sample of the diagram that links nodes through topics. It can be noticed how the /mavros node communicates with the /complementary_filter_node though the /mavros/imu/data_raw topic.

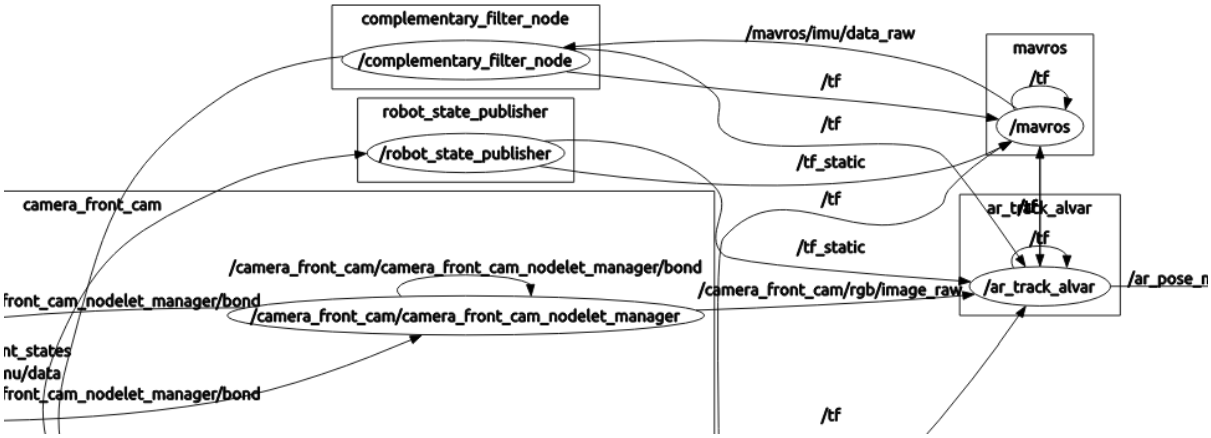


Figure 1.3. AdvanRobot's nodes and topics relations graph

1.2.3. Django

Some Python scripts have been developed alongside a Web application throughout the dissertation. As long as the scripts on the backend are implemented in Python and the frontend is implemented in HyperText Markup Language (HTML) and Javascript which do not support Python by themselves, a framework is needed to connect both parts.

Django is a powerful framework designed to use Python in Web applications. It is free and open source, and it also adds some features like a Structured Query Lite (SQLite) database and authentication support.[15] Django is commonly the solution chosen by professional web applications that want to use Python, for example Youtube, Dropbox or Google.

As is shown on the Figure 1.4. to call a Python function, Django uses a file which contains a mapping that links each function to an URL. Sending an HTTP Request to an URL returns an HTTP Response with the result of the function mapped.

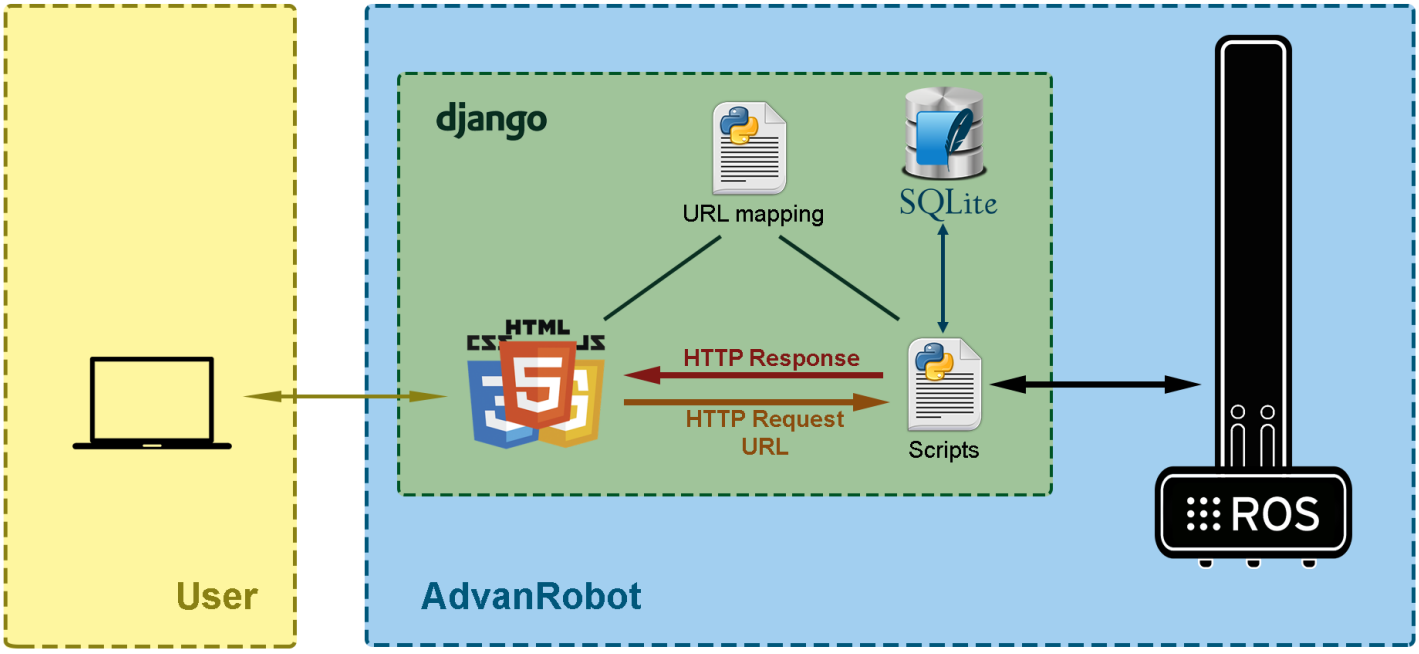


Figure 1.4. AdvanRobot Mainetance Tool implementation

On the code below is shown how the URL "/maintenance/diagnostics/" (where /maintenance is the root path) is mapped to the function diagnostics on the file views.py. On this example, the function initializes a ROS node to be used on the Diagnosis Tool and returns the template of its main page to be rendered.

```
urls.py:
url(r'^diagnostics/$', views.diagnostics, name='diagnostics')

views.py:
def diagnostics(request):
      rospy.init_node('diagnostics_ros')
      return render(request, 'maintenance/diagnostics.html')
```

The reason of choosing a Web application instead of a desktop or phone one is that AdvanRobot's application used to manage its operations is an Android app but it is truly developed as a Web application. This will ease their adaptation and unification if it is desired on a future.

1.3. The need of maintenance

Nowadays it is rare not to see a factory using automated machines at some point in the assembly line. But even with their accuracy they do not guarantee that the products will be manufactured as good as the original design. However, government laws, contracts and agreements with the clients or customers require that the product must have some established characteristics.

This forces the manufacturers, assemblers and sellers to add a last step on the manufacturing process where it is controlled the product's quality and it is checked if it has the characteristics to be delivered to the next step of the supply chain.

On a hardware based product like AdvanRobot a quality control does not guarantee either that it is going to keep its quality or work forever. AdvanRobot's maintenance consists on its periodically supervision and reparation of its components since its birth until it is considered not useful anymore. This includes both quality assessments of its behaviour and diagnoses of its components, repairing them when necessary.

However, the actual methods to detect which of the components is failing or not working how they should be, take too much time and effort. Furthermore, there is not a detailed process of how the quality of the robot's behaviour has to be tested.

1.4. Objectives

This dissertation aims to provide a set of procedures, good practices, guides and tools to empower the operator responsible of AdvanRobot (not necessary trained) to analyse and diagnose its performance. In order to accomplish this main goal, it has been set certain objectives:

A set of procedures has to be found in order to diagnose the components of the robot. These procedures have to be implemented into scripts.

For the robot's operation have also to be defined some performance testing procedures. Each test would be considered as passed by comparing its results with an acceptable threshold.

Finally, both diagnostics and quality assessment procedures have to be unified into an interface thought to be used by the operator.

1.5 Motivation

These objectives are established following the needs of Keonn which is the company that owns AdvanRobot. Keonn is a small company based in Barcelona that dedicates its activity on selling products based on Radio Frequency Identification (RFID) technology. They provide RFID solutions to retail shops in order to improve their businesses.

Keonn has offered this dissertation to work alongside their employees improving their product. It has also been used their workplace where AdvanRobot has enough space and tools to work adequately.

The main personal goal is ending up with some useful mechanisms that Keonn could really use on their interest, helping to push AdvanRobot into the Internet of Things market.

2. DIAGNOSTICS OF COMPONENTS

AdvanRobot is made of many components. If one of them fails, the whole robot's performance can be affected. This is a nightmare for the operator when he has to repair it if he does not have some help. For each of the components, the operator has to make many checkups through the command-line among other manual ones to try finding the disorder.

"A diagnosis is a set of disorders, the union of whose symptoms include all observed symptoms".[9] It is a common process in Artificial Intelligence (AI) looking for techniques to find out if there is any disorder in a system and if so, which of the components are causing it.

On this chapter is explained how have been developed some tools in order to find any disorder in AdvanRobot's system. This has been done by clustering the checkups the operator has to do into simple scripts which simplifies and automatizes his job. The scripts are developed in Python as ROS just provides developing libraries for Python and C++ programming languages.

As AdvanRobot's behavior depends of many components, the list of components to be diagnosed has been reduced to some of the most critical for its performance:

1- Laser

2- Camera

3- IMU

4- RFID system

Next is explained how each component's checkup has been solved into Python scripts. It is also explained how they have been tested forcing the components to fail, which could be the issues and some possible solutions to solve the disorder.

2.1. Robotic components

The Laser, the camera and the IMU belong to the ROS network and they all publish messages into topics when they are working so the methods to check their states are very similar. They consist in four steps which determine not only if there is an anomaly on the component but also at which level has occurred:

1- ROS - Checks if the ROS network is working properly.

2- Connection - Checks if the device is correctly plugged.

3- List of topics - Looks into the list of topics and checks if the topic is there.

4- Topic publication - Checks if the component is publishing messages into its topic.

The order of the three last steps goes from low-level to high-level in order that if one step fails the following ones will fail as well. The first one is a common script for each device and is necessary to be checked first to know if the ROS commands will work on the other steps. This order narrows the possible origins of the disorder into two of the steps and helps to discards possible causes of the problem.

2.1.1. Step 1: ROS

The script for the first step is exactly the same for the laser, camera and IMU diagnostic.

It is assumed that ROS is already installed into the robot, but even with it installed, the ROS own commands can fail if it is not configured correctly. Some of the next steps use these commands so the first thing to do is ensure that ROS works correctly and the problem is not coming from here.

To find out this, the script looks into the list of topics and checks if there is a topic called /rosout. As long as "ROS client libraries are required to publish console logging messages to the /rosout topic"[10] it has to be on the list of topics if the ROS network is operative.

```
rospyWorks = False
try:
      topiclist = subprocess.check_output(["rostopic", "list"])
      for i in range(0,len(topiclist)):
            if topiclist[i:i+7] == "/rosout":
                  rospyWorks = True
                  break
except:
      rospyWorks = False
```

It has been tested killing the /rosout topic through the command line and it fails the checkup.

Possible issues:	**Possible solutions:**
-ROS not installed. -/rosout topic dead.	-Install ROS. -Reboot AdvanRobot. -Reset ROS network.

3.1.3. Step 2: Connection

It is not unusual that on the assembling process some components are forgotten to be connected. They can also be disconnected due to continuous swing of the robot. This script looks up if the component is connected to the computer.

The camera and the IMU are connected via an Universal Serial Bus (USB). To detect if the camera and the IMU are connected the script looks into the /dev Linux directory. This directory contains special files for each device connected to the computer.[11] This also includes the USB ones which can be set by means of "udev rules", so the script checks if the name of the component is on the list of file names of the directory.

```
os.chdir("/dev")
connected_devices = os.listdir("/dev")
for i in range(0, len(connected_devices)):
      cameraIsConnected = False
      if(connected_devices[i] == camera_usb_name):
            cameraIsConnected = True
            break;
```

The laser is connected via an Ethernet connection so alternatively a ping is done to its IP. If the ping reaches the laser within a timeout means it is connected.

```
laser_response = os.system("ping -c 1 " + laser_ip)
if laser_response == 0:
      laserIsConnected = True
else:
      laserIsConnected = False
```

This is simply tested by unplugging the cables from the robot.

Possible issues:	**Possible solutions:**
-Component unplugged.	-Plug the component.
-Component not properly connected.	-Configure udev rules.
-Damaged connector.	-Change the cable.
-Udev rules not set.	-Change to another USB port
-Laser in different IP.	-Check laser IP.

3.1.2. Step 3: List of topics

Once the ROS commands are working, the next step consists on checking if the topic that is published when the device works is actually on the list of topics. This ensures that the topic have been initialized at some point but does not ensures that it is actually being published.

Each component has a node that publishes into their assigned topic. These topics are:

Laser → /scan_raw

Camera → /camera_front_cam/depth/points

IMU → /mavros/imu/data_raw

```
laserInTopiclist = False;
try:
      laser_response = subprocess.check_output(["rostopic", "list"])
      print laser_response
      for i in range(0,len(laser_response)):
            if laser_response[i:i+9] == "/scan_raw":
            laserInTopiclist = True
            break
except:
      laserInTopiclist = False
```

As the first step for some reason the topic could have been killed, so it has been tested by killing each topic.

Possible issues:	Possible solutions:
-Topic dead. -Device damaged	-Reboot AdvanRobot. -Reset ROS network.

3.1.4. Step 4: Topic publication

After knowing that the component is connected, the last remaining step reveals if it really works and it is publishing messages.

The rospy method *wait_for_message* creates a subscription to the topic given on the method's parameters. If any node publishes on that topic during a timeout it returns the first message published and then it unsubscribes from the topic.[12] Some topics have a little bit of delay so enough timeout is set to ensure receiving a response if the device does work.

```
laserPublishes = False
try:
     rospy.wait_for_message("/scan_raw", LaserScan, timeout=5)
     laserPublishes = True
except:
     laserPublishes = False
```

Killing the node that publishes into the topic returns a failure. It could happen that another node with the same name makes the other one die. The first try to repair it is to revive the node.

Possible issues:	Possible solutions:
-A new node with the same name has killed the correct one. -The device is damaged	-Reset the node. -Reboot AdvanRobot. -Reset ROS network. -Replace the device

2.2. RFID system

There are mainly two types of components on the RFID system: the readers and the antennas. They can be checked separately.

2.2.1. Readers

The antennas are connected to the readers so the readers have to be checked first. As they are also plugged to the computer through an Ethernet connection, it can also be checked if they are connected by doing a ping to their IP.

```
readerIsConnected = False
laser_response = os.system("ping -c 1 reader-0" + reader)
if laser_response == 0:
      readerIsConnected = True
else:
      readerIsConnected = False
```

It has been tested a disorder by unplugging the readers from the computer.

Possible issues:	**Possible solutions:**
-Reader unplugged from the computer. -Reader not connected properly. -Reader's connection damaged. -Damaged cable. -Damaged reader. -Bad connection between reader and power supply.	-Plug properly the Ethernet connections. -Replace the cables. -Replace the reader.

2.2.2. Antennas

Last components tested are the antennas which are neither on the ROS network so they do not publish messages into topics. Not only has to be checked that the robot read tags but also that each individual antenna does.

For this part, a previous setting has to be set by the operator because the antennas are going to read tags in order to know if they are working.

First of all, a strip of six tags is needed. There has to be a separation of 28 cm between tags, which is the distance between the centre of two adjacent antennas. Each tag will be assigned to two antennas as the strip is reused for both sides of the robot. This strip can be something flexible like a rope or a tape easy to store, or it can be a fixed structure (Figure 2.1.).

Figure 2.1. Fixed test structure of RFID tags

This tag and antenna assignment needs to be reflected on two configuration files that are stored on the robot's memory, one for each side of the robot. They contain the relation between each antenna, tag's EPC and reader.

The distribution of the readers, antennas and ports should also follow established rules:

-The reader number 1 contains the four antennas at the bottom of the reader, two from each side. The reader 2 contains the ones in the middle and the reader 3 the ones at the top.

-The antennas are numbered from 1 to 12, counting from right to left and from bottom to top.

-Each antenna on each reader has a port number that goes from 1 to 4, also counting from right to left and bottom to top.

The Figure 2.2. represents this distribution where X in AX is the antenna number and Y in PY is the port where it is connected.

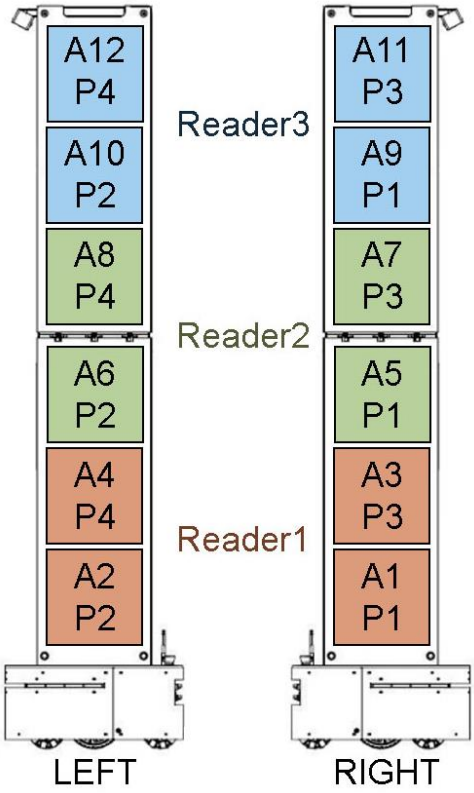


Figure 2.2. AdvanRobot's antennas, readers and ports distribution

So finally, the configuration files should look like the examples on the Figure 2.3. The three parameters separated by commas are: the EPC, the port and the reader.

```
000300000000000000000a,4,3
000300000000000000000b,2,3
000300000000000000000c,4,2
000300000000000000000d,2,2
000300000000000000000e,4,1
000300000000000000000f,2,1
```

```
000300000000000000000a,3,3
000300000000000000000b,1,3
000300000000000000000c,3,2
000300000000000000000d,1,2
000300000000000000000e,3,1
000300000000000000000f,1,1
```

Figure 2.3. Antennas configuration files

With this setting ready, the operator has to put the strip of tags in front of one robot's side at approximately 30 cm from it and run the script of the corresponding side.

The script opens a socket on the computer and on each reader and reads the data broadcasted by them. This string contains an eXtensible Markup Language (XML) formatted text for each tag that is being read on that moment. The XML lists information of the read like a timestamp, the tag's EPC and the name of the reader. A received XML looks like this:

```
ADVANNET/1.0
Content-Length:961
Content-Type:text/xml

<?xml version="1.0" encoding="UTF-8"?>
<inventory>
    <type>inventory</type>
    <ts>1460390332000</ts>
    <status>OK</status>
    <msg-version>2.3.0</msg-version>
    <op>inventory</op>
    <data>
        <advanNetId>AdvanNet-instance-54:4a:16:be:1c:4b--1</advanNetId>
        <deviceId>reader-03</deviceId>
        <inventory>
                <class>INVENTORY</class>
                <deviceId>reader-03</deviceId>
                <size>1</size>
                <items>
                        <item>
                                <class>READ_EVENT</class>
                                <epc>000300000000000000000001</epc>
                                <ts>1460390332924</ts>
                                <deviceId>reader-03</deviceId>
                                <data>
                                        <class>TAG_DATA</class>
                                        <hexepc>000300000000000000000001</hexepc>
                                        <props>
                                                <prop>TIME_STAMP:1460390332924,</prop>
                                                <prop>RSSI:-80,</prop>
                                                <prop>RF_PHASE:92,</prop>
                                                <prop>ANTENNA_PORT:4,</prop>
                                                <prop>FREQ:867341,</prop>
                                        </props>
                                </data>
                        </item>
                </items>
        </inventory>
    </data>
</inventory>
```

Figure 2.4. Example of a received XML

Then the XML is parsed storing the important data: the EPC and the port of the antenna that has read the tag. Once each XML has been parsed, it compares each read with the parameters taken from the configuration file and returns a list of the antennas indicating which ones have read their assigned tag and which ones have not.

The antennas are connected to the readers through Radio Frequency (RF) connectors which could be unplugged if they have not been properly connected on the assembly line. This has been replicated to test them. The operator should look on both sides of the connection and check if they are properly plugged in.

2.3. AdvanRobot' reliability

If an operator performs diagnostics periodically he can get at some point the robot's reliability. There are two important specification values that summarizes how reliable a system is. They are important to be known as they are interesting specifications that the client may ask for.

These measures are the Mean Time Between Failures (MTBF) and the Mean Time To Repair (MTTR).

2.3.1. MTBF

The MTBF measures the expected time between two failures, considering the failure is repaired immediately. The Bathtub curve (Figure 4.13.) represents the relation between the failure rate and time. The lifecycle of a technologic product typically has three phases: the Infant Mortality, the Useful Life and the End of Life.[16] On the Infant Mortality phase failures occur because the operator have not test the product sufficiently or it is not properly configured and connected or just because of factory defects. The Useful Life is the steady state of the robot where the first phase failures have been fixed and the maintenance needed has been attenuated. Finally, on the End of Life products begin to wear out because of its age or lack of maintenance.

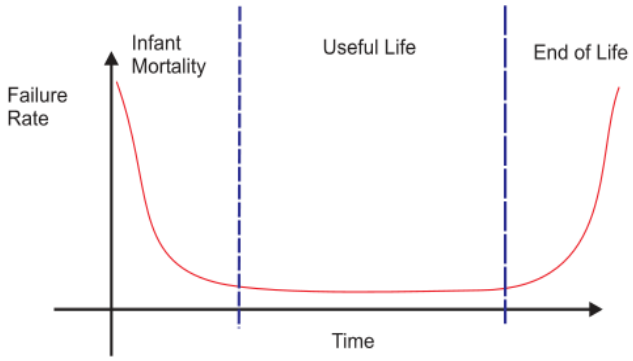


Figure 2.5. Bathtub curve [16]

Once AdvanRobot has a list of diagnostics, the MTBF can be easily measured. Taking a period of time that AdvanRobot has been working and dividing it by the times it has had a failure on that period gives the MTBF as a result.

$$MTBF = \frac{t_{final} - t_{init}}{n_{failures}}$$

2.3.2. MTTR

The MTBF does not give a useful specification by itself. It needs a complementary specification to have a useful meaning. The MTTR indicates how long it normally takes to repair the robot. It can be taken by dividing the sum of number of hours that took repairing it by the times it failed.

$$MTTR = \frac{Total\ repair\ hours}{n_{failures}}$$

Taking these two examples:

1- Robot-A works 1000 hours and fails five times. It takes 1 hour to repair each failure.

2- Robot-B works 1000 hours and fails five time. It takes 5 hours to repair each failure.

They both have a MTBF of 200 hours. However, Robot-A has a MTTR of 1 hour and the Robot-B 5 hours. If they both cost the same and they do the same job the Robot-A would be the best option.

Keonn can not only use these measures to know if they are improving the robot on forthcoming models but also to know if the components they are using are good enough. For example, if the IMU is failing often and takes lot of time to fix it, they could try another manufacturer expecting better results. So the MTBF and the MTTR are not only useful for the whole robot but also for each single hardware component.

3. QUALITY ASSESSMENTS

At this point the robot has a diagnosis method to detect any disorder during its lifetime. However, it has to be also checked the quality of its performance. This is a process done in almost every product on the market which are tested before being delivered to the shops or the client. Keonn neither can deliver an AdvanRobot to their clients without being sure its performance is acceptable.

This chapter shows the methodology that has been established to test the robot's operations and make quality assessments of the robot. This methodology could be executed by a Keonn's worker or by a distributor company's worker. In any case this is thought to be done by a qualified operator on an equipped test room with enough space and materials to carry out each of the proposed tests. On the explanations of each operation is detailed what is needed.

The quality assessment methodology established consist of different tests to 5 different robot operations:

1- Motion

2- Odometry

3- RFID system

4- Mapping

5- Inventory

Some of the tests' results are quantitative but others are qualitative so the operator has to judge himself if the robot has passed the test. Using the results of these tests can be taken conclusions and decide either if the robot is ready to be delivered or not. The robot will be considered ready to be used when each result of the tests is within a range of acceptable values.

The thresholds to decide if the robot is valid are taken depending on the needs of AdvanRobot. Some of them have been set but others have not been possible due to time constraints.

On the following pages is explained the processes established to perform correct tests for each operation in order to do a quality assessment. They are presented on a detailed step format that the operator has to follow in order to be valid.

3.1. Motion

The motion operation refers to the robot's ability to move in surfaces. It is divided in three different tests or obstacles that AdvanRobot could find on a working environment: a gap, a step and a ramp. The "Documento Básico SUA (Seguridad de Utilización y Accesibilidad)" includes the measures and characteristics a building must have in order to be safe and accessible under Spanish laws. It is taken as a reference to set some of the thresholds.

3.1.1 Gap

Test setting:

To emulate this on a test room it is needed two platforms of the same height which at least has to be 5 cm to do not let the wheels touch the floor. They have to be mobile and the top's surface has to be plain. They should not move when the robot goes from one platform to the other one, so they have to be fixable somehow (using walls and doorstops for example). They also have to have a ramp to place the robot on top.

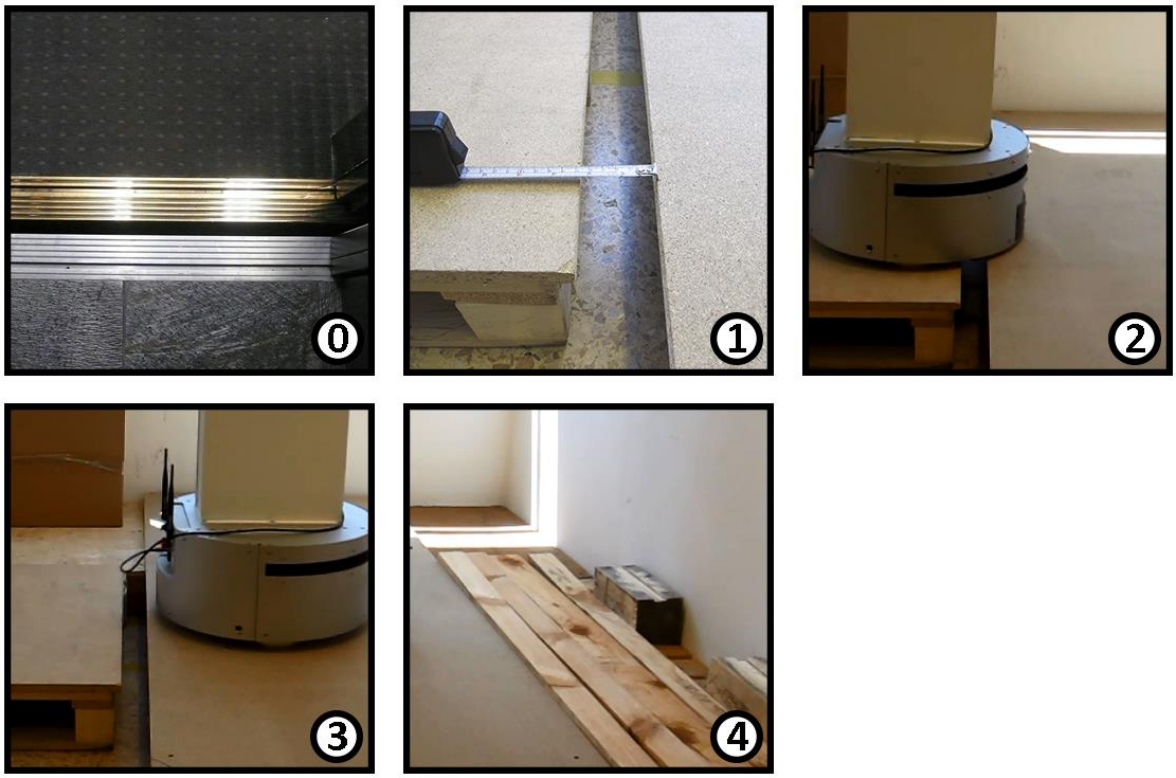


Figure 3.1. Gap test procedure

0- It is tested the capacity of going from one plain surface to another one through a gap. Gaps can be found for example in lifts' doors where the wheels can get trapped if the motors do not have enough power. The SUA dictates that the separation on the horizontal plane between the floor and the lift must be as much 2 cm so AdvanRobot must surpass this gap.

1- Move the platforms until there is a 1 cm gap and fix them to the floor. The platforms' edges that shape the gap have to be parallel to each other. Place the robot carefully on top of one of the platforms by disengaging and pushing it.

2- Once the robot is on top of one of the platforms looking to the other one, engage it and control it with the remote controller so it crosses the gap. The robot's forward direction has to be perpendicular to the gap. Keep your hands near to avoid it falling forwards or backwards when crossing the gap.

3- If the robot crosses properly without being helped, start again this process from the first step by adding 1 cm more to the gap. If it gets stuck on the gap the result of the test is the last gap distance it crossed satisfactorily.

3.1.2 Step

The distance between the floor and the bottom of the robot's base is 24 mm which is not much but enough to surpass some small steps that it can find on shops. Different type of objects considered as steps can be found on shops and AdvanRobot may not consider them as barriers so it tries to surpass them. For example, cables crossing the corridors, carpets or different floor types between shop sections.

Test setting:

It is needed four different boards of 5, 10, 15 and 20 mm of width. They have to be placed under a heavy object to fix it to the floor avoiding the robot moves it. It can also be supported to a wall.

Figure 3.2. Step test procedure

0- It is wanted AdvanRobot to surpass domestic appliance's cables like a vacuum cleaner. The biggest appilance's cables have typically a diameter of 6 mm, so the robot must surpass at least this height.

1- Place the 5 mm board on the floor and fix it.

2- Pull the robot and place it in front of the board as its forward direction is perpendicular to the front board's edge. Engage the robot and carefully make it surpass the board by using the remote control.

If the robot surpasses the board, repeat this process by using the next broader board. When the robot is not able to surpass a board means that the result of the test is the last board's width it did surpassed.

3.1.3 Ramp

Shops may have ramps that go from one floor to another one or just link different shop's levels. Because of the robot's height it has its centre of gravity quite high and it cannot be inclined a lot. But AdvanRobot can still go up through some ramps without falling.

Test setting:

An articulated ramp has to be built on the test room. The ramp must be able to adapt its inclination. It also has to have reinforcements along the ramp so the robot does not bend it. A possible solution is two boards united by hinges where one can be elevated and the other one forms a ramp to the floor. The edge that touches the floor has to be covered with a flexible non slippery mat to avoid the small step the board makes.

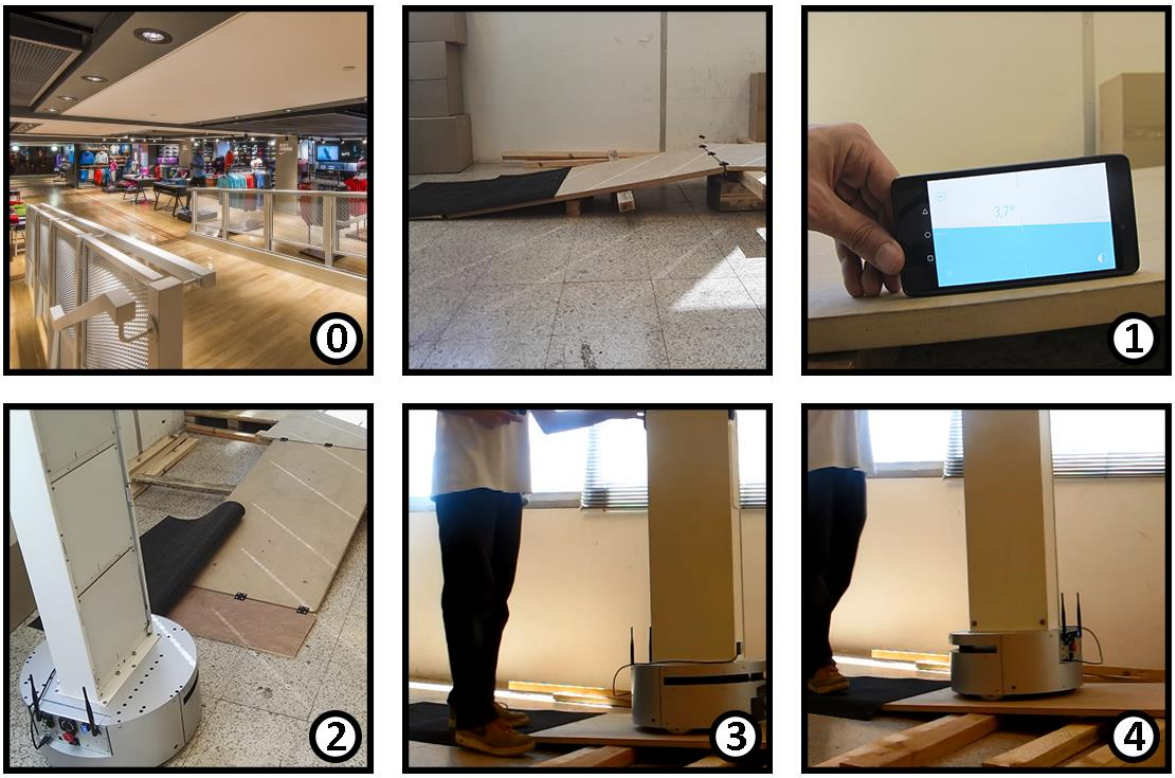


Figure 3.3. Ramp test procedure

0- The SUA says that accessible ramps must have an inclination of at most 12%, so shops may have ramps with this inclination. AdvanRobot has to be able to go upwards and downwards a 12% ramp without falling.

1- Download on a smartphone the 'Nivel de Burbuja' app for Android and calibrate it by following the steps on the menu.

2- Fix the ramp as it has an inclination of 5º (measure the inclination with the Nivel de Burbuja app. Push then the robot and place it in front of the ramp as its forward direction is parallel to the ramp's slope.

3- Engage the robot and make it move upwards the ramp using the remote controller. Situate yourself always behind the robot in case it falls.

4- Once it has travelled 1 meter on the ramp, make it turn using the remote controller and move it downwards. If it goes upwards and downwards without your help, move the ramp and fix it again as its inclination increases 1º. Repeat the process with this inclination. If the robot cannot go upwards or downwards, the result of the test is the last inclination it passed.

3.2. Odometry

The odometry is the estimation a robot does of its position thanks to the encoders of the wheels with traction power. It is relative to an initial point and includes its position and its rotation. As long as sensors have errors on their accuracy, the position computed by the robot is an estimation as it is probably not the actual one. [13]

Is important that each AdvanRobot has an acceptable odometry as it could get lost on the shop. On this test is looked the difference between where the robot thinks it is and where it actually is. This difference is the odometry error which can be divided in two different types of errors: systematic errors and non-systematic errors.

A systematic error is caused by imperfections of the robot's design, as it can be unequal wheel diameters and misalignment of wheels. On the other hand, a non-systematic error is caused by external influences like a slippery floor. The objective of this test is measuring the systematic errors.

The University of Michigan proposes a method for the quantitative measurement of systematic odometry errors. It is called the University of Michigan Benchmark test (UMBmark) and is followed on this section.[14]

Test setting:

The test room needs a large space where the robot can move around a 4x4 meters square. More than this area is needed given that it could go out of the square. Near one of the corners it is marked a cross with tape that gives a coordinate axis reference. Each tip of the cross measures 25 cm from the centre which is the radius of the robot's base.

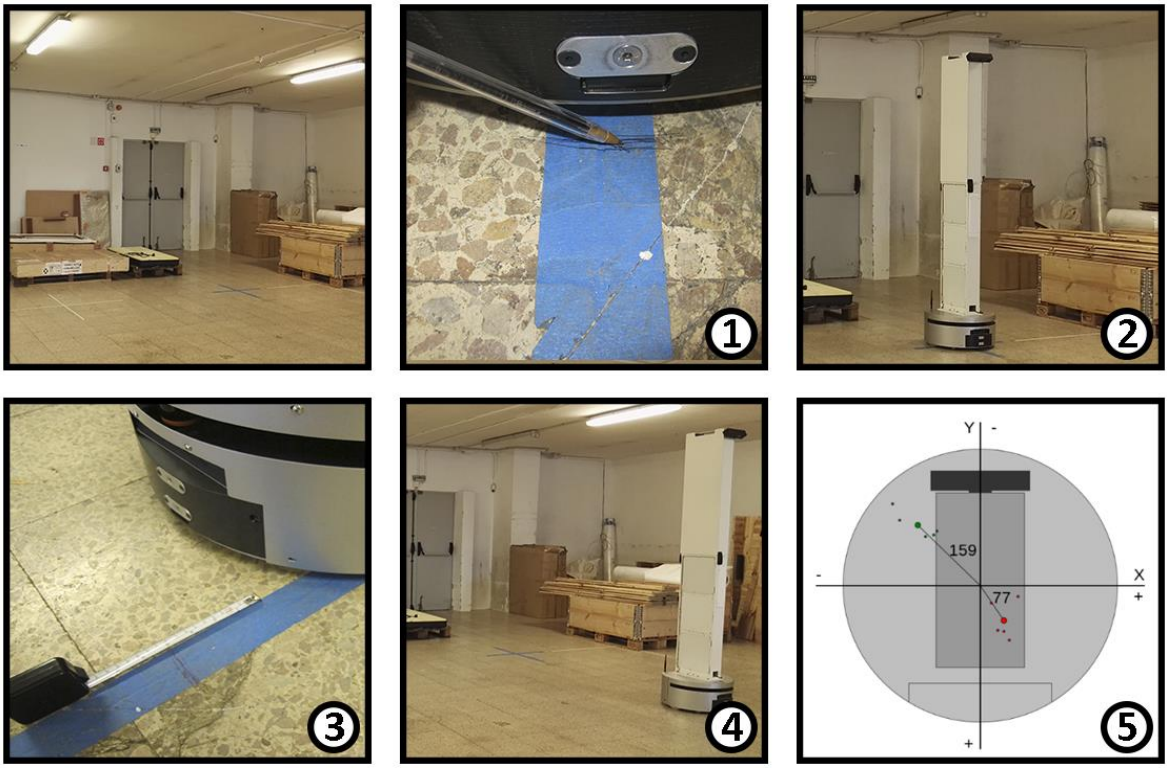


Figure 3.4. Odometry test procedure

1- Place the robot on the middle of the cross so the edges of the robot coincide with the tips of the cross. Place it looking forward to the positive Y (take the step X image as an axis reference).

2- Engage the robot and restart the odometry of the robot. Run the clockwise's UMBmark script that makes the robot move around the 4x4 meters. It moves forward at 10 cm per second and turns at 9º per second. This means it takes it 3 minutes and 20 seconds to complete the square. Keep an eye on the robot in case it deviates too much from its route.

3- Once the robot has completed the route, measure the distance from the centre of the robot's base to each axis' tips where the robot started. This is the absolute measure. Write down the point where it has ended alongside the endpoint computed by the robot, given as the result of the script.

4- Repeat the steps 2 and 3 five times for clockwise (CW) direction and five more for the robot travelling in counterclockwise (CCW) direction. Write down each measured and computed points.

5- Follow the following process to take the systematic odometry error.

By subtracting the computed point from the absolute point it ends up with the odometry error of each individual repetition:

$$\varepsilon x = x_{abs} - x_{comp}$$

$$\varepsilon y = y_{abs} - y_{comp}$$

Where εx and εy are the odometry error's coordinates. x_{abs} and y_{abs} are the absolute measure's coordinates. And x_{comp} and y_{comp} are the computed measure's coordinates.

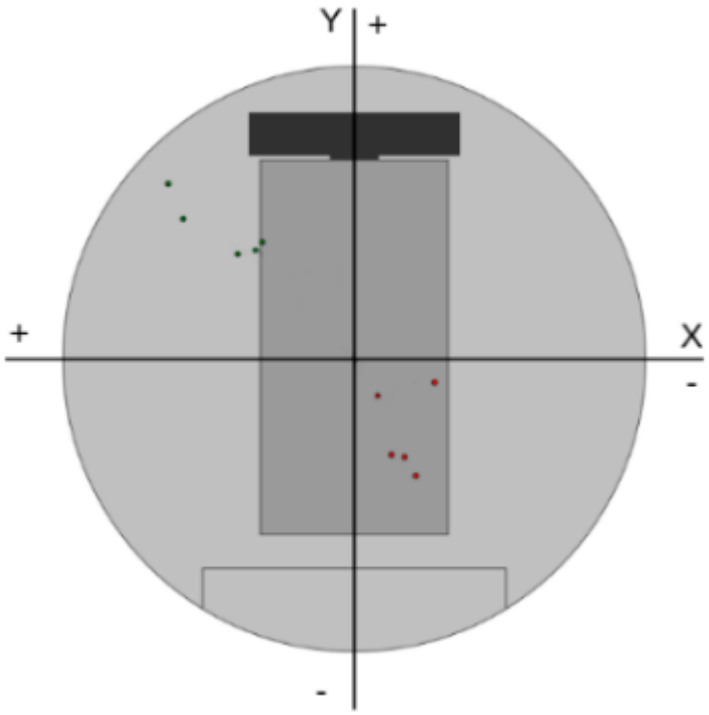


Figure 3.5. Odometry error clusters

The result is two clusters of points with five points each representing the odometry errors of both directions (CW/CCW). The Figure 3.5. shows an example of the odometry errors' clusters for the CW direction in red and the CCW direction in green. The dispersion of the points within a cluster is the non-systematic error. However, on the UMBMark paper is shown that on an uncalibrated robot traveling over a reasonably smooth floor, the contribution of systematic errors is notably larger than the contribution of non-systematic errors.

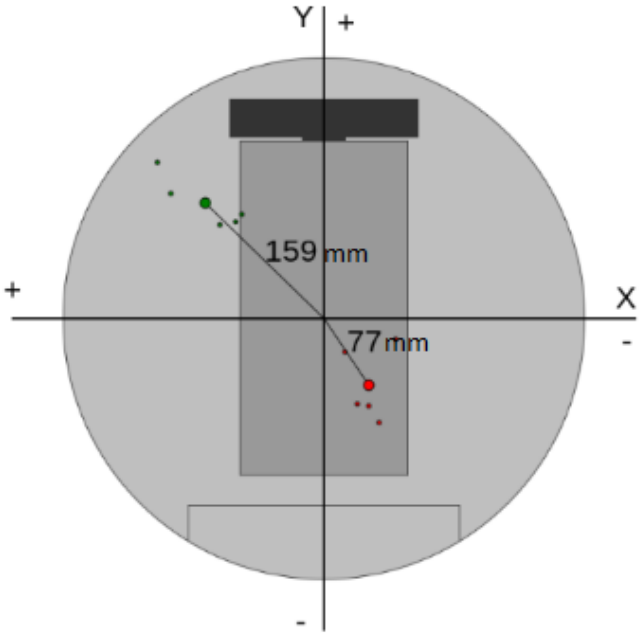


Figure 3.6. Odometry error offset

On the other hand, the systematic error is the offset of the cluster's gravity point to the origin of the coordinates system. The cluster's gravity point is the average of the points within it.

$$x_{gp,cw/ccw} = \frac{\sum_{i=0}^{n} \varepsilon x_{i,cw/ccw}}{n}$$

$$y_{gp,cw/ccw} = \frac{\sum_{i=0}^{n} \varepsilon y_{i,cw/ccw}}{n}$$

Where:

$x_{gp,cw/ccw}$ and $y_{gp,cw/ccw}$ are the gravity point's coordinates for CW or CCW directions.

n is the number of points within a cluster.

The offset of the gravity point to the origin of the coordinates is then:

$$r_{gp,cw/ccw} = \sqrt{\left(x_{gp,cw/ccw}\right)^2 - \left(y_{gp,cw/ccw}\right)^2}$$

As shown on the Figure 3.6. the CW cluster's offset is 77 mm and the CCW one is 159 mm

The systematic odometry error is finally considered as the maximum value between the clockwise offset and the counterclockwise as it has to be considered the worst scenario.

$$\varepsilon_{odometry} = Max(r_{gp,cw}, r_{gp,ccw})$$

On the example of the Figures 3.5 and 3.6 the systematic odometry error is 159 mm.

It has not been possible to set a threshold due to time constraints because it would need many tests to take a relevant value.

3.3. Antennas

The main operation of the robot and the reason of its existence is to read tags. What is tested in this section is not that the antennas read tags as it is checked on the diagnostics section, but the quality of the reads.

Going back to the Figure 2.4. there is a value on the XML that has been used to measure the quality of the reads. The Received Signal Strength Indicator (RSSI) expressed in dBm (taking 1 mW as a reference value) indicates the power of the received signal. The lower the RSSI is, the lower the received signal is.

Moreover, as it has been said AdvanRobot is an RFID passive system which means the power emitted by the tags is a reflection of the power emitted by the antennas. So the RSSI is directly related to the power emitted by the antenna.

This test includes also a script similar to the diagnostics one, but instead of reading one time it does 10 iterations of reads with 0.1 seconds between iterations. Once it has finished reading, it compares each read's RSSI and phase with the threshold.

To take the acceptable values it has been computed the RSSI mean value for each antenna:

$$\overline{RSSI} = \frac{\sum_{i=0}^{n} RSSI_i}{n}$$

But the mean does not give enough information because the values could be for example very dispersed which means some reads done on a test would be good but other ones bad. It is important to have a certain mean value with not dispersed values. To measure the dispersion, it has been also computed the standard deviation:

$$\sigma_{RSSI} = \sqrt{\frac{\sum_{i=0}^{n} (\overline{RSSI} - RSSI_i)^2}{n}}$$

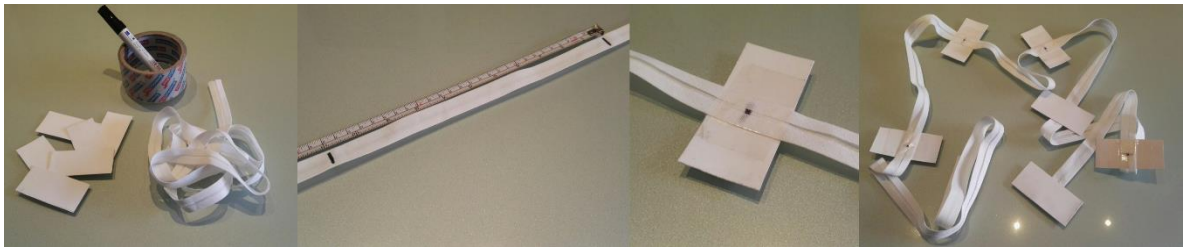
The threshold is taken from these values which is a confidence interval:

$$[\overline{RSSI} - 2 \cdot \sigma_{RSSI}\ ,\ \overline{RSSI} - 2 \cdot \sigma_{RSSI}]$$

These intervals give 95% of confidence that a given value will occur inside it. Each RSSI and phase values of each read have to be within these intervals in order to consider the test as passed. This threshold has not been taken because of time constraints.

Test setting:

The test room must be equipped with a stripe of six tags separated 28 cm between them. It can be either a fixed structure or a cloth strap. This can be easily built by attaching each tag into a cloth strap.



The configuration file on the robot must contain their EPCs correctly indicated alongside the reader and the port number of their assigned antennas.

Figure 3.7. Antennas test procedure

1- Place the strip at 30 cm from the robot's right side. Run the script corresponding to that side keeping the strip placed beside the robot until it finishes computing the result values which are shown on the console.

2- Repeat the same process for the left side.

3.4. Mapping

When the robot arrives into a shop it has to recognise it by making a map. This has to be tested before in the test room to check if it correctly draws a map that corresponds to the real scenario.

This is tested by carrying out the process that would be done on a shop.

Test setting:

The test room needs an emulated scenario made for example with boxes which simulate the corridors between shelves on a shop. It also needs at least two Quick Response (QR) codes placed on the walls at 1.5 meter height. These codes are the initial and end point.

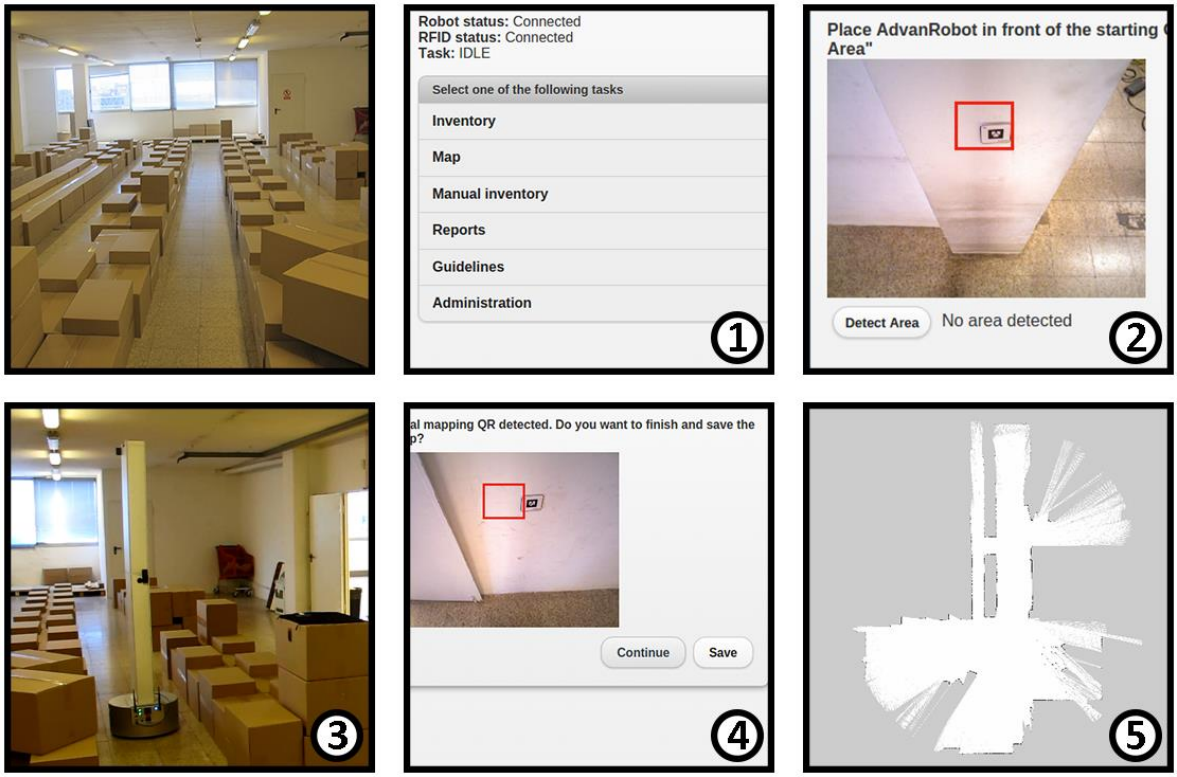


Figure 3.8. Mapping test procedure

1- Using a smartphone, choose the map option on the AdvanRobot's app's menu.

2- Place the robot in front of the QR code until the app reads the code and recognises which area is going to map.

3- Engage the robot and control it with the remote controller around the emulated shop ending on the next QR code. The robot marks waypoints on its path.

4- Once the robot is placed in front of the second code (the end point), it recognises it. Save the map.

5- The map is shown on the screen. To give a map as a good representation of the actual room it is necessary that its lines follow the same geometry rules as the real plan. If the walls are parallel for example the map should have parallel lines. It also should be each part of the map at the same scale. Evaluate it and take a decision according to these rules.

3.5. Inventory

The last operation tested is the most high-level one as it tests the entire robot's operation which is performing inventories. It is tested using the map generated on the last section. This is an end-to-end test so it depends on the other operations.

Test setting:

Keeping the scenario on the mapping process, 1000 tags are added around its way.

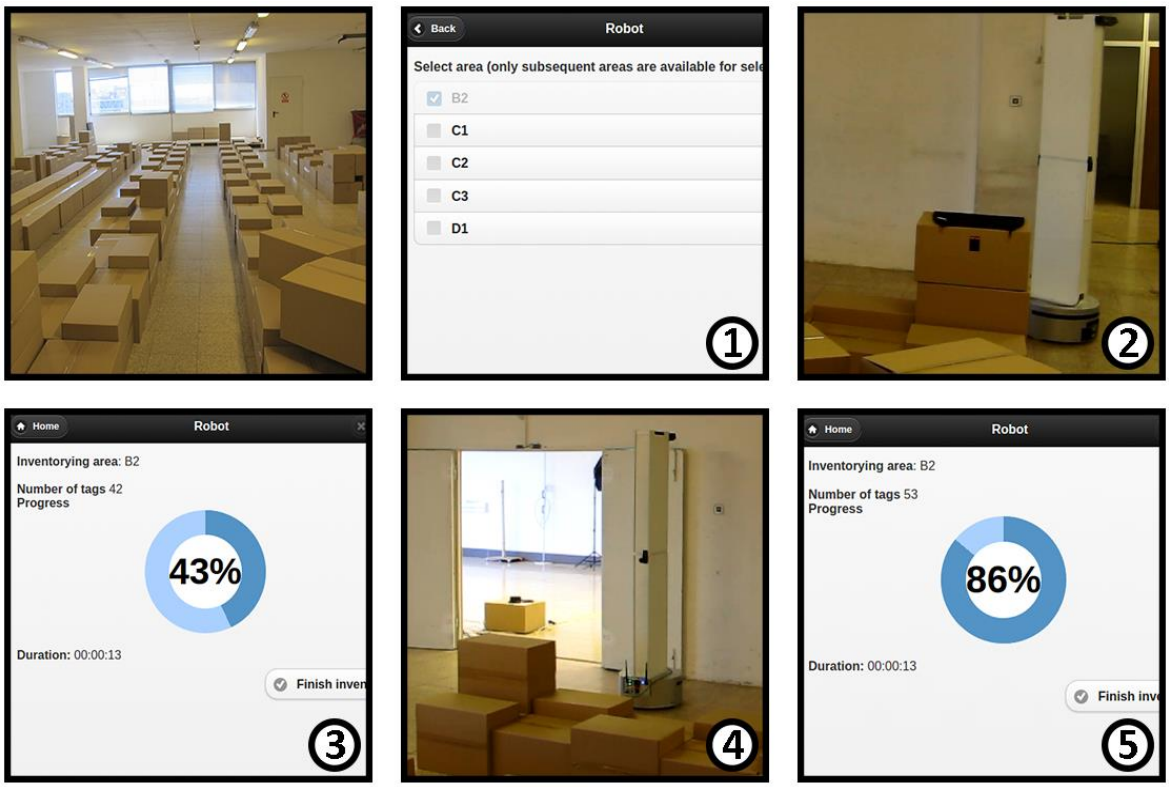


Figure 3.9. Inventory test procedure

1- Place the robot at the start QR code with the security button pulled. Choose on the app to perform an inventory. The robot recognises that the code corresponds to the starting point.

2- The robot turns 360º for a nearby recognition and straight after it starts moving through the path that was done on the mapping test.

3- The app displays the percentage of waypoints it has passed through.

4- When the robot finishes its path, it stops in front of the end point.

5- The app shows that it has completed the path and displays the number of tags it has read on the inventory process. Compare the tags that has read with the ones that were on the scenario. To pass the test, AdvanRobot has to move through the same path as the mapping process and finish exactly on the end QR. It must also read the 100% of the tags that were placed.

4. ADVANROBOT MAINTENANCE TOOL

Even though the scripts and procedures described previously help the operator, running the scripts, visualizing and interpreting the results through the command-line is not really user-friendly method. Neither is writing down in papers the results of the diagnostics and quality assessments.

For this reason, a Web application has been developed. It has been called AdvanRobot Maintenance Tool and it consists of a Diagnoses Tool, a Quality Assessments Tool, a Repair Tool and an Administration page.

The way is developed AdvanRobot right now the operator can access to the application through an URL on his Web browser when being on the same network as the robot. Once there, the application does most of the job and the operator just has to interact with it.

Figure 4.1. AdvanRobot Mainetance Tool main page

4.1. Diagnoses Tool

The Diagnoses Tool interface consists of five tabs, one for each device that can be tested and one that shows a summary of all them. Because the response from the devices can be delayed in some cases, a loading window appears until the response arrives so the operator knows it is processing the checkup.

4.1.1. Laser, camera and IMU

The tabs of the laser, the camera and the IMU are practically identical. They show a table with a step on each row and a button to test them individually. It has been programmed to test the steps in order, so each step can only be tested if the previous one has been tested and the result has been successful. This forces to do a diagnostic from low-level to high-level as it has been thought for.

Figure 4.3. Laser tab

4.1.2. Readers

As shown on the Figure 4.4. the Readers' tab allows to test each reader separately.

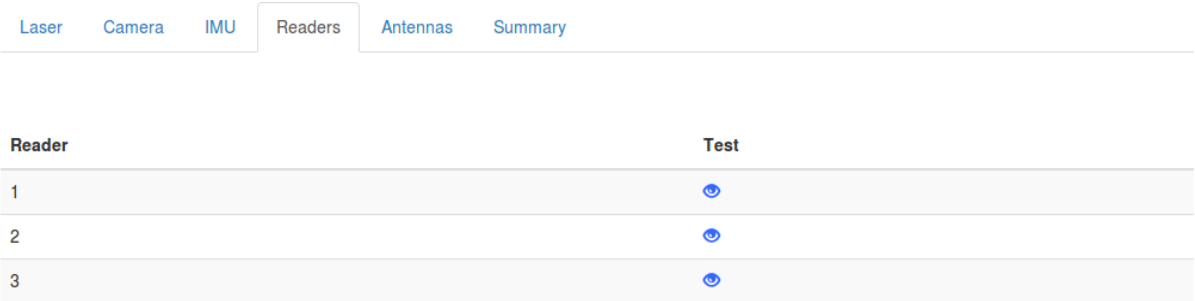


Figure 4.4. Readers tab

4.1.3. Antennas

The antennas' tab consists of two tables with six rows, one table for each side of the robot and one row for each antenna of that side. There is also a brief explanation of how to follow the procedure explained before.

A button under each table initiates the checkup and the result is displayed on the tables. It also fills the table with the EPC and port written in the configuration file.

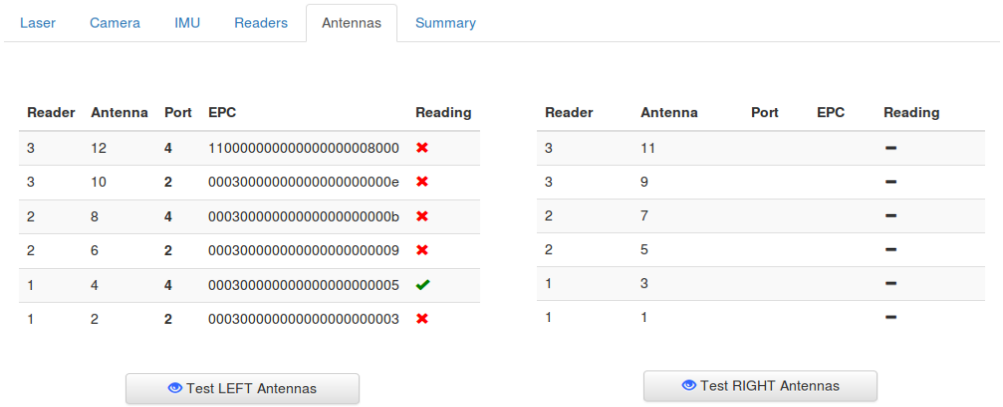


Figure 4.5. Antennas tab

4.1.4. Summary

Finally, the last tab shows a summary of every diagnosis result. At the bottom there is also a text field where the operator can write arisen observations. Lastly a button allows to save the diagnostic on the database.

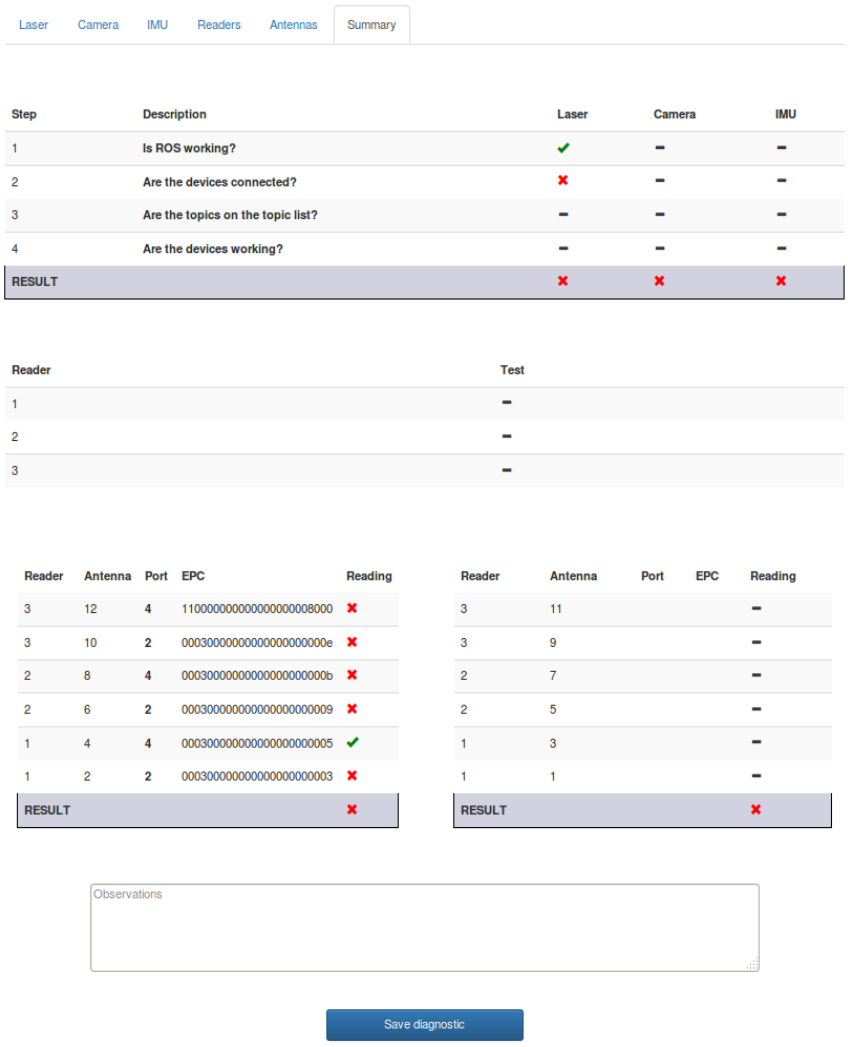


Figure 4.6. Summary tab

4.2. Quality Assessments Tool

The Quality Assessments Tool is made up of one tab for each operation described on the quality assessments chapter. This tool provides the description of the processes to test each operation in order to be followed by the operator. In some of the operation it also computes processes.

4.2.1 Motion

After each explanation there is an input field to write the gap, step and ramp values taken from the test. A save button finally saves into the summary tab.

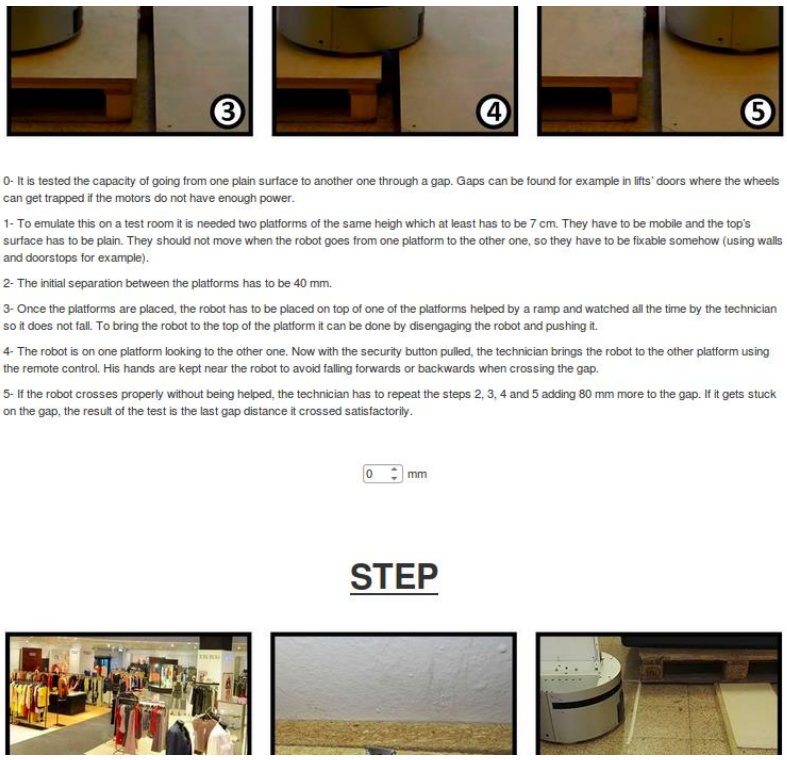


Figure 4.7. Motion tab sample

4.2.2. Odometry

In the odometry tab, after the UMBmark description there is a table with one row for each repetition the operator has to perform. After the operator measures the coordinates of an end point, he has to write them on the corresponding columns of the table alongside the ones computed by the robot.

The process of running the UMBmark scripts is not implemented on the tool because when the scripts are running the camera and laser do not work. For security reasons the operator has to run them himself through the command line.

Event	Direction	Measured X	Measured Y	Computed X	Computed Y
1	Clockwise	-550 mm	-480 mm	-380 mm	-460 mm
2	Clockwise	-560 mm	-500 mm	-430 mm	-500 mm
3	Clockwise	-576 mm	-478 mm	-430 mm	-540 mm
4	Clockwise	-554 mm	-448 mm	-410 mm	-500 mm
5	Clockwise	-566 mm	-501 mm	-450 mm	-550 mm
6	Counterclockwise	-250 mm	250 mm	220 mm	-230 mm
7	Counterclockwise	-225 mm	230 mm	180 mm	-210 mm
8	Counterclockwise	-219 mm	218 mm	590 mm	-500 mm
9	Counterclockwise	-239 mm	267 mm	440 mm	-350 mm
10	Counterclockwise	-250 mm	301 mm	380 mm	-310 mm

Save odometry

Figure 4.8. Odometry tab

4.2.3. RFID system

This tab has the same characteristics as the one in the Diagnostics Tool. The operator is asked to place the tag's strip on its place and press the button that makes the script read the tags. The screen remains disabled until it has done all the iterations. When it finishes it shows if each antenna has passed the test and also the times the antenna has read the tag alongside its RSSI and phase mean.

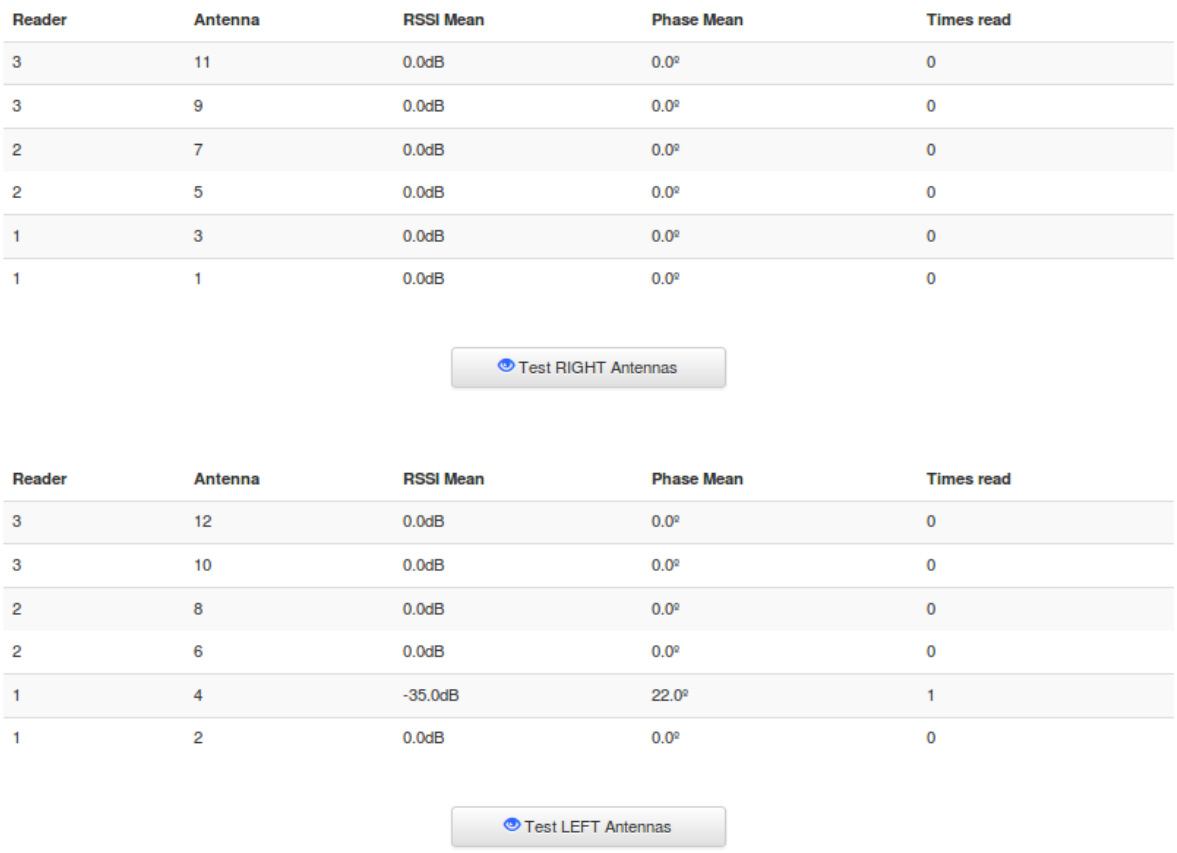


Figure 4.9. Antennas tab

4.2.4. Mapping and inventory

As the results of the mapping and inventory tests are qualitative, nothing is computed by the tool. A part of the explanation of the process it is asked to indicate either if it has passed the test or not.

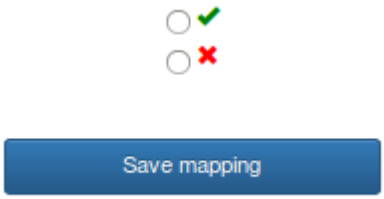


Figure 4.10. Mapping tab

4.2.5. Summary

As the equivalent tab in the Diagnostics Tool, the summary tab contains every result of the quality assessment. Additionally, for the odometry part it draws both clusters on an HTML canvas alongside the gravity points and the offsets like the Figure 3.6.

Finally, a button saves the quality assessment into the database.

4.3. AdvanRobot Maintenance Administrator

Django provides an administration page where can be administered the authentication and the database. It has been adapted to cover the necessities of the Maintenance Tool.

To access into the administration main page is done through the list icon that appear around the application next to the titles. A user and a password is needed to enter on this page and the root user can create other users assigning different edit and access roles.

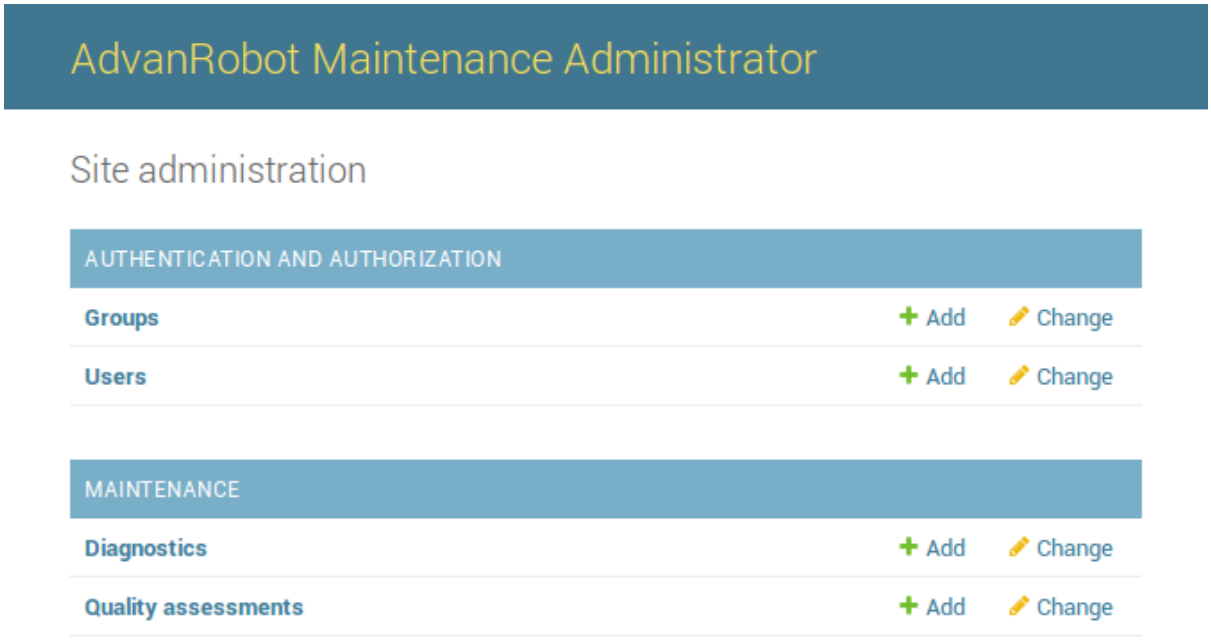


Figure 4.11. AdvanRobot Maintenance Administrator's main page

Below the Maintenance title are found the accesses to both of the database's tables.

4.3.1. Diagnostics administration

This part of the interface is really useful since it shows every diagnostic that has been saved ordered by the date that was performed. The Figure 4.12 shows an example of how is marked every component depending on the result of the diagnostic.

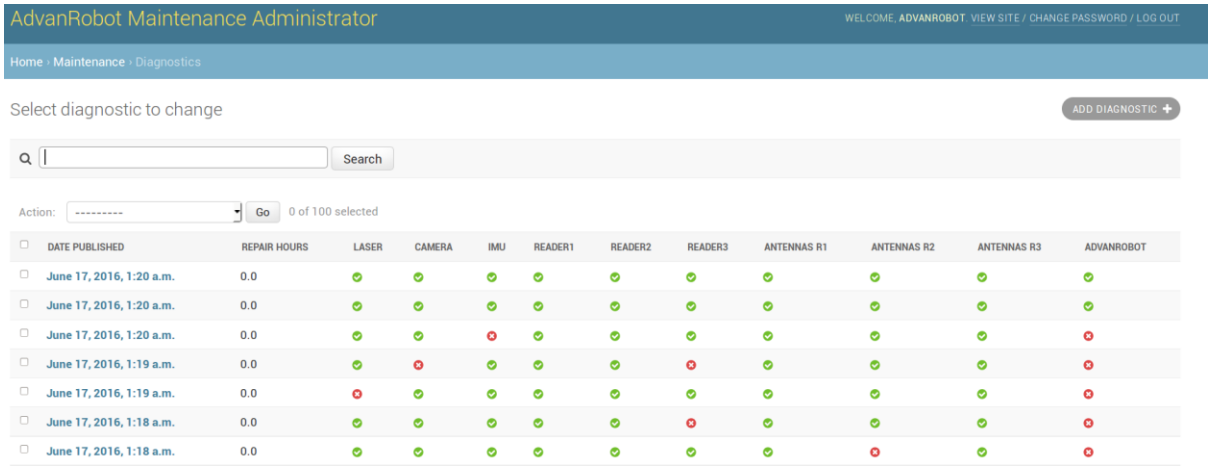


Figure 4.12. Diagnostics database

By clicking one of the diagnostics it redirects to a detailed diagnostic page where it can be edited three fields a part of the component's results.

One of these fields is the observations one that was filled by the operator before saving the diagnostic. The other ones are Repair Method and Repair Time. A good practice to use this fields would be the following:

Once the checkup has been saved, if any of the components have failed, the operator can start repairing the robot. As this process still belongs to the diagnosis he just saved, he should not use the Diagnosis Tool again but the Repair Tool. The Repair Tool is practically identical to the Diagnosis one but the components can be tested freely as the steps are always enabled. On the other hand, it cannot be saved on the database.

When AdvanRobot is totally repaired, the operator should go back to the diagnostics list and edit on that specific diagnosis the number of hours he has spent repairing it in the Repair Hours field which accepts decimal numbers. He also has to fill the field Repair Method explaining the steps he has followed to repair it. On the field Observations he can add a more detailed description about what was failing on the robot.

This fields are important to be filled as they are used to take the reliability of AdvanRobot as it has been explained before. It is also really useful when repairing it: If the operator does a diagnosis and one of the components fails, he can look on the database and search the stored

diagnostics that also failed because of that component. Looking on their observations field he can look for one that had a similar conduct and try to repair the robot by following the steps described on the Repair Method field. This significantly shortens the time the operator spends repairing the robot.

4.3.2. Quality assessments administration

Similarly to the diagnostics one, this page shows the list of the quality assessments. It helps to store a record of the robot's capabilities on its memory. When an anomaly occurs and is sent back from the shop to be repaired the operator could evaluate again its capabilities taking its wear by comparing them to its stored ones.

It also guarantees that the robot was delivered passing each of the tests.

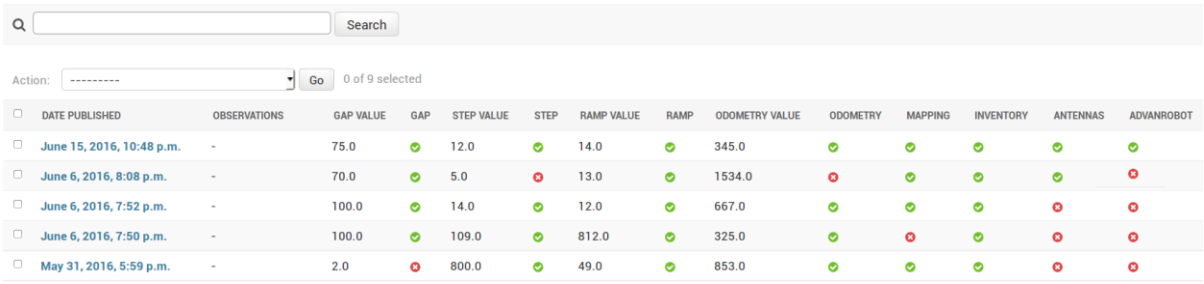


Figure 4.13. Quality assessments database

5. CONCLUSIONS

Fulfilling the objectives established, the result of this dissertation is a useful application that allows an operator to do diagnosis and quality assessments to an AdvanRobot. A tool that could be used by Keonn, an actual company, adding a value to their product.

Working with a really complex system like AdvanRobot is easy to understand how routine procedures that could look simple are not that easy to perform. Leaving this tool at the hands of the AdvanRobot's operators ensures that the effort put on this dissertation will have a fruitfulness and continuity on the future.

However, there is future work to be addressed:

-Some guidelines have been given about how the test room has to be. But these guidelines could be more defined on the future. Having the exactly same test circuit on the different test rooms would give a more valid results and the tool could be more automated and parametrized.

-The diagnoses tool could give more information about what is failing. At least it could print the exceptions that would be shown on the console.

-As long as the application is aimed to be used by a reduced number of users it has not taken much into account its design which could be improved.

-The application could also be securitized with an authentication to prevent customers entering on it and saving fake diagnoses or quality assessments.

-There have been implemented the tools to diagnose some important components but as AdvanRobot is made of many more they could also be implemented.

-The quality assessments could also be extended evaluating other operations of the robot.

-Finally, as it has been mentioned before it can be adapted to the actual application that AdvanRobot actually has.

References

[1] Ford. (2008). The evolution of mass production. Consulted on the 17th of May 2016 at http://www.ford.co.uk/experience-ford/Heritage/EvolutionOfMassProduction

[2] Anita Bunk. (2014). Infographic: Capitalizing on the Internet of Things. Consulted on the 17th of May 2016 at http://blog.bosch-si.com/categories/internetofthings/2014/05/infographic-capitalizing-on-the-internet-of-things/

[3] Bar Code Graphics, INC. (2012). What is RFID? Consulted on the 29th of April 2016 at http://www.epc-rfid.info/rfid

[4] Impinj. (2013). The different types of RFID systems. Consulted on the 29th of April of 2016 at http://www.impinj.com/resources/about-rfid/the-different-types-of-rfid-systems/

[5] Inteco. (2010) Guia sobre seguridad y privacidad sobre la tecnología RFID. COnsulted on the 29th of April of 2016 at https://www.agpd.es/portalwebAGPD/canaldocumentacion/publicaciones/common/Guias/Guia_RFID.pdf

[6] Klemens Sattlegger and Uli Denk. (2014). Navigating your way throgh the RFID jungle.Consulted on the 3rd of May 2016 at http://www.ti.com/lit/wp/slyy056/slyy056.pdf

[7] Ubuntu. (2015). Ubuntu. Consulted on the 13th of March 2016 at http://www.ubuntu.com/

[8] ROS. (2013). About ROS. Consulted on the 15th of March 2016 at http://www.ros.org/about-ros/

[9] Walter Hamscher. (1991). Principles of diagnosis: current trends and a report on the first International Workshop.

[10] ROS. (2013). Rosout. Consulted on the 2nd of June 2016 at http://wiki.ros.org/rosout

[11] The Linux Documentation Project. (2002). The /dev directory. Consulted on the 23rd of April 2016 at http://www.tldp.org/LDP/sag/html/dev-fs.html

[12] ROS. (2016) Module client. Consulted on the 25th of March 2016 at http://docs.ros.org/api/rospy/html/rospy.client-module.html#wait_for_message

[13] Cuentos Cuanticos. (2011) Robótica: Estimación de posición por odometría. Consulted on the 3rd of June 2016 at https://cuentos-cuanticos.com/2011/12/15/robotica-estimacion-de-posicion-por-odometria/

[14] Johann Borenstein and Liqiang Feng. (1995). UMBmark: A Benchmark Test for Measuring Odometry Errors in Mobile Robots. The University of Michigan.

[15] Django. (2014). Meet Django. Consulted on the 29th of April 2016 at https://www.djangoproject.com/

[16] IMC Networks. (2011) MTBF, MTTR, MTTF & FIT. Explanation of Terms. Consulted on the 5th of June 2016 at http://imcnetworks.com/wp-content/uploads/2014/12/MTBF-MTTR-MTTF-FIT.pdf