Wi-Fi Controlled Car

Test report

This document is the test report of the group AAD1 summer 2013 about the application Wi-Fi Controlled Car.

Document title	Test Report
Document ID	WCC-TestReport
Authors	Ambroise Dhenain Szymon Klepacz Anatolii Shakhov Alvaro Garcia Pierre Le Texier
Supervisor	Torben Gregersen
N°. pages	19
Date	07-02-2013

Document history

Date	Audit	Author	Description
06-12-2013	0.1	AS	Template + Information about successful tests.
06-12-2013	1.0	AD	Added software tests part.
08-12-2013	1.1	SzK	Changed test information. Corrected spelling.
09-12-2013	1.2	AS	Updated User Acceptance part, also improved the first part with initial components tests.
10-12-2013	1.3	AS	Updated Information about tests (according to supervisors remarks), removed conclusion.
12-12-2013	1.4	AD	Added information about User acceptance test and Software parts.
12-12-2013	1.5	SzK	?
12-12-2013	1.6	AS	Minor Changes.
12-12-2013	1.7	AD	Improved user acceptance tests. Added pictures.
12-12-2013	2.0	AS	Final version. Spelling checked and corrected in the last part of the report.

Approval

Author(s)	Ambroise Dhenain – 201301255 [AD] Szymon Klepacz – 201301204 [SzK] Anatolii Shakhov – 201301534 [AS] Alvaro Garcia – 201301220 [AG] Pierre Le Texier – 201301944 [PLT]
Project name	Wi-Fi controlled car
Document ID	WCC-TestReport
No. pages	19

By signing this document both parties accept, that this is the requirements for the development of the desired system.

Place and date: 13-12-2013

Authors Supervisor

Contents

Tables of figures

1. Opening

1.1. Purpose

The purpose of this document is to explain a number of tests within the system environment.

We will show how the system was tested, which bugs and errors occurred during the process and how the system works in a real environment.

1.2. Reading instruction

Chapter 2: Explanation about our way to test.

Chapter 3: Unit tests, one per functionality.

Chapter 4: Integration tests, using several functionality already tested by Unit tests to check a bigger environment.

Chapter 5: Non-regression tests, about how we managed the functionalities evolutions.

Chapter 6: Final test, one per version of the application.

2. Test system

During the project we had to work with different hardware and software. The main idea was to assemble a car which can be remotely controlled by a smartphone. We needed to start initial tests before considering on the components. We didn't know exactly the current and voltage level from the power supply and required by both engines. Each hardware component required special knowledge about its detailed schematics, special tools to provide a proper installation of the components. For the first tests we tried to understand how the different components work separately.

With the software tests, the situation was a bit different. To achieve the proper tests we had to use software together with the hardware. We could notice bugs and errors in the code from the IDE and we could test how the application works on the phone. The only concern was to test an application with the assembled hardware (the process of assembling took longer time due to some failed tests).

The last steps in testing the system would be create conditions to hold tests in communication between software and hardware in field Tests.

Each group was responsible for the tests on his part, respectively Car/Arduino and Android application. However, some tests need to use both, in this case the tests were done by the responsible of the Android application.

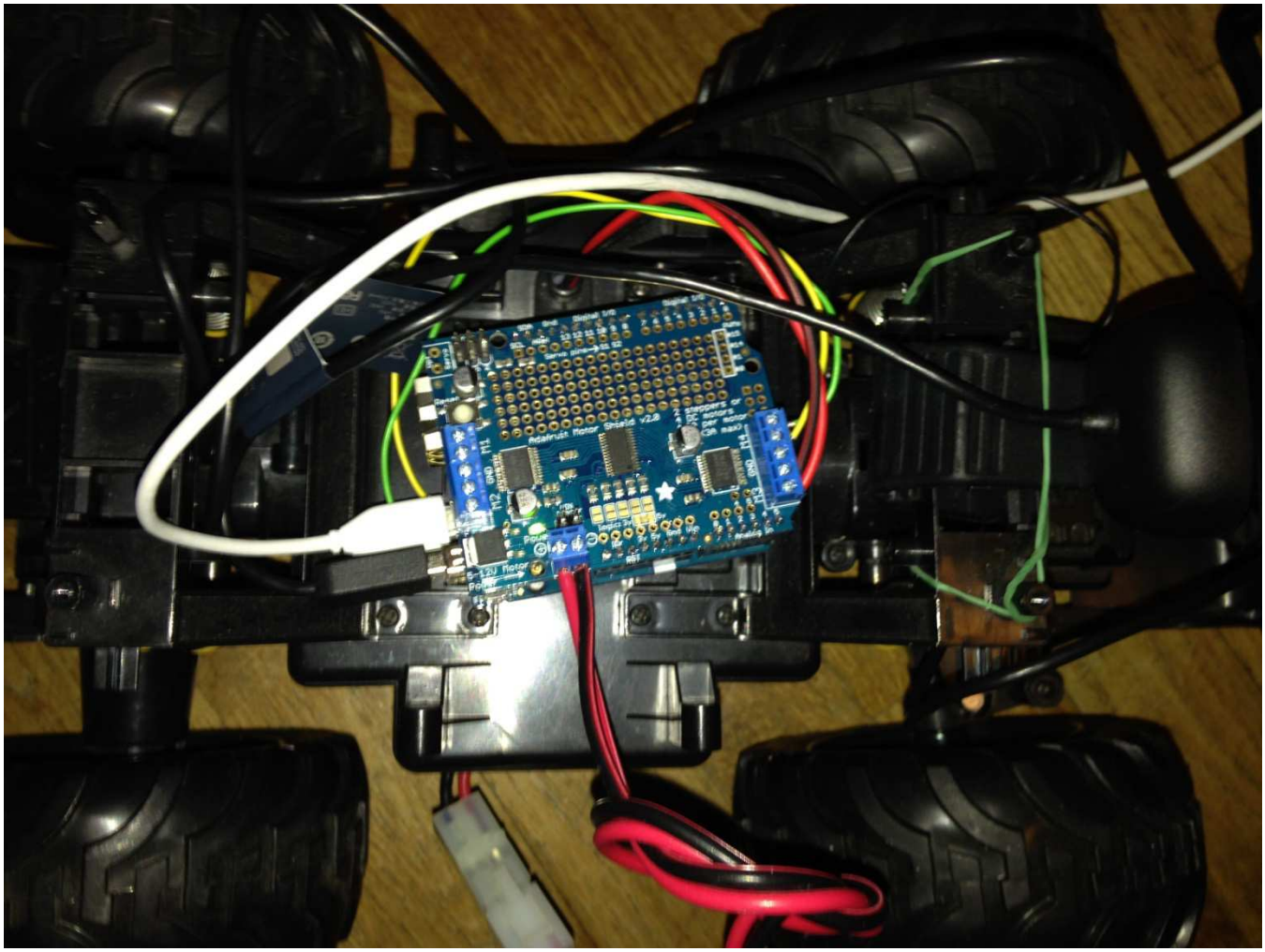

Figure 1 - Car photo during 2.0 tests (camera)

3. Unit tests

3.1. Hardware

To test the main board for the car we needed special software downloaded from the Arduino web site. With this software we could easily load any program on the board. One of the biggest benefits in it is the availability of test programs (they're called sketches) which can be loaded on the board to test different functions.

During those initial tests we tried to understand how the board behaves and which instructions change something during the operation process.

Functionality	Date	Person	Result
Initialization	07-10-2013	SK/AS	**YES**
Basic test operations	07-10-2013	SK/AS	**YES**
Wireless Connection	07-10-2013	SK/AS	**YES**

The initial tests passed successfully. We connected the Arduino board to PC, downloaded all the needed software for further development and started test sketches on the board. It ran all the programs successfully. When we turned on the board – we had a possibility to connect to wireless network created by Arduino. When connected we were able to log into the user interface and set up different settings.

The board was working properly out of the box. We didn't have to add anything else to it – everything needed was to connect the board to the laptop using a USB-MicroUSB wire.

As with the board we had to test the functionality of the engines of the car. These operations required some additional tools from us: digital Multimeter, battery and our car with engines.

Functionality	Date	Person	Result
Movement/Rotation	26-09-2013	Group	**YES**
Voltage	26-09-2013	Group	**YES**
Battery	26-09-2013	Group	**YES**

Those tests passed successfully as well, the battery worked perfectly and it could supply our board. The only problem during tests was the failure of multimeter (it didn't show the précised values) so we had to change it for a working one model.

After the initial tests we understood the working principle of different parts of our future project. The engines needed a power supply of at least 5V, Arduino board used a mixture between C and C++ languages and we had to load sketches on it to implement different functions, the battery provided 9.7V.

After we disassembled the car, we could not find any information about the engine, but we had a successful voltage and operation tests. If we want the car to move forward, we should connect black and red wires (wires from 1st engine) in a straight way. If we want the car to move backward, we had to switch them. This test showed us the need in motor shield which can provide the functionality to send different voltage.

The initial tests in hardware part succeeded and we could move further for integration testing (deeper and thorough tests to provide information about components integration and inter operation.

3.2. Software

The software tests part contains both Android and Arduino tests, because we can test the Android part separately from the Arduino part, but we have to test both at the same time to test the Arduino part.

All unit tests about Android and Arduino was validated once the functionality was completed, not before.

3.2.1. Android tests

Functionality	Date	Person	Result
Move	16-11-13	AD/SzK	**Yes**
Initialize car	04-12-13	AD/SzK	**Yes**
Change settings	16-11-13	AD/AG	**Yes**
Store picture	09-12-13	AG/AD	**Yes**
Video stream	11-12-13	AG/AD	**Yes**
Honk	17-11-13	AD/SzK	**Yes**

The Android tests could be done without use the Arduino or the car, we needed only the IDE to check the values or display the results. Basically it was to check if the strings sent by socket was correct or not, sometimes it was also to check if the user interface was correctly updated.

3.2.2. Arduino tests

Functionality	Date	Person	Result
Move	28-11-13	AD/SzK	**Yes**
Initialize car	06-12-13	AD	**Yes**
Change settings	06-12-13	AD	**Yes**
Take picture	01-12-13	PLT	**Yes**
Video stream	01-12-13	PLT	**Yes**
Honk	28-11-13	AD/SzK	**Yes**

The Arduino tests was the most difficult to do, because of the fact we destroyed two Arduinos, we couldn't test our functionalities without a real Arduino. We lost almost three weeks to wait to validate Arduino tests. It was difficult because of the Serial connection, for instance we couldn't display Integers in the console without some conversion, took some time to find out how to do it.

3.3. JUnit

We didn't use JUnit in our IDE (Android) to run the tests, we didn't use any framework to do it, our application doesn't need a test framework, and all tests were done manually by the developer in charge.

4. Integration tests

During these tests we tried to test an interaction between different components. Are they suitable for each other? Can they operate together to provide a proper result? Is the system solid and works as intended?

The new series of tests will show the working of the components and approval of their proper connection.

4.1. Hardware

During the initial tests we tested components, how they operate and what principles are important for them.

At the beginning we soldered headers to our new Adafruit motor shield. Then we plugged it into an Arduino board. The system worked fine, the only one concern was a jumper to provide the voltage for the Arduino board from the shield. When we connect the shield to Arduino – we need to add a power supply to the shield and if we want that supply to give energy to Arduino also, we need to put a small jumper near the power socket.

Functionality	Date	Person	Result
Initialization	13-11-2013	AS/SzK	YES
Moving	13-11-2013	AS/SzK	YES
Rotation	13-11-2013	AS/SzK	YES
Speed change	13-11-2013	AS/SzK	YES
Moving + Rotation	13-11-2013	AS/SzK	NO

First of all – we initialize the board and the whole system (connect wires, plug the power supply, assemble parts). The initialization is the first stage to understand that system turns on and functions. The second stage is to load the program on the board and see if the car starts operating as desired. The rotation tests – is to check how the engine responsible for rotation of the wheels functions. Another important part of it – is to check on how the speed of the car can be changed. At first, the car was moving too fast for us. But the speed can be controlled if the integer value inside the sketch is changed.

All the tests were successful except one. The rotation didn't work together with the movements. We could start them separately but indeed the program needed some special functions and improvements to provide this important functionality.

The car moved and we needed to wait for the Software part to check the operation of the whole system.

4.2. Software

The software integration tests took place at the same time than the **3.2.2**, actually when we tested the Arduino we tested also the Android and it was an integration test and a unit test at the same time.

This was true for all tests **except** for the following tests because there were more unit tests about them:

Functionality	Date	Person	Result
Take picture	12-12-13	AG/AD	Yes
Video stream	12-12-13	AG/AD	Yes

"Take picture" and "Video stream" functionalities were done by three different teammates, because of the Hardware part then Android part and improvements (bug correction, SSH improvement, etc.).

4.3. Overall system

These tests were held to test main functionality and stability of the whole system. Our car is connected with the phone using a Wi-Fi module and can be tested for its' main operation. Not all of the test stages passed successfully.

The tests are held in 3 stages, because we had technical issues (burnt boards) during our tests. Each test has important steps inside which show information about different car functions:

Initialization – the first boot, after connecting all the components and plugging in power supply.

Connection – checking the physical connection between the components of the car.

Movement – moving car forward and backward, controlling it by the phone.

Rotation – turning of the wheels in both sides.

Response – the car does actions when user interacts with the Android application.

Wireless Connection – the test to see if the Wireless connection is active between the phone and the board.

Application + Car – test to check if all the functions of the application operate properly and the car executes desired commands.

Play a sound (Honk) – the car has a built-in speaker and can produce sound when user presses the button in the application.

4.3.1. Stage 1

Functionality	Date	Person	Result
Initialization	14-11-2013	SzK/AS/AD	YES
Connection	14-11-2013	SzK/AS/AD	YES
Movement	14-11-2013	SzK/AS/AD	NO
Rotation	14-11-2013	SzK/AS/AD	NO
Response	14-11-2013	SzK/AS/AD	NO

We were able to start the system and even connect the phone to it. But the problem appeared when the battery discharged during tests (we thought it caused the problem). When we tried to load the program to test functionality of the board – battery died and Arduino switched off. We changed the battery but the board stopped working properly – all we could see is a single yellow LED flashing all the time. The built-in Wi-Fi module stopped responding to any commands and the hotspot didn't work. Later on we tried to recover the board, but all attempts failed.

4.3.2. Stage 2

In a couple of weeks we received a new Arduino and could set the tests again. We arranged a meeting and connected all the components. First tests were done with Arduino powered by a USB cable, and second part was with small jumper on the motor shield to provide a power from the battery.

Functionality	Date	Person	Result
Initialization	26-11-2013	SzK/AS/AD	YES
Connection	26-11-2013	SzK/AS/AD	YES
Movement	26-11-2013	SzK/AS/AD	YES
Rotation	26-11-2013	SzK/AS/AD	YES

Response	26-11-2013	SzK/AS/AD	**YES**
Wireless connection	26-11-2013	SzK/AS/AD	**NO**

As with the previous Arduino, this board caught the same system failure – the Linino OS on the board was burnt. The failure occurred when we connected a jumper to the Adafruit shield – we wanted to test the car without wires. The battery should power both shield and the main board, but Arduino YUN was the only one board which does not have a Voltage Regulator. Without it – it may support only 5V and nothing more. The battery gives 9.7V and cannot be used together with this jumper.

4.3.3. Stage 3

Successfully, our project manager took into account the possibility of another system failure and ordered the Arduino board on his name. It had arrived just a couple of days later after the second Arduino was burnt. The next series of stage started and finally we could test the system all together.

Functionality	Date	Person	Result
Initialization	28-11-2013	AD/SzK	**YES**
Connection	28-11-2013	AD/SzK	**YES**
Movement	28-11-2013	AD/SzK	**YES**
Rotation	28-11-2013	AD/SzK	**YES**
Response	28-11-2013	AD/SzK	**YES**
Application + Car	28-11-2013	AD/SzK	**YES**
Wireless	28-11-2013	AD/SzK	***NA***
Play a sound (Honk)	28-11-2013	AD/SzK	**YES**

During these tests we could successfully check the operation of the application together with the car. The only one trouble occurred was wrong commands (instead of moving back and forward the car was turning right/left and the same about turning). This problem was easily solved by changing a couple of lines in the code. The car was working and the application functioned well. The last step was to add a voltage regulator or another source of power to supply the main board (Stage 3 tests were held with Arduino connected to a laptop by wire).

5. Non-regression tests

Each time we added a new functionality, all previous tested functionalities have to be tested again to be sure the new functionality is not breaking something. These tests are called non-regression tests

We have not so many functionalities so it was easy to test all of these.

6. User acceptance tests

6.1. User acceptance test:

The tests are final ones. They are held with the final customer to be sure everything works the way we wrote it on the "*Requirement Specification*" document. Those tests will be done for each version of the entire application.
We chose our supervisor as tester because he knows well enough the application and the features from the user point of view.

6.1.1. Version 1.0

This is the very basic version of the car. All the functionalities are cut to the basic movements. It can only move backward and forward. The supervisor has a chance to test it for the first time and to see the further improvements. This is a basic version and test for 1.0 will be used to compare how new functions operate and if the car is still able to move.

Functions	Date	Tester	Result
Movement	28-11-13	Torben	**Yes**
Basic Funct.	28-11-13	Torben	**Yes**
Attractiveness	28-11-13	Torben	**Good**
Integrity	28-11-13	Torben	**Yes**

6.1.1.a. Details about 1.0 user acceptance tests

All the tests we done with the car on a big glass to avoid movements, the car wheels didn't touch the floor. We tested the following functionalities: Go forward, backward, turn left and right, make a honk sound.

Everything worked except for the turn, because we inversed the executed commands in the program so turn right button was actually turn left on the car. We fixed it and took a video which we uploaded on YouTube (https://www.youtube.com/watch?v=F-oRrcYfv8o), we sent the link to Torben to show him how the car was working.

Figure 2 - The car on a glass during tests

6.1.2. Version 1.1

The second version of the car implements new functions into its' behavior. Now the car is able to initialize the system, load special settings and run different functions (which user declares). Using settings user can control the operation of the car – regulate the max and minimum speed, rotation of the wheels and stop it when he needs to. Also, the car can be programmed to support executing pre-loaded instructions and execute them with the set up delays.

Function	Date	Tester	Result
Initialization	06-12-13	AD	**Yes**
Movement	06-12-13	AD	**Yes**
Control	06-12-13	AD	**Yes**
Performance	06-12-13	AD	**Excellent**

6.1.2.a. Details about 1.1 user acceptance tests

As the 1.0 user acceptance tests, we used a glass to avoid movements, this time we checked the previous functionalities, settings, acceleration and deceleration.

To test those settings, we changed the value of "Speed acceleration forward" from 1 to 3 and accelerated the car once we saved, we saw that the speed increased faster when moving forward than backward, so settings were correctly sent to the car each time the Activity was loaded.

6.1.3. Version 1.2

During these tests we should test the added functionality and improvement of the existing behavior. The car is packed with a camera and should be able to take pictures (approximately every 1-2 seconds) and show them on the phone screen as a whole video stream.

This version was not reached because Alvaro preferred work directly on the 2.0 version with a real video stream instead of a fake video stream using refreshing pictures.

6.1.4. Version 2.0

This is a major release of the car. Now a lot of minor bugs and issues were fixed. We can test the car and it provides a good performance.

One of the key goals is achieved – we can start recording video and send a real-time video stream to the screen of the smartphone.

Function	Date	Tester	Result
Initialization	12-12-13	AD	**Yes**
Movement	12-12-13	AD	**Yes**
Control	12-12-13	AD	**Yes**
Overall Performance	12-12-13	AD	**Good, troubles when out of range Wi-Fi**
User interface	12-12-13	AD	**Yes**
Video stream	12-12-13	AD	**Yes**
Photo stored	12-12-13	AD	**Yes**

6.1.4.a. Details about 2.0 user acceptance tests

Here we will describe in details tests we did on the 12th December when we tested the 2.0 version with all the functionalities from a user point of view. The following pages will explain in details how we tested the entire application and which issues we encountered.

From 9th December to 12th December, we worked on the video stream, the pictures both on hardware and software. But on the 12th December, the tester tested everything.

The tests were on both hardware and software but mainly on the software part.

We start by fixing all the components, because we wanted to use the car in a real environment, the speed could be huge and in case of crash we didn't want to break something.

Figure 3 - Hardware equipped with camera, regulator, YUN, Motor Shield, wires and batteries.

This was the first step, once it was done we started by powering the YUN using the regulator.

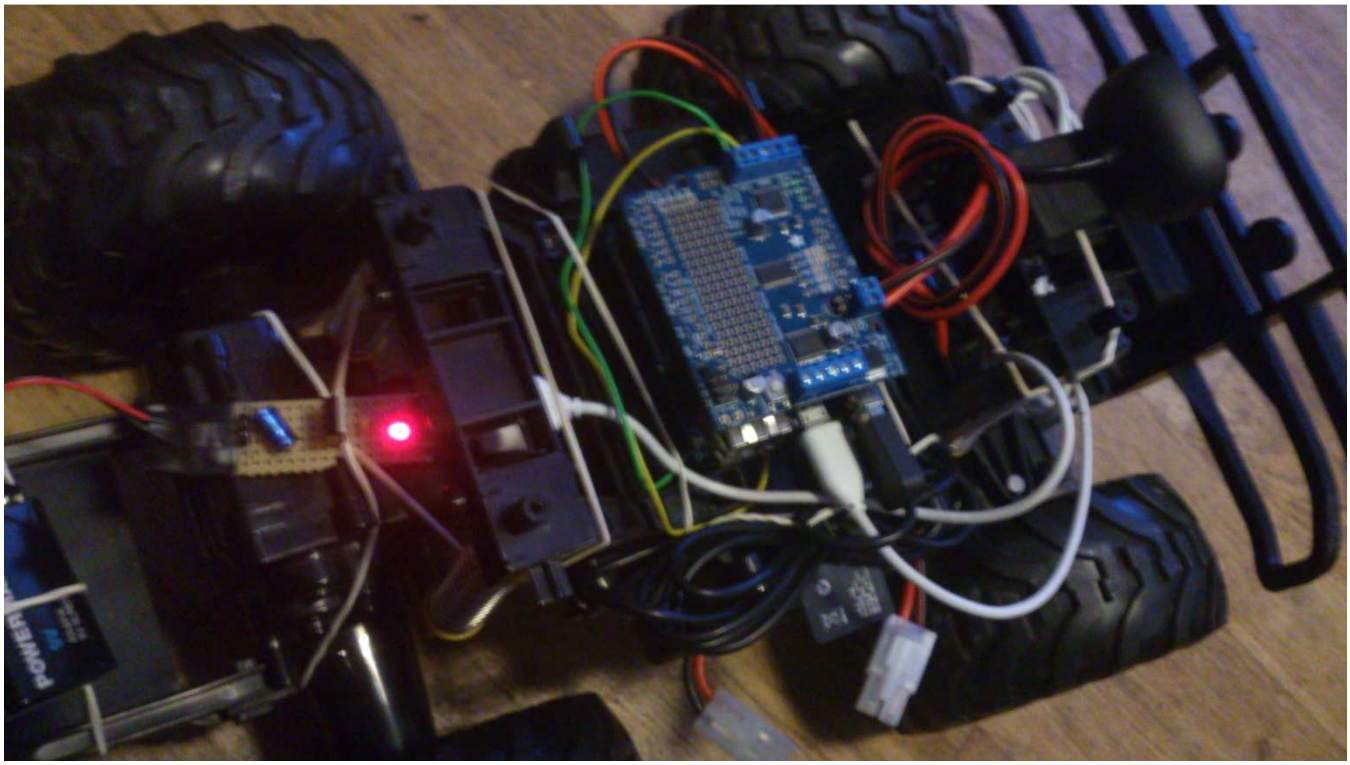

Figure 4 - Car powered by battery 9V using regulator

Then we powered the MotorShield using the big battery, stored below the car.

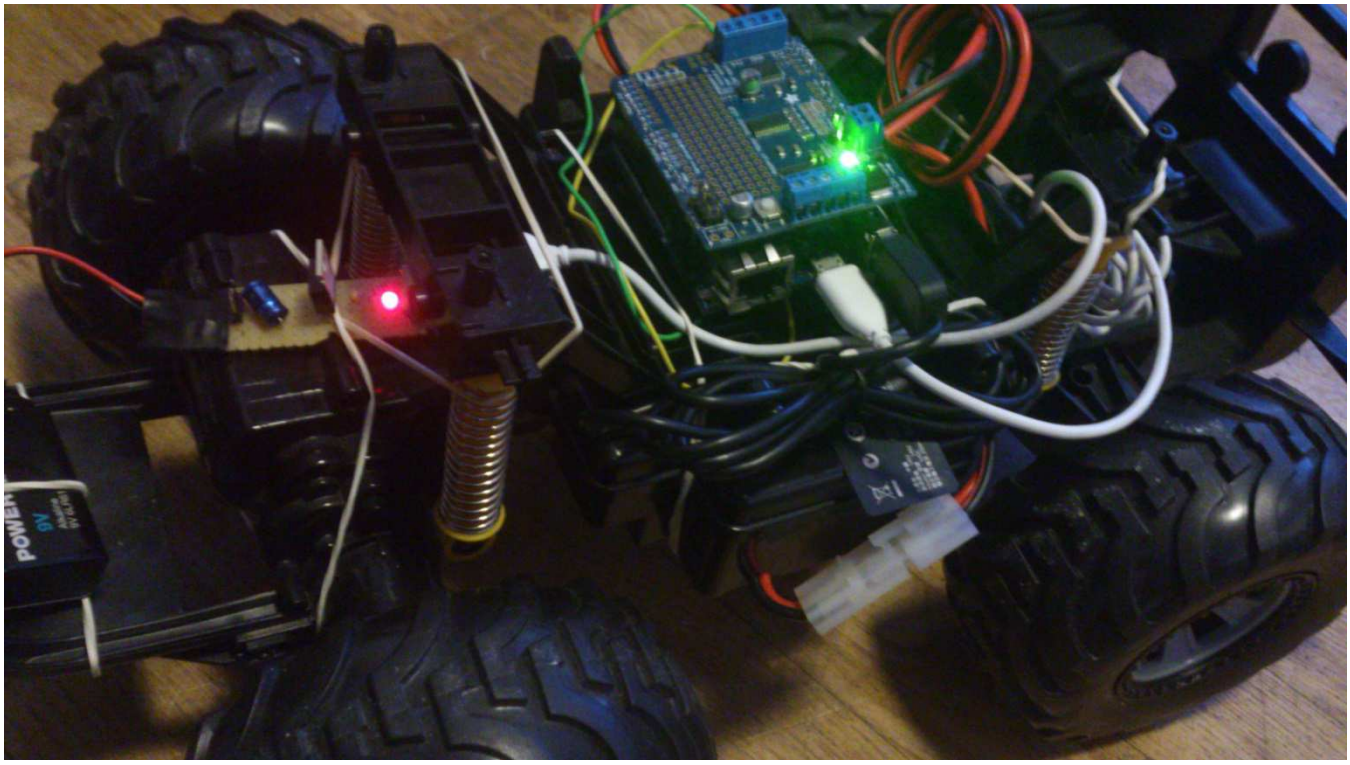

Figure 5 - YUN and MotorShield powered with their own battery

This first part was successful - the regulator worked well and the Arduino generated its' Wi-Fi network.

We connected to the network with the Android phone and ran the Android application to control the car - the camera was automatically started using the SSH command. The first screen was black, because the application has to be improved. Actually, if the camera was not running before, we don't get the video stream on the phone, we need to refresh the Activity to reload the video stream; this time the stream is available and we can see it on the phone. (This problem appears only the first time when the application is launched, if the camera wasn't running before)

We were able to control the car, do everything we designed such as take pictures and see the video stream on the phone. (Except the honk sound but it works, we forgot to add it)

Figure 6 – Car tested in a real environment

We encountered issues with the Wi-Fi range, the video stop streaming when the car was too far from the Android phone. The range is about 10 meters, depending on the environment; we tried on the floor of our apartment. Once the video stream stopped, we were not able to get it again even if we could still control the car; it's probably because the streaming needs more bandwidth than sending simple requests.

We got a problem there, because the Android application had some internal issues and wasn't responding, we didn't know what happened but we were not able to control the car or see the video stream anymore. All functionalities didn't work after the "out of range" issue.

We had to understand the problem, we unplugged the YUN and the Motor Shield, and then we restarted the YUN. Once the Wi-Fi network was reachable we executed the Android application. But still the same - no video and no car controls, something was wrong.

We still don't know exactly what happened by now, could be several different issues at the same time. We reset the YUN, no effect. We reset the YUN Wi-Fi/Linino then, which took us about 5-10 minutes. Once it was reset, we were able to control the commands, but still no luck with the camera.

So we used PuTTy to access to the camera but we got an unexpected error. Pierre helped us to fix it, but we didn't understand what the problem was, the packages were installed, the SD card was found, we couldn't reach internet to reinstall the packages so we quitted PuTTy.

Figure 7 - YUN powered by the computer during the debug

Then we tried to use the battery again and regulator powered YUN, but we were not able to control the car then. We thought it was because the camera was plugged on the YUN, we tried again after removing it, still the same.

In fact, the problem was the embedded battery; it was powerful enough to power the YUN but not enough to power the camera. We changed the 9V battery and tried again, everything worked.

By now, we think we did a lot of tests while the YUN was powered using the regulator battery instead of the computer and because the camera is plugged to the YUN and has to power a video stream, it consumed too much power too fast.

We tried again to use the car in a real environment, no problems occurred and everything worked as before.

Then we tried again to go out of range, we lost the video and the car control. This time we just refreshed the activity on the Android application and we got back the control and the car, but still nothing with the video.

We tried to disconnect and connect it again, but still the same, black screen.

We needed to unplug the YUN power and plug it to make the camera work again.

We think there is something with the Android application, maybe the thread used to connect to the Arduino is not killed when the Wi-Fi signal is lost and then there is still a "dead" thread running, but maybe it is killed well, but in this case it could be the Arduino application which is still in the infinite loop and don't know that its' client is dead, it could be and actually we think this is the only reason.

We tried some other tests that "confirmed" this hypothesis:

We moved away from the car with the Android smartphone and we lost the connection, the car still tried to go left even if we killed the application. We unplugged the YUN power and plugged it again, the car instantly tried to go left. It tried until the Arduino program restarted and shut down the motors some seconds later (~5-10).

One issue there is probably that the Arduino program doesn't know when its' client is disconnected when the Wi-Fi is lost, because it's not a proper disconnection. Maybe we need to do something more in the Arduino program to be aware that the client is lost even if he is not disconnected.

As we said before, we think there is another issue about the threads on the Android program. The thread that execute the SSH command is never killed for instance, but I don't think it is our issue there.

When the communication is lost because of the "out of range" reason, the thread that sends our commands to the Arduino is not killed but becomes inconstant, we don't really know but it looks like something still alive uses the thread and the socket as well so the Arduino cannot be reached from a new socket and the car cannot be controlled.

It's a hypothesis, but it could be true.

These issues don't appear all the time, sometimes it is one, sometime they are both, sometimes we need to restart the Arduino after an "out of range" and sometimes we just need to refresh the Android Activity, depending on the case. We still need to improve our applications to avoid those issues.

As conclusion for this user acceptance test, everything works fine while the car is not too far from the Android phone. A solution could be to use an antenna to have a better Wi-Fi signal; it is possible to do it with Arduino.

We made a short video about last test and uploaded it on YouTube.

(https://www.youtube.com/watch?v=Vfax7QpjkhE)

6.1.5. Version 2.1

We didn't reach this version, we didn't buy the proximity sensor nor we designed its' use, but we implemented the "Honk" functionality in 1.0 version instead of 2.1.