Wi-Fi Controlled Car

Requirements specification

This document contains the requirements specification about the project of the group AAD1 summer 2013.

Document title	Requirements specification
Document ID	WCC-RequirementsSpecification
Authors	Ambroise Dhenain Szymon Klepacz Anatolii Shakhov Alvaro Garcia Pierre Le Texier
Supervisor	Torben Gregersen
N°. pages	24
Date	30-08-2013

Document history

Date	Audit	Author	Description
02-10-2013	0.1	AD	First version of specification. Template only.
31-10-2013	1.0	AD	Opening, members.
01-11-2013	1.1	AD	General description, functional requirements.
02-11-2013	1.2	AD	System overview, functional requirements.
08-11-2013	1.3	AD	Corrections after meeting with Torben. Functional requirements, user interfaces.
14-11-13	1.4	AD	Limitation, future, user profile, development process, environment, non-functional requirements.
17-11-13	1.5	AD	Corrections after meeting with Torben. System functionalities, limitations of the system, development process, functional requirements, non-functional requirements, glossary.
21-11-13	1.6	AD	Corrections after meeting with Torben. System description, environment.
07-12-13	1.7	AD	Add picture. Add diagrams.
08-12-13	1.8	SzK	Corrected spelling, added missed non-functional elements.
09-12-13	1.9	AD	Add Activity diagrams.
09-12-13	1.10	PLT	Update diagrams.
11-12-13	2.0	AD	Final version. Update pictures. Corrections after meeting with Torben.

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-RequirementsSpecification
No. pages	22

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

Place and date:

Authors Supervisor

Contents

Tables of figures

1. Opening

1.1. Purpose

The purpose of this document is to describe the requirements for our project, a car controlled remotely by Wi-Fi. The system is a toy actually, a small car controlled by a phone application. The goal is to play with it and have access to more functionalities than a usual toy car like sound, recording, pictures and more.

Basically the system is pretty simple, you need the car and a phone application –we built only the android application- to control the car.

Once the application started, you have the view of the camera on the phone's screen and button to control the direction and the speed. You just have to touch the button and the car will move, following your orders.

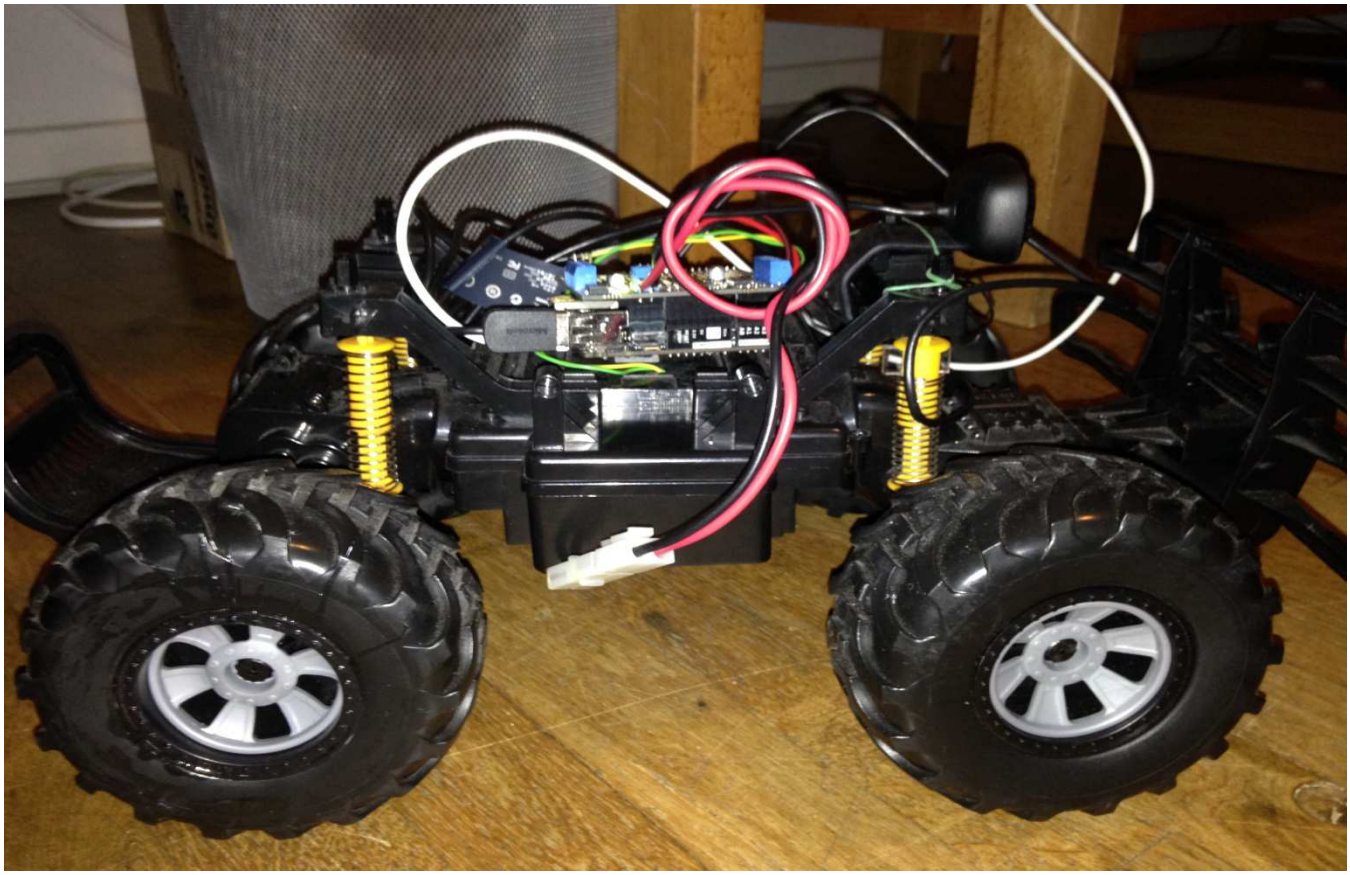

Figure 1 - Car borrowed at the Aarhus University

1.2. Reading instruction

Chapter 2: Contains the general description of the system. Such as system functionalities, limitations of the system, the future of the system and requirements.

Chapter 3: Describes the functional requirements with Use Cases diagrams.

Chapter 4: Describe the non-functional requirements such as motor shield, board, etc.

Chapter 5: Describes the user interfaces.

Chapter 6: Glossary which contains explanations about some words and abbreviations used in this document.

2. General description

2.1. System description

The “Wi-Fi controlled car” is to be designed to be easy and user friendly to use with some extra functionality which doesn’t exists on a simple car remote game like live video stream, pictures, collision warning and more.

The system will consist of two different parts, the car first, which is a simple car robot we borrowed from the University. It can move basically forward, backward, turn left and right. The second part is a phone application, actually we will use Android only for the moment. This could also be done for Windows or Apple phones. This application will shows to the user the video stream from the camera on board of the car and some buttons for control the car. (Turn, go forward, etc.)

The car will have its own battery for all the components on board (motors, camera and so on).

The phone application will be provided for free.

Figure 2 - System overview

2.1. System functionalities

To describe the system functionalities, we use the UML Use case diagrams. There is one diagram per version. The Use Cases diagrams illustrated below show the actors that can interact with the system and show also which kind of action they can use/do.

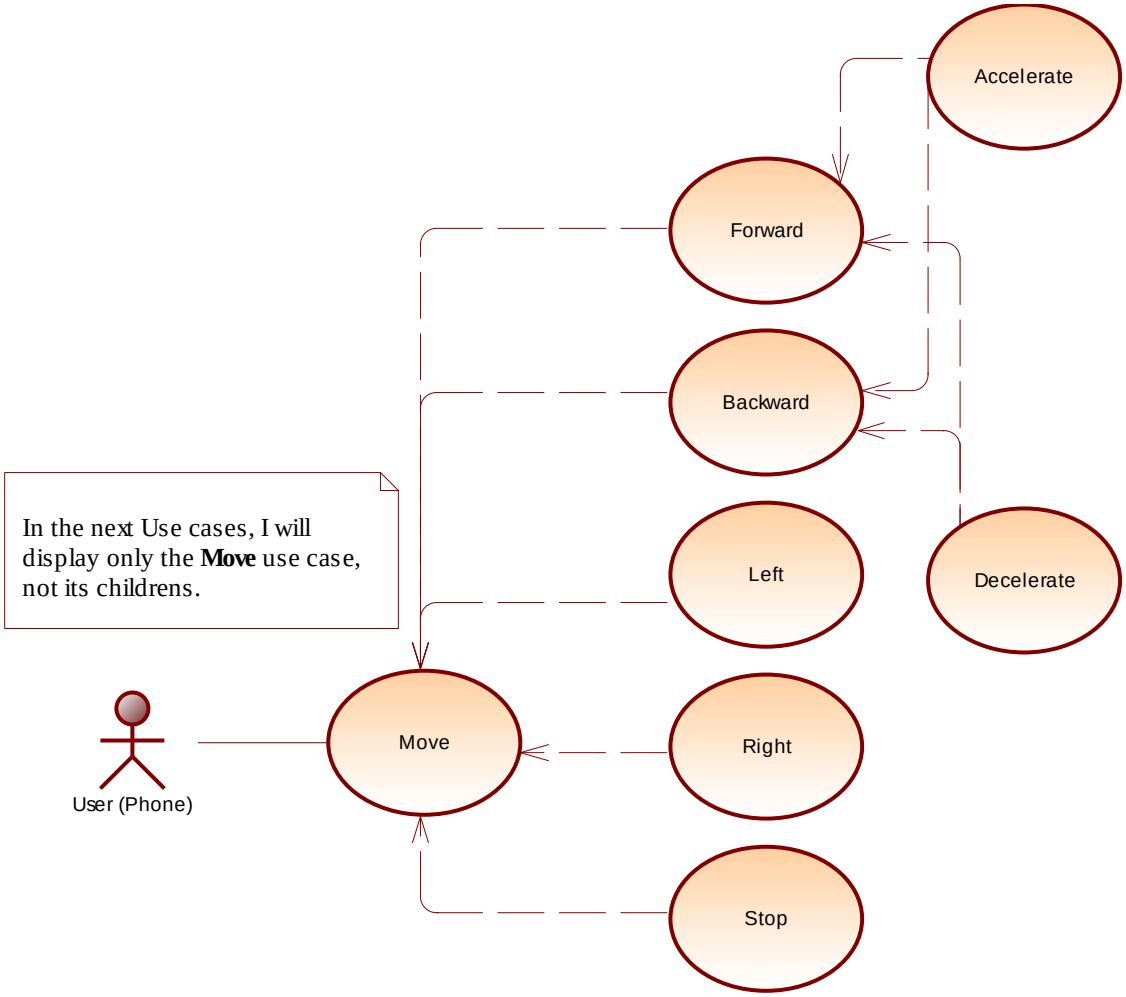

Figure 3 - Use case - System version 1.0

This is the version 1.0 of the system, which can basically only move, the details about all the possible movements are described too *and won't be describe in the next versions*.

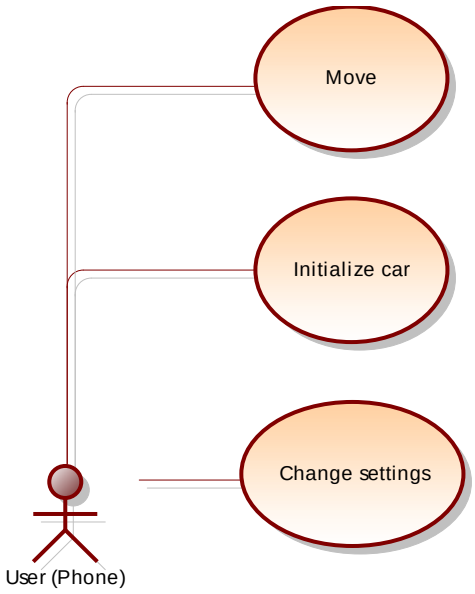

Figure 4 - Use case - System version 1.1

The version 1.1 improves the system with the settings and the car initialization to load these settings. Basically the settings can be used to limit the speed of the car, define its behaviour and so on.

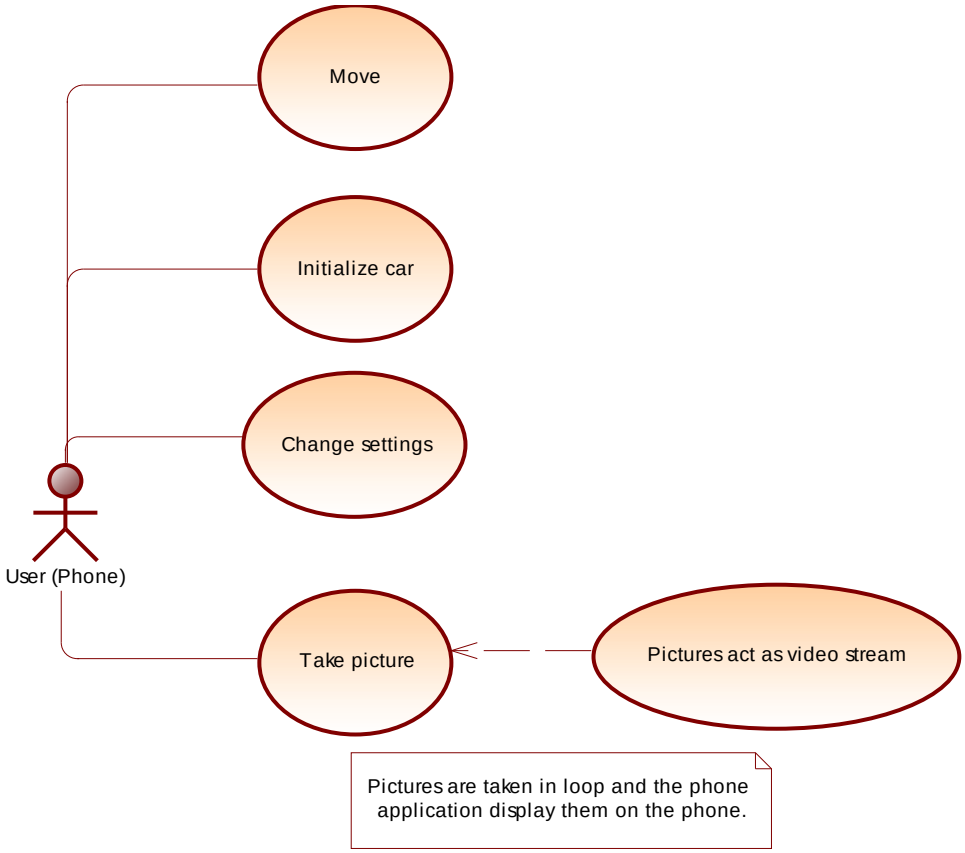

Figure 5 - Use case - System version 1.2

The version 1.2 improves the car with an embedded camera to takes pictures to display these pictures on the phone application. Basically this is a simulation where the pictures act as a video stream but it will be probably slower than a real video stream.

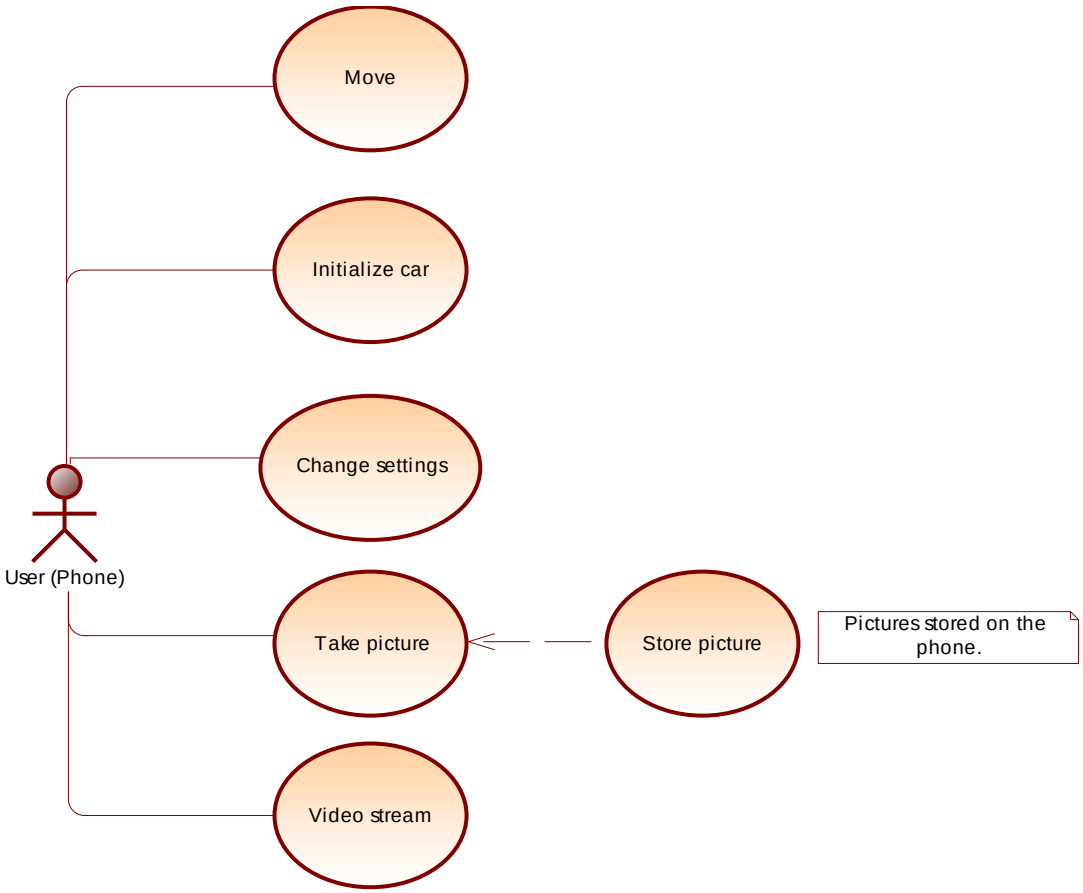

Figure 6 - Use case - System version 2.0

The version 2.0 is a major release of the system because we will reach our goal that is to have a video stream on the phone and control the car almost in real time with good performances. The camera will be able to share a video stream but also take pictures but this time the pictures will be stored on the SD card, not displayed on the phone screen.

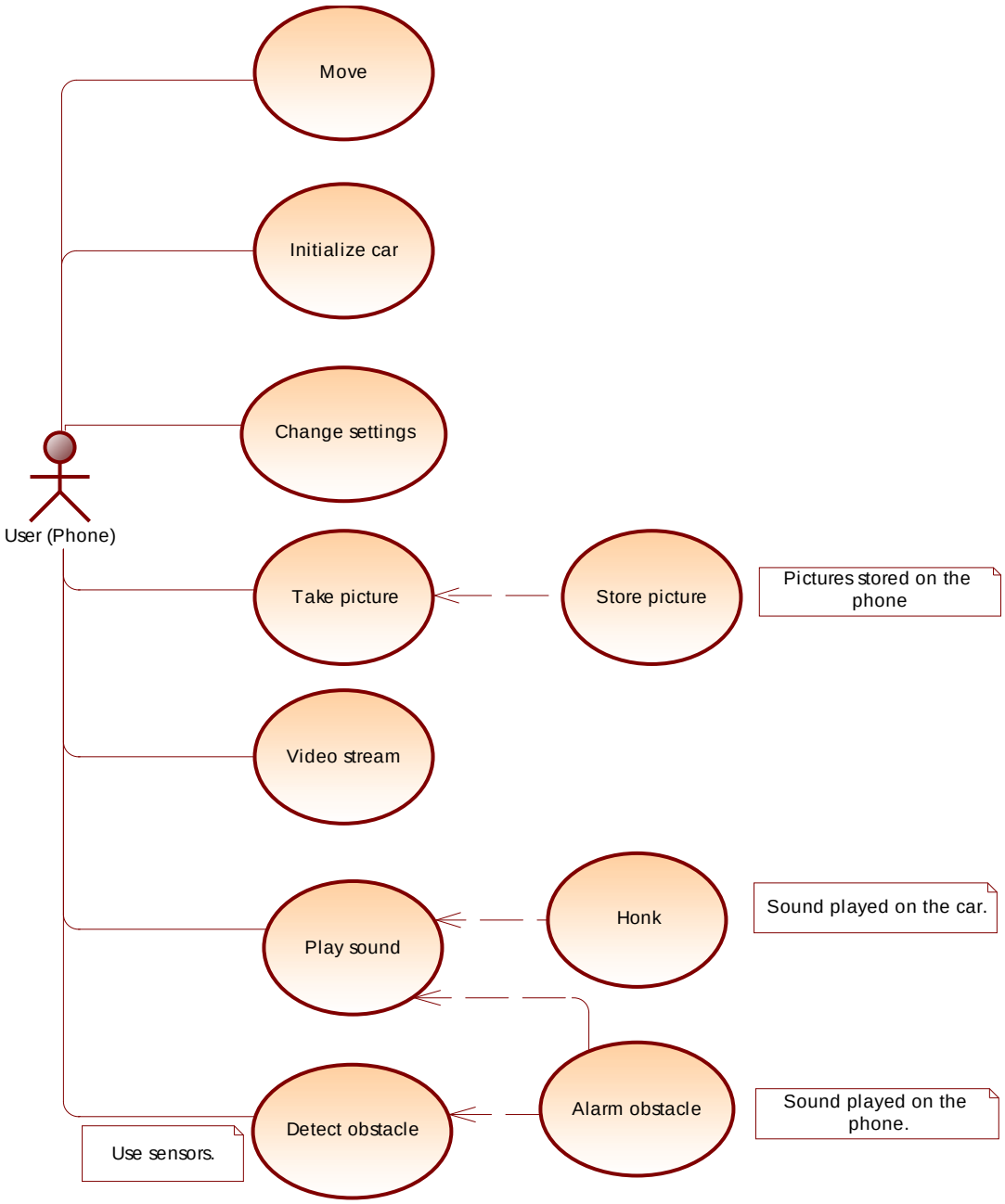

Figure 7 - Use case - System version 2.1

The version 2.1 of the system will add the sound to the car and to the phone with a proximity sensor to detect collisions. Basically it will be the same functionalities we can find on a real car by now.

All these versions are improvements (except 1.0), we **would like to reach the version 2.0** and have a video stream to show on the android application but it will depends on the issue(s) we will encounter on the software and the hardware.

2.2. Limitations of the system

The system has some limitations, for instance there is the radio communication because of the wireless limitation. We suppose be able to control the car in about 10 meters range before the Wi-Fi signal shutdown.

We designed the application to use video stream instead of pictures but we could have some limitations using the camera that could be not powerful enough for live stream, or the Wi-Fi, which could be not fast enough for transmit the live stream.

The car embedded a battery 9V. The user will need to charge or change it from time to time.

The user phone application will be translated only in English but could be easily translated to other languages.

The size of the program stored in the Arduino flash memory is limited, about 32Ko, it's too small and will maybe not be enough to do everything we want to do.

2.3. Future of the system

The system could be extended with new embedded hardware like buzzer, GPS tracker, light (night) and so on. By the way, the application could be upgraded too in order to manage new components. Actually, we could add components in the future by blocks that are independent from the others and it's really easy to do by this way.

It's also possible to use the SD card to add some scripts and increase the possibilities, because we are limited by the size of the program stored on the Flash memory, maybe we could extend it using external storage as SD card. But it won't be able to store binary/HEX code as the Flash memory, but we could use other language as PHP, Python or Javascript to extend the possibilities. It's possible to use Node.js for instance, really easy to use sockets by this way.

There are also a lot of settings about car management, actually, the user will be able to decide itself of the speed increment, the minimal and maximal speed and so on. It could be useful to limit the risk if a child plays with it, for instance.

2.4. User profile

The users that will be play with the system could be child (about 7 years old) or adults, they just have to know how to use a smartphone, launch an application and push some buttons. There is no really limitation about the age while they know how to use the phone and are careful because the speed of the car can be really quick.

2.5. Development process

The development process starts by an initial planning which describes all basics tasks and functionalities we want to implement in the system. Each time a task is done there can be also a Unit test to do (in most cases) and if the task is updating something there is non-regression test too. If the task is a final task that will be used directly by the user then there is also a User validation test to do.

If some tasks are completed early then we could add some other tasks on the planning.

We will use an iterative and incremental process, start with the basics and improve the application/car when we will reach our goals and discuss about the next steps with our mentor. Each step is represented by a Use case diagram which describes all functionalities.

2.6. Technologies

The car has on board an Arduino YUN with a program written in C++ language.

The phone application will be developed for Android smartphones only for the moment. The minimal required API version will be API 11, because some of the functionalities we need to use.

2.7. Environment

The car can be used indoor or outdoor, but it should not be used outside under the rains. It has, for the moment, no protection against water so it could kill the electronic components as the CPU board, the motor shield, or both.

For instance, if you use the car in a house, the Wi-Fi could be diminished because of the walls and the communication signal could be too weak to control the car correctly.

All electronic environments which affect the Wi-Fi signal can interfere with the communications.

3. Functional requirements

The purpose of these functional requirements is to describe each functionality using Activity UML diagrams to understand how the application will work. *These diagrams could be update in the future*.

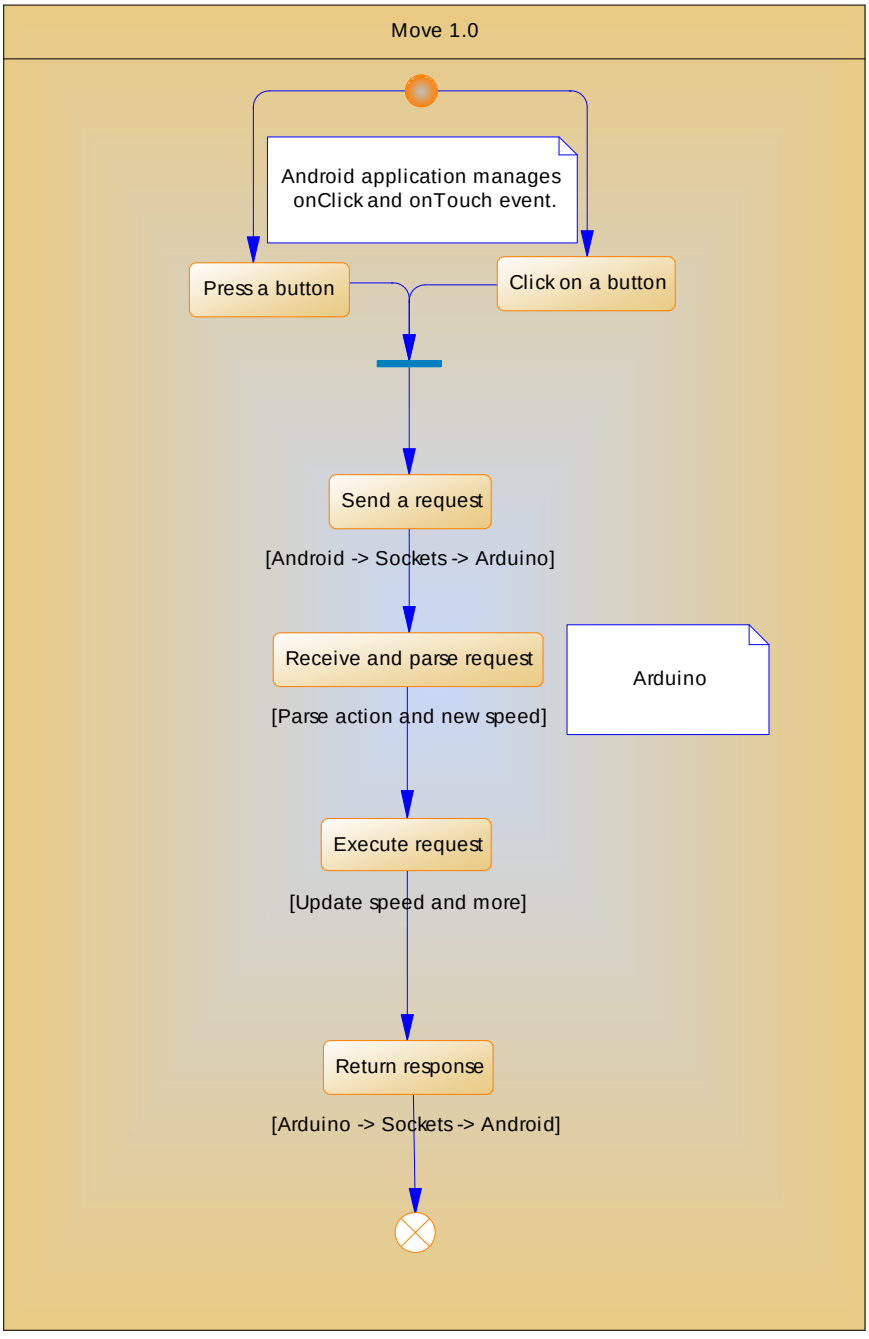

Figure 8 - Activity - Move (1.0)

The “Move” action is the first step, it’s the most basic action the car is able to do. It’s no more than go forward, backward, turn left and right, stop and control speed increase and decrease.

All these actions use the same request and act as the same way, they are sent using socket to the Arduino which will parse the request and get the action and the speed. It just has to update the car’s speed and execute the action requested.

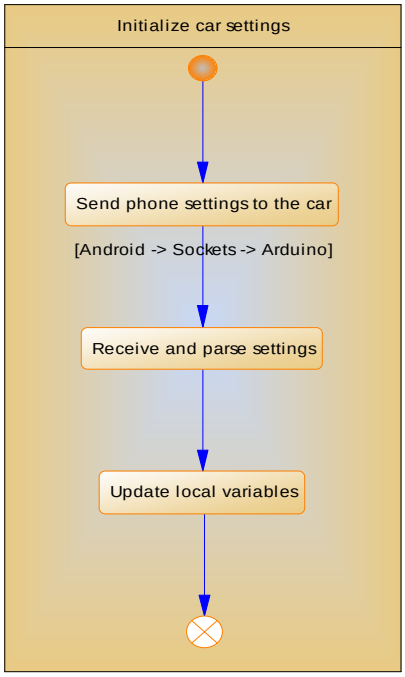

Figure 9 - Activity – Initialize car settings (1.1)

When the Android application starts, it initializes the process with the car and sends the phone application settings to the car such as the "honk" alarm frequency.

*The "**Initialize car**" is actually ambiguous, we are talking about initialize the car settings, because the car initializes itself, from the Android application we can only initialize the settings.*

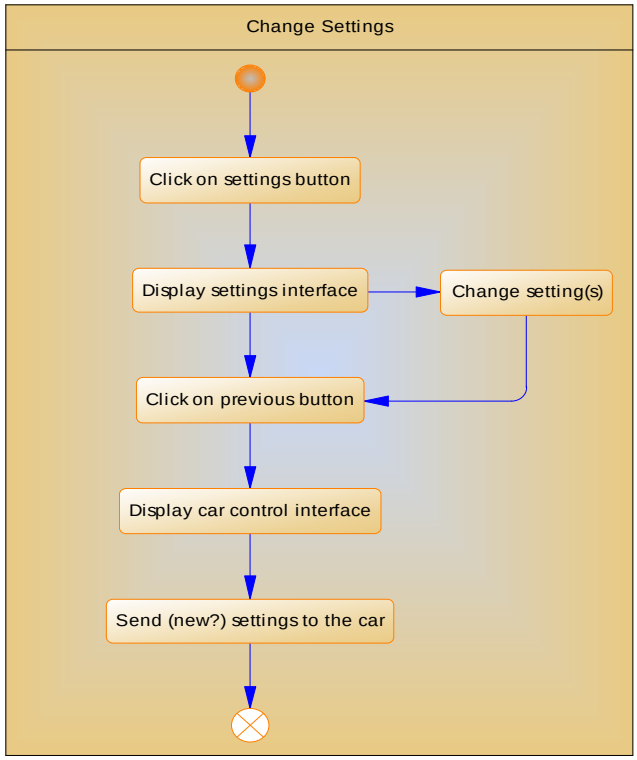

Figure 10 - Activity - Change settings (1.1)

The settings are automatically sent to the car each time the Car control interface is displayed (onCreate event), even if there was no change.

*The process is not really different between" **Initialize car**" and "**Change settings**", because when the settings are updated, the car settings are automatically refreshed*

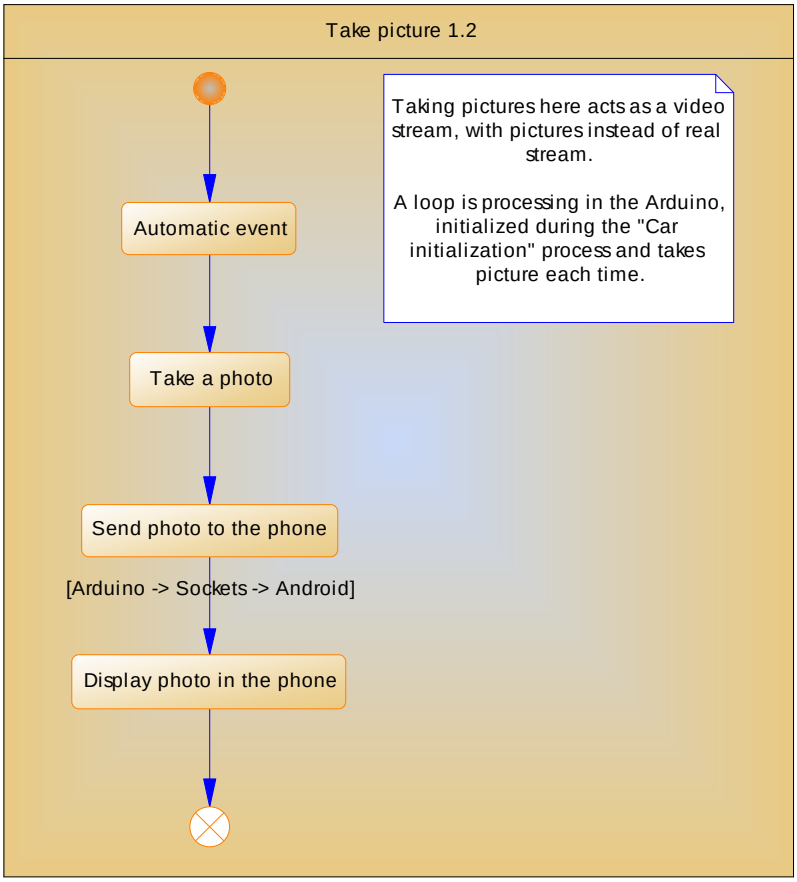

Figure 11 - Activity - Take picture (1.2)

This is the first version of the process "**Take picture"**. It's managed in the Arduino itself, in a loop started during the car initialization process.

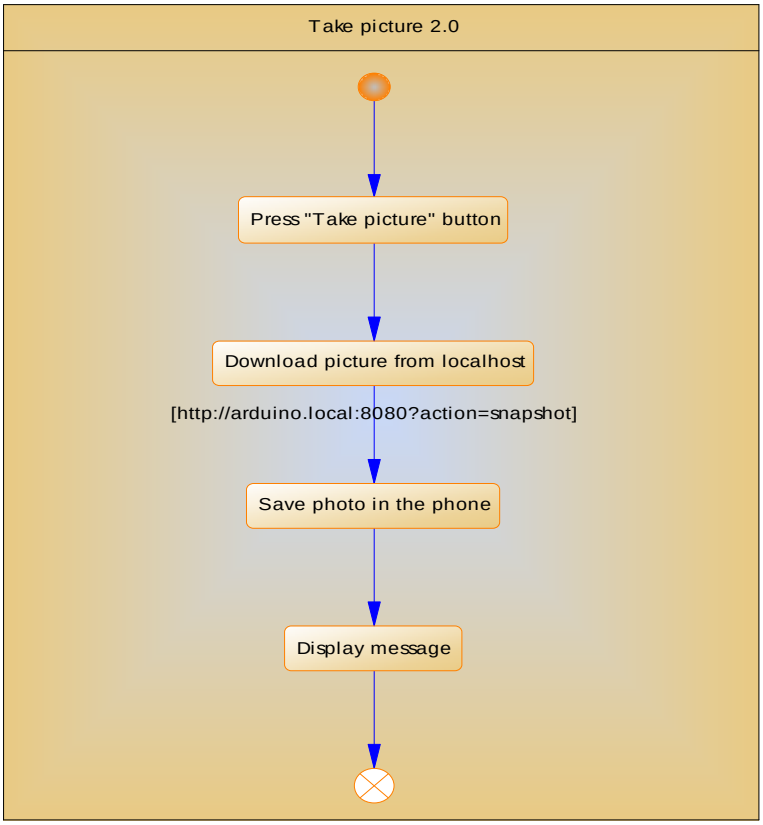

Figure 12 - Activity - Take picture (2.0)

In this second version of the process "**Take picture"**, the car can use both picture acting as stream and real stream. It can also take pictures on demand and in this case, store them on the SD card or display an error.

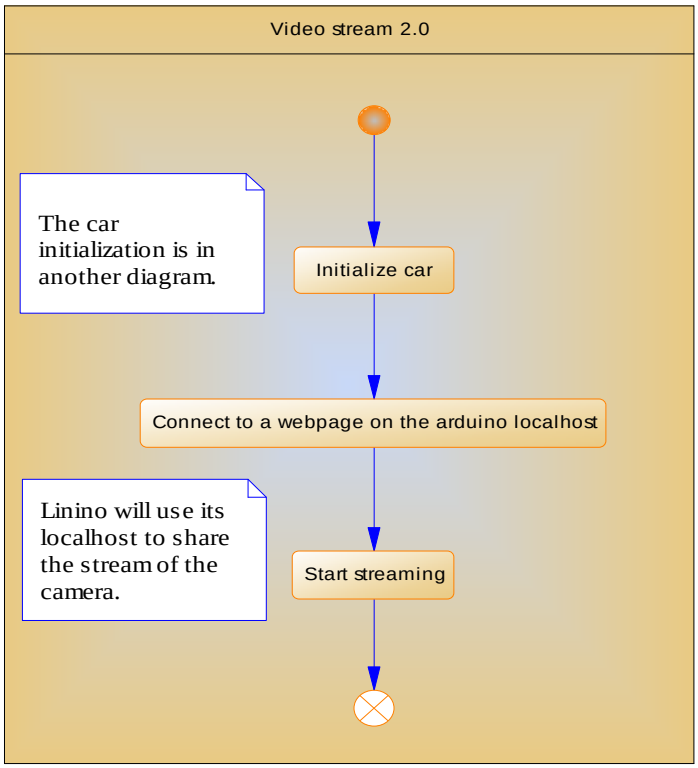

Figure 13 - Activity - Video stream (2.0)

The video steam is provided by an embedded camera, Linino use the stream and redirect it to the localhost network. All devices connected to the Arduino network are able to see the stream. The phone application will display the stream in its view. This process is initialized during the car initialization.

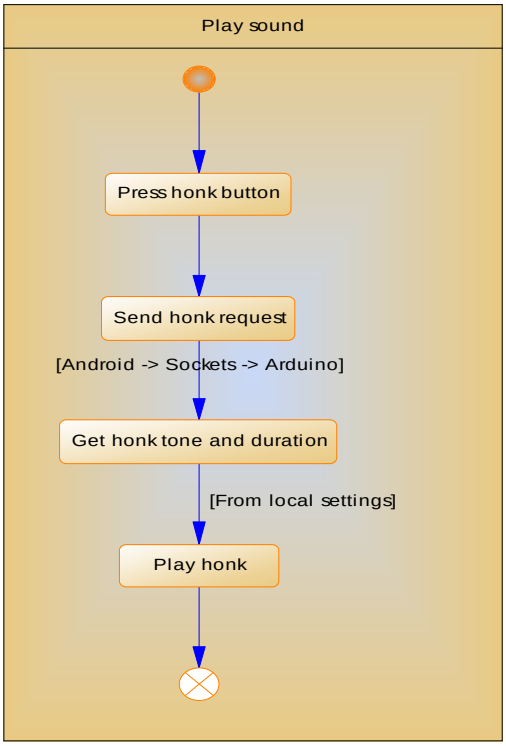

Figure 14 - Activity - Play sound (2.1)

The sound is played on the car, it acts as a honk. The user has a button on the Android interface to play the honk. The settings contain the tone and the duration to play, the user can manage these settings to the Android application, and they will be automatically refreshed in the Arduino program once saved.

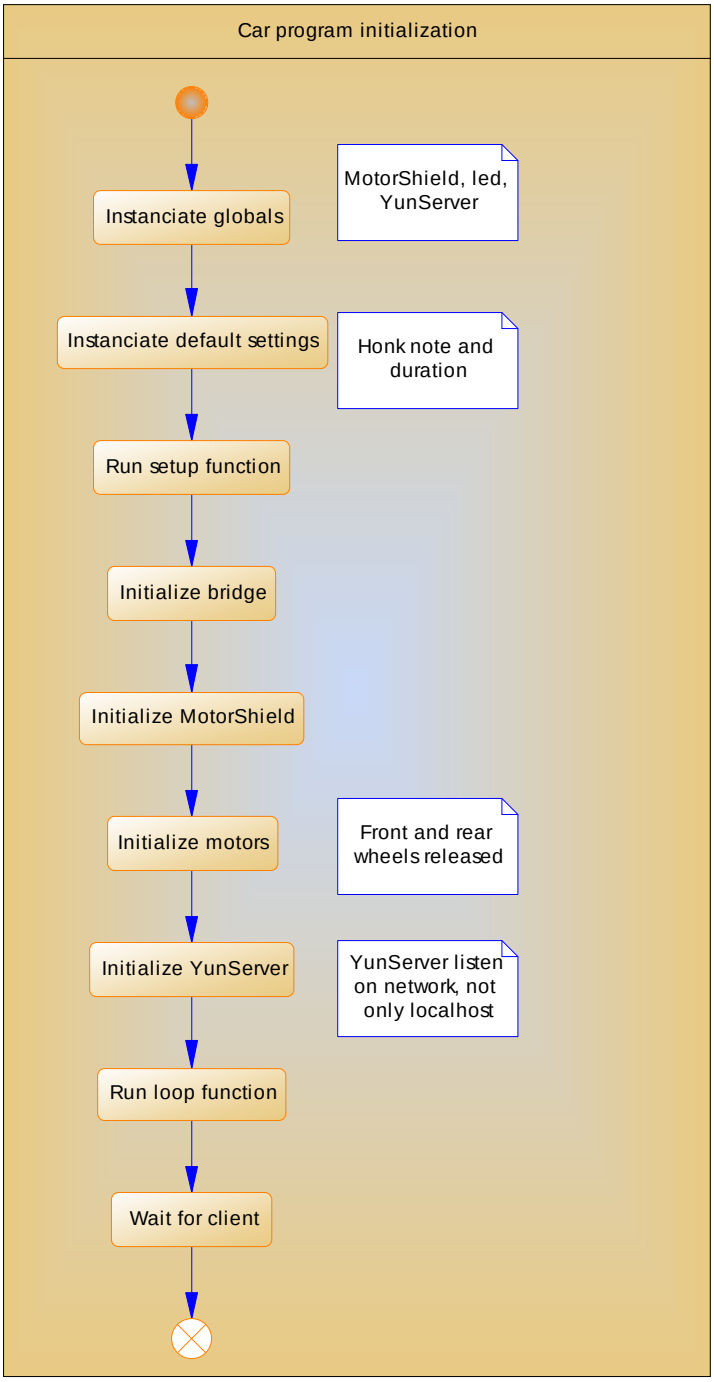

Figure 15 - Activity - Car program initialization

This activity diagram show how the Arduino program is initialized. Once the initialization completed, it wait in a loop for an available client to listen.

*The diagrams "**Detect obstacle**" and "**Sound obstacle**" are missing because we won't implement the proximity detection functionality, we don't have enough time and we don't have the component.*

4. Non-functional requirements

To build this project, we need some non-functional requirements, especially hardware:

- A car skeleton including wheels, wires, minimal design.
- A CPU board which is an Arduino YUN where the program will be stored to control the car.
- A motor shield which is an Adafruit Motor Shield to control the wheels to move and turn.
- A smartphone with Wi-Fi and the application installed.
- An area where control the car with enough space.
- A camera, Microsoft 3000 able to record and take photos.
- Voltage regulator
- Battery

5. User interface

Figure 16 - Mock up - Select device

This interface is about choosing the device to connect between several devices before connect to the car.
Actually, this interface won't be used in the Android application but could be used in the future.

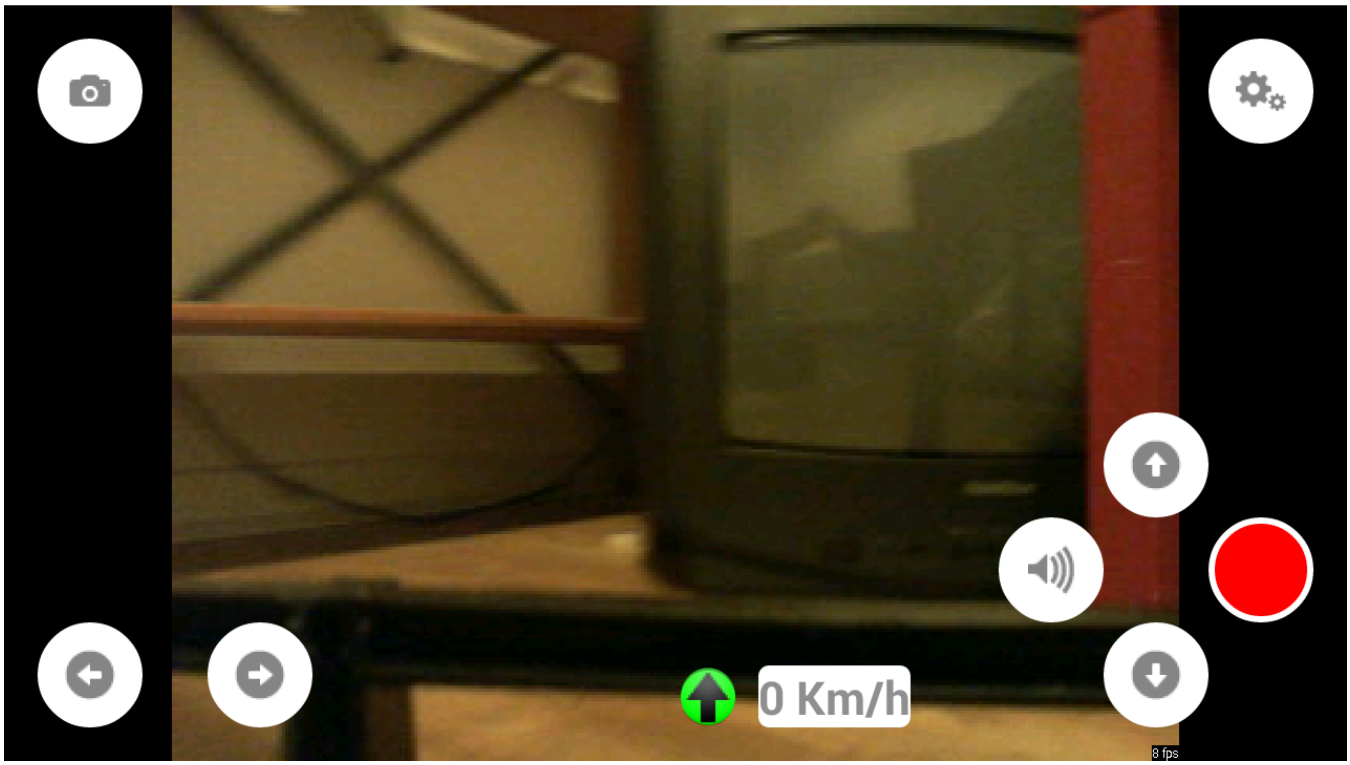

Figure 17 - Control car interface

This interface is the main interface, where the user can see the image from the embedded camera and choose direction, take pictures and more. Settings can be accessed by the top right button.

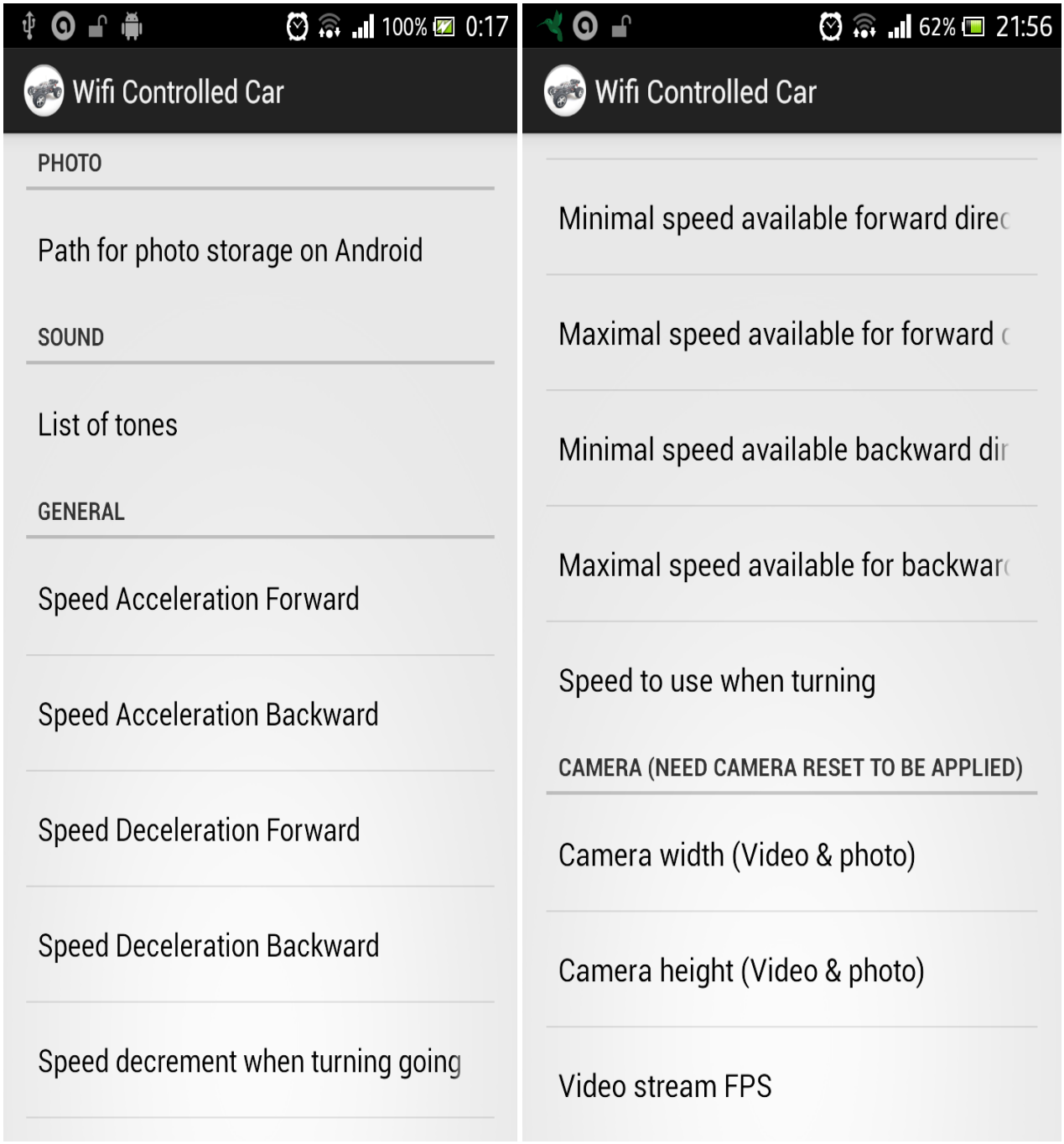

Figure 18 - Mock up - Settings

This interface is about phone settings, but these settings will be used in the car too for some of them, for instance, the “list of tones” will be automatically send to the car, because the honk is played on the car.

Most of the settings belong to the Android application, such as the motor management or the camera.

6. Glossary

WCC: Wi-Fi controlled car, the system as a whole.

Linino: Linux OS embedded on the Arduino board.

Arduino YUN: Arduino is the microcontroller used to control the car. We chose the YUN model.