A Requirements Specification to build a

Remote Controlled Car

9 May 2016

Nghia (Jason) Vuong

Table of Contents

1. Introduction

This document is a software requirement specification to my graduation project which involves building a remote controlled (RC) car and applying (programming) algorithm to make the RC car operate. In this document, I will first introduce the purpose and scope of this project in this section. Then, I will discuss in details what components a RC car has and how to put these components together as well as program to command the car in the later sections.

1.1. Purpose

The purpose of this Software Requirements Specification (SRS) document is to provide a complete description of both the purpose and functionality of the hardware and software of a RC car that is to be built. The main intended audience for this SRS is DIYers or RC car lovers who might be trying to understand the interactions of the system with other vehicle components. This document may also be helpful to RC-car manufacturers who are trying to improve and develop more features for their products. The tone of the document is fairly technical, but the goal is to deliver a simple-to-use product such that any DIYer with some programming and computer hardware knowledge can do it.

1.2. Scope

The product which I will build is a remote controller car, including a RC car and a controller (hardware) and the algorithm to operate the car and its add-on components (software). The controller allows users to control the car to move forward and backward (rear wheels) and change directions (front wheels). I will also add a sensor device to detect obstacles from the front. This features will help the car avoid obstacles when it is in the auto-ride mode. More add-on devices and features might be added to the project in version 2 or 3.

1.3. Definitions

Table 1.1: A listing of terms and acronyms used throughout this document.

SRS	Software Requirements Specification
RC	Remote controlled
DIY	Do It Yourself
L298N	Motor Drive Controller (control up to 2 motors)
RF24L01	Radio Frequency Transmission adapter
RAM	Random Access Memory
SRAM	Static Random Access Memory
EEPROM	Electrically Erasable Programmable Read-Only Memory
SDK	Software Development Kit
Wii Nunchuk	Wii controller for video games and simulation

1.4. References

1. [Autonomous Race Car | Jesper Birk | http://www.instructables.com/id/Autonomous-Race-Car]
2. [Homemade RC car | Abhishek18 | http://www.instructables.com/id/homemade-rc-car]
3. [Wiimote RC-Car Control | schoeppl.info | https://www.schoeppl.info/en/software/wiimote-rc-car-control/wiimote-rc-car-control.aspx]
4. [RC Truck to Arduino Robot with multiple sensors | immrroboto | http://www.instructables.com/id/RC-Truck-Arduino-Robot/]
5. [RF Control System For RC Vehicle Based On Arduino And nRF24L01 | mujahed1987 | http://www.instructables.com/id/RF-Control-System-For-RC-Vehicle-Based-On-Arduino]
6. [RF24L01 2.4GHz Radio/Wireless Transceivers How-To | Arduino-info @ WikiSpaces | https://arduino-info.wikispaces.com/Nrf24L01-2.4GHz-HowTo]
7. [Arduino Modules - L298N Dual H-Bridge Motor Controller| Reichenstein7| http://www.instructables.com/id/Arduino-Modules-L298N-Dual-H-Bridge-Motor-Controll/]
8. [ArduinoNunchuk GitHub | GabrielBianconi | https://github.com/GabrielBianconi/ArduinoNunchuk]
9. [Easy ultrasonic 4-pin sensor monitoring | Giedow | http://www.instructables.com/id/Easy-ultrasonic-4-pin-sensor-monitoring-hc-sr04/
10. [RF24 GitHub| Maniacbug | https://github.com/maniacbug/RF24]

1.5. Overview

This section #1 gives you a big picture what I am about to do in this project. In the following sections of this document, I will be giving information about:

- Overall description.
- Specific requirements of the project.
- Planning

2. Overall description

This section provides a more detailed overview of the system under development, including a description of the product's function and overarching constraints.

2.1. Product perspective

The final product of this project will be independent and totally self-contained, as it won't be a part of a larger system. It will retain its functionalities on its own.

2.1.1. System interfaces

The system includes two main components: a RC car and wireless remote controller. Each component would have a processor to manage and control small parts or sub-components in the component. The processors would have a piece of code or software on top to command and operate the components. Hardware and software interfaces will shortly discuss in the sub-sections below.

2.1.2. User interfaces

The system does not have a graphical user interface since it is a hardware system. A component (remote controller) would have a joystick allowing users to control car motions and directions. It also has a button to switch on/off to trigger the auto-ride mode. The other component is a RC car will receive commands from the controller to operate.

2.1.3. Hardware interfaces

There will be two microprocessors (Arduino Uno). One of them communicates with a motor controller, a distance sensor, and a radio frequency (RF) transmission adapter (receiver). The other microprocessor communicates with another RF transmission adapter (transmitter).

L298N Dual H Bridge: the motor controller controls two motors: one is a DC motor, one is a stepper motor.

HC-SR04: the ultrasonic sensor shoots ultrasonic waves and receive the coming back ultrasonic waves and calculate distance between the sensor and the obstacles.

nRF24L01: the radio communication module. One of them acts as a transmitter to transmit signals to another one (receiver). This will receive signals and send to the microprocessor.

Lithium-ion batteries: these will supply power to all the components listed above.

2.1.4. Software interfaces

During the development of my project I will use the following software and source code in order to accomplish the final objective:

- Arduino SDK
- Arduino driver for nRF24L01
- Arduino driver for Wii Nunchuk

2.1.5. Communications interfaces

The hardest part in building a RC car is how to build a communicating method between the car and the controller. There several ways to handle this such as using blue-tooth or Wi-Fi adapter. However, I found RF transmission adapters allow the components communicate and transmit in a wide range, so it becomes my selection.

2.1.6. Memory constraints

Arduino Uno (ATmega328 model) has the following memory properties:

- Flash Memory: 32 KB
- SRAM: 2 KB

- EEPROM: 1 KB

2.1.7. Operations

My project will be operable in two different modes. The first mode is called manual mode allowing users to control the car using a joystick (controller) to change the car motions and directions. The second mode is called auto-ride mode allowing the car control itself to move forward and when there is an obstacle in front of it, the car will move backward a little bit, change direction and move toward to that direction.

2.1.8. Site adaption requirements

To run the car, users simply control the controller. There is no installation requirements in this project.

2.2. Product functions

The system is composed of two components: a remote controlled car and a controller, serving one purpose of operating the car. The RC car is composed of several components: a car chassis, a DC motor, a stepper motor, a microprocessor, a motor controller, ultrasonic sensor, RF receiver, and batteries. Each component has one or many specific functions. Let's start with the car chassis.

- The chassis functions as a carrier, holding other components. The chassis comes along with four wheels and two motors. It also has an axles system. This system has two slots for DC motor and stepper motor and gears.
- The DC motor is the part make the car move. When the motor spin, the motors' movements will be transformed to the wheels with the axles system mechanism. The stepper motor play the role of change the car directions. This motor will control the front wheels turn 45 degree to left or right. Concurrently operating, the RC car will perform the same function as the actual car.
- The motor controller will control DC and stepper motors speeds and directions. And importantly, it is programmable. Based on my research, L298N Dual H Bridge is the best option for this project. It qualifies the need of being programmable and control motors' speed and directions.
- I choose Arduino Uno as the microprocessor. It performs programming car functions and other devices or components integrated into the car.
- Ultrasonic sensor performs as obstacles detector. It shoots ultrasonic waves to the front; and when the waves hit an object, they bounce and return going backward to the sensor. The sensor also has a receiver to catch those waves. The sensor processor will calculate the distance between the object and itself with the time of waves travel to the object and get back to the receiver.

- RF signal receiver is the communicating device helping the car receive signals from the controller. I use nRF24L01 for this project because it is the best-selling item on Amazon on Radio Frequency transmission device and perfectly matched my needs.
- Batteries supply power to the electronic components.

The controller is composed of three main parts: a microprocessor, the controller, and a nRF24L01 (RF transmitter).

- RF transmitter will receive commands from the controller through the microprocessor and send signals to the RF receiver on the car.
- The microprocessor is Arduino processor controlling all the attached devices (in this case is the controller and the RF transmitter.
- The controller comes along with the joystick (this will allow users to move the car in the directions they want to) and a button to switch the car between auto-ride mode and manual-ride mode. I use Wii Nunchuk and a special adapter to allow the microprocessor communicate with the nunchuck circuit board.

2.3. User characteristics

The system should not require a high level of technical knowledge to operate, and should be usable by anyone.

2.4. Constraints

The first one is the hardware limitations. Arduino Uno has small capacity of memory (see section 2.1.6) so the amount of code might be compacted to be fit in the memory. In details, adding more devices and features will consume more flash memory. Plus, 2KB SRAM might be good to just operate the car and few features. The more features, the more delay time the processor have to process and transfer data back and forth with RAM.

Moreover, my project needs parallel operations. Controlling motors and the sensors concurrently is necessary; especially with the motors. For instances, if the car wants to turn left, the DC motor will receive a signal from the microprocessor to move forward. At the time, the stepper motor will also receive another signal to turn the front wheels to the left. The combination of these two actions will make the car turn left. Based on my research, Arduino SDK is not a good option for parallel programming. However, I come up with a solution for the motors' actions: that is setting the stepper motor to turn left and delay for a little while when DC motor controls the car moving forward.

2.5. Assumptions & dependences

My project is compose of two aspects: hardware and software. The hardware mainly depends on components compatible with Arduino Uno; the software mainly depends on Arduino Software Development Kit (SDK). I assume that as long as the hardware components do not malfunction, they will work fine with programs or code written with Arduino SDK.

3. Specific requirements

This section will describe hardware and software requirements in detail as sub-sections which are interface requirements, functional requirements, performance requirements, design constraints, and software system attributes.

3.1. External interface requirements

3.1.1. Remote controlled car

- This car has to look like a normal car toy in the market.
- Its body must be flat or have box shape in order to carry electronic (and mechanical if necessary) components.
- It must have four wheels, motors, and source of power (batteries) to be able to commute.

3.1.2. Remote controller

- This controller has to have a compacted and handheld shape so that users can conveniently hold it.
- It must be operating to control the car.
- It must have enough buttons or a joystick to control the car.

3.2. Functional requirements

3.2.1. Remote controlled car

- The car must be able to move forward and backward.
- It must be able to turn left or turn right.

3.2.2. Remote controller

- The remote controller must work.
- Its buttons must function and allow the RC ca r switching mode between manual-ride and auto-ride.
- Its joystick must allow users to control the car (moving forward and backward and change directions).
- It must send signals to the car wirelessly to control the car.

3.3. Performance requirements

3.3.1. Remote controlled car

- The car must be in a regular car shape to reduce the air resistance.
- It must travel fast enough as any normal RC car in the market.
- It has to be able to travel on any types of road.

3.3.2. Remote controller

- The controller must send signals in an efficient amount of time (no delay when sending signal to the car).

3.4. Additional features requirements

3.4.1. Remote controlled car

- The car must be able to detect obstacles from distance and.
- The car must be able to avoid an obstacle after sensing it.

3.5. Design constraints

See sub-section 2.4 for details.

3.6. Software system attributes

3.6.1. Reliability

My system is mostly depending on hardware. These hardware reliability requirements are:

- Sensor data has to be true at least 99.9% of the time and data must get from the sensor at most 0.1 second.
- Communication between hardware has to be true at least 99% of the time and must be done at most 0.1 second.
- Communication between RF transmitter and receiver has to be done at most 0.1 second.

3.6.2. Usability

The product's target users mostly include anyone interested to RC car, they could be DIYers, RC car lovers, or kids with ages greater than eight. This project has very simple interface with standards of Wii Nunchuk that make the user's interaction as simple and efficient as possible.

3.6.3. Security

Since my project will not deal with sensitive private data, security is not a real concern. The only matter of interest in the security perspective is the data handled by the RF transmission devices. However, no one will interfere the radio frequency to just gain authority to control a RC car.

3.6.4. Maintainability

About hardware, though after delivering the final product as two pieces: RC car and controller, it is extremely simple to maintain the system. However, sub-components in each piece can be detached and replaced easily.

About software, maintainability of software depends on hardware. When replacing new hardware components, the components may have newer firmware so that I need to update the new code to work best with the hardware.

In addition, the batteries' life is another concern, since the battery life of a Lithium Ion battery lasts approximately one year.

3.6.5. Portability

The portability of my project in general is not well founded. The project will be created in Arduino SDK (mostly in Java), so I might not be able to port the RC car as a standalone system in different platforms. This system might also be written and work in Python environments. However, it is at best effectiveness working on Arduino environment.

4. Planning

4.1. Estimation

At the end of the semester I will plan to demonstrate my project by controlling the car manually and automatically. At the beginning of the semester, I have a lot of work which consists of researching on embedded devices and choosing hardware. After that, I will build it in three milestones: build the car with motor, build the controller and have it communicate with the car, add the ultrasonic sensor and program the car to avoid obstacles. Then, I will spend a week to test and debug the system before turn it in.

4.2. Milestones

#	Task	Scheduled Completion
1	Researching	04/25/16
2	Write proposal	04/28/16
3	Write SRS	05/09/16
4	Build the car with motor	05/16/16
5	Program the motors to move and change directions	05/23/16
6	Test the car without controller	05/30/16
7	Build the remote controller	06/06/16
8	Program the controller	06/13/16
9	Test the car with controller	06/20/16
10	Implement the ultrasonic sensor	06/27/16
11	Program auto-ride mode for the car	07/04/16
12	Test the auto-ride and obstacle avoidance	07/11/16
13	Wrap up everything and final test	07/18/16

5. Conclusion

In this document I have specified the basic requirements for my graduation project, which includes a remote controlled car and a controller to control the car motions wirelessly. Considering the difficulties when implementing components and sub-components into the system and writing algorithm to operate the system, I hope to give clear enough explanations of what is expected from the outcome of this project, all its functionalities and usages.