Software Requirements Specification

ARC - Autonomous RC

Tao Chen, Cierra Shawe, Daniel Stoyer

Version 0.1

November 2, 2016

Contents

1 INTRODUCTION

This section provides a scope description and overview of everything included in this SRS document. The purpose of this document and a list of abbreviations and definitions are provided.

1.1 Purpose

The purpose of this document is to give a detailed description of the requirements for the ”Autonomous RC System” or ARCS. The purpose and declaration for the development of ARCS will be explained. This document will also explain system constraints, interface decisions, and interactions with other external applications and hardware. This document is primarily intended to be a customer proposal for approval and a development team reference for the first version of the system.

1.2 Scope

ARCS is a software-hardware interface designed to retrofit RC cars for autonomous operation, using commodity hardware. The software and hardware specifications should be available free to download and modify at the users will.

Users should be able to purchase and install the specified hardware, and implement a version that can autonomously navigate to a given destination, or a within a pre- defined space.

The software will need to be installed on specific hardware, and flashed onto the main processing unit, along with control software on a computer or tablet. Hardware that we expect to need is a base station that includes a transceiver, an RC retrofitted with the main processing unit, a transceiver to send and receive information, a controller to send signals to sensors and actuation devices (motors, servos, etc...) and a vision system to aide in navigation.

ARCS will be expected to be able to receive input from a user base station, and react within the environment based on a destination that the system receives. It should also be able to navigate to the destination without user intervention, as fast as possible.

1.3 Definitions, Acronyms, and Abbreviations

1) **IMUs:** Inertial measurement unit. Used to measure acceleration, angular acceleration, and orientation of the vehicle.
2) **Operator/User:** The person who is giving commands such as destination to the system.
3) **Protocol:** defines the data format to be transferred.
4) **Telemetry Data:** data that contains the status information of the vehicle, such as speed, temperature, location, battery, etc.
5) **emergencies:** Emergencies include: The vehicle drifts off course significantly.
   Vehicle flips upside down.
   On-board components fall off.
   Etc.

6) **Visual Unit:** Visual components that provide image streams to the primary computer. The primary computer will extract information from the images, such as road condition, obstacles, and depth.
7) **Actuators:** Motors and servos.

1.4 References

[1] IEEE Software Engineering Standards Committee, ?IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications?, October 20, 1998.

1.5 Overview

The remainder of this document includes three sections and the appendixes.

Section two provides an overview of system functions and intersystem interaction. System constrains and assumptions are also addressed.

Section three provides the requirement specification in detailed terms and system interface descriptions. Different specification techniques are used in order to specify requirements for different audiences.

Section four priorities requirements and includes motivation for the chose prioritization and discusses why other methods were not chosen.

Appendixes at the end of the document include results of the requirement prioritization and a release plan based on the requirments.

2 Overall Description

This section provides a system overview. The system will be explained in its context to create a better understanding of how the system interacts with other systems and introduce the basic system functionality. This section will also describe what types of stakeholders will use the system and what functionality is available for each type. Lastly, constraints and assumptions for the system will be presented.

2.1 Product Perspective

The ARC system (ARCS) will be designed to integrate into an RC car using commodity hardware, and open to anyone who is interested in using it. This makes ARCS a component of a larger system, namely the RC car.

ARCS will consist of three parts:

1) Base-station used for user interaction
2) Hardware attached to RC car to be able to connect to base-station and operate the vehicle
3) Software implementation for control and communication between hardware and software

Fig. 1. Block diagram of hardware flow

In order for the user to interact with the vehicle, commands will be sent via some form of receiver to the car. This will need to be done via a base-station that has software able of providing a way to communicate with the receiver, which then transmits data to the receiver on the car, which is then handled by the on-board computer. The control flow is described in figure 1.

2.1.1 System interfaces

There are a total of 5 system interfaces where the system can communicate with the outside world.

1) Sensors: Sensors will have a two-way communication with a secondary computer unit, where filtering and smoothing will happen before reliable data will be passed to the primary computer. The programs that reside in the secondary unit will utilize various methods to generate reliable results based on the raw data. At start-up, there will be a script executed by the system to correctly configure and calibrate each sensor. Sensors may include: battery sensor, temperature sensor, GPS, speed sensor, and IMUs.
2) Radio: This is the portal of the system where operator/user can monitor the status of the vehicle. Different protocols will be implemented for telemetry data, which will be displayed to the operator/user. This portal also allows operator/user to take control over the computer in case of emergencies.
3) Visual unit: This is the interface where the visual unit can pass streams of images to the system.
4) Actuator: The system issues commands to the motors and servos via this interface.
5) User interface: This interface is a different interface than the radio interface even though they both allow humans

to interact with the system. The user interface will be disable after the vehicle starts maneuvering. This interface allows user to input operation modes and desired destinations into the system.

2.1.2 User interfaces

The user interface is a simple, concise GUI. Anyone who knows how to operate a mouse will interact with the interface with no problem. A map makes it easier for users to pin point destinations and view the current location of the vehicle. An error messages will be generated if destination is out of range, meaning that with the onboard battery the vehicle won't be able reach the desire destination. If the direct distance between the vehicle and the station exceeds the maximum radio range, an error message will be generated as well.

With proper training (in less than 30 minutes), one can understand all the indicators of the system to know the status of the vehicle.

The GUI is a single window/page arrangement. A large portion of the window is dedicated to the map. A little section at the bottom is the indicators. Error messages will directly appear on the map with alarm to warn the user.

2.1.3 Hardware Interfaces

As this is a research project, we are currently unsure of what hardware will be used.

2.1.4 Software Interfaces

- We require three operating systems. One for a remote PC for user input, one for the primary computer for on-car data analysis, and one for the secondary computer for on-car control.
- We require a user interface to be able to give the car destination commands.
- We require software that takes input from hardware, such as video cameras, and analyze the data
- Sensor analysis software needs to be able to read data from sensors, such IMUs and accelerometers, and generate usable information to pass on to
- Path finding software needs to be able to integrate and analyze data from the other software products and determine a path to follow in real time.
- The primary and secondary computers will need to talk with one another. The primary computer will need to receive data from the secondary computer, analyze the data and send corresponding commands back to the secondary computer. The secondary computer will need to receive commands from the primary computer, and send data to the primary computer.
- A software interface will required on the secondary computer to convert commands received from the computational computer to instructions usable by pre-existing RC car on-board controller.
- We will require separate interfaces between the secondary computer and the visual/spacial sensors, GPS, speed and accelerometer sensors, and the IMU.

2.1.5 Communications Interfaces

- Radio communication will required to be able to send commands to the car and to be able to receive feedback from the car. The radio frequency will be in 27 MHz or 49 MHz range. We need software that allows us to send and receive radio signals to and from the car.
- LAN communication will be used to transmit data between the on-car computers. We will utilize exist network protocols to perform the transmissions.
- A software interface will be required to send the commands the from the secondary controller computer to the pre-existing RC car on-board controller.

2.1.6 Memory constraints

2.1.7 Operations

2.1.8 Site Adaptation Requirements

2.2 Product Functions

2.3 User Characteristics

2.4 Constraints

2.5 Assumptions and Dependencies

2.6 Apportionment of Requirements

3 Specific Requirements

3.1 External Interfaces

3.2 Functions

3.3 Performance Requirements

3.4 Design Constraints

3.4.1 Standards Compliance

3.5 Software System Attributes

3.5.1 Reliability

3.5.2 Organizing the Specific Requirements

4 Other Nonfunctional Requirements

4.1 Performance Requirements

4.2 Safety Requirements

5 Other Requirements

6 Index

7 Appendix

7.1 Appendix A: Glossary

7.2 Appendix B: Analysis Models

7.3 Appendix C: To Be Determined List