|  |  |  |
| --- | --- | --- |
| Software Documentation | April 14  2016 | |
| Illustration and explanation of how the software operates and how to apply it in the solution. | | Sitraffic smartTL Connect | |

Contents

[Requirements 3](#_Toc448841649)

[Architecturally Significant Requirements 3](#_Toc448841650)

[Business Requirements 4](#_Toc448841651)

[Functional Requirements 5](#_Toc448841652)

[Quality-of-Service Requirements 6](#_Toc448841653)

[Implementation Requirements 7](#_Toc448841654)

[Architecture and Design 8](#_Toc448841655)

[Architecture Viewpoints 8](#_Toc448841656)

[Architecture Description 9](#_Toc448841657)

[End User Manual 10](#_Toc448841658)

[Tutorial 10](#_Toc448841659)

[Thematic 11](#_Toc448841660)

[Reference 12](#_Toc448841661)

[Business Model 13](#_Toc448841662)

[Technical Documentation 14](#_Toc448841663)

[Technical Documentation Embedded in Source Code 14](#_Toc448841664)

# Requirements

Attributes, capabilities, characteristics and qualities the system.

## Architecturally Significant Requirements

The purpose of this software is to manage the connections from several remote costume controller boards to a user focused web interface, in real time. To achieve this effect we have broken the software in four main components of development. From the lower level of complexity there is the controller board’s programming that should check and manage the state of the controller and send that information via communications over IP. Next we have the SimpleServer, an application that receives the information from *n* controller boards and puts it in a data configuration to be processed, or sends the controller board a message with the status it should be (in case the user changes it from the interface’s end). The converter verifies and compares the status received and stored to know how to process that information, with the ability to detect if a controller is not changing state, if the controller is not receiving messages, or any other communication error, and inform the user through the web interface. The communication exchange with the interface uses an open source SOAP (Simple Object Access Protocol) based solution named OCIT (Open Communication Interface for Traffic).

With this description an architecture was develop to be able to do these functions with some implicit compromises. The most limiting trade-off points in the converter application are the lack of adaptability of the messages sent and received as well as the need for the converter application to be pre-programmed with a specific number of field devices in order to be able to show the status correctly. In the field device we have a constraining limitation that influences the architecture of the solution, the need to pre-program each device to correctly represent its data to the server. A non-negotiable breaking point is the delay of receiving the messages, it means there is a chance that the state represented on the interface might not be the current state of the device. This delay breaks the real time monitoring of the devices but has no impact in performance because the needed delays are very small (2 to 10 seconds between board and converter and 5 to 10 seconds from the converter to smartGuard). The board’s software is also a fixed running detection cycle, because of the simple communication interface the commands it can process are limited, in the current state, there is no way to remotely change the configuration, name or fix a bug in the software running. A future achievement is to have a very solid remote update software running.

This software handles all the possible errors of communications, will always try to send a message in a set interval of time with exception of a unpredicted loss of power, due to battery concerns the software needs to send a message with priority before going offline. There is no interaction with the end user and the configuration is done at a source code level. The converter will be running in a Linux based system, the board’s software will be running in microprocessor.

## 

## Business Requirements

This project comes from a contractual requirement to implement remote monitoring on older controllers. When looking for a way to do this we found that several other cities had the same problem. They had old controllers, no central system and no money to replace all the controllers with new ones with remote access, this situation was a factor that influenced sales and the expansion of our products.

Having this solution as possibility opens new options for our clients, but we need to keep in mind that this project is only a temporary patch while we are introducing our controllers and software. We are planning on using it no longer than 3 years.

Planning for this life time we had to make sure to have a solid product that could endure the time and provide with all the features required. This was achieved by making simplicity and adaptability our guidelines.

Because the project was not originally thought out to support long distance maintenance and with the newly discovered possibility of providing the service to other countries, there is a new factor to consider, while having several possible solutions, can still be a misaligned expectation by the clients.

Plans for the future include the generalization of this solution with the goal of making it an adaptable platform for the integration of new devices into smartGuard, like low level add-on to the software. Keeping this solution under development can expand its capabilities, improve its quality and make it state of the art integration tool.

## Functional Requirements

## 

## Quality-of-Service Requirements

## 

## Implementation Requirements

# Architecture and Design

Overview of software. Includes relations to an environment and construction principles used in design of the software components.

## Architecture Viewpoints

## Architecture Description

Decision based

# End User Manual

Practical instructions for the end-user, system administrators and/or support staff.

## Tutorial

## Thematic

## Reference

# Business Model

How to market the product and analysis of the market demand.

# Technical Documentation

Source code, algorithms, interfaces, and APIs.

## Technical Documentation Embedded in Source Code