Understanding Rc Car Demo

Manual

Tritech

Table of Contents

[1 Document Information 3](#_Toc377114019)

[1.1 Version History 3](#_Toc377114020)

[1.2 References 3](#_Toc377114021)

[1.3 Definitions and Abbreviations 3](#_Toc377114022)

[2 Introduction 4](#_Toc377114023)

[2.1 Application fundamentals 4](#_Toc377114024)

[2.2 Two way communication 4](#_Toc377114025)

[2.3 Extra functionality 4](#_Toc377114026)

[2.4 Example of messages 4](#_Toc377114027)

[2.5 Mailing list 5](#_Toc377114028)

[3 Compiling and deployment 6](#_Toc377114029)

[3.1 Compiling 6](#_Toc377114030)

[3.2 Deployment 6](#_Toc377114031)

[4 Sending outgoing messages 7](#_Toc377114032)

[5 Future plans 8](#_Toc377114033)

# Document Information

This document describes the need and use of the MQTT to rest application

## Version History

| Version | Date | Author | Action |
| --- | --- | --- | --- |
| PA1 | 2014-05-09 | Fredrik Olsson | First version |

## References

| Ref | Title | Document id | Version |
| --- | --- | --- | --- |
| 1 |  |  |  |

## Definitions and Abbreviations

| Expression | Description |
| --- | --- |
| Pi | Raspberry Pi |

# Introduction

This document gives a short introduction to the “Rc Car demo”. How the two programs work and what parts are used.

## Application fundamentals

There are two applications used for the car demo, a server program and a client program. The server is located on the Pi, and the client can be located on any Windows computer. In the future there might be a third program, acting as a spectator. The spectator will see everything that the client sees, but it can’t interact with the server.

## Car Server

The car server is located on the raspberry pi, and is written in c/c++. It was originally written in pure C. But to add support for third party libraries the program was rewritten in c++. It’s not really objective oriented but has c++ syntax. The server has two threads, one input thread (the main thread). And one output thread. The input thread waits for a connection from the client (tcp), once this connection has been established the main thread will start the output thread. The main thread listens for commands from the client, the commands can be in the form of inputs, configurations, and camera variables. The inputs are used to set different servos and speed controllers to a desired value, basically the inputs controls the car. However the server might limit these inputs, so the car will not crash. Configurations are used to determine one which channels each servo is placed. For example the server needs to know on which channel the speed controller is, if it was to stop the car.

The output thread sends data to the client (UDP). The data can be either the camera or sensor values. The camera is captured using PiCam library. This data is in raw RGBA format and thus the image size is huge. Once the output thread has capture the camera image the image is compressed into a jpeg file. Then the image is sent to the client. The output thread also sends sensor values, such as GPS data, Voltage readings, current readings of different servos etc.

## Car Client

The client is located on any Windows computer, and is written in C#. The program uses MonoGame (previously XNA) to render textures and strings to the screen. Monogame is also used to capture keyboard and gamepad / wheel inputs. The program has two threads, one input and one output. The input thread, also known as the read thread reads camera images and sensor data from the server. When the client gets a message from the server the message the input thread lets the main thread know new data has arrived, the data is then parsed and handled accordingly.

The output thread, also known as the main thread does the hard work. The main thread handles: program updates, rendering of the screen, checking for new controller input, sending data to the server, and much more. Mono game works like this. At the start of the program the program calls a function “Init” where all initialization is done. After this the program loads external assets such as fonts and images. Then the program reaches the main loop, the main loop does two things, Update and Render. This cycle will repeat until the program is stopped. The update function checks for input and if any input is associated with a servo then that data is sent to the server. The update function also parses the data from the input thread. The GPS data is sent to google and google returns a map of the current location of the car. The render function renders everything to the screen, but before that everything is rendered to a texture. This is done so that in the future the texture could be sent to a third program, the spectator. The texture is then rendered to the screen.

## Parts used

Short list of what parts are used. Will be updated with more information later.

PCA9685 , 16 channel PWM board. For servo management.

MCP3204, 12-bit ADC.

Raspberry pi Revision B.

Nokia5110 LCD.

BR-355, GPS

Raspberry pi camera rev 1.3

Uni-Reg voltage divider.

Usb hub.

Relay switcher (unknown)