ROTA

Remotely Operated Transportation Apperatus

Rijksuniversiteit Groningen

Bastiaan van Houttum - S3146391
Oliver Strik - S3100693

November 1st 2017

Abstract

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Contents

1 Problem Description

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

2 Problem Analysis

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

3 Design

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

4 Program Code

The code of ROTA is split into three seperate programs.

- The controller - Written in Java. https://github.com/Kranex/ROTA-Controller
- The Server - Written in C. https://github.com/Kranex/ROTA-Server
- The Inperpreter - Written in Arduino C. https://github.com/Kranex/ROTA-Interpreter

In the interest of saving trees the 700+ lines of code are not included in this document, if you wish to find them please use the links above to access them. A description of each program however, is provided below.

4.1 The Controller

The controller was written in Java with the help of the Android Studio IDE. The main function of the controller is to create two floating joysticks to allow users to submit input to ROTA. It is essentually an xbox or playstation controller but without any buttons. One joystick is created on the left and one on the right, they will place themselves under the first touch on their half of the screen and the centre of the joystick will remain at that point until the finger is lifted again. This allows for some degree of personal adjustability; not everyone holds their phone the same way, and not all phones are the same size or shape, so the joysticks should not be fixed to one place on the screen.

The controller also tries to connect to a TCP socket on the IP address 192.168.12.1 port 55455, the Raspberry Pi was set up to maintain a consistent gateway address so that the controller would never have to ask for the IP address of the server, the server was also configured to only use port 55455. Once a connection has been created the controller will send an update packet every few miliseconds. This packet consists of 5 bytes {a,x1,y1,x2,y2}.

1. 'a' is an undefined byte. It can be used for anything; it was added in case some kind of status update or emergency stop button was introduced.
2. x1 and y1 are the influences of the left joystick. Their value will be between -127 to 127. (0,0) would mean the joystick is centered. (0,127) would mean the joystick was centered in the x axis but pushed all the way up on the y axis.
3. x2 and y2 are just the same as x1 and y1 but are for the right joystick.

The packet does not require any leading or ending byte as TCP guarentees that all packets will arrive and they will all arrive in the correct order. Thus, on the server, if the Server reads 5 bytes it will always read {a,x1,y1,x2,y2} if a packet exists to be read.

4.2 The Server

The server is by far the most convoluted program of the three, which is thanks to it having been programmed in C. It does only one thing: foward all packets received from the controller, straight to the Arduino over serial. It does however add 127 to each of the signed bytes as the serial connection seemed to really dislike the signed bytes. 127 is then immediately subtracted from the byte when it gets to the Arduino. This is all handled in about 5 lines of code, however there are nearly 300 lines surrounding it most of which is simply initialisation for the TCP server and serial connection. On the Raspberry Pi, the server is added to the list of programs to start at boot with systemd, thus no login is required, the Raspberry Pi must only be booted for the server to be created.

4.3 The Interpreter

The interpreter, for lack of a better name, is the program running on the arduino which 'interprets' the x and y influences recieved from the server and translates them into motor speed, servo angle or what ever else we happen to connect to ROTA. This job could have been done with the Raspberry Pi alone, however we would then have had the constraints of few GPIO to communicate with, as well as the lower voltage of the GPIO (3.3V instead of 5V) to deal with. Thus we decided to use the arduino as the step between the Raspberry Pi and the extra periperipherals.

After initialising the serial connections and any global variables, the arduino then reads a single byte from the serial buffer and acts apon it. It keeps count of which particular byte it is reading and then, as we have it currently set up, if it is the left y, uses it to set the motor speed and direction or if it is the right x, use it to set the servo angle. The particular stick or axis that controls each of these can easily be changed, the only reason we have it set up as so is that it made practical sense to split the controls of throttle and steering on to seperate sticks. As for why the sticks are cappable of moving in both axis when only one is required, it doesn't impare their function, it was easier to make two identical joysticks over two unique ones and most importantly, if we had the time and the servos to do it, we would have like to have added a moveable camera. Unfortunately, this was not the case, but now at least it features the ability to have such a camera with the addition of only two if statements.

5 Tests and Results

Throughout the development of ROTA there have been things that needed to be tested. Afterall there are three devices that could fail. The first thing that was programmed was the app, after fixing many issues that showed up throughout development, three worthy problems showed.

1. If the app is disconnected from the server, by loosing the wifi or something similar, it would never reconnect.

2. If the screen focus shifts, causing the app to leave fullscreen, it never reverts. this does not impact useability, however it is annoying.

3. If the user leaves the app, Android will 'suspend' it. This is great, however, it reaks havoc with the server connection. Very strange things can happen, the best you can hope for is that the app crashes, at worst, the app tries to reistablish the connection and you end up with 5 or 6 rogue threads spamming TCP Requests. To prevent this, I set the app up to destroy itself completely when minimised, this looses no functionality however as the app would have to be restarted when it comes out of suspension anyway, it's just easier to do that from the beginning.

There were never really any issues with the server, par some minor confusion caused by C's lack of a 'byte' type and chars being unsigned on some systems but not others. This was easily fixed by declaring all chars being used as bytes as signed.

Another problem came with the arduino, it would not accept signed bytes across serial. this meant I either needed to revise my transmission method or send unsigned bytes instead. as all bytes were between -127 and 127 adding 127 before sending and subtracting it after solved this problem.

The biggest problem of them all was how to supply power the Arduino, the Raspberry Pi and the motor shield. In the end we decided to power the Raspberry Pi and the motor shield with a large, two USB powerbank. The Raspberry Pi would then power the arduino. This seemed to work on the bench when the arduino was plugged into the PC for tuning the servo and running light tests on the motor. However the Raspberry Pi could not deliver the current to power the servo and the logic side of the motor controller, this would crash the Raspberry Pi and all hell would break loose. To counter this issue we gave the Arduino it's very own 9v battery. While it isn't perfect, it does seem to solve the problem. Idealy a dedicated Lithium Polymer battery would have been preferred, a 2200mah 7.4v battery would be small and light enough to fit on the frame, however they're not exactly cheap and we would need something to step the voltage down to 5 volts for the Raspberry Pi.

After the power problem was sorted, the car worked with almost perfectly, it can pickup quite some speed, every now and then there is a slight delay in the controls however not enough to cause serious issue. The only minor faults are that the servo is rather poorly attached to the steering allowing a small margin of free movement. this means that the steering is never exactly centered and the car needs constant steering adjustments to stay on course. We also noticed that the front wheels would lock on the frame when turning. This was because all the weight had crushed the suspension. To fix this the wheel arches were removed.

6 Evaluation