Team Rocket: Missile Launcher

1 Inhaltsverzeichnis

2 Grundgedanken

2.1 Ziel

Unser Ziel ist es den Dream Cheeky Missile Launcher live über eine Android-App steuern zu können.

2.2 Vorgehen

Folgende 3 Geräte brauchen wir:

- Missile Launcher
- Raspberry Pi
- Android Gerät

Der Missile Launcher wird mit dem Raspberry Pi per USB verbunden.

Vom Raspberry Pi soll aus soll man nun den Rocket Launcher steuern können. Dies wird mit NodeJS bewerkstelligt.

Der nächste Schritt ist es nun das Raspberry Pi zu einem Access Point zu konfigurieren, damit man eine Schnittstelle zwischen Smartphone und Raspberry Pi hat.

Auf dem Raspberry Pi erstellt man dann einen Socket.io Server, der folgende Events kennt:

- Up
- Down
- Left
- Right
- Stop
- Fire

Auf dem Client (Smartphone) läuft nun eine App. Diese App kann die obigen 6 Events senden.

Somit kann man den Missile Launcher über das Smartphone steuern.

3 Dream Cheeky Missile Launcher

Der Dream Cheeky Missile Launcher steht uns fertig zur Verfügung und an ihm muss nichts mehr konfiguriert werden.

3.1 Spezielles

Der Missile Launcher hat folgende Funktionen:

- Hoch
- Runter
- Links
- Rechts
- Feuern

Dabei gibt es zu beachten, dass der Rocket Launcher **nur jeweils eine** der obigen Funktionen auf einmal ausführen kann. Das heisst der Missile Launcher kann sich nicht diagonal bewegen, oder drehen und feuern.

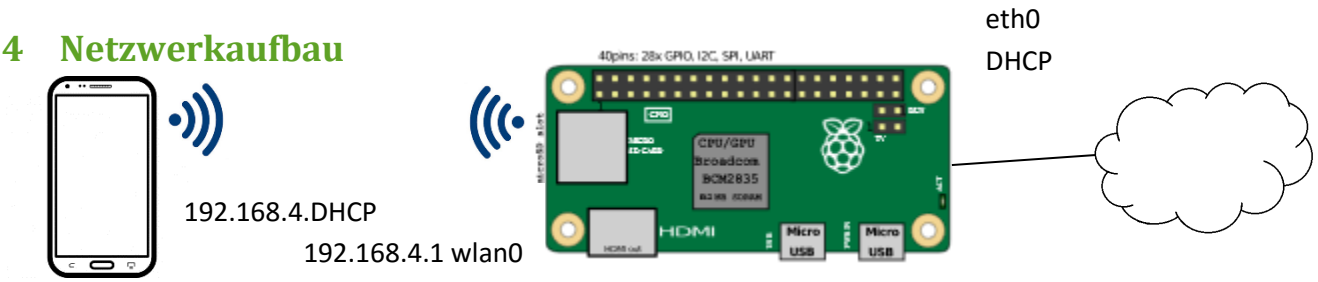
Ausserdem kann sich der Rocket Launcher sich nicht mehr als 360° bewegen.

Der Missile Launcher kommt also in alle 4 Richtungen irgendwann an einen Anschlag.

Die Zeit, in der der Missile Launcher feuert beträgt ca. 3-4 Sekunden.

Obige Faktoren wurden in der Entwicklung miteinberechnet.

4 Netzwerkaufbau



5 Raspberry Pi

Das Raspberry Pi erledigt die Arbeit als Server in unserem Projekt. Auf dem Raspberry Pi müssen einige Konfigurationen durchgeführt werden.

Zuerst jedoch muss man das „headless“ Raspberry Pi ansteuern. Dazu gibt es folgende Möglichkeiten:

- Maus, Tastatur und Bildschirm direkt anschliessen
- Mithilfe der IP-Adresse Zugriff über Netzwerkschnittstelle mit Putty

5.1 Installation Nodejs

Auf dem Raspberry haben wir NodeJS und NPM über NVM installiert. (Siehe: https://github.com/creationix/nvm). Zuerst hatten wird dabei Probleme, da NVM die binären Dateien ins richtige Verzeichnis heruntergeladen hat. Jedoch konnten wird das Problem mit einer Anleitung auf https://www.digitalocean.com/community/tutorials/how-to-install-node-js-with-nvm-node-version-manager-on-a-vps lösen.

5.2 Dream Cheeky Treiber installieren

Um über USB mit dem Missile Launcher zu kommunizieren, wollten wir zuerst ein Node-Modul herunterladen. Jedoch waren alle möglichen Module für die neuste Version des Rocket Launchers entwickelt worden und funktionierten somit nicht mit unserem. Das Problem konnten wir mit einer eigenen, umgeschriebenen Version des „thunder-connector“ Moduls lösen. Der Quellcode dieses Moduls ist hier (https://github.com/TimPietrusky/thunder-connector) zu finden. Bevor wir jedoch die Abhängigkeiten dieses Moduls erfolgreich installieren konnten, mussten wir noch „libusb“ auf dem Raspberry installieren.

5.3 Accesspoint auf dem Raspberry PI

Damit wir mit dem Smartphone auf den Socket.IO-Server des Raspberry's zugreifen können, mussten wir unser Raspberry zuerst in einen vollfunktionsfähigen Accesspoint umwandeln. Das Ganze war eigentlich relativ einfach, da wir genau dieser (https://menzerath.eu/artikel/raspberry-pi-als-wlan-access-point-nutzen/) Anleitung folgen konnten.

6 Smartphone

Das Smartphone fungiert als Client in unserem Projekt.

Remote Control auf dem Smartphone (Beispielfoto)

Über das Smartphone erfolgt die Steuerung des Missile Launchers.

6.1 Layout der App

Grobes Layout:

6.2 Installation Ionic Framework & Grundbefehle

Installation Cordova:

```
$ sudo npm install -g cordova
```

Installation Ionic:

```
$ sudo npm install -g ionic
```

Projekt erstellen:

```
$ ionic start Project blank
```

Plattform hinzufügen, damit die App auf Android funktioniert:

```
$ ionic platform add ios
```

App testen (Smartphone muss dazu angeschlossen sein):

```
$ ionic build ios
```

6.3 Konfiguration der App

Die App ist so konzipiert, dass es auf das Drücken reagiert und erst ein stopp sendet, wenn man wieder loslässt. Diese Funktion wurde mittels Javascript auf die einzelnen divs gelegt. Per JavaScript werden die Befehle über das WLAN an den Socket.io Server auf dem RaspberryPi gesendet.

6.4 Testen der Smartphone App

Um das Smartphone App testen zu können, musste man sich über WLAN mit dem RaspberryPi verbinden. Am Anfang stellte sich ein Problem heraus, welches wir nicht beachtet hatten. Wir hatten den Quellcode noch nicht für das Smartphone optimiert, sondern nur für den Computer und so funktionierte es nur stockend. Jedoch sahen wir, dass die Befehle über das WLAN auf den Socket.io-Server des RaspberryPi gesendet wurden. Nach einer kleinen Anpassung des Quellcodes, wo wir den Code für das Smartphone optimierten, funktionierte alles ohne Fehler und es gab kein stockende Bewegungen des Rocket Launchers mehr.

7 Schlusswort

Auch wenn wir immer wieder auf unvorhergesehene Probleme trafen, konnten wir dieses Projekt pünktlich beenden. Wir können nun auf unser gut vollendetes Projekt zurückschauen und uns am Rocket Launcher, welchen wir über das App steuern, erfreuen.