Team Rocket: Missile Launcher





Inhaltsverzeichnis

[Grundgedanken 3](#_Toc454529388)

[Ziel 3](#_Toc454529389)

[Vorgehen 3](#_Toc454529390)

[Dream Cheeky Missile Launcher 4](#_Toc454529391)

[Spezielles 4](#_Toc454529392)

[Raspberry Pi 5](#_Toc454529393)

[Installation Nodejs 5](#_Toc454529394)

[Dream Cheeky Treiber installieren (NPM Modul) 5](#_Toc454529395)

[Konfiguration des Treibers 5](#_Toc454529396)

[Netzwerkaufbau 5](#_Toc454529397)

[Netzwerkeinstellungen auf dem Raspberry Pi (Access-Point) 5](#_Toc454529398)

[Konfiguration des Socket.io Servers 5](#_Toc454529399)

[Smartphone 6](#_Toc454529400)

[Layout der App 6](#_Toc454529401)

[Installation Ionic Framework & Grundbefehle 6](#_Toc454529402)

[Konfiguration der App 7](#_Toc454529403)

[Testen der Smartphone App 7](#_Toc454529404)

[Schlusswort 7](#_Toc454529405)

# Grundgedanken

## Ziel

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

## Vorgehen

Folgende 3 Geräte brauchen wir:

* Missile Launcher
* Raspberry Pi
* Android Gerät

Der Missile Launcher wird mit dem Raspberry Pi verbunden. Das Raspberry Pi steuert den Missile Launcher über einen Treiber über die USB-Schnittstelle an.

Vom Raspberry Pi soll aus soll man nun den Rocket Launcher steuern können.

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.

Ein Access Point mit einem virtuellen Netzwerkadapter ist die ideale Lösung da:

* Raspberry Pi und Smartphone direkt im gleichen Netzwerk sind
* Der Access Point verschlüsselt Daten sendet und empfängt
* Man auf keine anderen Netzwerke angewiesen ist

Auf dem Raspberry Pi läuft nun ein 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.

# Dream Cheeky Missile Launcher

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



## 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.

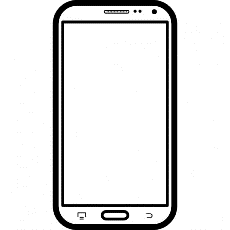
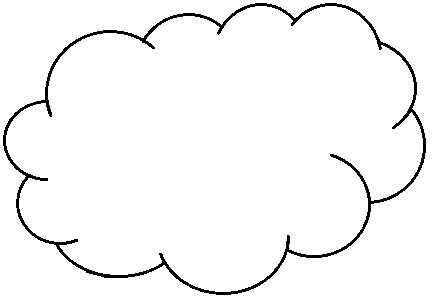
Der Missile Launcher kommt also in alle 4 Richtungen irgendwann an einen Anschlag.

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

Obige Faktoren wurden in der Entwicklung miteinberechnet.

# Netzwerkaufbau

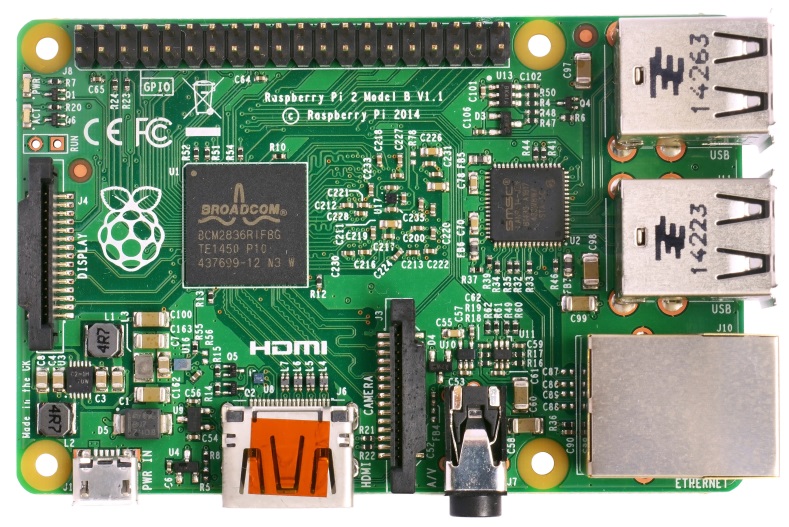
eth1 DHCP



192.168.4.DHCP  
 192.168..4.1 et0

# 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 (oder MAC-Adresse und einigen Netzwerktools) Zugriff über Netzwerkschnittstelle mit Putty

## Installation Nodejs

$ sudo apt-get install nodejs

NPM Installation

## Dream Cheeky Treiber installieren (NPM Modul)

### Konfiguration des Treibers

## Netzwerkeinstellungen auf dem Raspberry Pi (Access-Point)

## Konfiguration des Socket.io Servers

# Smartphone

Das Smartphone fungiert als Client in unserem Projekt.



Remote Control auf dem Smartphone (Beispielfoto)

Über das Smartphone erfolgt die Steuerung des Missile Launchers.

## Layout der App

Grobes Layout:

Feuer

UjjUp

## 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

## 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.

## 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.

# 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.