HTL | MÖSSINGERSTRASSE

HÖHERE TECHNISCHE BUNDESLEHRANSTALT
KLAGENFURT, MÖSSINGERSTRASSE

ABTEILUNG ELEKTRONIK UND TECHNISCHE INFORMATIK

PROJEKTDOKUMENTATION

RETROPI

03.02.2018

Inhaltsverzeichnis

1. Allgemein

Der Input für den RetroPi haben wir über eine Website (virtueller Controller) und einem selbst gebauten WLAN-Controller realisiert. Der Server läuft auf einem RaspberryPi zusammen mit der Benutzeroberfläche EmulationStation [1]. Die Datenübertragung von Client zu Server erfolgt mittels Websockets. Diese eignen sich für dieses Projekt gut eine persistente Verbindung zwischen Client und Server besteht und beide jederzeit Daten senden können. Noch ein Vorteil ist, da Websockets in vielen Programmiersprachen implementiert sind. Für die Server-Seite und den Hardware-Controller verwenden wir eine Open Source API [2] [3]. Die Webseite kommuniziert über JavaScript mit dem Server. Beim Hardware-Controller verwenden wir einen ESP-8266 der über die Arduino IDE programmiert wird. Bei EmulationStation gibt es für viele verschiedene Konsolen Emulatoren, wir haben uns dabei für die NES (Nintendo Entertainment System) entschieden.

2. Hinweis

Da es bei diesem Projekt sehr viel Source Code gibt, habe ich mich entschieden diesen nicht in der Dokumentation auszuarbeiten. Wenn Interesse zum Code besteht, besteht die Möglichkeit in diesen in unserem Github Repository einzusehen: https://github.com/quentinwendegass/RetroPi

3. Server

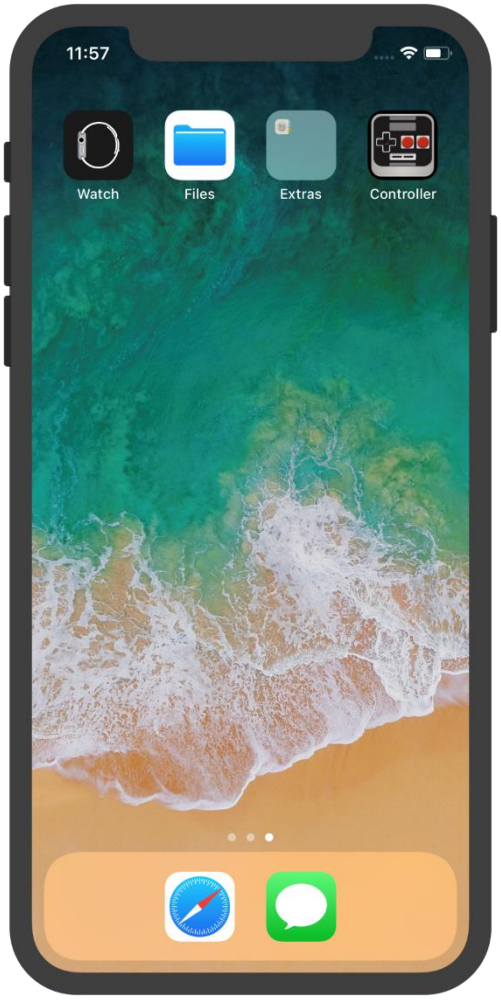
Den Websocket-Server und den Http-Server, haben wir auf dem RaspberryPi mit Python geschrieben. Wir haben für dieses Projekt Python gewählt, wegen den vorliegenden Vorteilen:

- Python-Code gilt als besonders gut lesbar, was mit dem vorgegebenen strukturierten Programmierstil zusammenhängt.
- Python-Programme sind in der Regel um einiges kürzer als in traditionellen Sprachen (wie z.B. in der Sprache Java) geschriebene Programme, die dasselbe leisten.
- Es wird eine umfangreiche Standard-Bibliothek von Modulen (z.B math, turtle) mitgeliefert.
- Python ist eine objektorientierte Programmiersprache, die dank eines hochgradig portablen Interpreters auf vielen Plattformen zur Verfügung steht.
- Python eignet sich für fast alle Anwendungsprobleme
- Python ist eine Skriptsprache und kann in Anwendersoftware eingebunden werden.

Die Funktion des Servers, ist es, den Tastendruck der vom Client geschickt wird zu verarbeiten. Für dieses Problem haben wir das Modul uinput [4] verwendet, welches Tastendrücke simulieren kann, die dann in den Spielen erkannt werden. Für Websockets haben wir eine Open Source API [2] gefunden, die unseren Ansprüchen gerecht ist. Sie wurde geschrieben von Johan Hanssen Seferidis und ist auf auf github.com frei zugänglich.

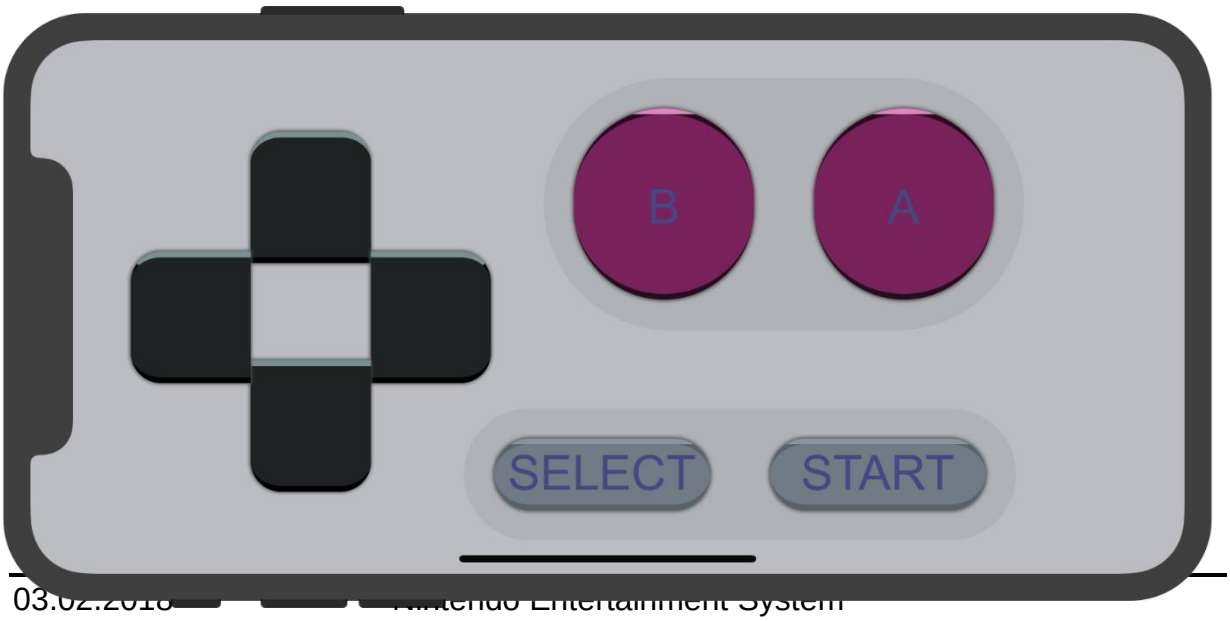
4. Virtueller-Controller

Der virtuelle Controller ist eine Http-Seite, die mittels JavaScript die Tastendrücke zum Server schickt. Da diese Art von Controller eigentlich nur für Smartphones gedacht wurde, ist dieser als Webanwendung implementiert. D.h. man kann den Controller als auf dem Home-Bildschirm speichern und obwohl es eigentlich eine Webseite ist, ist das Erscheinungsbild wie bei einer App.



Webapp am Home-Bildschirm

Status Menü



Eigentlicher NES-Controller

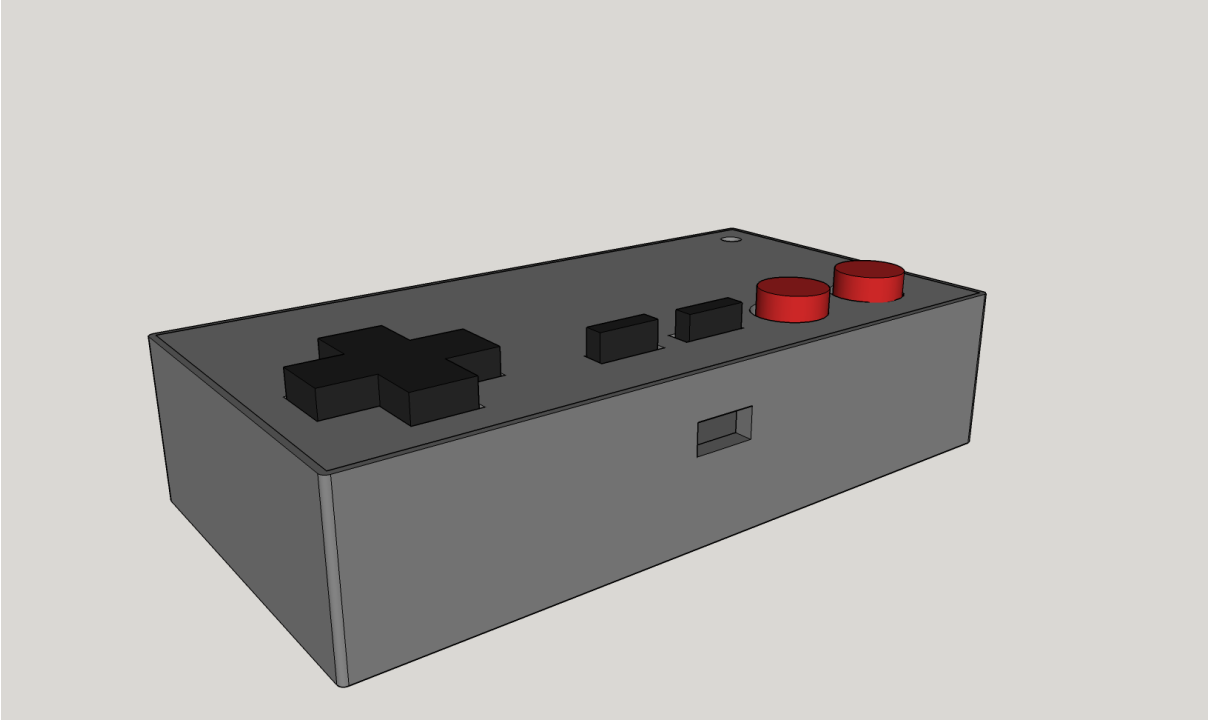
Die Webanwendung wurde auf verschiedenen Geräten (IOS und Android) getestet. Im Status-Menü kann man sich mit dem Server verbinden und trennen, sowohl auch den Verbindungsstatus und (wenn verbunden) den zugewiesenen Spieler ablesen. Im Querformat wird dann der Controller sichtbar. Bei jedem Tastendruck auf der Webseite werden zwei Datenpakete an den Server gesendet. Einmal für das drücken der Taste und wieder für das loslassen.

5. Hardware-Controller

Die Verbindung zu dem Server wird mittels WLAN hergestellt. Der ESP-8266 ist sinnvoll, da schon ein WLAN-Modul integriert ist und wegen seiner kleinen Bauform. Programmiert wurde er über die Arduino IDE. Es ist auch möglich diesen mit der Skriptsprache Lua zu programmieren, da ich aber mit der Arduino IDE schon vertraut bin, habe ich mich für diese entschieden. Beim starten des Controllers, verbindet er sich automatisch mit dem WLAN-Router. Darauf wird der Websocket-Client gestartet und mit dem Server verbunden. Für den Websocket verwenden wir eine Library von Chris Venter[3], die für das Arduino und den ESP-8266 geschrieben wurde. Die Taster werden an die GPIO-Pins angeschlossen. Sie werden über Interrupts ausgelesen und anschließend die Informationen an den Server geschickt. Wichtig ist das die Taster über eine Hardwareschaltung entprellt werden.

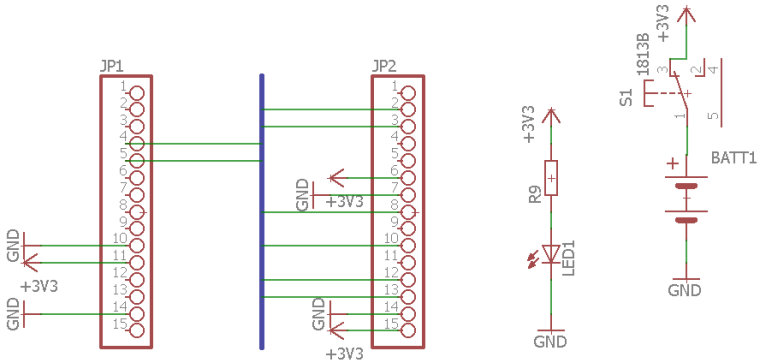
5.1. 3D Modell

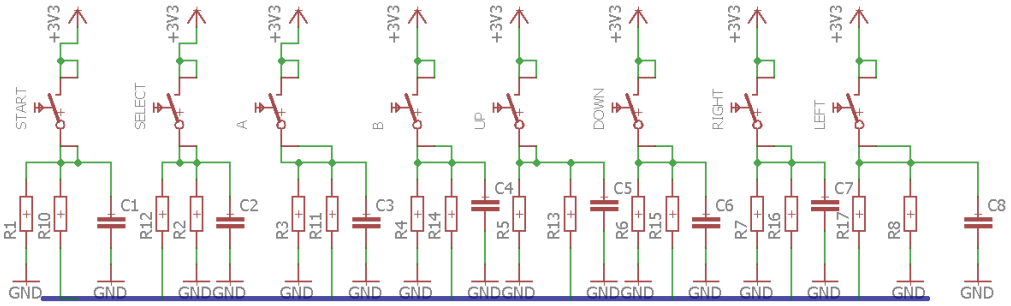
SketchUp ermöglicht einfaches Modellieren von 3D Objekten. Durch diese Software konnte ein NES Controller mit den optimalen Maßen die für die Hardware benötigt wird entworfen werden.

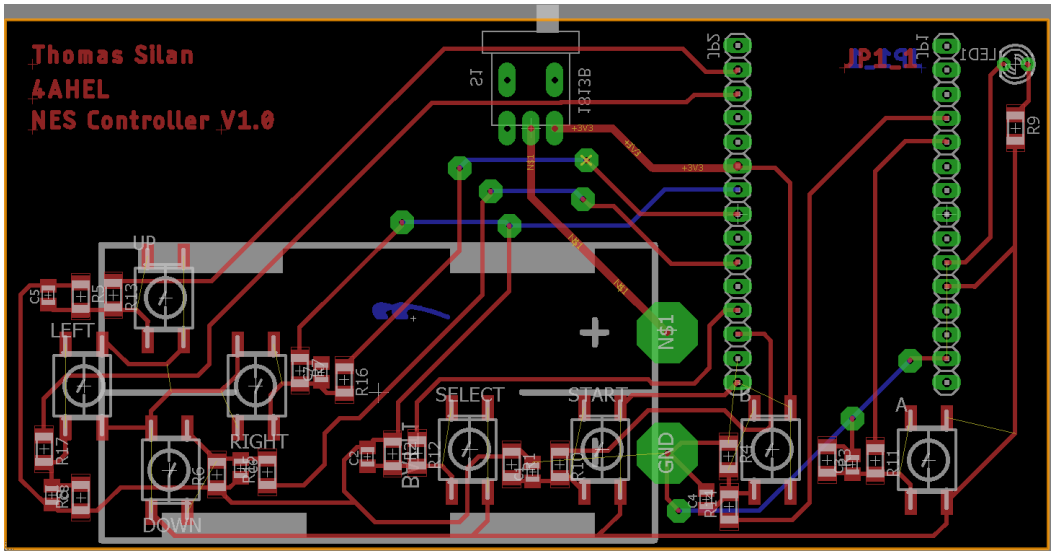


5.2. Schaltplan und Board Layout

Der ESP wird über Batterien mit Spannung versorgt und kann durch den Schalter ein bzw. Aus geschalten werden. Die Kondensatoren dienen zur Entprellung der Taster und werden durch die Widerstände auf Down gezogen, wenn der Taster nicht gedrückt wird. Aufgrund des Aufbaus der Taster können die können die offenen Verbindungen vernachlässigt werden.


JP1
JP2
GND
+3V3
R9
LED1
S1
1813B
BATT1


START
SELECT
A
B
UP
DOWN
RIGHT
LEFT
R1
R10
C1
R12
R2
C2
R3
R11
C3
R4
R14
C4
R5
R13
C5
R6
R15
C6
R7
R16
C7
R17
R8
C8
GND
+3V3


Thomas Silan
4AHEL
NES Controller V1.0
UP
LEFT
RIGHT
DOWN
SELECT
START
B
A
GND
N$1
R9

6. Weblinks

[1] https://retropie.org.uk

[2] https://github.com/Pithikos/python-websocket-server

[3] https://github.com/Links2004/arduinoWebSockets

[4] https://www.kernel.org/doc/html/v4.12/input/uinput.html