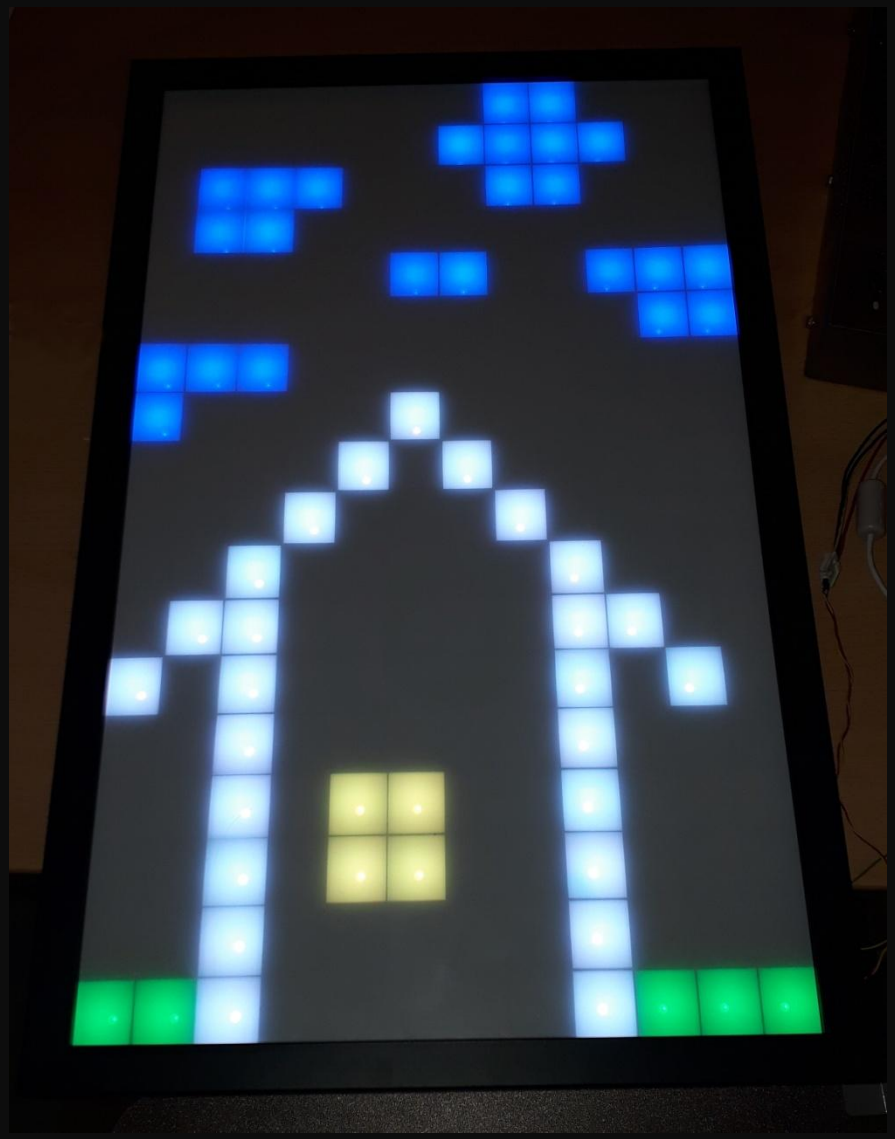
Pixel Wall

by enny



Inhalt

Mechanischer Aufbau
Wahl der Programmiersprache
Simulation

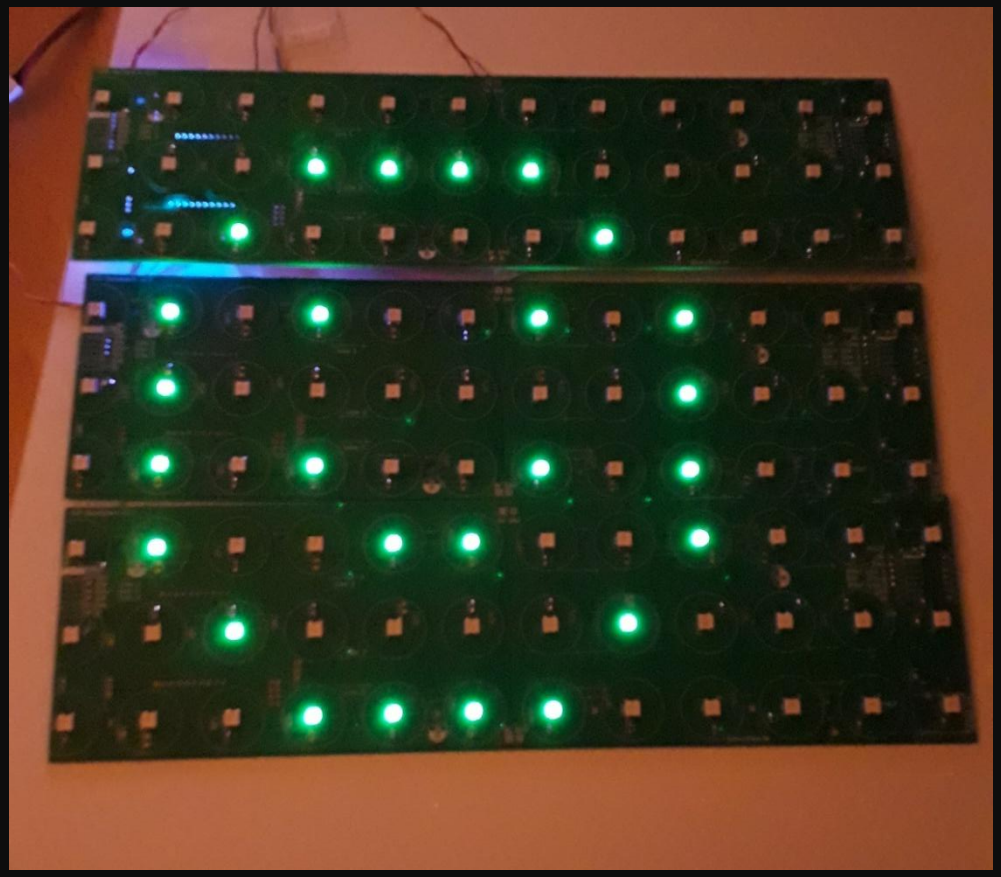
Programmaufbau
- Apps
- Webserver
- Websocket

Vorführung / Spielen

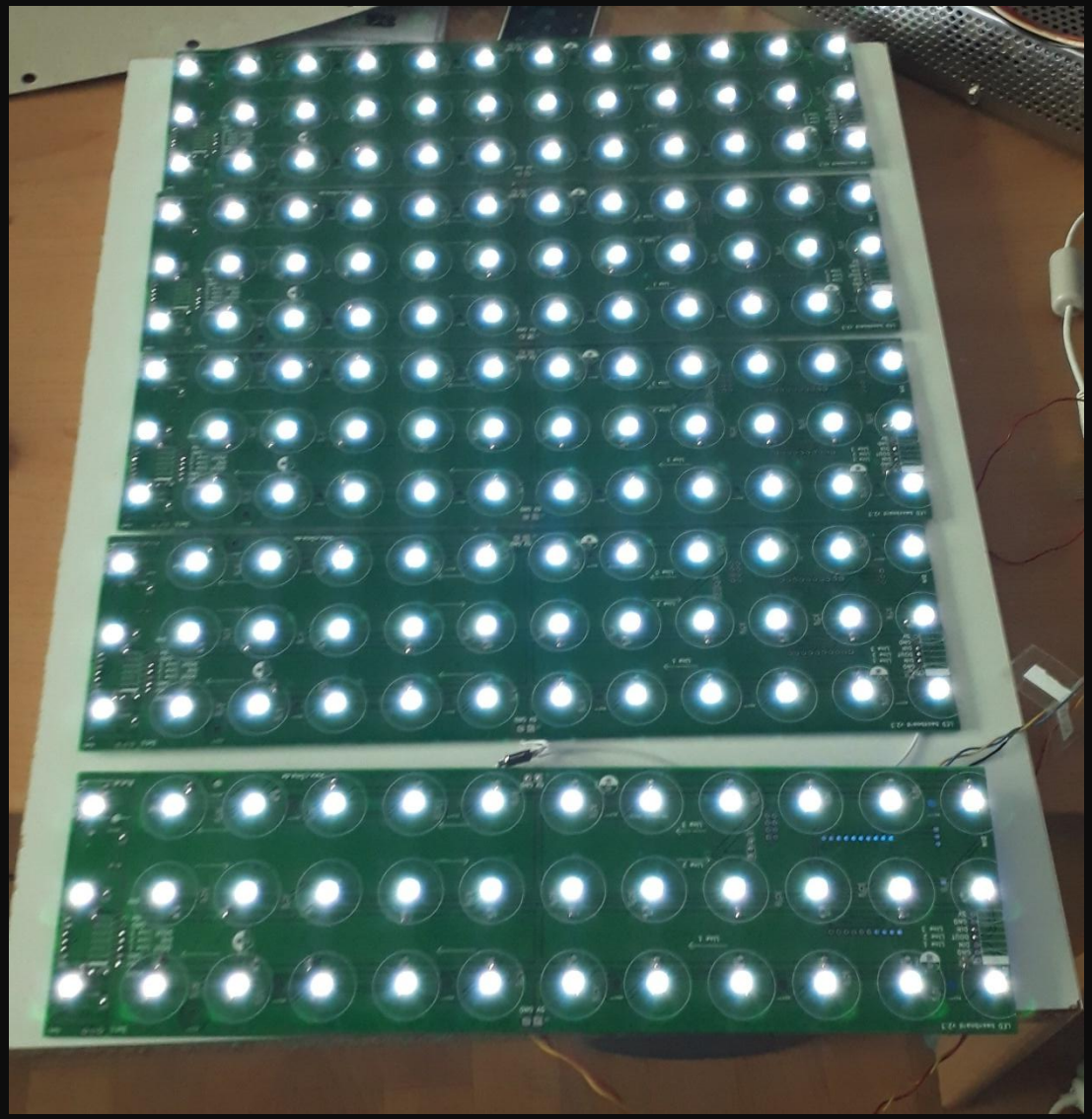
Mechanik

Mechanischer Aufbau

Mechanischer Aufbau



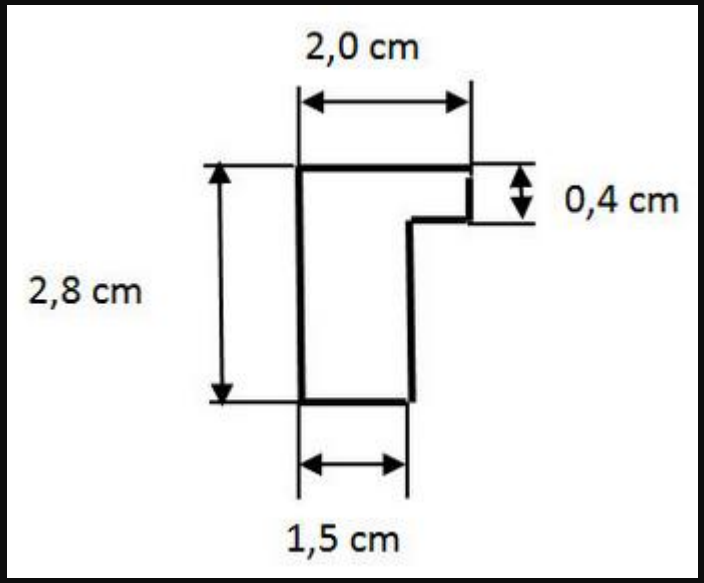
Mechanischer Aufbau



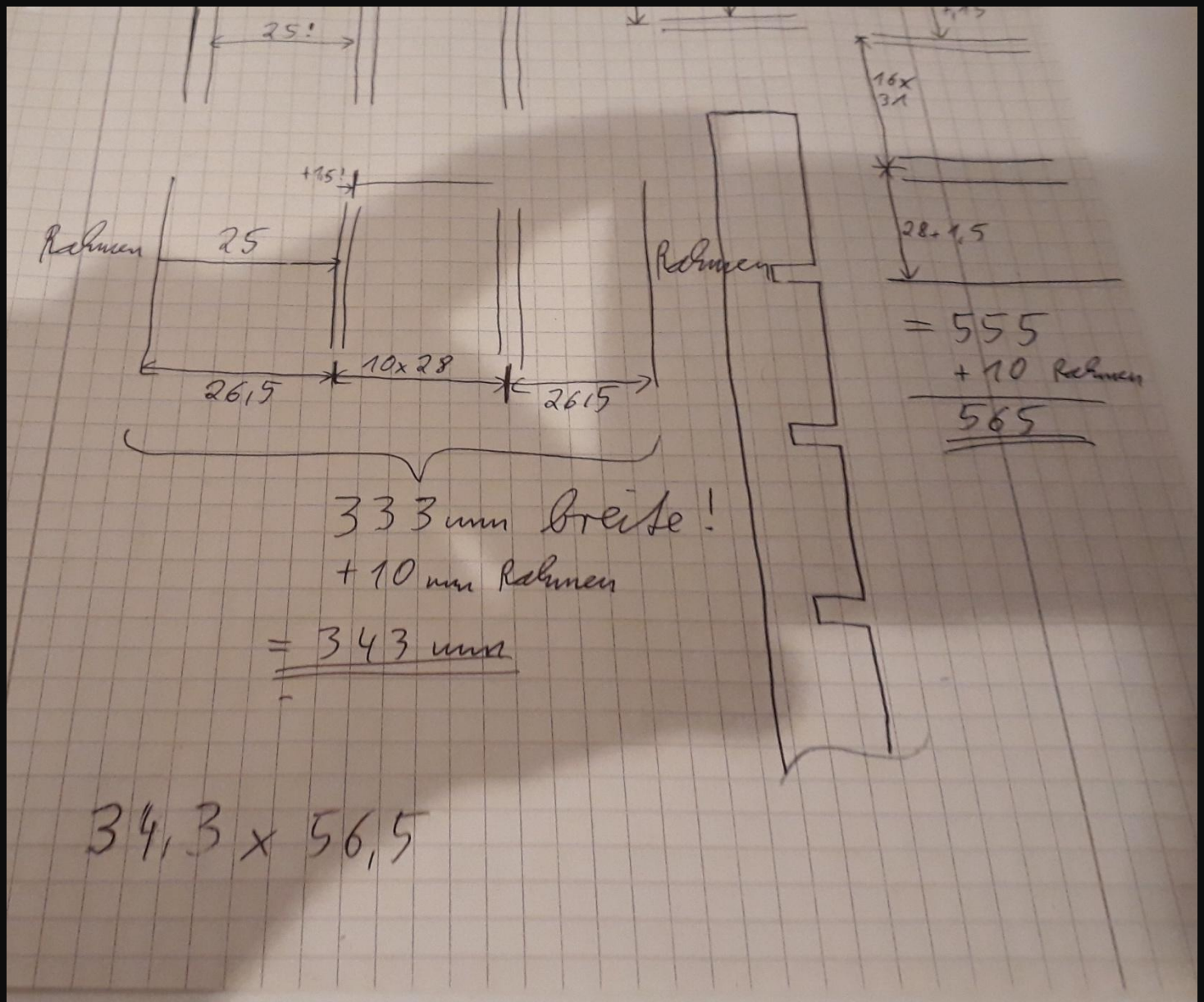
Mechanischer Aufbau

Bilderrahmen bestellt

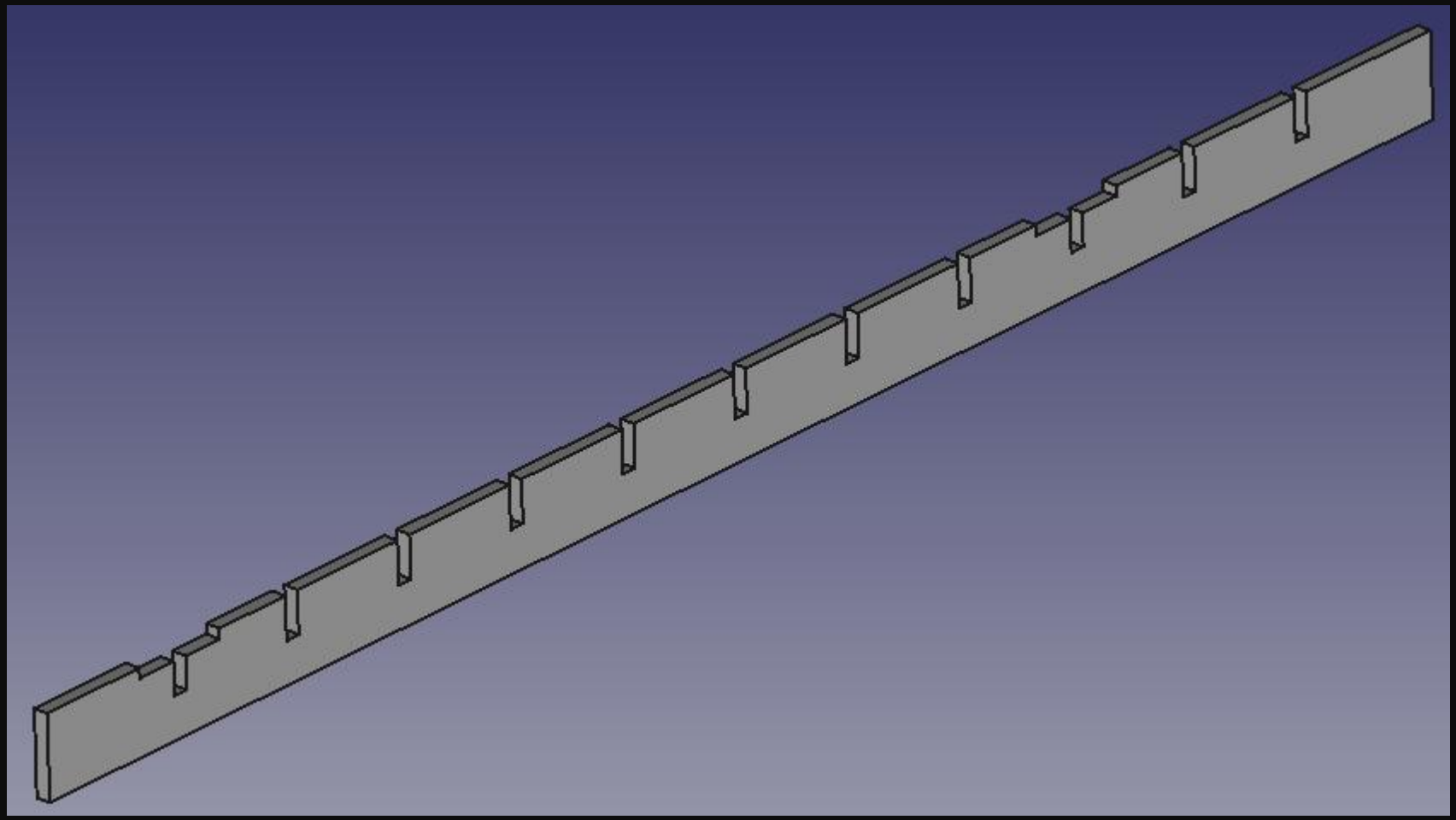
allesrahmen.de
AVE-86010-03-SZ
343 x 565 mm



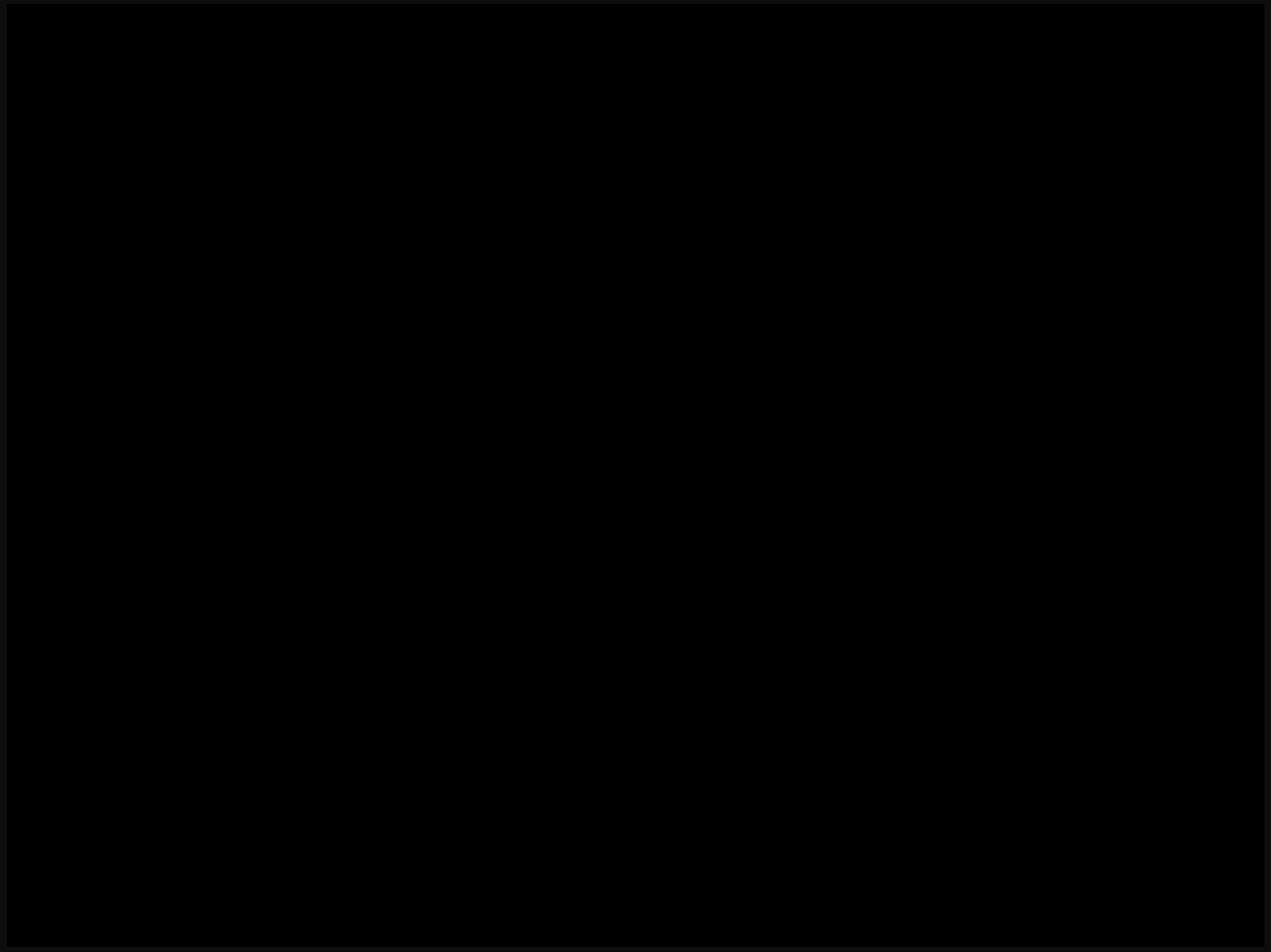
Mechanischer Aufbau



Mechanischer Aufbau

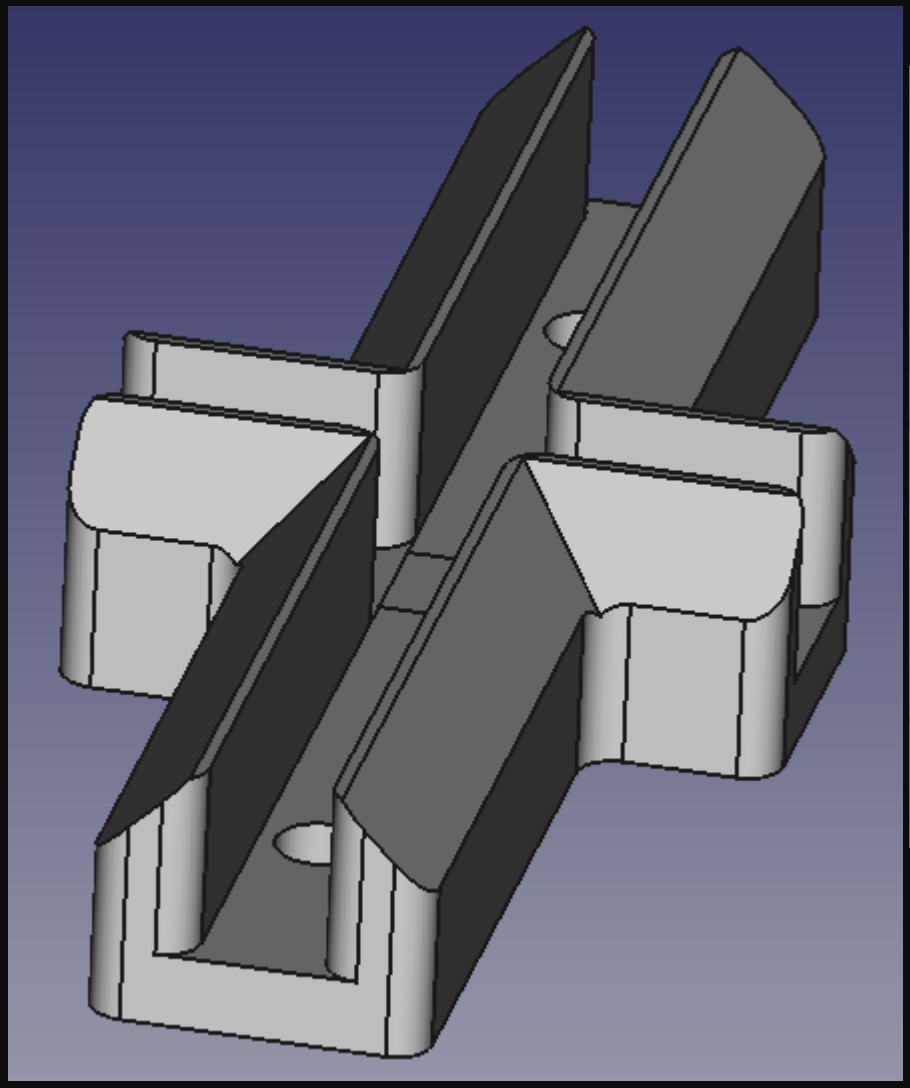


Mechanischer Aufbau



Mechanischer Aufbau

Mechanischer Aufbau

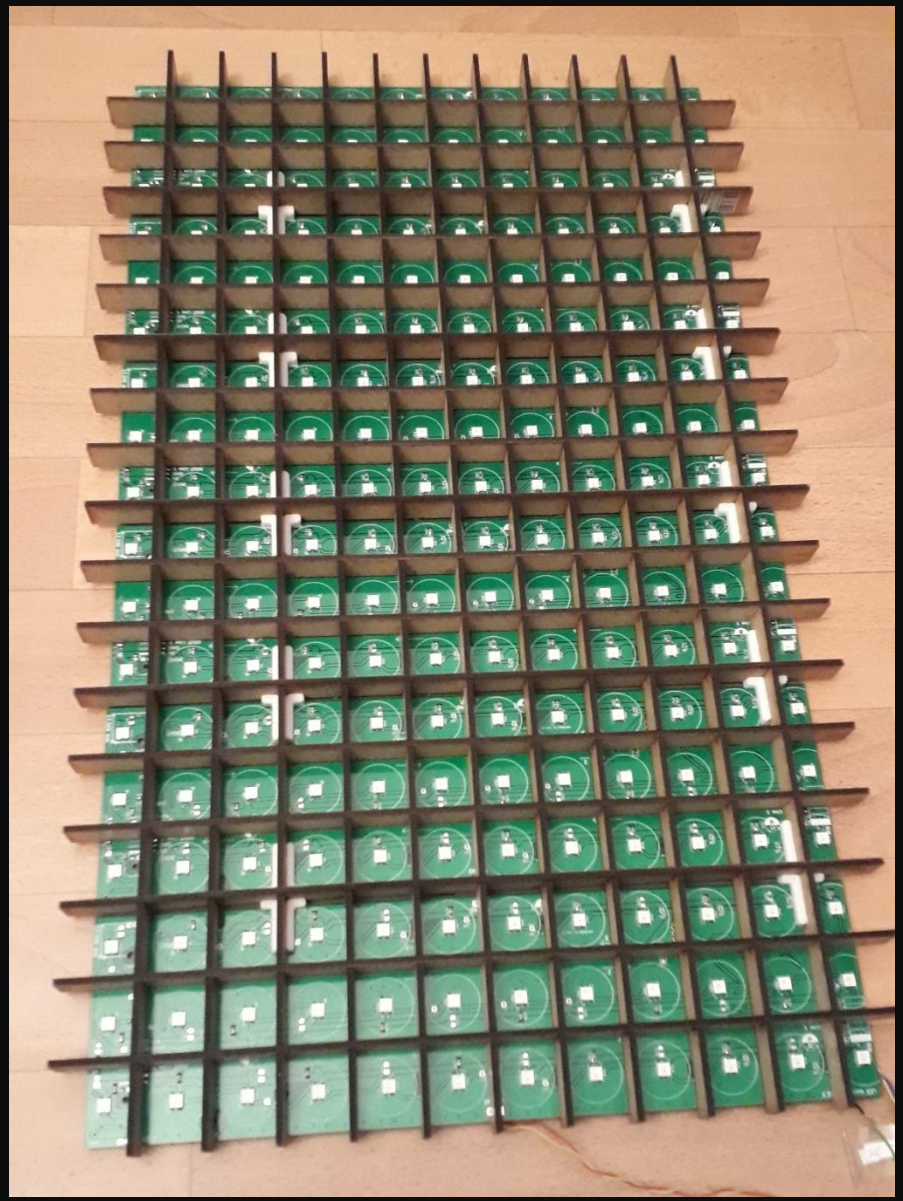


Befestigung der Leisten auf der Platine

Schrauben:
2 x 8 mm

+ Unterlegscheibe M2

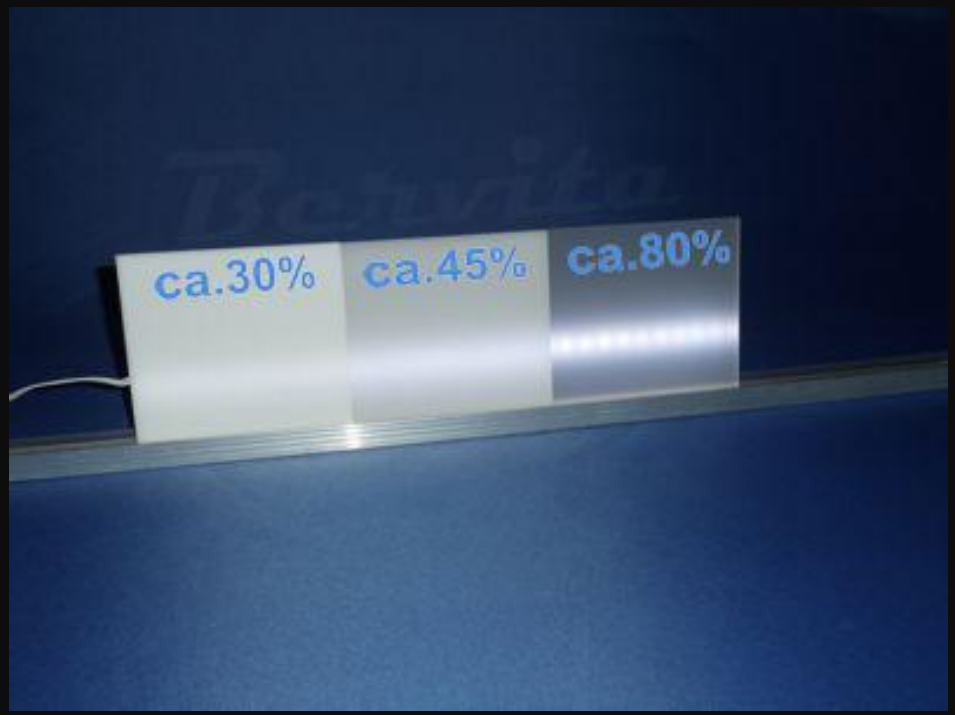
Mechanischer Aufbau



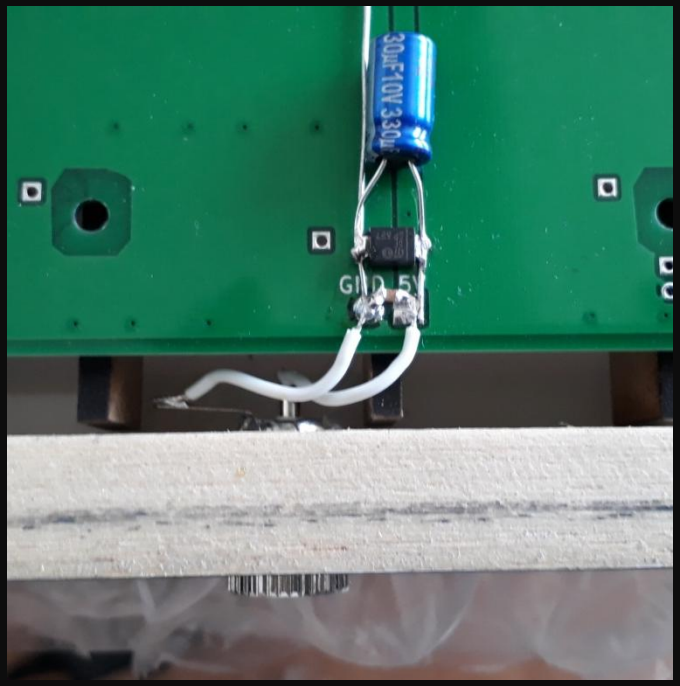
Mechanischer Aufbau

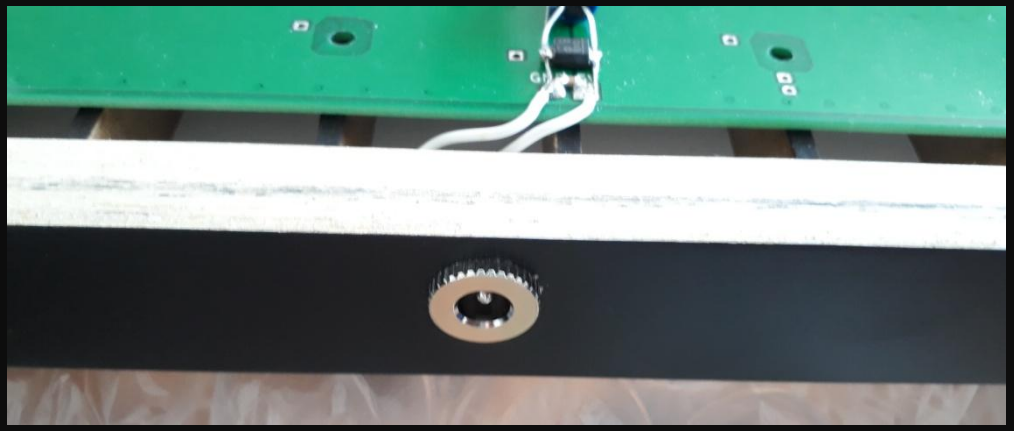
Plexiglas Platte bestellt

bervita.de
LD45%
567 x 344 x 2 mm



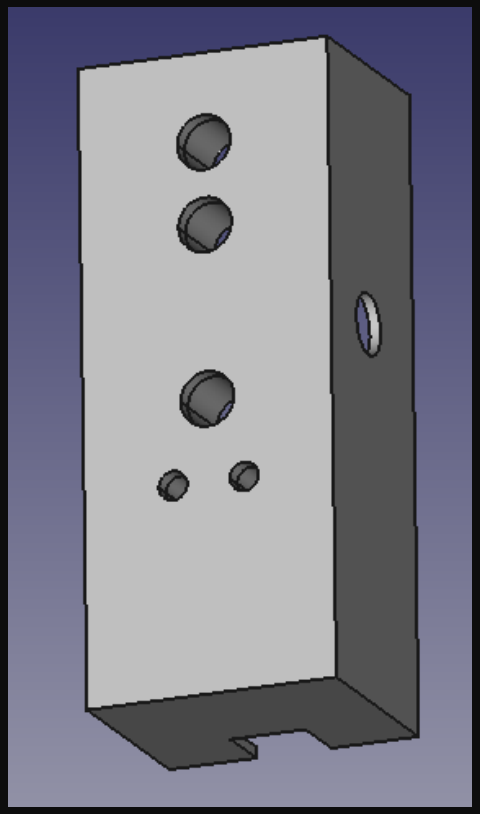
Spannungsversorgung

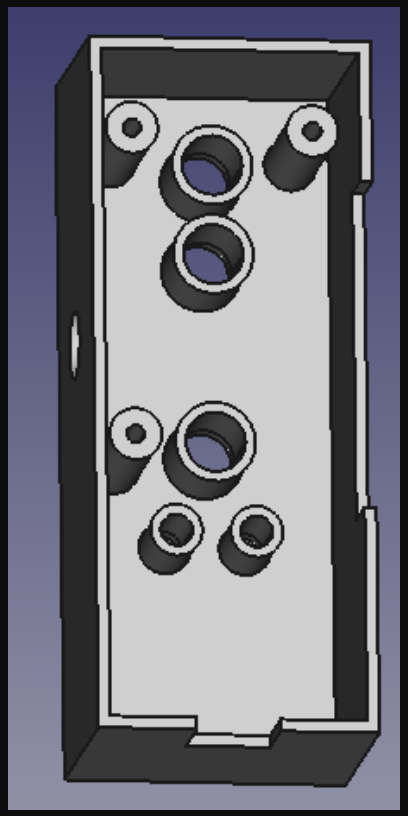




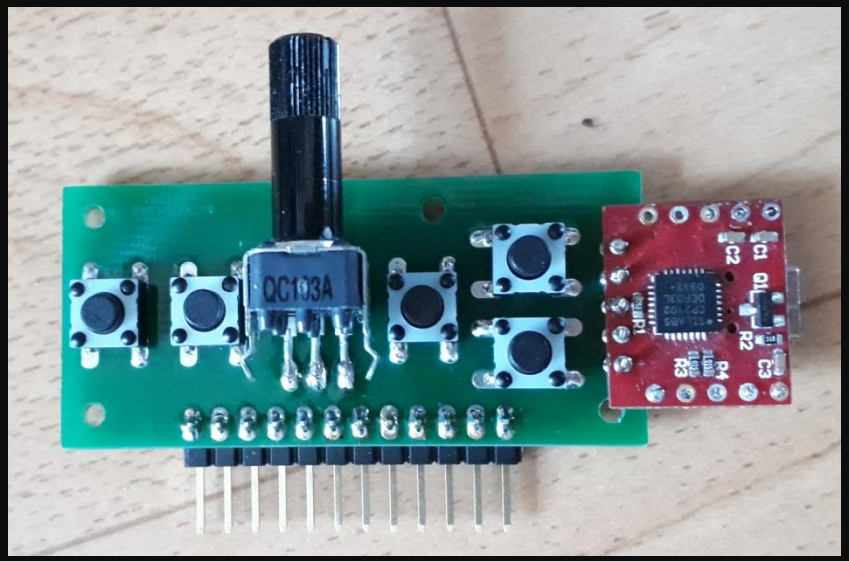
Bedieneinheit

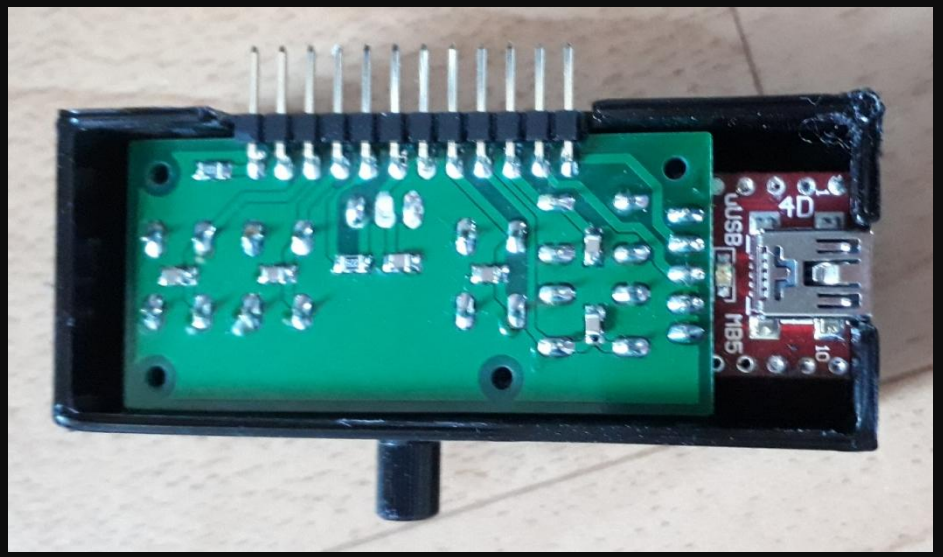
- Poti für Helligkeitssteuerung
- Tastaturelement (3x Bedienung + Flash + Reset)
- USB Anschluss zum flashen
- abnehmbar zwecks Optik





Bedieneinheit





Bedieneinheit

Up

Down

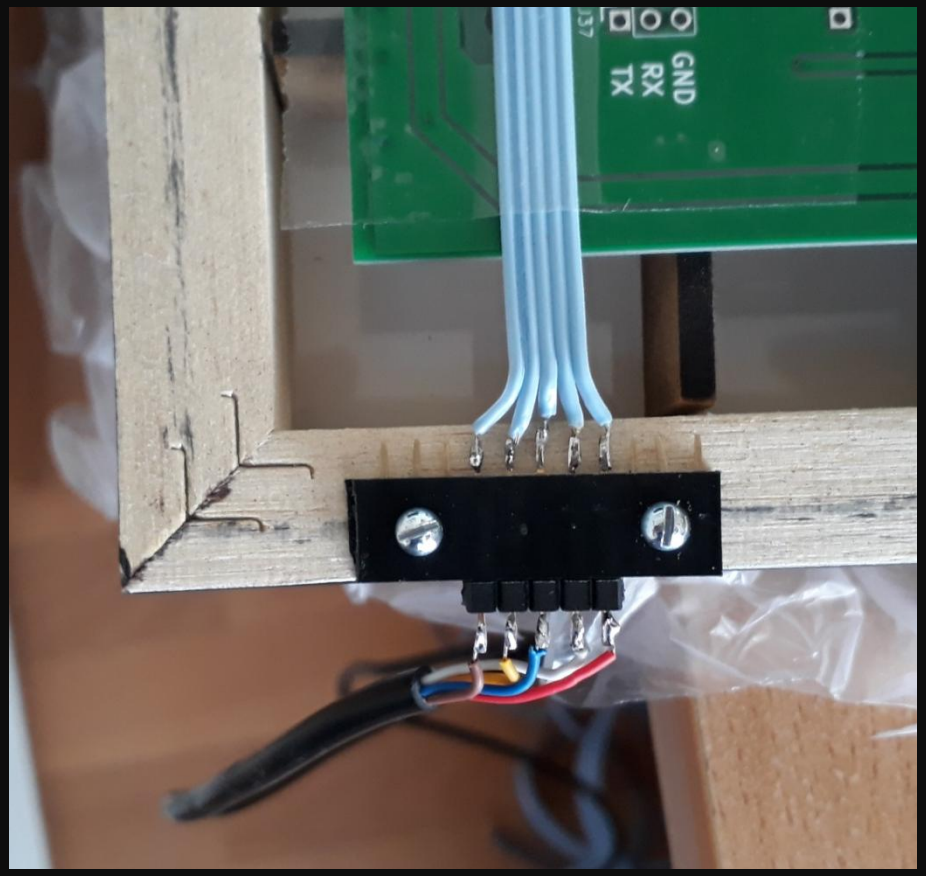
App select

Flash

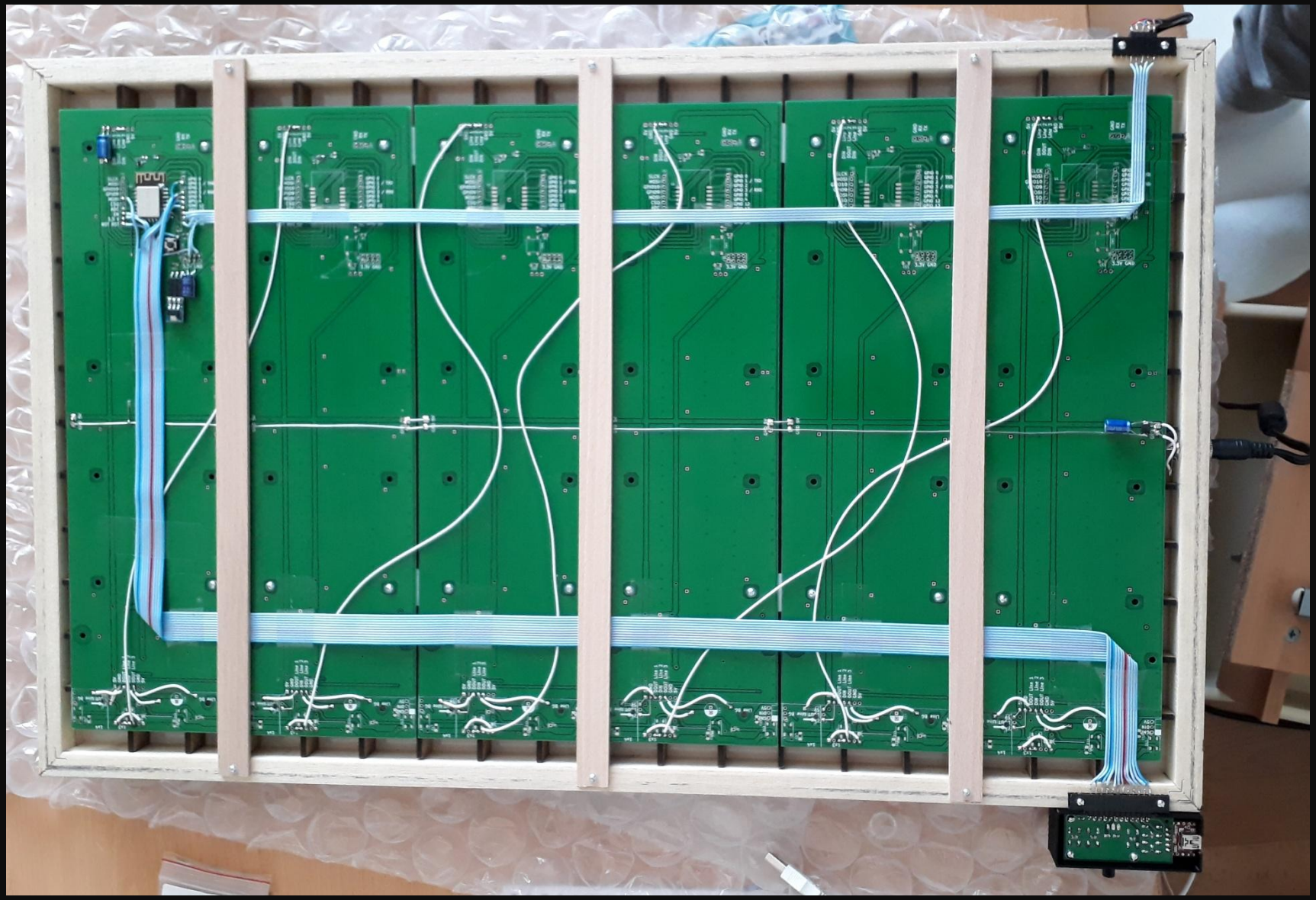
Helligkeit

Reset

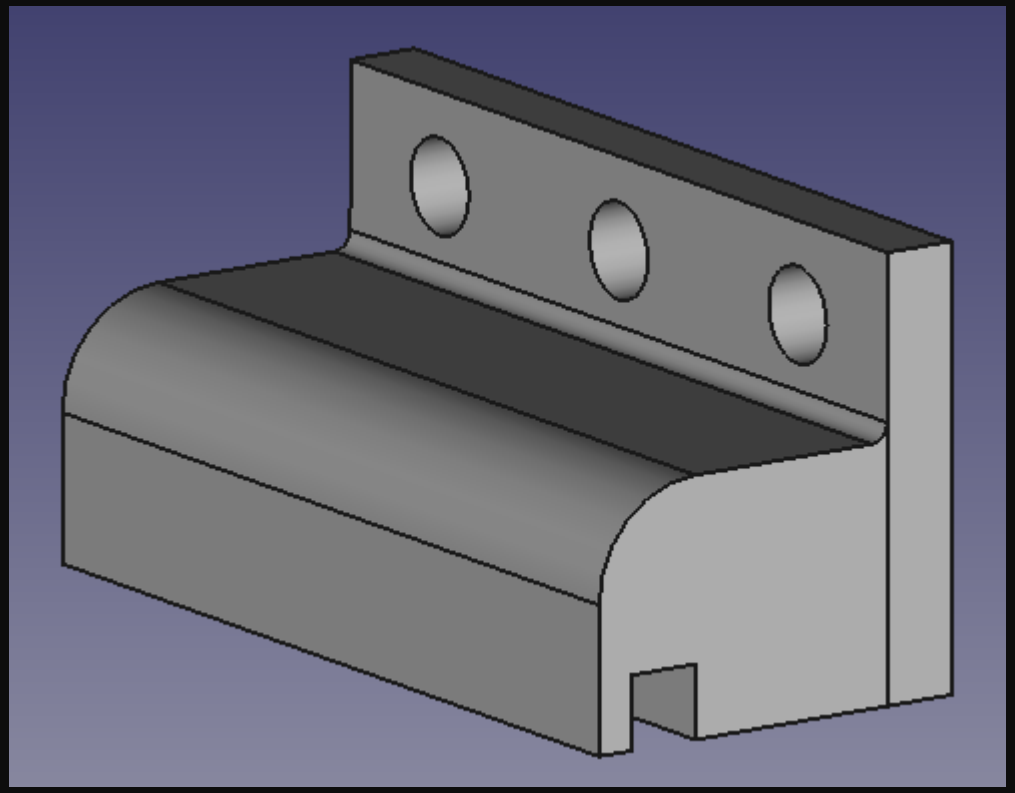
NES Controller

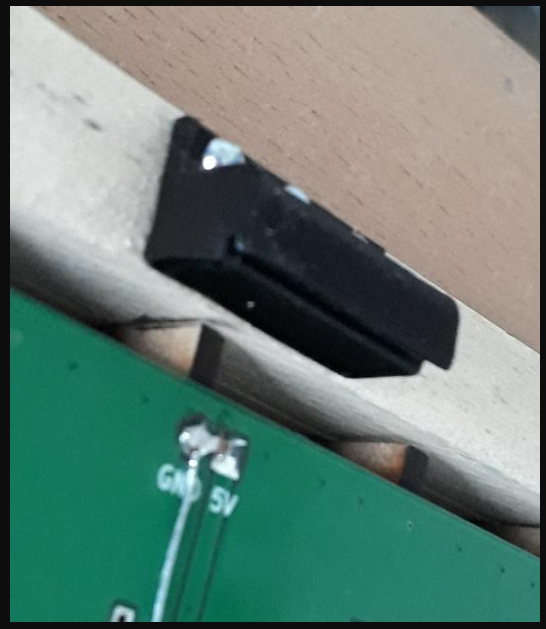


Gesamtaufbau

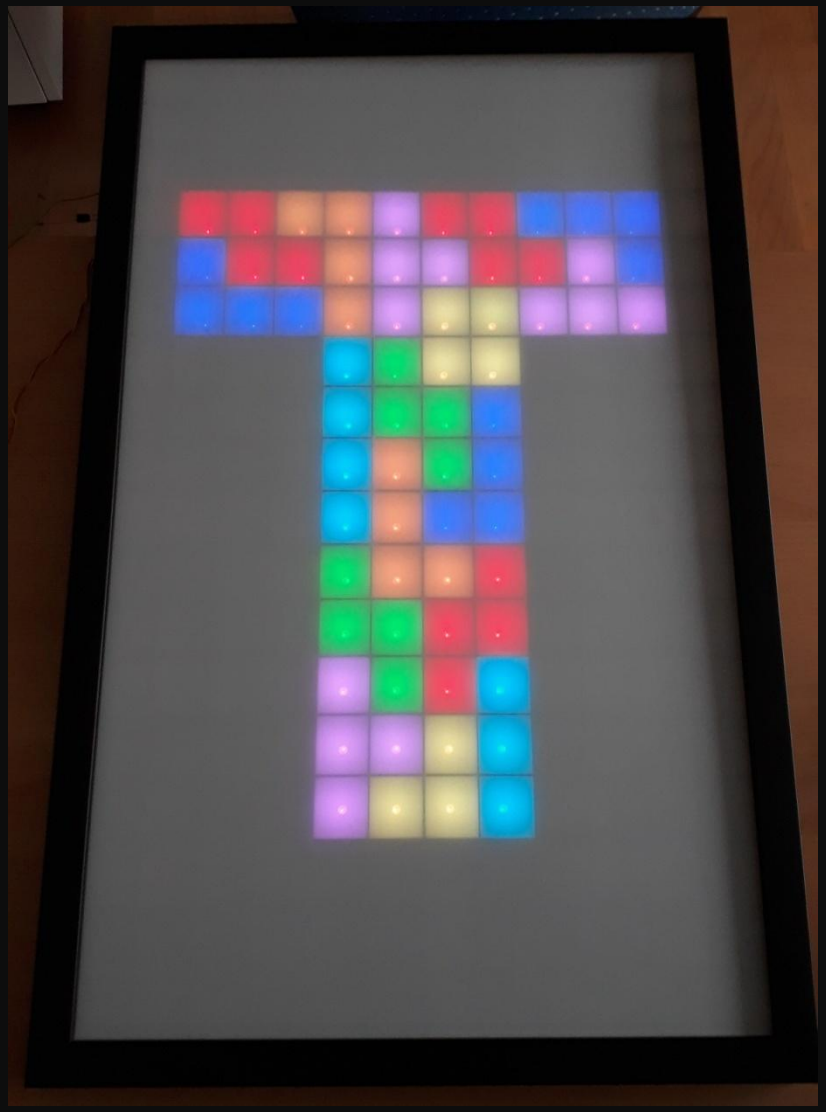


Aufhängung





Mechanischer Aufbau fertig ☺



Mechanischer Aufbau

Fazit Mechanik:

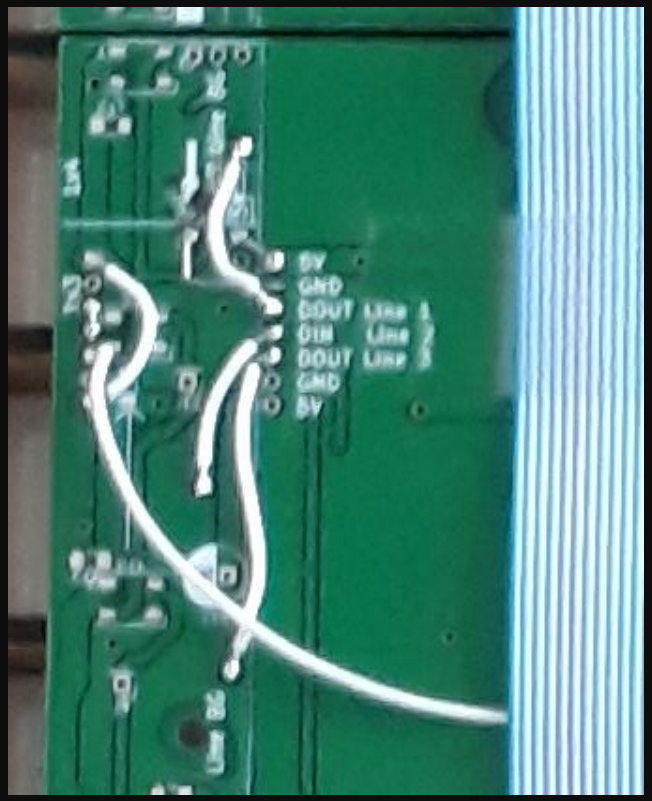
- 216 LEDs löten ist viel Arbeit
- Lasercutten ist perfekt geeignet: schnell, sehr präzise und flexibel
- Rahmen und Plexiglas am besten online bestellen
- 3D Drucker ist eine große Hilfe

→ mit dem Endergebnis sehr zufrieden ☺

Elektronik

Elektronik

- Verwendung der Wordclockplatine
- man nehme 6 Wordclock Platinen
- Randstück NICHT abtrennen, um 12 LEDs pro Reihe zu haben



Elektronische Modifikationen

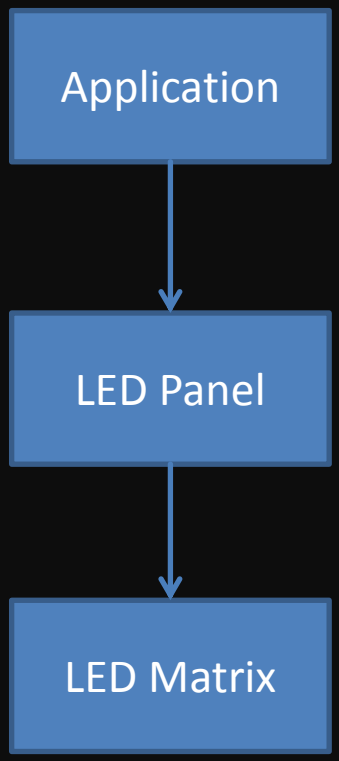
- RS232-USB Wandler ergänzt (in Bedieneinheit)
- Suppressordiode gegen ESD Schäden
- Zusätzliche Kondensatoren gegen HF Einstrahlung
- Zusätzliche Elkos an Versorgungsleitung
- Zusätzliche Masseverbindungen
- Pegelwandler für WS2812

Programmierung

Programmierung

- C++ auf Arduino Basis
  - in Visual Studio Code
- Simulation zum debuggen in C++/CLI
  - Visual Studio
  - ein paar Kompilerweichen notwendig

Programmaufbau



Anzeige auf dem Panel
Beantworten der Webserver requests
Senden/empfangen von Websockt Daten

Layer konzept
Darstellen von Zahlen, Buchstaben, Icons

Mapping der LEDs
Setzen der LEDs mittels Zeile /Spalte
Helligkeitseinstellung

LED Matrix

```
void setBrightness(unsigned int brightness);

void setLed(unsigned int col, unsigned int row, RGBColor color);
void setAll(RGBColor color);

void clearLed(unsigned int col, unsigned int row);
void clearAll();

void show();
```

LED Panel

```
void setLed(unsigned int col, unsigned int row, RGBColor color, unsigned int layer);
void clearLed(unsigned int col, unsigned int row, unsigned int layer);
bool isSet(unsigned int col, unsigned int row, unsigned int layer);
void clear(unsigned int layer);
void clear();
void setAll(RGBColor color, unsigned int layer);
void clearRow(unsigned int row, unsigned int layer);
```

```
void printImage(IPixelImage image, unsigned int col, unsigned int row, RGBColor color, b
void printDigit(unsigned int digit, unsigned int col, unsigned int row, RGBColor color,
void printFormatedNumber(int num, unsigned int numOfMinDigits, unsigned int col, unsigne
void printNumber(int num, unsigned int col, unsigned int row, RGBColor color, unsigned i
void printLineH(unsigned int col, unsigned int row, unsigned int num, RGBColor color, un
void printLineV(unsigned int col, unsigned int row, unsigned int num, RGBColor color, un
```

Application Konzept

- jede hat eigenen Namen
- wird über URL mit Name gestartet (z.B.: 192.168.4.1/tetris)
- URL kann auch GET parameter haben (192.168.4.1/settings?ssid=hallo&pass=xxx)
- Jede App muss eine eigene Website ausliefern
- Die Website kann Websocket Verbindung nutzen
- Die App kann per Websocket Daten an Webfrontend schicken
- Zusätzlich kann eine App per Bedienpanel oder NES Controller gesteuert werden

Application Interface

```
class IPixelApp
{
    public:
    virtual void start() = 0;
    virtual void end() = 0;
    virtual void loop() = 0;
    virtual void newWebsocketData(uint8_t * payload, size_t lenght) = 0;
    virtual WebsiteResponse_t getWebsiteResponse(String parameter) = 0;
    virtual void buttonEvent(Buttons::ButtonEvent_t btnEvent) = 0;

    /*  is called each 10ms
     *  do not do any blocking things! The funktion should return as soon
     *  do not send Websocket Data! */
    virtual void timerTick() = 0;
    virtual String getName() = 0;
};
```

Button Events

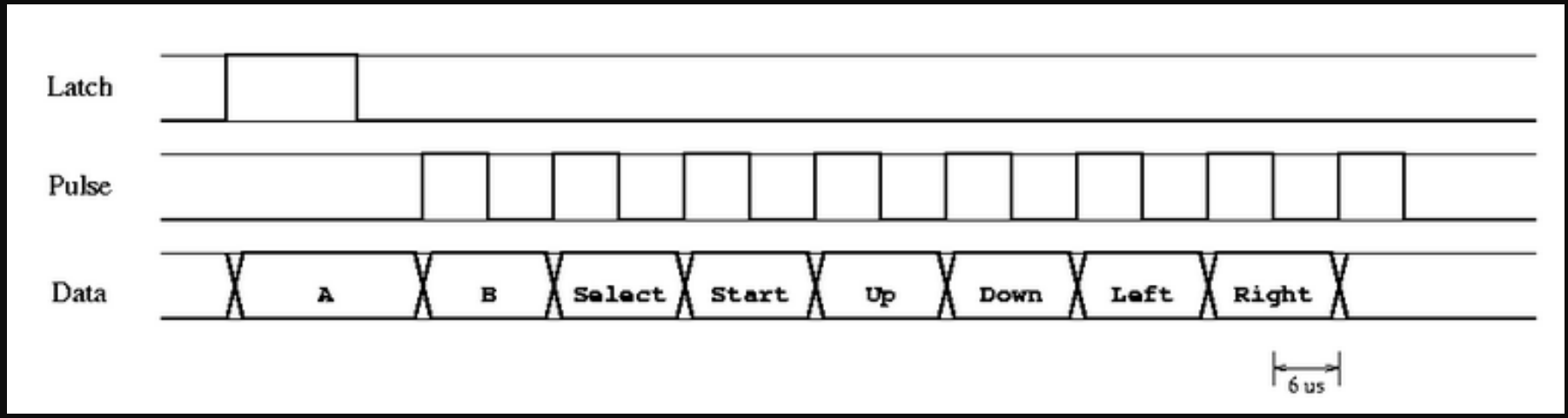
```
typedef enum Button_t{
    NONE,
    APP_SELECT,
    APP_UP,
    APP_DOWN,
    NES_UP,
    NES_RIGHT,
    NES_DOWN,
    NES_LEFT,
    NES_SELECT,
    NES_START,
    NES_A,
    NES_B
};
```

```
typedef enum Event_t{
    DOWN,
    UP
};

typedef struct ButtonEvent_t{
    Button_t button;
    Event_t event;
};
```

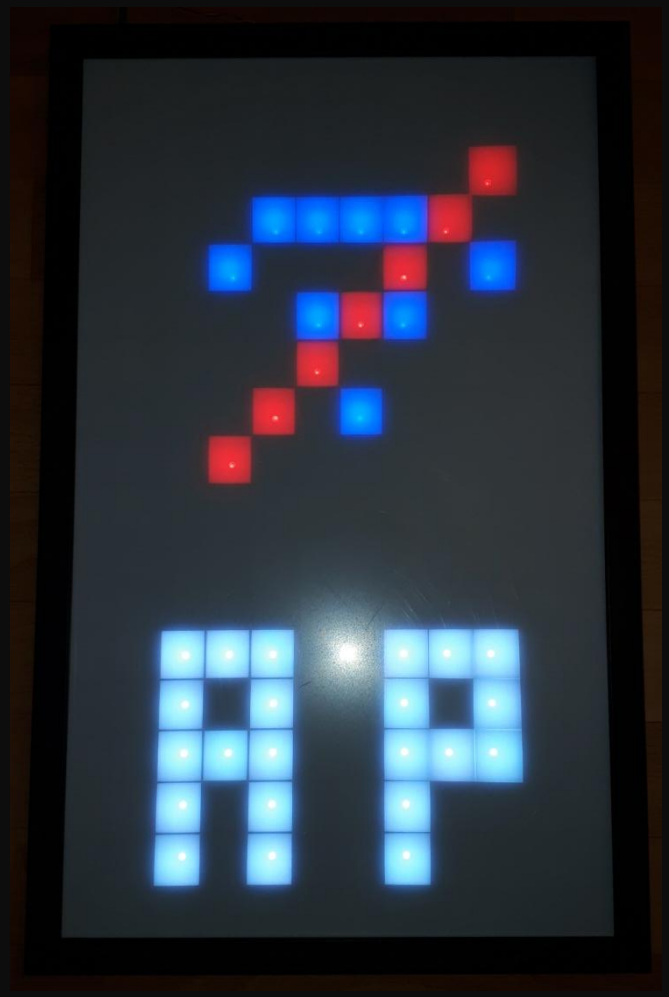
NES Controller

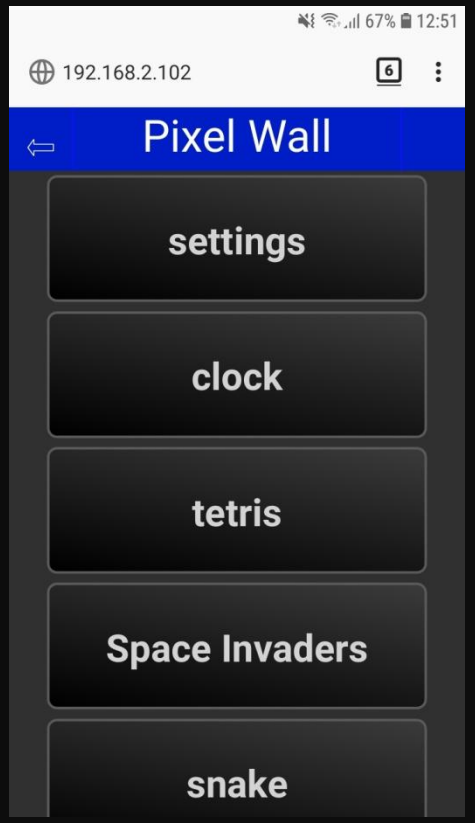
- Einfaches CMOS Schieberegister 4021
- Läuft ab 3V ☺



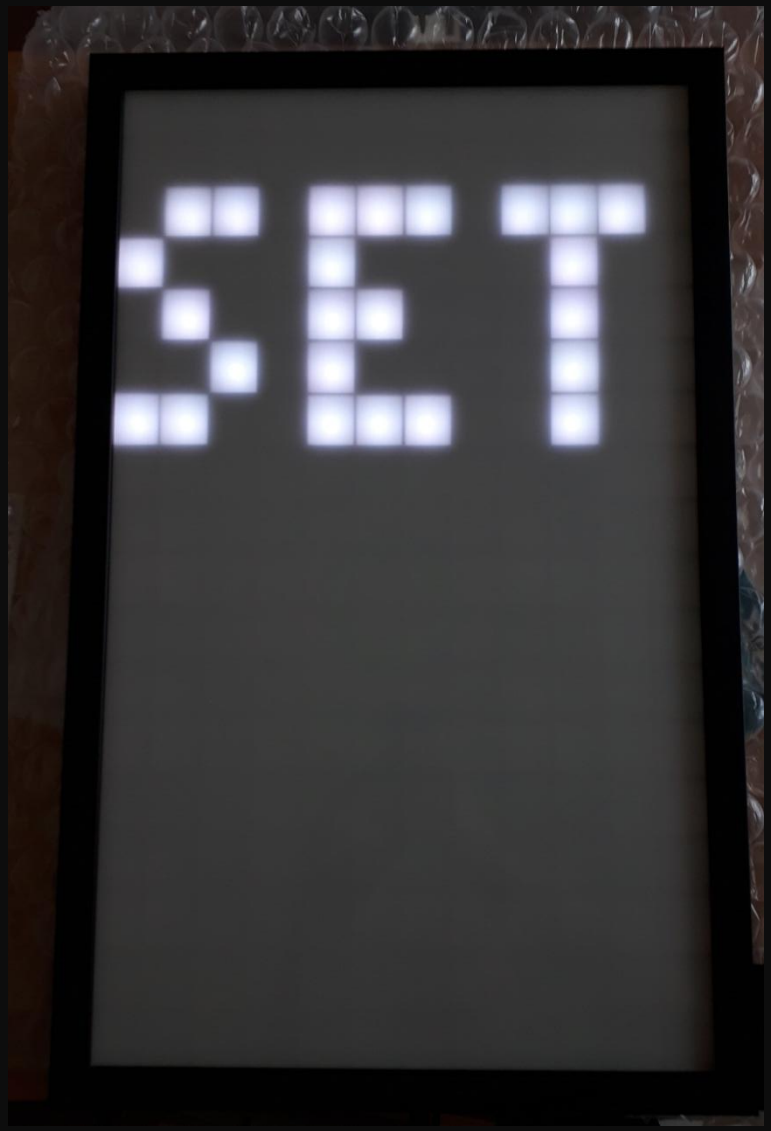
Default

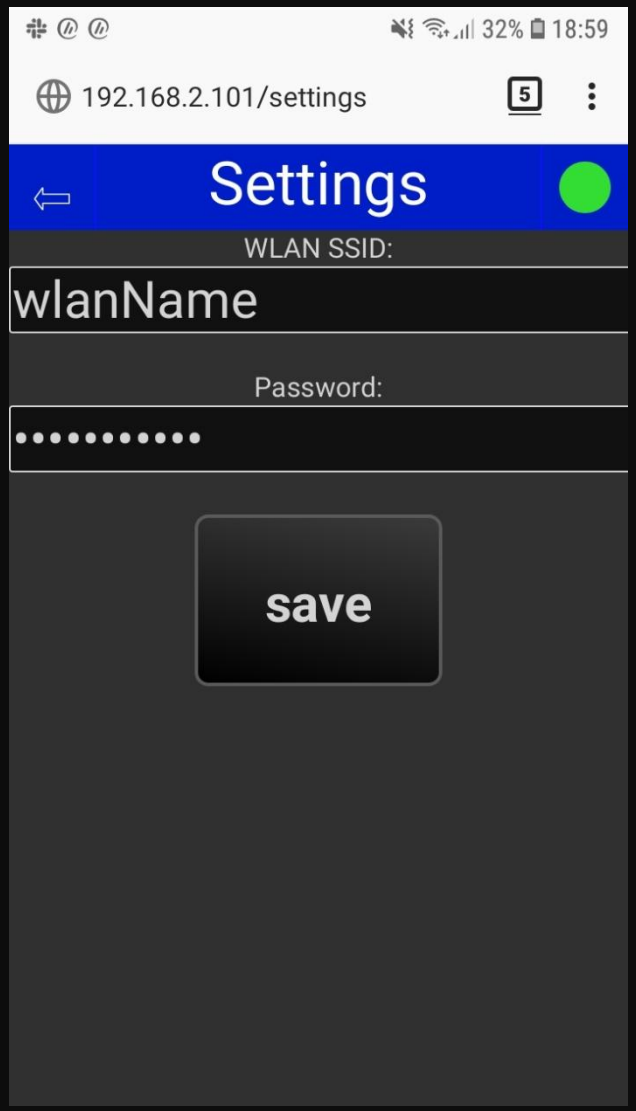

192
168


AP

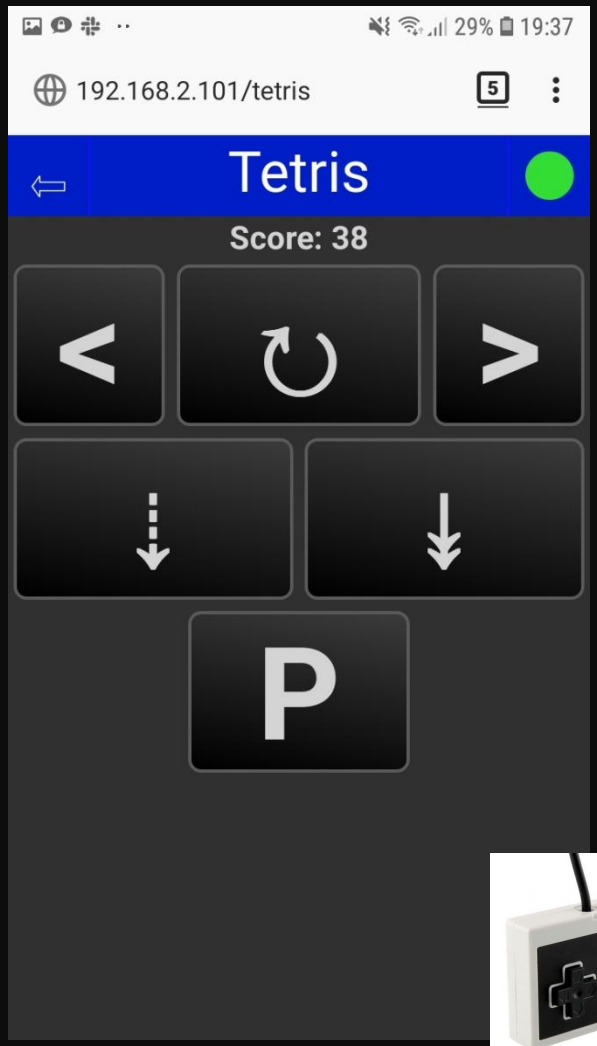

67% 12:51
192.168.2.102
Pixel Wall
settings
clock
tetris
Space Invaders
snake

Settings

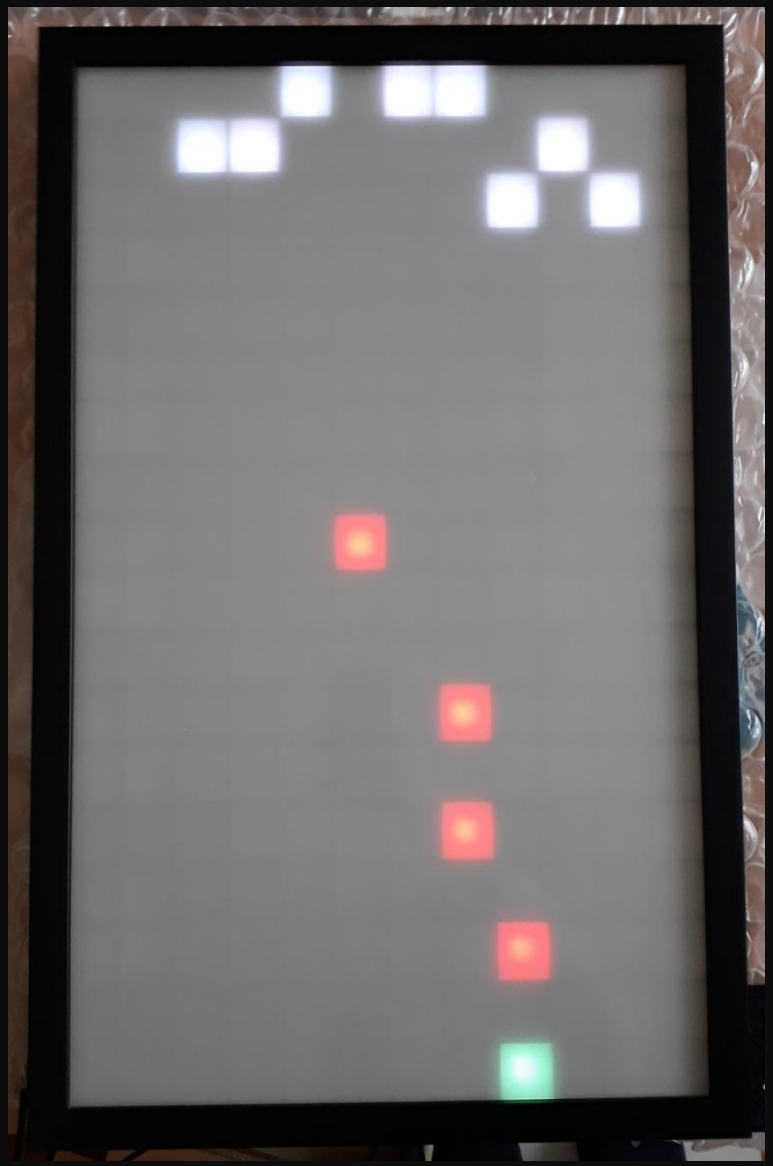

SET

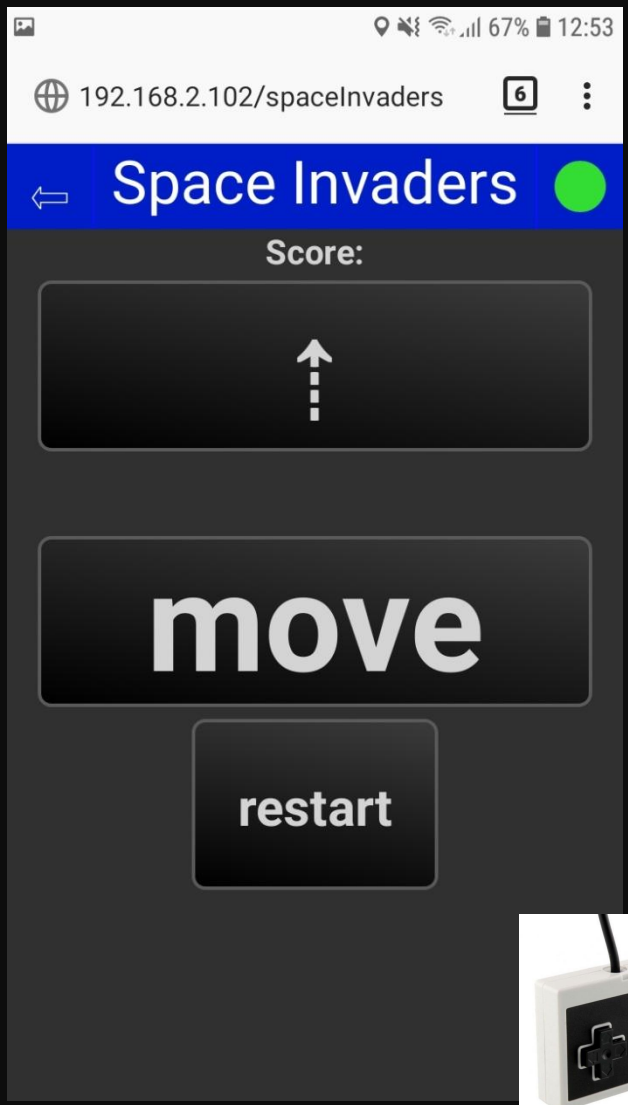

32% 18:59
192.168.2.101/settings
5
Settings
WLAN SSID:
wlanName
Password:
save

Tetris



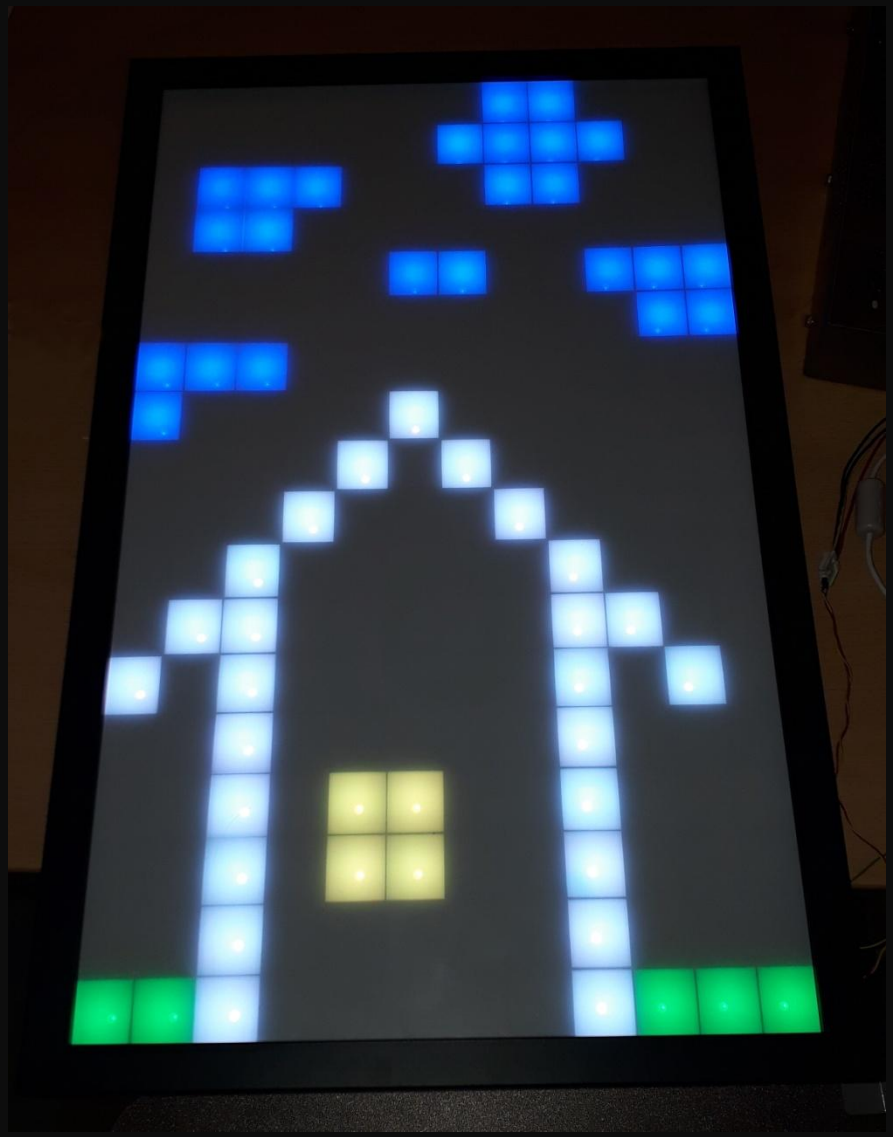
Space Invaders

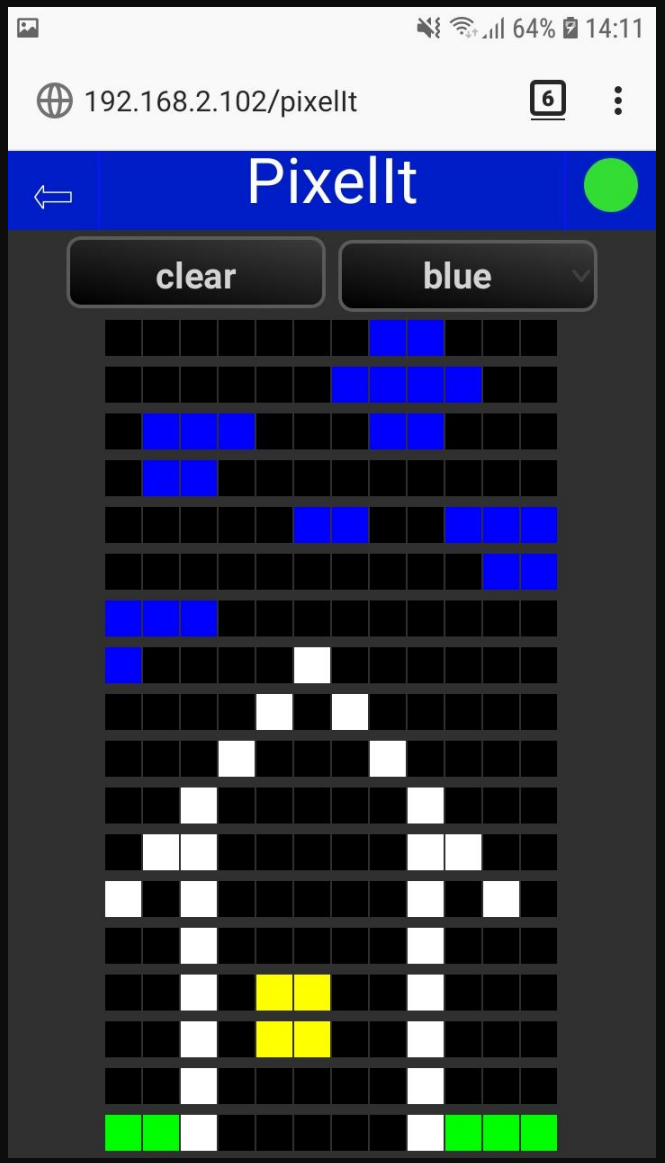




Snake

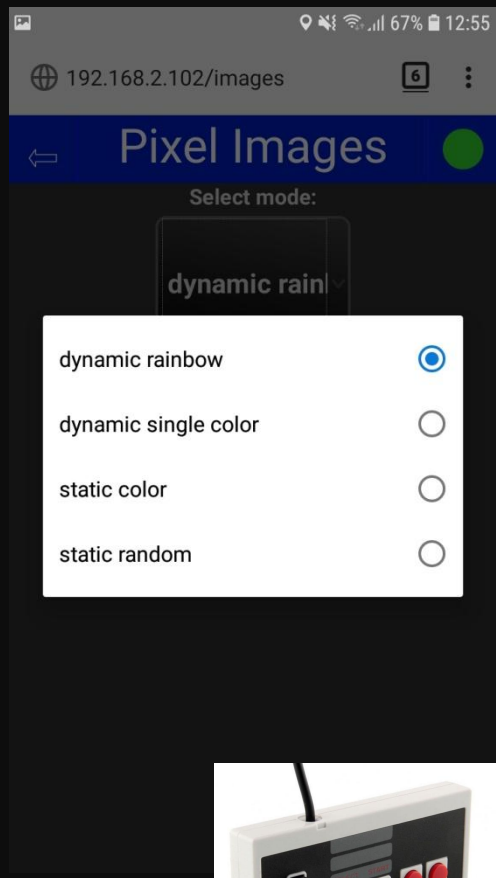
PixelIt





Images


67% 12:55
192.168.2.102/images
6
Pixel Images
Select mode:
dynamic rainl
slow
middle
fast


67% 12:55
192.168.2.102/images
6
Pixel Images
Select mode:
dynamic rainl
dynamic rainbow
dynamic single color
static color
static random

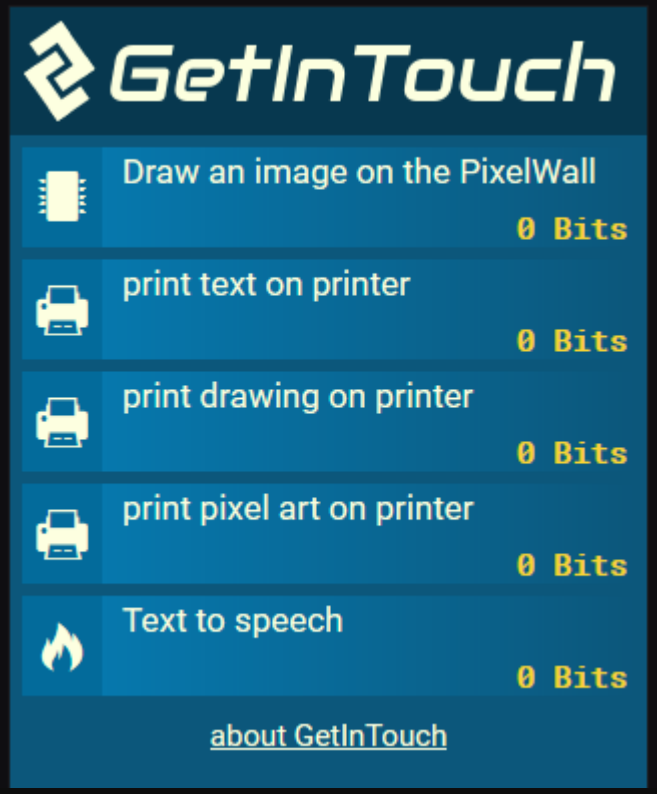
Programmierung

Fazit Programmierung:

- Verwende keine Strings für die Webseiten!
- Timer Interrupt erst starten wenn WLAN Verbindung aufgebaut
- Verwende Websocket für Echtzeit Steuerung
- Simulation sehr hilfreich
- ESP ist ein cooler Controller ☺

Control via GetInTouch

- Twitch Extension GetInTouch
- Lässt sich auch in den ESP mit rein kompilieren
- Ermöglicht Zuschauern das Zeichnen auf der PixelWall im LiveStream



Open Source

Instructable kommt demnächst:

www.instructables.com

Code ist auf Github veröffentlicht

https://github.com/C3MA/PixelWall

Fragen?