Titel

Team 1: Eggs & Bacon

*Tobias Gilgenreiner, Felix Dollinger, Marco Stoiber,*

*Maximilian Kritzenthaler, Eduard Schröder*

Untertitel

Inhalt

[Backend 2](#_Toc47967549)

[Systemhardware 2](#_Toc47967550)

[Einrichtung des Access Point 2](#_Toc47967551)

[Einrichten der Verbindung zwischen Backend und Arduino Mini 4](#_Toc47967552)

[Klasse zur seriellen Kommunikation 6](#_Toc47967553)

[RoboNova 7](#_Toc47967554)

[Einrichten des Wifi-Moduls - ESP8266 7](#_Toc47967555)

# Backend

Im Folgenden wird das Einrichten der Systeme als Access Point/Client, der Aufbau der Verbindungen vom/zum Backend und die laufenden Konsolen-Anwendungen genauer beschrieben.

## Systemhardware

*Getestet auf: Raspberry Pi 4, Raspberry Pi 3b+, Banana Pi M2+*

Die Systeme sollen auf Linux basieren. Raspbian als Betriebssystem ist am geeignetsten, da sämtliche inkludierten Bibliotheken direkt unterstützt werden sollten (besonders wiringPi.h/wiringSerial.h für UARTs am GPIO).

## Einrichtung des Access Point

*Getestet auf: Raspberry Pi 4, Raspberry Pi 3b+*

Die Einrichtung findet nach dem offiziell von Raspberry vorgestellten Ablauf statt (Raspberry Pi, 2020)[[1]](#footnote-1). Zu beachten ist, dass abhängig von dem Modell unterschiedliche Konnektivität-Stabilitäten zu erwarten sind (Raspberry Pi 4 >> Raspberry Pi 3b+). Über LAN wird ein weiteres Netzwerk überbrückt, d.h. das über den Router somit noch Zugang zum Internet für das System und seine verbundenen Geräte hergestellt werden kann.

Die Einrichtung findet über die Konsole statt. Code nach dem Befehl „nano“ wird entsprechend in die Datei eingefügt und abgespeichert.

1. ***hostapd* installieren und beim Startvorgang aktivieren:**

sudo apt install hostapd

sudo systemctl unmask hostapd

sudo systemctl enable hostapd

1. ***dnsmasq* installieren für Netzwerk Management Dienste (DNS, DHCP)**

sudo apt install dnsmasq

1. **Tools für Firewall Regeln (Speicherung/Wiederherstellung) installieren**

sudo DEBIAN\_FRONTEND=noninteractive apt install -y netfilter-persistent iptables-persistent

1. **Netzwerk konfigurieren: Feste IP mit 192.168.4.1 (darf nicht bei Nutzen eines Routers über LAN vergeben sein)**

sudo nano /etc/dhcpcd.conf

interface wlan0

static ip\_address=192.168.4.1/24

nohook wpa\_supplicant

1. **Routing und IP masquerading aktivieren (falls Internet erwünscht ist)**

sudo nano /etc/sysctl.d/routed-ap.conf

net.ipv4.ip\_forward=1

sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE

sudo netfilter-persistent save

1. **Konfigurieren der DHCP- und DNS-Dienste: Dynamische Zuweisung der Geräte. Zusätzlich sicherstellen, dass WiFi nicht blockiert wird.**

sudo mv /etc/dnsmasq.conf /etc/dnsmasq.conf.orig

sudo nano /etc/dnsmasq.conf

interface=wlan0 # Listening interface

dhcp-range=192.168.4.2,192.168.4.20,255.255.255.0,24h

# Pool of IP addresses served via DHCP

domain=wlan # Local wireless DNS domain

address=/gw.wlan/192.168.4.1

# Alias for this router

sudo rfkill unblock wlan

1. **Konfigurieren des Access Points: Nun die erwünschten Parameter eingeben.**

Netzwerk Name bei *ssid* angeben.

Netzwerk Passwort bei *wpa\_passphrase* angeben. (Zwischen 8-64 Chars)

sudo nano /etc/hostapd/hostapd.conf

country\_code=DE

interface=wlan0

ssid=RPI4

hw\_mode=g

channel=7

macaddr\_acl=0

auth\_algs=1

ignore\_broadcast\_ssid=0

wpa=2

wpa\_passphrase=INSERTPASSWORDHERE

wpa\_key\_mgmt=WPA-PSK

wpa\_pairwise=TKIP

rsn\_pairwise=CCMP

**Anm.:** Der Channel wird nicht automatisch festgelegt, d.h. falls es mit einem umgebenen Router eine Kollision gibt wird der Raspberry mit aller Wahrscheinlichkeit verlieren und kein Netzwerk anbieten! Dies ggf. dann an die Situation anpassen.

1. **Neustarten: Access Point wird fortan mit dem Starten aktiviert.**

sudo systemctl reboot

## Einrichten der Verbindung zwischen Backend und Arduino Mini

*Getestet auf: Raspberry Pi 4, Raspberry Pi 3b+, Banana Pi M2+*

Zur Kommunikation zwischen dem Backend und den Arduino Minis wird das RS232 UART-Protokoll verwendet. Da es sich bei UART um eine serielle Schnittstelle zum Senden und Empfangen von Daten handelt, muss auch auf dem Raspberry Pi die serielle Schnittstelle in den Systemeinstellungen aktiviert werden.

Auch hier wird zur Einrichtung die Konsole verwendet.

1. **Aktivieren der seriellen Schnittstelle unter Raspbian.**

Aufrufen der Systemeinstellungen:

sudo raspi.config

Anschließend werden der Reihe nach folgende Optionen ausgewählt:

* 1. Interfacing Options
  2. Serial
  3. No
  4. Yes
  5. Reboot

1. **Bluetooth deaktivieren, da dieser GPIO-UART-Ports verwendet.**

sudo nano /boot/config.txt

Folgende Zeile muss ergänzt werden:

dtoverlay = disable-bt

1. **GPIO-UART-Ports für als serielle Schnittstelle konfigurieren.**

sudo nano /boot/config.txt

Folgende Zeile muss ergänzt werden:

enable\_uart = 1

Da der Raspberry Pi 4 über vier weitere GPIO-UART-Ports verfügt, müssen diese durch die folgenden Zeilen separate aktiviert werden:

dtoverlay = uart2  
dtoverlay = uart3

dtoverlay = uart4

dtoverlay = uart5

1. **Hardware-Verbindung zwischen Raspberry Pi und Arduino Mini herstellen.**

Da der Arduino Mini Pro die Daten mit einer Spannung von 5V sendet, der Raspberry Pi am GPIO-Eingang jedoch nur für eine Maximalspannung von 3,3V ausgelegt ist, muss die Spannung des Signals auf 3,3V geregelt werden.

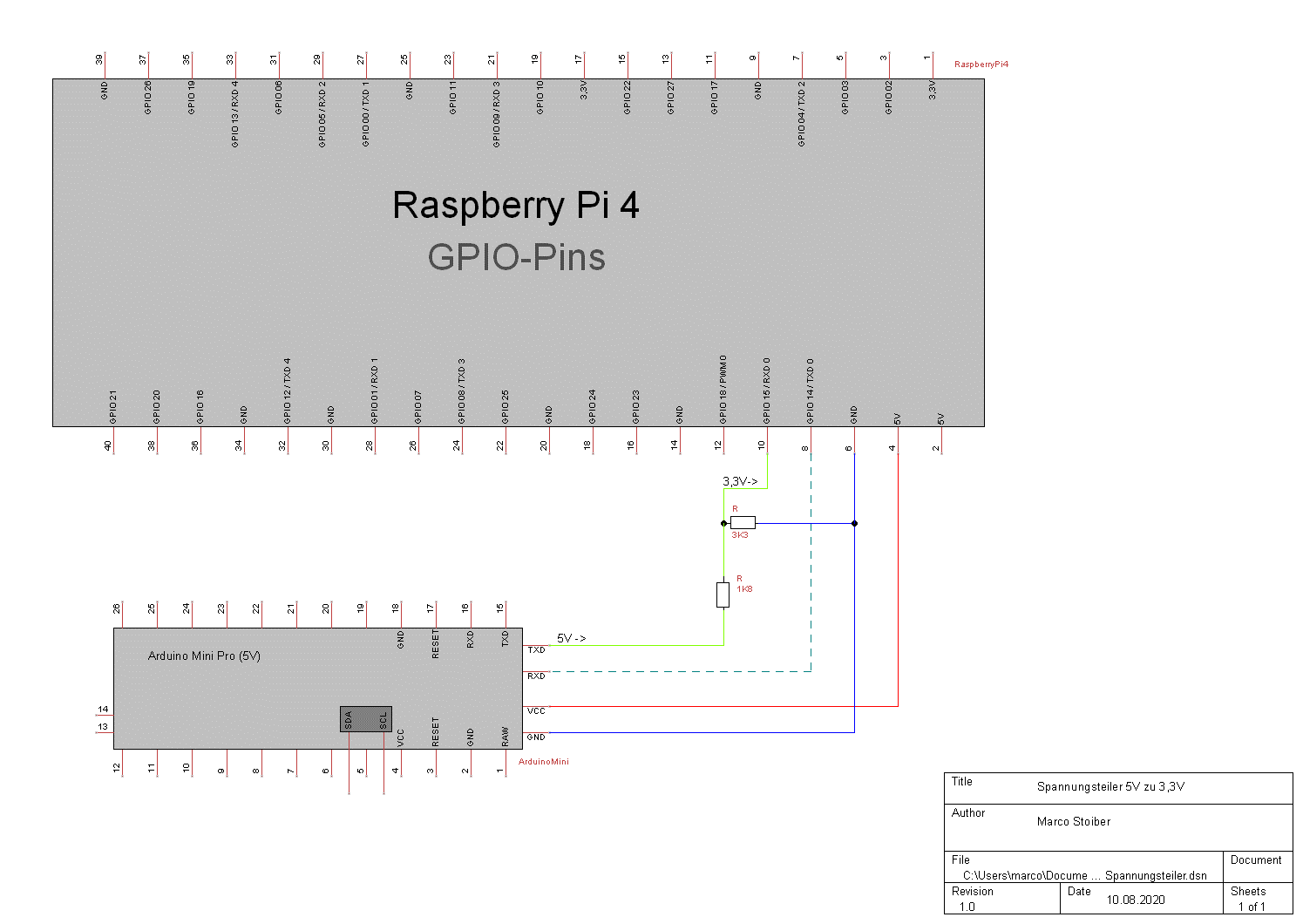
Hierzu wird ein Spannungsteiler wie folgt verwendet:

Abbildung 1: Schaltplan für Spannungsteiler (5V → 3,3V)

## Klasse zur seriellen Kommunikation

Zur einfacheren Verwendung der seriellen Schnittstellen aus dem Programm heraus, wird eine eigene Klasse zur seriellen Kommunikation verwendet. Diese wurde in der Programmiersprache C implementiert.

Nachfolgend werden die Kernelemente der Klasse kurz dargestellt.

1. **Einbinden der Bibliotheken:**

#include <wiringPi.h>

#include <wiringSerial.h>

1. **Öffnen eines seriellen Ports:**

wiringPiSetup();  
  
int fileDescriptor = serialOpen(char\* port, int baudrate);

1. **Auslesen eines Zeichens vom seriellen Port:**

If(serialDataAvail(fileDescriptor))

{  
 char input = serialGetChar(fileDescriptor);  
}

Prüft, ob Daten am Eingang vorhanden sind und liest anschließend ein Zeichen aus.

Folgende Tabelle zeigt die verwendeten UART-Ports und deren Adressen:

|  |  |  |  |
| --- | --- | --- | --- |
| **Port** | **Adresse** | **TX-Pin** | **RX-Pin** |
| USB UART | “/dev/ttyUSB0” | - | - |
| GPIO UART0 | “/dev/ttyAMA0” | GPIO14 | GPIO15 |
| GPIO UART1 | “/dev/ttyAMA1” | GPIO0 | GPIO1 |
| GPIO UART2 | “/dev/ttyAMA2” | GPIO4 | GPIO5 |
| GPIO UART3 | “/dev/ttyAMA3” | GPIO8 | GPIO9 |
| GPIO UART4 | “/dev/ttyAMA4” | GPIO12 | GPIO13 |

# RoboNova

## Einrichten des Wifi-Moduls - ESP8266

Zur Verwendung des ESP8266 wird die offizielle Anleitung des Herstellers *Simac Electronics* verwendet (Simac Electronics GmbH, 2020)[[2]](#footnote-2).

Als Entwicklungsumgebung wird, wie vom Hersteller empfohlen, die Arduino IDE verwendet.

Im Rahmen dieses Projekts wurde stets ein Arduino Uno zum Hochladen von Programmcode auf den ESP8266 genutzt.

1. <https://github.com/raspberrypi/documentation/blob/master/configuration/wireless/access-point-routed.md> [↑](#footnote-ref-1)
2. <https://joy-it.net/files/files/Produkte/SBC-ESP8266/SBC-ESP8266-Anleitung-20200626.pdf> [↑](#footnote-ref-2)