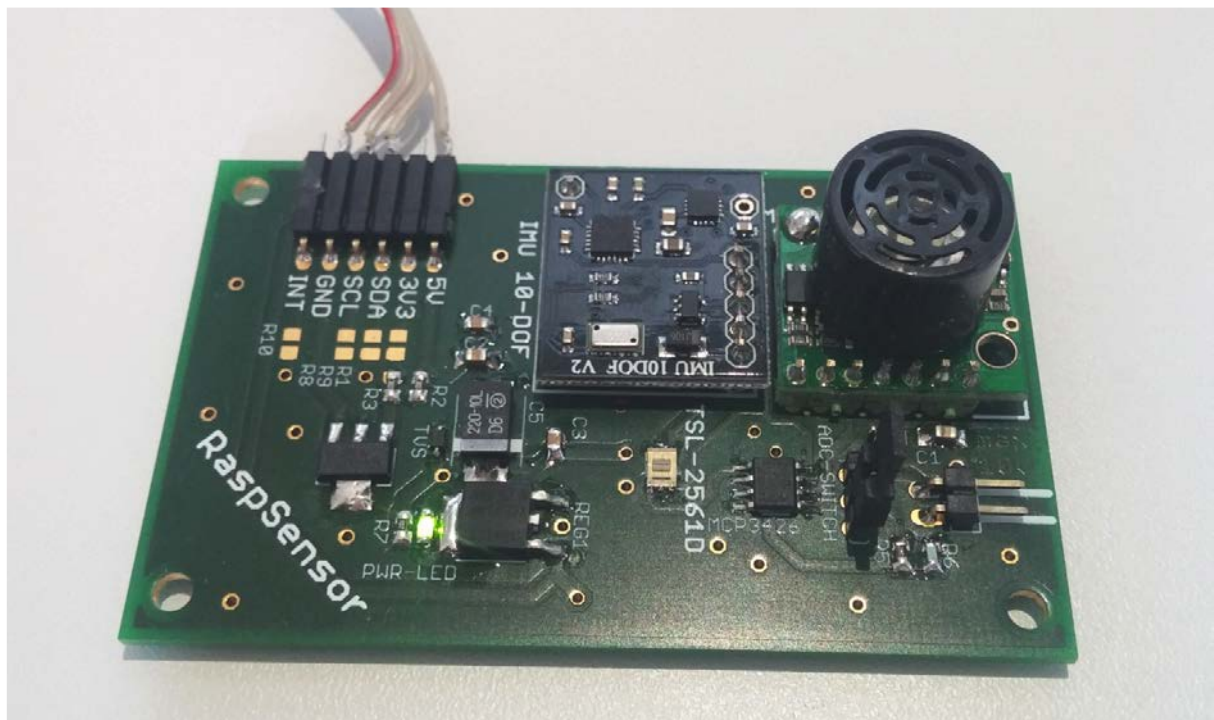


Projekt TI

Sensorikplatine für Raspberry PI SS 2014

Lukas Obkircher
Popetz Tobias



Inhaltsverzeichnis

1	Projektbeschreibung.....	1
2	Schaltplan	2
2.1	I2C	3
2.2	Supply.....	3
2.3	Sensoren	3
3	Layout	4
4	I2C-Bus aktivieren.....	5
5	Softwaredokumentation.....	6
6	Einrichtung des Webserver.....	7
7	Ergebnis	8
8	Schlusswort.....	9
	Anhang	9

1 Projektbeschreibung

Ziel des Projekts ist es eine Erweiterungsplatine für den Raspberry PI zu erstellen, auf der sich verschiedenste Sensorik befindet. Zur Kommunikation wird ein 3.3V I²C-Bus verwendet. Die Versorgungsspannung kann wahlweise 3.3V oder 5V betragen und ist über die Bus-Stiftleiste erreichbar. Um die Platine vor unsachgemäßem Anschließen zu schützen, wurde ein Verpolungsschutz mit integriert.

Folgende Sensoren befinden sich auf der Platine:

- 3-Achsen Beschleunigungssensor (MPU-6050)
- 3-Achsen Drehsensor (MPU-6050)
- 3-Achsen Kompass (HMC5883L)
- Luftdrucksensor (MS5611)
- Temperatursensor (MS5611)
- Lichtsensor (TSL-2561D)
- Ultraschall-Abstandsmesser (MaxSonar-EZ1)
- 2-Kanal AD-Wandler (MCP3426)

Sämtliche Sensorik wird mit 3.3V betrieben. Zur Abstandsmessung wird vom Ultraschall-Modul der analoge Ausgang verwendet, welcher über den ersten Kanal des ADCs eingelesen wird. Der zweite Kanal kann wahlweise die 5V-Versorgungsspannung oder eine externe Spannung von 0 bis 10V messen.

2 Schaltplan

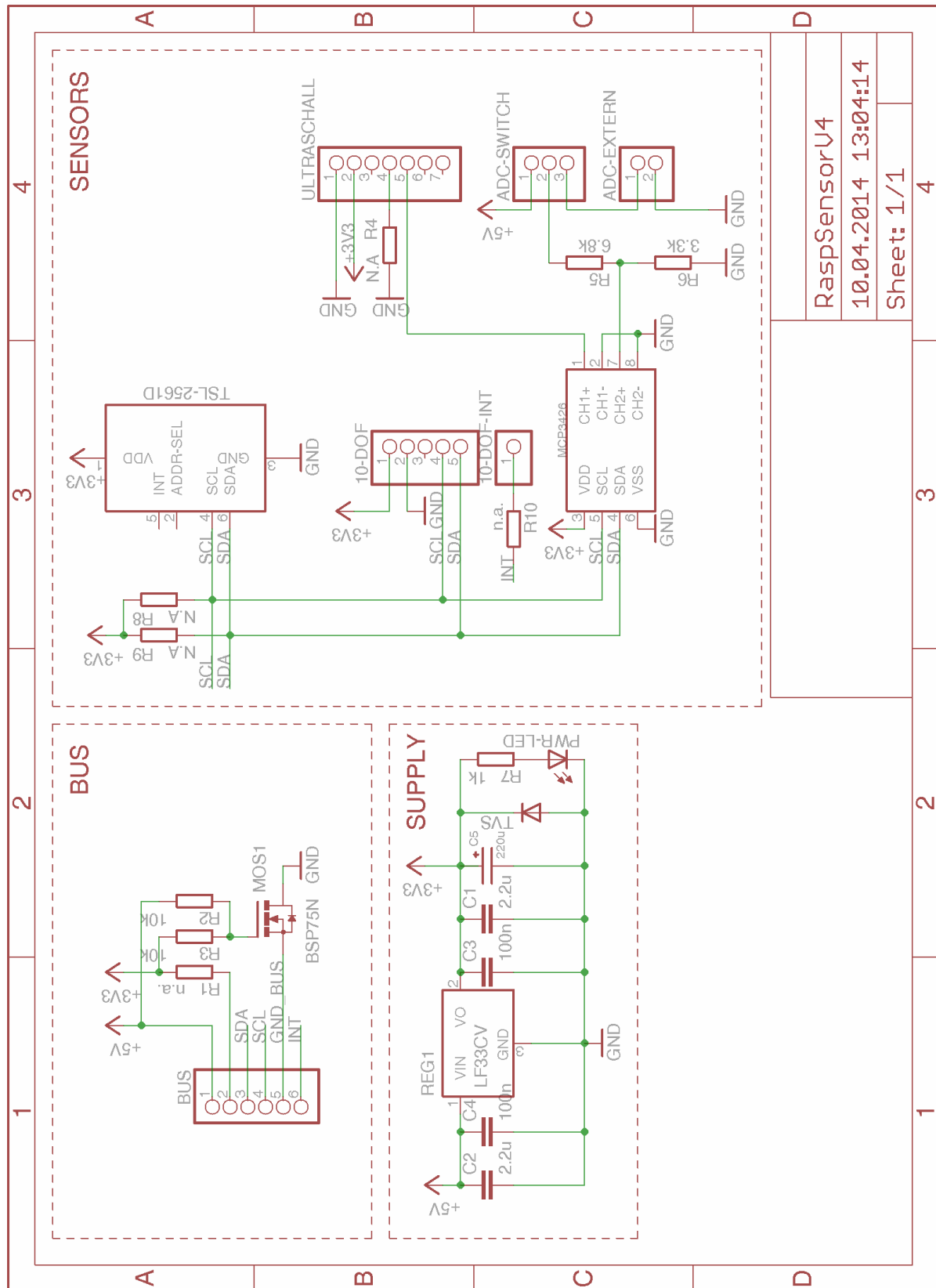


Abbildung 1: Eagle-Schaltplan

2.1 I2C

Der Anschluss an den Raspberry PI erfolgt über eine 6-Pin Stiftleiste. Diese beinhaltet neben den I2C-Signalen (SDA, SCL) auch verschiedene Spannungsversorgungen (5V und 3.3V), sowie ein Interrupt Signal. Die 3.3V Leitung wird nicht verwendet, sondern wie weiter unten beschrieben, aus der 5V-Spannung selbst erzeugt. Um die Platine vor unsachgemäßem Anschließen zu schützen, wurde ein Verpolungsschutz mittels eines N-MOSFET integriert. Dieser stellt eine Trennung zwischen dem internen und externen Groundsignal dar. Sobald eine positive Spannung an dem +5V-Pin anliegt, wird der MOSFET niederohmig und ermöglicht die Herstellung einer Versorgungsspannung.

2.2 Supply

Zur Erzeugung der nötigen 3.3V Spannungsversorgung wird der lineare Spannungsregler LF33CV verwendet. Dieser kann eine maximale Eingangsspannung von 16V in 3.3V bei maximal 500mA umwandeln. Da wir einen geschätzten Stromverbrauch von 20mA haben, ist keine Kühlung für den Spannungsregler nötig.

Zur Glättung und Auskopplung von Stromspitzen werden mehrere Kondensatoren sowie eine TVS-Diode verwendet. Eine LED signalisiert die einwandfreie Spannungsversorgung.

2.3 Sensoren

Beim **TSL-2561D** von TAOS handelt es sich um einen Lichtsensor, welcher mit einer Spannung von 2.7V bis 3.5V betrieben werden kann und über I²C kommuniziert. Der Sensor hat einen digitalen 16-Bit Ausgang und eine programmierbare interne Verstärkung. Um Störungen zu vermeiden, werden 50/60Hz-Signale automatisch unterdrückt.

Auf der **IMU-10DOF**-Platine von DroTek (<http://www.drotek.fr/>) befinden sich folgende Sensoren:

1x MPU-6050 (3-Achsen Beschleunigungssensor, 3-Achsen Drehsensor)

1x HMC5883L (3-Achsen Kompass)

1x MS561101BA (Barometer, Thermometer)

Alle drei Sensoren lassen sich über I²C auslesen und falls nötig konfigurieren.

Sowohl die MPU-6050 als auch der HMC5883L liefern Werte, die nicht mehr weiter angepasst werden müssen. Im PROM des MS561101BA befinden sich individuell bei der Herstellung eingespeicherte Kalibrierungswerte, die zur Umrechnung in reale Werte (Temperatur und Luftdruck) nötig sind.

Die **MPU-6050** kann mit bis zu 1kHz ausgelesen werden und liefert dabei 16-Bit-Werte für eine maximale Beschleunigung von 16G, sowie eine maximale Drehrate von 2000°/s.

Der Kompass **HMC5883L** kann bis maximal ± 8 Gs messen, hat eine Auflösung von 5 mGs und eine Aktualisierungsrate von bis zu 75Hz.

Der Luftdrucksensor **MS561101BA** benötigt für eine Messung etwa 10ms. Da zum Auswerten des Luftdrucks auch die Temperaturmessung nötig ist, verringert sich die maximale Sample rate auf 50Hz (Abwechselnde Messung von Luftdruck und Temperatur). Durch die hohe Auflösung des 24-Bit-ADCs, kann er einen Luftdruckunterschied von 0,012mBar, sowie 0,01°C messen.

Der Ultraschall-Sensor **LV-MaxSonar-EZ1** von Maxbotix befindet sich durch die Hardwarebeschaltung bereits in einem Modus, welcher ihn zum durchgehenden Messen veranlasst. Die Ausgabe der Abstandsmessung erfolgt analog, wobei 6.4mV einem Zoll entsprechen. Seine maximale Reichweite ist auf 6,45m begrenzt und die Updaterate beträgt 20Hz.

Beim **MCP3426** von Microchip handelt es sich um einen 2-Kanal 16Bit-ADC, welcher für dieses Projekt im 12Bit Modus betrieben wird und dabei eine Sample rate von 240Hz erreicht.

3 Layout

Das Layout wurde mit der Software Eagle gezeichnet und die Abmessungen entsprechend der für das Projekt zur Verfügung gestellten Vorlage.

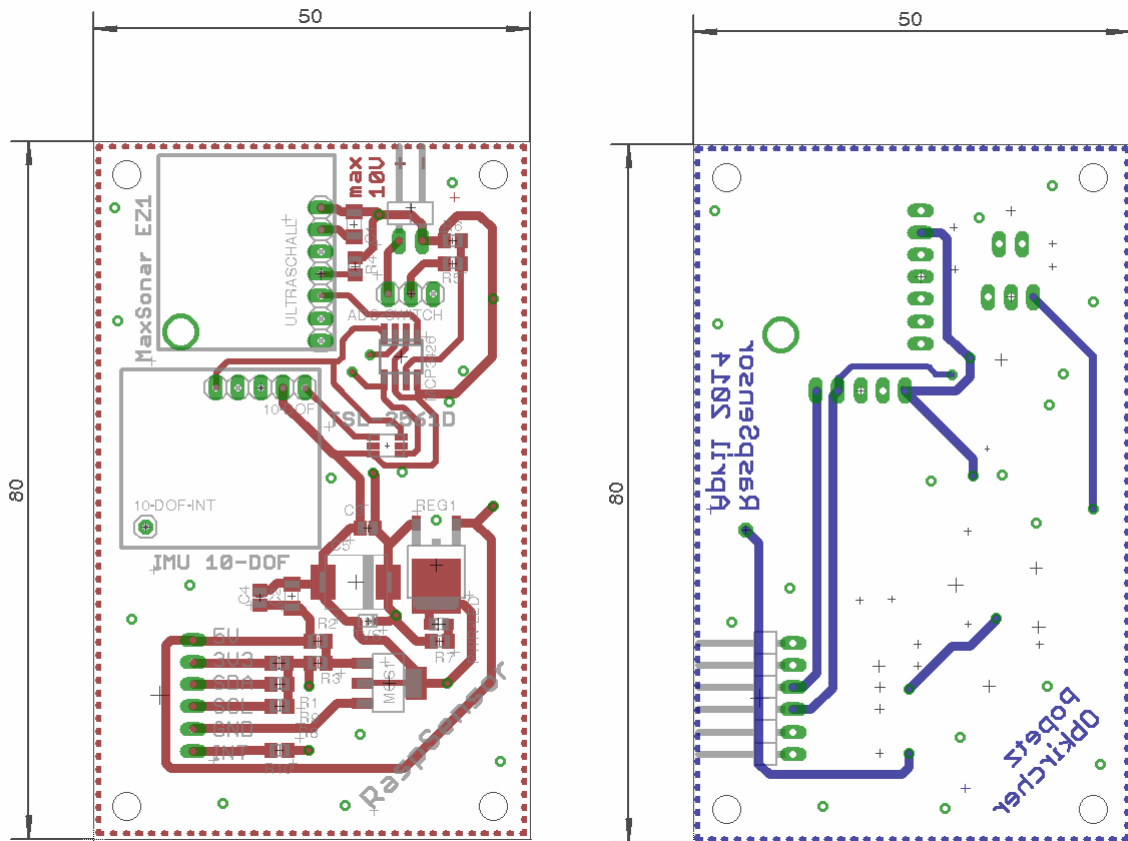


Abbildung 2: Eagle Layout (Top & Bottom)

4 I2C-Bus aktivieren

Um den I2C-Bus des Rasperrys zu aktivieren, sind die folgenden Schritte notwendig:

Raspberry-Config modifizieren:

```
sudo nano /etc/modprobe.d/raspi-blacklist.conf
```

Dort den i2c-Bus von der Blackliste entfernen.

Dies geschieht durch ein „#“ am Zeilenbeginn (entspricht Kommentar).

Nachträglich soll der Inhalt der Datei folgendermaßen aussehen:

```
# blacklist spi and i2c by default (many users don't need them)
blacklist spi-bcm2708
#blacklist i2c-bcm2708
```

I2C zum Kernel hinzufügen

Hierfür muss der I2C-Bus folgendermaßen hinzugefügt werden:

In der Datei:

```
sudo nano /etc/modules
```

Wird am Ende „i2c-dev“ hinzugefügt.

Der Inhalt der Datei sieht dann so aus:

```
# /etc/modules: kernel modules to load at boot time.
#
# This file contains the names of kernel modules that should be loaded
# at boot time, one per line. Lines beginning with "#" are ignored.
# Parameters can be specified after the module name.
```

```
snd-bcm2835
i2c-dev
```

Nun müssen für die Programmierung mittels Python einige Tools installiert werden.

Hierfür werden folgende Befehle der Reihe nach ausgeführt:

```
sudo apt-get update
sudo apt-get install i2c-tools
sudo apt-get install python-smbus
sudo adduser pi i2c
```

Abschließend wird der Raspberry mit „*sudo reboot*“ neu gestartet.

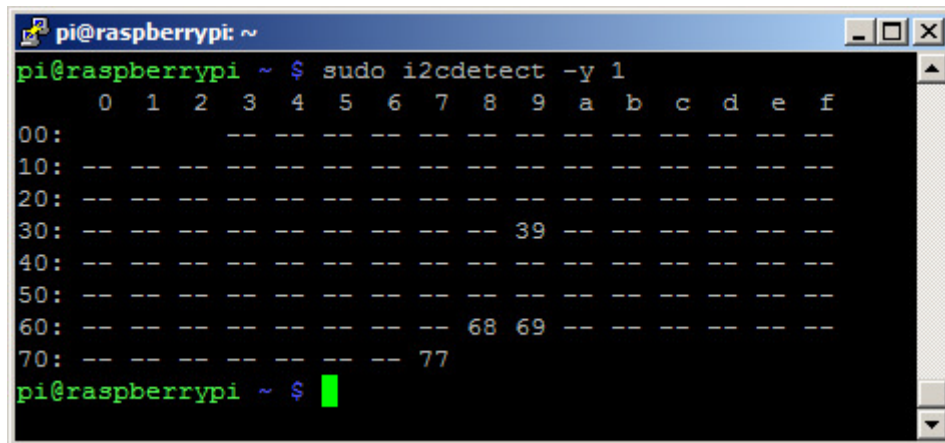
Nach dem Neustart werden durch den Befehl

```
i2cdetect -y 1
```

(Für den ursprünglichen Raspberry *i2cdetect -y 0*)

alle möglichen Adressen nach Geräten gesucht und entsprechend angezeigt.

Wenn alles ordnungsgemäß funktioniert hat, erscheint die folgende Ausgabe:



```
pi@raspberrypi: ~  
pi@raspberrypi ~ $ sudo i2cdetect -y 1  
    0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f  
00:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  
10:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  
20:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  
30:  --  --  --  --  --  --  --  --  39  --  --  --  --  --  --  
40:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  
50:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  
60:  --  --  --  --  --  --  --  68  69  --  --  --  --  --  --  
70:  --  --  --  --  --  --  77  --  --  --  --  --  --  --  --  
pi@raspberrypi ~ $
```

Abbildung 3: I2C-Gerätedetektion

Die Adressen sind hexadezimal und dahinter befinden sich folgende Ics:

0x39 = Lichtsensor

0x68 = ADC

0x69 = MPU

0x77 = MS5611

Der Kompass wird hier nicht angezeigt, da dieser mit einem separaten I2C-Bus mit der MPU verbunden ist. (Adresse vom Kompass: 0x1E)

5 Softwaredokumentation

Unsere Software besteht aus drei verschiedenen Dateien.

- RaspSensor.py
- broadcast.py
- RaspSensorWebpage.html

Die generelle Funktionsweise sieht folgendermaßen aus:

Das Programm „RaspSensor.py“ liest periodisch die Sensoren aus, rechnet die Daten in reale Werte um und kann diese, als String formatiert, anderen Python-Programmen anbieten.

Unser Broadcast-Server „broadcast.py“ ist ein auf Autobahn (<http://autobahn.ws/>) und Twisted (<https://pypi.python.org/pypi/Twisted>) basierender Python Websocket-Server, welcher auf Port 8000 geöffnet wird.

Dieser sendet periodisch an alle Clients den von „RaspSensor.py“ erhaltenen String.

„RaspSensorWebpage.html“: Das Auslesen und Anzeigen der Daten im Browser basiert auf JavaScript, wobei die im String übergeben Werte in eine Zeitleiste, ähnlich einem Oszilloskop übergeben und mittels der Diagramm-Bibliothek „Smoothie Charts“ (<http://smoothiecharts.org/>) angezeigt werden.

Die genauere Programmbeschreibung kann dem jeweiligen Source-Code entnommen werden.

6 Einrichtung des Webserver

Zuerst muss „lighttpd“ auf dem Raspberry installiert werden

Hierfür wird der folgende Befehl auf der Console ausgeführt:

„apt-get install lighttpd“

Anschließend muss noch die Webseite und der Smoothie.js auf dem Raspberry in den folgenden Ordner kopiert werden:

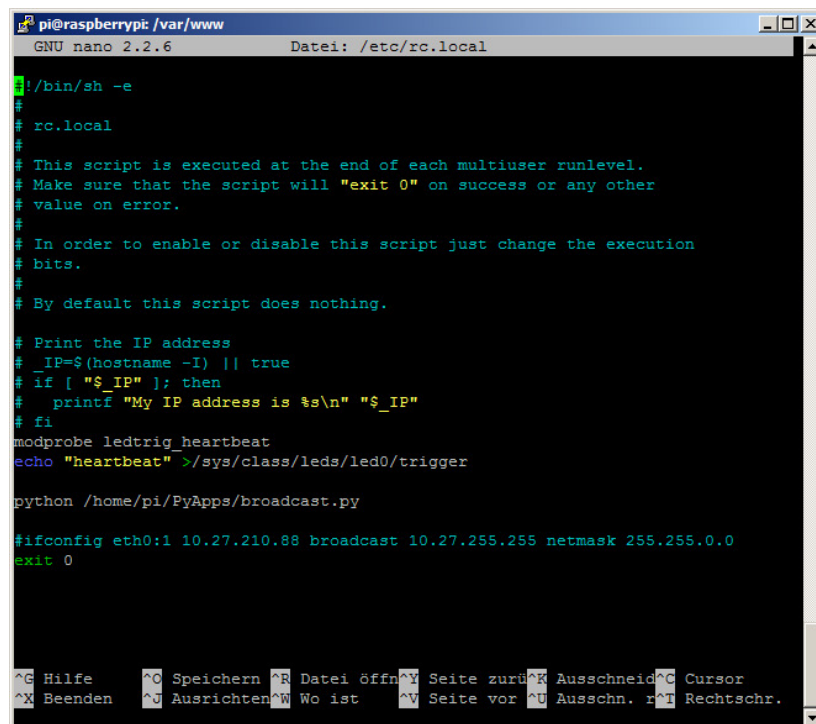
/var/www/

Wichtig hierbei ist, dass die Webseite den Namen „index.html“ besitzt.

Alternativ kann auch die Konfig-Datei des von Lighttpd verändert werden.

Nun muss noch in "etc/rc.local" (wird geöffnet mittels „sudo nano /etc/rc.local“) folgende Zeile hinzugefügt werden:

python /pfad/zur/datei/broadcast.py



```
pi@raspberrypi: /var/www
GNU nano 2.2.6      Datei: /etc/rc.local

#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.
#
# Print the IP address
# _IP=$(hostname -I) || true
# if [ "$_IP" ]; then
#   printf "My IP address is %s\n" "$_IP"
# fi
modprobe ledtrig_heartbeat
echo "heartbeat" >/sys/class/leds/led0/trigger
python /home/pi/PyApps/broadcast.py

#ifconfig eth0:1 10.27.210.88 broadcast 10.27.255.255 netmask 255.255.0.0
exit 0

^G Hilfe      ^C Speichern ^R Datei öffn ^Y Seite zurü ^K Ausschneid ^C Cursor
^X Beenden    ^O Ausrichten ^W Wo ist    ^V Seite vor ^U Ausschn. r ^T Rechtschr.
```

Abbildung 4: Bearbeitetes Startscript

7 Ergebnis

Unsere Hardware erfasst mit ca. 20Hz die gesamte Sensorik. Durch die Verwendung eines Broadcast-Servers ist es auch möglich von mehreren Clients gleichzeitig auf die Sensorik-Webseite zuzugreifen.

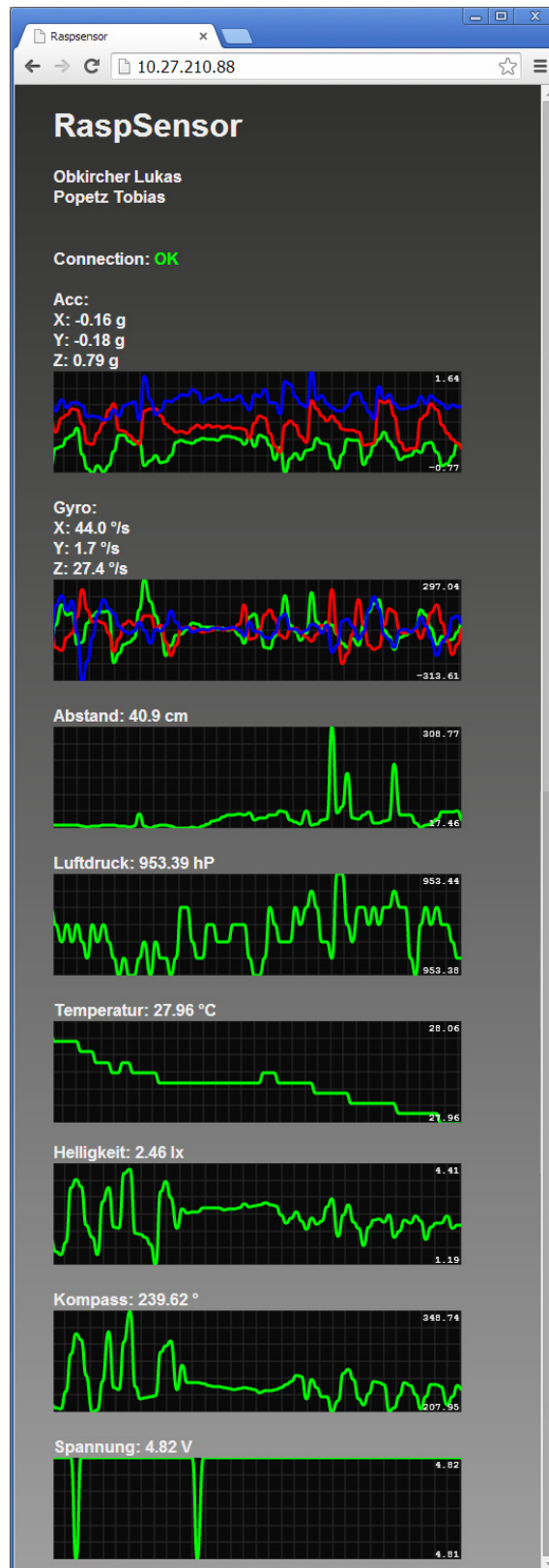


Abbildung 5: Finale Webseite

8 Schlusswort

Unserem Team hat es viel Freude bereitet mit dem RaspberryPI zu arbeiten und ihn näher kennenzulernen. Durch das Zeichnen der Platine und der Programmierung in Python konnten wir viele neue Erkenntnisse gewinnen, die uns im Verlauf des weiteren Studiums von großen Nutzen sein können. Einen großen Dank möchten wir Herrn Prof. Plate aussprechen, ohne den das Projekt keine so schnellen Fortschritte gemacht hätte.

Obkircher Lukas
Popetz Tobias

Anhang

Code:

- broadcast.py
- RaspSensor.py
- RaspSensorWebpage
- smoothie.js

Datenblätter:

- BSP 75N (NMOS)
- LF33CV (Linearregler)
- MCP3426 (ADC)
- MPU6050 (Acc+Gyro)
- MS5611 (Baro)
- TSL2561D (Licht)

Eagle:

- Schaltplan
- Layout