Entwicklung eines Boardcomputers zum Übertragen von Messdaten in Echtzeit an einen Webserver

AIT31V | Berufskolleg für Technik und Medien am Platz der Republik

PA 2019 ITA | Hr.Wächter

Lucas Wiedenfeld, Lasse Pittelkow, Sven Neuenhofer

2019

Inhaltsverzeichnis

[1. Projektbeschreibung 3](#_Toc25132202)

[1.1. Kurzbeschreibung 3](#_Toc25132203)

[1.2. Ist-Analyse 3](#_Toc25132204)

[1.3. Soll-Kriterien 4](#_Toc25132205)

[1.4. Kann-Kriterien 4](#_Toc25132206)

[1.5. Sicherheitsaspekte 4](#_Toc25132207)

[2. Projektplanung 5](#_Toc25132208)

[2.1. Netzplan 5](#_Toc25132209)

[2.2. Aufgabenverteilung 5](#_Toc25132210)

[3. Realisierung 6](#_Toc25132211)

[3.1. Raspberry-Pi 6](#_Toc25132212)

[3.1.1. Spannungsversorgung durch Autobatterie 6](#_Toc25132213)

[3.1.2. Auslesen und Glätten der Sensordaten 6](#_Toc25132214)

[3.1.3. Datenübertragung durch MQTT 8](#_Toc25132215)

[3.2. Server-Pi 9](#_Toc25132216)

[3.2.1. Installation von Mosquitto, Apache und PhP 9](#_Toc25132217)

[3.2.2. Webdesign 9](#_Toc25132218)

[4. Projektergebnisse 9](#_Toc25132219)

[4.1. Diskussion der erreichten Ziele 9](#_Toc25132220)

[4.2. Fazit 9](#_Toc25132221)

[5. Quellenverzeichnis 9](#_Toc25132222)

[6. Anhang 9](#_Toc25132223)

# 1. Projektbeschreibung

## 1.1. Kurzbeschreibung

Das Projekt beinhaltete mehrere Hauptaufgaben, welche in mehrere Teilaufgaben unterteilt sind. Diese Aufgaben führen nach und nach zum Gesamtergebnis des Boardcomputers, welcher mit einem Webinterface kommuniziert. Das Webinterface ruft die Daten in Echtzeit ab um sie grafisch darzustellen. Der Raspberry-Pi Zero wird mit einem Gyroskop mit Beschleunigungssensor an einem Fahrzeug angebracht. Das Gyroskop sendet die Daten an den Raspberry-Pi. Der Raspberry-Pi verbindet sich mit einen zuvor festgelegten Smartphone Hotspot und sendet die Daten an eine Domain mit hinterlegtem Webserver. Auf diesem Server befindet sich der SQL-Server, welcher die benötigten Daten speichert und das Webinterface, welches die Daten verarbeitet und gegeben falls ausgibt.

## 1.2. Ist-Analyse

Das Projekt an sich wurde bestimmt schon einmal durchgeführt und man kann bestimmt etwas im Internet dazu finden, aber es erneut durchzuführen ist dennoch sinnvoll. Es verbindet viele Bereiche der Informatik und Elektrotechnik miteinander und benutzt unter anderem nicht physische Verbindungen wodurch Datensicherheit auch eine wichtige Rolle spielt. Das lesen der Daten, welche vom Gyroskop gesendet werden und das verarbeiten wird mit einer maschinennahen Programmiersprache verwirklicht, wodurch man über Algorhythmen und andere Komplikationen nachdenken muss. Die Verbindung zum Webserver ist Teil der Netzwerktechnik und hat auch viel mit Datensicherheit zu tun durch z.B. Verschlüsselungen. Das Aufsetzen des Servers kann durch viele verschiedene Wege erreicht werden und der SQL-Server speichert die Daten sicher ab. Das Webinterface wird mit HTML, CSS und PHP realisiert. Alles in allem ist es ein umfangreiches und vielseitiges Projekt mit hohem Lerneffekt.

## 1.3. Soll-Kriterien

Am Ende des Projekts sollen die zuvor aufgestellten Hauptziele erreicht werden. Es sollen Daten von den Sensoren über den Raspberry-Pi ausgelesen werden. Diese Daten sollen durch einen Algorithmus geglättet werden um sie später besser verarbeiten zu können. Die Daten sollen dann durch den WLAN-Hotspot vom Smartphone über das Internet auf einen stationären Server gesendet werden. Die Daten werden dann in einer Datenbank gespeichert. Es soll außerdem noch ein Webinterface zum Auslesen und grafischen Darstellen der Daten erstellt werden. Die Datensicherung und Datensicherheit ist für uns erst einmal zweitrangig. Der Raspberry-Pi soll ebenfalls über die Motorradbatterie und eine externe Schaltung zur Spannungs- und Stromregulierung versorgt werden.

## 1.4. Kann-Kriterien

Neben der Datensicherung und der Datensicherheit kann noch eine Handy App entwickelt werden, welche die Daten in Echtzeit anzeigt. Dadurch würde die Übertragungsart zwischen dem Raspberry-Pi und dem Smartphone auf Bluetooth geändert werden und die Verbindung nur aufbauen, wenn die App geöffnet ist.

## 1.5. Sicherheitsaspekte

Die Verbindung vom Raspberry-Pi zum Smartphone ist theoretisch genug gesichert, solange nur der Raspberry-Pi sich im Hotspot befindet. Die generelle Verbindung zum Server sollte verschlüsselt sein, aber dies wird bei uns erst einmal zweitrangig gesehen. Die Datensicherheit wird erst einmal außen vorgelassen.

# 2. Projektplanung

## 2.1. Netzplan

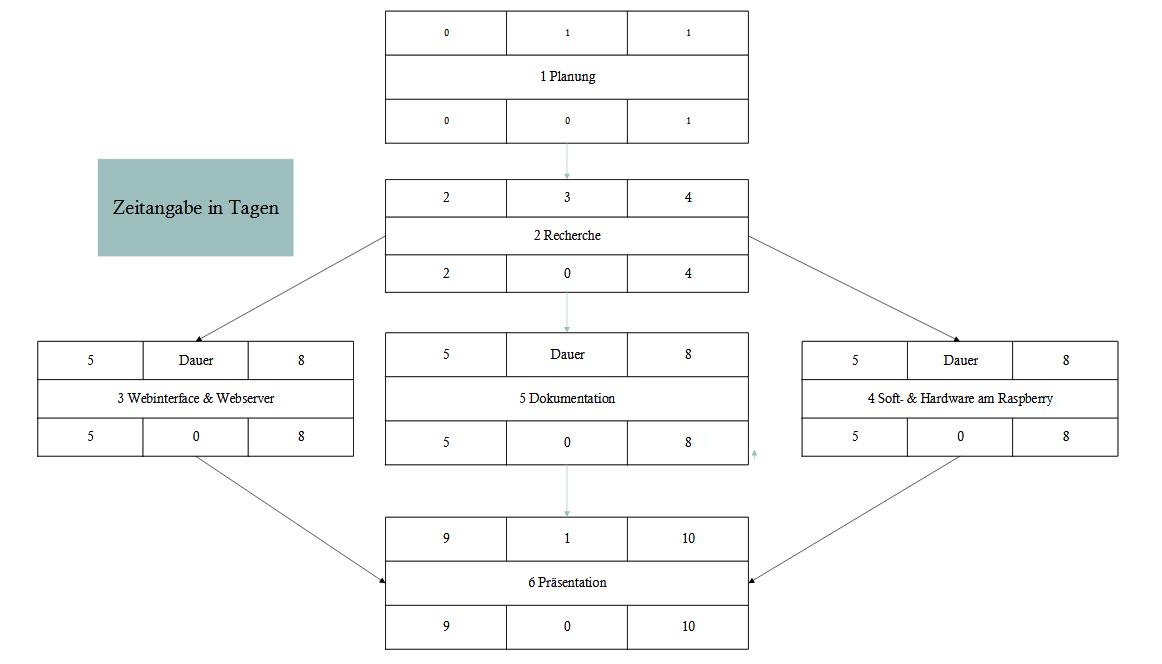


Bild 2.1.1: Netzplan

## 2.2. Aufgabenverteilung

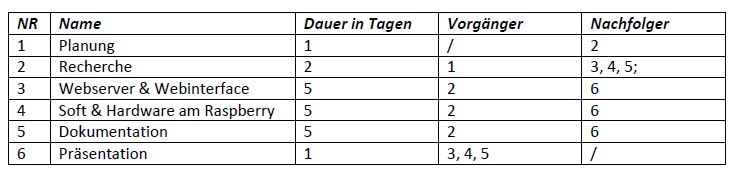


Bild 2.2.1: Tabellarische Darstellung der Aufgaben

Sven Neuenhofer: Planung, Recherche, teilw. Hardware am Raspberry und Dokumentation, Präsentation

Lasse Pittelkow: Planung, Recherche, Webserver & Webinterface, Präsentation

Lucas Wiedenfeld: Planung, Recherche, Soft & Hardware am Raspberry, Präsentation

# 3. Realisierung

## 3.1. Raspberry-Pi

### 3.1.1. Spannungsversorgung durch Autobatterie

Der Raspberry soll als erstes durch die Auto- oder Motorradbatterie versorgt werden. Dies erfolgt durch einen Spannungswandler, welcher die Spannung von der Batterie zum Raspberry-Pi auf 5V begrenzt. Diese Hauptspannungsversorgung wird an den jeweiligen Micro-USB Anschluss des Raspberry-Pi’s gelegt. Sobald die Spannung durch das Zündschloss unterbrochen wird, übernimmt eine separate Kondensatorschaltung die Spannungsversorgung für kurze Zeit. Ein GPIO des Raspberry-Pi’s wird als Überprüfung der Hauptspannung benutzt. Wenn keine Spannung am GPIO anliegt wird ein „shutdown“ Befehl ausgeführt. Die Kondensatoren halten den Raspberry-Pi ca. 30 Sekunden am Leben, so dass er sich herunterfahren kann ohne abzustürzen.

### 3.1.2. Auslesen und Glätten der Sensordaten

Das Auslesen der Sensordaten sollte ursprünglich über einen Raspberry-Pi Zero erfolgen. Der Raspberry-Pi Zero sorgte für Komplikationen, welche nicht behoben werden konnten. Aus diesem Grund werden die Daten nun über einen Raspberry-Pi 3B ausgelesen und geglättet. Das Programm auf dem Raspberry-Pi wird in Python geschrieben, da dies durch Bibliotheken und Quellen einfacher zu realisieren ist. Das Gyroskop gibt Neigungs- und Beschleunigungsdaten auf jeweils drei Achsen zurück. Es werden allerdings nur jeweils die y-Achsen und die x-Achse der Neigung benötigt. Der Sensor wird über das I²C Bussystem ausgelesen. Der Takt des Sensors (SCL = Serial Clock) wird auf den GPIO 3 gelegt. Die Daten (SDA = Serial Data) werden auf den GPIO 2 gelegt. Diese GPIO’s müssen benutzt werden, da sie das I²C Bussystem benutzen können. Die Spannungsversorgung des Sensors wird über einen 5V Pin des Raspberry-Pi’s geregelt und ein GND Pin ist ebenfalls mit dem Sensor verbunden.

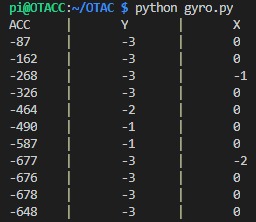


Bild 3.1.2.1: Ausgabe des Python-Programms zur Datenauffassung

Das Python Programm gibt die ACC (Acceleration = Beschleunigung) auf der y-Achse und die Neigung um die x und y-Achse aus. Die Beschleunigungswerte werden durch eine Funktion „gewichtet“ um einen geglätteten Mittelwert zu bilden. Die Formel beinhaltet zehn gewichtete Werte.

Formel:

x1\*0.05+x2\*0.05+x3\*0.06+x4\*0.07+x5\*0.08+x6\*0.1+x7\*0.12+x8\*0.15+x9\*0.15+xn\*0.17= gewichteter Mittelwert

Der Wert x1 ist der älteste Übergabewert und xn ist der Aktuelle. Bei jedem Aufruf werden die Werte um eine Position nach hinten verschoben.

Die von der Hardware gegebene Beschleunigung ist standardmäßig nicht auf null kalibriert. Deshalb wird drei Sekunden nach dem Start des Programms der Wert auf null gesetzt. Trotzdem weichen die Werte um +/- 100 ab.

### 3.1.3. Datenübertragung durch MQTT

MQTT (Message Queuing Telemetry Transport) ist ein Übertragungssystem. Es besteht aus „Clients“ und „Brokern“. Clients können senden und zuhören. Ein Client sendet eine Nachricht mit einem Topic an einen Broker, welcher diese Nachricht an alle Clients weiterleitet die dem Topic zuhören. Ein Client ist der Raspberry-Pi, welcher am Motorrad befestigt wird und die Sensordaten ausliest. Er agiert nur als Sender. Der als Server arbeitende Pi agiert als Broker und zuhörender Client. Der Broker wird durch die Software „MQTT Mosquitto“ erstellt und verwaltet.

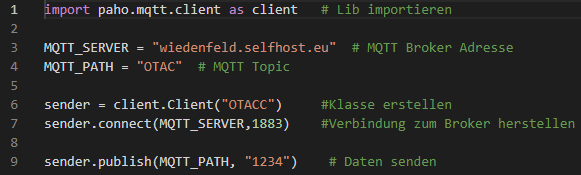


Bild 3.1.3.1: Codeschnipsel von MQTT aus dem Python Programm

Diese Codeschnipsel wurden aus dem Programm entnommen und kommentiert. Die erste Zeile importiert die MQTT-library um die entsprechenden Befehle verwenden zu können. Der MQTT\_Server wird in Zeile drei festgelegt. Die Domain war vorher schon vorhanden und wurde nicht im Ramen dieses Projekts angelegt. Im Beispiel heißt die Variable für das Topic „MQTT\_Path“. In Zeile sechs wird der Pi als Client gesetzt und im wird der Name „OTACC“ gegeben. Danach verbindet sich der Client mit dem Broker, welcher zuvor auf dem Server vorhanden sein muss. Zuerst wird die Adresse angegeben, danach folgt der jeweilige Port. Als letztes werden in Zeile neun die Daten an den Broker gesendet um von dort aus an alle zuhörenden Clients zu seden.

## 3.2. Server-Pi

### 3.2.1. Installation von Mosquitto, Apache und PhP

Vor der eigentlichen Installation der Pakete sollte ein „sudo apt-get update“ und ein „sudo apt-get upgrade“ durchgeführt werden um sicher zu gehen, dass alles ohne Probleme installiert wird.

Das Programm Mosquitto musste zuvor installiert werden um den Server als Broker zu benutzen. Dies erfolgt durch den Befehl „sudo apt-get install mosquitto mosquitto-client“. Danach muss der Broker so eingerichtet werden, dass er beim Serverstart ebenfalls startet. Der Befehl „sudo systemctl enable mosquitto.service“ erledigt dies. Der Status kann durch den Befehl „sudo service mosquitto status“ überprüft werden.

Apache installiert man durch den Befehl „sudo apt-get install apache2“.

PhP muss auf dem Server installiert werden, dadurch wird der Befehl „sudo apt-get install php“ nicht reichen. Man muss PhP durch den Befehl „sudo apt-get install php libapache2-mod-php“ installieren. Dieser Befehl installiert PhP auf dem Pi und auf dem Apache Server, damit dieser PhP übersetzen und anwenden kann.

### 3.2.2. Webdesign und empfang von Daten durch MQTT

# 4. Projektergebnisse

## 4.1. Diskussion der erreichten Ziele

## 4.2. Fazit

# 5. Quellenverzeichnis

# 6. Anhang