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

1.1. Kurzbeschreibung 3

1.2. Ist-Analyse 3

1.3. Soll-Kriterien 3

1.4. Kann-Kriterien 4

1.5. Sicherheitsaspekte 4

2. Projektplanung 5

2.1. Netzplan 5

2.2. Aufgabenverteilung 5

3. Realisierung 6

3.1. Raspberry-Pi 6

3.1.1. Spannungsversorgung durch Autobatterie 6

3.1.2. Auslesen und Glätten der Sensordaten 6

3.1.3. Datenübertragung durch MQTT 7

3.2. Server-Pi 8

3.2.1. Installation von Mosquitto, Apache und PhP 8

3.2.2. Webdesign und empfang von Daten durch MQTT 9

4. Projektergebnisse 9

4.1. Diskussion der erreichten Ziele 9

4.2. Fazit 9

5. Quellenverzeichnis 10

1. Projektbeschreibung

## 1.1. Kurzbeschreibung

Das Projekt beinhaltet mehrere Hauptaufgaben, welche in Teilaufgaben unterteilt sind. Diese Aufgaben führen nach und nach zum Gesamtergebnis des Boardcomputers, welcher mit einem Webserver kommuniziert. Der Raspberry-Pi wird mit einem Gyroskop und Beschleunigungssensor an einem Fahrzeug angebracht. Der Raspberry-Pi verbindet sich mit einem Smartphone Hotspot und sendet die gemessenen Daten übers Internet an einen Webserver. Auf diesem Server werden die Daten gespeichert und ein Webinterface gibt die daten aus.

## 1.2. Ist-Analyse

Das Projekt 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, PhP und JavaScript 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 und dieser Passwortgesichert ist. Die generelle Verbindung zum Server sollte verschlüsselt sein, aber dies ist zweiträngig. Die Datensicherheit wird erst einmal außen vorgelassen.

2. Projektplanung

## 2.1. Netzplan

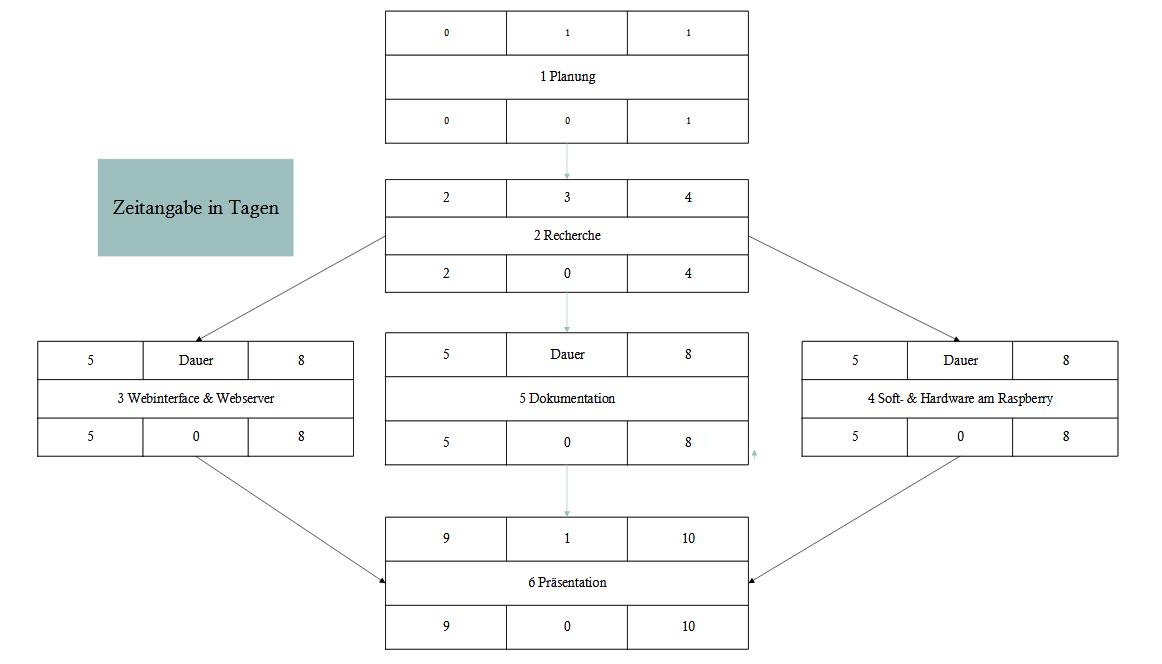


Bild 2.1.1: Netzplan

## 2.2. Aufgabenverteilung

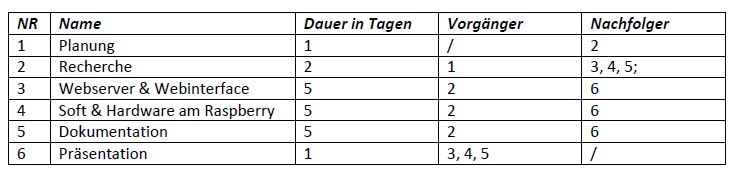


Bild 2.2.1: Tabellarische Darstellung der Aufgaben

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 durch die Auto- oder Motorradbatterie versorgt werden. Dies erfolgt durch einen Spannungswandler, welcher die Spannung von der Batterie auf 5V begrenzt (Versorgungsspannung des Pi). Diese wird an den jeweiligen 5V- und Masse Anschluss des Raspberry-Pi gelegt. Sobald die Spannung durch das Zündschloss unterbrochen wird, übernimmt eine separate Kondensatorschaltung die Spannungsversorgung für kurze Zeit. Ein GPIO des Raspberry-Pi wird als Überprüfung der Hauptspannung benutzt. Wenn keine Spannung am GPIO anliegt wird ein „shutdown“ Befehl ausgeführt. Die Kondensatoren erhalten die Spannung am Raspberry-Pi ca. 30 Sekunden lang, so dass er sicher herunterfahren kann.

### 3.1.2. Auslesen und Glätten der Sensordaten

Das Auslesen und Glätten der Sensordaten erfolgt über einen Raspberry-Pi 3B. Das Programm auf dem Raspberry-Pi ist in Python geschrieben. 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 nur sie das I²C Bussystem unterstützen. Die Spannungsversorgung des Sensors wird über einen der 3,3V Pins und einen der Masse (GND) Pins des Raspberry-Pi geregelt.

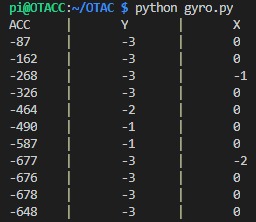


Bild 3.1.2.1: Ausgabe des Python-Programms zur Datenerfassung

Das Python Programm gibt die Beschleunigung (ACC) in Richtung 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

Message Queuing Telemetry Transport (MQTT) 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, der am Fahrzeug befestigt ist und die Sensordaten ausliest. Er agiert nur als Sender. Der als Server agiert als Broker und zuhörender Client. Als Broker arbeitet das Programm Mosquitto.

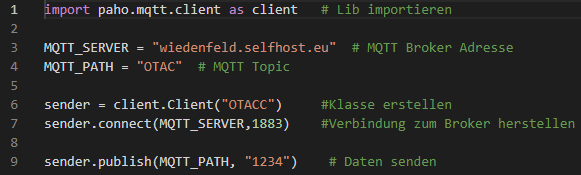


Bild 3.1.3.1: MQTT-Codeschnipsel aus dem Python Programm

Diese Codeschnipsel wurden aus dem Programm entnommen und kommentiert. Die erste Zeile importiert die MQTT-library um die entsprechenden Befehle verwenden zu können. Die Broker-Adresse wird in Zeile drei festgelegt. 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. Zuerst wird die Adresse angegeben, danach folgt der Port. Als letztes werden in Zeile neun die Daten an den Broker gesendet um von dort aus an alle zuhörenden Clients zu seden. Um schnell genug Daten zu senden wird nicht drauf geachtet ob diese auch wirklich ankommen. Der Sender bekommt keine Rückmeldung ob seine Nachricht vom Empänger erhalten wurde und es wird kein Handshake ausgeführt (QoS 0, Fire-and-Forget).

## 3.2. Server

### 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 muss 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 wird durch den Befehl „sudo apt-get install apache2“ installiert.

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

Die gestaltung des Webinterfaces ist so simpel wie möglich gehalten. Benötigt wird nur ein Login, welcher mit PhP realisiert wird und eine Hauptseite welche einen Graph zeigt der mit Javascript erstellt wird. Auch das Empfangen der Daten funktioniert über ein PhP-Skript. Diese werden dann in JavaScript übergeben und direkt zu einem Graph verarbeitet. Das Webinterface speichert und ruft Daten von der Datenbank ab. Beim Login werden verschlüsselte Daten abgerufen, unter anderem ein mit Hash verschlüsseltes Passwort.

4. Projektergebnisse

## 4.1. Diskussion der erreichten Ziele

Der Raspberry-Pi ist fast voll funktionstüchtig. Er kann über eine Autobatterie betrieben werden und das Programm überwacht die Spannungsversorgung und sendet Daten über MQTT an den Broker. Bei diesen Projektaufgaben gab es keinerlei Probleme, aber für die Spannungsversorgung nach dem Abschalten der Hauptstromquelle fehlen uns zwei Kondensatoren, welche den Raspberry-Pi noch ca. 30 Sekunden am Leben halten sollen. Dadurch überwacht der Pi jetzt zwar die Spannung, aber sobald diese abgeschnitten wird, wird der Pi sofort ausgeschaltet.

Der Server hingegen ist nicht so geworden, wie wir ihn uns vorgestellt hatten. Der Server besitzt einen Apache2-Server. Er besitzt ebenfalls PhPMyAdmin und einen SQL-Server. Dies ist auch soweit ganz gut gelaufen. Der Server bereitete später allerdings einige Probleme, da er die MQTT Bibliothek nicht finden konnte. Aus Verzweiflung haben wir den Server neu aufgesetzt, aber dies kostete uns viel Zeit, weswegen das Webinterface zwar einen funktionierenden Graphen besitzt, aber keine Echtzeitdaten erfassen und darstellen kann. Im Graphen werden stattdessen zufällige Daten angezeigt um die Funktionalität darzustellen.

## 4.2. Fazit

Wir sind uns alle einig, dass um ein solches Projekt ordentlich durchzuführen mehr Zeit benötigt wird. Wir lernten viel, da dieses Projekt fast jedes Fach vereint und damit fast jeden Bereich abdeckt. Unter anderem lernten wir viel über MQTT und den Apache Server. Es hat viel Spaß gemacht, da wir an einem Projekt arbeiten konnten was einige von uns auch privat interessiert. Deshalb werden wir dieses Projekt vielleicht auch noch in unserer Freizeit weiterführen.

5. Quellenverzeichnis

<https://pypi.org/project/paho-mqtt/>, <https://tutorials-raspberrypi.de/rotation-und-beschleunigung-mit-dem-raspberry-pi-messen/>, <https://github.com>, <https://StackOverflow.com>, <http://mqtt.org>, <https://phpmyadmin.net>, <https://canvasjs.com/php-charts/>;