|  |
| --- |
| Elektronikschule Tettnang |
| Projektdokumentation |
| Aufbau einer IoT-Anwendung zur Verarbeitung und Visualisierung von Sensordaten |

|  |
| --- |
| Niklas Schraff, Nils Simon EFI222  06.05.2024 – 08.05.2024 |



# **Inhalt:**

**I Abkürzungsverzeichnis**

**1. Ausgangssituation**

1.1 Projektziele

1.2 Teilaufgaben

1.3 Projektumfeld, Prozessschnittstellen

**2. Ressourcen- und Ablaufplanung**

2.1 Terminplanung

2.2 Personalplanung

2.3 Sachmittelplanung

2.4 Kostenplanung

**3. Durchführung- und Auftragsbearbeitung**

3.1 Aufbau des Mikrocontrollers

3.2 Programmierung des Microcontrollers

3.2.1 Verbindung des WLAN-Netzwerks

3.2.2 Verbindung des MQTT-Brokers

3.2.3 Ansteuerung der Sensoren

3.2.4 Senden der MQTT-Nachrichten

3.3 Konfiguration der Datenbank

3.4 NodeJs

3.4.1 Verbinden von MQTT mit der MySQL Datenbank

3.4.2 Abrufen und bereitstellen der Daten an das Frontend

3.4.3 Registrierungs- und Login System

3.5 Frontend Website

3.5.1 Anzeigen der Datendiagramme

3.6 Installation und Konfiguration des Raspberry Pi

**4. Knackpunkte, Abweichungen, Anpassungen, Entscheidungen**

**5. Qualitätssicherung**

5.1 Versionsverwaltung mit Git

5.2 Testfahrt

**6. Projektergebnisse**

6.1 Soll-Ist-Vergleich

6.2 Abnahme

6.3 Bewertung (Fazit, Ausblick)

**III Verzeichnis der Arbeits-/Hilfsmittel**

**IV Abbildungs- und Tabellenverzeichnis**

**V Anhang**

**VI Eidesstattliche Erklärung**

**I Abkürzungsverzeichnis**

|  |  |
| --- | --- |
| I2C | Inter-Integrated Circuit (serieller Datenbus) |
| SDA | Serial Data (I2C) |
| SCL | Serial Clock (I2C) |
| DHT11 / 22 | Digital-output relative humidity & temperature |
| BMP180 | Barometric pressure |
| Js | Javascript |
| MQTT | Message Queuing Telemetry Transport |
| MySQL | My Structured Query Language |
| PM2 | Process Manager |

**1. Ausgangsituation**

In diesem Projekt geht es um das Umsetzen eines Pilotprojekts zur Ermittlung von Höhenverläufen für das Hochwasserrisikomanagement der Stadt Tettnang. Dazu sollten verschiedene Sensoren Höhendaten liefern damit diese Verarbeitet und visualisiert werden können um später genaue Karten von potentiellen Überschwemmungsgebieten zu erstellen.

* 1. **Projektziele**

Ziel war es mithilfe von Sensoren, die über einen Mikrocontroller angesteuert werden, Höhendaten, Temperatur, Luftdruck sowie Luftfeuchtigkeit zu erfassen und diese in eine Datenbank zu speichern. Anschließend sollten diese auf einer Website übersichtlich und klar mittels Diagramme visualisiert werden.

* 1. **Teilaufgaben**

Das Projekt wurde in folgende Teilaufgaben unterteilt, um die Durchführung strukturierter und einfacher zu machen:

* Beschaffung der Hardware
* Aufbau des Mikrocontrollers und der Sensoren
* Konfiguration und Programmierung des Mikrocontrollers
* Aufsetzen eines Raspberry Pi als Web- und Datenbankserver
* Konfiguration der Datenbank
* Konfiguration des Webservers
* Entwickeln eines Server Backends zur Bereitstellung der Daten
* Visualisierung der Daten auf einer Website
* Durchführung einer Testfahrt
* Erstellung eines Angebots nach erfolgreichem Abschluss

**1.3 Projektumfeld und Schnittstellen**

Das Projekt wurde von der Stadt Tettnang über die Elektronikschule in Auftrag gegeben, um dann von einer Gruppe aus zwei Schülern einer Klasse umgesetzt zu werden.

Als Räumliches Umfeld wurde der Raum A.005 der Elektronikschule Tettnang genutzt sowie für die Testfahrt ein Privatfahrzeug. In diesem Raum wurde uns ein WLAN-Netzwerk sowie Stromversorgung für all unsere Geräte zu Verfügung gestellt.

In der jeweiligen Gruppe wurde dann ein Gruppenleiter ernannt, der für den Austausch mit anderen Gruppen über Probleme und Lösungen bei der Durchführung des Projekts zuständig war. Außerdem standen die zwei Lehrkräfte Herr Rauschmaier für IT-Themen sowie Frau Wattenbach für BWL-Themen als Ansprechpartner zur Verfügung.

**2.** **Ressourcen- und Ablaufplanung**

Im Folgenden werden Pläne zur Ressourcen- und Ablaufplanung dargestellt und erklärt.

**2.1 Terminplanung**

Das Projekt umfasste einen 3 Tageszeitraum vom 6. bis zum 8. Mai 2024.

**Tag 1** (6. Mai) umfasste die Planung des Projekts sowie der Aufbau und die Konfiguration des Mikrocontrollers.

**Tag 2** (7.Mai) umfasste das Aufsetzen und Konfiguration der Datenbank und des Webservers, sowie die Programmierung des Mikrocontrollers.

**Tag 3** (8.Mai) Umfasste das Entwickeln des Back-End sowie die Visualisierung der Daten. Außerdem wurde eine Testfahrt durchgeführt und das Projekt abgenommen.

Der genaue terminliche Ablauf ist dem Netzplan (Abb. 6) zu entnehmen.

**2.2** **Personalplanung**

|  |  |  |
| --- | --- | --- |
| **Aufgabe** | **Verantwortliche Person/en** | **Stunden** |
| Hardware-Beschaffung | Nils Simon, Niklas Schraff | 0,5 h |
| Mikrocontroller aufbauen | Niklas Schraff | 0,5h |
| Raspberry Pi aufsetzen | Nils Simon | 1h |
| Microcontroller programmieren | Niklas Schraff | 4h |
| Datenbank aufsetzen | Nils Simon | 1h |
| Webserver aufsetzen | Nils Simon | 0,5h |
| Backend schreiben | Nils Simon | 5h |
| Skript um MQTT und Datenbank zu Verbinden schreiben | Niklas Schraff | 2h |
| Frontend schreiben | Nils Simon, Niklas Schraff | 3h |
| Website auf Webserver ziehen | Nils Simon | 1h |
| Testfahrt durchführen | Nils Simon, Niklas Schraff | 1h |
| Angebot erstellen | Niklas Schraff | 2h |

In der folgenden Tabelle sind die verschiedenen Teilaufgaben mit den Verantwortlichen Personen aufgelistet. Die Verantwortlichkeiten wurden je nach Erfahrung in ihrem jeweiligen Fachgebiet zugeteilt.

Tab. Nr. 1: Personalplanung

**2.3 Sachmittelplanung**

Die folgende Sachmittelplanung listet benötigte Ressourcen getrennt in Hardware und Software auf. Hardware, die nicht verfügbar war, wurde von den Lehrkräften bereitgestellt. Zudem haben wir ausschließlich freiverfügbare Software genutzt, um die nötigen Programme zu entwickeln.

|  |  |  |  |
| --- | --- | --- | --- |
| **Hardware** | **Menge** | **Beschreibung** | **Vorhanden?** |
| ESP8266 Mikrocontroller | 1 | Mikrocontroller zur Abfrage der Sensordaten | Muss beschafft werden |
| Raspberry Pi | 1 | Zentraler Server für alle Services | Muss beschafft werden |
| Netzteil Raspberry Pi | 1 | Stromversorgung Raspberry Pi | Muss beschafft werden |
| Netzteil ESP8266 | 1 | Stromversorgung  Mikrocontroller | Muss beschafft werden |
| Sensor DHT22 | 1 | Sensor zur Messung von Luftdruck und Temperatur | Muss beschafft werden |
| Sensor BMP180 | 1 | Sensor zur Höhenmessung | Muss beschafft werden |

Tab. Nr. 2: Hardware-Sachmittelplanung

|  |  |  |
| --- | --- | --- |
| **Software** | **Beschreibung** | **Vorhanden?** |
| Visual Studio Code | Code Editor zur Programmierung | Ja, Freeware |
| MQTT | IoT Netzwerkprotokoll | Ja, Freeware |
| NodeJs | Server Software zur Datenübertragung | Ja, Freeware |
| Ubuntu Server22.04LTS | Betriebssystem für Raspberry Pi | Ja, Freeware |
| nginx | Webserver Software | Ja, Freeware |
| MySQL | Datenbank Software | Ja, Freeware |
| Micro Python | ESP8266Programmiersprache | Ja, Freeware |

Tab. 3 Software-Sachmittelplanung

**2.4 Kostenplanung**

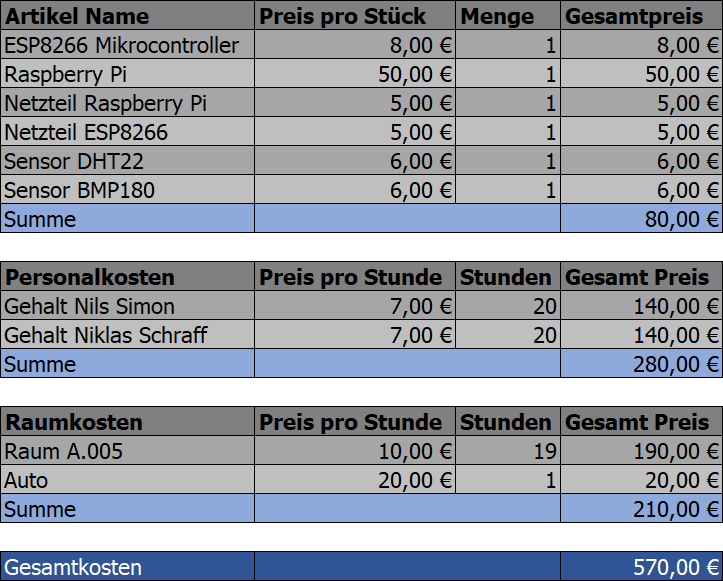
****Geplante anfallende Kosten für das Projekt werden hier ausführlich dargestellt. Dies beinhaltet auch Stundensätze und Raumkosten für die Arbeitszeit.

Abb. 1: Kostenplanung

**3. Durchführung- und Auftragsbearbeitung**

Im Folgenden wird die Durchführung, das Bearbeiten der einzelnen Teilaufgaben, sowie der genaue Ablauf des Projektes beschrieben.

**3.1 Aufbau des Mikrocontrollers**

Als Steuereinheit für die Sensoren kam ein ESP8266 Mikrocontroller zum Einsatz. Dieser eignete sich perfekt aufgrund seiner kompakten Bauform sowie den flexiblen Schnittstellen, welche unteranderem digitale und analoge Ein- und Ausgänge sowie einen I2C- Bus beinhalten.

Damit wurden die folgenden Sensoren angeschlossen und angesteuert:

* DHT11: Sensor zur Messung der Umgebungstemperatur sowie der Luftfeuchtigkeit
* BMP180: Sensor zur Messung der Höhe über dem Meeresspiegel sowie des Luftdrucks

Beide Sensoren wurden mithilfe von Stecksätzen miteinander verbunden, wodurch sich Löten vermeiden ließ.

**3.2 Programmierung des Mikrokontrollers**

Die Programmierung des Mikrocontrollers erfolgte in der Programmiersprache „C“ aufgrund der Hardwareunabhängigkeit sowie dem kompakteren Quellcode. Außerdem wurden verschiedene dritt Anbieter Bibliotheken. Außerdem wurde „Visual Studio Code“ inklusive des Plug-Ins „Platform IO“ als Entwicklungsumgebung verwendet.

**3.2.1 Verbindung des WLAN-Netzwerks**

Um Daten auf dem Mikrokontroller aus dem Internet senden und empfangen zu können, wurde dieser mit einem WLAN-Netzwerk verbunden. Hierfür wurde die Bibliothek „ESP8266 WiFi“ verwendet welche mit dem Übergeben von SSID und Passwort eine Verbindung aufbaut. Ein Access-Point wurde im Klassenraum von den Lehrkräften bereitgestellt.

**3.2.2 Verbindung des MQTT-Brokers**

Die Aufgabenstellung verlangte die Verwendung des Netzwerkprotokolls MQTT. Hierfür muss eine Verbindung zu einem MQTT-Broker hergestellt werden, welcher für die Verteilung der Daten zuständig war. Dies erfolgte über die frei verfügbare Bibliothek „PubSubClient“ mitwelcher wir durch Übergeben der URL sowie des Ports eine Verbindung herstellen konnten. Als MQTT-Broker wurde der öffentliche Broker von HiveMQ aufgrund der guten Erreichbarkeit genutzt.

**3.2.3 Ansteuerung der Sensoren**

Um die Sensoren über den Mikrocontroller anzusteuern und Messwerte abzufragen wurden die frei-verfügbaren Bibliotheken „Adafruit DHT sensor library“ für den DHT11 Sensor sowie „Adafruit BMP085 Library“ für den BMP180 Sensor verwendet.

Die Sensoren waren über folgende Schnittstellen angeschlossen:

* DHT11: Digital Pin 02 als Daten Pin
* BMP180: I2C über SDA (Pin 04) und SCL (Pin 05)

Das Erfassen der Messwerte erfolgte alle 60 Sekunden über die von den Bibliotheken bereitgestellte Funktionen.

**3.2.4 Senden der MQTT-Nachrichten**

Die erfassten Daten wurden zur weiteren Verarbeitung auf das MQTT-Topic „EST/EFI222/NSNS“ gesendet. Der Inhalt der Versendeten Nachrichten bestand aus einem String in der JSON-Notation, welcher durch die frei-verfügbaren Bibliothek „Arduino JSON“ erzeugt wird. Hierfür werden die Messwerte in die jeweiligen Schlüssel des JSON-Objekts geschrieben. Das Senden der Nachrichten erfolgte alle 60 Sekunden.

Beispiel des Inhalts einer Nachricht:

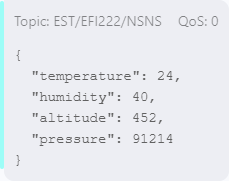


Abb. 2 MQTT-Nachricht

**3.3 Konfiguration der Datenbank**

Aufgabe war es die gemessenen Daten in einer Datenbank zu speichern. Hier haben wir uns für eine MySQL Datenbank entschieden, da wir diese bereits ausführlich im Unterricht besprochen haben und sich diese sehr einfach in das geplante NodeJs Backend einbinden lässt. In der Datenbank wurden dann folgende Spalten aufgesetzt:

* Timestamp: Diese Spalte speichert mittels der MySQL „now()“ Funktion den Zeitstempel zu welchem der Messeintrag aufgezeichnet wurde.
* Temperature: Diese Spalte speichert die gemessene Temperatur.
* Humidity: Hier wird die gemessene Luftfeuchtigkeit abgespeichert
* Pressure: Speichert den derzeitigen Luftdruck
* Altitude: Hier ist die Höhe gespeichert

Zusammen mit dem Zeitstempel und den gemessenen Werten ist es möglich die Daten übersichtlich dargestellt abzurufen und einfach zu analysieren.

**3.4 NodeJs**

Für sowohl das Backend als auch als Webserver wird NodeJs verwendet, da es effizient alle Anfragen verarbeiten kann und die Möglichkeit bietet APIs bereitzustellen, welche verwendet werden, um Daten einfach an das Frontend übertragen zu können.

Wir verwenden zusätzlich das Express Framework, um das Routing der verschiedenen Seiten möglichst einfach zu gestalten.

Allgemein bietet NodeJs eine elegante und effiziente Lösung für alle Full-Stack-Anwendungen, welche sowohl Front- als auch Backend Logik verwalten.

**3.4.1 Verbinden von MQTT mit der MySQL Datenbank**

Um die MQTT-Nachrichten in die MySQL Datenbank zu schreiben, wurde im Backend des NodeJs Servers eine Verbindung zwischen den beiden Punkten erstellt. Zuerst verbindet sich der Server mit dem MQTT-Broker und abonniert das Topic, auf welches der Mikrocontroller die Messdaten sendet. Sobald eine Nachricht empfangen wird, werden die Daten aufgeteilt und in eigene Variablen geschrieben. Danach werden diese Daten mittels eines einfachen SQL-Statements (Abb. 5) in die Datenbank unter der Tabelle „measurements“ eingefügt.

**3.4.2 Abrufen und bereitstellen der Daten an das Frontend**

Um die Daten im Frontend abrufen zu können, wurde ein API Endpoint im Backend eingerichtet. Im Code des Endpoints (Abb. 9) werden die neuesten 60 Einträge nach Eintrags-ID sortiert abgerufen und als JSON zurückgegeben. Im Frontend rufen wir einfach diesen Endpoint auf und schreiben dann die empfangenen Daten in eine Variable.

**3.4.3 Registrierungs- und Login-System**

Um das Frontend abzusichern wurden eine Registrierungsform und ein Login erstellt. Beim Registrieren gibt der Benutzer relevante Daten wie Name, E-Mail und Passwort, ein. Das Passwort bekommt dann ein generiertes Salt angehängt und wird dann über „BCrypt“, eine Bibliothek zum „hashen“ von Passwörtern, verschlüsselt und in die Datenbank geschrieben. Beim Login passiert mit dem eingegebenen Passwort das gleiche und der dann erstellte Hash wird dann mit dem aus der Datenbank verglichen. Bei einer Übereinstimmung wird der Nutzer eingeloggt.

**3.5 Frontend der Website**

Das Frontend besteht aus einer Navigationsleiste, einer Landingpage, zwei Seiten für die Graphen, der Registrierungs- und Login Seite sowie einer Accountseite.

Im nicht eingeloggten Zustand sind nur die Landingpage und die Login Seite verfügbar. Nachdem der Nutzer sich einloggt, hat er auch Zugriff auf alle weiteren Seiten.

Die Navigationsleiste ist da, um sich zwischen den Verschiedenen Seiten zu bewegen. Auch diese passt sich an, je nachdem ob der Nutzer eingeloggt ist oder nicht.

**3.5.1 Anzeigen der Datendiagramme**

Sobald der Nutzer einen Account erstellt und sich eingeloggt hat, bekommt er Zugriff auf die zwei Seiten, welche die Graphen beinhalten. Auf einer Seite findet man einen großen Graphen (Abb. 3) vor, in welchem alle 4 Liniendiagramme zu sehen sind. Einzelne Linien können hier ein- und ausgeblendet werden. Auf der anderen Seite sind 4 kleine Graphen (Abb. 4) zu sehen, welche jeweils ein Liniendiagramm anzeigen. Die Auswahl zwischen den beiden Ansichten liegt dann im eigenen Ermessen des Nutzers.

**3.6 Installation und Konfiguration des Raspberry Pi**

Auf dem Raspberry Pi wurde Ubuntu Server 22.04 installiert, da dieses Betriebssystem alle Voraussetzungen für eine nahtlose Verwendung erfüllt und auch von den Entwicklern weiterentwickelt und unterstützt wird. Weiterhin wurden auf dem Pi alle benötigten Programme installiert. Hierzu zählen NodeJs und PM2. NodeJs wird, wie bereits erwähnt, für das Backend und den Webserver verwendet. PM2 ist ein Monitoring Programm, welche das Verwalten von NodeJs Servern erleichtert und übersichtlicher gestaltet. Dazu zählen Vorzüge wie das Automatische neustarten von Webservern, falls diese abgestürzt sind, als auch das einfache Einsehen von Logs oder Fehlermeldungen des Servers.

**4. Knackpunkte, Abweichungen, Anpassungen, Entscheidungen**

Durch eine falsche Version der ChartJs Bibliothek haben einige Funktionen der Diagramme nicht ordnungsgemäß funktioniert. Das Problem wurde behoben durch Ersetzen der URL mit der fehlerhaften Version. Weiterhin kam es beim Aufsetzen der Website auf dem Raspberry Pi zu einem Fehler beim Datenbankzugriff. Grund hierfür war eine falsche Nutzerberechtigung des Root Zugangs der Datenbank. Die Lösung war einen neuen mit den korrekten Rechten anzulegen.

Abgewichen vom ursprünglichen Plan sind wir mit der Verwendung eines DHT11 anstatt eines DHT22 Sensors zur Messung der Temperatur sowie die Luftfeuchtigkeit.

Entschieden haben wir uns für den DHT11 aufgrund seiner besseren Kompatibilität mit der Mikrocontroller Bibliothek zur Ansteuerung.

**5. Qualitätssicherung**

**5.1 Versionsverwaltung mit Git**

Über die Versionsverwaltungssoftware Git ist es möglich, dass mehrere Entwickler gleichzeitig an Code arbeiten ohne, dass sie sich gegenseitig ihre Änderungen überschreiben. Hierfür spaltet sich jeder Entwickler seinen eigenen Branch (Zweig) ab, und arbeitet auf diesem. Ein Branch ist nur eine Kopie des Source-Codes. Sobald der Entwickler fertig mit seinen Änderungen ist, schiebt er die Änderungen wieder in den Haupt-Branch. Git vergleicht hierbei nur die Änderungen zum Zeitpunkt der Abspaltung des Branches. So wird genau dokumentiert, wer wann was verändert hat. Alle Änderungen sind also immer einsehbar.

**5.2 Testfahrt**

Zum umfangreichen Testen des aufgebauten Projekts beinhaltete die Aufgabenstellung eine 60-minütige Testfahrt mit einem Privat PKW, welche über Straßen mit möglichst großem Höhenunterschied führte. Diese wurde erfolgreich durchgeführt und es wurden umfangreich Daten gesammelt und visualisiert.

Die Testfahrt brachte eine hohe Zuverlässigkeit sowie Genauigkeit unserer Lösung als Ergebnis hervor und bewies uns die praktische Einsatzmöglichkeit des Projekts.

**6. Projektergebnisse**

**6.1 Soll-Ist-Vergleich**

Alle vorgegebenen Aufgaben wurden erfolgreich abgeschlossen. Optionale Aufgaben wurden aufgrund von Zeitüberschuss teilweise implementiert.

|  |  |  |
| --- | --- | --- |
| **Soll:** | **Ist:** | **Abweichung** |
| Mikrocontroller und Sensoren zusammenbauen | Mikrocontroller und Sensoren erfolgreich zusammengebaut und getestet | Keine Abweichung |
| Mikrocontroller programmiert | Mikrocontroller erfolgreich programmiert | Keine Abweichung |
| Datenbank eingerichtet | Datenbank erfolgreich eingerichtet | Keine Abweichung |
| Daten können in Datenbank geschrieben werden | Daten können korrekt in die Datenbank übertragen werden | Keine Abweichung |
| Website aufgesetzt | Website mit Login-System aufgesetzt | Optionales Login-System implementiert |
| Graphen in die Website implementiert | Graphen sind auf der Website abrufbar | Optionale 4 Graphenansicht implementiert |
| Daten können in Graphen visualisiert werden:  -Höhe | Höhe, Luftdruck, Temperatur und Luftfeuchtigkeit visualisiert | Optional Daten hinzugefügt |
| Testfahrt für Testdaten wurde durchgeführt | Testfahrt nach Plan abgeschlossen | Keine Abweichung |

Tab. 4: Soll-Ist-Vergleich

**6.2 Abnahme**

Die Abnahme verlief reibungslos und alle Aufgaben wurden erfüllt, siehe Abnahmeprotokoll (Abb. 8).

**6.3 Bewertung (Fazit, Ausblick)**

Der Ablauf des Projekts ging überwiegend reibungslos von statten, alle Aufgaben konnten zügig und korrekt erledigt werden. Kleinere Probleme wurden schnell gelöst und durch die gute Kommunikation innerhalb sowie außerhalb des Teams schnell gelöst. Unsere Lösung wäre sinnvoll für eine Ausweitung des Pilotprojekts auf die Flussläufe Tettnangs.

**III Verzeichnis der Arbeits- und Hilfsmittel**

Informationsquellen:

* <https://en.wikipedia.org/wiki/Node.js>
* <https://ejs.co/>
* <https://en.wikipedia.org/wiki/Cascading_Style_Sheets>
* <https://en.wikipedia.org/wiki/MySQL>
* <https://www.atlassian.com/git>
* <https://www.npmjs.com/package/pm2>
* <https://devarounder.de/der-bmp180-sensor-und-die-verwendung-mit-einem-esp32-oder-esp8266/>
* <https://randomnerdtutorials.com/esp32-dht11-dht22-temperature-humidity-sensor-arduino-ide/>
* <https://mylime.info/informatik/>
* <https://platformio.org/>
* <https://mqttx.app/>
* <https://www.chartjs.org/>

Genutzte Software/Tools:

* <https://code.visualstudio.com/>
* <https://platformio.org/>
* <https://nodejs.org/en>
* <https://ubuntu.com/download/server>
* <https://www.chartjs.org/>
* <https://getbootstrap.com/>

Bibliotheken:

* <https://registry.platformio.org/libraries/adafruit/DHT%20sensor%20library>
* <https://registry.platformio.org/libraries/adafruit/Adafruit%20Unified%20Sensor>
* <https://registry.platformio.org/libraries/knolleary/PubSubClient>
* <https://registry.platformio.org/libraries/adafruit/Adafruit%20BMP085%20Library>
* <https://registry.platformio.org/libraries/adafruit/Adafruit%20BusIO>
* <https://registry.platformio.org/libraries/bblanchon/ArduinoJson>

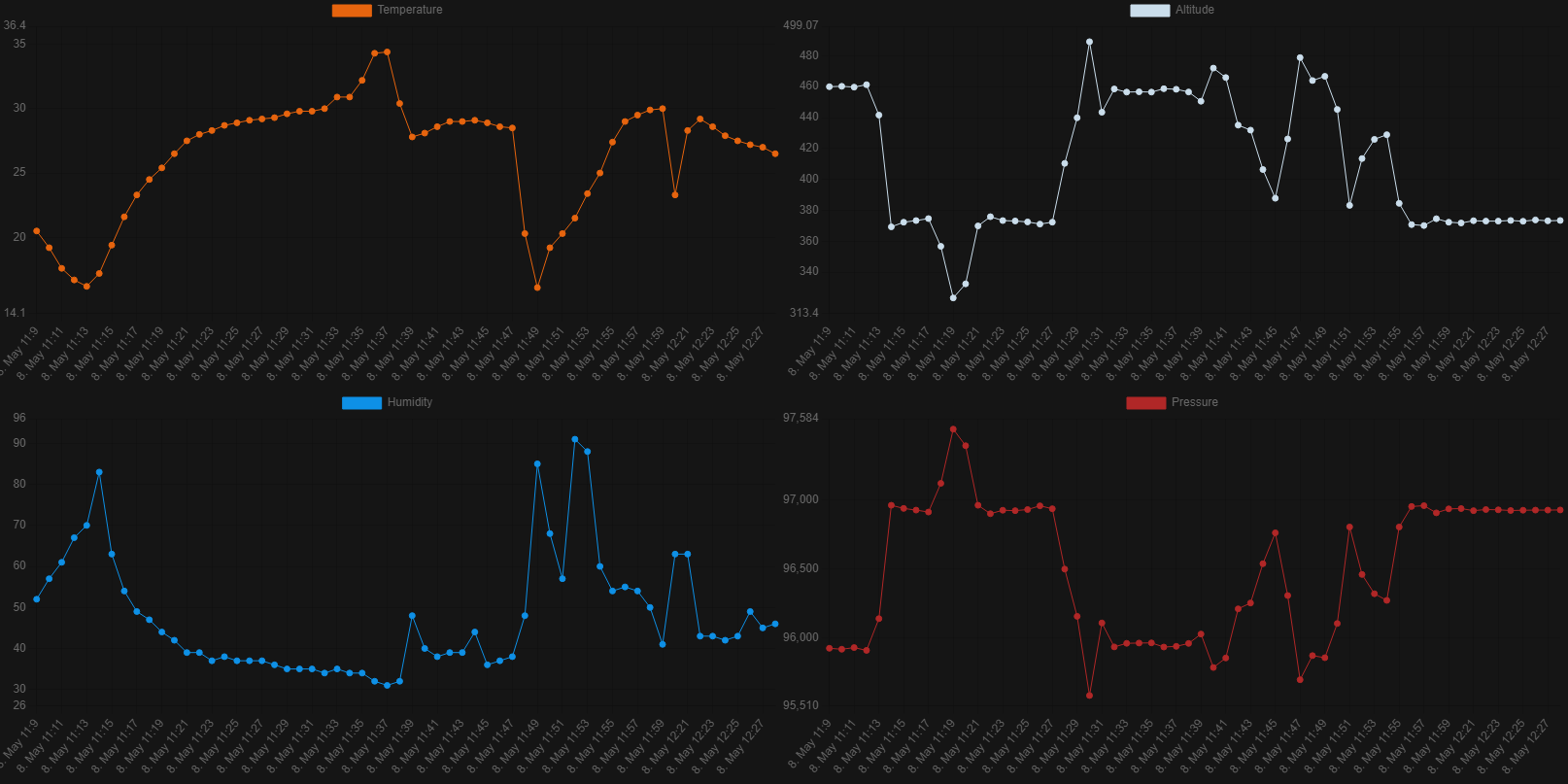
**IV Abbildungs- und Tabellenverzeichnis**

Abbildungen:

* Abb. 1: Kostenplanung
* Abb. 2: MQTT-Nachricht
* Abb. 3: Vier Graphen Seite
* Abb. 4: Kombinierter Graph mit ausgeblendetem Luftdruck
* Abb. 5: MQTT Message Handler
* Abb. 6: Netzplan
* Abb. 7: Gantt-Diagramm
* Abb. 8: Abnahmeprotokoll
* Abb. 9: API Endpoint
* Abb. 10: Angebot

Tabellen:

* Tab. 1: Personalplanung
* Tab. 2: Hardware-Sachmittelplanung
* Tab. 3: Software-Sachmittelplanung
* Tab. 4: Soll-Ist-Vergleich

**V Anhang**

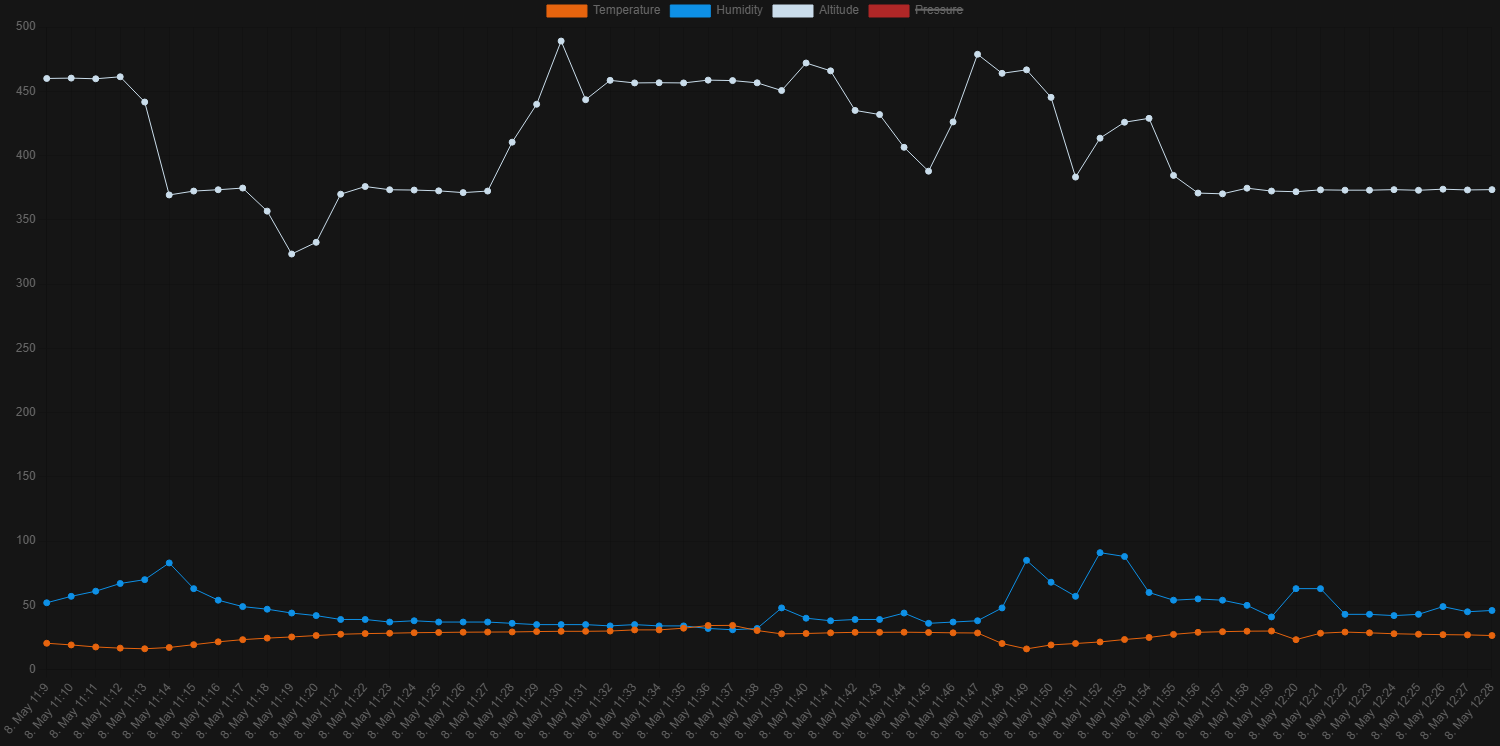
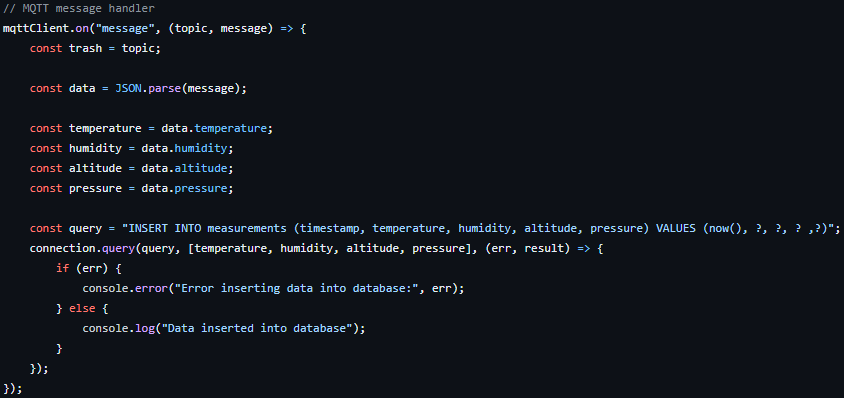
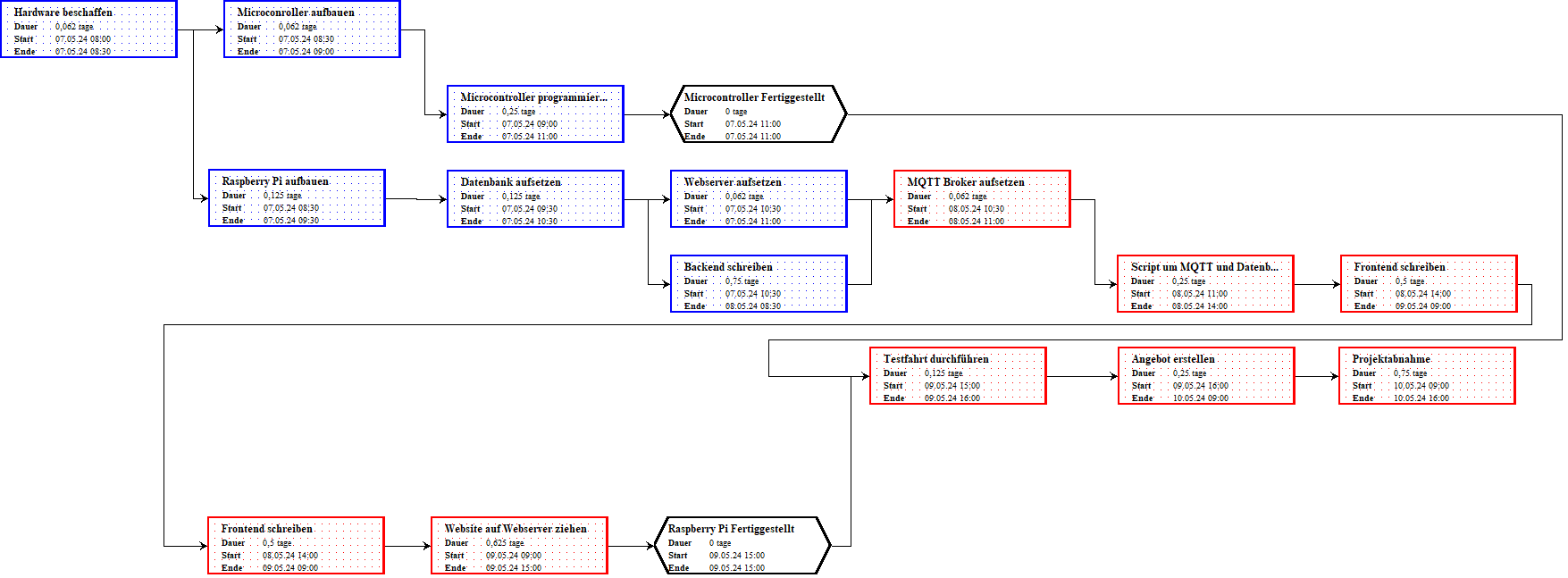
Abb. 3: Vier Graphen Seite

Abb. 4: Kombinierter Graph mit ausgeblendetem Luftdruck

****

****Abb. 5: MQTT Message Handler

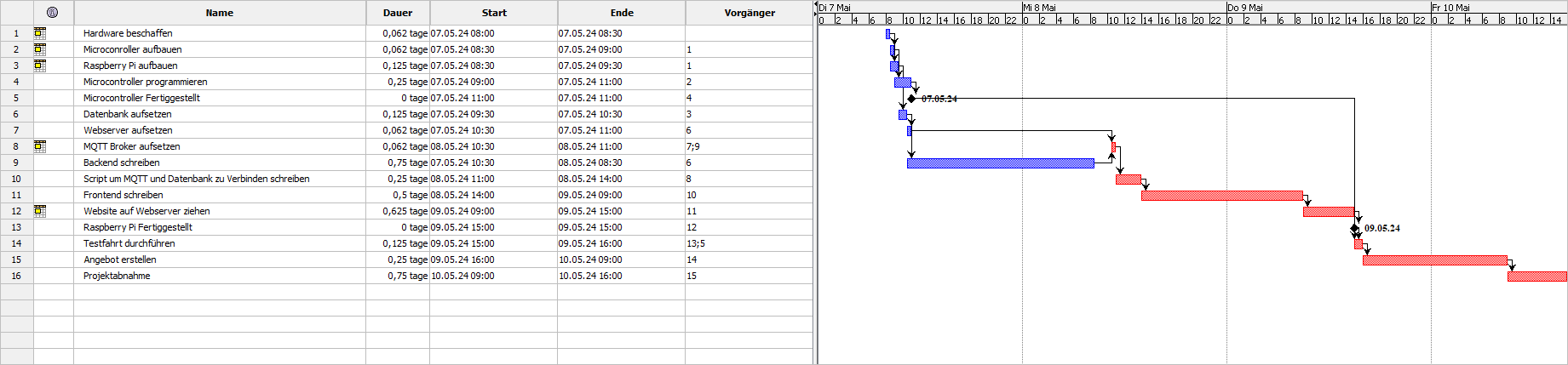
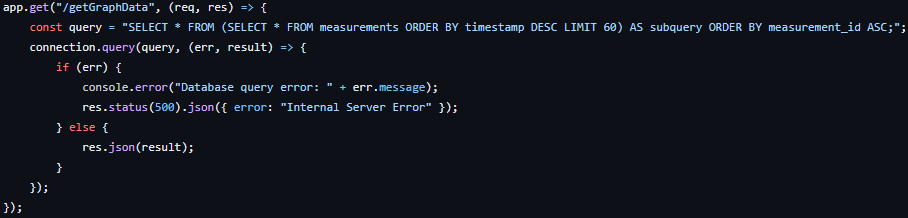
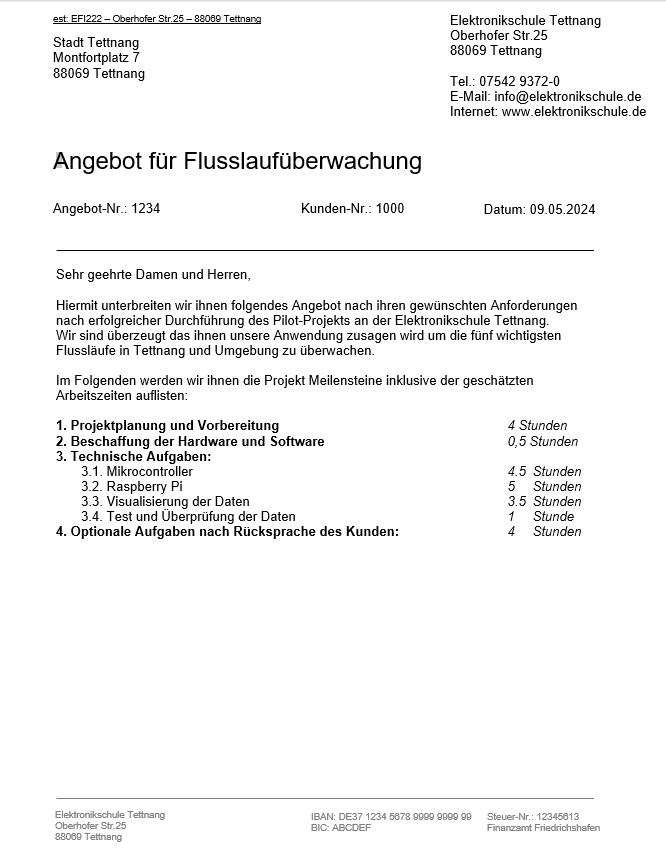
Abb. 6: Netzplan

Abb. 7: GANTT-Diagramm

Abb. 8: Abnahmeprotokoll

Abb. 9: API Endpoint



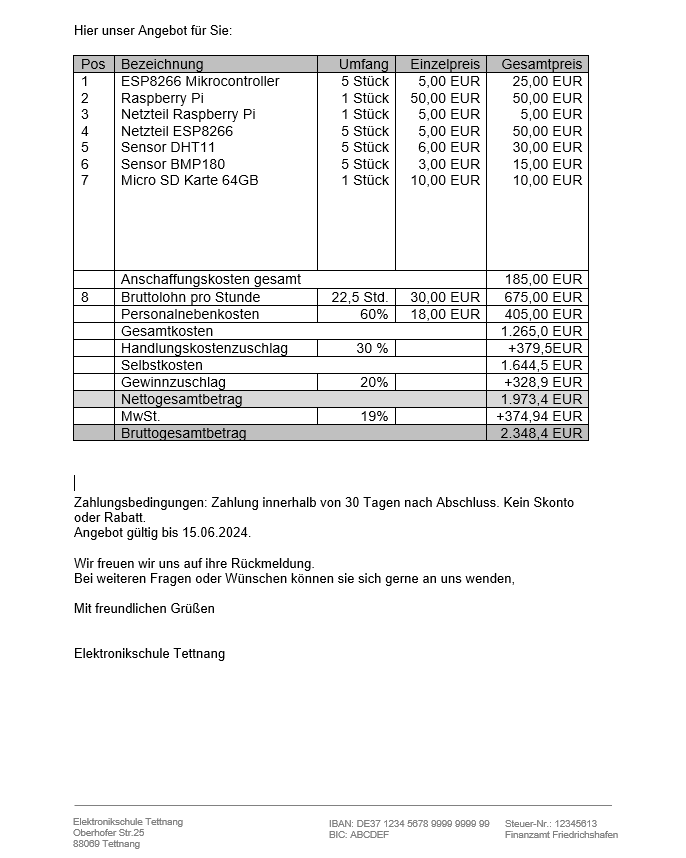


Abb 10**:** Angebot

**VI Eidesstattliche Erklärung**

Hiermit erklären wir, dass wir die vorliegende Arbeit selbstständig verfasst haben, dass wir sie zuvor an keiner anderen Berufsschule und in keiner anderen Ausbildung als Prüfungsleistung eingereicht haben und dass wir keine anderen als die angegebenen Quellen und Hilfsmittel benutzt haben. Alle Stellen der Arbeit, die wörtlich oder sinngemäß aus Veröffentlichungen oder aus anderweitigen fremden Äußerungen entnommen wurden, sind als solche kenntlich gemacht.



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ort, Datum Unterschrift