INTERNET DER DINGE

Wemos D1 Mini mit http-Server

Andre, Roman, Jaron – 10.06.2025

Gewerblich-Technische Schule Offenburg

BFK-SAE

Ein Bild, das Elektronik, Elektronisches Bauteil, Elektrisches Bauelement, passives Bauelement enthält.

KI-generierte Inhalte können fehlerhaft sein.

### Inhaltsverzeichnis

[Inhaltsverzeichnis 1](#_Toc201825408)

[Einleitung 2](#_Toc201825409)

[Projekterläuterung 2](#_Toc201825410)

[Optionale Erweiterungen 2](#_Toc201825411)

[Verwendete Hard- und Software 3](#_Toc201825412)

[Kommunikationsplan 4](#_Toc201825413)

[Arbeitseinteilung 4](#_Toc201825414)

[Kommunikationsart 4](#_Toc201825415)

[Technologien (HTML, XML und JSON) 5](#_Toc201825416)

[HTML (HyperText Markup Language) 5](#_Toc201825417)

[CSS (Cascading Style Sheet) 5](#_Toc201825418)

[XML (eXtensible Markup Language) 6](#_Toc201825419)

[JSON (JavaScript Object Notation) 6](#_Toc201825420)

[Hauptteil 7](#_Toc201825421)

[Vorgehensweise 7](#_Toc201825422)

[Erstinbetriebnahme 7](#_Toc201825423)

[Sensor Treiber 7](#_Toc201825424)

[Erstellung von Frontend 7](#_Toc201825425)

[Dateigenerierung von JSON und XML 8](#_Toc201825426)

[Umsetzung von Optionale Erweiterungen 8](#_Toc201825427)

[Erstellung von Video und Dokumentation 8](#_Toc201825428)

[Schluss 9](#_Toc201825429)

[Schlusswort 9](#_Toc201825430)

[Glossar 9](#_Toc201825431)

[Quellen 9](#_Toc201825432)

# Einleitung

## Projekterläuterung

Ziel des Projektes ist es, ein autark und persistentes laufendes System auf Basis des Wemos D1 Mini ESP 8266 Mikrocontroller zu entwickeln. Dieses System soll mithilfe eines Abstandssensor Sensordaten erfassen und diese über einen integrierten HTTP-Server in verschiedenen Formaten (HTML, XML, JSON) bereitstellen und gleichzeitig eine übersichtliche Weboberfläche mithilfe von HTML und CSS zur Visualisierung der Messwerte anbieten. Zusätzlich soll ein LED-Aktor auf Basis eines sinnvollen Triggers die Erfassung der Messdaten optisch anzeigen.

Ein Bild, das Text, Screenshot, Schrift, Quadrat enthält.

KI-generierte Inhalte können fehlerhaft sein.

### Optionale Erweiterungen

Zusätzlich zu den eigentlichen Projektzielen umfasst die Aufgabenstellung auch optionale Erweiterungen, die das System funktional erweitern und vertiefen sollen:

* Alle selbst erstellten Funktionen sollen mit einem passenden Docstring dokumentiert werden.
* Ein HTTP-Client soll in Python implementiert werden, der auf einem Rechner im gleichen Netzwerk läuft. Dieser Client empfängt regelmäßig die Daten vom HTTP-Server auf dem Wemos D1 Mini und gibt sie als JSON- oder XML-String in der Konsole aus.
* Die Messwert-Erfassung soll unabhängig vom Client alle paar Sekunden (z. B. alle 5 Sekunden) automatisch erfolgen. Dabei sollen stets nur die letzten 10 Messwerte in einer Liste gespeichert bleiben.
* Zusätzlich sollen die gespeicherten Messwerte als Tabelle auf der HTML-Seite dargestellt werden

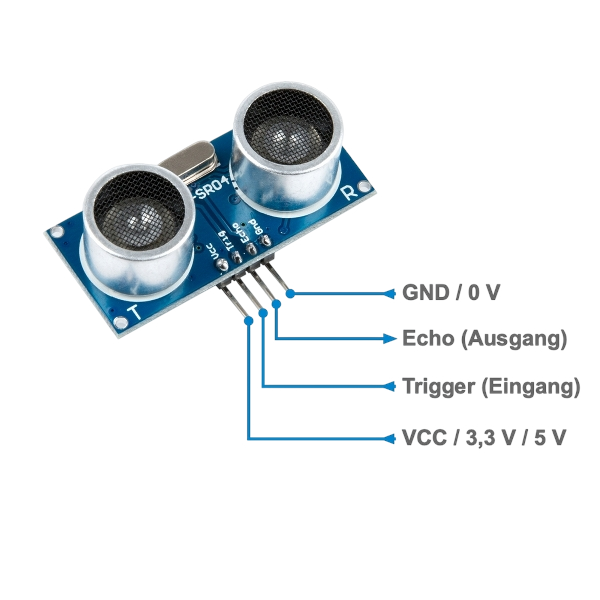
## Verwendete Hard- und Software

**Hardware:**

* **Wemos D1 Mini** als Mikrocontroller-Plattform, der die Sensordaten verarbeitet.
* **HC-SR04** als Abstandssensor, der die Abstandswerte misst und erfasst.
* **LED** auf dem Wemos D1 Mini zur optischen Ausgabe, die den Messrhythmus sichtbar macht.

**Software:**

* **HTTP-Server**, der die aktuellen Messwerte in den Formaten HTML, XML und JSON bereitstellt.
* **Bibliothek zur Sensordatenerfassung** für die Ansteuerung des Abstandssensors.
* **Web Frontend (HTML & CSS)** zur übersichtlichen Darstellung der Sensordaten auf einer Webseite.



## Kommunikationsplan

### Arbeitseinteilung

Zu Beginn des Projekts erfolgte eine gemeinsame Planung, in der die Umsetzungsschritte besprochen und die Aufgaben verteilt wurden:

**Andre:**

* Entwicklung des Skripts und Erstellung des Projektvideos
* Darstellung der Messdaten in Tabellenform
* Formatierung und Strukturierung des Quellcodes

**Roman:**

* Gestaltung der HTML-Webseite und des CSS-Frontends
* Einrichtung des GitHub-Repositories
* Automatisierte Erzeugung der Datenformate XML und JSON

**Jaron:**

* Erstellung der Projektdokumentation
* Umsetzung optionaler Erweiterungen
* Koordination und Verwaltung der Teamaufgaben
* Erstinbetriebnahme des Systems inklusive HTTP-Server und Sensorbibliothek

### Kommunikationsart

**Kontaktdaten:**

* **Andre:**
* **Roman:**
* **Jaron:**

Wir haben uns hauptsächlich in der Schule unterhalten, abgesehen davon aber auch in GitHub unsere Probleme und To-Dos beschrieben.



## Technologien (HTML, XML und JSON)

### HTML (HyperText Markup Language)

HTML ist die Standardsprache zur Erstellung von Webseiten. Sie strukturiert Inhalte und definiert die Darstellung im Browser. In unserem Projekt dient HTML zur strukturellen Anzeige der aktuellen Messwerte und Buttons.



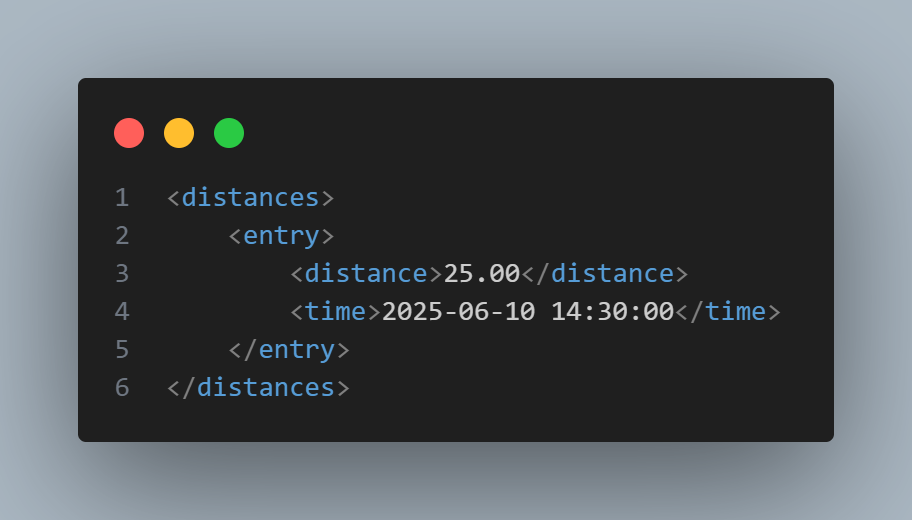
### CSS (Cascading Style Sheet)

CSS ist die sprache zur Gestaltung von Webseiten. Sie legt fest, wie Inhalte im Browser dargestellt werden – zum Beispiel Farben, Schriftgrößen, Abstände oder Layouts. In unserem Projekt wird CSS verwendet, um das Aussehen der Messwerte und Buttons einheitlich und ansprechend zu gestalten.



### XML (eXtensible Markup Language)

XML ist ein Format zur strukturierten Speicherung und Übertragung von Daten. Es nutzt Tags, um Informationen klar zu gliedern, ähnlich wie HTML.



### JSON (JavaScript Object Notation)

JSON ist ein leichtgewichtiges Datenformat zum Austausch von Informationen. Es verwendet eine klare, textbasierte Struktur aus Schlüssel-Wert-Paaren. Durch sein geringes Gewicht und die umfassende Unterstützung in JavaScript eignet es sich besonders gut für Webanwendungen.



# Hauptteil

## Vorgehensweise

### Erstinbetriebnahme

Zunächst wurde der Mikrocontroller mit einem USB-Kabel an den PC angeschlossen und die Installation von MicroPython auf dem ESP8266 durchgeführt. Anschließend wurde getestet, ob alles funktioniert, indem ein einfaches Python-Programm ausgeführt wurde, das die LED zum Leuchten bringt. Nachdem dies erfolgreich war, haben wir die bereits vorliegenden Dateien für den HTTP-Server und die WLAN-Verbindung an das vorgesehene Netzwerk angepasst. Daraufhin wurde auch dies getestet, indem der HTTP-Server gestartet und die Verbindung überprüft wurde.

Nachdem das Basis-Setup fertig war, haben wir es als [GitHub-Repository](https://github.com/RomanPleshka/Abstandmesser) hochgeladen, damit wir gemeinsam daran arbeiten können.

### Sensor Treiber

Nachdem die Erstinbetriebnahme erfolgreich verlief, haben wir nach einer passenden Bibliothek beziehungsweise einem Treiber für den Sensor HC-SR04 gesucht, um die Sensordaten korrekt auszulesen. Dabei sind wir auf ein einfaches Python-Skript gestoßen. Um dieses zu testen, mussten wir die richtigen Pins mit dem Sensor verbinden, was dank der klaren Beschriftung recht unkompliziert war. Nachdem wir die Pins mit Jumper-Kabeln verbunden hatten, konnten mithilfe der Dokumentation auf der GitHub-Seite der Bibliothek die Sensordaten erstmals in der Konsole ausgeben.

### Erstellung von Frontend

Während die Vorbereitungen getroffen wurden, wurde bereits das Frontend für die HTML-Seite erstellt. Dabei haben wir einen einfachen Ansatz gewählt und besonderen Wert auf Übersichtlichkeit und Funktionalität gelegt. Auf der Seite werden sowohl der aktuelle Messstand in HTML-Form angezeigt als auch Download-Funktionen für die XML- und JSON-Dateien bereitgestellt. Zudem gibt es direkte Links, die unmittelbar zu den jeweiligen Dateien führen.

### Dateigenerierung von JSON und XML

Für das Projektziel und die Speicherung der erfassten Sensordaten haben wir zwei Dateiformate implementiert: JSON und XML. Dabei werden stets die letzten Messwerte gespeichert. Zusätzlich wird jeder Messwert mit einem Zeitstempel versehen, der eine eindeutige Zuordnung der Datenpunkte in zeitlicher

Reihenfolge ermöglicht. Dies erleichtert sowohl die Analyse als auch die Nachverfolgung der Messwerte.

### Umsetzung von Optionale Erweiterungen

Da wir recht schnell waren, haben wir uns dazu entschieden, alle optionalen Erweiterungen in das Projekt zu integrieren.

Zunächst haben wir einen Client erstellt, welcher Daten vom HTTP-Server anfragt und diese als JSON ausgibt. Dieser ist unabhängig und benötigt lediglich den aktiven HTTP-Server des Mikrocontrollers. Er stellt eine HTTP-Verbindung zum Controller her und erstellt eine GET-Anfrage auf die distance.json-Datei. Die Antwort liest er aus und schreibt diese roh in die Konsole.

Zudem haben wir eine Liste erstellt, die stets die letzten 10 Messwerte speichert. Diese wird anhand einer Tabelle mithilfe von HTML und CSS auf der Seite angezeigt.

Außerdem haben wir für jede unserer Funktionen im Code einen Docstring erstellt, der eine Beschreibung sowie den Rückgabewert und die Parameter enthält.

### Erstellung von Video

Andre mach mal hinne

# Schluss

## Schlusswort

Das Projekt ist insgesamt sehr gut für uns verlaufen. Die Aufgabenteilung im Team verlief konfliktfrei, und jeder hat seine Aufgaben zuverlässig erfüllt. Es gab keine größeren Probleme beim Programmieren, und wir haben uns gegenseitig bei kleineren Schwierigkeiten unterstützt. Insgesamt war die Teamdynamik und Klima wirklich gut.

Das Projekt war jedoch für unseren Geschmack etwas zu einfach. Wir würden uns für das nächste Mal etwas Herausfordernderes wünschen oder noch mehr optionale Möglichkeiten, Punkte zu erzielen. Es wäre großartig, wenn wir uns mit komplexeren Aufgaben auseinandersetzen könnten.

## Glossar

|  |  |
| --- | --- |
| Fachwort | Definition |
| Autark | Unabhängig, Eigenständig |
| Persistent | Erhaltend  (Daten bleiben erhalten) |
|  |  |
|  |  |
|  |  |

## Quellen

[Hypertext Markup Language – Wikipedia](https://de.wikipedia.org/wiki/Hypertext_Markup_Language)

[Cascading Style Sheets – Wikipedia](https://de.wikipedia.org/wiki/Cascading_Style_Sheets)

[Extensible Markup Language – Wikipedia](https://de.wikipedia.org/wiki/Extensible_Markup_Language)

[JSON – Wikipedia](https://de.wikipedia.org/wiki/JSON)