PROJEKTDOKUMENTATION – CYBERPHYSISCHE SYSTEME

VERSION 1.0

Magnus Drolshagen

1 Inhaltsverzeichnis

2 Einstieg 2
2.1 Aufgabenstellung 2
2.2 Projektideen und -planung 2
3 Projektverlauf 2
4 Fazit 3
4.1 Endergebnis 3
4.1.1 Aufbau der Hardware 3
4.1.2 Aufbau der Software 4

2 Einstieg

2.1 Aufgabenstellung

Aufgabe war es, ein Projekt anhand eines ESP32-Mikrocontrollers durchzuführen und zu präsentieren. Es sollten Sensoren/Aktoren verwendet werden. Der Wert des Projektes lag dabei auf der Präsentation und Vorstellung des Projektes.

2.2 Projektideen und -planung

Die Idee war, einen Assistenten zu programmieren, welcher das Arbeiten am Arbeitsplatz vereinfacht und gewisse Aufgaben übernehmen kann. Im späteren Verlauf der Planung wurde deutlich, da aufgrund der kleinen Dimension des endgültigen Produktes die Funktionen des Produktes auf das Messen und Anzeigen/Benachrichtigen beziehen würden. Die zur Verfügung stehenden Sensoren, boten einen Funktionsumfang welcher hauptsächlich im Zusammenhang mit dem Raumklima stehen.

Mit diesen Faktoren wurde ein Projekt beschlossen, welches die Temperatur, Luftfeuchtigkeit, Helligkeit, Lautstärke und die Entfernung zum Bildschirm messen kann. Das Zeil war es möglichst funktionsreiche Sensoren zu verwenden, um nicht unnötig viele Sensoren verwenden zu müssen, was die Skalierbarkeit und Wartbarkeit verbessern.

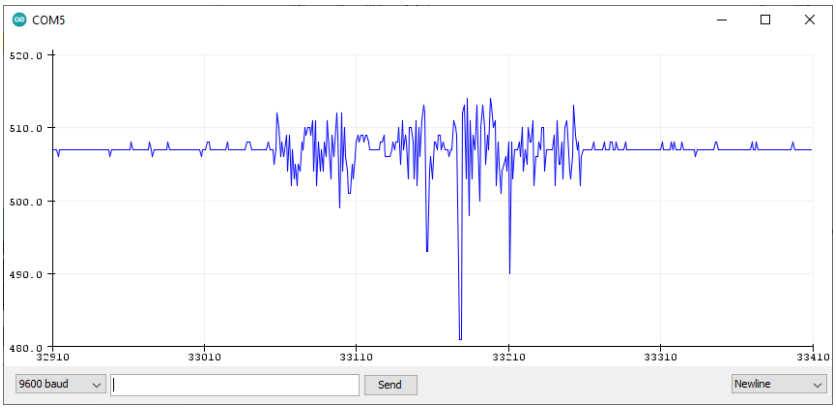
Als Ausgabe sollte der eigene Webbrowser dienen. Somit wäre das Gerät eigenständig es bedarf keiner physischen Anzeige am Gerät selbst. Daraus folgernd dient der ESP32 als Webserver, der eine WLAN-Verbindung benötigt. Die Werte der einzelnen Messungen sollten grafisch mit dem optimalen Bereich der Werte dargestellt werden.

3 Projektverlauf

Folgende Sensoren wurden für das Projekt ausgewählt:

- DHT11 → Temperatur
- DHT11 → Feuchtigkeit
- KY-037 → Lautstärke
- BH1750 → Helligkeit
- HC-SR04 → Entfernung

Zu Beginn des Projekts wurde das Augenmerk auf die Integration der Sensoren gelegt. Die geschieht wie oben genannt über die WLAN-Schnittstelle. Der Sensor konnte nicht wie gewünscht implementiert werden. Dieser



schlägt nämlich bei lauten, kurzen Geräuschen aus. Nicht jedoch bei einem lauten Hintergrundrauschen

beispielsweise. Aus diesem Grund wurde der Sensor mitsamt, aufgrund fehlender Alternative und Zeitdruck, der Funktionalität aus dem Projekt entfernt.

Als nächster Schritt stand die Einbindung des ESP32 in das lokale Netzwerk an. Die Schwierigkeiten, die hierbei entstanden, waren hauptsächlich aufgrund der fehlenden Updates der Entwicklungsumgebung entstanden. Diese Probleme wurden jedoch schnell beseitigt.

Darauf folgte die Einrichtung der Webdienste auf dem Mikrocontroller. Diese erweisen sich größtenteils als problemlos. Probleme brachte lediglich die Verzeichnisstruktur, die den hochgeladenen Dateien zugrunde liegt.

Die Erstellung der Benutzeroberfläche benötigte mit Abstand am Meisten Zeit. Diese wurde mithilfe von HTML und CSS erledigt und lief problemlos vonstatten.

4 Fazit

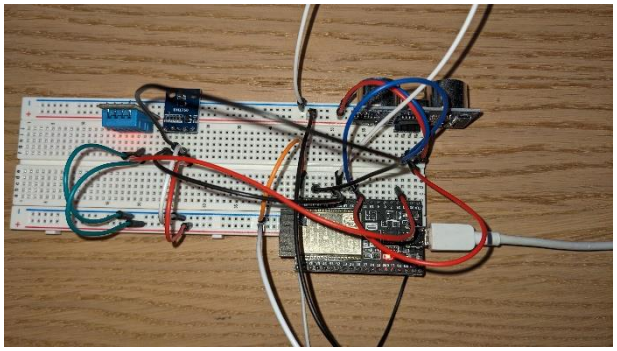
4.1 Endergebnis

4.1.1 Aufbau der Hardware

Die Sensoren werden mit dem ESP32 auf ein Breadboard gesteckt und wie folgt verdrahtet.

Sensoren	Anschluss Sensor	Anschluss ESP32
DHT11	GND	GND
	DATA	GPIO15
	VCC	3.3V
BH1750	VCC	3.3V
	GND	GND
	SCL	GPIO22
	SDA	GPIO21
	ADDR	Nicht verbinden
HC-SR40	GND	GND
	Echo	GPIO18
	Trig	GPIO5
	VCC	5V

Hinweis: Pins für die Stromversorgung können auch zusammengefasst werden.



4.1.2 Aufbau der Software

Folgende Bibliotheken werden benötigt:

- Wire
  https://www.arduino.cc/reference/en/language/functions/communication/wire/
- WiFi
  https://www.arduino.cc/reference/en/libraries/wifi/
- BH1750
  https://www.arduino.cc/reference/en/libraries/bh1750/
- DHT
  https://www.arduino.cc/reference/en/libraries/dht-sensor-library/
- ESPAsyncWebServer
  https://www.arduino.cc/reference/en/libraries/espasyncwebserver/
- SPIFFS
  https://github.com/espressif/arduino-esp32/blob/master/libraries/SPIFFS/src/SPIFFS.h

4.1.2.1 INO-Datei

1. Einbinden der Bibliotheken
2. Starten des Webservers
3. WLAN-Zugangsdaten festlegen
4. Variablen für die Erfassung der Werte erstellen
5. Variablen für die Werte erstellen
6. Werte für die Warnzeichen erstellen

4.1.2.2 Setup()

1. Seriellen Monitor starten
2. WLAN-Verbindung starten
3. Dateisystem auf den ESP32 laden
4. Dateien für die Website übertragen
5. Sensorerfassung starten

4.1.2.3 Loop()

1. Werte_auslesen()
2. Werte_anzeigen()
3. Erfasste Werte interpretieren

4.1.2.4 Werte_auslesen()

1. Methoden zur Erfassung der Werte aufrufen

4.1.2.5 Werte_anzigen()

1. Werte auf dem Seriellen Monitor ausgeben

4.1.2.6 String processor(String var)

1. Erfasse Werte an die HTML weiterleiten