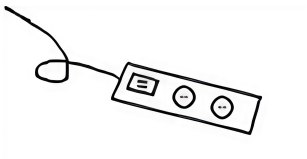
Smarte Steckdosenleiste

Jonathan Westphal (16)

February 21, 2025



Contents

1 Einleitung

1.1 Wer bin Ich

Ich bin ein technikbegeisterte Jugendlicher und besuche aktuell das Bismark-Gymansium in Karlsruhe. In den vergangenen Jahren habe ich mich intensiv mit Elektronik beschäftigt und konnte dabei bereits wertvolle Erfahrungen sammeln. So habe ich am CanSat-Wettbewerb [1] teilgenommen und dabei den zweiten Platz erreicht. Darüber hinaus habe ich zahlreiche eigene Elektronikprojekte zu Hause entwickelt und erfolgreich umgesetzt, darunter ein Fahrradbremslicht sowie eine Wetterstation mit Serveranbindung.

[1] Deutscher Wettbewerb zum Bau eines Minisatellitenbau in Dosenformat (cansat.de)

1.2 Wie kam ich auf die Idee

Die Idee für dieses Projekt entstand bei dem Blick auf meinen Schreibtisch. Hier sind viele verschiedne elektronische Gräte angeschlossen (PC, Monitor, Laptop, Labornetzteil, Lötkolben, Receiver, Lautsprecher, diverse Ladegeräte ...). Dabei hatte ich folgende Gedanken

- Status LEDs brennen dauerhaft
- Standby Stromverbrauch auch bei ausgeschaltetem Gerät
- Steckdosenleiste mit unzureichendem Überspannungsschutz
- Welchen Energie- / Stromverbrauch besitzen meine Geräte

Anfangs hatte ich das Problem einfach dadurch gelöst, dass ich eine schaltbare Steckdosenleiste verwendete. Doch mit der Zeit vielen wir weitere Probleme auf

- Unnötiges Warten auf das Abschalten der Steckdosenleiste, insbesondere wenn am PC noch ein Update und Compilierungsjob läuft
- Spitzenströme beim Einschalten führten zur defektem Schalter in der Steckdosenleiste und zur Auslösung der Haushaltssicherung. Diese

musste ich dann immer zwei Stockwerke tiefer im Keller wieder einschalten. Hinzu kam, das ich diverse elektronische Geräte wieder stellen musste (Bsp. elektronische Uhr).

Deshalb beschloss ich mich einen Tages, dass ich genug habe und begann daraufhin damit eine Lösung zu entwickeln, die diese beiden Probleme ein für alle Mal beseitigen würde.

1.3 Was habe ich erarbeitet

So entstand die Idee der smarten Steckdosenleiste. Doch welche Funktionen sollte diese haben? Zuerst habe ich festgelegt, dass sie erkennen muss, wenn ich mich vom Schreibtisch entferne. Am einfachsten und universellsten ließ sich das durch eine Leistungsmessung des Stromverbrauchs realisieren. Außerdem sollte die Steckdosenleiste die Möglichkeit haben, beim Einschalten kurzzeitig einen Widerstand zuzuschalten. Dadurch wird vermieden, dass die Sicherung durch den plötzlichen Einschaltstrom auslöst. Als weiteres Feature hatte ich die Idee, die Steckdosenleiste in ein Smart-Home-Netzwerk integrieren zu können um sie ebenso von Unterwegs aus steuern zu können und auch damit es einfacher ist beim entwickeln einen neuen Code hochzuladen. Als letztes Entschied ich mich noch dafür Sicherheitsfeatures einzubauen die meiner Meinung nach jede Steckdosenleiste haben sollte, so bekam sie noch einen Überspannungsschutz für Blitzeinschläge und eine Backup Schmelzsicherung.

2 Umsetzung

Die Umsetzung nam ich analog meiner anderen Projekte vor

1. Idee und Funktion - diese war mit der smarten Steckdose bereits geboren
2. Designphase - grober Aufbau und Festlegung der Funktionen
3. Entwicklung - Schaltplanentwurf und Layout der Platine
4. Realisierung - Bestückung der Platine
5. Programmierung - "Try and Error" Erstellung des Codes in C++

6. Test - Erster Test und Optimierug der Steckdosenleiste für den Alltag

Figure 1: Zeigt die fertig entwickelte Steckdosenleiste

2.1 Elektrischer Aufbau

Den gesamten elektrischen Aufbau, sowie die Schaltpläne habe ich alle mit meinen Vorkenntnissen in KiCad[2] erstellt, beim Design der Platinen habe ich vor allem auf eine möglichst einfache Bestückung, sowie einen möglichst großen Sicherheitsabstand zwischen dem "Hochspannungsbreich" 230V und dem "Niederspannungsbereich" 5V/3.3V des Mikrocontrollers geachtet. Die Abbildungen 3/4 zeigen hier noch die Leiterbahnen auf der Leiterplatte.

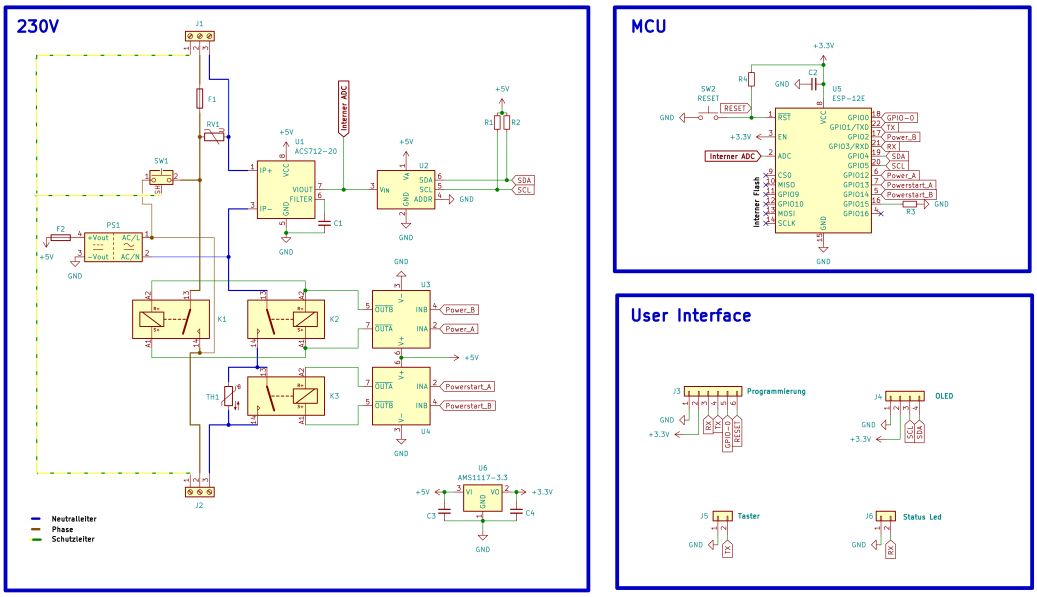


Figure 2: Zeigt einen vereinfachten Schaltplan der Gesamtplatine

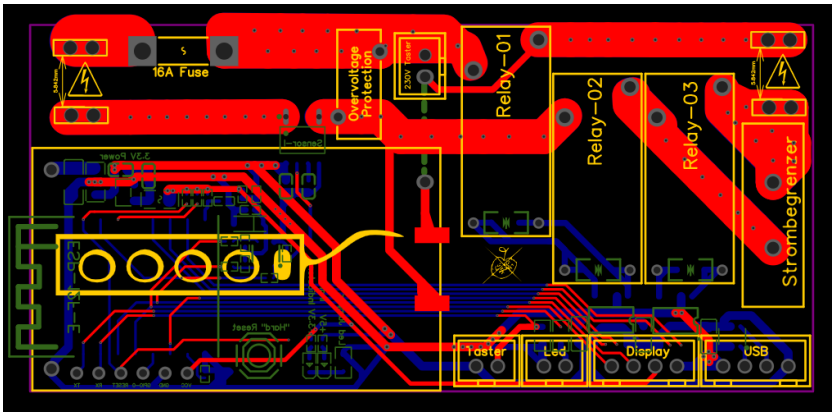


Figure 3: Zeigt die Leiterbahnen auf der Oberseite der Leiterplatte

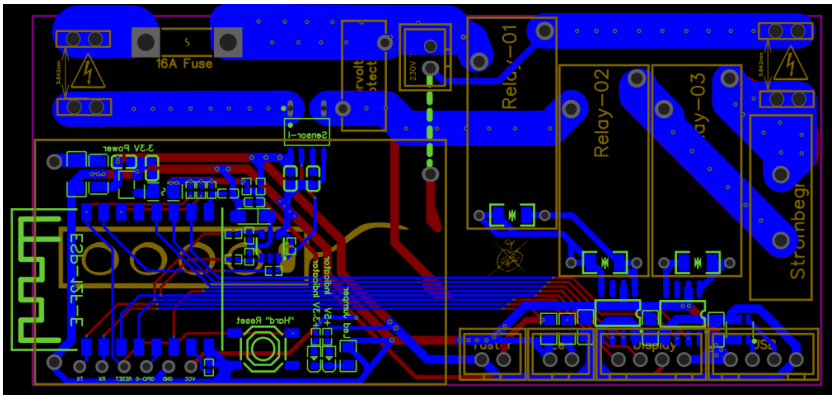


Figure 4: Zeigt die Leiterbahnen auf der Unterseite der Leiterplatte

[2]Open-Source-Software zur Erstellung von Leiterplatten und elektrischen Schaltplänen (kicad.org)

2.1.1 Stromversorgung

Um die Stromversorgung möglichst sicher und stromsparend zu gestalten, habe ich mir eine Abschaltautomatik ausgedacht. Nach dem Ausschalten der Steckdosenleiste wird auch diese vom Netz getrennt. Dies hat zur Folge, dass es keinerlei Standby Energieverbrauch gibt. Nachteil hierbei die Steckdosenleiste muss einmalig zum Einschalten manuell über einen Taster bedient werden. Allerdings habe ich auch die Möglichkeit eines Jumpers auf der Platine vorgesehen, so dass die Steckdosenleiste auch dauerhaft im Smart Home Netzwerk zur Verfügung steht. Der Rest der Stromversorgung ist dann wesentlich einfacher, die Wechselspannung wird durch ein Schaltnetzteil in 5V Gleichspannung umgewandelt, dieser Zwischenschritt ist notwendig, da die Relais nur mit dieser Spannung schalten können. Und dann wird die Spannung in einem letzten Schritt noch in 3,3V für den Mikrocontroller umgewandelt. Die Abbildung 2 zeigt hierbei auch nochmal den Schaltplan.

2.1.2 Leistungsmessung

Um eine Leistungsmessung der Stromstärke zu ermöglichen, habe ich mich für die Verwendung eines Halleffektsensors anstelle eines Messwandlers entschieden, um den Aufbau so klein und einfach wie möglich zu halten. Beim Hall-Effekt-Sensor fließt der Strom nämlich direkt durch den Sensor anstatt durch eine

Ferritkernspule und erzeugt somit direkt ein elektrisches Feld, das mit einem Magnetsensor gemessen werden kann, anstatt eine Spannung in einer Ferritkernspule. Die Berechnung der Leistung wird in 2.2.2 ausführlich beschrieben.

2.1.3 230V Regulierung

Eine der kompliziertesten Schaltungen, die ich dann entwerfen musste, war die für die 230V-Regelung, die einfache Schaltung war zwar nicht schwierig, aber die Toleranzen und der Einschaltstrom machten es kompliziert. Die Schaltung ist aber recht einfach zu erklären, als erstes kommt ein Varistor[3] als Überspannungsschutz für Blitzeinschläge, dann kommt eine Zusatzsicherung, die für 10A ausgelegt ist, dann kommen 2 Spezialrelais, die für besonders hohe Einschaltströme geeignet sind und abschließend kommt nur noch ein Überbrückungsrelais, das am Anfang einen NTC-Widerstand[4] dazwischen schaltet, um die Einschaltströme so zu begrenzen, dass die einfache Haushaltssicherung nicht mehr auslößt. Dies ist auch noch einmal im Schaltplan in der Abbildung 2 zu sehen.

[3]Spannungsabhängiger Widerstand

[4]Ein temperaturabhängiger Widerstand wird verwendet, um sicherzustellen, dass der Widerstand bei einem Versagen des Überbrückungsrelais nicht durchbrennt, sondern lediglich warm wird (NTC = Negative Temperature Coefficient).

2.2 Programmierung

Um wenigstens die Programmierung relativ einfach zu halten, habe ich mich entschieden, die gesamte Programmierung in C++ (genauer Arduino C++) zu schreiben, dabei habe ich für einfache Funktionen wie z.B. das Anzeigen von Grafiken und Texten auf dem Display Standardbibliotheken verwendet und den Rest des Codes sowie die Leistungsberechnung (2.2.2) selbst programmiert, dabei sind mir zwar oft Fehler unterlaufen, aber durch die vielen kleinen Glücksmomente, wenn der Code funktioniert hat, hat es dann doch Spaß gemacht.

2.2.1 Hauptprogrammroutine

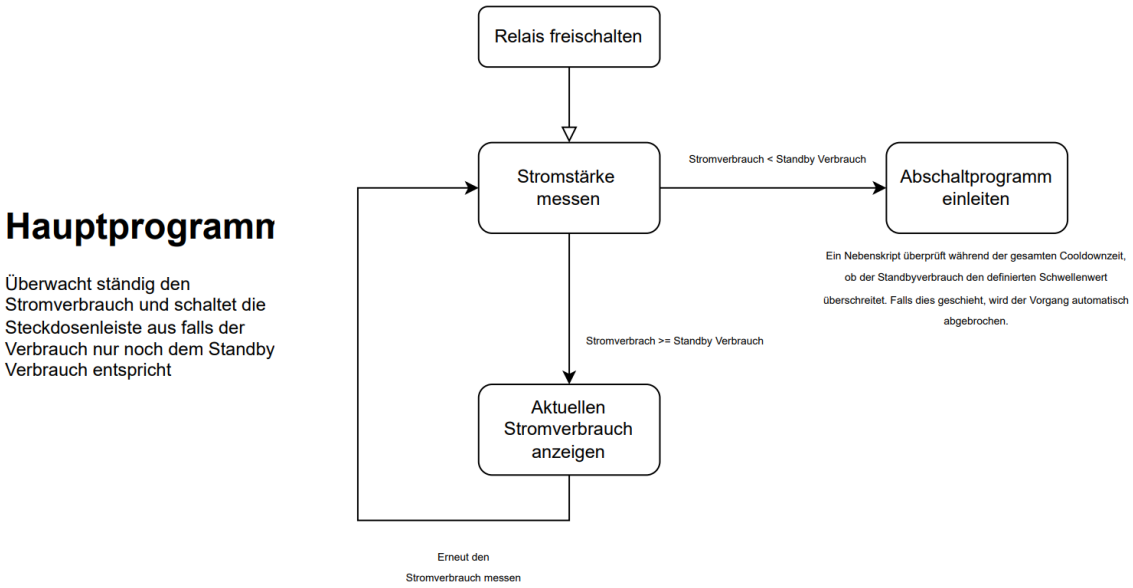


Figure 5: Zeigt ein Flussdiagramm der Hauptroutine des Codes

Die Abbildung 5 zeigt die vereinfachte Hauptroutine des Codes, die im Wesentlichen daraus besteht, ständig neue Leistungsmessungen durchzuführen und dann zu entscheiden, ob der gemessene Wert bereits dem Standby-Verbrauch entspricht. Während diese Routine allein noch recht einfach zu programmieren war, wurde es recht komplex, den gesamten Code ohne Blocking-Funktionen (Funktionen, die eine Wartezeit beinhalten, in der der Prozessor quasi schläft) zu schreiben. Denn sonst würden andere Teile des Programms, wie z.B. die Erkennung von Tastendrücken, nicht mehr erkannt werden und der gesamte Teil mit der Smart Home Integration würde nicht mehr funktionieren, da sonst Abfragen nicht mehr bearbeitet werden würden.

2.2.2 Leistungsberechnung

Eine der schwierigsten Hürden während der gesamten Programmierung war die korrekte Messung bzw. Berechnung der Stromstärke, was aber vor allem an der Natur von Schaltnetzteilen liegt, da diese ihre Leistung nicht wie ein

ohmscher Widerstand entlang der gesamten sinusförmigen Spannungskurve abrufen, sondern aufgrund ihrer Kondensatoren immer etwas Energie gespeichert haben und somit ihre Leistung immer nur entlang der Spitze der Spannungskurve abrufen.

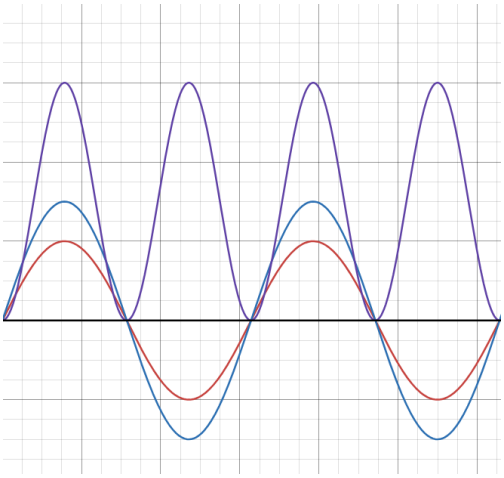


Figure 6: Ohmer Wiederstand

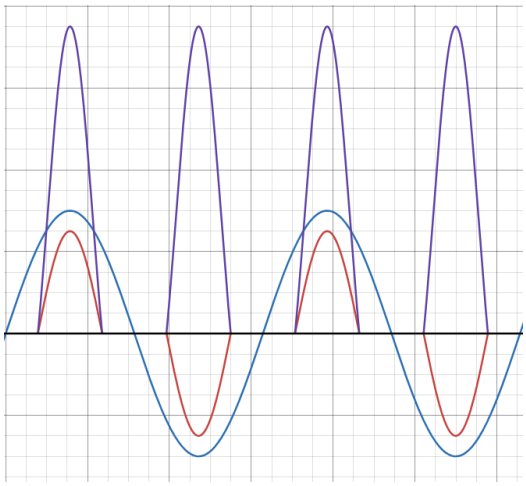


Figure 7: Schaltnetzteil

Legende zu Abbildung 2 und 3:
Lila = Elektrische Leistung
Blau = Spannungskurve
Rot = Ampere-Kurve

So einfach war das Problem aber noch nicht gelöst, denn je nach Schaltnetzteil ist die Amperekurve nicht immer eine Sinuskurve, so dass man den Flächeninhalt der Kurve berechnen muss, um auf die wahre Leistung zu kommen. Zusätzlich muss dieser Wert noch durch 2 dividiert werden, um den Effektivstromwert für Wechselstrom zu erhalten.

```
int current_messen() {
  for (int i = 0; i < 1000; i++) {
    messwert = analogRead(current);
    if (messwert > nullwert) {
      alle_werte += messwert - nullwert;
      anzahl_werte++;
    }
    if (messwert < nullwert) {
      alle_werte += nullwert - messwert;
      anzahl_werte++;
    }
    delayMicroseconds(100);
  }

  av = alle_werte / anzahl_werte;
  av = av / 1.41421  //Wurzel 2

  av = (int)av //Zahl in dezimalzahl umwandeln

  //Werte Zuruecksetzten um Overflow zu verhindern
  alle_werte = 0;
  anzahl_werte = 0;

  return av;
}
```

Mein derzeitiger Code-Ansatz ist es, 5 Ampere-Kurven zu messen und dann den Mittelwert zu bilden, dies tue ich, damit sich eventuelle Messfehler innerhalb der 1000 Messungen nicht so stark auswirken.

2.2.3 Taster Programmierung

Die Programmierung des Tasters war dann eine ähnlich so schwierige Herausforderung. Die Hauptroutine, die den Taster kontrolliert, ist in Abbildung 8 dargestellt. Sie besteht im Wesentlichen aus zwei Unterprogrammen, das erste überprüft, ob der Taster gedrückt wurde und speichert ab, wie lange, das zweite Unterprogramm entscheidet dann, ob dieser Tastendruck als kurz oder lang anzusehen ist und speichert dies in einem Array ab, bei weiteren Tastendrücken prüft es dann, ob die gedrückte Kombination mit einer der

Aufgaben übereinstimmt und führt diese dann aus.

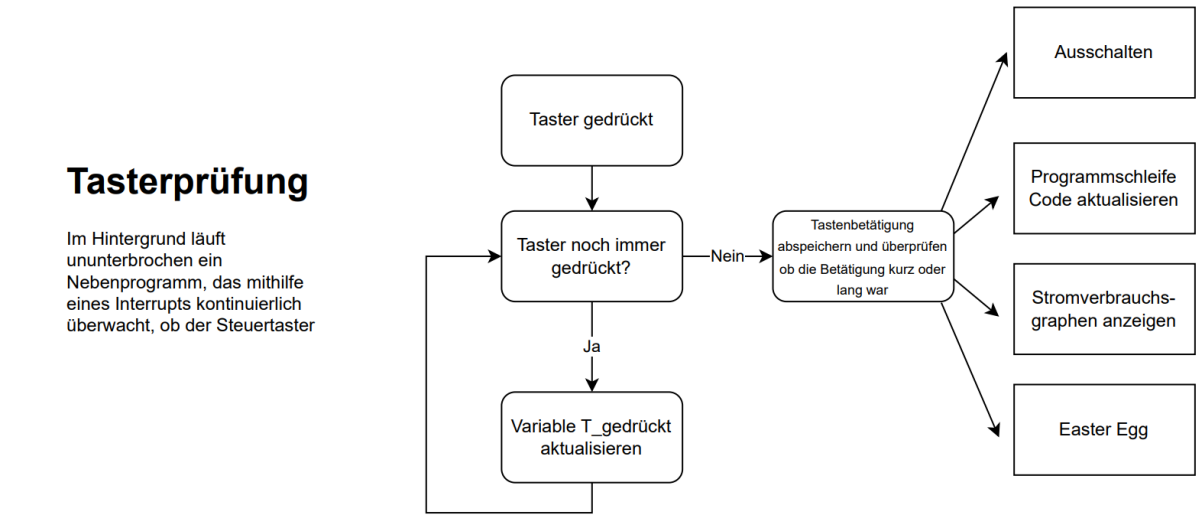


Figure 8: Zeigt das Flussdiagramm der Tasterroutine des Codes

2.2.4 Smart Meter OS

Als letztes musste ich noch eine Art "Betriebssystem" für die Steckdosenleiste schreiben. Der Benutzer der Steckdosenleiste sieht auf dem Display dabei immer die gerade ausgeführte Tätigkeit und wird mit den nötigen Informationen versorgt, dabei beginnt das OS zuerst mit der Initialisierung, währenddessen wird eine Art Icon der Steckdosenleiste dargestellt, danach beginnt die Steckdosenleiste sich mit dem Wlan zu verbinden, dies wird dem Benutzer dann wieder auf dem Display dargestellt, egal ob die Verbindung erfolgreich war oder nicht fährt die Steckdosenleiste dann mit der Hauptroutine fort, dabei wird dem Benutzer dann immer ein ungefährer Messwert der aktuellen Wattzahl angezeigt, in dieser Anzeige bleibt die Steckdosenleiste dann solange, bis der Benutzer entweder mit ihr interagiert oder den Arbeitsplatz verlässt, dann wird entweder ein Shutdown Bildschirm angezeigt oder auf die gewünschte Anzeige umgeschaltet. All dies ist auch noch einmal in Abbildung 9 dargestellt.

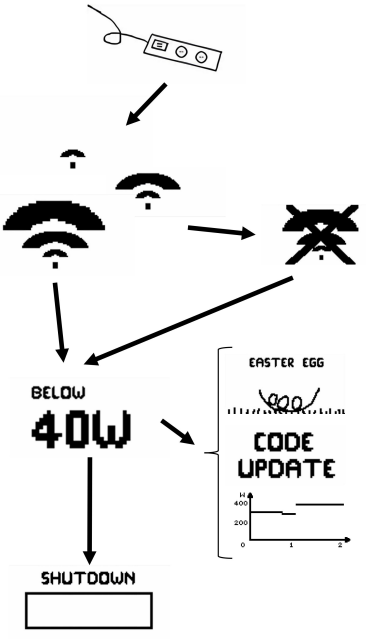


Figure 9: Zeigt den Programmablauf

2.2.5 Code Updates

Da es beim Programmieren besonders praktisch ist, möglichst schnell möglichst neue Programmierungen auszuprobieren, habe ich mich dazu entschlossen, die Programmierung über Wlan zu ermöglichen, genauer gesagt habe ich dazu die Arduino OTA[5] Bibliothek verwendet.

[5]Arduino Bibliothek die es ermöglicht neue Sketches über Wlan hochzuladen (github.com/JAndrassy/ArduinoOTA)

2.2.6 Möglichkeiten zur Smart Home Integration

Hinsichtlich der Smart Home Integration wollte ich einerseits dem Benutzer die Möglichkeit geben, die Messdaten der Steckdosenleiste einzusehen und

Funktionen der Steckdosenleiste, wie z.B. das ferngesteuerte Ausschalten, zu ermöglichen. Andererseits wollte ich aber auch anderen Smart Home Geräten die Möglichkeit geben, mit der Steckdosenleiste zu interagieren, weshalb ich mich schließlich dazu entschloss, zwei Benutzeroberflächen zu erstellen, eine für den Benutzer besonders benutzerfreundliche und eine für andere Smart Home Geräte, die nur einen Json String mit den benötigten Informationen ausgibt.

Figure 10: Zeigt die Benutzoberfläche


```
{
    "status": "angeschaltet",
    "seit": "5min",
    "letzter_messwert": "131Watt"
}
```


Figure 11: Zeigt die JSON Ausgabe

3 Ausblick

Obwohl die Steckdosenleiste sowohl programmtechnisch als auch physisch einwandfrei funktioniert und bei mir bereits im Einsatz ist, denke ich immer noch darüber nach, eine neue, verbesserte Version zu entwickeln, die dann ein farbiges Display mit höherer Auflösung und eventuell auch eine Dauerstrom-steckdose hat, damit es einfacher ist, einfach nur sein Handy aufzuladen, ohne dass der Computer eingeschaltet sein muss, außerdem hatte ich auch daran gedacht, einen noch leistungsstärkeren Prozessor und einen Spannungssensor einzubauen, damit genauere Leistungsmessungen möglich sind.