7. Test und Inbetriebnahme

Nach der Entwicklung der Hardware sowie der Implementierung der Software folgt der entscheidende Schritt, in dem beide Komponenten erstmals gemeinsam in Betrieb genommen werden. Um die zuvor erarbeiteten Funktionen zuverlässig und sicher nutzen zu können, ist eine strukturierte Test- und Inbetriebnahmephase unerlässlich. Die Inbetriebnahme umfasst zunächst eine sorgfältige Sicht- und elektrische Prüfung der fertig bestückten Platine, um mögliche Fehlerquellen auszuschließen, die im ersten Betrieb zu Schäden an den Bauteilen oder Zerstörung der Schaltung führen könnten. Erst wenn diese Basisprüfungen erfolgreich abgeschlossen sind, erfolgt der eigentliche Funktionstest, bei dem die Software auf den Chip geflasht wird und sämtliche Funktionen systematisch überprüft werden. Ziel dieser Phase ist es, die korrekte Zusammenarbeit von Hardware und Software sicherzustellen und einen zuverlässigen Betrieb des Gesamtsystems zu gewährleisten.

Die Inbetriebnahme folgt dabei einem reproduzierbaren Schema mit festem Ablauf in 5 Schritten. Im Folgenden sollen diese Schritte erklärt und die Ergebnisse diskutiert werden.

1. Schritt: Sichtprüfung

   Hier geht es darum von vornherein erkennbare Fehler zu finden und diese zu beheben. Dafür wird als erstes die Ausrichtung der Bauteile kontrolliert. Dafür ist sowohl auf dem Bauteil als auch auf der Platine eine Kennzeichnung für Pin 1 für jedes Bauteil mit mindestens 4 Pins hinzugefügt. Diese Markierungen müssen für jeden IC übereinanderliegen, damit das Bauteil korrekt ausgerichtet ist. Neben der Ausrichtung sollte auch die Position der Bauteile noch einmal abschließend kontrolliert werden, ob es beim Lötvorgang zu Verschiebungen auf den Pads kam. Auf der bestellten Platine war das für alle Bauteile der Fall. Diese Prüfung ist vor allem bei händisch selbstbestückten Platinen obligatorisch, bei maschinell bestückten Platinen kann dieser Schritt auch entfallen.

   Der zweite Teil der Sichtprüfungen betrifft die Lötverbindungen. Hier wird geprüft, ob es bereits sichtbare Lötbrücken gibt. Das sind ungewollte Verbindungen zwischen Pins eines meist mehrbeinigen Chips. Diese verursacht einen Kurzschluss zwischen den Pins und kann irreparable Schäden am Bauteil verursachen. Diese müssen daher unbedingt vor dem ersten betrieb entfernt werden. Auf der anderen Seite muss auch auf fehlende Lötverbindungen geachtet werden. Diese fügen den Bauteilen zwar keinen Schaden zu, verhindern aber ebenfalls eine korrekte Funktion der Schaltung. Auch hier waren keine Mängel auf der Platine zu erkennen. Zuletzt wird bei der Sichtprüfung kontrolliert, dass alle ausgeschlossenen Bauteile unbestückt geblieben sind. Dabei handelt es sich meist um Brücken oder 0Ω Widerstände, welche Systeme zu Testzwecken trennen oder Alternativschaltungen vom System abkapseln, um sie nur bei Bedarf zu

nutzen. Im vorliegenden System wäre das der Widerstand R404, welcher zur Einstellung der Alternativen Frequenz des Funkmoduls genutzt wird. Auch diese Prüfung besteht die Platine.

2. Schritt: Widerstandsmessung

In diesem Schritt werden die Widerstände zwischen den Ausgängen aller Spannungswandler gegen Masse geprüft, um Kurzschlüsse innerhalb der Versorgungsspannung frühzeitig zu erkennen. Ein Kurzschluss auf einem Pfad der Versorgungsspannung führt mit sehr hoher Wahrscheinlichkeit zu Zerstörung des Spannungswandlers und häufig auch zur Beschädigung von Bauteilen, welche mit dieser Spannung gespeist werden. Entsprechend wird bei dieser Messung ein möglichst hoher Widerstand erwartet. Die Messung findet immer gegen Ground statt, entsprechend bezeichnet R_{48V} den Widerstand zwischen 48V und Masse.

$$R_{48V} => 20\ M\Omega$$
$$R_{12V} = 4.97\ k\Omega$$
$$R_{5V} = 150\ k\Omega$$
$$R_{3.3V} = 3.4 M\Omega$$

Die sehr unterschiedlichen Widerstände lassen sich dadurch erklären, dass es sich bei dem Spannungswandler auf 3.3V um einen Linearregler handelt, welcher anders funktioniert als die DCDC-Module, welche für 12V und 5V eingesetzt werden. Bei den Modulen lässt sich der sehr geringe Widerstand durch unbekannte interne Beschaltungen, zum Beispiel Filter oder ähnliches, erklären. Zur Referenz wurden die Bauteile unverbaut gemessen und haben ähnliche, vergleichbare Werte ergeben.

3. Schritt: Spannungsmessung

Nachdem sichergestellt ist, dass keine Kurzschlüsse in den Pfaden der Versorgung vorliegen und die Platine auf Lötbrücken überprüft wurde, kann nun zum ersten Mal Spannung angelegt werden. Dabei wird für jede Versorgungsspannung eine Spannungsmessung durchgeführt. Damit soll sichergestellt werden, dass die Spannungswandler korrekt arbeiten. Auftretende Probleme könnten zum Beispiel zu hohe Ausgangsspannungen sein, welche Bauteile zerstören können, zu geringe Spannungen, sodass die Schaltungen nicht korrekt arbeiten oder eine instabile, schwankende Ausgangsspannung. Zur Messung wird die Eingangsspannung eingespeist, um das gesamte System auf einmal zu bestromen.

Beim vorliegenden System werden 48V eingespeist und folgende Spannungen gemessen:

$$U_{12V} = 11.89V$$
$$U_{5V} = 5.03V$$
$$U_{3.3V} = 3.29V$$

Mit den gemessenen Werten ergebenen sich folgende Abweichungen:

$$\Delta_{12V} = \frac{11.89}{12} - 1 = -0.92\%$$
$$\Delta V_{5V} = \frac{5.03}{5} - 1 = 0.6\%$$
$$\Delta V_{3.3} = \frac{3.29}{3.30} - 1 = -0.4\%$$

Die Spannungen liegen damit alle innerhalb $\mp 1\%$. Damit erfüllen alle Spannungswandler die nötigen Anforderungen.

4. Schritt: Stecker bestücken

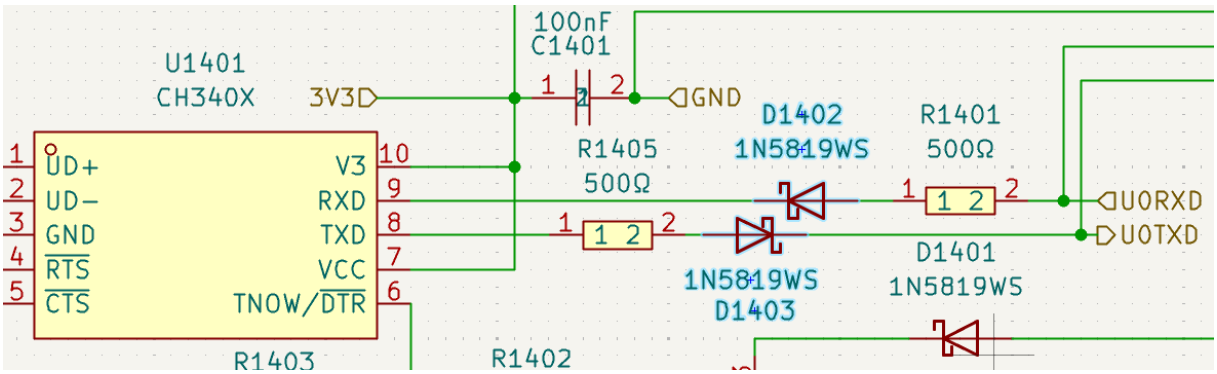
Der vierte Schritt betrifft vor allem Platinen, welche händisch mit Reflow-Lötverfahren bestückt wurden. Da die Stecker oft die Temperaturen nicht aushalten, werden sie nachträglich per Hand bestückt. Außerdem handelt es sich bei Steckern oft um sehr teure Bauteile. Daher werden diese, auch bei anderen Lötverfahren, oft weggelassen und erst dann bestückt, wenn die Platine alle restlichen elektrischen System- und Qualitätstests erfolgreich durchlaufen hat. Im Vorliegenden Fall wurden die Stecker bei der Bestellung bestückt, sodass hier keine Stecker mehr bestückt werden müssen. Stattdessen wurden die Stiftleisten bei der Bestückung ignoriert, da diese für die meisten Systeme nicht relevant sind, sondern nur Schnittstellen zum Debugging und Testen liefern. Auf der ersten Platine, die von einem System bestückt wird, bringen diese sehr großen Nutzen, sodass sie bestückt werden sollen.

5. Schritt: Software flashen

Im letzten Schritt wird nun, nachdem die elektrische Spannungsversorgung validiert ist, die Software aufgespielt, damit im nächsten Schritt die Gesamtfunktionstests durchgeführt werden können. Dafür ist im eine Mikro-USB Buchse vorgesehen mit entsprechender Beschaltung, um die Software über USB vom PC aus aufspielen zu können. Hier treten für die Platine die ersten Fehler auf und das Aufspielen der Software schlägt fehl.

Nachdem im letzten Schritt der Inbetriebnahme, beim Aufspielen der Software, Fehler aufgetreten waren, sollen diese nun dargelegt, erklärt und im besten Fall behoben werden. Der Fehler äußert sich dadurch, dass keine Verbindung zwischen dem PC und dem Mikrocontroller mithilfe des USB-To-Serial Wandlers aufgebaut werden kann. Das liegt meist daran, dass der PC als Master versucht eine Kommunikation zu initiieren, der Mikrocontroller darauf aber nicht reagiert. Da er keine Antwort erhält bricht der PC den verbindungsversuch ab. Bei der Fehlersuche vielen Folgende Fehler auf:

Der USB-To-Serial Wandler ist falsch an den Mikrocontroller angeschlossen, wie in Schaltplan 21 gut zu erkennen ist. Wenn Nachrichten vom Mikrocontroller gesendet werden, werden diese vom Pin U0TXD gesendet. Diese werden vom CH340X empfangen. Das müsste am Pin RXD passieren. In der Vorliegenden Schaltung ist das nicht der Fall, sodass keine Kommunikation zwischen USB und Mikrocontroller stattfinden kann. Dieser Fehler ist nicht behebbar, ohne eine geänderte Version der Platine zu bestellen. Zu Testzwecken könnten die beiden Leiterbahnen aufgekratzt werden und durch Kabel überbrückt werden, das ist platztechnisch aber nur schwer umsetzbar. Zudem ist auf der Platine eine Stiftleiste vorgesehen, welche die Programmierschnittstellen nach außen führt, sodass das Flashen über diese Schnittstelle versucht werden soll.



Schaltplan 21: Schaltplanausschnitt für die USB-Signalleitungen

Dieser Pinheader führt direkt die beiden Pins R0TXD und R0RXD nach außen, sodass nicht direkt per USB über die Kontakte neue Software aufgespielt werden kann. Stattdessen wird eine bereits bestehende Schaltung, wie zum Beispiel ein Entwicklungsboard, verwendet, das für die Platine die Schnittstelle zwischen Mikrocontroller und PC zur Verfügung stellt. Die Schnittstelle als Pinheader bietet den Vorteil, dass die angeschlossenen Kabel einfach vertauscht werden können, damit sich eine korrekte Verbindung ergibt. Für diese Arbeit wird dafür ein ESP32 Devkit V4 verwendet. Dieses führt über Pins seine U0TXD und R0RXD Signale nach außen. Das sind die Signale, welche vom USB-To-Serial Wandler bereits in eine UART-Kommunikation umgewandelt wurden. Die Signale werden damit zwischen Wandler und Mikrocontroller abgegriffen. Diese Signale werden über Kabel auf den Pinheader geführt. Wird nun über die USB-Schnittstelle eine Software aufgespielt, passiert das gleichzeitig auf zwei Chips. Dafür muss der ESP32 auf

der eigenen Platine noch in den korrekten Boot-Modus versetzt werden. Dafür wird GPIO 0 gegen Ground gezogen. Dieser wird auch über die Stiftleiste nach außen geführt, sodass dieser über ein Kabel nur auf Masse gelegt werden muss. Da die beiden Systeme aktuell von zwei unterschiedlichen Spannungsquellen versorgt werden, das Devkit über den PC, die Platine über 48V Spannung vom Labornetzteil, wird zusätzlich noch eine Groundleitung gezogen, um beide Systeme auf dasselbe Massepotenzial zu beziehen. Das Enable-Signal wird nicht beachtet, das sorgt lediglich dafür, dass darüber für die Schaltung automatisch ein Neustart erzwungen wird. Das kann aber auch über einen manuellen Neustart passieren. Leider war auch hierrüber kein Flashen möglich. Zusätzlich wurde als Fehler eine falsche Pinbelegung identifiziert. Im Betrieb sind die Pins für eine UART-Schnittstelle beliebig definierbar, so wie es auch in der Funktion Init_Ports() für die UART-Schnittstelle gemacht wird. Diese sind aber zum Zeitpunkt des Aufspielens der Software noch unbekannt, weswegen dort nur die Standardbelegung verwendet werden kann. Das sind für den ESP32 die Pins GPIO 1 und GPIO 3. Für einen Testaufbau kann dieses Problem umgangen werden, indem einzelne Kabel direkt an die beiden entsprechenden Pins des Mikrocontrollers angelötet werden. Diese werden dann wie bereits vorher mit den Pins U0TXD und U0RXD des Devkit verbunden, um darüber flashen zu können. Die Masseverbindung sowie der Massebezug von GPIO 0 über den Pinheader werden dafür weiterhin genutzt. Das Problem lässt sich darüber jedoch auch nicht beheben. Da der Mikrocontroller nicht mit Software bespielbar ist und rund um die USB-Beschaltung bereits mehrere Fehler aufgetaucht sind, muss das System in diesem Hinblick als Funktionsunfähig betrachtet werden. Alle Versuche das Problem zu reparieren oder zu umgehen haben keine Besserung gebracht. Daher muss die USB-Beschaltung grundlegend für eine Version 2 überarbeitet werden. Auch die Pinbelegung muss angepasst werden, sodass die UART-Schnittstelle die standardmäßig vorgesehenen Pins nutzt, und die Verbindung zwischen Sende- und Empfangspins muss korrigiert werden. Leider ist es aufgrund der Fertigungs- und Lieferzeit von mehreren Wochen nicht möglich eine verbesserte Version noch im Rahmen dieser Bachelorarbeit zu betrachten. Der letzte fehlende Prozess ist der Test aller Funktionen. Durch die fehlende Möglichkeit Software auf die Platine aufzuspielen, ist ein umfassender Funktionstest der elektrischen Schaltung nicht möglich. Die einzige Funktion, welche vollständig ohne den Mikrocontroller auskommt, ist die Erzeugung des Notaus Signals. Zum Test der Software besteht die Möglichkeit das Devkit zu nutzen. Leider ist auch hier nur ein Teil der Software testbar, da die Schnittstellen für CAN, sowie die Porterweiterung oder Zusatzbeschaltungen fehlen. Auch ist keines der über I^2C oder SPI anzusteuernden Geräte testbar, da diese nicht als bedrahtete Bauteile vorrätig sind, um die Schaltung mittels eines Steckbretts aufzubauen. Unter den testbaren Funktionen befindet sich ein Großteil der im Setup ausgeführten Initialisierungsfunktionen. Bis auf Init_GPIO_Exp_Ports() und init_CAN2() sind alle Funktionen im Setup ausführbar. Die Initialisierung für die beiden anderen sind nicht möglich, da die beiden Bauteile für das Devkit nicht zur Verfügung stehen und somit die Schreibfunktionen über SPI und I^2C fehlschlagen und Fehler verursachen würden. Die Funktionen init_Interrupts() und

init_ports() werden ausgeführt, um Fehler im Funktionsablauf zu entdecken. Die tatsächlichen Funktionen der Pins werden nicht einzeln geprüft. Ähnliches gilt für die Funktion init_CAN1(). Die Funktion wird ausgeführt, um zu verhindern, dass bei der Initialisierung Fehler auftreten. Die tatsächliche CAN-Kommunikation kann nicht getestet werden, da kein CAN-Transceiver verfügbar ist. Der CAN-Controller hingegen ist für CAN1 im ESP integriert. Die Funktion init_storage() wird aktiv getestet. Dafür wird die Funktion print_NVS() jeden 5000. Loopdurchlauf aufgerufen und gibt den Inhalt des persistenten Speichers textuell über die Serielle Schnittstelle aus. So wird jede Änderung im NVS ausgegeben, um für den Entwickler nachvollziehbar zu sein. Die Änderungen im Speicher nimmt unter anderem eine der Webseiten vor, welche über den Webserver zur Verfügung gestellt wird. Dafür wird die Funktion init_wifi() ausgeführt, welche den ESP als Accesspoint konfiguriert. Die Netzwerkschnittstelle ist die einzige, welche komplett ohne Peripherie nur vom Mikrocontroller zur Verfügung gestellt wird. Daher können alle Funktionen vollständig getestet werden. Zum einen kann der Aufbau der Webseiten kontrolliert werden und bei Notwendigkeit optisch angepasst werden. Zum anderen werden die Funktionen getestet. So im Fall der Livedaten Seite. Hier wird der Loop genutzt, um Livedaten zu simulieren. Der Loop übernimmt dabei die Funktion der Auswertung von CAN1 und beschreibt dieselben Variablen, welche auf der Livedatenseite dargestellt werden. Dieser neue Codeabschnitt ist in Programmcode 47 dargestellt.

```
174     loop_ctr++;
175
176     // Dummy Werte statt CAN
177     if(loop_ctr % 900 == 0){
178       CAN_speed += 5.3;
179       CAN_battery_temp += 0.1;
180       CAN_soc -= 0.2;
181       if(CAN_speed > 120.0){CAN_speed = 0.0;}
182       if(CAN_battery_temp > 45.0){CAN_battery_temp = 20.0;}
183       if(CAN_soc < 0.0){CAN_soc = 100.0;}
184     }
```

Programmcode 47: Simulation von variablen Daten statt CAN für Livedaten

Damit wird vor allem getestet, ob die Webseite mit der Datenrate zurechtkommt, oder ob es Probleme beim Laden der Seite oder dem Aufrechthalten der Verbindung gibt. All das ist nicht der Fall, die Livedaten Funktionalität kann problemlos erfüllt werden. Selbiges gilt für die Einstellungen und die Herstellerseite. Die Funktionalität ist vollständig testbar. Vor allem die Änderung der Einstellungen kann gut nachvollzogen werden durch die Funktion print_NVS(). Lediglich das Anlernen der Funkfernbedinung kann nicht überprüft werden, da weder das Funkmodul, noch die Porterweiterung verfügbar sind. Ansonsten ist die Netzwerkschnittstelle vollständig testbar und funktionsfähig. Alle über das Netzwerk realisierten Funktionen funktionieren Fehlerfrei.