



University of Applied Sciences

**Fachhochschule  
EMDEN·LEER**

Fachbereich Technik, Abteilung E+I

# **Projektarbeit**

## **WS 2021 / 2022:**

### **Entwurf, Design, Herstellung und Test eines Maximum Power Point Trackers für Solarpanel**

**Studiengang:**  
Elektrotechnik im Praxisverbund

**Bearbeitet von:**  
Hans Wilhelm Zeuschner

**Betreuer:**  
Prof. Dr.-Ing. Gavin Kane

# Inhaltsverzeichnis

<b>1 Eidesstattliche Erklärung</b>	<b>4</b>
<b>2 Einleitung</b>	<b>5</b>
2.1 Aufgabenstellung . . . . .	6
2.2 Anforderungen an das neue Design . . . . .	7
2.3 Bisheriger technischer Stand . . . . .	8
2.3.1 Bilder des bisherigen MPPT . . . . .	8
2.3.2 Probleme mit dem bisherigen MPPT . . . . .	10
<b>3 Hauptteil</b>	<b>12</b>
3.1 Systemüberblick: Blockschaltbild . . . . .	13
3.2 Verwendete CAD-Software für das Platinendesign . . . . .	14
3.3 Auswahl der Bauelemente . . . . .	14
3.4 Entwicklung des Schaltplans . . . . .	15
3.4.1 Mikrocontroller, Kommunikation und lokale Spannungsversorgung . . . . .	16
3.4.1.1 Unterstützende Bauelemente . . . . .	17
3.4.1.2 Lokale Spannungsversorgung . . . . .	19
3.4.1.3 Debugging-/ Programmierinterfaces und Testpunkte . . . . .	22
3.4.1.4 EEPROM . . . . .	24
3.4.1.5 CAN Bus Transceiver . . . . .	24
3.4.2 Hochsetzsteller . . . . .	25
3.4.2.1 Auswahlentscheidungen Hochsetzsteller . . . . .	26
3.4.2.2 Gate-Treiber für den Hochsetzsteller . . . . .	30
3.4.2.3 INA226 Leistungssensor . . . . .	32
3.4.2.4 Ausgangsspannungsmessnetzwerk . . . . .	33
3.4.3 Ein- und Ausgangsschutzbeschaltung . . . . .	35
3.4.4 mechanische Elemente . . . . .	36
3.5 Platinenlayout . . . . .	37
3.5.1 Kupferlagen der Platine . . . . .	37
3.5.2 3D-Render der Platine . . . . .	40
3.5.3 Ein- und Ausgangsschutzschaltung . . . . .	42
3.5.4 Hochsetzsteller . . . . .	43
3.5.4.1 Leistungsmessung . . . . .	43
3.5.4.2 Leistungsteil . . . . .	44
3.5.4.3 Gate Treiber . . . . .	45
3.5.5 Spannungsversorgung des Mikrocontrollers . . . . .	46
3.5.6 Mikrocontroller . . . . .	48
3.5.6.1 Spannungsteiler zur Messung der Ausgangsspannung und Oszillatoren . . . . .	49
3.5.6.2 Debug-Interfaces . . . . .	49
3.5.6.3 CAN-Bus . . . . .	50
3.6 Bestücken der Platine . . . . .	51
3.7 Entwicklung der Software . . . . .	53
3.7.1 Allgemeines . . . . .	53

3.7.2	Funktion des MPP-Algorithmus . . . . .	56
3.7.3	Weitere Programmkomponenten . . . . .	61
3.7.3.1	Parametrisierung . . . . .	61
3.7.3.2	INA226 Leistungssensor . . . . .	61
3.7.3.3	MPPT IO . . . . .	62
3.7.3.4	CAN Schnittstelle . . . . .	62
3.8	Verifikation und Messung . . . . .	63
3.8.1	Sicherheitsrelevante Messungen . . . . .	64
3.8.1.1	Transient / Sprungantwort . . . . .	64
3.8.1.2	Zu niedrige Eingangsspannung . . . . .	65
3.8.2	Verifikation des MPPT-Algorithmus . . . . .	66
3.8.2.1	Theorie . . . . .	66
3.8.2.2	Messungen . . . . .	68
3.8.3	Belastungstest . . . . .	69
3.8.4	Wirkungsgrad . . . . .	71
3.8.5	Paralleler Betrieb mehrerer MPPT an einem gemeinsamen Akku	74
3.8.6	Messungen mit Solarpanel und Messungen an einem Li-Ion Akku . . . . .	74
<b>4</b>	<b>Schlusswort und Ausblick</b>	<b>75</b>
4.1	Einflüsse der COVID-19 Pandemie . . . . .	75
4.2	Weitere Schritte . . . . .	76
4.3	Endergebnis . . . . .	76
4.4	Danksagung . . . . .	78
<b>5</b>	<b>Anhang</b>	<b>80</b>
5.1	Bildverzeichnis . . . . .	80
5.2	Quellcode . . . . .	81
5.2.1	mppt_main.cpp . . . . .	81
5.2.2	mppt_config.h . . . . .	83
5.2.3	mppt_pins.h . . . . .	85
5.2.4	ina_sensor.h . . . . .	86
5.2.5	ina_sensor.cpp . . . . .	87
5.2.6	mppt_control_algo.h . . . . .	89
5.2.7	mppt_control_algo.cpp . . . . .	90
5.2.8	mppt_io.h . . . . .	93
5.2.9	mppt_io.cpp . . . . .	94
5.2.10	platformio.ini . . . . .	97
5.3	Schaltplan . . . . .	98
5.4	Platinenlayout . . . . .	104
5.5	Abkürzungsverzeichnis . . . . .	127

# **1 Eidesstattliche Erklärung**

Ich, der Unterzeichnende, erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit selbständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Alle Quellenangaben und Zitate sind richtig und vollständig wiedergegeben und in den jeweiligen Kapiteln und im Literaturverzeichnis wiedergegeben. Die vorliegende Arbeit wurde nicht in dieser oder einer ähnlichen Form ganz oder in Teilen zur Erlangung eines akademischen Abschlussgrades oder einer anderen Prüfungsleistung eingereicht. Mir ist bekannt, dass falsche Angaben im Zusammenhang mit dieser Erklärung strafrechtlich verfolgt werden können.

Emden, den 31.03.2022

## 2 Einleitung

Das Solarboot-Team der Hochschule Emden Leer nimmt mit einem selbst entwickelten und gebauten Solarboot an zahlreichen Wettbewerben teil. Ziel dieser Wettbewerbe ist es, möglichst schnell zu fahren oder innerhalb einer festen Zeit, mit nur einer Akkuladung und der Unterstützung durch Sonnenenergie, eine möglichst weite Strecke zurückzulegen.

Die Regularien der Rennen [13] schreiben genau vor, wie groß die maximale Solarfläche sein darf. Daher ist es von großer Bedeutung, die zur Verfügung stehende Sonnenenergie so effizient wie möglich zu nutzen.

Die elektrische Leistung, die ein Solarpanel zur Verfügung stellt, ist von vielen Faktoren abhängig. Der wichtigste Einfluss ist die Intensität der Sonneneinstrahlung. Weitere Faktoren sind Temperatur, Einfallsinkel und Verschmutzungsgrad. Alle aufgelisteten Parameter lassen sich im Anwendungsfeld Solarboot kaum beeinflussen. Die Solarpanels sind fest montiert und lassen sich nicht verstehen oder neigen. Auch der Verschmutzungsgrad bzw. Verdeckungsgrad während einer Fahrt über Wasser variiert.

Neben den bereits genannten Faktoren gibt es eine weitere wichtige Möglichkeit, die Ausgangsleistung eines Solarpanels zu maximieren, den sogenannten MPPT [12]. Im Anwendungsgebiet Solarboot ist dieser besonders interessant, da sich durch spezielle Elektronik der Maximum Power Point, kurz MPP, finden lässt. Dieser verschiebt sich durch die Veränderung der Sonneneinstrahlung etc. Ein Maximum Power Point Tracker, kurz MPPT, kann diesen MPP intelligent kontinuierlich tracken und somit gewährleisten, dass zu jedem Zeitpunkt das Maximum an Leistung aus einem Panel gewonnen werden kann.

Der neu entwickelte MPPT ist erheblich kleiner, leichter und etwas günstiger in der Herstellung bzw. Beschaffung, als die alte Entwicklung (vgl. Tab. 2). Außerdem ermöglicht die Neuentwicklung, dass alle Solarpanel des Bootes an einem eigenen MPPT angeschlossen werden, was den Gesamtwirkungsgrad anhebt.



Abbildung 1: Frontansicht des neu-entwickelten MPPT

## 2.1 Aufgabenstellung

Diese Projektarbeit befasst sich damit, einen Maximum Power Point Tracker zu entwickeln, herzustellen und zu testen. Dieser ist genau auf die Anwendung in dem Solarboot zugeschnitten, es ist aber auch eine Anwendung in einer anderen Umgebung vorstellbar.

In der Vergangenheit wurde an der Hochschule bereits ein MPPT für den Anwendungsfall Solarboot entwickelt. Dieser war allerdings nie im Einsatz, da pro Boot mehrere MPPTs nötig sind und nie genug von dem Modell gebaut wurden. Auf den bisherigen MPPT wird in Abschnitt 2.3 genauer eingegangen.

Jedes Solarpanel sollte idealerweise an einem eigenen MPPT angeschlossen werden, da die Sonneneinstrahlung pro Panel unterschiedlich sein kann. Wenn mehrere Panels in Serie oder parallel betrieben werden, so sinkt in der Praxis der Wirkungsgrad. Dies war bis zum Bau der in dieser Ausarbeitung beschriebenen MPPT der Fall. Im Solarboot wurde zu diesem Zeitpunkt ein BlueSolar MPPT 150/35 der Firma Victron verwendet verwendet. An diesem waren alle Panels des Bootes in einer Serien-/Parallelschaltung angeschlossen.

## 2.2 Anforderungen an das neue Design

Die Tabelle 1 fasst die relevanten Anforderungen an das neue Design zusammen. Diese wurden in Kooperation mit dem bestehenden Solarboot-Team ausgearbeitet. Die Anforderungen der Eingangsspannung und des Eingangsstroms richten sich an den verbauten Solarpanels aus. Da ein Lithium-Ionen-Akku mit Ladeschlussspannung von  $50,4V$  verwendet wird, muss die Leistungsstufe des MPPT mindestens diese Ausgangsspannung erzeugen können. Aus den Werten der Tab. 1 resultiert somit eine theoretische Gesamtleistung von  $36V * 6A = 216W$  am Eingang, mit der der Akku geladen werden kann. In der Praxis wird die Leistung eines einzelnen Solarpanels allerdings nie dieses Niveau erreichen, da die Panelspannung unter Last absinkt. Somit ist eine maximale Leistung von  $26V * 6A = 156W$  realistischer.

<b>Eingangsspannung</b>	$\leq 36V$
<b>Eingangsstrom</b>	$\leq 6A$
<b>Ausgangsspannung</b>	$\leq 51V$

Tabelle 1: Anforderungen an den MPPT

Weitere Anforderungen an den MPPT sind:

- gleichzeitiger Betrieb mehrerer MPPT an einem Akku
- geringere Kosten pro Gerät (ca. 70€)
- leichte und kleine Bauform
- hoher Wirkungsgrad ( $> 90\%$ )
- CAN-Interface für Daten-Logging
- Konformität mit den Regularien der Solar-Sport-One-Rennen [13]

### **2.3 Bisheriger technischer Stand**

Der zuvor im Rahmen mehrerer studentischer Arbeiten entwickelter MPPT funktioniert grundsätzlich, hat aber einige Probleme und Nachteile. Daher wurde entschlossen, diesen nicht weiter zu verbessern, sondern eine Neuentwicklung zu beginnen. Die Tab. 2 stellt die wesentlichen Unterschiede zwischen dem alten und neuen Design dar.

	Länge	Breite	Höhe	Gewicht	Kosten	Gesamtersparnis pro Boot
<b>alter MPPT</b>	100mm	100mm	70mm	800g	ca. 100€	-
<b>neuer MPPT</b>	80mm	65mm	24mm	80g	ca. 70€	3,42kg bzw. 180€

Tabelle 2: Vergleich der MPPTs

Die Gesamtersparnis wurde basierend auf der Annahme berechnet, dass pro Boot sechs MPPT verwendet werden. Pro verbauten Solarpanel wird ein MPPT benutzt.

### 2.3.1 Bilder des bisherigen MPPT

Im Folgenden sind drei Fotos des bisherigen MPPT dargestellt.

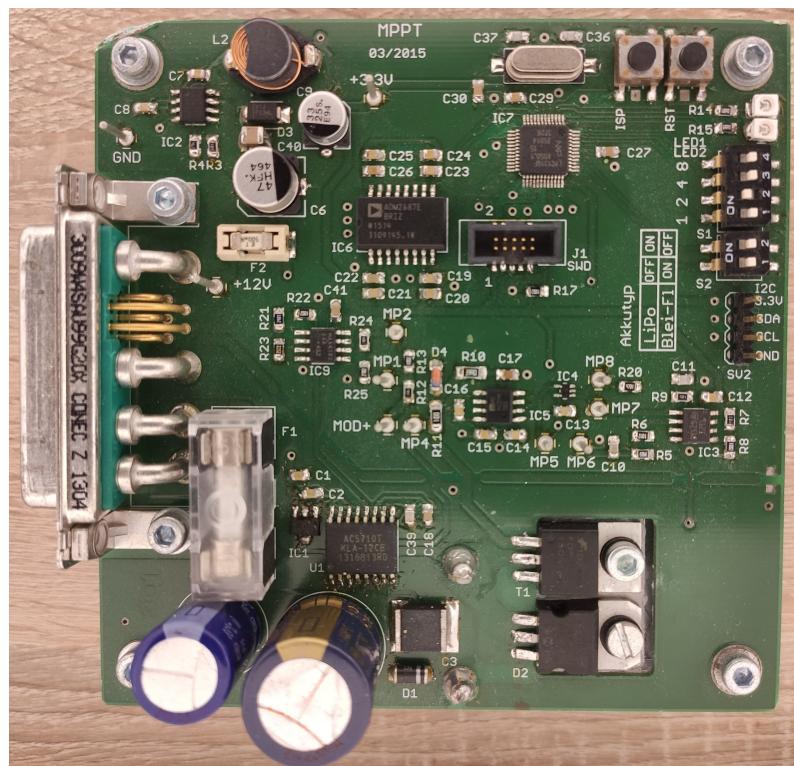


Abbildung 2: Frontansicht bisheriger MPPT

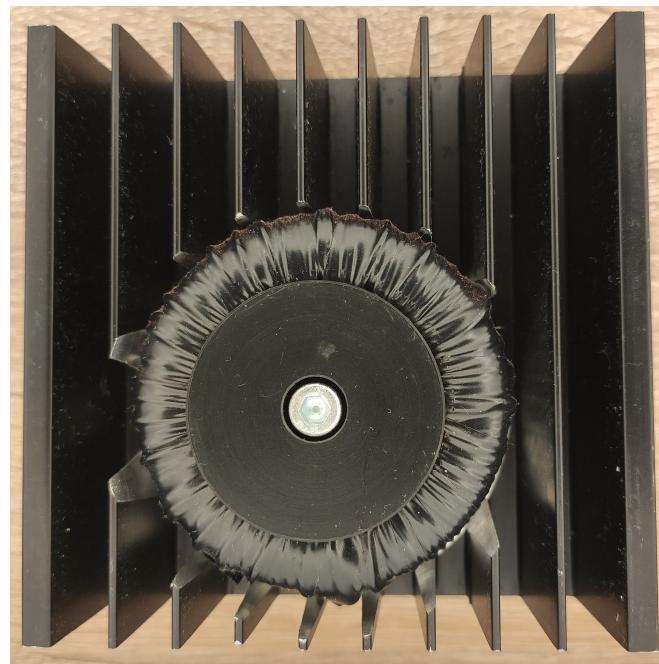


Abbildung 3: Unterseite bisheriger MPPT

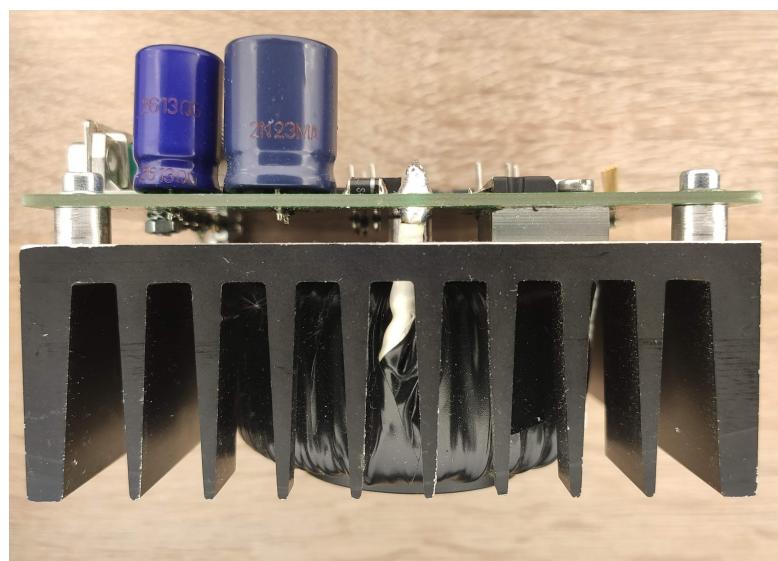


Abbildung 4: Seitenansicht bisheriger MPPT

### 2.3.2 Probleme mit dem bisherigen MPPT

Wie in Tab. 2 dargestellt, zeichnet sich das bisherige Design vor allem durch die große Bauform und ein hohes Gewicht aus. Dies wird auch auf den Abb. 2 bis 6 deutlich.

Der Aluminiumkühlkörper ist sehr groß und schwer. Außerdem muss in diesen eine Aussparung hineingefräst werden, was mit zusätzlicher Arbeit und Kosten verbunden ist. Die Spule ist eine Sonderanfertigung, die nicht über übliche Distributoren für elektronische Bauelemente zu kaufen ist.

Wie der Abb. 2 zu entnehmen ist, ist der MPPT durch einen Sturz beschädigt worden. Durch das hohe Gewicht der Spule und des Kühlkörpers war der Aufprall so stark, dass die obere linke Ecke der Platine eingedrückt wurde und der Ferritkern der kleinen SMD-Spule gebrochen ist. Ein leichteres Design ist weniger anfällig für solche Beschädigungen.

Die Größe der Spule und des Kühlkörpers geben die Platinengröße vor, die trotz der relativ geringen Anzahl von Bauelementen aus diesem Grund relativ groß ist.

Insgesamt handelt es sich um ein sehr großes und schweres Gerät.

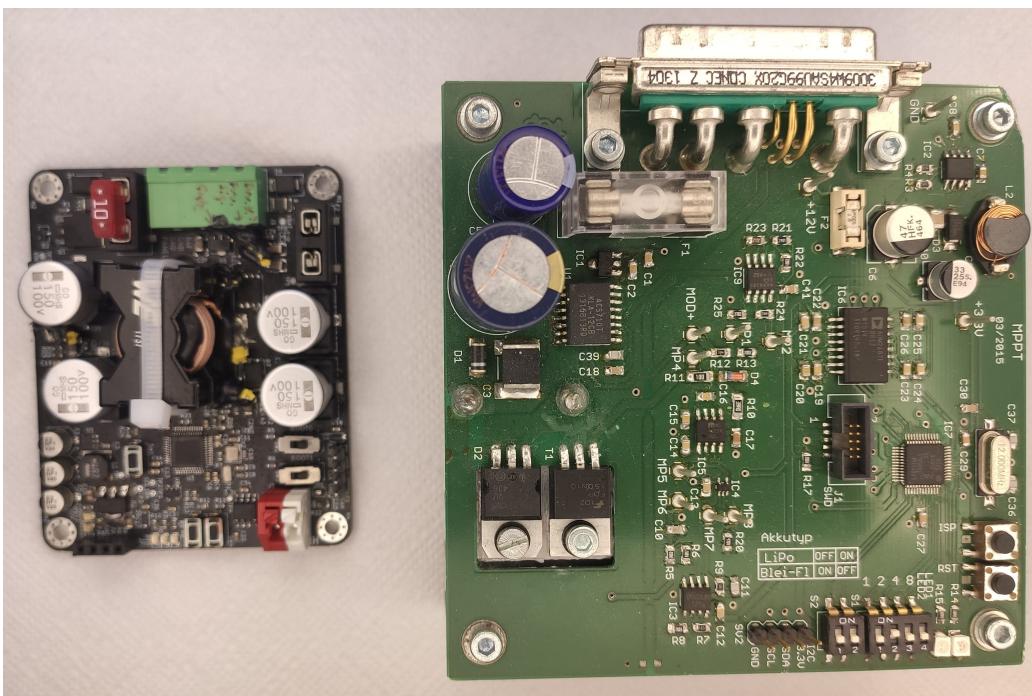


Abbildung 5: Vergleich zwischen dem neuen MPPT (links) und dem alten MPPT (rechts): Frontansicht

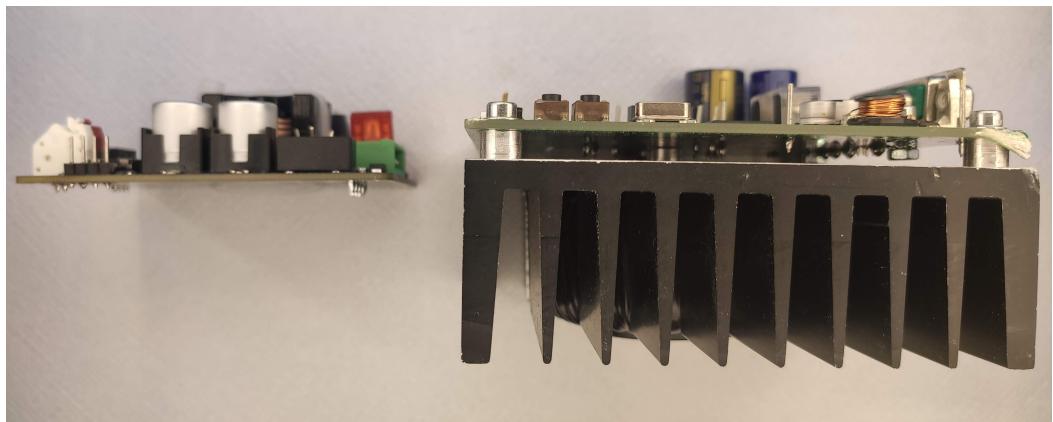


Abbildung 6: Vergleich zwischen dem neuen MPPT (links) und dem alten MPPT (rechts): Seitenansicht

### **3 Hauptteil**

In diesem Kapitel werden alle Etappen der Hardware-Entwicklung, der Verifikation und der Software-Programmierung beschrieben.

Die Abb. 7 zeigt das Blockschaltbild des MPPTs. In den folgenden Kapiteln werden die einzelnen Komponenten des Blockschaltbilds aufgegriffen und die relevanten technischen Designentscheidungen werden erläutert.

### 3.1 Systemüberblick: Blockschaltbild

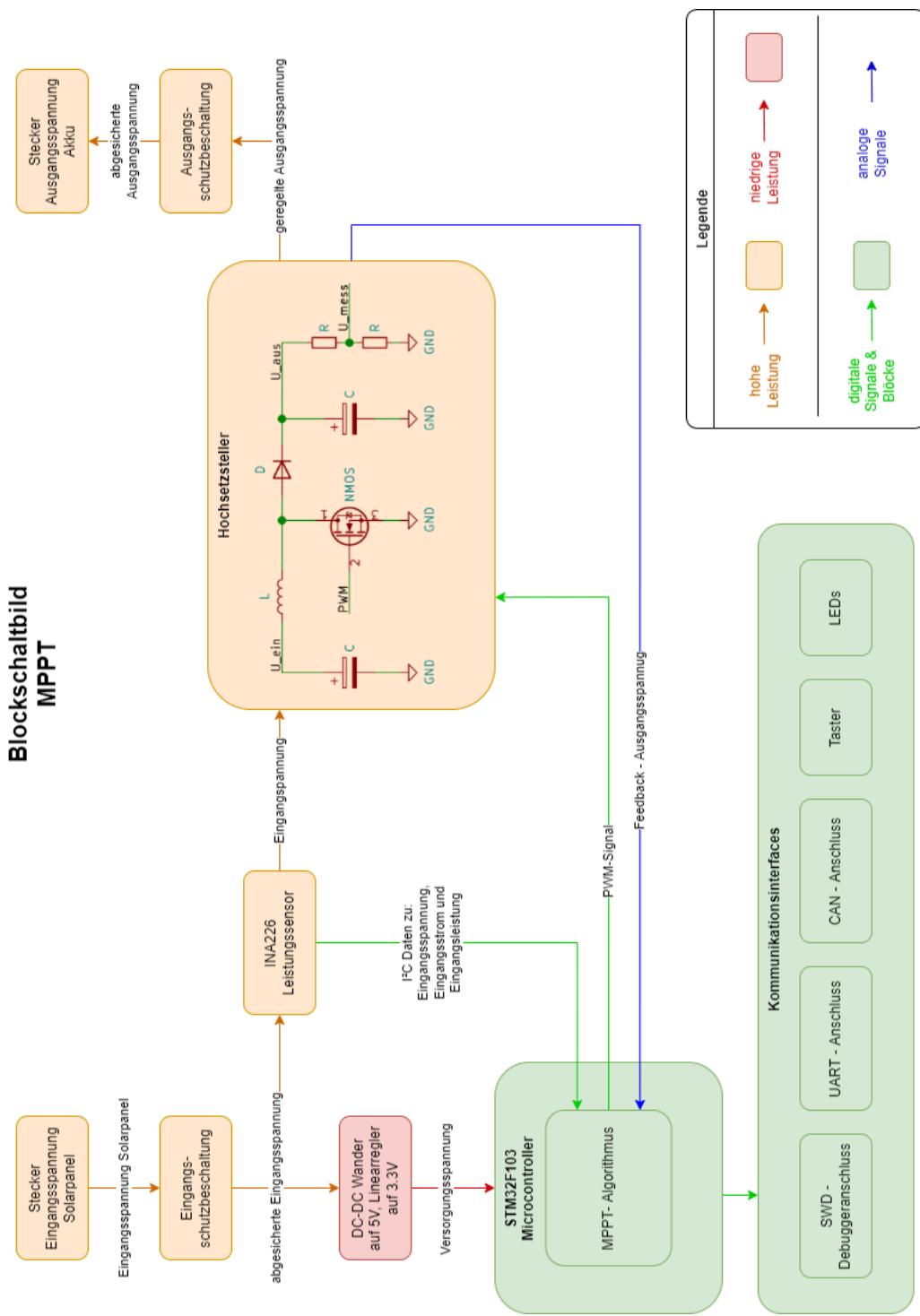


Abbildung 7: Blockschaltbild des MPPT

## 3.2 Verwendete CAD-Software für das Platinendesign

Im Rahmen dieser Projektarbeit wurde die Open-Source Software KiCAD zum Erstellen der Schaltpläne und dem Platinenlayout verwendet.

## 3.3 Auswahl der Bauelemente

Grundsätzlich musste eine Vielzahl an Bauelementen ausgewählt werden für die Platine. Dabei mussten je nach Art des Bauteils verschiedene Faktoren gegeneinander abgewogen werden. Außerdem wurde darauf geachtet, so wenige individuelle Bauteilvarianten zu verwenden, wie möglich. Dies ist insbesondere relevant für passive Bauelemente, wie Widerstände und Kondensatoren. Je weniger verschiedene Werte von z.B. Widerständen auf der Platine zu verlöten sind, desto einfacher und schneller ist die Bestückung. Außerdem ist es unwahrscheinlicher, dass fälschlicherweise zwei völlig verschiedene Widerstandswerte beim Bestücken miteinander verwechselt werden. In vielen Fällen ist es so, dass die Schaltung auch mit einem leicht modifizierten Widerstand / Kapazität vollkommen korrekt funktioniert. Daher ist es in der Regel möglich, z.B. anstelle eines  $4,7k\Omega$  Widerstands einen  $10k\Omega$  Widerstand zu verwenden. Allerdings gilt diese Regel keinesfalls immer. Es muss individuell für jedes Bauteil entschieden werden, ob eine Substitution in Frage kommt.

Da die Platine möglichst klein sein sollte, aber die Bauteile noch per Hand verlötet werden sollten, wurde entschieden, dass die Bauform 0603 für die SMD-Bauteile (Widerstände und Kondensatoren) verwendet wird. Bei den anderen Bauelementen wurde darauf geachtet, dass das kleinste vom Hersteller angebotene Gehäuse verwendet wurde, das noch von Hand verlötet werden konnte. Häufig wird ein Bauteil in mehreren verschiedenen SMD oder THT Bauformen angeboten.

Um Platz auf der Platine zu sparen, wurden mit Ausnahme der Stecker nur SMD-Bauelemente verwendet.

Zu den wichtigsten Parametern zählen der Preis, die Größe und die Verfügbarkeit. Hinzu kommen eine Vielzahl von bauteilspezifischen Eigenschaften. Wenn zum Beispiel ein Elektrolytkondensator für ein Schaltnetzteil verbaut werden soll, so ist die Kapazität relevant. Außerdem ist wichtig, dass der interne Widerstand möglichst niedrig ist. Aber auch die maximal zulässige Betriebstemperatur und die vom Hersteller angegebene Lebensdauer sind wichtig, um zu gewährleisten, dass das Bauteil nicht verfrüht zerstört wird.

Zur Auswahl der Bauteile wurde die Seite des Elektronikbauteile-Distributors Dikey verwendet. In manchen Fällen, falls zum Beispiel ein gut geeignetes Bauteil nicht auf Lager war, wurde auf der Seite eines anderen Anbieters (Mouser) nachgesehen, ob es dort verfügbar war. Beide Anbieter verfügen auf ihrer Seite über eine parametrierbare Such- und Filterfunktion, die sehr hilfreich bei der Eingrenzung der Zahl von relevanten Ergebnissen ist bei der Suche nach einem Bauteil.

Die Auswahl der Bauteile und die Erstellung des Schaltplans wurden parallel abgearbeitet. Für jedes Bauteil ist in KiCAD das passende Footprint zu hinterlegen. Es

ist daher sinnvoll, das Bauteil direkt auszusuchen, damit das Footprint direkt beim Zeichnen des Schaltplans hinterlegt werden kann.

Bei einigen Bauteilen existierte noch kein vorinstalliertes Footprint und teilweise auch kein Symbol, das auf dem Schaltplan zur angezeigt wird. In diesen Fällen musste das fehlende Symbol oder Footprint in dem eingebauten Editor gezeichnet werden.

### 3.4 Entwicklung des Schaltplans

Bei der Entwicklung des Schaltplans wurde mit der grundsätzlichen Strukturierung der einzelnen Komponenten in Blöcke begonnen. So wurden zum Beispiel alle Bauelemente, die unmittelbar zum Hochsetzsteller gehören auf einem Blatt des Schaltplans gruppiert, die Bauteile, die primär zur Funktion des Mikrocontrollers gehörten, wurden wiederum auf einer eigenen Seite gesammelt. Die Abb. 8 zeigt die Unterteilung des Schaltplans in die folgenden Blöcke:

- Mikrocontroller
- Hochsetzsteller
- Ein- und Ausgangsschutzbeschaltung
- mechanische Elemente (Befestigungslöcher etc.)

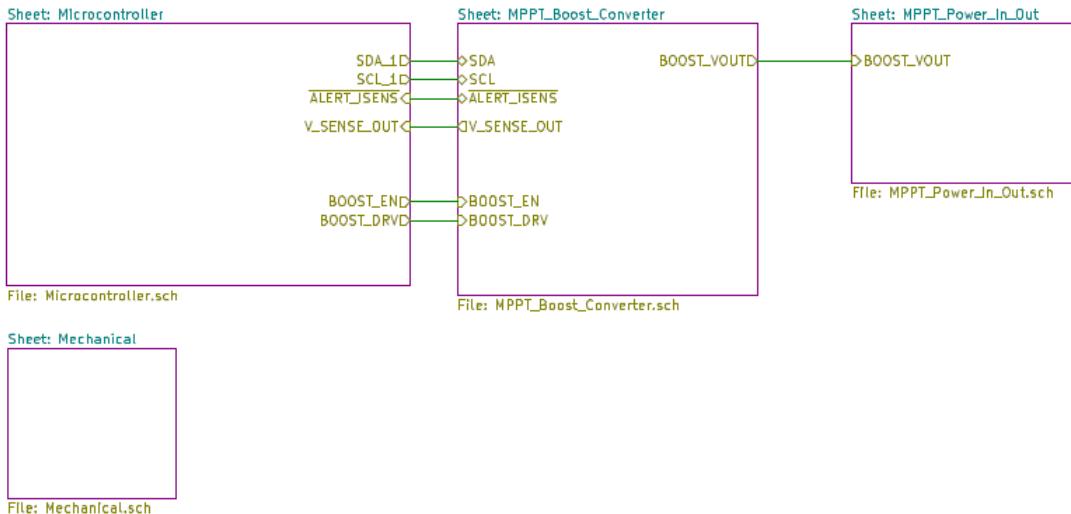


Abbildung 8: Übersicht Schaltplan

Wichtig zu beachten ist, dass globale Netzlabel, wie z.B. Masse und Spannungslabel nicht auf dieser Übersicht zu sehen sind. Das bedeutet, dass nicht nur eine einzige elektrische Verbindung zwischen dem Block MPPT\_Boost\_Converter und MPPT\_Power\_In\_Out besteht. Der Übersichtlichkeit halber sind auf dieser Übersicht nur die relevanten Signale dargestellt.

### 3.4.1 Mikrocontroller, Kommunikation und lokale Spannungsversorgung

Der MPPT benötigt einen Mikrocontroller, auf dem der Regelalgorithmus ausgeführt wird. Dieser muss ausreichend schnell arbeiten, damit sichergestellt ist, dass der MPP kontinuierlich gefunden und gehalten werden kann. Aufgrund vorheriger Erfahrungen mit dem Bauteil, wurde entschieden, dass ein STM32F103C8<sup>1</sup> [15] eingesetzt werden soll. Dieser Mikrocontroller läuft mit einem max. Takt von 72MHz, hat viel Peripherie und ist somit gut für den Einsatzzweck geeignet. Weiteres, wichtiges Kriterium bei der Auswahl war, dass der Mikrocontroller mit der Arduino IDE programmierbar sein sollte. Die IDE ermöglicht eine vergleichsweise simple Programmierung von eingebetteten Systemen und es existiert eine große Zahl von Open-Source Libraries, so dass die Treiber verwendeter HW-Komponenten nicht alle neu entwickelt werden mussten.

Die Abb. 9 zeigt die Verschaltung des Mikrocontrollers. Es wurden alle Signale unter Verwendung von Netzlabels angeschlossen. Die Spannungsversorgung von 3.3V ist aufgeteilt zwischen dem digitalen und analogen Teil des Prozessors. Die Masseverbindung des analogen Anschlusses ist im Schaltplan ebenfalls dezidiert gezeichnet worden, um die Trennung von analogem und digitalen / störungsbelasteten Teil zu unterstreichen. Die Abb. 12 zeigt die zusätzliche Filterung der analogen Versorgungsspannung. Diese ist nötig, damit der ADC so genau wie möglich arbeitet und Störungen, die z.B. durch das Schalten digitaler Gatter etc. auftreten, nicht unmittelbar auf den sensitiven ADC-Eingang übertragen werden.

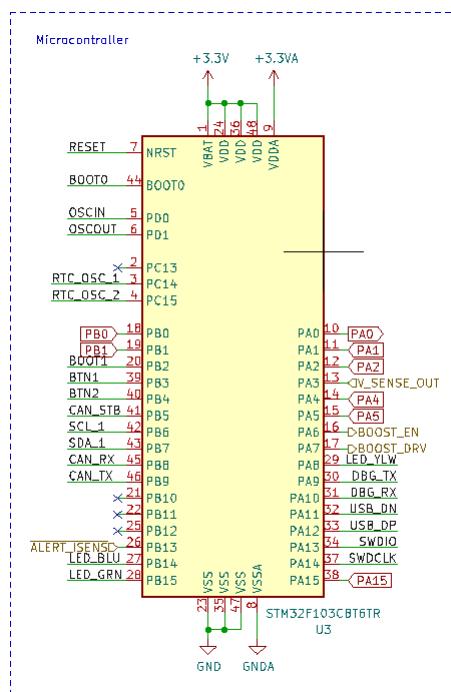


Abbildung 9: Anschlüsse am Mikrocontroller

<sup>1</sup>Alternativ kann auch der Mikrocontroller mit der Teilenummer STM32F103CB eingesetzt werden. Dieser besitzt zusätzlichen Speicher, ist aber kompatibel mit dem STM32F103C8.

### 3.4.1.1 Unterstützende Bauelemente

Die Abb. 10 zeigt die Bauteile, die für die normale Funktion des Mikrocontrollers nötig sind.

Der Reset-Taster (Abb. 10 l.o.) kann verwendet werden, um den Mikrocontroller, zum Beispiel während der Softwareentwicklung, zurückzusetzen. Es handelt sich hierbei um ein active-low Signal, das heißt, dass der Reset erfolgt, wenn das Signal am Eingang auf Massepotential ist. Im Mikrocontroller ist ein schwacher Pull-Up-Widerstand integriert, der das Signal am Reset-Pin auf 3.3V zieht. Über den Kondensator C2 ist somit in Verbindung mit dem internen Widerstand ein RC-Tiefpass realisiert. Dieser hat die Aufgabe, beim Einschalten des Systems ein High-Signal am Reset-Eingang in etwa so lange zu verzögern, bis alle Betriebsspannungen stabil sind. Nach dieser Zeit ist C2 geladen und der Mikrocontroller wechselt aus dem Reset in Run. Wenn der Taster Reset gedrückt wird, so wird C2 entladen und der Reset-Pin auf Masse gezogen. Sobald der Taster losgelassen wird, erfolgt der gleiche Ladeprozess des Kondensators wie soeben geschildert.

Der STM32-Mikrocontroller besitzt verschiedene Boot-Modi. Über die Schalter Boot0 und Boot1 (Abb. 10 l.u.) können diese bei einem Reset ausgewählt werden. Diese Schalter sind im Regelfall nicht notwendig, da ein Wechsel des Boot-Modus im Kontext dieses Projekts nicht nötig ist. Wenn die Programmierung des Mikrocontrollers über UART vorgenommen werden soll, dann ist es nötig, in den passenden Boot-Modus zu wechseln. Allerdings kann der Mikrocontroller auch über die SWD-Schnittstelle programmiert werden, was keinen Wechsel des Boot-Modus erfordert. Daher wurde auch nur diese Methode verwendet.

Die sechs Abblockkondensatoren (Abb. 10 r.o.) dienen dazu, die Betriebsspannung an den Versorgungspins des Mikrocontrollers stabil zu halten. Durch die schnellen internen Schaltvorgänge sind häufig kurzzeitig hohe Ströme zum Laden und Entladen von Gattern nötig. Die Kondensatoren können diesen Strom bereitstellen.

Der Arbeitstakt des Mikrocontrollers wird durch den Quarzoszillator Y2 (Abb. 10 r.u.) erzeugt. Hierbei handelt es sich lediglich um einen 8MHz Oszillator. Intern kann der Mikrocontroller durch einen PLL einen höheren Takt aus diesem 8MHz Signal erzeugen. Zusätzlich hierzu ist ein 32,768kHz Oszillator (Y1) vorhanden. Dieser kann dazu verwendet werden, die interne RTC des STM32 mit einem Takt zu versorgen. An beiden Oszillatoren sind Lastkapazitäten angeschlossen. Diese sind in ihrem Wert abhängig von dem ausgewählten Oszillator. Es ist wichtig, dass der korrekte Wert verwendet wird, damit die Oszillationsfrequenz des Kristalls nicht verschoben wird.

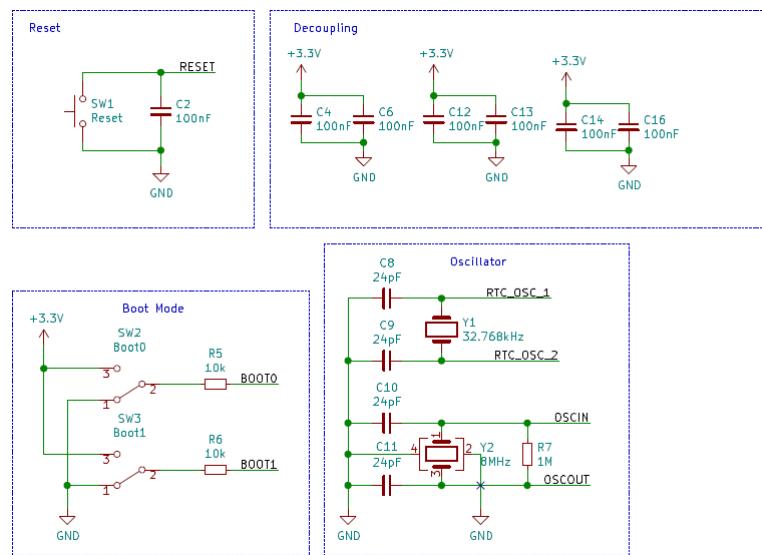


Abbildung 10: Unterstützende Bauelemente für den Mikrocontroller

### 3.4.1.2 Lokale Spannungsversorgung

Der Mikrocontroller und die restliche digitale Elektronik auf der Platine benötigen eine stabile Spannungsversorgung. Diese wird aus der Solarspannung gebildet. Die Solarspannung fluktuiert stark und ist außerdem mit bis zu 36V deutlich zu hoch. Daher wird ein synchroner Tiefsetzsteller (Buck-Converter) 11 verwendet, um die Spannung zu verringern und zu stabilisieren. Der Vorteil eines synchronen Buck-Converters gegenüber einem nicht synchronen ist der gesteigerte Wirkungsgrad. Dieser ist höher, da die Verluste, die durch den Spannungsfall der Diode eines regulären Tiefsetzstellers im Falle eines synchronen Tiefsetzstellers nicht auftreten. In diesem ist anstelle einer Diode ein FET verbaut, der in richtigen Moment durch die interne Elektronik eingeschaltet wird. Sowohl am Eingang als auch am Ausgang des Buck-Converters sind Elektrolytkondensatoren vorgesehen. Diese stabilisieren lokal die Spannung und sind wichtig für eine stabile Operation der Spannungsregelung. Die 5V-Spannung wird benötigt für den CAN-Bus.

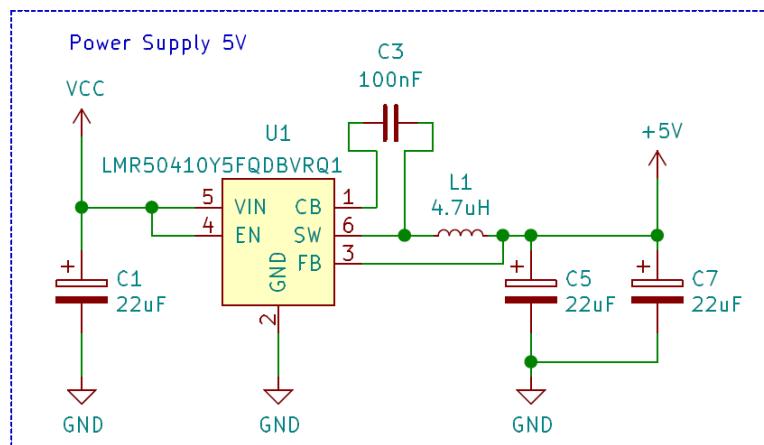


Abbildung 11: 5V Tiefsetzsteller

Der Mikrocontroller und zahlreiche andere ICs arbeiten mit einer Spannungsversorgung von 3.3V. Diese wird vom Linearregler U2 (Siehe Abb. 12) zur Verfügung gestellt. Es wurde ein Linearregler eingesetzt, da der gesamte Laststrom der digitalen Schaltkreise nicht über 100mA beträgt. Somit ist die Verlustleistung, die im Linearregler entsteht vernachlässigbar (Siehe Gl. 1). Daher war es nicht nötig, einen teureren Tiefsetzsteller einzusetzen. Besonders wichtig hierbei ist, dass das Spannungs differential zwischen Ein- und Ausgang des Reglers klein ist, um die Verlustleistung zu minimieren. Daher wird die 5V-Versorgung verwendet und nicht die Solarspannung.

Damit der Linearregler stabil arbeitet, sind zwei Keramikkondensatoren vorgesehen.

Berechnung der Verlustleistung des Linearreglers:

$$P_V = U_{Diff} * I_{Last} = (5V - 3,3V) * 0,1A = 0,17W \quad (1)$$

Die Versorgungsspannung des ADC muss mit Hilfe des Filternetzwerks (Abb. 12) gesiebt werden, um Störungen zu unterdrücken. Diese würden sich negativ auf die Genauigkeit der ADC-Messungen auswirken. Die Abb. 13 zeigt die Eigenschaften des Filternetzwerks im Frequenzbereich von 1 Hz bis 10 MHz. Bei der Dimensionierung wurde darauf geachtet, dass die eventuellen Störungen durch das Schalten des Hochsetzstellers, mit dem die Batterie geladen wird, herausgefiltert werden. Der Hochsetzsteller schaltet mit einer Frequenz von 100kHz, bei dieser beträgt die Dämpfung des Filters 9dB. Die Vielfachen, die durch das rechteckförmige Schalten auftreten, werden noch effektiver gefiltert.

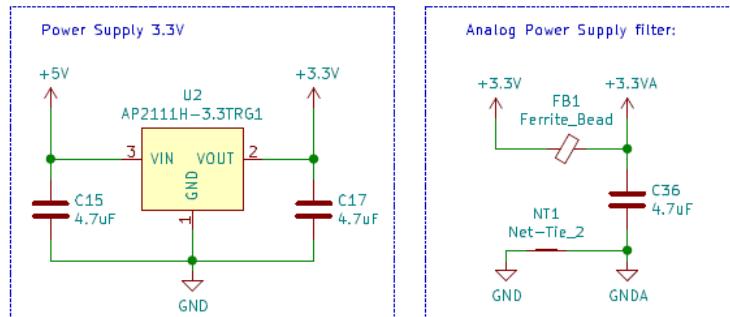


Abbildung 12: 3.3V Linearregler und Filter

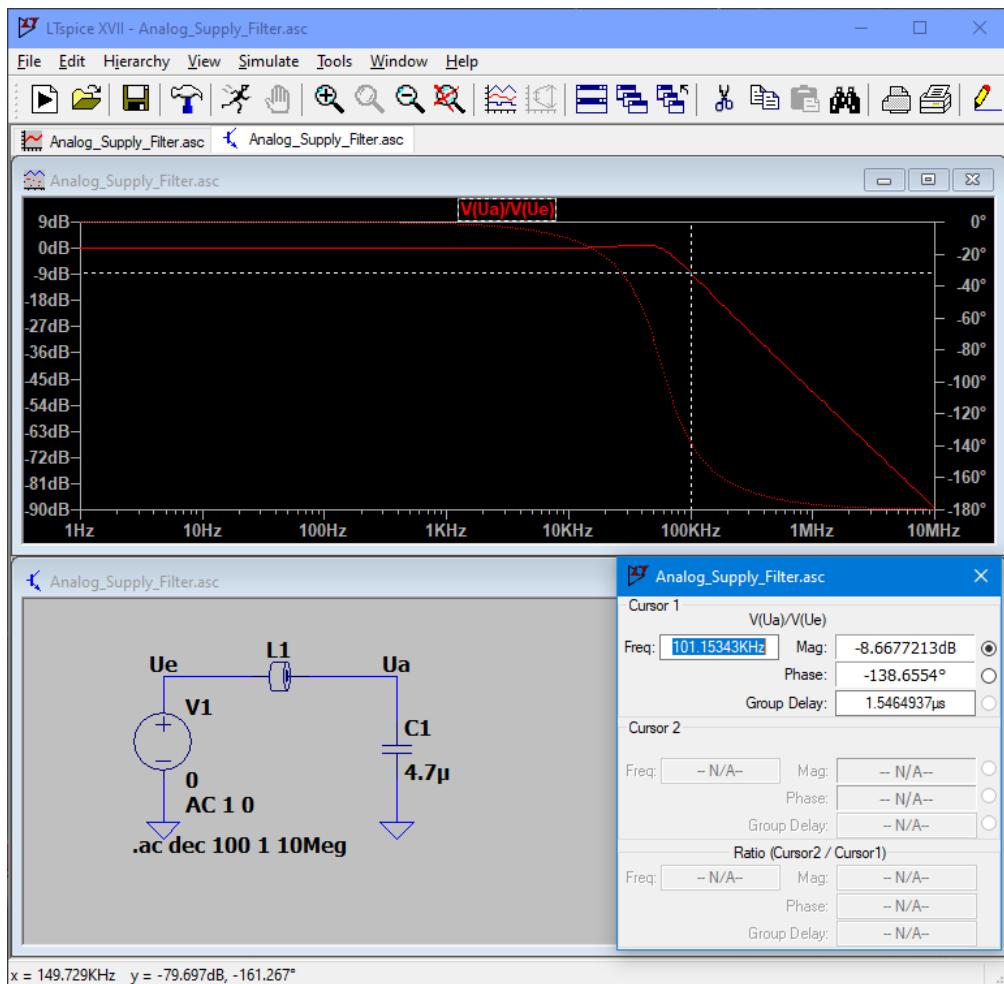


Abbildung 13: Simulation: Verhalten des Filters

### 3.4.1.3 Debugging-/ Programmierinterfaces und Testpunkte

Um den Mikrocontroller programmieren zu können, wurde ein Steckverbinder für das SWD-Interface vorgesehen. Über diesen lässt sich der STM32 mit Hilfe eines ST-Link-Programmers flashen. Die Diode D3 wurde eingesetzt, um zu verhindern, dass die Betriebsspannung des MPPT-Boards in den ST-Link-Programmer eingespeist wird. Allerdings ist es in manchen Fällen nützlich, den Mikrocontroller auch programmieren zu können, wenn nicht das gesamte Board an eine Spannungsversorgung angeschlossen ist. In diesem Szenario leitet die Diode D3 und der Mikrocontroller ist mit 3.3V versorgt.

Zusätzlich zum Programmier-Interface ist außerdem ein UART-Stecker vorhanden. Über diesen können während der Software-Debugging-Phase Nachrichten ausgegeben werden, um den Quellcode zu debuggen.

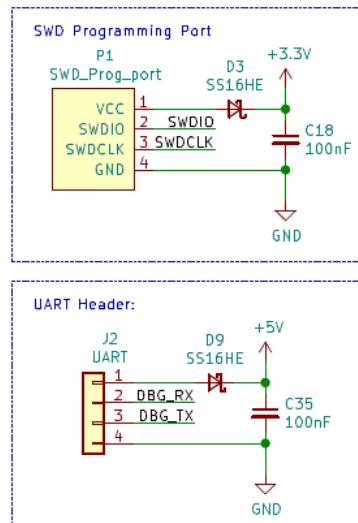


Abbildung 14: SWD-Programmierananschluss und UART

Um dem Nutzer generelle Informationen zum Status des MPPT zu liefern, sind fünf LEDs verbaut (Siehe Abb. 15 links). Zwei dieser LEDs (D4 und D11) leuchten, wenn die betreffende Betriebsspannung vorhanden ist. Die anderen drei LEDs (D5, D6 und D10) können über GPIO-Pins des Mikrocontrollers geschaltet werden.

Über die Taster SW4 und SW5 (Siehe Abb. 15 rechts) können Eingaben getätigt werden. Die Taster sind an GPIO-Pins des Mikrocontrollers angeschlossen. Sie werden über Pull-Up-Widerstände auf die Betriebsspannung gezogen. Wenn ein Taster betätigt wird, so liegt das Massepotential am Eingang an. Das Entprellen der Taster muss in Software geschehen, hierfür wurde keine zusätzliche Hardware auf der Platine vorgesehen.

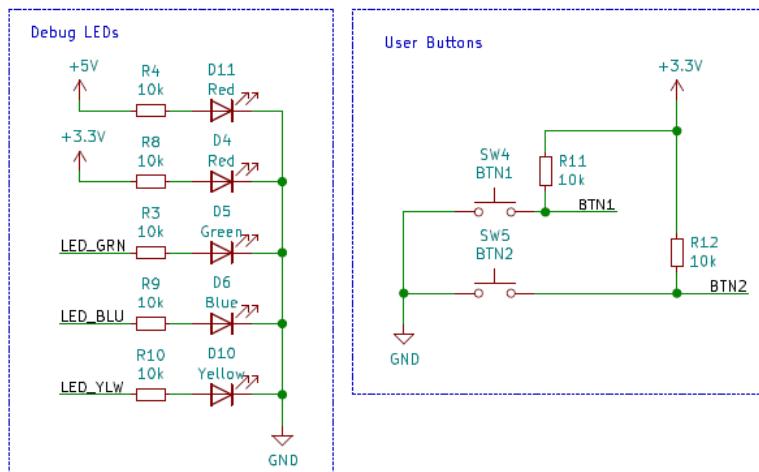


Abbildung 15: LEDs und Taster

Zu Testzwecken sind einige der relevanten Signale und Spannungen auf Testpads verfügbar gemacht (Siehe Abb. 16).

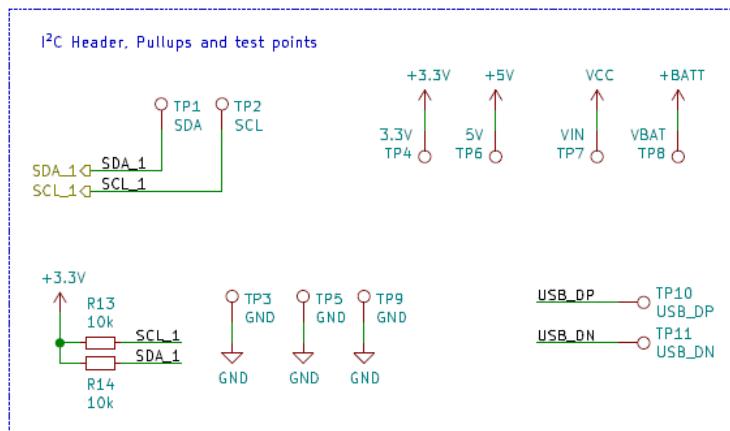


Abbildung 16: Testpunkte und Pull-Up Widerstände

### 3.4.1.4 EEPROM

Auf der Platine ist ein kleiner EEPROM verbaut. In diesem kann die Konfiguration des jeweiligen MPPT gespeichert werden und beim Start des Moduls geladen werden.<sup>2</sup>

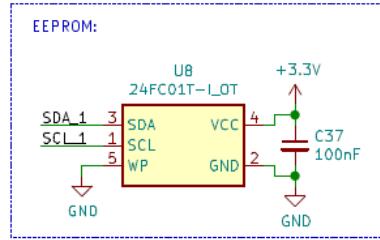


Abbildung 17: externer EEPROM-IC

### 3.4.1.5 CAN Bus Transceiver

Mit Hilfe des CAN-Bus-Transceivers ist eine Kommunikation mit der MPPT-Platine über CAN möglich. Im Solarboot existiert eine zentrale Datenlogger-Platine, die Daten von den verschiedenen elektronischen Steuergeräten und Modulen sammelt. Über die Lötbrücke JP1 kann ein Terminierwiderstand dazugeschaltet werden, sofern dies erwünscht sein sollte.

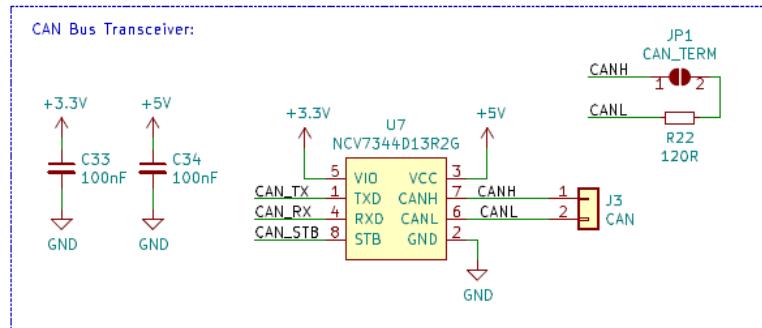


Abbildung 18: CAN Bus Transceiver

---

<sup>2</sup>Zum Zeitpunkt der Abgabe dieser Arbeit ist diese Funktion noch nicht in der Software implementiert worden

### 3.4.2 Hochsetzsteller

Der Hochsetzsteller (vlg. Abb. 19) ist zentrales Element der MPPT-Platine. Dieser besteht im Wesentlichen aus vier Arten von Bauteilen:

- Kondensatoren am Ein- und Ausgang
- Spule
- MOSFET
- Diode

Da die korrekte Funktion des Hochsetzstellers (engl. Boost-Converter) elementar für die Funktion des MPPT ist, wurde die Auswahl der einzelnen Komponenten mit besonderer Sorgfalt durchgeführt. Bei der Dimensionierung wurde der Anwendungsbericht *Basic Calculation of a Boost Converter's Power Stage*[5] der Firma Texas Instruments als Grundlage verwendet.

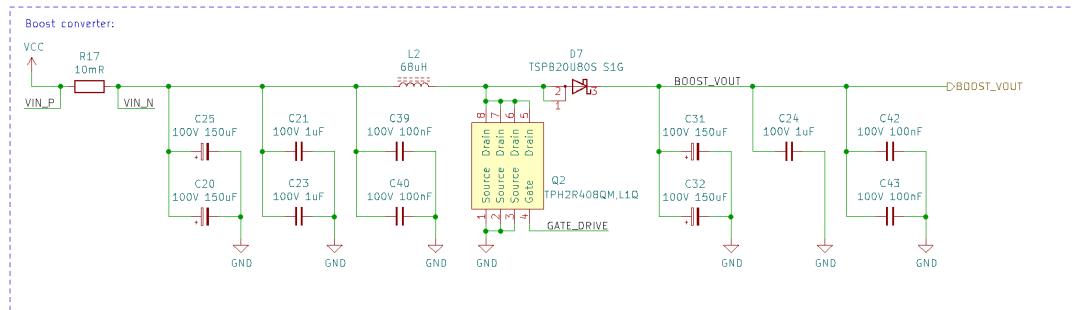


Abbildung 19: Schaltplan Hochsetzsteller

### 3.4.2.1 Auswahlentscheidungen Hochsetzsteller

Im folgenden sind die Auswahlentscheidungen der Komponenten des Boost-Converters genau erklärt.

#### Kondensatoren am Ein- und Ausgang:

Grundsätzlich ist zu bemerken, dass sowohl an Ein- als auch am Ausgang der Leistungsstufe jeweils zwei Arten von Kondensatoren zum Einsatz kommen. Es sind Elektrolytkondensatoren und Keramikkondensatoren parallel verbaut. Beide Arten bieten ihre eigenen Vor- und Nachteile, in Kombination können die Vorteile der beiden Technologien miteinander kombiniert werden. Elektrolytkondensatoren bieten eine große Kapazität, haben aber einen höheren Innenwiderstand. Somit sind sie grundsätzlich gut geeignet, um niederfrequente Spannungen zu sieben bzw. zu stabilisieren. Im Gegensatz dazu haben Keramikkondensatoren erheblich geringere Kapazitäten. Durch den geringeren Innenwiderstand und die kleinere Bauform, die auch zu einer Reduktion der parasitären Induktivität der Anschlüsse führt, können diese kurzzeitig deutlich höherer Ströme bereitstellen. Die dem Schaltplan zu entnehmen ist, wurden zwei Größen von Keramikkondensatoren verwendet, um den Effekt noch stärker auszunutzen.

Im Anwendungsgebiet der Leistungselektronik, bei der, wie im Fall dieses Hochsetzstellers, hochfrequent große Ströme geschaltet werden, bildet die Kombination beider Kondensatortechnologien das Optimum.

Sowohl am Ein- als auch am Ausgang des Hochsetzstellers werden die gleichen Kondensatoren eingesetzt. Zwar ist die Eingangsspannung nie so hoch, wie die Ausgangsspannung werden könnte, so ist der Unterschied zwischen den beiden Spannungen im Regelfall aber auch nie größer als 20V. Es wurden die selben Werte gewählt, um die Zahl der verschiedenen BOM-Einträge so niedrig wie möglich zu halten. Es wurden die Kondensatoren vom Hersteller Nippon Chemi-Con mit der Teilenummer *EM-HS101ARA151MKG5S* ausgewählt. Diese zeichnen sich durch eine kleine Bauform, einen geringen ESR und eine lange Lebensdauer aus. Der äquivalente Serienwiderstand bei 20 °C beträgt nur  $130m\Omega$  [2]. Die Kondensator-Nennspannung wurde mit 100V bewusst etwas über der maximalen Ausgangsspannung des MPPT gewählt, da in der Vergangenheit die Software durch Bugs Probleme bei der Begrenzung der Ausgangsspannung bei sehr geringer Last hatte. Dadurch, dass die Nennspannung der Kondensatoren nun deutlich höher gewählt wurde, gibt es im Falle solcher Probleme eine zusätzliche Sicherheitsbarriere, bevor es zu bleibenden Schäden kommt.

## Spule:

Die wichtigsten Kriterien bei der Auswahl der Spule waren:

- Induktivität
- parasitärer Widerstand
- Sättigungsstrom
- SMD-Bauform / Größe
- Preis

Die Induktivität der Spule beeinflusst unmittelbar, wie hoch die Ströme sein werden, die bei gleicher Last durch den MOSFET geschaltet werden müssen. Auch die Frequenz, mit der gearbeitet wird, ist ein wichtiger Parameter. Je höher diese gewählt wird (im Rahmen der realistischen Grenzen), desto geringer wird der zu schaltende Strom.

Die max. Schaltfrequenz ist durch die Hardware, die im Mikrocontroller vorhanden ist, begrenzt. Durch eine Abwägung zwischen möglichst hoher Frequenz und gleichzeitig noch möglichst hoher PWM-Auflösung, wurde die Schaltfrequenz auf 100kHz festgelegt.

Der Anwendungsbericht von TI [5] bietet viele Formeln, mit deren Hilfe die Parameter berechnet werden können. Die Zahl der Unbekannten ist vor allem zu Beginn der Parametrisierung sehr groß. Daher müssen einige der Bauteilwerte festgelegt werden, um eine Berechnungsgrundlage zu haben.

Es wurde daher entschieden, dies im Fall der Induktivität zu tun. Nach ausgiebigen Recherchen und Vergleichen, wurde die Spule mit der Teilenummer 74437429203680 vom Hersteller Würth Elektronik ausgewählt. Die Spule besitzt eine Induktivität von  $68\mu H$ . Der ausschlaggebendste Grund für die Wahl der Spule war der geringe parasitäre Widerstand und der hohe Sättigungsstrom des Kernmaterials.

## Berechnungen der beteiligten Größen:

Um die maximalen Schaltströme zu berechnen, wurden die Formeln aus [Ti] verwendet.

Die Pulsweite, mit der die Spule geschaltet wird wurde folgendermaßen berechnet<sup>3</sup>:

$$D = 1 - \frac{V_{IN(min)} * n}{V_{OUT}} = 1 - \frac{26V * 0,9}{50,4V} = 0,54 \quad (2)$$

Im Anschluss konnte Rippelstrom durch die Spule berechnet werden:

$$\Delta I_L = \frac{V_{IN(min)} * D}{f_s * L} = \frac{26V * 0,54}{100kHz * 68\mu H} = 2,04A \quad (3)$$

Mit diesen Werten konnte nun der maximale Schaltstrom berechnet werden:

$$I_{SW(max)} = \frac{\Delta I_L}{2} + \frac{I_{OUT(max)}}{1 - D} = \frac{2,04A}{2} + \frac{3A}{1 - 0,54} = 7,54A \quad (4)$$

---

<sup>3</sup>Hierbei wurde ein Wirkungsgrad von 90% angenommen

## MOSFET:

Anforderung an den MOSFET ist, dass mindestens der in Gl. 4 berechnete Strom geschaltet werden muss. Außerdem ist die On-Resistance sehr wichtig, damit die Schaltverluste so niedrig wie möglich sind. Ebenfalls wichtig in Bezug auf die Schaltverluste ist eine möglichst geringe Gate-Kapazität, da diese bei jedem Schaltvorgang ge- bzw. entladen wird. Je länger dies dauert, desto länger befindet sich der FET in der linearen, dh. sehr verlustbehafteten Region. Bei der Auswahl muss auch darauf geachtet werden, dass die max. zulässige Drain-to-Source-Spannung unter der maximalen Ausgangsspannung liegt. Andernfalls würde der FET zerstört werden, wenn die gewollte Ausgangsspannung erreicht wird.

Unter Berücksichtigung der genannten Anforderungen wurde der FET mit der Teilenummer *TPH2R408QM,L1Q* vom Hersteller Toshiba gewählt. Es handelt sich hierbei um einen MOSFET mit sehr niedrigem Widerstand im eingeschaltetem Zustand. Der max. Drain-Strom mit 120A ist deutlich höher als die Mindestanforderung nach Gl. 4.

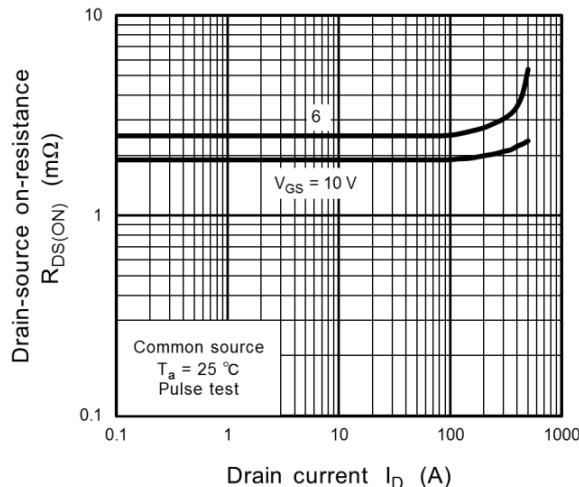
Berechnung der Verlustleistung des MOSFETs bei maximaler Ausgangslast<sup>4</sup>:

$$P_V = R_{DS(ON)} * I_{Last}^2 = 1,9m\Omega * (3A)^2 = 17mW \quad (5)$$

Wie die Rechnung zeigt, sind die angenäherten Verluste durch den MOSFET sehr gering.

Der Wert für  $R_{DS(ON)} = 1,9m\Omega$  stammt aus dem Datenblatt [3] des FETs (vgl. Abb. 20).

Da das Gate in dieser Anwendung sogar mit 18V geladen wird, im Datenblatt hierzu aber keine Angaben existieren, wurden die Daten für 10V gewählt.



**Fig. 8.5  $R_{DS(ON)}$  -  $I_D$**

Abbildung 20:  $R_{DS(ON)}$  des MOSFETs [3]

---

<sup>4</sup>Unter Vernachlässigung der Verluste während des Schaltvorgangs

## Diode:

Ein weiteres zentrales Bauelement eines Hochsetzstellers ist die Diode. Die Suche nach einem passenden Bauelement wurde von Beginn an auf Schottky-Dioden beschränkt. Dieser Diodentyp schaltet besonders schnell und hat eine niedrigere Schwellspannung als herkömmliche Siliziumdioden. Für das Einsatzgebiet ist eine schnelle Schaltzeit der Diode von Vorteil, da der Hochsetzsteller mit hoher Frequenz geschaltet wird.

Bei der Suche nach einer Diode musste darauf geachtet werden, dass die Schwellspannung (unter Last) möglichst niedrig ist und dass die Sperrspannung ausreichend hoch ist.

Nach ausgiebigen Vergleichen mit vielen anderen Bauteilen, wurde eine Schottkydiode vom Hersteller Taiwan Semiconductor mit der Teilenummer *TSPB20U80S S1G* ausgewählt.

$$P_V = U_F * I_{Last} = 0,4V * 3A = 1,2W \quad (6)$$

Die in der Diode auftretenden Wärmeverluste sind deutlich höher als die im MOSFET auftretenden Verluste.

Der Wert für  $U_F = 0,4V$  stammt aus dem Datenblatt [11] der Diode (vgl. Abb. 21). Hierbei wurde eine Betriebstemperatur von ca.  $40^{\circ}\text{C}$  und ein Strom von 3A angenommen.

**FIG. 2 TYPICAL FORWARD CHARACTERISTICS**

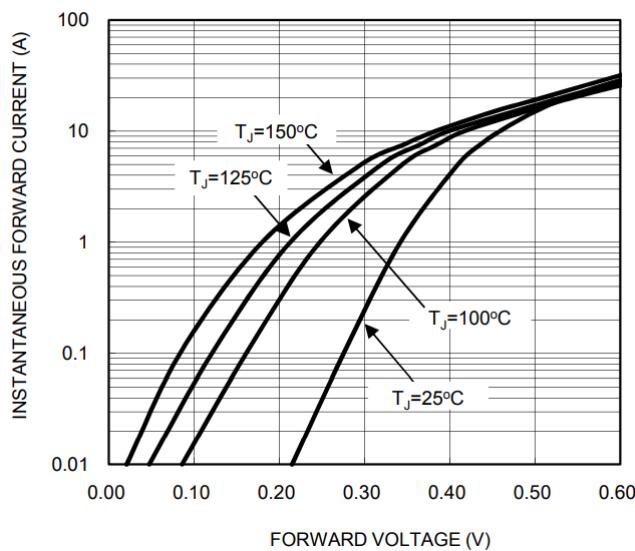


Abbildung 21:  $U_F$  der Diode in Abhängigkeit von Strom und Temperatur [11]

$$a \quad (7)$$

### 3.4.2.2 Gate-Treiber für den Hochsetzsteller

Um die Schaltverluste, die in dem MOSFET des Hochsetzstellers auftreten, so gering wie möglich zu halten, muss das Gate des FETs möglichst schnell geladen und entladen werden. Der MOSFET wird über einen GPIO-Pin des Mikrocontrollers angesteuert. Wenn die Ansteuerung direkt erfolgen würde, so wäre die Einschaltzeit des FETs lang. Dies verdeutlichen die folgenden Rechnungen.

Ein GPIO-Pin des STM32 kann laut Datenblatt [14], S. 65 bis zu 20mA liefern. Die Versorgungsspannung beträgt 3,3V. Aus diesen Angaben wird die Größenordnung der Ausgangsimpedanz eines einzelnen GPIO-Pins angenähert:

$$R_{GPIO} \cong \frac{U_{GPIO}}{I_{GPIO(MAX)}} = \frac{3.3V}{20mA} = 165\Omega \quad (8)$$

Die Gate-Kapazität des MOSFETs beträgt laut Datenblatt 6nF. Der Gate-Eingangswiderstand beträgt 1,9Ω [3]. Als Annäherung für die Zeit, die für das Laden des Gate vergeht wurde die Zeitkonstante  $\tau$  berechnet.

$$\tau = (R_{GPIO} + R_{Gate}) * C_{Gate} = (165\Omega + 1,9\Omega) * 6nF = 1\mu s \quad (9)$$

Nach ca.  $5 * \tau$  ist die Gatekapazität vollständig geladen. Dies entspricht einer Zeit von  $5\mu s$ . Die Schaltfrequenz des Hochsetzstellers beträgt 100kHz, daraus folgt eine Periodendauer von  $\frac{1}{100kHz} = 10\mu s$ . Daraus folgt, dass der Entladeprozess des Gate beginnt, sobald es annähernd voll geladen wurde. Somit ist es nicht möglich, das Gate nur unter Verwendung des GPIO-Pins ausreichend schnell zu laden und zu entladen.

Hinzu kommt außerdem die niedrige Spannung, die ein GPIO-Pin liefert. Die Gate-Spannung sollte im Fall des eingesetzten FET mindestens 10V betragen, damit der Drain-to-Source-Widerstand möglichst niedrig ist [3].

Aus diesen Gründen wurde ein Gate-Treiber IC benötigt. Dieser ist speziell für den Anwendungsfall entwickelt und ermöglicht es, beide zuvor identifizierten Probleme beim Schalten des Gate zu beheben.

Die maximale Gate-to-Source-Spannung des eingesetzten MOSFETs beträgt 20V [3]. Die Eingangsspannung des MPPT kann aber höher sein als 20V, daher war es nötig, diese Spannung zu begrenzen. Hierfür wurde der Linearregler U5 (Siehe Abb. 22 l.u.) vorgesehen. Dieser regelt die Spannung herunter auf 18V. Es wurde ein Linearregler eingesetzt, da der Laststrom nicht groß ist. Dieser setzt sich maßgeblich nur aus dem Strom zusammen, der für das Laden und Entladen des Gates nötig ist.

Es wurde der Gate-Treiber mit der Teilenummer *UCC27531DBVR* vom Hersteller Texas Instruments ausgewählt. Dieser IC erfüllt die Anforderung, dass das Gate so schnell wie möglich geschaltet werden soll. Der Treiber kann bis zu 2,5A bzw. 5A beim Laden und Entladen zur Verfügung stellen [7]. Der Widerstand R23 (Siehe Abb. 22) am Eingang wurde vorgesehen, um Rise-Time des GPIO-Pins zu limitieren und den GPIO nicht zu überlasten. R21 kann mit einem anderen Wert als  $0\Omega$  bestückt werden, um die Laderate des Gate zu limitieren.

Außerdem sind noch mehrere Abblockkondensatoren vorgesehen. Dabei handelt es sich um Keramikkondensatoren (sog. MLCC), da der Gate-Treiber im Schaltmoment sehr hohe Ströme erfordert. Hierfür sind MLCCs besonders geeignet.

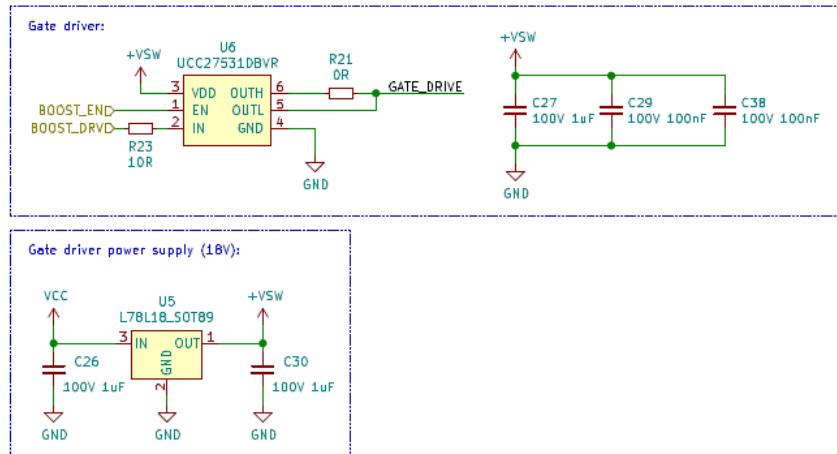


Abbildung 22: Gate-Treiber für den Hochsetzsteller

### 3.4.2.3 INA226 Leistungssensor

Für die Operation des MPPT ist es sehr wichtig, ständig genaue Messungen der Eingangsleistung zu haben. Aus diesem Grund wurde entschieden, einen dezidierten IC zu verwenden, der diese Aufgabe übernimmt. Der Leistungssensor misst den Spannungsfall über einen Shunt-Widerstand (Siehe R17 Abb. 19). Außerdem misst der Leistungssensor-IC auch die Eingangsspannung (VCC). Aus den beiden Messwerten (I und U) wird durch den IC die Leistung, die durch den Shunt übertragen wird, berechnet.

Bei der Auswahl eines geeigneten ICs gab es keine große Auswahl, da die Anforderungen an die maximal messbare Spannung, die unterstützt wird, mit  $\geq 36V$  relativ hoch war. Außerdem war erwünscht, dass der IC bereits viele der Messungen eigenständig ausführen kann und diese Resultate per Bussystem zu einem Host-Mikrocontroller übertragen kann. Somit entfiel die Notwendigkeit komplexe und ggf. fehleranfällige analoge Schaltungen auf der MPPT-Platine aufzubauen. Infolgedessen wurde der IC mit der Teilenummer *INA226AIDGSR* vom Hersteller Texas Instruments ausgewählt. Dieser Chip erfüllt alle genannten Anforderungen.

Im Fall des MPPTs wird eine High-Side-Messung verwendet. High-Side bedeutet, dass der Shunt-Widerstand aus Sicht des Verbrauchers mit der Versorgungsspannung in Serie geschaltet ist. Im Gegensatz hierzu, wäre der Messwiderstand bei einer Low-Side-Messung auf der Masseseite in Serie verschaltet. Eine High-Side-Messung ist in der Regel schaltungstechnisch komplexer durchzuführen [1], der ausgewählte Chip unterstützt jedoch beide Verfahren. Laut [1] sind einige der Vorteile einer High-Side-Messung:

- keine Probleme mit Masse-Schleifen
- Messtechnische Erkennung von Masseschlüssen möglich

Daher wurde entschieden, den Strom mittels High-Side-Messung zu erfassen.

Die Schaltung auf Abb. 23 zeigt den Leistungssensor. Die Eingänge vom Messwiderstand sind gefiltert. Bei der Dimensionierung der Bauteile wurde die Vorschläge aus dem Datenblatt [6], Abschnitt 7.4.2 verwendet.

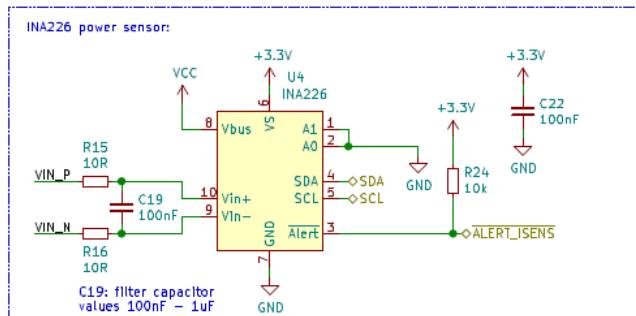


Abbildung 23: INA226 Leistungssensor

### 3.4.2.4 Ausgangsspannungsmessnetzwerk

Die Ausgangsspannung des Hochsetzstellers muss gemessen werden. Mit dieser Spannung wird die LiIon-Batterie geladen. Der Regelalgorithmus benötigt stets die aktuelle Ausgangsspannung zur Berechnung der einzustellenden Pulswerte.

Die maximale Eingangsspannung des Mikrocontrollers an den analogen Eingängen beträgt 3.3V, die Akkuspannung kann aber bis zu 50,4V betragen. Daher wird ein resistiver Spannungsteiler eingesetzt, um die Ausgangsspannung proportional zu verringern.

Die Werte für den Spannungsteiler wurden so ermittelt, dass eine geringe Überspannung nicht direkt zu einer zu hohen Spannung am analogen Eingang führt:

$$V_{IN(MAX)} = V_{ADC(MAX)} * \frac{R18 + R19 + R20}{R20} = 3,3V * \frac{21k\Omega}{1k\Omega} = 69,3V \quad (10)$$

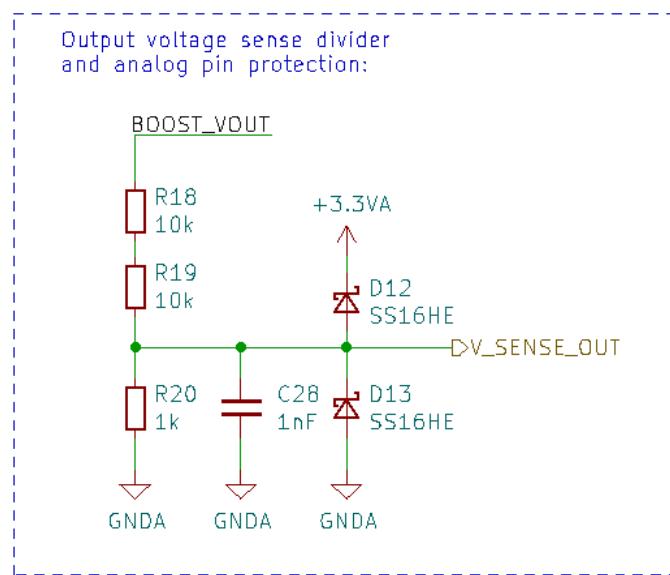
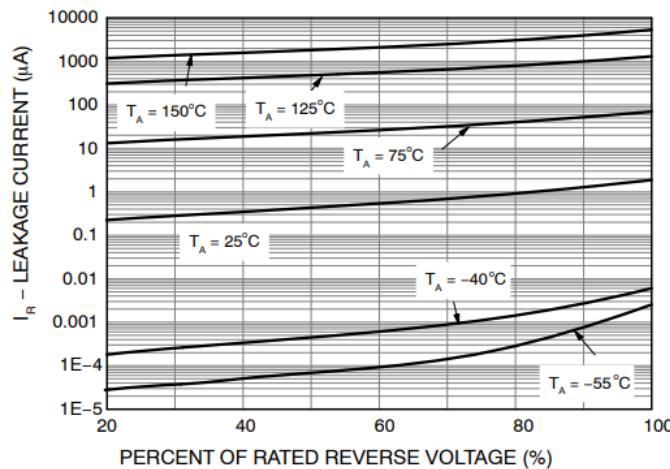


Abbildung 24: Spannungsteiler zur Messung der Ausgangsspannung

Der Kondensator C28 (Siehe Abb. 24) dient einer Filterung, um das Messsignal etwas zu glätten.

Die Schottkydioden D12 und D13 bieten einen zusätzlichen Schutz für den Fall, dass die Spannung an V\_SENSE\_OUT im Fehlerfall höher sein sollte als 3.3V oder niedriger als 0V. Falls die Eingangsspannung des Teilers zu hoch ist, so leitet die Diode D12 und die Spannung am Eingang des ADC wird begrenzt auf  $3.3V + U_{F(D12)}$ . Sofern die Spannung unter 0V sinken sollte, dann leitet die Diode D13 und begrenzt die Spannung auf  $-U_{F(D13)}$ .

Damit die ADC-Messung nicht verfälscht wird, ist es sehr wichtig, dass die Dioden einen möglichst niedrigen Leckstrom besitzen. Wenn dieser Strom zu groß wäre, so würde die Ausgangsspannung des Spannungsteilers infolge der hohen Widerstände der Teilelemente stark beeinflusst werden. Bereits ein kleiner Strom führt bei den großen Widerständen zu einem erheblichen Fehler. Daher ist eine 3.3V Zenerdiode parallel zu R20 nicht geeignet, da diese Dioden typischerweise deutlich höhere Leckströme aufweisen (Siehe Abb. 26). Wenn die Spannung am ADC-Eingang 3.3V betragen würde, dann würden bereits 5mA durch die Zenerdiode fließen, was den Spannungsteiler stark belastet und das Messergebnis verfälscht. Der Leckstrom der Schottkydiode in diesem Anwendungsfall beträgt nur ca. 200nA<sup>5</sup> (Siehe Abb. 25).



**Figure 6. Typical Reverse Characteristics – SS16HE**

Abbildung 25: Leckstrom der Schottkydioden D12 und D13 [10]

ELECTRICAL CHARACTERISTICS ( $T_A = 25^\circ\text{C}$  unless otherwise noted,  $V_F = 0.9 \text{ V Max.} @ I_F = 10 \text{ mA for all types}$ )

Device*	Device Marking	Zener Voltage (Note 2)			Zener Impedance			Leakage Current			$\theta V_Z$ (mV/k) @ $I_{ZT}$		C @ $V_R = 0$ $f = 1 \text{ MHz}$
		V <sub>Z</sub> (Volts)			@ $I_{ZT}$			$I_R @ V_R$					
		Min	Nom	Max	mA	$\Omega$	$\Omega$	mA	$\mu\text{A}$	Volts	Min	Max	pF
MM3Z2V4T1G	00	2.2	2.4	2.6	5	100	1000	0.5	50	1.0	-3.5	0	450
MM3Z2V7T1G	01	2.5	2.7	2.9	5	100	1000	0.5	20	1.0	-3.5	0	450
MM3Z3V0T1G	02	2.8	3.0	3.2	5	100	1000	0.5	10	1.0	-3.5	0	450
MM3Z3V3T1G	05	3.1	3.3	3.5	5	95	1000	0.5	5	1.0	-3.5	0	450
MM3Z3V6T1G	06	3.4	3.6	3.8	5	90	1000	0.5	5	1.0	-3.5	0	450

Abbildung 26: Leckstrom der Zenerdiode MM3Z3V3T1G [8]

<sup>5</sup> Annahme: 20 % der Rückwärtsspannung bei  $25^\circ\text{C}$

### 3.4.3 Ein- und Ausgangsschutzbeschaltung

Der Eingang des MPPT ist mit verschiedenen Maßnahmen gegen Beschädigung durch Überstrom, ESD und Verpolung geschützt. Die Abbildung 27 zeigt die Schutzmaßnahmen des Eingangs. Die Schmelzsicherung F1 schützt vor einem zu hohen Dauerstrom. Im Falle eines massiven Schadens an der Platine (z.B. Kurzschluss zwischen Masse und Versorgungsspannung) schützt die Sicherung die vorgeschalteten Bauelemente und Leitungen vor der zu hohen Belastung. D1 und RV1 schützen die Schaltung vor kurzen Spannungsspitzen, wie diese durch ESD auftreten können. Die Bauelemente sind elektrisch nach der Sicherung verschaltet, so dass die Sicherung auslösen sollte, falls eines der Bauteile nach einem ESD-Event einen Kurzschluss darstellen sollte.

Q3, D2 und R1 bilden den Verpolungsschutz. Anstelle einer einfachen Diode als Schutz wurde ein MOSFET verwendet, da dieser deutlich niedrigere Verluste aufweist als eine Diode. Bei Q3 handelt es sich um einen P-Kanal MOSFET. Dieser leitet, wenn die Gate zu Source-Spannung unter einen vom Bauteil abhängigen Schwellwert sinkt. Dieser Effekt wird bei der Schaltung ausgenutzt. Wenn die Versorgungsspannung in verkehrter Polarität angeschlossen wird, so liegt an Drain Masse der Quelle an und an dem Gate über den Widerstand R1 die positive Versorgungsspannung an. Durch die Diode D2 liegt nahezu die volle Versorgungsspannung an Source an. Die Spannungsdifferenz von Gate zu Source ist nicht negativ genug, um den P-Kanal MOSFET einzuschalten.

Wenn die Versorgungsspannung in korrekter Polarität angeschlossen wird, so leitet die interne Body-Diode des FET, auch wenn der FET im Zuschaltmoment der Versorgungsspannung noch aus ist. Somit liegt nahezu die volle Eingangsspannung an Source. Das Gate ist mit Masse verbunden. Somit ist die Gate zu Source-Spannung annähernd -(Versorgungsspannung). Dadurch schaltet der FET ein.

Die Zenerdiode D2 dient dazu, die Gate zu Source-Spannung auf ein zulässiges Maß zu begrenzen, für den Fall, dass die Eingangsspannung größer ist, als die maximal zulässige Gate zu Source-Spannung des MOSFETs. Die Zenerdiode ist in Sperrrichtung verbaut. Wenn die Gate zu Source-Spannung zu negativ wird, so wird die Zenerspannung der Diode erreicht und sie leitet. Der Widerstand R1 begrenzt den Strom und die Gate zu Source-Spannung wird begrenzt auf -(Zenerspannung).

Der Ausgang des MPPT (Siehe Abb. 28) ist ähnlich abgesichert, wie der Eingang. Es ist eine Sicherung und ESD-Schutz vorgesehen.

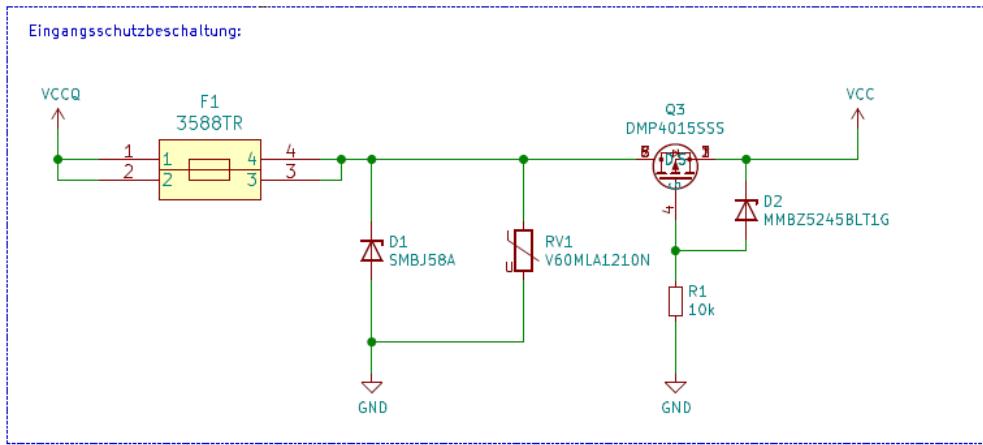


Abbildung 27: Eingangsschutzbeschaltung

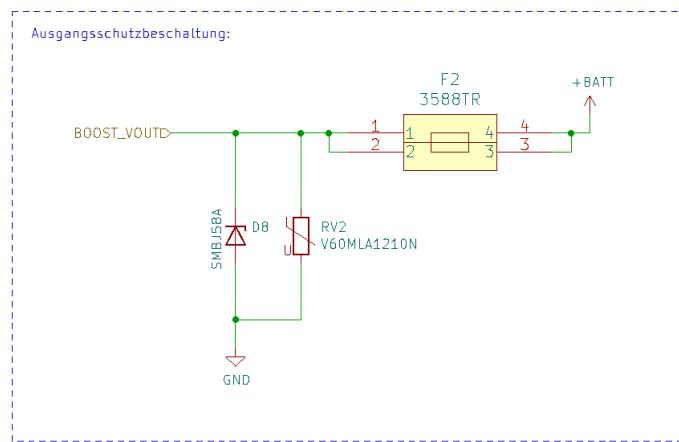


Abbildung 28: Ausgangsschutzbeschaltung

### 3.4.4 mechanische Elemente

Um die Platine später in ein Gehäuse einbauen zu können, wurden vier Montagelöcher vorgesehen (Siehe Abb. 29).

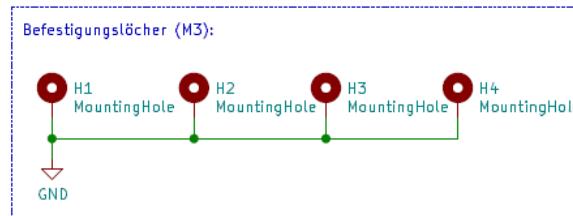


Abbildung 29: Montagelöcher

## 3.5 Platinenlayout

Die Abb. 35, 37 und 36 zeigen 3D-Render der Platine. Die Maße der Platine betragen: 80mm \* 65mm. Die Platine hat vier Kupferlagen, von denen die beiden äußersten primär für Signale verwendet wurden. Die beiden inneren Lagen dienen ausschließlich dafür, eine gute Masse-Referenz zu haben und sind dementsprechend nahezu komplett mit Masse gefüllt.

Beim Layout wurde versucht, alle Signale auf der obersten Lage zu routen. Falls dies nicht möglich war, so wurde über eine Durchkontaktierung (Via) auf die Platinenrückseite gewechselt. Um das Layout übersichtlicher und aufgeräumter zu halten, wurden Leiterbahnen auf der Vorderseite primär vertikal und auf der Rückseite primär horizontal geroutet. Dies war nicht in allen Fällen möglich oder sinnvoll, dann wurde von dieser Regel abgewichen.

### 3.5.1 Kupferlagen der Platine

Der Übersichtlichkeit halber ist die oberste Kupferlage doppelt dargestellt. Auf der Abb. 30 ist das Layout ohne den sog. Ground-Fill oder Ground-Pour dargestellt. Das bedeutet, dass die Massefläche nicht komplett eingefüllt ist. Somit sind die Details besser zu erkennen. Die Abb. 31 zeigt das Layout der ersten Lage mit Ground-Fill. Die Abb. 32 zeigt die Rückseite der Platine ohne einen Ground-Fill. Die Abb. 33 und 34 zeigen die beiden inneren Lagen mit Ground-Fill.

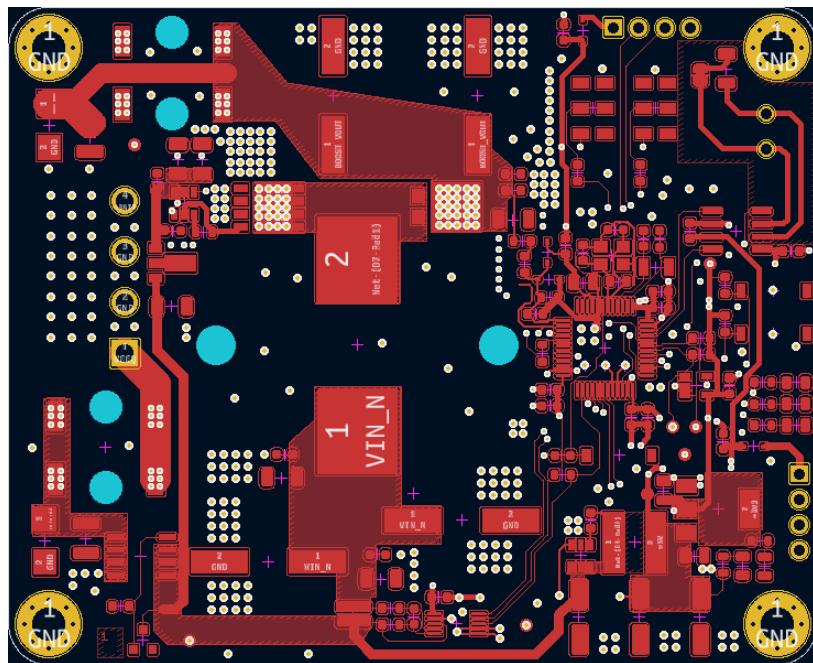


Abbildung 30: Oberste Kupferlage ohne Ground-Fill

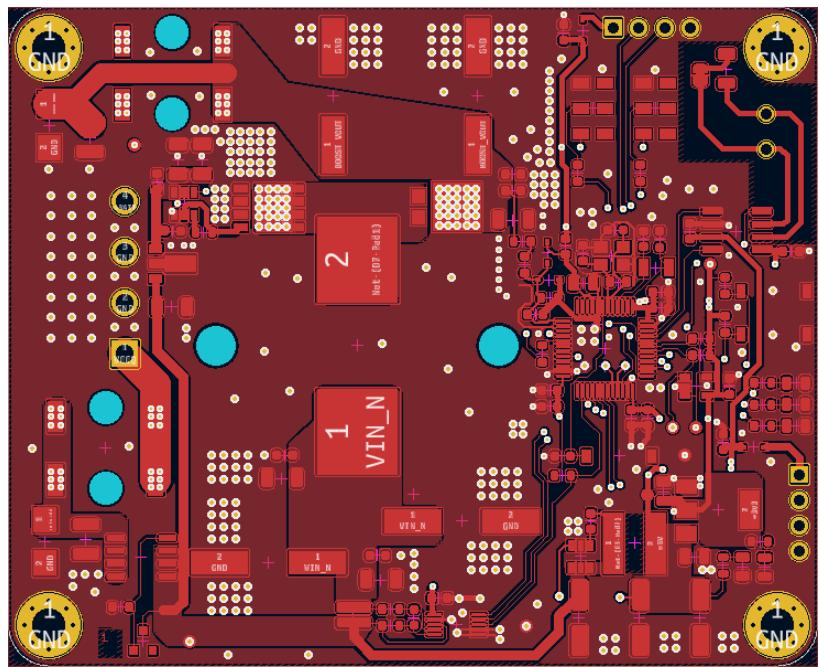


Abbildung 31: Oberste Kupferlage mit Ground-Fill

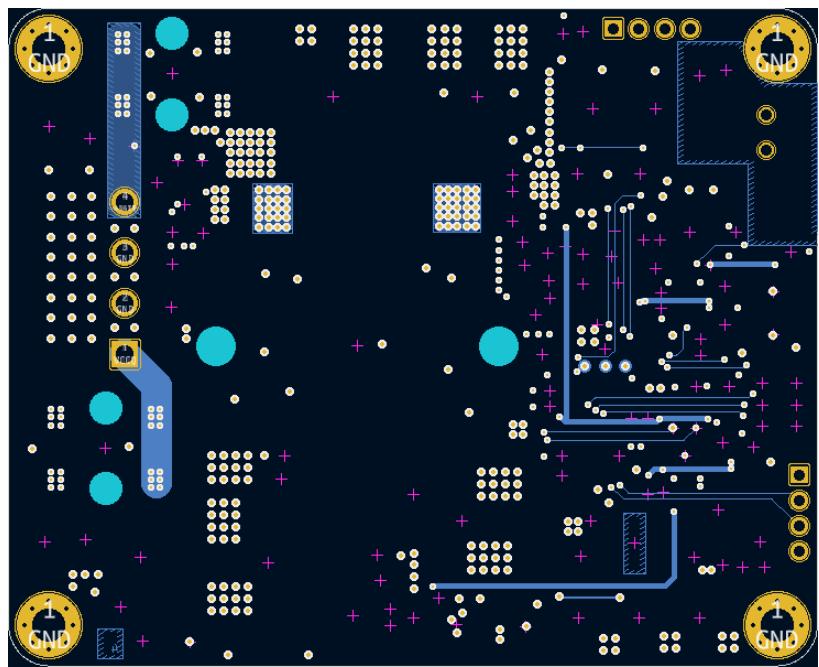


Abbildung 32: Unterste Kupferlage ohne Ground-Fill

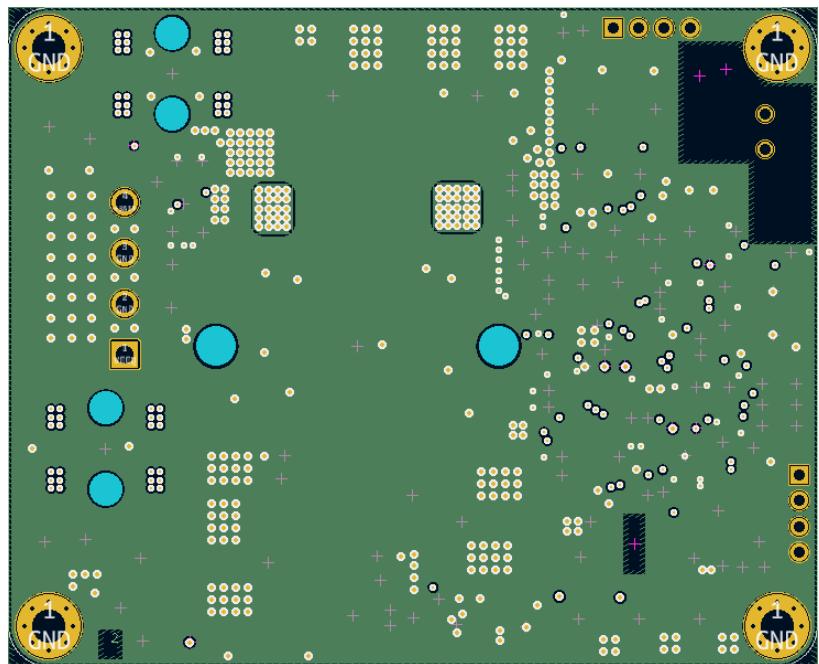


Abbildung 33: Interne Kupferlage 1 mit Ground-Fill

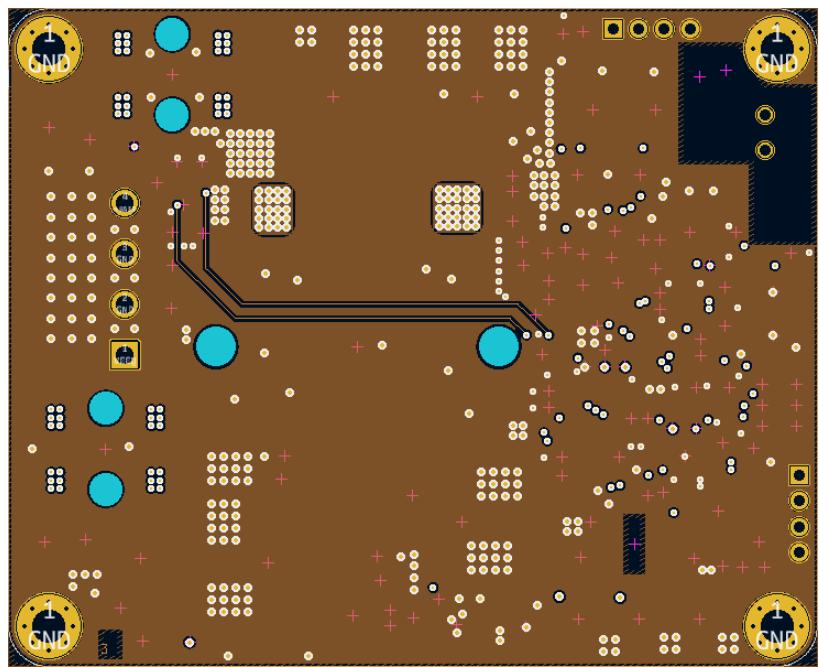


Abbildung 34: Interne Kupferlage 2 mit Ground-Fill

### 3.5.2 3D-Render der Platine

Die Abb. 35, 36 und 37 zeigen ein 3D-Render der Platine. Die Darstellung der Oberseite, auf der die Komponenten aufgelötet werden ist sowohl mit (Abb. 35) als auch ohne (Abb. 37) Bauteile dargestellt.

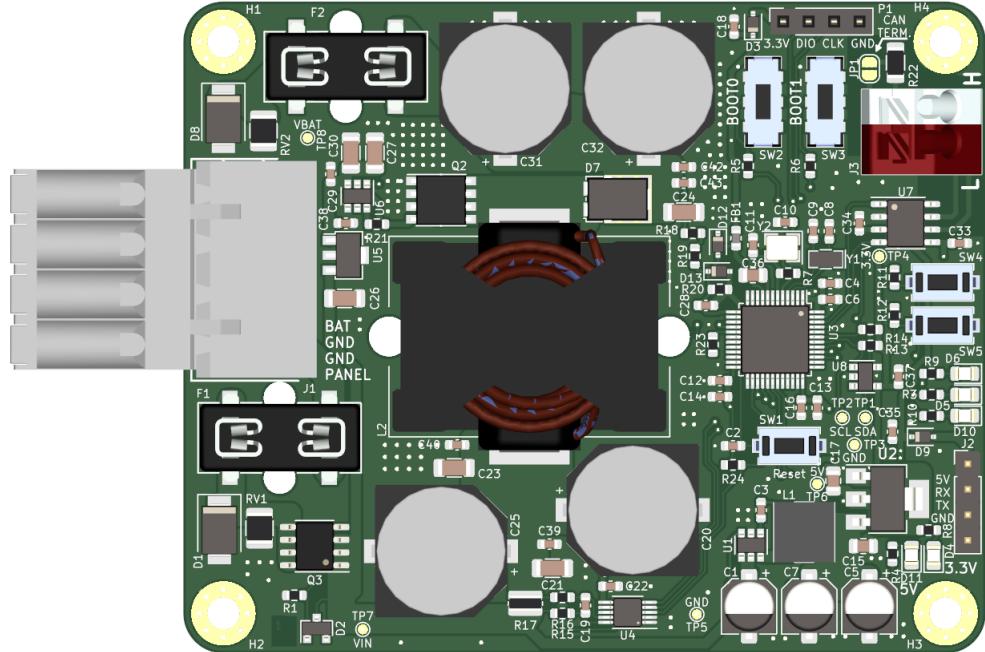


Abbildung 35: Oberseite der bestückten Platine

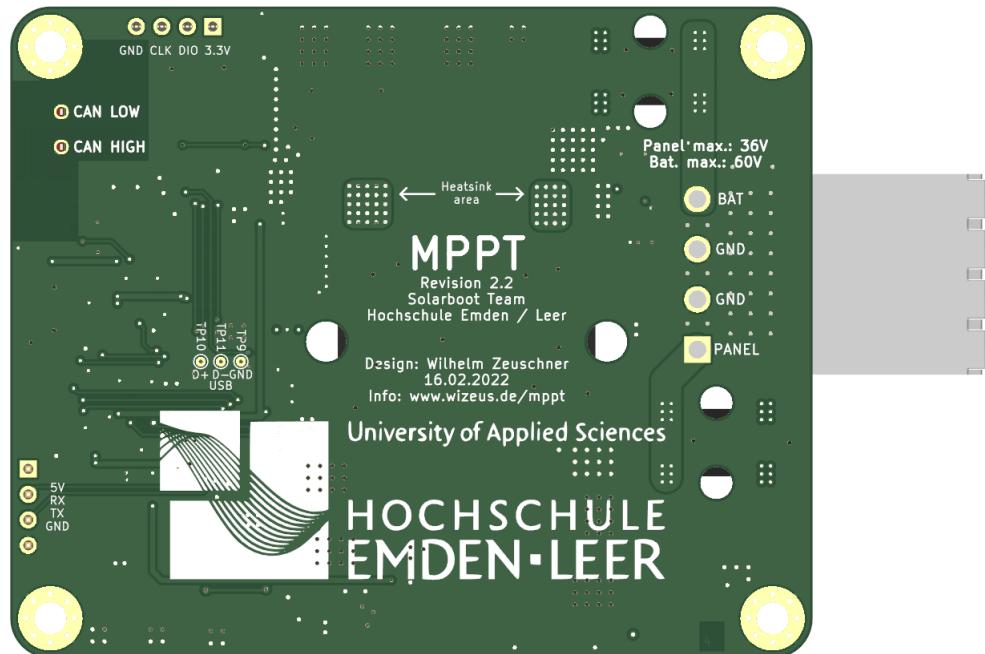


Abbildung 36: Unterseite der Platine

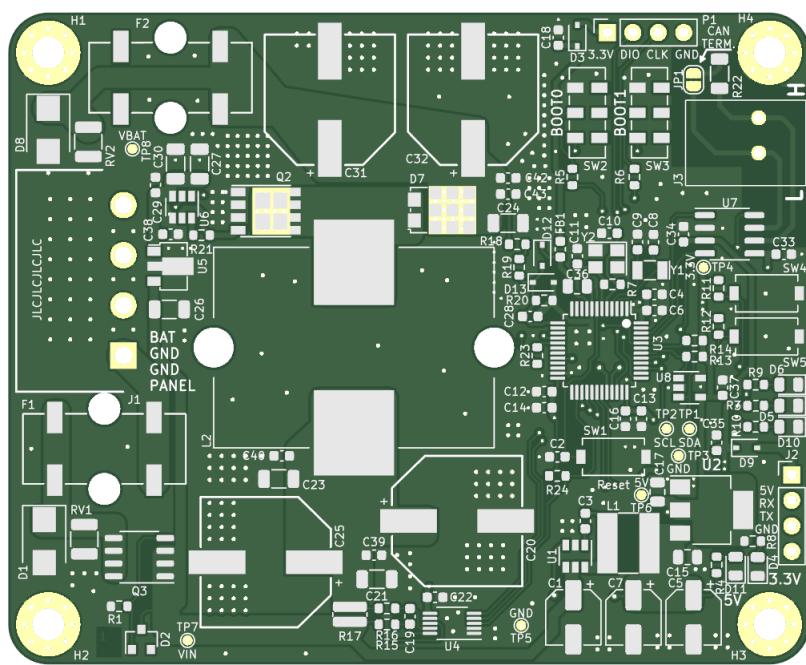


Abbildung 37: Oberseite der unbestückten Platine

### 3.5.3 Ein- und Ausgangsschutzschaltung

*In den folgenden Abschnitten sind die Ground-Fills ausgeblendet, um das Layout auf den Bildern leichter erkennen zu können.*

Beim Layout der Schutzbeschaltung (Siehe Abb. 38) mussten keine Besonderheiten berücksichtigt werden, da nur niederfrequente Ströme uns Spannungen vorliegen. Die Leiterbahnen für die leistungsführenden Teile wurden so breit wie möglich ausgelegt. Zum Teil wurde die Leiterbahn sowohl auf der Oberseite als auch auf der Unterseite geroutet, um den Widerstand der Leiterbahn zu minimieren. Die Segmente sind somit parallel geschaltet.

Bei den TVS-Bauteilen wurde darauf geachtet, die Leiterbahnen besonders kurz und breit zu halten. Somit haben diese einen niedrigen Widerstand und eine geringe Induktivität. Dies ist wichtig, da im Fehlerfall kurzzeitig ein hoher Strom fließen kann bzw. muss.

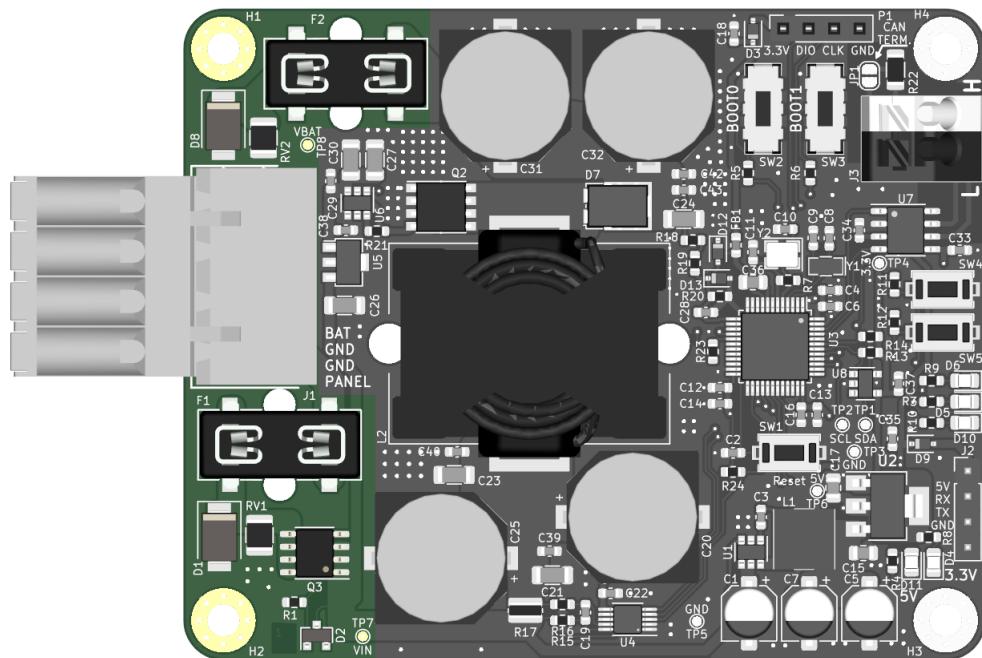


Abbildung 38: Ein- und Ausgangsschutzschaltung markiert

### 3.5.4 Hochsetzsteller

Der Hochsetzsteller (Siehe Abb. 39) ist das zentrale Element der Platine. Es werden große Ströme mit hoher Frequenz geschaltet, daher kommt dem Layout eine zentrale Rolle zu, um eine fehlerfreie und störarme Funktion zu gewährleisten. Das Layout der einzelnen Teile des Hochsetzstellers werden in den folgenden Abschnitten erläutert.

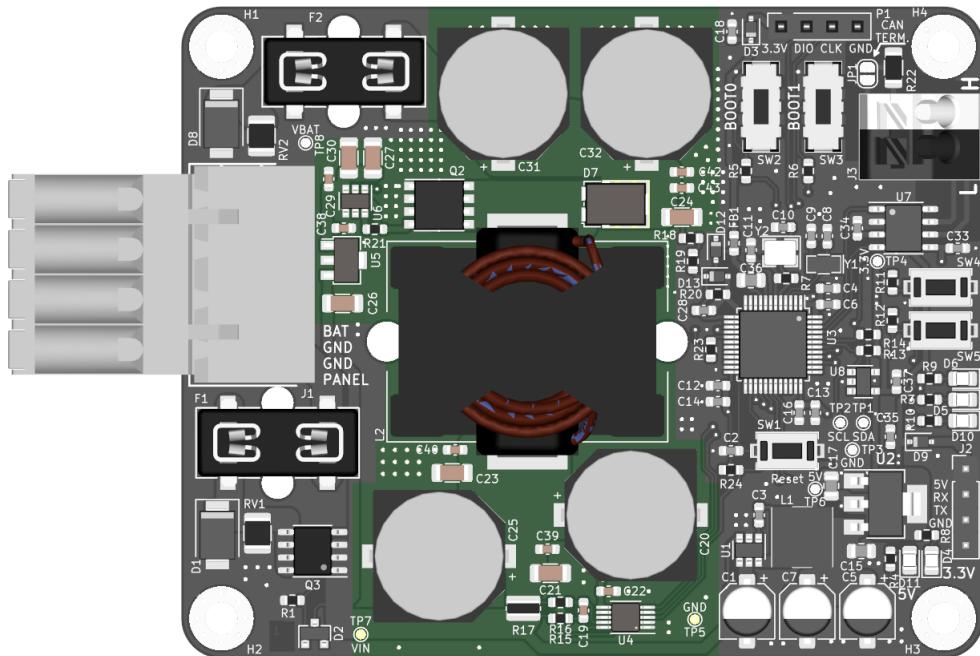


Abbildung 39: Hochsetzsteller markiert

### 3.5.4.1 Leistungsmessung

Die Leistungsmessung am Eingang liefert wichtige Informationen für den MPPT-Algorithmus. Da ein dezidierter IC verwendet wird, muss keine komplexe analoge Schaltung aufgebaut werden.

Es wird ein Shtunt-Widerstand eingesetzt, über den der IC den Spannungsfall misst. Die Abgriffe am Shunt sind so ausgelegt, dass keine zusätzlichen Spannungsfälle, z.B. durch den Widerstand der Leiterbahnen das Ergebnis verfälschen (Siehe Abb. 40). Die passiven Bauelemente, wie Kondensatoren und Widerstände sind möglichst nah am IC und große Schleifen werden vermieden. Eine gute Masseverbindung wird durch zahlreiche Vias zwischen den Masseflächen garantiert.

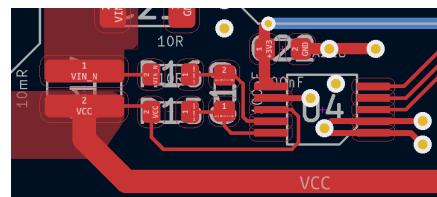


Abbildung 40: Layout des Leistungssensors

### 3.5.4.2 Leistungsteil

Die Abb. 41 zeigt das Layout des Leistungsteils. Auf dem rechten Teil der Abb. wurden sie relevanten Segmente markiert.

Am Eingang und Ausgang des Hochsetzstellers sind mehrere Kondensatoren, die über möglichst breite Leiterbahnen angeschlossen sind, um die Impedanz der Verbindung zu minimieren. Am Eingang des Boost-Konverters befindet sich keine aktive Elektronik.

Am Ausgang sind der MOSFET und die Diode verbaut. Diese sind so nah wie möglich an der Spule, die Eingang und Ausgang verbindet, verbaut. Auf diese Weise sind die Impedanzen der Verbindungen niedrig und die hochfrequenten Ströme fließen nicht durch große Schleifen. Der MOSFET schaltet die Spule auf Masse. Daher sind in der Nähe des MOSFETs sehr viele Vias, die eine sehr solide Masseverbindung herstellen. Dies trifft auch auf die Masseanschlüsse aller Kondensatoren zu. Die jeweils größten SMD-Pads der Diode und des FETs sind mit vielen Vias versehen, um die im Halbleiter entstehende Abwärme auf die andere Seite der Platine zu leiten. Sofern eine zusätzliche Kühlung nötig sein sollte, kann auf der Rückseite ein Kühlkörper aufgeklebt werden.

Die Spule kann mit einem Kabelbinder an die Platine befestigt werden, hierfür sind zwei Löcher vorgesehen. Diese sind türkis auf der Abb. 41 markiert. Die Befestigung soll zusätzliche Resistenz gegen Vibratoren bieten. Die Spule ist ein großes und verhältnismäßig schweres Bauteil.

Da die inneren Lagen vollständig mit Masse gefüllt sind, besteht ein sehr niederohmiger Rückweg für die Ströme, die beim Schalten auftreten. Dies reduziert Emissionen von elektromagnetischen Feldern.

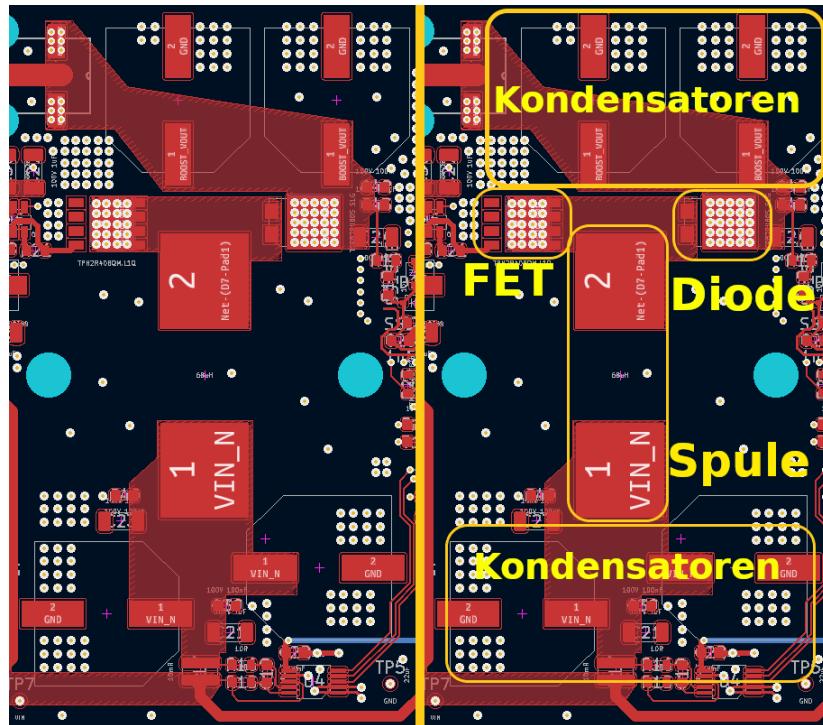


Abbildung 41: Layout des Leistungsteils

### 3.5.4.3 Gate Treiber

Die Schaltung des Gate-Treibers ist links vom MOSFET (Siehe Abb. 41) verbaut. Der Gate-Treiber lädt und entlädt das Gate des MOSFETs sehr schnell. Damit dies möglichst effektiv ist, ist die Leiterbahn zum Gate breit und kurz. Die Abblockkondensatoren sind ebenfalls so nah wie möglich am IC, um die Impedanz zu minimieren. Vias zu Masse sorgen für einen kurzen Return-Path für die Ströme, mit denen das Gate geladen wird.

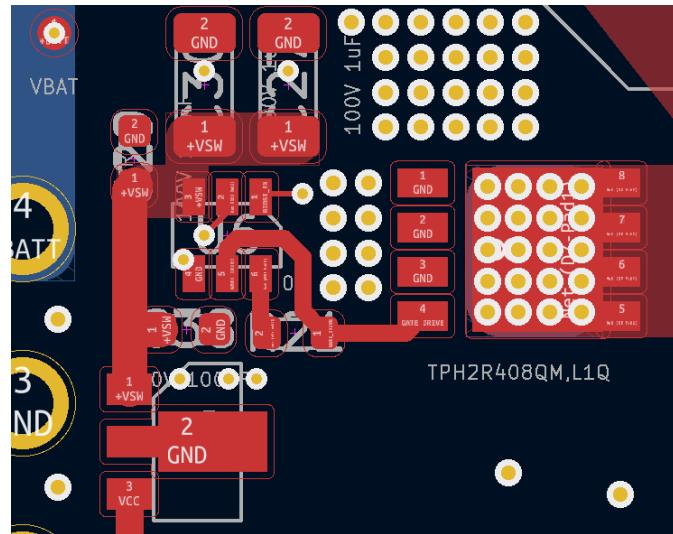


Abbildung 42: Layout des Gate-Treibers

### 3.5.5 Spannungsversorgung des Mikrocontrollers

Die Abb. 43 zeigt, wo die Spannungsversorgung des Mikrocontrollers erzeugt wird. Um die 5V-Spannung zu erzeugen, kommt ein Tiefsetzsteller-IC (engl. Buck-Converter) zum Einsatz. Da dieser eine Induktivität mit einer hohen Frequenz schaltet, muss das Layout sorgfältig erfolgen.

Die 3.3V-Versorgung wird durch einen 3.3V Linearregler erzeugt.

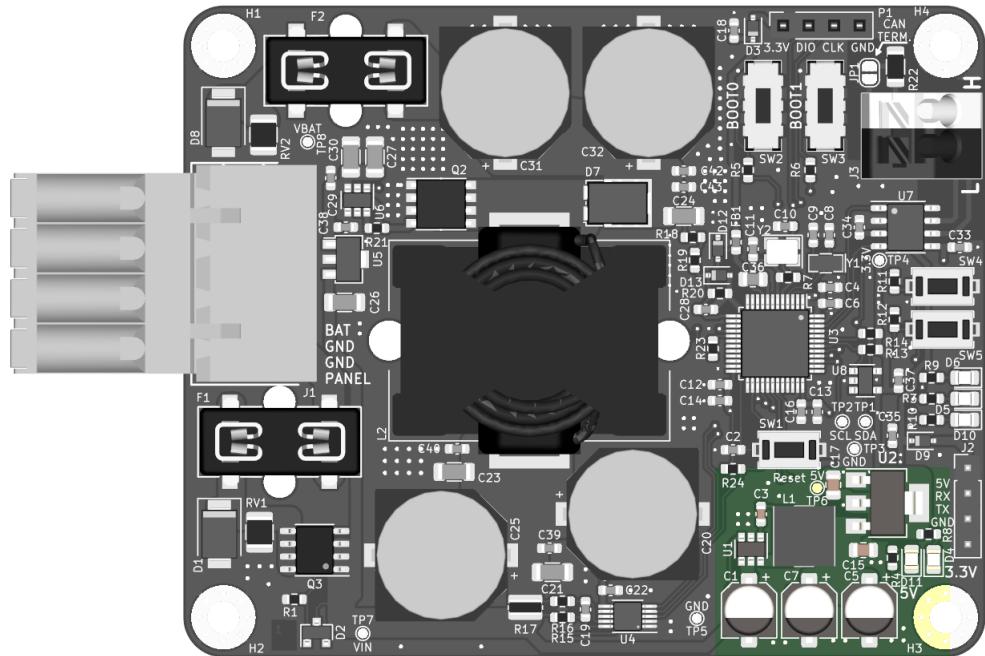


Abbildung 43: Spannungsversorgung des Mikrocontrollers markiert

Die Abb. 44 zeigt das Layout beider Spannungsregler. Der Tiefsetzsteller befindet sich auf der linken Seite. Der Linearregler ist oben rechts angeordnet.

Der Buck-Converter hat Kondensatoren am Ein- und Ausgang, die möglichst nah und mit breiten Leiterbahnen angeschlossen sind. Die Spule des Tiefsetzstellers ist ebenfalls so nah wie möglich am Tiefsetzsteller-IC. Es kommen ausgefüllte Flächen zum Einsatz, da diese in ihrer Geometrie genau an die Schaltung angepasst werden können und besonders niederohmig sind aufgrund der größeren Fläche im Gegensatz zu Leiterbahnen, die die Komponenten verbinden. Auch beim Tiefsetzsteller wurde, wie beim Hochsetzsteller, darauf geachtet, dass stets eine sehr gute Masseverbindung bei allen Bauteilen vorhanden ist, bei denen hohe bzw. hochfrequente Ströme zu erwarten sind. Außerdem wurden Stromschleifen auf ein Minimum reduziert.

Die Kondensatoren am Linearregler sind ebenfalls so nah wie möglich am IC wie möglich. Unter dem Linearregler ist eine Kupferfläche vorgesehen, um ggf. entstehende Abwärme etwas besser an die Umgebung abzugeben.

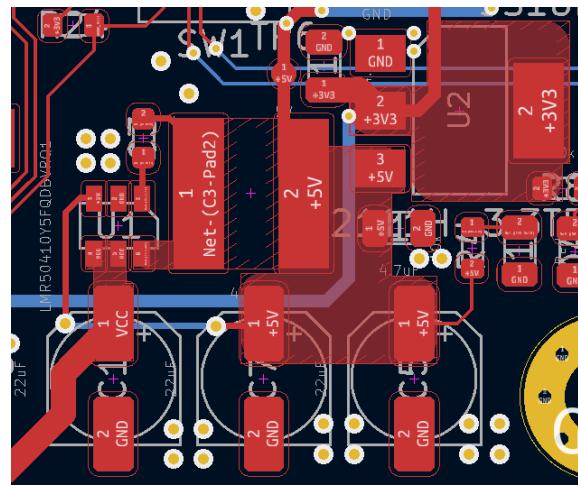


Abbildung 44: Layout der Spannungsversorgung

### 3.5.6 Mikrocontroller

Die Sektion Mikrocontroller umfasst den STM32, einen EEPROM, CAN-Transceiver und einige Bauteile für das Debuggen und die Programmierung des Mikrocontrollers. Auf der Abb. 45 sind diese Bauteile markiert. Größtenteils handelt es sich bei dem Layout dieser Signale um digitale Signale. Lediglich der Spannungsteiler zur Messung der Ausgangsspannung arbeitet analog.

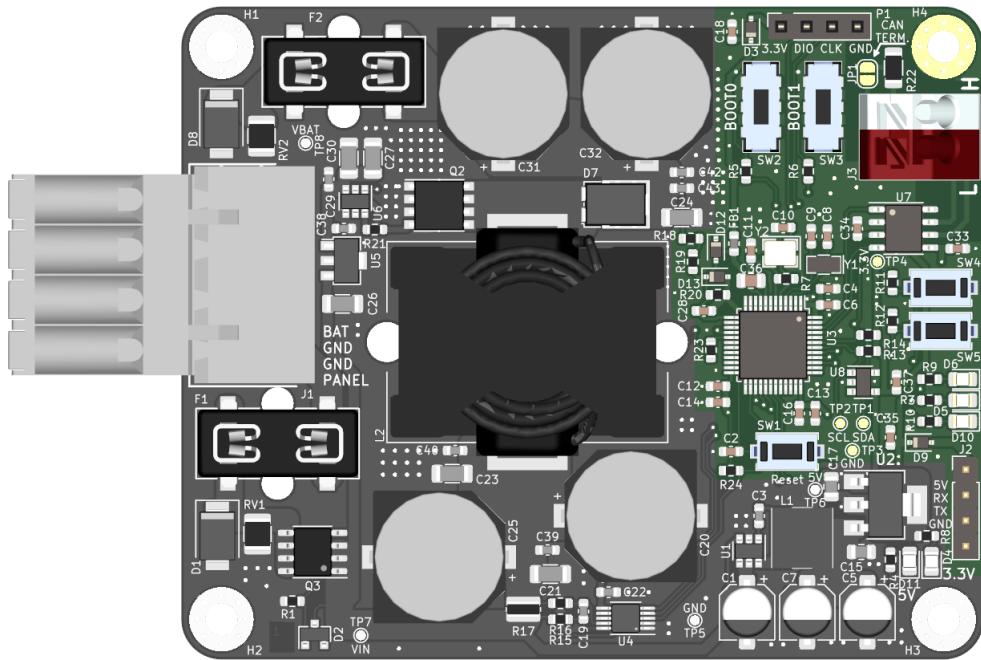


Abbildung 45: Mikrocontroller und dazugehörige Bauteile markiert

### 3.5.6.1 Spannungsteiler zur Messung der Ausgangsspannung und Oszillatoren

Der Spannungsteiler zur Messung der Ausgangsspannung ist aus mehreren Widerständen und Dioden zum Schutz des ADC-Eingangs aufgebaut (Siehe Abb. 46 l.o.). Beim Layout wurde darauf geachtet, die Bauteile in einem möglichst kleinen Areal unterzubringen und große aufspannende Schleifen zu vermeiden. Es gibt nur einen gemeinsamen Massepunkt zwischen der digitalen Masseverbindung des Mikrocontroller-Gehäuses und dem analogen Teil. Dies verhindert, dass Störungen durch galvanische Kopplung auftreten.

Die beiden Oszillatoren (Siehe Abb. 46 r.o.) sind nah am Mikrocontroller platziert und die Zuleitungen kurz, annähernd gleichlang und nah beieinander ausgeführt. Es wurde auf eine solide Masseanbindung gesorgt. Außerdem wurde eine Guard-Trace zwischen dem 8MHz-Taktsignal und den umgebenden Traces vorgesehen, um die kapazitiven Überkopplungen zu minimieren.

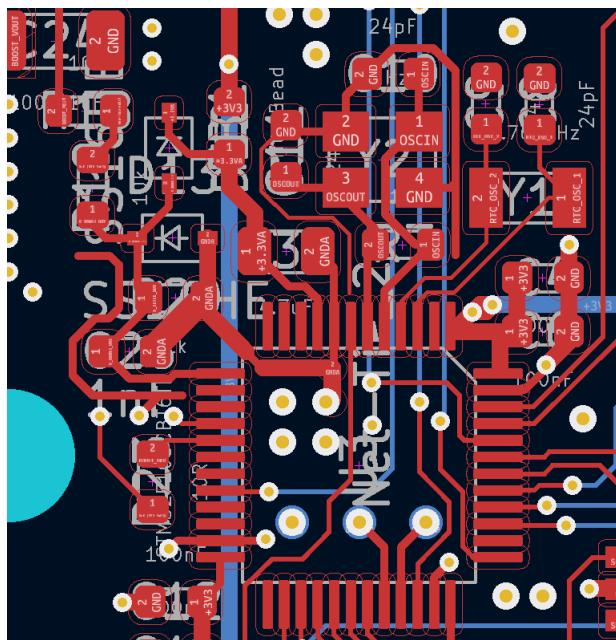


Abbildung 46: Layout des Spannungsteilers und der Oszillatoren

### 3.5.6.2 Debug-Interfaces

Bei den Debug-Interfaces handelt es sich um den SWD-Bus sowie ein UART Interface. Beides sind digitale Bussysteme, die verhältnismäßig robust sind. Daher waren keine besonderen Maßnahmen nötig.

### 3.5.6.3 CAN-Bus

Am CAN-Bus können unter Umständen und je nach bedarf eine Menge Geräte angeschlossen werden. Außerdem werden lange Leitungen verwendet. Das Boot stellt eine harsche Umgebung dar, was elektromagnetische Störungen angeht. Ein Teil dieser Störungen wird auch in das CAN-Signal eingekoppelt. Der Bus ist sehr robust gegenüber Störungen, da es sich um ein differentielles Signal handelt. Nichtsdestotrotz wurde der Teil der Signale, der die CAN-Low und High Signale beinhaltet gesondert behandelt. Die Masseflächen wurden nicht gefüllt, um den Teil gut von Rest der Schaltung zu trennen (Siehe Abb. 47). Um dies zu verdeutlichen wurde auf der Abb. 47 die Masseflächen im Gegensatz zu den anderen Abb. gefüllt.

Über den leicht zugänglichen Löt-Jumper (Siehe Abb. 47 l.o.) kann ein Terminierungswiderstand zwischen CAN-Low und High geschaltet werden.

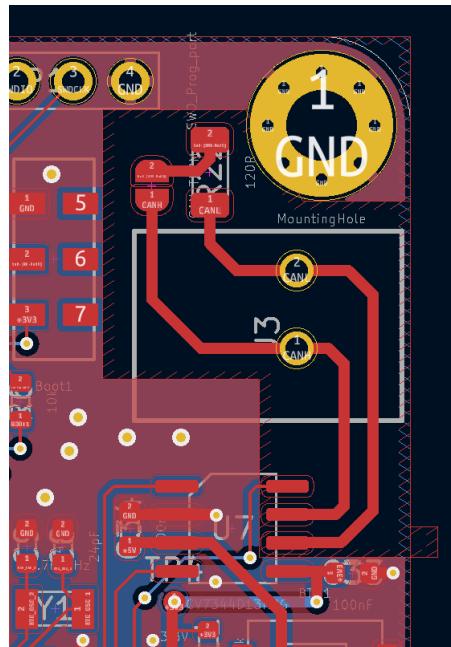


Abbildung 47: Layout CAN-Bus

### 3.6 Bestücken der Platine

Die entwickelte Platine wurde bei einem Dienstleister (JLCPCB) bestellt. Zunächst wurde mit einer Spritze Lötpaste auf alle SMD-Pads aufgebracht (Siehe Abb. 48). Im Anschluss wurden die Bauteile mit einer Pinzette an der passenden Position platziert (Siehe Abb. 49). Dabei handelte es sich um einen arbeitsintensiven Prozess. Danach wurde die Platine in einem Reflow-Ofen erhitzt und das Lötzinn verschmolz und stellte alle Verbindungen her.

Nach dem Verlöten wurde die Platine an eine Spannungsquelle mit Strombegrenzung angeschlossen. Es floss ein sehr hoher Strom, was auf einen Kurzschluss zwischen Masse und Eingangsspannung zurückzuführen war (Siehe Abb. 50). Nach Nachbeserungen mit einem Lötkolben war die Platine bereit, programmiert zu werden.

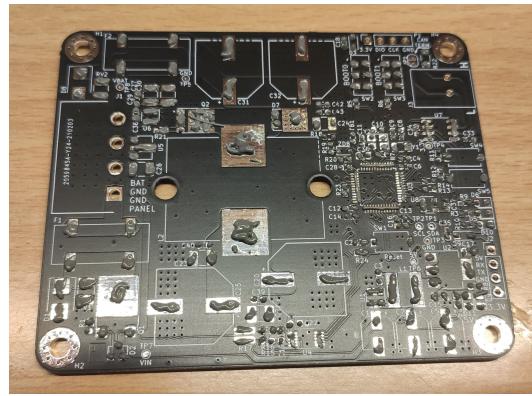


Abbildung 48: Platine mit aufgebrachter Lötpaste

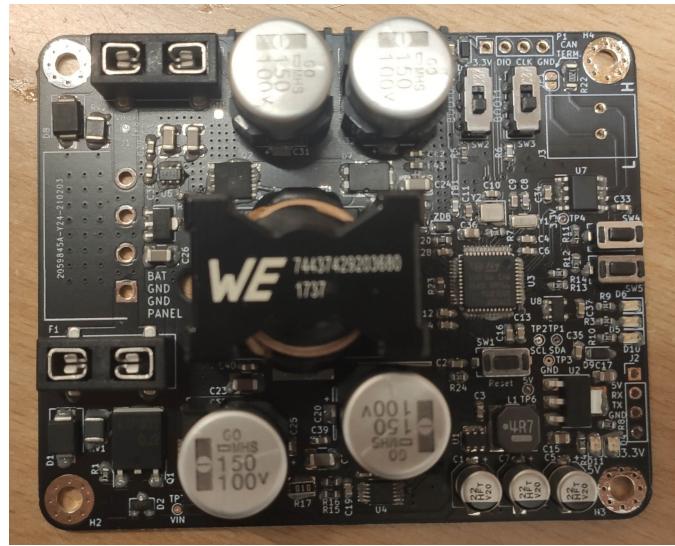


Abbildung 49: Platine mit aufgebrachter Lötpaste und Bauteilen

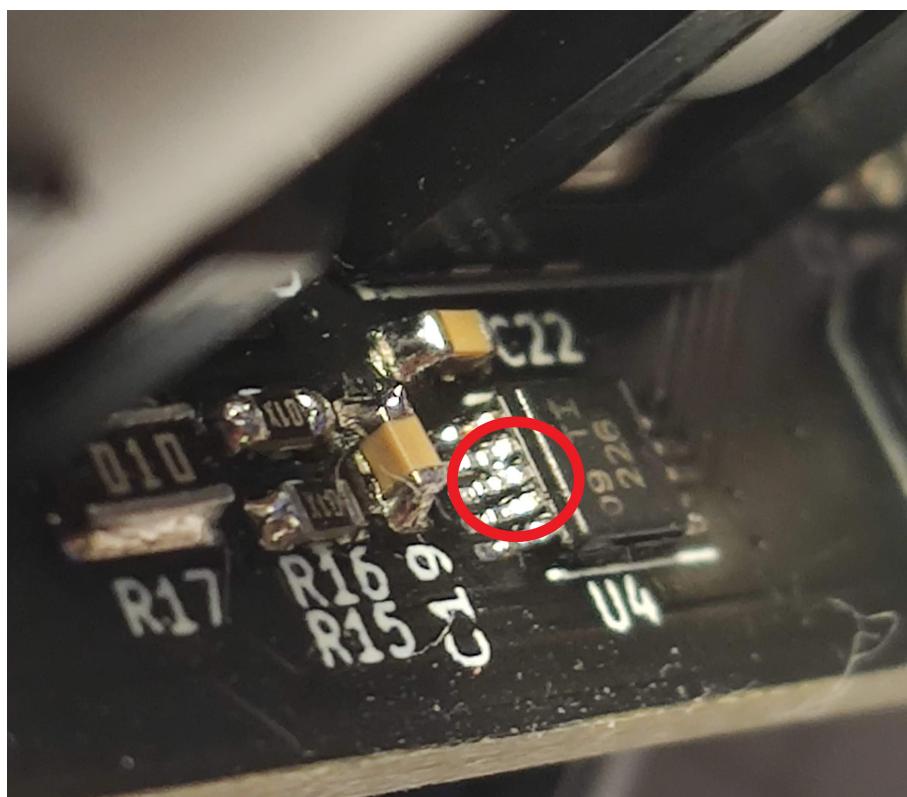


Abbildung 50: Ungewollte Lötbrücke mit Markierung

## 3.7 Entwicklung der Software

### 3.7.1 Allgemeines

Die Firmware, die auf dem Mikrocontroller ausgeführt wird, wurde in der Arduino-Umgebung entwickelt. Dabei wurde die PlatformIO IDE verwendet (Siehe Abb. 51). Diese bietet einige Vorteile gegenüber der Arduino IDE, die direkt von der Firma Arduino angeboten wird.

Ausschlaggebende Kriterien waren im Kontext dieser Arbeit:

- Code-Autocomplete
- guter Support zahlreicher Mikrocontroller
- bessere Benutzeroberfläche
- schnelleres Kompilieren von Code
- simple Verwaltung von Libraries

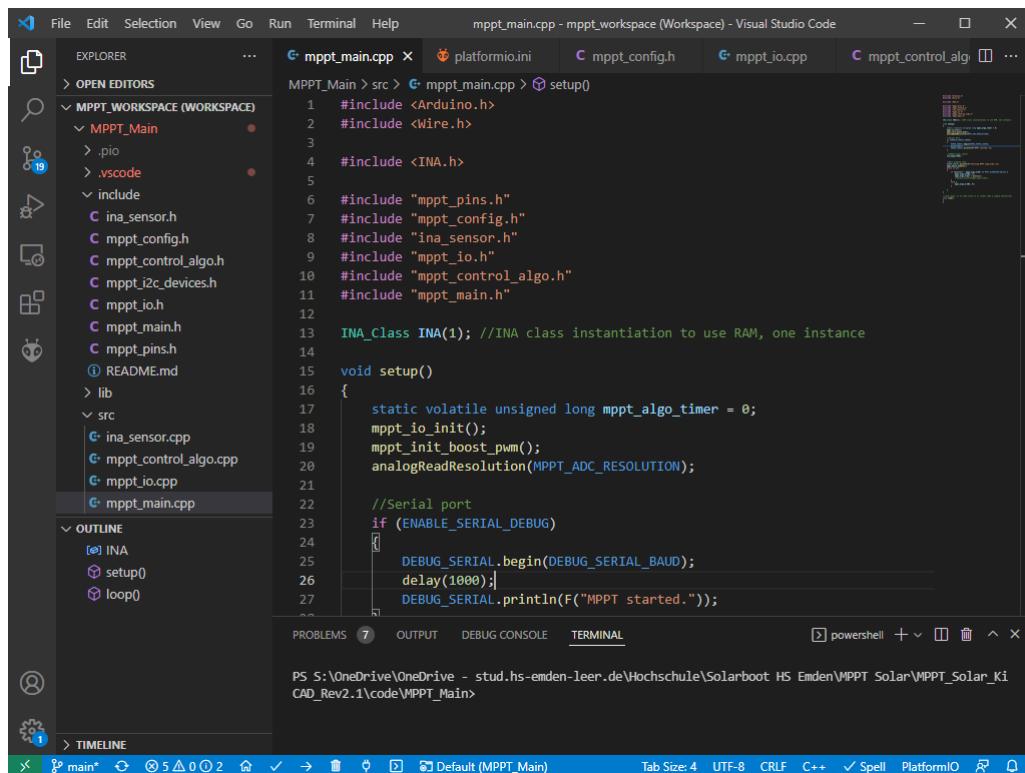


Abbildung 51: IDE PlatformIO

Der Sourcecode wurde über das Versionsverwaltungstool Git verwaltet (Siehe Abb. 52). Dies ermöglichte eine vereinfachte Arbeit auf mehreren Geräten und ermöglichte es, Änderungen, die zeitlich schon etwas zurücklagen leicht nachzuvollziehen.

The screenshot shows the GitHub Desktop application interface. At the top, it displays the repository 'MPPT\_Solar\_KiCAD\_Rev2.1'. Below this, there are tabs for 'Changes' (19) and 'History'. The 'Changes' tab is selected, showing a list of commits from various authors (Wilhelm Zeuschner) across different branches ('wip', 'before sym lib migration', 'before migration to v6', 'C: Generate Gerbers and Associated Artifacts', etc.). A specific commit titled 'mppt algorithm fixed!' is highlighted. The right side of the window shows a detailed diff view for this commit, comparing the code between the 'main' branch (old version) and the 'mppt algorithm fixed!' branch (new version). The diff highlights changes in the 'code\MPPT\_Main\src\mppt\_control\_algo.cpp' file, specifically in the 'mppt\_algo\_2' function. The new code includes comments explaining the algorithm's simplicity and how it tracks the maximum power point based on previous readings and current values.

```

@@ -13,6 +13,104 @@ MPPT_algorithm
13 //extern volatile float in_voltage, in_power, out_voltage;
14 volatile float in_power_1, in_power_2, in_power_initial;
15
16 */
17 //This algorithm is pretty simple.
18 //It is based the preverb and observe algorithm to track the maximum power point
19 //input power gets raised by raising the output voltage, respecting the maximum allowed
20 //output voltage.
21 +
22 //potential issues:
23 // p80 gets skipped if one of the previous conditions is true -> implement return
24 // or ignore?
25 */
26 void mppt_algo_2(INA_Class &INA, bool run_preturb)
27 {
28     //Reference values for previous power readings (last preverb and observe cycle)
29     //These get compared to the new values and based on the result of the comparison
30     //the pulse width gets adjusted
31     static volatile bool mppt_pwm_change_direction = 0; //=> PWM was decreased;
32     static volatile float mppt_prev_in_power = 0; //Measured input power of the last p80 run
33     //
34     //Measure current values
35     mppt_io_read_output_voltage(MPPT_ADC_READING_AVG_COUNT, out_voltage);
36     ina_read_voltage(INA, in_voltage);
37     //
38     //Check that the maximum output voltage is not exceeded
39     if (out_voltage > MPPT_MAX_OUT_VOLTAGE)
40     {
41         mppt_dec_boost_pwm();
42         mppt_leds_state(0);
43     }

```

Abbildung 52: Applikation GitHub Desktop für die Versionsverwaltung

Bei der Entwicklung wurde mit einem Oszilloskop gearbeitet, um relevante Signale messtechnisch zu erfassen (Siehe Abb. 53). Zusätzlich waren bis zu vier Multimeter im Einsatz, um Ströme und Spannungen zu messen. Die Schaltung wurde über ein Labornetzteil gespeist, das eine Strombegrenzung bot. Somit konnte der maximale Eingangsstrom im Fehlerfall begrenzt werden. Vor dem Eingang des MPPT wurde ein niederohmiger Widerstand verschaltet. An diesem fiel abhängig vom Eingangstrom verschiedene viel Spannung ab. Dieser Widerstand war wichtig für den Test des MPPT-Algorithmus.

Der Ausgang des MPPT wurde mit einer variablen elektronischen Last belastet. Außerdem wurden relevante Variablenwerte über UART ausgegeben, um das Debugging zu vereinfachen.

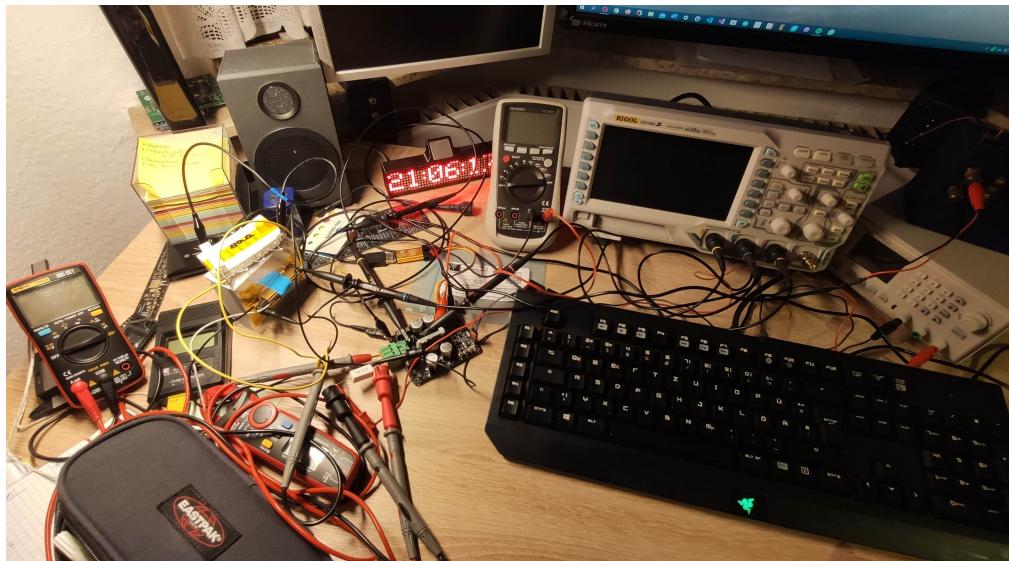


Abbildung 53: Aufbau während der Softwareentwicklung

### 3.7.2 Funktion des MPP-Algorithmus

Der gesamte Quellcode befindet sich im Anhang 5.2 dieser Ausarbeitung.

Der MPPT Algorithmus arbeitet nach dem sogenannten "perturb and observe" Verfahren. Dabei misst der Mikrocontroller kontinuierlich die Eingangsleistung, die von den Solarpanelen bereitgestellt wird. Ziel ist es, die Solarleistung zu maximieren. Je nach dem, wie die Umgebungsbedingungen sind und wie hoch der Laststrom am Solarpanel ist, schwankt der sog. Maximum Power Point des Panels, an dem das Produkt aus Solarspannung und Solarstrom (entspr. Solarleistung) maximal ist. Die Abb. 54 zeigt exemplarisch den Zusammenhang zwischen dem Strom und der Spannung, die von einem Solarpanel geliefert werden. Die Aufgabe des MPPT-Algorithmus besteht darin, das Maximum kontinuierlich zu finden und zu halten. Die Abb. 54 zeigt die Leistung-zu-Spannung-Abhängigkeit zweier Panel. Es wird deutlich, dass jedes Panel unterschiedlich ist. Daher ist es nicht möglich, einfach einen festen Leistungswert vorzugeben, der für jedes mögliche Panel funktioniert. Es besteht zwangsläufig die Notwendigkeit einer dynamischen Nachjustierung.

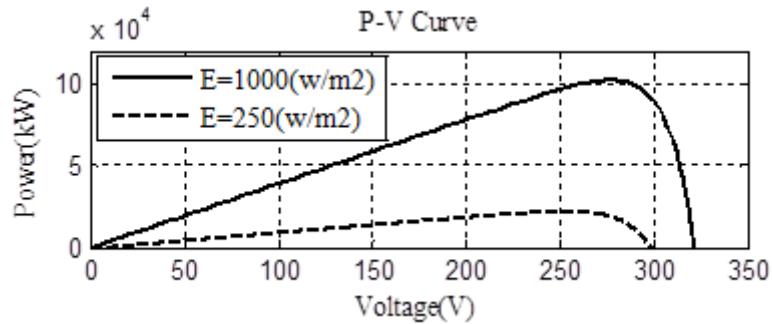


Abbildung 54: Maximum Power Point versch. Solarpanel [4]

Übertragen auf dieses Projekt bedeutet dies, dass der Mikrocontroller über den INA226 Leistungssensor die Eingangsleistung misst. Anhand der aktuellen Messung und der vorherigen Messung kann so festgestellt werden, ob die Leistung am Eingang zugenommen, abgenommen oder unverändert geblieben ist. Die Tabelle 3 zeigt, wie auf die verschiedenen Szenarien reagiert wird. Auf der Abb. 58 ist das Verhalten des Algorithmus im Detail dargestellt.

<b>Eingangsleistung</b>	<b>Reaktion der Software</b>	<b>Effekt</b>
Zunahme	Pulsweite anheben	Ausgangsspannung steigt
Abnahme	Pulsweite absenken	Ausgangsspannung sinkt
keine Änderung	PWM unverändert	Ausgangsspannung bleibt gleich

Tabelle 3: Aktionen des MPPT Algorithmus

Ein Anheben der Ausgangsspannung bewirkt unmittelbar eine Zunahme der Ausgangsleistung, wenn davon ausgegangen wird, dass eine ohmsche Last angeschlossen ist. Da der MPPT einen Li-Ion Akku lädt, bewirkt eine höhere Ausgangsspannung einen höheren Ladestrom. Somit kann die Annahme bestehen bleiben, dass die Eingangsleistung über die Änderung der Ausgangsspannung angehoben und abgesenkt werden kann.

Die Ausgangsspannung darf nicht unendlich weit angehoben werden. Wenn der Akku beispielsweise bereits voll geladen ist und kein weiterer Verbraucher angeschlossen ist, dann wird die Ausgangsspannung auf die Ladeschlussspannung des Akkus begrenzt. Sofern dies erfolgt, so kann der MPP nicht erreicht werden, da die Leistung nicht benötigt wird.

Außerdem kann über die Information der Eingangsleistung die maximale Leistung, die vom MPPT umgesetzt wird, beschränkt werden. Wenn das zulässige Limit erreicht wurde, so darf die Ausgangsspannung nicht weiter angehoben werden, da dies eine Leistungszunahme zur Folge hätte.

Da die Arduino-Umgebung verwendet wurde, gibt es bereits ein bestehendes Gerüst aus den Methoden `void setup()` und `void loop()`, die automatisch beim Start des Mikrocontrollers aufgerufen werden. `setup()` läuft in der Regel einmalig, wobei `loop()` kontinuierlich aufgerufen wird. Bei `loop()` handelt es sich daher im Prinzip um eine Endlosschleife. Allerdings werden im Hintergrund noch weitere Operationen ausgeführt, auf die der Programmierer keinen Einfluss hat. Somit ist der Code etwas schneller, wenn in `setup()` eine Endlosschleife eingefügt wird, in der der MPPT-Algorithmus aufgerufen wird. `loop()` wird somit nie ausgeführt.

In `setup()` werden die Variablen und die Klassen für die IO, die PWM und den ADC initialisiert. Außerdem wird der INA226 Leistungssensor initialisiert. Sofern das entsprechende `#define` aktiv ist, wird auch ein Debugging-Interface über UART aktiviert. Die Aufrufe, die von `setup()` ausgehen sind auf der Abb. 55 dargestellt.

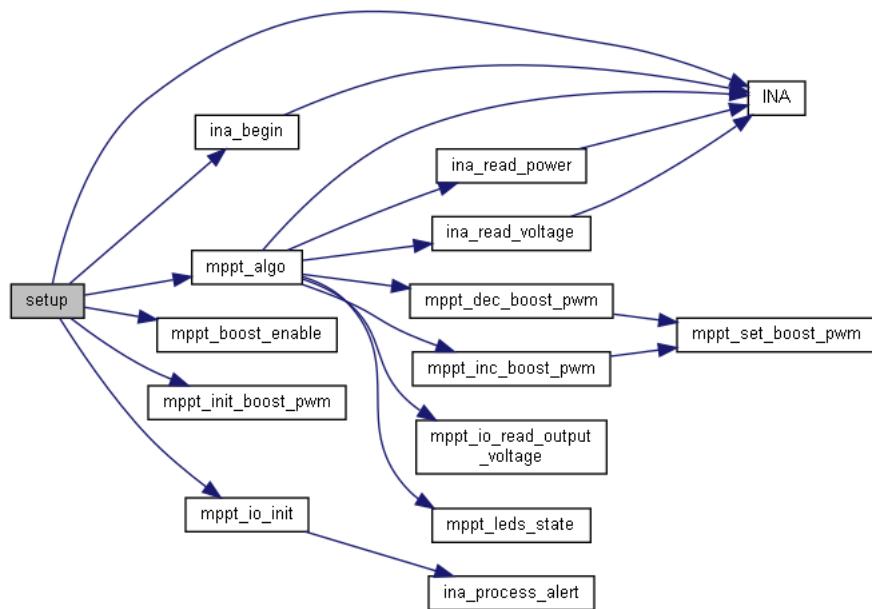


Abbildung 55: Aufrufgraph `setup()`

Im Anschluss beginnt das Programm mit der Endlosschleife, in der kontinuierlich der MPPT-Algorithmus aufgerufen wird. Dieser Aufruf ist ebenfalls auf Abb. 55 unter dem Namen mppt\_algo dargestellt. Zwecks besserer Lesbarkeit zeigt die Abb. 56 die Aufrufe, die ausschließlich von der Methode mppt\_algo() ausgehen.

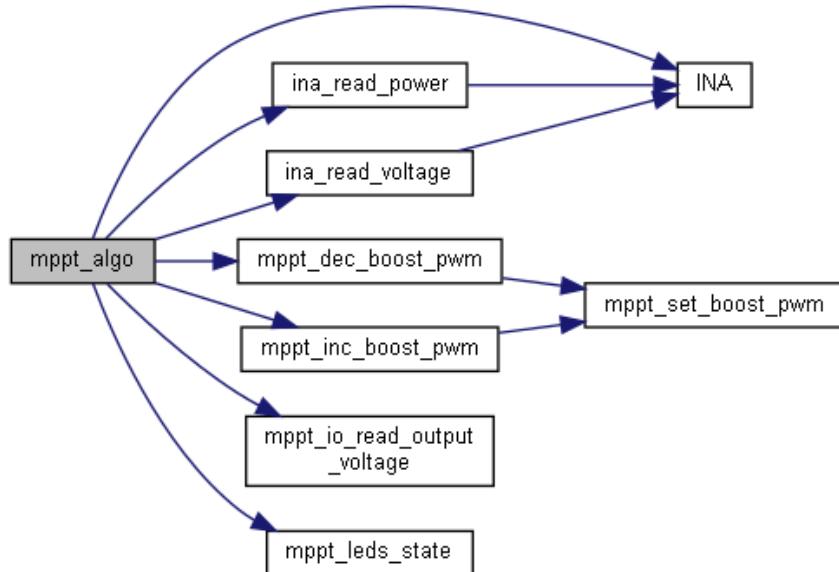


Abbildung 56: Aufrufgraph mppt\_algo()

Da der MPPT-Algorithmus essenziell für die Funktion ist, wird dieser im Anschluss detailliert erläutert.

Zu Beginn wird sichergestellt, dass die Spannungsgrenzen an Ein- und Ausgang eingehalten werden. Wenn die Eingangsspannung zu niedrig ist, dann ist eine zuverlässige Funktion des MPPT nicht mehr gewährleistet. Es können Probleme auftreten mit dem Hochsetzsteller. Zum einen ist die Gate-Treiber-Spannung von der Eingangsspannung abhängig. Sinkt die Eingangsspannung zu niedrig, so wird das Gate nicht mehr zuverlässig durchgeschaltet. Außerdem wird mit abnehmender Eingangsspannung das Verhältnis von Ausgangsspannung zu Eingangsspannung immer größer. Dies hat zur Folge, dass die Pulsweite der PWM immer weiter zunehmen muss. Auch hier gibt es Grenzen. Daher wurde die Mindestspannung des Eingangs auf 10V festgelegt. Wie viele andere Parameter kann dieser leicht verändert werden, da alle Parameter in einer Datei (mppt\_config.h) gesammelt sind. Sofern Änderungen nötig sein sollten, können diese übersichtlich und zentral vorgenommen werden. Die Ausgangsspannung muss begrenzt werden, um den Akku nicht zu überladen. Sofern eines der oben genannten Probleme durch das Programm erkannt wurde, wird die Pulsweite abgesenkt, was dazu führt, dass die Ausgangsspannung absinkt. Dies geschieht so lange, wie der Fehler besteht, ggf. bis zu einer Pulsweite von 0%. Zusätzlich zu den soeben erwähnten Sicherheitsmaßnahmen gibt es außerdem eine zweite Überprüfung der Ausgangsspannung. Falls diese 5V über dem zulässigen Maximum liegt, so wird die Pulsweite schlagartig auf 0% abgesenkt. Dies kann nötig sein, wenn sie Last schlagartig entfernt wird. In diesem Fall kann es zu lange dauern, bis die Pulsweite pro Aufruf etwas abgesenkt wurde, was zur Folge hat, dass

die Ausgangsspannung unzulässig hoch werden würde. Diese Überprüfung hat sich beim Debugging als äußerst nützlich erwiesen. Nach der Überprüfung der Sicherheitsparameter beginnt die Logik des "perturb and observe" Algorithmus. Dieser arbeitet nach den Vorgaben, die in Tabelle 3 dargelegt sind. Die Abb. 58 zeigt den Entscheidungsbaum, nach dem der Algorithmus arbeitet. Wichtig zu beachten ist, dass der Algorithmus zusätzlich zur Änderung der Eingangsleistung außerdem beachtet, welche Aktion im vorangegangenen Schritt erfolgte. Hiermit ist gemeint, ob die Pulsweite angehoben oder abgesenkt wurde. Dies ist notwendig, da der MPP sich sowohl links als auch rechts der aktuellen Leistung befinden kann. Die Abb. 57 verdeutlicht dieses Konzept.

Für den Fall, dass die aktuelle Leistung dem blau markierten Punkt entspricht, so würde eine Erhöhung der Pulsweite eine Zunahme der Leistung zur Folge haben. Das Gegenteil trifft zu für die rote Markierung, diese liegt rechts vom MPP. Daher muss die Pulsweite abgesenkt werden, um eine höhere Leistung zu erreichen.

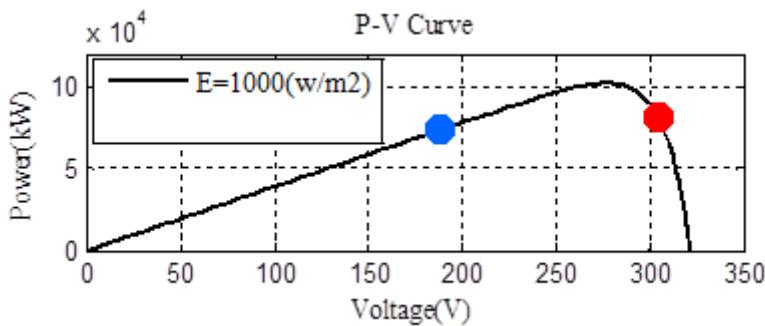


Abbildung 57: Maximum Power Point eines Solarpanels mit Markierungen [4]

Der MPP-Algorithmus wird von der Dauerschleife im Hauptprogramm zyklisch aufgerufen. Dies geschieht allerdings nicht mit der größtmöglichen Frequenz, die der Mikrocontroller erreichen kann. Grund hierfür ist, dass eine Änderung der Pulsweite sich nicht sofort auf die Ausgangsspannung auswirkt. Dementsprechend wären die zu messenden Werte beim folgenden Aufruf des Algorithmus möglicherweise noch nicht die finalen stabilen Werte. Daher wurde ein konfigurierbares Intervall von 1ms für den Aufruf festgelegt.

Die sicherheitsrelevanten Überprüfungen der Ein- und Ausgangsspannung dürfen allerdings nicht mit dieser niedrigen Frequenz geprüft werden. Daher wurde ein Parameter zur Methode, die den Algorithmus beinhaltet hinzugefügt. Wenn dieser "1" ist, so wird der gesamte Algorithmus ausgeführt, andernfalls erfolgt nur eine Überprüfung der sicherheitskritischen Werte.

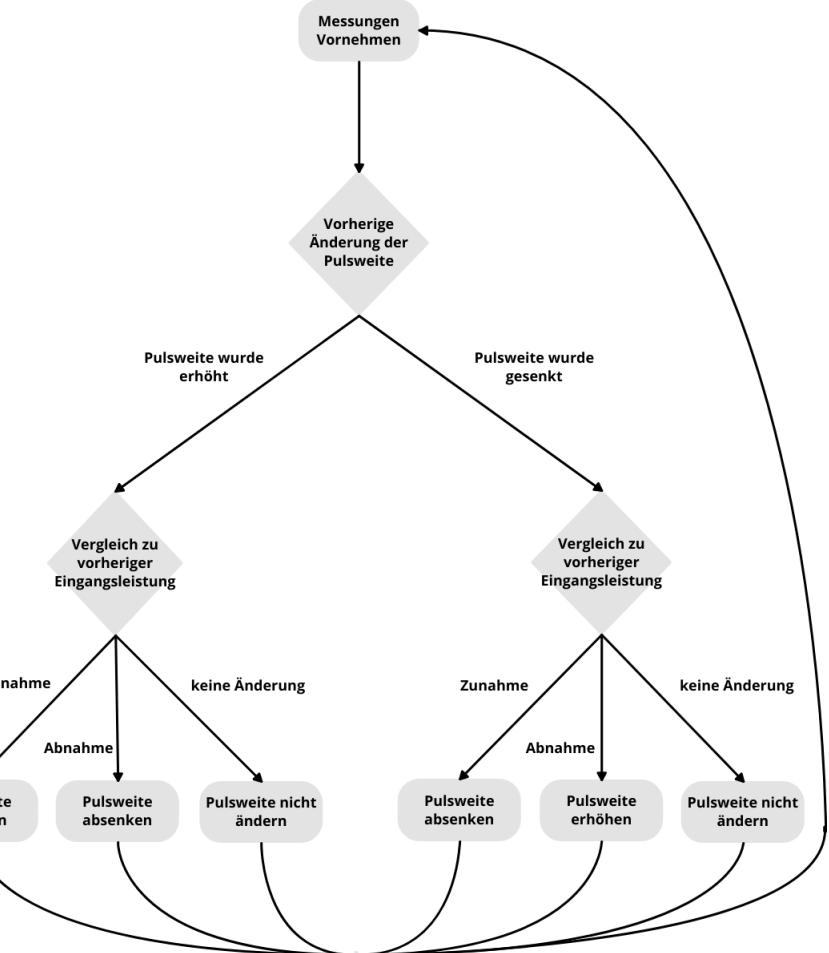


Abbildung 58: Entscheidungsbaum des Preturb-and-Observe Algorithmus

### 3.7.3 Weitere Programmkomponenten

#### 3.7.3.1 Parametrisierung

Wie in den vorangegangenen Kapiteln bereits erwähnt wurde, sind zahlreiche Konstanten über die Dateien mppt\_config.h, mppt\_pins.h und ina\_sensor.h parametrierbar.

In der Datei mppt\_config.h sind Parameter für die Anpassung der Arbeitsweise des MPPT-Algorithmus zu finden.

Die Datei mppt\_pins.h beinhaltet die Zuordnung der IO-Pins und ihrer Funktion.

In der Datei ina\_sensor.h können spezifische Parameter des Leistungssensors angepasst werden.

#### 3.7.3.2 INA226 Leistungssensor

Der Treiber für den INA226 Leistungssensor wird von einer Open-Source Library zur Verfügung gestellt. Der IC ist sehr komplex und es gibt über das Interface der Library zahlreiche Einstellmöglichkeiten. Analog zu vielen anderen Parametern, die ggf. angepasst werden müssen, sind diese in der Datei ina\_sensor.h verfügbar. Die Parameter wurden größtenteils unverändert bei den in einem Beispiel der Library

vorgegebenen Werten belassen. Es wurden lediglich die Werte angepasst, die die Konversionszeit beeinflussen. Diese Werte wurden im Rahmen der Entwicklung experimentell bestimmt. Es wurden niedrige Zeiten gewählt, damit die Konversion nicht zu viel Zeit in Anspruch nimmt. In einer Umgebung mit mehr Störungen kann es hilfreich sein, die Konversionszeit zu erhöhen oder die Zahl der Mittelwertbildungen zu erhöhen. Im Fall des MPPT ist dies aber nicht so oft erforderlich, wie im Beispiel.

### **3.7.3.3 MPPT IO**

Diese Komponente stellt nützliche Methoden bereit, die dazu dienen Aufgaben, die mit den IO-Pins des Mikrocontrollers zu tun haben, zu erfüllen.

Dazu gehören das Initialisieren der IO-Pins, das Lesen der Ausgangsspannung über den ADC und die Pulsweitenmodulation über eine spezielle Hardwarekomponente, die im STM32 integriert ist. Außerdem wird eine Schnittstelle zur Ansteuerung der Debugging-/Status-LEDs bereitgestellt.

### **3.7.3.4 CAN Schnittstelle**

Die Schnittstelle zum Austausch von Daten über den CAN-Bus ist zum Zeitpunkt der Abgabe dieser Arbeit nicht implementiert.

### 3.8 Verifikation und Messung

Um eine sichere Operation des MPPTs sicherzustellen und zu verifizieren, dass der Algorithmus zuverlässig den MPP findet, wurden eine Reihe von Messungen vorgenommen.

Der Aufbau auf Abb. 59 wurde für die Messungen verwendet.

Verwendete Messinstrumente und Laborgeräte:

- Oszilloskop: Rigol DS1054Z
- Elektronische Last: EastTester ET5411
- Labornetzteil: Riden RD6006
- Leistungswiderstand (umschaltbar zwischen  $5\Omega$ ,  $2,5\Omega$  und  $0\Omega$ )
- diverse Multimeter



Abbildung 59: Messaufbau

### 3.8.1 Sicherheitsrelevante Messungen

#### 3.8.1.1 Transient / Sprungantwort

Es wurden Sprungantworten aufgenommen für Eingangsspannungssprünge und Ausgangslastsprünge. Hierbei ist wichtig, dass der MPPT zuverlässig auf dynamische Änderungen reagiert und es nicht z.B. zu einer überhöhten Ausgangsspannung kommt und dass die Ausgangsspannung sich schnell wieder erholt nach dem Zuschalten einer Last.

Exemplarisch gezeigt wird hier eine Sprungantwort auf das Zuschalten der Ausgangslast bei 36V Eingangsspannung, 50V Ausgangsspannung und 60W Ausgangsleistung nach dem Zuschalten der Last 60.

Kanal 1: Eingangsspannung, Kanal 2: Ausgangsspannung.

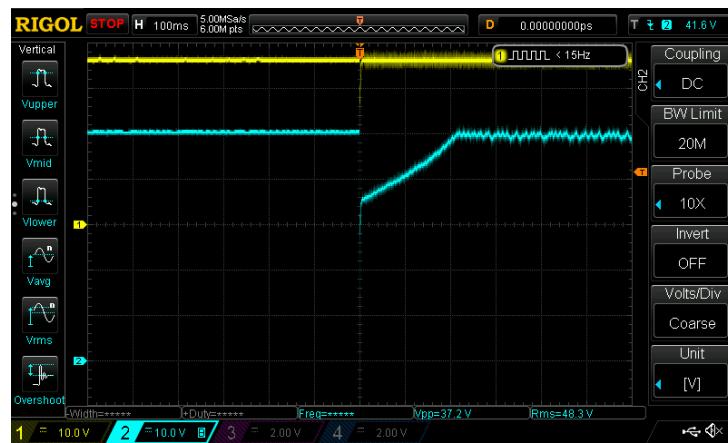


Abbildung 60: Sprungantwort Ausgangslast

Die Abb. 61 zeigt die Sprungantwort auf eine von 36V auf 16V fallende Eingangsspannung bei 60W Ausgangslast.

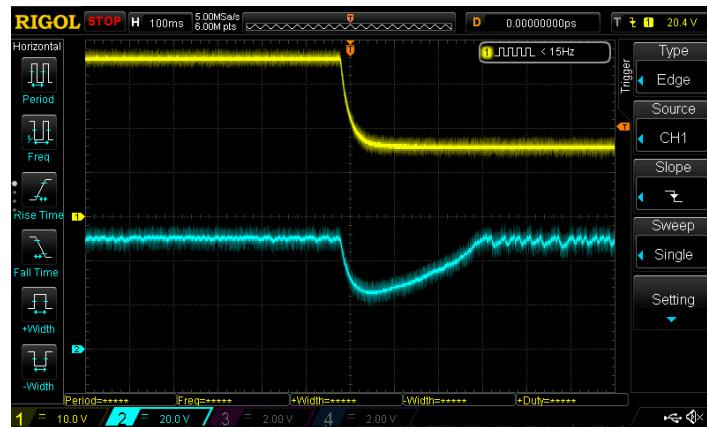


Abbildung 61: Sprungantwort fallende Eingangsspannung

### 3.8.1.2 Zu niedrige Eingangsspannung

Der MPPT muss den Hochsetzsteller abschalten, wenn die Eingangsspannung unter den festgelegten Wert fällt. Die Abb. 62 zeigt die Antwort des Reglers auf das Absecken der Eingangsspannung von 20V auf 10V. Wie zu sehen ist, wird der Ausgang abgeschaltet, wenn die Eingangsspannung zu niedrig für eine zuverlässige Operation ist. Kanal 1: Eingangsspannung, Kanal 2: Ausgangsspannung.

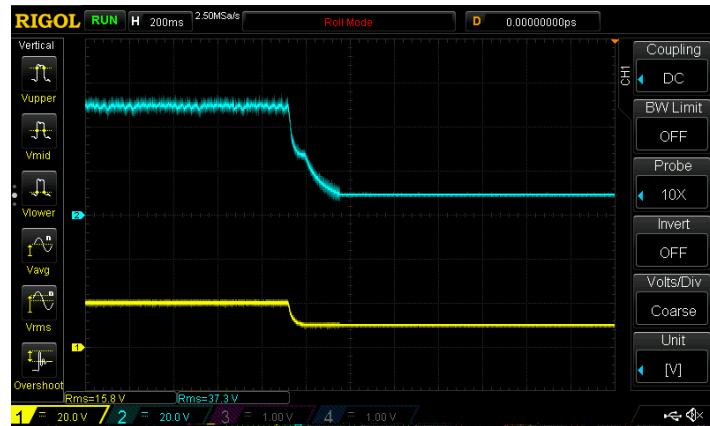


Abbildung 62: Sprungantwort fallende Eingangsspannung

### 3.8.2 Verifikation des MPPT-Algorithmus

#### 3.8.2.1 Theorie

Der Maximum Power Point Tracking Algorithmus wurde im Labor verifiziert. Daher war es nicht möglich, den MPPT an einem Solarpanel zu betreiben. Außerdem wären Messungen an einem Solarpanel nicht wiederholbar gewesen, da die Sonneninstrahlung nie gleich ist.

Aus diesem Grund wurde der Messaufbau (Siehe Abb. 63) verwendet.

Dieser besteht aus folgenden Komponenten:

Ein Schaltnetzteil am Eingang des Aufbaus stellte eine konstante Eingangsspannung zur Verfügung. Das Netzteil lieferte außerdem dank eingebauter Elektronik den Wert für den Gesamtstrom und die Gesamtleistung.

Nach dem Schaltnetzteil wurde ein Leistungswiderstand in Serie zur positiven Versorgungsleitung verschaltet. An diesem Widerstand fiel proportional zum Laststrom eine Spannung ab. Wenn die Last (der MPPT) einen sehr hohen Strom aus dem Schaltnetzteil abrufen sollte, so würde an dem Leistungswiderstand beinahe die gesamte Eingangsspannung abfallen. Das Gegenteil gilt für den Fall, in dem der MPPT nur einen sehr geringen Strom benötigt. In dem Szenario würde am Widerstand nahezu keine Spannung abfallen. Somit wurde mit Schaltnetzteil (CV) und Leistungswiderstand eine Spannungsquelle mit Innenwiderstand modelliert. Auf die genaue Bedeutung dieser Tatsache wird im nächsten Absatz genauer eingegangen.

Nach dem Vorwiderstand wurde der MPPT angeschlossen, auf dem der MPPT-

Algorithmus lief. Am Eingang des MPPT lag die durch den Vorwiderstand stromabhängig reduzierte Eingangsspannung an.

Am Ausgang des MPPT wurde eine elektronische Last angeschlossen, die im Constant-Voltage-Betrieb arbeitete.

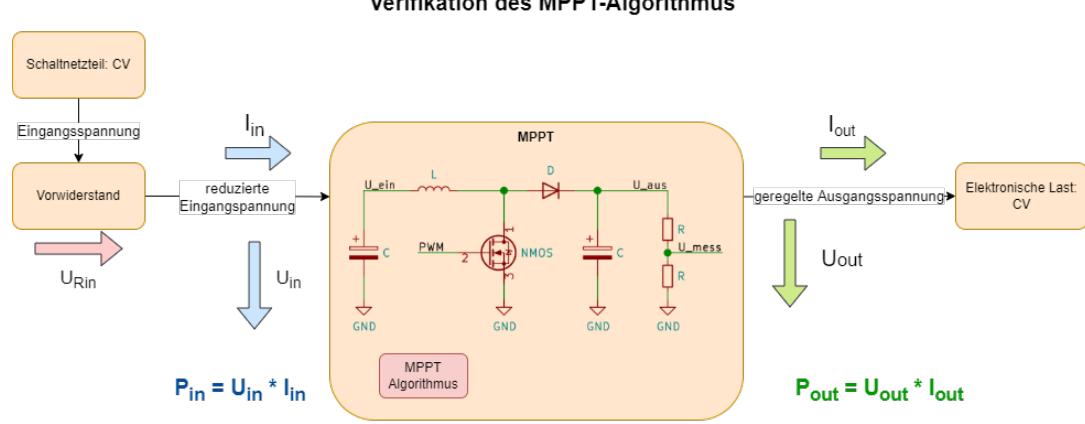


Abbildung 63: Messaufbau: Verifikation des MPPT-Algorithmus

Eine Spannungsquelle mit Innenwiderstand liefert im Falle der Leistungsanpassung die maximale Leistung. Diese Tatsache wurde bei der Messung des MPPT-Algorithmus ausgenutzt. Die Abb. 64 verdeutlicht diesen Zusammenhang. Die Hälfte der Quellenspannung fällt an dem Widerstand ab.

Ein gut funktionierender MPPT-Algorithmus wird dieses Optimum finden und verfolgen. Die veränderliche Solarspannung lässt sich simulieren, indem die Eingangsspannung verändert wird. Dadurch verschiebt sich das Optimum, in dem die maximale Eingangsleistung von der Quelle mit Innenwiderstand zur Verfügung gestellt wird. Der Algorithmus muss in der Lage sein, dieses neue Optimum zu finden.

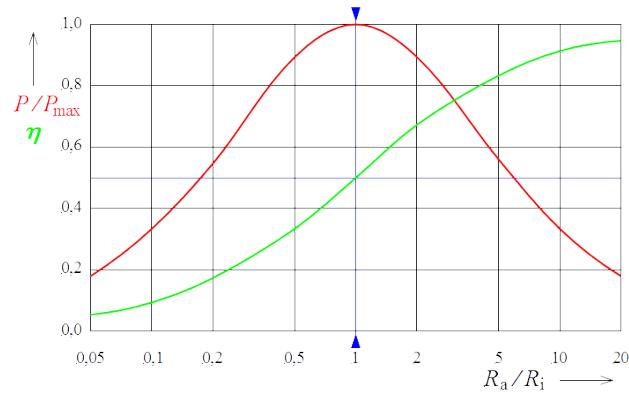


Abbildung 64: Leistungsanpassung: Spannungsquelle mit Innenwiderstand [9]

### 3.8.2.2 Messungen

Zur Verifikation der Funktion des MPPT-Algorithmus wurde der Vorwiderstand auf den Wert  $5\Omega$  eingestellt. Die Eingangsspannung betrug 30V, die Ausgangsspannung des MPPT maximal 50V.

Bei optimaler Leistungsanpassung fallen 15V am Vorwiderstand ab 11. Die Eingangsleistung des MPPT (abzüglich der Verluste im Vorwiderstand) beträgt 45W (Siehe Gl. 12 und Gl. 12).

$$U_{R_{vor}} = 30V - 15V \quad (11)$$

$$I_{R_{vor}} = I_{MPPT_{ein}} = \frac{U_{R_{vor}}}{R_{vor}} = \frac{15V}{5\Omega} = 3A \quad (12)$$

$$P_{R_{vor}} = P_{MPPT_{ein}} = U_{R_{vor}} * I_{R_{vor}} = 15V * 3A = 45W \quad (13)$$

Diese Hypothese wurde mit Hilfe mehrerer Messungen belegt. Bei den Messungen wurde der Ausgangstrom mit Hilfe der elektronischen Last variiert. Die Tab. 4 fasst die Ergebnisse der Messungen zusammen. Zu jeder Messung wurde eine Oszilloskopaufnahme angefertigt.

Wie die Messergebnisse belegen, funktioniert der MPPT-Algorithmus wie erwartet. Ab einer Eingangsleistung von ca. 45W senkt der MPPT seine Ausgangsspannung ab, um den Eingangsstrom zu beschränken, da dies Vorteilhaft ist für die Gesamtleistung. Somit findet der MPPT erfolgreich den Maximum Power Point.

MPPT Eingang			Vorwiderstand		Last		
Vin/V	Iin/A	Pin/W	R_vor/Ω	P_R_vor/W	Vout/V	Iout/A	Pout/W
30	0,025	0,75	5	0,003125	50	0	0
30	0,2	6	5	0,2	46	0,1	4,6
28	0,4	11,2	5	0,8	48,8	0,2	9,76
26,9	0,6	16,14	5	1,8	49,1	0,3	14,73
26	0,8	20,8	5	3,2	47,5	0,4	19
23,5	1,38	32,43	5	9,522	49,3	0,6	29,58
19	2,25	42,75	5	25,3125	49,5	0,8	39,6
15,4	2,99	46,046	5	44,7005	41,9	1	41,9
15,3	3	45,9	5	45	35,3	1,2	42,36
15,2	3	45,6	5	45	30,2	1,4	42,28
15,2	3	45,6	5	45	26,4	1,6	42,24

Tabelle 4: Messergebnisse zur Verifikation des MPP-Algorithmus

### 3.8.3 Belastungstest

Der MPPT wurde über eine Dauer von 30 Minuten mit 2,9A am Ausgang belastet. In der Firmware des MPPT wurde ein Limit von 130W Eingangsleistung festgelegt. Dieses Limit wurde eingehalten, indem der MPPT seine Ausgangsspannung auf 45V absenkte, obwohl bei niedrigerer Last 50V als Ausgangsspannung konfiguriert wurden. Die Abb. 65 zeigt, dass die Pulsweite auf ca. 45% beschränkt bleibt, was zu einer Ausgangsspannung von ca 45V führt.

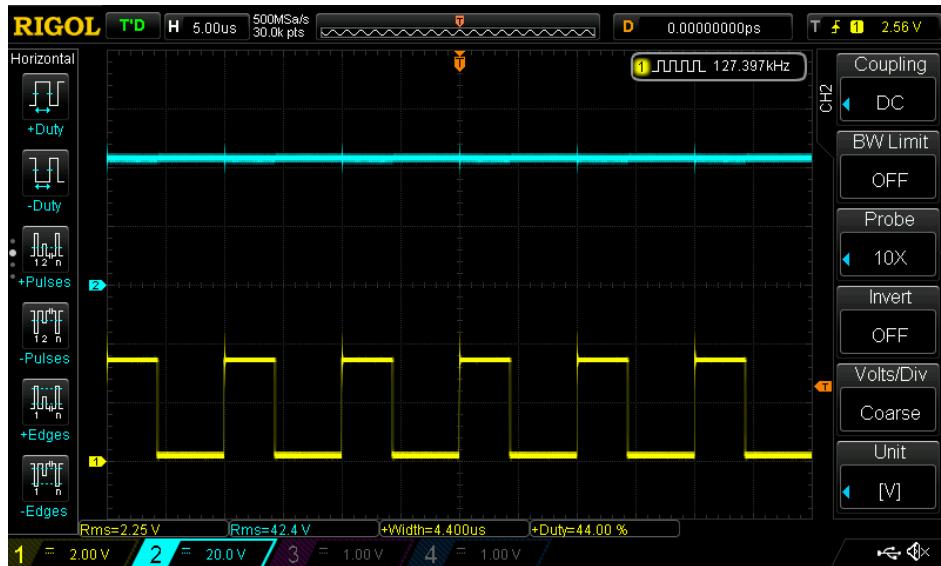


Abbildung 65: Belastungstest: Leistungsbegrenzung

Die beim Ende des Tests gemessene maximale Temperatur betrug 52°C, gemessen direkt unter der Diode des Hochsetzstellers. Die Temperatur des MOSFETs betrug 50°C (Umgebungstemperatur: 19 °C). Da beide Bauelemente nah beieinander platziert sind und die Platine eine hohe thermische Leitfähigkeit besitzt, haben sich beide Temperaturen angenähert. Es wird jedoch deutlich, dass die Verluste der Diode etwas höher sind. Die Abb. 66 und 67 zeigen eine Wärmebildaufnahme der Platine. Aufgrund der sehr niedrigen Sensorsauflösung (8x8 Pixel) und der Interpolation der Messdaten lassen sich nur qualitative Aussagen treffen. Auf Abb. 67 sind die Platine, der MOSFET und die Diode markiert. Die Aufnahme zeigt, dass sich die beiden Halbleiter am meisten erhitzen.

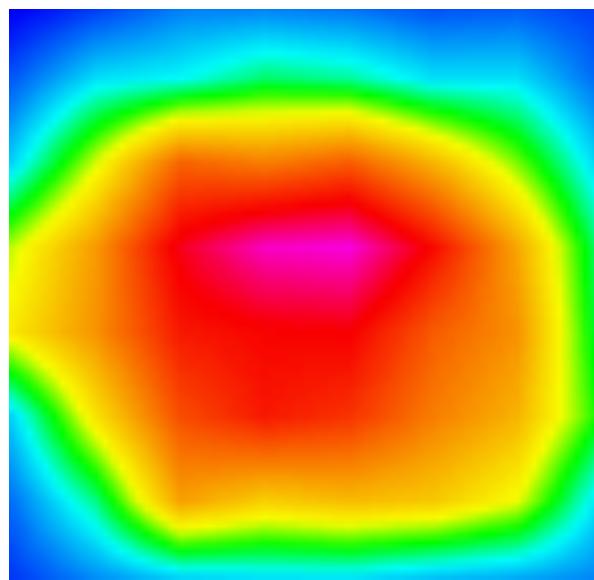


Abbildung 66: Wärmebildaufnahme ohne Markierungen

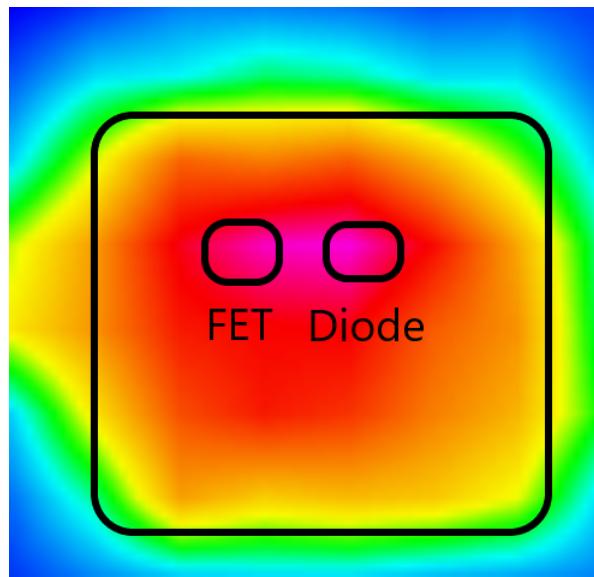


Abbildung 67: Wärmebildaufnahme mit Markierungen

### 3.8.4 Wirkungsgrad

Der Wirkungsgrad des MPPTs wurde bei verschiedenen Eingangsspannungen und einer einstellbaren Ausgangslast ermittelt. Für die Messung des Wirkungsgrads wurden die Eingangsspannungen 16V, 26V und 36V verwendet. Die Ausgangsspannung wurde in der MPPT Firmware auf 50V eingestellt. Ein Eingangsstrom von 6A wurde nicht überschritten. Die maximale Ausgangsleistung betrug 115W bei 36V Eingangsspannung. Die Messergebnisse sind in den Tabellen 5, 6 und 7 dargestellt.

Die Abb. 68 und 69 zeigen einen graphischen Vergleich der Wirkungsgrade in Abhängigkeit von der Eingangsspannung und der Ausgangsleistung.

Die Messergebnisse verdeutlichen, dass der MPPT einen sehr guten Wirkungsgrad aufweist.

Die Panelspannung beträgt im Leerlauf ca. 36V und sinkt bei Belastung ab. Daher wurden die Messwerte aus der Tabelle 6 als Basis gewählt. Die erwartete Leistung beträgt >70W pro Panel. Dementsprechend liegt der zu erwartende Wirkungsgrad bei >93% bei der beim Solarboot zu erwartenden Solarleistung.

<b>Uin</b>	<b>Iin</b>	<b>Pin</b>	<b>Uout</b>	<b>Iout</b>	<b>Pout</b>	<b>Wirkungsgrad</b>
16	0,50	8	48,10	0,15	7,3593	0,91
16	0,90	14,4	46,50	0,28	13,113	0,91
16	1,50	24	48,70	0,46	22,4994	0,93
16	2,19	35,04	48,80	0,67	32,8424	0,93
16	2,86	45,76	48,80	0,87	42,5536	0,92
16	3,54	56,64	48,80	1,07	52,216	0,92
16	4,30	68,8	49,00	1,30	63,504	0,92
16	4,95	79,2	48,90	1,48	72,372	0,91
16	5,70	91,2	49,00	1,68	82,32	0,90

Tabelle 5: Messdaten: Wirkungsgrad bei 16V Eingangsspannung

<b>Uin</b>	<b>Iin</b>	<b>Pin</b>	<b>Uout</b>	<b>Iout</b>	<b>Pout</b>	<b>Wirkungsgrad</b>
26,00	0,20	5,20	30,00	0,14	4,20	0,80
26,00	0,54	14,04	49,60	0,26	12,90	0,91
26,00	0,85	22,10	49,50	0,42	20,79	0,94
26,00	1,35	35,10	49,20	0,68	33,36	0,95
26,00	1,73	44,98	49,20	0,86	42,16	0,93
26,00	2,18	56,68	49,30	1,09	53,74	0,94
26,00	2,47	64,22	49,50	1,23	60,89	0,94
26,00	2,88	74,88	49,60	1,43	70,93	0,94
26,00	3,40	88,40	49,20	1,68	82,51	0,93
26,00	3,80	98,80	49,20	1,89	92,99	0,94
26,00	4,20	109,20	49,20	2,07	101,84	0,93
26,00	4,82	125,32	49,30	2,35	115,86	0,92

Tabelle 6: Messdaten: Wirkungsgrad bei 26V Eingangsspannung

<b>Uin</b>	<b>Iin</b>	<b>Pin</b>	<b>Uout</b>	<b>Iout</b>	<b>Pout</b>	<b>Wirkungsgrad</b>
36	0,18	6,30	48,00	0,10	4,94	0,78
36	0,27	9,65	49,60	0,17	8,28	0,85
36	0,36	12,96	48,60	0,23	11,37	0,87
36	0,52	18,72	49,20	0,35	17,12	0,91
36	1,04	37,26	49,50	0,70	34,65	0,92
36	1,30	46,80	49,50	0,90	44,65	0,95
36	1,51	54,36	49,50	1,05	51,98	0,95
36	1,83	65,88	49,60	1,26	62,50	0,94
36	2,10	75,60	49,50	1,46	72,27	0,95
36	2,40	86,40	49,50	1,66	82,17	0,95
36	2,91	104,76	49,60	2,00	99,20	0,94
36	3,40	122,40	49,60	2,33	115,57	0,94

Tabelle 7: Messdaten: Wirkungsgrad bei 36V Eingangsspannung

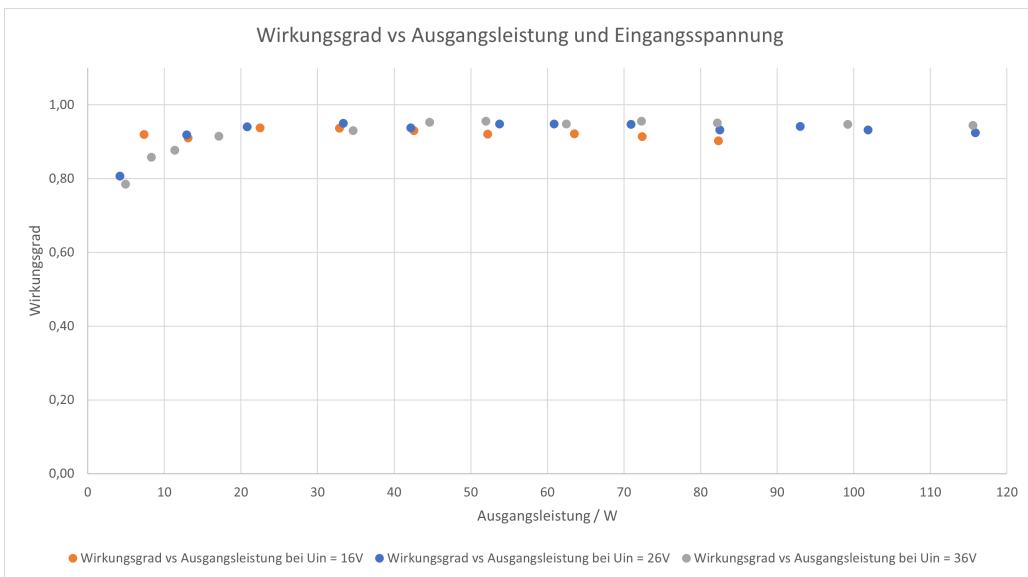


Abbildung 68: Vergleich der Wirkungsgrade bei verschiedenen Eingangsspannungen

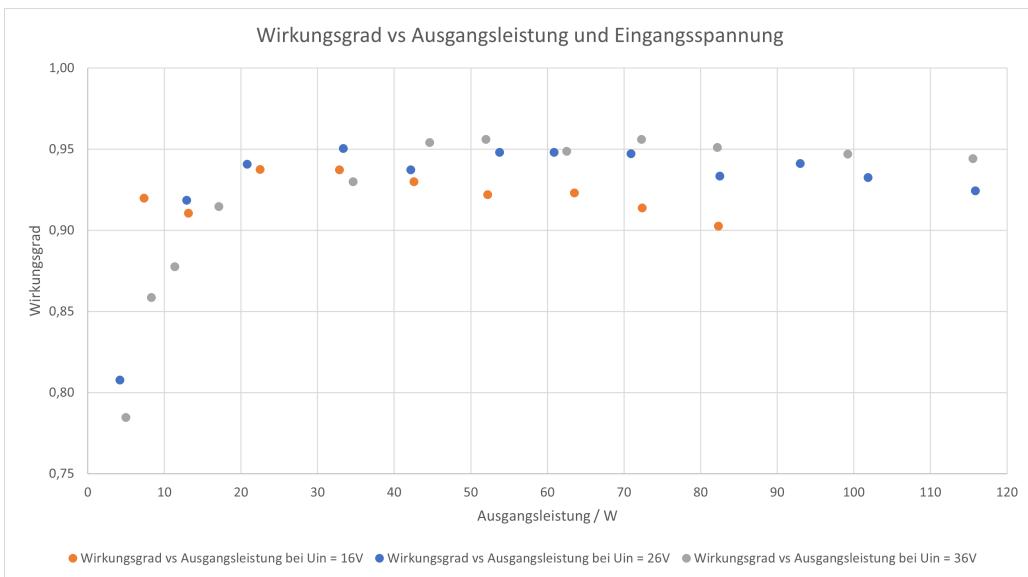


Abbildung 69: Vergleich der Wirkungsgrade bei verschiedenen Eingangsspannungen, verbesserte Auflösung

### **3.8.5 Paralleler Betrieb mehrerer MPPT an einem gemeinsamen Akku**

Im späteren Einsatz im Boot sind ca. 6 bis 8 MPPTs an einem gemeinsamen Akku angeschlossen, der von den MPPTs geladen wird. Der Akku kann als Spannungsquelle mit niedrigem Innenwiderstand aufgefasst werden. Somit würde der Akku die Ausgänge aller MPPTs auf die selbe Spannung beschränken. Je nach Sonneneinstrahlung würde dann jeder MPPT verschieden viel Strom in den Akku leiten.

Aufgrund fehlender Bauteile war es leider nicht möglich, mehr als eine MPPT-Platine vollständig zu bestücken (Siehe 4.1). Dementsprechend konnte diese Messung nicht durchgeführt werden.

### **3.8.6 Messungen mit Solarpanel und Messungen an einem Li-Ion Akku**

In Absprache mit dem Beteuer dieser Arbeit wurde entschieden, dass diese Messungen nicht vorzunehmen sind. Die Funktion des MPPT wurde bereits ausreichend demonstriert durch die hier geschilderten Messungen im Labor.

## 4 Schlusswort und Ausblick

### 4.1 Einflüsse der COVID-19 Pandemie

Aufgrund der globalen Maßnahmen zur Eindämmung des COVID-19-Virus kam es u.a. in der Halbleiterindustrie und Elektronikindustrie zu großen Lieferproblemen. Hiervon war auch dieses Projekt betroffen. Die Platine musste insgesamt zwei Mal verändert werden, da Bauteile nicht lieferbar waren und alternative Bauteile andere Footprints hatten.

Dies war mit einem erheblichen Mehraufwand verbunden.

Da die Bauteile fehlten, war es nur möglich, einen einzigen MPPT vollständig zu bestücken und zu verifizieren.

Die Bauteile wurden über den Distributor Digikey bezogen. Wie die Abb. 70 zeigt, sind eine Vielzahl der Bauteile (Stand 24.03.2022) nicht lieferbar. Unter diesem Gesichtspunkt macht es zum Zeitpunkt der Abgabe dieser Arbeit keinen Sinn, zu versuchen, das gesamte Boot mit den neu entwickelten MPPTs auszustatten. Essentielle Bauelemente, wie MOSFETs und Mikrocontroller nicht verfügbar. Die Preise vieler Bauteile haben sich ebenfalls erheblich erhöht. Insgesamt sind 14 von 51 Bauteilen nicht lieferbar, dies entspricht 27%. Hiervon sind 12 Halbleiter, bei zweien handelt es sich um Elektrolytkondensatoren. Teilweise wäre eine Substitution durch ein anderes Bauteil ohne größeren Aufwand möglich, im Falle des Mikrocontrollers oder des Buck-Reglers für die 5V-Spannungsversorgung müssten aber erhebliche Änderungen am Design erfolgen. Auch der Tausch des MOSFETs oder der Power-Diode ist wahrscheinlich nicht ohne Weiteres möglich, da sich das Footprint bei den Herstellern unterscheidet.

	296-29034-1-ND INA226AIDGSR Texas Instruments IC MONITOR PWR/CURR B/DIR 10MSOP <input type="checkbox"/> 38	Customer Reference	2	Backorder	3,26000	6,52 €
Check Lead Time						
	497-7280-1-ND L78L18ACUTR STMicroelectronics IC REG LINEAR 18V 100MA SOT89-3 <input type="checkbox"/> 39	Customer Reference	2	Immediate	0,48000	0,96 €
Check Lead Time						
	296-LMRS0410YSFQDBVR01CT-ND LMRS0410YSFQDBVR01 Texas Instruments AUTOMOTIVE QUALIFIED SIMPLE SWIT <input type="checkbox"/> 40	Customer Reference	2	Backorder	1,60000	3,20 €
Check Lead Time						
	NCV734D1032GOSCT-ND NCV734D1032G onsemi IC TRANSEIVER HALF 1/1 8SOIC <input type="checkbox"/> 41	Customer Reference	2	Backorder	0,81000	1,62 €
Check Lead Time						
	497-1734B-1-ND STM32F103C8T6TR STMicroelectronics IC MCU 32BIT 128KB FLASH 48LQFP <input type="checkbox"/> 42	Customer Reference	2	Backorder	6,66000	13,32 €
Lead time not available						
	296-35581-1-ND UCC27510BVR Texas Instruments IC GATE DRVR LOW-SIDE SOT23-6 <input type="checkbox"/> 43	Customer Reference	2	Backorder	1,34000	2,68 €
Check Lead Time						
	SER4076CT-ND FC-135R 32.7680KF-A3 EPSON CRYSTAL 32.7680HZ 12.5PF SMD <input type="checkbox"/> 44	Customer Reference	1	Backorder	0,79000	0,79 €

Abbildung 70: Beispiel für nicht lieferbare Bauteile (Stand 24.03.2022)

## 4.2 Weitere Schritte

Um das Design vollständig zu verifizieren, sind noch einige Messungen mit mehreren MPPTs an einem Akku nötig. Außerdem steht der Praxistest in einem Solarboot noch aus. Dieser kann erst dann erfolgen, wenn die Bauteile wieder lieferbar sind.

Die Platine, die in dieser Ausarbeitung auf allen Fotos gezeigt wurde, ist eine etwas frühere work-in-progress Version. Bei dieser gab es einige kleinere Mängel, die bereits korrigiert wurden. Aufgrund fehlender Bauteile wurde diese verbesserte Version allerdings nicht aufgebaut.

Es ist denkbar, dass das Design über eine Plattform, wie zum Beispiel GitHub, als Open-Source Projekt der Öffentlichkeit zugänglich gemacht wird. Die endgültige Entscheidung hierüber ist bei Abgabe der Arbeit jedoch noch nicht getroffen worden. Falls die Veröffentlichung erfolgen sollte, so wird das Projekt unter folgendem Link verfügbar sein: Link zum GitHub-Repository.

## 4.3 Endergebnis

Am Ende dieser Arbeit existiert ein zu großen Teilen geprüftes und funktionsfähiges MPPT-Design, das im Solarboot der Hochschule Emden eingesetzt werden könnte. Das Design ist in vielerlei Hinsicht überlegen gegenüber der vorherigen Version (Siehe Abb. 72).

Nicht alle Messungen, die nötig wären, um die volle Funktionsfähigkeit des Designs zu belegen, konnten durchgeführt werden (Siehe 4.1).

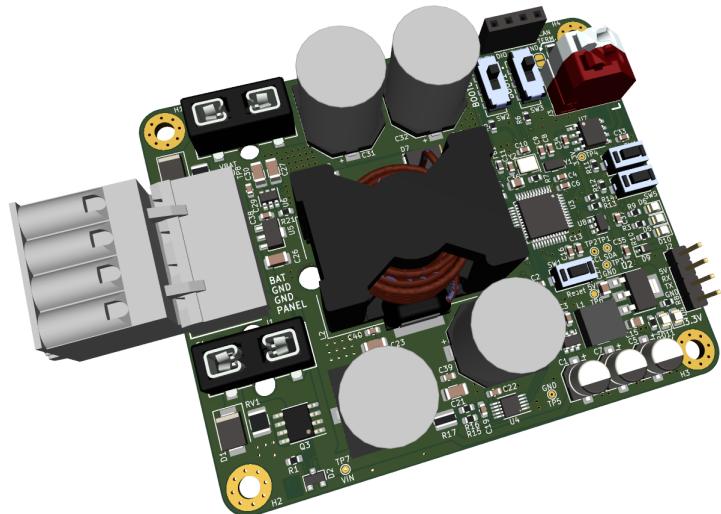


Abbildung 71: 3D Render der Platine

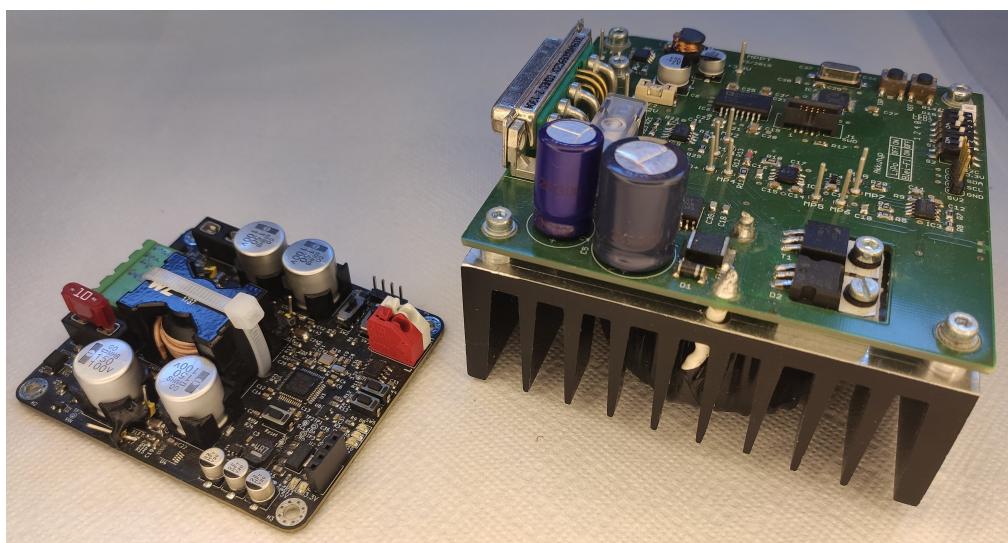


Abbildung 72: Vergleich der MPPTs

## **4.4 Danksagung**

Besonders bedanken möchte ich mich bei meinem Betreuer Prof. Dr.-Ing. Gavin Kane, der mich während meiner Arbeit begleitet hat und mir viele wertvolle Ratschläge gegeben hat.

Außerdem hat die Zusammenarbeit mit dem Solarboot-Team wunderbar funktioniert, nicht zuletzt wegen der großen Motivation und Hilfsbereitschaft der Teammitglieder.

# Literatur

- [1] Steve Arar. *Resistive current sensing: Low-side vs. high-side sensing*. Aug. 2021. URL: <https://www.allaboutcircuits.com/technical-articles/resistive-current-sensing-low-side-versus-high-side-sensing/>.
- [2] Nippon Chemi-Con. *EMHS101ARA151MKG5S Datasheet*. URL: [https://www.mouser.de/datasheet/2/420/NIPC\\_S\\_A0012699746\\_1-2524722.pdf](https://www.mouser.de/datasheet/2/420/NIPC_S_A0012699746_1-2524722.pdf).
- [3] Toshiba Electronic Devices und Storage Corporation. *TPH2R408QM Datasheet*. 2021. URL: <https://toshiba.semicon-storage.com/info/docget.jsp?did=67915%5C&prodName=TPH2R408QM>.
- [4] Er.gauravvats. *Power voltage PV curve*. 2013. URL: [https://en.wikipedia.org/wiki/Maximum\\_power\\_point\\_tracking#/media/File:Power-voltage\\_\(P\\_-V\)\\_curve.png](https://en.wikipedia.org/wiki/Maximum_power_point_tracking#/media/File:Power-voltage_(P_-V)_curve.png).
- [5] Brigitte Hauke. *Basic Calculation of a Boost Converter's Power Stage*. Techn. Ber. Texas Instruments, 2014. URL: <https://www.ti.com/lit/an/slva372c/slva372c.pdf>.
- [6] Texas Instruments. *INA226 Datasheet*. URL: <https://www.ti.com/lit/ds/symlink/ina226.pdf>.
- [7] Texas Instruments. *UCC27531DBVR Datasheet*. URL: <https://www.ti.com/lit/ds/symlink/ucc27531.pdf>.
- [8] ONsemi. *MM3ZxxxT1G Series Datasheet*. 2021. URL: [https://www.mouser.de/datasheet/2/308/1/MM3Z2V4T1\\_D-2316321.pdf](https://www.mouser.de/datasheet/2/308/1/MM3Z2V4T1_D-2316321.pdf).
- [9] Saure. *Maximum Power Transfer Graph*. 11. Sep. 2017. URL: [https://commons.wikimedia.org/wiki/File:Maximum\\_Power\\_Transfer\\_Graph\\_de.png](https://commons.wikimedia.org/wiki/File:Maximum_Power_Transfer_Graph_de.png).
- [10] ON Semi. *SS16HE Datasheet*. URL: <https://www.onsemi.com/pdf/datasheet/ss13he-d.pdf>.
- [11] Tawian Semiconductor. *TSPB20U80S Datasheet*. URL: [https://www.mouser.de/datasheet/2/395/TWSC\\_S\\_A0001020858\\_1-2522484.pdf](https://www.mouser.de/datasheet/2/395/TWSC_S_A0001020858_1-2522484.pdf).
- [12] Mohammadmehdi Seyedmahmoudian u. a. „Maximum power point tracking of partial shaded photovoltaic array using an evolutionary algorithm: A particle swarm optimization technique“. In: *Journal of Renewable and Sustainable Energy* (2014).
- [13] SolarSportOne. *Technical Regulations 2020 version*. 2020. URL: <https://solarsportone.org/2018/wp-content/uploads/2019/10/2020-Technical-Regulations-9-10-19.pdf>.
- [14] STMicroelectronics. *STM32F103C8 Datasheet*. URL: <https://www.st.com/resource/en/datasheet/stm32f103c8.pdf>.
- [15] STMicroelectronics. *STM32F103C8 Overview Page*. URL: [https://www.st.com/content/st\\_com/en/products/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus/stm32-mainstream-mcus/stm32f1-series/stm32f103/stm32f103c8.html#overview](https://www.st.com/content/st_com/en/products/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus/stm32-mainstream-mcus/stm32f1-series/stm32f103/stm32f103c8.html#overview).

## 5 Anhang

### 5.1 Bildverzeichnis

#### Abbildungsverzeichnis

1	Frontansicht des neu-entwickelten MPPT . . . . .	5
2	Frontansicht bisheriger MPPT . . . . .	8
3	Unterseite bisheriger MPPT . . . . .	9
4	Seitenansicht bisheriger MPPT . . . . .	9
5	Vergleich zwischen dem neuen MPPT (links) und dem alten MPPT (rechts): Frontansicht . . . . .	10
6	Vergleich zwischen dem neuen MPPT (links) und dem alten MPPT (rechts): Seitenansicht . . . . .	11
7	Blöckschaltbild des MPPT . . . . .	13
8	Übersicht Schaltplan . . . . .	15
9	Anschlüsse am Mikrocontroller . . . . .	16
10	Unterstützende Bauelemente für den Mikrocontroller . . . . .	18
11	5V Tiefsetzsteller . . . . .	19
12	3.3V Linearregler und Filter . . . . .	20
13	Simulation: Verhalten des Filters . . . . .	21
14	SWD-Programmieran schluss und UART . . . . .	22
15	LEDs und Taster . . . . .	23
16	Testpunkte und Pull-Up Widerstände . . . . .	23
17	externer EEPROM-IC . . . . .	24
18	CAN Bus Transceiver . . . . .	24
19	Schaltplan Hochsetzsteller . . . . .	25
20	$R_{DS(ON)}$ des MOSFETs [3] . . . . .	28
21	$U_F$ der Diode in Abhängigkeit von Strom und Temperatur [11] . . . . .	29
22	Gate-Treiber für den Hochsetzsteller . . . . .	31
23	INA226 Leistungssensor . . . . .	32
24	Spannungsteiler zur Messung der Ausgangsspannung . . . . .	33
25	Leckstrom der Schottkydioden D12 und D13 [10] . . . . .	34
26	Leckstrom der Zenerdiode MM3Z3V3T1G [8] . . . . .	34
27	Eingangsschutzbeschaltung . . . . .	36
28	Ausgangsschutzbeschaltung . . . . .	36
29	Montagelöcher . . . . .	36
30	Oberste Kupferlage ohne Ground-Fill . . . . .	37
31	Oberste Kupferlage mit Ground-Fill . . . . .	38
32	Unterste Kupferlage ohne Ground-Fill . . . . .	38
33	Interne Kupferlage 1 mit Ground-Fill . . . . .	39
34	Interne Kupferlage 2 mit Ground-Fill . . . . .	39
35	Oberseite der bestückten Platine . . . . .	40
36	Unterseite der Platine . . . . .	40
37	Oberseite der unbestückten Platine . . . . .	41
38	Ein- und Ausgangsschutzschaltung markiert . . . . .	42

39	Hochsetzsteller markiert . . . . .	43
40	Layout des Leistungssensors . . . . .	43
41	Layout des Leistungsteils . . . . .	44
42	Layout des Gate-Treibers . . . . .	45
43	Spannungsversorgung des Mikrocontrollers markiert . . . . .	46
44	Layout der Spannungsversorgung . . . . .	47
45	Mikrocontroller und dazugehörige Bauteile markiert . . . . .	48
46	Layout des Spannungsteilers und der Oszillatoren . . . . .	49
47	Layout CAN-Bus . . . . .	50
48	Platine mit aufgebrachter Lötpaste . . . . .	51
49	Platine mit aufgebrachter Lötpaste und Bauteilen . . . . .	51
50	Ungewollte Lötbrücke mit Markierung . . . . .	52
51	IDE PlatformIO . . . . .	53
52	Applikation GitHub Desktop für die Versionsverwaltung . . . . .	54
53	Aufbau während der Softwareentwicklung . . . . .	55
54	Maximum Power Point versch. Solarpanel [4] . . . . .	56
55	Aufrufgraph setup() . . . . .	58
56	Aufrufgraph mppt_algo() . . . . .	59
57	Maximum Power Point eines Solarpanels mit Markierungen [4] . . . . .	60
58	Entscheidungsbaum des Preturb-and-Observe Algorithmus . . . . .	61
59	Messaufbau . . . . .	63
60	Sprungantwort Ausgangslast . . . . .	64
61	Sprungantwort fallende Eingangsspannung . . . . .	64
62	Sprungantwort fallende Eingangsspannung . . . . .	65
63	Messaufbau: Verifikation des MPPT-Algorithmus . . . . .	66
64	Leistungsanpassung: Spannungsquelle mit Innenwiderstand [9] . . . . .	67
65	Belastungstest: Leistungsbegrenzung . . . . .	69
66	Wärmebildaufnahme ohne Markierungen . . . . .	70
67	Wärmebildaufnahme mit Markierungen . . . . .	70
68	Vergleich der Wirkungsgrade bei verschiedenen Eingangsspannungen .	73
69	Vergleich der Wirkungsgrade bei verschiedenen Eingangsspannungen, verbesserte Auflösung . . . . .	73
70	Beispiel für nicht lieferbare Bauteile (Stand 24.03.2022) . . . . .	75
71	3D Render der Platine . . . . .	76
72	Vergleich der MPPTs . . . . .	77

## 5.2 Quellcode

### 5.2.1 mppt\_main.cpp

```

1 #include <Arduino.h>
2 #include <Wire.h>
3
4 #include <INA.h>
5
6 #include "mppt_pins.h"
7 #include "mppt_config.h"
8 #include "ina_sensor.h"

```

```

9 #include "mppt_io.h"
10 #include "mppt_control_algo.h"
11 #include "mppt_main.h"
12
13 INA_Class INA(1); //INA class instantiation to use RAM, one
14     instance
15
16 void setup()
17 {
18     static volatile unsigned long mppt_algo_timer = 0;
19     mppt_io_init();
20     mppt_init_boost_pwm();
21     analogReadResolution(MPPT_ADC_RESOLUTION);
22
23     //Serial port
24     if (ENABLE_SERIAL_DEBUG)
25     {
26         DEBUG_SERIAL.begin(DEBUG_SERIAL_BAUD);
27         delay(1000);
28         DEBUG_SERIAL.println(F("MPPT started."));
29     }
30
31     //INA226 power sensor
32     ina_begin(INA);
33
34     //Main program loop
35     DEBUG_SERIAL.println(F("Starting MPPT algorithm."));
36     mppt_boost_enable();
37     while (true)
38     {
39         if (millis() - mppt_algo_timer >= MPPT_ALGORITHM_DELAY) {
40             mppt_algo(INA, 1);
41             mppt_algo_timer = millis();
42             //Serial.println(mppt_algo_timer);
43         }
44         else {
45             mppt_algo(INA, 0);
46         }
47     }
48 }
49
50
51 //void loop() is no used since it is slower than a simple while(
52     true)
53 void loop()
54 {

```

### 5.2.2 mppt\_config.h

```
1  /*
2 Configuration file
3 */
4 #include <Arduino.h>
5
6 #ifndef __MPPT_CONFIG
7 #define __MPPT_CONFIG
8
9 //Debug serial (UART)
10#define ENABLE_SERIAL_DEBUG true
11#define DEBUG_SERIAL Serial
12#define DEBUG_SERIAL_BAUD 115200
13
14//CAN Bus functionality
15//(not implemented)
16#define ENABLE_CAN_BUS false
17
18//MPPT related settings
19#define MPPT_MAX_OUT_VOLTAGE 50 //MAXIMUM battery
20// voltage / MPPT output (default 50.2)
21#define MPPT_MIN_IN_VOLTAGE 10 //MINIMUM panel / input
22// voltage
23#define MPPT_ALGORITHM_DELAY 1 //How often the perturb
24// and observe algorithm gets executed, value is the time (in ms)
25// before the algorithm gets called again => frequency = 1/
26// MPPT_ALGORITHM_DELAY
27
28//Boost converter
29#define MPPT_BOOST_MAX_PWM_VAL 1023
30#define MPPT_BOOST_MIN_PWM_VAL 0
31#define MPPT_BOOST_PWM_RES RESOLUTION_10B_COMPARE_FORMAT
32// "resolution" for PWM
33#define MPPT_BOOST_PWM_STEP_SIZE 5
34
35//ADC accuracy / bits for output voltage measurement
36#define MPPT_ADC_RESOLUTION 12
37const int MPPT_ADC_MAX_READING = pow(2, MPPT_ADC_RESOLUTION)
38- 1;
39const float MPPT_ADC_REF_VOLTAGE = 3.3;
40#define MPPT_ADC_READING_AVG_COUNT 1
41
42//MPPT output voltage divider resistor values
43const unsigned int MPPT_OUT_VOLTAGE_R_UPPER = 20000; //Upper
44// resistor in the divider
45const unsigned int MPPT_OUT_VOLTAGE_R_LOWER = 1000; //Lower
46// resistor in the divider, the ADC measures across this resistor
47const unsigned int MPPT_OUT_VOLTAGE_R_SUM =
48// MPPT_OUT_VOLTAGE_R_LOWER + MPPT_OUT_VOLTAGE_R_UPPER; //Sum
49// of both, makes calculation easier
50
51const float MPPT_OUT_VOLTAGE_DIVIDER_FACTOR = (float)
52// MPPT_OUT_VOLTAGE_R_SUM / (float)MPPT_OUT_VOLTAGE_R_LOWER;
53
54//INA226 current sensor settings
```

```
43 #define MPPT_INA_SHUNT          10000
44
45
46 #endif
```

### 5.2.3 mppt\_pins.h

```
1 /*  
2 Pin definitions  
3 */  
4  
5 #ifndef __MPPT_PINS  
6 #define __MPPT_PINS  
7  
8 //LEDs for status indication etc.  
9 #define LED1_PIN PB_14 //Blue LED  
10 #define LED2_PIN PB_15 //Green LED  
11 #define LED3_PIN PA_8 //Red LED  
12 #define LED_BLUE_PIN LED1_PIN  
13 #define LED_GREEN_PIN LED2_PIN  
14 #define LED_RED_PIN LED3_PIN  
15  
16 //Buttons  
17 #define BTN1_PIN PB_3 //Top (SW4)  
18 #define BTN2_PIN PB_4 //Bottom (SW5)  
19  
20 //CAN Bus Pins  
21 #define CAN_RX_PIN PB_8  
22 #define CAN_TX_PIN PB_9  
23 #define CAN_STBY_PIN PB_5  
24 #define CAN_STBY_ON 1  
25 #define CAN_STBY_OFF 0  
26  
27 //Boost control pins  
28 #define BOOST_DRIVE_PIN PA_7  
29 #define BOOST_EN_PIN PA_6  
30 #define BOOST_EN_ON 1  
31 #define BOOST_EN_OFF 0  
32  
33 //Output voltage sensing  
34 #define V_SENSE_OUT_PIN PA_3  
35  
36 //INA226 signals  
37 #define ALERT_ISENS_PIN PB_13 //Overcurrent alert input to  
38 STM32  
39 #define SDA_1 PB_7  
40 #define SCL_1 PB_6  
41  
42  
43 #endif
```

#### 5.2.4 ina\_sensor.h

```
1 /*  
2 Declarations for the INA226 power sensor  
3 */  
4  
5 #ifndef __INA_SENSOR  
6 #define __INA_SENSOR  
7  
8 #include <INA.h> //Library  
9 #include "mppt_config.h"  
10  
11 //INA config  
12 const uint32_t SHUNT_MICRO_OHM = MPPT_INA_SHUNT;           // < Shunt  
    resistance in Micro-Ohm, e.g. 100000 is 0.1 Ohm  
13 const uint16_t MAXIMUM_AMPS = 10;                            // < Max  
    expected amps, clamped from 1A to a max of 1022A  
14 const uint32_t BUS_CONVERSION = 148;                         // pre-set:  
    8500  
15 const uint32_t SHUNT_CONVERSION = 148;                       // pre-set:  
    8500  
16 const uint16_t INA_MODE = INA_MODE_CONTINUOUS_BOTH;  
17 const uint32_t ALERT_VOLTAGE = 5000;                          //  
    Millivolts!  
18 const uint8_t AVERAGING_TIMES = 4;                           // pre-set:  
    128  
19  
20 //Function declarations  
21  
22 //Initialize sensor  
23 void ina_begin(INA_Class &INA);  
24  
25 //Processes an alert signal from the INA  
26 volatile void ina_process_alert();  
27  
28 //Read current input voltage and power  
29 void ina_read_voltage_and_power(INA_Class &INA, volatile float &  
    in_voltage, volatile float &in_pwr);  
30  
31 //Read current input voltage  
32 void ina_read_voltage(INA_Class &INA, volatile float &in_voltage);  
33  
34 //Read current input power  
35 void ina_read_power(INA_Class &INA, volatile float &in_pwr);  
36  
37 #endif
```

### 5.2.5 ina\_sensor.cpp

```
1  /*
2  Code for the INA226 power sensor
3  */
4
5 #include <INA.h>
6
7 #include "ina_sensor.h"
8 #include "mppt_pins.h"
9
10 void ina_begin(INA_Class &INA)
11 {
12     //Initialize INA sensor
13     uint8_t devicesFound = INA.begin(MAXIMUM_AMPS, SHUNT_MICRO_OHM);
14     ;
15     while (devicesFound == 0)
16     {
17         Serial.println(F("No INA device found, retrying in 1 second
..."));
18         delay(1000);                                // Wait 1 second before retrying
19         devicesFound = INA.begin(MAXIMUM_AMPS, SHUNT_MICRO_OHM); // Expected max Amp & shunt resistance
20     }
21     Serial.println(F("Found INA226 sensor."));
```

22 //Set conversion values and warning threshold  
23 INA.setBusConversion(BUS\_CONVERSION); // Maximum  
conversion time 8.244ms (8500)  
24 INA.setShuntConversion(SHUNT\_CONVERSION); // Maximum  
conversion time 8.244ms (8500)  
25 INA.setAveraging(AVERAGING\_TIMES); // Average  
each reading n-times  
26 INA.setMode(INA\_MODE); // Bus/shunt  
measured continuously  
27 INA.alertOnBusOverVoltage(true, ALERT\_VOLTAGE); // Trigger  
alert if over ALERT\_VOLTAGE on bus

```
28 }
29
30 volatile void ina_process_alert()
31 {
32     //Serial.println(F("Alert"));
33     digitalToggleFast(LED_RED_PIN);
34 }
```

35 //Read current input voltage and power  
36 void ina\_read\_voltage\_and\_power(INA\_Class &INA, volatile float &  
in\_voltage, volatile float &in\_pwr)

```
37 {
38     in_pwr = INA.getBusMicroWatts() / 1000.0;
39     in_voltage = INA.getBusMilliVolts() / 1000.0;
40 }
41
42 //Read current input voltage  
43 void ina_read_voltage(INA_Class &INA, volatile float &in_voltage)
```

```
45 {
46     in_voltage = INA.getBusMilliVolts() / 1000.0;
47 }
48
49 //Read current input power
50 void ina_read_power(INA_Class &INA, volatile float &in_pwr)
51 {
52     in_pwr = INA.getBusMicroWatts() / 1000.0;
53 }
```

### 5.2.6 mppt\_control\_algo.h

```
1 /*  
2 MPPT algorithm  
3 */  
4  
5 #ifndef __MPPT_CONTROL_ALGO  
6 #define __MPPT_CONTROL_ALGO  
7  
8 #include "mppt_pins.h"  
9 #include "ina_sensor.h"  
10 #include "mppt_io.h"  
11  
12  
13  
14 //Main MPPT control function  
15 void mppt_algo(INA_Class &INA, bool run_preturb);  
16  
17 #endif
```

### 5.2.7 mppt\_control\_algo.cpp

```
1  /*
2  MPPT algorithm
3  */
4
5 #include "mppt_control_algo.h"
6 #include "mppt_pins.h"
7 #include "ina_sensor.h"
8 #include "mppt_io.h"
9 #include "mppt_main.h"
10
11 #define MAX(a, b) (((a) > (b)) ? (a) : (b)) //Determine maximum of
12     a and b
13
14 /*extern*/ volatile float in_voltage, in_power, out_voltage;
15 volatile float in_power_1, in_power_2, in_power_initial;
16
17 /*
18 This algorithm is pretty simple.
19 It is based the preturb and observe algorithm to track the
20     maximum power point
21 Input power gets raised by raising the output voltage, respecting
22     the maximum allowed output voltage.
23
24 potential issues:
25     p&o gets skipped if one of the previous conditions is true ->
26         implement return or ignore?
27 */
28 void mppt_algo(INA_Class &INA, bool run_preturb)
29 {
30     //Reference values for previous power readings (last preturb
31     and observe cycle)
32     //These get compared to the new values and based on the result
33     //of the comparision the pulse width gets adjusted
34     static volatile bool mppt_pwm_change_direction = 0; //0-> PWM
35     was decreased; 1-> PWM was increased
36     static volatile float mppt_prev_in_power = 0;           //Measured
37     input power of the last p&o run
38
39     //Measure current values
40     mppt_io_read_output_voltage(MPPT_ADC_READING_AVG_COUNT,
41     out_voltage);
42     ina_read_voltage(INA, in_voltage);
43
44     //Check that the maximum output voltage is not exceeded
45     if (out_voltage > MPPT_MAX_OUT_VOLTAGE)
46     {
47         mppt_dec_boost_pwm();
48         mppt_leds_state(0);
49
50         //Serial.println(F("OUT_V"));
51     }
52     //Check that the panel voltage is not under the allowed minimum
53     else if (in_voltage < MPPT_MIN_IN_VOLTAGE)
54     {
```

```

46     mppt_dec_boost_pwm();
47     mppt_leds_state(0);
48
49     //Serial.println(F("PAN_V"));
50 }
51
52 //MPPT algorithm (preturb and observe)
53 //Runs only with the set frequency (mppt_config.h)
54 //The caller specifies if this should run by setting
55 //run_preturb to true
56 else if (run_preturb)
{
57     //Read the current input power
58     ina_read_power(INA, in_power);

59     //The pulse width was increased in the previous run
60     if (mppt_pwm_change_direction)
61     {
62         //Check if the input power has increased as well
63         //If the power has increased, increase the pulse width
64         further
65         if (in_power > mppt_prev_in_power)
66         {
67             mppt_pwm_change_direction = 1;
68             mppt_inc_boost_pwm();
69         }
70         //If the input power has decreased, start to decrease
71         //the pulse width
72         else if (in_power < mppt_prev_in_power)
73         {
74             mppt_pwm_change_direction = 0;
75             mppt_dec_boost_pwm();
76         }
77
78         //The pulse width was decreased in the previous run
79     else
80     {
81         //Check if the input power has increased
82         //If the power has increased, decrease the pulse width
83         further
84         if (in_power > mppt_prev_in_power)
85         {
86             mppt_pwm_change_direction = 0;
87             mppt_dec_boost_pwm();
88         }
89         //If the input power has decreased, start to increase
90         //the pulse width
91         else if (in_power < mppt_prev_in_power)
92         {
93             mppt_pwm_change_direction = 1;
94             mppt_inc_boost_pwm();
95         }
96     }
97
98     //Set reference for the next run

```

```
97     mppt_prev_in_power = in_power;
98 }
99 ///////////////
100
101 //Last resort fail safe; output voltage is way too high
102 if (out_voltage > (MPPT_MAX_OUT_VOLTAGE + 5))
103 {
104     Serial.println(F("!FAIL!"));
105     while (out_voltage > MPPT_MAX_OUT_VOLTAGE)
106     {
107         mppt_dec_boost_pwm();
108         mppt_leds_state(255);
109         mppt_io_read_output_voltage(MPPT_ADC_READING_AVG_COUNT,
110             out_voltage);
111     }
112 }
```

### 5.2.8 mppt\_io.h

```
1 /*  
2 Declarations for IO related functions  
3 */  
4  
5 #ifndef __MPPT_IO  
6 #define __MPPT_IO  
7  
8 #include "mppt_pins.h"  
9  
10 //Sets all IO to the specified function / state  
11 void mppt_io_init();  
12  
13 //Read the output voltage of the boost converter  
14 void mppt_io_read_output_voltage(int avg_count, volatile float &  
    voltage);  
15  
16  
17 //Increase PWM value  
18 void mppt_inc_boost_pwm();  
19 //Decrease PWM value  
20 void mppt_dec_boost_pwm();  
21  
22 //Initialize PWM for the boost converter  
23 void mppt_init_boost_pwm();  
24 // "Arm" / "disarm" the gate driver for the boost converter; PWM  
    signal get's passed on to the mosfet's gate  
25 void mppt_boost_enable();  
26 void mppt_boost_disable();  
27  
28 //Set the PWM value  
29 void mppt_set_boost_pwm(volatile int &pwm_val);  
30  
31 //LEDs  
32 void mppt_leds_state(byte state);  
33  
34 #endif
```

### 5.2.9 mppt\_io.cpp

```
1  /*
2  IO related functions
3 */
4 #include <Arduino.h>
5
6 #include "mppt_pins.h"
7 #include "mppt_io.h"
8 #include "ina_sensor.h"
9 #include "mppt_config.h"
10 #include "mppt_main.h"
11
12 // Automatically retrieve TIM instance and channel associated to
13 // pin
14 // This is used to be compatible with all STM32 series
15 // automatically.
16 TIM_TypeDef *Instance = (TIM_TypeDef *)pinmap_peripheral(
17     digitalPinToPinName(BOOST_DRIVE_PIN), PinMap_PWM);
18 uint32_t channel = STM_PIN_CHANNEL(pinmap_function(
19     digitalPinToPinName(BOOST_DRIVE_PIN), PinMap_PWM));
20
21 // Instantiate HardwareTimer object. Thanks to 'new' instantiation,
22 // HardwareTimer is not destructed when setup function is finished
23 .
24 HardwareTimer *MyTim = new HardwareTimer(Instance);
25
26 /*extern*/ volatile int boost_pwm_val;
27
28 //Sets all IO to the specified function / state
29 void mppt_io_init()
30 {
31     //Inputs
32     pinMode(BTN1_PIN, INPUT_PULLUP);
33     pinMode(BTN2_PIN, INPUT_PULLUP);
34     pinMode(V_SENSE_OUT_PIN, INPUT); //INPUT_ANALOG?
35
36     //Input with interrupt
37     pinMode(ALERT_ISENS_PIN, INPUT_PULLUP);
38     attachInterrupt(ALERT_ISENS_PIN, ina_process_alert, FALLING);
39
40     //Outputs
41     pinMode(LED1_PIN, OUTPUT);
42     pinMode(LED2_PIN, OUTPUT);
43     pinMode(LED3_PIN, OUTPUT);
44
45     pinMode(CAN_STBY_PIN, OUTPUT);
46     digitalWriteFast(CAN_STBY_PIN, CAN_STBY_ON); //Set CAN
47     transceiver to standby
48
49     pinMode(BOOST_EN_PIN, OUTPUT);
50     digitalWriteFast(BOOST_EN_PIN, BOOST_EN_OFF); //Turn gate
51     driver off
52
53     //PinMode not needed, if it gets configured by using the PWM
54     functionality
```

```

46     pinMode(BOOST_DRIVE_PIN, OUTPUT);
47     digitalWriteFast(BOOST_DRIVE_PIN, 0); //Turn gate signal off
48 }
49
50 //Read the output voltage of the boost converter
51 void mppt_io_read_output_voltage(int avg_count, volatile float &
52     voltage)
53 {
54     unsigned int adc_readings = 0;
55     for (int i = 0; i < avg_count; i++)
56     {
57         adc_readings += analogRead(V_SENSE_OUT_PIN);
58     }
59     voltage = (adc_readings / avg_count) *
60     MPPT_OUT_VOLTAGE_DIVIDER_FACTOR * ((float)MPPT_ADC_REF_VOLTAGE /
61     (float)MPPT_ADC_MAX_READING);
62 }
63
64 // "Arm" / "disarm" the gate driver for the boost converter; PWM
65 // signal get's passed on to the mosfet's gate
66 void mppt_boost_enable()
67 {
68     digitalWriteFast(BOOST_EN_PIN, BOOST_EN_ON);
69 }
70 void mppt_boost_disable()
71 {
72     digitalWriteFast(BOOST_EN_PIN, BOOST_EN_OFF);
73 }
74
75 //Initialize PWM for the boost converter
76 void mppt_init_boost_pwm()
77 {
78     MyTim->setMode(channel, TIMER_OUTPUT_COMPARE_PWM1,
79     BOOST_DRIVE_PIN); //PWM mode setting: hardware PWM channel, PWM
80     mode, PWM pin
81     // MyTim->setPrescaleFactor(8); // Due to setOverflow with
82     MICROSEC_FORMAT, prescaler will be computed automatically based
83     on timer input clock
84     MyTim->setOverflow(10, MICROSEC_FORMAT); // This sets the PWM "frequency" in microseconds!! -> 1/setting =
85     pwm_freq; setting of 10: 100kHz
86     MyTim->setCaptureCompare(channel, 0, MPPT_BOOST_PWM_RES); // Initial value 0 (PWM OFF)
87
88     MyTim->resume(); //Start PWM
89 }
90
91 //Increase PWM value
92 void mppt_inc_boost_pwm()
93 {
94     if (boost_pwm_val < MPPT_BOOST_MAX_PWM_VAL)
95     {
96         boost_pwm_val += MPPT_BOOST_PWM_STEP_SIZE;
97         mppt_set_boost_pwm(boost_pwm_val);
98     }

```

```

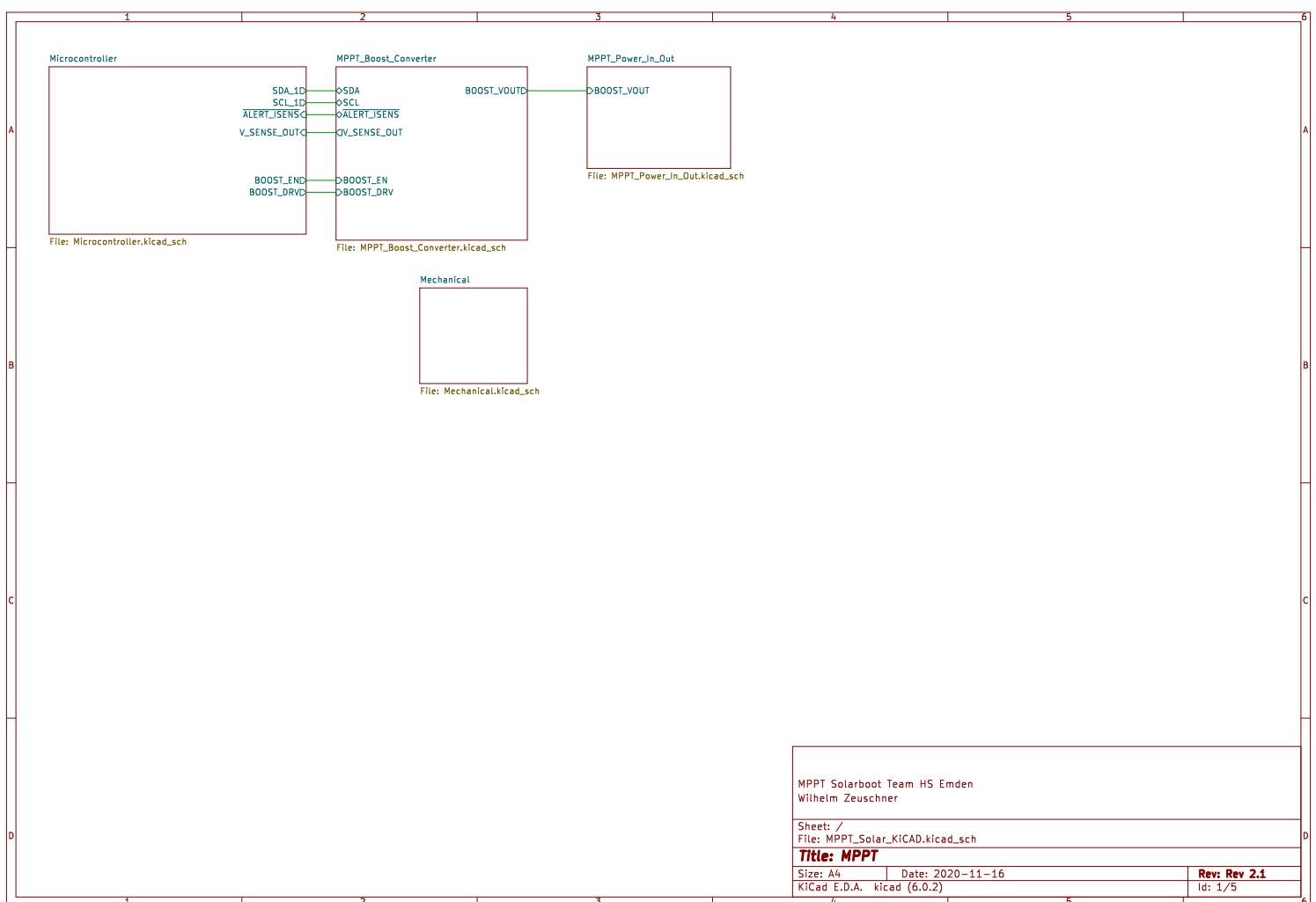
91 }
92
93 //Decrease PWM value
94 void mppt_dec_boost_pwm()
95 {
96     if (boost_pwm_val > MPPT_BOOST_MIN_PWM_VAL)
97     {
98         boost_pwm_val -= MPPT_BOOST_PWM_STEP_SIZE;
99         mppt_set_boost_pwm(boost_pwm_val);
100    }
101 }
102
103 //Set the PWM value
104 void mppt_set_boost_pwm(volatile int &pwm_val)
105 {
106     MyTim->setCaptureCompare(channel, pwm_val, MPPT_BOOST_PWM_RES);
107 }
108
109 //LEDs
110 void mppt_leds_state(byte state)
111 {
112     switch (state)
113     {
114         case 0: //ERROR
115             digitalWriteFast(LED_RED_PIN, 1);
116             digitalWriteFast(LED_GREEN_PIN, 0);
117             digitalWriteFast(LED_BLUE_PIN, 0);
118             break;
119         case 1: //PWM value gets decreased
120             digitalWriteFast(LED_RED_PIN, 1);
121             digitalWriteFast(LED_GREEN_PIN, 0);
122             digitalWriteFast(LED_BLUE_PIN, 0);
123             break;
124         case 2: //PWM value gets increased by MPPT algo
125             digitalWriteFast(LED_RED_PIN, 0);
126             digitalWriteFast(LED_GREEN_PIN, 1);
127             digitalWriteFast(LED_BLUE_PIN, 0);
128             break;
129         case 3: //PWM value gets decreased by MPPT algo
130             digitalWriteFast(LED_RED_PIN, 0);
131             digitalWriteFast(LED_GREEN_PIN, 0);
132             digitalWriteFast(LED_BLUE_PIN, 1);
133             break;
134         default: //All LEDs off if invalid value
135             digitalWriteFast(LED_RED_PIN, 0);
136             digitalWriteFast(LED_GREEN_PIN, 0);
137             digitalWriteFast(LED_BLUE_PIN, 0);
138             break;
139     }
140 }

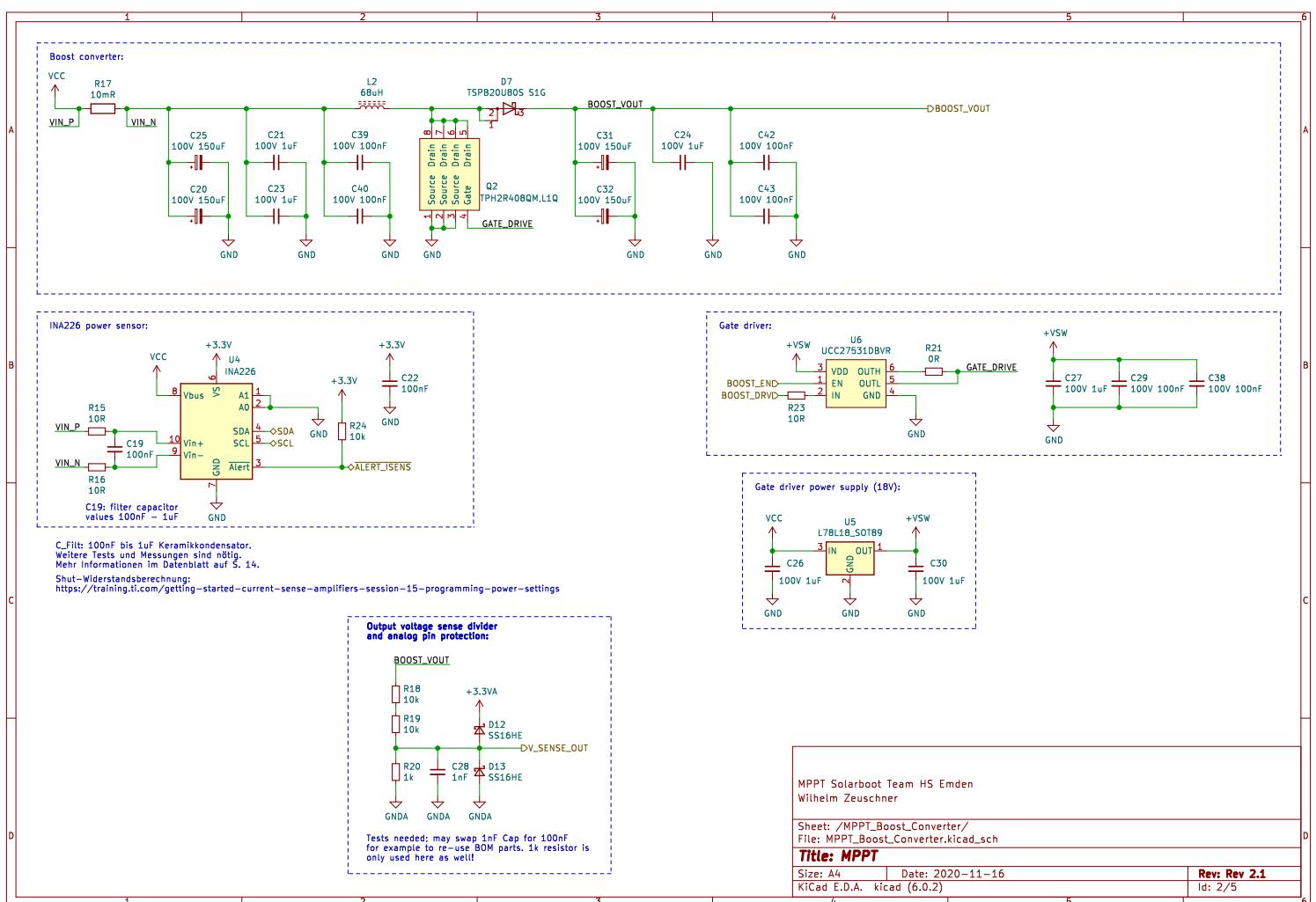
```

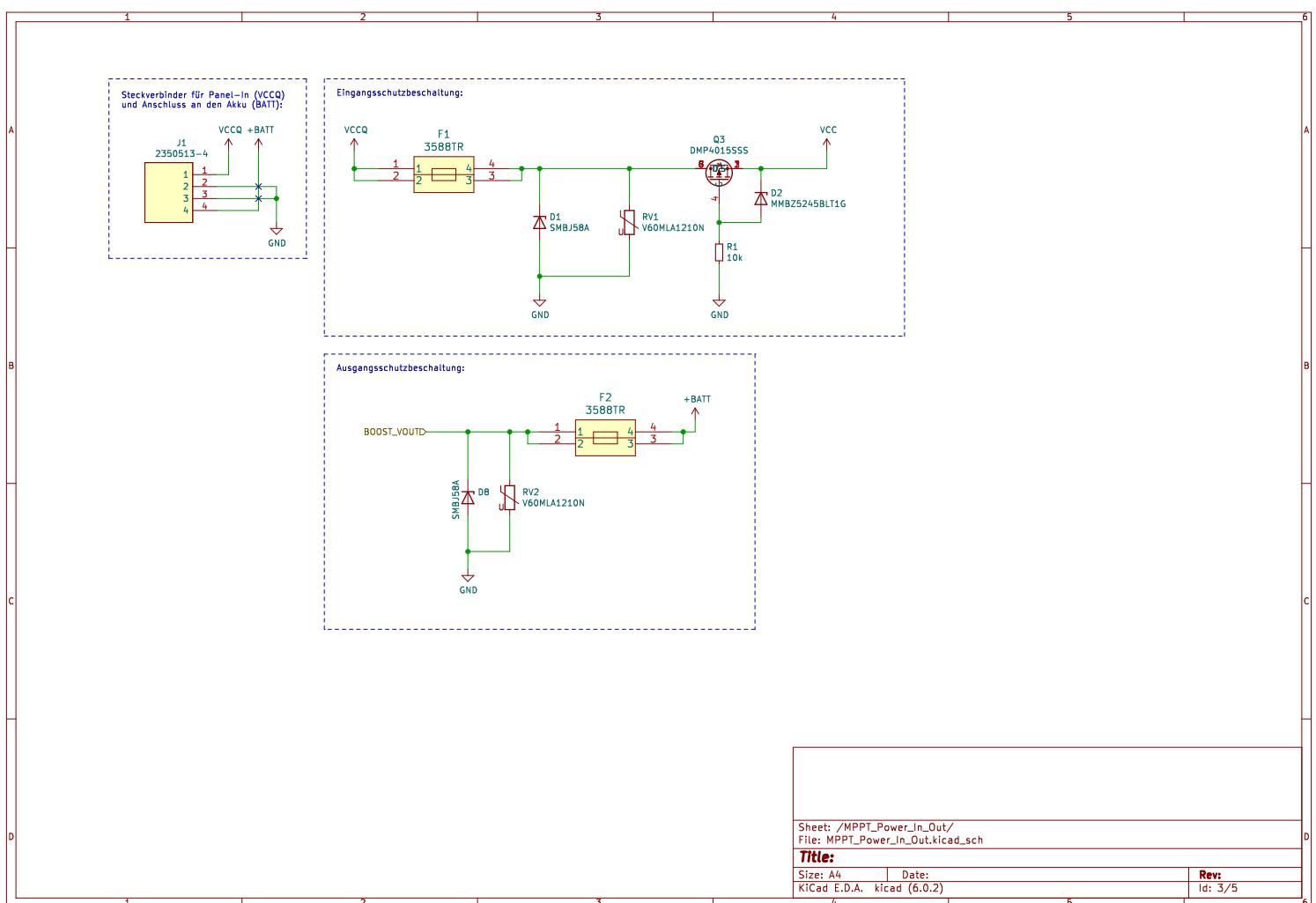
### 5.2.10 platformio.ini

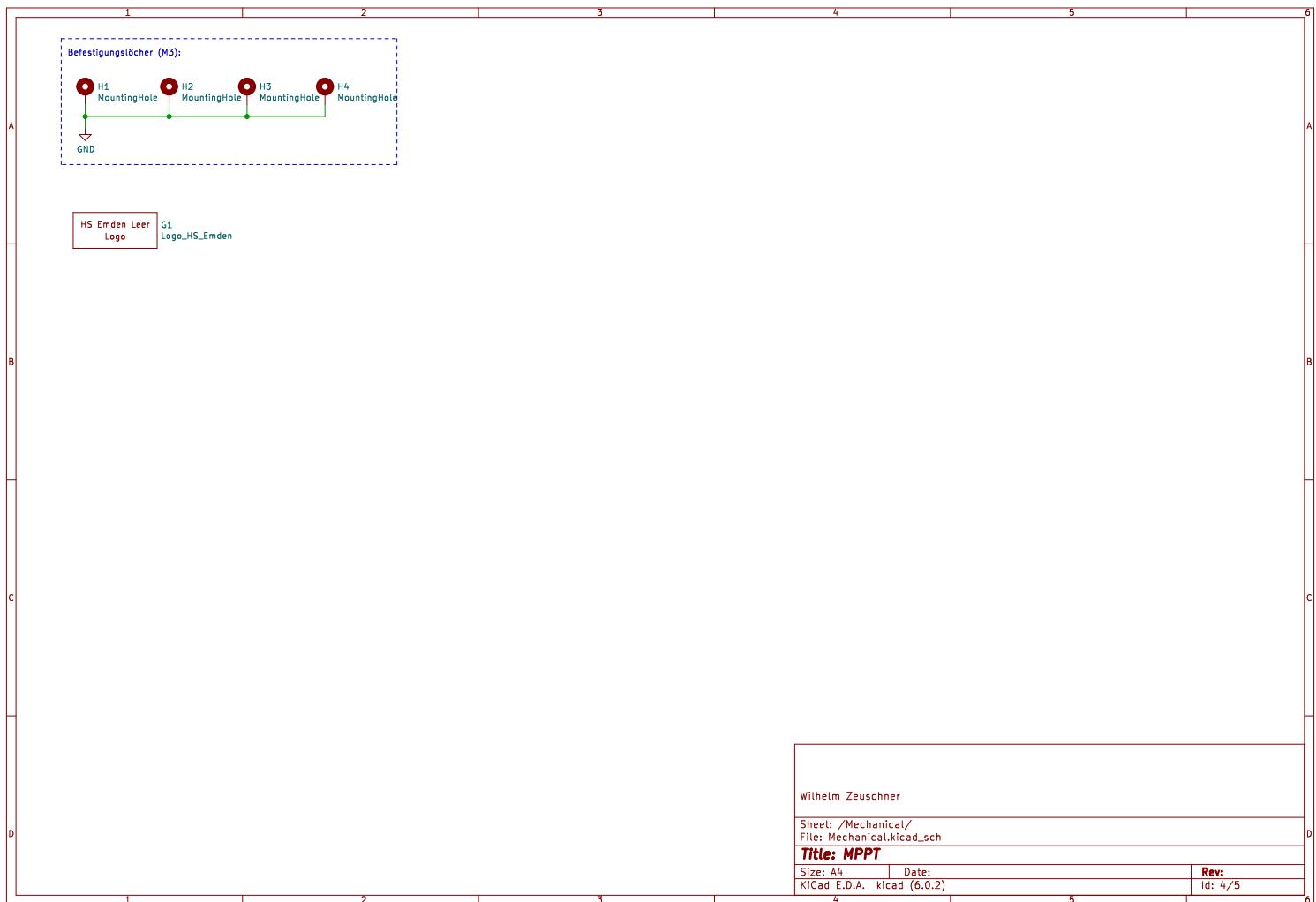
```
1 ; PlatformIO Project Configuration File
2 ;
3 ;     Build options: build flags, source filter
4 ;     Upload options: custom upload port, speed and extra flags
5 ;     Library options: dependencies, extra library storages
6 ;     Advanced options: extra scripting
7 ;
8 ; Please visit documentation for the other options and examples
9 ; https://docs.platformio.org/page/projectconf.html
10
11 [env:genericSTM32F103CB]
12 platform = ststm32
13 board = genericSTM32F103CB
14 framework = arduino
15 monitor_speed = 115200
16
17
18 lib_deps =
19
20     # RECOMMENDED
21     # Accept new functionality in a backwards compatible manner
22     # and patches
23     sv-zanshin/INA2xx @ ^1.1.0 #https://github.com/Zanduino/INA2xx
```

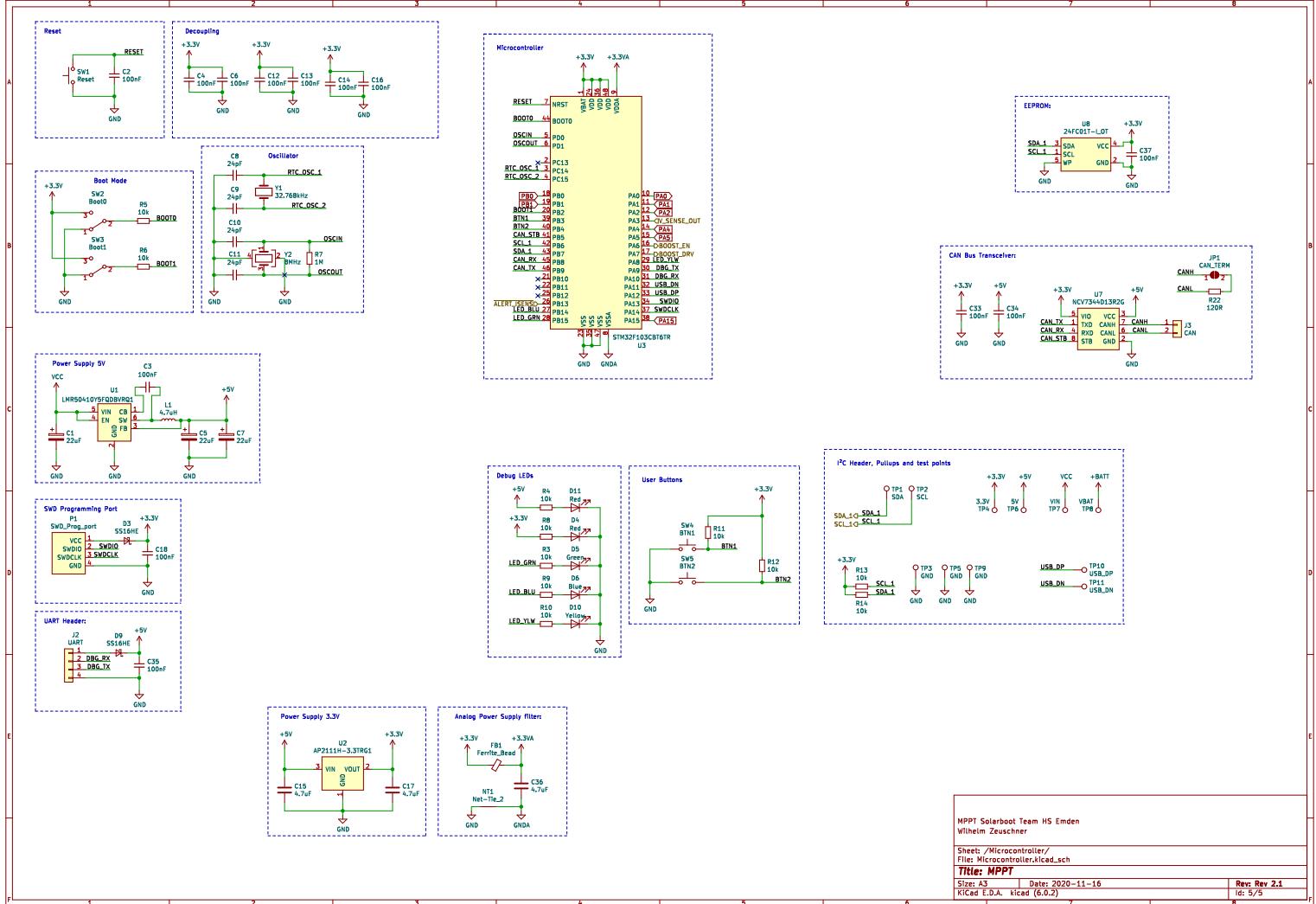
### **5.3 Schaltplan**



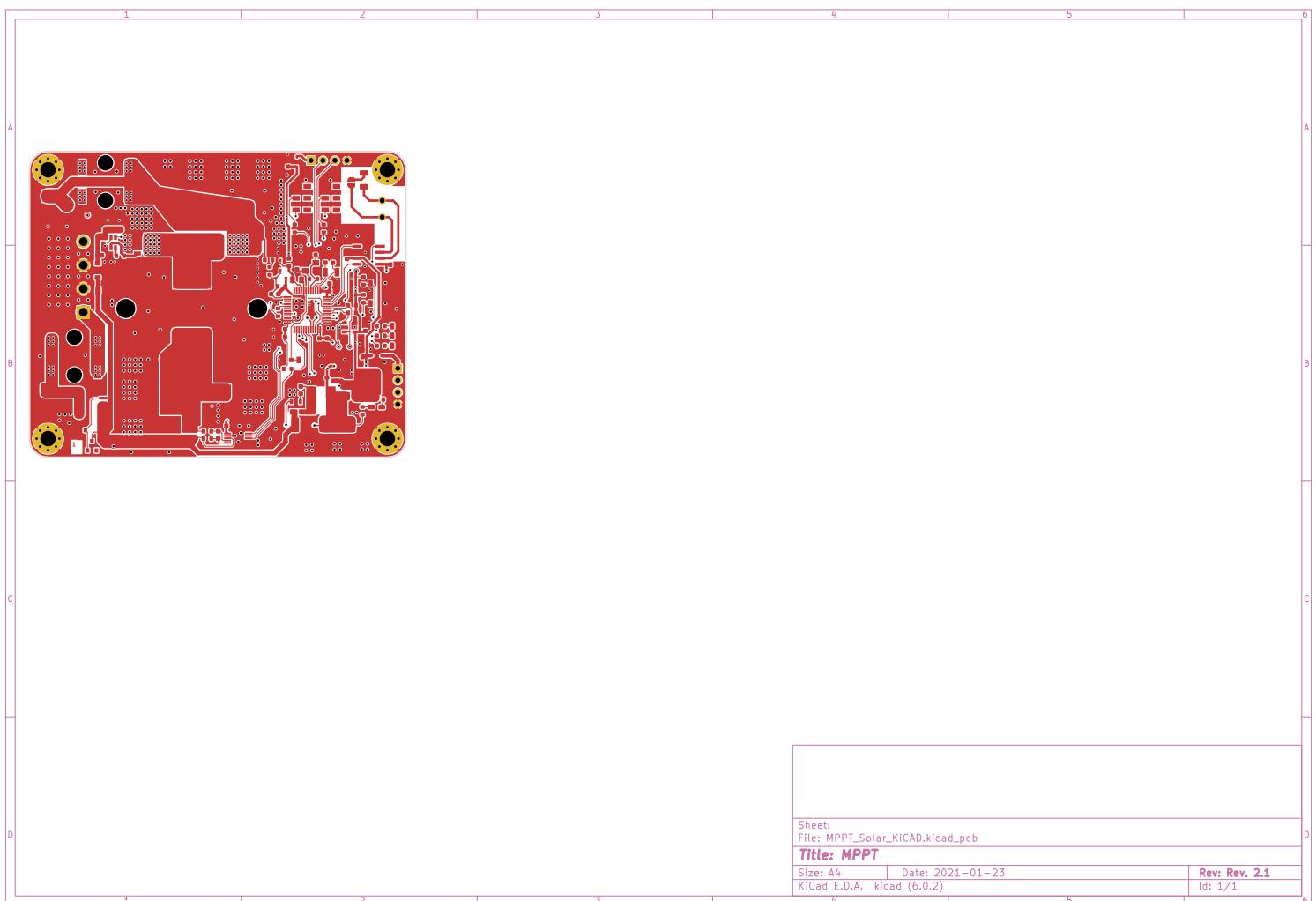


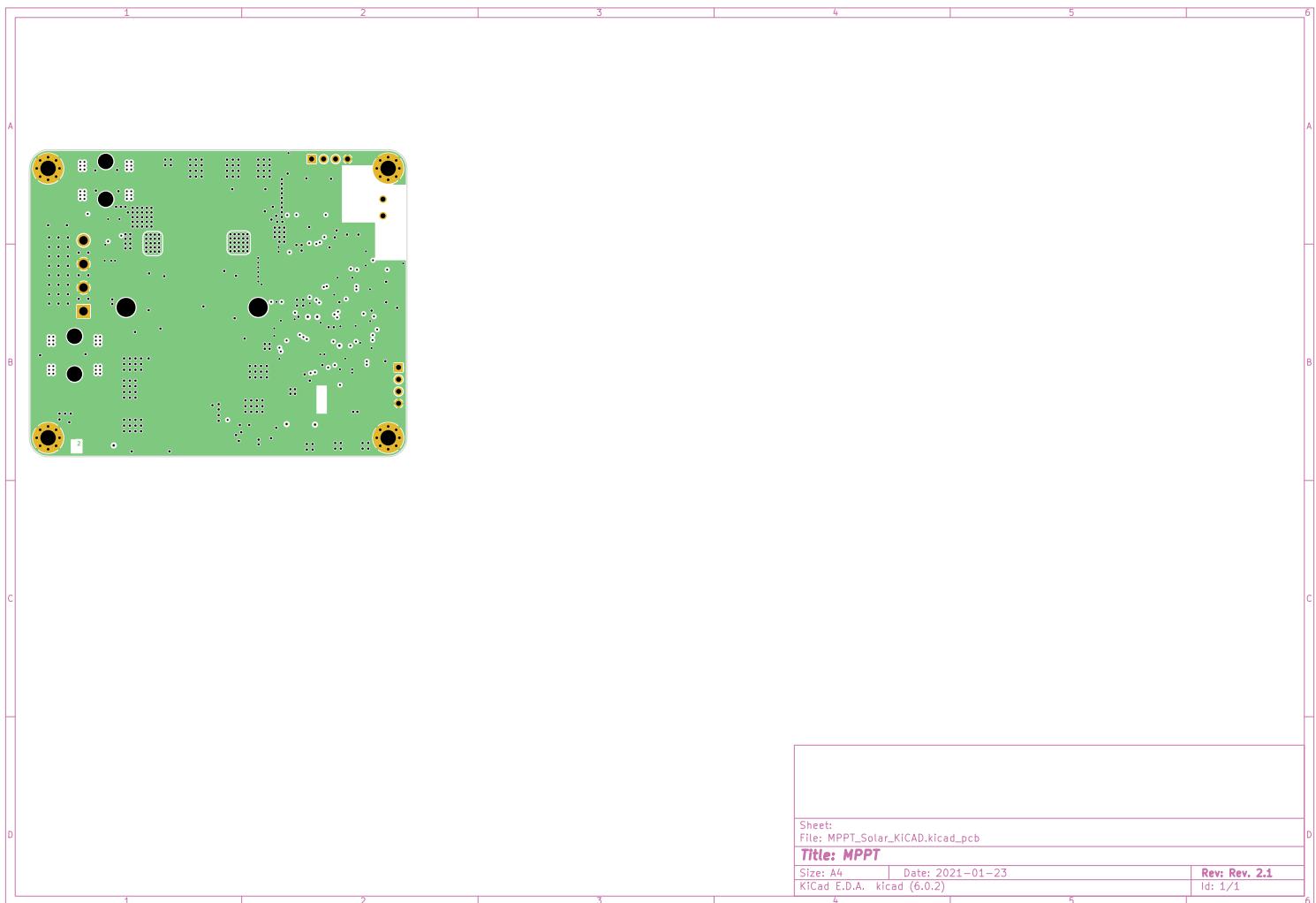


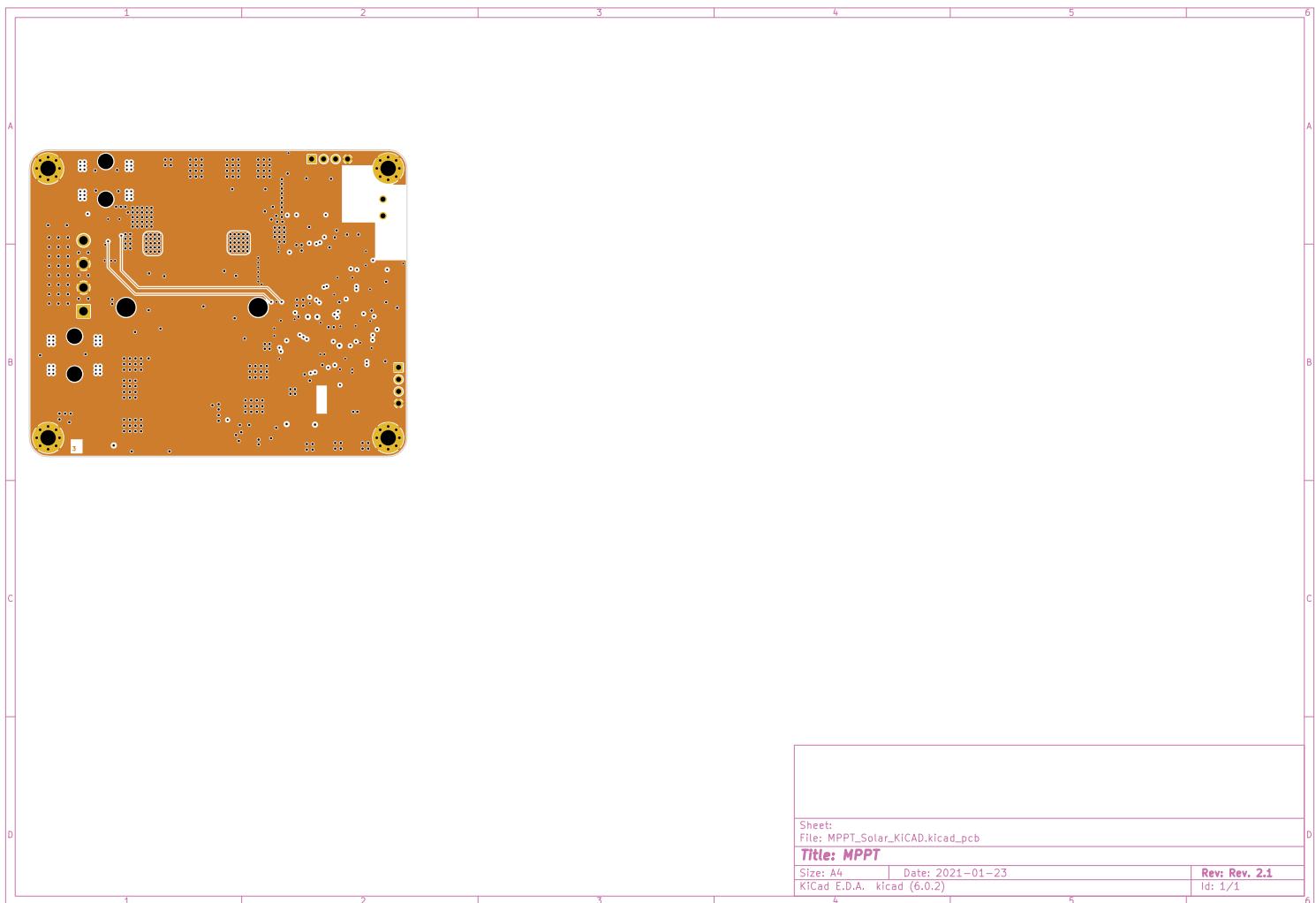


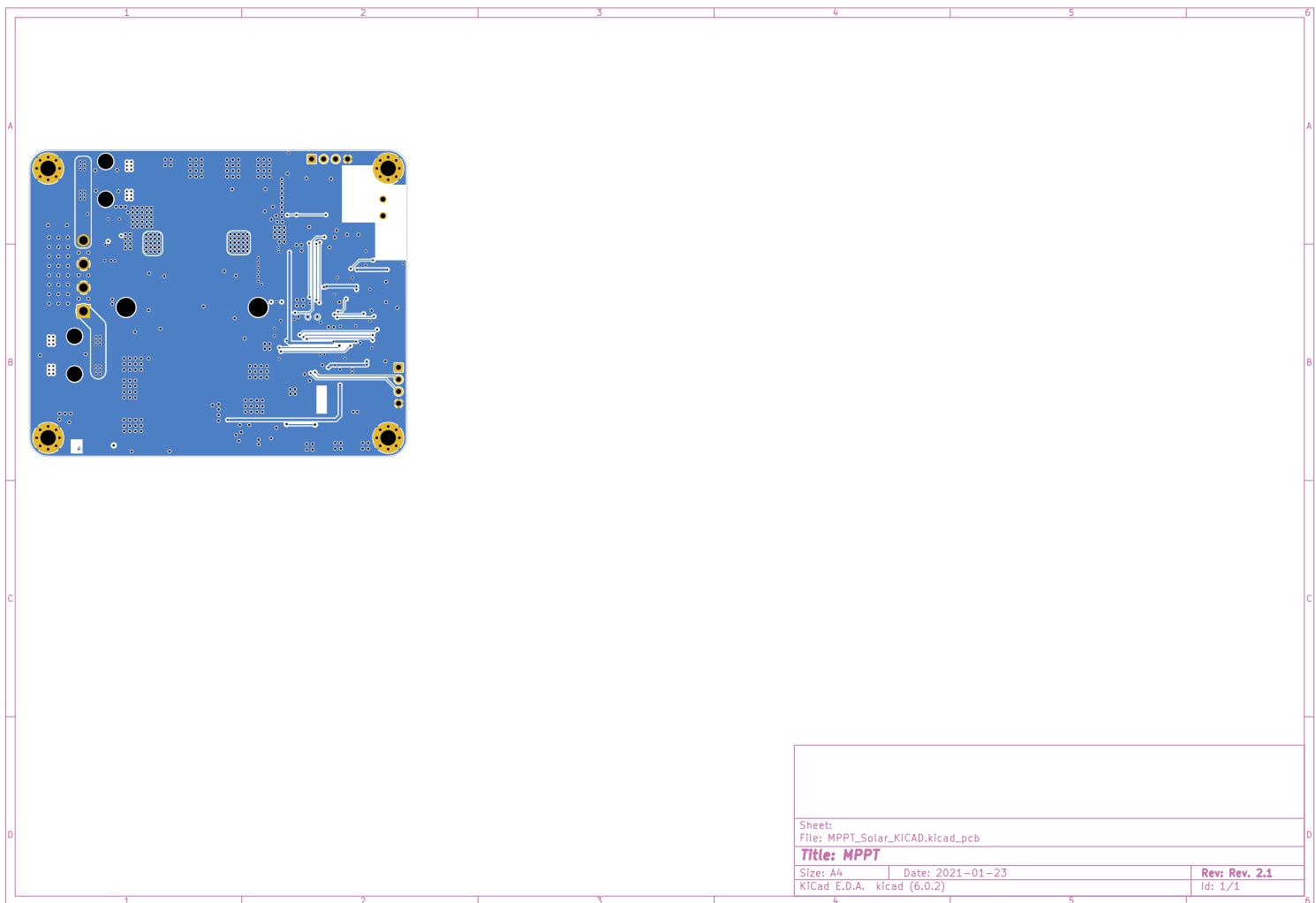


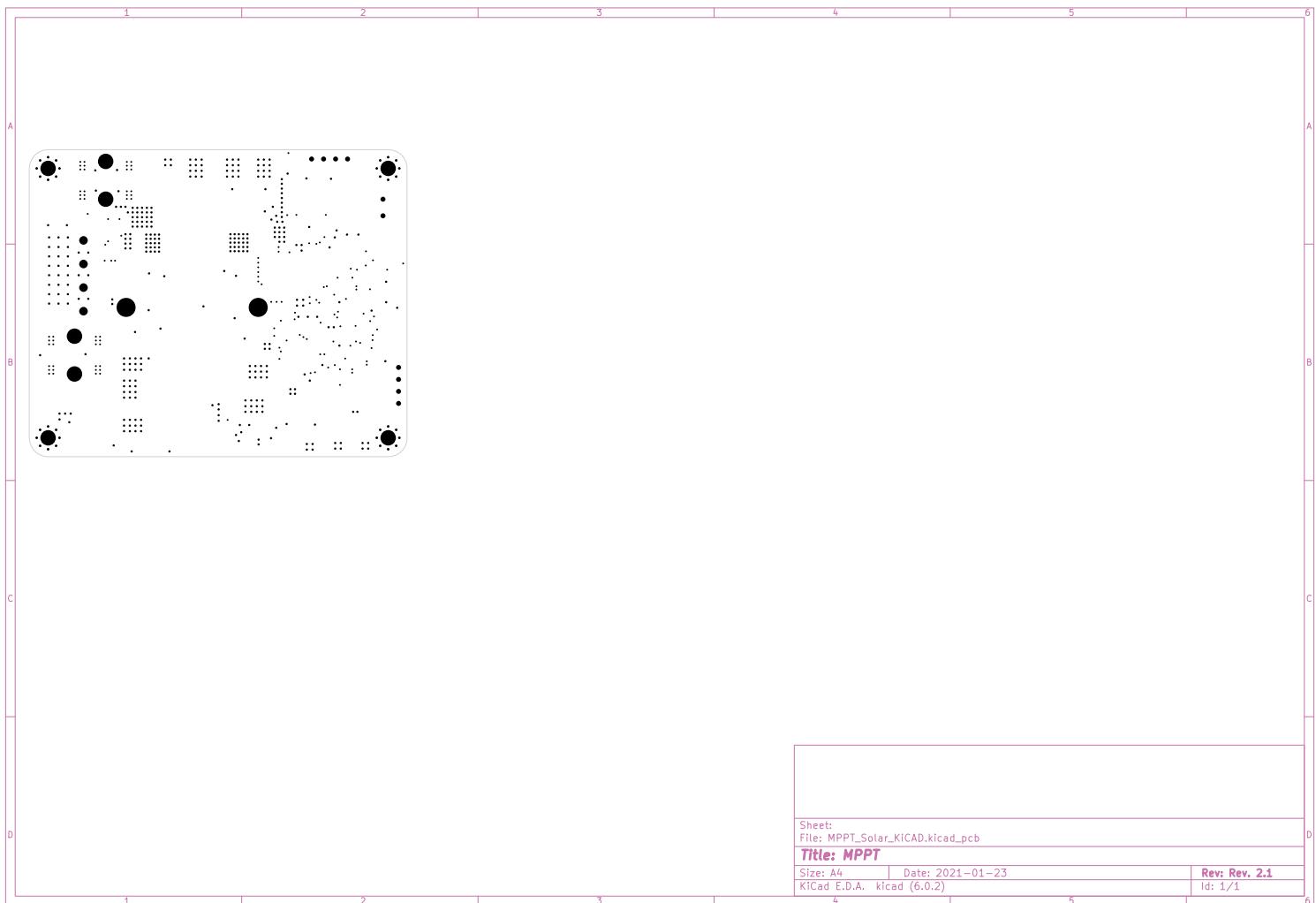
## **5.4 Platinenlayout**

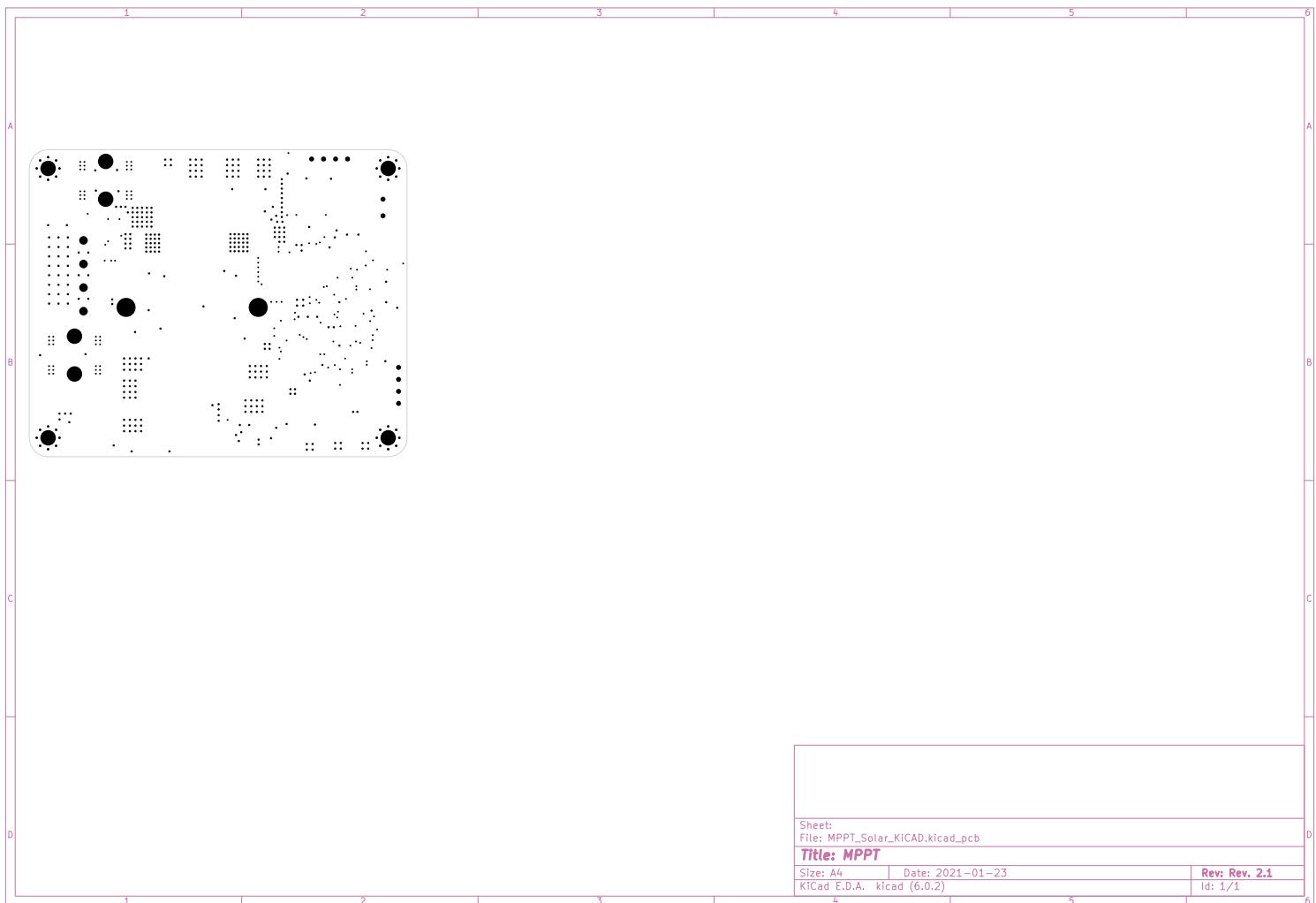


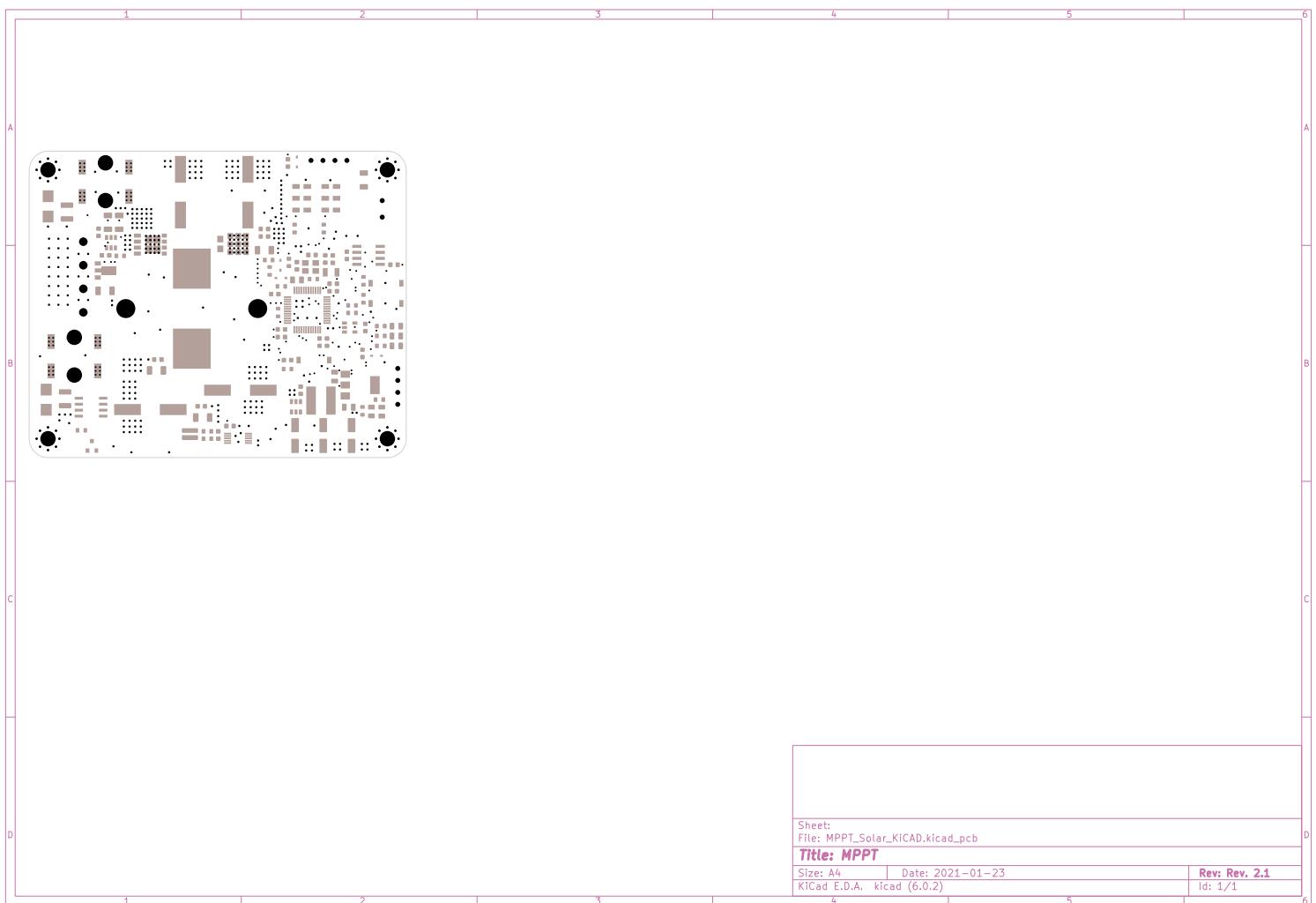


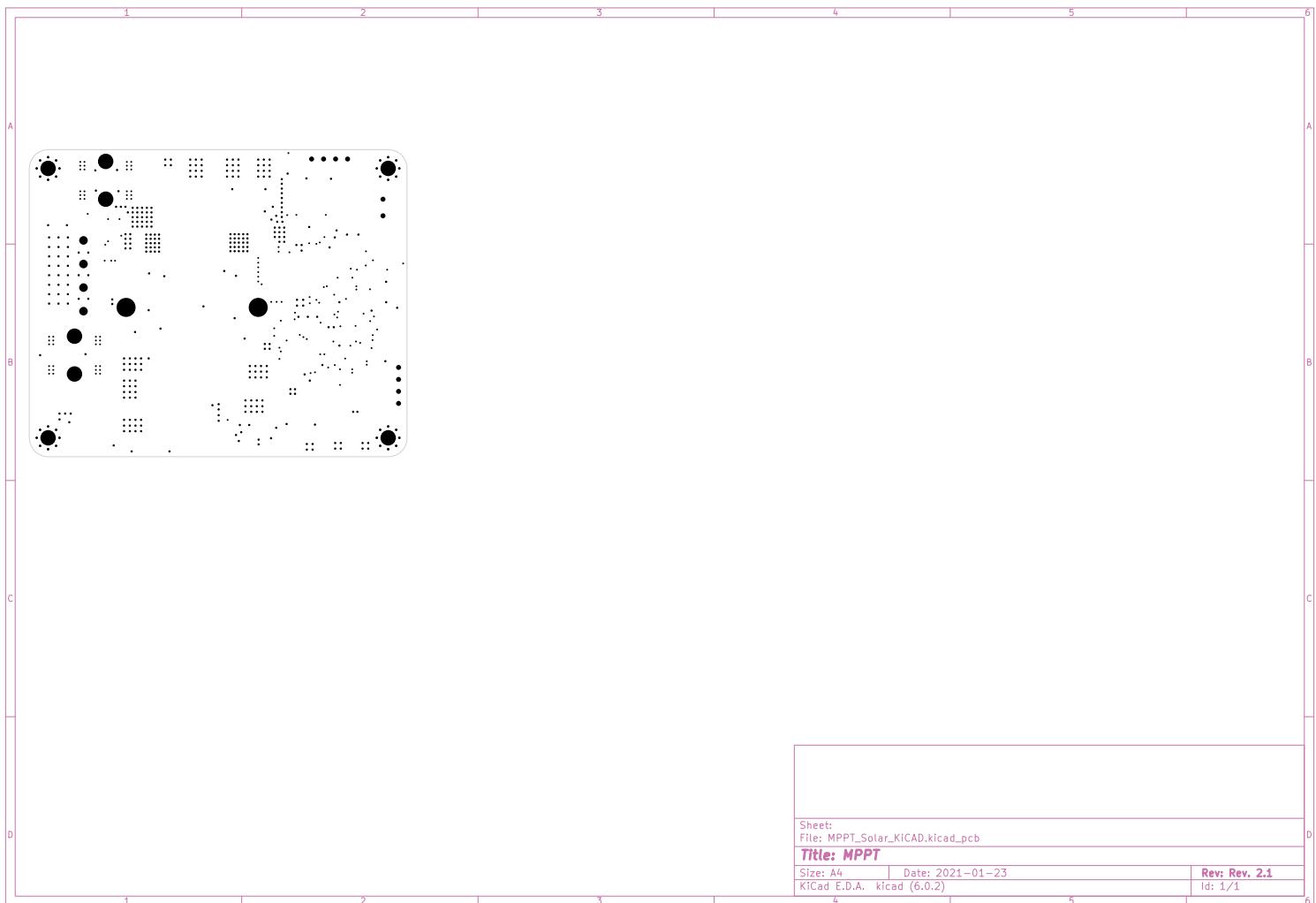


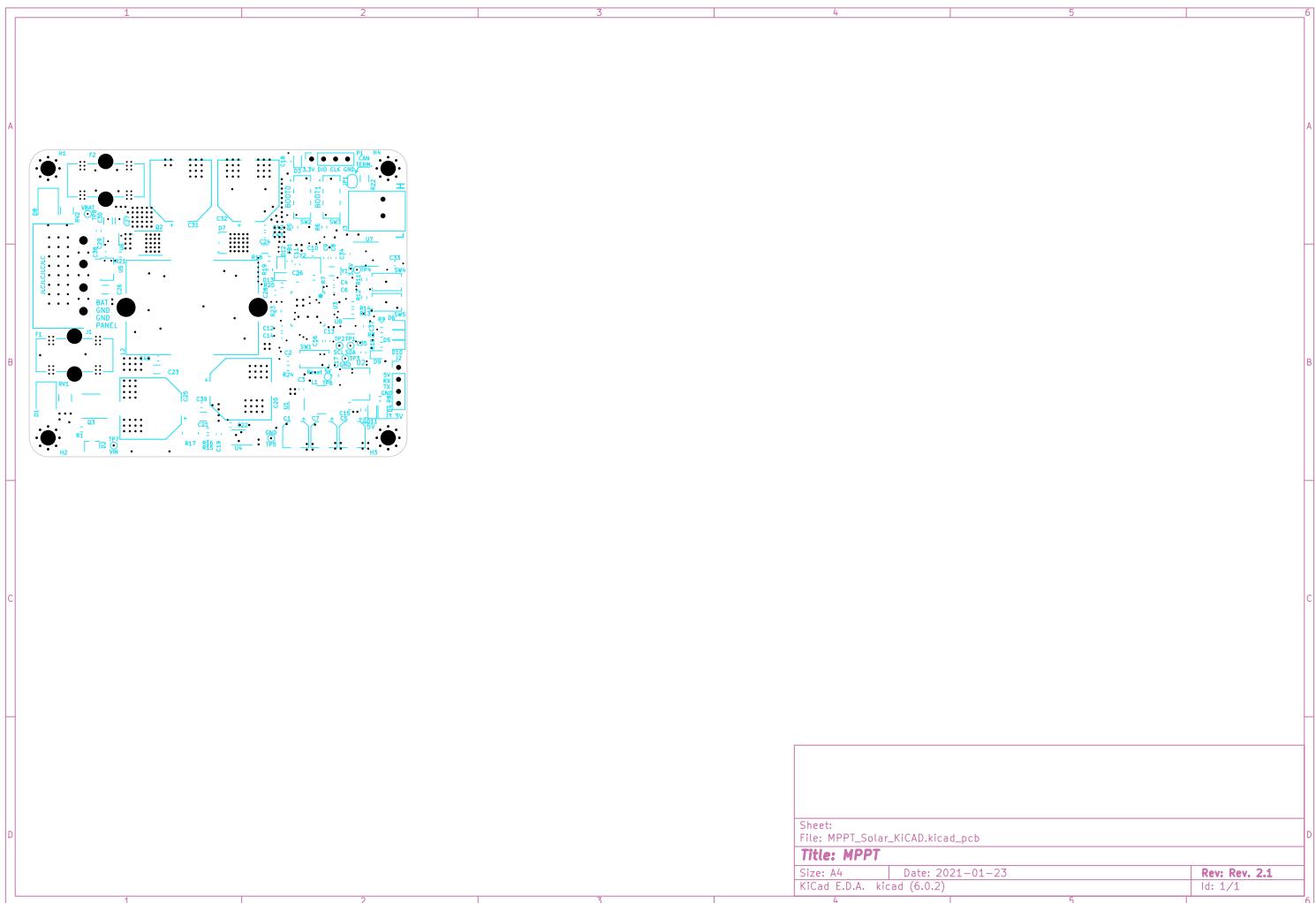


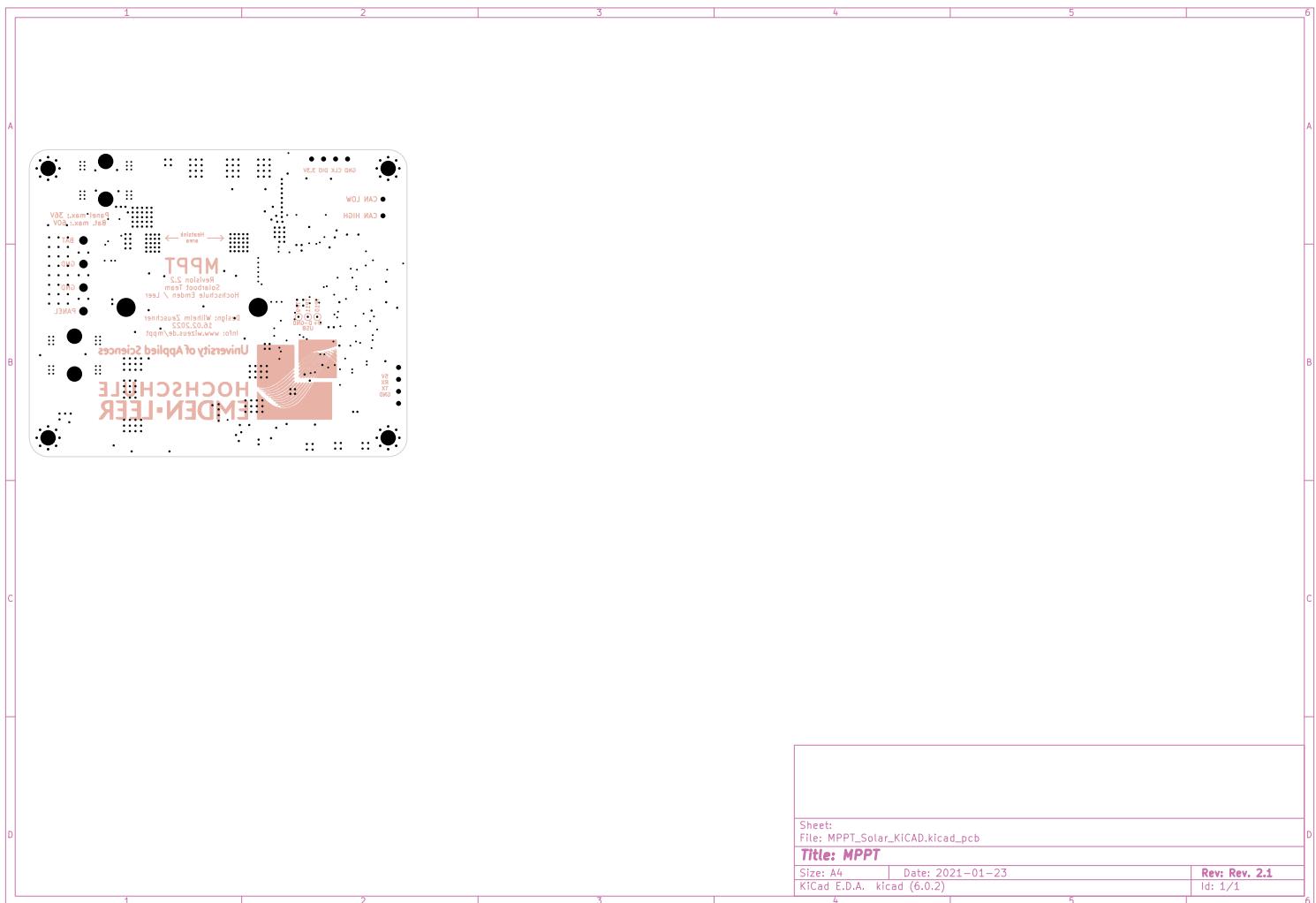


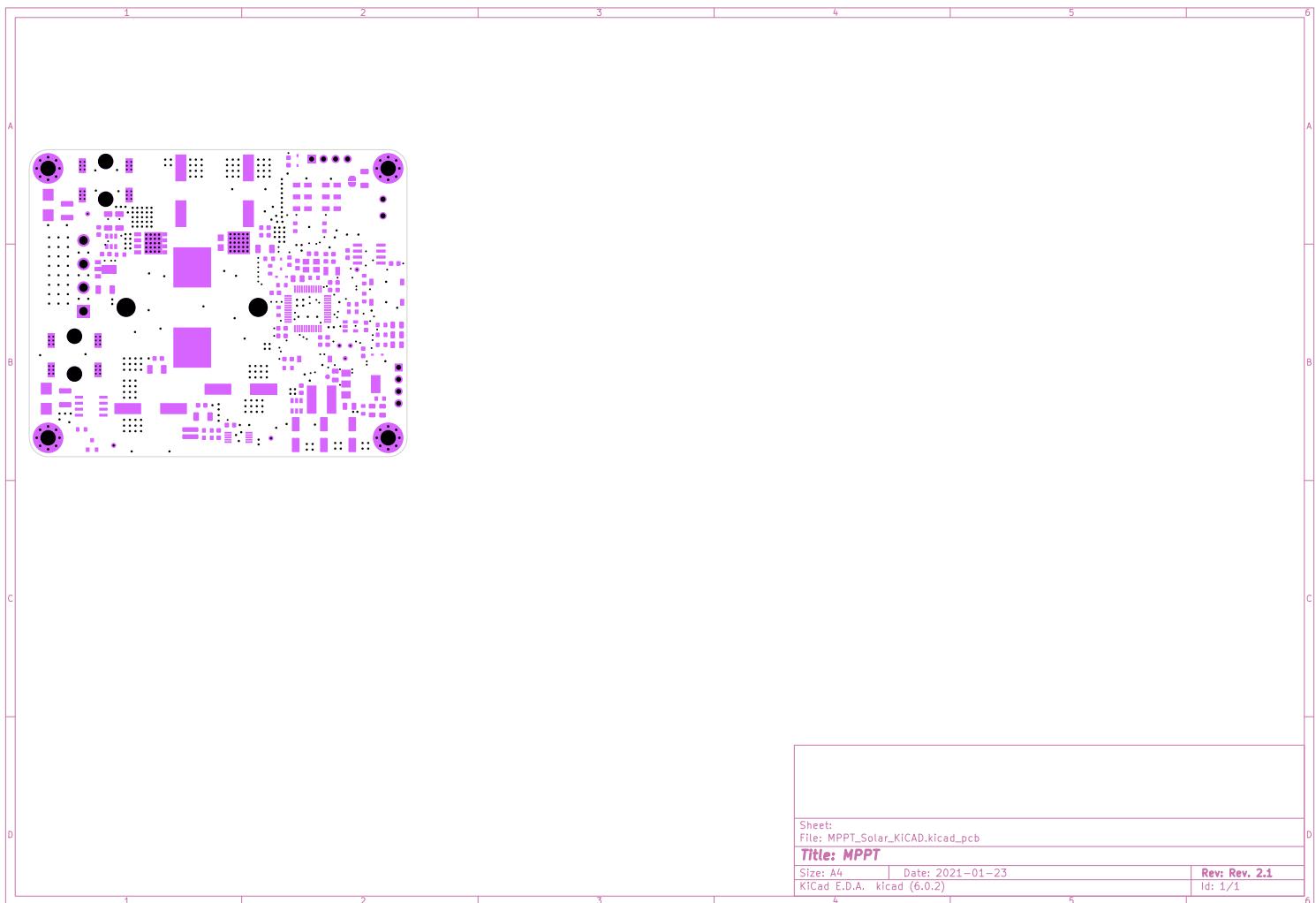


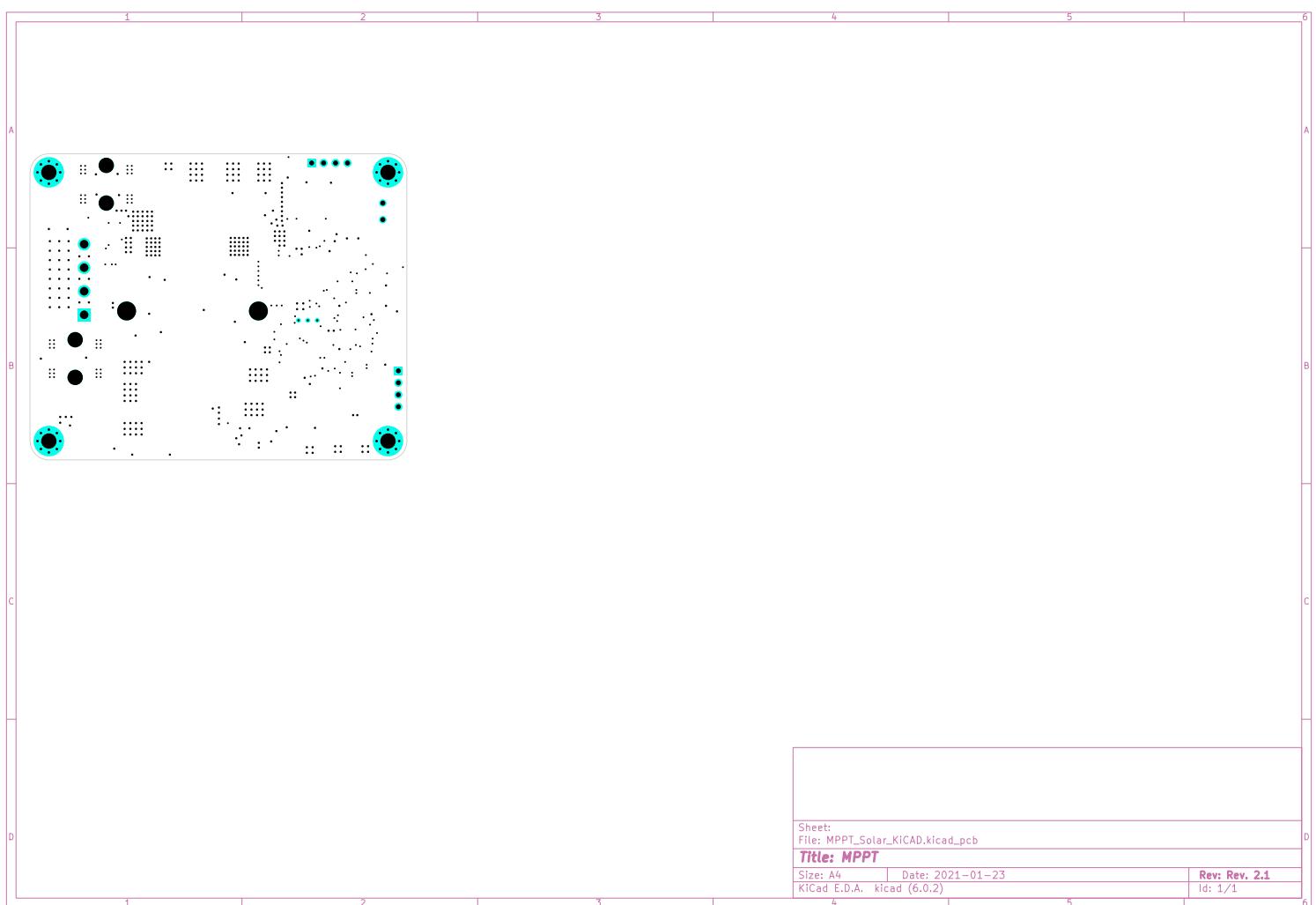


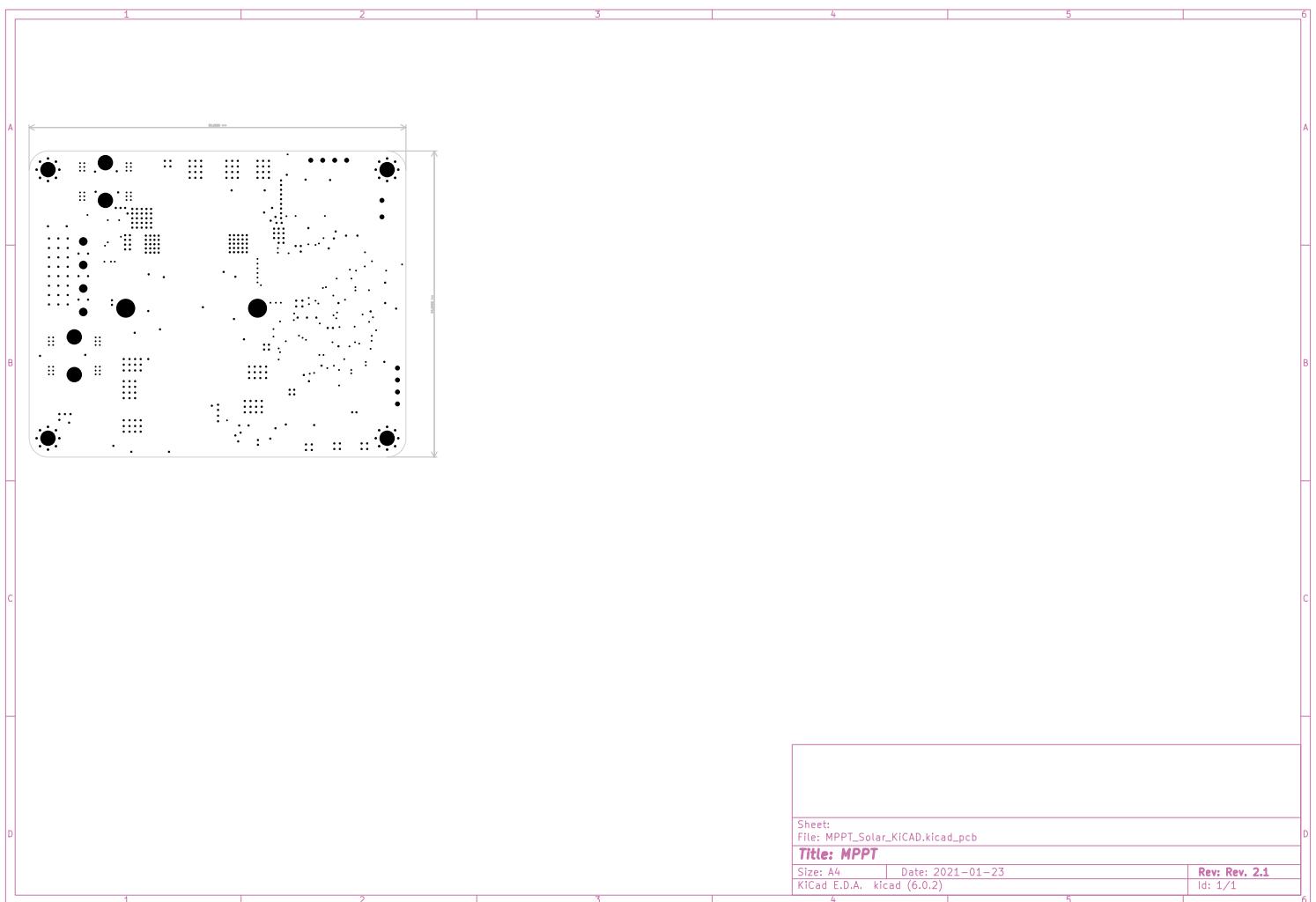


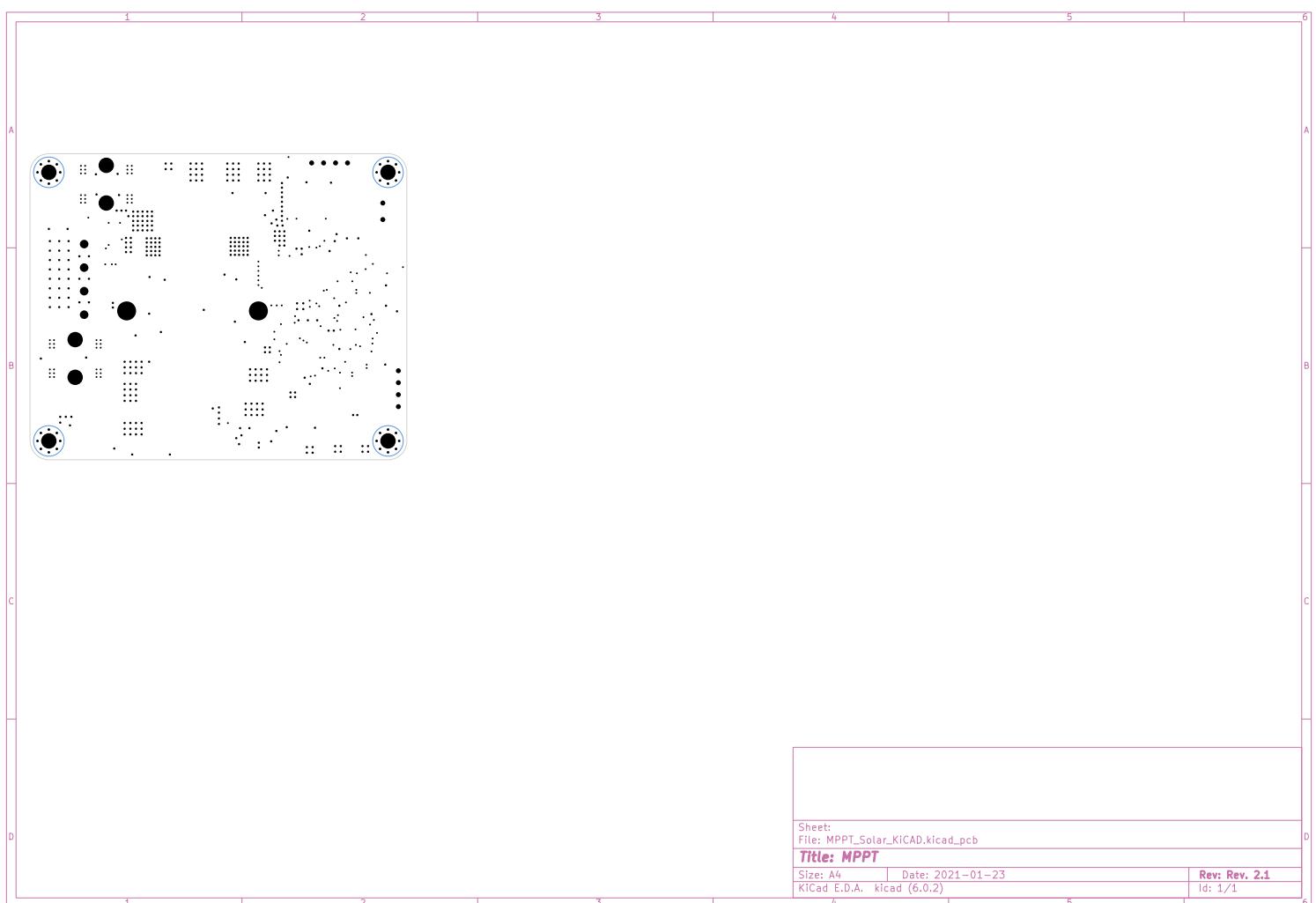


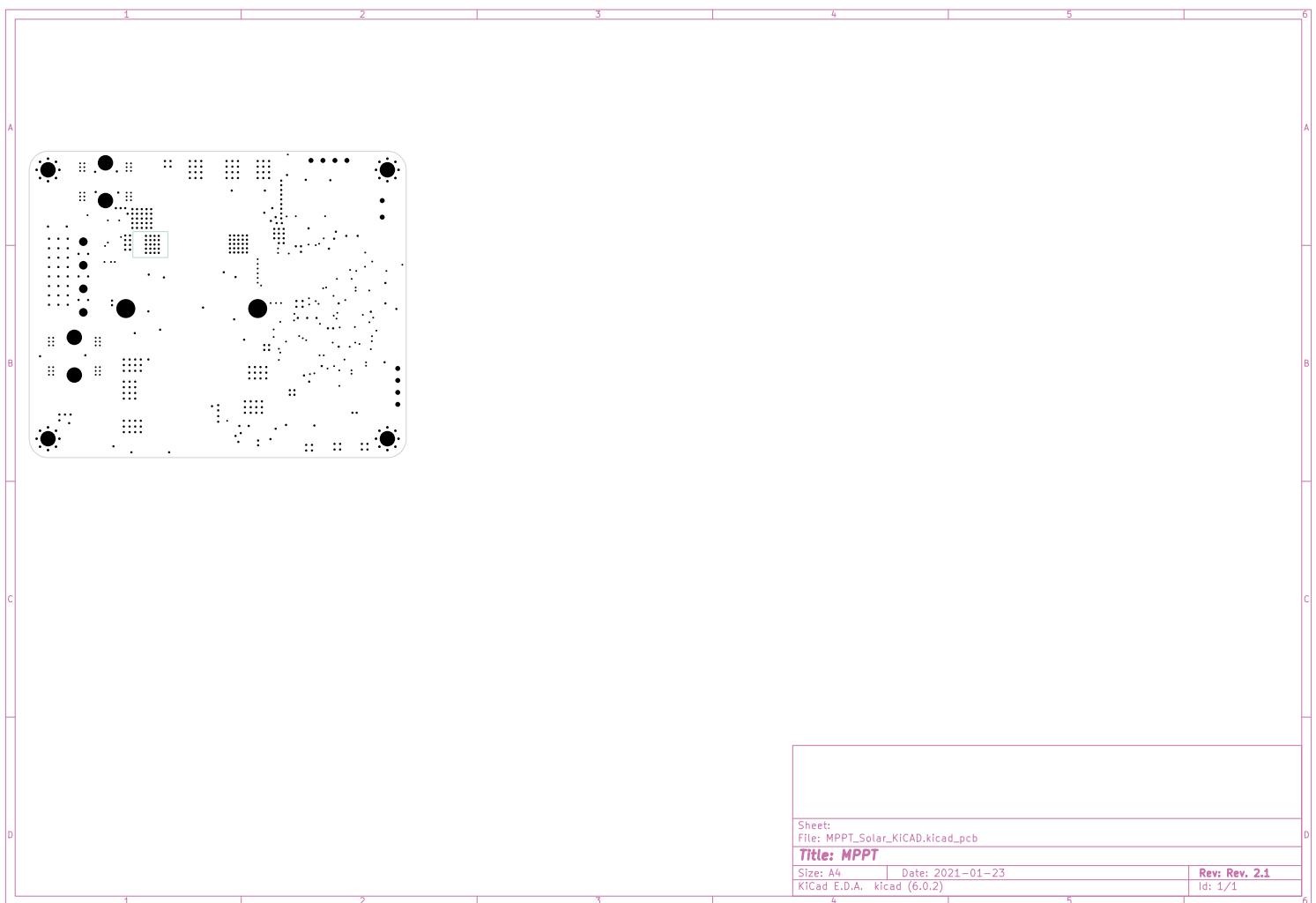


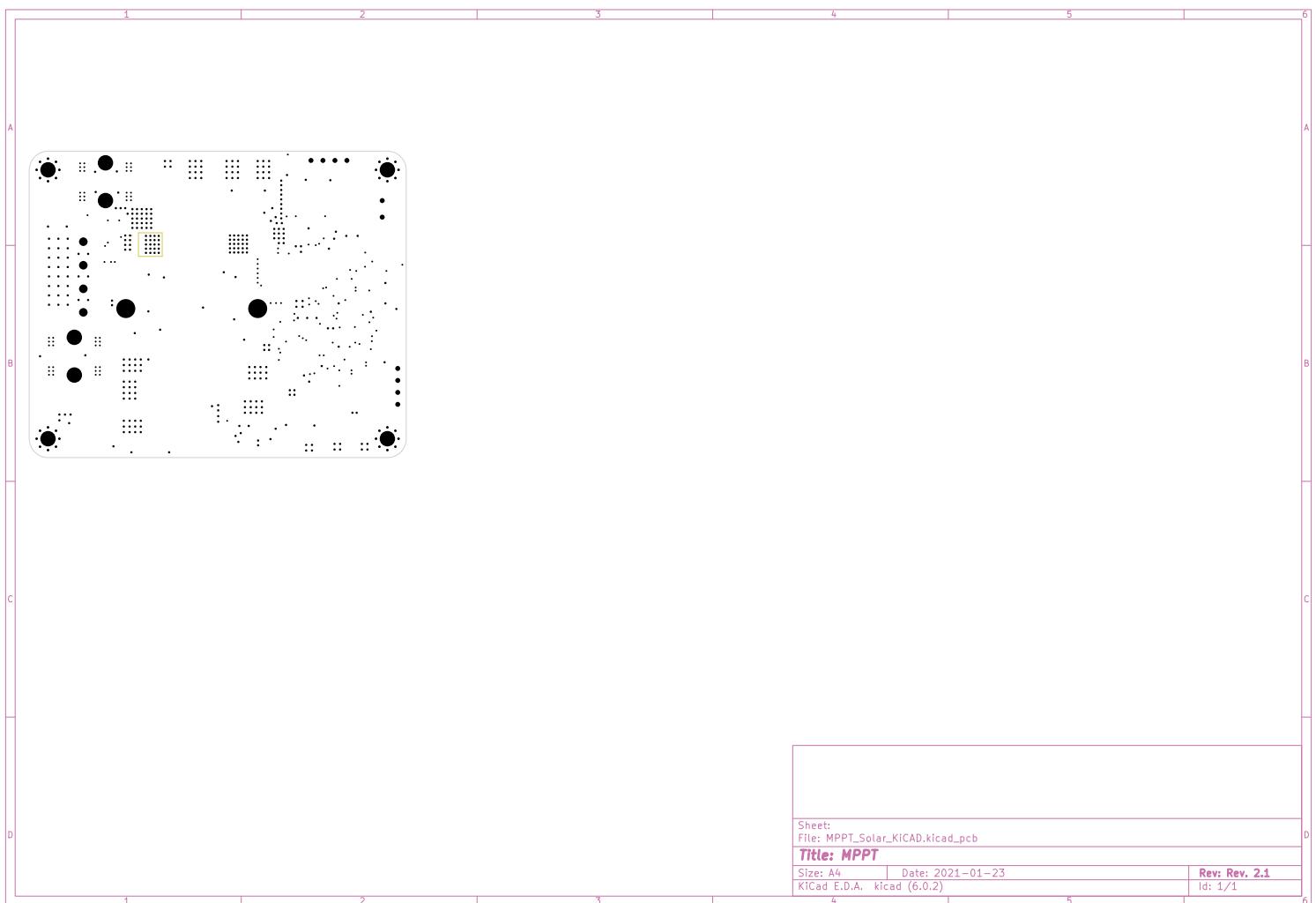


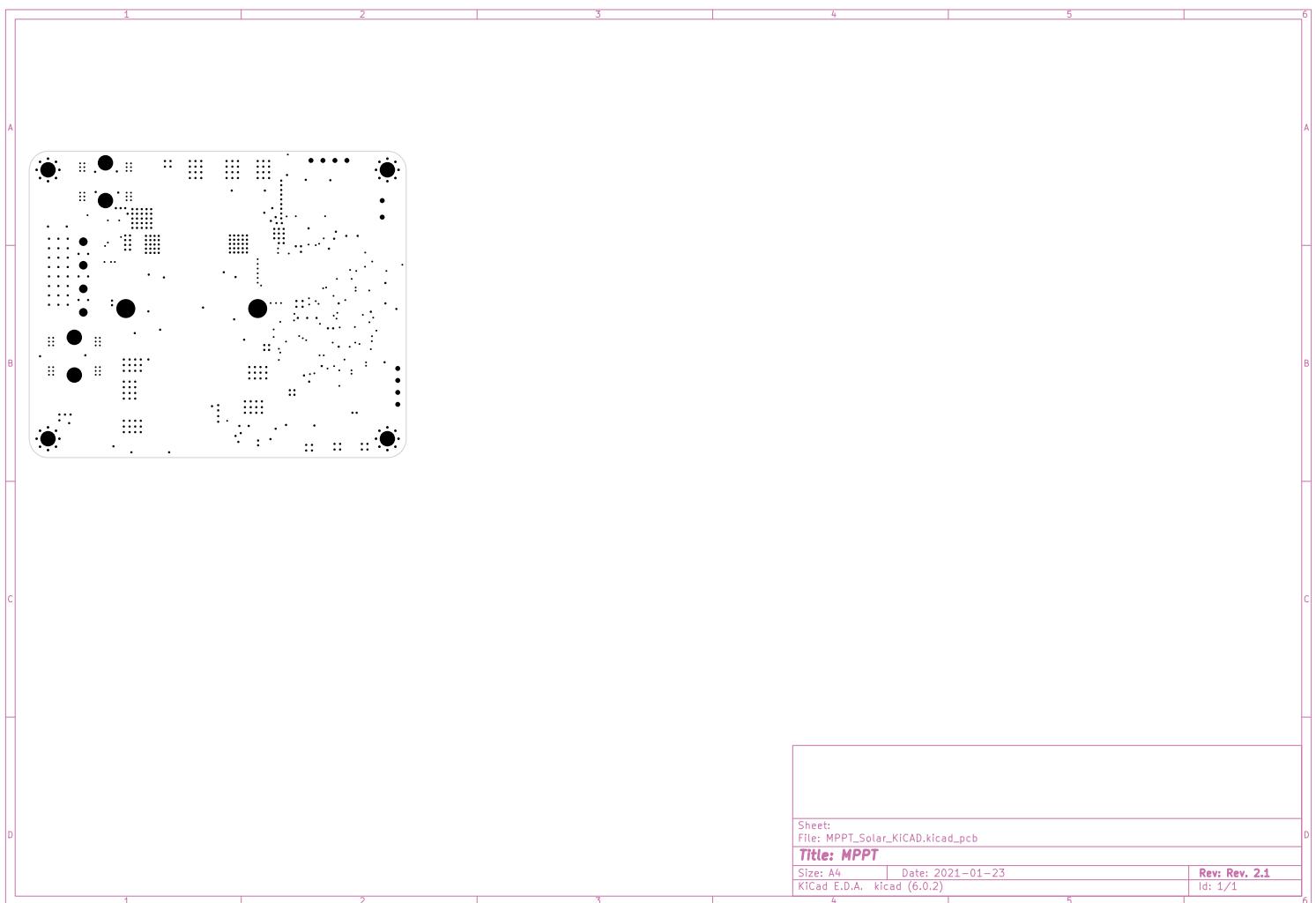


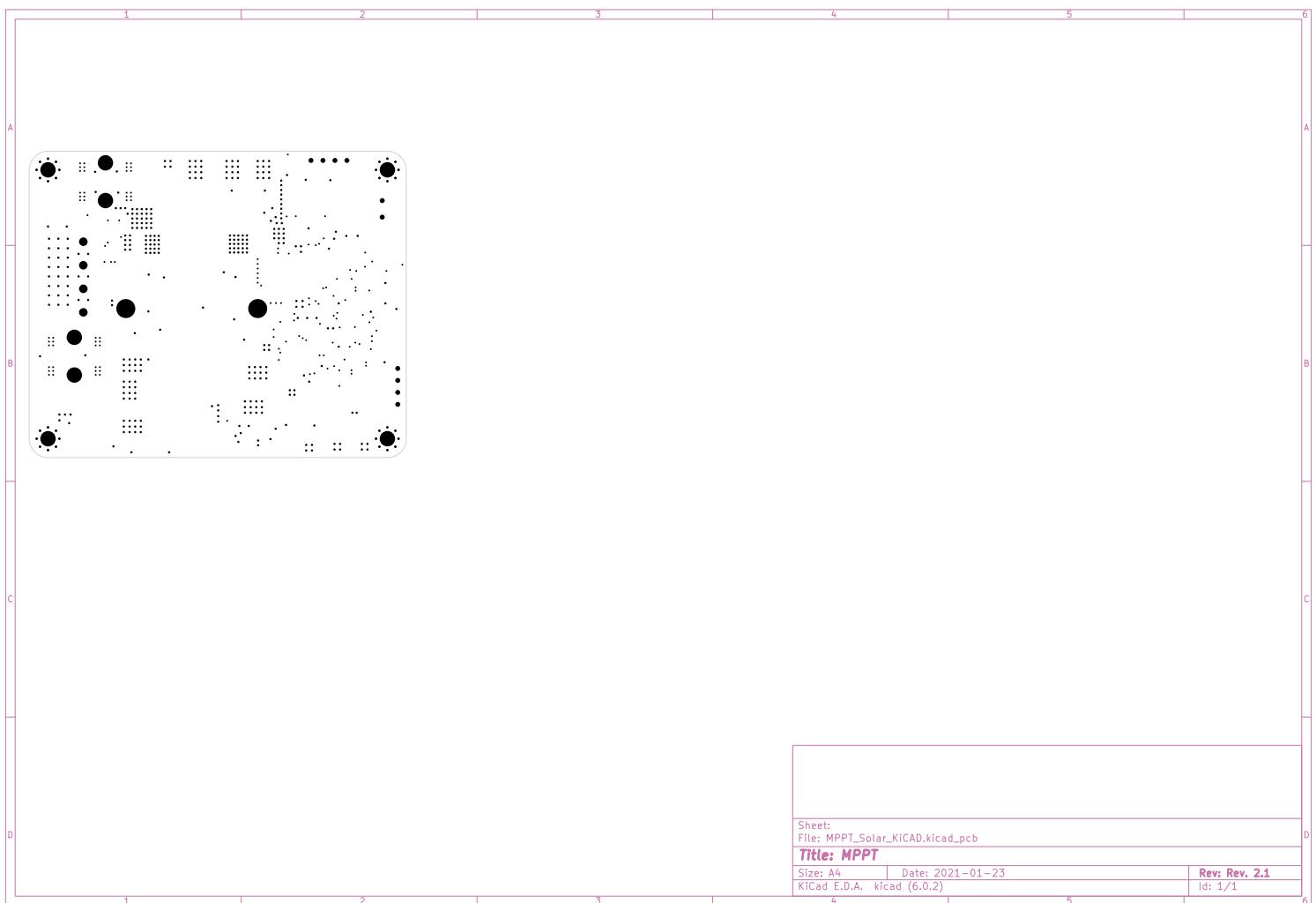


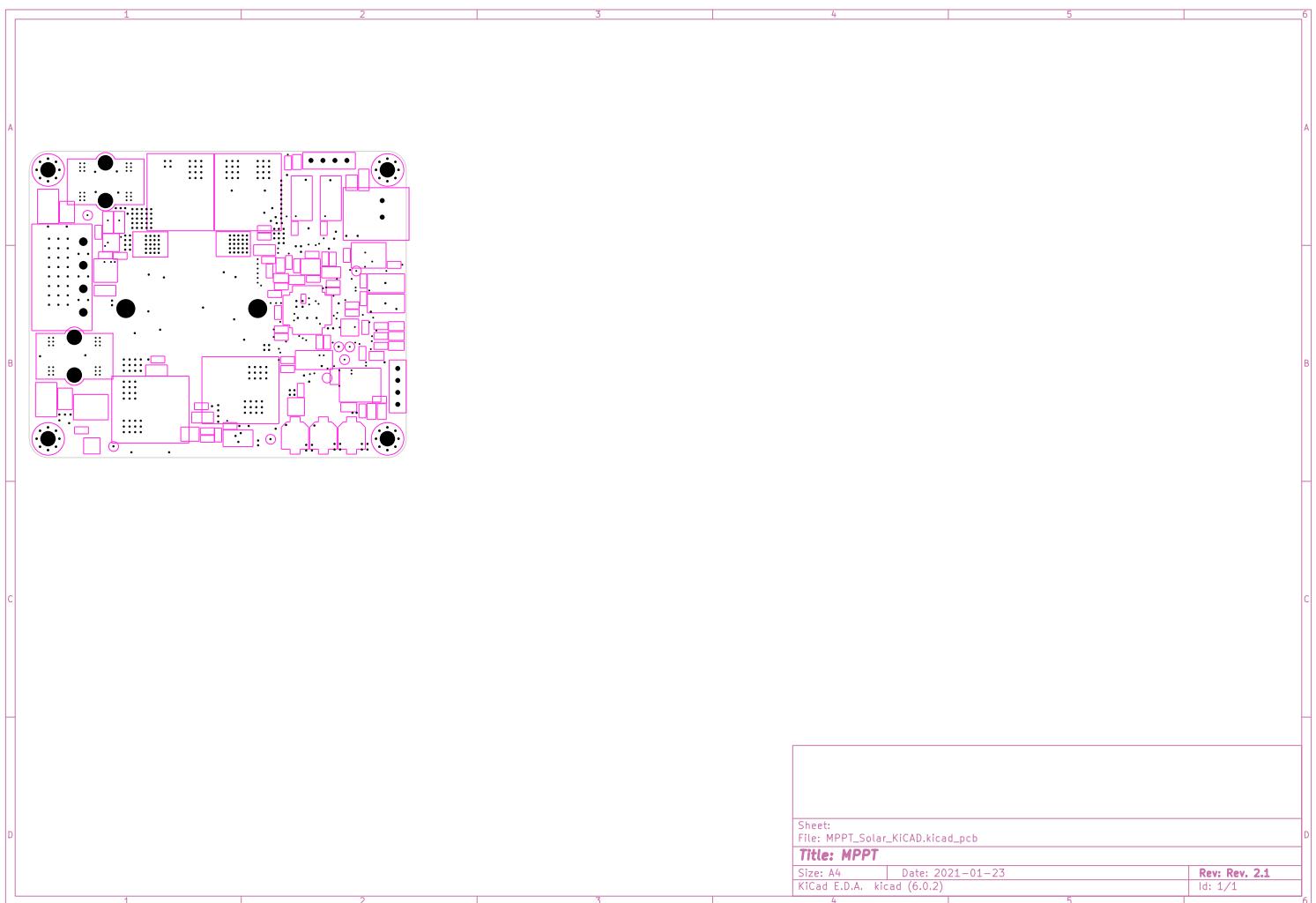


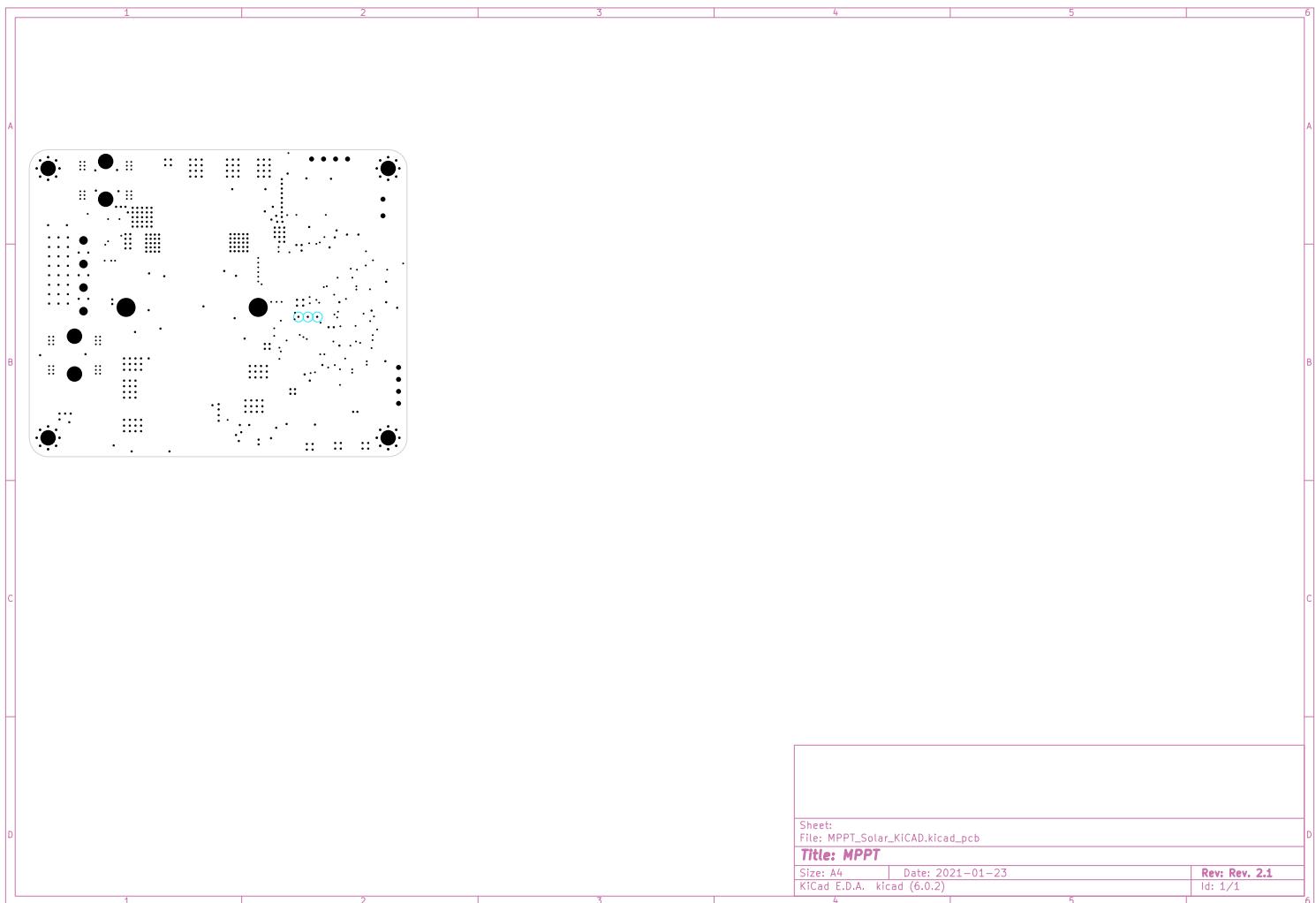


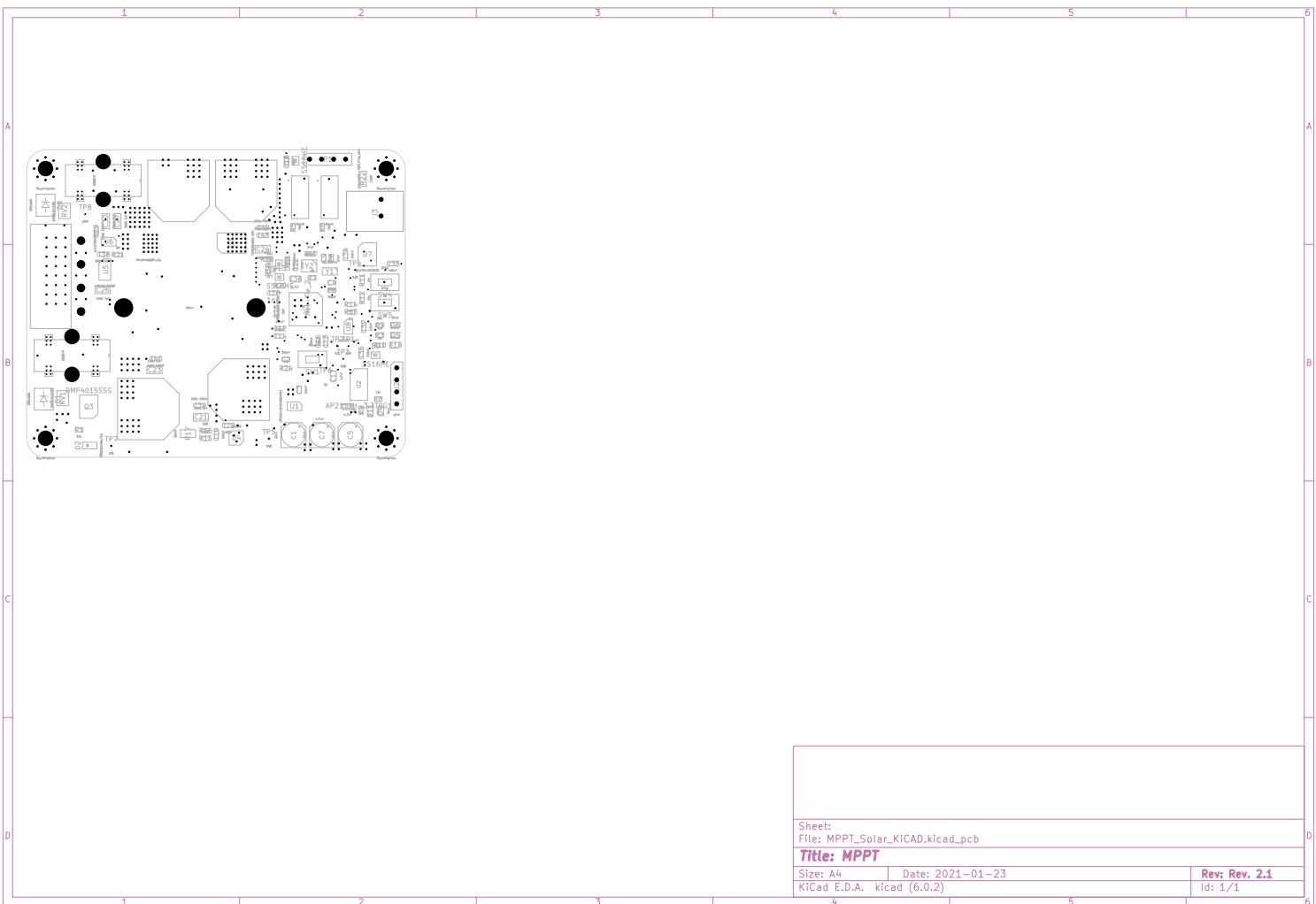


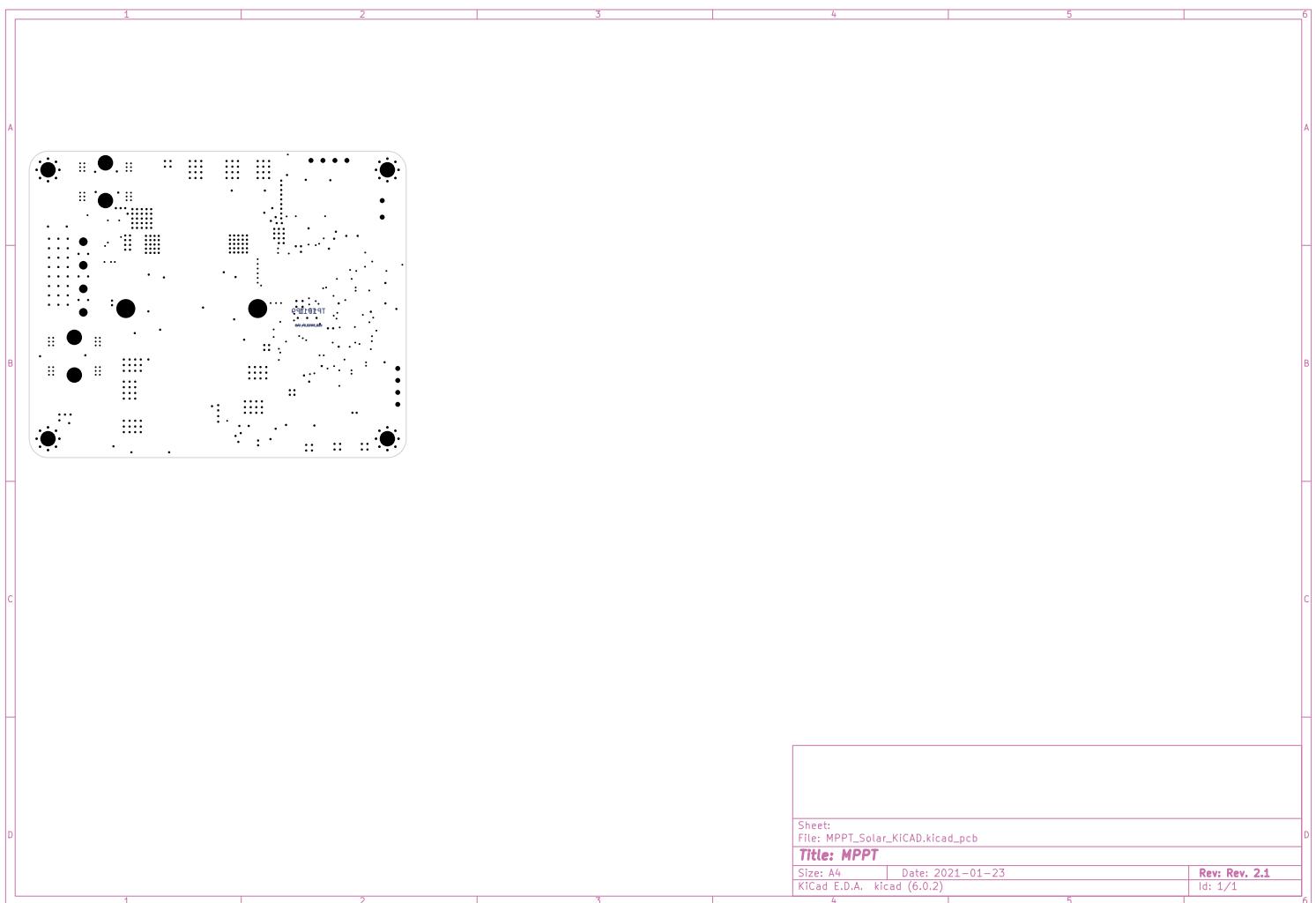












## **5.5 Abkürzungsverzeichnis**

### **Akronyme**

**ADC** Analog to Digital Converter.

**BOM** Bill Of Materials.

**CAN** Controller Area Network.

**EEPROM** Electrically Erasable Programmable Read-Only Memory.

**FET** Feld Effekt Transistor.

**GPIO** General Purpose Input / Output.

**IC** Integrated Circuit.

**IDE** Integrated Development Environment.

**LED** Light Emitting Diode.

**MLCC** Multi Layer Ceramic Capacitor.

**MPP** Maximum Power Point.

**MPPT** Maximum Power Point Tracker.

**PLL** Phase Locked Loop.

**RTC** Real Time Clock.

**SMD** Surface Mounted Device.

**SWD** Serial Wire Debug.

**THT** Through Hole Technology.

**UART** Universal Asynchronous Receiver-Transmitter.