Design and Implementation of a 300 W Educational MPPT Synchronous Buck Converter with ESP32 Control

1. Introduction

This document presents a comprehensive overview of the design, implementation, and operation of a 300 W Synchronous Buck Converter developed for Photovoltaic (PV) Maximum Power Point Tracking (MPPT) applications. The system employs an ESP32 microcontroller for real-time digital control and integrates dual INA228 precision power sensors to measure both input and output voltage and current parameters with high accuracy.

The project was carried out with the MSPE Internship at the Chair of Renewable and Sustainable Energy Systems, Technical University of Munich (TUM). Its primary objective was to create an educational and demonstrative platform that enables students to understand the working principles of MPPT algorithms, power electronics hardware, and embedded control systems in a hands-on, visual, and interactive manner.

Beyond providing a functional MPPT converter, the design emphasizes clarity, transparency, and accessibility both in hardware layout and firmware structure to facilitate its use as a teaching and research tool. Through the integration of a web-based user interface, real-time data visualization, and open-source firmware, the project bridges the gap between theoretical learning and practical experimentation in renewable energy system education.

2. Project Overview

The primary objective of this project was to independently design, implement, and validate a robust and efficient 300 W synchronous buck converter capable of extracting maximum power from a photovoltaic (PV) source through Maximum Power Point Tracking (MPPT). The work includes the complete development cycle from schematic design and component selection to PCB layout, power and control stage integration, firmware development, and system-level validation.

In addition to its technical performance, the project places strong emphasis on educational value. The converter is designed not only to perform effective MPPT but also to serve as an instructional tool, offering a clear and accessible demonstration of key power electronics and embedded control concepts. The printed circuit board (PCB) includes informative graphical elements and labeling, while the control firmware employs simplified yet functional algorithms, making the system ideal for use in laboratory courses and teaching environments.

2.1. Key Features

- **300 W Synchronous Buck Converter**
  Solely designed and implemented to achieve high efficiency and stable operation under varying PV input conditions. The converter demonstrates both high-side and low-side MOSFET switching, driven by a dedicated gate driver IR2104 for optimal performance.
- **MPPT Algorithm Implementation**
  Employs a basic yet effective MPPT algorithm to continuously track the maximum power point of the PV array. The algorithm is Perturb and Observe and it prioritizes simplicity and interpretability, ensuring students can easily follow its operation and behavior.
- **ESP32-Based Control**
  Utilizes the **ESP32 microcontroller** as the central control unit, responsible for executing the MPPT algorithm, managing power measurements, and handling user interaction. The board supports USB-based programming and communication for rapid firmware iteration.
- **Dual INA228 Precision Power Sensors**
  Integrates two Texas Instruments INA228 digital power monitors for precise measurement of input and output voltage and current. The dual-sensor configuration enables real-time efficiency and power-flow monitoring essential for MPPT performance evaluation.
- **Web-Based User Interface (UI)**
  A lightweight HTML-based web application provides remote control and visualization of the converter's operation. Using Chart.js, the interface displays live I–V curves, operating points, and performance data, allowing users to observe MPPT behavior in real time.
- **End-to-End Project Ownership**
  The system was developed entirely from the ground up including schematic design, PCB layout, component selection, firmware programming, debugging, and experimental validation demonstrating full-cycle hardware and software integration.

- **Educational Focus and Visualization**
  The converter emphasizes teachability through its visual PCB design, intuitive firmware structure, and web-based visualization. These features collectively support students in exploring the principles of MPPT control, power electronics, and system measurement in a practical context.

3. Hardware Design

This section details the hardware components and their integration into the synchronous buck converter system. The PCB design also includes informative visualizations to enhance its educational value.

3.1. Synchronous Buck Converter Topology (Power and Gate Driver Stage)

A synchronous buck converter topology was chosen for its higher efficiency than a regular buck converter, particularly at higher power levels, due to the replacement of the free-wheeling diode with a MOSFET to decrease losses.

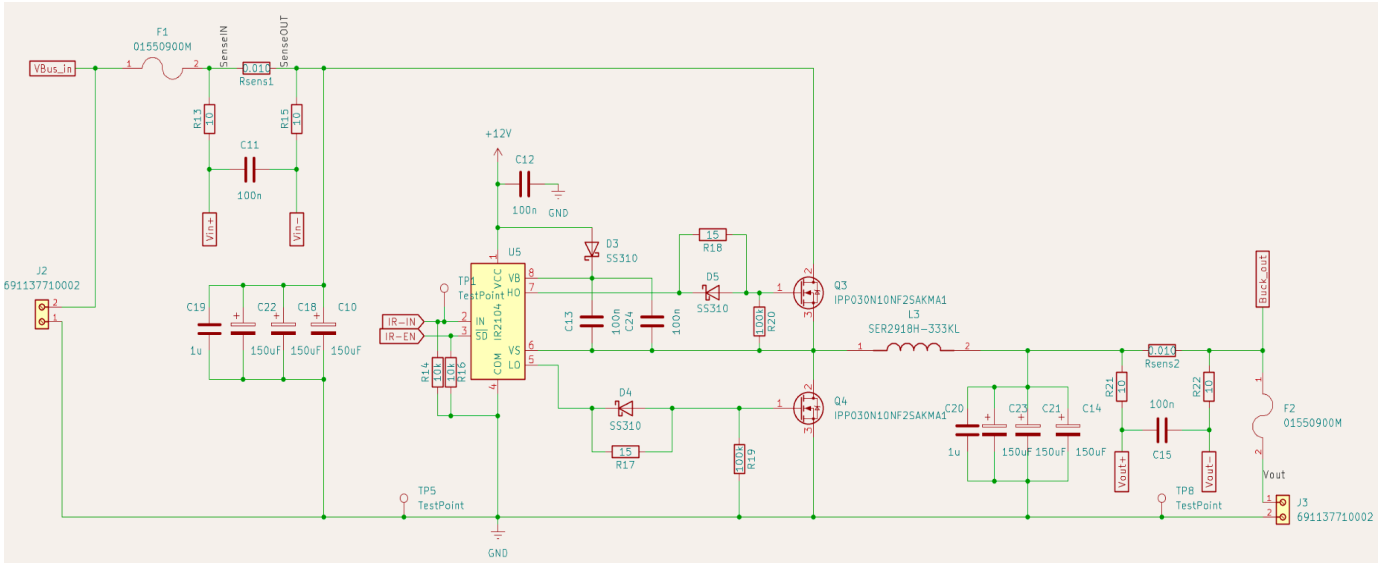


Figure 1: Synchronous Buck Converter Schematics with IR2104 and sense branch

The PV source used in this project was a JWS-Solar 1210 ProductLink, which can reach an open-circuit voltage of 38.8, a short-circuit current of 9.4 A, and operates at a maximum power point (MPP) of 31.9 V and 8.9 A. These parameters defined the electrical boundaries for the converter design, including voltage, current, and power ratings of the power stage components. The converter operates at a switching frequency of 39 kHz, a typical and efficient choice for ESP32-based PWM control.

To determine suitable values for the buck inductor and both input and output capacitors, I initially used an online Excel-based calculator view only to estimate component requirements based on target ripple levels and operating conditions.

MPPT: SYNCHRONOUS BUCK DESIGN CALCULATOR			
REQUIRED PARAMETERS	**VALUE**	**UNIT**	**NOTES**
V_{mp}	31.9	V	Solar panel maximum powerpoint voltage (found in solar panel back sticker)
I_{mp}	8.9	A	Solar panel maximum powerpoint current (found in solar panel back sticker)
V_{batt}	16	V	Maximum battery voltage of your setup
f_{sw}	39	kHz	MPPT buck converter pwm switching frequency (Visit Sheet #2)
ASSUMED PARAMETERS	**VALUE**	**UNIT**	**NOTES**
V_{ripple}	0.05	V	MPPT output ripple voltage (50mV is a good and ideal value)
$\%I_{ripple}$	35	%	
$\%_{efficiency}$	95	%	MPPT buck conversion efficiency (use 100% for ideal computation, 96% for actual computation)
SOLVED PARAMETERS	**VALUE**	**UNIT**	**NOTES**
Solar Power (P_{solar})	283.91	W	The maximum solar panel power output
Duty Cycle (D)	52.79656822	%	The PWM duty cycle % at given parameter conditions
Ripple Current (dI)	6.21053125	A	Maximum continous current for MOSFET
Peak Inductor Current (Ipk	20.84964063	A	Maximum current rating for inductor (selected inductor must have a higher saturation current)
Inductance (L)	34.65847585	uH	Inductance for your MPPT's synchronous buck inductor (select nearest value)
Output Capacitor (Cout)	398.1109776	uF	Capacitance of your MPPT's output capacitor to achieve a ripple voltage less than or equal to V_{ripple} (select any value above it)

Figure 2: Necessary INductor and Capacitor Values with Max Current Calculation

These preliminary calculations were then refined through a series of iterative LTSpice simulations, which incorporated realistic models of the gate driver (IR2104), MOSFETs, diodes(SS310), and the PV panel. The simulations allowed detailed evaluation of dynamic behavior such as ripple amplitude, transient response, and current stress on each component.

The capacitor selection focused on achieving stable operation under varying PV input and load conditions. I selected aluminum–polymer capacitors for both input and output filtering, as they provide low equivalent series resistance (ESR), high ripple current capability, and sufficient voltage margin for the converter's operating range. Multiple capacitors were connected in parallel to further reduce ESR and increase total current handling capacity. In addition, a 1 µF MLCC capacitor was placed in parallel to suppress high-frequency switching noise.

For the inductor, I selected a component that provides the required inductance value while maintaining high saturation current capability to ensure linear behavior under full-load operation. The selected inductor effectively prevents magnetic saturation even at the maximum current expected from the PV source.

To ensure safe operation, I also included fuse protection of 15 Amps sized according to the converter's maximum current limits, providing protection against overcurrent and short-circuit conditions. All the components are given in appendix BOM.

3.1.1 Gate Driver — IR2104 and MOSFETs

The IR2104 high/low-side gate driver is the central interface between the ESP32 PWM signals and the power MOSFETs in the synchronous buck stage. It provides the level shifting and bootstrap high-side supply required to drive the high-side N-channel MOSFET, while also driving the low-side device. The schematic implementation on the board closely follows the IR2104 application recommendations from the datasheet (bootstrap diode, bootstrap capacitor, decoupling, and gate network).

Functional role

- **High/low side driving:** The IR2104 accepts the logic-level PWM input from the ESP32 and produces the complementary HO (high-side) and LO (low-side) gate signals referenced to the switching node (VS) and ground respectively.
- **Bootstrap supply:** A bootstrap diode and capacitor between VCC/VB and VS create the floating high-side supply for the IR2104, enabling it to drive the high-side gate to the required VB voltage. (See bootstrap diode/capacitor on the schematic.)
- **Internal deadtime:** The IR2104 provides an internal deadtime (~540 ns typical) to prevent direct conduction (shoot-through) of the high-side and low-side MOSFETs during switching transitions. This hardware deadtime relieves the software from enforcing a minimum off time, improving robustness.

Gate drive network — resistors, diode and discharge resistor

The gate network uses a small series gate resistor (15 Ω) in combination with a Schottky diode (SS310) in parallel and a 100 kΩ gate-to-source discharge resistor. This arrangement is intentionally asymmetric to control turn-on and turn-off characteristics independently:

- **Series gate resistor (15 Ω)** — placed in series between the driver output and the MOSFET gate:
  • Limits the instantaneous gate charging/discharging current, which reduces ringing and electromagnetic interference (EMI) caused by parasitic inductances.
  • Controls the MOSFET switching speed in the *turn-on* direction (when current must charge the gate capacitance), reducing di/dt and voltage overshoot across the MOSFET and the switching node.
- **Schottky diode (SS310) in parallel with the series resistor** — oriented so that current bypasses the resistor in the opposite direction:
  Creates an asymmetrical gate drive: slower turn-on through the resistor, faster turn-off through the diode. The diode provides a low-impedance path for the faster transition, enabling the gate to be discharged quickly and minimizing the time both MOSFETs could be partially on. The Schottky type is chosen because of

its low forward voltage and fast switching, improving turn-off speed while keeping forward losses low. Asymmetrical gating is useful to tune trade-offs between switching losses, EMI, and safe deadtime behavior: slower turn-on reduces switching stress and ringing; faster turn-off reduces the overlap interval that can create shoot-through if a fault occurs.

- **Gate-to-source discharge resistor (100 kΩ)** — to pull the gate to the source when the driver is inactive: Prevents the gate from floating and accidentally turning the MOSFET partially on. Ensures the gate is at a safe, defined potential when the driver is unpowered or during assembly and testing.

On the PCB the series resistor should be placed as close as possible to the MOSFET gate pin and the diode close to the driver/gate node to minimize parasitic inductance and to ensure the intended directional behaviour is not altered by trace impedance. The 100 kΩ discharge resistor should also be placed close to the MOSFET gate-source pins.

MOSFET selection: IPP030N10NF2SAKMA1 (N-channel)

The power stage uses IPP030N10NF2SAKMA1 N-channel MOSFETs. The selection criteria and how the device fits the design goals are:

- **Low RDS(on):** A low on-resistance minimizes conduction losses during the MOSFET conduction interval, directly improving converter efficiency — a primary objective in a 400 W synchronous buck.
- **High current capability:** The MOSFET's continuous and peak current ratings provide margin for the expected maximum current from the PV source and for transient events.
- **Sufficient voltage rating (100 V):** A large voltage margin above the PV open-circuit voltage protects against PV transients, spikes during switching events, and makes the design more robust.

Schematic notes (as implemented)

- The schematic includes the **bootstrap diode and 2×100 nF bootstrap capacitor** between VB and VS as recommended for IR2104 high-side bias.
- Gate resistors (15 Ω), Schottky diodes (SS310) and 100 kΩ gate-to-source resistors are placed per the driver datasheet guidance and tuned with iterative simulations and oscilloscope measurements for the tradeoff between EMI, switching loss and safety in an educational lab context.

3.2. Auxiliary Voltage Levels

The converter requires multiple auxiliary voltage rails to power different subsystems. Specifically, the IR2104 gate driver requires a supply voltage of at least 10 V, while both the ESP32 microcontroller and INA228 power sensors operate at 3.3 V logic levels. The primary input to the control board is the 5 V supply provided via the USB Type-C port, capable of delivering up to approximately 2 A in its default configuration.

To generate the required voltage levels, two auxiliary converters were implemented:

- A step-up (boost) converter to produce 13 V for the gate driver circuit.
- A low-dropout (LDO) linear regulator to produce 3.3 V for the ESP32 and sensors.

Because the voltage difference between 5 V and 3.3 V is relatively small, a linear regulator was chosen for simplicity and low noise. The AP2114H-3.3TRG1 LDO was selected for this purpose, offering sufficient current capability and stable operation. Although linear regulators are generally less efficient for large voltage drops, in this case the efficiency loss is minimal due to the small voltage difference and moderate current demand.

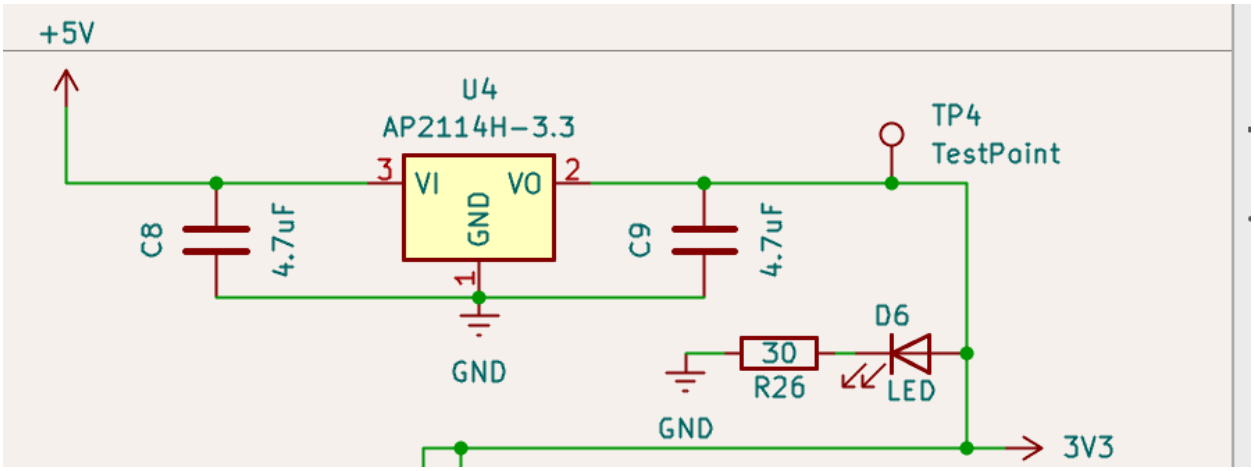


Figure 3: LDO 5v to 3.3V Conversion

For the gate driver, which requires a higher supply voltage, a boost converter based on the AP3012KTR-G1 IC was implemented to step the 5 V USB input up to 13 V. According to the datasheet, the output voltage of this converter can be set by selecting appropriate inductor and feedback resistor values. The target of 13 V was chosen because it comfortably meets the IR2104's recommended operating range of 10–20 V, and also utilized components already available from previous designs.

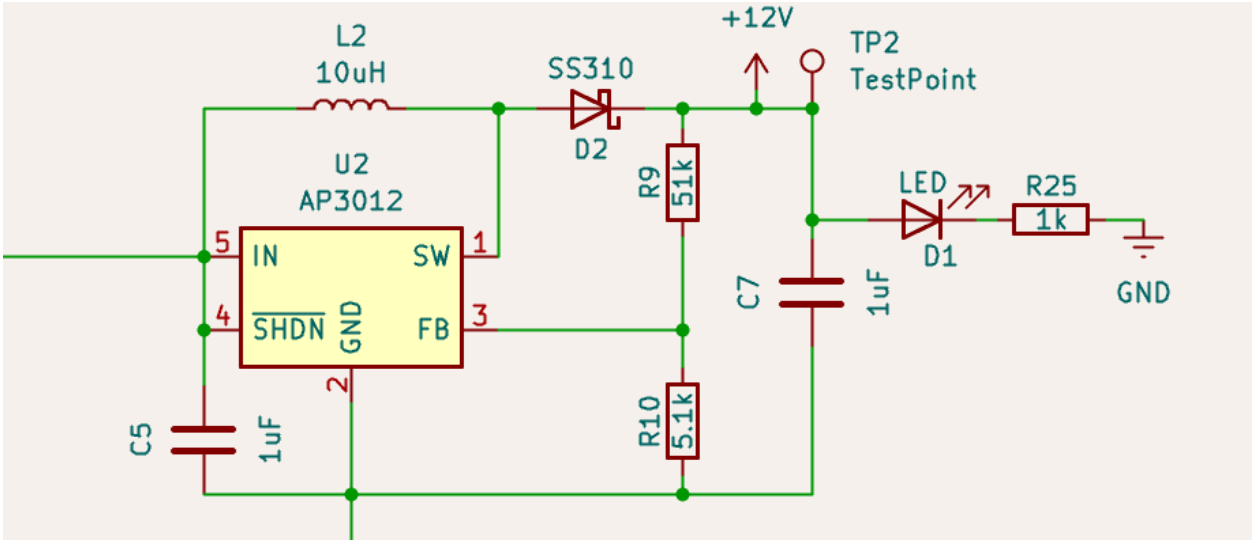


Figure 4: Boost Converter for 5v to 13 V conversion

This configuration ensures reliable operation of all control and driver circuits while minimizing complexity and component count. The 13 V rail supplies the IR2104 high-side and low-side driver sections, whereas the 3.3 V rail powers all digital and sensing circuitry on the board.

Additionally, a direct 5 V supply connection has been integrated to power the entire circuit when no USB cable is connected. This ensures continuous operation and flexibility in power sourcing.

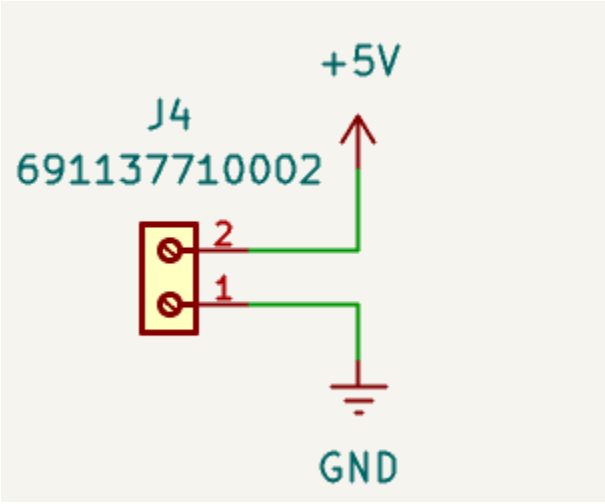


Figure 5: External 5V

3.3. Power Sensors: INA228

Two INA228 Datasheet digital power monitors are used as the backbone of the measurement system: one on the input (PV side) and one on the output (load side). These chips provide the precise, high-resolution voltage and current measurements that the MPPT algorithm relies on, and they also compute power/energy/charge internally so the MCU can read stable, filtered results

3.3.1. INA228 Overview

The INA228 is a high-precision digital power monitor with an I2C interface, capable of measuring bus voltage, shunt voltage, current, and power. Also it has internal Ultra-high precision ADC (20-bit) and very low offset/gain error make the INA228 suitable for both small and large current sensing with high accuracy.
Wide common-mode range (–0.3 V to +85 V) and a bus voltage input up to 85 V make it appropriate for typical PV and DC-DC converter topologies.
It supports programmable conversion times and averaging, letting you trade measurement speed for noise performance useful when you want either fast MPPT updates or very low noise measurements.

The devices' address pins (A0/A1) were set according to the datasheet to avoid bus collisions; the INA228 supports multiple pin-selectable addresses for multi-device systems

- **Input INA228:** I2C Address 1000000 (binary)
- **Output INA228:** I2C Address 1000100 (binary)

3.3.3. Sense Resistor

A 0.01 Ohm sense resistor is used with both INA228 sensors for current measurement. The corresponding shunt calibration in the firmware has been adjusted accordingly, ensuring high precision readings. The INA228 allows differential full-scale shunt ranges (example values in the datasheet): **±163.84 mV** chosen **0.01 Ω** shunt:

- Full-scale current (±163.84 mV range):
  IFS=0.16384 V0.01 Ω=16.384 AIFS=0.01 Ω0.16384 V=16.384 A.

Which is suitable for our PV panel converter since our operating currents are below this.

For both sensors I implemented **input filtering** and a careful **Kelvin (4-wire) layout** to maximize measurement accuracy and immunity to switching noise:

- **Input filtering:** small passive filters were placed on the INA228 voltage inputs (shunt and bus lines) following the recommendations in the datasheet. These filters suppress high-frequency switching spikes from the power stage, reduce ADC sampling jitter, and improve reported power stability without slowing the effective measurement bandwidth needed by the MPPT loop.
- **Kelvin (4-wire) sense layout:** the shunt resistors are routed with dedicated Kelvin sense traces to the INA228, keeping the sense leads short and separate from the high-current return path. The INA228 devices are located close to their respective shunts, local decoupling is provided, and star grounding / appropriate return routing is used to avoid voltage drops and coupling from the high-current switching loop. These layout choices minimize error caused by trace resistance, ground bounce, and EMI from the switching stage.

The chosen shunt value (0.01 Ω) and the INA228 configuration were reflected in the firmware calibration constants so the readings report correct current, voltage and power. For firmware integration and calibration I used the **Adafruit INA228 Arduino library**, which provided convenient helper routines to set the SHUNT_CAL and read scaled voltage/current/power registers; this library was used as the basis for calibration, averaging configuration, and runtime measurements in the MPPT control loop.

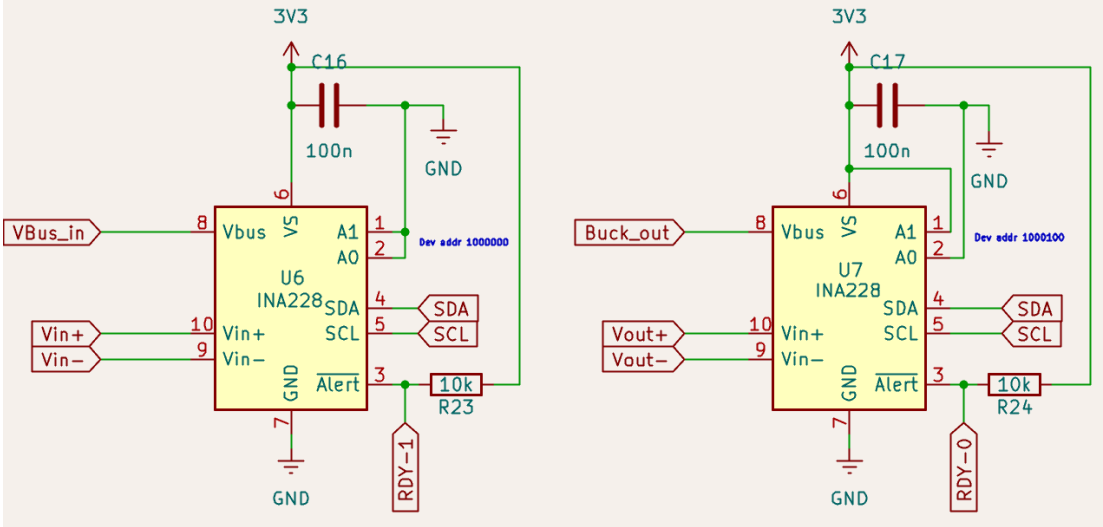


Figure5: Necessary INA228 Configurations (Input Side Left Output Right)

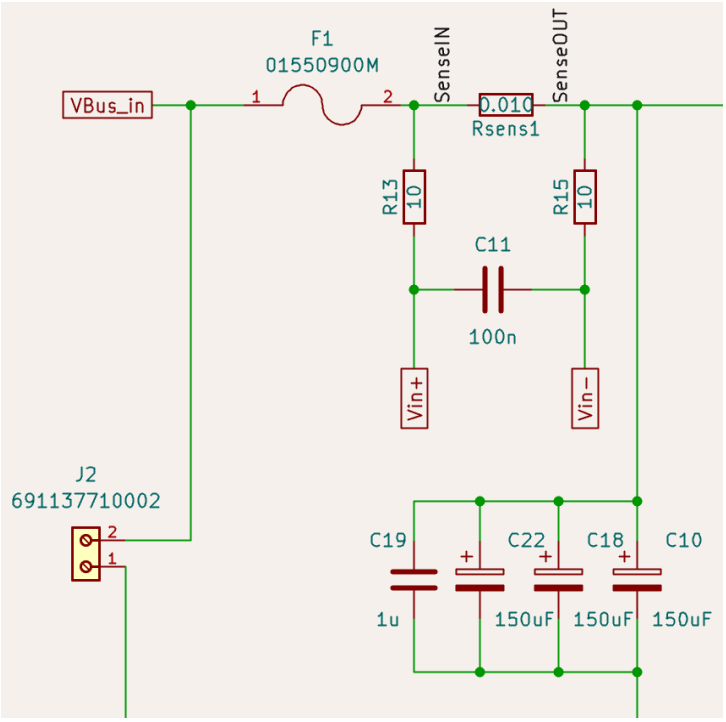


Figure6: Sense Resistor and Filter Network Schematics

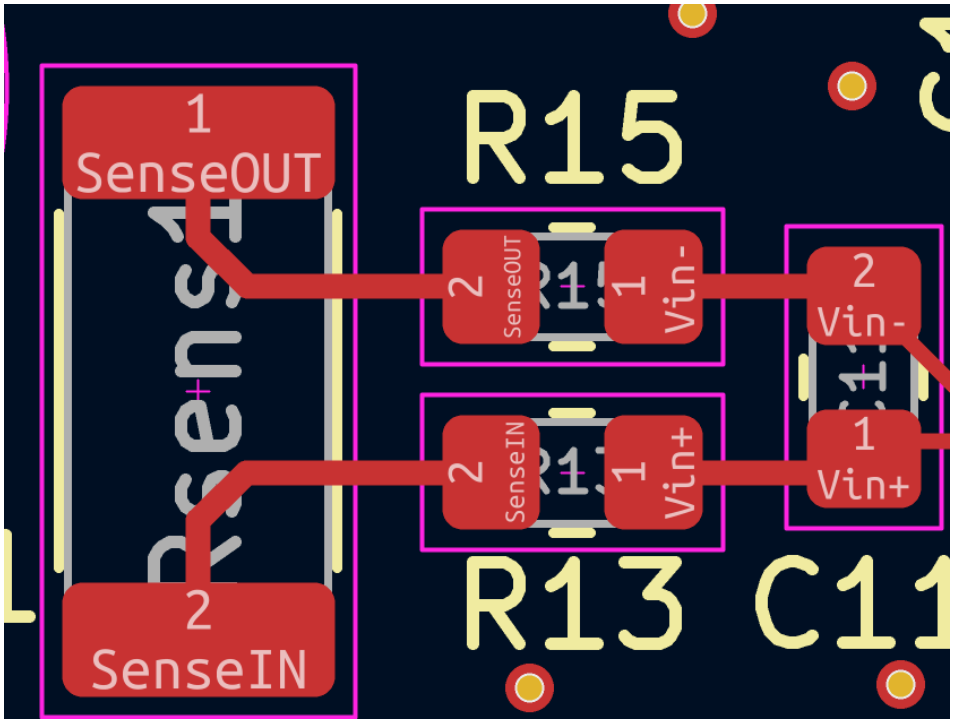


Figure7: Kelvin Traces

3.4. Microcontroller: ESP32 and USB Connection (USB-C receptacle + CH340C TX/RX)

The ESP32 serves as the central control unit for the MPPT synchronous buck converter: it executes the MPPT algorithm, reads the INA228 power sensors over I^2C, generates the PWM signals for the gate driver, hosts the web UI, and provides a USB serial programming/debug interface for students.

3.4.1 Key features of the ESP32 (used in this project)

- **Dual-core processor** : provides sufficient compute headroom to run the MPPT control loop, background telemetry tasks and a lightweight web server concurrently.
- **Wi-Fi and Bluetooth**: built-in 2.4 GHz Wi-Fi and Bluetooth enable the web interface and future wireless extensions (OTA updates, telemetry, remote control).
- **Hardware PWM (LEDC) and timers**: used to generate the synchronized gate-driver PWM with the chosen switching frequency (39 kHz) and to implement precise deadtime/timing in software.
- **I^2C interface**: used to communicate with the two INA228 sensors for voltage/current/power measurements. The ESP32's I^2C peripheral is configured for a robust bus speed that balances responsiveness and noise immunity.
- **UART interface(s)**: UART0 is used for programming and serial debugging via the USB-UART bridge (CH340C).
- **GPIOs, ADCs and peripheral flexibility**: ESP32's GPIO matrix and ADCs make it easy to integrate status LEDs, push-buttons (start/stop MPPT), and other experimental I/O for students.

3.4.2 USB-C receptacle and CH340C (USB → UART bridge)

To provide a familiar and easy programming experience for students, the board includes a **USB-C receptacle** wired as a power/data input and a **CH340C USB-to-UART** bridge that connects the PC USB port to the ESP32's UART0 (TX/RX).

- **USB-C receptacle (VBUS/CC/GND):** the USB-C connector is wired following the USB Type-C recommendations so the board can draw VBUS when plugged into a host/charger. The CC pin handling was implemented per the USB Type-C specification (pull-up / pull-down resistor configuration) so the source advertises a default current level; this lets the board reliably receive 5 V from typical USB Type-C hosts. Note that the USB Type-C specification defines discrete default current advertisement levels which are selected by appropriate CC resistor values. The board's CC termination was implemented in accordance with the spec and the component datasheets in our project, its 5V and 2A.

- **CH340C USB-UART bridge:** the CH340C is used to convert USB data to a TTL UART for the ESP32. The CH340C's TXD pin is connected to the ESP32 RX0 pin and CH340C RXD to ESP32 TX0 for programming and serial console access. Where auto-programming/reset functionality was desired, the CH340C control lines were wired to the ESP32's EN and IO0 lines following the standard USB-to-ESP32 auto-boot circuit patterns described in reference designs; otherwise manual reset/boot is used for flash mode with extra switch buttons. The CH340C device selection and wiring follow the CH340C datasheet recommendations for decoupling and USB line layout.
- **USB D+ / D−** were routed as a differential pair and length-matched to minimize skew equal trace lengths and controlled routing to reduce USB errors. Even on a two-layer board (without a dedicated controlled-impedance plane), matching skew and keeping the pair tightly coupled minimizes common-mode radiation and susceptibility.

3.4.3 Implementation notes and rationale

- **Powering the control electronics:** VBUS from the USB-C receptacle supplies the board's 5 V rail. That 5 V is used as the input to the AP3012 boost and the AP2114 3.3 V regulator (as described earlier). The USB-C CC resistor configuration was chosen so that common hosts/chargers present a reliable 5 V source; the design follows the USB Type-C and charger behavior guidance in the official spec and application notes.
- **Signal levels & isolation:** the ESP32 operates at 3.3 V logic. The CH340C and its PCB implementation were chosen/implemented so that UART lines are compatible with 3.3 V TTL levels.
- **Wiring for reliable programming:** TXD ↔ RX0 and RXD ↔ TX0 are used for the serial console and flashing. If implemented, the DTR/RTS handshake lines from the CH340C were routed to the ESP32 EN/IO0 pins with the recommended capacitor/resistor network so the host can toggle the ESP32 into the bootloader automatically (this follows common ESP32 dev-board reference designs). If the auto-boot circuit is not present, manual reset + IO0 pushbutton are used for entering the bootloader.
- The USB-C + CH340C arrangement gives students a plug-and-play experience: plug a USB cable, and with VSCode PlatformIO program/debug the ESP32. The serial console is also used by the firmware for realtime logs.

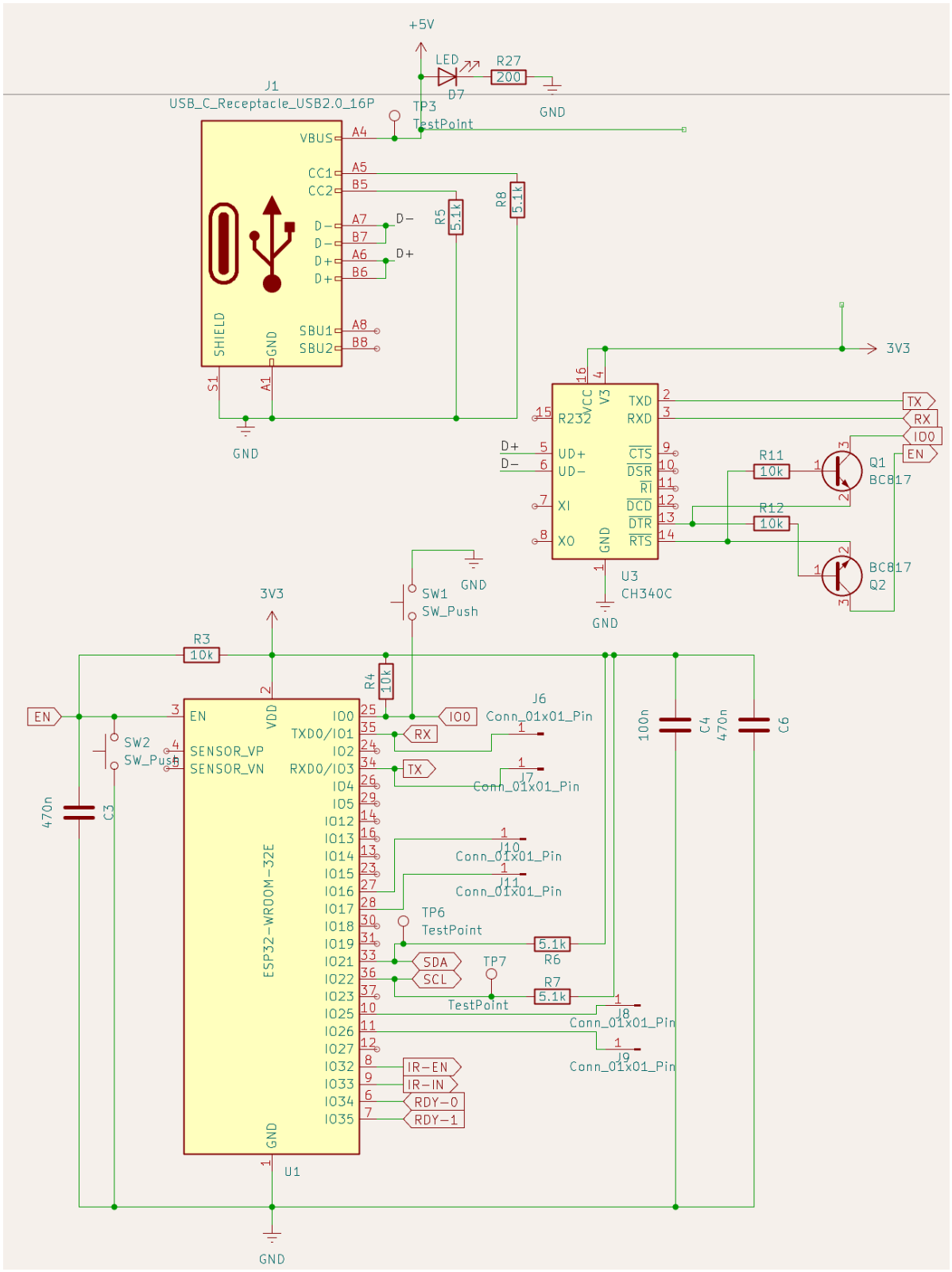


Figure 8: Schematics of the Receptacle/CH340C and ESP32 connections

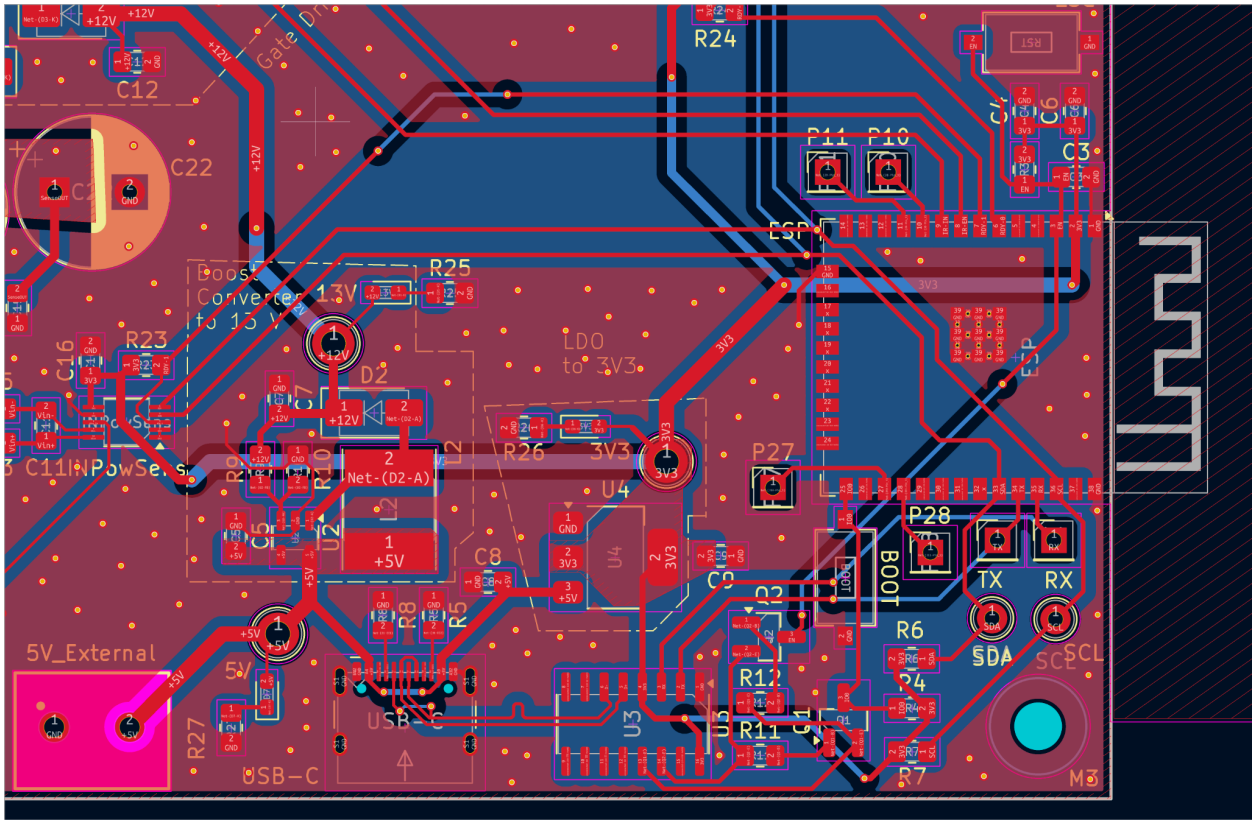


Figure 9: PCB Layout of the Receptacle/CH340C and ESP32 connections

3.4.2 General PCB Design and Implementation Analogy

The PCB layout was driven by three parallel objectives: electrical performance (signal integrity & power delivery), measurement accuracy (sensing & noise reduction), and teachability / accessibility for laboratory use. The board was fabricated and tested successfully; screenshots of the full final schematics and layout are included at appendix to illustrate the implementation choices described below.

Layout objectives

- Keep high-current power paths short and wide to minimize resistive losses and loop inductance.
- Keep sensitive measurement and logic traces (INA228 sense, I²C, ESP32 signals) physically separated from the switching loops and noisy nodes.
- Place critical passive components (bootstrap cap, input/output bulk caps, shunt resistor, gate network) as close as possible to the ICs/MOSFETs they serve.
- Provide convenient access for debugging and education: test points, LEDs, user connectors, and clear silkscreen labels.

- A **2-layer PCB** and a compact **≤10 × 10 cm** form factor were chosen to reduce fabrication cost while still satisfying layout constraints for 300 W. Small design also improved signal integrity by reducing trace length and loop areas.

PCB Layout Strategy and Design Considerations

- The **power stage** (MOSFETs, inductor, input/output capacitors, shunt) is grouped tightly to minimize the high-current loop area. Input capacitors are placed close to the MOSFET drains and PV input, and the output caps are located near the output node and sense resistor. This reduces ripple and radiated EMI and improves transient response.
- The **gate driver** (IR2104) and its bootstrap diode/capacitor are placed adjacent to the MOSFETs and the switching node (VS) so the bootstrap loop is minimal. The series gate resistors and the Schottky diodes for the asymmetric gate network are placed as close to the MOSFET gate pins as possible to control parasitic inductance and guarantee the intended gate timing.
- The **measurement subsystem** (INA228s and 0.01 Ω shunts) is located with short Kelvin sense traces; the INA228 devices are placed very close to their respective shunts to reduce error from PCB trace resistance and switching noise coupling.
- The **control & communication area** (ESP32, USB-C, CH340C, LDO/boost converters) is partitioned away from the power stage. This physical separation reduces coupling of switching noise into digital rails and analog measurements.
- High-current nets use **wide traces and copper pours** rather than thin traces; this lowers I^2R loss and acts as a heatsink for dissipating MOSFET/inductor losses. Where extra carrying capacity was needed, two layer pours and stitched vias are used to increase cross-sectional area.
- The design intentionally uses **a two-layer board** for cost effectiveness. I tried to use the top layer for power and signal routing and bottom layer for ground fills mainly so I tried to minimize ground layer cuts as much as possible. Where the design required crossing nets, bottom layer routing was used sparingly to preserve continuous copper pours for ground and thermal conduction on the bottom plane. After routing, the board was filled with ground pours across both layers to provide broad low-impedance reference planes and thermal mass.
- Both the top and bottom PCB layers were used as **copper fills for high-current paths**, significantly increasing the current-carrying capability and reducing resistive losses. These top and bottom copper areas were **interconnected using multiple stitched vias**, effectively creating a low-impedance, parallel current path between layers.
- **Via stitching** (multiple vias across the copper fills) was applied between top and bottom copper areas to reduce thermal resistance, increase current carrying capability, and lower the impedance of the return paths. Stitching also helps

reduce the effective loop area for switching returns and mitigates parasitic antenna behaviour.
- Large copper areas under and around the MOSFETs act as heatsinks; thermal vias and stitched copper assist in moving heat to the bottom layer. Components with significant loss (MOSFETs, shunt resistors) have generous copper planes and thermal relief to avoid hot spots.
- Decoupling capacitors are placed close to IC power pins (ESP32, CH340C, INA228, regulator inputs/outputs). Small MLCCs (for HF decoupling) are in parallel with bulk aluminum-polymer capacitors (for ripple current handling) at both input and output rails.
- Critical digital control traces (I^2C, UART) are kept short and routed away from the switching node and inductor to reduce injected noise; I^2C pull-ups are located close to the master (ESP32) to maintain signal integrity.

Testability and usability for teaching

- Multiple **test points** and labelled **LEDs** were added for the 5 V, 13 V and 3.3 V rails and for critical signals (GND, SDA, SCL, gate driver input / PWM, VIN, VOUT, TX/RX). These access points simplify oscilloscope probing, measurement and debugging during lab demonstrations.
- **Pin headers** expose the ESP32 TX/RX, 4 general-purpose IO pins and other useful signals so students can attach probes, sensors, or expansion modules without desoldering.
- **Tactile switches** were provided for EN and IO0 to allow manual reset and manual bootloader entry for flashing. The serial-bridge (CH340C) connections were routed for standard auto-boot behavior where feasible.
- Clear **silkscreen labelling** and simple visual icons were added so students can quickly identify connectors, test points and polarity. M3 mounting holes are placed at the corners for classroom fixtures or enclosures.

Overall, the PCB was laid out with standard power-electronics best practices in mind: minimize switching loop areas, place decoupling and bootstrap components close to their pins, separate noisy and sensitive domains, use wide copper for current carrying nets, and provide generous testability and clear lab-friendly labeling. These implementation choices support both robust operation in the lab and an accessible learning experience for students experimenting with MPPT and power electronics.

4. Firmware Development

The firmware implements real-time MPPT control, sensor acquisition, logging, and a web-based user interface. It is written in C++ for the ESP32 using VSCode PlatformIO/Arduino framework and is divided into modular components (filesystem, measurement, MPPT logic, PWM control, webserver, telemetry and logging). The firmware is organized into separate modules to improve readability, enable reuse in teaching, and simplify testing.

4.1. Development Environment

The firmware for the MPPT Synchronous Buck Converter was developed using the **PlatformIO** environment within **Visual Studio Code**. PlatformIO provides integrated project management, dependency handling, and an efficient build system, simplifying embedded software development and deployment.

Target Board: ESP32

Framework: Arduino (selected for its robust ecosystem, ease of use, and availability of well-documented libraries)

Programming and Debugging Interface:

- USB connection via **CH340C** USB-to-UART bridge
- Serial monitor speed: **115200 baud**

File System: LittleFS was used for storing configuration files, calibration data, and web assets (HTML, CSS, JS) used in the onboard web interface.

4.2. Dependencies and Libraries

The firmware utilizes several open-source libraries to enable key system functions such as web hosting, sensor data acquisition, JSON handling, and over-the-air updates.

Library / Repository	Functionality
`ESPAsyncWebServer`	Asynchronous web server enabling a responsive local web UI for system monitoring and control.
`WebSockets`	Enables real-time data exchange between the ESP32 and the browser interface.

`ArduinoJson`	Used for structured data serialization/deserialization (sensor readings, configuration data).
`Adafruit_INA228`	Interface library for current, voltage, and power measurements using the INA228 sensor; used for calibration and live telemetry.
`ElegantOTA`	Provides a web-based OTA (Over-The-Air) update mechanism for easy firmware updates without physical connection.

Additional Build Flags:
`-DELEGANTOTA_USE_ASYNC_WEBSERVER=1`

> Enables compatibility between ElegantOTA and the asynchronous web server framework.

4.2. Software Architecture

The firmware is structured into three main folders:

- **src folder:** Contains .cpp files implementing the core functionalities, including MPPT algorithm, sensor readings, and buck converter control and also main.
- **include folder:** Contains .h header files defining classes, functions, and global variables.
- **data folder:** Stores files related to the web server, such as HTML, CSS, and JavaScript files for the user interface.

The firmware is structured into the following modules:

- **Main (`main.cpp`)** — program entry, system initialization, and top-level loop. Spawns tasks (FreeRTOS cores) and calls into module update functions on a fixed cadence. The system utilizes multiple cores to ensure synchronous operation, preventing mutual interference and ensuring all executions are completed independently.
- **Filesystem (`edugrid_filesystem.*`)** — wrapper around LittleFS that handles persistent configuration (Wi-Fi credentials, log name) and CSV logging.
- **Web server (`edugrid_webserver.*` + `data/www/`)** — Async web server and websocket handler; serves the HTML/CSS/JS UI from LittleFS and provides JSON endpoints for IV data and control.
- **Measurement (`edugrid_measurement.*`)** — INA228 configuration, sensor reads, zero-offset calibration, and filtering/zero-deadband logic.

- **PWM control (`edugrid_pwm_control.*`)** — LEDC PWM setup, duty management, manual-ramp functionality and safety limiting for the converter duty cycle.
- **MPPT algorithm (`edugrid_mpp_algorithm.*`)** — MPPT state machine including AUTO tracking and IV sweep mode; aligns its cadence to the INA228 averaging window.
- **Telemetry / Logging (`edugrid_telemetry.*`, `edugrid_logging.*`)** — runtime telemetry output (serial) and CSV persistence of operating data for offline analysis.

  `setup()` initializes hardware, mounts LittleFS, configures INA228 sensors, boots Wi-Fi and the webserver, and initializes PWM and MPPT modules. The top-level `loop()` runs at a fixed cadence (defined by `TASK_LOOP_INTERVAL_MS`) and performs the following actions: read sensors, update MPPT state machine (AUTO / MANUAL / IV_SWEEP), service manual ramp requests, log data, and handle telemetry. The firmware uses FreeRTOS primitives on the ESP32: time-critical or periodic computations are split into tasks where appropriate to ensure deterministic sampling cadence and to keep the webserver responsive during heavy operations. Whole firmware code is commented heavily to be able to understand as much as possible for another developer.

4.3. Measurement module — INA228 usage & calibration

Files reference: `include/edugrid_measurement.h`,`src/edugrid_measurement.cpp`

- I²C addresses: `0b1000000` (PV), `0b1000100` (LOAD).
- INA228 Adafruit library is used and its configuration snippets is used such as;
- Shunt resistor: `0.01 Ω`. SHUNT_CAL at `_configureInaDevice()`
- Averaging and timing (INA_STEP_PERIOD_MS, INA_AVG_SAMPLES, INA_CONV_US) is used

The measurement module uses two INA228 devices on I²C (addresses 0x40/0x44 — documented in the design section). Adafruit library is used extensively for easier setup. Each INA228 is configured with the matching full-scale shunt range and averaging settings so that the measurement cadence (INA_STEP_PERIOD_MS) aligns with the MPPT update period. A 0.01 Ω shunt is used on both sides; the firmware executes a one-time `calibrateZeroOffsets()` routine at boot (300 samples by default) to capture and subtract zero offsets. The Adafruit INA228 library is used to simplify register setup and runtime reads. The most important adjustment is the step period of calculation which is two INA conversions (shunt + bus) at **1,052 µs** each gives **2 × 1,052 µs = 2,104 µs per sample**; with **128 averaging samples** that becomes **2,104 µs × 128 = 269,312 µs ≈ 270 ms** after millisecond rounding. Adding the configured **120 ms extra settle time** yields **270**

ms + 120 ms = 390 ms, which is stored as `INA_STEP_PERIOD_MS` and used by the MPPT logic—so an automatic duty-step change happens every **390 ms**.

4.4. MPPT algorithm and IV sweep

4.4. MPPT Algorithm and IV Sweep

The firmware supports three operational modes for controlling and characterizing the PV + buck system:

- **AUTO (continuous MPPT):** a real-time tracking mode that continuously searches for and maintains operation at the PV maximum power point.
- **IV_SWEEP:** a diagnostic/data-collection mode that steps the converter through the duty-cycle range to capture a full I–V and P–V dataset for visualization and analysis.
- **MANUAL:** User just adjusts the duty cycle with the slider on the webUI slider has a 40ms throttle delay to avoid instantaneous duty changes it goes smoothly and continuously to the set value.

Control decisions are synchronized with the INA228 measurement cadence (averaging/conversion time). Aligning control updates with the measurement window reduces noise-induced missteps and ensures each power estimate is based on a stable average.

AUTO — Perturb & Observe (P&O)

The AUTO mode implements a **Perturb & Observe (P&O)** style hill-climb algorithm chosen for its simplicity and robustness in an educational setting. Key behavioral points:

- The algorithm perturbs the PWM duty by a small step (positive or negative), measures the resulting change in input power, and keeps the perturbation direction if power improves; otherwise it reverses direction.
- Duty updates are rate-limited (software slew) to avoid abrupt transients and to give the power stage time to settle between steps. This also reduces overshoot and oscillation around the MPP.
- P&O decision timing is synchronized with INA228 averaging so that each power comparison uses an averaged power value, improving stability in the presence of switching noise.
- Safety bounds clamp commanded duty between configured `PWM_MIN_DUTY_PCT` and `PWM_MAX_DUTY_PCT` to prevent the converter from commanding unsafe operating points.

IV_SWEEP — Full Duty Range Characterization

IV_SWEEP is intended for **visualization and teaching** rather than live tracking. Its behavior:

- The firmware steps the PWM duty through a configured range (in this project: **5% → 95% duty**) in fixed increments (`IV_SWEEP_STEP_PCT`) and at each step waits long enough for the INA228 averaging window and the power stage settling time to produce a stable measurement.
- For each step the firmware records averaged voltage and current samples and computes instantaneous power; all points are collected into an I–V and P–V dataset.
- When the sweep completes the full range, the dataset is served to the web interface where **Chart.js** renders the I–V and P–V curves and highlights the measured MPP. The sweep data can also be logged to LittleFS as a CSV for offline analysis.
- The sweep range intentionally avoids the extreme 0% and 100% duty positions (hence 5–95%) to prevent hard-saturation, long deadtime behavior, or out-of-range operating conditions in the power stage.
- **UI integration:** IV_SWEEP returns JSON arrays over the web API that the frontend (Chart.js) consumes to plot the curve in real time. During AUTO mode the UI displays the live operating point and a running IV snapshot so students can observe tracking behavior.

4.4.1. PWM Control

PWM generation uses the ESP32 LEDC peripheral configured at 39 kHz. The `edugrid_pwm_control` module exposes `setPWM()` and `requestManualTarget()` functions. Safety bounds (`PWM_MIN_DUTY_PCT` / `PWM_MAX_DUTY_PCT`) always clamp commanded duty to prevent accidental overdrive; a software ramping mechanism limits duty changes to `MANUAL_SLEW_STEP_PCT` per `MANUAL_SLEW_INTERVAL_MS` to avoid sudden transients during manual operation. All the timings and averaging samples are explained in the code as comments.

4.6 Web Server Implementation

The ESP32 hosts an asynchronous web server that provides a lightweight, responsive, and interactive user interface for controlling, monitoring, and logging the MPPT converter. The web subsystem is implemented using the **ESPAsyncWebServer** and **WebSockets** libraries, serving static assets from **LittleFS** and exposing REST-like endpoints and WebSocket channels for live telemetry and control.

4.6.1 Front-end technologies

- **HTML / CSS / JavaScript** — the UI is a single-page application served from LittleFS. All web assets are stored on LittleFS so the interface remains available without external servers. The firmware includes a small upload endpoint to update these assets for iterative UI improvements.
- **Chart.js** — used for interactive plotting of the live I–V and P–V curves and for visualizing the live operating points. **A large bright green dot is the current operating point while small faded green dots are the last 10 operating points.**
- **WebSockets** (or WebSocket-like real-time transport) — used to push live IV points and state updates from the ESP32 to the browser with minimal latency. The webpage subscribes to the WebSocket for live updates and appends incoming points to the Chart.js dataset. This allows smooth, low-latency plotting of IV points and an instantaneous display of the operating point.

4.6.2 Back-end architecture (ESP32)

- **Static file server:** HTML, JS, CSS, and Chart.js libraries are stored in LittleFS and served by the AsyncWebServer so the browser receives a self-contained UI after connecting to the ESP32.
- **WebSocket channel:** a persistent socket pushes live measurement samples and status messages to the UI in real time. This enables smooth chart updates without polling.
- **File upload / OTA:** the webserver handles file uploads to update web assets and also works alongside **ElegantOTA** to permit over-the-air firmware updates via the web UI.
- **Non-blocking model:** using the asynchronous server avoids blocking the main control loop when serving files or handling UI requests, preserving control timing and consistent MPPT cadence. Using `ESPAsyncWebServer` avoids blocking the MPPT loop while serving large resources or handling file transfers.
- **Synchronized sampling:** web updates use the same averaged measurement values produced by the INA228 calibration and averaging settings, avoiding noisy raw samples in plots.

4.6.3 Functionality exposed to users

- **Control:** start/stop MPPT, switch modes (AUTO / MANUAL / IV_SWEEP), manual duty control.

- **Real-time telemetry:** input/output voltage, input/output current, computed input/output power, duty cycle, mode state.
- **IV Curve visualization:** live plotting of sampled IV points during full IV sweep (IV_SWEEP mode). The maximum power point is highlighted on the I-V and P–V plots. Also at the bottom of the page after IV_Sweep is finished MPP values (V I W) are written. For live operating points a large bright green dot is the current operating point while small faded green dots are the last 10 operating points
- **Data logging & download:** CSV log files stored on LittleFS can be listed and downloaded via the UI for offline analysis.
- **Firmware updates:** ElegantOTA provides an OTA update page integrated into the web UI for flashing new firmware images.

5. System Validation

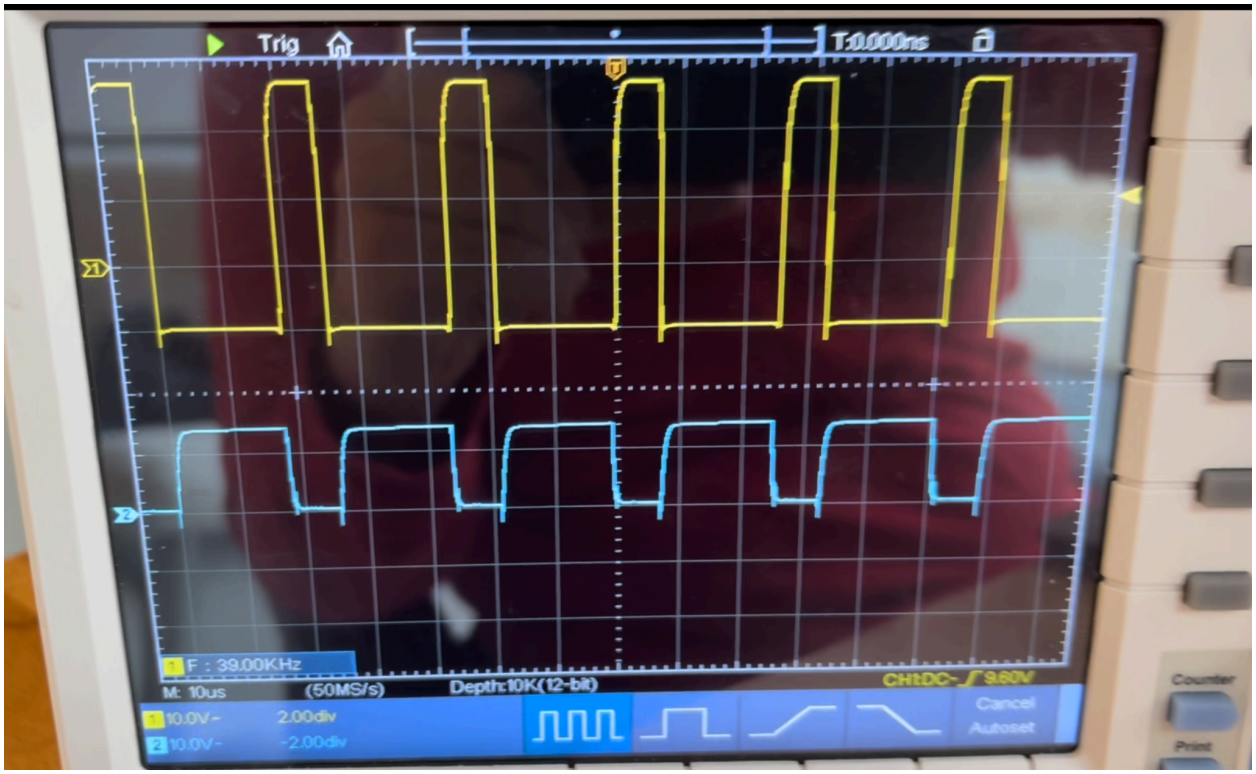
5.1. Initial Power-Up and Testing

Calibration Procedure:
At the first system startup (whether powered via the USB-C port or an external 5 V supply) no PV input or load should be connected. During this stage, the firmware automatically performs a zero-offset calibration routine to improve current and voltage measurement accuracy. This calibration captures samples of both input and load current measurements while assuming zero input conditions. The computed offsets are stored in RAM and continuously subtracted during runtime to compensate for ADC and amplifier bias errors.

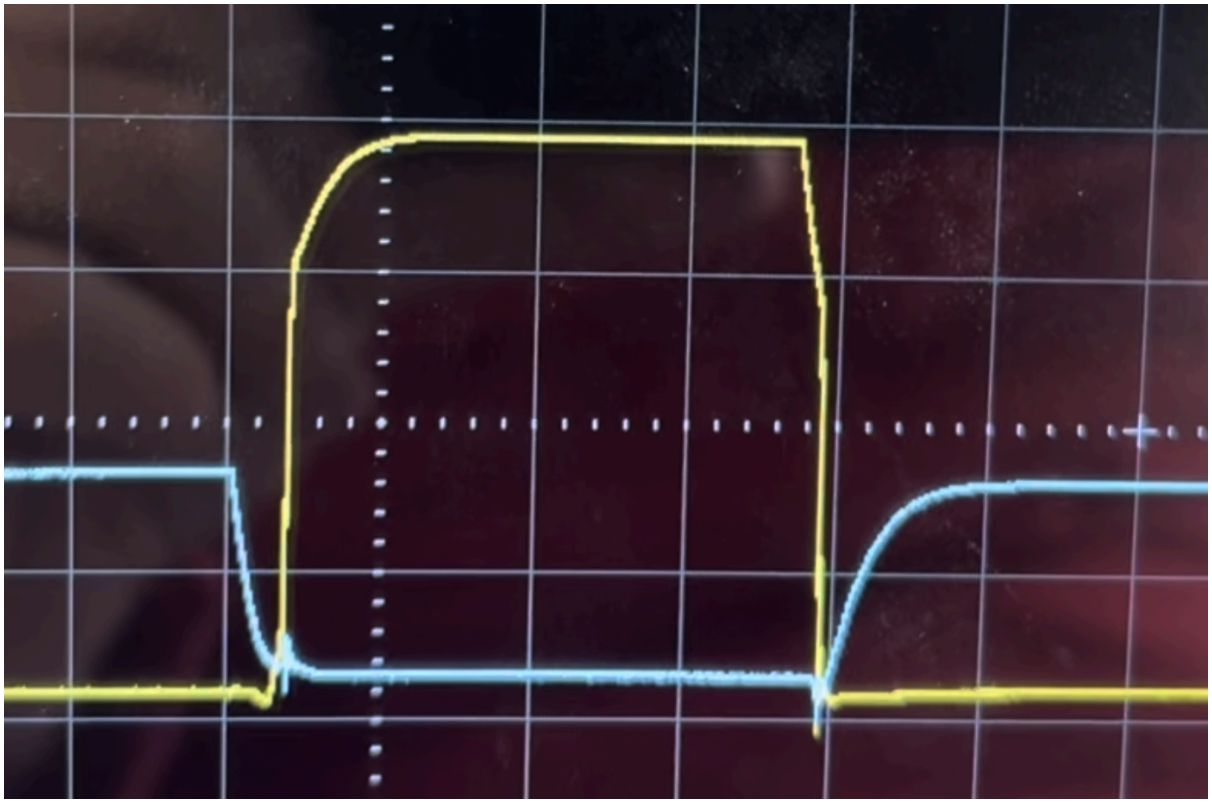
For any change in the system or environment one can make another new zero calibration offset through webUI as well but must be disconnected from the input PV panel. Performing this initial calibration ensures that subsequent measurements of PV voltage, current, and power are accurate and reliable during MPPT operation.

5.2. Oscilloscope Verification

I verified the gate waveforms on a digital oscilloscope: the high-side and low-side PWM signals are complementary, correctly timed and synchronous, and the enforced deadtime is sufficient to prevent shoot-through while remaining small enough to avoid excessive conduction loss. The scope capture shows the two gate channels and the interval between their falling/rising edges; this image is included as evidence of correct timing. For clarity the capture is annotated with channel labels (HS/LS), the timebase and voltage scale, and cursors that measure the deadtime between transitions.



Gate signals CH1 is top side mosfet Ch2 is low side mosfet



No Overshoot happens so deadtime works well

6. Performance Analysis

6.1. Efficiency Measurements

- **Input Power vs. Output Power:** Analysis of the converter's efficiency across different load conditions gives that around MPP region converter efficiency goes up to the %95.5 which can also be observed from the GUI. At very low duty (%5-10) the efficiency is low.

6.2. MPPT Tracking Performance

- **Tracking Speed:** At first power-up the firmware performs a timed measurement window and uses the result to decide when to change the PWM duty. I configured the INA measurement timing so that each automatic duty-step occurs every 390 ms. The math behind this is straightforward: the INA chip performs two conversions per sample (shunt and bus), each taking 1,052 µs, so one averaged sample requires `2 × 1,052 µs = 2,104 µs`. With 128 averaging samples the total conversion time becomes `2,104 µs × 128 = 269,312 µs` (rounded up to 270 ms for millisecond timing). To allow the power stage and sensors to settle after a duty change I add an extra 120 ms dwell, giving a final step period of 270 ms + 120 ms = 390 ms, which is stored in the firmware as `INA_STEP_PERIOD_MS`. In practice this means the MPPT algorithm will update the duty only when the 390 ms interval has elapsed, ensuring each change is based on a full averaged measurement plus a conservative settle margin. (If faster stepping is desired the averaging count or the extra-settle value can be reduced, or `INA_STEP_PERIOD_MS` can be overridden, but that trades off noise and settling safety.) So it takes 400 ms to change %1 duty cycle therefore it should be calculated with that information.
- **Tracking accuracy:** The MPPT reliably converges to the maximum power point within the resolution set by the controller — each duty change is **1%**, so the smallest controllable step is 1% of duty. In practice this means the algorithm finds the MPP to within the 1% duty-step granularity (plus measurement noise), which is more than sufficient for teaching and demonstration purposes.
- **Sensor precision:** Independent multimeter checks confirm the INA228's voltage and current readings are consistent and repeatable across the tested range. Measurements show no discernible bias or instability, so the INA228 provides reliably precise inputs for the MPPT algorithm.

7. Future Work and Enhancements

- **Advanced MPPT Algorithms:** Exploring more sophisticated MPPT algorithms for improved performance in dynamic conditions, which could be introduced in advanced modules for students.
- **Thermal Management:** Further optimization of thermal management for extended operation of higher power ratings. Also requires a more current capacity inductor as well (may require a custom winding)
- **Fault Detection and Protection:** Implementing additional fault detection and protection mechanisms. Using sensors and gate driver
- **Battery Charging Integration:** Extending the system to include battery charging capabilities.

8. Conclusion

The successful design and implementation of the 300W Synchronous Buck Converter for PV MPPT with ESP32 demonstrates a robust and efficient solution for solar power optimization. The end-to-end ownership of the project, encompassing hardware and firmware development, has resulted in a functional prototype with precise control and monitoring capabilities. Furthermore, the project's educational focus, with informative PCB visualizations and simplified control algorithms, makes it an excellent platform for students to learn about MPPT systems.

9. Appendix

9.1. Github Link (Hardware+Software Related) and Youtube Video

Github: https://github.com/UtkuDenizAltiok/EDUGrid_Final
Youtube Demo Video : https://youtu.be/TUlADqGg1uM

9.2. Schematic Diagrams

- Full Buck Converter Power Stage with IR2104 Gate Driver Circuit and Sense Network

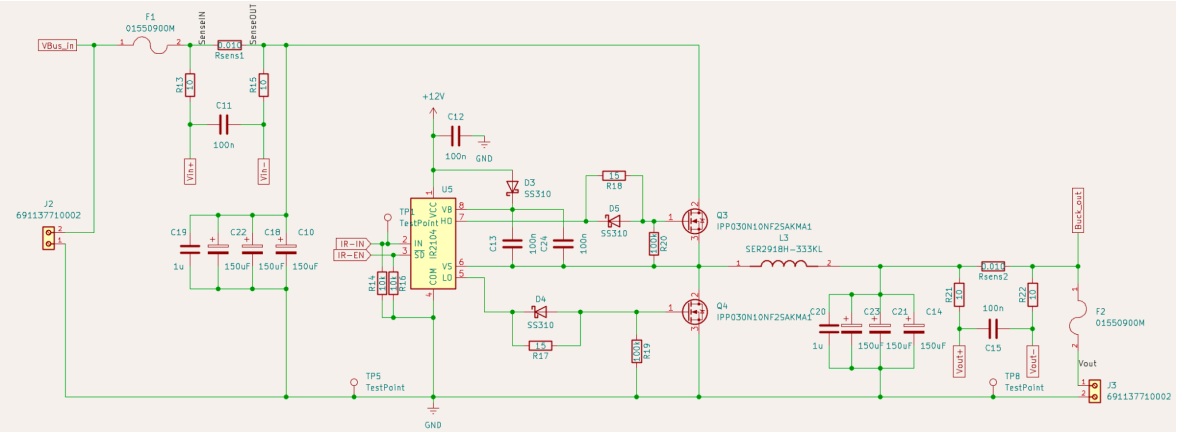


Figure 10: Schematics of Full Buck Converter Power Stage with IR2104 Gate Driver Circuit and Sense Network

- INA228 Sensor Integration Both Input and Output

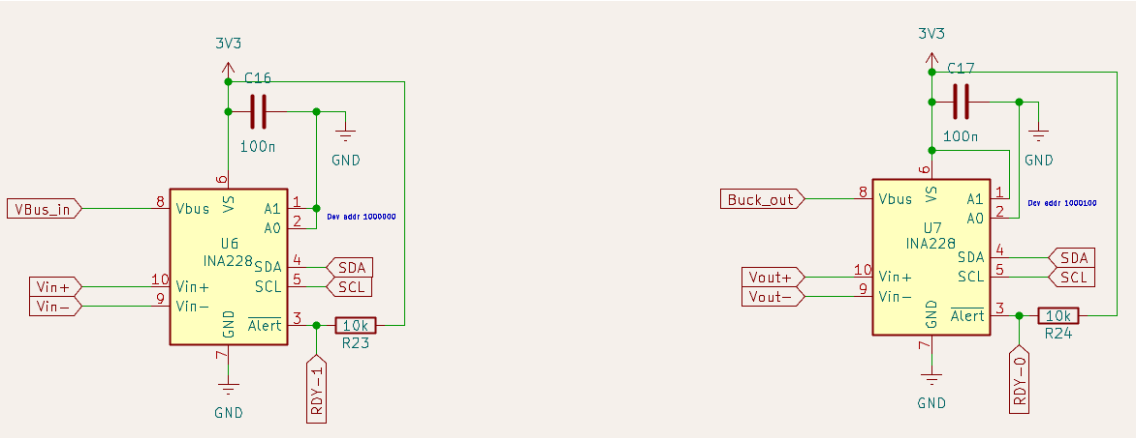


Figure 11: Schematics of INA228 Sensor Integration left input, right output

- Full ESP32 Connections and Auxiliary Voltages

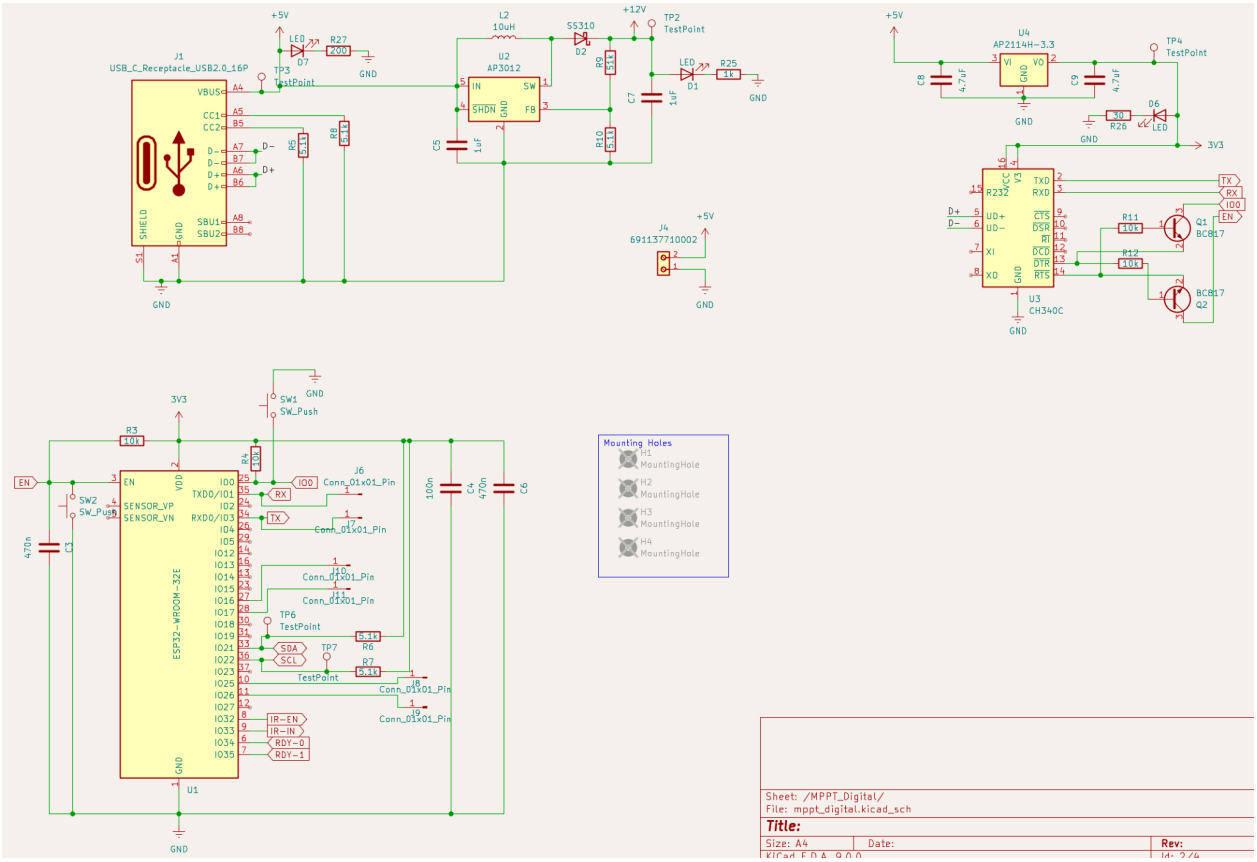


Figure 12: Full Schematics of ESP32 Connections and Auxiliary Voltages

9.3. PCB Layout

- Top Layer

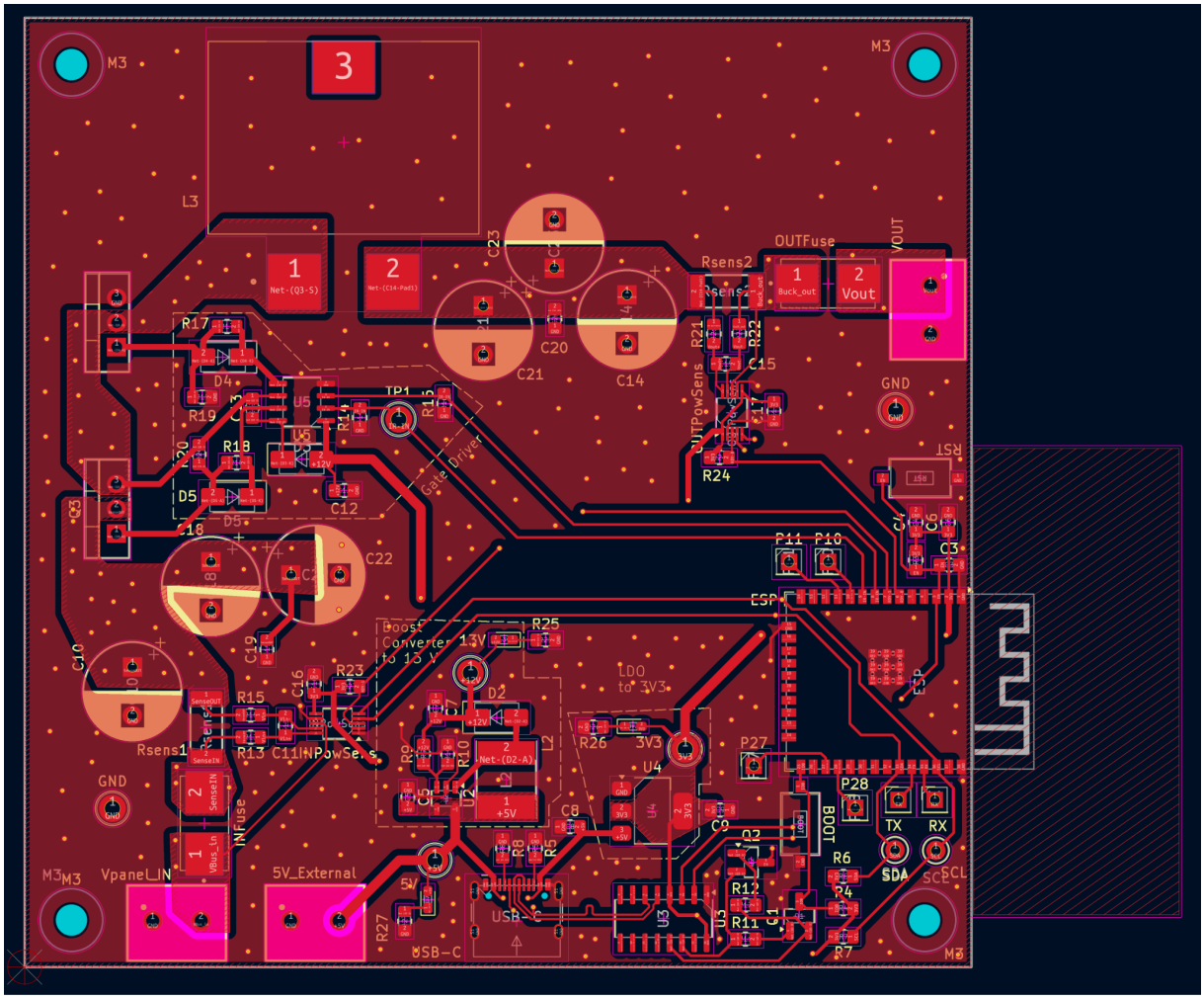


Figure 13: Only Top layer PCB layout from KiCad

- Bottom Layer

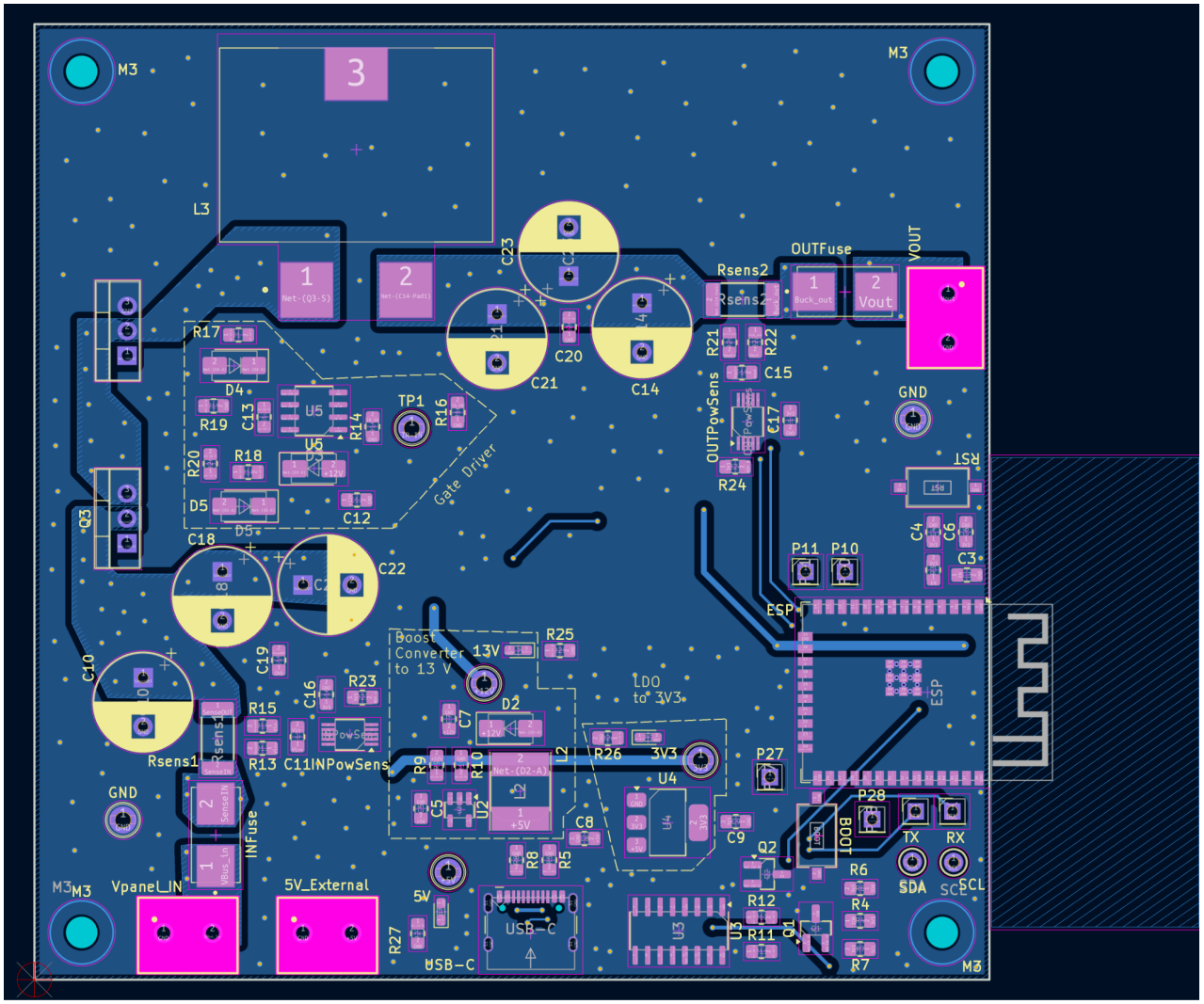


Figure 14: Only Bottom layer (mostly ground) PCB layout from KiCad

- Top+Bottom from Top Layer ViewF

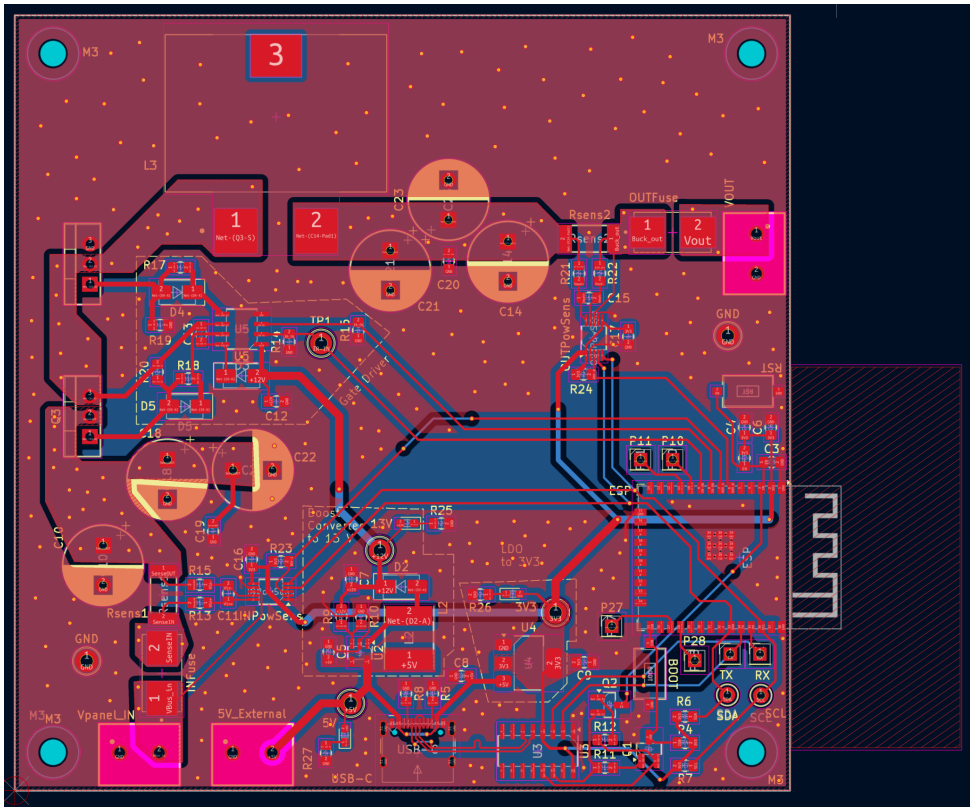


Figure 15: Both layers from top layer PCB layout from KiCad

- 3D View

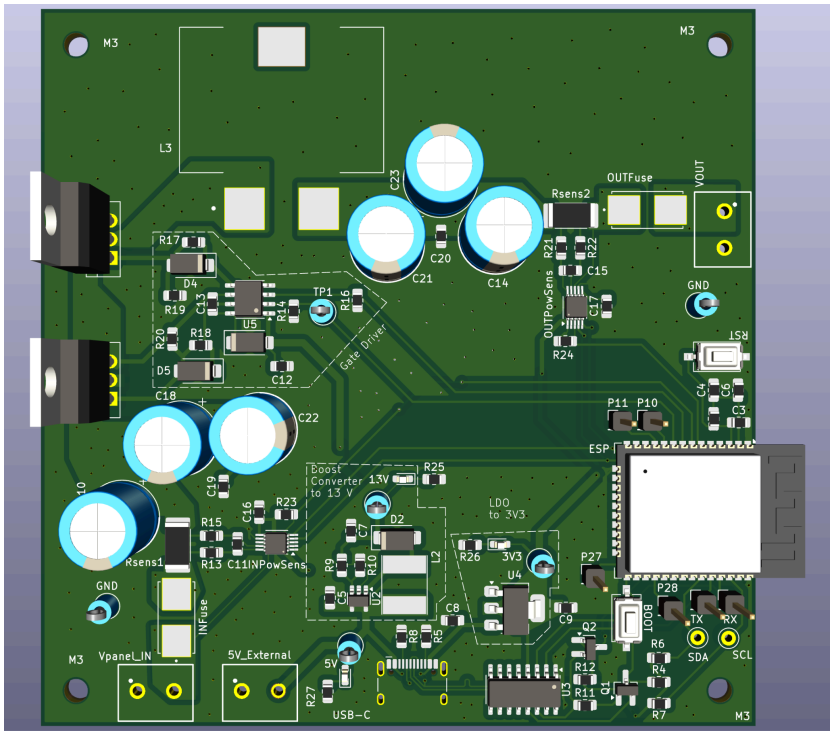


Figure 16: 3D PCB View from KiCad

9.4. Used Equipments and Hardware Pictures

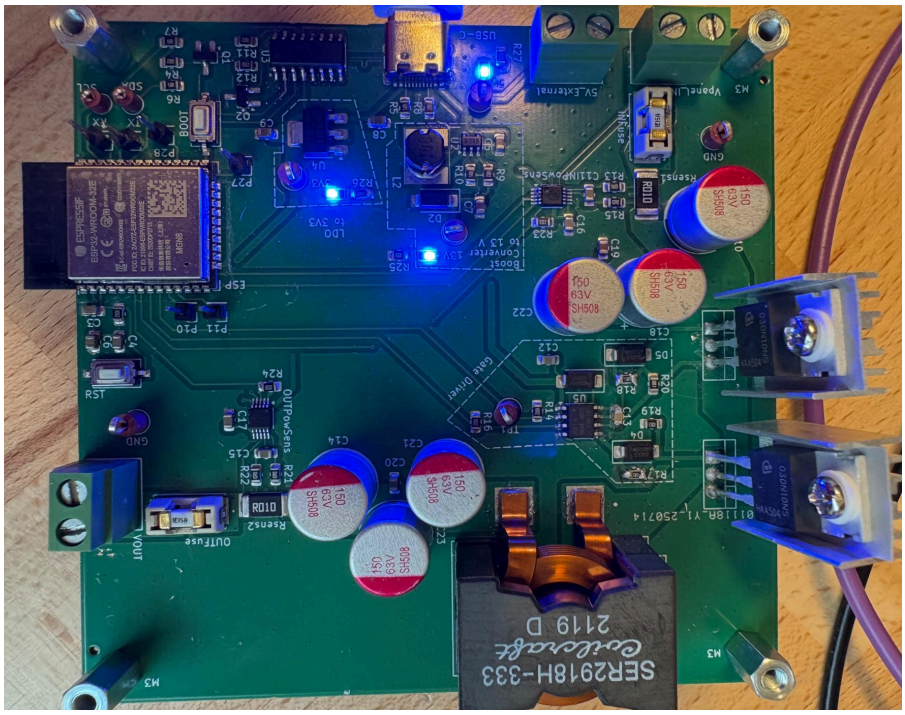


Figure 17: PCB Top View

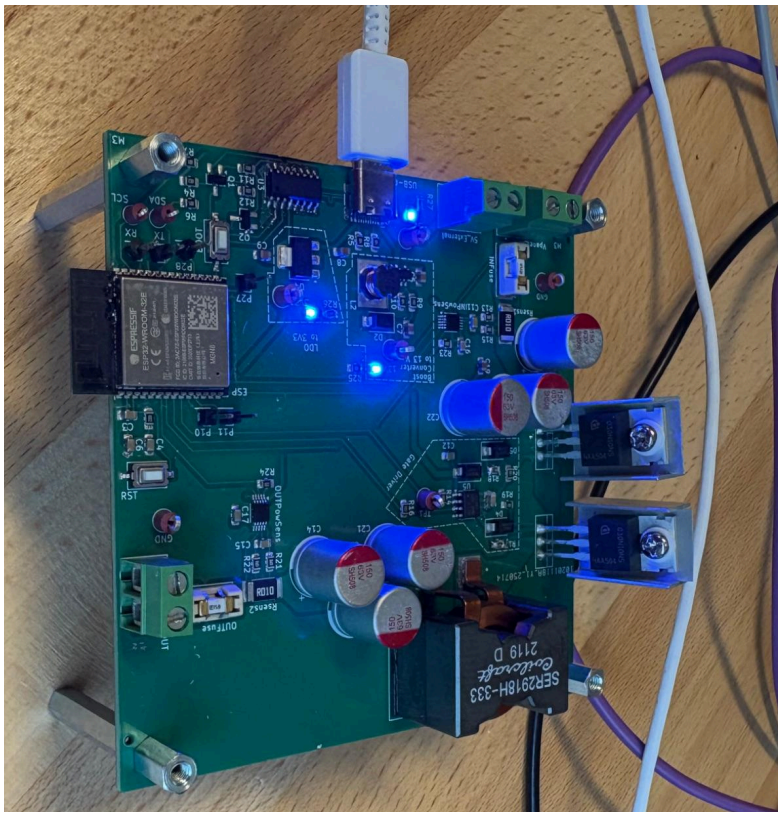


Figure 18: PCB Side View

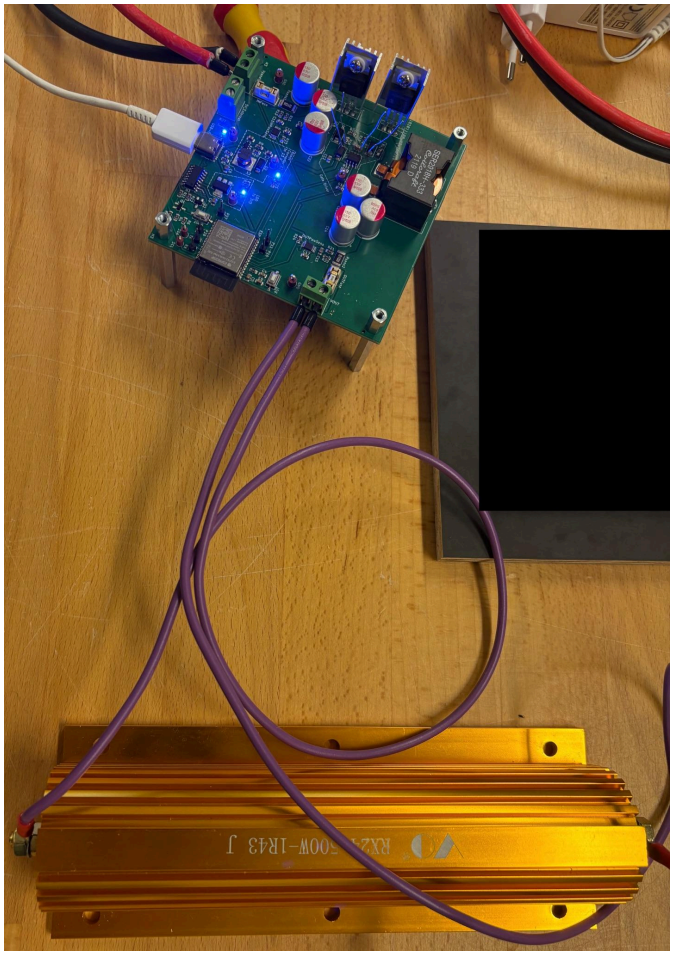


Figure 19: Connection with 1.43 Ohm 500W Rated Resistor

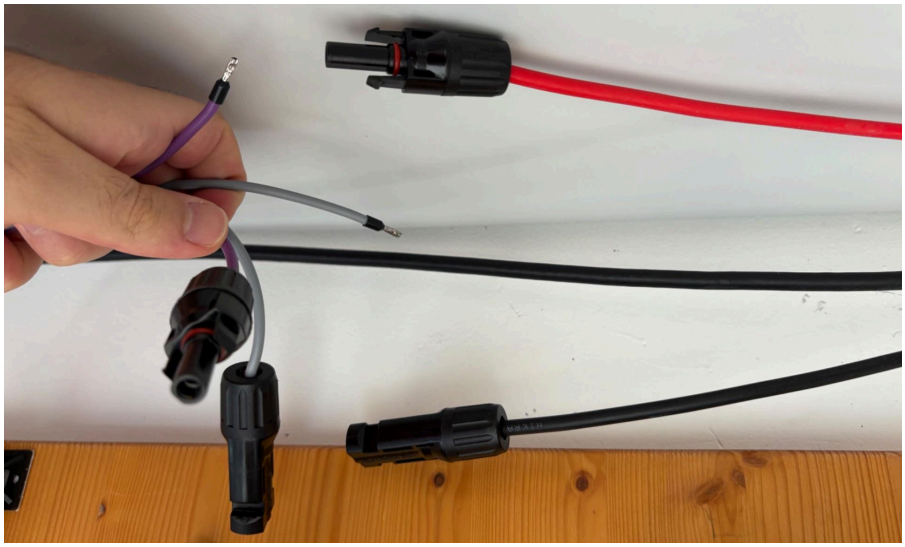


Figure 20: MC4 Cable Connector for high power PV Panels

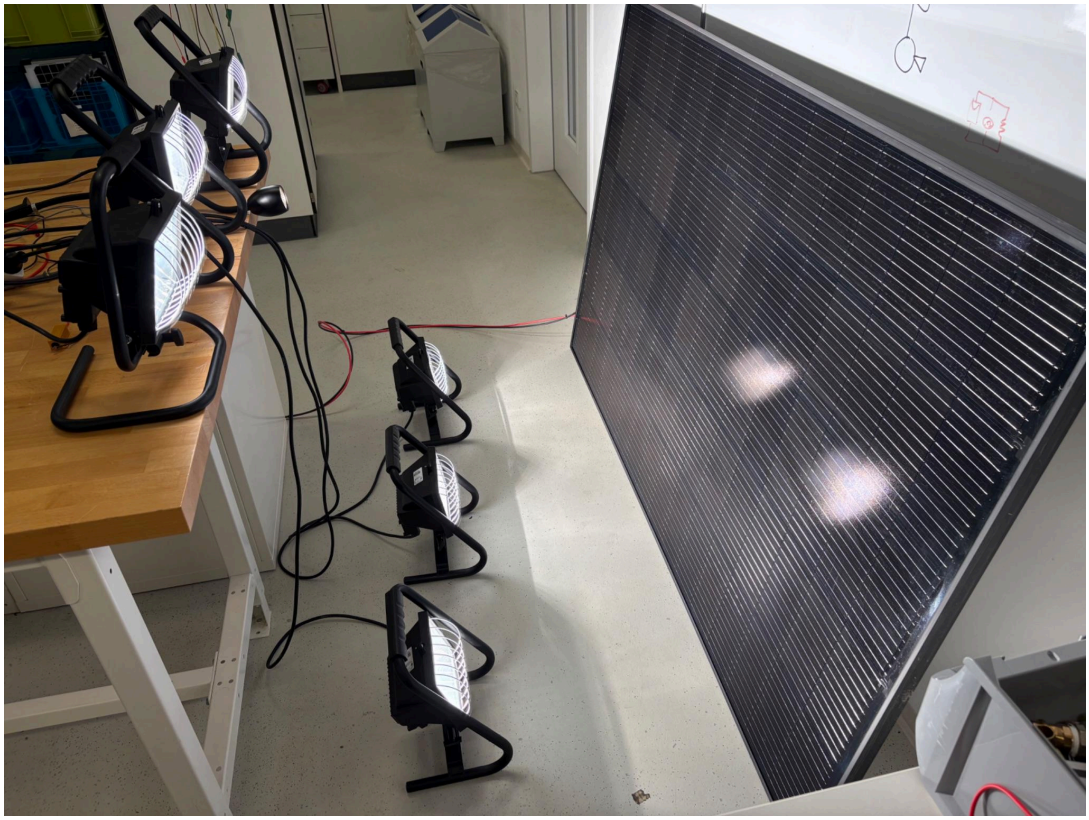


Figure 21: 400 Watt Solar Panel with 3kW (6×500W) Lightning Test Setup

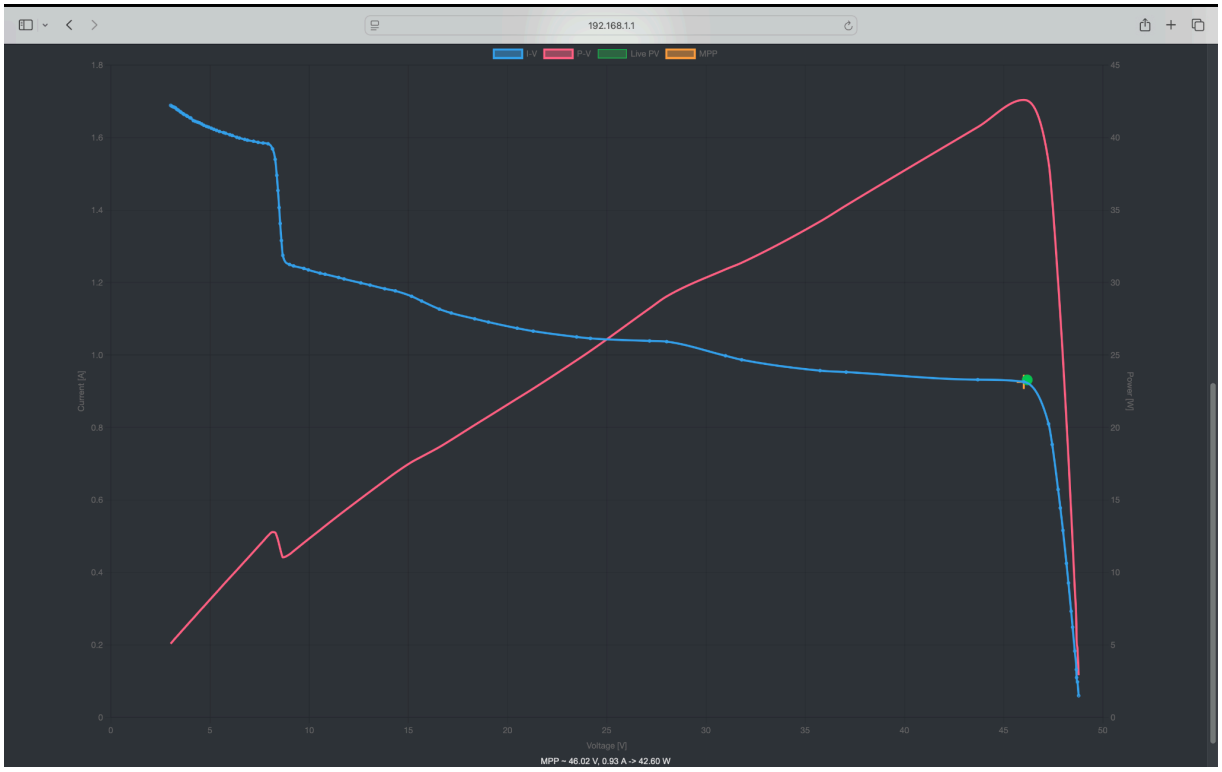


Figure 22: IV Curve while operating at max power point (its not completely flat since sunlight changed during IV Sweep)

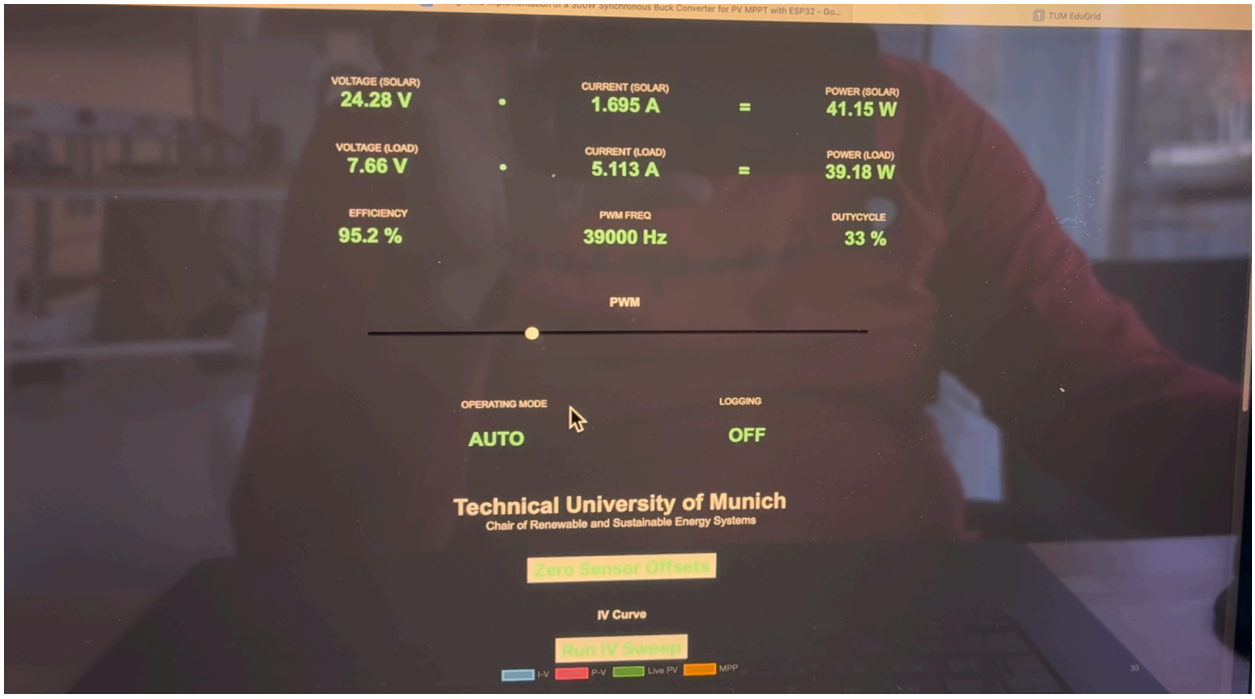


Figure 23: Web UI when operating at AUTO mode and found the Max Power Point

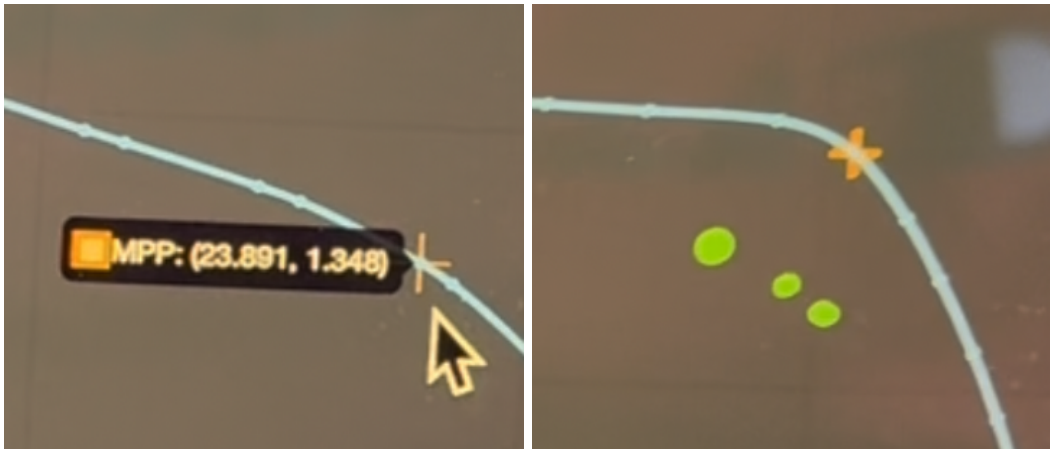


Figure 24: Marking the MPP Point and showing Live Operating Points