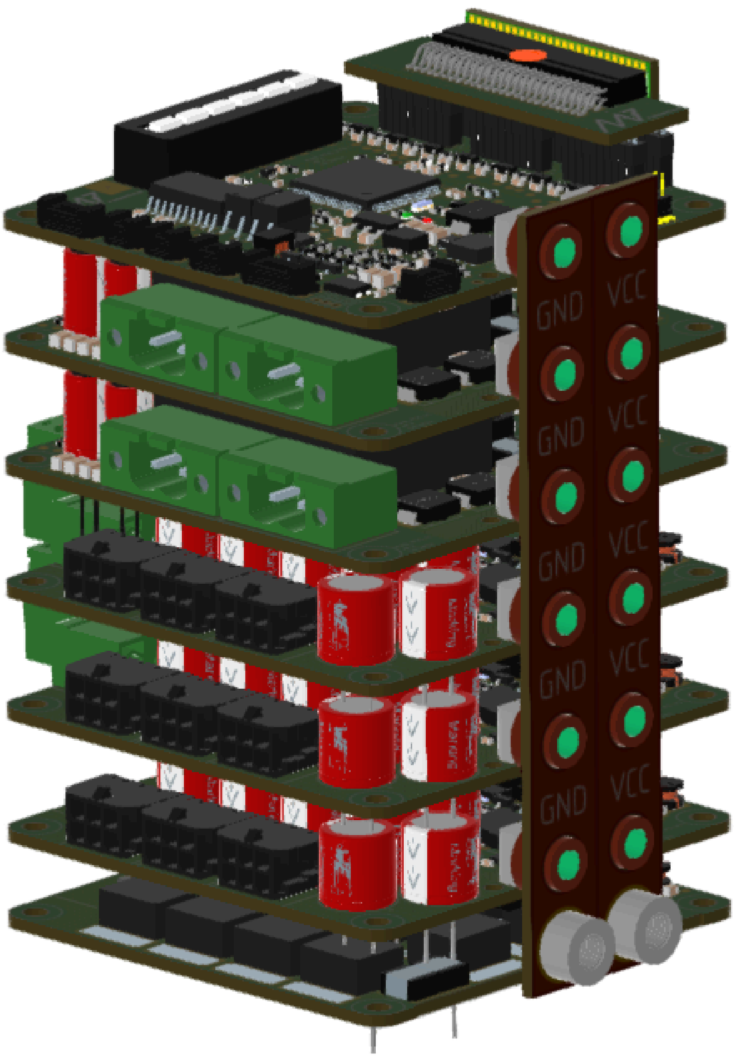
MAB ROBOTICS

MAB Robotics
Power Distribution System
MAB PDS



User manual

v1.0

02.2025

www.mabrobotics.pl
contact@mabrobotics.pl

1. Introduction

1.1 Overview

For robots, particularly mobile ones, it's crucial to have robust power distribution systems that ensure reliable, safe operation, energy efficiency, and seamless integration with components like motors, sensors, onboard computers, and subsystems. Additionally, the challenge is to make these systems compact and lightweight. MAB PDS addresses all these requirements, offering a comprehensive, integrated solution that simplifies the construction of robots while meeting all essential power management needs in a small, efficient form factor.

With the MAB PDS, robotic researchers, engineers, and system integrators can focus on higher-level development instead of being limited by hardware and embedded firmware design, which can be time-consuming and complex.

The MAB PDS is built on a modular philosophy designed to meet the core needs of every robot:

1. high-power bus for motors with on/off control,
2. safe power bus for the computers and sensors with on/off control,
3. handling regenerative energy from motors,
4. monitoring all the parameters related to power supplies and energy consumption,
5. power switch to enable/disable the robot,
6. fast and reliable handling of safety triggers,
7. fast telemetry and control from a higher-level controller,
8. scalability for flexible system expansion.

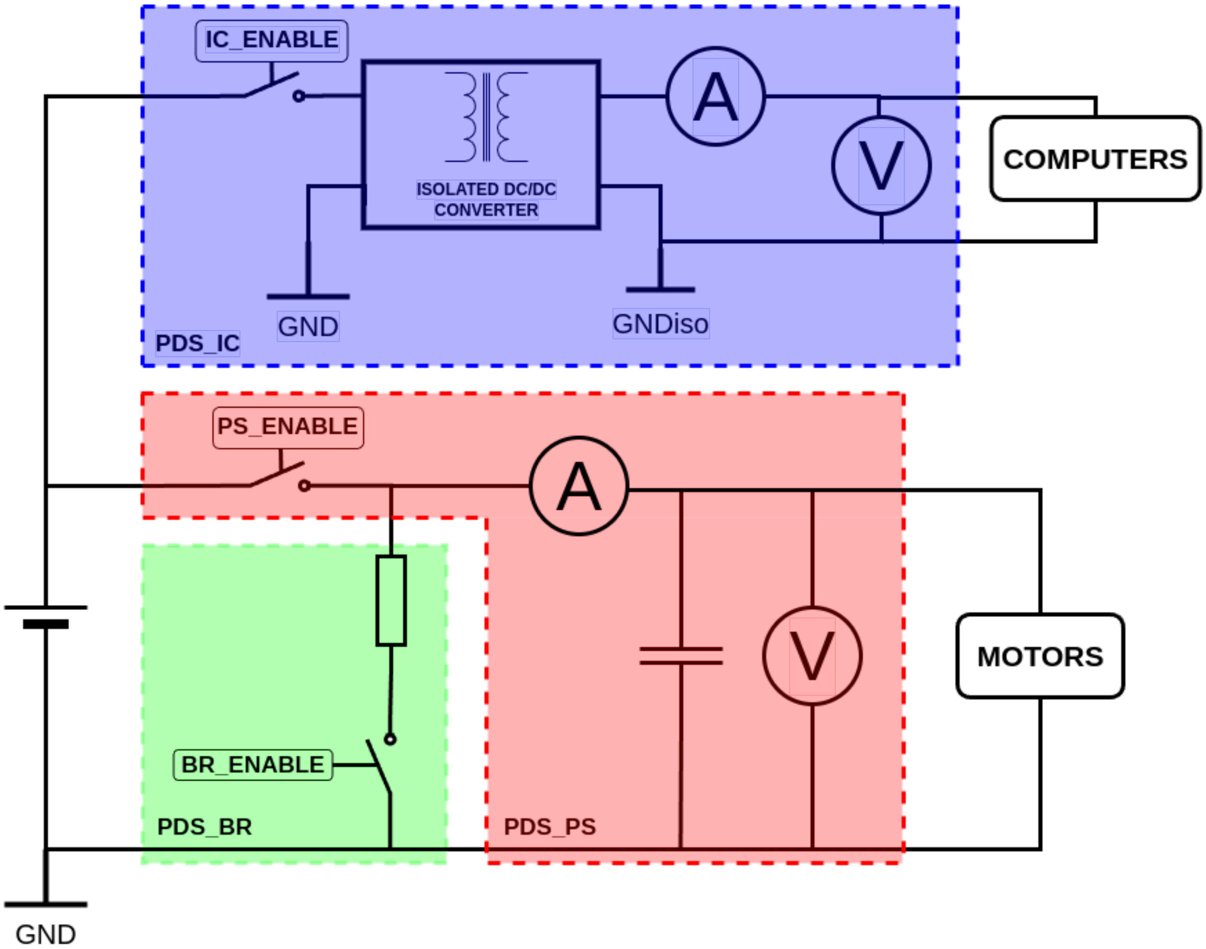
1.2 Modules

To handle all these features, the MAB PDS is split into four different types of modules:

- control (CTRL) - master module of the system,
- isolated DC/DC converter (IC) - a module with an isolated DC/DC converter for peripherals like computers and sensors,
- power stage (PS) - a module that handles high-power delivery to the motors,
- brake resistor (BR) - complementary subsystem with the Power Stage module.

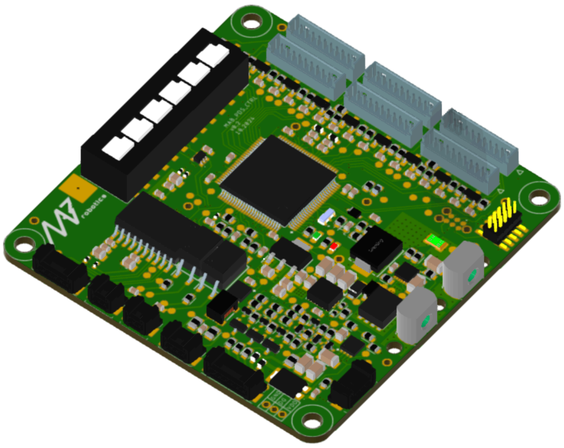
2. Specification

The overview of what exactly the PDS slave modules do is presented in the simple scheme below:



The master (CTRL) module manages ENABLE signals, acquires current/voltage measurements, and communicates with external systems.

2.1 Control (CTRL) module



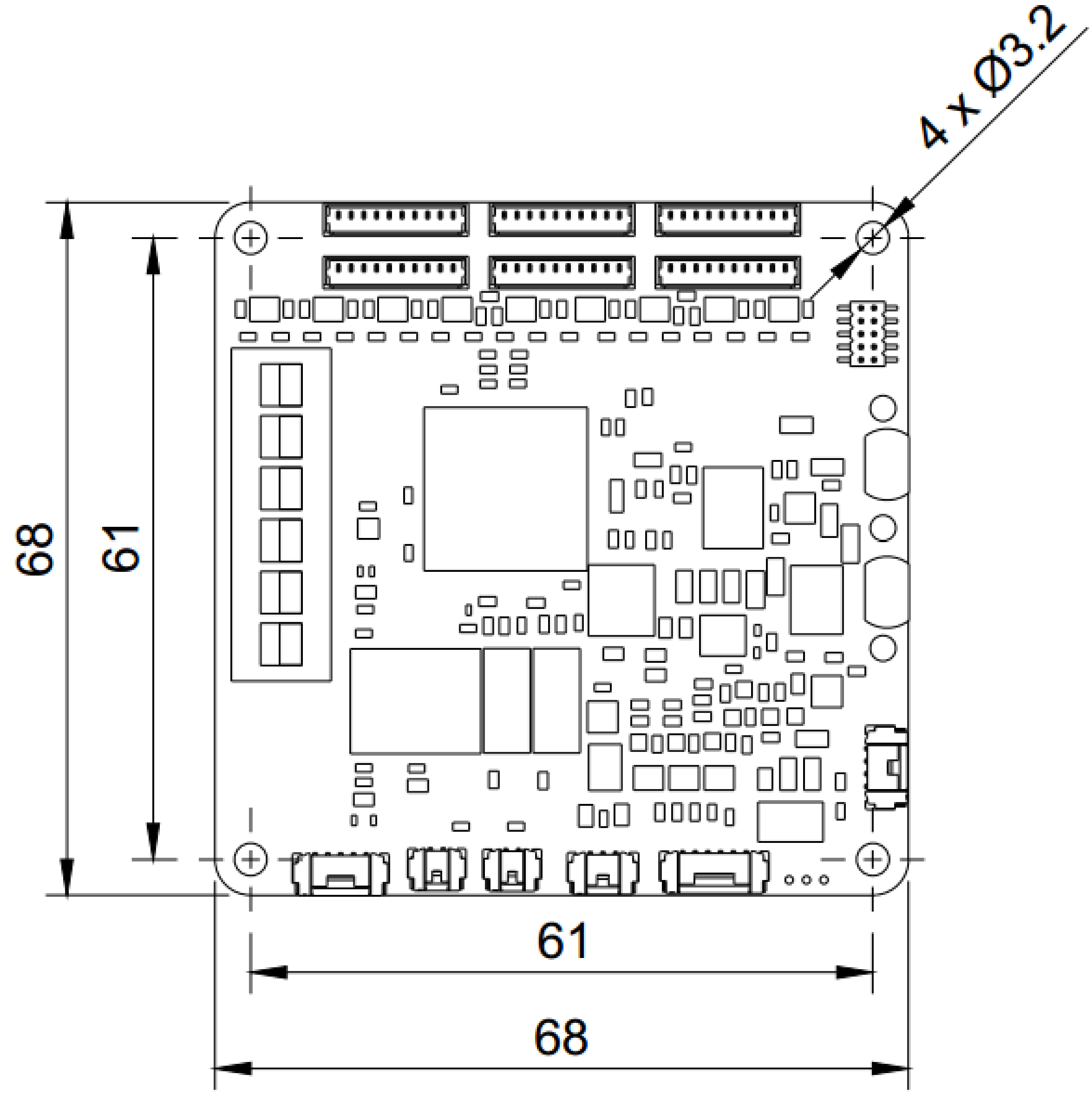
The PDS_CTRL module serves as the central controller in the system. Its main functions include:

- managing and acquiring data from up to six connected modules,
- facilitating communication over CAN FD with support for integration into the CANdle ecosystem,
- handling the RGB switch for powering the system on and off, with status reporting via the RGB diodes,
- the monitoring input voltage,
- tracking temperature across the system,
- managing safe Turn Off (STO) inputs ensures safe operation in critical situations.

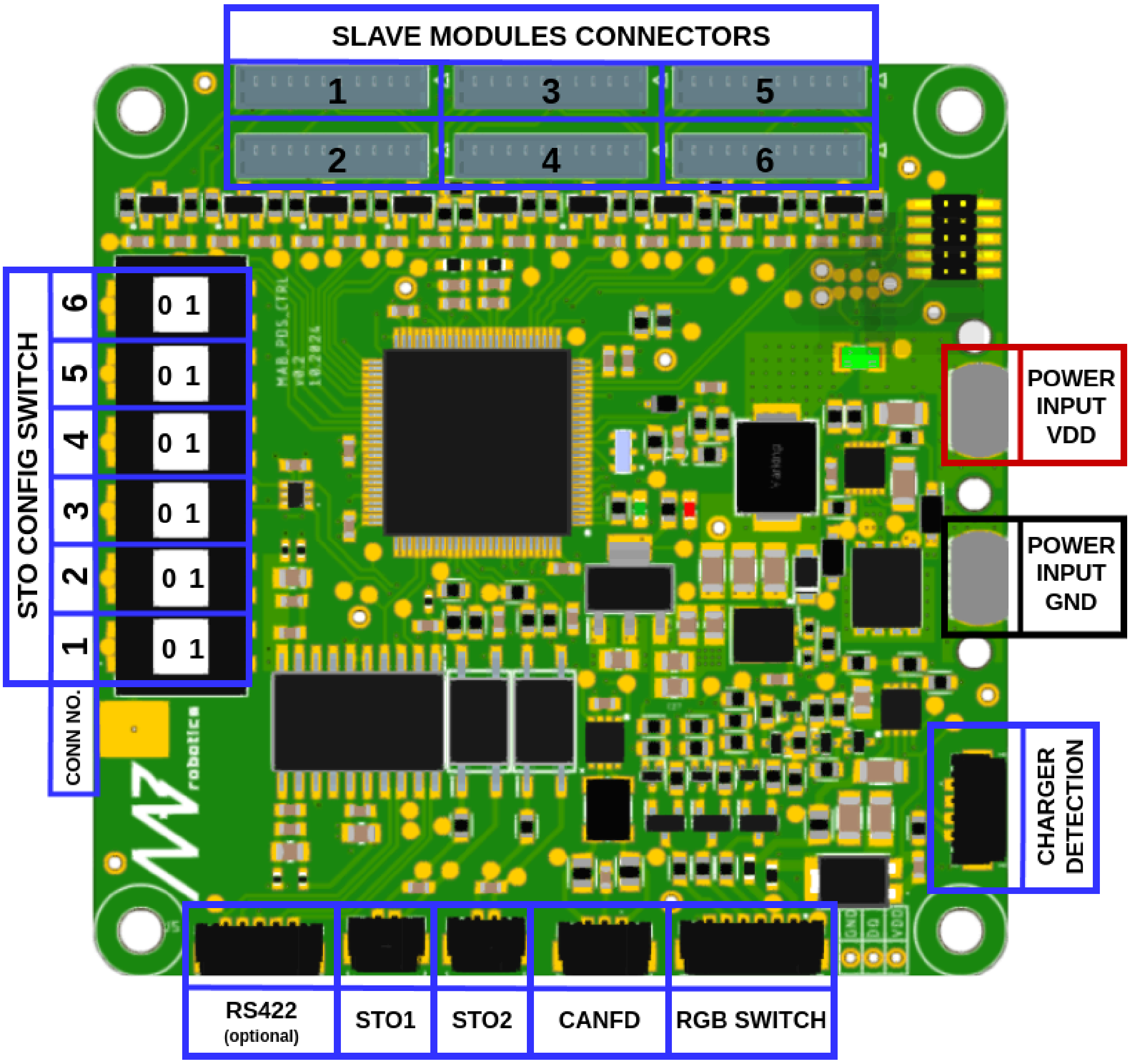
2.1.1 General specification:

Parameter	Value
Input voltage	10-54V
Max number of slave modules	6
FDCAN Baudrate (adjustable)	1/2/5/8 Mbps
Current consumption on standby (without slave modules)	37.5 mA
The quiescent current in the shutdown	11.63 uA
Mass	36g
Ambient Temperature (Operating)	0-40℃
Ambient Temperature (non-operating)	0-60℃
Maximum Humidity (Operating)	up to 95%, non-condensing at 40 °C
Maximum Humidity (Non-Operating)	up to 95%, non-condensing at 60 °C
Altitude (Operating)	-400 m to 2000 m

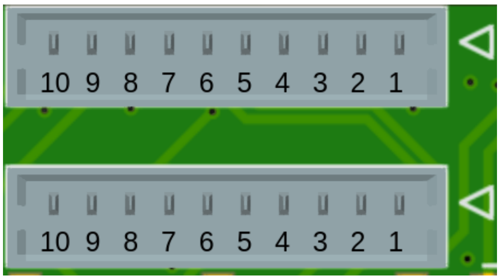
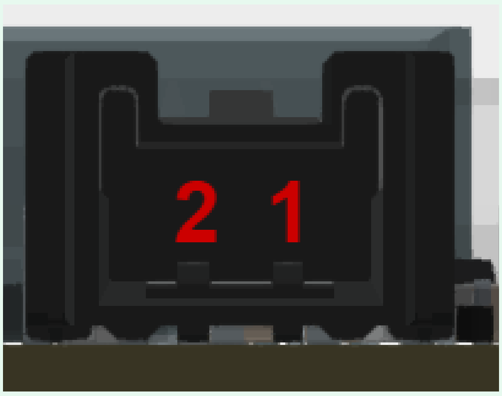
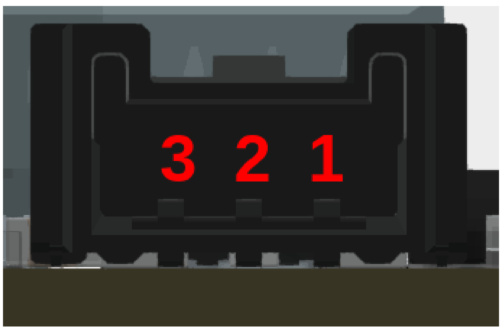
2.1.2 Drawings

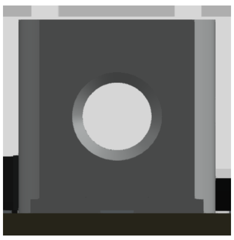
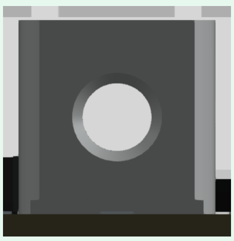


2.1.3 Board Overview



2.1.4 Connectors list

Connector name	Included cable specification	Connector and cable pinout	Wiring details
SLAVE SOCKET	SIGNAL BAR dedicated to requested PDS configuration.	 1 - 5V 2 - GND 3 - ENABLE 4 - VOLTAGE_MEASUREMENT 5 - OVER_CURRENT_DETECTION 6 - CURRENT_MEASUREMENT_P 7 - CURRENT_MEASUREMENT_N 8 - TEMPERATURE 9 - RESERVED 10 - RESERVED	Connectors to slave modules. The signal bar is installed by default.
STO1/2	Molex Microlock pigtail cable, 1m length	 1 - STOx_RTN - black 2 - STOx_IN - orange	Safe Turn Off/Kill switch port number 1. The port is handled with an optocoupler. Connect STOx_IN to the 12-30V safe voltage, like the computer supply in series with a normally close safety switch. The STOx_RTN connects to the corresponding ground.
CANFD	Molex Microlock pigtail cable, 1m length	 1 - CAN_H - grey 2 - GND - black 3 - CAN_L - yellow	Non-isolated CANFD port to communicate with robot electronic systems. This port can be connected to the CANdle ecosystem. Termination on this port can be mounted if necessary. Connect to CANFD lines of the CANdle or custom CANFD

			controller.
CHARGER DETECTION	Optional	 1 - CHRG_DET_RTN - black 2 - CHRG_DET_IN - orange 3 - PWR_BTN_TERMINAL1 - white 4 - PWR_BTN_TERMINAL2 - purple	Port to detect charging. This port is usable in battery-powered applications to be woken up by the charger plugging event. Connect to the charging add-on module.
RS422	Optional		Auxiliary galvanically isolated RS422 port with isolated power. This port is not included in standard PDS configuration. For more information, please contact us.
RGB SWITCH	Molex Microlock cable with the RGB switch, 1m length	 1 - PWR_BTN_TERMINAL2 - purple 2 - PWR_BTN_TERMINAL1 - white 3 - LED_RED - red 4 - LED_GREEN - green 5 - LED_BLUE - blue 6 - GND - black	This is the port for the RGB power switch. The user can enable the whole PDS over this port. RGB diodes indicate the system's state. Connect the assembled RGB here.
POWER INPUT VDD	M3 screw mount point for power bus bar.		Power input - positive(VDD) connector. Nominal current: 50A. Thread: M3 The power bus bar is installed by default.
POWER INPUT GND	M3 screw mount point for power bus bar.		Power input - negative(GND) connector. Nominal current: 50A. The power bus bar is installed by default.

2.1.5 Ports specification

2.1.5.1 Slave modules ports

These ports follow a generic structure and are used for controlling and monitoring slave modules. The sections of the manual dedicated to the respective slave modules provide detailed descriptions of the functionalities associated with each signal line.

2.1.5.2 RS422

Feature	Description
Transceiver	12-Mbps isolated RS422 transceiver with integrated DC/DC converter.
Transmission Type	Full duplex
Termination	Split termination soldered on the board
Signal Protection	TVS diodes on signal lines
Isolation Rating	5000V
Surge Test Voltage	7.8 kVPK (basic insulation); upgradeable to 10 kVPK (reinforced insulation) upon request.

RS422 interface can serve as a redundant communication channel alongside CANFD, offering a secure backup option, or as the primary communication method in applications where CANFD is not available, such as when integrating with PLCs in industrial automation systems.
This port is available for request.

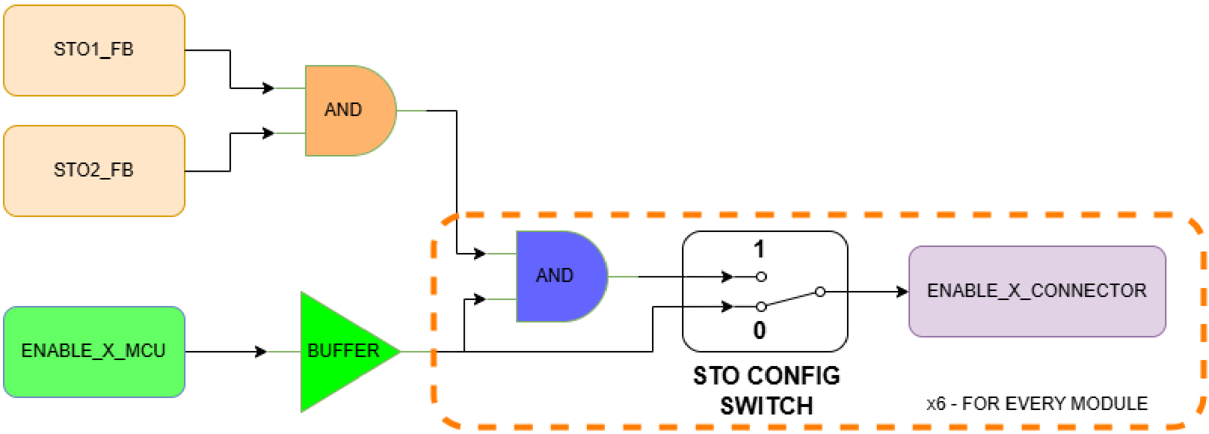
2.1.5.3 STO - Safe Turn Off

The PDS incorporates a safety mechanism for controlled power shut-off, similar to those commonly used in motor controllers. It features two input channels that can independently cut off power from the connected loads, ensuring safety during operation.

The inputs are protected against reverse polarity.

Parameter	Value
STO_X_IN wrt STO_X_RTN voltage	12-30V
Rated transient overvoltage immunity	6kVpeak

The STO signal mechanics is presented below:



For every slave module the STO input influence of PDS behavior is configurable with the mechanical switch (STO CONFIG SWITCH on the graphs above). The truth table for the system is presented below:

STO1_FB	STO2_FB	ENABLE_X_MCU	STO_CONFIG_ SWITCH	ENABLE_X_ CONNECTOR **MODULE ENABLE INPUT STATE**
X	X	0	X	0
X	X	1	0	1
0	0	1	1	0
1	0			0
0	1			0
1	1			1

Summarizing:

1. If STO_CONFIG_SWITCH is in position 0, the ENABLE state set by MCU is passed directly to the output, regardless of the status of STO1_FB and STO2_FB
2. If STO_CONFIG_SWITCH is in position 1, the ENABLE state set by MCU is passed to the output only when the STOx_FB signals state is 1

This mechanism offers full flexibility in handling safety triggers. For instance, it allows for an immediate power cut-off from the PS modules, which supply power to the actuators, while keeping the IC modules that power the computers active.

The STO input is implemented purely in hardware, with minimal delays primarily due to signal propagation through integrated circuits and the turn-off time of the power MOSFETs. The time between the rising edge of the STOx_FB signal and the power cut-off is typically no more than a few microseconds.

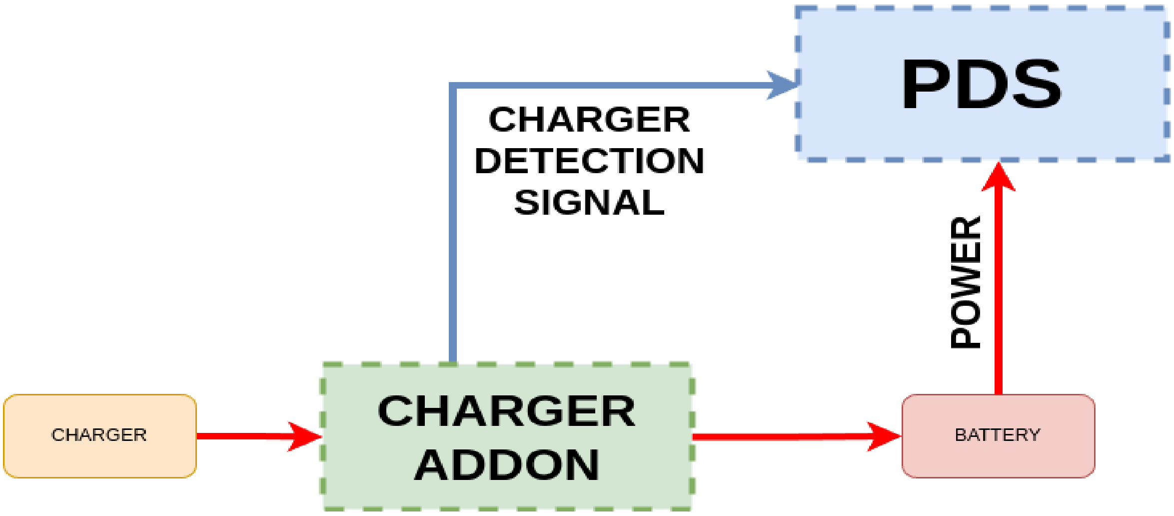
2.1.5.4 CANFD

Feature	Description
Transceiver	8-Mbps non-isolated CANFD transceiver
Termination	Split termination soldered on the board
Signal Protection	TVS diode and CMC on signal lines, ESD Protection on CANH and CANL up to ±13 kV

The CAN FD interface in the PDS is designed to provide essential telemetry data, including information on currents, voltages, and temperature, as well as to facilitate the configuration and control of the system. For comprehensive details regarding the protocol and integration with the CANdle ecosystem, please refer to the following sections of the manual, where these aspects are explained in detail.

2.1.5.5 CHARGER DETECTION*

PDS can detect the charger in battery-powered applications. It can be used when the battery is installed in the device. To get this functionality, it is necessary to use a charging addon between the charger and the battery. When this feature is used, connecting the charger will wake up the PDS and cause the RGB switch to slow blinking(more details in the 2.1.5.6 paragraph). The idea is presented in the scheme below:

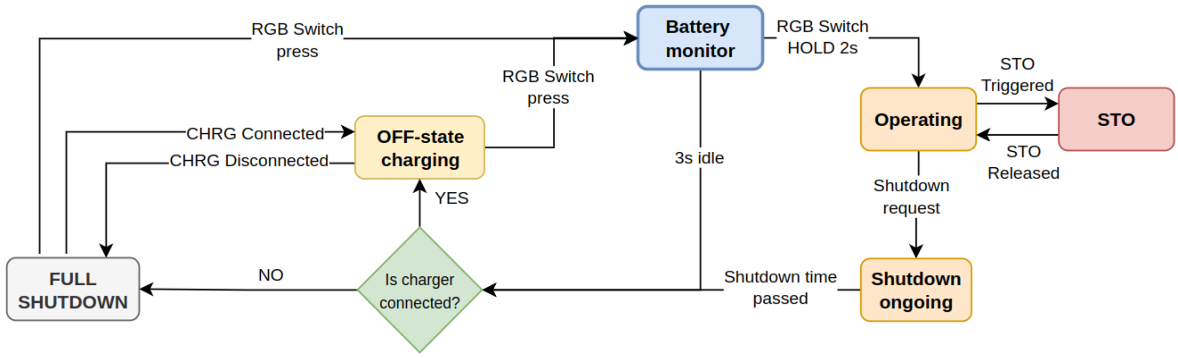


*under development, will be available soon.

2.1.5.6 RGB SWITCH

This port is dedicated to connecting the RGB switch which enables the PDS and reports the system state with the RGB diodes.

The states and transitions are presented in the scheme below:

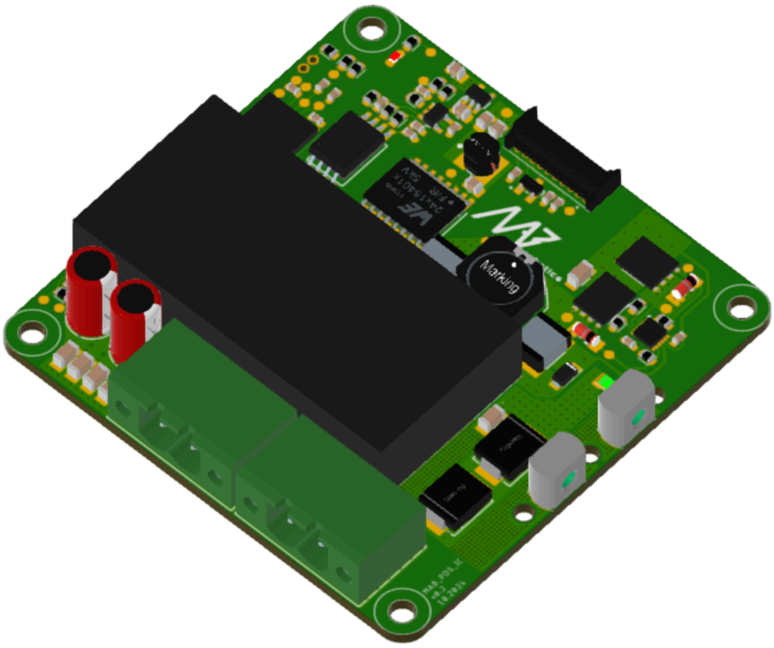


State name	State definition
Full shutdown	PDS is turned off with a quiescent current ~20uA.
Battery monitor	PDS shows the battery charge level with respect to the user-predefined voltage levels. Refer to the “Battery voltage levels” in section 3.4.4.3 for detailed description. In this state, all slave modules are turned off.
Off state charging	PDS slave modules are turned off. The CTRL module is enabled. The battery is charging. RGB button blinks slowly with color dependent on the battery voltage.
Operating	Normal operation. RGB switch indicates battery level same way as in battery monitor state. If the charger is connected the RGB switch additionally blinks slowly with BLUE/<BAT LEVEL COLOR> alternately.
STO	Device generally works as in operating state but RGB Led is indicating STO event by blinking RED / YELLOW.
Shutdown ongoing	All submodules except Isolated converters are disabled. RGB switch turns solid WHITE. Shutdown ongoing flag is set in Control Board status register and device will turn into full shutdown state after time specified by the shutdown time property.

2.1.5.7 POWER INPUT

The power input connector consists of a pair of M3 right-angle connectors, specifically designed for connection to the bus bar, as detailed in the subsequent sections of the manual.

2.2 Isolated converter(IC) module



The PDS_IC module features an isolated DC/DC converter, providing a secure and stable power supply for sensitive electronic systems, such as computers and subsystems. Key characteristics include:

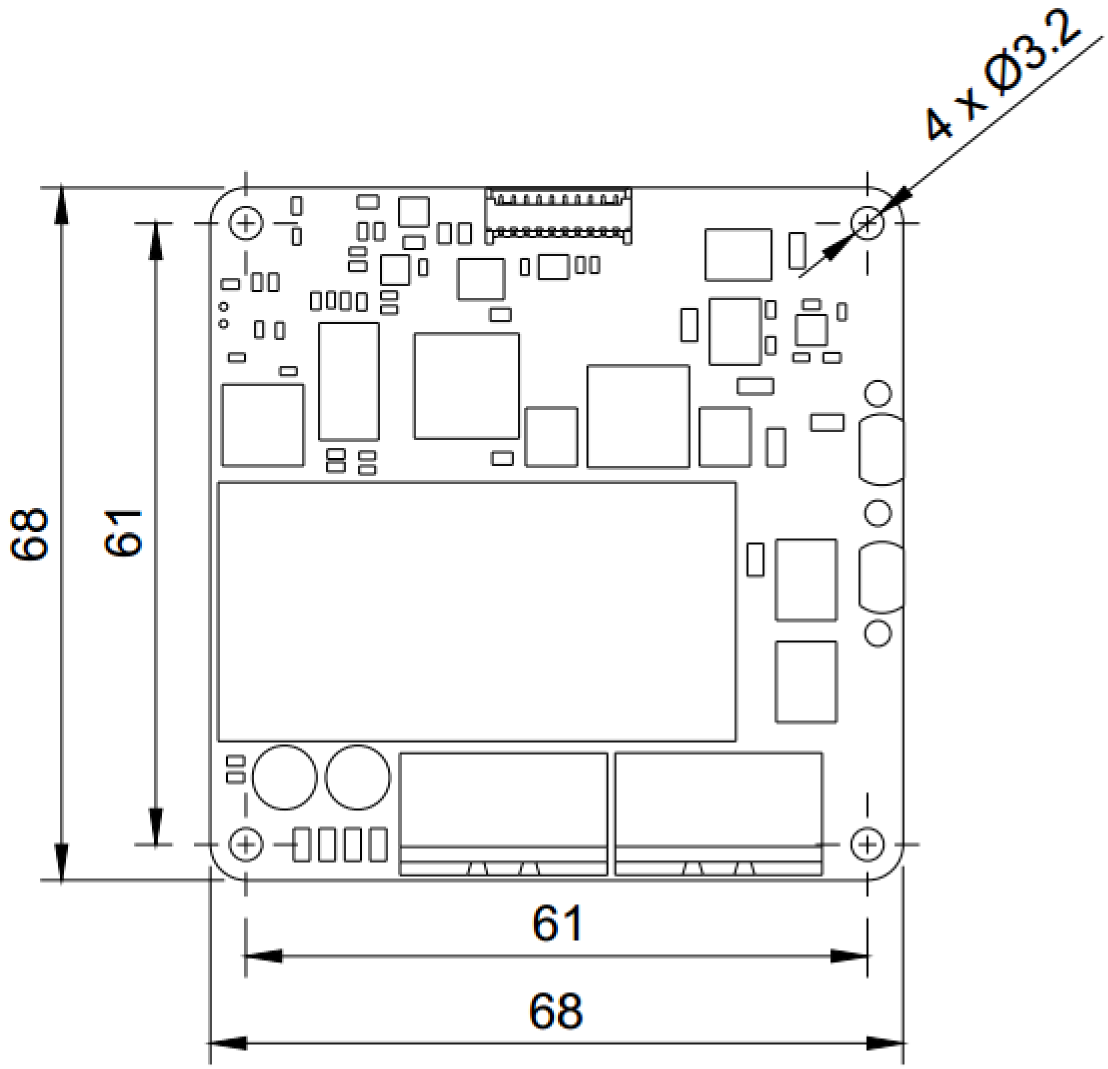
- isolated power line with a configurable output voltage,
- isolated voltage and current measurements on the isolated bus,
- temperature monitoring for reliable operation,
- reverse polarity, over-voltage, and over-current protection for safety,
- integrated EMC filter to reduce electromagnetic interference,
- transient protection to guard against sudden voltage spikes.

2.2.2 General specification:

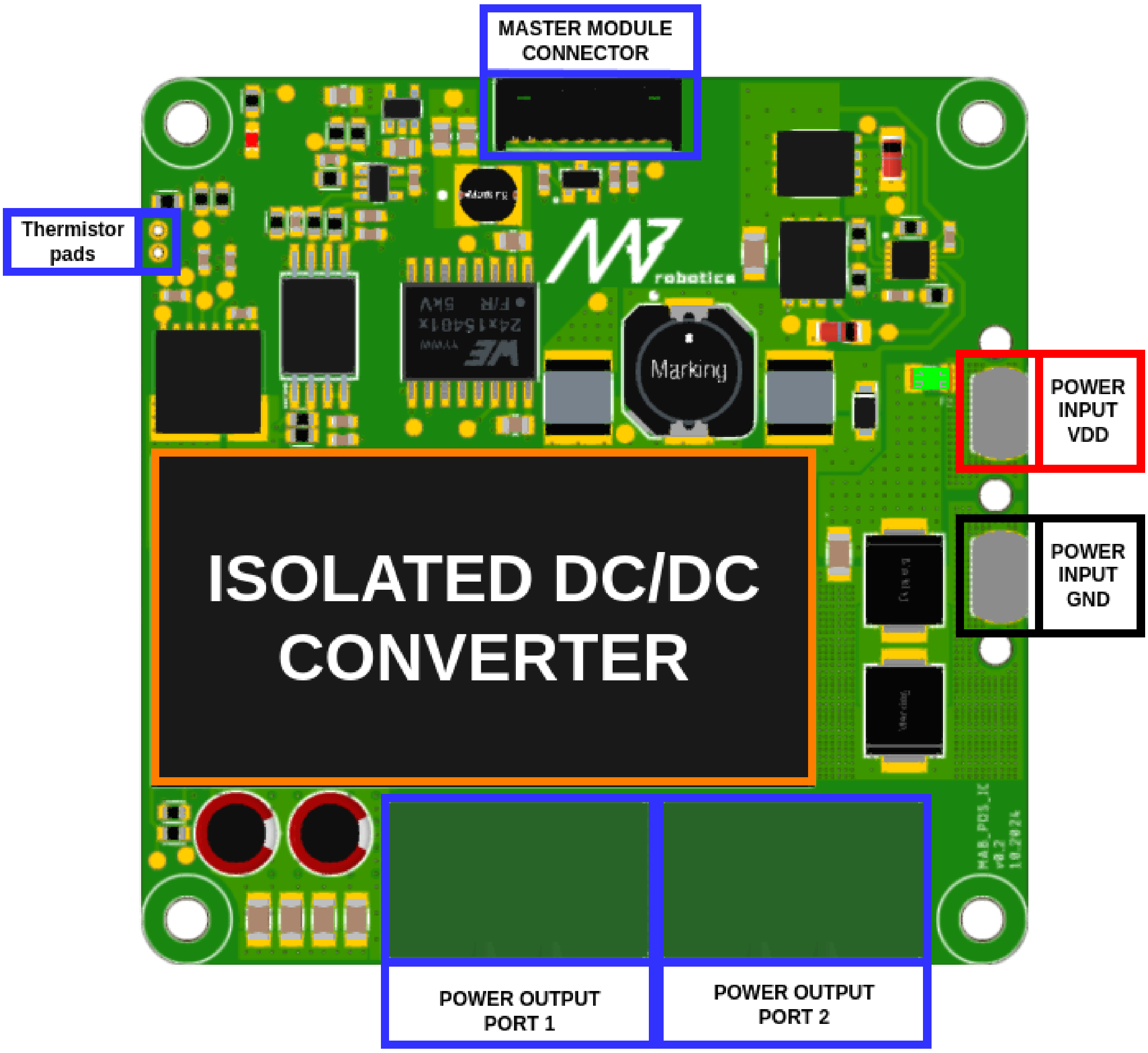
Parameter	Value
Input voltage	10-54V
Input protection	reverse polarity down to -54V overvoltage, overcurrent(with fuse), transients.
Power	60W
DC/DC converter output	Available options: 3.3VDC 12A 5VDC 12A 12VDC 5A 15VDC 4A 24VDC 2.5A

Output protection	Short circuit (Continuous), overload, over-voltage, over temperature, input undervoltage lockout.
Current consumption on standby(no load)	7,8 mA
The quiescent current in the shutdown	6,57 uA
Mass	74g
Ambient Temperature (Operating)	0-40°C
Ambient Temperature (non-operating)	0-60°C
Maximum Humidity (Operating)	up to 95%, non-condensing at 40 °C
Maximum Humidity (Non-Operating)	up to 95%, non-condensing at 60 °C
Altitude (Operating)	-400 m to 2000 m

2.2.3 Drawings

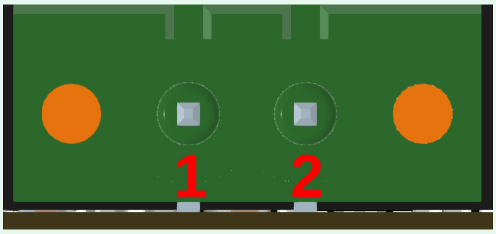
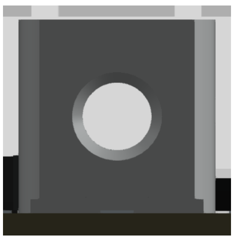
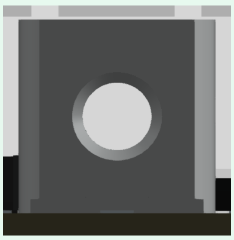
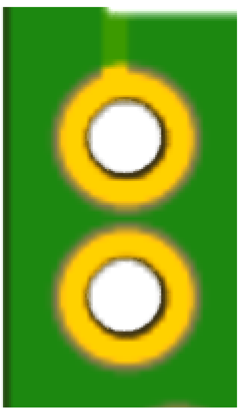


2.2.4 Board Overview



2.2.5 Connectors list

Connector name	Included cable specification	Pinout	Wiring details
MASTER MODULE	SIGNAL BAR dedicated to requested PDS configuration.	 1 - 5V 2 - GND 3 - DC/DC converter enable 4 - OUTPUT_VOLTAGE_SIGNAL 5 - OVER_CURRENT_DETECTION	Connector to the master CTRL module. The signal bar is installed by default.

		6 - OUTPUT_CURRENT_P 7 - OUTPUT_CURRENT_N 8 - CONVERTER_TEMPERATURE 9 - RESERVED 10 - RESERVED	
POWER OUTPUT PORT 1/2	2 pcs of [691340500002]() included, without the cables	 1 - V_ISO 2 - GND_ISO	Isolated DC/DC converter port 1. Mate connector: Horizontal: [691340500002]() Vertical: [691343510002]() One of these connectors can be used to connect the load to the outputs of isolated DC/DC converters. Supported cable diameter: 12 to 24AWG.
POWER INPUT VDD	M3 screw mount point for power bus bar.		Power input - positive(VDD) connector. Nominal current: 50A. Thread: M3. The power bus bar is installed by default.
POWER INPUT GND	M3 screw mount point for power bus bar.		Power input - negative(GND) connector. Nominal current: 50A. The power bus bar is installed by default.
THERMISTOR INPUT	Soldering pads. According to IPC-2222B wire diameter can be from 0,1mm (AWG38) to 0,6mm (AWG23)		External thermistors mounted on the DC/DC converter case/heatsink have to be mounted to these pads. The PDS_IC comes with a factory-assembled thermistor.

2.2.6 Ports specification

2.2.6.1 MASTER CONNECTOR

This connector is used for controlling and monitoring the PDS_IC module.

2.2.6.2 POWER OUTPUT

This port is directly connected to the output of the isolated DC/DC converter. It features two symmetrical connectors, allowing power to be distributed to multiple points. Both current and voltage at these ports are measured and monitored. The DC/DC converter output has adjustable overcurrent and overvoltage limits, which can be configured via software. Measurement is done using isolated sensors, ensuring galvanic isolation from all other potentials in the PDS, including the PDS_CTRL.

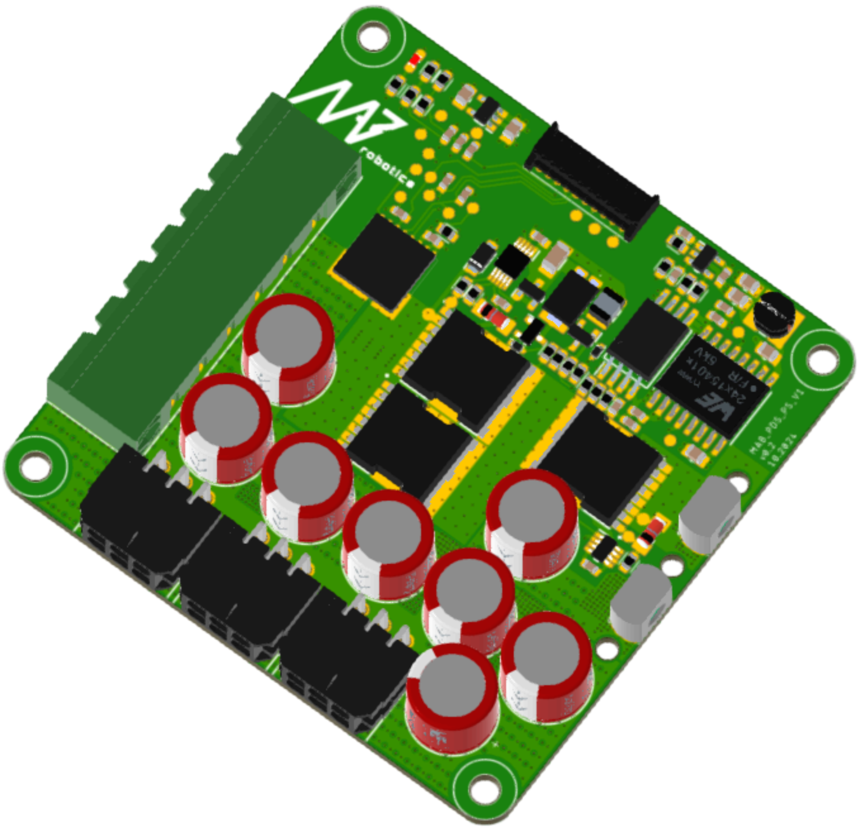
2.2.6.3 POWER INPUT

The power input connector consists of a pair of M3 right-angle connectors, specifically designed for seamless connection to the bus bar. Detailed instructions on how to integrate this with the bus bar are provided in the following sections of the manual.

2.2.6.4 THERMISTOR INPUT

The temperature of the DC/DC converter is monitored using an external thermistor, which should be mounted on the converter's casing with heat-transferring adhesive to ensure accurate readings. The IC module comes with a thermistor pre-assembled at the DC/DC converter case. This setup allows for effective thermal management and precise temperature monitoring, ensuring optimal performance and safety.

2.3 Power Stage (PS) module



The PDS_PS module is a high-power power path designed to provide a safe and reliable power source for robot actuators. Its features include:

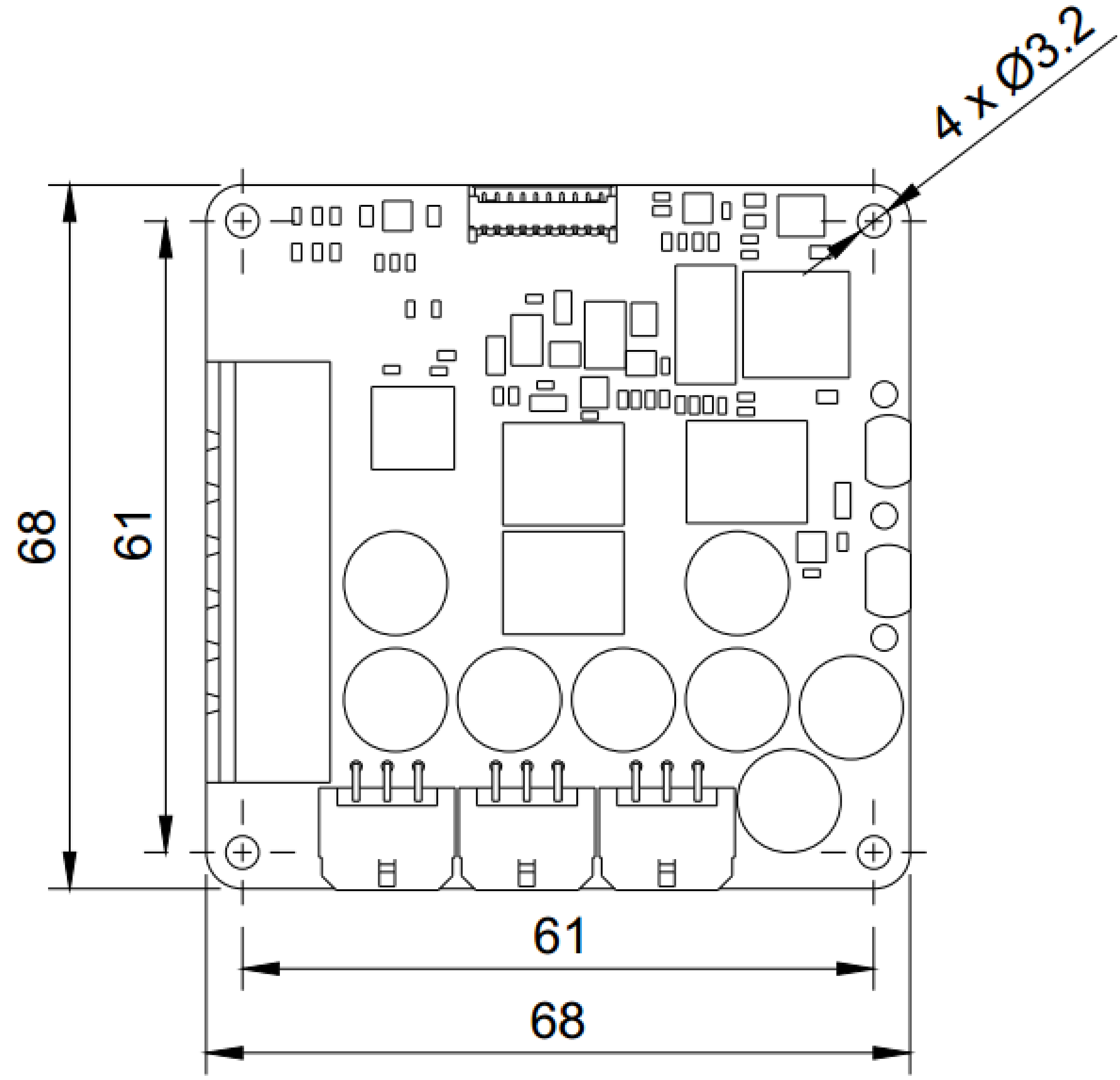
- continuous current capacity of up to 50A,
- isolated voltage and current measurements for precise monitoring,
- bus-controlled with an electronic high-side switch for efficient power management
- inrush current limitation to prevent damage from startup surges,
- high-quality bulk capacitance for stable operation,
- protection against reverse polarity, over-voltage, and over-current to ensure safety,
- transient protection for resilience against voltage spikes,
- MOSFET temperature feedback for thermal monitoring and safe operation.

2.3.1 General specification:

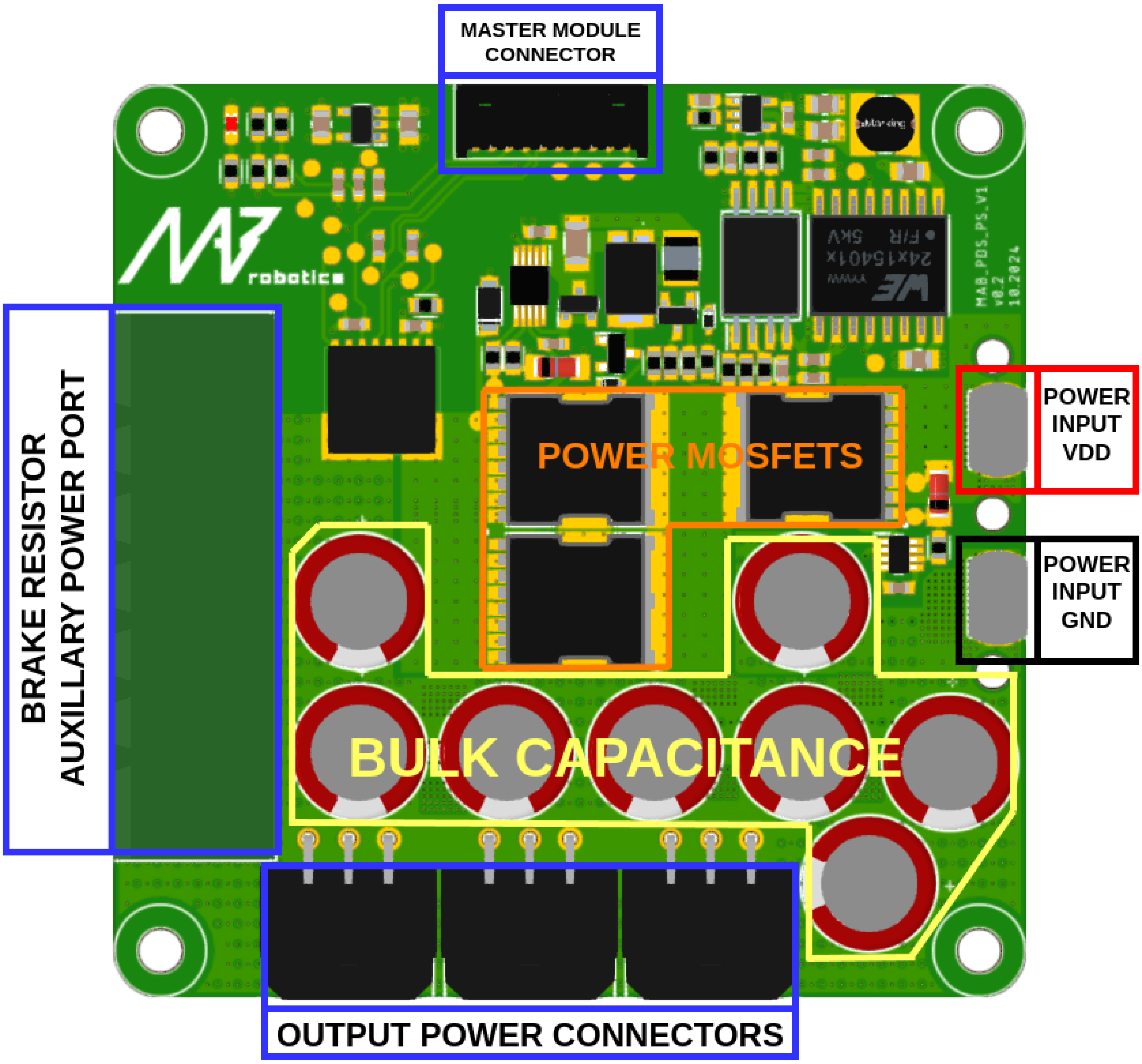
Parameter	Value
Input voltage	10-54V
Input protection	reverse polarity down to -54V overvoltage, overcurrent (no fuse on the board, based on current monitoring), transients.

Output current	up to 50A
Delivered power	up to 2500W
Total internal resistance	1.35 mohm
Conduction losses at 50A current(excluding losses on the connectors' resistance)	3.375 W
Power transfer efficiency	up to 99.8%
Output protection	Over-current, over-voltage, over temperature, transient.
Internal bus capacitance	376uF(low ESR Aluminum Hybrid Polymer Capacitors)
Current consumption on standby(no load)	11.56 mA
The quiescent current in the shutdown	1.97 uA
Mass	43g
Ambient Temperature (Operating)	0-40℃
Ambient Temperature (non-operating)	0-60℃
Maximum Humidity (Operating)	up to 95%, non-condensing at 40 °C
Maximum Humidity (Non-Operating)	up to 95%, non-condensing at 60 °C
Altitude (Operating)	-400 m to 2000 m

2.3.2 Drawings

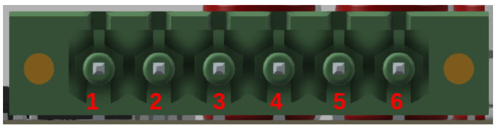
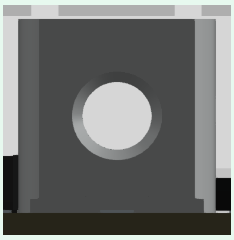
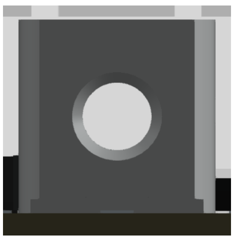


2.3.3 Board Overview



2.3.4 Connectors list

Connector name	Included cable specification	Pinout	Wiring details
MASTER MODULE	Signal bar dedicated to chosen PDS stack.	 1 - 5V 2 - GND 3 - BUS_ENABLE	Connector to the master CTRL module. The signal bar is installed by default.

		4 - VOLTAGE_MEASUREMENT 5 - OVER_CURRENT_DETECTION 6 - OUTPUT_CURRENT_P 7 - OUTPUT_CURRENT_N 8 - MOSFET_TEMPERATURE 9 - RESERVED 10 - RESERVED	
POWER OUTPUT PORT 1/2/3	No cables included.	 1 - CAN_H 2 - GND 3 - V+ 4 - V+ 5 - GND 6 - CAN_L The CANFD lines are NOT connected between POWER OUTPUT PORTS.	Power delivery to the actuators. Connect the the actuators to Molex Micro-Fit ports.
BRAKE RESISTOR/ AUXILLARY POWER SUPPLY	[691340500006](691340500006) included, without the cables.	 1 - V+ 2 - GND 3 - V+ 4 - GND 5 - V+ 6 - GND	Power delivery to the actuators/brake resistor connection. 691340500006 or 691343510006 can be used in port to serve as an auxiliary power supply connector.
POWER INPUT VDD	M3 screw mount point for power bus bar.		Power input - positive(VDD) connector. Nominal current: 50A. Thread: M3. The power bus bar is installed by default.
POWER INPUT GND	M3 screw mount point for power bus bar.		Power input - negative(GND) connector. Nominal current: 50A. Thread: M3. The power bus bar is installed by default.

*Various options available for the request.

**More information in the Ports specification section below

2.3.5 Ports specification

2.3.5.1 MASTER CONNECTOR

This connector facilitates the control and monitoring of the PDS_PS module.

2.3.5.2 POWER OUTPUT

The voltage on the three ports with Molex Micro-Fit 3.0 connectors is available when the bus is enabled by the CTRL board. Each connector can handle a continuous current of up to 10A. It is important to avoid overloading a single connector by connecting too many actuators in one chain. To ensure proper planning of the robot wiring, refer to the table below.

The provided data is based on tinned connectors with six fully loaded circuits in each connector. Please note that these values are interpolated per terminal (since each power line has two terminals). For more detailed information, refer to the Molex Micro Fit 3.0 documentation or contact us for additional support.

AWG wire size	Microfit 3.0 current [A]
18	6.5
20	5
22	4.5
24	4.5
26	4
28	3
30	3

The standard connection cable used with the MD80 motor controller is rated to handle up to 10A of current.

2.3.5.3 BRAKE RESISTOR/AUXILIARY POWER SUPPLY

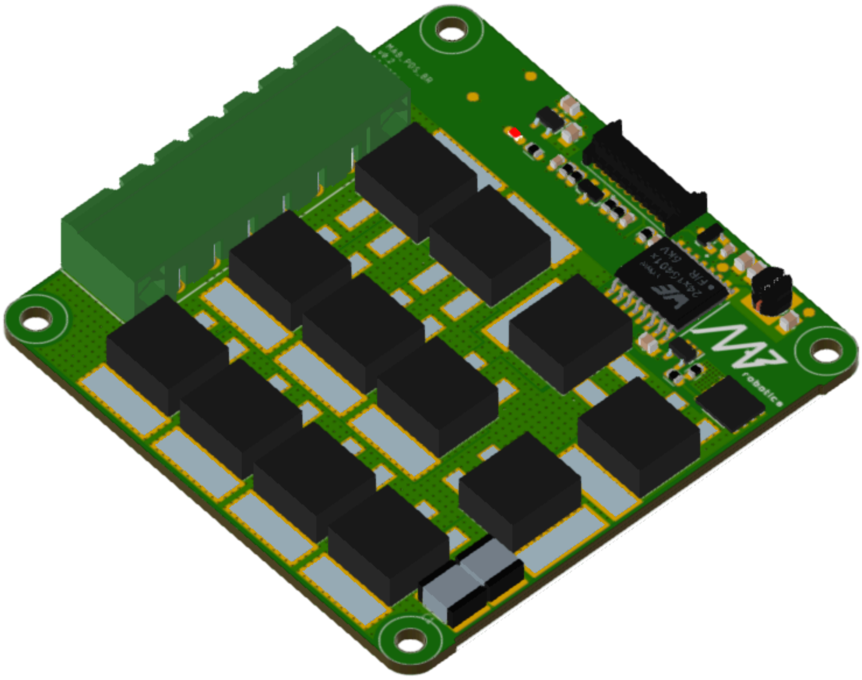
This port has three potential use cases:

a. **Brake Resistor Module Connection**: This is necessary for applications where regenerative braking occurs, and the PDS is used to protect against over-voltage actively. The connection can be made using the connector or by soldering the header between the PDS_PS and PDS_BR boards, depending on system requirements.
b. **Alternative Power Port**: If used instead of Molex Micro-Fit connectors, this port has 3 pairs of positive and negative poles, each capable of handling up to 20A, for a total of 60A.
c. **Mixed Usage**: This configuration allows for the simultaneous connection of both the braking resistor and power to the load.

2.3.5.4 POWER INPUT

The power input connector consists of a pair of M3 right-angle connectors. This port is designed for connection to the bus bar, which is further detailed in the following sections of the manual.

2.4 Braking resistor (BR) module



The PDS_BR module is a low-profile, high-power resistor with an integrated control and monitoring system. It simplifies the management of overvoltage situations in the system, typically caused by regenerative braking in motors. Key features include:

- onboard power resistors with a peak nominal power of 420W,
- built-in temperature monitoring for system safety,
- transient protection to prevent electrical surges,
- braking control through a MOSFET switch,
- adjustable trigger voltage via software, providing flexibility.

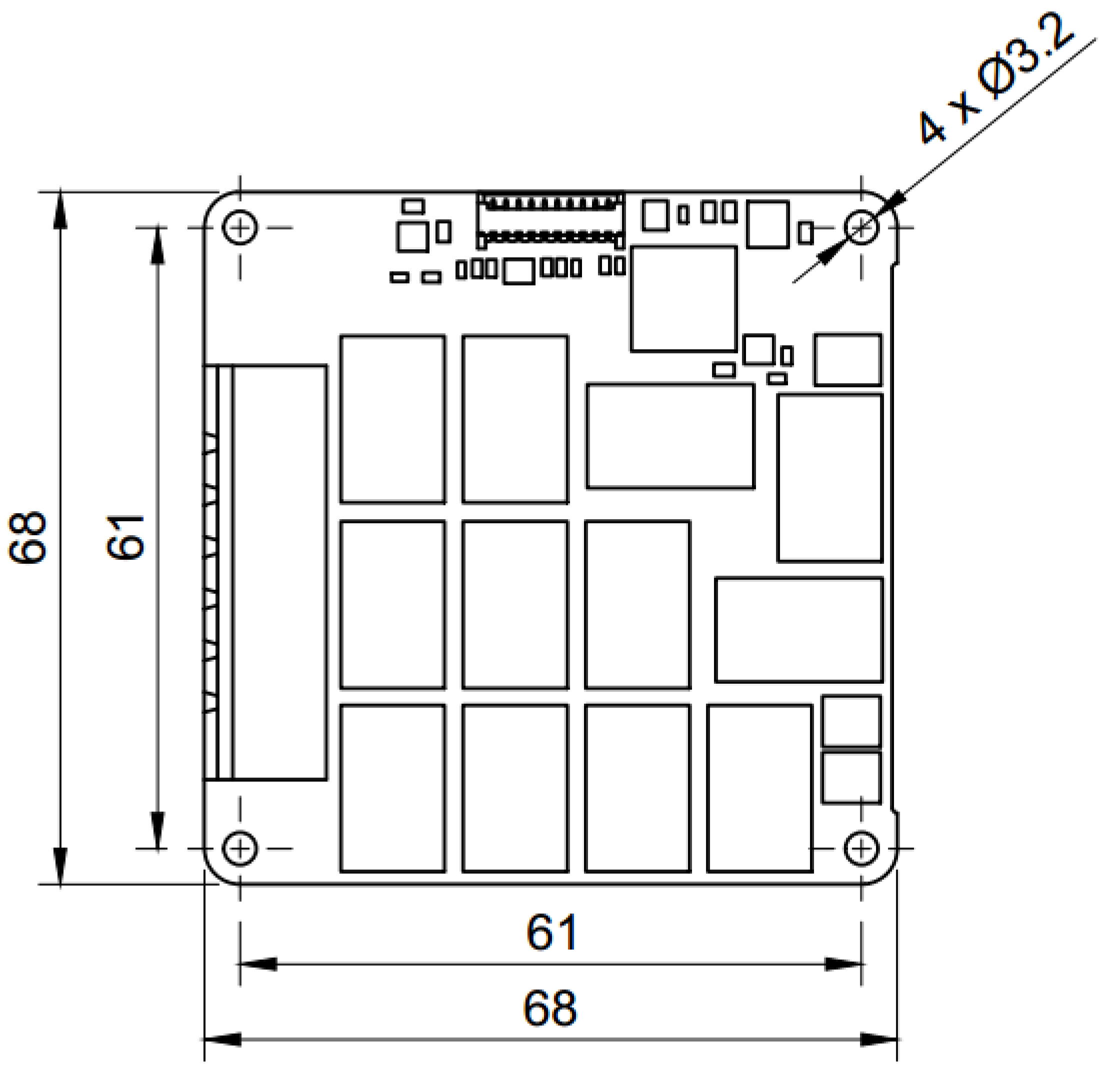
2.4.1 General specification:

Parameter	**Value**
Input voltage	10-54V
Input protection	transients, over temperature.
Maximum dissipated power	420W
Current consumption on standby(resistors disabled)	2,7 mA
Mass	42g
Ambient Temperature (Operating)	0-40°C
Ambient Temperature (non-operating)	0-60°C
Maximum Humidity (Operating)	up to 95%, non-condensing at 40 °C

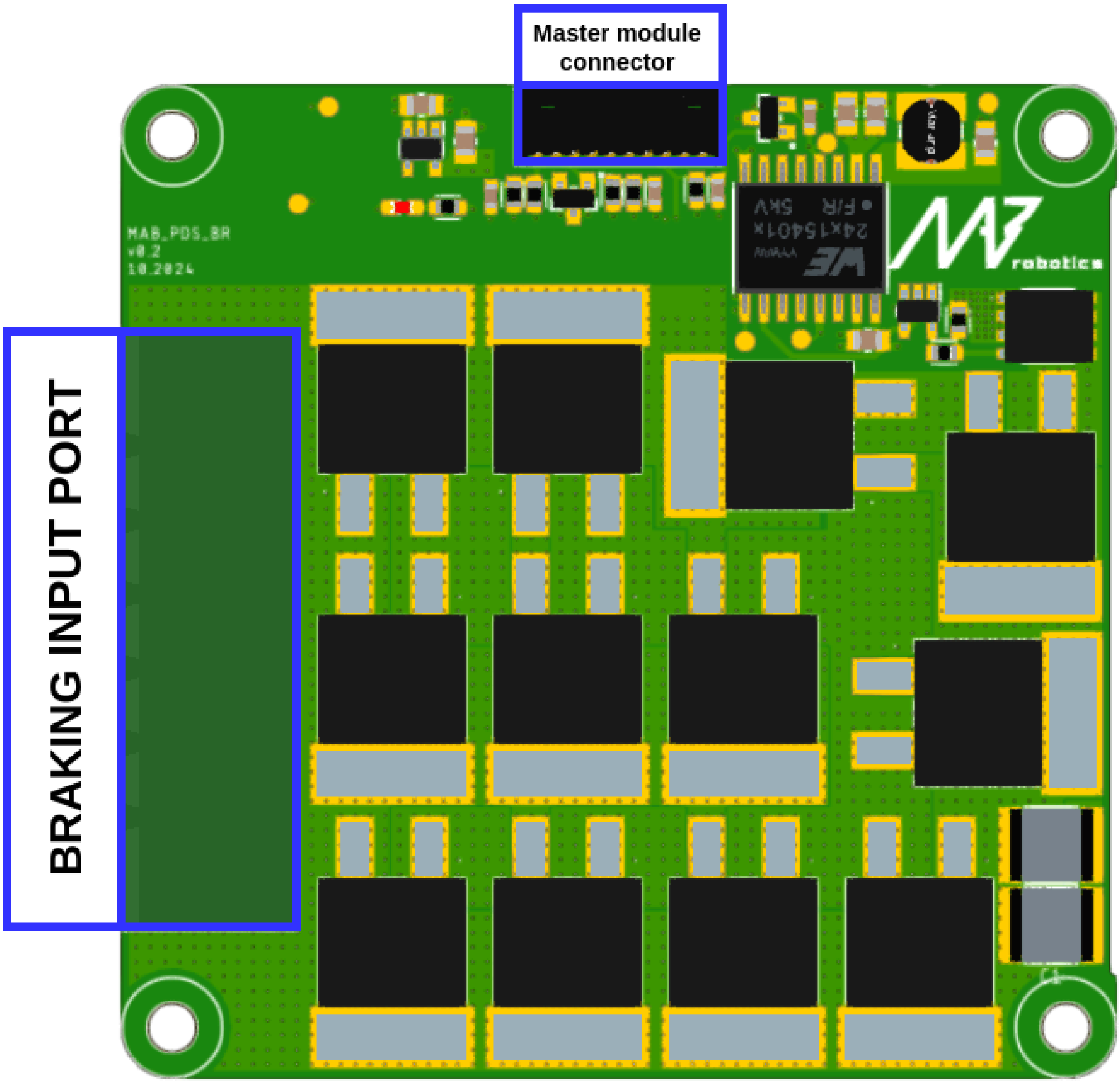
Maximum Humidity (Non-Operating)	up to 95%, non-condensing at 60 °C
Altitude (Operating)	-400 m to 2000 m

The BRAKE_ENABLE is triggered when an overvoltage condition occurs on the PS module connected to the BR module. In such cases, excess energy is dissipated through the braking resistors. It's important to ensure proper cooling of the BR module to prevent overheating and thermal shutdown.

2.4.2 Drawings



2.4.3 Board Overview



2.4.4 Connectors list

Connector name	**Included cable specification**	**Pinout**	**Wiring details**
MASTER MODULE	Signal bar dedicated to chosen PDS stack.	 1 - 5V 2 - GND 3 - BRAKING ENABLE 4 - RESERVED 5 - RESERVED 6 - RESERVED 7 - RESERVED 8 - TEMPERATURE 9 - RESERVED 10 - RESERVED	Connector to the master CTRL module. The signal bar is installed by default.
BRAKING INPUT PORT	691340500006 included, without the cables	 1 - V+ 2 - GND 3 - V+ 4 - GND 5 - V+ 6 - GND	Power delivery to the actuators/brake resistor connection. 691340500006 or 691343510006 can be used to connect to the PS module. If ordered with a preassembled connection to the PS module, there's no need to care about it.

2.4.5 Ports specification

2.4.5.1 MASTER CONNECTOR

This connector enables the control and monitoring of the PDS_BR module.

2.4.5.2 BRAKING INPUT PORT

This port is designed for connecting the PDS_PS output. In the event of overvoltage caused by regenerative braking, this port allows excess energy to be safely dissipated through an external brake resistor or redirected to energy recovery systems. This ensures the system operates within safe voltage limits, protecting components from potential damage.

2.5 Safety disclaimer

All power components of the modules are galvanically isolated from the master CTRL module to prevent potential ground loops and ensure stable system operation. However, the slave modules are not designed to operate independently of the master CTRL module. Some protection mechanisms are software-based, and without the CTRL module, hardware damage may occur. If the system is to be used in a configuration other than the recommended stack, please contact MAB Robotics for support.

Additionally, the power supply **must be protected** by an appropriate fuse or alternative protection mechanism on the power input line. While the MAB PDS includes short circuit protection on the output, the power supply itself, such as Li-Po batteries, requires further safeguards like a Battery Management System (BMS) to prevent risks like fire or explosion.

3. Integration

3.1 Guidelines for selecting PDS modules for your robot

To correctly select the PDS modules, answer these three key questions:

1. What is the number of actuators, and what is their expected current consumption?
2. Is the system battery-powered, and what is the system voltage?
3. How many computers or sensors does the robot have, and what are their operating voltage and current consumption?

Example Scenario: Four-wheeled robot with steerable axes, six DOF robotic arm, and a three-axis gimbal

Actuators Specifications:

Actuator Types	Number of actuators	Nominal Power	Nominal Current	Peak Power	Peak Current
MA-p-60-IP66	9(6 for arm, 3 for gimbal)	150 W	3.6 A	330 W	7.9 A
MA-p-100-IP66	6(4 in wheels, 2 to control axes angle)	450 W	10 A	800 W	19 A

Battery Specification

- Type: 10S Li-Po
- Maximum Voltage: 42 V

Additional Components

1. GPU Unit (AI and Control):

- Supply Voltage: 9–20 V
- Max Power: 60 W

2. LIDAR (Sensors):
   - Supply Voltage: 24 V
   - Max Power: 30 W
3. 5G Router (Connectivity):
   - Supply Voltage: 24 V
   - Max Current: 0.6 A

Modules selection

1. PS Modules (Power Stage)

MA-p-60-IP66 Actuators (9 Units):

- Nominal Current Load:
  - 9×3.6 A = 32.4 A
- Peak Current Load (4 Overloaded Actuators):
  - 5×3.6 A+4×7.9 A = 49.6 A

MA-p-100-IP66 Actuators (6 Units):

- Nominal Current Load:
  - 6×10 A = 60 A
- Peak Current Load (2 Overloaded Actuators):
  - 2×19 A+4×10 A = 48 A

Solution: Use three PS modules, splitting the actuators evenly:

- 1st module handles 3 **MA-p-100-IP66** actuators (nominal: 30 A, peak: 48 A). Connect each actuator to a single POWER OUTPUT port(Molex Microfit connectors),
- 2nd module handles 3 **MA-p-100-IP66** actuators (nominal: 30 A, peak: 48 A). Connect each actuator to a single POWER OUTPUT port(Molex Microfit connectors),
- 3rd module handles 9 **MA-p-60-IP66**(nominal: 32.4 A, peak: 49.6 A). Connect three daisy-chained actuators to a single POWER OUTPUT port(Molex Microfit connectors).

---

2. BR Modules (Braking Resistor)

In a battery-powered system, the battery can absorb the recovered energy, and actuators can often dissipate energy during overvoltage conditions, making additional braking resistors generally unnecessary. However, under certain conditions—such as operating with heavy loads or when the battery is fully charged—there is a risk of overvoltage. In these cases, installing a BR module can be a crucial mitigation strategy.

In the analyzed example, there is risk related to the robotic arm placed on the top of the robot:

Configuration:

- Base DOFs (3 Actuators):
  - DOF 1: Rotation around the robot's vertical axis.
  - DOF 2: Tilting in one axis.
  - DOF 3: Tilting in a perpendicular axis, moving the base joint of the arm.
- Joint DOFs (3 Actuators):
  - DOF 4: Movement of the upper arm joint.
  - DOF 5: Movement of the forearm joint.
  - DOF 6: Movement of the wrist joint.

Mitigation Strategy for Overvoltage:

1. Group the Actuators with High Energy Dissipation Potential
   - Joints 2, 3, and 4 typically experience higher dynamic loads and can generate significant energy during deceleration or sudden stops. These actuators should be grouped on the same PS module.
2. Install a BR Module for the Selected PS
   - Add a BR module to the PS managing these joints to ensure excess energy generated during regenerative braking is safely dissipated.
3. Reason for Omission of DOF 1:
   - The actuator for DOF 1, which rotates the robot around its vertical axis, generates relatively low regenerative energy compared to the joint actuators. It can operate safely without requiring a BR module.

Calculation Example for DOFs 2, 3, and 4:

- Nominal Power Requirements:
  Each actuator operates at nominal power during regular motion, but the braking phase can generate energy proportional to its dynamic load.
- Peak Energy Dissipation During Braking:
  When all three joints brake simultaneously, the generated peak power could overload the battery or the PS without a BR module. For example, if each actuator generates up to 100 W during braking, the total energy dissipation required is 300 W.

  Solution: Install a BR module for the PS serving these joints. The maximum dissipation power of the module is 420W.

---

3. IC Modules (Isolated DC/DC Converters)

Converter Selection:

- **GPU (60 W):** Use a dedicated converter. Options:
  - **12 V / 5 A** (I = 60W/12V = 5A) or **15 V / 4 A** (I = 60W/15V = 4A)
- **LIDAR + 5G Router** (Combined 45 W):
  - Choose a 24 V / 2.5 A converter (I = 45W/24V = 1.9A).

Solution: Use 2 IC modules:

- 1 for GPU, 12V / 5A,
- 1 for LIDAR + Router, 24 V / 2.5 A.

---

4. CTRL Module

Modules in Use:

- 3 PS modules,
- 1 BR module,
- 2 IC modules.

Total Slave Modules: 6 (max number of the slave modules per CTRL is 6).

Solution: Use 1 CTRL module to manage the system.

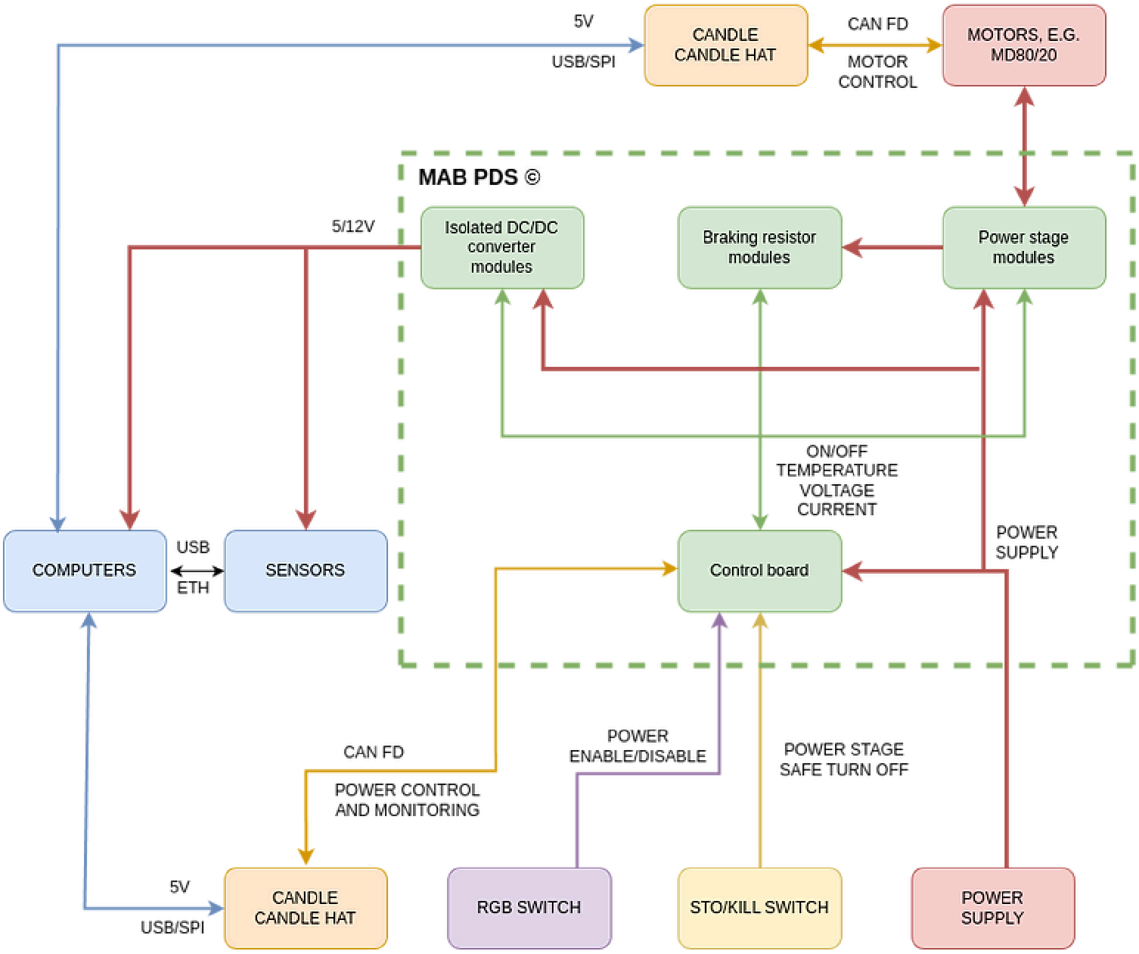
If you need assistance to choose the proper setup, contact us for support.

3.2 Mechanical integration

For the requested PDS configuration, we provide a complete CAD model of the system. You can find the 3D model of each module, and example configurations here. The stack is assembled using M3 standoffs. Mounting holes with the appropriate spacing are required to integrate the system into your construction. Installation is straightforward and involves securing the system to the frame with M3 screws, ensuring a robust connection. It is essential to position the system in an area with adequate airflow to prevent overheating, especially in applications where braking resistors are heavily utilized during overvoltage events.

3.3 Electrical integration

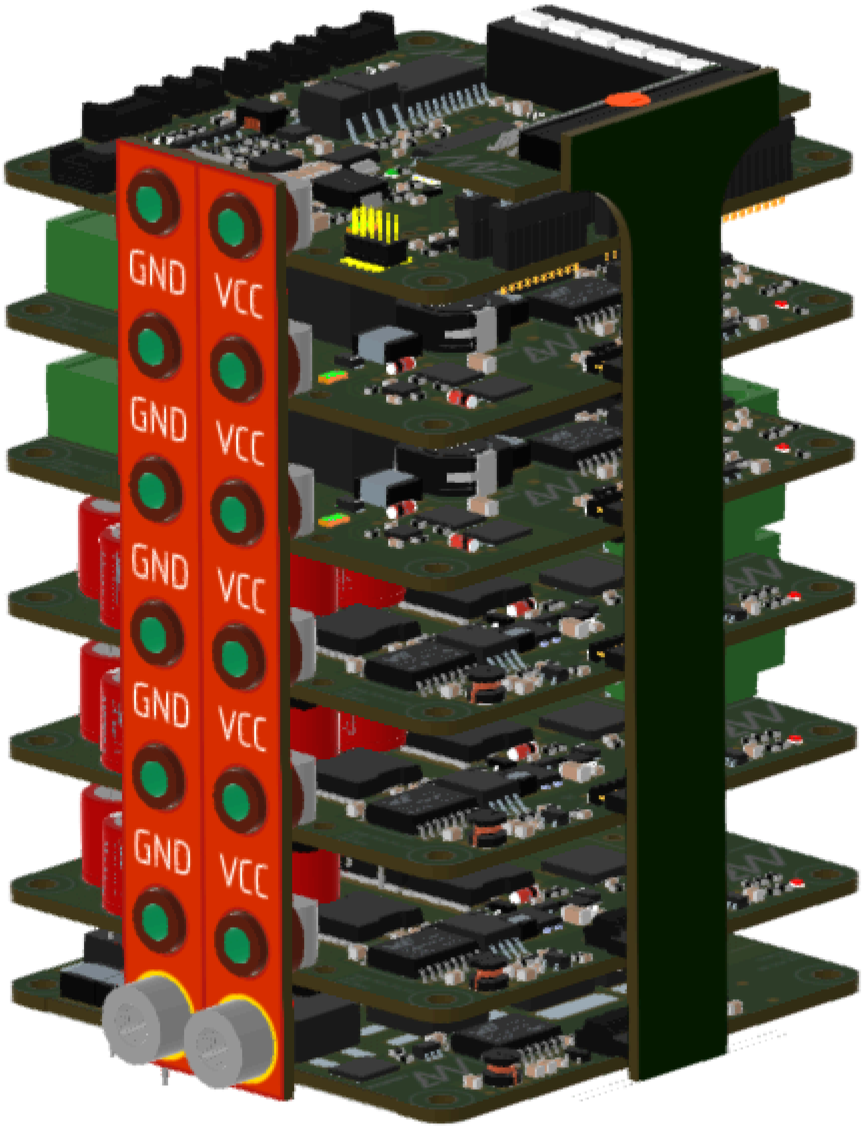
The PDS module serves as an intermediary between the power supply and the robot's electrical system. Below is an example integration diagram for reference:



As can be seen on the scheme diagram, the PDS ports that need to be wired are:

- CANFD
- STO/KILL SWITCH
- POWER SUPPLY
- ISOLATED CONVERTER OUTPUTS
- POWER STAGES OUTPUTS

The connection of signals between the CTRL and slave modules is established via the SIGNAL BAR, which is included with the ordered PDS stack. Power distribution across the stack is managed using the integrated POWER BUS BAR. Both bars are pre-installed to ensure seamless integration. Below is a visualization of an example stack featuring the mentioned bars.



Together with the PDS, a whole setup of wires is mentioned in tables in the “connectors list” in all modules. The last cable that is included, is the power cable described below:

Power cable specification	**Wiring details**
Silicon 1,5 mm2 RED - V+ BLACK - GND Ring terminal on the one side, leaded wires on the second side length 0,5m	Connect the leaded terminals to the device power supply, like the battery or bench supply. The cable is mounted to the POWER BUS BAR with M5 screws. <u>For quick start and evaluation purposes only. MAX current 16A.</u>

3.4 FDCAN protocol

3.4.1 Overview

Communication with a PDS device is done with the use of the “properties” concept. Each data or setting is a property of a module that forms a PDS device stack. Each property has its own underlying data type and access rights. Each property data occupies 4 bytes on the CAN frame. When communicating with the PDS, the host controller has to form a request frame that contains:

- Message type (which indicates what the host device is about to achieve),
- Type of the module that the message is addressed to,
- Socket index where the particular module is connected,
- Number of properties that it is going to read or write,
- A set of properties (when reading) or a property/value pairs (when writing).

NOTE:
As in all MAB CAN-based devices, the CAN ID is used to target specific devices. PSD is not an exception.

When the frame is successfully received and processed by the PDS device, its response body depends on the message type.

When the host sends a request that writes the properties to the PDS module, it will respond with a frame that contains the protocol status code, the number of properties received (for confirmation purposes), and the status code for each property that the host device was about to write.

Please note that it is possible that the host will try to write multiple properties where a few of them have the “read-only” access rights. In such a case the status codes for properties that were written properly will have the “OK” status, and “NO ACCESS” for those with read-only access rights.

To standardize the form of communication with all parts of the PDS device, the Control board is also considered as a module (from the protocol perspective) and is always assigned to the virtual socket with index 0. If the host wants to communicate with the control board it should simply choose the control board module type and 0 as its socket index.

When the host sends a request to read properties the PDS device responds similiary with the protocol status code, number of properties and a set of property status / value that corresponds to the properties set in the request.

Please refer to the “Communication frames” section for examples.

3.4.2 Commands

The message frame body depends on the command code that is always placed at the beginning of the frame. For simplicity, there are only two command codes supported by the PDS device:

Command name	Command code [HEX]	Description
Read properties	0x20	Read properties from the PDS module
Write properties	0x21	Write properties to the PDS module

3.4.3 Module type & Socket index

Command code is followed by a pair of module type code and socket index number. To prevent mistakes, when creating a frame, the correct type and socket index must be provided. This brings a solution to address a particular module even if there is more than one module of the same type connected to the control board and also allows the PDS to reject the message if there is no module of the desired type at the given socket index.

Module name	Type code
Control board	0x01
Brake resistor	0x02
Isolated Converter	0x03
Power Stage	0x04
Undefined	0x00

Socket indexing is quite straightforward. Socket number 1 has the corresponding index 1, etc…
The maximum number of sockets is 6. Since the control board module is the root one we are communicating with over the FDCAN and other PDS modules are connected to it, it is always available at socket index 0.

3.4.4 Properties

3.4.4.1 Overview

Properties are the values representing some state or parameter of the module like temperature, voltage, CAN Baudrade, or module type connected to socket number 5. Obviously, not all properties are available on each module and also there are some common properties that each module must have. For example:

- STATUS_WORD and TEMPERATURE properties are available on all modules,
- CAN Baudrate and submodules connectivity properties are exclusively available on the control board (The only one that has CAN Interface and the one that we are connecting submodules to),
- Output current/power properties are available on both the Power stage and Isolated converter modules as those modules are able to measure the current / voltage at its output,
- The brake resistor index is available on the Power stage module as they are intended to be bound together in runtime.

Above are only a few examples. Refer to the particular module section for available properties and their details.

Each property has its own ID number, underlying data type and access rights. For simplicity, the property ID numbering is global so the same property on every module will have the same ID.
This limits the overall property number in the whole system to 256.

Regardless of the underlying data type, property values are transferred on the 32-bit placeholders.
Refer to the communication frames for details.

3.4.4.2 Properties summary table

NAME	ID	ACCESS RIGHTS	TYPE	DESCRIPTION	AVAILABILITY ON MODULES			
					CB	PS	BR	IC
STATUS WORD	0x00	R	U32 - btiwise	A register were each bit represent different state of the module	✅	✅	✅	✅
STATUS CLEAR	0x01	W	U32 - bitwise	A register used to clear status bits in the STATUS WORD	✅	✅	✅	✅
ENABLE	0x02	RW	Boolean	Used to enable / disable particular module (when write) or simply check if it is enabled / disabled (when read)	✅	✅	✅	✅
TEMPERATURE	0x03	R	Float	Module temperature	✅	✅	✅	✅
TEMPERATURE LIMIT	0x04	RW	Float	Temperature limit. When module temperature exceeds this value it will become disabled and unable to be enabled back until cooldown	✅	✅	✅	✅
BUS VOLTAGE	0x05	R	U32	Power supply input bus voltage on control board / Power output voltage on power stage and Isolated converter	✅	✅	❌	✅
...	...	...	...	...	...	...	...	...
LOAD CURRENT	0x10	R	S32	Output current of the module	❌	✅	❌	✅
LOAD POWER	0x11	R	S32	Output power of the module	❌	✅	❌	✅
TOTAL DELIVERED ENERGY	0x12	R	U32	Total amount of energy (in WH) that was consumed by device connected to the module	❌	✅	❌	✅
TOTAL RECUPERATED ENERGY	0x13	R	U32	Total amount of energy (in WH) that was generated by the device connected to the module	❌	✅	❌	❌
...	...	...	...	...	...	...	...	...
CAN ID	0x20	RW	U16	CAN ID	✅	❌	❌	❌
CAN BAUDRATE	0x21	RW	ENUM*	CAN Baudrate (1M / 2M / 5M / 8M)	✅	❌	❌	❌
SOCKET 1 MODULE	0x22	R	ENUM*	The type of the module that is connected to socket number 1	✅	❌	❌	❌
SOCKET 2 MODULE	0x23	R	ENUM*	The type of the module that is connected to socket number 2	✅	❌	❌	❌
SOCKET 3 MODULE	0x24	R	ENUM*	The type of the module that is connected to socket number 3	✅	❌	❌	❌
SOCKET 4 MODULE	0x25	R	ENUM*	The type of the module that is connected to socket number 4	✅	❌	❌	❌
SOCKET 5 MODULE	0x26	R	ENUM*	The type of the module that is connected to socket number 5	✅	❌	❌	❌
SOCKET 6 MODULE	0x27	R	ENUM*	The type of the module that is connected to socket number 6	✅	❌	❌	❌

SHUTDOWN_TIME	0x28	RW	U32	The time (in mS) that PDS device waits before it turns off after receiving shutdown request (either from power button or from host device)	✅	❌	❌	❌
BATTERY_LEVEL_1	0x29	RW	U32	Levels (in mV) that defines ranges of battery energy available: 0 - L1 :: Almost discharged (RED indication) L1 - L2 :: Nominal charge (YELLOW indication) L2 < :: Fully charged (GREEN indication)	✅	❌	❌	❌
BATTERY_LEVEL_2	0x2A	RW	U32		✅	❌	❌	❌
...	...	...	...	...	...	...	...	...
BR SOCKET INDEX	0x30	RW	ENUM*	Socket index of the brake resistor that we are going to bind with the power stage	❌	✅	❌	❌
BR TRIGGER VOLTAGE	0x31	RW	U32	The value of the power stage output voltage above which the brake resistor will be triggered (in mV)	❌	✅	❌	❌
...	...	...	...	...	...	...	...	...
OCD_LEVEL	0x40	RW	U32	The value of current (in mA) above which the the module will enter OCD state if this value is exceeded for a time equal or grater than the "OCD Delay" time of the same module.	❌	✅	❌	✅
OCD_DELAY	0x41	RW	U32	The time (in mS) that the OCD level has to be excedeed for to turn the module into OCD state	❌	✅	❌	✅
...	...	...	...	...	...	...	...	...
HW_VERSION	0xFD	R	ENUM*	Hardware version of the module	✅	✅	✅	✅
FW_VERSION	0xFE	R	U32 - bytewise	Firmware version of the Control board MCU	✅	❌	❌	❌
COMMAND	0xFF	W	ENUM*	Property used to send various commands to the PDS device like (power off, reboot, etc...). See	✅	❌	❌	❌

- CB - Control board ; PS - Power stage ; BR - Brake resistor ; IC - Isolated converter,
- Access rights: R - Read only ; W - write only ; RW - Read & write,
- ENUM* types and their coding are described within each property description.

3.4.4.3 Properties detailed description

- **Status and status clear words**

The terms 'Status' and 'Status Clear' indicate basic binary states or events within a module, such as whether it is enabled or a limit has been exceeded. The U32 bitwise type means that each bit in the value represents a different purpose rather than a singular numerical value. Like properties, status bits are module-specific—each module defines its own set, which may or may not align with those of other modules.

NAME	BIT POSITION	DESCRIPTION	AVAILABILITY ON MODULES			
			CB	PS	BR	IC
Enabled	0	Indicates whenever the module is enabled or not. Note that in case of Control board it indicates entire device power on status. This flag indicates the same information as the "Enable" property.	✅	✅	✅	✅
Over temperature	1	The temperature limit of the module has been exceeded	✅	✅	✅	✅
Over current	2	The current limit of the module has been exceeded (OCD state)	❌	✅	❌	✅
...	...		...	...	...	...
STO1	10	Safe Torque Off at input 1 has been triggered	✅	❌	❌	❌
STO2	11	Safe Torque Off at input 2 has been triggered	✅	❌	❌	❌
FDCAN Timeout	12	CAN message processing timeout	✅	❌	❌	❌
Submodule 1 error	13	Some error occurs on the module connected to the socket (1 - 6). Refer to the particular module status word for more details about the issue	✅	❌	❌	❌
Submodule 2 error	14		✅	❌	❌	❌
Submodule 3 error	15		✅	❌	❌	❌
Submodule 4 error	16		✅	❌	❌	❌
Submodule 5 error	17		✅	❌	❌	❌
Submodule 6 error	18		✅	❌	❌	❌
Charger detection	19	Indicates whenever charger has been connected to the device	✅	❌	❌	❌
Shutdown schedule	20	Indicates that the PDS device shutdown has been scheduled and the PDS device is about to be turned off within the time specified in the "Shutdown time" property of the control board. This flag could be used by host controlers to detect that they are going to be disabled soon and could handle user-defined shutdown procedures.	✅	❌	❌	❌
...	...		...	...	...	...

- **Temperature (float, read-only)**

  Module temperature sensor reading in ℃

- **Temperature limit (float, read-write)**

  When the temperature property exceeds this limit, the “Over temperature” bit in status word will be set and the module will stop operating.

- **Bus voltage**

  This property is the result of voltage measurements on different system points depending on the module from which it is being read:
  - On the **Control Board,** it is the voltage that supplies the PDS Device (for example the battery voltage)
  - On the **Power stage,** it is the voltage on the output from the module. It might but not have to be the same as input voltage. Note that the power stage module provides the ideal diode behavior when disabled and the voltage on its output might come from recuperated energy from drives. This voltage is also used to trigger the brake resistor connected to the power stage.
  - On the **Isolated Converter,** it is the output voltage from the DC/DC Converter (12V or 5V depending on the chosen IC Variant)

- **Load current**

  Holds the value of current that is drained by the devices connected to the output of a module.

- **Load power**

  The momentary power at the output of the module.

- **Total delivered energy**

  Accumulated positive power in the time domain (in mWH)

- **Total recuperated energy**

  Accumulated negative power in the time domain (in mWH)

- **CAN ID**

  This property is used to change the CAN ID of the PDS device

- **CAN Baudrate**

  CAN Baudrate of the control board. The table below shows available baud rate codes that could be passed as the property value:

CODE	BAUDRATE
0x00	1M
0x01	2M
0x02	5M
0x03	8M

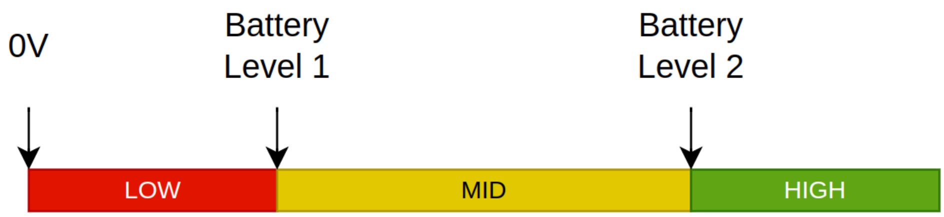
- **Socket X module**

  The value of this property (x : [1 - 6]) is a module type code (see Module type & Socket index section for details). Its purpose is to get the module type connected to the particular socket.

- **Shutdown time (u32 / read-write)**

  The time (in mS) that the PDS device waits before it turns off after receiving a shutdown request (either from the power button or from the host device)

- **Battery level 1 & 2 (u32 / read-write)**
  User-defined battery voltage levels that determine the LOW / MID / HIGH Battery energy ranges. These levels are used by the battery monitor state (short RGB switch press). Note that because the PDS device can operate in a wide range of supply voltage and sources (not limited to batteries) by default both levels are set to 0V so if not configured the Battery monitor will always indicate HIGH battery energy. When setting up these levels make sure that correct order has been maintained. The property write operation will result in an **INVALID_DATA** error if the user tries to set up battery level 2 lower than level 1 or the opposite, so when overwriting default values, ensure that level 2 will be set up before level 1.

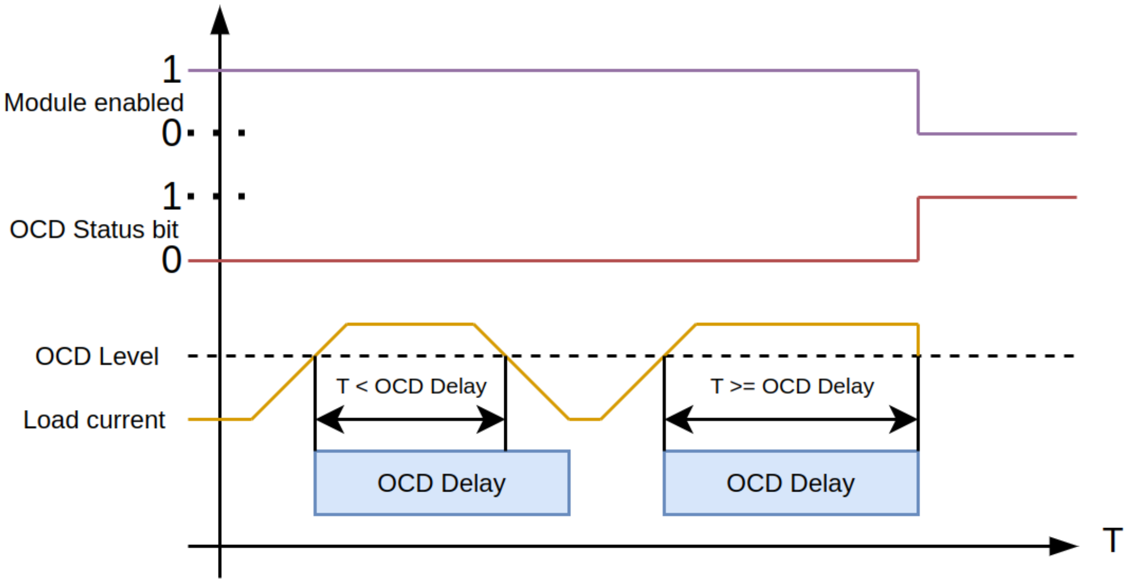


- **Brake resistor Socket index**

  The value of this property is a socket index code. It is available on the Power stage module only and is used to bind the Power Stage module with the brake resistor module (Power Stage and Brake resistor modules have special power connections and this connection is reflected in software with this property. Please look at 2.3.5.3 paragraph for details)

- **Brake resistor trigger voltage**
  The voltage level (in mV) above which the Brake resistor will be triggered. Note that the BR has to be bound with PS first to make use of this property. This property is also exclusively available on the power stage and the power stage takes full control over the brake resistor. Although the user is theoretically able to turn the BR on independently, it is not recommended! BR is intended to be used only in companion with the Power Stage module.

- **OCD level & delay**
  The graph below describes the relation between the OCD level, delay and status bit for a module. In simple words if the load current exceeds the OCD level for a time greater then OCD delay, the OCD status bit is set and the module stops operating until the host device explicitly clears the OCD status bit and reenable the device.



- **HW & FW Version**
  HW Version indicates the PCB revision of the module in use (including Control board PCB). FW Version is related to the Firmware that drives while PDS device and is available to read only on control board

- **Command**
  This is a special purpose, Control board inclusive property to send various commands to the PDS device.

Command	Code	Description
NULL	0x00	NULL command. Could be used to ping the device. (No processing - fast response)
SHUTDOWN	0x01	Schedule PDS device shutdown. The device will turn off after time specified within the "Shutdown time" property of the control board
REBOOT	0x02	Reboot the device
SAVE CONFIG	0x03	Save current configuration in the device non volatile memory.

3.4.4.4 Response status codes

Response to correct read or write request frames contains two different status code types. The overall message status code (protocol status) and per property status code (property status).

Overall message status codes:

Status	Code [HEX]	Description
OK	0x00	Request frame parsed correctly
Invalid message length	0x01	The received request frame length is incorrect. For example when the number of properties byte indicates 4 properties but only 3 types are provided. Or if the received frame length is less than the expected header.
Invalid module type	0x02	Module type code is out of range
Invalid socket index	0x03	Socket index code is out of range
No module at socket	0x04	There is no module connected to the desired socket index
Wrong module at the socket	0x05	The desired module type does not match the one actually connected to the desired socket

Properties status codes:

Status	Code [HEX]	Description
OK	0x00	Property read/write operation success
No property	0x01	The addressed module does not support the requested property
Invalid access rights	0x02	Requested read operation from write-only property or write operation to read-only property
Invalid data argument	0x03	Given data value during the write operation is invalid. For example when the user is trying to exceed some limit or pass some value that is out of valid enumeration range.
…	…	…

3.4.4.5 Communication frames

Generic frame body of the "Write property" (0x21) message:

nBytes	1	1	1	1	1	4	1	4	...	1	4	1	4
Frame field	Write property command code [0x21]	Module type	Socket index	Number of properties	Property 1 type	Property 1 value	Property 2 type	Property 2 value	...	Property N-1 type	Property N-1 value	Property N-type	Property N value

The first row indicates the number of bytes occupied by the corresponding frame field.

Generic response to properties writes frame body:

nBytes	1	1	1	1	...	1	1
Frame field	STATUS	Number of properties	Property 1 Status	Property 2 Status	...	Property N-1 Status	Property N Status

Generic frame body of the "Read properties" (0x20) message:

nBytes	1	1	1	1	1	1	...	1	1
Frame field	Read properties command [0x20]	Module type	Socket index	Number of properties	Property 1 type	Property 2 type	...	Property N-1 type	Property N-type

Generic response to properties read frame body:

nBytes	1	1	1	4	1	4	...	1	4	1	4
Frame field	STATUS	Number of properties	Property 1 Status	Property 1 value	Property 2 Status	Property 2 value	...	Property N-1 Status	Property N-1 value	Property N Status	Property N value

Please note that there are no additional data integrity-checking mechanisms added to the frames. The underlying FDCAN interface handles all those features.

3.4.4.6 Communication scenario examples

For all examples below the assumed physical setup of the PDS device is as follows:

Socket 1 :: Power stage module
Socket 2 :: Power stage module
Socket 3 :: No module
Socket 4 :: Brake resistor
Socket 5 :: No module
Socket 6 :: Isolated Converter 12V

1. Reading the submodules stack

Request frame:

1	2	3	4	5	6	7	8	9	10
0x20	0x01	0x00	0x06	0x22	0x23	0x24	0x25	0x26	0x27

Request frame breakdown:

Byte	Value	Meaning
1	0x20	"Read property" message ID
2	0x01	Control board module type. (Submodules assignments to socket are the Control board exclusive properties.)
3	0x00	Socket index 0 means that we are targeting the control board
4	0x06	We are about to read 6 different properties
5	0x22	Socket 1 module property ID
6	0x23	Socket 2 module property ID
7	0x24	Socket 3 module property ID
8	0x25	Socket 4 module property ID
9	0x26	Socket 5 module property ID
10	0x27	Socket 6 module property ID

If the request frame was successfully received and parsed by the PDS device it should respond with the following response frame:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0x00	0x06	0x00	0x05	0x00	0x00	0x00	0x00	0x05	0x00	0x00	0x00	0x00	0x00	0x00	0x00

17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
0x00	0x00	0x02	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x03	0x00	0x00	0x00

Response frame breakdown:

Byte	Value	Meaning
1	0x00	Message status code OK (no errors)
2	0x06	6 properties was read
3	0x00	1st property status code OK
4 - 7	0x00000005 (Little endian)	1st socket module type == Power Stage
8	0x00	2nd property status code OK
9 -12	0x00000005 (Little endian)	2nd socket module type == Power Stage
13	0x00	3rd property status code OK
14 - 17	0x00000000 (Little endian)	3rd socket module type == NO MODULE
18	0x00	4th property status code OK
19 - 22	0x00000002 (Little endian)	4th socket module type == Brake resistor
23	0x00	5th property status code OK
24-27	0x00000000 (Little endian)	5thsocket module type == NO MODULE
28	0x00	6th property status code OK
29 - 32	0x00000003 (Little endian)	6th socket module type == Isolated converter

2. Binding Brake Resistor with the Power Stage

Request frame:

1	2	3	4	5	6	7	8	9	10	11	12	13	14
0x21	0x04	0x02	0x02	0x30	0x00	0x00	0x00	0x04	0x31	0x00	0x00	0x00	0x04

Request frame breakdown:

Byte	Value	Meaning
1	0x21	“Write property” message ID
2	0x04	Power Stage module type
3	0x02	Socket index 2
4	0x02	2 different properties to write
5	0x30	Property ID: Brake Resistor Socket Index
6 - 9	0x00000004	Property data: 4 (Socket index of BR to bind)
10	0x31	Property ID: Brake Resistor trigger voltage
11-14	0x0000C350	Property data: 50 000 (BR Trigger voltage in mV)

if the request frame was successfully received and parsed by the PDS device it should respond with the following response frame:

1	2	3	4
0x00	0x02	0x00	0x00

Response frame breakdown:

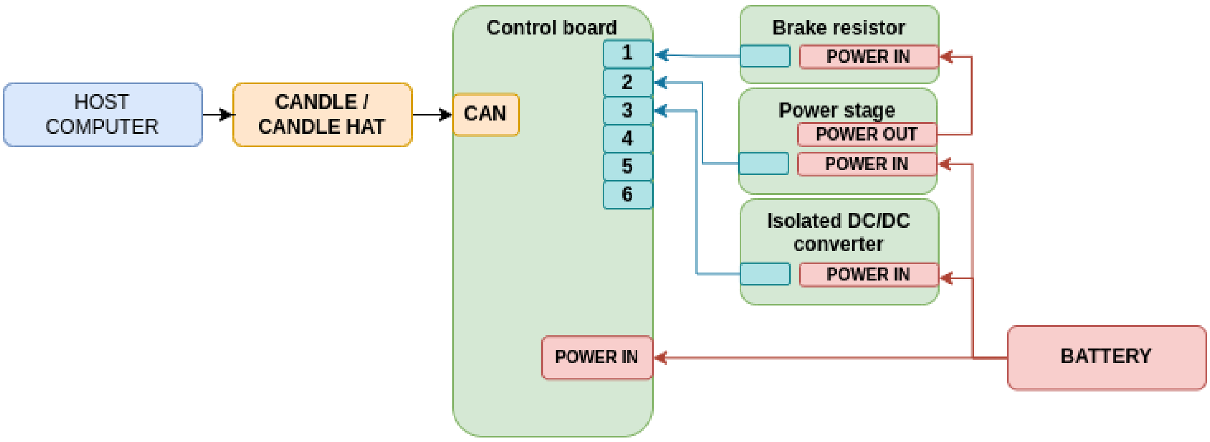
Byte	Value	Meaning
1	0x00	Message status code OK (no errors)
2	0x02	2 properties was written
3	0x00	1st property status code OK
4	0x00	2nd property status code OK

4. CANdle SDK API guide

4.1 Intro

The best way to get familiar with all the features that CANdle SDK API provides for the PDS device is to study source code (especially the PDS - examples), but here we are explaining some basic concepts of using it with our CANdleSDK C++ API.

We will start with a very simple setup. Let's get a look at the image below:



As you can see there are three different submodules connected to the Control board:

- Brake resistor (Socket index 1)
- Power Stage (Socket index 2)
- Isolated DC/DC Converter (Socket index 3)

Additionally, the Brake resistor at socket 1 is powered from the output of the Power stage at socket 2. It is important to properly bind them together what will be covered a bit later in this document.

Note that in the real application, there will be many more elements of the presented modules like output from the DC/DC Converter which is intended to supply the Host SBC, or plenty of additional inputs to the control board but for simplicity, they are hidden here.

4.2 Example 1 - Basic SDK usage

First of all, the PDS API is designed to reflect the physical setup as much as possible. Since our SDK is intended to be used in conjunction with one of our CANdle devices ([CANdle](#) or [CANdleHAT](#)) and it is the first element on the path from the perspective of the computer that will run our code, the CANdle device object is the first one that we are going to create:

```
Candle candle(mab::CAN_BAUD_1M, true);
```

Now if the CANdle object is created we can declare our PDS object. When creating an instance we have to pass the CAN ID (100 in this example) and a reference to the previously created CANdle object that the PDS is connected to:

```
constexpr u16 PDS_CAN_ID = 100;
Pds pds(PDS_CAN_ID, candle);
```

Before using the PDS device we have to call the **init()** method in order to read a hardware setup from it, so the PDS class will be able to make some assertions if we try to make some dumb operations like reading data from a non-existing module.

```
pds.init();
```

Once the PDS object is created and initialized, we can start using it. In the first example, we will simply read the submodule stack to check if it is connected as expected. For this purpose, we need a structure that holds the module types available on each socket:

```
Pds::modulesSet_S pdsModules = pds.getModules();
```

Now it is time to print it! Pds class provides a few helper methods like "moduleTypeToString" that could be used alongside with our logging feature embedded into the CANdleSDK:

```
Logger _log;
_log.m_tag = "PDS Example";
_log.info("PDS have the following numbers of connected modules:");
_log.info("\t1\t:: %s", Pds::moduleTypeToString(pdsModules.moduleTypeSocket1));
_log.info("\t2\t:: %s", Pds::moduleTypeToString(pdsModules.moduleTypeSocket2));
...
```

If everything goes fine the output in the console should look like this:

```
[PDS Example] PDS have the following set of connected modules:
[PDS Example]     Socket 1 :: ISOLATED_CONVERTER
[PDS Example]     Socket 2 :: POWER_STAGE
[PDS Example]     Socket 3 :: BRAKE_RESISTOR
[PDS Example]     Socket 4 :: NO MODULE CONNECTED
[PDS Example]     Socket 5 :: NO MODULE CONNECTED
[PDS Example]     Socket 6 :: NO MODULE CONNECTED
```

When calling the PDS class methods that reads or writes data to the PDS device we are actually communicating with its control board which similarly to the submodules also provides some other features like temperature and bus voltage measurement, or controlling the FDCAN Interface parameters. Below are a few example methods that shows how to access those features:

```
controlBoardStatus_S pdsStatus     = {0};
u32                  pdsBusVoltage = 0;
f32                  temperature   = 0.0f;


pds.getStatus(status);
pds.getBusVoltage(busVoltage);
pds.getTemperature(temperature);

_log.info("Enabled :: [ %s ]", status.ENABLED ? "YES" : "NO");
_log.info("STO2 :: [ %s ]", status.STO2_EVENT ? "YES" : "NO");
_log.info("STO1 :: [ %s ]", status.STO1_EVENT ? "YES" : "NO");
_log.info("CAN Timeout :: [ %s ]", status.FDCAN_TIMEOUT_EVENT ? "ON" : "NO");
_log.info("Over temperature :: [ %s ]", status.OVT_EVENT ? "YES" : "NO");
_log.info("Voltage :: [ %.2f ]", static_cast<float>(busVoltage / 1000.0f));
```

```
[PDS Example] Enabled : YES
[PDS Example] STO1 : ON
[PDS Example] STO2 : ON
[PDS Example] CAN Timeout : NO
[PDS Example] Over temperature : NO
[PDS Example] Bus voltage : 24960
```

4.3 Example 2 - Using the submodules

When using submodules we have to pay attention to the physical setup of our PDS device! Since we could have any set of connected submodules, we have to ensure the correct addressing of the ones that we want to use. Actually this only requires the user to provide correct socket indexe assignment when attaching particular submodule objects.

With our example physical setup the submodules addressing can look like this:

```
constexpr socketIndex_E BRAKE_RESISTOR_SOCKET_INDEX      = socketIndex_E::SOCKET_1;
constexpr socketIndex_E POWER_STAGE_SOCKET_INDEX         = socketIndex_E::SOCKET_2;
constexpr socketIndex_E ISOLATED_CONVERTER_SOCKET_INDEX = socketIndex_E::SOCKET_3;
```

As in the previous example. The first thing we want to do is to create the CANdle and PDS objects

```
Candle candle(mab::CAN_BAUD_1M, true);
Pds    pds(PDS_CAN_ID, candle);
pds.init();
```

If we want to use the submodules we cannot simply create a new instance of it but need to fetch the pointer to the already created one. This approach helps the Pds class to provide interface only to the submodules that are actually available in the system. Each type of the submodule has a dedicated attaching method. Below is the example attachment of the submodules objects:

```
   auto powerStage = pds.attachPowerStage(POWER_STAGE_SOCKET_INDEX);
   auto brakeResistor = pds.attachBrakeResistor(BRAKE_RESISTOR_SOCKET_INDEX);
   auto isolatedConverter = pds.attachIsolatedConverter12(ISOLATED_CONVERTER_SOCKET_INDEX);
```

Good practice is to check if we attached the modules correctly by comparing them against the null pointer. We can also expect some error messages if the PDS does not detect desired modules on the provided socket index.

```
if (powerStage == nullptr) exit(EXIT_FAILURE);
if (brakeResistor == nullptr) exit(EXIT_FAILURE);
if (isolatedConverter == nullptr) exit(EXIT_FAILURE);
```

Now if the submodules are properly attached, we can start configuring them! Lets start with the Power stage as it is the most “interactive” module.

First we are going to set up the protection levels of our Power Stage module. Those protection levels are: max temperature, OCD level, OCD delay and brake resistor trigger voltage.

```
powerStage->setTemperatureLimit(90.0f);                    // 90 Celsius degrees
powerStage->setOcdLevel(25000);                            // 25 A OCD level
powerStage->setOcdDelay(1000);                             // 1 mS delay
powerStage->setBrakeResistorTriggerVoltage(25000);  // 25V DC
```

Note that brake resistor trigger voltage is the Power Stage property because brake resistor does not measure the DC Bus voltage. Brake resistors are controlled by the Power stage module that they're bind to. Binding mechanism reflects the physical connection from the Power stage output to the Brake resistor power input (see the diagram at the beginning of the **Quick start** section). To bind them together we should simply call the binding method from the Power stage class:

```
powerStage->bindBrakeResistor(brakeResistor->getSocketIndex());
```

We can also set up some limits for the Brake resistor itself and Isolated converter as well:

```
brakeResistor->setTemperatureLimit(90.0f);      // 90 Celsius degrees
isolatedConverter->setTemperatureLimit(70.0f);  // 70 Celsius degrees
isolatedConverter->setOcdLevel(4000);           // 4 A OCD level
```

After we set up our modules to operate we can enable the Power stage output and monitor its parameters. We can actually monitor any property available in the system but for simplicity this example will only cover Power Stage parameters. Below are a few examples of reading Power Stage properties:

```
powerStage->enable();
sleep(1);
```

Because of the quite big capacitance at the Power Stage module output it is wise to wait some time before reading its data. Here we are waiting for 1 second but 50ms should be fair enough. Otherwise the first readings after enabling this module could be taken when the output voltage is still rising and their values might be misunderstood.

```
PowerStage::status_S powerStageStatus = {};
float                temperature      = 0.0f;
u32                  outputVoltage    = 0;
s32                  outputCurrent    = 0;

powerStage->getStatus(powerStageStatus);
powerStage->getOutputVoltage(outputVoltage);
powerStage->getLoadCurrent(outputCurrent);

_log.info("Power stage");
_log.info("Enabled :: [ %s ]", powerStageStatus.ENABLED ? "ON" : "OFF");
_log.info("Over current event :: [ %s ]", powerStageStatus.OCD_EVENT ? "YES" : "NO");
_log.info("Over temperature event :: [ %s ]", powerStageStatus.OVT_EVENT ? "YES" : "NO");
_log.info("Temperature :: [ %.2f ]", temperature);
_log.info("Voltage :: [ %.2f ]", static_cast<float>(outputVoltage / 1000.0f));
_log.info("Current :: [ %.2f ]", static_cast<float>(outputCurrent / 1000.0f));
```

4.4 Example 3 - Battery monitor, configuration save and shutdown procedure

This example demonstrates how to set up a battery voltage levels for the battery monitor feature, how to save the current configuration in device memory and how to handle the device shutdown procedure.

Battery voltage levels (See section 3.4.4.3 Properties detailed description or …)
(From 0V to Level 1: Red indication / battery low, From Level 1 to Level 2: Yellow indication / battery normal, From Level 2 : Battery full)

```
constexpr u32 BATTERY_LEVEL_1 = 20000; // 20V
constexpr u32 BATTERY_LEVEL_2 = 24000; // 24V

constexpr u32 SHUTDOWN_TIME = 5000; // 5 seconds
```

Init stage:

```
Candle candle(mab::CAN_BAUD_1M, true);
Pds    pds(PDS_CAN_ID, candle);
pds.init();
```

Setting up the voltage levels:

```
pds.setBatteryVoltageLevels(BATTERY_LEVEL_1, BATTERY_LEVEL_2);
```

Now when the levels are set, the PDS RGB Switch should indicate the battery level with corresponding color.

Because the PDS Device is intended to control the power supply of the host computers that runs the application software, we need to allow the application developers to handle the shutdown procedure before PDS will ordinarily cut off the supply voltage. For this purpose when turning off the device, no matter if by the RGB switch or the API command, the PDS will delay the actual shut down for the time specified by the shutdownTime property. By default the shutdown time is set to 10 seconds. We can change this time by calling:

```
pds.setShutdownTime(SHUTDOWN_TIME);
```

In order to save the settings, so device will boot up with current configuration by default, we need to call:

```
pds.saveConfig();
```

Now it's time to let our robot rest and call the shutdown command. You can also omit this and test the behavior by manually holding the RGB switch for 2 seconds.

```
pds.shutdown();
```

PDS Device is not able to initialize the communication with the Host computer so in order to detect if our device is about to turn off we need to monitor the "Shutdown scheduled" flag in the control board status property. If it is set, the application could expect that the host will be turned off in a while and can perform additional actions before.
In real application it is good practice to have a separate thread for that purpose.

```
Pds::status_S status       = {};
while(!status.SHUTDOWN_SCHEDULED)
{
        pds.getStatus(status);
}
// userAppShutdown();
```

5. Firmware update

User can update PDS Device firmware by using the candletool* - our swiss knife CLI tool to operate with MAB device ecosystem. To update the firmware the PDS need to be connected to external computer with CANdle/CANdle Hat. The wiring should be done like in the point 2.c of the Quick start guide presented below.

```
$ candletool update pds < CAN ID > -f < PATH TO .mab FILE >
```

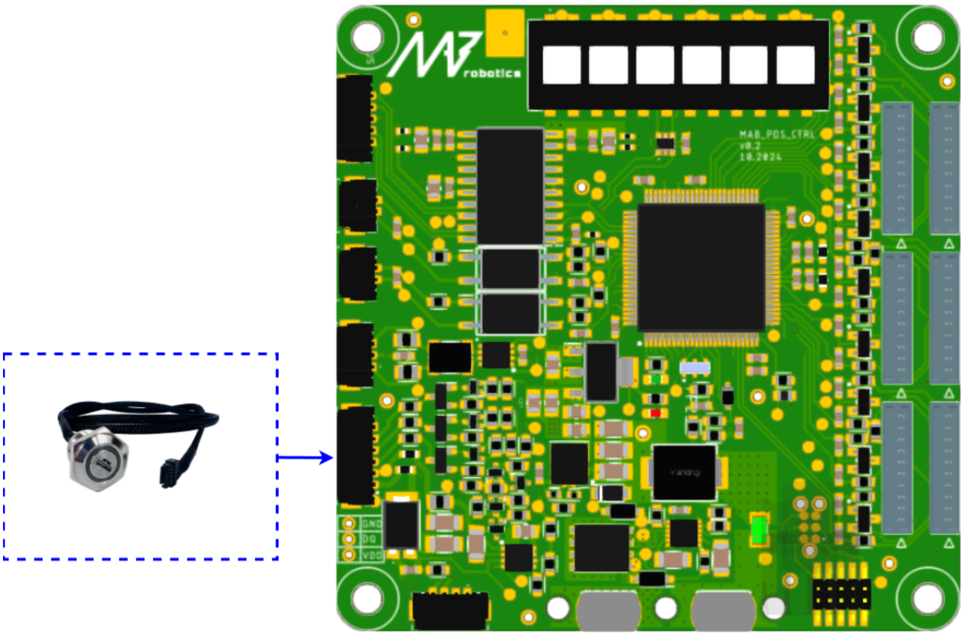
*candletool is a part of the CANdleSDK software package and it is still in development phase so distributed with limited capabilities. This application user manual is out of scope of this document.

6. Quick start guide

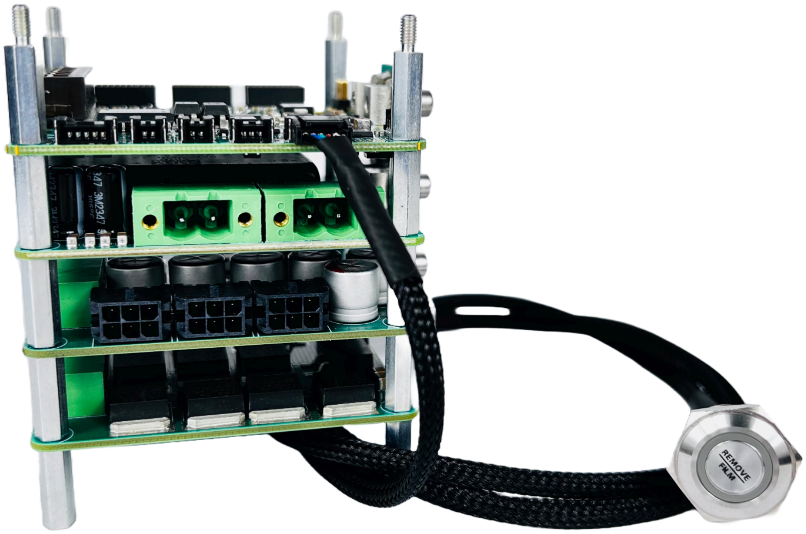
1. Mount the PDS to the frame,
2. Prepare all the harnesses delivered with PDS and connect them to PDS_CTRL:
    a. RGB SWITCH:
    Together with the PDS, there is included the switch presented below:

After connecting, the switch can be used to enable the PDS and monitor its state. Please remember to set the thresholds if the visual battery monitoring is needed.

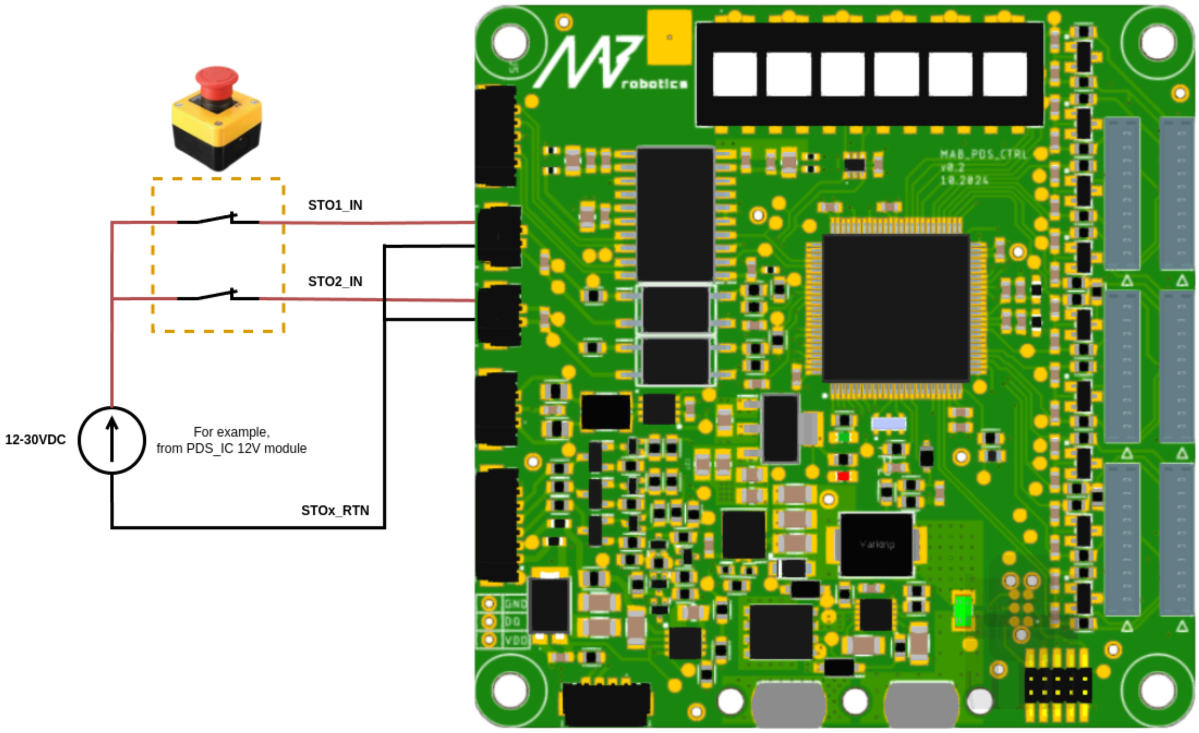
The switch need to be connected as below:



Example photo of PDS with connected switch:

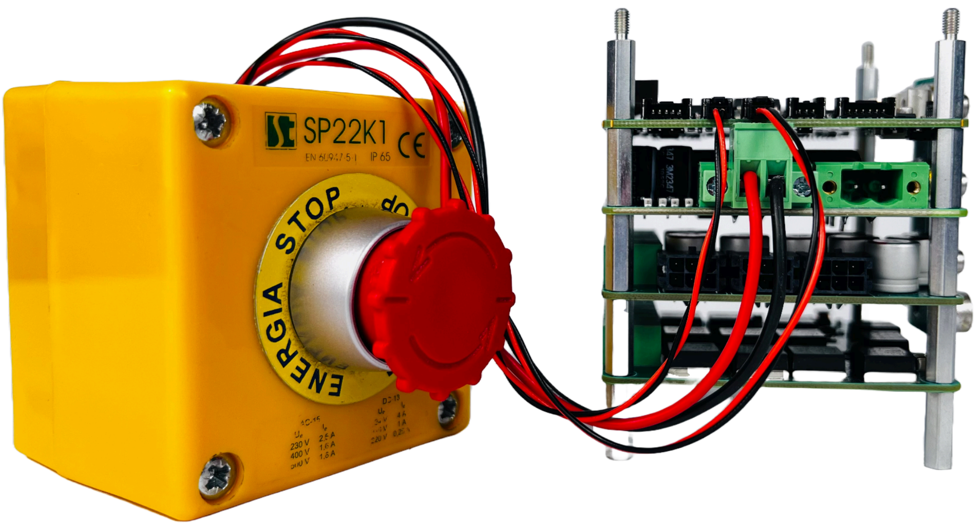


b. STO1/2
An example schematic of E-stop connection is presented below. The mechanism can be realized with pair of the normally closed contacts, or two different E-stops - like one mechanical, and the second one wireless to increase the safety. The STOx inputs need to be polarized with external voltage. It can be for example 12V from PDS_IC module, or from other power supply dedicated to safety in the application.



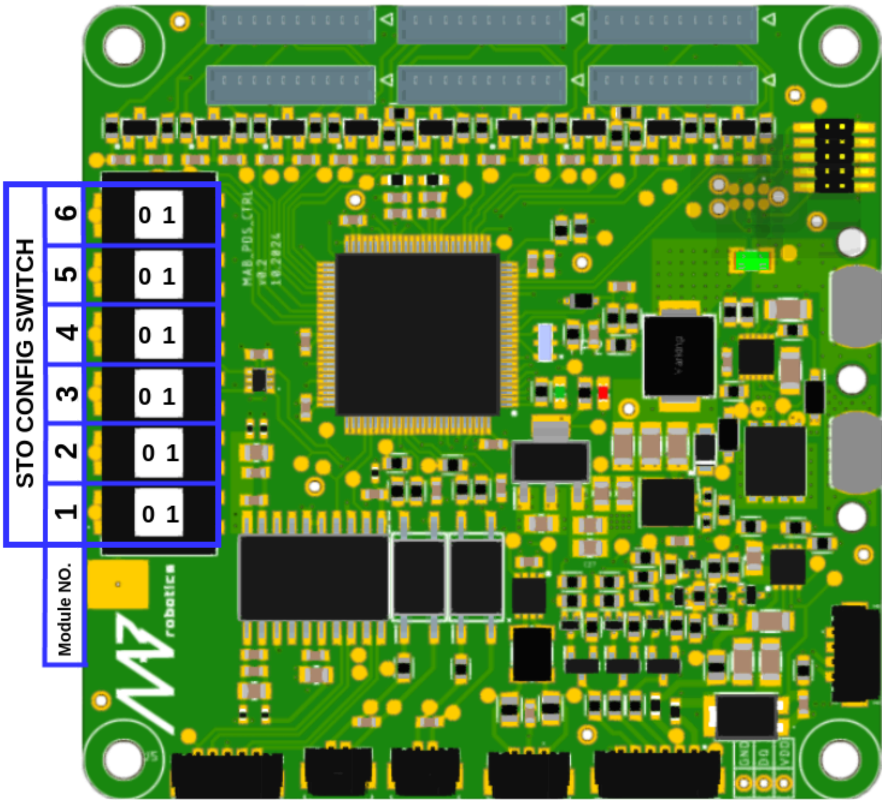
To make connection between the E-stop and PDS, use those cables:

Example wired system can look like on the photo. Please note that is only for reference.



Beside STO inputs, there is presented connection to the PDS_IC module. Such cable is not included in the set, but the housing is. The E-stop button is also NOT included. Please note that the proposed solution is for reference only - the safety considerations has to be evaluated and analyzed properly to ensure that the solution match the requirements. Usage of the voltage from the IC module for supplying the E-stop circuits can be acceptable in the end application or not, dependly on the system architecture and needs.

The last step is configuration of the STO switch. The schematic diagram of the STO configuration is presented below:



The switch determines what happens with the connected modules after STO triggering. If set to 0 (left positon), module remains on. If set to 1 (right position), the power is cut off from particular module. Table below shows the effects of setting the switch in position 1 for different modules:

Module type	Enabled STO effect
PDS_IC	The power supply is cut off from the DC/DC converter. The results can be, for example, computers shut down.
PDS_PS	The load is disconnected from the power supply. Example result can be de-energization of the connected actuators - robot will fall down as the torque is no longer generated.
PDS_BR	The energy is no longer dissipated in the resistor, independently of the voltage level on the DC bus.

For real world application, the proper strategy can be to enable STO on the PS modules, and disable on the IC and BR module. In such case, the motors are instantaneously turned off, the computers are still working, and the potential regenerative energy that can appear on the motors bus is dissipated safely. Please note that the signal of the STO triggering is transmitted by the CTRL module over the CAN FD independently on the setting on the switch. There is possibility to implement other safety strategy, like setting the torque references to 0 on the actuators, or enabling the strong damping on the joints to, for example, make robot slowly fall down in controllable way, or enable breaking by shorting the motor phases. Please remember that such mechanism can be much slower and less reliable than the mechanism based on the hardware STO circuits, which are purely hardware based and only delay comes from the signals propagation delay and the bus capacitance discharging.

c. CANFD(to CANdle/CANdle HAT)
Use that cable to connect PDS to candle:

CANdle or CANdle HAT should be connected to the CAN port:

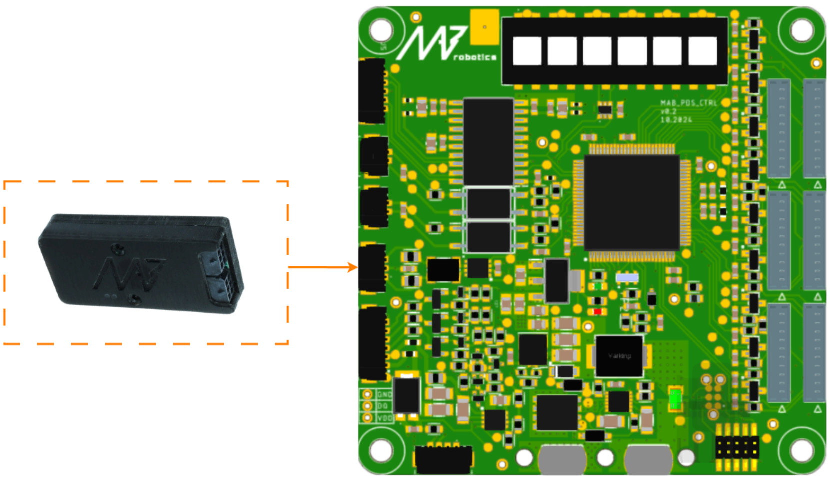
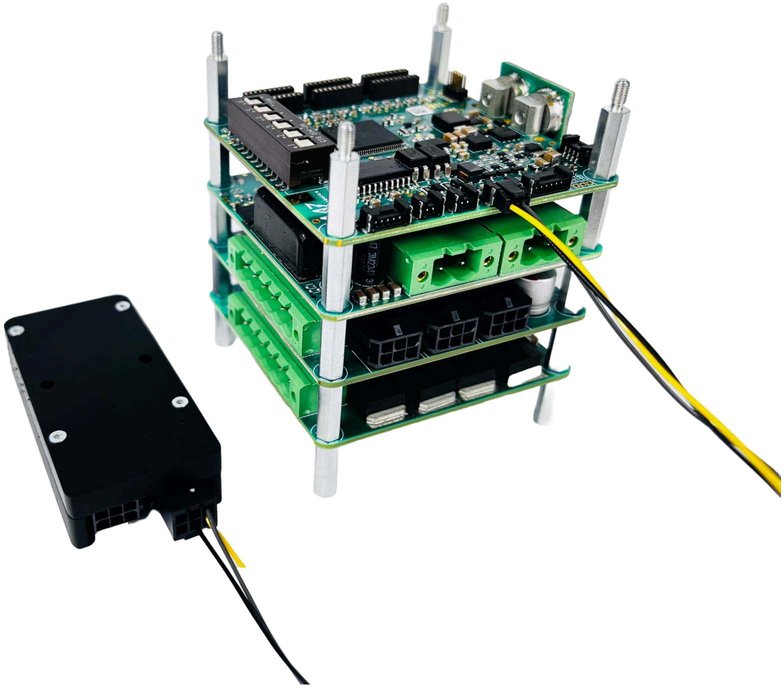
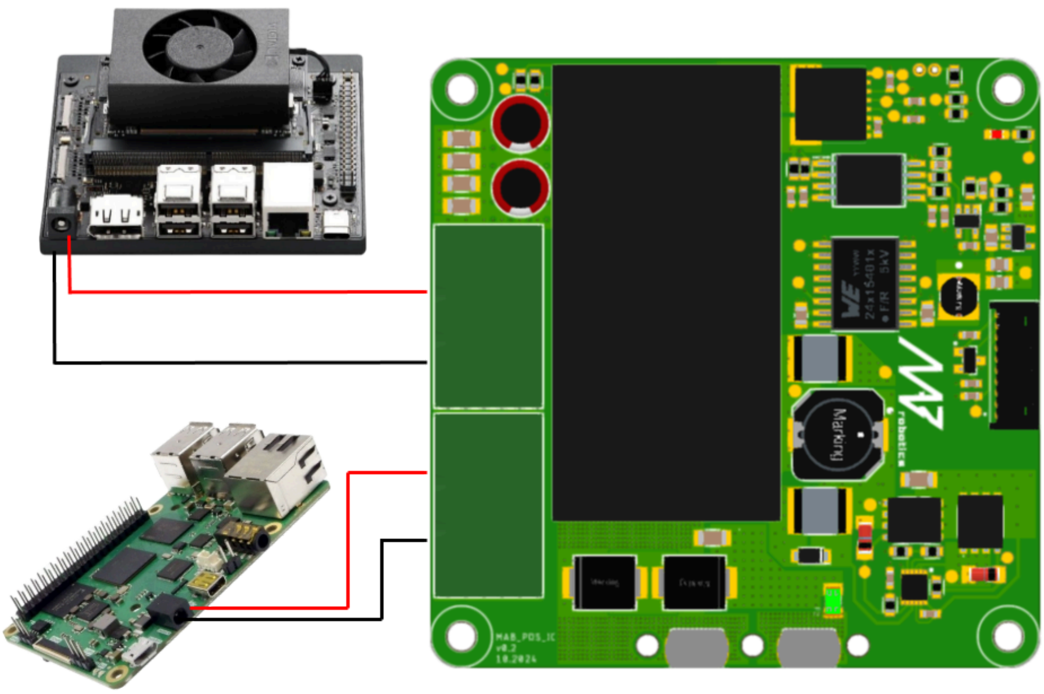


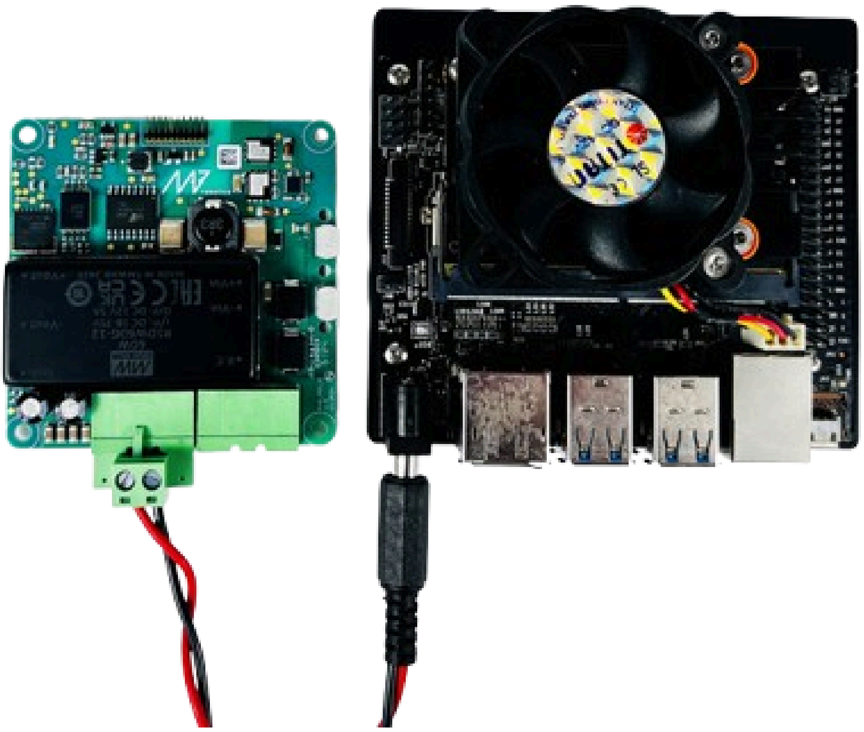
Photo of the connected modules:



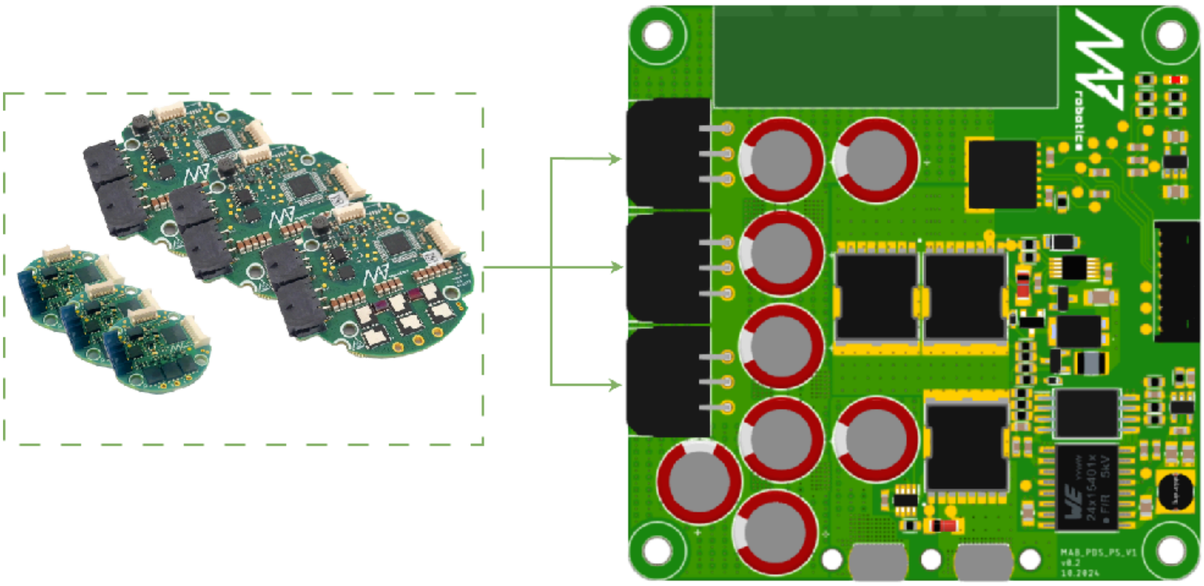
3. Prepare the harnesses for the IC modules and connect them. Ensure proper polarization before turning on the power supply to prevent damage to the load.
   The example schematic of wiring the PDS_IC module with computers:



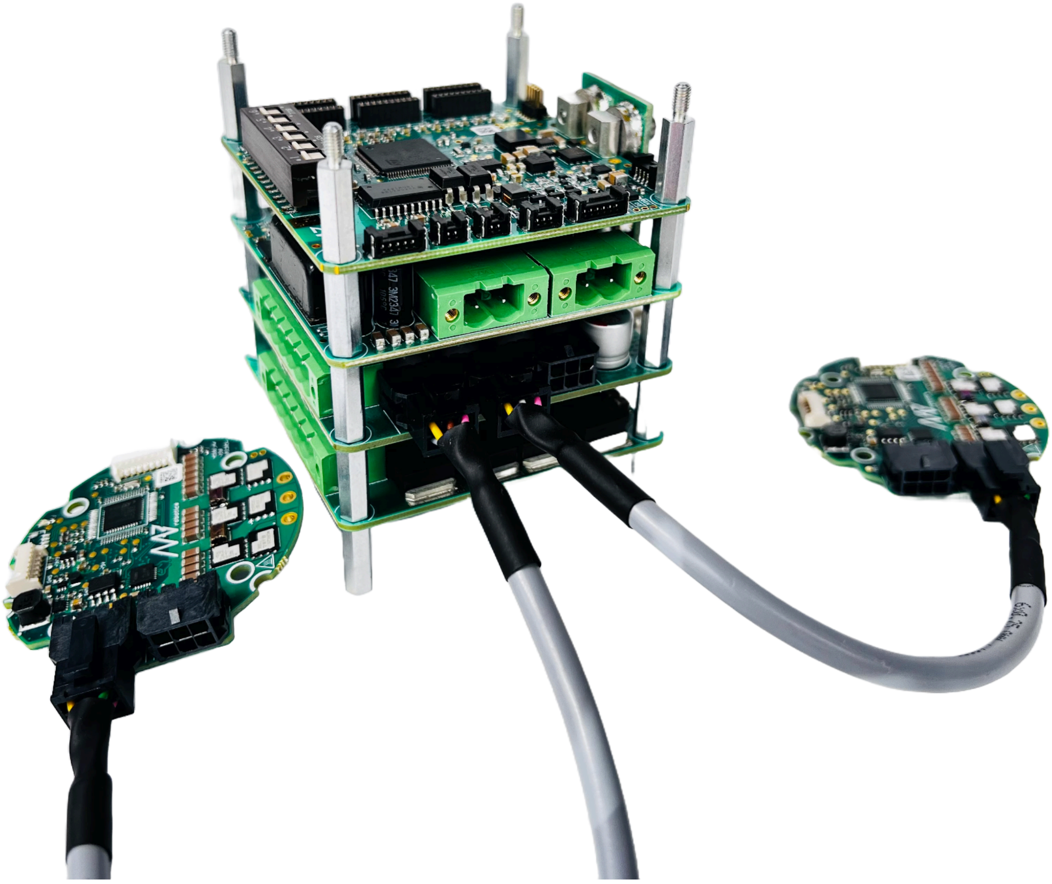
Example photo of the connection between the computer and the PDS_IC module:

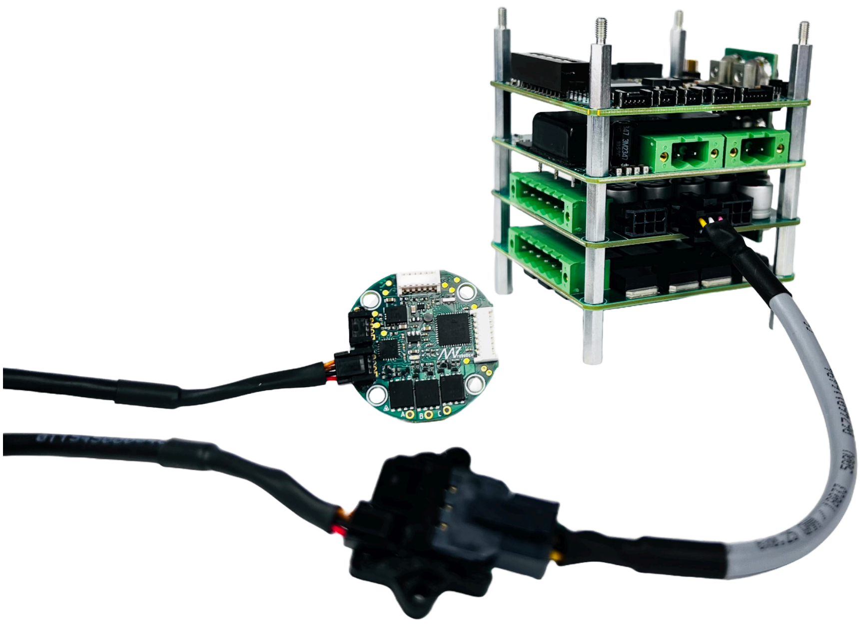


4. Connect the actuators to PS modules:



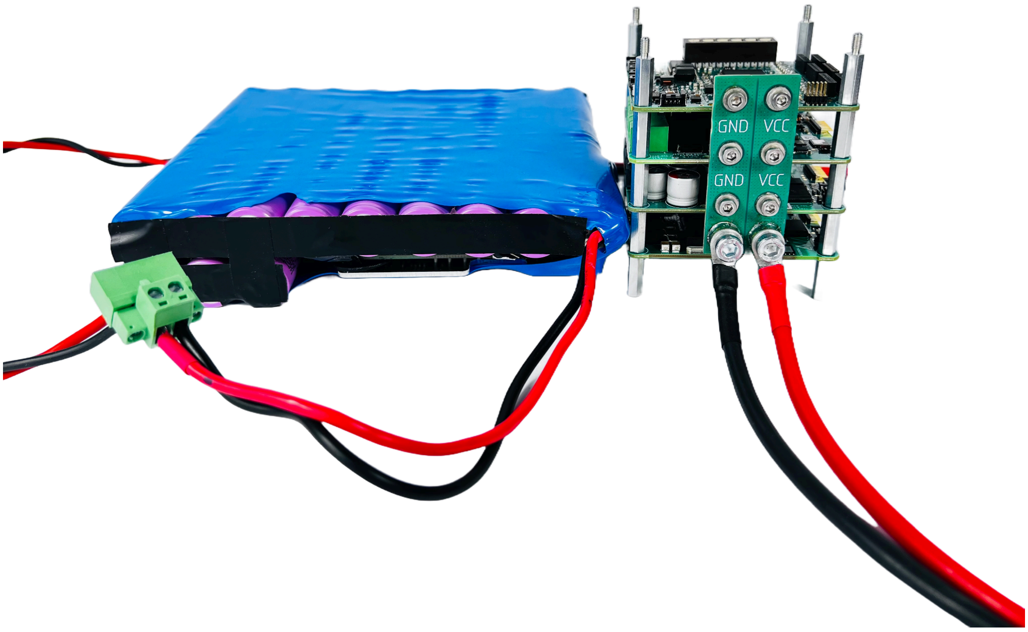
Example connection between PDS_PS and MD80/MD20 motor controllers:





5. Make connections between the power cable with the power supply safely. Please use proper protection on the power line, especially in battery-powered applications. The main power line need to be equipped with overcurrent protection, like fuse, in case of the failures/short circuits between the PDS module and power supply. To the PDS there is included power cable and M5 screws to make connection with the power bus bar. This cable can be used for start-up and test.

It’s maximum current is 16A. For the final integration ensure that the used wire diameter/AWG can handle the current drawn by the load from power source.



6. Push the RGB button for two seconds (until it turns purple) to turn on the system,
7. Run examples to become familiar with PDS functions,
8. Build power management application dedicated to your robot’s needs.