

# DPL Classic & DPL-VU

---

Handbuch v2.0.0

Body-Lights-Platinen fuer Astromech-Droiden

Firmware DPL Firmware\_Nano-v1.1  
Plattform Arduino Nano (ATmega328)

## Danksagung

Die Firmware basiert auf der hervorragenden **AstroCAN BodyLights**-Codebasis von **RealNobser** aus der R2-Builders-Community. Vielen Dank an RealNobser fuer den grossartigen Code, ohne den die DPL Classic und DPL-VU Platinen so nicht moeglich waeren.

Doku v2.0.0 — 2026-04-26

(c) Printed-Droid · alle Rechte vorbehalten / all rights reserved

# Haftungsausschluss

DPL Classic & DPL-VU ist ein DIY-Produkt fuer Astromech-Droiden-Builder. Aufbau, Verkabelung, Konfiguration und Betrieb umfassen Arbeiten, die bei unsachgemaesser Ausfuehrung Sach- oder Personenschaden verursachen koennen.

Mit der Nutzung dieser Dokumentation, Hardware und Firmware erkennst du an:

- Du bist alleine verantwortlich fuer alle Arbeiten an deinem System.
- Du verfuegst ueber Grundkenntnisse in Loeten und Elektronik oder holst dir qualifizierte Hilfe.
- Du pruefst Spannung, Polung und Verkabelung vor jedem Einschalten.
- Du verwendest passende Komponenten (5V-Versorgung, korrekte LED-Strips, etc.).

**Printed-Droid und alle Beteiligten uebernehmen keinerlei Haftung** fuer Schaeden durch falsche Verkabelung, ungeeignete Komponenten, unsachgemaesse Bedienung oder unzureichende Sicherheitsmassnahmen.

Dokumentation und Firmware werden "wie besehen" bereitgestellt. Im Zweifel qualifizierte Hilfe einholen, bevor am System gearbeitet wird.

Diese Dokumentation: **(c) Printed-Droid, alle Rechte vorbehalten.** Keine Genehmigung zur kommerziellen Weiterverwendung, Vervielfaeltigung oder Bearbeitung ohne ausdrueckliche Zustimmung. Firmware (AstroCAN BodyLights) von RealNobser, alle Rechte beim Original-Autor.

# Inhaltsverzeichnis

<b>Haftungsausschluss</b>	<b>2</b>
<b>Inhaltsverzeichnis</b>	<b>3</b>
<b>DPL Classic &amp; DPL-VU — User-Handbuch</b>	<b>5</b>
■ Quickstart in 5 Minuten . . . . .	5
Inhaltsverzeichnis . . . . .	5
1. Was ist DPL Classic / DPL-VU . . . . .	6
Externe VU-Erweiterung am DPL Classic . . . . .	8
Andere Variante: RGB-DPL . . . . .	10
2. Erstinbetriebnahme . . . . .	10
Setup-Schritte . . . . .	10
3. Hardware-Anschlüsse & Pin-Belegung . . . . .	11
3.1 System-Architektur — DPL als zentrale Steuerung . . . . .	11
3.2 Externe LED-Module . . . . .	11
3.3 PCB-Breakouts der DPL-Platinen . . . . .	13
3.4 Vollständige Pin-Belegung Arduino Nano . . . . .	14
3.5 Hinweise zur Eigenentwicklung . . . . .	15
4. Firmware flashen . . . . .	16
5. Befehlssyntax . . . . .	16
So konfigurierst du ein Modul — Schritt für Schritt . . . . .	17
6. Module und Tokens . . . . .	17
Hauptmodule . . . . .	17
DPL-Sub-Bereiche (gehören zur MAX7219-Matrix der DPL) . . . . .	18
7. Animations-Referenz pro Modul . . . . .	18
DP (DPL — Container, koordiniert alle Sub-Bereiche + VU) . . . . .	18
CB (CBI) . . . . .	19
DC (DCBI) . . . . .	19
LD (LDPL) . . . . .	19
UA (UAL) . . . . .	19
VU (VUBar) . . . . .	20
DPL-Sub-Bereiche . . . . .	20
8. Voltage Monitor konfigurieren . . . . .	20
Aktivieren . . . . .	21

Spannungsteiler-Bestückung . . . . .	21
Schwellwerte setzen . . . . .	21
Anzeige . . . . .	21
<b>9. Türen-Sensoren (LeftDoor / RightDoor) . . . . .</b>	<b>22</b>
So aktivierst du die Schalter . . . . .	22
Wenn keine Schalter und keine Brücke angeschlossen sind . . . . .	23
Verkabelungs-Tipp . . . . .	23
<b>10. System-Befehle . . . . .</b>	<b>23</b>
<b>11. Troubleshooting . . . . .</b>	<b>23</b>
Befehl wird nicht akzeptiert (kein "OK" zurück) . . . . .	23
LEDs leuchten nicht . . . . .	24
Falsche Farben (z.B. Rot wird Grün gezeigt) . . . . .	24
VU-Bar reagiert nicht auf Sound . . . . .	24
Voltage-Anzeige zeigt falschen Wert . . . . .	24
EEPROM-Settings nach Firmware-Update verloren . . . . .	24
<b>12. FAQ . . . . .</b>	<b>24</b>
<b>13. Eigene Firmware schreiben . . . . .</b>	<b>26</b>
<b>14. Cheat-Sheet (Schnell-Referenz) . . . . .</b>	<b>26</b>
Modul-Tokens . . . . .	26
Befehlsformat . . . . .	27
Parameter . . . . .	27
System-Befehle . . . . .	27
Häufige Animationen (AN-Nummer) . . . . .	27
Befehlsbeispiele (Copy-Paste) . . . . .	27
Pin-Map (Arduino Nano on-board) . . . . .	28

# DPL Classic & DPL-VU — User-Handbuch

**Firmware DPL Firmware\_Nano-v1.1 · Plattform Arduino Nano (ATmega328) · Doku 2.0.0 · Stand 2026-04-26**

Dieses Handbuch richtet sich an User der Printed-Droid-Platinen DPL Classic und DPL-VU. Es deckt Setup, Hardware-Anschlüsse, Bedienung, alle Befehle, alle Animationen und Troubleshooting ab.

## ■ Quickstart in 5 Minuten

Wenn du die Platine gerade ausgepackt hast und es einfach laufen sehen willst:

1. **5V** an die **5V IN**-Schraubklemme — ein USB-Kabel vom PC reicht aus
2. **USB-Kabel** an den on-board Arduino Nano stecken
3. **Türschalter-Header brücken** — kurze Drahtbrücke zwischen den zwei Pins der Header **Left Door** UND **Right Door** setzen (sonst bleiben DPL und CBI dunkel!)
4. **Firmware flashen** — [DPL Firmware\\_Nano-v1.1.hex](#) mit XLoader laden (siehe *Abschnitt 4*)
5. **Serielles Terminal** öffnen mit **115200 Baud, Zeilenende CR** und z.B. [/DP/BR/12](#) senden → DPL leuchtet

Funktioniert? Super. Für Details, externe Module (CBI/LDPL/UAL), Animationen und Voltage-Monitor weiterlesen.

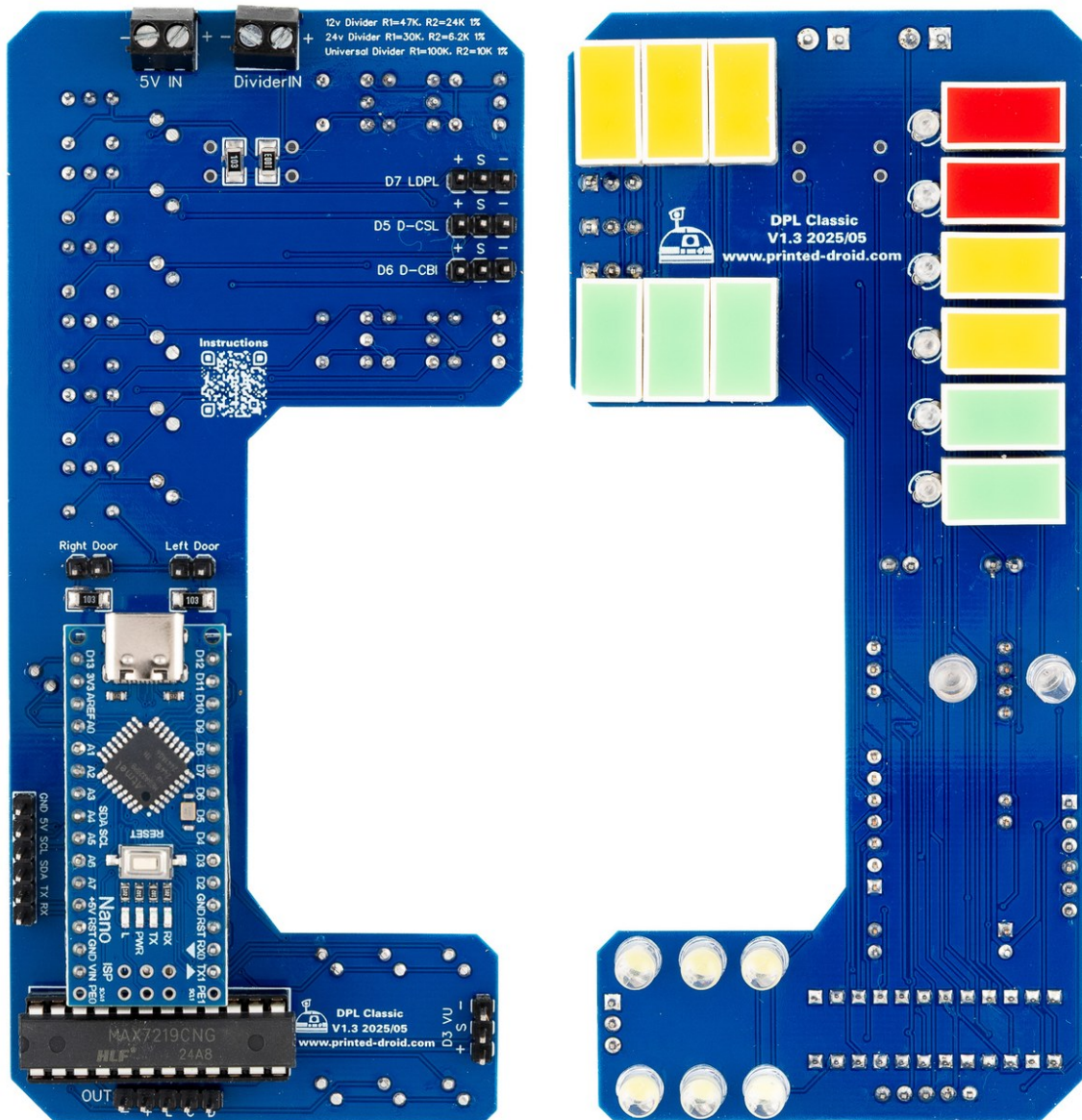
Funktioniert nicht? → *Abschnitt 11 Troubleshooting*, als erstes die Türschalter-Brücken prüfen.

## Inhaltsverzeichnis

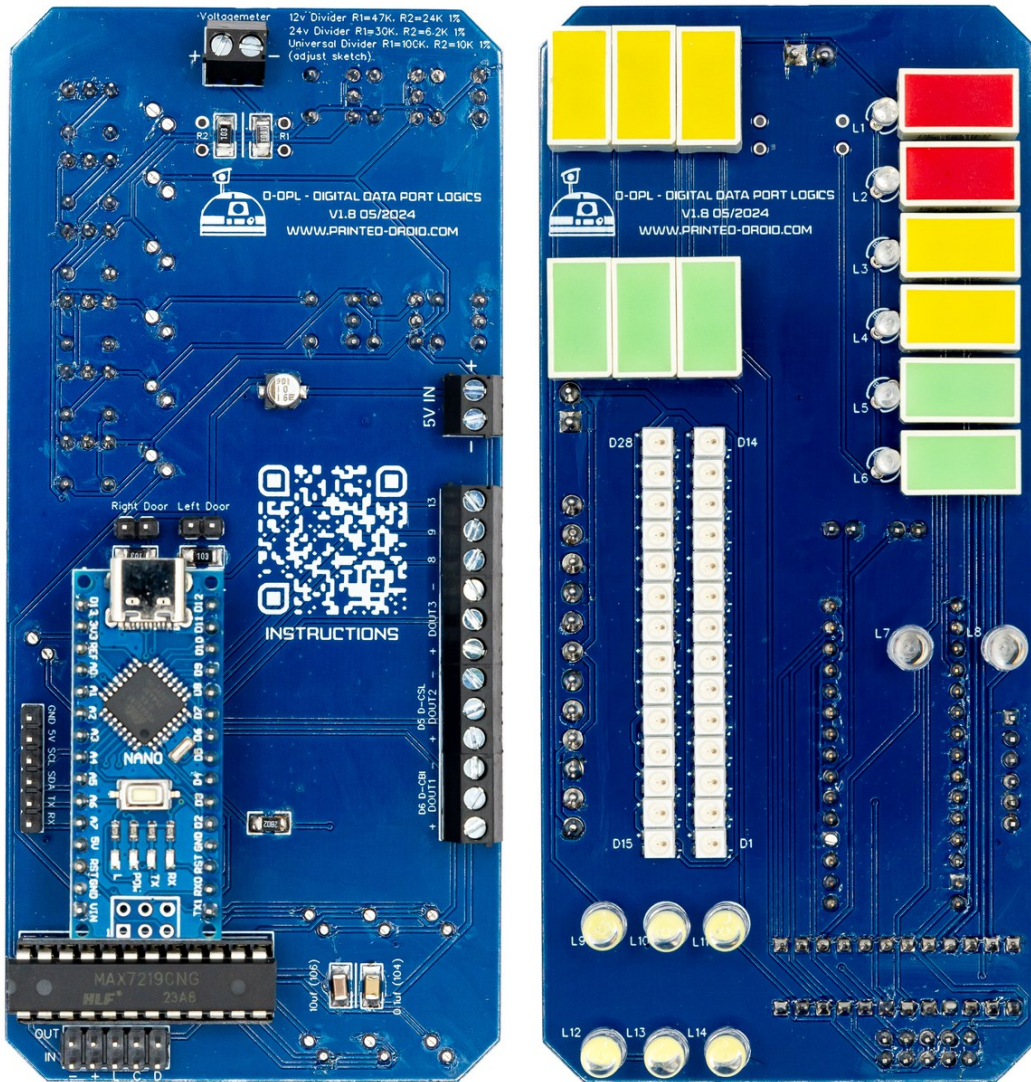
1. *Was ist DPL Classic / DPL-VU*
2. *Erstinbetriebnahme*
3. *Hardware-Anschlüsse & Pin-Belegung*
4. *Firmware flashen*
5. *Befehlssyntax*
6. *Module und Tokens*
7. *Animations-Referenz pro Modul*
8. *Voltage Monitor konfigurieren*
9. *Türen-Sensoren (LeftDoor / RightDoor)*
10. *System-Befehle*
11. *Troubleshooting*
12. *FAQ*
13. *Eigene Firmware schreiben*
14. *Cheat-Sheet (Schnell-Referenz)*

## 1. Was ist DPL Classic / DPL-VU

Beide sind LED-Controller-Platinen für Astromech-Droiden (R2-D2 und ähnliche). Sie ersetzen ältere Einzel-Boards für DPL, CBI, CSL, UAL durch eine kombinierte Platine mit on-board Arduino Nano.



DPL Classic — Rückseite (links) mit Anschlüssen, Vorderseite (rechts) mit den DPL-LEDs.



DPL-VU — Rückseite (links) mit Anschlüssen, Vorderseite (rechts) mit DPL-LEDs plus integrierter VU-Reihe (28 WS2812B im mittleren Bereich, D28-D14 / D15-D1).

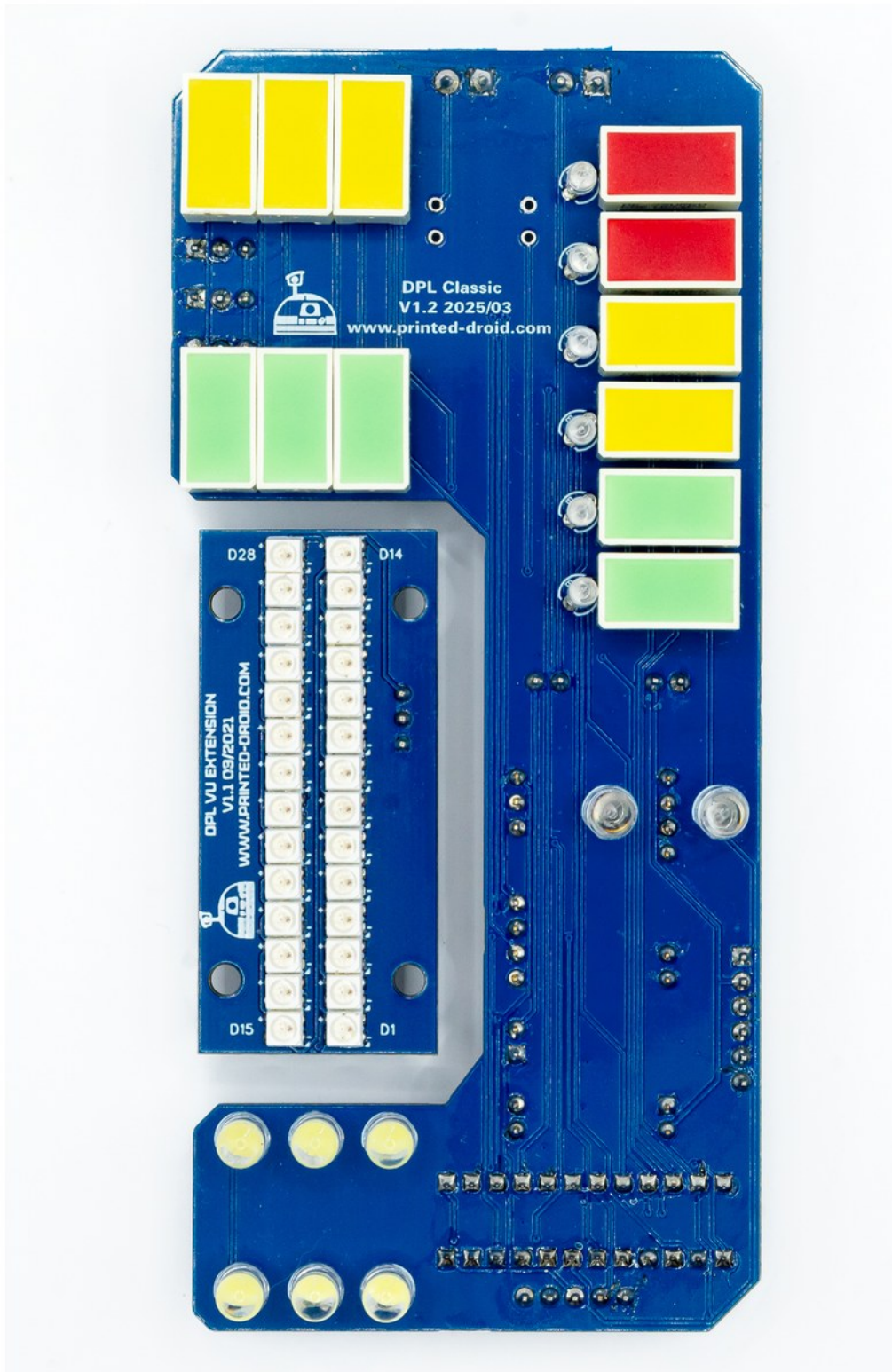
Eigenschaft	DPL Classic	DPL-VU
Microcontroller	Arduino Nano on-board	Arduino Nano on-board
MAX7219 LED-Treiber für DPL-Frontside-LEDs	■	■
Anschluss für externes CBI Plus / D-CBI	■	■
Anschluss für LDPL-Strip	■	■
Anschluss für UAL-Strip	■	■
<b>VU-LEDs auf Platine bestückt</b>	■	■ (28 LEDs in 2x 14er-Reihe)
Externe VU-Erweiterung an D3 anschließbar	■	■ (nicht nötig)
Sound-reaktive Effekte	■ (siehe Hinweis)	■ (siehe Hinweis)
4 Voltage-Schwellen (Akku-Anzeige)	■	■
Türschalter-Eingänge	■	■

Eigenschaft	DPL Classic	DPL-VU
USB-Upload via on-board Nano	■	■

**Wichtiger Hinweis zur "VU"-Anzeige:** Der Name "VU" suggeriert Audio-Reaktivität, aber **die Firmware verwendet keinen Audio-Eingang**. Die Effekte (VU-Meter, BPM-Pulse, Juggle-Sweep) basieren auf einer synthetischen 62-BPM-Sinuswelle. Das sieht aus wie ein VU-Meter, reagiert aber nicht auf tatsächlichen Sound.

### Externe VU-Erweiterung am DPL Classic

Für den DPL Classic gibt es eine separat erhältliche **VU Extension** (Printed-Droid):



DPL Classic mit angesteckter externer VU-Extension (28 WS2812B). Die Extension wird am 3-Pin-Header **D3 VU** der DPL Classic angeschlossen — die Firmware fährt den Pin sowieso, kein Update nötig.

## Andere Variante: RGB-DPL

Wenn du **alle Panels in voller RGB-Farbe** statt der hier dokumentierten MAX7219-Mischung haben willst, gibt es bei Printed-Droid auch die **RGB-DPL-Platine**. Sie nutzt durchgängig WS2811/WS2812-LEDs auf allen Panels (DPL, CBI, LDPL, CSL) und kommt mit einer eigenen Firmware (Profile, Color Schemes, Personality-Modi). Andere Befehlssyntax (`BRIGHTNESS 50` statt `/DP/BR/15`).

Doku & Firmware: <https://github.com/PrintedDroid/RGB-DPL-Firmware>

## 2. Erstinbetriebnahme

Die DPL-Platine ist die **zentrale Steuereinheit**. Alle anderen Module (CBI Plus, D-CBI, LDPL, UAL, optionale VU-Extension) werden an sie angeschlossen — siehe *Abschnitt 3* für Details.

Die DPL hat zwei Türschalter-Eingänge (`Left Door`, `Right Door`). Solange diese **nicht** angeschlossen sind, hält die Firmware **DPL und CBI dunkel** — als ob die Klappen geschlossen wären. **Das ist Absicht**, sorgt aber für 80% aller "geht nicht"-Meldungen. Beim ersten Test daher: entweder Reed-Schalter anschließen, **oder** eine **Drahtbrücke** zwischen den zwei Pins jedes Headers setzen. Details siehe *Abschnitt 9*.

### Setup-Schritte

#### 1. Stromversorgung

**5V** an die `5V IN`-Schraubklemme (z.B. vom Bordnetz-5V-Schiene des R2). Die Platine wird **ausschließlich mit 5V betrieben** — kein 12V auf diesem Eingang!

2. Optional **Bordnetz-Spannung** (12V / 24V Akku-Spannung) an die `Voltage IN`-Schraubklemme. **Das ist NUR ein Mess-Eingang** für die Akku-Anzeige, keine zusätzliche Versorgung.

3. **Strombedarf**: Bei den Default-Helligkeiten reicht ein **handelsübliches USB-Kabel direkt am PC** als alleinige Versorgung — auch dann, wenn DPL + analoges CBI (CBI Plus) + digitales D-CBI gleichzeitig angeschlossen sind. Erst bei sehr hellen Setups (alle WS2812B-Strips auf voller Helligkeit) wird ein separates 5V-Netzteil mit ein paar Ampere sinnvoll.

4. **LED-Module anschließen** (alle optional, je nach Aufbau)

**CBI Plus** (analoges CBI mit eigenem MAX7219) → 5-Pin-Header `-/+L/C/D` an der DPL

5. **D-CBI** (digitales CBI mit WS2812B) → 3-Pin-Header `D6 D-CBI` (5V/Signal/GND)

6. **LDPL-Strip** → 3-Pin-Header `D7 LDPL` (5V/Signal/GND)

7. **UAL-Strip** → Signal-only-Pin `D8` (5V/GND von einem freien Power-Pin)

8. Bei DPL Classic mit nachgerüsteter externer **VU Extension** → 3-Pin-Header `D3 VU`

9. **Türschalter** (siehe Warning oben — DPL und CBI bleiben sonst dunkel!)

Reed-Schalter / Mikroschalter / Hall-Sensor an die 2-Pin-Header `Right Door` und `Left Door`

10. Oder zum Testen: Drahtbrücke zwischen den zwei Pins jedes Headers

11. Details und alle Varianten in *Abschnitt 9*

12. **USB-Kabel an den on-board Arduino Nano stecken** (zum Konfigurieren über Serial-Terminal)

#### 13. Erste Funktionsprüfung

Beim ersten Start spielt eine Boot-Sequenz (Startup-Animation der DPL) ab

14. Anschließend laufen alle Module mit den Default-Animationen

15. Wenn nichts leuchtet: Türschalter prüfen oder `/RESTART` senden

## 3. Hardware-Anschlüsse & Pin-Belegung

Dieser Abschnitt zeigt erst die System-Architektur (welche Module gibt es, wie verbinden sie sich), dann die konkreten Anschlüsse auf der DPL-Platine und schließlich die vollständige Nano-Pin-Belegung für Anwender mit eigenem Code.

### 3.1 System-Architektur — DPL als zentrale Steuerung

Die DPL-Platine ist die **Master-Platine** mit on-board Arduino Nano. Alle weiteren LED-Module sind separate Platinen, die an die DPL angeschlossen werden:

Externes Modul	Anschluss an der DPL	Befehls-Token
<b>D-CBI</b> (digitales Charge Bay Indicator, WS2812B)	3-Pin-Header <b>D6</b> <b>D-CBI</b> (5V / Signal / GND)	<code>/DC/...</code>
<b>CBI Plus</b> (analoges Charge Bay Indicator, MAX7219)	5-Pin-Header <code>-/+/L/C/D</code>	<code>/CB/...</code>
<b>LDPL</b> (Large DataPanel Logics)	3-Pin-Header <b>D7</b> <b>LDPL</b> (5V / Signal / GND)	<code>/LD/...</code>
<b>UAL</b> (Utility Arm Lights)	Signal-only-Pin <b>D8</b> (5V/GND separat)	<code>/UA/...</code>
<b>VU Extension</b> (nur DPL Classic)	3-Pin-Header <b>D3</b> <b>VU</b> (5V / Signal / GND)	<code>/VU/...</code>

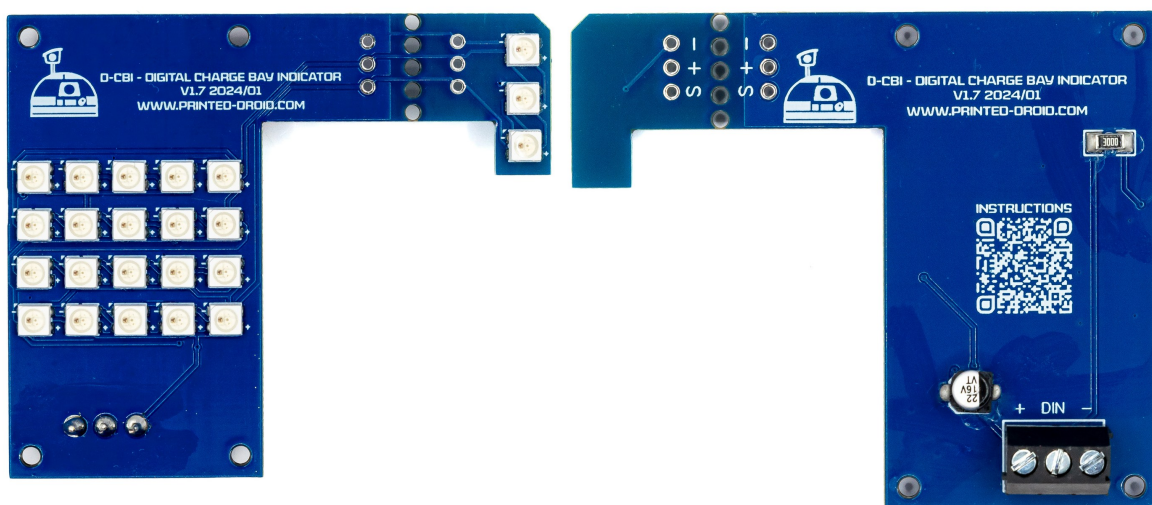
#### Was bedeutet das für die Nutzung:

- Du musst nicht alle Module gleichzeitig haben — die DPL läuft auch standalone
- Wenn ein externes Modul nicht angeschlossen ist, läuft die zugehörige Animation einfach ins Leere — kein Fehlverhalten, einfach nichts sichtbar
- CBI Plus und D-CBI können auch **gleichzeitig** angeschlossen sein — sie hängen an unterschiedlichen Anschlüssen und werden über getrennte Token (`/CB/...` für CBI Plus, `/DC/...` für D-CBI) konfiguriert

### 3.2 Externe LED-Module

Die folgenden Module werden separat von Printed-Droid angeboten und an die DPL angeschlossen:

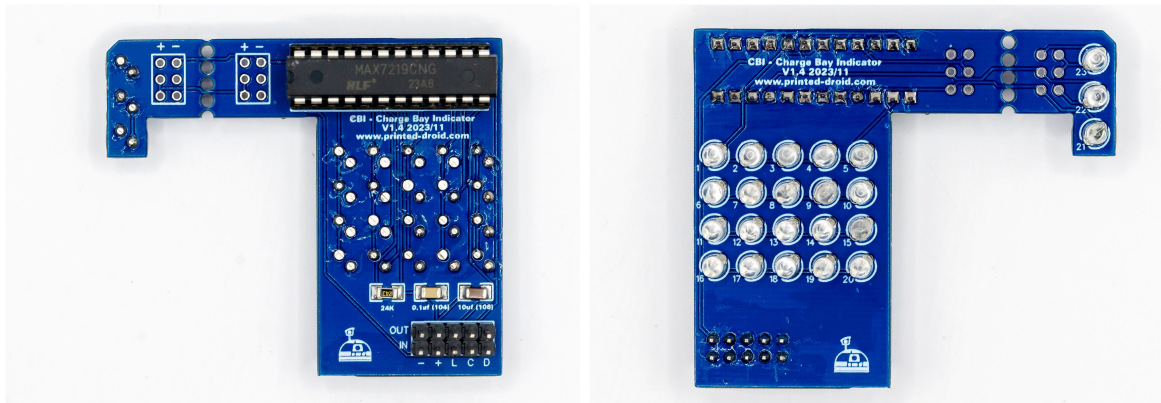
#### D-CBI (digitales Charge Bay Indicator)



D-CBI — Rückseite (links) mit DIN-Schraubklemme (+ **DIN** -) für die Hauptverbindung zur DPL. Vorderseite (rechts) mit 23 WS2812B-LEDs (20 Hauptbereich + 3 Status-LEDs).

- **Volle RGB-Farbtiefe:** WS2812B-LEDs erlauben **jede beliebige Farbe** (16,7 Mio. Farben pro LED) — Grundfarbe pro `/DC/CO/R/G/B` frei wählbar, Animationen wie Rainbow, Color-Fade, Heart-Beat etc. nutzen die volle Farb-Vielfalt
- Anschluss an die DPL über das 3-Pin-Terminal `D6 D-CBI` (5V / Signal / GND), oder über die DIN-Schraubklemme auf der D-CBI-Platine
- Befehle: `/DC/...` (siehe *Abschnitt 6*)

### CBI Plus (analoges Charge Bay Indicator)



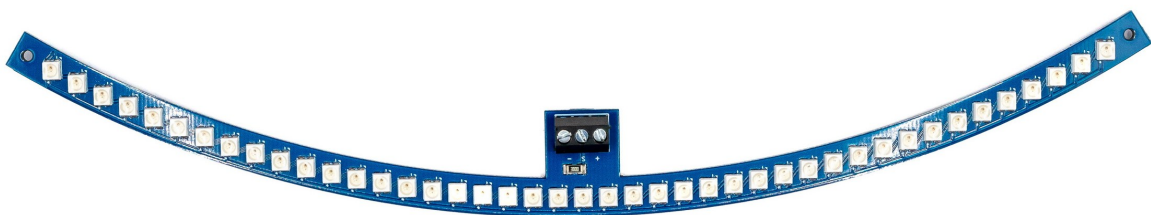
CBI Plus — Rückseite (links) mit eigenem MAX7219CNG + 5-Pin `IN/OUT`-Header für Daisy-Chain. Vorderseite (rechts) mit den klassischen 3-mm-LED-Indikatoren.

- **Feste Einzelfarben:** Die Vorderseite ist mit klassischen analogen 3-mm-LEDs bestückt — jede LED hat ihre Farbe ab Werk fest verbaut (rot/gelb/grün/blau in den jeweiligen Bereichen). Es gibt keine RGB-Steuerung, der `/CB/CO/...`-Befehl wirkt sich nicht auf einzelne LEDs aus
- Anschluss an die DPL über den 5-Pin-Header `-/+/L/C/D`
- Befehle: `/CB/...`

**Welches CBI-Modul?** Wer das klassische R2-Look mit fixen LED-Farben will, nimmt das **CBI Plus**. Wer mehr Spielraum für Farben und Animationen möchte, nimmt das **D-CBI**. Beide können auch parallel betrieben werden — sie hängen an unterschiedlichen Pins und werden separat konfiguriert.

**Trennbare PCB für sehr alte R2-Bodies:** Sowohl D-CBI als auch CBI Plus haben einen Bereich mit Lochreihen ("Bruchkante"), an dem die Platine bei Bedarf in zwei Hälften getrennt werden kann (für sehr alte R2-Body-Generationen mit anderem Layout). Die getrennten Hälften werden anschließend über die zwei nebeneinander liegenden 3-Pin-Header (oben auf der Rückseite, jeweils `+ S -` beschriftet) mit kurzen Drähten oder Pin-Brücken wieder elektrisch verbunden. Wer die Standard-PCB-Form behält, lässt diesen Bereich unverändert.

### LDPL (Large DataPanel Logics)



LDPL — gebogener WS2812B-Strip mit 43 LEDs für die Bauchklappen-Beleuchtung. Mittig 3-Pin-Schraubklemme (`- S +`) zum Anschluss an `D7` der DPL.

- Anschluss über das 3-Pin-Terminal `D7 LDPL` an der DPL

- Volle RGB-Farbtiefe per [/LD/CO/...](#)
- Befehle: [/LD/...](#)

### UAL (Utility Arm Lights)

- WS2812B-Strip mit 19 LEDs für die Werkzeug-Arme
- Anschluss über den **Signal-only-Pin D8** an der DPL — 5V/GND aus einem freien Power-Pin (z.B. I2C-Stiftleiste oder MAX7219-Header) holen
- Befehle: [/UA/...](#)

### VU Extension (nur DPL Classic)

Die DPL-VU hat die VU-LEDs bereits on-board. Die DPL Classic kann optional auf VU aufgerüstet werden — siehe Bild in *Abschnitt 1* (DPL Classic mit angesteckter VU Extension). Anschluss an den 3-Pin-Header [D3 VU](#) an der DPL Classic.

## 3.3 PCB-Breakouts der DPL-Platinen

Beide DPL-Varianten (Classic + VU) führen die gleichen Funktionen heraus, aber in unterschiedlicher Form: bei DPL-VU sind die LED-Anschlüsse als **Schraubklemmen** ausgeführt, bei DPL Classic ist nur Power als Schraubklemme — alle anderen Anschlüsse sind **Stiftleisten**.

Funktion	Nano-Pin(s)	DPL-VU	DPL Classic
Power In: <b>5V</b> + GND	—	<b>Schraubklemme</b>	<b>Schraubklemme</b>
<b>Voltage In</b> (Bordnetz für Akku-Messung, KEINE Stromversorgung!)	A0 via Spannungsteiler	<b>Schraubklemme</b>	<b>Schraubklemme</b>
3-Pin LED-Out <b>DCBI</b> (5V / Signal / GND)	D6	Schraubklemme	Stiftleiste
3-Pin LED-Out (D5, reserviert) (5V / Signal / GND)	D5	Schraubklemme	Stiftleiste
3-Pin LED-Out <b>LDPL</b> (5V / Signal / GND)	D7	Schraubklemme	Stiftleiste
Signal-only <b>UAL</b>	D8	Stiftleiste	Stiftleiste
Signal-only <b>frei #1</b>	D9	Stiftleiste	Stiftleiste
Signal-only <b>frei #2</b>	D13	Stiftleiste	Stiftleiste
<b>Türschalter Left</b> (D2 + GND)	D2	2-Pin-Header	2-Pin-Header
<b>Türschalter Right</b> (D4 + GND)	D4	2-Pin-Header	2-Pin-Header
<b>I2C-Breakout</b> (5V / SDA / SCL / GND)	A4 (SDA), A5 (SCL)	Stiftleiste	Stiftleiste
<b>UART-Breakout</b> (5V / RX / TX / GND)	D0 (RX), D1 (TX)	Stiftleiste	Stiftleiste
<b>MAX7219 / CBI-Plus-Daisy-Chain</b> : -, +, L, C, D	GND, 5V, D12, D11, D10	5-Pin Stiftleiste	5-Pin Stiftleiste

**Pin-Map des MAX7219-/CBI-Headers** (5-Pin-Reihe für die Daisy-Chain zu CBI Plus):

Header-Label	Bedeutung	Nano-Pin
–	GND	GND
+	5V	5V
L	LOAD (CS)	D12
C	CLOCK	D11
D	DATA	D10

### 3.4 Vollständige Pin-Belegung Arduino Nano

Diese Tabelle ist hilfreich für Anwender, die **eigene Firmware** auf die Platine schreiben wollen. Sie zeigt jeden Pin des Arduino Nano, was darauf bestückt ist und welche Pins bei der Original-Firmware ungenutzt sind.

Nano-Pin	Default-Funktion (Arduino)	DPL-Firmware-Belegung	Bauteil auf Platine	Frei für eigenen Code?
D0	UART RX	—	RX/TX-Stiftleiste	■ teilt sich mit USB-Serial
D1	UART TX	—	RX/TX-Stiftleiste	■ teilt sich mit USB-Serial
D2	INT0	<a href="#">SWITCH_LEFTDOOR</a> (INPUT_PULLUP)	2-Pin Türschalter-Header (links / Dataport-Tür)	nur wenn Doors-Logik nicht gebraucht
D3	INT1, PWM	<a href="#">VU_DATA</a> (WS2812B Out)	VU-LED-Reihe (DPL-VU on-board) bzw. 3-Pin-Header <a href="#">D3 VU</a> für externe VU Extension (DPL Classic)	nur wenn keine VU-LEDs / externe VU
D4	—	<a href="#">SWITCH_RIGHTDOOR</a> (INPUT_PULLUP)	2-Pin Türschalter-Header (rechts / Charge-Bay-Tür)	nur wenn Doors-Logik nicht gebraucht
D5	PWM	reservierter WS2812B-Out	3-Pin-Anschluss (5V / Signal / GND)	■ frei für eigenen Code
D6	PWM	<a href="#">DCBI_DATA</a> (WS2812B Out)	3-Pin-Anschluss <a href="#">D6 D-CBI</a> (5V / Signal / GND)	nur wenn kein DCBI-Strip
D7	—	<a href="#">LDPL_DATA</a> (WS2812B Out)	3-Pin-Anschluss <a href="#">D7 LDPL</a> (5V / Signal / GND)	nur wenn kein LDPL-Strip
D8	—	<a href="#">UAL_DATA</a> (WS2812B Out)	Signal-only Pin (Power separat)	nur wenn kein UAL-Strip
D9	PWM	— (ungenutzt)	Signal-only Pin	■ frei
D10	SPI SS, PWM	<a href="#">MAX7219_DATA</a> (Bit-Bang)	MAX7219-Header-Pin <b>D</b> (zur CBI-Plus-Daisy-Chain)	nur wenn keine MAX7219-Anzeigen (DPL-LEDs / CBI Plus)
D11	SPI MOSI, PWM	<a href="#">MAX7219_CLOCK</a> (Bit-Bang)	MAX7219-Header-Pin <b>C</b>	nur wenn keine MAX7219-Anzeigen

Nano-Pin	Default-Funktion (Arduino)	DPL-Firmware-Belegung	Bauteil auf Platine	Frei für eigenen Code?
D12	SPI MISO	MAX7219_LOAD (Bit-Bang CS)	MAX7219-Header-Pin L	nur wenn keine MAX7219-Anzeigen
D13	SPI SCK, LED_BUILTIN	— (ungenutzt)	Signal-only Pin	■ frei (Vorsicht: on-board LED am Pin)
A0	ADC	ANALOGPIN (Voltage-Divider)	An Spannungsteiler 100kΩ/10kΩ (SMD; THT-Pads daneben für eigene Werte) → Voltage-In-Klemme	nur wenn keine Spannungsmessung
A1	ADC	—	—	■ frei
A2	ADC	—	—	■ frei
A3	ADC	—	—	■ frei
A4	ADC, I2C SDA	—	I2C-Stiftleiste	■ frei (für I2C-Erweiterungen vorgesehen)
A5	ADC, I2C SCL	—	I2C-Stiftleiste	■ frei (für I2C-Erweiterungen vorgesehen)
A6	ADC only	—	—	■ frei (Achtung: nur analog, kein digital I/O)
A7	ADC only	—	—	■ frei (Achtung: nur analog, kein digital I/O)
RST	Reset	—	Reset-Pad / Auto-Reset über USB	(Hardware-Reset)
5V	Power Bus	—	5V-Schiene (gespeist aus 5V IN-Klemme)	—
3V3	Power Out (50 mA max)	—	nur am Nano-Pin abgreifbar	—
GND	Ground	—	GND-Schiene	—
VIN	Power In (raw, Nano-Pin)	—	<b>nicht für Bordnetz nutzen</b> — die "Voltage-In"-Klemme der DPL ist GETRENNT, nur ADC-Mess-Eingang	—
AREF	ADC-Referenz	— (Default = AVcc)	—	ändern bei eigener ADC-Skalierung

### 3.5 Hinweise zur Eigenentwicklung

- **MAX7219 belegt D10/D11/D12 als Bit-Bang-SPI** (LedControl-Library, kein Hardware-SPI). Damit ist Hardware-SPI nicht ohne Konflikte nutzbar. Zwei Optionen für eigene SPI-Geräte: LedControl entfernen und Hardware-SPI auf D10 (SS) / D11 (MOSI) / D13 (SCK) nutzen — D12 wäre dann MISO.
- Software-SPI / I2C / UART für Zusatzgeräte verwenden.

**D13 hat die on-board User-LED** des Nano. Wenn du D13 als Eingang oder als Output gegen das LED-Verhalten verwendest, blinkt die LED u.U. mit oder die LED wirkt als  $\sim 1\text{ k}\Omega$ -Pull-Down. Für Sensor-Eingänge eher A1-A3 nehmen.

**A6/A7 sind nur ADC** — sie können nicht als digital I/O verwendet werden (Hardware-Limitation des ATmega328P im TQFP-Package).

**D0/D1 (RX/TX)** sind für die serielle Konsole reserviert. Wenn du eigene Firmware schreibst und keine Serial-Konsole brauchst, sind sie als digitale I/O-Pins frei. **Wichtig:** das USB-Upload-Protokoll nutzt RX/TX — nach dem Flashen hast du keinen Konflikt mehr, aber während des Uploads dürfen die Pins nicht extern getrieben werden.

**GND-Verteilung:** Alle GND-Pins (Schraubklemme, Header, Türschalter, MAX7219-Header, I2C, UART) sind miteinander verbunden — eigene Geräte können einen beliebigen GND-Pin nutzen.

## 4. Firmware flashen

Beide Platinen verwenden dieselbe Firmware. Die aktuelle Datei heißt: [DPL Firmware\\_Nano-v1.1.hex](#)

### Empfohlene Tools:

- **XLoader** (Windows) — von Printed-Droid empfohlen  
(<https://www.printed-droid.com/kb/dpl-classic-data-port-logics/> → Downloads)
- **avrdude** (Linux/macOS) — Kommandozeile

### XLoader-Konfiguration:

- Hex-File: [DPL Firmware\\_Nano-v1.1.hex](#)
- Device: **Nano (ATmega328)**
- COM-Port: der Port des angeschlossenen Nano (Geräte-Manager prüfen)
- Baud rate: **115200** (manche ältere Nanos brauchen 57600)
- Upload klicken — der Nano blinkt während des Uploads, fertig in ca. 5-10 Sekunden

### Nach dem Flash:

- Die EEPROM-Settings bleiben in der Regel erhalten
- Wenn ein Firmware-Update das interne Speicher-Layout ändert, werden die User-Settings (Helligkeit, Farben, Animationen) automatisch auf Defaults zurückgesetzt und müssen neu gesetzt werden — das wird im Changelog der jeweiligen Version vermerkt

## 5. Befehlssyntax

Befehle werden über die **serielle Schnittstelle** des Arduino Nano gesendet:

- **Baud-Rate:** 115200
- **Zeilenende:** **CR (\r)** — manche Terminals senden nur LF (\n), das wird ignoriert. CRLF funktioniert (LF wird ignoriert, CR triggert die Auswertung).
- **Maximale Befehlslänge:** 32 Zeichen inklusive Slashes

### Format:

```
/MODUL/PARAMETER/WERT[ /WERT2/WERT3... ]
```

Antwortet die Platine mit `OK`, wurde der Befehl akzeptiert. Keine Antwort heißt: Befehl ist falsch geformt oder das Modul existiert in der aktuellen Firmware nicht.

### Beispiele:

Befehl	Wirkung
<code>/DP/BR/15</code>	DPL Helligkeit auf 15 (Skala 0–15 für MAX7219)
<code>/LD/BR/100</code>	LDPL Helligkeit auf 100 (Skala 0–255 für WS2812B)
<code>/LD/CO/255/0/0</code>	LDPL Grundfarbe Rot
<code>/CB/AN/3</code>	CBI Heart-Animation
<code>/DC/AN/6</code>	DCBI Rainbow-Animation
<code>/DP/VL/11.5/12.0/12.5/13.0</code>	Voltage-Schwellen rot/gelb/grün/laden
<code>/DP/RR/100000.0/10000.0</code>	R1=100kΩ, R2=10kΩ (Default)
<code>/DP/VM/1</code>	Voltage-Monitor aktivieren
<code>/RESTART</code>	Soft-Reboot
<code>/FACTORY</code>	EEPROM auf Defaults zurücksetzen + Reboot

## So konfigurierst du ein Modul — Schritt für Schritt

Beispiel: **LDPL** soll dauerhaft in einem leuchtenden Rot mit mittlerer Helligkeit laufen.

### 1. Helligkeit setzen:

→ Antwort: `OK`. Helligkeit ist 0-255 für WS2812B-Module.

### 2. Grundfarbe setzen (R, G, B je 0-255):

→ reines Rot. Beispiele: `/LD/CO/255/255/0` = Gelb, `/LD/CO/0/0/255` = Blau.

### 3. Animation wählen:

→ `fadeOutAnimation` (Knight-Rider mit Fade). Andere Optionen siehe *Abschnitt 7*.

4. **Fertig.** Settings sind automatisch im EEPROM gespeichert und überleben einen Reboot. Beim nächsten Einschalten startet LDPL direkt mit diesen Werten.

Das Schema gilt für **alle Module**: `BR` für Helligkeit, `CO` für Farbe (nur WS2812B-Module — D-CBI, LDPL, UAL, VU), `AN` für Animation.

**Tip:** Wenn du was kaputt-konfiguriert hast und nicht mehr weißt was los ist, einfach `/FACTORY` senden. Das setzt alles auf Default-Werte zurück und du fängst neu an.

## 6. Module und Tokens

Die Firmware kennt diese Module:

### Hauptmodule

Token	Modul	Wo physisch?	Hardware-Treiber	Helligkeits-Skala
<code>DP</code>	Data Port Lights (DPL) — Container für alle Sub-Bereiche + VU	<b>Auf der DPL-Platine selbst</b>	on-board MAX7219 ( <code>dev0</code> )	0–15

Token	Modul	Wo physisch?	Hardware-Treiber	Helligkeits-Skala
CB	Charge Bay Indicator (CBI)	Externes <b>CBI-Plus</b> -Modul (an MAX7219-Header)	CBI-Plus-eigener MAX7219 ( <code>dev1</code> ) per Daisy-Chain	0–15
DC	Digital Charge Bay Indicator (D-CBI)	Externes <b>D-CBI</b> -Modul (an D6)	WS2812B 23 LEDs	0–255
LD	Large DataPanel Logics (LDPL)	Externer <b>LDPL</b> -Strip (an D7)	WS2812B 43 LEDs	0–255
UA	Utility Arm Lights (UAL)	Externer <b>UAL</b> -Strip (an D8)	WS2812B 19 LEDs	0–255
VU	VU-Bar	DPL-VU on-board / bei Classic optional <b>VU Extension</b> an D3	WS2812B 28 LEDs	0–255

**Was passiert bei nicht angeschlossenen Modulen?** Die Firmware fährt alle Pins, egal ob ein Modul angeschlossen ist oder nicht. Fehlt z.B. CBI Plus, läuft die `/CB/...`-Animation nur "im Code" — keine Fehlermeldung, einfach nichts sichtbar. Das ist OK und kein Anlass zur Sorge.

### DPL-Sub-Bereiche (gehören zur MAX7219-Matrix der DPL)

Diese Tokens steuern einzelne Reihen/Spalten der DPL-Matrix isoliert. Normalerweise ist das nicht nötig, weil `/DP/AN/...` alle Sub-Bereiche koordiniert — aber für Spezialeffekte hilfreich:

Token	Bereich	Position auf der DPL-Matrix
TB	Top Bar	Reihe 4+5
BL	Blue Lights	Reihe 0
BG	Bar Graph	Reihe 2+3 (linker Block)
RL	Red Lights	Reihe 2+3 Spalte 6
WL	White Lights	Reihe 1

## 7. Animations-Referenz pro Modul

Jedes Modul hat eine eigene Liste an Animationen. Setzen mit `/MODUL/AN/NUMMER`.

### DP (DPL — Container, koordiniert alle Sub-Bereiche + VU)

AN	Name	Beschreibung
0	none	Alles aus
1	defaultAnimation	Alle Sub-Bereiche im Default-Modus, VU im VU-Effekt-Wechsel
2	randomVUAnimation	Sub-Bereiche im Default, VU im Random-Bargraph-Modus
3	startupAnimation	Boot-Sequenz (Default beim ersten Start nach <code>/FACTORY</code> )

## CB (CBI)

AN	Name	Beschreibung
0	none	Aus
1	defaultAnimation	Standard-Animations-Sequenz
2	randomAnimation	Zufalls-LED-Muster
3	heartAnimation	Herz-Symbol-Sequenz
4	chargingAnimation	Lade-Symbol-Sequenz (wird automatisch aktiviert, wenn die gemessene Spannung den <code>charge</code> -Schwellwert überschreitet)

## DC (DCBI)

AN	Name	Beschreibung
0	none	Aus
1	defaultAnimation	Standard-Sequenz in der eingestellten Grundfarbe ( <code>/DC/CO/...</code> )
2	randomAnimation	Zufalls-LED-Muster
3	heartAnimation	Herz-Symbol
4	randomFadeAnimation	Zufallspunkte mit Fade-Out
5	colorFadeAnimation	Zufalls-LEDs in Regenbogen-Farben mit Fade-Out
6	rainbowAnimation	Klassischer Regenbogen-Verlauf
7	heartBeatAnimation	Pulsender Herzschlag
8	chargingAnimation	Lade-Symbol

## LD (LDPL)

AN	Name	Beschreibung
0	none	Aus
1	defaultAnimation	Knight-Rider-Lauflicht (hart, ohne Fade)
2	fadeOutAnimation	Knight-Rider mit Fade-Out (Default)
3	rainbowAnimation	Lauflicht in Regenbogen-Farben

## UA (UAL)

AN	Name	Beschreibung
0	none	Aus
1	defaultAnimation	Knight-Rider mit Fade in der eingestellten Grundfarbe ( <code>/UA/CO/...</code> , Default)

AN	Name	Beschreibung
2	movieAnimation	Filmsequenz: 18 LEDs in der Grundfarbe, dann ein grüner LED am Ende. Stoppt automatisch nach Position 26.
3	abortAnimation	Roter LED bei Position 18 als Abbruch-Indikator

Hinweis: Auf den Printed-Droid-Platinen ist kein dedizierter UAL-Schalter-Pin vorgesehen. Die Movie-Animation wird daher **nur per Serial-Befehl** ausgelöst (`/UA/AN/2`), nicht automatisch durch einen Hardware-Trigger.

## VU (VUBar)

AN	Name	Beschreibung
0	none	Aus
1	defaultAnimation	Wechselt automatisch alle 5 Sekunden zwischen drei Effekten: VU-Meter, BPM-Pulse, Juggle-Sweep
2	randomVUAnimation	Klassischer VU-Bargraph mit zufälligem Drift, grün/gelb/rot je nach Pegel

## DPL-Sub-Bereiche

Alle Sub-Bereiche haben mindestens `0` (none) und `1` (defaultAnimation):

Token	AN 0	AN 1	AN 2	AN 3
<code>TB</code>	none	defaultAnimation	—	—
<code>BL</code>	none	defaultAnimation	staticTimeAnimation	bargraphAnimation
<code>BG</code>	none	defaultAnimation	—	—
<code>RL</code>	none	defaultAnimation	—	—
<code>WL</code>	none	defaultAnimation	—	—

Hinweis: Wenn `/DP/AN/1` gesetzt wird, überschreibt der DP-Container die Sub-Animationen. Direkter Eingriff (`/BL/AN/2` etc.) wird beim nächsten DP-Animation-Wechsel überschrieben.

## 8. Voltage Monitor konfigurieren

Die DPL-Platine kann eine externe Bordnetz-Spannung messen und den Lade-Zustand anzeigen.

**Wichtig:** Der Mess-Eingang `Voltage IN` ist **getrennt** von der `5V IN`-Versorgung — die Platine wird mit 5 V betrieben, der Voltage-IN-Eingang ist ausschließlich ein hochohmiger Mess-Anschluss (über Spannungsteiler an A0).

Die Status-Anzeige läuft über das CBI Plus (drei MAX7219-LEDs in Spalte 5) und/oder das D-CBI (drei dedizierte WS2812B-LEDs).

## Aktivieren

```
/CB/VM/1 Voltage-Monitor an
/CB/VM/0 Voltage-Monitor aus
```

**VM** wirkt global auf alle Module mit VCC-Funktion (CBI, DCBI). Das Token kann auch `/DP/VM/1` etc. sein — der Effekt ist derselbe.

## Spannungsteiler-Bestückung

Auf den Printed-Droid-Platinen (DPL Classic + DPL-VU) ist der Spannungsteiler **ab Werk mit R1 = 100 kΩ und R2 = 10 kΩ als SMD bestückt** und passt für 0–30 V Bordnetz-Messung. Diese Default-Werte sind in der Firmware bereits hinterlegt — du musst sie normalerweise **nicht** ändern.

**Falls andere Werte gebraucht werden** (z.B. um bei höheren Spannungen die Messauflösung zu verbessern oder bei sehr niedrigen Spannungen besser aufzulösen):

1. **SMD-Widerstände entfernen** — die zwei SMD-Bauteile (R1 und R2) auf der Platine vorsichtig auslöten oder mit einem feinen LötKolben gegeneinander hochkippen
2. **Eigene THT-Widerstände einlöten** — direkt **neben den SMD-Pads** sind Durchsteckmontage-Pads (THT) für Standard-Bedraht-Widerstände vorgesehen. Dort beliebige Werte einlöten
3. **Software-Skalierung anpassen** — die neuen Werte mit `/DP/RR/<R1_in_Ohm>/<R2_in_Ohm>` (oder `/CB/RR/...` etc.) der Firmware mitteilen, sonst stimmt die angezeigte Spannung nicht

```
/DP/RR/100000.0/10000.0 R1=100k, R2=10k (Default – bei SMD-Bestückung)
/DP/RR/220000.0/22000.0 Beispiel: für höheres Spannungs-Range
```

**Wichtig:** Der `/RR`-Befehl ändert nur die Software-Skalierung (was die Firmware aus dem ADC-Wert errechnet), **nicht** die Hardware. Wenn die SMD-Widerstände drauf sind und du `/RR` auf andere Werte setzt, zeigt die Firmware Mist an. Im Zweifel nach versehentlichem Verstellen `/FACTORY` ausführen — dann sind die Defaults wieder gesetzt.

**Auslegung:** Egal welcher Bordnetz-Wert: nach dem Spannungsteiler darf am ADC-Pin (A0) maximal 5V anliegen. Formel:  $V_{ADC} = V_{Bordnetz} \times R2 / (R1+R2)$ . Mit Default-Werten:  $12\text{ V} \times 10/(100+10) = 1.09\text{ V} \checkmark$ ;  $24\text{ V} \times 10/(100+10) = 2.18\text{ V} \checkmark$ . Sicherheitsabstand zu 5V immer einplanen.

## Schwellwerte setzen

```
/DP/VL/11.5/12.0/12.5/13.0 red / yellow / green / charge
```

Schwellwert	Bedeutung
red	Unterhalb dieser Spannung: rote LED leuchtet (kritisch niedrig)
yellow	Zwischen red und yellow: gelb
green	Zwischen yellow und green: grün (normal)
charge	Oberhalb dieser Spannung: Charging-Animation aktiv (Ladegerät erkannt)

Die Default-Werte (11.5/12.0/12.5/13.0) sind für 12V-Blei-Akku ausgelegt. Bei 3S-Lipo (~10.5–12.6V Betriebsspannung) sollten die Schwellwerte tiefer gesetzt werden, z.B. `/DP/VL/10.8/11.4/12.0/12.6`.

## Anzeige

- **Auf CBI Plus (wenn angeschlossen):** drei dedizierte LEDs in Spalte 5 der MAX7219-Matrix (rot/gelb/grün)

- **Auf D-CBI (wenn angeschlossen):** drei dedizierte WS2812B-LEDs (Index 20=grün, 21=gelb, 22=rot)
- **Charging-Animation** wird automatisch gestartet, wenn die Spannung über `charge` steigt (z.B. Ladegerät angeschlossen). Beim Sinken unter `charge` schaltet die Firmware zurück auf `defaultAnimation`.

## 9. Türen-Sensoren (LeftDoor / RightDoor)

Die Platine hat zwei Türschalter-Eingänge auf eigenen 2-Pin-Headern. Beide sind intern mit Pull-Up beschaltet — ein Schalter muss den jeweiligen Pin nur **gegen GND ziehen**.

Eingang	Header	Nano-Pin	Pin LOW (gegen GND)	Pin HIGH (nicht angeschlossen)
<b>LeftDoor</b> (Dataport-Tür)	2-Pin links	D2	Tür offen → DPL + VU laufen	Tür geschlossen → DPL + VU komplett aus
<b>RightDoor</b> (Charge-Bay-Tür)	2-Pin rechts	D4	Tür offen → CBI + DCBI laufen	Tür geschlossen → CBI + DCBI komplett aus

**Logik:** LOW = "Tür offen" = LEDs an. HIGH = "Tür zu" = LEDs aus. Das spart Strom und vermeidet Lichtaustritt durch geschlossene Klappen.

### So aktivierst du die Schalter

Es gibt mehrere Möglichkeiten, je nach Bauart deines Droiden:

#### Variante 1 — Reed-Schalter mit Magnet (empfohlen, Standard im R2-Builders-Szenario)

- Reed-Kontakt vom Typ **NO (Normally Open)** — schließt, wenn ein Magnet in die Nähe kommt
- Anschluss am 2-Pin-Header: ein Pin auf D2 (bzw. D4), der andere auf GND — die Polung ist egal, ein Reed ist nicht gerichtet
- Magnet so platzieren, dass er **bei geöffneter Tür neben dem Reed sitzt** → Reed schließt → Pin = LOW → DPL läuft
- Bei geschlossener Tür wandert der Magnet weg → Reed öffnet → Pin = HIGH → DPL aus

**Tipp Druckteile:** Auf Makerworld / Thingiverse gibt es fertige Magnet- und Reed-Halter für die R2-Türen, die direkt zum Servo-Mechanismus passen.

#### Variante 2 — Mechanischer Mikroschalter

- Tasterschalter mit Hebel, der durch den Türmechanismus betätigt wird
- Anschluss am Header: Schließer-Kontakt zwischen D2/D4 und GND
- Bei offener Tür: Hebel gedrückt → Pin = LOW
- Bei geschlossener Tür: Hebel los → Pin = HIGH

#### Variante 3 — Hall-Effekt-Sensor

- Digitaler Hall-Sensor (z.B. A3144 mit Open-Drain-Ausgang)
- VCC = 5V der Platine, GND = GND, Output am Header an D2/D4
- Magnet-Erkennung wie Reed, aber kontaktlos und langlebiger

### Variante 4 — Externer Controller / Servo-Steuerung

- Wenn ein anderer Microcontroller (z.B. Body-Master, Bridge) den Tür-Status kennt, kann dieser den D2/D4-Eingang direkt mit GND verbinden bzw. floaten lassen
- Achtung: Beide Seiten müssen denselben GND haben

### Variante 5 — Schalter komplett weglassen, LEDs immer an

- Drahtbrücke zwischen dem D2- bzw. D4-Pin und GND auf der 2-Pin-Header → Pin permanent LOW → DPL/CBI laufen immer (auch bei geschlossener Tür)
- Sinnvoll bei Show-/Display-Aufbau ohne bewegliche Türen, oder zum Testen

### Wenn keine Schalter und keine Brücke angeschlossen sind

Pins bleiben durch den internen Pull-Up auf HIGH → DPL und CBI sind **immer aus**. Das ist der Auslieferungszustand. Erste Inbetriebnahme: entweder Schalter anschließen oder Drahtbrücke setzen, sonst leuchtet nichts.

### Verkabelungs-Tipp

Für die typische R2-Verkabelung ist es praktisch, beide GND-Pins der zwei Türschalter-Header miteinander zu verbinden und nur noch zwei Signal-Drähte zur jeweiligen Tür zu führen. Wer feste Reed-Halter mit JST-Stecker hat, lötet einfach passende JST-Buchsen auf die 2-Pin-Header.

## 10. System-Befehle

Diese Befehle haben keinen Modul-Token, nur den Slash-Prefix:

Befehl	Wirkung
<code>/RESTART</code>	Soft-Reboot des Arduino Nano. EEPROM bleibt erhalten. Nützlich nach Settings-Änderungen, die einen Restart brauchen.
<code>/FACTORY</code>	EEPROM auf Defaults zurücksetzen, dann Restart. <b>Alle User-Settings gehen verloren</b> . Nützlich wenn die Konfiguration korrupt ist oder nach einem Firmware-Update mit geändertem Speicher-Layout.

Settings werden bei jedem `/MODUL/PARAM/WERT`-Befehl automatisch in EEPROM gesichert — du musst sie nicht manuell speichern.

## 11. Troubleshooting

### Befehl wird nicht akzeptiert (kein "OK" zurück)

1. **Zeilenende prüfen:** Terminal muss CR senden. Im Arduino-IDE-Serial-Monitor: "Both NL & CR" wählen.
2. **Baud-Rate prüfen:** 115200, sonst kommt nur Müll an.
3. **Befehlslänge prüfen:** Maximal 32 Zeichen. Lange Werte (z.B. `/DP/VL/11.5/12.0/12.5/13.0`) kratzen am Limit — keine zusätzlichen Leerzeichen.

4. **Modul-Token prüfen:** Tippfehler? Das Token ist Groß-/Kleinschreibungs-sensitiv (Großbuchstaben funktionieren immer).
5. **Modul existiert nicht:** Liste der gültigen Modul-Tokens in Abschnitt 6 prüfen.

## LEDs leuchten nicht

- Türen-Schalter HIGH? → DPL/VU aus (LeftDoor) bzw. CBI/DCBI aus (RightDoor)
- Helligkeit auf 0? → `/DP/BR/10` setzen
- Animation auf 0 (none)? → `/DP/AN/1` setzen
- Stromversorgung des LED-Strangs? Bei langen WS2812B-Strängen (LDPL = 43 LEDs) braucht es bis zu 2.5A bei voller Helligkeit
- Datenleitung defekt? Reihenfolge prüfen: Erste LED muss am Pin (DCBI=D6, LDPL=D7, UAL=D8, VU=D3) hängen, danach Daisy-Chain

## Falsche Farben (z.B. Rot wird Grün gezeigt)

WS2812B-Strips gibt es in mehreren Farb-Reihenfolgen (RGB, GRB, BGR). Die Firmware ist auf **GRB** kompiliert (Standard für WS2812B). Strips mit anderer Reihenfolge zeigen vertauschte Farben — Lösung: anderer Strip-Typ oder Firmware-Anpassung (Firmware-Modifikation, kein User-Setting).

## VU-Bar reagiert nicht auf Sound

Der VU-Effekt ist **synthetisch** (62-BPM-Sinuswelle), nicht audio-reaktiv. Siehe Abschnitt 1.

## Voltage-Anzeige zeigt falschen Wert

1. Spannungsteiler-Werte (R1/R2) prüfen — Default ist 100k/10k SMD-bestückt. Falls eigene THT-Widerstände eingelötet wurden, mit `/DP/RR/<R1>/<R2>` auf die echten Werte korrigieren.
2. Limit-Werte prüfen — `/DP/VL/RED/YELLOW/GREEN/CHARGE`. Reihenfolge muss aufsteigend sein (red < yellow < green < charge).
3. ANALOGPIN A0 darf maximal 5V sehen — Spannungsteiler muss die Bordnetz-Spannung auf < 5V herunterteilen. Mit Default-Werten:  $12V \times 10 / (100 + 10) = 1.09V$  ✓,  $24V \times 10 / (100 + 10) = 2.18V$  ✓.

## EEPROM-Settings nach Firmware-Update verloren

Wenn ein Firmware-Update das interne Speicher-Layout ändert, werden alle User-Settings (Helligkeit, Farben, Animationen, Voltage-Schwellen) automatisch auf Defaults zurückgesetzt. Bei Updates vorher die eigenen Settings notieren, damit sie nach dem Flashen wieder gesetzt werden können.

# 12. FAQ

## Was ist auf der DPL-Platine selbst zu sehen, was ist extern?

Auf der DPL-Platine (Classic + VU) sind die LEDs für die Data-Port-Anzeige selbst (TopBar, BlueLights, BarGraph, RedLights, WhiteLights). Bei DPL-VU zusätzlich die VU-Reihe. **Nicht** auf der DPL: das Charge-Bay-Indicator-Display (D-CBI oder CBI Plus), der LDPL-Bauchklappen-Strip, der UAL-Werkzeugarm-Strip — das sind separate Platinen, die an die DPL angeschlossen werden.

### Was passiert, wenn ich kein D-CBI / kein CBI Plus / kein LDPL angeschlossen habe?

Nichts Schlimmes — die Firmware treibt die Pins sowieso, das angeschlossene Modul (oder eben keines) entscheidet, was leuchtet. Befehle wie `/CB/AN/3` sind weiterhin gültig, du siehst die Animation einfach nicht.

### D-CBI oder CBI Plus — was ist der Unterschied?

**D-CBI** ist die digitale WS2812B-Variante mit **voller RGB-Farbtiefe** — Grundfarbe per `/DC/CO/...` frei wählbar, Animationen wie Rainbow, Color-Fade, Heart-Beat etc. nutzen die ganze Farb-Palette. **CBI Plus** ist die analoge Variante mit klassischen 3-mm-LEDs in **fixen Einzelfarben** (rot/gelb/grün/blau ab Werk verbaut) — wirkt wie das klassische CBI im Original-R2. Welches du nimmst, ist Geschmackssache — viele Builder kombinieren auch beide.

### Kann ich D-CBI UND CBI Plus gleichzeitig anschließen?

Ja. Sie hängen an unterschiedlichen Anschlüssen (CBI Plus am 5-Pin-Header, D-CBI am 3-Pin-Header an D6) und werden über getrennte Tokens (`/CB/...` und `/DC/...`) konfiguriert. Beide animieren parallel mit ihren eigenen Settings.

### Kann ich beide DPL-Platinen (Classic + VU) parallel betreiben, eine pro Datalink-Tür?

Ja, das ist sogar der typische Use-Case bei vielen R2-Builds. Jede Platine hat ihren eigenen Arduino — sie können unabhängig konfiguriert werden.

### Kann die DPL Classic später auf VU aufgerüstet werden?

Ja — eine externe VU Extension ist bei Printed-Droid separat erhältlich. Sie wird an den 3-Pin-Header `D3 VU` der DPL Classic angeschlossen. Die Firmware muss nicht angepasst werden.

### Funktioniert die Firmware mit jedem Arduino Nano?

Mit dem ATmega328-basierten Nano ja (Original-Nano oder kompatible Klone). Mit Nano Every (ATmega4809) oder Nano ESP32 NICHT — das sind andere Microcontroller mit anderer Firmware-Anforderung.

### Wie viele LEDs kann ich an LDPL / D-CBI / UAL anschließen?

Die Firmware erwartet die Standard-Werte (LDPL = 43 LEDs, D-CBI = 23 LEDs, UAL = 19 LEDs, VU = 28 LEDs). Mehr LEDs werden nicht angesteuert (überschüssige bleiben dunkel), weniger LEDs sind kein Problem (überschüssige Indizes werden geschrieben, existieren aber physisch nicht).

### Gibt es eine Web-UI?

Nein — die DPL Classic / DPL-VU werden ausschließlich über die serielle Schnittstelle konfiguriert.

### Wo finde ich aktuelle Firmware-HEX-Files?

<https://www.printed-droid.com/kb/dpl-classic-data-port-logics/> → Downloads. Neue Versionen werden von Printed-Droid bereitgestellt.

### Kann ich den I2C-Header an der DPL nutzen, um z.B. ein Display anzuschließen?

Mit der DPL-Firmware nicht — die nutzt I2C nicht. Für eigene Erweiterungen ist der Header (A4=SDA, A5=SCL, 5V, GND) frei verfügbar, allerdings müsstest du dafür eigene Firmware schreiben.

## Mein D-CBI / CBI Plus passt nicht in mein altes R2-Body — kann man die Platine teilen?

Ja. Beide Platinen haben einen Bereich mit Lochreihen ("Bruchkante"), an dem die PCB getrennt werden kann. Die zwei Hälften lassen sich anschließend über die zwei nebeneinander liegenden 3-Pin-Header (+S – jeweils) auf der Rückseite mit kurzen Pin-Brücken oder Drähten wieder verbinden. Wer den Standard-Form-Faktor behält, lässt diesen Bereich unverändert.

## Wo bekomme ich Hilfe, wenn etwas nicht funktioniert?

Drei Anlaufstellen:

- **Produktseiten** mit FAQ-Updates: <https://www.printed-droid.com>
- **Facebook-Gruppe** (Community + Direktkontakt): <https://www.facebook.com/groups/printeddroid/>
- **GitHub-Repo**: [https://github.com/PrintedDroid/DPL-VU\\_and\\_DPL-Classic](https://github.com/PrintedDroid/DPL-VU_and_DPL-Classic)

## 13. Eigene Firmware schreiben

Die Original-Firmware nutzt den Arduino Nano nur teilweise aus. Wer eigenen Code schreiben oder die Firmware modifizieren will, findet die nötigen Infos in diesem Handbuch:

- **Pin-Belegung** der Platine — siehe *Abschnitt 3.4*
- **Welche Pins frei sind** — D9, D13, A1-A7 sowie der I2C-Bus (A4/A5) und die UART-Pins (D0/D1) sind komplett unbeschaltet
- **Welche Pins fest verdrahtet sind** — MAX7219 (D10/11/12), WS2812B-Strips (D3/D6/D7/D8) und Spannungsteiler (A0). Diese Bauteile sind permanent angeschlossen — eigener Code sollte sie entweder bewusst ansteuern oder die Pins als Eingang ohne Pull-Up belassen
- **Konflikte** — siehe *Abschnitt 3.5* (Hardware-SPI vs. MAX7219-Bit-Bang, A6/A7 nur ADC, D13 mit on-board LED, D0/D1 USB-Konflikt während Upload)

**Upload eigener Firmware:** gleiches Verfahren wie das Firmware-Update in *Abschnitt 4* — XLoader für `.hex`-Dateien, oder direkt aus der Arduino-IDE über USB-Serial-Bootloader (kein ICSP-Programmer nötig, der on-board Nano hat einen Bootloader vorinstalliert).

## 14. Cheat-Sheet (Schnell-Referenz)

Eine Seite zum Ausdrucken und neben den Lötplatz / PC legen.

### Modul-Tokens

Token	Modul
DP	Data Port Lights (DPL on-board)
CB	CBI Plus (analog Charge Bay, externes Modul)
DC	D-CBI (digital Charge Bay, externes Modul)
LD	LDPL (Large DataPanel Logics, externes Modul)
UA	UAL (Utility Arm Lights)
VU	VU-Bar
TB BL BG RL WL	DPL-Sub-Bereiche

## Befehlsformat

```
<MODUL>/<PARAM>/<WERT>[/<WERT2>/<WERT3>...]  
Zeilenende: CR @ 115200 Baud Max. 32 Zeichen
```

## Parameter

Token	Bedeutung	Bereich
BR	Helligkeit	0-15 (DP/CB) oder 0-255 (DC/LD/UA/VU)
AN	Animation-Nummer	0-N
AS	Animation-Speed	ms
CO	Grundfarbe	R/G/B je 0-255 (nur WS2812B-Module)
VM	Voltage-Monitor	0/1
VL	Voltage-Schwellen	4 Floats: red/yellow/green/charge
RR	Spannungsteiler	R1/R2 in Ohm (Default 100000/10000)

## System-Befehle

Befehl	Wirkung
/RESTART	Soft-Reboot
/FACTORY	EEPROM zurücksetzen + Reboot

## Häufige Animationen (AN-Nummer)

Modul	0	1	2	3	4	5	6	7	8
DP	none	default	randomV U	startup					
CB	none	default	random	heart	charging				
DC	none	default	random	heart	randFade	colorFade	rainbow	heartBeat	charging
LD	none	default	fadeOut	rainbow					
UA	none	default	movie	abort					
VU	none	cycle	randomBa r						

## Befehlsbeispiele (Copy-Paste)

```
/DP/BR/15          DPL hell  
/LD/CO/0/0/255    LDPL Blau  
/LD/AN/2          LDPL Knight-Rider mit Fade  
/DC/AN/6          D-CBI Rainbow  
/CB/AN/3          CBI Heart  
/UA/AN/2          UAL Movie-Sequenz  
/VU/AN/1          VU automatisch zwischen Effekten wechseln  
/DP/VM/1          Voltage-Monitor an  
/DP/VL/11.5/12.0/12.5/13.0  Schwellen 12V Blei (rot/gelb/gruen/laden)  
/DP/VL/10.8/11.4/12.0/12.6  Schwellen 3S-Lipo
```

## Pin-Map (Arduino Nano on-board)

Pin	Funktion	Pin	Funktion
D2	Türschalter Left	D8	UAL Out
D3	VU Out	D9	frei
D4	Türschalter Right	D10	MAX7219 Data (Header <a href="#">D</a> )
D5	reserviert (3-Pin)	D11	MAX7219 Clock (Header <a href="#">C</a> )
D6	D-CBI Out	D12	MAX7219 Load (Header <a href="#">L</a> )
D7	LDPL Out	D13	frei (on-board LED)
A0	Voltage IN	A1-A3	frei (ADC)
A4	I2C SDA (frei)	A5	I2C SCL (frei)
A6/A7	frei (nur ADC)	D0/D1	UART (USB-Serial)