

Decoder: Miningen Light 2

Miningen Light 2 ist DER Decoder auf unserer Modellbahn-Anlage.de. Er soll die Verbindung zwischen DCC-Zentralen und Hardware sein.

Ein- und Ausgänge des Decoders

```
///////////////////////////////////////////////////////////////////////////////////////////////////////////////////  
/////////////////////////////////////////////////////////////////////////////////////////////////////////////////  
/////////////////////////////////////////////////////////////////////////////////////////////////////////////////  
/////////////////////////////////////////////////////////////////////////////////////////////////////////////////  
//  
// DARSTELLUNG DES DECODERS, SEINER EIN- UND AUSGÄNGE  
//  
//  
//      PC6          REST          RESET-o|1____28|o-Druckknopf/I/O 7 analog  
5        PC5  
//      PD0         digital 0 (RX)    I/O 3-o|2       27|o-I/O 6           analog  
4        PC4  
//      PD1         digital 1 (TX)    I/O 5-o|3       26|o-DKnopf LED       analog  
3        PC3  
//      PD2         digital 2          DCC IN-o|4       25|o-I/O 2           analog  
2        PC2  
//      PD3         digital 3 ~       WS2812B-o|5       24|o-I/O 1           analog  
1        PC1  
//      PD4         digital 4         BOARD LED-o|6       23|o-I/O 4           analog  
0        PC0  
//  
//              VCC-o|7       22|o-GND  
//              GND-o|8       21|o-AREF  
//      PB6          CLOCK-o|9       20|o-VCC  
//      PB7          CLOCK-o|10      19|o-LED5  
digital 13      PB5  
//      PD5         digital 5 ~       LED1-o|11      18|o-LED4  
digital 12      PB4  
//      PD6         digital 6 ~       LED3-o|12      17|o-LED6           ~  
digital 11      PB3  
//      PD7         digital 7          LED2-o|13      16|o-LED7           ~  
digital 10      PB2  
//      PB0         digital 8          Relais-o|14_____15|o-Servo/LED8     ~  
digital 9       PB1  
//  
  
// Define Pins  
#define IO3 0  
#define IO5 1  
#define DCC 2
```

```
#define WS2812B 3
#define BOARD_LED 4
#define LED1 5
#define LED3 6
#define LED2 7
#define Relais 8
#define Servo 9      #(oder LED8)
#define LED7 10
#define LED6 11
#define LED4 12
#define LED5 13

void setup()
{
    pinMode(RED, OUTPUT);
    pinMode(GREEN, OUTPUT);
    pinMode(BLUE, OUTPUT);
    digitalWrite(RED, HIGH);
    digitalWrite(GREEN, LOW);
    digitalWrite(BLUE, LOW);
}
```

LED per DCC ein- und ausschalten

```
#include <NmraDcc.h>

NmraDcc Dcc;
const int ledPin = 13;
const int dccPin = 2; // DCC-Eingang

void setup() {
    pinMode(ledPin, OUTPUT);
    Dcc.pin(dccPin, 0); // DCC-Eingang initialisieren
    Dcc.begin();        // DCC-Decoder starten
}

void loop() {
    Dcc.process();      // DCC-Befehle verarbeiten
}

// Diese Funktion wird aufgerufen, wenn ein DCC-Befehl für diese Adresse
// kommt
void notifyDccAccTurnout(uint16_t addr, uint8_t direction, uint8_t
outputPower) {
    if (addr == 1) { // Adresse 1
        if (direction == 1) {
            digitalWrite(ledPin, HIGH);
        } else {
```

```
        digitalWrite(ledPin, LOW);  
    }  
}  
}
```

LED faden mit DCC

```
const int dccInputPin = 2; // Eingang vom DCC-Decoder  
const int ledPin = 9;      // LED-Pin (muss PWM-fähig sein: 3, 5, 6, 9, 10, 11)  
  
int brightness = 0;  
int fadeAmount = 5; // Schrittweite für das Faden  
bool fadeIn = true; // Richtung: ein- oder ausfaden  
  
void setup() {  
    pinMode(dccInputPin, INPUT);  
    pinMode(ledPin, OUTPUT);  
}  
  
void loop() {  
    if (digitalRead(dccInputPin) == HIGH) {  
        // LED langsam ein- und ausfaden  
        analogWrite(ledPin, brightness);  
        brightness = brightness + fadeAmount;  
  
        if (brightness <= 0 || brightness >= 255) {  
            fadeAmount = -fadeAmount; // Richtung umkehren  
        }  
        delay(30); // Verzögerung für sanftes Faden  
    } else {  
        analogWrite(ledPin, 0); // LED aus  
        brightness = 0;  
    }  
}
```

LED schaltet hier sofort aus, wenn der DCC Befehl kommt

```
const int dccInputPin = 2; // Eingang vom DCC-Decoder  
const int ledPin = 9;      // LED-Pin (PWM-fähig: 3, 5, 6, 9, 10, 11)  
  
int brightness = 0;  
int fadeAmount = 5;  
bool fadeIn = true;  
  
void setup() {  
    pinMode(dccInputPin, INPUT);
```

```
pinMode(ledPin, OUTPUT);  
}  
  
void loop() {  
  if (digitalRead(dccInputPin) == HIGH) {  
    // LED langsam ein- und ausfaden  
    analogWrite(ledPin, brightness);  
    brightness = brightness + fadeAmount;  
  
    if (brightness <= 0 || brightness >= 255) {  
      fadeAmount = -fadeAmount;  
    }  
    delay(30);  
  } else {  
    // LED sofort ausschalten, wenn DCC-Befehl LOW ist  
    analogWrite(ledPin, 0);  
    brightness = 0;  
  }  
}
```

Mehrere LEDs, wobei jede LED auf eine eigene DCC Adresse hört

- DCC-Signal an Arduino-Pin 2 (über Optokoppler oder DCC-Shield)
- Jede LED an einen PWM-fähigen Pin (z.B. 3, 5, 6, 9, 10, 11)
- Jede LED hat eine eigene DCC-Adresse und wird unabhängig gesteuert.
- Der Fade-Effekt läuft nur, wenn der DCC-Befehl (direction == 1) aktiv ist.
- Bei direction == 0 wird die LED sofort ausgeschaltet.
- Du kannst die Anzahl der LEDs und Adressen einfach anpassen.

```
#include <NmraDcc.h>  
  
NmraDcc Dcc;  
const int dccPin = 2; // DCC-Eingang  
  
// LED-Pins und zugehörige DCC-Adressen  
struct LED {  
  int pin;  
  int dccAddr;  
  int brightness;  
  int fadeAmount;  
  bool isFading;  
};  
  
LED leds[] = {  
  {3, 1, 0, 5, false}, // LED an Pin 3, DCC-Adresse 1  
  {5, 2, 0, 5, false}, // LED an Pin 5, DCC-Adresse 2
```

```
{6, 3, 0, 5, false}, // LED an Pin 6, DCC-Adresse 3
{9, 4, 0, 5, false}, // LED an Pin 9, DCC-Adresse 4
{10, 5, 0, 5, false}, // LED an Pin 10, DCC-Adresse 5
{11, 6, 0, 5, false} // LED an Pin 11, DCC-Adresse 6
};
const int ledCount = 6;

void setup() {
  for (int i = 0; i < ledCount; i++) {
    pinMode(leds[i].pin, OUTPUT);
  }
  Dcc.pin(dccPin, 0);
  Dcc.begin();
}

void loop() {
  Dcc.process();
  for (int i = 0; i < ledCount; i++) {
    if (leds[i].isFading) {
      leds[i].brightness += leds[i].fadeAmount;
      if (leds[i].brightness <= 0 || leds[i].brightness >= 255) {
        leds[i].fadeAmount = -leds[i].fadeAmount;
      }
      analogWrite(leds[i].pin, leds[i].brightness);
      delay(30);
    }
  }
}

// Wird aufgerufen, wenn ein DCC-Accessory-Befehl kommt
void notifyDccAccTurnout(uint16_t addr, uint8_t direction, uint8_t
outputPower) {
  for (int i = 0; i < ledCount; i++) {
    if (leds[i].dccAddr == addr) {
      if (direction == 1) {
        leds[i].isFading = true; // LED faden lassen
      } else {
        leds[i].isFading = false;
        analogWrite(leds[i].pin, 0); // LED sofort aus
        leds[i].brightness = 0;
      }
    }
  }
}
```

Programm Schweißlicht im BW

Hier ist ein Arduino-Programm, das 12 LEDs unabhängig voneinander über DCC-Befehle steuert. Jede LED flackert mit einer individuell einstellbaren Frequenz, solange der zugehörige DCC-Ausgang aktiv

ist. Das Programm nutzt die NmraDcc-Library, um DCC-Befehle zu decodieren.

Hardware-Voraussetzungen:

- Arduino (z.B. Uno, Nano, Mega)
- DCC-Schnittstelle (z.B. Optokoppler oder DCC-Shield) an Pin 2
- 12 PWM-fähige Pins für die LEDs (z.B. 3, 5, 6, 9, 10, 11 und ggf. weitere, je nach Board)
- 12 Widerstände (z.B. 220 Ohm) für die LEDs

```
#include <NmraDcc.h>

// DCC-Eingang
const int dccPin = 2;

// Struktur für jede LED
struct FlackerLED {
    int pin;           // Arduino-Pin
    int dccAddr;       // DCC-Adresse
    bool isFlackern;    // Flackerzustand
    unsigned long lastChange; // Letzte Änderung
    int flackerDelay;   // Flackerfrequenz (ms)
};

FlackerLED leds[] = {
    {3, 1, false, 0, 50}, // LED an Pin 3, DCC-Adresse 1, 50ms
    {5, 2, false, 0, 70}, // LED an Pin 5, DCC-Adresse 2, 70ms
    {6, 3, false, 0, 90}, // LED an Pin 6, DCC-Adresse 3, 90ms
    {9, 4, false, 0, 110}, // LED an Pin 9, DCC-Adresse 4, 110ms
    {10, 5, false, 0, 130}, // LED an Pin 10, DCC-Adresse 5, 130ms
    {11, 6, false, 0, 150}, // LED an Pin 11, DCC-Adresse 6, 150ms
    {A0, 7, false, 0, 170}, // LED an Pin A0, DCC-Adresse 7, 170ms (als
digitaler Pin)
    {A1, 8, false, 0, 190}, // LED an Pin A1, DCC-Adresse 8, 190ms
    {A2, 9, false, 0, 210}, // LED an Pin A2, DCC-Adresse 9, 210ms
    {A3, 10, false, 0, 230}, // LED an Pin A3, DCC-Adresse 10, 230ms
    {A4, 11, false, 0, 250}, // LED an Pin A4, DCC-Adresse 11, 250ms
    {A5, 12, false, 0, 270}, // LED an Pin A5, DCC-Adresse 12, 270ms
    {4, 13, false, 0, 290}, // LED an Pin 4, DCC-Adresse 13, 290ms
    {7, 14, false, 0, 310}, // LED an Pin 7, DCC-Adresse 14, 310ms
    {8, 15, false, 0, 330}, // LED an Pin 8, DCC-Adresse 15, 330ms
    // Hier ggf. weitere LEDs ergänzen
};

const int ledCount = 12; // Anzahl der LEDs

NmraDcc Dcc;

void setup() {
    // Alle LED-Pins als Ausgang setzen
    for (int i = 0; i < ledCount; i++) {
        pinMode(leds[i].pin, OUTPUT);
    }
}
```

```
    digitalWrite(leds[i].pin, LOW);
}

// DCC initialisieren
Dcc.pin(dccPin, 0);
Dcc.begin();
}

void loop() {
    Dcc.process(); // DCC-Befehle verarbeiten

    // Flackern für jede LED steuern
    for (int i = 0; i < ledCount; i++) {
        if (leds[i].isFlackern) {
            if (millis() - leds[i].lastChange > leds[i].flackerDelay) {
                // LED-Zustand umschalten
                digitalWrite(leds[i].pin, !digitalRead(leds[i].pin));
                leds[i].lastChange = millis();
            }
        } else {
            digitalWrite(leds[i].pin, LOW); // LED aus
        }
    }
}

// Wird aufgerufen, wenn ein DCC-Accessory-Befehl kommt
void notifyDccAccTurnout(uint16_t addr, uint8_t direction, uint8_t
outputPower) {
    for (int i = 0; i < ledCount; i++) {
        if (leds[i].dccAddr == addr) {
            leds[i].isFlackern = (direction == 1); // Bei direction=1 flackern,
sonst aus
            if (!leds[i].isFlackern) {
                digitalWrite(leds[i].pin, LOW); // Sofort ausschalten
            }
        }
    }
}
```

From:
<https://wiki.modellbahn-anlage.de/> - Wiki der Modellbahn-Anlage.de

Permanent link:
<https://wiki.modellbahn-anlage.de/decoder/modellbahn-anlage.de/miningen-light-2?rev=1767767238>

Last update: 07.01.2026 07:27

