FEUERHAND BEAT LANTERN

Vollstaendiger Bau- und Programmierplan

Referenzdesign V2.0 - 14. September 2026

Feuerhand-Sturmlaterne als akustisch reaktive LED-Lichtskulptur
4 Modi | 3000 K | RGBW | Beat-Erkennung | Peak-Flash | USB-C

1. Projektziel und Design-Freeze

Ziel ist der reversible bzw. moeglichst substanzschonende Umbau einer klassischen Feuerhand-Sturmlaterne zu einer vollstaendig elektrischen, akustisch reaktiven Lichtskulptur. Die originale Aussenwirkung bleibt erhalten. Der sichtbare Dochtknopf dient als Moduswaehler; Stromschalter und USB-C sitzen unauffaellig an der Unterseite.

Parameter	Festlegung
Licht unten	7x RGBW, echtes Warmweiss 3000 K, diffuser Flammenkoerper
Licht oben	Cree XP-G4 3000 K, kurzer Hochleistungs-Peak bis ca. 3 A
Mikrocontroller	Seeed Studio XIAO ESP32-C3
Audio	INMP441 I2S MEMS; Bassband ca. 45-180 Hz
Bedienung	ON/OFF unten; originaler Dochtknopf ueber 24-Rastungen-Encoder
Akku	1x geschuetzter Keeppower P2160TC 21700, 6000 mAh / 21.6 Wh, USB-C
Laden	5 V USB-C, ca. 1.5 A; Herstellerwert ca. 4 h, Ziel praxisnah 4-5 h
Laufzeit	Entwicklungsziel 7-8 h im gemischten Musikbetrieb
Modi	1 FLAME, 2 PULSE, 3 COLOR BEAT, 4 COLOR BEAT + FLASH

Hinweis: Die genannten Innenmasse stammen aus manuellen Messungen der vorhandenen Laterne: Tankboden ca. 110 mm nutzbare Breite und 25-30 mm Hoehe; obere Flash-Zone ca. 52 mm Breite x 25 mm Hoehe plus ca. 50 x 30 mm Zusatzraum. Vor Bohren, Fraesen oder 3D-Druck muessen diese Masse am konkreten Exemplar mit Messschieber verifiziert werden.

Sicherheit: Nach diesem Umbau darf die Laterne nicht mehr mit Petroleum oder offener Flamme betrieben werden. Vor mechanischer Bearbeitung eines gebrauchten Tanks muss er vollstaendig entleert, fachgerecht gereinigt und frei von Brennstoffdaempfen sein. Li-Ion-Zelle niemals loeten, oeffnen, quetschen oder in Metallkontakt bringen.

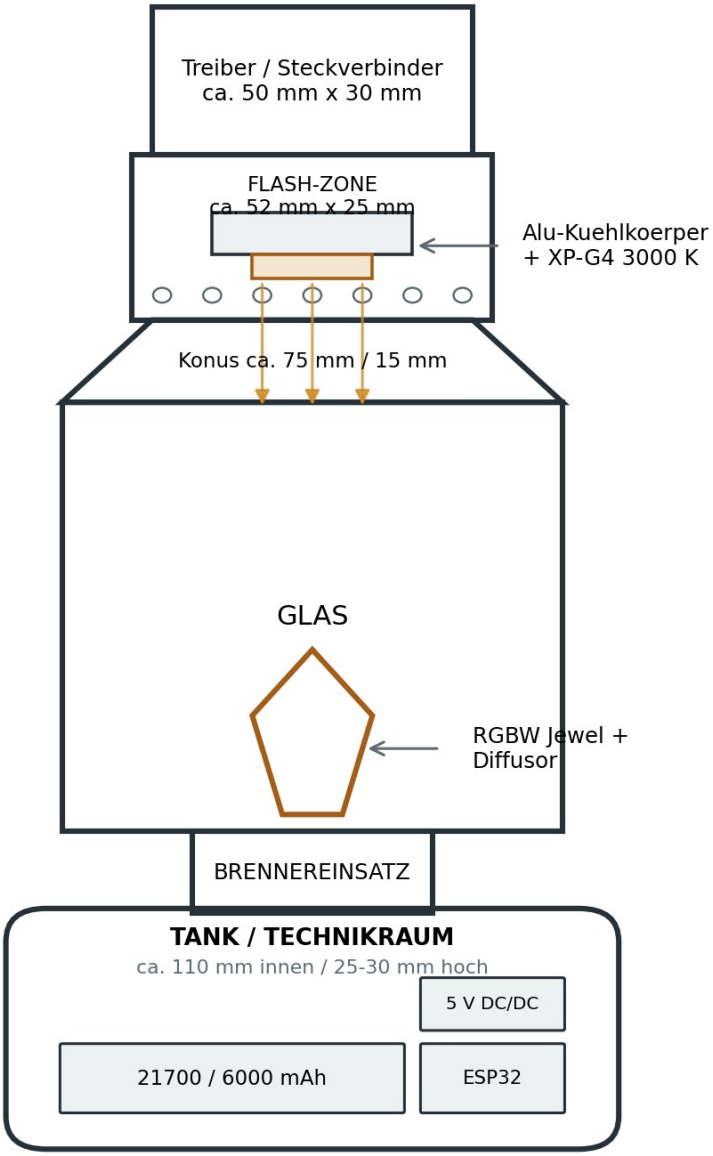
2. Funktionslogik der vier Modi

Modus	Verhalten	Oberer Flash
1 - FLAME	Organisches, asynchrones 3000-K-Flackern; warmweiss + etwas Bernstein/Rot.	Aus
2 - PULSE	FLAME bleibt Grundlage; Bass/Kick hebt Helligkeit mit schneller Attack und weichem Decay an.	Aus
3 - COLOR BEAT	RGB-Farbe und Helligkeit reagieren auf Bassenergie; kleiner Warmweissanteil erhaelt Laternencharakter.	Aus
4 - COLOR BEAT + FLASH	Wie Modus 3. Bei seltenen Extrem-Peaks wird die untere Quelle kurz gedimmt und der obere 3000-K-Flash gezuendet.	50-80 ms; mindestens 700 ms Sperrzeit; optionaler Double-Flash nur bei Superpeak

Gestaltungsprinzip: Der obere Flash darf nicht zur Stroboskop-Lichtorgel werden. Er ist eine zweite dramaturgische Ebene. Ein normaler Beat bewegt nur die untere Lichtquelle; erst ein statistisch deutlich ueber dem laufenden Bassniveau liegender Peak aktiviert die obere Quelle.

3. Mechanische Architektur

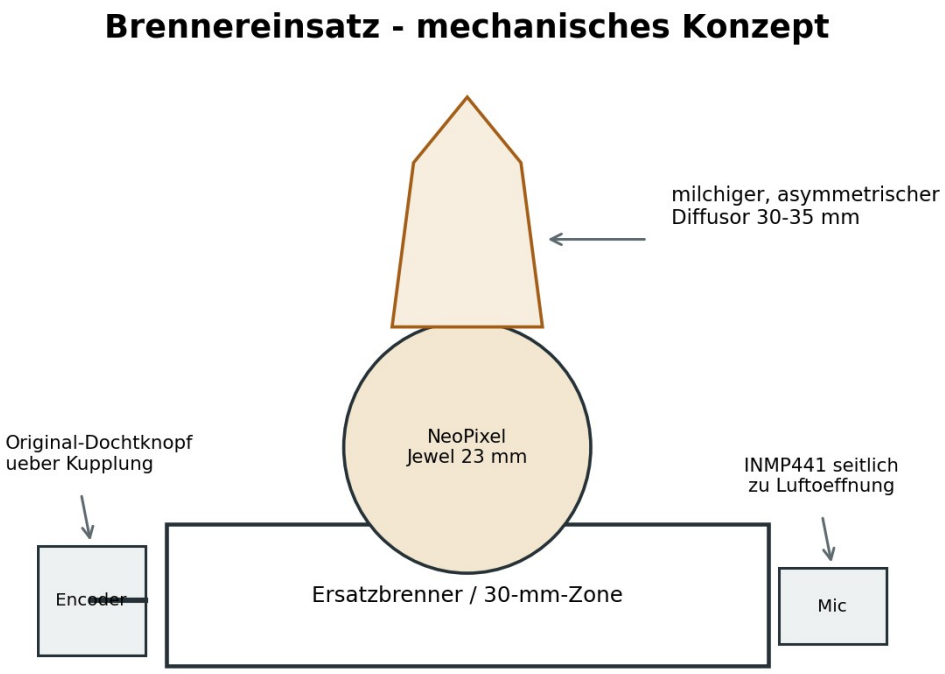
Mechanischer Referenzaufbau - Schnitt (nicht massstaeblich)



3.1 Tank / Unterseite

- Akku horizontal in einem isolierenden, formschluessigen Halter am flachen Tankboden. Zielmass der Zelle: ca. 77 x 22 mm.
- XIAO, Pololu-Regler, Pegelwandler, Sicherung und Steckverbinder auf G10/FR4- oder 3D-gedrucktem Elektroniktraeger; kein blankes Leiterplattenmaterial direkt auf Lampenmetall.
- USB-C-Einbaubuchse wird ueber ein kurzes 1:1-Panelkabel direkt mit dem USB-C-Port des P2160TC verbunden. Laden nur bei Hauptschalter OFF.
- Hauptschalter trennt die gesamte Last hinter Akku/Sicherung. Er muss mindestens 5 A DC bei Niederspannung sicher schalten koennen.
- 5-A-Sicherung so nah wie praktisch moeglich am Akku-Ausgang; danach sternfoermige Verteilung zu 5-V-Zweig und oberem Flash-Zweig.

3.2 Brennerzone

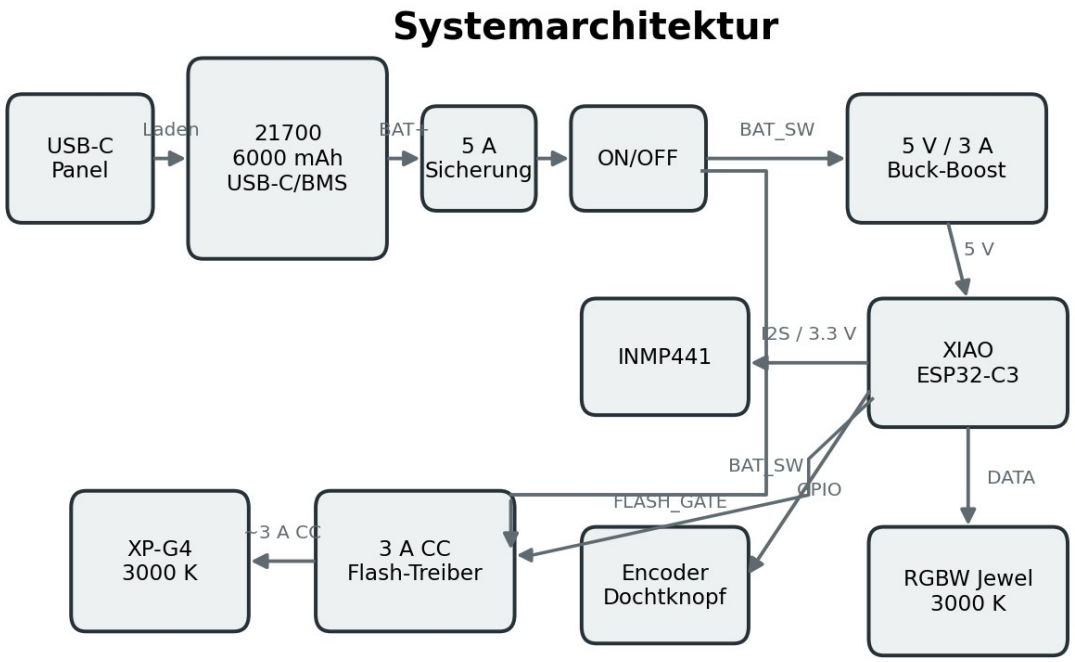


- Empfehlung: separaten Feuerhand-B-276-Ersatzbrenner umbauen und das Originalteil aufbewahren.
- NeoPixel Jewel mittig unter einem milchigen, asymmetrischen Diffusor montieren. Kein direkter Blick auf einzelne LEDs.
- INMP441 seitlich im Brennerraum, Mikrofonoefnung zu einer vorhandenen Luftoeffnung; 10-20 mm Abstand zu Schaltregler und Hochstromleitung.
- Die originale Dochtknopf-Achse wird mechanisch vom Dochttransport entkoppelt und ueber eine kurze Kupplung auf den 6-mm-Encoder gefuehrt. Externe Optik bleibt unveraendert.
- Kupplung bevorzugt als geschlitzte Klemmkupplung/3D-Druck-Adapter mit Madenschraube; Achsdurchmesser am Original vor CAD-Erstellung messen.

3.3 Oberkopf / Peak-Flash

- Cree XP-G4 3000 K auf 19.9-mm-Starplatine, Chip nach unten in den Glaszylinder.
- Starplatine mit duennem Waermeleitpad oder Paste auf 35-40-mm-Aluminium-Waermeverteiler/Kuehlkoerper schrauben. Vorhandene umlaufende Luftloecher bleiben frei.
- Keine enge Optik. Der native ca. 120-Grad-Abstrahlwinkel ist erwuenscht, weil das ganze Glas kurz aufleuchten soll.
- Flash-Treiber in den darueberliegenden ca. 50 x 30 mm Technikraum setzen, nicht direkt auf die LED-Platine.
- Kabel entlang einer hinteren Seitenstrebe unauffaellig nach oben fuehren: BAT+, GND und FLASH_GATE. Fuer BAT+/GND AWG22-24 Silikonkabel; Steuerleitung AWG28-30.

4. Elektrische Architektur



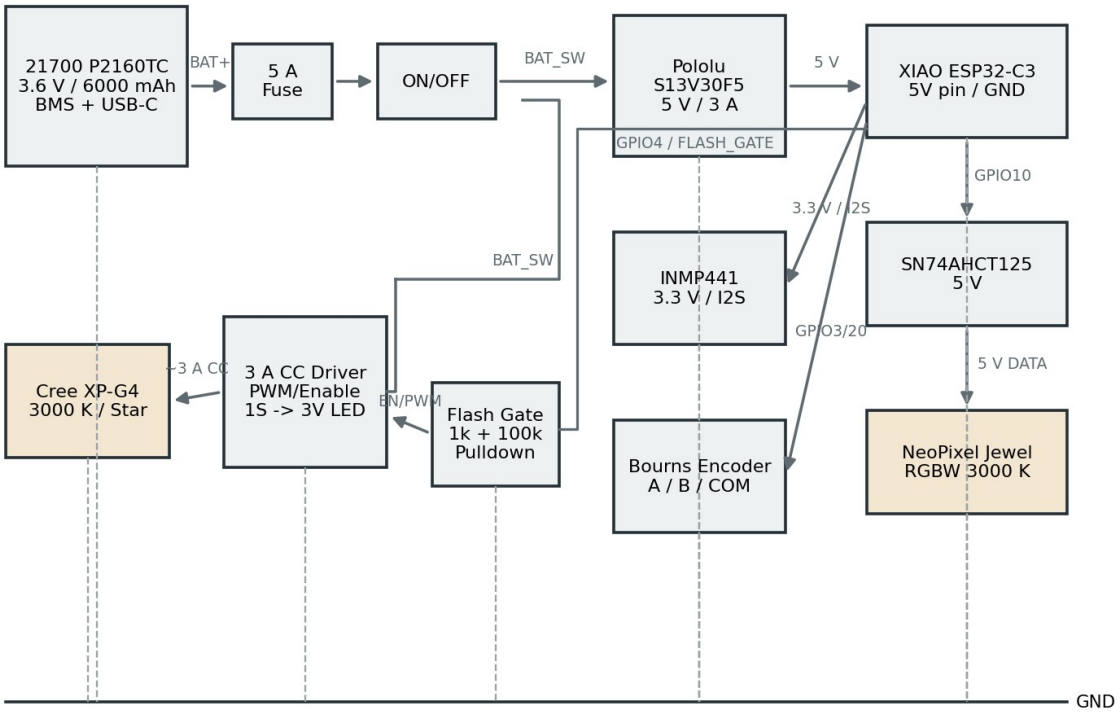
Hauptschalter trennt beide Lastzweige; USB-C-Laden bleibt bei OFF direkt am Akku erreichbar.

4.1 Leistungszweige

Der Akku speist zwei getrennte Lastzweige hinter Sicherung und Hauptschalter. Zweig A wird ueber einen 5-V-Buck-Boost-Regler stabilisiert und versorgt Mikrocontroller, Pegelwandler und RGBW-Jewel. Zweig B versorgt den oberen Hochleistungs-Flash direkt aus der Li-Ion-Spannung ueber eine Konstantstromstufe. Diese Trennung verhindert, dass ein 3-A-Flash den 5-V-Logikzweig sichtbar einbrechen laesst.

Elektrische Verdrahtung - Referenzdesign

Alle Massen gemeinsam. Sicherung unmittelbar hinter dem Akku; Hauptschalter danach vor der Lastverzweigung.



Netz	Quelle	Last / Zweck	Leitung
BAT_RAW	P2160TC nach Sicherung/Schalter	Pololu 5-V-Regler und Flash-Treiber	AWG22-24 fuer Hochstrompfade
5V_SYS	Pololu S13V30F5	XIAO 5V, SN74AHCT125 VCC, NeoPixel V+	AWG24-26
3V3	XIAO 3V3	INMP441 VDD	AWG28-30
GND	gemeinsame Masse	alle Module	sternnah, Hochstromruecklauf nicht durchs Mic fuehren
FLASH_GATE	GPIO4	Enable/PWM des Flash-Treibers	AWG28-30; 1k Serie + 100k Pulldown

4.2 Pinbelegung XIAO ESP32-C3

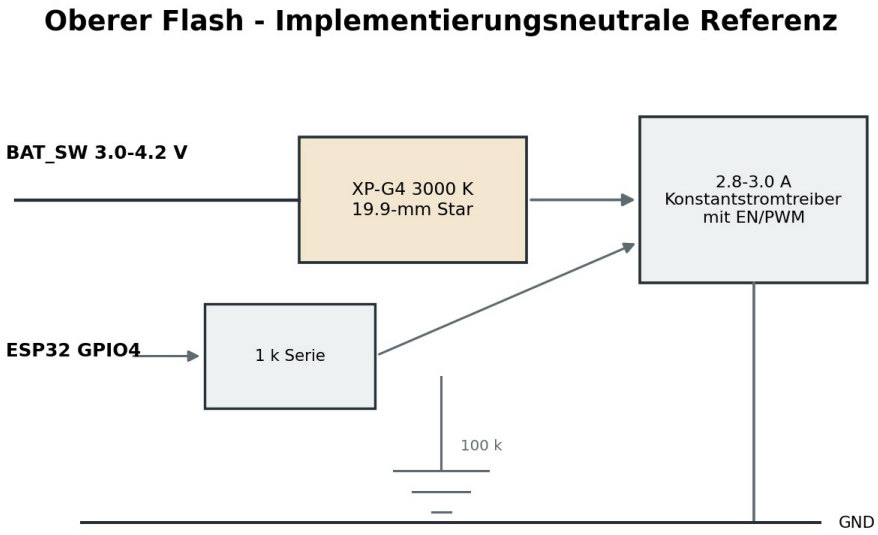
Funktion	Board-Pin	GPIO	Verbindung
NeoPixel DATA	D10	GPIO10	via SN74AHCT125, danach 330 Ohm zum DIN
Encoder A	D1	GPIO3	INPUT_PULLUP
Encoder B	D7	GPIO20	INPUT_PULLUP
Flash Gate	D2	GPIO4	1k Serie; 100k Pulldown nach GND
I2S BCLK	D4	GPIO6	INMP441 SCK/BCLK
I2S WS/LRCLK	D5	GPIO7	INMP441 WS
I2S DATA	D6	GPIO21	INMP441 SD
3V3	3V3	-	INMP441 VDD
5V	5V	-	aus Pololu 5V_SYS
GND	GND	-	gemeinsame Masse

Hinweis: GPIO2, GPIO8 und GPIO9 sind beim ESP32-C3 Boot-/Strapping-relevant. Das Referenzdesign vermeidet sie fuer externe Signalquellen. GPIO4 fuer FLASH_GATE bekommt zusaetzlich einen Hardware-Pulldown, damit der Flash waehrend Reset/Boot sicher AUS bleibt.

4.3 RGBW-Jewel

- NeoPixel Jewel 7x RGBW Warm White 3000 K, ca. 23 mm Durchmesser.
- Zwischen 5V_SYS und GND direkt am Jewel: 470-1000 uF Elektrolytkondensator plus 100 nF Keramik.
- DATA: XIAO GPIO10 -> SN74AHCT125 -> 330-Ohm-Serienwiderstand -> DIN.
- Software-Masterhelligkeit im Referenzcode 55 %. Damit bleibt die Worst-Case-RGBW-Leistung begrenzt; in den realen Flammenmodi liegt der Durchschnitt deutlich darunter.

4.4 Oberer 3000-K-Flash



Ziel: 2.8-3.0 A Peak, 50-80 ms, mind. 700 ms Sperrzeit. Referenz: modifizierter 3040-mA-Taschenlampentreiber oder gleichwertiger PWM-Treiber.

Referenzziel: ca. 3.0 A Peak durch die XP-G4, 50-80 ms. Ein kommerzielles 17-mm-AMC7135x8-Taschenlampenmodul mit 3040 mA kann als mechanische/elektrische Basis dienen, muss aber von einer fachkundigen Person fuer eine externe PWM-/Enable-Steuerung umgebaut werden. Alternativ ist jeder kompakte, PWM-faehige 3-A-Konstantstromtreiber fuer 1S-Li-Ion und eine einzelne 3-V-LED zulässig.

- XP-G4 Anode an BAT_RAW; Kathode an Konstantstromsenke.
- Treiber-Nennstrom 2.8-3.0 A. Keine direkte MOSFET-Ansteuerung ohne Strombegrenzung.
- FLASH_GATE aktiv HIGH, 3.3 V Logik; Hardware-Pulldown 100 kOhm und 1 kOhm Serienwiderstand.
- Bei niedriger Akkuspannung darf die Flash-Helligkeit etwas abfallen; das ist im Referenzdesign akzeptiert und reduziert Komplexitaet/Waerme.
- Thermik: nur gepulst; Firmware erzwingt mindestens 700 ms zwischen Hauptblitzen und typischerweise 65 ms On-Time.

5. Akku, Laden und Laufzeit

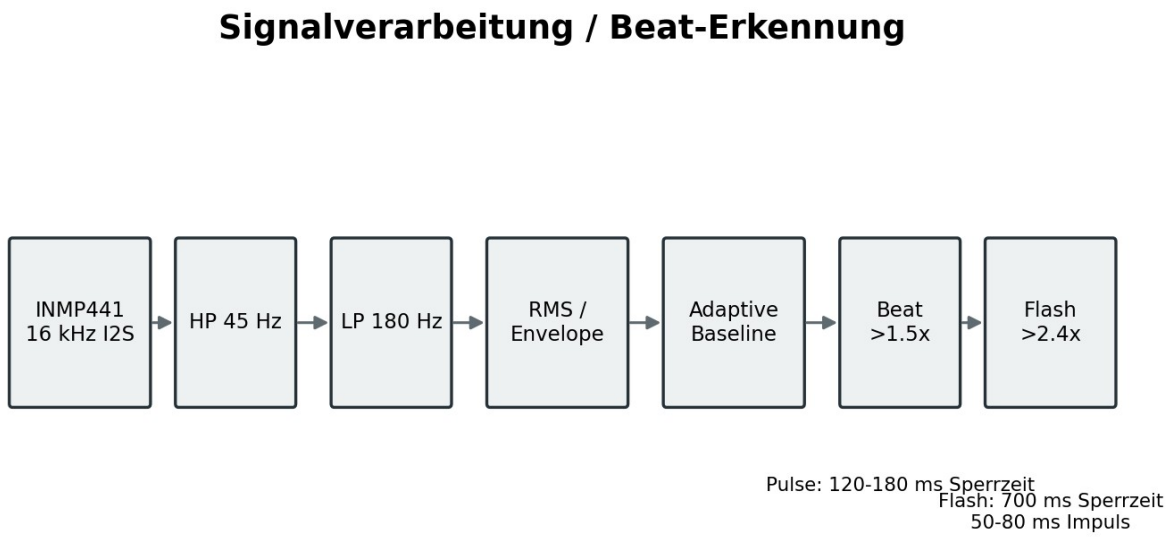
Groesse	Wert
Akku	Keeppower P2160TC, geschuetzter 21700 mit Type-C
Nennspannung	3.6 V
Kapazitaet	6000 mAh / ca. 21.6 Wh
Abmessungen	ca. 21.8 x 76.8-77 mm
Max. kontinuierliche Entladung	10 A
USB-C Ladestrom	ca. 1.5 A
Hersteller-Ladezeit	ca. 4 h Standardwert
Projektziel	4-5 h Laden; 7-8 h Betrieb

Energiebudget: 21.6 Wh nominell. Bei grob 85-90 % nutzbarer Systemenergie stehen ca. 18-19 Wh zur Verfuegung. Fuer 8 h Betrieb entspricht das etwa 2.25 W mittlerer Leistungsaufnahme. Das Referenzdesign liegt im Normalbetrieb darunter; der sehr helle obere Flash hat wegen seiner kurzen Einschaltdauer nur einen kleinen Einfluss auf den Energieverbrauch.

Szenario	Geschaetzte mittlere Leistung	Erwartung
FLAME	ca. 0.8-1.3 W	deutlich >8 h moeglich
PULSE	ca. 1.0-1.5 W	typ. >8 h
COLOR BEAT	ca. 1.2-1.8 W	ca. 8-12 h je Helligkeit
COLOR BEAT + FLASH	ca. 1.4-2.2 W	Ziel 7-8 h mit Reserve

Hinweis: Laufzeiten sind Konstruktionswerte, keine Garantie. Sie muessen am fertigen Prototyp mit realer Musik, finaler Helligkeit und Akku bei Raumtemperatur gemessen werden. Falls 7 h unterschritten werden, zuerst PIXEL_MASTER reduzieren; Flash-Peaks sind energetisch weniger relevant als dauerhaft hohe RGBW-Helligkeit.

6. Audioerkennung und Lichtdramaturgie



Das Mikrofon wird mit 16 kHz I2S abgetastet. Ein digitaler Hochpass bei ca. 45 Hz entfernt tieffrequenten Drift, ein Tiefpass bei ca. 180 Hz reduziert Stimmen, Hi-Hats und obere Mitten. Aus dem Band wird ein RMS-/Envelope-Wert gewonnen. Ein schneller Envelope folgt Peaks, ein langsamer Envelope bildet den adaptiven Musikpegel. Dadurch passt sich die Lampe automatisch an unterschiedliche Lautstaerken an.

Ereignis	Startwert	Sperrzeit / Envelope
Normaler Beat	fast/slow >= 1.50	mind. 145 ms; weicher Decay
Flash-Peak	fast/slow >= 2.40	mind. 700 ms
Superpeak	fast/slow >= 3.10	optional 65 ms + 45 ms Pause + 25 ms zweiter Impuls

Diese Schwellen sind bewusst Startwerte. Nach dem Einbau veraendert das Metallgehaeuse den akustischen Pegel. Der Erbauer sollte die Schwellen im fertigen Gehaeuse mit dem vorgesehenen E-Piano/PA-System justieren. Ziel ist: Modus 2 reagiert oft, Modus 3 musikalisch, Modus 4 nur bei echten Hoehepunkten.

7. Software - vollstaendiger Referenzcode

Hinweis: Der Code ist ein Referenz- und Kalibrierstand fuer Arduino-ESP32. Vor finalem Einbau auf dem verwendeten Arduino-ESP32-Core kompilieren und auf einem Tischaufbau testen. Bibliothek: Adafruit NeoPixel. Die I2S-API stammt aus ESP-IDF/Arduino-ESP32.

```
/*
  Feuerhand Beat Lantern - reference firmware V2.0
  Target: Seeed Studio XIAO ESP32-C3 / Arduino framework
  Lower light: Adafruit NeoPixel Jewel RGBW 3000 K
  Microphone: INMP441 I2S, L/R pin = GND (left channel)
  Upper flash: external 3 A constant-current stage, active HIGH on GPIO4

  IMPORTANT: Thresholds below are starting values. Calibrate in the finished metal housing.
*/

#include <Arduino.h>
#include <Adafruit_NeoPixel.h>
#include <Preferences.h>
#include "driver/i2s.h"

// XIAO ESP32-C3 GPIO numbers (not board D-labels)
static constexpr int PIN_NEOPIXEL = 10; // D10 / GPIO10
static constexpr int PIN_ENC_A    = 3;  // D1  / GPIO3
static constexpr int PIN_ENC_B    = 20; // D7  / GPIO20
static constexpr int PIN_FLASH    = 4;  // D2  / GPIO4
static constexpr int PIN_I2S_BCLK = 6;  // D4  / GPIO6
static constexpr int PIN_I2S_WS   = 7;  // D5  / GPIO7
static constexpr int PIN_I2S_SD   = 21; // D6  / GPIO21

static constexpr int NUM_PIXELS = 7;
static constexpr uint32_t SAMPLE_RATE = 16000;
static constexpr int AUDIO_BLOCK = 64;

Adafruit_NeoPixel pixels(NUM_PIXELS, PIN_NEOPIXEL, NEO_GRBW + NEO_KHZ800);
Preferences prefs;

uint8_t mode = 1;                // 1..4
uint32_t lastModeChange = 0;
bool modeNeedsSave = false;

// Audio / filter state
float hpY = 0.0f, hpXPrev = 0.0f, lpY = 0.0f;
float fastEnv = 0.0f, slowEnv = 0.0020f;
float beatPulse = 0.0f;
uint32_t lastBeatMs = 0;
uint32_t lastFlashMs = 0;

// Flash state machine
struct FlashState {
  bool on = false;
  uint8_t phase = 0; // 0 idle, 1 main on, 2 gap, 3 second on
  uint32_t untilMs = 0;
} flash;

// Encoder polling state
uint8_t oldAB = 0;
int8_t encAcc = 0;

// Flicker state
float flicker[NUM_PIXELS] = {0.82,0.90,0.78,0.94,0.86,0.80,0.91};
float flickerTarget[NUM_PIXELS] = {0.82,0.90,0.78,0.94,0.86,0.80,0.91};
uint32_t nextFlickerTargetMs = 0;
uint16_t colorHue = 0;

// Tuning constants
static constexpr float HP_FC = 45.0f;
static constexpr float LP_FC = 180.0f;
static constexpr float MIN_AUDIO = 0.00035f;
static constexpr float BEAT_RATIO = 1.50f;
static constexpr float FLASH_RATIO = 2.40f;
static constexpr float SUPERFLASH_RATIO = 3.10f;
static constexpr uint32_t BEAT_REFRACT_MS = 145;
static constexpr uint32_t FLASH_REFRACT_MS = 700;
static constexpr uint32_t FLASH_MAIN_MS = 65;
static constexpr uint32_t FLASH_GAP_MS = 45;
static constexpr uint32_t FLASH_SECOND_MS = 25;
static constexpr bool ENABLE_DOUBLE_FLASH = true;
static constexpr float PIXEL_MASTER = 0.55f; // keeps RGBW current sane

void setupI2S() {
  i2s_config_t cfg = {};
```

```
  cfg.mode = (i2s_mode_t)(I2S_MODE_MASTER | I2S_MODE_RX);
  cfg.sample_rate = SAMPLE_RATE;
  cfg.bits_per_sample = I2S_BITS_PER_SAMPLE_32BIT;
  cfg.channel_format = I2S_CHANNEL_FMT_ONLY_LEFT;
  cfg.communication_format = I2S_COMM_FORMAT_STAND_I2S;
  cfg.intr_alloc_flags = ESP_INTR_FLAG_LEVEL1;
  cfg.dma_buf_count = 4;
  cfg.dma_buf_len = AUDIO_BLOCK;
  cfg.use_apll = false;
  cfg.tx_desc_auto_clear = false;
  cfg.fixed_mclk = 0;

  i2s_pin_config_t pins = {};
  pins.bck_io_num = PIN_I2S_BCLK;
  pins.ws_io_num = PIN_I2S_WS;
  pins.data_out_num = I2S_PIN_NO_CHANGE;
  pins.data_in_num = PIN_I2S_SD;

  i2s_driver_install(I2S_NUM_0, &cfg, 0, nullptr);
  i2s_set_pin(I2S_NUM_0, &pins);
  i2s_zero_dma_buffer(I2S_NUM_0);
}

float readBassRMS() {
  int32_t samples[AUDIO_BLOCK];
  size_t bytesRead = 0;
  i2s_read(I2S_NUM_0, samples, sizeof(samples), &bytesRead, portMAX_DELAY);
  int n = bytesRead / sizeof(int32_t);
  if (n <= 0) return 0.0f;

  const float dt = 1.0f / SAMPLE_RATE;
  const float rcHP = 1.0f / (2.0f * PI * HP_FC);
  const float aHP = rcHP / (rcHP + dt);
  const float rcLP = 1.0f / (2.0f * PI * LP_FC);
  const float aLP = dt / (rcLP + dt);

  double sumSq = 0.0;
  for (int i=0; i<n; ++i) {
    // INMP441 gives 24-bit audio inside a 32-bit slot. Relative scaling is sufficient.
    float x = (float)samples[i] / 2147483648.0f;
    hpY = aHP * (hpY + x - hpXPrev);
    hpXPrev = x;
    lpY += aLP * (hpY - lpY);
    sumSq += (double)lpY * (double)lpY;
  }
  return sqrtf((float)(sumSq / n));
}

void updateAudio(float rms) {
  // Fast peak envelope, slow adaptive baseline.
  fastEnv = max(rms, fastEnv * 0.86f);
  const float alphaSlow = (rms > slowEnv) ? 0.0040f : 0.0100f;
  slowEnv += alphaSlow * (rms - slowEnv);
  slowEnv = max(slowEnv, MIN_AUDIO);

  uint32_t now = millis();
  float ratio = fastEnv / slowEnv;

  if (rms > MIN_AUDIO && ratio >= BEAT_RATIO && now - lastBeatMs >= BEAT_REFRACT_MS) {
    beatPulse = min(1.0f, 0.40f + (ratio - BEAT_RATIO) * 0.55f);
    lastBeatMs = now;
  }

  // Flash only in mode 4 and only for rare peaks.
  if (mode == 4 && rms > MIN_AUDIO && ratio >= FLASH_RATIO &&
      now - lastFlashMs >= FLASH_REFRACT_MS && flash.phase == 0) {
    flash.phase = 1;
    flash.on = true;
    digitalWrite(PIN_FLASH, HIGH);
    flash.untilMs = now + FLASH_MAIN_MS;
    lastFlashMs = now;
    // Store super-peak decision in phase's high bit via a simple flag.
    if (ENABLE_DOUBLE_FLASH && ratio >= SUPERFLASH_RATIO) flash.phase = 11;
  }

  beatPulse *= 0.92f;
```

```
  if (beatPulse < 0.01f) beatPulse = 0.0f;
}

void serviceFlash() {
  uint32_t now = millis();
  if (flash.phase == 0 || (int32_t)(now - flash.untilMs) < 0) return;

  if (flash.phase == 1) { // normal single flash ends
    digitalWrite(PIN_FLASH, LOW);
    flash.on = false;
    flash.phase = 0;
  } else if (flash.phase == 11) { // super flash: main ended, create gap
    digitalWrite(PIN_FLASH, LOW);
    flash.on = false;
    flash.phase = 2;
    flash.untilMs = now + FLASH_GAP_MS;
  } else if (flash.phase == 2) {
    digitalWrite(PIN_FLASH, HIGH);
    flash.on = true;
    flash.phase = 3;
    flash.untilMs = now + FLASH_SECOND_MS;
  } else if (flash.phase == 3) {
    digitalWrite(PIN_FLASH, LOW);
    flash.on = false;
    flash.phase = 0;
  }
}

void updateEncoder() {
  // Ben Buxton style 4-state transition table; polling is sufficient for a slow wick knob.
  static const int8_t table[16] = {0,-1,1,0, 1,0,0,-1, -1,0,0,1, 0,1,-1,0};
  uint8_t ab = (digitalRead(PIN_ENC_A) << 1) | digitalRead(PIN_ENC_B);
  uint8_t idx = ((oldAB & 0x3) << 2) | (ab & 0x3);
  oldAB = ab;
  encAcc += table[idx];

  if (encAcc >= 4) {
    encAcc = 0;
    mode = (mode >= 4) ? 1 : mode + 1;
    lastModeChange = millis(); modeNeedsSave = true;
    beatPulse = 0.7f; // warm visual acknowledgement
  } else if (encAcc <= -4) {
    encAcc = 0;
    mode = (mode <= 1) ? 4 : mode - 1;
    lastModeChange = millis(); modeNeedsSave = true;
    beatPulse = 0.7f;
  }
}

uint8_t scale8(float v) {
  v = constrain(v * PIXEL_MASTER, 0.0f, 1.0f);
  return (uint8_t)lroundf(v * 255.0f);
}

void updateLowerLight() {
  uint32_t now = millis();
  if (now >= nextFlickerTargetMs) {
    nextFlickerTargetMs = now + random(22, 55);
    for (int i=0;i<NUM_PIXELS;i++) flickerTarget[i] = random(65,101) / 100.0f;
  }
  for (int i=0;i<NUM_PIXELS;i++) flicker[i] += 0.22f * (flickerTarget[i] - flicker[i]);

  colorHue += 55 + (uint16_t)(beatPulse * 220.0f);

  for (int i=0;i<NUM_PIXELS;i++) {
    float local = flicker[i];
    if (mode == 1 || mode == 2) {
      float pulse = (mode == 2) ? (1.0f + 0.38f * beatPulse) : 1.0f;
      float k = constrain(local * pulse, 0.0f, 1.15f);
      uint8_t w = scale8(0.78f * k);
      uint8_t r = scale8(0.30f * k);
      uint8_t g = scale8(0.08f * k);
      pixels.setPixelColor(i, pixels.Color(r,g,0,w));
    } else {
      // Color modes: colored beat body with a small warm-white component to retain lantern character.
      uint16_t hue = colorHue + i * 3800;
```

```
      uint8_t val = (uint8_t)constrain(110 + 105*local + 35*beatPulse, 0, 255);
      uint32_t c = pixels.gamma32(pixels.ColorHSV(hue, 225, val));
      uint8_t r=(c>>16)&0xFF, g=(c>>8)&0xFF, b=c&0xFF;
      float dip = (mode == 4 && flash.on) ? 0.68f : 1.0f; // contrast + peak-current relief
      r = scale8((r/255.0f)*dip); g = scale8((g/255.0f)*dip); b = scale8((b/255.0f)*dip);
      uint8_t w = scale8((0.10f + 0.10f*local) * dip);
      pixels.setPixelColor(i, pixels.Color(r,g,b,w));
    }
  }
  pixels.show();
}

void setup() {
  pinMode(PIN_FLASH, OUTPUT);
  digitalWrite(PIN_FLASH, LOW); // hardware 100k pulldown is still mandatory
  pinMode(PIN_ENC_A, INPUT_PULLUP);
  pinMode(PIN_ENC_B, INPUT_PULLUP);
  oldAB = (digitalRead(PIN_ENC_A)<<1) | digitalRead(PIN_ENC_B);

  pixels.begin(); pixels.clear(); pixels.show();
  randomSeed(esp_random());
  setupI2S();

  prefs.begin("beatlamp", false);
  mode = constrain((int)prefs.getUChar("mode", 1), 1, 4);
}

void loop() {
  updateEncoder();
  float bassRms = readBassRMS();
  updateAudio(bassRms);
  serviceFlash();
  updateLowerLight();

  if (modeNeedsSave && millis() - lastModeChange > 1500) {
    prefs.putUChar("mode", mode);
    modeNeedsSave = false;
  }
}
```

8. Firmware-Kalibrierung

1. Zunaechst nur XIAO + INMP441 + NeoPixel auf dem Tisch betreiben; Flash-Zweig abgeklemmt lassen.
2. Mit realer Musik in ca. geplanter Entfernung beobachten, ob Modus 2 auf Kick/Bass und nicht auf Sprache reagiert. Bei zu hektischer Reaktion BEAT_RATIO erhoehen; bei zu wenig Reaktion senken.
3. COLOR BEAT bei maximal vorgesehener Lautstaerke testen. Falls zu grell oder Laufzeit kritisch: PIXEL_MASTER von 0.55 auf 0.45-0.50 reduzieren.
4. Flash-Treiber zunaechst mit Labornetzteil und elektronischer Last/LED testen. Strom auf 2.8-3.0 A begrenzen.
5. Erst danach Flash mit ESP32 verbinden. FLASH_RATIO so einstellen, dass maximal wenige Peaks pro Minute ausloesen. Kein permanentes Blitzen.
6. 30 Minuten Musik-Dauertest im montierten Oberkopf: Temperatur des Alu-Traegers, Treibers und benachbarter Kunststoffteile kontrollieren.
7. Finaler Laufzeittest: voll laden, 8 Stunden typisches Programm laufen lassen und Restspannung/Abschaltpunkt dokumentieren.

9. Fertigungsreihenfolge

1. Laterne vollstaendig entleeren/reinigen; bei ehemaligem Petroleumtank erst nach sicherer Entgasung mechanisch bearbeiten.
2. Innenmasse mit Messschieber erfassen und CAD-Halter fuer Akku/Elektronik sowie Brenneradapter anpassen.
3. Ersatzbrenner zerlegen; Docht und Dochtmechanik entfernen, Originalknopf und Welle soweit moeglich erhalten.
4. Encoderkupplung anfertigen und mechanisch auf leichtgaengige Rastung pruefen.
5. RGBW-Jewel, Diffusor und Mikrofon in den Brennereinsatz integrieren.
6. Akkuhalter, Sicherung, Hauptschalter, USB-C-Paneldurchfuehrung und Elektroniktraeger im Tank montieren.
7. 5-V-System verdrahten und alleine testen.
8. Oberkopf: XP-G4 auf Alu-Kuehlkoerper montieren; Flash-Treiber oberhalb davon; Kabelzug durch Seitenstrebe.
9. Gesamtsystem elektrisch pruefen: Polaritaet, Kurzschluss, Ruhestrom, 5-V-Stabilitaet, Flash-Strom.

10. Firmware laden, Modi kalibrieren und 8-h-Burn-in mit Musik durchfuehren.
11. Kabel gegen Scheuern sichern, Steckverbinder verriegeln, alle Metallkanten mit Tuelle/Schrumpfschlauch isolieren.

10. Abnahmekriterien

Pruefung	Bestanden, wenn...
Aussenoptik	Von vorne keine moderne Elektronik sichtbar; originaler Dochtknopf bleibt Bedienorgan.
Schalter	OFF trennt alle Verbraucher. USB-C laedt bei OFF.
Flamme	Keine sichtbaren Einzelpixel aus normalem Betrachtungsabstand; warm und organisch.
Beat	PULSE reagiert reproduzierbar auf Bass/Kick und ignoriert weitgehend Sprache/Hi-Hats.
Flash	Nur Modus 4; extrem hell, warm, kurz; kein zufaelliges Boot-Flackern.
Thermik	Nach 30 min harter Musik keine unzulaessige Erwaermung von Akku, Kabeln oder Kunststoff; LED-Traeger bleibt innerhalb sicherer Grenzwerte.
Laden	Volladung in ca. 4-5 h bei geeignetem 5-V-USB-C-Netzteil; keine gleichzeitige Nutzung.
Laufzeit	Mindestens 7 h typischer gemischter Betrieb nach Vollladung.
Service	Brenner-/Elektronikeinsatz und Akku ohne Zerstoerung der Laterne erreichbar.

11. Fehlersuche

Symptom	Wahrscheinliche Ursache / Massnahme
NeoPixel flackert zufaellig	Masse/Datenpegel pruefen; 74AHCT125, 330 Ohm und Elko direkt am Jewel; Leitung kuerzen.
ESP32 resettiert beim Flash	Flash-Hochstromruecklauf von Logikmasse trennen/sternfoermig fuehren; Steckverbinder/Kabelquerschnitt pruefen.
Mikrofon reagiert auf Elektronikgeraeusch	Mic weiter vom DC/DC-Regler, eigene Massefuehrung, mechanisch entkoppeln; Kabel nicht parallel zum Flash-Pfad.
Flash zu oft	FLASH_RATIO erhoehen (z.B. 2.6), Sperrzeit verlaengern, Mic-Empfindlichkeit/Position pruefen.
Flash zu schwach bei fast leerem Akku	Bei linearem 1S-Treiber normal. Optional professionellen Buck-Boost-Konstantstromtreiber einsetzen.
Encoder springt Modi	Mechanische Entprellung/Kupplung pruefen; Polling schneller; ggf. 10-nF-Kondensatoren nach GND und Software-Debounce.
Laufzeit <7 h	PIXEL_MASTER reduzieren; RGB-Saettigung/Helligkeit begrenzen; 5-V-Regler-Wirkungsgrad und Ruhestroeme messen.

12. Quellen und Referenzdaten

Quelle	URL
Seeed Studio / XIAO ESP32-C3	https://wiki.seeedstudio.com/XIAO_ESP32C3_Getting_Started/
Reichelt XIAO ESP32-C3	https://www.reichelt.de/de/de/shop/produkt/xiao_esp32c3_wifi_bt_ohne_header-358356
Adafruit NeoPixel Jewel RGBW 3000K	https://www.adafruit.com/product/2858
Eckstein NeoPixel Jewel Preis	https://eckstein-shop.de/Adafruit-NeoPixel-Jewel-7x-5050-RGBW-LED-with-Integrated-Drivers-Warm-White-3000K
BerryBase INMP441	https://www.berrybase.de/inmp441-mems-omnidirektionales-mikrofonmodul-i2s-interface
BerryBase Bourns Encoder	https://www.berrybase.de/bauelemente/passive-bauelemente/potentiometer/drehimpulsgeber
BerryBase SN74AHCT125N	https://www.berrybase.de/sn74ahct125n-quad-level-shifter-dil-14
Pololu S13V30F5	https://www.pololu.com/product/4082
LED-Tech Cree XP-G4 3000K Star	https://www.led-tech.de/de/CREE-XP-G4-C4-3000K-auf-Star
Keeppower P2160TC Daten	https://keeppower.com/product/p2160tc-type-c-usb-21700-6000mah-li-ion-battery/
Convoy 17mm AMC7135x8 3040mA	https://convoylight.com/products/17mm-7135-8-driver-12groups-3040ma

Quelle	URL
Feuerhand B-276 Ersatzbrenner	https://feuerhand.com/products/burner-with-wick-for-baby-special-276
InLine USB-C Panelkabel 33441G	https://www.inline-info.com/InLine-USB-3.2-Gen.2-Adapterkabel-Stecker-C-auf-Einbaubuchse-C-0-20m

Preis- und Verfuegbarkeitsangaben im separaten Einkaufslisten-PDF entsprechen dem Recherchezeitpunkt 14.09.2026 und koennen sich aendern.