LUMINOUS

Dokumentation

Inhalt

Installationsanleitung

Dieser Leitfaden beschreibt die genaue Installation und Konfiguration des Luminous-Backends. Sie benötigen einen Linux-Server, in diesem Fall einen Raspberry Pi 4 B und möglicherweise weitere erforderliche Ressourcen.

Raspberry Pi

1. Beginnen Sie mit der Installation einer beliebigen Linux-Version auf Ihrem Raspberry Pi. In diesem Guide verwenden wir Raspberry Pi OS Lite 64-Bit, aber jede andere Linux-Version sollte ebenfalls funktionieren.
2. Sobald die Linux-Distribution erfolgreich installiert wurde, fahren Sie fort mit der Konfiguration des Raspberry Pi für Luminous.
3. Öffnen Sie ein Terminal oder eine SSH-Verbindung zum Raspberry Pi.
4. Aktualisieren Sie zunächst das System mit den folgenden Befehlen:

   ```
   sudo apt update
   sudo apt upgrade
   ```

5. Installieren Sie Git mit dem folgenden Befehl:

   ```
   sudo apt install git
   ```

6. Sobald die Installation abgeschlossen ist, können Sie überprüfen, ob Git erfolgreich installiert wurde, indem Sie den Befehl `git --version` ausführen. Wenn Git erfolgreich installiert wurde, wird die installierte Version angezeigt.
7. Überprüfen Sie, ob Python 3.9.2 oder eine mit OLA kompatible Version installiert ist:

   ```
   python3 --version
   ```

   Wenn Python 3.9.2 installiert ist, wird die Version angezeigt. Falls nicht, installieren Sie Python 3.9.2 mit den folgenden Befehlen:

   ```
   cd /opt
   sudo wget https://www.python.org/ftp/python/3.9.2/Python-3.9.2.tgz
   sudo tar -xzf Python-3.9.2.tgz
   cd Python-3.9.2
   ```

```
sudo ./configure --enable-optimizations
sudo make altinstall
```

(Hinweis: Der Befehl `make altinstall` wird verwendet, um die Standard-Python-Installation des Systems nicht zu überschreiben. Der Python-Interpreter heißt dann `python3.9`.)

8. Überprüfen Sie die Installation von Python 3.9 ggf. erneut:

```
# Wenn manuell installiert:
python3.9 --version
# Wenn über apt installiert oder bereits vorhanden:
python3 --version
```

(Passen Sie die Python-Befehle in den späteren Schritten ggf. auf `python3.9` an, falls Sie manuell installiert haben.)

Ihre Raspberry Pi-Umgebung ist jetzt vorbereitet, um mit der Installation von Luminous fortzufahren.

OLA

Allgemeine Informationen

Wir werden hier eine umfassende Anleitung zur Installation von OLA bereitstellen. Sollten jedoch Schwierigkeiten auftreten, verweisen wir auf die folgenden Ressourcen, die ebenfalls erklären, wie man OLA installiert:

- The Newbie Guide for OLA on Ubuntu
- OLA - Linux install

Diese Anleitungen bieten zusätzliche Informationen und können bei Problemen hilfreich sein. Vieles in diesem Text basiert jedoch auf den beiden oben genannten Anleitungen und wurde für eine einfache Installation auf dem Raspberry Pi angepasst.

Abhängigkeiten installieren

Um alle erforderlichen Abhängigkeiten für die ordnungsgemäße Funktion von OLA zu installieren, benötigen Sie einige Bibliotheken. Einige davon sind als Pakete in Linux-Distributionen verfügbar, während andere manuell heruntergeladen und kompiliert werden müssen.

Zuerst benötigen Sie mindestens die folgenden Abhängigkeiten:

- cppunit
- uuid or ossp uuid
- pkg-config
- curses
- lex (or flex) *(wir nutzen flex)*
- yacc (or bison) *(wir nutzen bison)*

- the protocol buffers library http://code.google.com/p/protobuf/ (version 2.3.0 or later)
- microhttpd ftp://ftp.gnu.org/gnu/libmicrohttpd/ (wenn Sie die Web-Benutzeroberfläche möchten). Sie benötigen mindestens Version 0.4.0 von microhttpd.
- libtool
- automake
- autoconf

Hier ist eine einfache Möglichkeit, diese Abhängigkeiten zu installieren:

```
sudo apt-get install libcppunit-dev libcppunit-1.13-0 uuid-dev pkg-config
libncurses5-dev libtool autoconf automake g++ libmicrohttpd-dev libmicrohttpd10
protobuf-compiler libprotobuf-lite10 python-protobuf libprotobuf-dev libprotoc-dev
zlib1g-dev bison flex make libftdi-dev libftdi1 libusb-1.0-0-dev liblo-dev
libavahi-client-dev python-numpy
```

oder

```
sudo apt install build-essential autoconf libtool pkg-config libcppunit-dev
libmicrohttpd-dev zlib1g-dev libftdi-dev libusb-1.0-0-dev protobuf-compiler
libprotobuf-dev python3-protobuf libprotoc-dev liblua5.3-dev
```

Die obige Methode wird empfohlen, aber wenn sie nicht funktioniert (z. B. weil keine direkte Internetverbindung besteht), können Sie die Pakete auch manuell herunterladen und installieren.

ANMERKUNG: Verwenden Sie für das manuelle Kompilieren ein geeignetes temporäres Verzeichnis, z. B. in Ihrem Home-Verzeichnis oder unter `/tmp`, anstelle des nicht standardmäßig vorhandenen "Documents"-Ordners.

1. Googeln Sie nach dem Quellpaket.
2. Laden Sie das Archiv herunter (`$FILENAME.tar.gz`).
3. Entpacken Sie den Inhalt in ein temporäres Verzeichnis (z.B. `~/ola_dependencies/build`).
4. Öffnen Sie ein Terminal.
5. Geben Sie `cd $FILEPATH` ein und drücken Sie ENTER (Beispiel: `cd ~/ola_dependencies/build/bison-3.8`).
6. Geben Sie `./configure` ein und drücken Sie ENTER. Warten Sie, bis der Vorgang abgeschlossen ist.
7. Geben Sie `make` ein und drücken Sie ENTER. Warten Sie, bis der Vorgang abgeschlossen ist.
8. Tippen Sie `make check` (optional) und drücken Sie dann ENTER. Warten Sie, bis der Vorgang abgeschlossen ist.
9. Geben Sie `sudo make install` ein und drücken Sie ENTER. Warten Sie, bis der Vorgang abgeschlossen ist.

Tun Sie dies für alle Abhängigkeiten, die nicht über `apt` installiert werden konnten.

Sobald alle Abhängigkeiten installiert sind, geben Sie `sudo ldconfig` in die Konsole ein, um die neuen Bibliotheken nutzbar zu machen.

```
sudo ldconfig
```

Dadurch sollten alle erforderlichen Abhängigkeiten für OLA installiert sein.

Installation von OLA

Prüfen Sie das Git-Repositorium mit dem folgenden Befehl:

```
git clone https://github.com/OpenLightingProject/ola.git ola
cd ola
```

Stellen Sie sicher, dass Sie die Version 0.10.9 von OLA heruntergeladen haben, da dies für die Installation von entscheidender Bedeutung sein kann. (Ggf. mit `git checkout <version>` die Version wechseln).

Wenn Sie OLA zum ersten Mal installieren, führen Sie den Befehl mit `-i` aus, um die fehlenden Dateien automatisch zu installieren:

```
autoreconf -i
```

Sie haben die Möglichkeit, zusätzliche Optionen an `./configure` zu übergeben. Verwenden Sie:

```
./configure --help
```

um alle verfügbaren Optionen anzuzeigen. Eine der beliebtesten Optionen ist `--enable-python-libs`, um das Python Client Module zu erstellen. Wenn Sie die RDM-Responder-Tests nutzen möchten, fügen Sie außerdem `--enable-rdm-tests` hinzu.

Für unsere Installation empfehlen wir die Verwendung von `--enable-python-libs` und `--enable-rdm-tests`. Wir verwenden auch `make -j N`, wobei N die Anzahl der CPU-Kerne angibt. Da der Raspberry Pi 4b über 4 Kerne verfügt, verwenden wir `make -j 4`:

```
./configure --enable-rdm-tests --enable-python-libs
make -j 4
make check
sudo make install
```

Abschließend führen Sie `sudo ldconfig` aus, um die neuen Bibliotheken nutzen zu können:

```
sudo ldconfig
```

OLA sollte nun erfolgreich installiert worden sein.

OLA Ausführen

Um OLA manuell auszuführen, geben Sie `olad` im Terminal ein.

```
olad
```

Besuchen Sie die OLA-Web-Oberfläche auf Ihrem Gerät unter folgender Adresse `http://<Ihre-Pi-IP>:9090/`. Wenn Sie die Web-Oberfläche sehen, haben Sie OLA korrekt installiert.

Einrichtung des Autostarts für OLA (ola.service)

Damit der Luminous-Backend-Server zuverlässig funktioniert und insbesondere der Autostart von Luminous gelingt (der von einem laufenden OLA-Dienst abhängt), muss der OLA-Daemon (`olad`) ebenfalls automatisch beim Systemstart gestartet werden. Dies geschieht durch die Einrichtung eines Systemd-Dienstes. Je nach Installation kann das automatisch eingerichtet worden sein. Wenn nicht, folgen Sie diesen Schritten:

1. **Überprüfen Sie den Pfad zu `olad`:**
   Führen Sie `which olad` im Terminal aus. Die Ausgabe ist normalerweise `/usr/local/bin/olad` (wenn aus Quellen kompiliert) oder `/usr/bin/olad` (wenn über Paketmanager installiert). Merken Sie sich diesen Pfad.

   ```
   which olad
   ```

2. **Erstellen Sie die Service-Datei:**
   Öffnen Sie eine neue Datei namens `ola.service` im Verzeichnis `/etc/systemd/system/` mit einem Texteditor (z. B. `nano`):

   ```
   sudo nano /etc/systemd/system/ola.service
   ```

3. **Fügen Sie folgenden Inhalt ein:**
   Passen Sie den Pfad bei `ExecStart=` an, falls Ihre Ausgabe von `which olad` abweicht.

   ```
   [Unit]
   Description=OLA Daemon (Open Lighting Architecture)
   After=network-online.target
   Wants=network-online.target

   [Service]
   # Führen Sie 'which olad' aus, um den korrekten Pfad zu finden
   ExecStart=/usr/local/bin/olad
   User=root
   Restart=on-failure
   ```

```
StandardOutput=journal
StandardError=journal

[Install]
WantedBy=multi-user.target
```

(Hinweis: Sie können Optionen zu `ExecStart=` hinzufügen, z.B. `ExecStart=/usr/local/bin/olad -l 3` für ein anderes Log-Level.)

4. **Speichern und schließen Sie die Datei** (in `nano`: `STRG+X`, dann `Y`, dann `Enter`).
5. **Laden Sie die Systemd-Konfiguration neu:**

```
sudo systemctl daemon-reload
```

6. **Aktivieren Sie den OLA-Dienst:**
   Dadurch wird er bei jedem Systemstart automatisch gestartet.

```
sudo systemctl enable ola.service
```

7. **Starten Sie den OLA-Dienst jetzt manuell** (oder starten Sie den Raspberry Pi neu):

```
sudo systemctl start ola.service
```

8. **Überprüfen Sie den Status des Dienstes:**

```
sudo systemctl status ola.service
```

   Es sollte angezeigt werden, dass der Dienst "active (running)" ist. Sie können die Logs auch mit `sudo journalctl -u ola.service` einsehen.

Mit diesen Schritten ist sichergestellt, dass der OLA-Daemon automatisch startet und für das Luminous-Backend bereitsteht.

Backend Server

Der nächste Schritt ist die Installation von Luminous auf dem Raspberry Pi. Wechseln Sie dazu in das Verzeichnis `/home/pi/`:

```
cd /home/pi/
```

Herunterladen

Laden Sie das Git-Repository herunter und wechseln Sie in das entsprechende Verzeichnis:

```
git clone https://github.com/le0ntt/Luminous.git
cd Luminous
```

Abhängigkeiten installieren

Stellen Sie sicher, dass alle notwendigen Abhängigkeiten installiert sind. Führen Sie dazu die folgenden Befehle aus:

```
cd backend
pip install -r requirements.txt
```

Starten des Backend-Servers

Starten Sie den Backend-Server mit folgendem Befehl (verwenden Sie ggf. `python3.9`, wenn Sie manuell installiert haben):

```
python3 server.py
```

Zu diesem Zeitpunkt möchten Sie nur überprüfen, dass keine Fehlermeldungen auftreten, die den Start verhindern. Der Output in der Konsole sollte so oder so ähnlich aussehen:

```
pi@raspberrypi:~/Luminous/backend $ python3 server.py
Ignored channels when checking: {1: [18, 16, 13], 2: [201, 202, 203, 204, 205,
207, 208, 209, 210, 211, 213, 214, 215, 216, 217, 219, 220, 221, 222, 223, 225,
226, 227, 228, 229, 231, 232, 233, 234, 235, 237, 238, 239, 240, 241]}
Setting up...
Setup done
/home/pi/Luminous/backend/server/led_control.py:22: RuntimeWarning: This channel
is already in use, continuing anyway.  Use GPIO.setwarnings(False) to disable
warnings.
  GPIO.setup(LED_PIN, GPIO.OUT)
Possibly the MIDI interface is not connected. unknown port 'E-MU XMidi2X2:E-MU
XMidi2X2 Midi Out 2  28:1'
No dmx_channel key for non-master channel
 * Serving Flask app 'server'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
```

```
 * Running on http://192.168.178.46:5000
Press CTRL+C to quit
```

Falls dies nicht so aussieht und gravierende Fehler auftreten, suchen Sie sich Hilfe.

Wenn der Output so oder so ähnlich aussieht, können Sie den Server mit `STRG+C` wieder stoppen und zum nächsten Schritt übergehen.

Einrichtung des Autostarts

1. **Erstellen Sie das Log-Verzeichnis** (falls es nicht bereits existiert):

   ```
   mkdir -p /home/pi/Luminous/logs
   ```

2. Erstellen Sie eine neue Service-Datei im Verzeichnis `/etc/systemd/system/`. Ein passender Name wäre `Luminous.service`. Verwenden Sie einen Editor wie `nano`:

   ```
   sudo nano /etc/systemd/system/Luminous.service
   ```

3. Öffnen Sie die Datei und fügen Sie folgenden Inhalt ein. **Achten Sie darauf, dass die Pfade korrekt sind und `python3` ggf. durch `python3.9` ersetzt wird, falls Sie manuell installiert haben**:

   ```
   [Unit]
   Description=Luminous Backend Service
   After=network-online.target ola.service
   Wants=network-online.target

   [Service]
   User=pi
   # Pfad zu Python anpassen, falls manuell installiert (z.B.
   /usr/local/bin/python3.9)
   ExecStart=/usr/bin/python3 /home/pi/Luminous/backend/server.py
   WorkingDirectory=/home/pi/Luminous/backend/
   StandardOutput=journal+tty
   StandardError=journal+tty+file:/home/pi/Luminous/logs/error.log
   # TTYPath=/dev/tty3
   Restart=always

   [Install]
   WantedBy=multi-user.target
   ```

4. Speichern Sie die Datei nach dem Bearbeiten (in `nano`: `STRG+X`, dann `Y`, dann `Enter`).

5. Laden Sie die systemd Konfiguration neu:

```
sudo systemctl daemon-reload
```

6. Aktivieren Sie den Dienst, damit er beim Booten gestartet wird:

```
sudo systemctl enable Luminous.service
```

7. Starten Sie den Dienst sofort (Optional):

```
sudo systemctl start Luminous.service
```

8. Führen Sie einen Neustart des Raspberry Pi durch, um den Autostart zu überprüfen:

```
sudo reboot
```

9. Nach dem Neustart überprüfen Sie den Status und die Logs des Dienstes, um sicherzustellen, dass alles korrekt funktioniert (und dass `olad` ebenfalls läuft):

```
sudo systemctl status Luminous.service
sudo journalctl -u Luminous.service -f
```

Durch diese Schritte richten Sie den Autostart für den Luminous-Dienst ein und stellen sicher, dass der Dienst beim Hochfahren des Systems automatisch gestartet wird.

Frontend (App) installieren

Die Installation des Frontends ist im Vergleich zum Backend deutlich einfacher. Es ist keine separate Konfiguration auf dem Client-PC notwendig.

1. Besuchen Sie die Releases-Seite des Projekts.
2. Laden Sie von der neuesten Version die `.exe`-Installationsdatei herunter (z.B. `luminous-setup-1.2.x.exe`).
3. Führen Sie die heruntergeladene Datei aus und folgen Sie den Anweisungen des Installationsassistenten.

Empfehlung: Wenn der PC hauptsächlich für die Steuerung mit Luminous verwendet wird, empfiehlt es sich, die App zum Windows-Autostart hinzuzufügen.

Admin-Handbuch

Dieser Guide beschreibt die grundlegende Verwaltung der Luminous-App für Administratoren. Die folgenden Abschnitte führen durch die häufigsten administrativen Aufgaben und bieten Hilfestellungen bei auftretenden Problemen.

Problembehebung

Die meisten ungewöhnlichen Probleme lassen sich durch einen Systemneustart lösen. Starten Sie sowohl den Raspberry Pi als auch die Luminous-App neu. Sollte das Problem weiterhin bestehen, könnte es sich um einen Fehler im Code handeln. In diesem Fall können Sie die Entwickler per E-Mail kontaktieren: pietasdarwin@gmail.com, l.hoelzel@icloud.com

Allgemeine Verwaltung

Die Verwaltungsfunktionen der Luminous-App befinden sich oben links über das Zahnrad-Icon. Besonders relevant sind die ersten beiden Menüpunkte „Einstellungen“ und „Geräte“:

Einstellungen

Im ersten Reiter befinden sich allgemeine Benutzereinstellungen:

- Sprache ändern
- Design anpassen

Administrator

Der Administratorbereich erfordert ein Passwort. Aktuell lautet das Admin-Passwort `licht`. Nach einem Datenbank-Reset ist das Passwort standardmäßig leer und kann danach geändert werden.

Sollten Sie das Passwort vergessen, ist der Zugriff nicht trivial, da das Passwort verschlüsselt in der Datenbank gespeichert ist. Als Notlösung können Sie per SSH auf den Raspberry Pi zugreifen und die Datenbank durch eine ältere Version ersetzen.

Weitere Funktionen:

- **OLA-Übersicht öffnen**: Hilfreich zum Debuggen von DMX-Signalen, da hier die aktuellen Channel-Werte angezeigt werden.
- **Server URL und Port ändern**: Besonders nützlich, wenn sich die Netzwerkkonfiguration (z.B. WLAN) geändert hat.
- **Datenbank verwalten**:
  - **Backup herunterladen**: Empfohlen vor jeder Änderung.
  - **Datenbank zurücksetzen**: Alle Daten werden gelöscht, jedoch wird auf dem Pi ein Backup erstellt.
  - **Datenbank hochladen**: Eine eigene Version hochladen und die bestehende ersetzen.

Studio-Übersicht

Anpassungen der Lampenpositionen und Icons zur Aktualisierung der Studio-Ansicht:

- Definieren Sie das **Hauptgrid** über Zeilen und Spalten.
- Weisen Sie jeder Zelle rechts in der Übersicht Lampen mit passenden Icons oder „None“ zu, falls diese leer ist.
- Greenscreen und Traversenleuchten sind im Erscheinungsbild vordefiniert. Hier ist die Zuordnung entscheidend oder „None“, um nichts anzuzeigen.
- **Custom-Lampen** außerhalb des Grids platzieren Sie frei per Pixelkoordinaten (Abstand oben/links). Sie können sie auch „flippen“, falls sie nach unten zeigen sollen.

Speicheroptionen:

- **Temporär speichern**: Einstellungen gehen beim Neustart verloren.
- **Speichern** ohne den Haken für temporäres Speichern (per Admin-Passwort): Einstellungen dauerhaft in der Datenbank sichern.

Geräteverwaltung

Im Menü „Geräte“ können Sie Lampen hinzufügen oder bestehende Geräte konfigurieren:

Grundeinstellungen je Gerät:

- **Universum**: DMX-Universum, dem das Gerät zugeordnet ist.
- **Nummer**: Kennzeichnung im Studio; bestimmt auch die Reihenfolge in der App.
- **Gerätename**
- **Typ**:
    - **RGBDim**: RGB-Lampen (z.B. Traverse)
    - **BiColor**: LEDs z.B. für Greenscreen
    - **Spot/Fill**: Gleiche Funktion, aber unterschiedliche Icons in der App
    - **HMI**: Automatisch bei Blendenöffnung aktiv; manuelles Ausschalten erforderlich
    - **Misc**: Sonstige Geräte

Kanäle und Kanalnamen:

- Jedes Gerät hat mindestens einen „Main“-Kanal (Hauptfader).
- Zusätzliche Kanäle können benannt und frei gewählt werden (Inputfelder).
- Kanäle müssen nicht lückenlos folgen. Der **Startkanal** hilft nur beim Zuordnen.

Übersicht DMX-Belegung:

- Graue Markierung: Aktuell ausgewähltes Gerät
- Farbig hinterlegt: Belegte Kanäle
- Rot markiert: Kanalüberschneidungen

Speichern und Löschen erfolgt nach Eingabe des Admin-Passworts.

Szenenverwaltung

Szenen können auf den Seiten „Studio“, „LightFX“ oder „Szenen“ gespeichert werden:

- Es werden alle aktuell eingeschalteten Lampen mit deren Kanalwerten gespeichert. Der Masterfader wird aktuell nicht berücksichtigt.
- In LightFX empfiehlt sich das Aktivieren des „Solo“-Modus, um nur die Auswahl zu speichern.

Speichern einer Szene

- Namen vergeben
- Die Option „Als Standard Szene setzen“ aktiviert die dauerhafte Speicherung nach Passworteingabe.
- Ohne diese Option wird die Szene beim Neustart gelöscht.

Szenenübersicht

- Bookmark-Icon mit Haken: dauerhaft gespeichert
- Bookmark-Icon mit Plus: Möglichkeit, mit Passwort dauerhaft zu speichern
- Mülltonnen-Icon: Szene löschen

> **Hinweis:**
> Eine direkte Bearbeitung gespeicherter Szenen ist nicht möglich. Es empfiehlt sich folgendes Vorgehen:
>
> - Szene mit "Solo" aktivieren
> - Änderungen an der Lichtstimmung vornehmen
> - Als neue Szene speichern
> - Alte Szene löschen

Entwickler-Setup

Dieser Leitfaden beschreibt die grundlegenden Schritte zur Einrichtung einer lokalen Entwicklungsumgebung für das Luminous-Projekt. Er richtet sich an Entwickler, die Änderungen am Code vornehmen oder zur Weiterentwicklung beitragen möchten.

Plattform & Empfehlungen

Die Entwicklung ist grundsätzlich auf Windows, macOS und Linux möglich. Für eine nahtlose Integration und das Testen mit dem OLA-Daemon (Open Lighting Architecture), der für die DMX-Ansteuerung benötigt wird, wird eine Unix-basierte Umgebung (Linux oder macOS) empfohlen. Für die reine Entwicklung der Frontend- oder Backend-Logik ohne direkte DMX-Ausgabe ist dies jedoch nicht zwingend erforderlich.

Als Entwicklungsumgebung wird Visual Studio Code (VS Code) empfohlen.

Voraussetzungen

Stellen Sie sicher, dass die folgenden Werkzeuge auf Ihrem System installiert sind:

- **Node.js:** Version 18 oder höher.
- **Python:** Version 3.9 oder höher.
- **Git:** Zur Versionsverwaltung.

Einrichtung der Umgebung

1. **Repository Forken:**
   Erstellen Sie zunächst einen persönlichen Fork des offiziellen Luminous-Repositories auf GitHub: https://github.com/LE0Ntt/Luminous/

2. **Repository Klonen:**
   Klonen Sie Ihren erstellten Fork auf Ihr lokales System. Öffnen Sie ein Terminal oder eine Git-Bash und führen Sie aus:

   ```
   git clone <URL-Ihres-Forks> Luminous-Dev
   cd Luminous-Dev
   ```

3. **Projekt in VS Code öffnen:**
   Öffnen Sie den geklonten Ordner (`Luminous-Dev`) in Visual Studio Code.

4. **Backend Abhängigkeiten installieren:**
   Navigieren Sie im integrierten Terminal von VS Code (oder einem separaten Terminal) in das `backend`-Verzeichnis. Installieren Sie die erforderlichen Python-Pakete:

   ```
   cd backend
   pip install -r requirements.txt
   ```

```
# Oder verwenden Sie 'python3 -m pip install -r requirements.txt', falls
'pip' nicht auf Python 3 verweist
```

5. **Frontend Abhängigkeiten installieren:**
   Wechseln Sie im Terminal in das Frontend-Verzeichnis `electron-vite-project`. Installieren Sie die Node.js-Abhängigkeiten:

```
cd ../electron-vite-project
# Alternativ von Hauptordner: cd electron-vite-project
npm install
```

Anwendung starten (Entwicklungsmodus)

Um die Anwendung lokal für Entwicklungszwecke zu starten, benötigen Sie in der Regel zwei laufende Prozesse:

1. **Backend starten:**
   Öffnen Sie ein Terminal im `backend`-Verzeichnis und starten Sie den Python-Server:

```
# Im Verzeichnis 'backend'
python server.py
# Oder 'python3 server.py'
```

   Das Backend läuft nun und wartet auf Verbindungen (standardmäßig oft auf Port 5000).

2. **Frontend starten:**
   Öffnen Sie ein *zweites* Terminal im `electron-vite-project`-Verzeichnis und starten Sie den Frontend-Entwicklungsserver mit Hot-Reloading:

```
# Im Verzeichnis 'electron-vite-project'
npm run dev
```

   Die Luminous Electron-Anwendung sollte nun starten und sich automatisch mit dem lokal laufenden Backend verbinden. Änderungen am Frontend-Code werden in der Regel live übernommen.

Anwendung bauen (Produktion)

Um eine verteilbare Version der Electron-Anwendung zu erstellen (z.B. für Windows, macOS oder Linux), führen Sie den Build-Befehl im Frontend-Verzeichnis aus:

```
# Im Verzeichnis 'electron-vite-project'
npm run build
```

Die gebauten Installationsdateien finden Sie anschließend im dafür vorgesehenen Ausgabeordner `release` innerhalb von `electron-vite-project`.

Zusätzlich sollte die Web-Version der Anwendung erstellt werden, um die Benutzeroberfläche auch über einen Webbrowser zugänglich zu machen:

```
npm run build:web
```

Die dabei generierten Dateien werden automatisch in den `static`-Ordner des `backend`-Verzeichnisses kopiert. Der Python-Server liefert sie dann über den definierten Port aus (z. B. `http://192.168.2.119:5000/`).
Für den Zugriff über andere Geräte im lokalen Netzwerk muss die IP-Adresse des Raspberry Pi bzw. Hosts verwendet werden.

> Diese Schritte sollten eine grundlegende Entwicklungsumgebung für Luminous bereitstellen.

Kontakt

Dies schließt die vorliegende Dokumentation ab. Wir hoffen, dass die bereitgestellten Informationen Ihnen bei der Installation, der täglichen Verwaltung und dem initialen Entwickler-Setup der Luminous-Software von Nutzen sind.

Für weiterführende Fragen, bei Problemen, Anregungen oder Feedback stehen Ihnen die Entwickler jederzeit zur Verfügung. Bitte zögern Sie nicht, uns zu kontaktieren:

pietasdarwin@gmail.com, l.hoelzel@icloud.com

Ihr Luminous Entwicklungsteam