

# Technische Anleitung – README

## Selbstlernseinheit 12-Kanal-EKG

### Zusammenfassung

Dieses README deckt die Inbetriebnahme der am Medizinisch Interprofessionellen Trainingszentrum des Universitätsklinikum Dresden entwickelten EKG-Selbstlernseinheit für Studierende. Es beschreibt materielle und technische Voraussetzungen, ein empfohlenes räumliches Setup, Konfiguration und die Inbetriebnahme. Für substanzelle Teile dieses Guides wird Unterstützung der Haus-IT benötigt, das Guide ist geschrieben für Ubuntu Linux.

## 1 Materielle Voraussetzungen 📦

Zur idealen Verwendung des Projektes wird benötigt

- Eine medizinische Übungspuppe in Lebensgröße
- Ein EKG-Gerät mit 10 Steckelektroden + Klebepads
- Zwei USB-Kameras mit hoher Auflösung
- Ein PC zum Ausführen der Projektsoftware
- Ein Drucker, Klebeband und ein Klebestift

### 1.1 USB-Kameras

Für stabile Anwendung sind Kameras mit hoher Auflösung, 4K UHD oder höher, empfohlen. Das Originalsetup basiert auf zwei „HP 960 4K“. Das Verwenden verschiedener Modelle ist meist möglich, solange die Kameras mit der gleichen Auflösung aufnehmen.

### 1.2 PC

Dieser Teil sollte mit der IT-Abteilung besprochen werden 🏢

Der PC benötigt mind. zwei freie USB-Ports (USB 2.0+). Das Projekt wurde mit Zielplattform Ubuntu Linux (Version 24.04.1 LTS) entwickelt.

Auf dem PC muss die Umgebungsverwaltung Conda installiert sein. Aus Rechts- und Lizenzgründen sollte die über das Community-Projekt [Conda-Forge](#) bereitgestellte Variante („miniconda“) genutzt werden. Das Projekt wurde mit Conda-Forge Version 24.9.2 entwickelt, sollte aber auch mit anderen Versionen funktionieren. Sollte miniconda direkt in die homedirectory des Benutzeraccounts installiert werden, können der bereitgestellte Desktop-shortcut und das dazugehörige shell script direkt verwendet werden.

Nachdem conda installiert wurde, kann es im Terminal mit dem Befehl `conda` verwendet werden. Conda stellt isolierte Laufzeitumgebungen zur Verfügung und muss noch über das Terminal konfiguriert werden:

1. Neue Umgebung erstellen: `conda create -n hybparc python=3.9`

## 2. Neue Umgebung Aktivieren: `conda activate hybparc`

Nun ist das offene Terminal auf die neue Umgebung gesetzt. Es müssen zuerst „Pip“ (Python Paketmanager) und danach via Pip noch Python Pakete installiert werden. Sollte im Terminal nachgefragt werden, ob zusätzliche Abhängigkeiten (dependencies) mitinstalliert werden sollen, sollte dies mit `y` bestätigt werden.

1. Pip installieren: `conda install pip`
2. OpenCV installieren: `pip install opencv-contrib-python`
3. PyQt6 installieren: `pip install pyqt6`

Es sind alle dependencies des Projektes installiert.

## 2 Referenzmarker

Zur Nutzung des Projekts werden sogenannte Arucomarker benötigt. Diese müssen auf Papier ausgedruckt werden (Empfehlung mind. 100g/m<sup>2</sup>, vollweiß) und an der Übungspuppe bzw. an den Elektroden befestigt werden. Den Projektdaten liegt hierzu ein PDF bei, dieses sollte mit Skalierung 100% ausgedruckt werden um die vorgesehenen Maße einzuhalten.

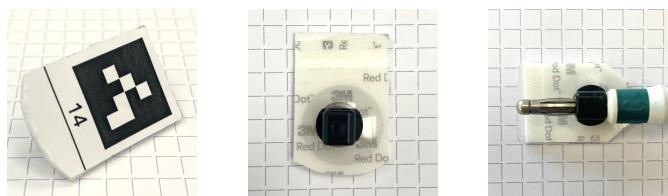
Die mitgelieferte Konfiguration deckt ein 12-Kanal EKG ab. Es werden also 15 Marker (10 Elektroden + 5 Ausrichtungsmarker) benötigt.

### 2.1 Elektrodenmarker

**TODO rework into step by step graphics**

1. Marker ausdrucken und anhand der Trennlinien ausschneiden
2. Die Marker links, rechts und oben trimmen", sodass auf den drei Seiten ca. 2mm weiße Umrundung übrigbleibt.
3. Jeden Marker von **hinten** mit einem Streifen Klebeband bekleben (Klebeseite zum Marker hin)
4. Den Marker Kantenparallel auf ein EKG-Klebepad kleben, sodass der Mittelpunkt des Markers der oberen Kante des Befestigungsclips des Pads ca. 8mm übersteht.
5. Nun den unteren Teil des Markers zurechtstutzen.
6. Den Marker mithilfe des Pad-Clips an die Elektrode anstecken, und so weit nach hinten schieben, dass bei Benutzung des Setups Platz für das eigentliche Pad ist.

Ideales Endergebnis:



## 2.2 Ausrichtungsmarker

Alle Ausrichtungsmarker müssen flach auf der Puppe angebracht werden. Hier stehen coole informative Details von Eva und darunter ist ein Beispelfoto.

## 2.3 Positionierung

Dieser Teil ist relevant, wenn die mitgelieferten Lagedaten verwendet werden sollen 📦

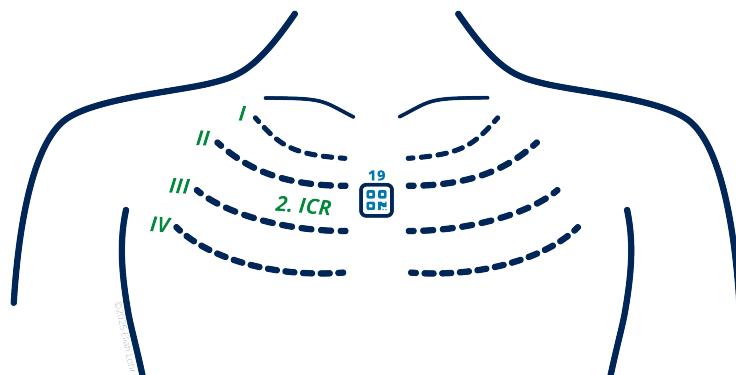
Die Marker müssen ihrer Nummer nach an den zugehörigen Elektroden angebracht werden:

Nr.	11	12	13	14	15	16	21	31	41	51
Name	V1	V2	V3	V4	V5	V6	Rot	Gelb	Schw.	Grün

Um die Position der Elektroden zu bestimmen, müssen die Ausrichtungsmarker auf der Puppe angebracht werden. Diese sind entsprechend zugeordnet:

Nr.	19	29	39	49	59
Name	Brust	Arm R	Arm L	Fuß R	Fuß L

Der Brust-Ausrichtungsmarker (Nummer 19) sollte dabei mit der oberen Kante auf Höhe der zweiten echten Rippe angebracht werden, sodass der Marker selbst innerhalb des zweiten ICR liegt.



Die Arm marker

## 3 Projektaufstellung, Konfiguration ✨

Dieser Teil sollte mit der IT-Abteilung besprochen werden 🧑

Es ist empfohlen, einen vorgefertigten Release von [Twilio?](#) [Github?](#) herunterzuladen. Bei Bedarf steht der aktuelle Sourcecode auf [GitHub](#) verfügbar. Falls der in 1.2 angebotene Desktop-shortcut und sein shell-Script verwendet werden sollen, muss das Projekt in der homedirectory im Ordner repos installiert werden (Gesamtpfad `~repos/hybpars_aruco/`).

Sollten die verwendeten Kameras die **exakt** gleiche Auflösung (3840x2160) wie die des Original-setups haben, und MJPEG unterstützen, sollte™ das Projekt (unter Ubuntu Linux) nun ohne Probleme funktionieren.

### 3.1 Kamera-IDs

Potentiell müssen die interface IDs angepasst werden, diese können mit den v4l2 command line tools (müssen potenziell nachinstalliert werden) ausgelesen werden: `v4l2-ctl -list-devices`. Die Konfigurations-Reihenfolge"der Kameras (Unterkörper vs. Oberkörper) spielt keine Rolle.

### 3.2 Andere Auflösung

Bei anderen Auflösungen müssen im Code unter `/hybparc_aruco/main.py` fast zu Beginn der `__init__`-Funktion die entsprechenden Parameter angepasst werden:

```

32
33
34
35
36
37
38
39
40
41
42
43
# Setup and warm up cameras
#! Adjustments are specific to the HP 960 4K in our physical setup
self.stream0 = cv.VideoCapture(index=0, apiPreference=cv.CAP_ANY)
self.stream0.set(cv.CAP_PROP_FOURCC, cv.VideoWriter.fourcc('M', 'J', 'P', 'G'))
self.stream0.set(cv.CAP_PROP_FRAME_WIDTH, 3840) ←
self.stream0.set(cv.CAP_PROP_FRAME_HEIGHT, 2160) ←
self.stream1 = cv.VideoCapture(index=4, apiPreference=cv.CAP_ANY)
self.stream1.set(cv.CAP_PROP_FOURCC, cv.VideoWriter.fourcc('M', 'J', 'P', 'G'))
self.stream1.set(cv.CAP_PROP_FRAME_WIDTH, 3840) ←
self.stream1.set(cv.CAP_PROP_FRAME_HEIGHT, 2160) ←

```

### 3.3 Kein MJPEG 😞

Das Projekt wurde ausschließlich mit MJPEG getestet. Bei unbedingtem Bedarf kann jedoch probiert werden, das Format umzustellen. Die unterstützten Formate einer Kamera bzw. des Interfaces mit der Nr. X können via `v4l2-ctl -d /dev/videoX -list-formats-ext` ausgelesen werden. Der entsprechende Codec muss im [FourCC Format](#) unter `/hybparc_aruco/main.py` fast zu Beginn der `__init__`-Funktion angepasst werden:

```

32
33
34
35
36
37
38
39
40
41
42
43
# Setup and warm up cameras
#! Adjustments are specific to the HP 960 4K in our physical setup
self.stream0 = cv.VideoCapture(index=0, apiPreference=cv.CAP_ANY)
self.stream0.set(cv.CAP_PROP_FOURCC, cv.VideoWriter.fourcc('M', 'J', 'P', 'G')) ←
self.stream0.set(cv.CAP_PROP_FRAME_WIDTH, 3840) ←
self.stream0.set(cv.CAP_PROP_FRAME_HEIGHT, 2160) ←
self.stream1 = cv.VideoCapture(index=4, apiPreference=cv.CAP_ANY)
self.stream1.set(cv.CAP_PROP_FOURCC, cv.VideoWriter.fourcc('M', 'J', 'P', 'G')) ←
self.stream1.set(cv.CAP_PROP_FRAME_WIDTH, 3840) ←
self.stream1.set(cv.CAP_PROP_FRAME_HEIGHT, 2160) ←

```

### 3.4 Desktop shortcut ✨

Um Nutzenden einfachen Zugang zu ermöglichen, empfiehlt sich die Einrichtung eines Desktop-shortcuts. Da das Projekt in einem Conda-environment läuft sind dazu mehrere Schritte empfohlen. Empfohlenerweise liegt auf dem Desktop eine Verknüpfung `starte-ekg.desktop` mit dem Inhalt

```
[Desktop Entry]
Name=Starte EKG
```

```
Comment=MITZ Selbstlernerheinheit
Exec=/home/selbstlern/repos/starte-ekg.sh
Terminal=false
Type=Application
```

(das Projekt liegt hier in der homedirectory des Nutzers „selbstlern“, im Ordner /repos/)  
Die durch den shortcut aufgerufene starte-ekg.sh enthält lediglich

```
cd ~/repos/hybparc_aruco
~/miniforge3/envs/hybparc/bin/python3 main.py
```

wobei Zeile 1 den Ausführungsort auf den Projektordner setzt, und Zeile 2 zum ausführen der main.py den direkten Pfad der python-executable nutzt, die in das hybparc conda-environment installiert ist. So kann erreicht werden, dass Nutzende ausschließlich das user interface sehen.

Die beiden Zeilen können *nicht* vereinigt werden! Der change directory (cd) Befehl ist nötig um das Projekt in seinem Ordner auszuführen, sonst zerbrechen Abhängigkeiten.

### 3.5 Auswertung konfigurieren

to be filled

## 4 🚨 Troubleshooting 🚨

Um das Projekt über das Terminal zu starten, oder um Änderungen am conda environment vorzunehmen, muss das environment bei jedem Terminal neustart wieder aktiviert werden (conda activate hybparc).

**OpenCV Package Type** Das Guide empfiehlt opencv-contrib-python zu installieren. Dies ist eine Ubuntu Linux spezifische Empfehlung. Sollte das Projekt unter MacOS und Windows verwendet werden empfiehlt sich das Package opencv-python.

**Kontrollbilder** Zur Kontrolle können die für die Auswertung genutzten Bilder abgespeichert werden. Hierfür muss im Projektordner einfach ein Ordner „results“ erstellt werden.

⚠ Nach Kontrolle Ordner löschen, die Bilder sind unkomprimiert und damit sehr groß! ⚡

## 5 Known Issues 😞

**Crash noch vor der Auswertung** Leider sind die Zeiten mit denen Kameras für die Software verfügbar sind abhängig vom Betriebssystem, (gefühlt) der Raumtemperatur(?), dem PC und dessen Tagesstimmung usw. Es kann also vorkommen, dass die Software abstürzt, sollte „zu schnell“ auf auswerten gedrückt werden. Hier muss die Software einfach neugestartet werden und vor dem Auswerten 15-30s gewartet werden. Sobald einmal eine Auswertung vorgenommen wurde, sind die Kameras an die Software gebunden und solche crashes treten nicht mehr auf. (Preventative fix austehend)

**Langsamer Autofokus** Leider sind die Auswertungen davon abhängig, dass der Autofokus der Kameras gut funktioniert. Hat eine Kamera einen schlechten Tag, so kann es sein, dass ein zweites Mal ausgewertet werden muss.



Stand 21. März 2025  
Gefördert von:



**Stiftung  
Innovation in der  
Hochschullehre**

„Technische Anleitung - Readme“ von Elijah Lohr ist lizenziert unter CC BY SA 4.0  
(<https://creativecommons.org/licenses/by-sa/4.0/>). 

