Welcome to the **Cinemate** documentation.

Overview

TBA

Installing CinePi and Cinemate

This guide walks you through installing the `cinepi-raw` fork and the Cinemate UI on a fresh Bookworm installation. Lite version of Bookworm also works.

Dependencies

Node Version Manager

```
wget -qO- https://raw.githubusercontent.com/nvm-sh/nvm/v0.40.3/install.sh |
bash
source ~/.bashrc
nvm install --lts
```

cpp-mjpeg-streamer cinemate fork

```
sudo apt install -y libspdlog-dev libjsoncpp-dev
cd /home/pi
git clone https://github.com/Tiramisioux/cpp-mjpeg-streamer.git --branch
cinemate
cd cpp-mjpeg-streamer && mkdir build && cd build
cmake .. && make
make install-here
```

cinepi-raw cinemate fork

Cinemate depends on a custom branch of [cinepi-raw](https://github.com/Tiramisioux/cinepi-raw/tree/rpicam-apps_1.7_custom_encoder, created by Csaba Nagy.

```
git clone https://github.com/Tiramisioux/cinepi-raw.git --branch rpicam-
apps_1.7_custom_encoder
cd /home/pi/cinepi-raw
sudo rm -rf build (if you have a previous build)
export PKG_CONFIG_PATH=/home/pi/cpp-mjpeg-streamer/build:$PKG_CONFIG_PATH
sudo meson setup build
sudo ninja -C build
meson install -C build
```

> Join the CinePi Discord here!

CineMate `settings.json` User Guide

This file controls how the camera behaves and how your buttons, switches and displays are mapped. It lives in `~/cinemate/src/settings.json` on the Raspberry Pi. You can edit it with any text editor; the settings take effect the next time you start CineMate.

The configuration is structured as JSON. Each top-level key describes a feature area of the system. Below is a tour of every section and what the options do.

1. `welcome_message` and `welcome_image`

Text or image displayed briefly when Cinemate starts.

```
"welcome_image": null
"welcome_message": "THIS IS A COOL MACHINE",
```

Set `welcome_image` to the path of a bitmap file to show a logo instead of text.

Example path: `/home/pi/welcome_image.bmp`.

If `welcome image` path is set, this will override the text message.

2. `system`

```
"system": {
  "wifi_hotspot": {
    "name": "CinePi",
    "password": "11111111",
    "enabled": false
  }
}
```

- **name** – the Wi-Fi network name (SSID) broadcast by the Pi when hotspot mode is enabled.
- **password** – password for joining the hotspot.
- **enabled** – set to `true` to start the hotspot automatically on boot. If set to `false`, CineMate will still start its web ui but stream it on whatever network the Pi is connected to.

CinePi-raw CLI User Guide

This guide explains how to start **CinePi-raw** from the command line. The tool is a fork of the `rpicam-apps` project and allows capturing CinemaDNG files using Raspberry Pi cameras. The examples below assume you have installed the software and its dependencies as described in the repository README.

1. Checking available options

Before running the program you can view all command-line flags with:

```
cinepi-raw -h
```

This prints a long list of options supported by the application. It includes the standard parameters from `rpicam-apps` (such as resolution and exposure settings) plus additional flags specific to the CinePi project. If you just want to confirm that your build works, you can also display the version number using:

```
cinepi-raw --version
```

2. Camera modes

CinePi-raw uses **Libcamera** to talk to your Raspberry Pi camera module. Each sensor supports one or more *modes*, which define the resolution and bit depth of the RAW images that the sensor can produce. A mode is written as:

```
<width>:<height>:<bit-depth>[:<packing>]
```

- `width` and `height` select the active pixel area of the sensor.
- `bit-depth` is usually 12 or 16 bits per pixel.
- `packing` can be `P` for packed or `U` for unpacked data.

The mode must match the sensor you are using. For example, an IMX477 camera can run at `4056:3040:12` (full sensor) or at smaller cropped resolutions. When specifying a mode you typically also set the output `--width` and `--height` which control the size of the image written to disk. These can be equal to the mode values or smaller when scaling is applied.

7.3 CineMate "Pseudo-CLI"

CineMate doesn't use a real shell parser. Instead, a background thread reads simple text commands from SSH or the serial port and calls the corresponding controller methods.

Available Commands

Command	Input type	Example	Discussion
`rec` / `stop`	none	`rec`	Toggle recording on or off
`set iso <value>`	int	`set iso 800`	Set ISO to nearest allowed step
`inc iso` / `dec iso`	none	`inc iso`	Step ISO up or down
`set shutter a <angle>`	float	`set shutter a 180`	Set actual shutter angle (snaps unless free/sync)
`inc shutter a` / `dec shutter a`	none	`inc shutter a`	Cycle through shutter angles
`set shutter a nom <angle>`	float	`set shutter a nom 180`	Set nominal shutter angle for motion blur
`inc shutter a nom` / `dec shutter a nom`	none	`inc shutter a nom`	Step the nominal shutter angle
`set fps <value>`	float	`set fps 24`	Change frame rate (snaps unless free)
`inc fps` / `dec fps`	none	`inc fps`	Step through FPS list
`set wb [<Kelvin>]`	int or none	`set wb 5600`	Set white balance or cycle presets

How Cinemate launches cinepi-raw

`cinemate/src/module/cinepi_multi.py` starts one `cinepi-raw` process per connected camera. It takes user settings from `sensor_detect.py` and `settings.json` to influence the command-line flags passed to `cinepi-raw`.

Here it how it works:

1. Detecting Cameras

When CineMate starts, `CinePiManager` runs `cinepi-raw --list-cameras`. Each line of output describes a connected sensor. The manager parses this output and stores basic information about every camera:

- **index** – numeric index passed to `--camera`
- **model** – sensor model name (e.g. `imx477`)
- **mono** – whether the camera is monochrome

This information is kept in the `CameraInfo` class and written to Redis under the `cam_info` keys so that other modules know which sensors are present.

2. Loading Resolution Data

`cinepi_multi.py` relies on `sensor_detect.py` to look up valid resolutions and frame rates for each sensor. The mapping lives in `src/module/sensor_detect.py` and is organised like this:

```
sensor_resolutions = {
    'imx477': {
        0: {'width': 2028, 'height': 1080, 'bit_depth': 12, 'fps_max': 50},
        1: {'width': 2028, 'height': 1520, 'bit_depth': 12, 'fps_max': 40},
        # ...
    },
    'imx585_mono': {
        0: {'width': 1928, 'height': 1090, 'bit_depth': 12, 'fps_max': 87},
    }
}
```

If you add support for a new sensor or want to tweak maximum frame rates, modify this dictionary. `cinepi_multi` calls `get_resolution_info()` to fetch the entry for the detected model and sensor mode (stored in Redis as `sensor_mode`).

CinePi Controller Methods

CineMate exposes most of its runtime features through the `CinePiController` class in `src/module/cinepi_controller.py`. Buttons, the pseudo-CLI and the web UI all call these methods. Below is an overview of the most useful ones and what they do.

Recording

- `rec()` – Toggle recording on or off depending on the current state.
- `start_recording()` – Begin recording if storage is mounted and space is available.
- `stop_recording()` – Stop the current recording.

Exposure settings

These methods adjust ISO, shutter angle and frame rate. Increment/decrement helpers step through the arrays defined in `settings.json` unless free mode is active.

- `set_iso(value)` – Set ISO to a specific value.
- `inc_iso()` / `dec_iso()` – Step ISO up or down.
- `set_shutter_a(value)` – Set the *actual* shutter angle. In normal mode the value snaps to the nearest valid angle.
- `inc_shutter_a()` / `dec_shutter_a()` – Cycle through shutter angles.
- `set_shutter_a_nom(value)` – Set the *nominal* shutter angle used for motion-blur calculations.
- `inc_shutter_a_nom()` / `dec_shutter_a_nom()` – Step the nominal shutter angle.
- `set_fps(value)` – Apply a new frame rate while respecting locks and sync mode.
- `inc_fps()` / `dec_fps()` – Step through the configured FPS list.

White balance

- `set_wb(kelvin=None, direction='next')` – Set white balance to a specific Kelvin temperature or cycle through presets if no value is given.
- `inc_wb()` / `dec_wb()` – Move to the next or previous white balance preset.

Redis API quick start

Cinemate talks to the cinepi-raw recorder through a local Redis server. Parameters such as ISO, FPS or the recording state are stored as simple keys. Two pub-sub channels (`cp_controls` and `cp_stats`) carry notifications and status updates.

Here is an overview of how the pieces fit together and how you can experiment with them using `redis-cli` or your own Python scripts.

How CineMate and cinepi-raw interact

Cinepi-raw exposes an API over Redis. Cinemate acts as the user interface. When you change a value in Cinemate (for example by pressing a button or turning a rotary encoder) it writes the new value to Redis and publishes the key name on the `cp_controls` channel. `cinepi-raw` subscribes to this channel and reacts to changes.

Conversely, cinepi-raw periodically publishes camera statistics on the `cp_stats` channel. Cinemate listens and updates the on-screen GUI.

The `cp_controls` channel

CineMate writes values and immediately publishes the key name. The recorder only reacts when it receives that publish event.

Any key may be sent this way. For example, to adjust the preview zoom:

```
# Set preview zoom level

redis-cli SET zoom 1.5
redis-cli PUBLISH cp_controls zoom
```

> Execpt for the recording trigger **is_recording**. Here, the Cinemate cinepi-raw fork *immediately* starts and stops recording upon edge detection (the variable changes from 0 to 1 or vice versa). The reason for this exception has to do with how the cinepi-raw fork handles recording with multiple cameras

```
# Start recording
redis-cli SET is_recording 1                    # triggers 0 → 1 edge

# Stop recording
redis-cli SET is_recording 0                    # triggers 1 → 0 edge
```

Redis key reference

This page lists all Redis keys used by Cinemate and cinepi-raw. Values are simple strings so you can read or write them with `redis-cli`.

Each entry explains which component normally writes the key and what happens when you change it manually.

Key	Written by	Description	Safe to change manually?
anamorphic_factor	Cinemate	Preview squeeze for anamorphic lenses	Yes (publish key to apply)
iso	Cinemate → cinepi-raw	Sensor gain in ISO	Yes
shutter_a	Cinemate → cinepi-raw	Actual shutter angle in degrees	Yes
shutter_angle_nom	Cinemate	Desired shutter angle before sync/free adjustments	Yes
shutter_a_sync_mode	Cinemate	Keep exposure constant when changing FPS	Yes
fps	Cinemate → cinepi-raw	Target frames per second	Yes
sensor_mode	Cinemate → cinepi-raw startup	Active sensor resolution/mode	Yes (causes pipeline restart)
wb	Cinemate → cinepi-raw	White-balance temperature (Kelvin)	Yes