Cinemate

User & Builder Guide

None

None

Table of contents

Welcome to the **Cinemate** documentation.

1. Getting started

1.1 Hardware requirements

- Raspberry Pi 5
- Official HQ or Global Shutter camera
- HDMI monitor or a phone/tablet for monitoring

1.2 Installation

1. **Burn the Cinemate image** to an SD card (8 GB or larger).
2. **Connect the Pi and the camera sensor board.**

   Important: Ensure the Pi is powered off before attaching the camera ribbon cable. Hot-swapping the cable is not advised.
3. **Boot the Pi.** CineMate should start automatically.
4. **Previewing the image**
5. Plug in an HDMI monitor **or** connect your phone/tablet to the Wi-Fi network `CinePi` (password `11111111`).
6. Open a browser and go to `cinepi.local:5000` to see the interface. A clean video feed without the GUI is available at `cinepi.local:8000/stream`.
7. **Recording footage**
8. Attach a high-speed drive: an **SSD** (Samsung T7 recommended), an **NVMe drive**, or the **CFE Hat**.
9. Format the drive as `ext4` and give it the label `RAW`.
10. Connect a button between **GPIO5** and **GND** (or briefly short these pins with a paper clip). When using the phone preview, you can also start/stop recording by tapping the preview.

    That's it—your bare-bones CineMate build is ready.

    Remember to power everything down before disconnecting hardware!

2. Manual installation

Here is how you can manually install libcamera, cinepi-raw, cinemate and accompanying software on the Raspberry Pi.

Although Raspberry Pi 4 (and even 3) has been known to work with the stack below, a Raspopberry Pi 5B or Compute Module 5 is recommended. Also note that for high speed USB 3, a Raspberry Pi 4 or 5 is needed.

This guide assumes fresh Raspbery Pi Bookworm installation running kernel 6.12.20+.

If you run Raspberry Pi OS Lite, begin by installing the following packages:

```
sudo apt install -y python-pip git python3-jinja2
```

2.1 libcamera 1.7.0 raspberry pi fork

```
git clone https://github.com/raspberrypi/libcamera && \
sudo find ~/libcamera -type f \( -name '*.py' -o -name '*.sh' \) -exec chmod +x {} \; && \
cd libcamera && \
sudo meson setup build --buildtype=release \
  -Dpipelines=rpi/vc4,rpi/pisp \
  -Dipas=rpi/vc4,rpi/pisp \
  -Dv4l2=true \
  -Dgstreamer=enabled \
  -Dtest=false \
  -Dlc-compliance=disabled \
  -Dcam=disabled \
  -Dqcam=disabled \
  -Ddocumentation=disabled \
  -Dpycamera=enabled && \
sudo ninja -C build install && \
cd
```

```
cd ~/libcamera/utils && sudo chmod +x *.py *.sh && sudo chmod +x ~/libcamera/src/ipa/ipa-sign.sh && cd ~/libcamera && sudo ninja -C build install
```

```
sudo apt-get install --reinstall libtiff5-dev && sudo ln -sf $(find /usr/lib -name "libtiff.so" | head -n 1) /usr/lib/aarch64-linux-gnu/libtiff.so.5 &&
export LD_LIBRARY_PATH=/usr/lib/aarch64-linux-gnu:$LD_LIBRARY_PATH && sudo ldconfig
```

```
sudo apt install -y python3-pip git python3-jinja2 libboost-dev libgnutls28-dev openssl pybind11-dev qtbase5-dev libqt5core5a meson cmake python3-yaml
python3-ply libglib2.0-dev libgstreamer-plugins-base1.0-dev libgstreamer1.0-dev libavdevice59
```

2.2 nvm

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

Cinemate uses nvm (Node Version Manager) for its web ui. If you plan to use cionepi-raw only, nvm is not needed.

2.3 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
```

Cinemate uses a custom fork of cpp-mjpeg-streamer. If you plan to use only cinepi-raw, you can use the original app found at https://github.com/nadjieb/cpp-mjpeg-streamer

2.4 cinepi-raw

2.4.1 Dependencies

```
sudo apt install -y cmake libepoxy-dev libavdevice-dev build-essential cmake libboost-program-options-dev libdrm-dev libexif-dev libcamera-dev libjpeg-dev
libtiff5-dev libpng-dev redis-server libhiredis-dev libasound2-dev libjsoncpp-dev libpng-dev meson ninja-build libavcodec-dev libavdevice-dev libavformat-dev
libswresample-dev && sudo apt-get install libjsoncpp-dev && cd ~ && git clone https://github.com/sewenew/redis-plus-plus.git && cd redis-plus-plus && mkdir
build && cd build && cmake .. && make && sudo make install && cd ~
```

2.4.2 cinepi-raw cinemate fork

```
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
sudo meson install -C build
```

Cinemate depends on a custom branch of cinepi-raw created by Csaba Nagy. If you plan to use the original version you can find it adapted for rpicam-apps 0.7 here: https://github.com/Tiramisioux/cinepi-raw/tree/rpicam-apps_1.7

Join the CinePi Discord here!

2.4.3 imx585 driver cinemate fork

```
sudo apt install linux-headers dkms git
```

```
git clone https://github.com/Tiramisioux/imx585-v4l2-driver.git
cd imx585-v4l2-driver/
./setup.sh
```

The imx585 is written by Will Whang. For original drivers and startup guides, visit https://github.com/will127534/StarlightEye

2.4.4 imx283 driver cinemate fork

```
sudo apt install linux-headers dkms git
```

```
git clone https://github.com/Tiramisioux/imx283-v4l2-driver.git
cd imx283-v4l2-driver/
./setup.sh
```

The imx283 is written by Will Whang. For original drivers and startup guides, visit https://github.com/will127534/imx283-v4l2-driver

2.5 Enabling I²C

```
sudo apt update && apt upgrade
sudo raspi-config nonint do_i2c 0
```

Enabling I2C is needed for using the camera modules.

2.6 Hostname

```
sudo hostnamectl set-hostname cinepi
```

You will find the pi as `cinepi.local` on the local network, or at the hotspot Cinemate creates

2.7 Add camera modules to config.txt

```
sudo nano /boot/firmware/config.txt
```

Paste this into your file, and uncomment the sensor you are using.

Also specify which physical camera port you have connected your sensor to.

```
# Raspberry Pi HQ camera
#camera_auto_detect=1
#dtoverlay=imx477,cam0

# Raspberry Pi GS camera
#camera_auto_detect=1
#dtoverlay=imx296,cam0

# OneInchEye
#camera_auto_detect=0
#dtoverlay=imx283,cam0

# StarlightEye
camera_auto_detect=0
dtoverlay=imx585,cam0

# StarlightEye Mono
camera_auto_detect=0
#dtoverlay=imx585,cam1,mono

# CFE Hat (pi 5 only)
dtparam=pciex1
dtparam=pciex1_gen=3

dtoverlay=disable-bt
```

And at the very bottom of the file:

```
[all]
avoid_warnings=1
disable_splash=1
```

2.8 Add the IMX585 tuning file (optional)

```
curl -L -o /home/pi/libcamera/src/ipa/rpi/pisp/data/imx585.json \
  https://raw.githubusercontent.com/will127534/libcamera/master/src/ipa/rpi/pisp/data/imx585.json
sed -i '8s/"black_level": *[0-9]\+/"black_level": 0/' /home/pi/libcamera/src/ipa/rpi/pisp/data/imx585.json
# cp /home/pi/libcamera/src/ipa/rpi/pisp/data/imx585.json /usr/local/share/libcamera/ipa/rpi/pisp/
```

For the mono sensor use `imx585_mono.json` instead.

2.9 IR filter switch script (optional)

```
wget https://raw.githubusercontent.com/will127534/StarlightEye/master/software/IRFilter -O /usr/local/bin/IRFilter
sudo chmod +x /usr/local/bin/IRFilter
```

Cinemate has its own way of handling the IR switch but the installation above can be convenient for use outside of Cinemate

2.10 Change the console font (optional)

```
sudi apt update
sudo apt install console-setup kbd
sudo dpkg-reconfigure console-setup  # choose Terminus / 16x32
```

Verify `/etc/default/console-setup` contains:

```
FONTFACE="Terminus"
FONTSIZE="16x32"
```

Then enable the service:

```
sudo systemctl enable console-setup.service
sudo systemctl start console-setup.service
```

This can be useful if running the Pi on a small HD field monitor

2.11 Create post-processing configs

Paste this into the terminal and hit enter:

```
sudo bash -c 'cat > post-processing.json << EOF
{
    "sharedContext": {},
    "mjpegPreview": {
        "port": 8000
    }
}
EOF' && \
sudo chmod +x post-processing.json && \
sudo bash -c 'cat > post-processing0.json << EOF
{
    "sharedContext": {},
    "mjpegPreview": {
        "port": 8000
    }
}
EOF' && \
sudo chmod +x post-processing0.json && \
sudo bash -c 'cat > post-processing1.json << EOF
{
    "sharedContext": {},
    "mjpegPreview": {
        "port": 8001
    }
}
EOF' && \
sudo chmod +x post-processing1.json
```

2.12 Install PiShrink

```
wget https://raw.githubusercontent.com/Drewsif/PiShrink/master/pishrink.sh
sudo install -m755 pishrink.sh /usr/local/bin/pishrink
```

PiShrink is a great tool for compressing SD image file backups of the SD card. See here for instructions

2.12.1 Reboot before installing Cinemate:

```
reboot
```

You should now have a working install of cinepi-raw. To try it out, see this section [TBA]

To continue installing Cinemate, follow the steps below:

2.13 Create a Python virtual environment

```
sudo apt update && apt install -y python3-venv
python3 -m venv /home/pi/.cinemate-env
echo "source /home/pi/.cinemate-env/bin/activate" >> ~/.bashrc
source /home/pi/.cinemate-env/bin/activate
```

2.14 Grant sudo privileges and enable I²C

```
echo "pi ALL=(ALL) NOPASSWD: /home/pi/.cinemate-env/bin/*" | sudo tee /etc/sudoers.d/cinemate-env
sudo chown -R pi:pi /home/pi/.cinemate-env
sudo chown -R pi:pi /media && chmod 755 /media
sudo usermod -aG i2c pi
sudo modprobe i2c-dev && echo i2c-dev | sudo tee -a /etc/modules
echo "pi ALL=(ALL) NOPASSWD: /home/pi/run_cinemate.sh" | sudo tee -a /etc/sudoers.d/pi_cinemate
```

Reboot so the group changes take effect:

```
reboot
```

2.15 Cinemate

2.15.1 Dependencies

```
source /home/pi/.cinemate-env/bin/activate
python3 -m pip install --upgrade pip setuptools wheel
sudo apt-get install -y i2c-tools portaudio19-dev build-essential python3-dev python3-pip python3-smbus python3-serial git
pip3 install adafruit-circuitpython-ssd1306 watchdog psutil Pillow redis keyboard pyudev sounddevice smbus2 gpiozero RPI.GPIO evdev termcolor pyserial
inotify_simple numpy rpi_hardware_pwm
pip3 uninstall -y Pillow && pip3 install Pillow
pip3 install sugarpie flask_socketio board adafruit-blinka adafruit-circuitpython-seesaw luma.oled grove.py pigpio-encoder gpiod
sudo apt install python3-systemd e2fsprogs ntfs-3g exfatprogs console-terminus
```

2.15.2 Replace RPi.GPIO with lgpio

```
sudo apt install -y swig python3-dev build-essential git
git clone https://github.com/joan2937/lg
cd lg && make
sudo make install
cd .. && pip install lgpio
```

2.15.3 Clone the Cinemate repo

```
git clone https://github.com/Tiramisioux/cinemate.git
```

2.15.4 Allow Cinemates `main.py` to run with sudo

Edit the sudoers file:

```
sudo visudo
```

add this to the end of the file:

```
pi ALL=(ALL) NOPASSWD: /home/pi/cinemate/src/main.py
pi ALL=(ALL) NOPASSWD: /bin/mount, /bin/umount, /usr/bin/ntfs-3g
pi ALL=(ALL) NOPASSWD: /home/pi/cinemate/src/logs/system.log
pi ALL=(ALL) NOPASSWD: /sbin/mount.ext4
```

2.15.5 Enable NetworkManager

```
sudo systemctl enable NetworkManager --now
```

2.15.6 Rotate logs

Paste this into the terminal and hit enter:

```
# tee /etc/logrotate.d/general_logs <<'EOP'
/var/log/*.log {
   size 100M
   rotate 5
   compress
   missingok
   notifempty
}
EOP
```

2.15.7 Seed Redis with default keys

```
redis-cli <<'EOF'
SET anamorphic_factor 1.0
PUBLISH cp_controls anamorphic_factor
SET bit_depth 12
PUBLISH cp_controls bit_depth
...
EOF
```

(See the settings guide for the full list.)

2.15.8 Add a convenience alias

Append to `~/.bashrc`:

```
alias Cinemate='python3 /home/pi/Cinemate/src/main.py'
```

Then, inside `cinemate` folder:

```
make install
```

2.16 Cinemate services

Cinemate with two small helper services under `services/`:

- **storage-automount** – mounts and unmounts removable drives such as SSDs, NVMe enclosures and the CFE HAT. Partitions named `RAW` are attached at `/media/RAW`; all others are mounted under `/media/<LABEL>`.
- **wifi-hotspot** – keeps a simple Wi-Fi hotspot running via NetworkManager so you can reach the web UI even without other networking. The SSID and password come from the `system.wifi_hotspot` section of `settings.json`.

Install and enable both services with:

```
cd /home/pi/cinemate/services
sudo make install
sudo make enable
```

You can manage each one individually with `make <action>-<service>`, for example `make status-wifi-hotspot`.

2.17 Starting Cinemate

If you are not using the service file for autostart, anywhere in the terminal, type:

```
cinemate
```

This would be the recommended way of trying out Cinemate as you will get extended logging in the terminal which can be helpful when troubleshooting. The Cinemate logger also relays logging messages from the running cinepi-raw instance.

3. Configuration

3.1 Connecting to the Pi with SSH

This guide shows how to log into the Raspberry Pi that runs Cinemate.

3.1.1 Join the same network

Connect your computer and the Pi to the same network. If you are using the preinstalled image file, the system automatically starts a built-in hotspot: join the **CinePi** Wi-Fi with password `11111111`.

You can change this behaviour later in the settings file.

3.1.2 Find the Pi's address

Open a terminal (on Windows you can use PowerShell).

Try the hostname first:

```
ssh pi@cinepi.local
```

If this fails you can list devices on the network:

```
arp -a
```

Look for an entry labelled `cinepi` or note the new IP address that appears.

3.1.3 Connect

Use the hostname or IP address with SSH:

```
ssh pi@cinepi.local
# or
ssh pi@<ip-address>
```

When asked about the host key, type `yes`. Enter the default password `1` when prompted.

You will now see the `pi@cinepi` prompt, meaning you are logged in.

If you are installing Cinemate manually, the hostname has not yet been set to cinepi. Then you will have to identify which ip address on the network is actually the Raspberry Pi and use that ip address.

3.1.4 Next steps

From here you can run `cinemate` to start the interface or use `make` commands to manage the service. For security you should change the password with `passwd` after the first login.

3.2 User settings

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.

3.2.1 welcome message

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.

3.2.2 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.

3.2.3 geometry

Controls image orientation for each camera port (`cam0`, `cam1`, etc.). These settings let you mount cameras in any orientation and still get an upright preview and recording. Example:

```
"geometry": {
  "cam0": { "rotate_180": false, "horizontal_flip": false, "vertical_flip": false },
  "cam1": { "rotate_180": false, "horizontal_flip": false, "vertical_flip": false }
}
```

- **rotate_180** – flip the image upside-down.
- **horizontal_flip** – mirror the image left/right.
- **vertical_flip** – mirror the image top/bottom.

3.2.4 output

Maps each camera to an HDMI connector. Use `-1` for automatic selection.

```
"output": {
  "cam0": { "hdmi_port": 0 },
  "cam1": { "hdmi_port": 1 }
}
```

3.2.5 preview

Adjusts zoom levels for the HDMI/browser preview.

```
"preview": {
  "default_zoom": 1.0,
  "zoom_steps": [1.0, 1.5, 2.0]
}
```

- **default_zoom** – magnification factor used at startup.
- **zoom_steps** – list of zoom factors you can cycle through with the `set_zoom_step` command.

3.2.6 anamorphic_preview

For stretching the preview when using anamorphic lenses.

```
"anamorphic_preview": {
  "default_anamorphic_factor": 1,
  "anamorphic_steps": [1, 1.33, 2.0]
}
```

- **default_anamorphic_factor** – factor loaded when Cinemate starts.
- **anamorphic_steps** – selectable squeeze factors; values above `1.0` widen the image.

3.2.7 gpio_output

Defines pins used for visual feedback or sync signals.

```
"gpio_output": {
  "pwm_pin": 19,
  "rec_out_pin": [6, 21]
}
```

- **pwm_pin** – outputs a strobe for shutter sync or external devices.
- **rec_out_pin** – list of pins pulled high while recording (useful for tally LEDs).

3.2.8 arrays

Preset lists for exposure and frame-rate settings. Cinemate will step through these values unless you enable free mode, either in the settings file or during runtime.

```
"arrays": {
  "iso_steps": [100, 200, 400, 640, 800, 1200, 1600, 2500, 3200],
  "shutter_a_steps": [1, 45, 90, 135, 172.8, 180, 225, 270, 315, 346.6, 360],
  "fps_steps": [1, 2, 4, 8, 12, 16, 18, 24, 25, 33, 40, 50],
  "wb_steps": [3200, 4400, 5600]
}
```

3.2.9 settings

General options for runtime behaviour.

```
"settings": {
  "light_hz": [50, 60],
  "conform_frame_rate": 24
}
```

- **light_hz** – list of mains frequencies used to calculate flicker-free shutter angles. These are added to the shutter angle array and also dynamically calculated upon each fps change. This way, there is always a flicker free shutter angle value close by, when toggling through shutter angles, either via the cli or using buttons/pots/rotary encoder.
- **conform_frame_rate** – frame rate intendend for project conforming in post. This setting is not really used by CineMate except for calculating the recording timecode tracker in redis but might be used in future updates.

3.2.10 analog_controls

Maps Grove Base HAT ADC channels to analogue dials (potentiometers). Use `null` to disable a dial.

```
"analog_controls": {
  "iso_pot": 0,
  "shutter_a_pot": 2,
  "fps_pot": 4,
  "wb_pot": 6
}
```

***Note** that even if you are using a Grove Base Hat, it might be useful to disable the dials not connected to pots, since noise from these connectors might trigger false readings.*

3.2.11 free_mode

When enabled, ignores the preset arrays and exposes the full range supported by the sensor.

```
"free_mode": {
  "iso_free": false,
  "shutter_a_free": false,
  "fps_free": true,
  "wb_free": false
}
```

3.2.12 buttons

Defines GPIO push buttons. Each entry describes one button and the actions it triggers.

```
{
  "pin": 5,
  "pull_up": true,
  "debounce_time": 0.1,
  "press_action": {"method": "rec"}
}
```

- **pin** – BCM pin number the button is connected to.
- **pull_up** – set `true` if the pin idles high (internal pull-up). Use `false` for pull-down wiring.
- **debounce_time** – ignore additional presses within this time window (seconds).
- **press_action**, **single_click_action**, **double_click_action**, **triple_click_action**, **hold_action** – actions to perform for each type of interaction. Actions call Cinemate CLI commands with optional `args`.

HOW “INVERSE” (1-0-1) BUTTONS ARE AUTO-DETECTED

Some push-buttons are wired closed = logic 1 and open = 0. At start-up, CineMate automatically detects buttons in state `true` and reverses them. This way the user can use any type of push buttons, both 1-0-1 and 0-1-0 types.

3.2.13 two_way_switches

Latching on/off switches. Cinemate triggers an action whenever the state changes.

```
{
  "pin": 27,
  "state_on_action":  {"method": "set_all_lock", "args": [1]},
  "state_off_action": {"method": "set_all_lock", "args": [0]}
}
```

3.2.14 rotary_encoders

Rotary encoders used for fine adjustment of settings. These can be wired straight to the GPIO pins of the Pi.

```
{
  "clk_pin": 9,
  "dt_pin": 11,
  "encoder_actions": {
    "rotate_clockwise":        {"method": "inc_iso"},
    "rotate_counterclockwise": {"method": "dec_iso"}
  }
}
```

- **clk_pin** and **dt_pin** – the two pins of the encoder.
- **encoder_actions** – commands to run when turning the dial.

3.2.15 quad_rotary_encoders

Support for the Adafruit Neopixel Quad I2C rotary encoder breakout with four dials. Each entry assigns a dial to a setting and clones the behaviour of a button pin.

```
"quad_rotary_encoders": {
  "0": {"setting_name": "iso", "gpio_pin": 5},
  "1": {"setting_name": "shutter_a", "gpio_pin": 16},
  "2": {"setting_name": "fps", "gpio_pin": 26},
  "3": {"setting_name": "wb", "gpio_pin": 5}
}
```

3.2.16 i2c_oled

Configuration for the optional OLED status screen. This can be useful for presenting extra information appart from the HDMI/ web display.

```
"i2c_oled": {
  "enabled": true,
  "width": 128,
  "height": 64,
  "font_size": 30,
  "values": ["write_speed_to_drive"]
}
```

- **enabled** – turn the OLED display on or off.
- **width / height** – pixel dimensions of your screen.
- **font_size** – size of the displayed text.
- **values** – list of Redis keys or pseudo-keys to show (for example `cpu_temp`).

  Available keys come from `src/module/i2c_oled.py`. Here are some examples:

- `iso`, `fps` – basic camera settings.
- `shutter_a` – shown as **SHUTTER** with a `°` suffix.
- `wb_user` – shown as **WB** with a trailing `K`.
- `space_left` – displayed as **SPACE** in gigabytes.
- `write_speed_to_drive` – write speed in MB/s.
- `resolution` – prints `width×height@bit_depth` on the first line.
- `is_recording` – draws a bullet `●` when recording.
- `cpu_load`, `cpu_temp`, `memory_usage` – Pi system statistics.

  Other keys will display their name in uppercase and the raw value from Redis.

3.3 cinepi-raw terminal commands

Here is how you can operate **CinePi-raw** from the command line.

3.3.1 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 Cinemate.

3.3.2 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:

```
--mode 2028:1080:12:U
```

- `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.

3.3.3 Low-resolution (lores) stream

```
--lores-width 1280 --lores-height 720
```

CinePi-raw can produce a secondary low-resolution stream alongside the full-resolution RAW frames.

3.3.4 Preview window

By default the program opens an HDMI preview so you can see what the camera captures. The size and position of this window are controlled with:

```
-p 0,30,1920,1020
```

This positions the preview 30 pixels from the top of the screen with a 1920×1020 window.

3.3.5 Tuning files

```
--tuning-file /home/pi/cinemate/resources/tuning_files/imx477.json
```

Describes the camera’s colour and lens characteristics. Point to a file supplied with Libcamera (for example `imx477.json` for the HQ camera)

3.3.6 Post processing

--post-process-file /home/pi/post-processing.json

For cinepi-raw, this file defines the port used by cpp-mjpeg-streamer (default cinepi.local:8000)

If you have more than one camera connected to the Pi, and activated in `boot/firmware/config.txt`, the camera commected to physical cam0 will use /home/pi/post-processing0.json and the camera connected to cam1 will use /home/pi/post-processing1.json

3.3.7 Cinemate-specific flags

The CineMate fork introduces several extra options:

Flag	Argument	Description
`--cam-port`	`cam0` | `cam1`	Select which CSI camera port to use.
`--hdmi-port`	`0` | `1` | `-1`	Choose the HDMI connector for the preview (`0` = HDMI-0, `1` = HDMI-1, `-1` = auto-detect).
`--same-hdmi`	*(none)*	Force both capture and controller GUI to share the same HDMI output.
`--keep16`	`true` | `false`	Save full 16-bit DNGs instead of 12-bit packed files.

At this moment though, Cinemate is 12bit only. The flag is for future updates of the IMX585 16bit clear HDR modes.

3.3.8 Example commands

Below are sample commands for different sensors and modes.

IMX477 (12-bit, full width)

```
cinepi-raw --mode 4056:2160:12 --width 4056 --height 2160 \
           --lores-width 1280 --lores-height 720 \
           -p 0,30,1920,1020 \
           --post-process-file /home/pi/post-processing.json \
           --tuning-file /home/pi/libcamera/src/ipa/rpi/pisp/data/imx477.json \
```

IMX585 (12-bit unpacked)

```
cinepi-raw --mode 1928:1090:12:U --width 1928 --height 1090 \
           --lores-width 1280 --lores-height 720 \
           -p 0,30,1920,1020 \
           --post-process-file /home/pi/post-processing.json \
           --tuning-file /home/pi/libcamera/src/ipa/rpi/pisp/data/imx585.json \
```

Now, with an SSH shell running redis-cli you should be able to capture RAW footage from the command line!

```
redis-cli
> set is_recording 1
> publish cp_controls is_recording
```

3.4 Cinemate terminal commands

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.

3.4.1 Available Commands

Command	**Input type**	**Example**	**Function**
`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
`inc wb` / `dec wb`	none	`inc wb`	Cycle white balance steps
`set resolution [<mode>]`	int or none	`set resolution 2`	Apply or cycle sensor mode
`set anamorphic factor [<float>]`	float or none	`set anamorphic factor 1.33`	Set or toggle anamorphic stretch
`set zoom [<float>]`	float or none	`set zoom 2`	Change digital zoom; omit to cycle
`inc zoom` / `dec zoom`	none	`inc zoom`	Step preview zoom factor
`mount` / `unmount`	none	`mount`	Mount or unmount external storage
`toggle mount`	none	`toggle mount`	Mount if not mounted, otherwise unmount
`time`	none	`time`	Show system and RTC time
`set rtc time`	none	`set rtc time`	Copy system time to the RTC
`space`	none	`space`	Report remaining SSD space
`get`	none	`get`	Print all current settings
`set shutter a sync [0/1]`	0/1 or none	`set shutter a sync 1`	Enable exposure sync mode
`set iso lock [0/1]`	0/1 or none	`set iso lock`	Lock or unlock ISO setting
`set shutter a nom lock [0/1]`	0/1 or none	`set shutter a nom lock`	Lock or unlock nominal shutter
`set shutter a nom fps lock [0/1]`	0/1 or none	`set shutter a nom fps lock 1`	Lock nominal shutter and FPS together
`set fps lock [0/1]`	0/1 or none	`set fps lock 1`	Lock or unlock the frame rate
`set all lock [0/1]`	0/1 or none	`set all lock 0`	Toggle all exposure locks at once
`set fps double [0/1]`	0/1 or none	`set fps double`	Instant or toggled 2× FPS mode
`reboot` / `shutdown`	none	`reboot`	Safely reboot or halt the Pi
`restart camera`	none	`restart camera`	Restart the libcamera pipeline

Command	Input type	Example	Function
`restart cinemate`	none	`restart cinemate`	Restart the Cinemate process
`set iso free [0/1]`	0/1 or none	`set iso free 1`	Allow any ISO instead of presets
`set shutter a free [0/1]`	0/1 or none	`set shutter a free 0`	Allow any shutter angle
`set fps free [0/1]`	0/1 or none	`set fps free 1`	Allow any FPS
`set wb free [0/1]`	0/1 or none	`set wb free`	Allow any white balance
`set filter <0/1>`	0/1	`set filter 1`	Toggle IR-cut filter (IMX585)

Commands without an explicit argument will toggle the current state when possible (e.g. `set fps lock` flips the lock; `set fps lock 1` forces it on).

4. Operating the camera

4.1 Simple GUI

Simple GUI is available via browser and/or attached HDMI monitor.

- Red color means camera is recording.
- Purple color means camera detected a drop frame
- Green color means camera is writing buffered frames to disk. You can still start recording at this stage, but any buffered frames from the last recording will be lost.

Buffer meter in the lower left indicates number of frames in buffer. Useful when testing storage media.

When a compatible USB microphone is connected, VU meters appear on the right side of the GUI so you can monitor audio levels.

4.2 Compatible sensors

Sensor	Cinemate sensor mode	Resolution	Aspect Ratio	Bit Depth	Max FPS*	File S (MB)
IMX283	0	2736 x 1538	1.80	12	40	7.1
	1	2736 x 1824	1.53	12	34	8.2
IMX296	0	1456 x 1088	1.33	12	60	2
IMX477	0	2028 x 1080	1.87	12	50	4.3
	1	2028 x 1520	1.33	12	40	5.3
	2	1332 x 990	1.34	10	120	2.7
IMX585	0	1928 x 1090	1.77	12	87	4
	1	3840 x 2160	1.77	12	34	4

Note that maximum fps will vary according to disk write speed. For the specific fps values for your setup, make test recordings and monitor the output. Purple background in the monitor/web browser indicates drop frames. You can cap Cinemates max fps values for your specific build by editing the file `cinemate/src/module/sensor_detect.py`

4.3 Audio recording (experimental)

Cinemate can capture audio alongside the image sequence. Support is currently limited to a few USB microphones with hard coded configurations: - **RØDE VideoMic NTG** – recorded in stereo at 24-bit/48 kHz. - **USB PnP microphones** – recorded in mono at 16-bit/48 kHz.

Audio is written as `.wav` files into the same folder as the `.dng` frames. The implementation is still experimental and audio/video synchronization needs further investigation.

.asoundrc Setup

For `dsnoop` support, create a `~/.asoundrc` in home directory:

```
nano ~/.asoundrc
```

Paste this into the file:

```
    pcm.dsnoop_24bit {
        type dsnoop
        ipc_key 2048
        slave {
            pcm "hw:Device,0"
            channels 2
            rate 48000
            format S24_3LE
            period_size 1024
            buffer_size 4096
        }
    }

    pcm.dsnoop_16bit {
        type dsnoop
        ipc_key 2049
        slave {
            pcm "hw:Device,0"
            channels 1
            rate 48000
            format S16_LE
            period_size 1024
            buffer_size 4096
        }
    }

    pcm.mic_24bit {
        type plug
        slave.pcm "dsnoop_24bit"
    }

    pcm.mic_16bit {
        type plug
        slave.pcm "dsnoop_16bit"
    }
```

Exit nano editor using ctrl+x.

4.4 Speed ramping

Speed ramping is the process of changing the camera's frame rate during a shot so that playback speed varies once the footage is conformed to a constant frame rate in post production. Ramping up the frame rate produces slow motion while ramping down speeds up the action.

4.4.1 Speed ramping in Cinemate

Cinemate exposes frame rate control through the `CinePiController` class. The simplest way to change speed on the fly is the CLI command:

```
set fps <value>
```

For quick 2× changes Cinemate also implements `set_fps_double` which toggles between the current FPS and twice that value:

```
    def set_fps_double(self, value=None):
        target_double_state = not self.fps_double if value is None else value in (1, True)
        if target_double_state:
            if not self.fps_double:
                self.fps_saved = self.fps
                target_fps = min(self.fps * 2, self.fps_max)
                self.set_fps(target_fps)
        else:
            if self.fps_double:
                self.set_fps(self.fps_saved)
        self.fps_double = target_double_state
```

The controller contains an experimental `_ramp_fps` helper that gradually steps the frame rate up or down using `ramp_up_speed` and `ramp_down_speed` delays. This can be adapted if smoother transitions are desired.

4.4.2 Shutter angle synchronisation

When frame rate changes the shutter angle can either remain fixed (preserving motion blur) or adjust to keep the exposure time constant. This behaviour is controlled by `shutter_a_sync_mode`.

```
        if self.shutter_a_sync_mode == 0:
            # keep motion-blur constant
            self.initialize_shutter_angle_steps()
            self.shutter_angle_actual = min(
                self.shutter_a_steps_dynamic,
                key=lambda x: abs(x - self.shutter_angle_actual))
        else:
            # keep exposure-time constant
            self.shutter_angle_actual = round(
                self.exposure_time_nominal * self.current_fps * 360, 1)
            self.shutter_angle_actual = min(360.0,
                                            max(1.0, self.shutter_angle_actual))
```

Mode `0` keeps the motion blur consistent because the physical shutter angle does not change. As the FPS increases the exposure time gets shorter, resulting in a darker image. Mode `1` stores the current exposure time and recalculates the shutter angle whenever the FPS is adjusted so that brightness stays the same.

Cinemate updates the nominal exposure time when the user sets a new angle:

```
    if self.shutter_a_sync_mode == 1:
        self.exposure_time_nominal = (new_angle / 360) / self.current_fps
        self.shutter_angle_actual = new_angle
        self.is_shutter_angle_transient = True
        self.redis_controller.set_value(ParameterKey.SHUTTER_A_TRANSIENT.value, 1)
        threading.Timer(0.5, self.end_shutter_angle_transient).start()
```

When the transient period ends, the FPS is recalculated from the stored exposure time:

```
    def end_shutter_angle_transient(self):
        self.is_shutter_angle_transient = False
        self.redis_controller.set_value(ParameterKey.SHUTTER_A_TRANSIENT.value, 0)
        if self.shutter_a_sync_mode == 1:
            adjusted_fps = (self.shutter_angle_nom / 360) / self.exposure_time_nominal
            self.update_fps(round(adjusted_fps, 1))
```

5. System management

5.1 System services

5.1.1 cinemate-autostart.service

```
make install   # copy service file
make enable    # start on boot
make start     # launch now
make stop      # stop it
make status    # check status
make disable   # disable autostart
make clean     # remove the service
```

Note that in order for the web ui to work properly you have to run `make install` once in the `/home/pi/cinemate` folder, even if you are not using the autostart service.

5.1.2 storage-automount.service

`storage-automount` is a systemd service that watches for removable drives and mounts them automatically. The accompanying Python script reacts to udev events and the CFE-HAT eject button so drives can be attached or detached safely.

It understands `ext4`, `ntfs` and `exfat` filesystems. Partitions labelled `RAW` are mounted at `/media/RAW`; any other label is mounted under `/media/<LABEL>` after sanitising the name. This applies to USB SSDs, NVMe drives and the CFE-HAT slot.

To manually install and enable the service:

```
cd cinemate/services/storage-automount
sudo make install
sudo make enable
```

You can stop or disable it later with:

```
sudo make stop
sudo make disable
```

5.1.3 wifi-hotspot.service

`wifi-hotspot` keeps a small access point running with the help of NetworkManager so you can always reach the web interface. The SSID and password are read from `/home/pi/cinemate/src/settings.json` under `system.wifi_hotspot`.

Install and enable it with:

```
cd cinemate/services/wifi-hotspot
sudo make install
sudo make enable
```

As with `storage-automount`, you can stop or disable the hotspot with `make stop` and `make disable`.

5.1.4 Adjusting config.txt for different sensors:

```
sudo nano /boot/firmware/config.txt
```

Uncomment the section for the sensor being used, and make sure to comment out the others. Reboot the Pi for changes to take effect.

Exit the editor by pressing Ctrl+C

5.1.5 Compiling cinepi-raw

For easy later rebuilding and installation of cinepi-raw you can create the file compile-raw.sh.

```
nano compile-raw.sh
```

Paste this into the file

```
sudo meson install -C build
```

Exit by pressing Ctrl+C

Make it exectutable:

```
sudo chmod +x compile-raw.sh
```

Now, from the same folder, to build and install cinepi-raw:

```
./compile-raw.sh
```

5.2 Hotspot logic

If `wifi_hotspot` in `settings.json` is `true` and no hotspot is active, Cinemate starts its own hotspot `nmcli device wifi hotspot` using your chosen SSID and password. If the Pi is already connected to wifi (for example WiFi settings set with `sudo raspi-config`) this connection will be replaced by Cinemates hotspot. Set `enabled: false` to keep wlan0 free for regular Wi-Fi use.

Note that Cinemate still streams its web gui on whatever network the Pi is connected to, with GUI at :5000 and clean preview without GUI on :8000/stream

5.3 Backing up the SD card

Create a compressed image:

```
sudo dd if=/dev/mmcblk0 bs=4M conv=sparse,noerror status=progress | \ gzip -c > /media/RAW/Cinemate_$(date +"%Y%m%d_%H%M%S").img.gz
```

Or use PiShrink for a smaller file:

```
sudo bash -euo pipefail -c '
  ts=$(date +%Y%m%d_%H%M%S)
  raw="/media/RAW/Cinemate_${ts}.img"
  final="/media/RAW/Cinemate_${ts}.img.gz"
  dd if=/dev/mmcblk0 of="$raw" bs=4M conv=sparse,noerror status=progress
  pishrink.sh -v -z "$raw" "$final"
  rm -f "$raw"
'
```

6. Redis

6.1 Redis API quick start

You can think of the toolchain like this:

camera sensor > libcamera > cinepi-raw <> redis <> cinemate

It is a bit more complicated than that but in the context of describing redis and Cinemate it is pretty accurate

- 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.

6.1.1 The `cp_controls` channel

Both cinepi-raw and CineMate writes values and immediately publish 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
```

6.1.2 The `cp_stats` channel

Every frame, cinepi-raw sends a small JSON object containing live statistics.

```
    Json::Value data;
    Json::Value histo;
    data["framerate"] = completed_request->framerate;
    data["colorTemp"] = info.colorTemp;
    data["focus"] = info.focus;
    data["frameCount"] = app_->GetEncoder()->getFrameCount();
    data["bufferSize"] = app_->GetEncoder()->bufferSize();
    redis_->publish(CHANNEL_STATS, data.toStyledString());
```

CineMate’s `RedisListener` parses these messages and updates Redis keys like `framecount` , `BUFFER` and `fps_actual` .

6.1.3 Inspecting and changing values with `redis-cli`

Because everything is plain Redis you can poke around from the command line. Here are a few handy commands:

```
# List all keys
redis-cli KEYS '*'

# Read the current ISO value
redis-cli GET iso

# Start a recording (same as pressing the Rec button)
redis-cli SET is_recording 1
redis-cli PUBLISH cp_controls is_recording
```

6.1.4 Controlling the camera from your own script

Below is a very small example using `redis-py`. This is basically what Cinemate does: it keeps track of variables being set by cinepi-raw, and also setting the same, or other variable by itself.

```
import redis
r = redis.Redis(host='localhost', port=6379, db=0)

# toggle recording
current = r.get('is_recording')
new_value = b'0' if current == b'1' else b'1'
r.set('is_recording', new_value)
r.publish('cp_controls', 'is_recording')
```

Note that in this example, the publishing of the is_recording key is not strictly needed for recording to start/stop, but for formality's sake I think we should keep the publish command.

6.2 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
zoom	Cinemate	Digital zoom for preview streams	Yes
ir_filter	Cinemate → cinepi-raw	Toggle IR-cut filter (IMX585 only)	Yes
rec / is_recording	Cinemate → cinepi-raw	Start/stop recording when toggled	Yes (edge-triggered)
bit_depth	Cinemate → cinepi-raw startup	Sensor bit depth (10 or 12)	No (set at startup)
height / width	Cinemate → cinepi-raw startup	Active sensor resolution	No
lores_width / lores_height	cinepi-raw startup	Preview stream resolution	No
cg_rb	Cinemate → cinepi-raw	White-balance gain pair "1/R,1/B"	Yes (advanced)
fps_user	Cinemate	Temporary storage for the UI slider	No
fps_last	Cinemate	Previous stable fps from stats	No
fps_actual	cinepi-raw → Cinemate	Measured FPS from pipeline	No
framecount	cinepi-raw → Cinemate	Total frames recorded	No
buffer	cinepi-raw → Cinemate	Raw frames currently in RAM	No
buffer_size	cinepi-raw → Cinemate	Size of RAM buffer in frames	No
is_buffering	cinepi-raw → Cinemate	1 while buffer pre-fills	No
is_writing	cinepi-raw → Cinemate	1 while frames are flushing to disk	No
is_writing_buf	Cinemate	Internal countdown after recording stops	No
is_mounted	Cinemate (SSD monitor)	1 when storage is mounted	No
storage_type	Cinemate (SSD monitor)	Drive type (NVME/USB/SD)	No
space_left	Cinemate (SSD monitor)	Remaining space in GB	No

Key	Written by	Description	Safe to change manually?
write_speed_to_drive	Cinemate (SSD monitor)	Current write speed MB/s	No
file_size	Cinemate	Bytes per frame for current mode	No
last_dng_cam0/1	cinepi-raw → Cinemate	Path to last written DNG frame	No
recording_time	Cinemate	HH:MM:SS:FF timer while recording	No
memory_alert	Cinemate	1 if RAM usage high	No
cam_init	cinepi-raw	Internal flag during startup	No
cameras	cinepi-raw	JSON list of detected cameras	No
gui_layout	Cinemate	Path to GUI layout preset	No
pi_model	Cinemate	Raspberry Pi model string	No
sensor	cinepi-raw	Active camera model	No
tc_cam0/tc_cam1	cinepi-raw → Cinemate	SMPTE time code per camera	No
shutter_angle_actual	Cinemate	Calculated shutter angle applied after clamping or sync	No
shutter_angle_transient	Cinemate	Temporary value during ramping	No
exposure_time	Cinemate	Current exposure time in seconds	No
wb_user	Cinemate	Kelvin value set before converting to cg_rb	No