Cinemate

User & Builder Guide

None

None

Table of contents

1. Cinemate Docs 6

2. Quick start 7

2.1 Quick start 7

2.1.1 Hardware requirements 7

2.1.2 Installation 7

2.2 Simple GUI 8

3. Using the camera 10

3.1 CLI user guide 10

3.1.1 Checking available options 10

3.1.2 Camera modes 10

3.1.3 Low-resolution (lores) stream 11

3.1.4 Preview window 11

3.1.5 Tuning files 11

3.1.6 Post processing 11

3.1.7 Cinemate-specific flags 12

3.1.8 Example commands 12

3.2 Commands reference 14

3.2.1 Available Commands 15

3.3 Compatible sensors 20

3.4 Audio recording (experimental) 22

3.5 Speed ramping 25

3.5.1 Speed ramping in Cinemate 25

3.5.2 Shutter angle synchronisation 26

3.6 Dual sensors 27

4. Customising your build 28

4.1 Connecting to the Pi with SSH 28

4.2 Settings file 30
4.2.1 welcome message 30
4.2.2 system 30
4.2.3 geometry 31
4.2.4 output 31
4.2.5 preview 32
4.2.6 anamorphic_preview 32
4.2.7 gpio_output 32
4.2.8 arrays 33
4.2.9 settings 33
4.2.10 analog_controls 34
4.2.11 free_mode 34
4.2.12 buttons 35
4.2.13 two_way_switches 35
4.2.14 rotary_encoders 36
4.2.15 quad_rotary_controller 37
4.2.16 i2c_oled 38
4.3 Configuring the Wi-Fi hotspot 40
5. System management 41
5.1 System services 41
5.1.1 cinemate-autostart.service 41
5.1.2 storage-automount.service 41
5.1.3 wifi-hotspot.service 42
5.2 Modifying config.txt 43
5.2.1 Adjusting config.txt for different sensors: 43
5.3 Recompiling cinepi-raw 44
5.4 Overclocking the Pi 45
5.5 Backing up the SD card 46

6. Reference & troubleshooting 47

6.1 Redis API quick start 47

6.1.1 Redis-cli 47

6.1.2 cp_controls 47

6.1.3 cp_stats 48

6.1.4 Controlling the camera from your own script 49

6.2 Redis key reference 50

6.3 Hardware overview 57

7. Cinemate builds 60

8. Development & contributing 62

8.1 Installation 62

8.1.1 Camera stack 62

8.1.2 Cinemate 69

8.1.3 Todo 76

8.2 Acknowledgements 79

1. Cinemate Docs

Welcome to the **Cinemate** project - an open-source boiler plate for building your own digital cinema camera using a Raspberry Pi 5. It combines a lightweight Python interface with the CinePi-raw recorder by Csaba Nagy for capturing 12-bit CinemaDNG footage.

To begin, follow the steps in Quick start. Later chapters explain how to customise the system to tailor Cinemate to your needs.

For sharing your build with others, inspiration and discussion, make sure to join the CinePi Discord.

Download this documentation in pdf format for easy offline reference.

2. Quick start

2.1 Quick start

2.1.1 Hardware requirements

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

2.1.2 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 T 7 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 **GPIO** 5 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!

2.2 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.

TBA

3. Using the camera

3.1 CLI user guide

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

3.1.1 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.

3.1.2 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 IMX `477` 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.

3.1.3 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.

3.1.4 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.

3.1.5 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)

3.1.6 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 cam 0 will use /home/pi/post-processing 0 .json and the camera connected to cam 1 will use /home/pi/post-processing 1 .json

3.1.7 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 12 bit only. The flag is for future updates of the IMX585 16 bit clear HDR modes.

3.1.8 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
```

3.2 Commands reference

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.

3.2.1 Available Commands

Command	Input type	Example
`rec` / `stop`	none	`rec`
`set iso <value>`	int	`set iso 800`
`inc iso` / `dec iso`	none	`inc iso`
`set shutter a <angle>`	float	`set shutter a 180`
`inc shutter a` / `dec shutter a`	none	`inc shutter a`
`set shutter a nom <angle>`	float	`set shutter a nom 180`
`inc shutter a nom` / `dec shutter a nom`	none	`inc shutter a nom`
`set fps <value>`	float	`set fps 24`
`inc fps` / `dec fps`	none	`inc fps`

Command	Input type	Example
`set wb [<Kelvin>]`	int or none	`set wb 5600`
`inc wb` / `dec wb`	none	`inc wb`
`set resolution [<mode>]`	int or none	`set resolution 2`
`set anamorphic factor [<float>]`	float or none	`set anamorphic factor 1.33`
`set zoom [<float>]`	float or none	`set zoom 2`
`inc zoom` / `dec zoom`	none	`inc zoom`
`mount` / `unmount`	none	`mount`
`toggle mount`	none	`toggle mount`
`time`	none	`time`
`set rtc time`	none	`set rtc time`

Command	Input type	Example
`space`	none	`space`
`get`	none	`get`
`set shutter a sync [0/1]`	0 / 1 or none	`set shutter a sync 1`
`set iso lock [0/1]`	0 / 1 or none	`set iso lock`
`set shutter a nom lock [0/1]`	0 / 1 or none	`set shutter a nom lock`
`set shutter a nom fps lock [0/1]`	0 / 1 or none	`set shutter a nom fps lock`
`set fps lock [0/1]`	0 / 1 or none	`set fps lock 1`
`set all lock [0/1]`	0 / 1 or none	`set all lock 0`
`set fps double [0/1]`	0 / 1 or none	`set fps double`

Command	Input type	Example
`reboot` / `shutdown`	none	`reboot`
`restart camera`	none	`restart camera`
`restart cinemate`	none	`restart cinemate`
`set iso free [0/1]`	0 / 1 or none	`set iso free 1`
`set shutter a free [0/1]`	0 / 1 or none	`set shutter a free 0`
`set fps free [0/1]`	0 / 1 or none	`set fps free 1`
`set wb free [0/1]`	0 / 1 or none	`set wb free`
`set filter <0/1>`	0 / 1	`set filter 1`

> 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).

3.3 Compatible sensors

Sensor	Cinemate sensor mode	Resolution	Aspect Ratio	Bit Depth	Max FPS *	Fil Siz (M
IMX283	0	2736 x 1538	1.80	12	40	7
	1	2736 x 1824	1.53	12	34	8
IMX296	0	1456 x 1088	1.33	12	60	2
IMX477	0	2028 x 1080	1.87	12	50	4
	1	2028 x 1520	1.33	12	40	5
	2	1332 x 990	1.34	10	120	2
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`

3.4 Audio recording (experimental)

Cinemate records 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.

3.5 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.

3.5.1 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 $2\times$ changes Cinemate also implements `set_fps_double` which toggles between the current FPS and twice that value. This can be used for designing a slow-motion button. Here is how you would to it in the settings file, button section:

```
{
  "pin": 18,
  "pull_up": true,
  "debounce_time": 0.1,
  "press_action": {"method": "set_fps_double"}
}
```

> No argument is needed here. For methods such as `set_fps_double`, calling the method without an argument will simply toggle the control, in tis caseturning the slow motion on and off. If the user provides an argument, the control will be set explicitly to that value.

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.

3.5.2 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`.

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 **exposure time** stays the same.

Cinemate updates the nominal exposure time when the user sets a new angle. FPS is recalculated from the stored exposure time:

3.6 Dual sensors

CineMate automatically detects each camera connected to the Raspberry Pi and spawns a separate `cinepi-raw` process per sensor. By default:

- **Primary camera** (first detected) displays its preview on HDMI port `0`.
- **Secondary cameras** run with `--nopreview` and map to subsequent HDMI outputs (cam `1` → HDMI `1`, cam `2` → HDMI `2`, etc.).
- Preview windows are centered and sized according to your `geometry` settings.

Cameras are synchronized with cam `0` being the server and cam `1` being the client.

You can override default HDMI mappings in `settings.json` under the `output` section.

4. Customising your build

4.1 Connecting to the Pi with SSH

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.

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.

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.

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.

4.2 Settings file

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.

4.2.1 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.

4.2.2 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.

Use the hotspot when you need a direct connection in the field. Disable it during development so the Pi can join your regular Wi-Fi and reach the internet. If you are connected to the Pi via Ethernet you can keep the hotspot on.

4.2.3 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_f
lip": false },
  "cam1": { "rotate_180": false, "horizontal_flip": false, "vertical_f
lip": false }
}
```

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

4.2.4 output

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

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

4.2.5 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.

4.2.6 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.

4.2.7 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).

4.2.8 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]
}
```

4.2.9 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.

4.2.10 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.*

4.2.11 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
}
```

4.2.12 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`.

Automatic detection of inverse push buttons

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.

4.2.13 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]}
}
```

4.2.14 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.

4.2.15 quad_rotary_controller

Support for the Adafruit Neopixel Quad I 2 C rotary encoder breakout. Each entry maps one of the four dials to a setting and defines the push button actions similar to the `buttons` section.

```
"quad_rotary_controller": {
  "enabled": true,
  "encoders": {
    "0": {"setting_name": "iso", "button": {"press_action":
{"method": "rec"}}},
    "1": {"setting_name": "shutter_a", "button": {"press_action": {"me
thod": "set_fps_double"}}},
    "2": {
      "setting_name": "fps",
      "button": {
        "press_action": "None",
        "single_click_action": {"method": "set_resolution"},
        "double_click_action": {"method": "restart_cinemate"},
        "triple_click_action": {"method": "reboot"},
        "hold_action": {"method": "toggle_mount"}
      }
    },
    "3": {"setting_name": "wb", "button": {"press_action": {"method":
"rec"}}}
  }
}
```

- **enabled** – turn the quad rotary controller on or off.
- **encoders** – mapping of each dial to a setting and button actions.

4.2.16 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/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.

4.3 Configuring the Wi-Fi hotspot

The built-in hotspot ensures you can always reach Cinemate even when there is no other network available. When `wifi_hotspot` in `settings.json` is set to `true` and no hotspot is active, Cinemate runs `nmcli device wifi hotspot` using your chosen SSID and password.

This is handy when shooting in the field. Connect your phone or laptop directly to the hotspot and browse to the GUI to control the camera. If the Pi was previously connected to a Wi-Fi network, that connection is replaced by the hotspot.

During development you may want the Pi to join your normal Wi-Fi so it has internet access. Set `system.wifi_hotspot.enabled` to `false` and configure Wi-Fi through `raspi-config` or the desktop tools. The web interface will appear on the Pi's regular network address, letting you stay connected to both the Pi and the internet.

If you plug an Ethernet cable into the Pi, you can keep the hotspot running while also having a wired connection for internet and local networking.

> 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. 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`.

5.2 Modifying config.txt

5.2.1 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

5.3 Recompiling cinepi-raw

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
```

5.4 Overclocking the Pi

5.5 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 -Eeuo pipefail -c '
  # Timestamp like 2025-07-19_19-38-33
  ts=$(date +%F_%H-%M-%S)

  # Paths on /media/RAW
  raw="/media/RAW/Cinemate_${ts}.img"         # working image
  final="/media/RAW/cinemate_${ts}.img.xz"    # desired end-result

  # 1 — Image the SD-card (pads bad blocks, keeps sparsity)
  dd if=/dev/mmcblk0 of="$raw" \
     bs=4M conv=noerror,sync,sparse status=progress

  # 2 — Shrink + parallel-xz compress **in place**
  /usr/local/bin/pishrink.sh -s -v -Z -a "$raw"

  # 3 — Rename the freshly-made .xz to the lowercase style you want
  mv "${raw}.xz" "$final"

  # 4 — Remove the now-unused raw image
  rm -f "$raw"
'
```

6. Reference & troubleshooting

6.1 Redis API quick start

6.1.1 Redis-cli

```
# 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
```

You can also type:

```
redis-cli
```

This will open the redis cli.

6.1.2 cp_controls

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
```

> Note that for the **is_recording** key Cinemate 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
```

6.1.3 cp_stats

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`.

6.1.4 Controlling the camera from your own script

Below is a very small example using `redis-py`.

```
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.

Info

This is basically what Cinemate does: it keeps track of variables being set by cinepi-raw, and also sets variables itself.

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

Key	Written by	Description	Safe to change manually?
ir_filter	Cinemate → CinePi-raw	Toggle IR-cut filter (IMX 585 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

Key	Written by	Description	Safe to change manually?
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
write_speed_to_drive	Cinemate (SSD monitor)	Current write speed MB/s	No
file_size	Cinemate	Bytes per frame for current mode	No

Key	Written by	Description	Safe to change manually?
last_dng_cam 0 / 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_cam 0 /tc_cam 1	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

Key	Written by	Description	Safe to change manually?
exposure_time	Cinemate	Current exposure time in seconds	No
wb_user	Cinemate	Kelvin value set before converting to cg_rb	No

6.3 Hardware overview

CineMate image file comes pre-installed with: - StarlightEye - CFE Hat - Grove Base HAT

7. Cinemate builds

TBA

8. Development & contributing

8.1 Installation

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

Stack is confirmed to work on Raspberry pi `4` and `5` models.

This guide assumes fresh Raspbery Pi Bookworm installation running kernel `6.12.34`.

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

```
sudo apt-get install python3-jinja2 python3-ply python3-yaml
```

8.1.1 Camera stack

Tools & dependencies

```
sudo apt update -y
sudo apt upgrade -y
```

```
sudo apt install -y git 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 ~
```

libcamera 1.7.0 raspberry pi fork

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

```
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
```

```
git clone https://github.com/raspberrypi/libcamera.git && \
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 ~/libcamera/utils && sudo chmod +x *.py *.sh && sudo chmod +x ~/
libcamera/src/ipa/ipa-sign.sh && cd ~/libcamera && sudo ninja -C
build install
```

cpp-mjpeg-streamer

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

CinePi-raw cinemate fork

```
git clone https://github.com/Tiramisioux/cinepi-raw.git --branch
rpicam-apps_1.7_custom_encoder
cd cinepi-raw
sudo rm -rf build
sudo meson setup build
sudo ninja -C build
sudo meson install -C build
cd
sudo ldconfig
```

Seed Redis with white balance default keys

```
redis-cli <<EOF
SET cg_rb 2.5,2.2
PUBLISH cp_controls cg_rb
EOF
```

Additional drivers

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

imx585 driver

```
git clone https://github.com/will127534/imx585-v4l2-driver.git --
branch 6.12.y
cd imx585-v4l2-driver/
./setup.sh
cd
```

Add IMX585 tuning files

```
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/
```

```
curl -L -o /home/pi/libcamera/src/ipa/rpi/pisp/data/imx585_mono.json
https://raw.githubusercontent.com/will127534/libcamera/master/src/ipa/
rpi/pisp/data/imx585_mono.json && sudo cp /home/pi/libcamera/src/ipa/
rpi/pisp/data/imx585_mono.json /usr/local/share/libcamera/ipa/rpi/
pisp/
```

IR filter switch script

```
sudo 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

imx283 driver

```
git clone https://github.com/will127534/imx283-v4l2-driver.git
cd imx283-v4l2-driver/
./setup.sh
cd
```

Enabling I²C

```
sudo raspi-config nonint do_i2c 0
```

> Enabling I 2 C is needed for using the camera modules.

Setting hostname

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

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

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 (example shows imx `4 7 7` activated)

```
# 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
```

Exit with Ctrl+x. System will ask you to save the file. Press "y" and then enter.

Change the console font (optional)

```
sudo apt install console-setup kbd
sudo dpkg-reconfigure console-setup

# choose: UTF-8
#         Guess optimal character set
#         Terminus
#         16x32 (framebuffer only)
```

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

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
```

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

Reboot:

```
sudo reboot
```

Trying out CinePi from the terminal

You should now have a working install of cinepi-raw. To see if your camera is recognized by the system:

```
cinepi-raw --list-cameras
```

Try it out with a simple cli command:

```
cinepi-raw --mode 2028:1080:12:U --width 2028 --height 1080 --lores-
width 1280 --lores-height 720
```

For more details on running CinePi-raw from the command line, see this section.

8.1.2 Cinemate

System wide packages

```
sudo apt update
sudo apt install -y \
    git build-essential python3-dev python3-pip python3-venv \
    i2c-tools python3-smbus python3-pyudev \
    libgpiod-dev libgpiod2 python3-libgpiod gpiod \
    portaudio19-dev python3-systemd \
    e2fsprogs ntfs-3g exfatprogs \
    console-terminus
```

Create a Python virtual environment

```
python3 -m venv ~/.cinemate-env
source /home/pi/.cinemate-env/bin/activate
echo "source /home/pi/.cinemate-env/bin/activate" >> ~/.bashrc
```

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:

```
sudo reboot
```

Python packages

> If you previously installed the `board` Python package, remove it with `pip3 uninstall board`.

```
pip install \
    gpiozero \
    adafruit-blinka adafruit-circuitpython-ssd1306 adafruit-
circuitpython-seesaw \
    luma.oled grove.py pigpio-encoder smbus2 rpi_hardware_pwm \
    watchdog psutil pillow redis keyboard pyudev numpy termcolor sound
device \
    evdev inotify_simple sysv_ipc flask_socketio sugarpie
```

Alternative GPIO back-end

```
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
```

Clone the Cinemate repo

```
  git clone https://github.com/Tiramisioux/cinemate.git --branch
cinemate-3.1
```

Allow Cinemate 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
```

Exit with Ctrl+x. System will ask you to save the file. Press "y" and then enter.

Enable NetworkManager

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

Rotate logs

Paste this into the terminal and hit enter:

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

Seed Redis with default keys

```
redis-cli MSET \
anamorphic_factor 0 bit_depth 0 buffer 0 buffer_size 0 cam_init 0 came
ras 0 cg_rb 3.5,1.5 \
file_size 0 fps 24 fps_actual 24 fps_last 24 fps_max 1 fps_user 24 fra
mecount 0 \
gui_layout 0 height 0 ir_filter 0 is_buffering 0 is_mounted 0 is_recor
ding 0 \
is_writing 0 is_writing_buf 0 tc_cam0 0 tc_cam1 0 iso 100
lores_height 0 lores_width 0 \
pi_model 0 rec 0 sensor 0 sensor_mode 0 shutter_a 0 space_left 0 stora
ge_type 0 \
wb 5600 wb_user 5600 width 0 memory_alert 0 \
shutter_a_sync_mode 0 shutter_angle_nom 0 shutter_angle_actual 0 shutt
er_angle_transient 0 \
exposure_time 0 last_dng_cam1 0 last_dng_cam0 0 \
zoom 0 write_speed_to_drive 0 recording_time 0
```

(See the settings guide for the full list.)

Add alias

```
nano ~/.bashrc
```

Add to the end of the file:

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

Exit with Ctrl+x. System will ask you to save the file. Press "y" and then enter.

Reload .bashrc

```
source ~/.bashrc
```

Cinemate services

storage-automount

Mounts and unmounts removable drives such as SSDs, NVMe enclosures and the CFE HAT.

wifi-hotspot

Keeps a simple Wi-Fi hotspot running via NetworkManager so you can reach the web UI while in the field. 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 start  # starts the service
sudo make enable # makes the service start on boot
```

You can also start and enable the service individually, by entering their respective folders and issuing the `sudo make` command

Note that if you were connected to the Pi via wifi, this connection is now broken due to the Pi setting up its own hotspot.

To connect again, check your available wifi networks. There should now be a network available named CinePi. Connect to it using password `11111111`

Now you shuld be able to ssh to the Pi this command:

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

You should also be able to find the Pi by opening a terminal and typing:

```
arp -a
```

You will see something like

```
❯ arp -a

? (10.42.0.1) at e4:5f:1:a9:72:a7 on en0 ifscope [ethernet]
...
```

During development/building your rig you might prefer the Pi to use your normal Wi-Fi instead of its own hotspot so you remain online while tinkering.

Disable the hotspot by setting `system.wifi_hotspot.enabled` to `false` `settings.json` *and* by stopping the service with `sudo systemctl stop wifi-hotspot`.

To stop the hotspot from starting on boot, type `sudo systemctl disable wifi-hotspot`.

If you plug in an Ethernet cable you can keep the hotspot active while the wired connection provides internet access. See Hotspot logic for more details on how the hotspot works.

Connect to the Pi (if not already connected):

```
ssh pi@10.42.0.1

# password: 1
```

Starting Cinemate

Now, back on the Pi, anywhere in the terminal, type:

```
cinemate
```

Make sure things are running smoothly and then you can move on to enabling the cinemate-autostart service:

cinemate-autostart.service

```
cd /home/pi/cinemate/

sudo make install   # copy service file
sudo make enable    # start on boot
make start          # launch now
```

After enabling the service, Cinemate should autostart on boot.

You now have a 12 bit RAW image capturing system on your Raspberry Pi!

8.1.3 Todo

- simple_gui.py adaptive layout for non 1920x1080 screens
- 16 bit modes for imx 585
- support for imx 294
- overclocking of ISP
- optional auto-exposure
- hardware sync of sensor frame capture, perhaps via a pico
- rendering mode, for creating proxy files in camera (using https://github.com/mrjulesfletcher/dng_to_video)
- automatic detection of attached sensor and dynamic dtoverlay

8.2 Acknowledgements

The **Cinemate** stack is built on top of several open-source projects. Special thanks to all authors!

- **CinePi-raw** – Csaba Nagy
- **IMX 585 and IMX 283 drivers** – Will Whang
- **libcamera** – Ideas on board
- **cpp-mjpeg-streamer** – Nadjieb Mohammadi
- **lgpio** – Joan

Also thanks to Simon at Altcinecam for support and assistance!

Get your sensors and CFE Hats here: https://www.tindie.com/stores/will 123321/