Welcome

Thank you for choosing Freenove products!

How to Start

Please follow this tutorial to set up the screen.

Get Support

Encounter problems? Don't worry! Refer to “TroubleShooting.pdf” or contact us.

When there are packaging damage, quality problems, questions encountering in use, etc., just send us an email. We will reply to you within one working day and provide a solution.

support@freenove.com

About

Freenove provides open source electronic products and services.

Freenove is committed to helping customers learn programming and electronic knowledge, quickly implement product prototypes, realize their creativity and launch innovative products. Our services include:

- Kits for learning programming and electronics
- Kits compatible with Arduino®, Raspberry Pi®, micro:bit®, etc.
- Kits for robots, smart cars, drones, etc.
- Components, modules and tools
- Design and customization

To learn more about us or get our latest information, please visit our website:

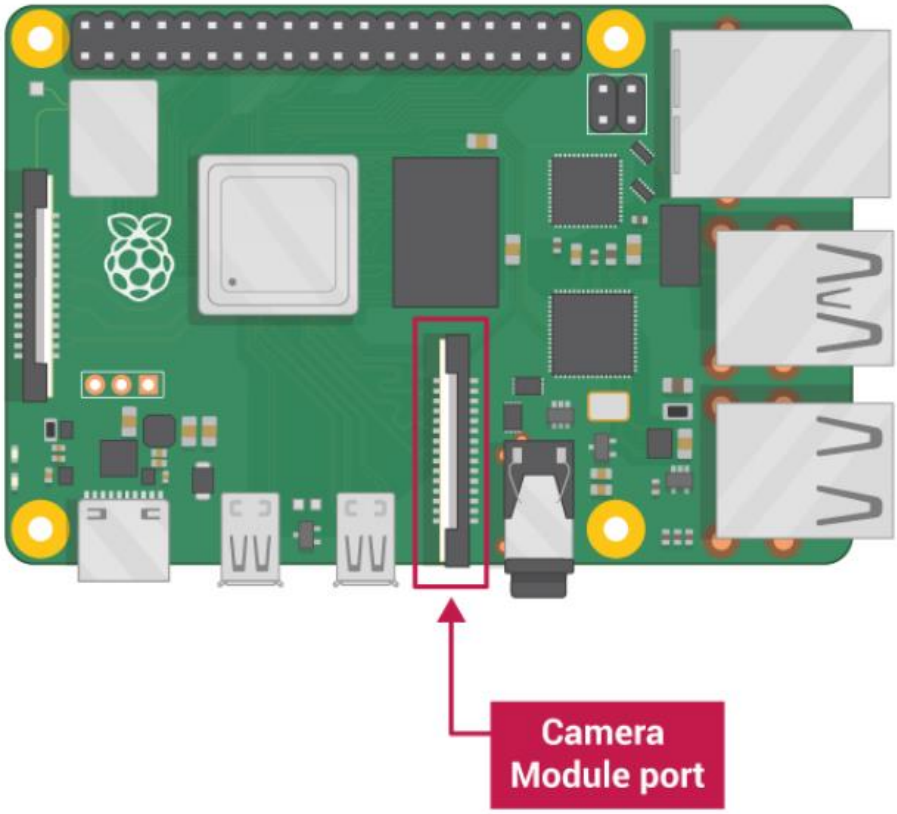
http://www.freenove.com

Copyright

Step 1 What you will need

Raspberry Pi computer with a Camera Module port

All current models of Raspberry Pi have a port for connecting the Camera Module.



Note: If you want to use a Raspberry Pi Zero, you need a Camera Module ribbon cable that ?ts the Raspberry Pi Zero's smaller Camera Module port.

Raspberry Pi Camera Module

There are two versions of the Camera Module:

The standard version (https://www.raspberrypi.org/products/camera-module-v2/**),** which is designed to

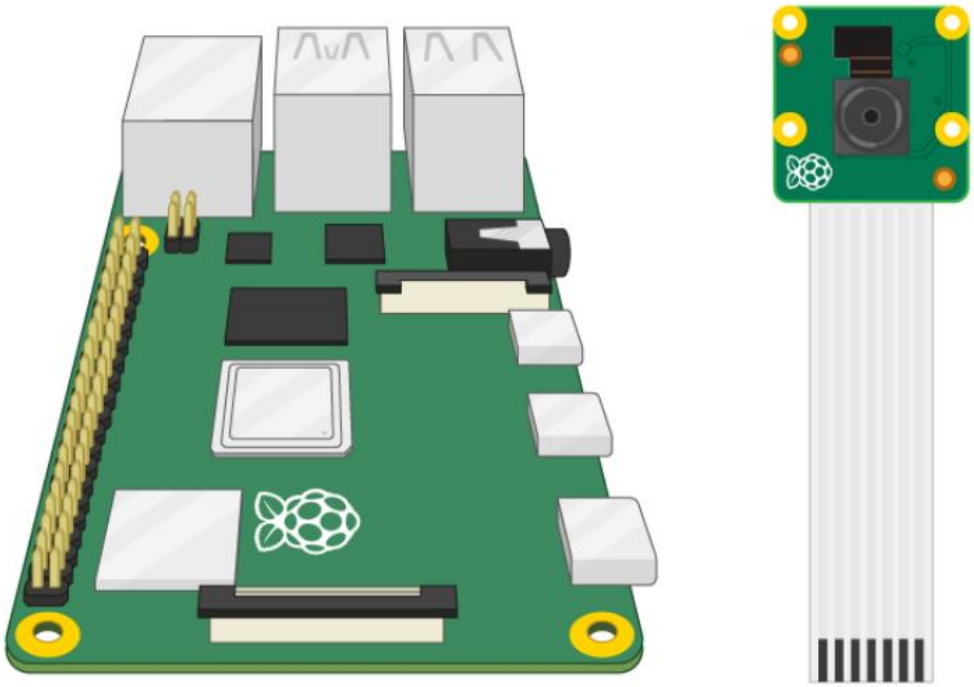
take pictures in normal light

The NoIR version (https://www.raspberrypi.org/products/pi-noir-camera-v2/**),** which doesn't have an infrared ?lter, so you can use it together with an infrared light source to take pictures in the dark.

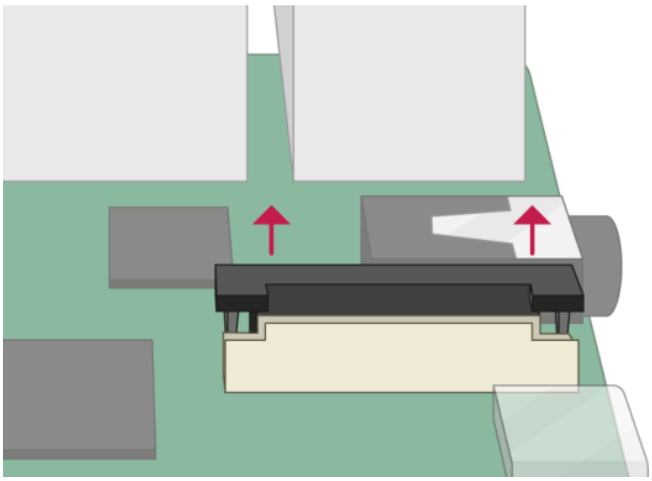
Step 2 Connect and install the Camera Module

Ensure your Raspberry Pi is turned off..

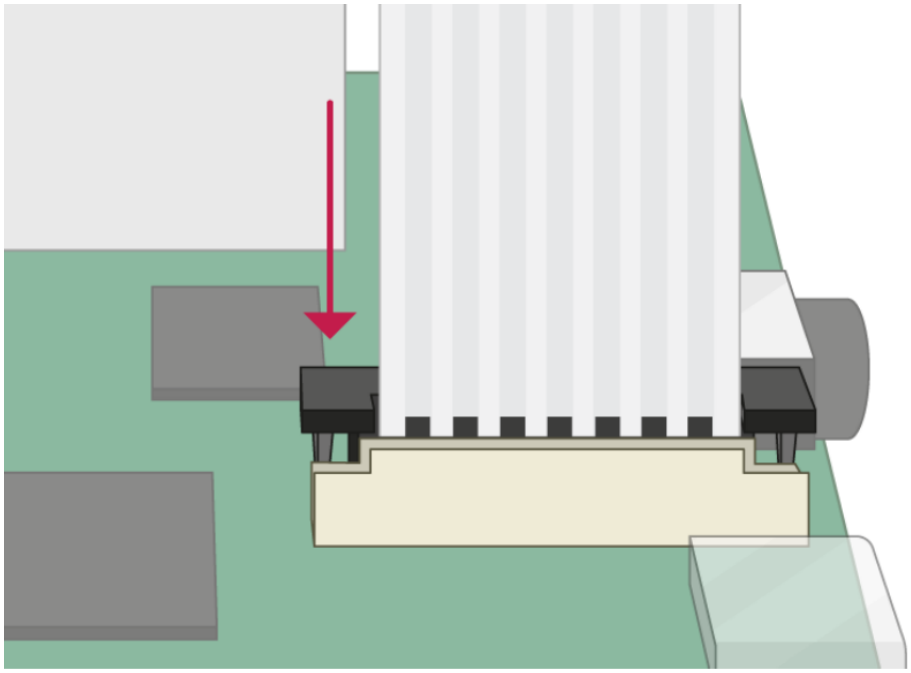
1. Locate the Camera Module port.



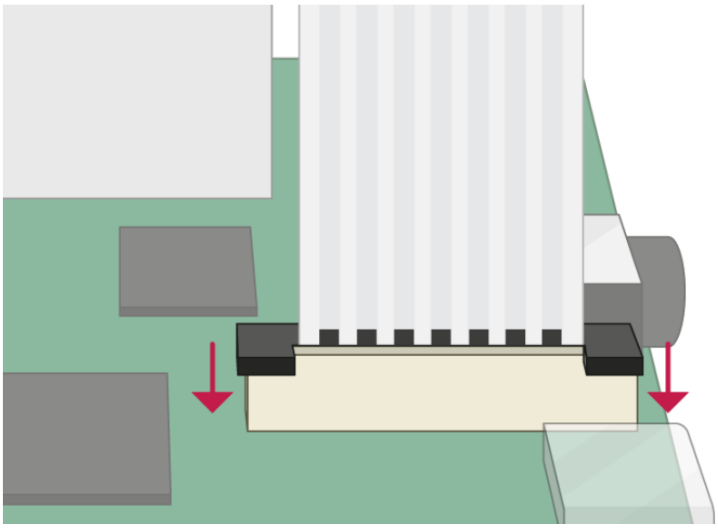
2. Gently pull up on the edges of the port's plastic clip.

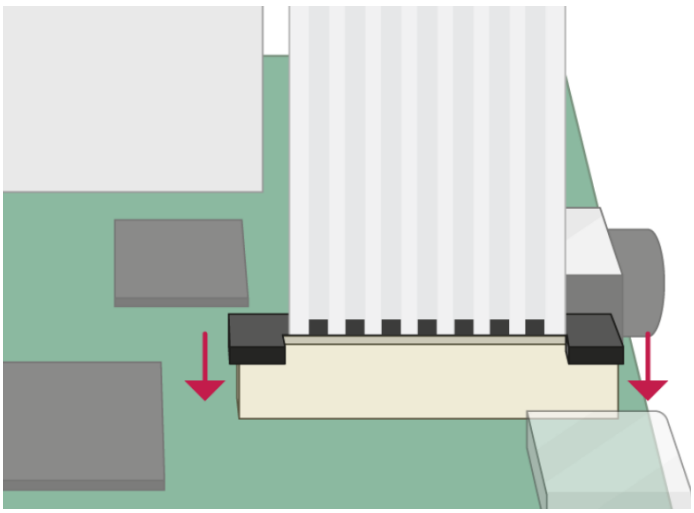


3. Insert the Camera Module ribbon cable; make sure the connectors at the bottom of the ribbon cable are facing the contacts in the port.

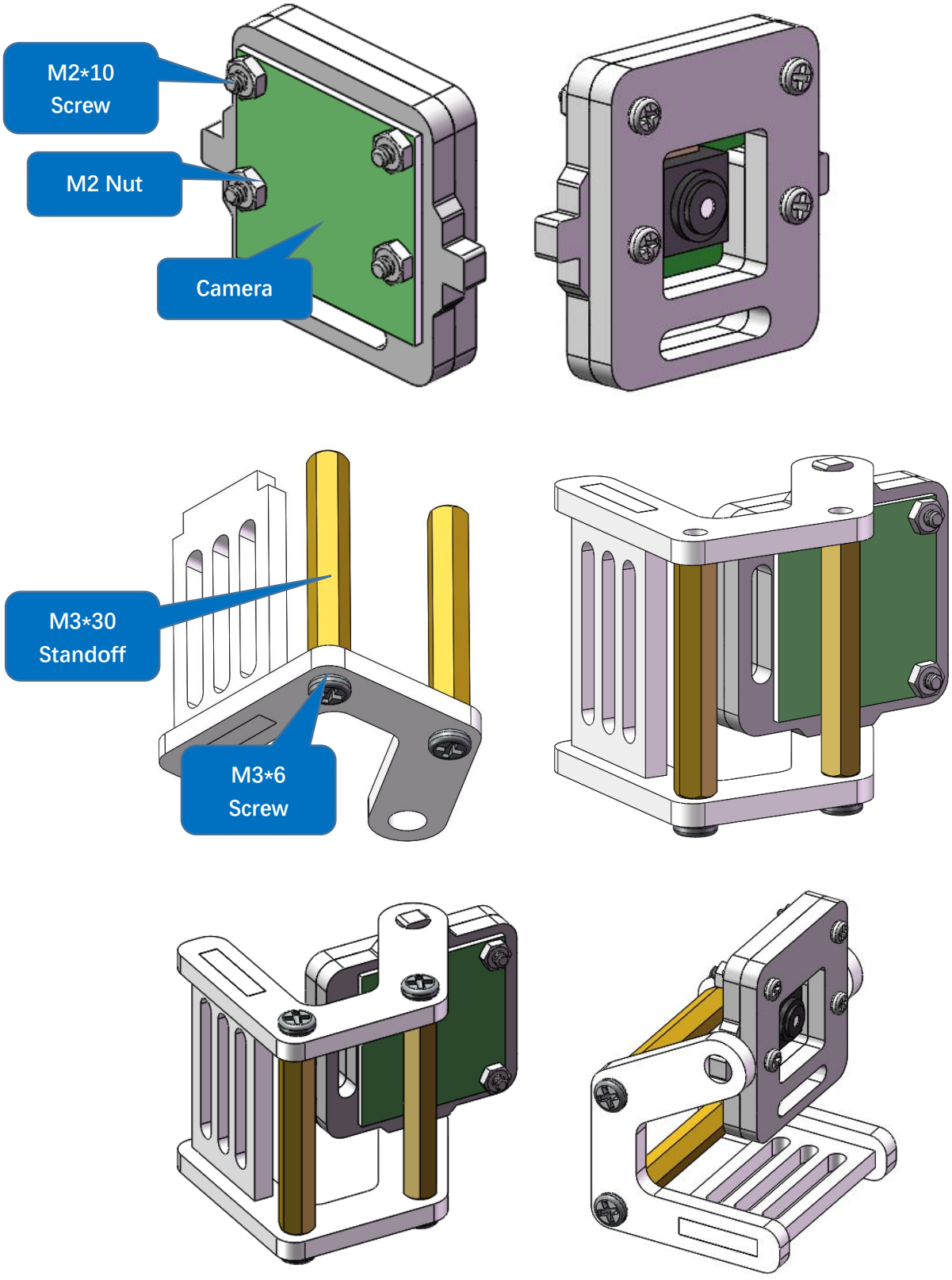


4. Push the plastic clip back into place.





Assemble the camera support



Start up your Raspberry Pi.

Step 3 How to control the Camera Module via the command line

The Python Picamera2 module currently uses the latest version of the Raspberry PI operating system (called Bullseye) by default.

Picamera2 is designed for systems running either Raspberry Pi OS version or Raspberry Pi OS Lite, using a Bullseye or later image. It is pre-installed in current images downloaded from the Raspberry Pi website, or can be installed using the Raspberry Pi Imager tool. Users with older images should consider updating them or proceed to the installation instructions.

Picamera2 can operate in a headless manner, not requiring an attached screen or keyboard. When first setting up such a system we would recommend attaching a keyboard and screen if possible, as it can make trouble-shooting easier. Raspberry Pi OS Bullseye and later images by default run the libcamera camera stack, which is required for Picamera2.

Open a terminal window by clicking the black monitor icon in the taskbar:

You can check that libcamera is working by opening a command window and typing:

```
libcamera-hello
```

You should see a camera preview window for about five seconds. If you do not, please refer to the Raspberry Pi camera documentation.

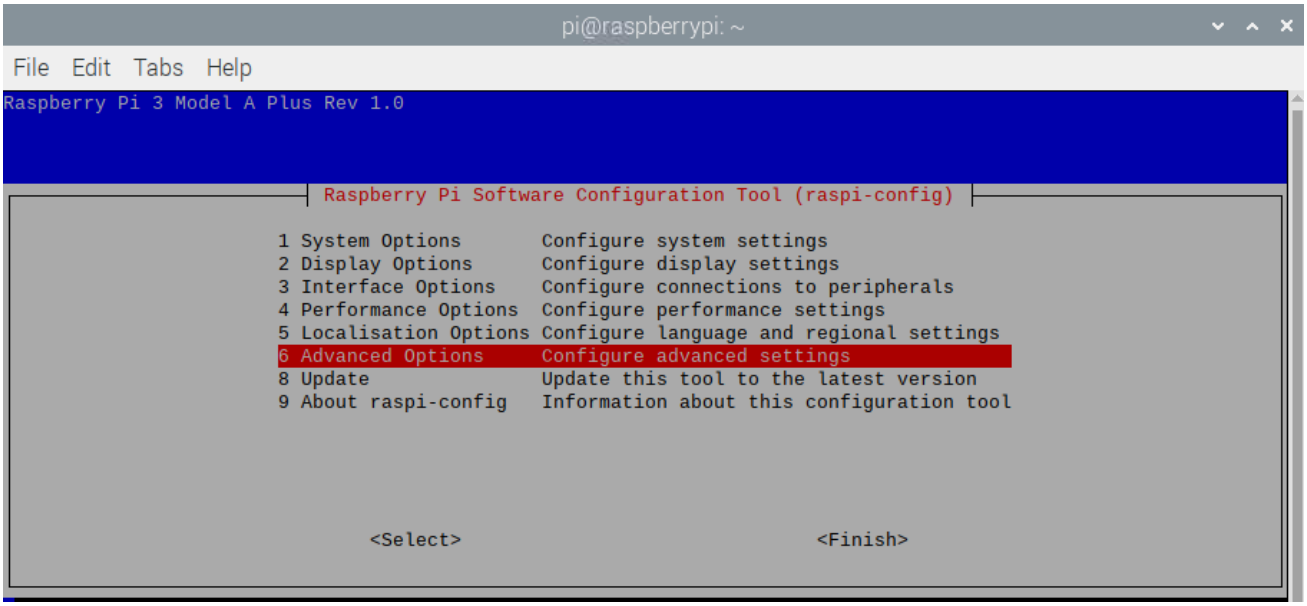
If you have any concerns, please contact us via email:support@freenove.com

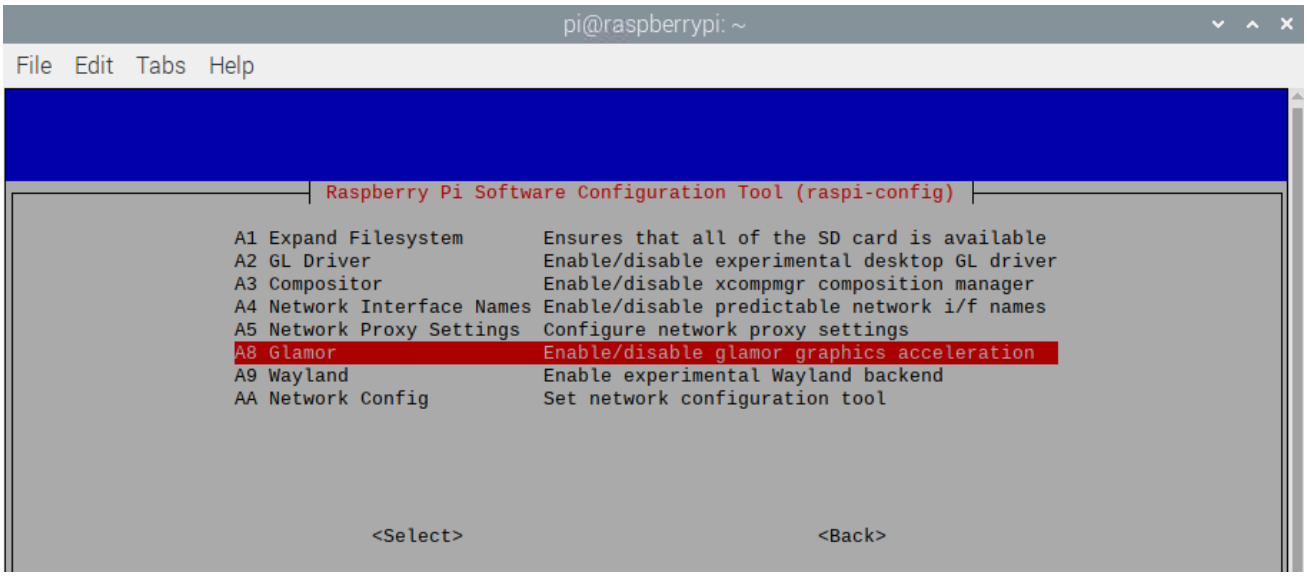
libcamera-apps does not work properly on Pi 0 to 3 devices when running the latest Bullseye images.

A workaround is to open a terminal, run "**sudo raspi-config**", navigate to "**Advanced Options**" and enable "**Glamor**" graphic acceleration. Then reboot your Pi.

```
sudo raspi-config
```

```
pi@raspberrypi:~ $ sudo raspi-config
```

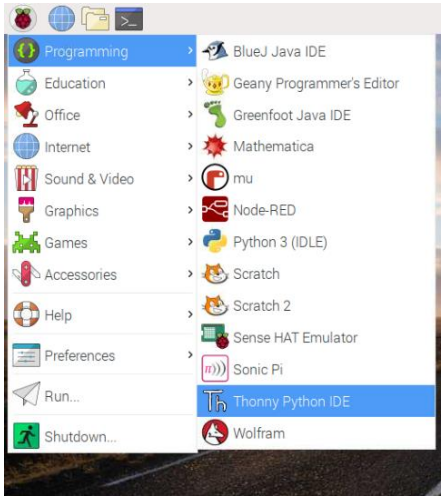




reboot your Pi.

Step 4 How to control the Camera Module with Python code

The Python picamera library allows you to control your Camera Module and create amazing projects.
Open a Python 3 editor, such as **Thonny Python IDE**:



Open a new file and save it as camera.py.
Note: it's important that you never save the file as picamera2.py.
Enter the following code:

```
1  from picamera2 import Picamera2, Preview
2  import time
3  picam2 = Picamera2()
4  camera_config = picam2.create_preview_configuration()
5  picam2.configure(camera_config)
6  picam2.start_preview(Preview.QTGL)
7  picam2.start()
8  time.sleep(2)
9  picam2.capture_file("test.jpg")
10 picam2.close()
```

The following script will:
1. Open the camera system
2. Generate a camera configuration suitable for preview
3. Configure the camera system with that preview configuration
4. Start the preview window
5. Start the camera running
6. Wait for two seconds and capture a JPEG file (still in the preview resolution)

NOTE:
Users of Raspberry Pi 3 or earlier devices will need to enable Glamor in order for this example script using X Windows to work. To do this, run sudo raspi-config in a command window, choose Advanced Options and then enable Glamor graphic acceleration. Finally reboot your device.

In addition, you can set Preview window parameters,
The preview implementation takes exactly the same parameters:
•x - the x-offset of the preview window
•y - the y-offset of the preview window
•width - the width of the preview window
•height - the height of the preview window
•transform - a transform that allows the camera image to be horizontally and/or vertically flipped on the display
All the parameters are optional, and default values will be chosen if omitted. The following example will place an
800x600 pixel preview window at (100, 200) on the display, and will horizontally mirror the camera preview image:
Enter the following code:

```
from picamera2 import Picamera2, Preview
from libcamera import Transform
picam2 = Picamera2()
picam2.start_preview(Preview.QTGL, x=100, y=200, width=800, height=600,
transform=Transform(hflip=1))
picam2.start()
```

The supported transforms are:
•Transform() - the identity transform, which is the default
•Transform(hflip=1) - horizontal flip
•Transform(vflip=1) - vertical flip
•Transform(hflip=1, vflip=1) - horizontal and vertical flip (equivalent to a 180 degree rotation)
It's important to realise that the display transform discussed here does not have any effect on the actual images received from the camera. It only applies the requested transform as it renders the pixels onto the screen. We'll encounter camera transforms again when it comes actually to transforming the images as the camera delivers them.Please also note that in the example above, the **start_preview()** function must be called before the call to picam2.start() .Finally, if the camera images have a different aspect ratio to the preview window, they will be letter- or pillar-boxed to fit, preserving the image's proper aspect ratio.

Next use the camera to capture videos:
In Picamera2, the process of capturing and encoding video is largely automatic. The application only has to define what encoder it wants to use to compress the image data, and how it wants to output this compressed data stream.
The mechanics of taking the camera images that arrive, forwarding them to an encoder, which in turn sends the results directly to the requested output, is entirely transparent to the user. The encoding and output all happens in a separate thread from the camera handling to minimise the risk of dropping camera frames.
Here is a first example of capturing a ten-second video.
Enter the following code:

```
from picamera2.encoders import H264Encoder
from picamera2 import Picamera2
import time
picam2 = Picamera2()
```

```
video_config = picam2.create_video_configuration()
picam2.configure(video_config)
encoder = H264Encoder(bitrate=10000000)
output = "test.h264"
picam2.start_recording(encoder, output)
time.sleep(10)
picam2.stop_recording()
```

In this example we use the H.264 encoder. For the output object we can just use a string for convenience; this will be interpreted as a simple output file. For configuring the camera, the create_video_configuration is a good starting point, as it will use a larger buffer_count to reduce the risk of dropping frames.

We also used the convenient start_recording and stop_recording functions, which start and stop both the encoder and the camera together. Sometimes it can be useful to separate these two operations, for example you might want to start and stop a recording multiple times while leaving the camera running throughout. For this reason, start_recording could have been replaced by:

```
encoder.output = 'test.h264'
picam2.start_encoder(encoder)
picam2.start()
```

and stop_recording by:

```
picam2.stop()
picam2.stop_encoder()
```

All the video encoders can be constructed with parameters that determine the quality (amount of compression) of the output, such as the bitrate for the H.264 encoder. For those not so familiar with the details of these encoders, these parameters can also be omitted in favour of supplying a quality to the start_encoder or start_recording functions.

Thepermitted quality parameters are:

•Quality.VERY_LOW

•Quality.LOW

•Quality.MEDIUM - this is the default for both functions if the parameter is not specified

•Quality.HIGH

•Quality.VERY_HIGH

This quality parameter only has any effect if the encoder was not passed explicit codec-specific parameters. It could be used like this:

```
import time
from picamera2.encoders import H264Encoder, Quality
from picamera2 import Picamera2
picam2 = Picamera2()
video_config = picam2.create_video_configuration()
picam2.configure(video_config)
encoder = H264Encoder()

picam2.start_recording(encoder, 'low.h264', quality=Quality.VERY_LOW)
print(encoder._bitrate)
time.sleep(5)
picam2.stop_recording()
```

```
13
14  picam2.start_recording(encoder, 'medium.h264', quality=Quality.MEDIUM)
15  print(encoder._bitrate)
16  time.sleep(5)
17  picam2.stop_recording()
18
19  picam2.start_recording(encoder, 'high.h264', quality=Quality.VERY_HIGH)
20  print(encoder._bitrate)
21  time.sleep(5)
22  picam2.stop_recording()
```

You can also check out the official documentation to learn more about picamera2:
https://datasheets.raspberrypi.com/camera/picamera2-manual.pdf
There are so many interesting examples for you to learn in picamera2.
The official library link is as follows:
https://github.com/raspberrypi/picamera2