🫐 Projects

Getting started with picamera

Get started with the Raspberry Pi camera module, using Python and picamera

`Electronic components` `Pi Camera Module`
`Python`

Step 1   What you will make

Get started with the Raspberry Pi Camera Module, using Python and picamera. You'll take still pictures, record video, and apply image effects.

What you will learn

By following the 'Getting started with picamera' resource, you will learn:

- How to connect the Camera Module to the Raspberry Pi
- How to use Python to control the Camera Module
- How to use `start_preview()` and `stop_preview()` to control the camera preview
- How to take still pictures with `capture()`
- How to record video with `start_recording()` and `stop_recording()`
- How to play back video with `omxplayer`
- How to alter the brightness and contrast
- How to apply image effects and exposure modes

This resource covers elements from the following strands of the **Raspberry Pi Digital Making Curriculum (https://www.raspberrypi.or g/curriculum/)**:

- **Use basic programming constructs to create simple programs (https://www.raspberrypi.org/curriculum/programming/creator)**
- **Use basic digital, analogue, and electromechanical components (https://www.raspberrypi.org/curriculum/physical-computing/creator)**

Step 2	**What you will need**

Hardware

- Raspberry Pi Camera Module

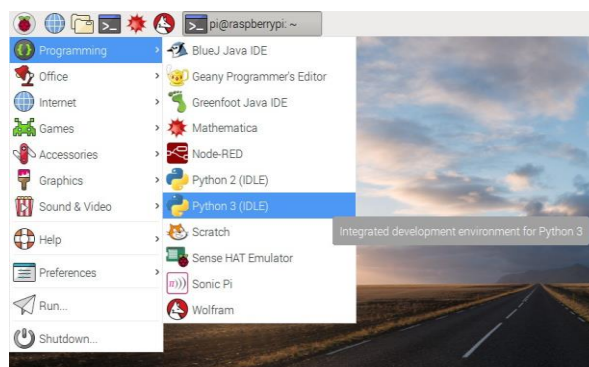| Step 3	**Getting started with picamera**

The Camera Module is a great accessory for the Raspberry Pi, allowing users to take still pictures and record video in full HD.

Step 4	**Camera preview**

Now your camera is connected and the software is enabled, you can get started by trying out the camera preview.

- Open **Python 3** from the main menu:

- Open a new file and save it as `camera.py`. It's important that you **do not** save it as `picamera.py`.

- Enter the following code:

```
from picamera import PiCamera
from time import sleep

camera = PiCamera()

camera.start_preview()
sleep(10)
camera.stop_preview()
```

- Save with **Ctrl + S** and run with **F5**. The camera preview should be shown for 10 seconds, and then close. Move the camera around to preview what the camera sees.

  The live camera preview should fill the screen like so:

  Note that the camera preview only works when a monitor is connected to the Pi, so remote access (such as SSH and VNC) will not allow you to see the camera preview

- If your preview was upside-down, you can rotate it with the following code:

```
camera.rotation = 180
camera.start_preview()
sleep(10)
camera.stop_preview()
```

You can rotate the image by 90, 180, or 270 degrees, or you can set it to 0 to reset.

- You can alter the transparency of the camera preview by setting an alpha level:

```
from picamera import PiCamera
from time import sleep

camera = PiCamera()

camera.start_preview(alpha=200)
sleep(10)
camera.stop_preview()
```

alpha can be any value between 0 and 255.

Step 5   Still pictures

The most common use for the Camera Module is taking still pictures.

- Amend your code to reduce the sleep and add a camera.capture() line:

```
camera.start_preview()
sleep(5)
camera.capture('/home/pi/Desktop/image.jpg')
camera.stop_preview()
```

It's important to sleep for at least 2 seconds before capturing, to give the sensor time to set its light levels.

- Run the code and you'll see the camera preview open for 5 seconds before capturing a still picture. You'll see the preview adjust to a different resolution momentarily as the picture is taken.

- You'll see your photo on the Desktop. Double-click the file icon to open it:
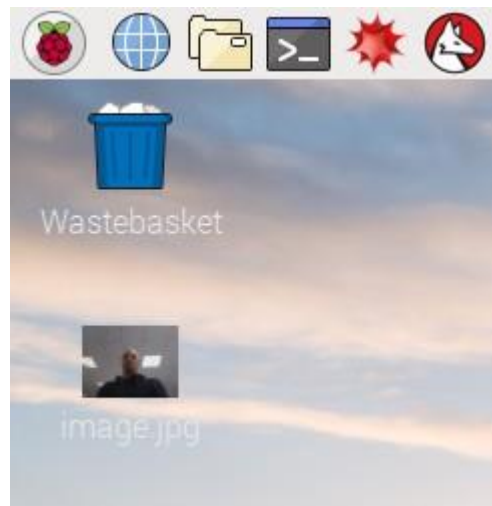
- Now try adding a loop to take five pictures in a row:

```python
camera.start_preview()
for i in range(5):
    sleep(5)
    camera.capture('/home/pi/Desktop/image%s.jpg' % i)
camera.stop_preview()
```

The variable `i` contains the current iteration number, from `0` to `4`, so the images will be saved as `image0.jpg`, `image1.jpg`, and so on.

- Run the code again and hold the camera in position. It will take one picture every five seconds.

- Once the fifth picture is taken, the preview will close. Now look at the images on your Desktop and you'll see five new pictures.

Step 6   Recording video

Now you've used the camera to take still pictures, you can move on to recording video.

- Amend your code to replace `capture()` with `start_recording()` and `stop_recording()`:

```python
camera.start_preview()
camera.start_recording('/home/pi/video.h264')
sleep(10)
camera.stop_recording()
camera.stop_preview()
```

- Run the code; it will record 10 seconds of video and then close the preview.

- To play the video, you'll need to open a terminal window by clicking the black monitor icon in the taskbar:

- Type the following command and press **Enter** to play the video:

```
omxplayer video.h264
```
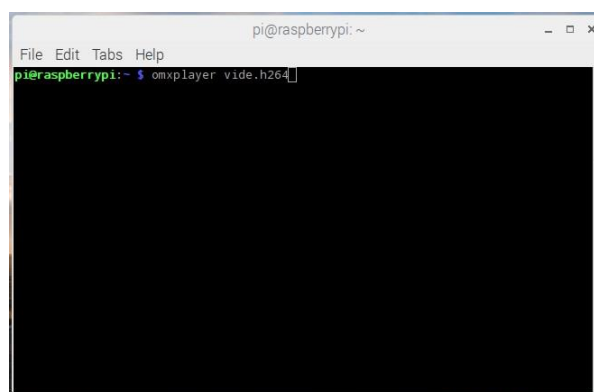

- The video should play. It may actually play at a faster speed than what has been recorded, due to `omxplayer`'s fast frame rate.

Step 7	**Effects**

At the beginning, you created a `camera` object with `camera = PiCamera()`. You can manipulate this `camera` object in order to configure its settings. The camera software provides a number of effects and other configurations you can apply. Some only apply to the preview and not the capture, others apply to the capture only, but many affect both.

- The resolution of the capture is configurable. By default it's set to the resolution of your monitor, but the maximum resolution is 2592 x 1944 for still photos and 1920 x 1080 for video recording. Try the following example to set the resolution to max. Note that you'll also need to set the frame rate to `15` to enable this maximum resolution:

```python
camera.resolution = (2592, 1944)
camera.framerate = 15
camera.start_preview()
sleep(5)
camera.capture('/home/pi/Desktop/max.jpg')
camera.stop_preview()
```

- The minimum resolution allowed is 64 x 64. Try taking one at that resolution.

- You can easily add text to your image with `annotate_text`. Try it:

```python
camera.start_preview()
camera.annotate_text = "Hello world!"
sleep(5)
camera.capture('/home/pi/Desktop/text.jpg')
camera.stop_preview()
```

- You can alter the brightness setting, which can be set from `0` to `100`. The default is `50`. Try setting it to another value:

```python
camera.start_preview()
camera.brightness = 70
sleep(5)
camera.capture('/home/pi/Desktop/bright.jpg')
camera.stop_preview()
```

- Try adjusting the brightness in a loop, and annotating the display with the current brightness level:

```python
camera.start_preview()
for i in range(100):
    camera.annotate_text = "Brightness: %s" % i
    camera.brightness = i
    sleep(0.1)
camera.stop_preview()
```

- Similarly, try the same for the contrast:

```python
camera.start_preview()
for i in range(100):
    camera.annotate_text = "Contrast: %s" % i
    camera.contrast = i
    sleep(0.1)
camera.stop_preview()
```

- You can set the annotation text size with the following code:

```python
camera.annotate_text_size = 50
```

Valid sizes are `6` to `160`. The default is `32`.

- You can also alter the annotation colours. First of all, ensure that `Color` is imported by amending your `import` line at the top:

```python
from picamera import PiCamera, Color
```

Then amend the rest of your code as follows:

```python
camera.start_preview()
camera.annotate_background = Color('blue')
camera.annotate_foreground = Color('yellow')
camera.annotate_text = " Hello world "
sleep(5)
camera.stop_preview()
```
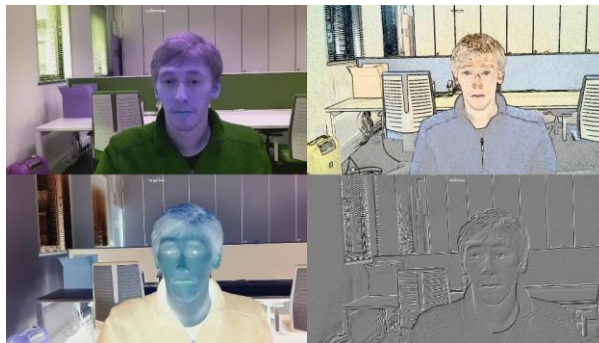
- You can use `camera.image_effect` to apply a particular image effect. The options are: `none`, `negative`, `solarize`, `sketch`, `denoise`, `emboss`, `oilpaint`, `hatch`, `gpen`, `pastel`, `watercolor`, `film`, `blur`, `saturation`, `colorswap`, `washedout`, `posterise`, `colorpoint`, `colorbalance`, `cartoon`, `deinterlace1`, and `deinterlace2`. The default is `none`. Pick one and try it out:

```python
camera.start_preview()
camera.image_effect = 'colorswap'
sleep(5)
camera.capture('/home/pi/Desktop/colorswap.jpg')
camera.stop_preview()
```

- Try looping over the various image effects in a preview to test them out:

```python
camera.start_preview()
for effect in camera.IMAGE_EFFECTS:
    camera.image_effect = effect
    camera.annotate_text = "Effect: %s" % effect
    sleep(5)
camera.stop_preview()
```


- You can use `camera.awb_mode` to set the auto white balance to a preset mode to apply a particular effect. The options are: `off`, `auto`, `sunlight`, `cloudy`, `shade`, `tungsten`, `fluorescent`, `incandescent`, `flash`, and `horizon`. The default is `auto`. Pick one and try it out:

```
camera.start_preview()
camera.awb_mode = 'sunlight'
sleep(5)
camera.capture('/home/pi/Desktop/sunlight.jpg')
camera.stop_preview()
```

You can loop over the available auto white balance modes with `camera.AWB_MODES`.

- You can use `camera.exposure_mode` to set the exposure to a preset mode to apply a particular effect. The options are: `off`, `auto`, `night`, `nightpreview`, `backlight`, `spotlight`, `sports`, `snow`, `beach`, `verylong`, `fixedfps`, `antishake`, and `fireworks`. The default is `auto`. Pick one and try it out:

```
camera.start_preview()
camera.exposure_mode = 'beach'
sleep(5)
camera.capture('/home/pi/Desktop/beach.jpg')
camera.stop_preview()
```

You can loop over the available exposure modes with `camera.EXPOSURE_MODES`.

Step 8   What next?

Now you've got started with the Camera Module, you could try adding GPIO controls using **GPIO Zero (https://gpiozero.readthedocs.org/)**, integrate with Minecraft Pi, or even post your pictures to Twitter! Try some more camera resources:

* **Push button stop-motion (https://projects.raspberrypi.org/en/projects/push-button-stop-motion/)**
* **Minecraft photobooth (https://projects.raspberrypi.org/en/projects/minecraft-photobooth/)**
* **Tweeting Babbage (https://projects.raspberrypi.org/en/projects/tweeting-babbage/)**
* **Parent detector (https://projects.raspberrypi.org/en/projects/parent-detector/)**

There's also an infrared version of the camera (called Pi NoIR) which gives you everything the regular Camera Module offers, with one difference: it doesn't use an infrared filter. This gives you the ability to see in the dark with infrared lighting. See the **Infrared bird box (https://projects.raspberrypi.org/en/projects/infrared-bird-box/)** resource for making the most of the Pi NoIR camera.

Also, see the extensive **picamera documentation (https://picamera.readthedocs.org/)** for more information.