

# OpenCV 기초 사용법

김성영교수  
금오공과대학교  
컴퓨터공학과

## 학습 목표

---

- 디지털 영상의 구조 및 유형에 대해 설명할 수 있다.
- 컴퓨터비전과 영상처리를 구분하여 설명할 수 있다.
- 라즈베리파이용 카메라 모듈의 특징을 설명할 수 있다.
- 내부 명령어를 사용하여 정지 영상과 동영상을 촬영할 수 있다.
- Python을 사용하여 정지 영상과 동영상을 촬영할 수 있다.
- 센서 기반의 움직임 검출과 이메일 발송을 할 수 있다.
- OpenCV를 활용하여 간단한 영상 처리를 수행할 수 있다.

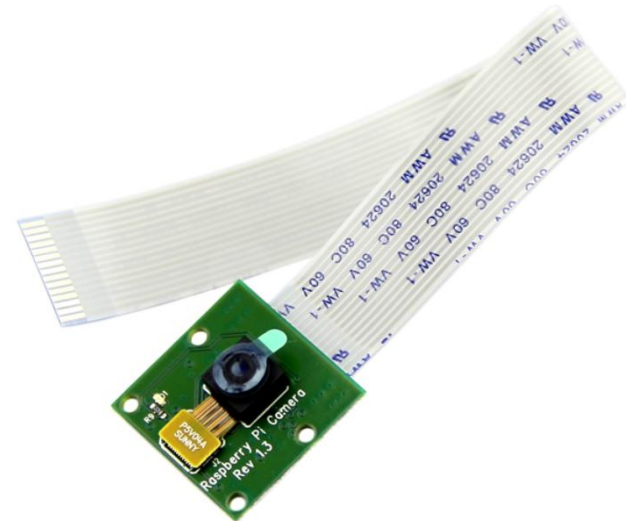
# 라즈베리 파이에 사용가능한 카메라

- RPi 카메라 모듈 또는 USB Webcam



## 라즈베리 파이 용 카메라 모듈 사양 (V1)

- 5MP sensor
  - 2592 x 1944 stills
  - 1080p30 및 720p60 video
- CSI(Camera Serial Interface) – 15pin
- 25 x 20 x 9 mm, 3g
- Raspberry Pi A+ 이후 모델과 호환



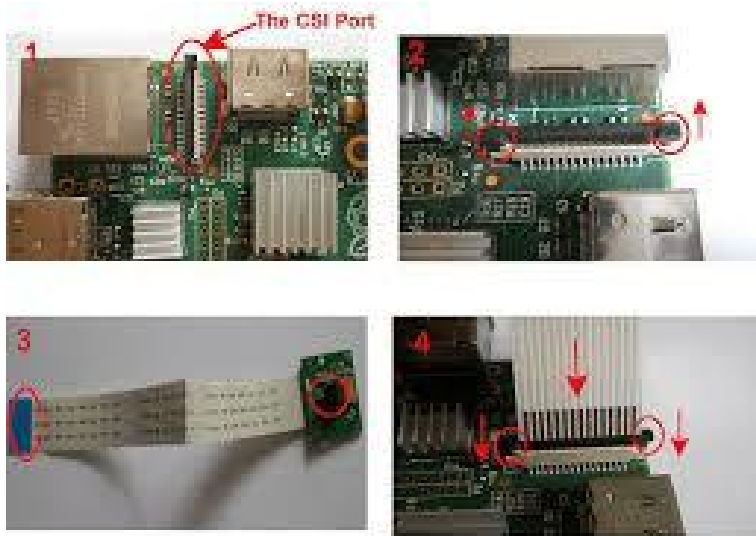
## 라즈베리 파이 용 카메라 모듈 사양 (V2)

---

- 8MP sensor
  - 3280x2464 stills
  - 1080p30 및 720p60 video
- 25 x 23 x 9 mm, 3g
- Raspberry Pi A+ 이후 모델과 호환

# 라즈베리 파이 용 카메라 연결

- CSI(Camera Serial Interface) 포트 사용
- 카메라의 파란색 면이 RPi의 USB 포트 방향으로 연결
  - CSI 커버가 부러지지 않도록 주의



# 라즈베리 파이의 비디오 처리

---

- GPU

- 3B+: VideoCore IV MP2 400MHz with OpenGL ES 1.1, 2.0 (24 GFLOPS)
- 4B: VideoCore IV 500MHz with OpenGL ES 1.1, 2.0, 3.0, 3.1

- 내부 코덱

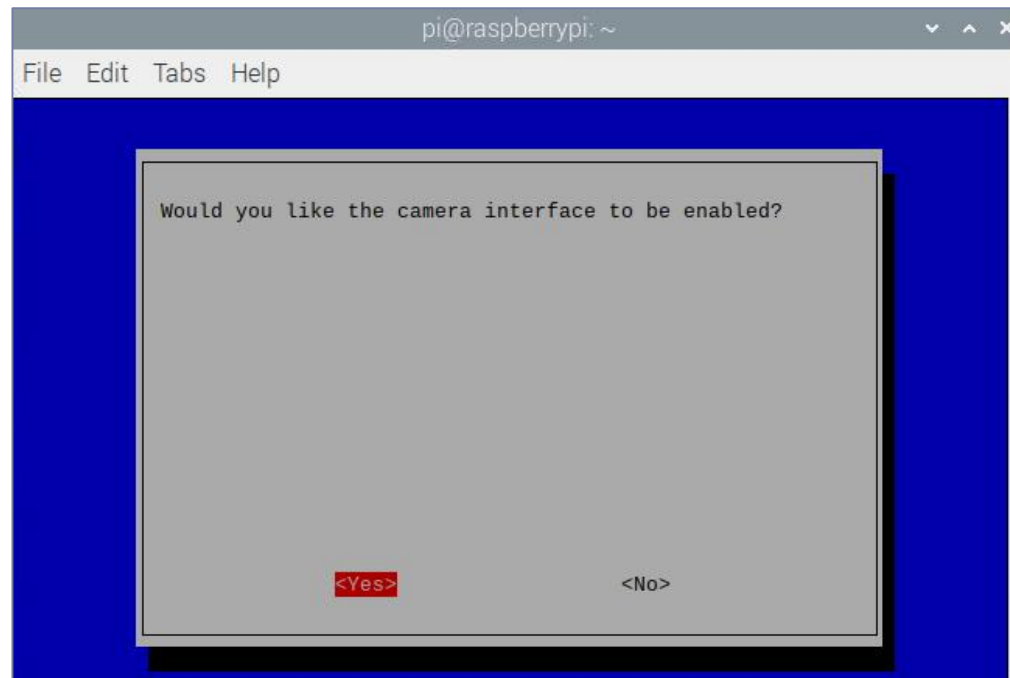
- 3B+: 라이선스 받은 MPEG-2 그리고 VC-1, 1080p30 h.264 / MPEG-4 AVC CODEC
- 4B: H.265 4Kp60, H.264 1080p60

- 카메라 모듈 접근

- MMAL 또는 V4L API
- 파이썬 기반의 Picamera

# 카메라 활성화

- \$ sudo raspi-config
- 5 Interfacing Options
  - P1 Camera
- 재부팅





# 라즈비안에서 사진/동영상 찍기

---

- 정지 영상 촬영

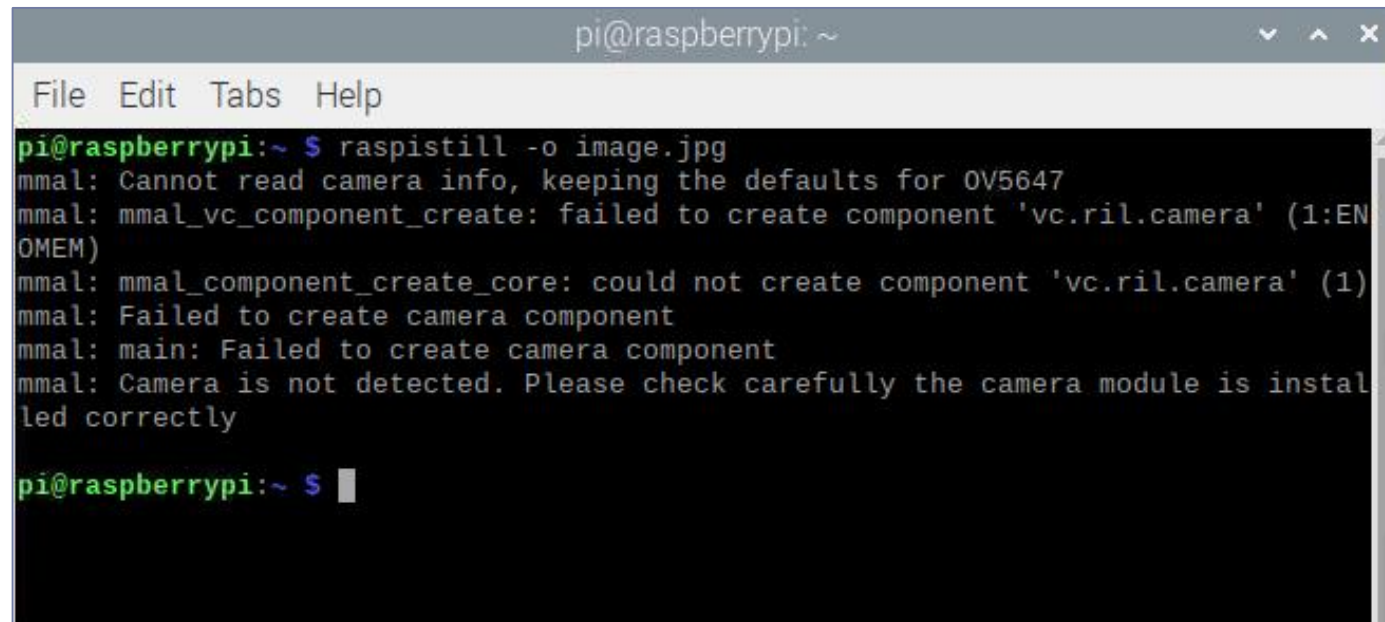
- 방법: `$ raspistill -o [파일명]`
- 예시: `$ raspistill -o image.jpg`
- 옵션 확인: `$ raspistill`

- 동영상 촬영 (기본 5초)

- 방법: `$ raspivid -o [파일명]`
- 예시: `$ raspivid -o video.h264`
- 옵션 확인: `$ raspivid`

## 카메라 연결 확인 (1)

- 다음 에러가 발생하는 경우 카메라 연결 재확인 필요
  - 부팅후에 다시 동작 확인

A terminal window titled 'pi@raspberrypi: ~' with a menu bar (File, Edit, Tabs, Help). The terminal output shows the command 'raspistill -o image.jpg' and several error messages from the mmal module. The errors indicate that the camera component 'vc.ril.camera' could not be created, likely due to memory (OMEM) issues. The final message states 'Camera is not detected. Please check carefully the camera module is installed correctly'. The prompt 'pi@raspberrypi:~ \$' is shown at the bottom with a cursor.

```
pi@raspberrypi:~ $ raspistill -o image.jpg
mmal: Cannot read camera info, keeping the defaults for OV5647
mmal: mmal_vc_component_create: failed to create component 'vc.ril.camera' (1:EN
OMEM)
mmal: mmal_component_create_core: could not create component 'vc.ril.camera' (1)
mmal: Failed to create camera component
mmal: main: Failed to create camera component
mmal: Camera is not detected. Please check carefully the camera module is instal
led correctly

pi@raspberrypi:~ $
```

## 카메라 연결 확인 (2)

---

- 카메라 연결 확인

`$ vcgencmd get_camera`

- 다음의 값이 출력되면 카메라 연결 상태

`supported=1 detected=1`

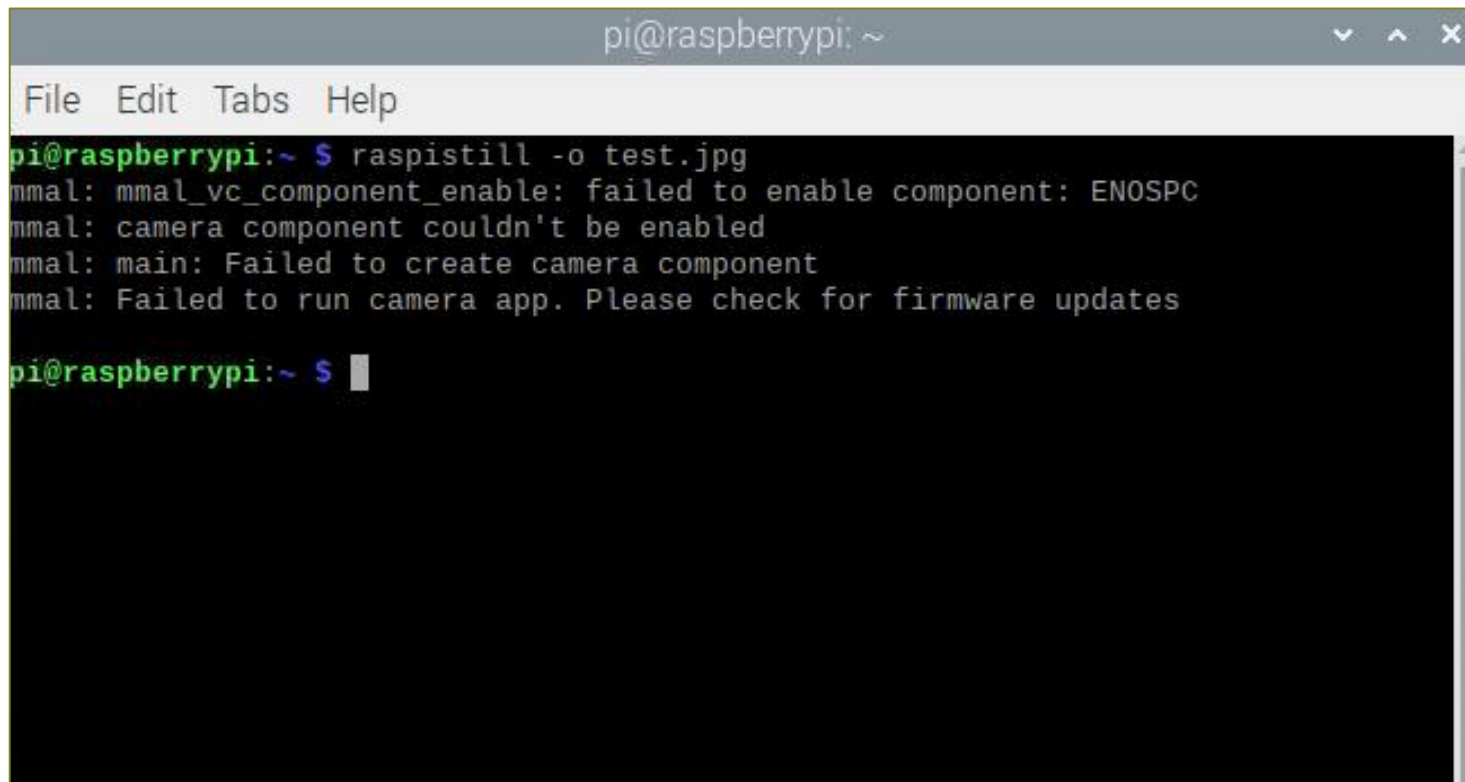
□ 카메라 연결 상태이지만 영상 촬영이 되지 않으면 H/W에 문제가 있을 가능성 있음

- 다음의 값이 출력되면 카메라 미연결 상태

`supported=1 detected=0`

## 카메라 연결 확인 (3)

- 다음 에러가 발생하는 경우 카메라가 이미 사용중임

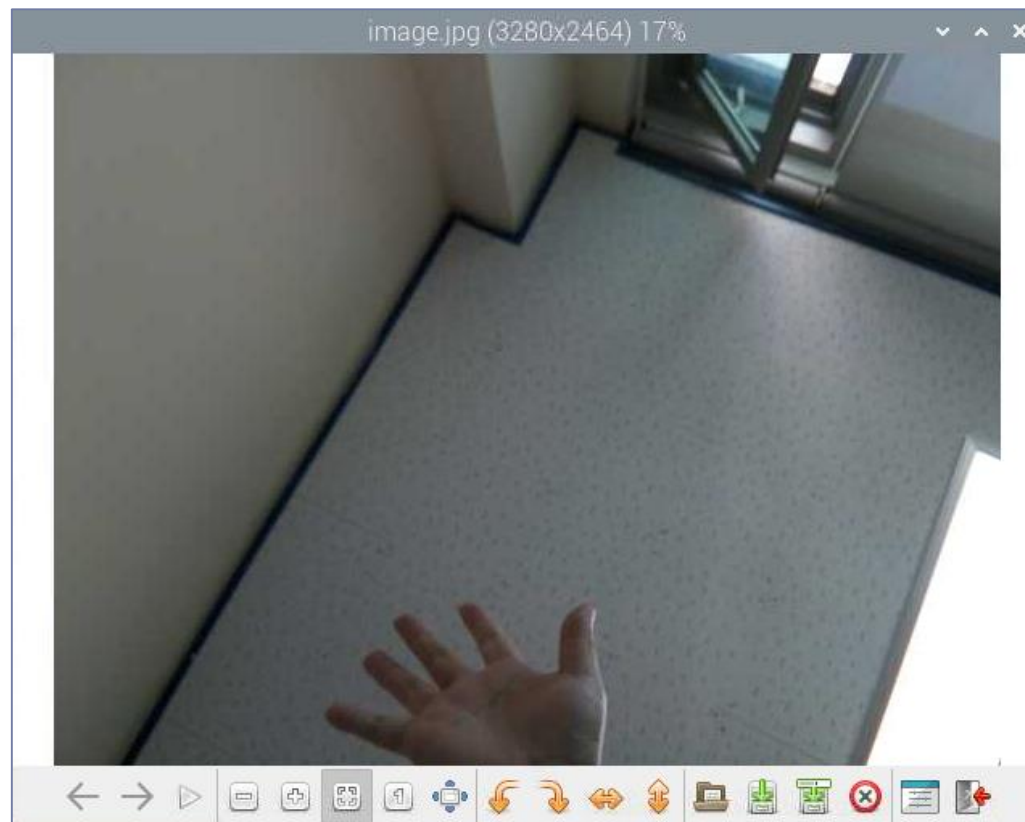
A terminal window titled 'pi@raspberrypi: ~' with a menu bar containing 'File', 'Edit', 'Tabs', and 'Help'. The terminal shows the command 'raspistill -o test.jpg' being executed. The output consists of four lines of error messages: 'mmal: mmal\_vc\_component\_enable: failed to enable component: ENOSPC', 'mmal: camera component couldn't be enabled', 'mmal: main: Failed to create camera component', and 'mmal: Failed to run camera app. Please check for firmware updates'. The prompt 'pi@raspberrypi:~ \$' is visible at the bottom of the terminal window.

```
pi@raspberrypi:~ $ raspistill -o test.jpg
mmal: mmal_vc_component_enable: failed to enable component: ENOSPC
mmal: camera component couldn't be enabled
mmal: main: Failed to create camera component
mmal: Failed to run camera app. Please check for firmware updates

pi@raspberrypi:~ $
```

## 정지 영상 확인

- 파일 탐색기를 사용하여 저장된 영상을 더블 클릭 ⇒ Image Viewer에서 확인 가능



# 라즈비안 동영상 재생

---

## ● 재생 순서

- 비디오 파일이 저장된 디렉토리로 이동
- `$ omxplayer video.h264`
  - 단, 원격접속 환경에서는 비디오 재생 불가
- 파일 탐색기를 사용하여 비디오 파일 재생
  - VLC media player: 원격접속 환경에서는 재생 속도가 느림

## ● 라즈비안의 샘플 비디오 재생

- `$ omxplayer /opt/vc/src/hello_pi/hello_video/test.h264`

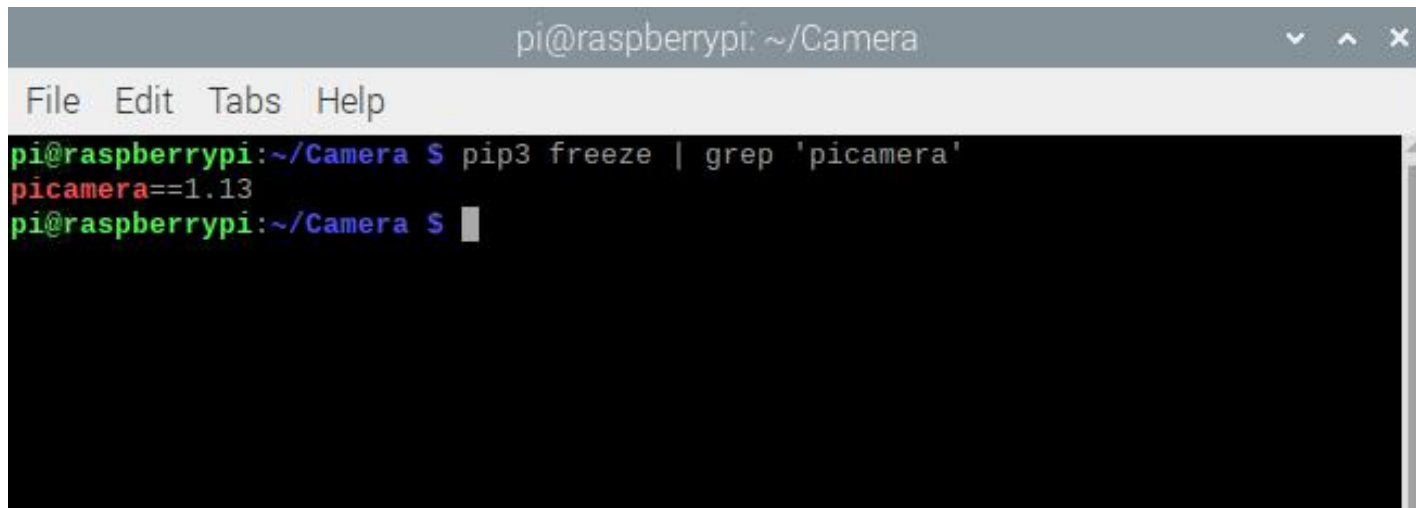
# python으로 카메라 제어

---

- picamera 설치 확인
- 정지 영상 촬영
- 동영상 촬영
- OpenCV 활용

## picamera 설치 확인

- [https://picamera.readthedocs.io/en/release-1.13/api\\_camera.html](https://picamera.readthedocs.io/en/release-1.13/api_camera.html)
- `$ pip3 freeze | grep 'picamera'`



```
pi@raspberrypi: ~/Camera
File Edit Tabs Help
pi@raspberrypi:~/Camera $ pip3 freeze | grep 'picamera'
picamera==1.13
pi@raspberrypi:~/Camera $
```



## 참조: picamera 설치 (필요한 경우만 사용)

---

- Python 2용 picamera library 설치
  - `$ pip install "picamera[array]"`
- Python 3용 picamera library 설치
  - `$ pip3 install "picamera[array]"`

# 정지 영상 촬영

image\_capture.py

```
import picamera
import time

camera = picamera.PiCamera()

camera.hflip = True
camera.vflip = True

camera.start_preview()
time.sleep(1)
camera.stop_preview()

camera.capture('image.jpg')
```

원격접속 환경에서는  
Preview 확인 불가

```
$ cd ~/OpenCVTest
$ python3 image_capture.py
```

~/OpenCVTest 디렉토리에 저장

## 카메라 설정과 초기값

---

```
camera.sharpness = 0
camera.contrast = 0
camera.brightness = 50
camera.saturation = 0
camera.ISO = 0
camera.video_stabilization = False
camera.exposure_compensation = 0
camera.exposure_mode = 'auto'
camera.meter_mode = 'average'
camera.awb_mode = 'auto'
camera.image_effect = 'none'
camera.color_effects = None
camera.rotation = 0
camera.hflip = False
camera.vflip = False
camera.crop = (0.0, 0.0, 1.0, 1.0)
```

## sleep()을 이용한 제어

```
import picamera
from time import sleep

camera = picamera.PiCamera()
camera.capture('image1.jpg')
sleep(2)
camera.capture('image2.jpg')
```

image\_capture2.py

```
import picamera
from time import sleep

camera = picamera.PiCamera()
camera.start_preview()
for n in range(100):
    camera.brightness = n
    sleep(0.2)
```

brightness.py


모니터에서 실행 결과 확인

# 동영상 촬영

```
import picamera
from time import sleep
camera = picamera.PiCamera()

camera.start_recording('video.h264')
sleep(5)
camera.stop_recording()
```

video\_capture.py



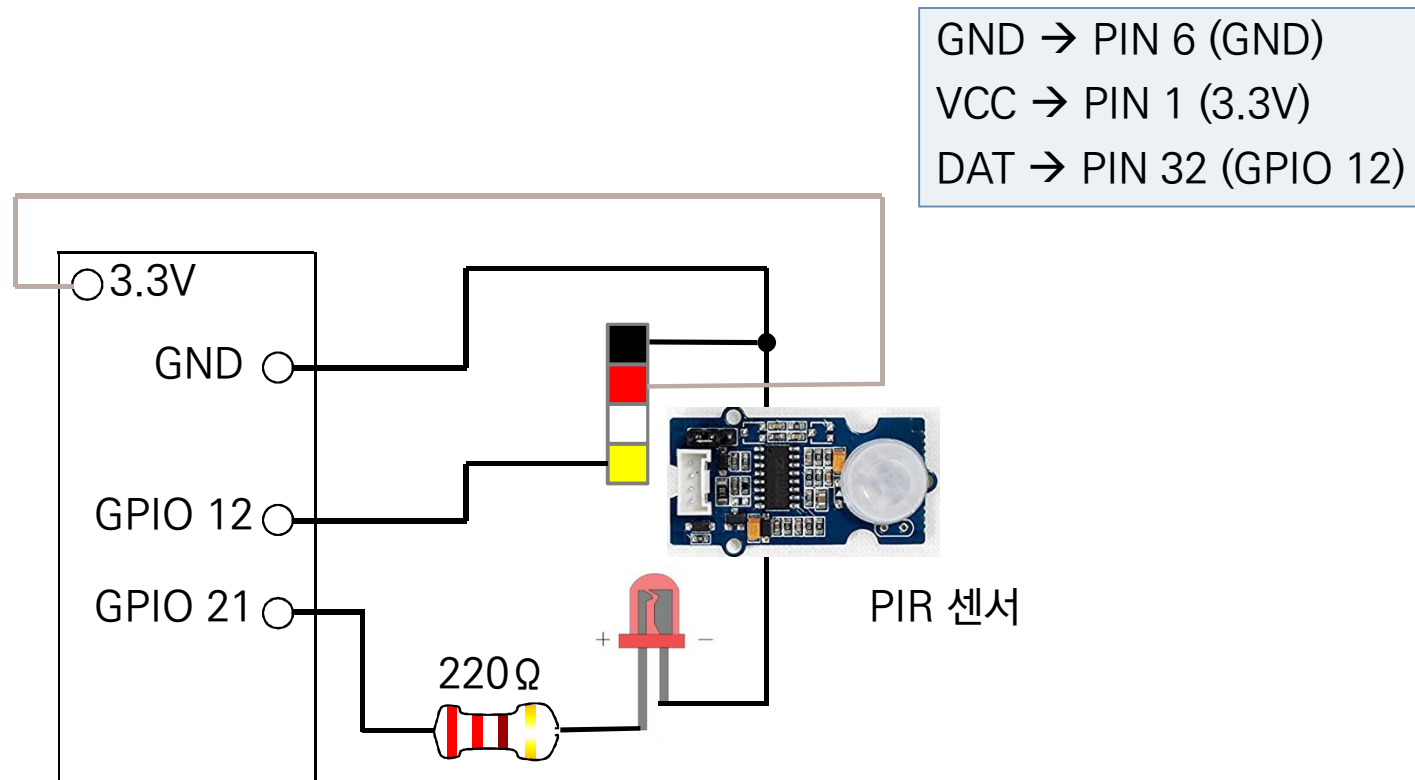
5초간 동영상을 촬영하여  
video.h264 파일에 저장

## 예제: 카메라 제어

---

- 움직임을 감지한 경우 사진 촬영
  - PIR 센서를 사용하여 움직임 감지  
(PIR 센서는 Grove Pi 없이 직접 라즈베리 파이에 연결)
  - 사진 촬영동안 LED 점등

# 움직임 검출 및 카메라 제어: 회로 구성



# 움직임 검출 및 카메라 제어: 소스 코드 (움직임 검출)

```
import RPi.GPIO as GPIO
import picamera
import time

GPIO.setmode(GPIO.BCM)

PIR = 12
RLED = 21

GPIO.setup(PIR, GPIO.IN)
GPIO.setup(RLED, GPIO.OUT)

camera = picamera.PiCamera()
```

pir\_camera.py

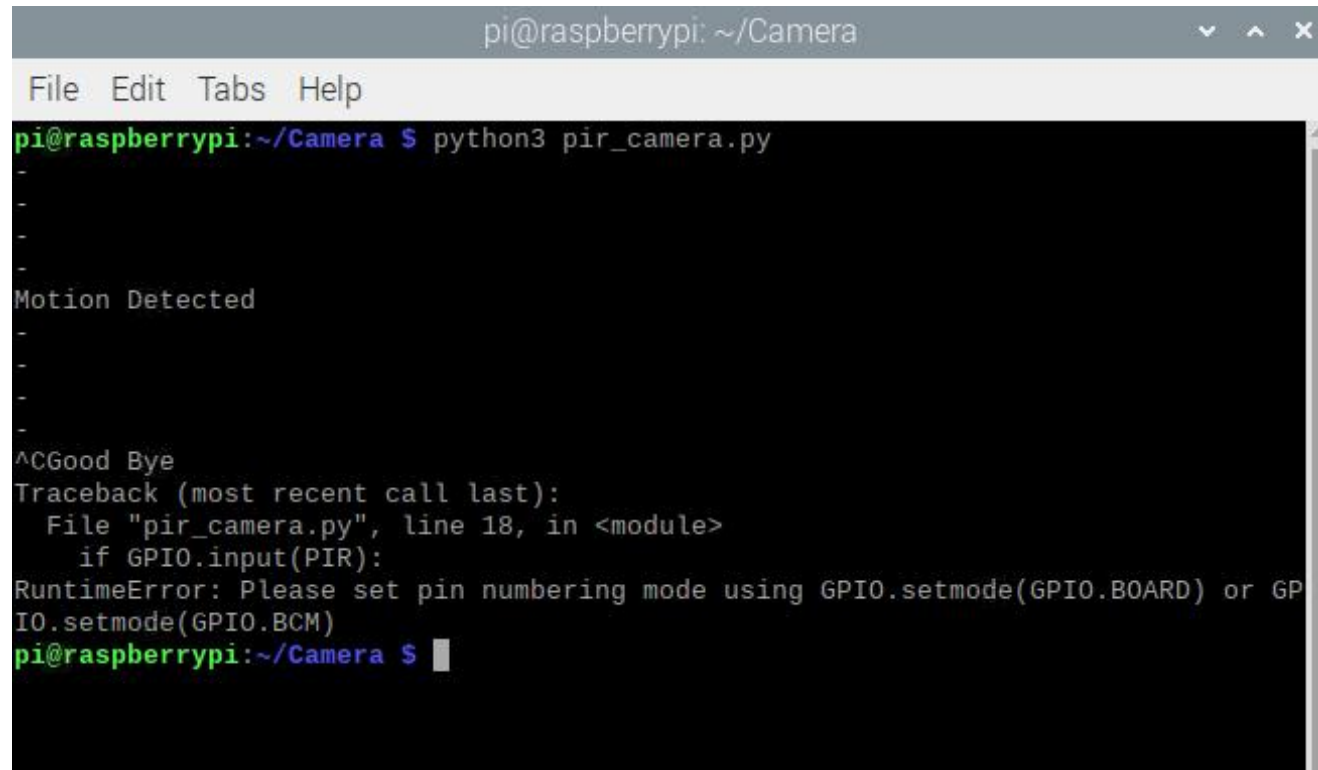


```
while True:
    try:
        # Sense motion, usually human, within the target range
        if GPIO.input(PIR):
            print('Motion Detected')

            GPIO.output(RLED, 1)
            camera.capture('image0.jpg')
            time.sleep(1)
            GPIO.output(RLED, 0)
        else:
            print('-')

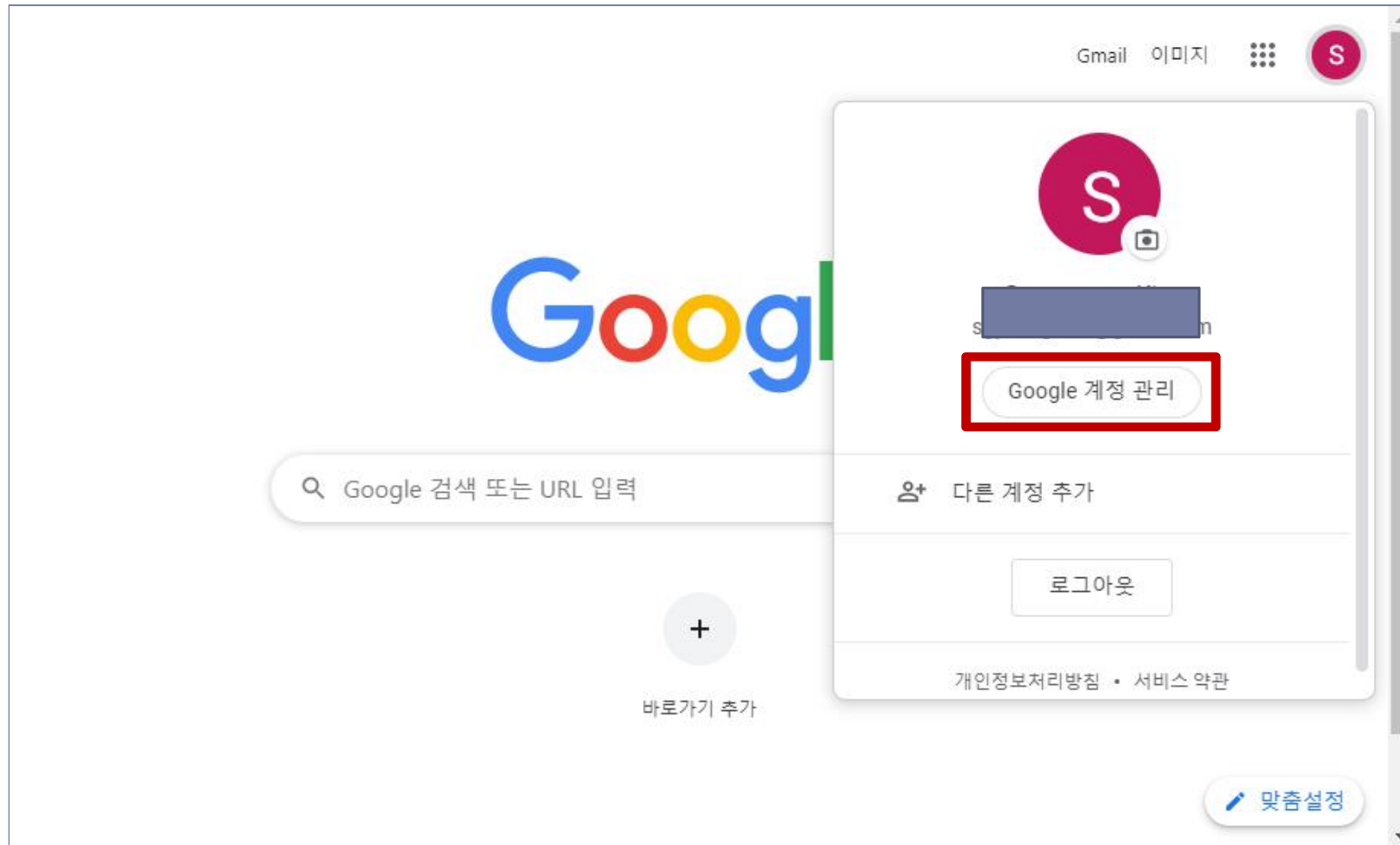
        # if your hold time is less than this,
        # you might not see as many detections
        time.sleep(1)
    except IOError:
        print("Error")
        GPIO.cleanup()
    except KeyboardInterrupt:
        print("Good Bye")
        GPIO.cleanup()
```

# 움직임 검출 및 카메라 제어: 실행 화면 (움직임 검출)

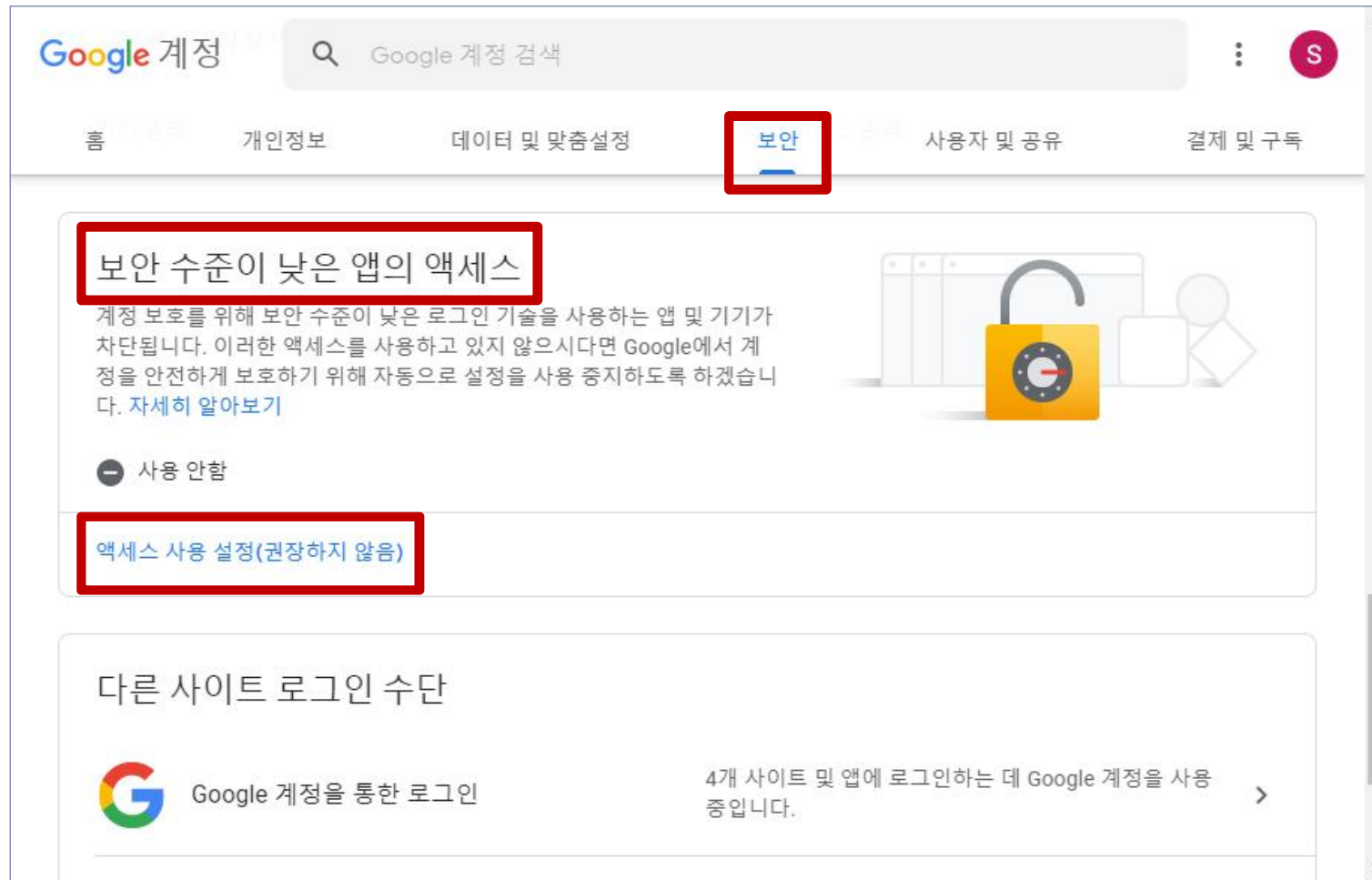


```
pi@raspberrypi: ~/Camera
File Edit Tabs Help
pi@raspberrypi:~/Camera $ python3 pir_camera.py
-
-
-
-
Motion Detected
-
-
-
-
^C Good Bye
Traceback (most recent call last):
  File "pir_camera.py", line 18, in <module>
    if GPIO.input(PIR):
RuntimeError: Please set pin numbering mode using GPIO.setmode(GPIO.BOARD) or GPIO.setmode(GPIO.BCM)
pi@raspberrypi:~/Camera $
```

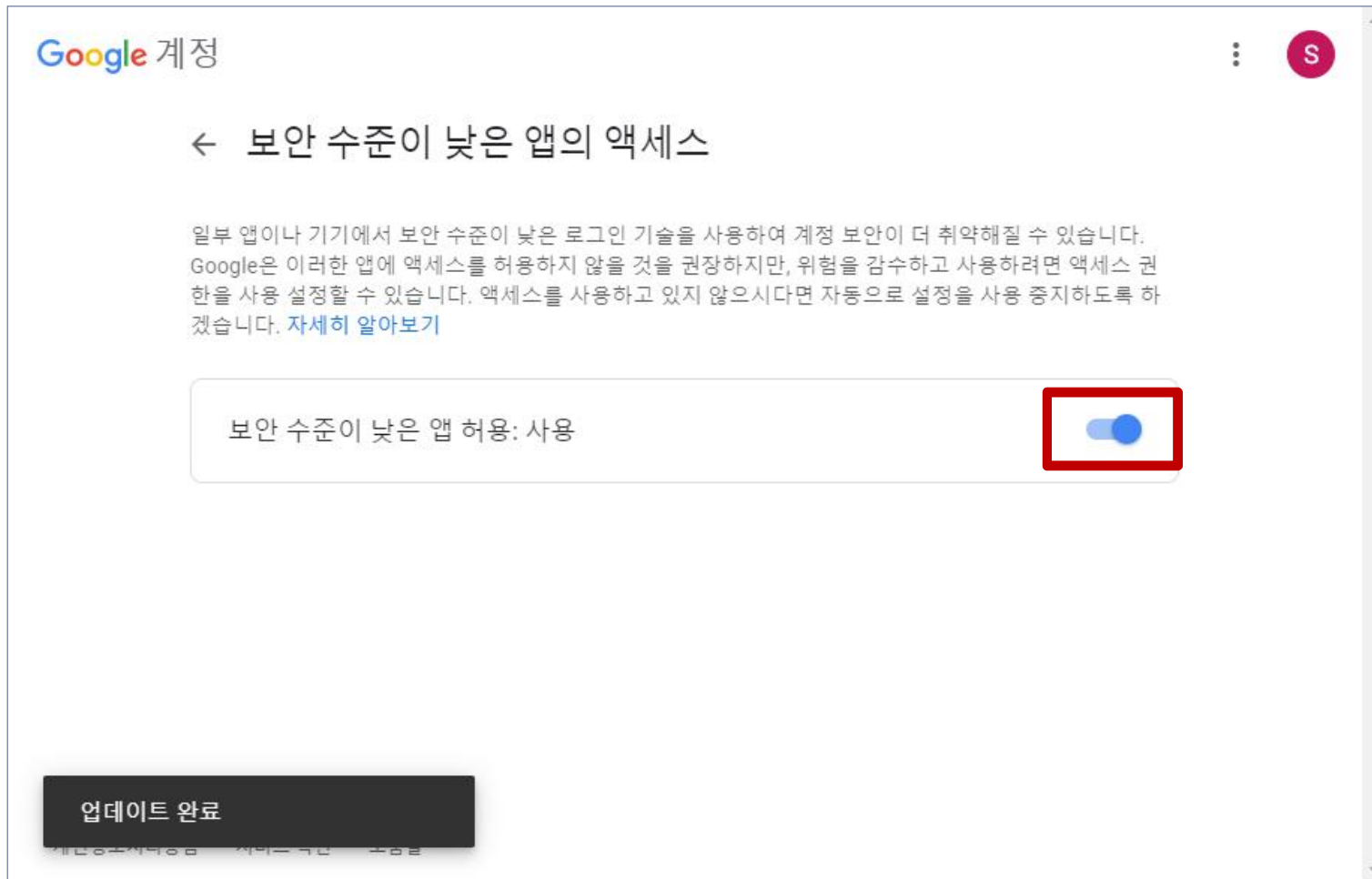
# 움직임 검출 및 카메라 제어: 이메일 발송을 위한 gmail 설정 (1)



## 움직임 검출 및 카메라 제어: 이메일 발송을 위한 gmail 설정 (2)



## 움직임 검출 및 카메라 제어: 이메일 발송을 위한 gmail 설정 (3)



# 움직임 감출 및 카메라 제어: 소스 코드 (이메일 발송)

pir\_camera\_email.py

```
import RPi.GPIO as GPIO
import picamera
import smtplib, subprocess, time, datetime
from email.mime.text import MIMEText

GPIO.setmode(GPIO.BCM)

PIR = 12
RLED = 21

GPIO.setup(PIR, GPIO.IN)
GPIO.setup(RLED, GPIO.OUT)

camera = picamera.PiCamera()
```

```
# settings for sending e-mail
print("System Working")

SMTP_USERNAME = '. . .' # Gmail id of the sender
SMTP_PASSWORD = '. . .' # Gmail Password of the sender
SMTP_RECIPIENT = '. . .' # Mail id of the receiver
SMTP_SERVER = 'smtp.gmail.com' # Address of the SMTP server
SSL_PORT = 465

while True:
    try:
        # Sense motion, usually human, within the target range
        if GPIO.input(PIR):
            print('Motion Detected')

            GPIO.output(RLED, 1)
            camera.capture('image0.jpg')
            time.sleep(1)
            GPIO.output(RLED, 0)
```

```
# connecting e-mail server
p = subprocess.Popen(["runlevel"], stdout=subprocess.PIPE)
out, err = p.communicate()
if out[2] == '0':
    print('Halt detected')
    exit(0)
if out [2] == '6':
    print('Shutdown detected')
    exit(0)
print("Connected to mail")
```



```
# Create the container (outer) email message
TO = SMTP_RECIPIENT
FROM = SMTP_USERNAME

localtime = datetime.datetime.now()
strTime = localtime.strftime("%Y-%m-%d %H:%M")
msg = MIMEText('Motion Detected in RPi: '+strTime)
msg['Subject'] = 'Raspi Noti'
#msg['From'] = SMTP_RECIPIENT+'@gmail.com'
msg['To'] = SMTP_RECIPIENT

# Send the email via Gmail
print("Sending the mail")
server = smtplib.SMTP_SSL(SMTP_SERVER, SSL_PORT)
server.login(SMTP_USERNAME, SMTP_PASSWORD)
#server.send_message(msg)
server.sendmail(FROM, TO, msg.as_string())
server.quit()

print("Mail sent")
```

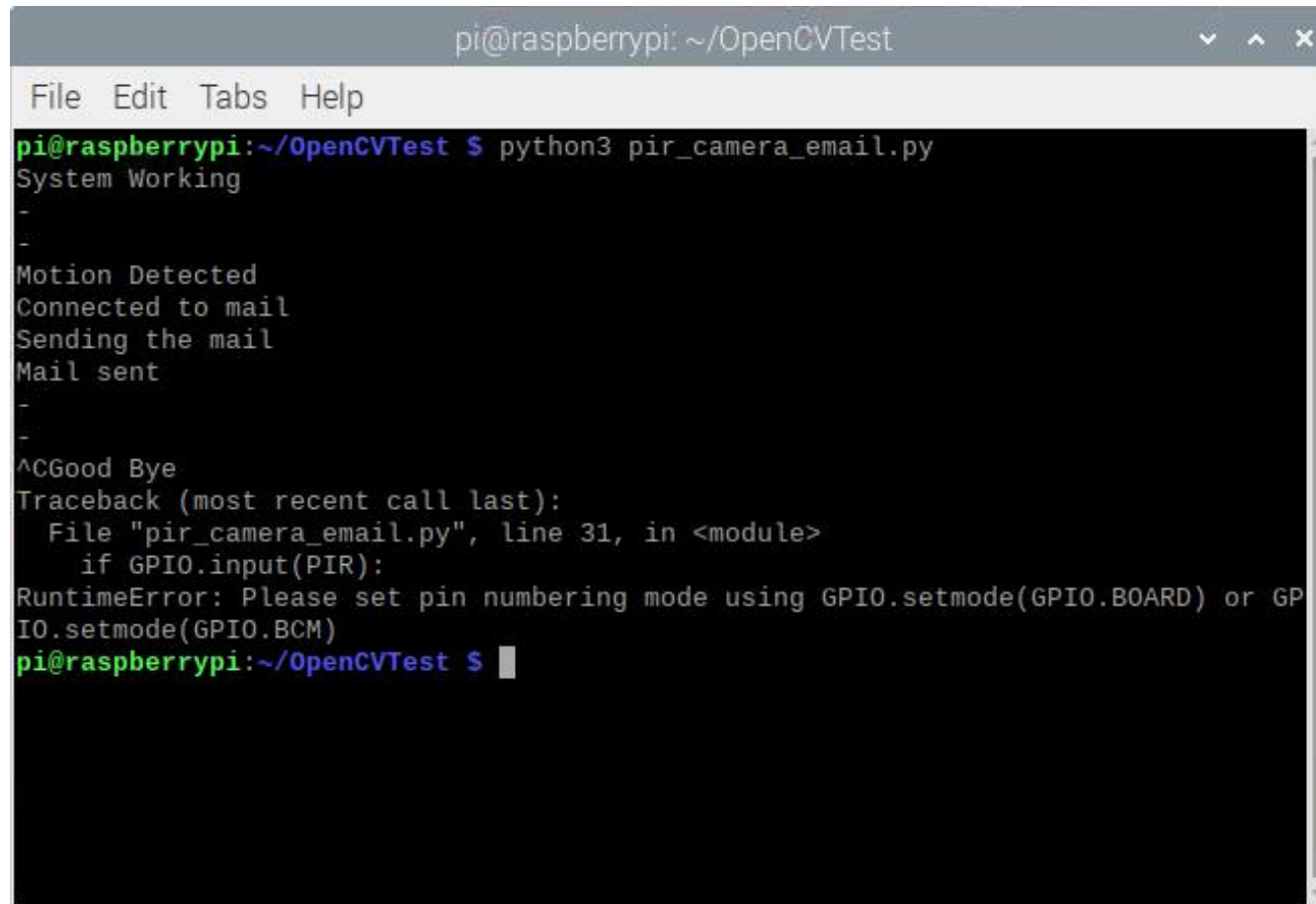
```
    else:
        print('-')

    # if your hold time is less than this,
    # you might not see as many detections
    time.sleep(1)

except IOError:
    print("Error")
    GPIO.cleanup()

except KeyboardInterrupt:
    print("Good Bye")
    GPIO.cleanup()
```

## 움직임 검출 및 카메라 제어: 실행 화면 (이메일 발송)



```
pi@raspberrypi: ~/OpenCVTest
File Edit Tabs Help
pi@raspberrypi:~/OpenCVTest $ python3 pir_camera_email.py
System Working
-
-
Motion Detected
Connected to mail
Sending the mail
Mail sent
-
-
^C Good Bye
Traceback (most recent call last):
  File "pir_camera_email.py", line 31, in <module>
    if GPIO.input(PIR):
RuntimeError: Please set pin numbering mode using GPIO.setmode(GPIO.BOARD) or GPIO.setmode(GPIO.BCM)
pi@raspberrypi:~/OpenCVTest $
```

# 움직임 검출 및 카메라 제어: 실행 화면 (이메일 발송)

Raspi Noti ☆

보낸사람

sgyoung.kim@gmail.com

받는사람

sykim@kumoh.ac.kr

☒ 이미지 표시

☐ 이미지 차단

☐ 모든 HTML 태그 차단

[이 주소에서 온 메일을 항상 이미지 표시](#)

Motion Detected in RPi: 2021-01-06 23:43

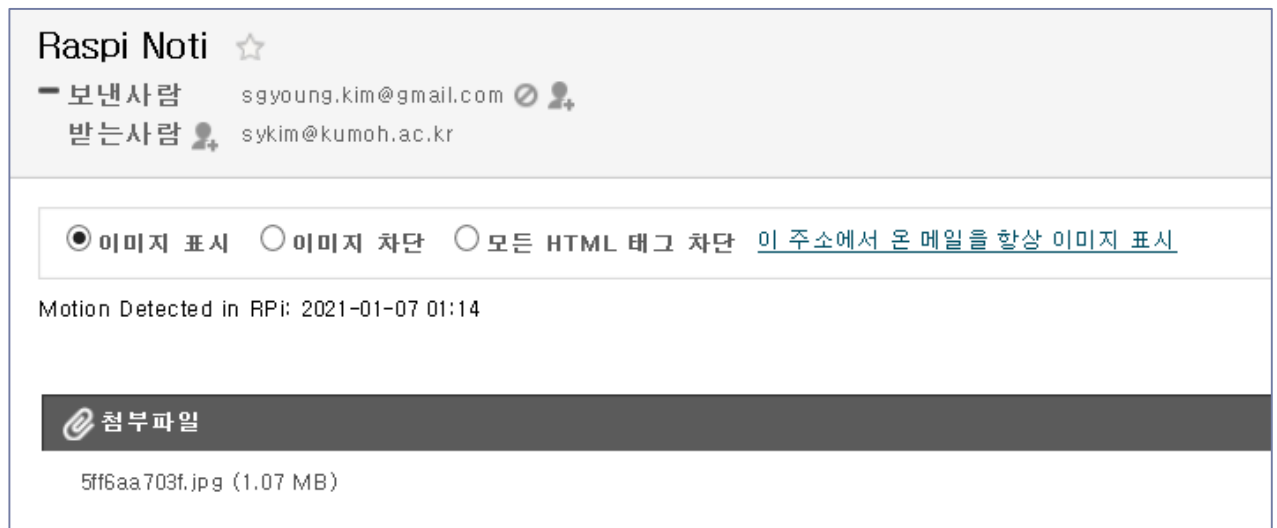
## 움직임 검출 및 카메라 제어 (실습)

- 다음 URL을 참조하여 캡처한 영상을 함께 이메일 전송하고 SMS를 발송하도록 코드 수정

□ <https://www.dexterindustries.com/GrovePi/projects-for-the-raspberry-pi/raspberry-pi-motion-sensing/>

□ coolsms

- <https://console.coolsms.co.kr>
- <https://developer.coolsms.co.kr>

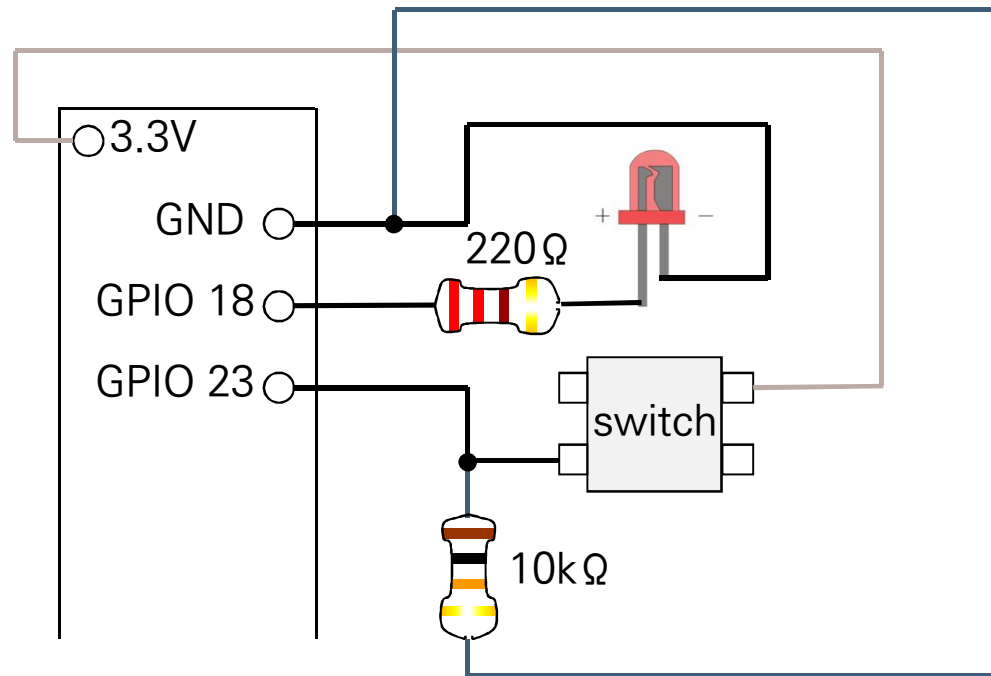


## 버튼으로 동영상 촬영하기

---

- 버튼을 누르면 5초 동안 동영상 촬영
  - 풀다운 저항 방식을 사용

## 버튼으로 동영상 촬영하기: 회로 구성



# 버튼으로 동영상 촬영하기: 소스 코드

```
import RPi.GPIO as GPIO
import picamera
import time

GPIO.setmode(GPIO.BCM)
LED = 18                                # GPIO 18번으로 설정
PUSH = 23                              # GPIO 23번으로 설정

GPIO.setup(LED, GPIO.OUT)              # LED(18)번 채널을 출력으로 설정
GPIO.setup(PUSH, GPIO.IN)              # LED(23)번 채널을 입력으로 설정

camera = picamera.PiCamera()
try:
    while (True):
        if GPIO.input(PUSH) == GPIO.HIGH:
            print("start recording")
```

button\_video.py

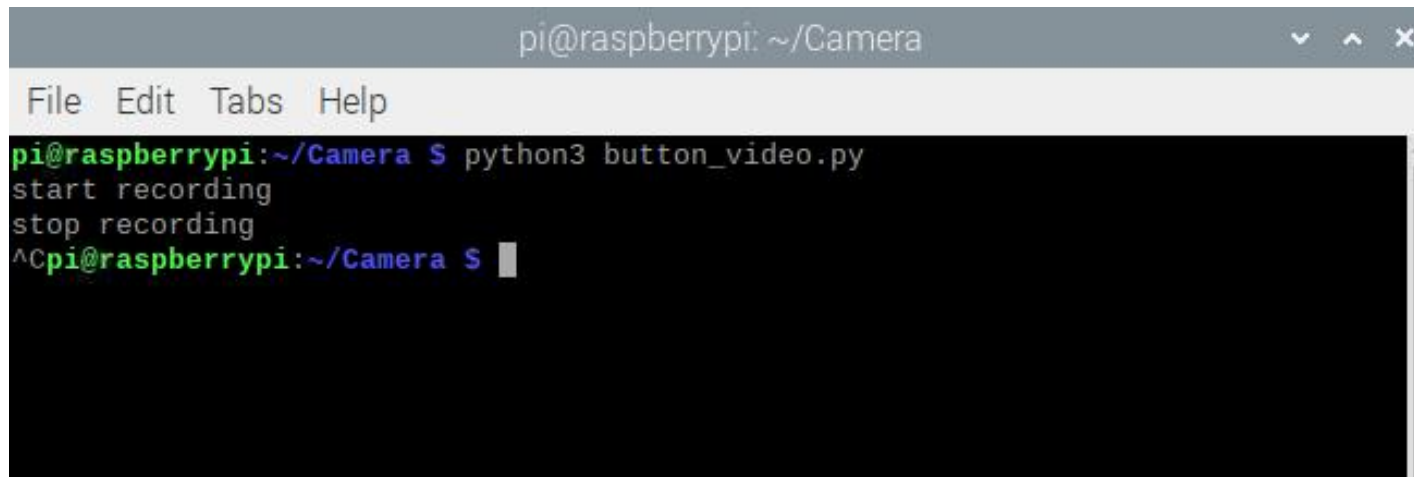


계속

```
GPIO.output(LED, GPIO.HIGH)
camera.start_recording('video.h264')
time.sleep(5)
camera.stop_recording()
GPIO.output(LED, GPIO.LOW)
print("stop recording")

except KeyboardInterrupt:          # 예외: 키보드 중단(Ctrl+C 등)
    GPIO.output(LED, GPIO.LOW)    # LED(18)에 LOW 상태 인가
    GPIO.cleanup()
```

## 버튼으로 동영상 촬영하기: 실행 화면



A terminal window titled "pi@raspberrypi: ~/Camera" with a menu bar containing "File", "Edit", "Tabs", and "Help". The terminal output shows the execution of a Python script:

```
pi@raspberrypi:~/Camera $ python3 button_video.py
start recording
stop recording
^Cpi@raspberrypi:~/Camera $
```

# OpenCV 사용 설정

---

- 가상 환경 활성화 (새로운 이미지에서 사용)

```
$ source ~/start_py3cv4.sh
```

- 쉘 프롬프트에 (py3cv4) 표시

- imutils 패키지 설치 (오늘 필요)

```
$ sudo pip3 install imutils
```

□ <https://github.com/jrosebr1/imutils>

# OpenCV를 사용한 카메라 접근

cv\_camera.py

```
from imutils.video import VideoStream
import imutils
import time
import cv2

#vs = VideoStream(src=0).start() #0: first cam, 1: second cam
vs = VideoStream(usePiCamera=True, resolution=(640, 480)).start()
time.sleep(2.0)

while True:
    # grab the frame
    frame = vs.read()

    # codes for image processing

    # show the output frame
    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1) & 0xFF
    if key == ord("q"):
        break

cv2.destroyAllWindows()
vs.stop()
```

## 간단한 영상처리 (1)

---

- 영상을 흐릿하게 변환

- `frame = cv2.GaussianBlur(frame, (5, 5), 0)`

- 칼라 영상을 회색음영 영상으로 변환

- `frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)`

- 영역의 경계 검출

- `frame = cv2.Canny(frame, 100, 200)`

## 간단한 영상처리 (2)

---

- 회색음영 영상을 이진 영상으로 변환

- `th, frame = cv2.threshold(frame, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)`

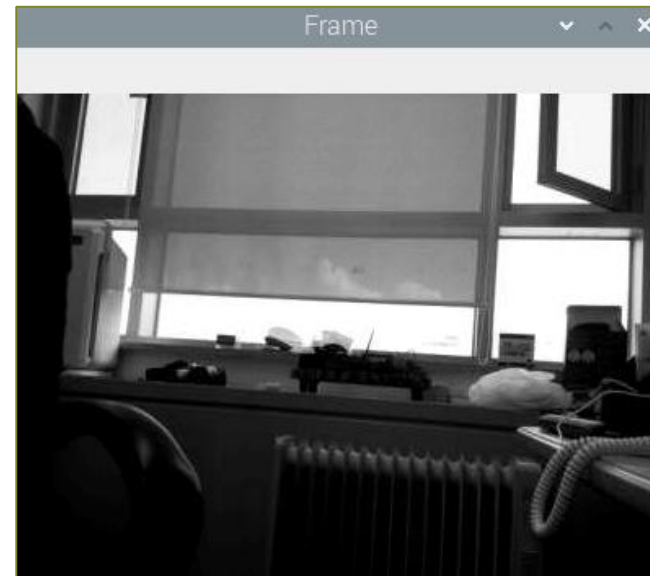
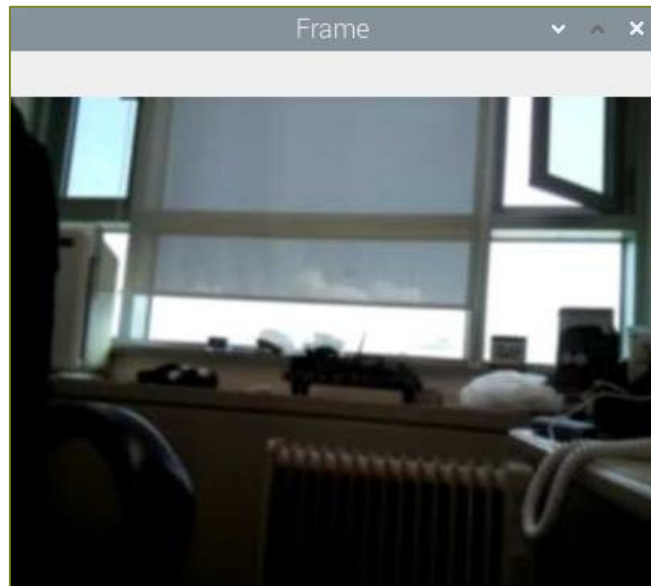
- 모폴로지 변환 (이진 영상에 적용 가능)

- 회색음영 영상을 이진 영상으로 변환하는 작업과 함께 사용

- `K = cv2.getStructuringElement(cv2.MORPH_RECT, (5,5))`

- `frame = cv2.morphologyEx(frame, cv2.MORPH_OPEN, K)`

- `frame = cv2.morphologyEx(frame, cv2.MORPH_CLOSE, K)`



## 카메라 설정 변경 방법

---

- Method #1: GUI method — guvcview
- Method #2: Command line option — v4l2-ctl
- Method #3: OpenCV's API combined with imutils.video
- Method #4: PiCamera's API combined with imutils.video



# 카메라 설정 변경 (1)

- guvcview

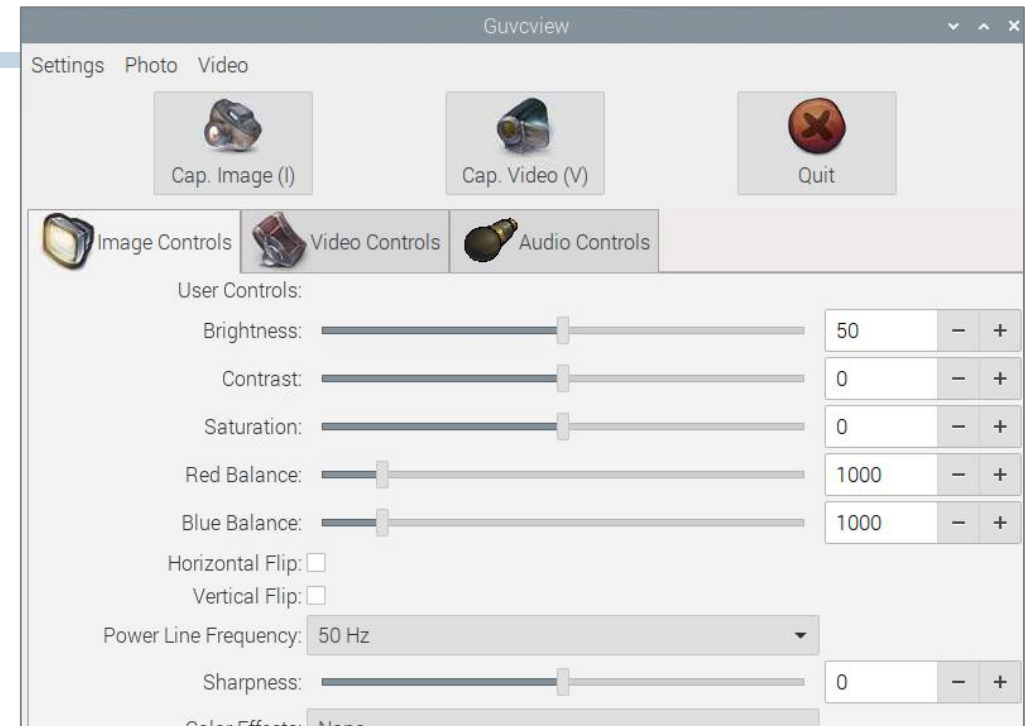
- GTK Viewer application (GUI 기반)
- 변경한 설정은 unplug 할 때까지 유효
- Gvvcview 설치 및 실행

```
$ sudo apt-get install guvcview
```

```
$ guvcview
```

- V4l2-ctl

- v4l2-ctl -h
- v4l2-ctl -c brightness=200 -c contrast=10



## 카메라 설정 변경 (2)

---

- OpenCV API를 사용한 웹 캠 설정 변경
  - `cv2.CAP_PROP_CONTRAST`
  - `cv2.CAP_PROP_BRIGHTNESS`
  - `cv2.CAP_PROP_FOCUS`
  - `cv2.CAP_PROP_ISO_SPEED`
  - `cv2.CAP_PROP_AUTO_WB`

## 카메라 설정 변경 - 시연

---

- USB 캠 설정 변경 ~ `change_usb_parameters.py`
- picamera 설정 변경 ~ `change_picamera_parameters.py`

## 예제: 얼굴 검출

- 실시간으로 얼굴을 검출하여 표시

- haarcascade.zip을 다운로드하여 소스코드와 동일 디렉토리에 압축 해제 (디렉토리 사용하지 않음)

- <http://image.kumoh.ac.kr/data/haarcascade.zip>
    - haarcascade\_frontalface\_default.xml



# 얼굴 검출: 소스코드

detect\_faces.py

```
. . .

frame = vs.read()

# load the face detector and detect faces in the image
detector = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
rects = detector.detectMultiScale(frame, scaleFactor=1.05, \
                                   minNeighbors=9, minSize=(40, 40),
                                   flags=cv2.CASCADE_SCALE_IMAGE)
print("[INFO] detected {} faces".format(len(rects)))

# draw a rectangle around each face
for (x, y, w, h) in rects:
    cv2.rectangle(frame, (x,y), (x+w, y+h), (0,255,0), 2)

# show the output frame
cv2.imshow("Frame", frame)

. . .
```

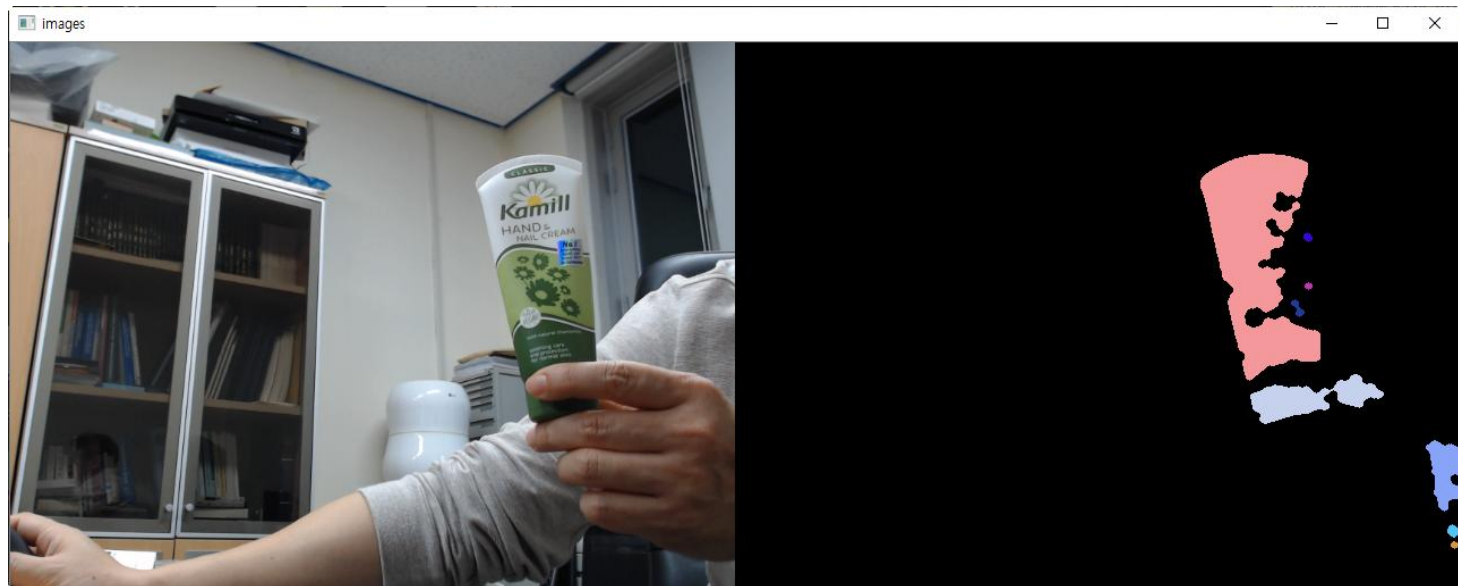
## 실습: 상반신 및 전신 검출

---

- haarcascade\_fullbody.xml과 haarcascade\_upperbody.xml 파일을 활용하여 상반신과 전신 검출
- detectMultiScale( ) 함수의 인자 설정 변경

## 실습: 움직이는 물체 검출

- 실시간으로 움직이는 물체를 검출하여 표시
  - `cv.createBackgroundSubtractorMOG2()` 사용
  - [https://docs.opencv.org/4.1.2/d1/dc5/tutorial\\_background\\_subtraction.html](https://docs.opencv.org/4.1.2/d1/dc5/tutorial_background_subtraction.html) 참고



# 도움말

---

- picamera documentation

- <http://picamera.readthedocs.org/en/release-1.10/>

- picamera simple instruction

- <https://www.raspberrypi.org/documentation/usage/camera/python/README.md>

- OpenCV-Python tutorial

- <http://docs.opencv.org>



## 참고: Raspi 보드 버전과 OS 버전 확인하기

---

- 보드 유형: `$ cat /proc/device-tree/model`  
Raspberry Pi 4 Model B Rev 1.2p
- 일반적인 커널 정보: `$ uname -a`  
Linux raspberrypi 4.14.79-v7+ #1159 SMP Sun Nov 4 17:50:20  
GMT 2018 armv7l GNU/Linux
- OS 버전에 대한 정보1: `$ cat /etc/issue`  
Raspbian GNU/Linux 9 \n \l
- OS 버전에 대한 정보2: `$ cat /etc/*release*`  
PRETTY\_NAME="Raspbian GNU/Linux 10 (buster)"  
. . .

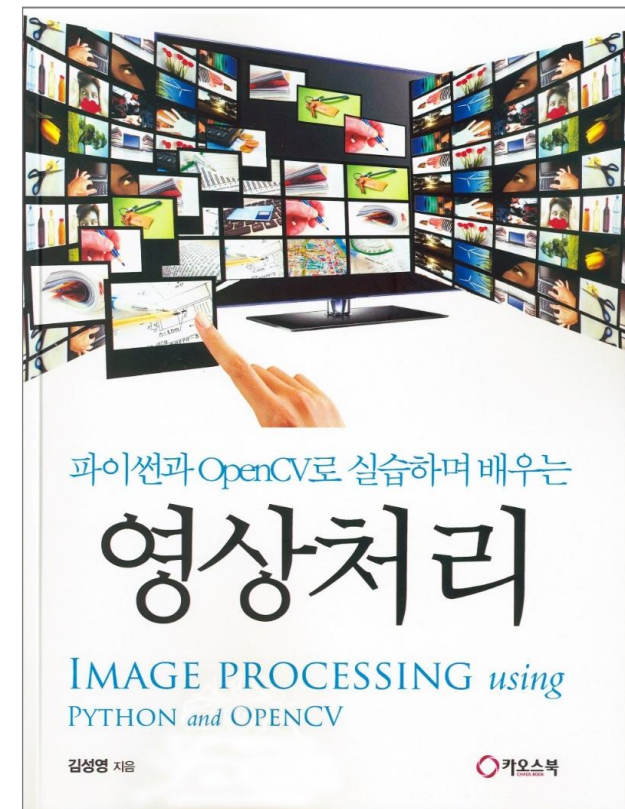
## 참고: Raspi 보드 RAM 용량 확인

- 용량 확인: \$ free -h

	total	used	free	shared	buff/cache
Mem:	3.8Gi	191Mi	3.2Gi	68Mi	341Mi
3.4Gi					
Swap:	99Mi	0B	99Mi		

## 참고문헌

- 파이썬과 OpenCV로 실습하며 배우는 영상처리, 김성영 저, 카오스북



## 다음 강의 주제

---

- 카메라 기반의 OpenCV 활용