

어플리케이션 만들기

Raspberry Pi Camera

Raspberry Pi Camera Tool

라즈베리 파이 스타일 카메라 접근하기 - 1

- raspistill : 파이 캠을 이용하여 사진 캡처할 때 사용하는 기본 제공 명령어
- raspivid : 파이 캠을 이용하여 동영상을 녹화할 때 사용하는 기본 제공 명령어

라즈베리 파이 스타일 카메라 접근하기 - 파이캠 활성화 하기

- sudo raspi-config
- 6. Enable Cam Enble Enable 설정

Raspberry Pi Software Configuration Tool (raspi-config)

1 Expand Filesystem	Ensures that all of the SD card storage is available to the OS
2 Change User Password	Change password for the default user (pi)
3 Boot Options	Choose whether to boot into a desktop environment or the command line
4 Wait for Network at Boot	Choose whether to wait for network connection during boot
5 Internationalisation Options	Set up language and regional settings to match your location
6 Enable Camera	Enable this Pi to work with the Raspberry Pi Camera
7 Add to Rastrack	Add this Pi to the online Raspberry Pi Map (Rastrack)
8 Overclock	Configure overclocking for your Pi
9 Advanced Options	Configure advanced settings
0 About raspi-config	Information about this configuration tool

<Select>

<Finish>

라즈베리 파이 캠을 이용한 동영상 녹화 – raspivid 사용예제

- 320x240 사이즈의 동영상을 10초간 video.h264의 이름으로 녹화후 저장할시

```
#raspivid -w 320 -h 240 -t 10000 -o video.h264
```

라즈베리 파이 캠을 이용한 동영상 녹화 – raspivid 옵션

- -o : output file name 설정
- -w : 너비 설정
- -h : 높이 설정
- -t : 촬영 시간 설정
- -fps : 초당 프레임 설정
- -cd : 코덱 설정 H264 or MJPEG 사용 설정
- -sh : 선명도 설정 (-100 ~ 100)
- -co : 대비값 설정(-100~100)
- -sa : 채도값 설정(-100~100)
- -br : 밝기값 설정(0~100)

라즈베리 파이 캠을 이용한 동영상 녹화 – raspistill 사용예제

1024x768 사이즈의 사진을 jpg 타입으로
10초 타이머 후 picture.jpg으로 저장할 시

```
#raspistill -w 1024 -h 768 -e jpg -o picture.jpg
```

라즈베리 파이 스타일 카메라 접근하기 - raspistill 옵션

- -o : output file name 설정
- -w : 너비 설정
- -h : 높이 설정
- -t : 타이머 설정
- -e : 인코딩 설정(jpg, bmp, gif, png)
- -sh : 선명도 설정 (-100 ~ 100)
- -co : 대비값 설정(-100~100)
- -sa : 채도값 설정(-100~100)
- -br : 밝기값 설정(0~100)

USB Camera 사진 캡처

- fswebcam 사용

- USB 카메라 캡처 패키지 fswebcam 이용
- `sudo apt-get install fswebcam`
- 사용법 : `fswebcam <옵션> 파일명`
- 캡처 테스트 해보기

```
#fswebcam image.jpg
```

```
root@raspberrypi:/home/pi/tst# fswebcam image.jpg
--- Opening /dev/video0...
Trying source module v4l2...
/dev/video0 opened.
No input was specified, using the first.
Adjusting resolution from 384x288 to 352x288.
--- Capturing frame...
Captured frame in 0.00 seconds.
--- Processing captured image...
Writing JPEG image to 'image.jpg'.
root@raspberrypi:/home/pi/tst#
```

USB Camera 사진 캡처

- fswebcam 옵션

- -b : 백그라운드에서 실행
- -D : 타이머 설정
- -r : 해상도 설정 ex)-r 1280x720
- --no-banner : 하단 날짜, 시간배너 제거
- -invert : 색상 반전
- -grayscale : 흑백 설정

Linux Video Interface

V₄L2

V4l2 소개

Video for linux 2는 video for linux의 두 번째 버전이다.

유니릭스 리눅스 시스템에서
실시간 비디오 캡처를 지원하기 위한
어플리케이션과 드라이버의 API 모음이다.
커널에는 유니릭를 지원하는 드라이버가 있으며
캐릭터 드라이버의 한 종이다.

V4L2 캡처

- V4L2 드라이버를 적재 시킨다.

```
#sudo modprobe bcm2835-v4l2
```

- 예제를 컴파일 후 실행!!

V4l2 예제

```
#define VIDEODEV      "/dev/video0 "
...
int main(int argc, char ** argv)
{
    for (;;) {
        int idx;
        int c;

        c = getopt_long(argc, argv,
                        short_options, long_options, &idx);
        ...

    }
```

open_device();	→	장치 열기
init_device();	→	UVC 드라이버 설정
start_capturing();	→	캡처 시작
mainloop();	→	영상 복사하기
savelmage(argv[1], inimg);	→	bmp 파일로 저장
stop_capturing();	→	캡처 정지
uninit_device();	→	장치 해제
close_device();	→	장치 닫기
return 0;		

V4l2 예제 흐름

```
open( "/dev/video0" ..);
```

```
ioctl( VIDIOC_S_FMT,);
```

```
·  
·
```

```
fb_mem = mmap();
```

```
ioctl( VIDIOC_STREAMON,);
```

```
*(char *)fb_mem
```

```
ioctl( VIDIOC_STREAMOFF,);
```

```
close();
```

← 어플리케이션 메모리로 영상 복사

장치 열기 - open_device()

```
static void open_device(void)
{
    dev_name = "/dev/video0";
    ...
    fd = open(dev_name, O_RDWR /* required */ | O_NONBLOCK, 0);
    if (-1 == fd) {
        fprintf(stderr, "Cannot open '%s': %d, %s\n",
            dev_name, errno, strerror(errno));
        exit(EXIT_FAILURE);
    }
}
```

open 함수를 이용하여 /dev/video0에 접근한다.

```
# cat /sys/class/video4linux/video0/dev
81:0

# cat /proc/devices
81 video4linux

# ls -al /dev/video0
crw-rw----+ 1 root video 81, 0 Jun 28 03:23 /dev/video0
```


장치 열기

- V4l2 장치 파일

/dev/video 0–63 Video Capture Interface

/dev/radio 64–127 AM/FM Radio Devices

/dev/vtx 192–223 Teletext Interface Chips

/dev/vbi 224–239 Raw VBI Data
(Intericast/teletext)

참고:

https://linuxtv.org/downloads/legacy/video4linux/API/V4L1_API.html

Documentation/video4linux

장치 초기화 – init_device()

```
static void init_device(void)
{
    ...

    CLEAR(fmt);
    fmt.type = V4L2_BUF_TYPE_VIDEO_CAPTURE;

    fmt.fmt.pix.width      = 640;
    fmt.fmt.pix.height     = 480;
    fmt.fmt.pix.pixelformat = V4L2_PIX_FMT_YUYV;
    fmt.fmt.pix.field      = V4L2_FIELD_INTERLACED;

    if (0 > ioctl(fd, VIDIOC_S_FMT, &fmt)) // 비디오 포맷 설정
        errno_exit("VIDIOC_S_FMT");

    init_mmap(); // 비디오 메모리 매핑
}
```

V4l2 ioctl cmd

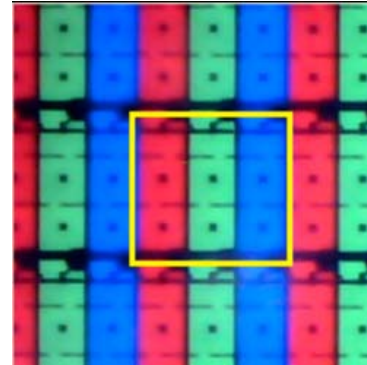
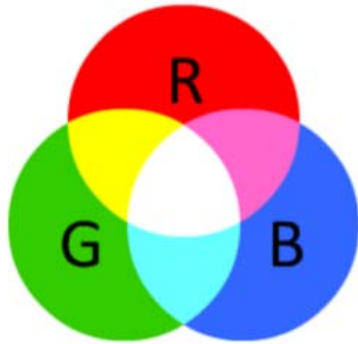
op. code	I/O	structure
VIDIOC_QUERYCAP	IOR	struct v4l2_capability
VIDIOC_RESERVED	IO	1
VIDIOC_ENUM_FMT	IOWR	struct v4l2_fmtdesc
VIDIOC_G_FMT	IOWR	struct v4l2_format
VIDIOC_S_FMT	IOWR	struct v4l2_format
VIDIOC_G_COMP	IOR	struct v4l2_compression
VIDIOC_S_COMP	IOW	struct v4l2_compression
VIDIOC_REQBUFS	IOWR	struct v4l2_requestbuffers
VIDIOC_QUERYBUF	IOWR	struct v4l2_buffer
VIDIOC_G_FBUF	IOR	struct v4l2_framebuffer
VIDIOC_S_FBUF	IOW	struct v4l2_framebuffer
VIDIOC_OVERLAY	IOWR	int
VIDIOC_QBUF	IOWR	struct v4l2_buffer
VIDIOC_DQBUF	IOWR	struct v4l2_buffer
VIDIOC_STREAMON	IOW	int
VIDIOC_STREAMOFF	IOW	int
VIDIOC_G_PARM	IOWR	struct v4l2_streamparm
VIDIOC_S_PARM	IOW	struct v4l2_streamparm
VIDIOC_G_STD	IOR	v4l2_std_id
VIDIOC_S_STD	IOW	v4l2_std_id
VIDIOC_ENUMSTD	IOWR	struct v4l2_standard

linux/videodev2.h

VIDIOC_ENUMINPUT	IOWR	struct v4l2_input
VIDIOC_G_CTRL	IOWR	struct v4l2_control
VIDIOC_S_CTRL	IOW	struct v4l2_control
VIDIOC_G_TUNER	IOWR	struct v4l2_tuner
VIDIOC_S_TUNER	IOW	struct v4l2_tuner
VIDIOC_G_AUDIO	IOWR	struct v4l2_audio
VIDIOC_S_AUDIO	IOW	struct v4l2_audio
VIDIOC_QUERYCTRL	IOWR	struct v4l2_queryctrl
VIDIOC_QUERYMENU	IOWR	struct v4l2_querymenu
VIDIOC_G_INPUT	IOR	int
VIDIOC_G_OUTPUT	IOWR	int
VIDIOC_S_OUTPUT	IOWR	int
VIDIOC_ENUMOUTPUT	IOWR	struct v4l2_output
VIDIOC_G_AUDOUT	IOWR	struct v4l2_audioout
VIDIOC_S_AUDOUT	IOW	struct v4l2_audioout
VIDIOC_G_MODULATOR	IOWR	struct v4l2_modulator
VIDIOC_S_MODULATOR	IOW	struct v4l2_modulator
VIDIOC_G_FREQUENCY	IOWR	struct v4l2_frequency
VIDIOC_S_FREQUENCY	IOW	struct v4l2_frequency
VIDIOC_CROPCAP	IOR	struct v4l2_cropcap
VIDIOC_G_CROP	IOWR	struct v4l2_crop

Image pixel data format

- RGB



- YUV 422/420

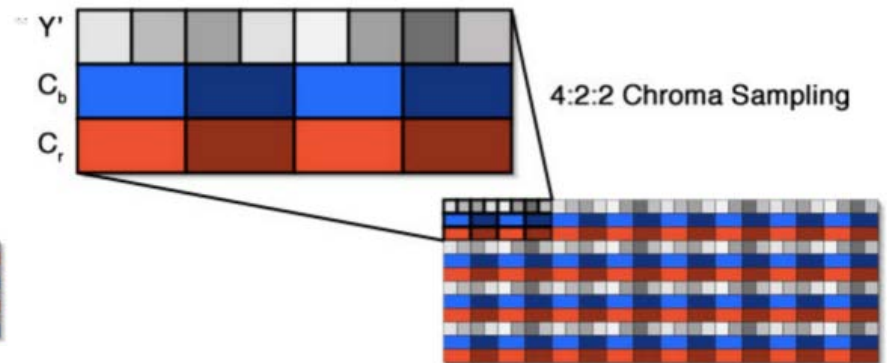
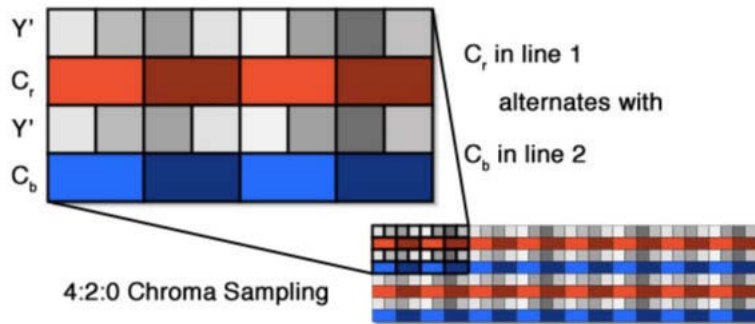


Image pixel data 변환

- YUV → RGB

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1.402 \\ 1 & -0.34414 & -0.71414 \\ 1 & 1.772 & 0 \end{bmatrix} \begin{bmatrix} Y \\ C_b - 128 \\ C_r - 128 \end{bmatrix}$$

- RGB → YUV

$$\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.16874 & -0.3313 & 0.500 \\ 0.500 & -0.4187 & -0.0813 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix}$$

Image 변환 코드

```
static void process_image(const void *p, int size)
{
```

```
    ...
```

```
    for(y = 0; y < height; y++)
    {
```

```
        for(j = 0, x = 0; j < HEIGHT * 2; j += 4, x += 2)
        {
```

```
            if(j >= width*2)
            {
```

```
                location++; location++;
                continue;
            }
```

```
            y0 = in[j];
```

```
            u = in[j+1]-128;
```

```
            y1 = in[j+2];
```

```
            v = in[j+3]-128;
```

```
            r = clip((298*y0 + 409*v + 128) >> 8, 0, 255);
```

```
            g = clip((298*y0 - 100*u - 208*v + 128) >> 8, 0, 255);
```

```
            b = clip((298*y0 + 516*u + 128) >> 8, 0, 255);
```

```
            inimg[(height-y-1)*width*3+count++] = b;
```

```
            inimg[(height-y-1)*width*3+count++] = g;
```

```
            inimg[(height-y-1)*width*3+count++] = r;
```

```
            pixel = ( (r>>3)<<11 ) | ( (g>>2)<<5 ) | (b>>3);
```

픽셀 변환

첫 번째 픽셀

```
            r = clip((298*y1 + 409*v + 128) >> 8, 0, 255);
```

```
            g = clip((298*y1 - 100*u - 208*v + 128) >> 8, 0, 255);
```

```
            b = clip((298*y1 + 516*u + 128) >> 8, 0, 255);
```

```
            inimg[(height-y-1)*width*3+count++] = b;
```

```
            inimg[(height-y-1)*width*3+count++] = g;
```

```
            inimg[(height-y-1)*width*3+count++] = r;
```

```
            pixel = ( (r>>3)<<11 ) | ( (g>>2)<<5 ) | (b>>3);
```

픽셀 변환

두 번째 픽셀

```
        }
        in += istride;
```

```
        count = 0;
```

```
    }
```

비디오 메모리 매핑 - init_mmap()

```
static void init_mmap(void)
{
    struct v4l2_requestbuffers req;

    CLEAR(req);

    req.count = 4;
    req.type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
    req.memory = V4L2_MEMORY_MMAP;

    ... if (0 > ioctl(fd, VIDIOC_REQBUFS, &req)) {
    ...

        struct v4l2_buffer buf;

        CLEAR(buf);

        buf.type      = V4L2_BUF_TYPE_VIDEO_CAPTURE;
        buf.memory     = V4L2_MEMORY_MMAP;
        buf.index      = n_buffers;

        if (0 > ioctl(fd, VIDIOC_QUERYBUF, &buf))
            errno_exit("VIDIOC_QUERYBUF");

        buffers[n_buffers].length = buf.length;
        buffers[n_buffers].start =
            mmap(NULL /* start anywhere */,
                buf.length,
                PROT_READ | PROT_WRITE /* required */,
                MAP_SHARED /* recommended */,
                fd, buf.m.offset);

        if (MAP_FAILED == buffers[n_buffers].start)
            errno_exit("mmap");
    }
}
```

메모리 크기. 614400 = 640x480x2

메모리 선택 주소

메모리 매핑

캡처 시작 – start_capturing()

```
static void start_capturing(void)
{
    unsigned int i;
    enum v4l2_buf_type type;

    for (i = 0; i < n_buffers; ++i) {
        struct v4l2_buffer buf;

        CLEAR(buf);
        buf.type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
        buf.memory = V4L2_MEMORY_MMAP;
        buf.index = i;

        if (0 > ioctl(fd, VIDIOC_QBUF, &buf)) → Buffer 생성
            errno_exit("VIDIOC_QBUF");
    }

    type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
    if (0 > ioctl(fd, VIDIOC_STREAMON, &type)) → 버퍼에 데이터 수집
        errno_exit("VIDIOC_STREAMON");
    break;
}
```


영상 가져오기 - main_loop()

```
static void mainloop(void)
{
...
    for (;;) {
...
        r = select(fd + 1, &fds, NULL, NULL,
            &tv);

        if (-1 == r) {
            if (EINTR == errno)
                continue;
            errno_exit("select");
        }

        if (0 == r) {
            fprintf(stderr, "select timeout\n");
            exit(EXIT_FAILURE);
        }

        if (read_frame())
            break;
    }
...
}
```

영상 데이터 복사/변환

read_frame()

```
static int read_frame(void)
{
...
    CLEAR(buf);

    buf.type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
    buf.memory = V4L2_MEMORY_MMAP;

    if (o > ioctl(fd, VIDIOC_DQBUF, &buf)) {
        switch (errno) {
            case EAGAIN:
                return o;

            case EIO:
                /* Could ignore EIO, see spec. */

                /* fall through */

            default:
                errno_exit("VIDIOC_DQBUF");
        }
    }

    assert(buf.index < n_buffers);

    process_image(buf.index, buf.start, buf.bytesused);

    if (o > ioctl(fd, VIDIOC_QBUF, &buf))
        errno_exit("VIDIOC_QBUF");
...
    return 1;
}
```

—————→ Data Queue empty or Fill

—————→ YUV → RGB

파일 저장 - saveImage()

```
void saveImage(char *filename, unsigned char *inimg)
```

```
{
    RGBQUAD palrgb[256];
    FILE *fp;
    int imgfd;
    BITMAPFILEHEADER bmpHeader;
    BITMAPINFOHEADER bmpInfoHeader;

    MEMZERO(bmpHeader);
    bmpHeader.bfType = 0x4d42;
    bmpHeader.bfOffBits = sizeof(BITMAPFILEHEADER) + sizeof(BITMAPINFOHEADER);
    bmpHeader.bfSize += bmpHeader.bfOffBits;
    bmpHeader.bfSize += WIDTH*HEIGHT*NUMCOLOR;
    bmpHeader.bfOffBits += sizeof(RGBQUAD) * 256;
```

Bmp 헤더 생성

```
    MEMZERO(bmpInfoHeader);
    bmpInfoHeader.biSize = sizeof(BITMAPINFOHEADER); //40;
    bmpInfoHeader.biWidth = WIDTH;
    bmpInfoHeader.biHeight = HEIGHT;
    bmpInfoHeader.biPlanes = 1;
    bmpInfoHeader.biBitCount = NUMCOLOR*8;
    bmpInfoHeader.biSizeImage = WIDTH*HEIGHT*bmpInfoHeader.biBitCount/8;
    bmpInfoHeader.biXpelsPerMeter = 0x0B12;
    bmpInfoHeader.biYpelsPerMeter = 0x0B12;
```

```
    if((fp = fopen(filename, "wb")) == NULL)
    {
        fprintf(stderr, "Error : Failed to open file...\n");
        exit(EXIT_FAILURE);
    }
```

```
    //////////// BITMAPFILEHEADER
    fwrite(&bmpHeader.bfType, sizeof(char), 2, fp);
    fwrite(&bmpHeader.bfSize, sizeof(unsigned int), 1, fp);
    fwrite(&bmpHeader.bfReserved1, sizeof(unsigned short int), 1, fp);
    fwrite(&bmpHeader.bfReserved2, sizeof(unsigned short int), 1, fp);
    fwrite(&bmpHeader.bfOffBits, sizeof(unsigned int), 1, fp);
```

Bmp 헤더/info 저장

```
    //////////// BITMAPINFOHEADER
    fwrite(&bmpInfoHeader.biSize, sizeof(unsigned int), 1, fp);
    fwrite(&bmpInfoHeader.biWidth, sizeof(unsigned int), 1, fp);
    fwrite(&bmpInfoHeader.biHeight, sizeof(unsigned int), 1, fp);
    fwrite(&bmpInfoHeader.biPlanes, sizeof(unsigned short int), 1, fp);
    fwrite(&bmpInfoHeader.biBitCount, sizeof(unsigned short int), 1, fp);
    fwrite(&bmpInfoHeader.biCompression, sizeof(unsigned int), 1, fp);
    fwrite(&bmpInfoHeader.biSizeImage, sizeof(unsigned int), 1, fp);
    fwrite(&bmpInfoHeader.biXpelsPerMeter, sizeof(unsigned int), 1, fp);
    fwrite(&bmpInfoHeader.biYpelsPerMeter, sizeof(unsigned int), 1, fp);
    fwrite(&bmpInfoHeader.biClrUsed, sizeof(unsigned int), 1, fp);
    fwrite(&bmpInfoHeader.biClrImportant, sizeof(unsigned int), 1, fp);
    fwrite(palrgb, sizeof(RGBQUAD), 256, fp);
    fwrite(inimg, sizeof(unsigned char), WIDTH*HEIGHT*3, fp);
```

이미지 저장

```
    fclose(fp);
}
```

캡처 정지 – stop_capturing()/uninit_device()/close_device()

```
static void stop_capturing(void)
{
```

```
    enum v4l2_buf_type type;
```

```
    type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
```

```
    if (0 > ioctl(fd, VIDIOC_STREAMOFF, &type))
```

```
        errno_exit("VIDIOC_STREAMOFF");
```

```
}
```

→ 캡처 정지

```
static void uninit_device(void)
{
```

```
    unsigned int i;
```

```
    for (i = 0; i < n_buffers; ++i)
```

```
        if (-1 == munmap(buffers[i].start, buffers[i].length))
```

```
            errno_exit("munmap");
```

```
    free(buffers);
```

```
}
```

→ 메모리 해제

```
static void close_device(void)
{
```

```
    if (-1 == close(fd))
```

```
        errno_exit("close");
```

```
    fd = -1;
```

```
}
```

→ 장치 닫기

Thank you!!

and QnA