Embedded System Software

Webcam controller App & Device

컴퓨터공학과

120230200

김지섭

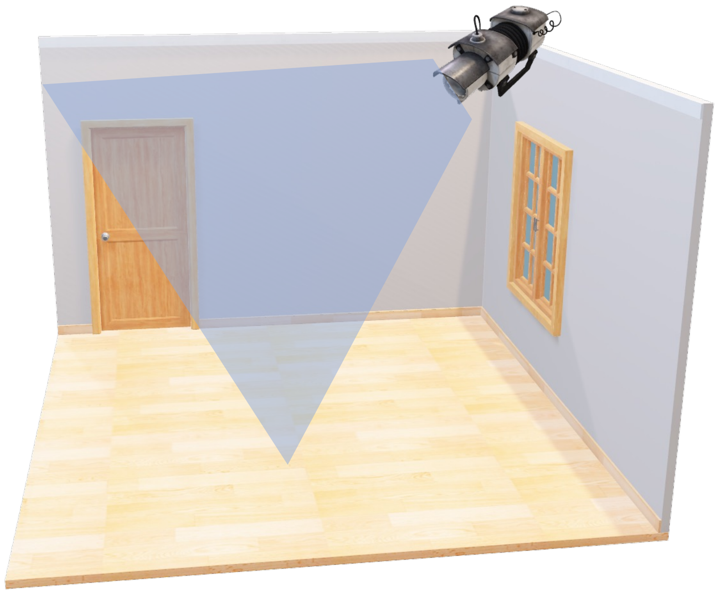
DISCOS

Data-Intensive Computing & Systems Laboratory

Sogang University, Seoul
South Korea

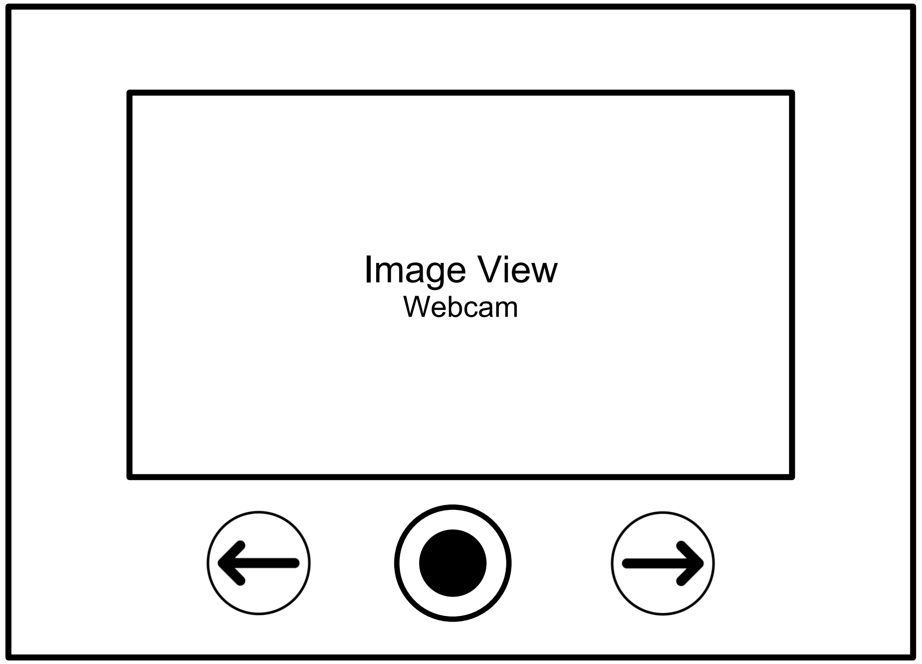
Contents

Introduction



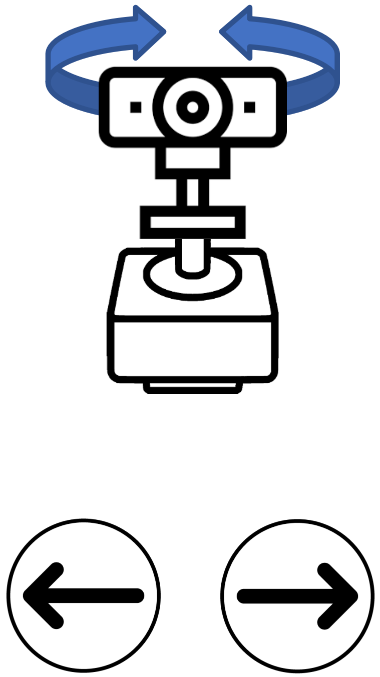
Service architecture

Android UI

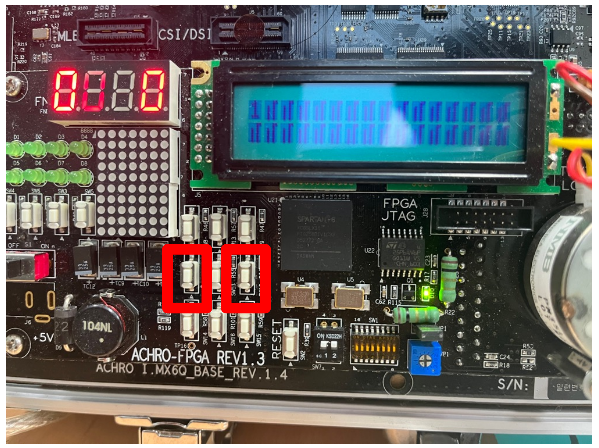


왼쪽 회전 촬영 버튼 오른쪽 회전

Hardware Design

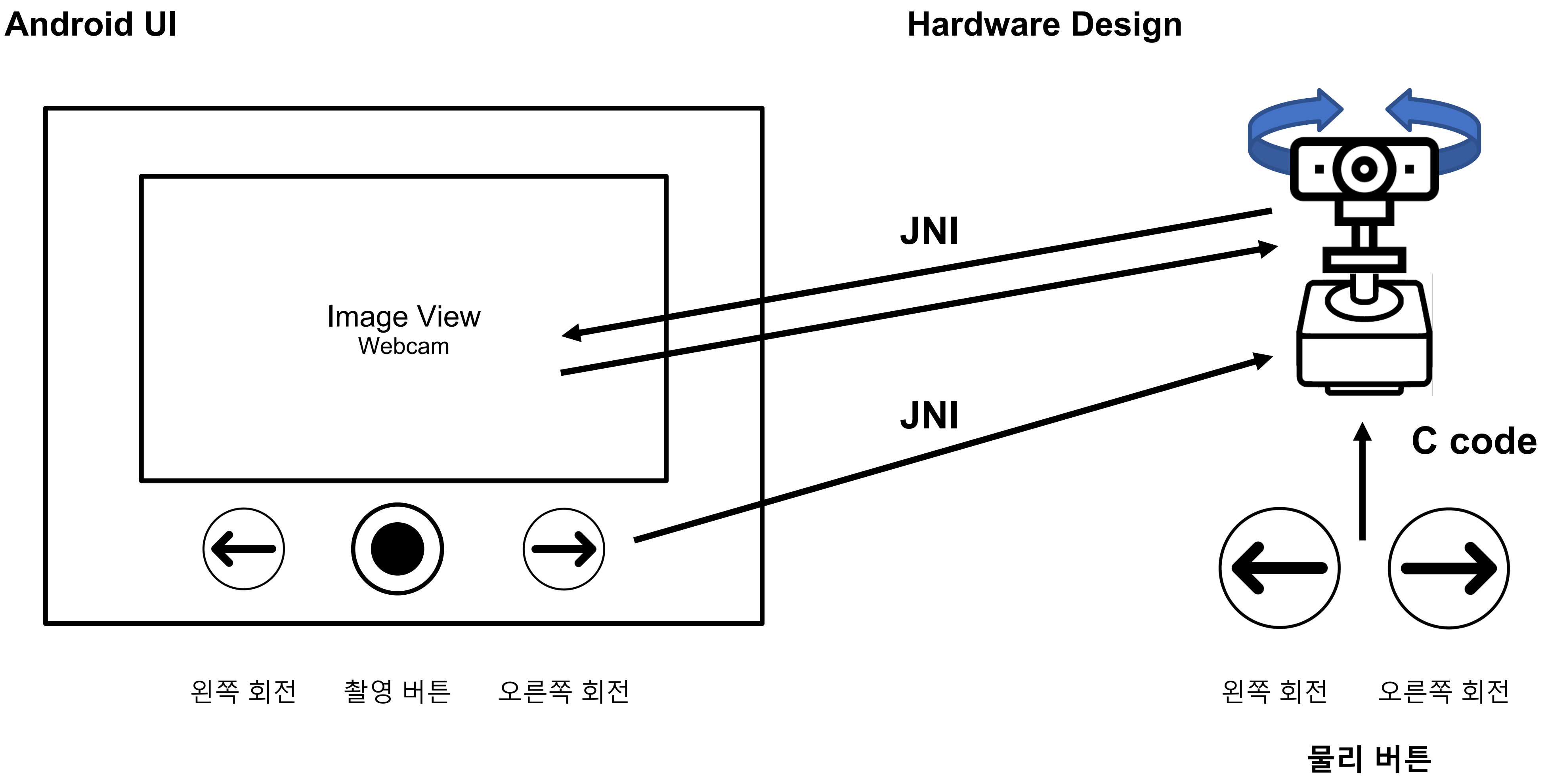


왼쪽 회전 오른쪽 회전

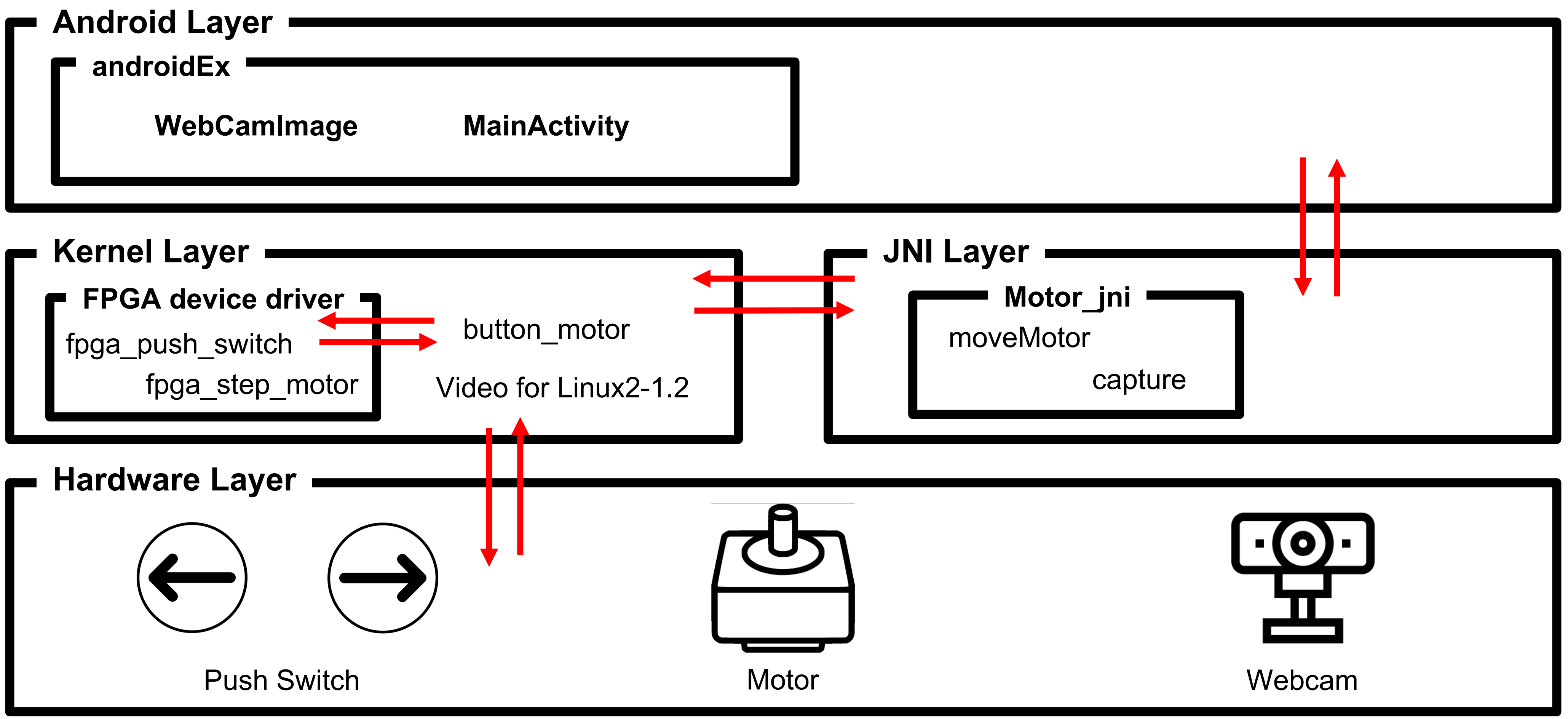


물리 버튼

Service architecture



Service architecture

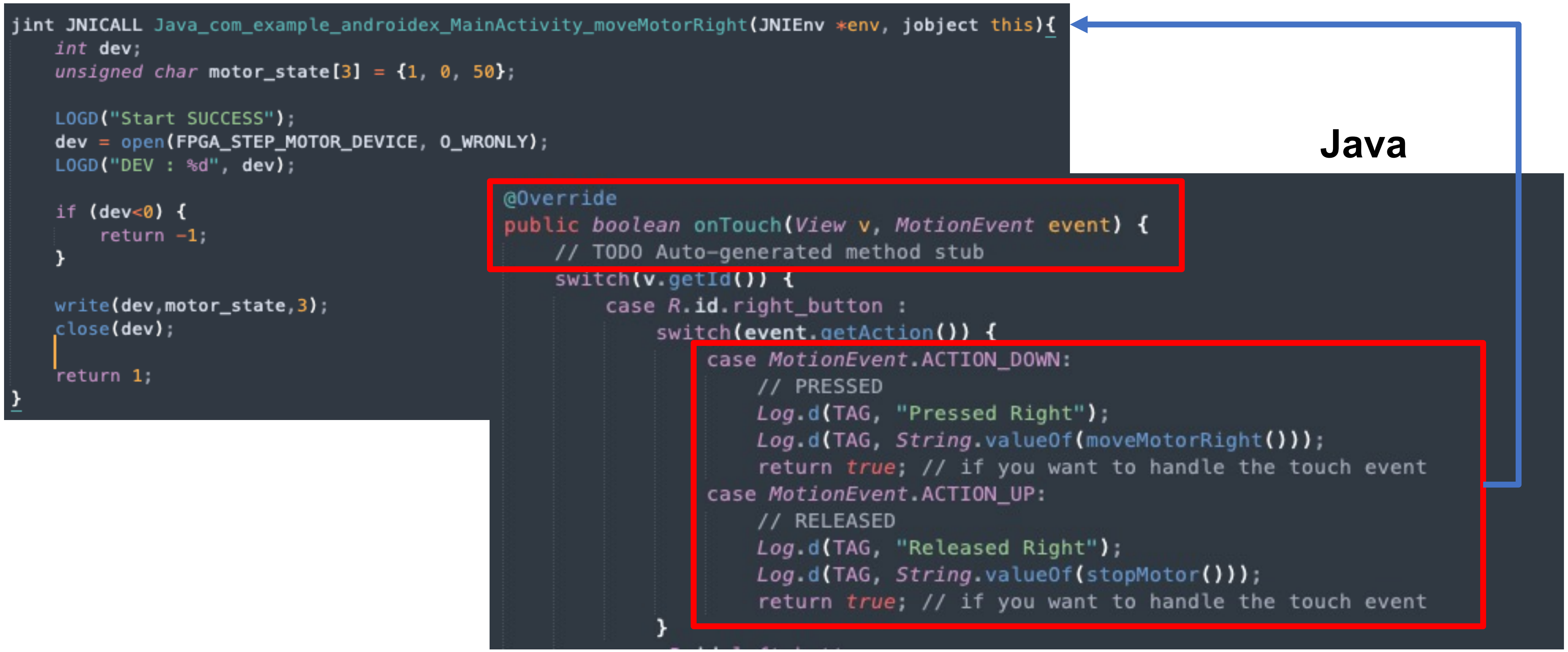


Implementation

Motor 구동(JNI)

JNI

Java

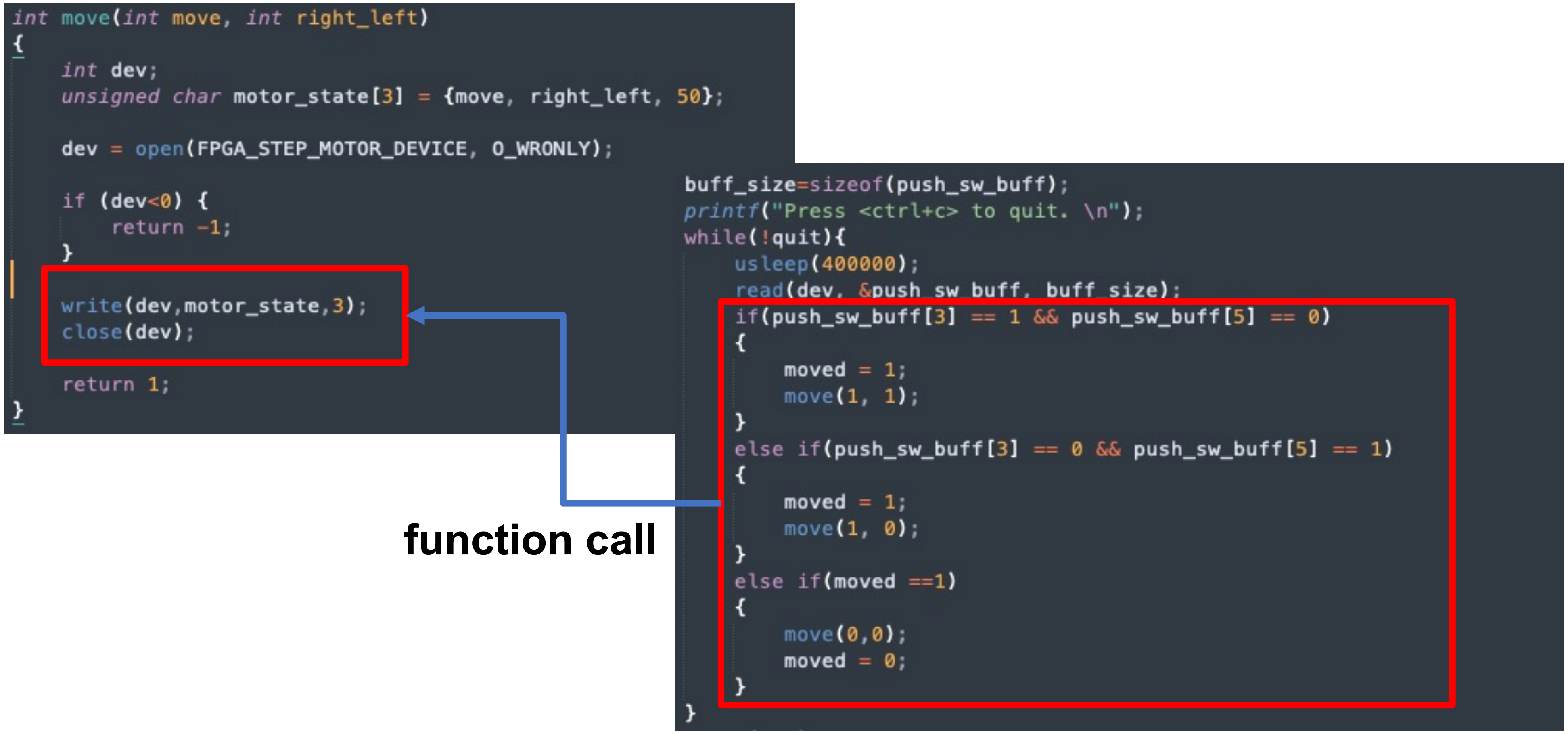


```c
jint JNICALL Java_com_example_androidex_MainActivity_moveMotorRight(JNIEnv *env, jobject this){
    int dev;
    unsigned char motor_state[3] = {1, 0, 50};

    LOGD("Start SUCCESS");
    dev = open(FPGA_STEP_MOTOR_DEVICE, O_WRONLY);
    LOGD("DEV : %d", dev);

    if (dev<0) {
        return -1;
    }

    write(dev,motor_state,3);
    close(dev);

    return 1;
}
```

```java
@Override
public boolean onTouch(View v, MotionEvent event) {
    // TODO Auto-generated method stub
    switch(v.getId()) {
        case R.id.right_button :
            switch(event.getAction()) {
                case MotionEvent.ACTION_DOWN:
                    // PRESSED
                    Log.d(TAG, "Pressed Right");
                    Log.d(TAG, String.valueOf(moveMotorRight()));
                    return true; // if you want to handle the touch event
                case MotionEvent.ACTION_UP:
                    // RELEASED
                    Log.d(TAG, "Released Right");
                    Log.d(TAG, String.valueOf(stopMotor()));
                    return true; // if you want to handle the touch event
            }
```

Implementation

Motor 구동(버튼)

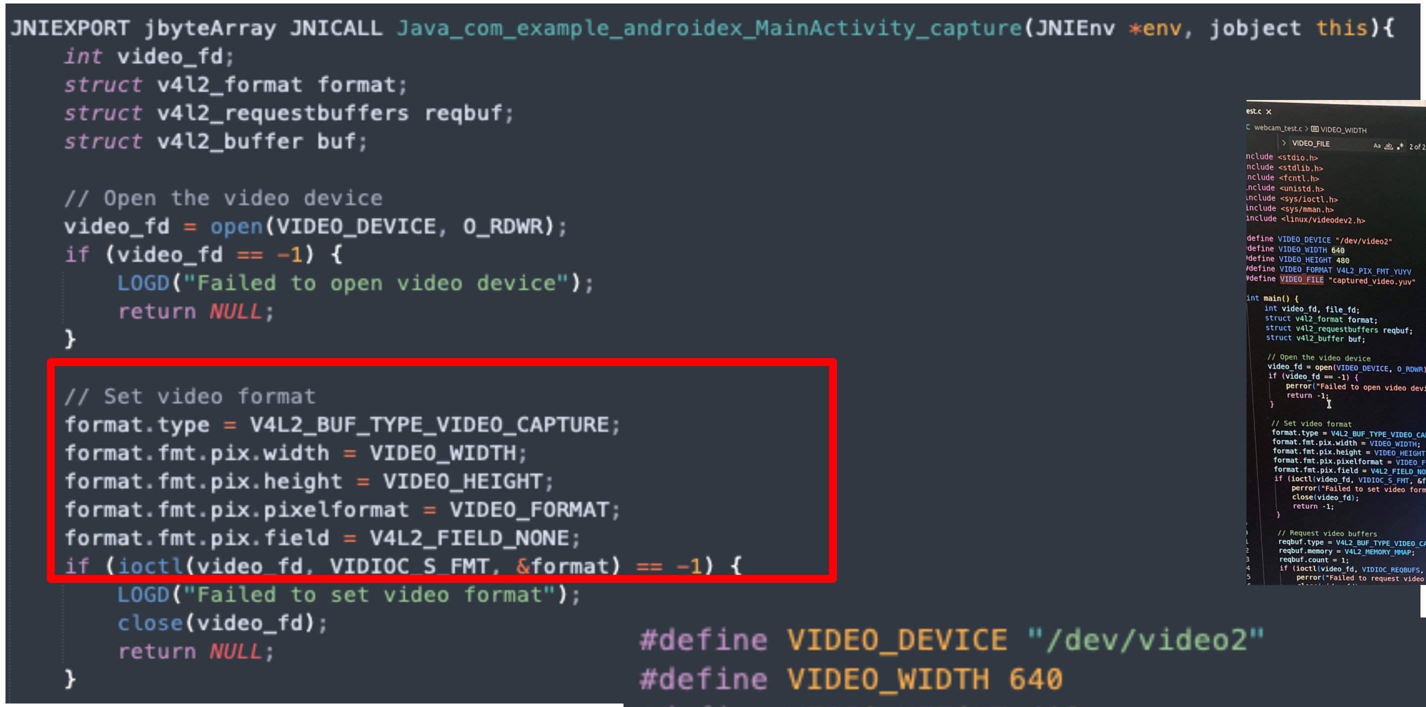


```c
int move(int move, int right_left)
{
    int dev;
    unsigned char motor_state[3] = {move, right_left, 50};

    dev = open(FPGA_STEP_MOTOR_DEVICE, O_WRONLY);

    if (dev<0) {
        return -1;
    }

    write(dev,motor_state,3);
    close(dev);

    return 1;
}
```

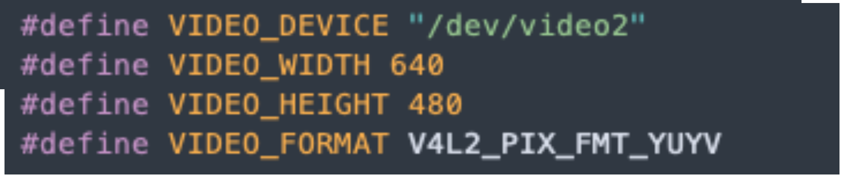
```c
buff_size=sizeof(push_sw_buff);
printf("Press <ctrl+c> to quit. \n");
while(!quit){
    usleep(400000);
    read(dev, &push_sw_buff, buff_size);
    if(push_sw_buff[3] == 1 && push_sw_buff[5] == 0)
    {
        moved = 1;
        move(1, 1);
    }
    else if(push_sw_buff[3] == 0 && push_sw_buff[5] == 1)
    {
        moved = 1;
        move(1, 0);
    }
    else if(moved ==1)
    {
        move(0,0);
        moved = 0;
    }
}
```

Implementation

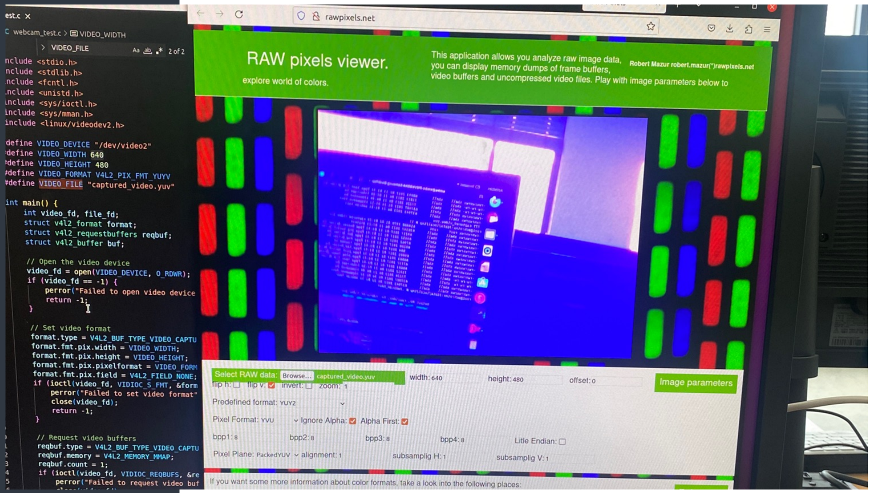
Webcam 이미지 촬영



```
JNIEXPORT jbyteArray JNICALL Java_com_example_androidex_MainActivity_capture(JNIEnv *env, jobject this){
    int video_fd;
    struct v4l2_format format;
    struct v4l2_requestbuffers reqbuf;
    struct v4l2_buffer buf;

    // Open the video device
    video_fd = open(VIDEO_DEVICE, O_RDWR);
    if (video_fd == -1) {
        LOGD("Failed to open video device");
        return NULL;
    }

    // Set video format
    format.type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
    format.fmt.pix.width = VIDEO_WIDTH;
    format.fmt.pix.height = VIDEO_HEIGHT;
    format.fmt.pix.pixelformat = VIDEO_FORMAT;
    format.fmt.pix.field = V4L2_FIELD_NONE;
    if (ioctl(video_fd, VIDIOC_S_FMT, &format) == -1) {
        LOGD("Failed to set video format");
        close(video_fd);
        return NULL;
    }
```



```
#define VIDEO_DEVICE "/dev/video2"
#define VIDEO_WIDTH 640
#define VIDEO_HEIGHT 480
#define VIDEO_FORMAT V4L2_PIX_FMT_YUYV
```

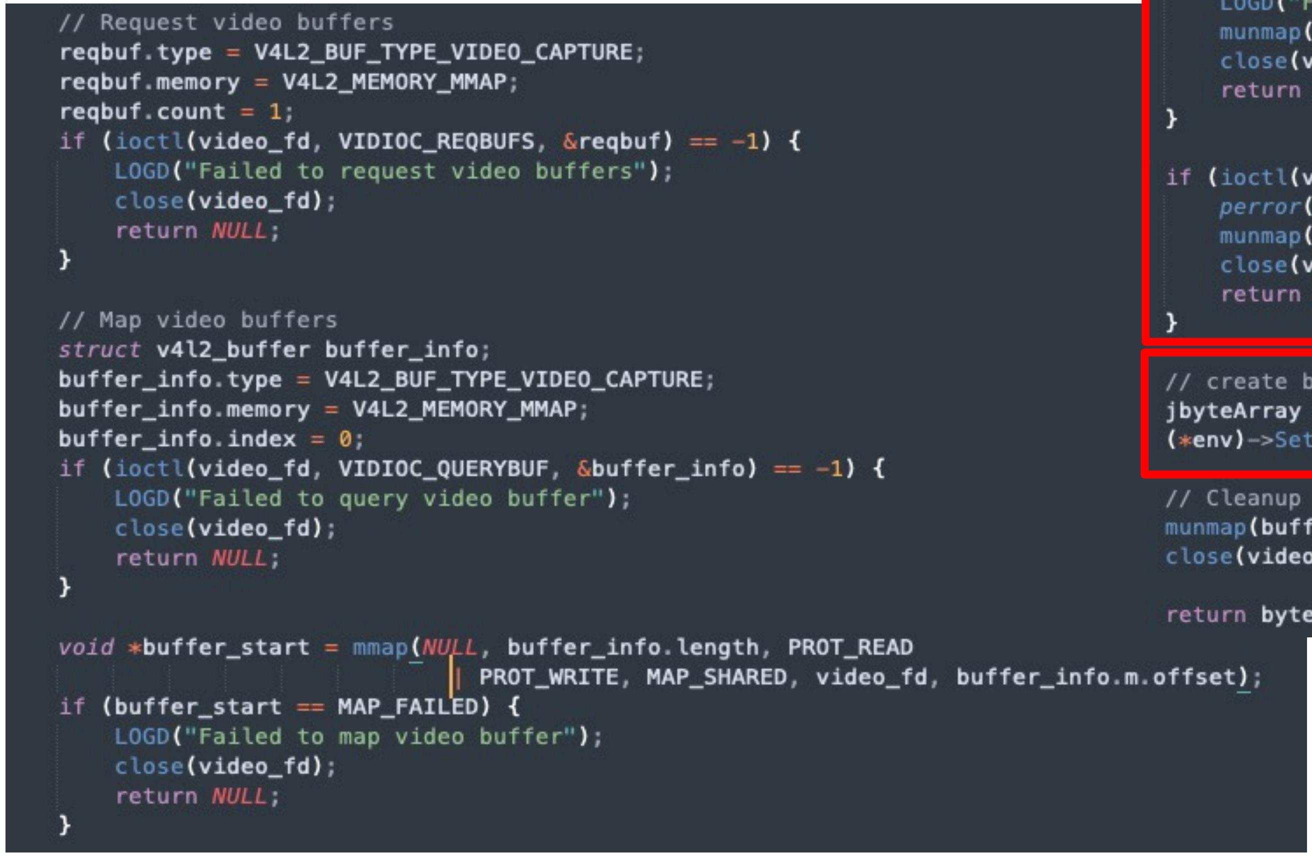


```
est.c ×
C webcam_test.c > VIDEO_WIDTH
  > VIDEO_FILE        Aa ab .*  2 of 2
nclude <stdio.h>
nclude <stdlib.h>
nclude <fcntl.h>
nclude <unistd.h>
include <sys/ioctl.h>
include <sys/mman.h>
include <linux/videodev2.h>

define VIDEO_DEVICE "/dev/video2"
define VIDEO_WIDTH 640
define VIDEO_HEIGHT 480
#define VIDEO_FORMAT V4L2_PIX_FMT_YUYV
#define VIDEO_FILE "captured_video.yuv"

int main() {
    int video_fd, file_fd;
    struct v4l2_format format;
    struct v4l2_requestbuffers reqbuf;
    struct v4l2_buffer buf;

    // Open the video device
    video_fd = open(VIDEO_DEVICE, O_RDWR);
    if (video_fd == -1) {
        perror("Failed to open video device
        return -1;
    }

    // Set video format
    format.type = V4L2_BUF_TYPE_VIDEO_CAPTU
    format.fmt.pix.width = VIDEO_WIDTH;
    format.fmt.pix.height = VIDEO_HEIGHT;
    format.fmt.pix.pixelformat = VIDEO_FORM
    format.fmt.pix.field = V4L2_FIELD_NONE;
    if (ioctl(video_fd, VIDIOC_S_FMT, &form
        perror("Failed to set video format"
        close(video_fd);
        return -1;
    }

    // Request video buffers
    reqbuf.type = V4L2_BUF_TYPE_VIDEO_CAPTU
    reqbuf.memory = V4L2_MEMORY_MMAP;
    reqbuf.count = 1;
    if (ioctl(video_fd, VIDIOC_REQBUFS, &re
        perror("Failed to request video buf
```

rawpixels.net
RAW pixels viewer.
explore world of colors.
This application allows you analyze raw image data, you can display memory dumps of frame buffers, video buffers and uncompressed video files. Play with image parameters below to
Select RAW data: Browse... captured_video.yuv
width: 640
height: 480
offset: 0
Image parameters
Predefined format: YUY2
Pixel Format: YVU
Ignore Alpha
Alpha First
bpp1: 8
Little Endian
Pixel Plane: PackedYUV
alignment: 1
subsamplig H: 1
subsamplig V: 1
If you want some more information about color formats, take a look into the following places:

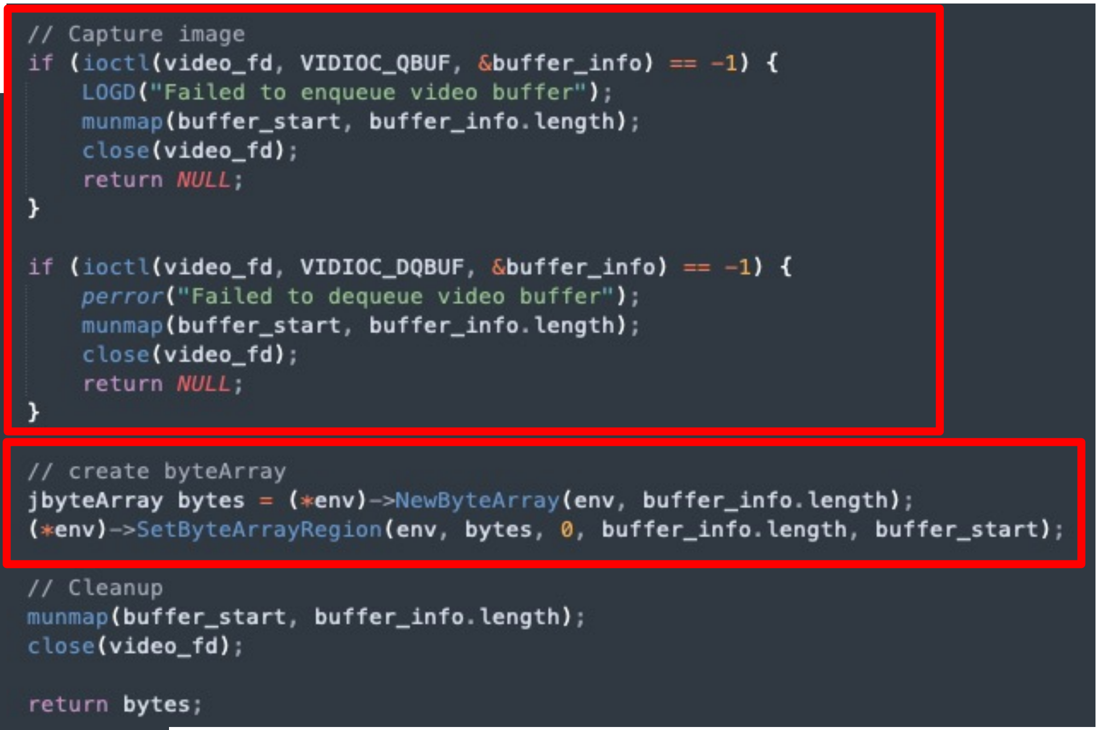
Implementation

Webcam 이미지 촬영



```
// Request video buffers
reqbuf.type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
reqbuf.memory = V4L2_MEMORY_MMAP;
reqbuf.count = 1;
if (ioctl(video_fd, VIDIOC_REQBUFS, &reqbuf) == -1) {
    LOGD("Failed to request video buffers");
    close(video_fd);
    return NULL;
}

// Map video buffers
struct v4l2_buffer buffer_info;
buffer_info.type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
buffer_info.memory = V4L2_MEMORY_MMAP;
buffer_info.index = 0;
if (ioctl(video_fd, VIDIOC_QUERYBUF, &buffer_info) == -1) {
    LOGD("Failed to query video buffer");
    close(video_fd);
    return NULL;
}

void *buffer_start = mmap(NULL, buffer_info.length, PROT_READ
                          | PROT_WRITE, MAP_SHARED, video_fd, buffer_info.m.offset);
if (buffer_start == MAP_FAILED) {
    LOGD("Failed to map video buffer");
    close(video_fd);
    return NULL;
}
```

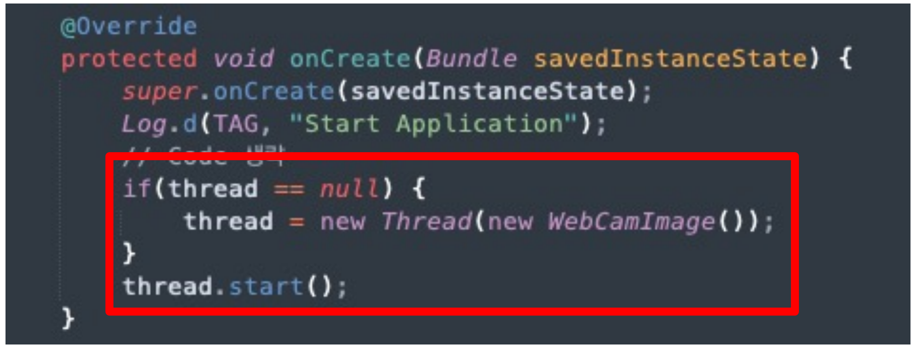
Image file을 jni의 response로 보내기 위해 jbyteArray를 활용



```
// Capture image
if (ioctl(video_fd, VIDIOC_QBUF, &buffer_info) == -1) {
    LOGD("Failed to enqueue video buffer");
    munmap(buffer_start, buffer_info.length);
    close(video_fd);
    return NULL;
}

if (ioctl(video_fd, VIDIOC_DQBUF, &buffer_info) == -1) {
    perror("Failed to dequeue video buffer");
    munmap(buffer_start, buffer_info.length);
    close(video_fd);
    return NULL;
}

// create byteArray
jbyteArray bytes = (*env)->NewByteArray(env, buffer_info.length);
(*env)->SetByteArrayRegion(env, bytes, 0, buffer_info.length, buffer_start);

// Cleanup
munmap(buffer_start, buffer_info.length);
close(video_fd);

return bytes;
```

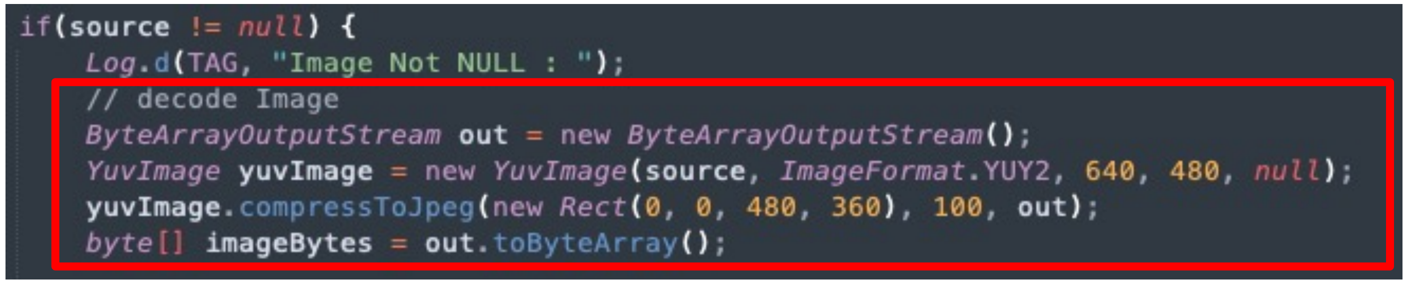
Implementation

ImageView update

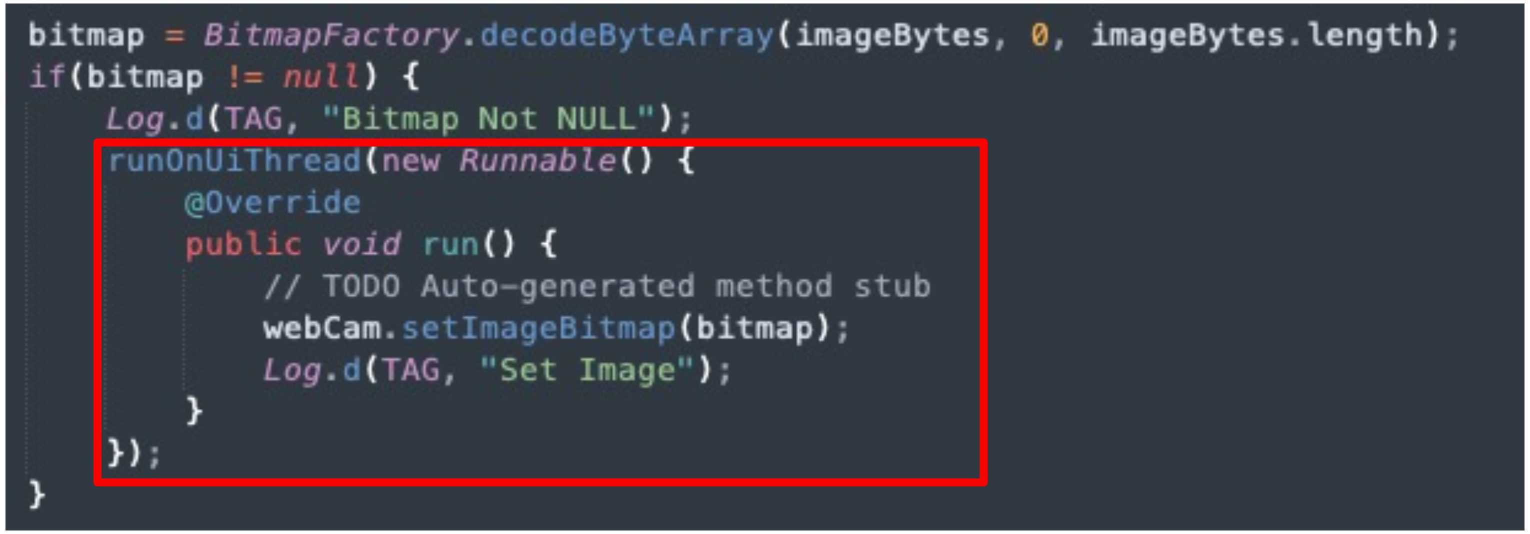


```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    Log.d(TAG, "Start Application");
    // Code 생략
    if(thread == null) {
        thread = new Thread(new WebCamImage());
    }
    thread.start();
}
```

Android가 yuv format을 지원하지 않아 jpg형태로 변환 후 imageView update



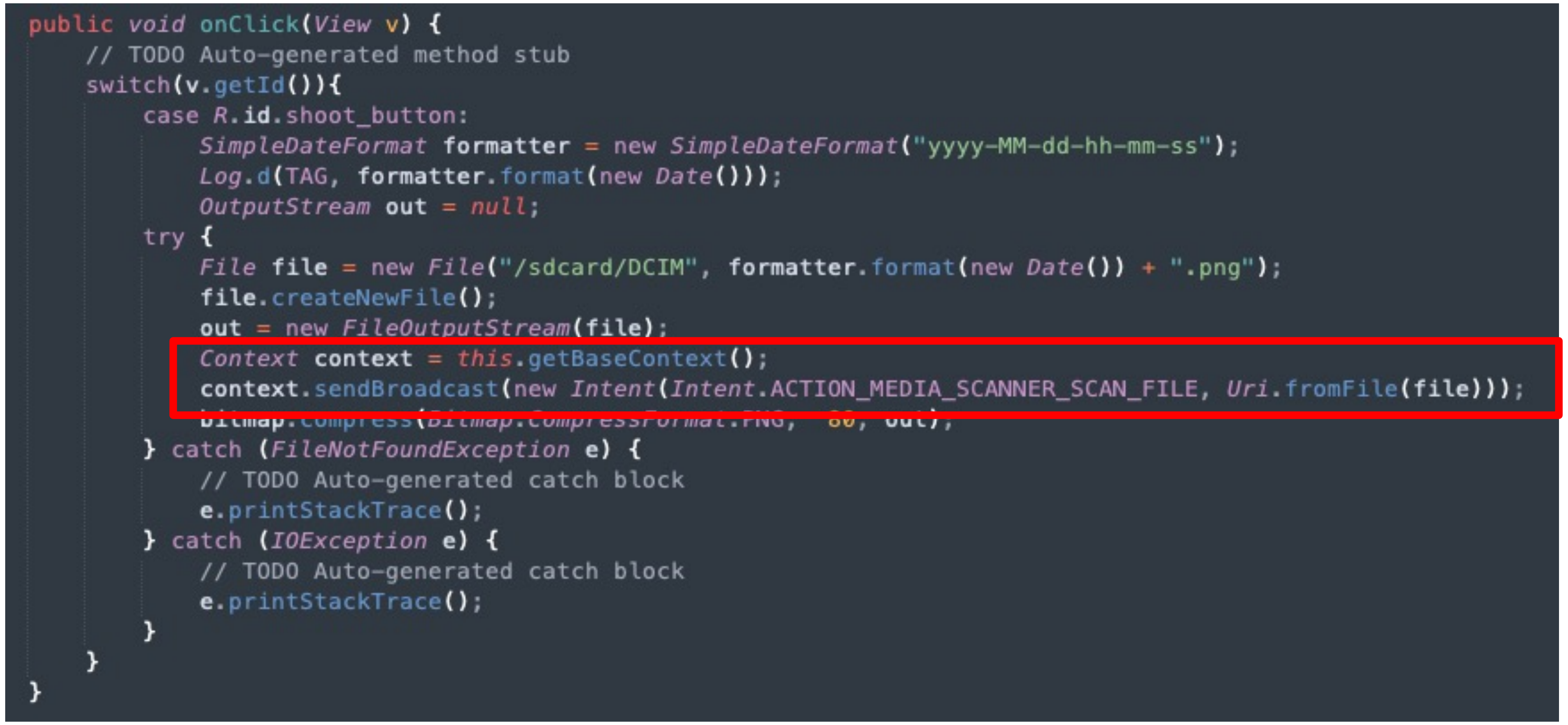
```
if(source != null) {
    Log.d(TAG, "Image Not NULL : ");
    // decode Image
    ByteArrayOutputStream out = new ByteArrayOutputStream();
    YuvImage yuvImage = new YuvImage(source, ImageFormat.YUY2, 640, 480, null);
    yuvImage.compressToJpeg(new Rect(0, 0, 480, 360), 100, out);
    byte[] imageBytes = out.toByteArray();
```



```
bitmap = BitmapFactory.decodeByteArray(imageBytes, 0, imageBytes.length);
if(bitmap != null) {
    Log.d(TAG, "Bitmap Not NULL");
    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            // TODO Auto-generated method stub
            webCam.setImageBitmap(bitmap);
            Log.d(TAG, "Set Image");
        }
    });
}
```

Implementation

Image 저장



```
public void onClick(View v) {
    // TODO Auto-generated method stub
    switch(v.getId()){
        case R.id.shoot_button:
            SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM-dd-hh-mm-ss");
            Log.d(TAG, formatter.format(new Date()));
            OutputStream out = null;
        try {
            File file = new File("/sdcard/DCIM", formatter.format(new Date()) + ".png");
            file.createNewFile();
            out = new FileOutputStream(file);
            Context context = this.getBaseContext();
            context.sendBroadcast(new Intent(Intent.ACTION_MEDIA_SCANNER_SCAN_FILE, Uri.fromFile(file)));
            bitmap.compress(Bitmap.CompressFormat.PNG, 80, out);
        } catch (FileNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```

저장된 file을 Disk로 broadcast

Implementation

결론

Linux의 기본 Device driver인 **Video for Linux2**를 활용하여, Webcam 영상을 화면에 송출하고, 수업시간에 배운 내용들을 활용하여, 모터와 버튼을 통해 Webcam을 조작하는 Application과 Embedded Device를 구현하였습니다.

Webcam Device의 Memory 구조, Data 처리 방식 등 **Device에 대한 정보가 없어**, 정확한 Format을 찾고, 해당 format을 android에서 지원하는 format으로 변환하는 과정에서 많은 Overhead가 있었습니다.

향후 외부 Device에서의 Control이나, Device Driver Custom 등 다양한 것들을 추가적으로 진행하면 좋은 공부가 될 것 같습니다.

Demonstration

시 연