KETI 전자부품연구원
Korea Electronics Technology Institute

SOFTWARE RELEASE NOTE

For LVRK-RK3399

2019.10.

VR/AR 연구센터

목차

1. 개요

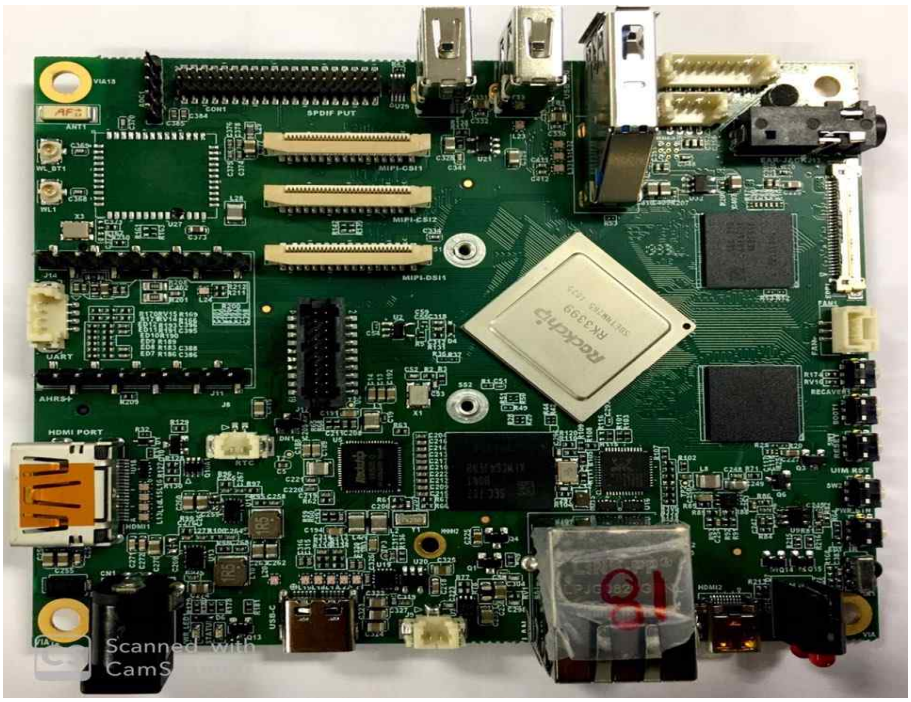
본 문서는 다음의 LVRK-3399 보드와 함께 제공하는 소프트웨어와 사용 방법을 포함하여 설명한다.

LVRK-3399 보드는 Rockchip사의 RK3399 기반 고성능 ARM 프로세싱 보드이고 다양한 포트와 인터페이스를 지원한다.

보드 크기는 105 x 90 mm 이고, 4GB LPDDR3 RAM, 64GB eMMC, 온보드 2.4G & 5G dual-band WiFi 모듈과 PCIe타입 LTE모뎀을 지원하며, GPU와 VPU 가속을 지원하며, 안드로이드 7.x & 8.x 버전을 지원한다.

비디오 입출력장치로 MIPI-CSI 듀얼 카메라 인터페이스, eDP 디스플레이, HDMI 2.0 인터페이스를 지원한다. Type-C/DP, USB3.0, USB2.0, MicroSD, Gbps Ethernet, 3.5mm audio jack 등이 마련되어 있다.

RK3399 SoC는 내장 Mali-T860 GTPU를 가지고 있어 강력한 3D 및 HD H.265/H.264 비디오 스트림을 처리할 수 있다. 듀얼 카메라 입력, 듀얼 ISP 인터페이스는 13MPix/s까지 이미지 처리를 할 수 있어 AI 및 Deep Learning을 위한 완벽한 플랫폼이 될 수 있다.



2. 하드웨어 스펙

- SoC: Rockchip RK3399
  - CPU: big.LITTLE , Dual-Core Cortex-A72(up to 2.0GHz) + Quad-Core Cortex-A53(up to 1.5GHz)
  - GPU: Mali-T864 GPU , supports OpenGL ES1.1/2.0/3.0/3.1, OpenVG1.1, OpenCL, DX11, and AFBC
  - VPU: 4K VP9 and 4K 10bits H265/H264 60fps decoding, Dual VOP, etc
- PMU: RK808-D PMIC, cooperated with independent DC/DC, enabling DVFS, sofeware power-down, RTC wake-up, system sleep mode
- RAM: Dual-Channel 4GB LPDDR3-1866
- Flash: 64GB eMMC 5.1 Flash
- Ethernet: Native Gigabit Ethernet
- Wi-Fi/BT: 802.11a/b/g/n/ac, Bluetooth 4.1, Wi-Fi and Bluetooth combo module, 2x2 MIMO, dual antenna interface
- Video Input: one or two 4-Lane MIPI-CSI, dual ISP, up to 13MPix/s , supports simultaneous input of dual camera data
- Video output
  - HDMI: HDMI 2.0a, supports 4K@60Hz , HDCP 1.4/2.2
  - DP on Type-C: DisplayPort 1.2 Alt Mode on USB Type-C
  - LCD Interface: one eDP 1.3 (4-Lane , 10.8Gbps) , one or two 4-Lane MIPI-DSI
- Audio Out: 3.5mm Dual channel headphone jack, or HDMI
- Audio In: 3-Pin 2.54mm microphone interface
- USB 2.0: 2 independent native USB 2.0 Host A interfaces
- USB 3.0: 1 native USB 3.0 Host A type interface
- USB Type-C: Supports USB3.0 Type-C and DisplayPort 1.2 Alt Mode on USB Type-C
- PCIe: One M.2 M-Key PCIe x4 socket, compatible with PCIe 2.1, Dual operation mode; Onboard M3 PCB nut for mounting M.2 2280 module
- microSD Slot x 1
- Debug: one Debug UART, 4 Pin 2.54mm header, 3V level, 1500000bps
- Keys: PowerKey, Reset, MASKROM(BOOT), Recovery
- IR reciver: Onboard IR reciver, Acceptes 38KHz carrier frequency
- RTC Battery: 2 Pin 1.27/1.25mm RTC battery input connector

- Cooling: two 2.5mm PCB nuts for mounting heat sink; 3 Pin 12V cooling fan interface with PWM
- Power supply: DC 12V/2A
- Ambient Operating Temperature: -20℃ to 70℃

3. 준비물

응용 프로그램 개발자가 보드 사용 전에 필요한 준비물들은 다음과 같다.

- LVRK-3399 보드
- Type-C 케이블
- MicroSD 카드: Class 10 or 이상, 최소 8GB SDHC
- USB top Serial 아답터(optional, for debugging or access from PC host)
- DC12V/2A 파워 아답터
- HDMI 모니터 및 케이블
- USB 키보드, 마우스, 허브 등
- 호스트 컴퓨터 (Ubuntu 18.04 or 16.04.5 LTS 64bit)

4. Access Hardware

4.1 Access Serial Interface

사용자는 현재 UART4만 추가적으로 할당하여 사용이 가능하다.

Serial Interface	Serial Device
UART0	Used by Bluetooth
UART1	Used by Gbps Ethernet
UART2	Used by Serial Debug Port
UART3	Used by Gbps Ethernet
UART4	Available, device name is /dev/ttyS4

4.2 Flash Image to eMMC

LVRK-3399는 eMMC에 이미지를 다운로드하기위해 다음의 3가지 방법이 있다.

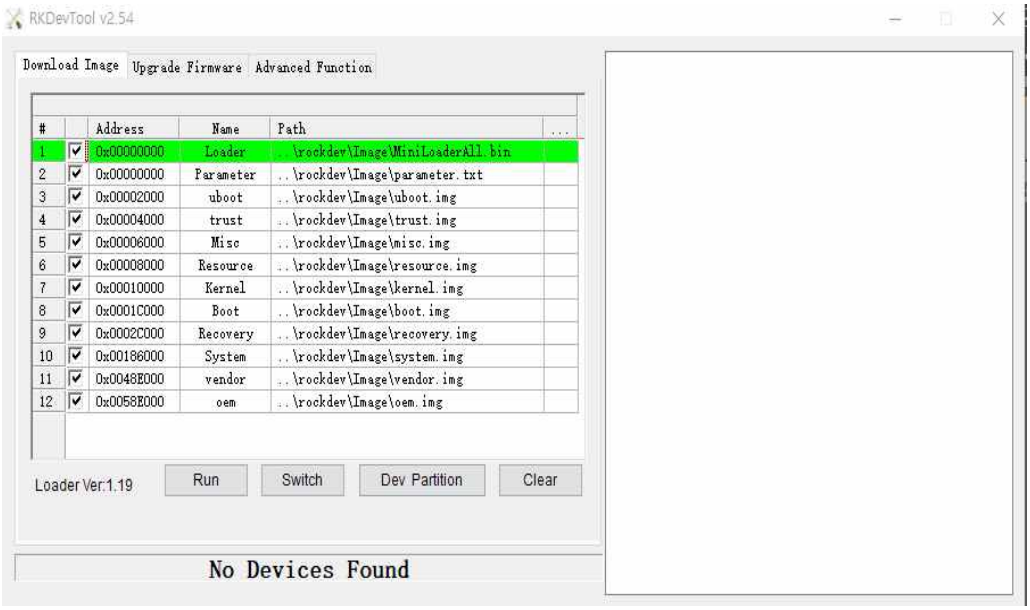
- Rockchip사에서 제공하는 윈도우용 유틸리티 "AndroidTool_Release_v2.54"를 사용하여 eMMC에 Type-C를 통하여 플래쉬 한다.
- Rockchip사에서 제공하는 리눅스용 유틸리티 "Linux_Upgrade_Tool_1.27"을 사용하여 Type-C를 통하여 eMMC에 플래쉬 한다.
- EFlasher를 사용하여 부팅용 SD카드를 만들어, 이미지를 eMMC에 플래쉬한다.

본 문서는 윈도우용 유틸리티인 "AndroidTool_Re3leasae_v2.54"를 사용하여 플래쉬하는 방법을 기술한다.

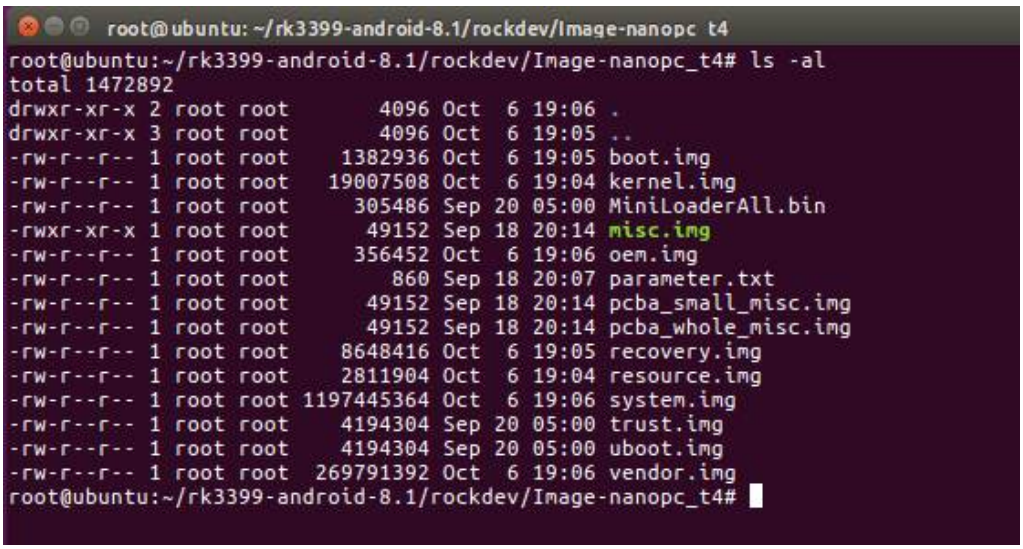
- "DriverAssistant_v4.5.tgz"를 다운로드하고, 압축을 풀어 PC에 설치한다.
- "AndroidTool_Release_v2.54.zip"을 다운로드하고, 압축을 풀어 Administrator로 실행을 한다.
- AndroidTool을 실행하면 초기 언어 설정이 중문으로 되어있는데, 필요시 config.ini 파일을 아래와 같이 "Selected=2"로 수정하여 영문으로 변경할 수 있다.

```
[Language]
Kinds=2
Selected=2
LangPath=Language\

Lang1File=Chinese.ini
Lang1FontName=宋体
Lang1FontSize=9

Lang2File=English.ini
Lang2FontName=Arial
Lang2FontSize=9
```

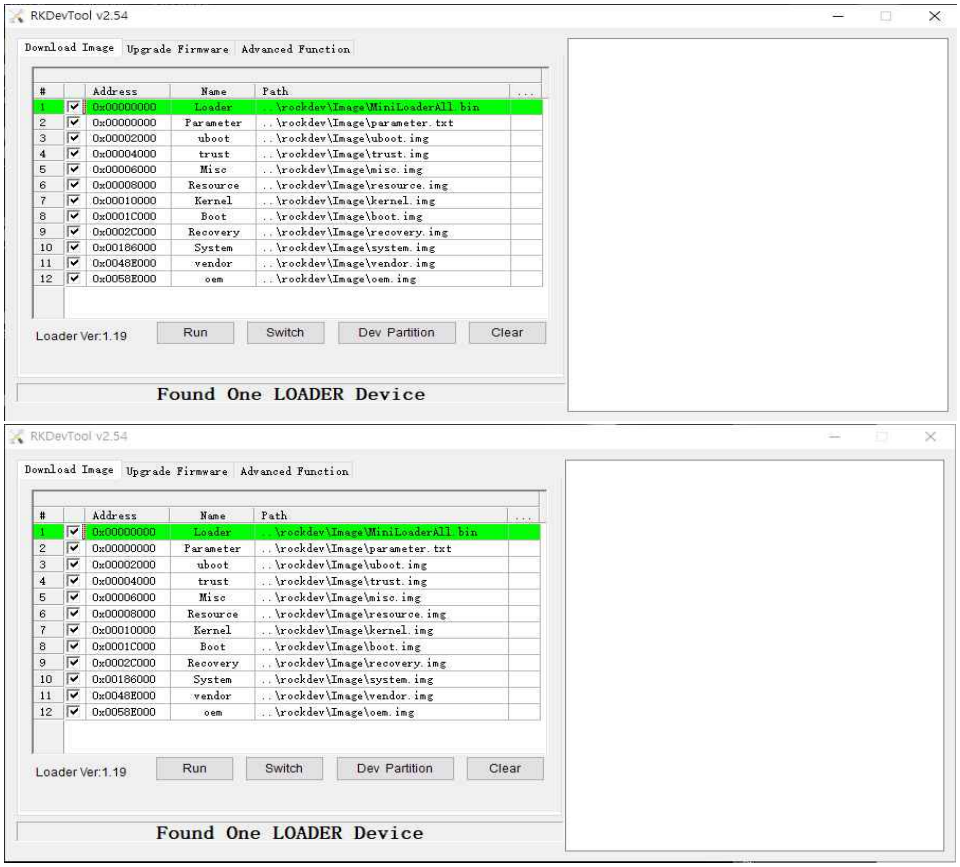
- 다음 그림과 같이 AndroidTool을 관리자 권한으로 실행을 하고, "Download Image" 탭 화면에서 각 이미지 파일을의 경로를 설정해 준다.



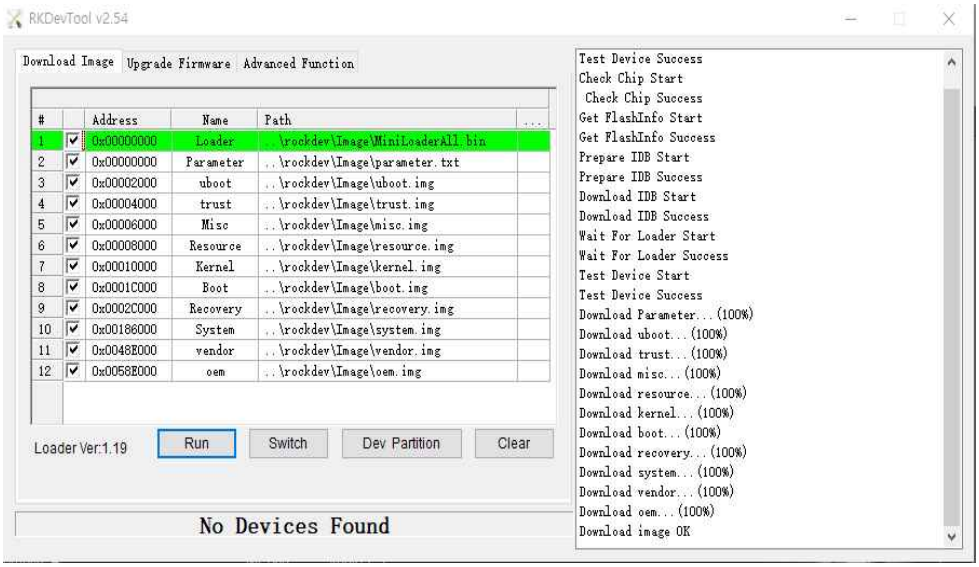
- 업데이트된 바이너리 이미지를 플래쉬하기 위해서 안드로이드 소스를 빌드 후 "/rockdev/Image-nanopc-t4" 디렉토리 아래에 다음과 같은 이미지 바이너리파일들이 생성한다. 다음 그림에서 보이는 파일들을 모두 복사하여 "AndroidTool_Release_v2.54/rockdev" 디렉토리에 복사하면 별도의 경로설정 없이 디폴트로 사용할 수도 있다.



- LVRK-3399 보드에 Type-C 케이블을 꼽아 PC와 연결하고, 전원을 인가하면 부팅이 시작된다. "Recovery" 버튼을 누르고, 동시에 "Reset" 버튼을 누르고 1초정도 후에 "POWER" 버튼을 2초정도 누르고 떼면, 창 하단부에 "Found One LOADER Device" 혹은 이미지가 훼손된 경우나 다운로드 최초의 경우 "Found One MASKROM Device" 메시지가 표시된다.



- 이미지 경로 설정을 마치고 "Run"버튼을 클릭하면, 플래쉬 과정이 진행된다. 디바이스 테스트 과정을 성공하면 이미지 다운로드가 진행되고, 문제가 없이 완료되면 시스템이 자동으로 리부팅된다.



5. Android 8.1 Build

5.1 개발환경 셋업

안드로이드 소스를 컴파일하기 위해서 Ubuntu 16.04 버전을 권장하고, 다음의 패키지들을 설치한다.

```
sudo apt-get install bison g++-multilib git gperf libxml2-utils make python-networkx zip
sudo apt-get install flex curl libncurses5-dev libssl-dev zlib1g-dev gawk minicom
sudo apt-get install openjdk-8-jdk
sudo apt-get install exfat-fuse exfat-utils device-tree-compiler liblz4-tool
```

5.2 크로스 컴파일러 설치

리눅스 커널과 u-boot를 컴파일하기위해 "aarch64-linux-gcc 6.4' 버전을 설치한다.

```
git clone https://github.com/friendlyarm/prebuilts.git -b master --depth 1
cd prebuilts/gcc-x64
cat toolchain-6.4-aarch64.tar.gz* | sudo tar xz -C /
```

그리고 컴파일러의 디렉토리를 ~/.bashrc 파일의 PATH 변수에 다음과 같이 추가하고, 반영이 되도록 쉘 커맨드라인에서 실행을 한다.

```
export PATH=/opt/FriendlyARM/toolchain/6.4-aarch64/bin:$PATH
export GCC_COLORS=auto
```

```
. ~/.bashrc
```

설치한 컴파일러는 64bit용이며, 정상 설치되었는지 다음과 같이 확인하도록 한다.

```
aarch64-linux-gcc -v
Using built-in specs.
COLLECT_GCC=aarch64-linux-gcc
COLLECT_LTO_WRAPPER=/opt/FriendlyARM/toolchain/6.4-
aarch64/libexec/gcc/aarch64-cortexa53-linux-gnu/6.4.0/lto-wrapper
Target: aarch64-cortexa53-linux-gnu
Configured with: /work/toolchain/build/aarch64-cortexa53-linux-
gnu/build/src/gcc/configure --build=x86_64-build_pc-linux-gnu
--host=x86_64-build_pc-linux-gnu --target=aarch64-cortexa53-linux-gnu
--prefix=/opt/FriendlyARM/toolchain/6.4-aarch64
--with-sysroot=/opt/FriendlyARM/toolchain/6.4-aarch64/aarch64-
cortexa53-linux-gnu/sysroot --enable-languages=c,c++
```

```
--enable-fix-cortex-a53-835769 --enable-fix-cortex-a53-843419 --with-cpu=cortex-a53
...
Thread model: posix
gcc version 6.4.0 (ctng-1.23.0-150g-FA)
```

5.3 소스코드 설치

본 문서와 별도로 제공해드리는 경로에서 소스코드 압축파일을 다운로드하고 압축을 풀어 설치한다. 빌드 서버는 Ubuntu 16.04.5 LTS 버전을 사용할 것을 권장하며, 디스크 용량은 최소 250GB 이상, 램은 8GB이상 사용할 것을 권장한다.

리눅스 전용 머신 및 윈도우 환경에서의 가상 머신에서의 개발이 모두 가능하다. 가상 머신의 경우 메모리 용량이 16GB정도로 충분하면 좋다.

5.4 이미지 파일 생성

직접 코드 수정을 하여 안드로이드 소스코드를 컴파일할 수 있으며, 또한 업데이트된 이미지 파일을 생성할 수 있다.

```
cd rk3399-android-8.1
./build-nanopc-t4.sh –F -M
```

빌드 옵션을 다르게 하여 각 모듈별 빌드 및 이미지를 생성할 수도 있다. 다음은 현재 가능한 옵션을 나열하였다. 빌드 쉘스크립트명 뒤에 "-F –M"대신에 아래 옵션 중에 다른 것을 사용하여도 된다.

```
function usage()
{
        echo "Usage: $0 [ARGS]"
        echo
        echo "Options:"
        echo "    -a              build Android"
        echo "    -B              build U-Boot"
        echo "    -K              build Linux kernel"
        echo "    -W               build Wi-Fi drivers (.ko)"
        echo
```

```
        echo "   -F, --all   build all (U-Boot, kernel, wifi, Android)"
        echo "   -M           make rockdev image"
        echo "   -u           generate update.img"
        echo
        echo "   -h           show this help message and exit"
        exit 1
}
```

5.3 시스템 업데이트

소스코드 컴파일이 완료되면 새로운 이미지 파일들이 "rockdev/image-nanopc-t4/" 디렉토리에 생성된다. Type-C를 사용하여 AndroidTool을 사용해도 되고,

EFlasher를 사용해 만든 부팅용 SD카드를 Ubuntu 머신에 연결하면 자동으로 마운트시키고, "rockdev/image-nanopc-t4/" 디렉토리 아래의 모든 파일을 복사하여 SD카드의 "FRIENDLYARM" 파티션의 android8 디렉토리에 붙여 넣기 한다. 이후 SD카드를 LVRK-3399 보드에 넣고 Flash과정을 실행한다.

제품 대량 양산을 위해 bootable SD카드를 사용해야 한다면, SD를 사용한 업데이트와 관련해 좀 더 자세한 내용은 아래 링크를 통하여 참조하도록 한다.

https://github.com/friendlyarm/sd-fuse_rk3399

6. Customization

6.1 Logo On/Off

'device/rockchip/common/BoardConfig.mk' 파일에서 다음과 같이 수정한다.

```
Change:
BOOT_SHUTDOWN_ANIMATION_RINGING := false
to:
BOOT_SHUTDOWN_ANIMATION_RINGING := true
```

6.2 Boot Logo and Animation

안드로이드 소스 디렉토리에서 다음의 파일들을 새로이 만들어 대체시킨다. 해당 파일을 만드는 상세 내용은 본 문서에서는 다루지 않으며, LVRK-3399보드 제공 시에 샘플로 KETI로고 이미지를 탑재하여 제공하였다. Bootanimation.zip 및 Shutdownanimation.zip 샘플은 필요시 Github등의 저장소를 통하여 제공할 수 있으며, 자체적으로 필요한 애니메이션 파일을 마련할 수 있다고 본다. 중요한 것은 zip파일을 생성시 압축을 하지 않는 옵션을 선택해야한다.

Boot animation

Kernel/logo.bmp Kernel/logo_kernel.bmp Device/rockchip/common/bootshutdown/bootanimation.zip

Shutdown animation

Device/rockchip/common/bootshutdown/shutdownanimation.zip

Animation zip파일을 만드는 방법들은 각종 인터넷에 정보가 많다. 그 중에 아래 웹사이트를 참조하면 좋을 듯 하다. 제작된 zip파일은 위 경로에 위치하여 이미지를 빌드하면 된다.

https://blog.naver.com/PostView.nhn?blogId=wlscjf518&logNo=220278976242

6.3 Audio Codec Path Route

LVRK-3399 보드는 오디오 코덱을 ALC5651을 사용하였으며, 3.5mm헤드셋을 통하여 스테레오 스피커 및 핸즈프리 마이크를 지원한다.

핸즈프리 마이크는 오디오 코덱의 IN3P에 연결하였다. 해당 입력 포트를 사용하기 위해 리눅스 커널드라이버와 안드로이드 설정파일에서 Path Route와 볼륨 값 등을 수정하여 반영하였다. 응용프로그램의 필요에 따라 볼륨 레벨을 튜닝하여 사용할 수 있다.

Kernel/sound/soc/codecs/rt5651.c Hardware/rockchip/audio/tinyalsa_hal/codec_config/rt5651_config.h

레코딩 볼륨을 조절하기 위해서는 위 드라이버 파일에서 고정하기에 앞서 아래처럼 디버

깅용 터미널 창이나 adb쉘을 통하여 amix 명령어를 사용해 테스트 해 볼 수 있다.

```
Amix "IN3 Boost" 8
Amix "ADC Capture Volume" 100
```

7. Driver Libraries for Visual and Pose Tracking

7.1 USB2.0 UVC Camera Driver Library and Demo Applications

2019년 9월중에 Preliminary Release로 소스 저장소를 제공하였으며, LVRK-3399보드에서 USB2.0 인터페이스를 통하여 카메라를 안드로이드 응용프로그램에서 사용할 수 있는 드라이버 및 샘플프로그램을 제공한다.

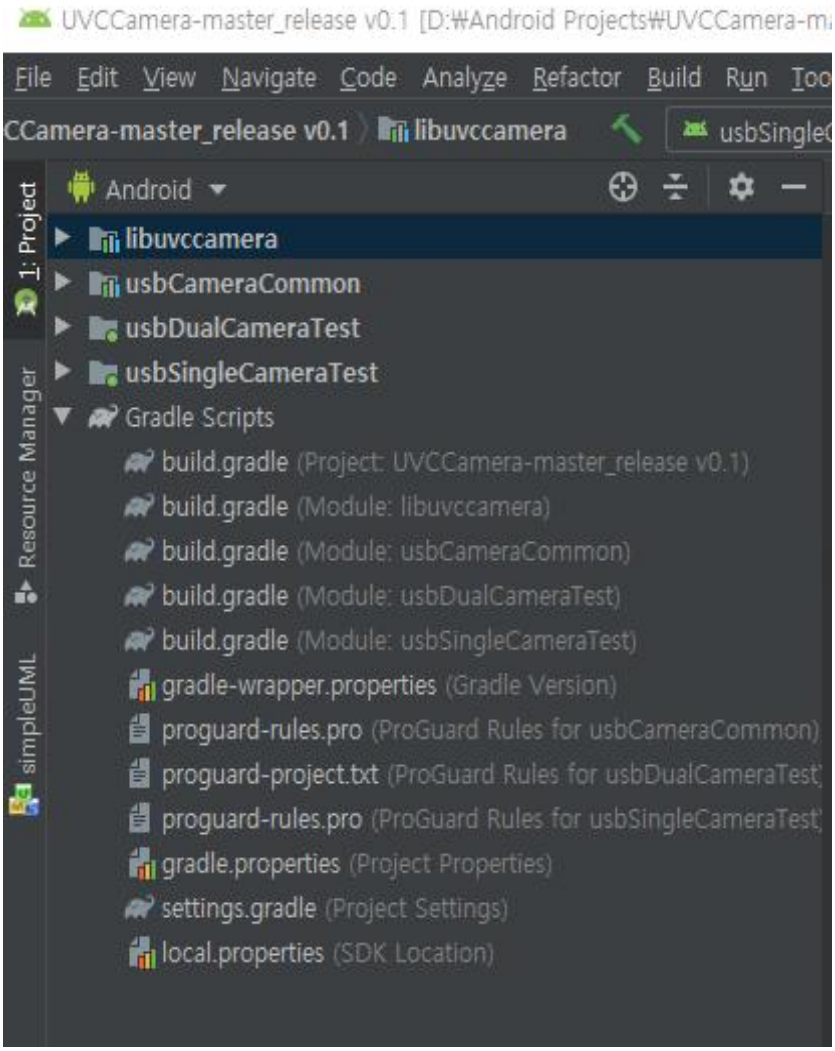
다음은 해당 github주소이다. 최신 소스코드를 다운로드 할 수 있으며, 일반적인 USB2.0 WebCAM이 동작할 수 있는 라이브러리이다. USB3.1기반 4K고화소 카메라는 현재 지원하지 않으며 추가 확장 개발이 필요하다.

https://github.com/jackcha76/UVCCamera-master_release-v0.1

7.1.1 라이브러리 및 어플리케이션 컴파일

Gradle 빌드시스템이 NDK파트를 포함하여 전체 프로젝트를 빌드 할 것이며, 아래 그림과 같이 라이브러리로 제공된 프로젝트의 libuvccamera, usbCameraCoomon 모듈을 프로젝트에 포함시킨다.

- UVCCamera 프로젝트 구조:



안드로이드 스튜디오가 local.properties 파일을 생성시키는데, 실제 설치된 SDK 및 NDK를 알맞게 설정한다. 제공하는 프로젝트 소소는 NDK-r12b 버전을 사용하였으며, 최신버전 NDK는 에러가 있었다.

변경 전:

```
## This file must *NOT* be checked into Version Control Systems,
# as it contains information specific to your local configuration.
#
# Location of the SDK. This is only used by Gradle.
# For customization when using a Version Control System, please read the
# header note.
```

```
#Thu Oct 17 17:11:15 KST 2019
sdk.dir=C\:\\Users\\PC2\\AppData\\Local\\Android\\Sdk
```

변경 후:

```
## This file must *NOT* be checked into Version Control Systems,
# as it contains information specific to your local configuration.
#
# Location of the SDK. This is only used by Gradle.
# For customization when using a Version Control System, please read the
# header note.
#Wed Aug 14 16:11:20 KST 2019
#ndk.dir=C\:\\Users\\PC2\\AppData\\Local\\Android\\Sdk\\ndk-bundle
ndk.dir=C\:\\Users\\PC2\\AppData\\Local\\Android\\android-ndk-r12b
sdk.dir=C\:\\Users\\PC2\\AppData\\Local\\Android\\Sdk
```

위같이 수정하여 프로젝트를 Synchronize 및 Make project 를 실행한다.

7.1.2 사용법

카메라 라이브러리를 사용하기 위해서는 프로젝트로 같이 포함하여 제공한 usbSingleCameraTest 및 usbDualCameraTest 샘플 프로젝트들을 잘 참조하여 이해할수 있도록 한다.

Android API14 이상 호환되고, USB host function 기능이 필수로 요구된다.

제공하는 프로젝트는 Saki4510t/UVCCamera 프로젝트를 기본으로 하였으며, UVC호환 카메라를 지원하고, 응용프로그램 개발자가 몇 개의 단순한 API만을 사용하여 USB카메라를 쉽게 사용할 수 있도록 해준다.

라이브러리를 사용하여 USB카메라의 Detect 및 Connect를 간단하게 할 수 있으며, 사진을 찍거나 mp4 비디오를 녹화하면서 해상도를 바꾸고, Contrast/Brightness 등을 변경할 수 있다.

- **Add to your Android project**

step 1. build.gradle 파일에 Saki4510t repository 추가

```
allprojects {
   repositories {
      …
```

```
      maven { url
'http://raw.github.com/saki4510t/libcommon/master/repository/' }
   }
}
```

Step 2. libuvccamera/usbCameraCommon 프로젝트 Dependency를 추가

```
dependencies {
…
   implementation("com.serenegiant:common:${commonLibVersion}") {
            exclude module: 'support-v4'
   }
}
```

Step 3. usbCameraCommon 프로젝트에 libuvccamera Dependency 추가

```
dependencies {
…
   implementation project(':libuvccamera')
}
```

Step 4. Sample App 프로젝트에 libuvccamera 및 usbCameraCommon dependencies 추가

```
dependencies {
…
   implementation project(':libuvccamera')
   implementation project(':usbCameraCommon')
}
```

- **API Introduction**

어플리케이션에서 라이브러리를 올바르게 사용하기 위해서는 다음의 CameraViewInterface, USBMonitor 및 UVCCameraHandler 인스턴스를 onCreate() 메소드에서 생성한다.

예로 usbSingleCameraTest 프로젝트에서 MainActivity.java를 살펴보자.

```
@Override
   protected void onCreate(final Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
…
      final View view = findViewById(R.id.camera_view);
      view.setOnLongClickListener(mOnLongClickListener);
      mUVCCameraView = (CameraViewInterface)view;
```

```
      mUVCCameraView.setAspectRatio(PREVIEW_WIDTH / (float)PREVIEW_HEIGHT);
…

      mUSBMonitor = new USBMonitor(this, mOnDeviceConnectListener);
      mCameraHandler = UVCCameraHandler.createHandler(this,
mUVCCameraView, USE_SURFACE_ENCODER ? 0 : 1, PREVIEW_WIDTH,
PREVIEW_HEIGHT, PREVIEW_MODE);
```

onStart(), onStop(), onDestroy() 메소드들을 오버라이딩하여 재정의하고, UI에 배치한 버튼들의 이벤트 핸들러를 다음과 같이 등록한다. 핸들러 등록은 어플리케이션 개발자가 선호하는 방식으로 작성하도록 한다.

```
/**
 * event handler when click camera / capture button
 */
private final OnClickListener mOnClickListener = new OnClickListener() {
   @Override
   public void onClick(final View view) {
      switch (view.getId()) {
      case R.id.capture_button:
         if (mCameraHandler.isOpened()) {
            if (checkPermissionWriteExternalStorage() && checkPermissionAudio()){
               if (!mCameraHandler.isRecording()) {
                   mCaptureButton.setColorFilter(0xffff0000); // turn red
                   mCameraHandler.startRecording();
               } else {
                   mCaptureButton.setColorFilter(0); // return to default color
                   mCameraHandler.stopRecording();
               }
            }
         }
         break;
      case R.id.brightness_button:
         showSettings(UVCCamera.PU_BRIGHTNESS);
         break;
      case R.id.contrast_button:
         showSettings(UVCCamera.PU_CONTRAST);
         break;
      case R.id.reset_button:
         resetSettings();
         break;
      }
   }
};
```

OnLongClickListener를 등록하여 원하는 기능을 추가할 수 도 있다. 다음은 카메라 프리뷰 화면에서 롱-클릭을 하면 스틸 이미지를 저장하는 기능을 실행하도록 리스너를 등록한 예제이다.

```
/**
```

```
 * capture still image when you long click on preview  image(not on buttons)
 */
private final OnLongClickListener mOnLongClickListener = new
OnLongClickListener() {
   @Override
   public boolean onLongClick(final View view) {
      switch (view.getId()) {
      case R.id.camera_view:
         if (mCameraHandler.isOpened()) {
            if (checkPermissionWriteExternalStorage()) {
               mCameraHandler.captureStill();
            }
            return true;
         }
      }
      return false;
   }
};
```

mOnDeviceConnectListener 리스너를 등록하면서 onConnect() 메소드륽 오버라이딩하여 Preview를 실행하도록 재정의하였으며, startPreview() 메소드에서는 mCameraHandler 인스턴스의 startPreview() 메소드에 렌더링할 mUVCCameraView SurfaceTexture를 인자로 넘겨준다.

```
private final OnDeviceConnectListener mOnDeviceConnectListener = new
OnDeviceConnectListener() {
…
   @Override
   public void onConnect(final UsbDevice device, final UsbControlBlock
ctrlBlock, final boolean createNew) {
      if (DEBUG) Log.v(TAG, "onConnect:");
         mCameraHandler.open(ctrlBlock);
         startPreview();
         updateItems();
   }
```

```
private void startPreview() {
   final SurfaceTexture st = mUVCCameraView.getSurfaceTexture();
   mCameraHandler.startPreview(new Surface(st));
   runOnUiThread(new Runnable() {
      @Override
      public void run() {
         mCaptureButton.setVisibility(View.VISIBLE);
      }
   });
   updateItems();
}
```

라이브러리 사용시 해상도 및 프리뷰 모드 등 기본 설정 값을 다음과 같이 초기화 하도록 할 수 있다.

```
/**
```

```
 * set true if you want to record movie using MediaSurfaceEncoder
 * (writing frame data into Surface camera from MediaCodec
 *  by almost same way as USBCameratest2)
 * set false if you want to record movie using MediaVideoEncoder
 */
 private static final boolean USE_SURFACE_ENCODER = false;

/**
 * preview resolution(width)
 * if your camera does not support specific resolution and mode,
 * {@link UVCCamera#setPreviewSize(int, int, int)} throw exception
 */
 private static final int PREVIEW_WIDTH = 640;

/**
 * preview resolution(height)
 * if your camera does not support specific resolution and mode,
 * {@link UVCCamera#setPreviewSize(int, int, int)} throw exception
 */
 private static final int PREVIEW_HEIGHT = 480;

/**
 * preview mode
 * if your camera does not support specific resolution and mode,
 * {@link UVCCamera#setPreviewSize(int, int, int)} throw exception
 * 0:YUYV, other:MJPEG
 */
 private static final int PREVIEW_MODE = 0;
```

usbDualCameraTest 프로젝트 및 기타 라이브러리 설명은 생략한다. 본 샘플 프로젝트의 베이스라인이 된 saki4510t github를 clone하여 다양한 예제를 구동시켜 보기를 바란다. 그리고 UVC카메라로의 이미지를 OpenCV로 처리하기 위한 예제가 필요하다면 다음의 github repository를 참조하길 바란다. (https://github.com/saki4510t/OpenCVwithUVC)

- **Camera Devices Compatibility**

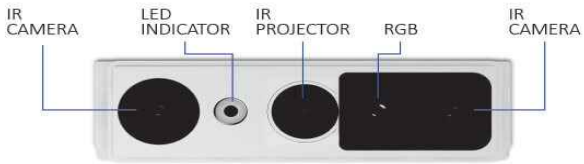
본 카메라 데모프로젝트에 사용된 카메라는 Logitech C170, Microsoft LifeCam Studio, FriendlyARM사의 FA-CAM202 2M-Pixel USB Camera 등의 USB2.0 디바이스 등이다.

혹시 UVC호환기종이라고 하더라도 USB3.1기반의 4K 고속 카메라 및 본 프로젝트 드라이버에 등록되지 않은 제조사(refer to USBVendorId class)의 제품은 본 데모프로젝트와 호환되지 않을 수 있으며, 추가 개발이 필요하다.

7.2 Astra Stereo-S Driver and Depth Stream Demo

Astra Stereo S 모듈은 Orbbec사의 최초의 Active Stereo IR 3D Camera Sensor이다.

Depth 검출 범위는 30cm ~ 3M 까지 동작하며, USB2.0, Linux UVC 호환, RGB 1920x1080@30fps, Depth 640x480@30fps, 전력소모는 Avg 2.36W 이다.



7.2.1 SDK Installation

센서 모듈의 기능을 안드로이드에서 곧바로 개발하기에 앞서 Windows 및 Linux환경에서 SDK을 사용하여 기능을 선 검증한 뒤, 안드로이드로 포팅을 하는 것을 권장한다. Android SDK가 제공되기는 하지만, Manual 및 예제가 아직 제공되는 것이 없어 개발이 어렵다.

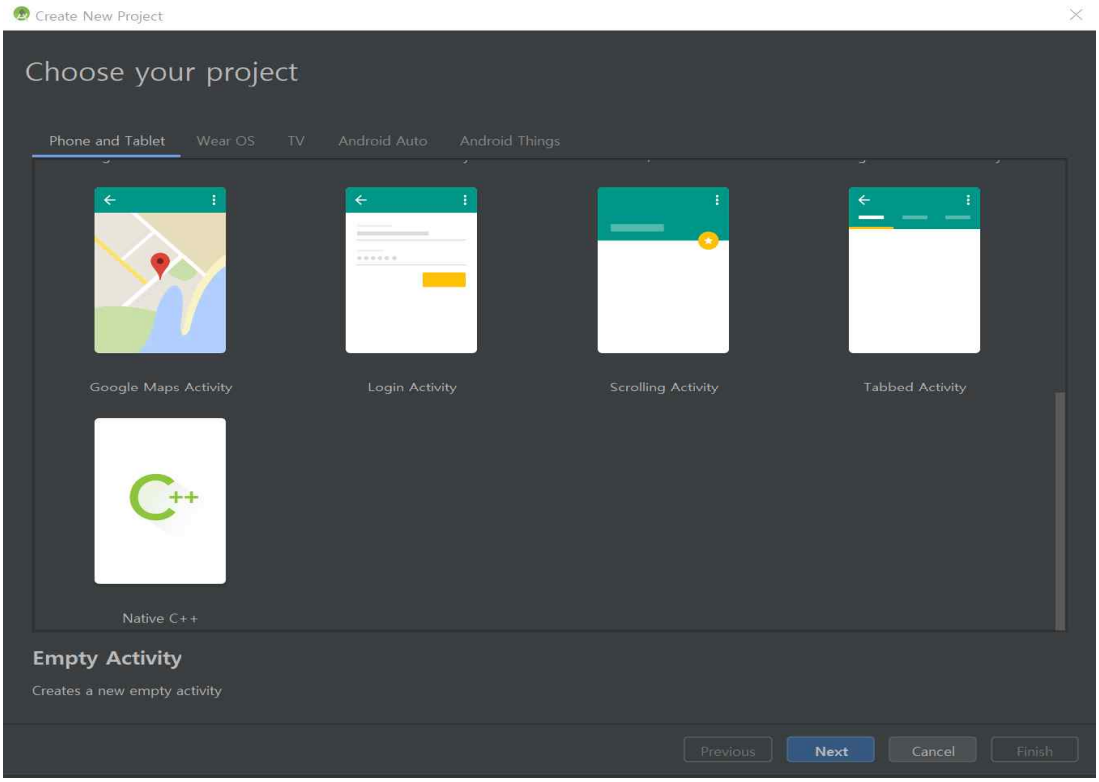
Orbbec의 Astra센서 시리즈 카메라들은 OpenNI(Open Natural Interaction) Protocol을 모두 지원한다. 따라서 이번에 제공하는 안드로이드 프로젝트는 OpenNI기반의 Orbbec SDK를 사용하여 3D센서 정보를 얻을 수 있다.

Astra 센서모듈을 사용하는 담당자는 별도로 제공해드리는 "OpenNI2 Coding Instruction" 문서를 숙지하기 바란다.

OpenNI2 SDK를 윈도우나 리눅스 등에서 검증할 필요가 있는데, 개발환경 셋업은 "Orbbec SDK Development Manaul.pdf"를 참조하여 선호하는 운영체제에서 SDK를 설치하여 기능 동작을 테스트 해보기 바란다. 본 문서에서는 Android환경설정을 다루도록 하겠다.

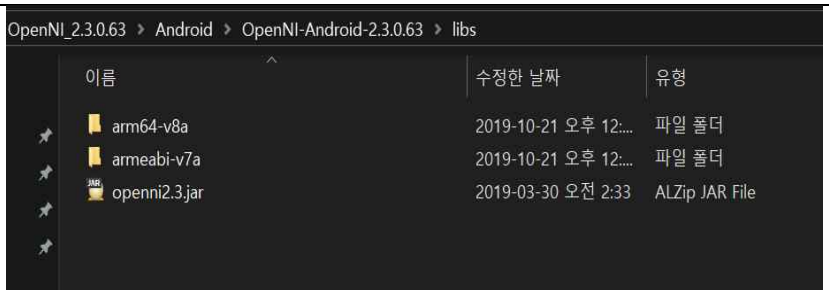
7.2.2 Android 환경 설정

- OpenNI2 SDK를 안드로이드 프로젝트에 포함시키기 위해서는 아래 그림의 프로젝트 생성시 좌측 하단의 Native C++을 선택한다.



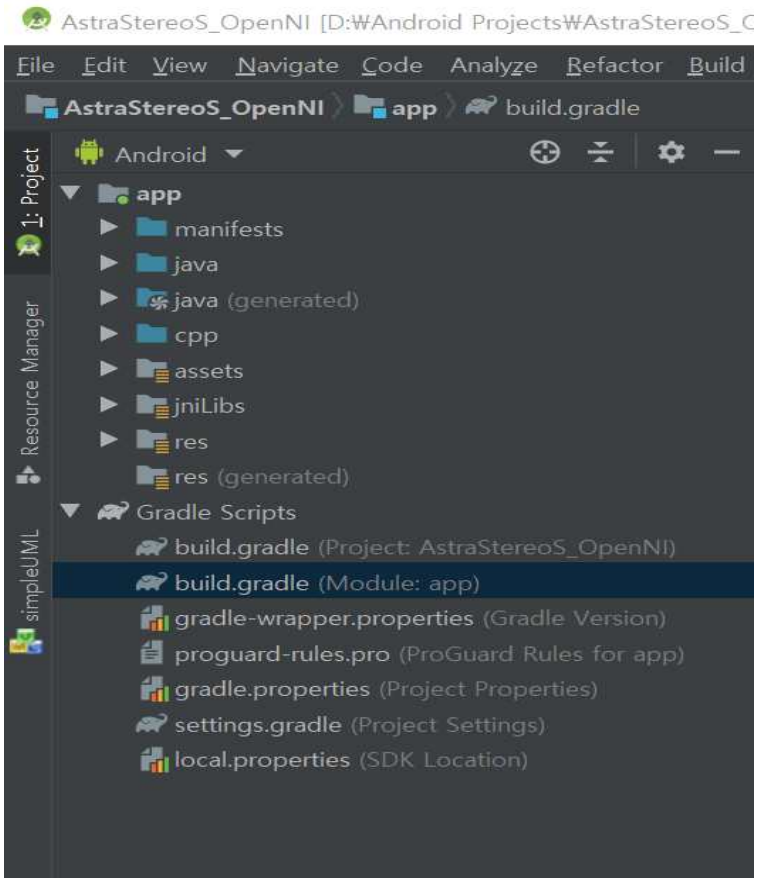
- OpenNI SDK의 libs 디렉토리내의 모든 파일을 안드로이드 프로젝트내 경로로 복사한다.

```
From:
\Astra Orbbec\2.0.17\OpenNI_2.3.0.63\Android\OpenNI-Android-2.3.0.63\libs
To:
\AstraStereoS_OpenNI\app\libs
```



- Gradle 및 CMakeLists 파일에서의 프로젝트 설정

`\AstraStereoS_OpenNI\app\build.gradle` 파일을 더블 클릭하여 오픈한다.



아래와 같이 ndk의 abiFilters를 defaultConfig에 포함시킨다.

```
android {
    compileSdkVersion 28
    buildToolsVersion "29.0.2"
    defaultConfig {
        applicationId "com.orbbec"
        minSdkVersion 19
        targetSdkVersion 28
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
        externalNativeBuild {
            cmake {
                cppFlags ""
            }
        }

        ndk {
            abiFilters 'armeabi-v7a'
        }
    }
```

buildTypes 이후에 sourceSets를 라이브러리파일 디렉토리를 설정하기위해 추가한다.

```
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
        }
    }
    sourceSets {
        main {
            jniLibs.srcDirs = ['libs']
        }
    }
```

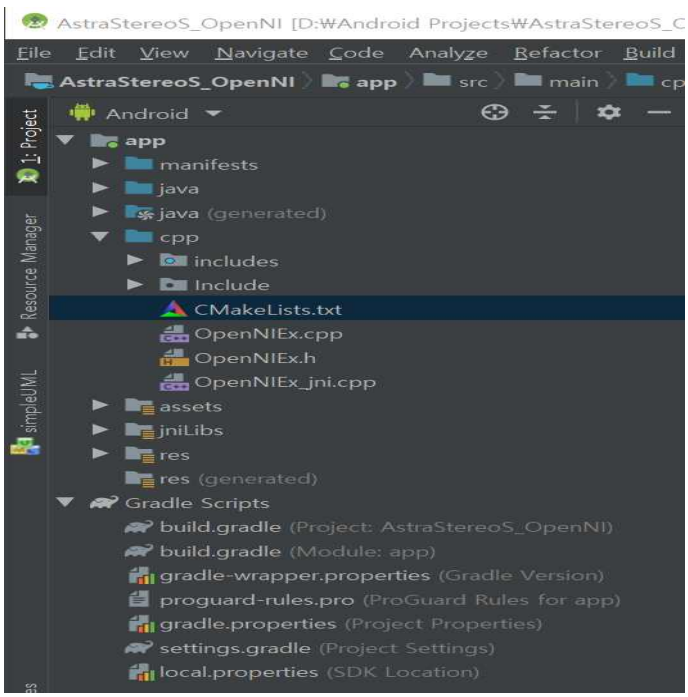
컴파일을 위해 적절한 openni2.3 jar 패키지를 dependencies에추가한다.

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'androidx.appcompat:appcompat:1.1.0'
    implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'androidx.test:runner:1.2.0'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.2.0'
    implementation files('libs/openni2.3.jar')
}
```

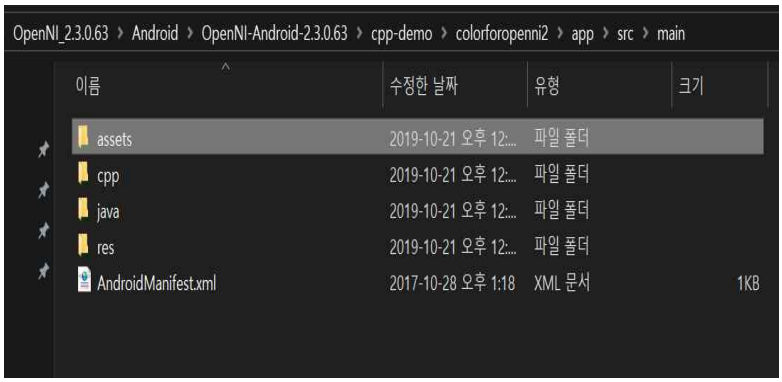
다음 그림과 같이 CMakeLists.txt 파일에서 cpp 관련설정을 한다.

```
# For more information about using CMake with Android Studio, read the
# documentation: https://d.android.com/studio/projects/add -native-code.html

# Sets the minimum version of CMake required to build the native library.

cmake_minimum_required(VERSION 3.4.1)

add_compile_options( -std=c++11)

# Creates and names a library, sets it as either STATIC
# or SHARED, and provides the relative paths to its  source code.
# You can define multiple libraries, and CMake builds them for you.
# Gradle automatically packages shared libraries with your APK.

add_library(OpenNI2
            SHARED
            IMPORTED
           )

set_target_properties(OpenNI2
        PROPERTIES IMPORTED_LOCATION
        ../../../../libs/armeabi-v7a/libOpenNI2.so)

add_library( # Sets the name of the library.
        #native-lib
        OpenNIEx
```

```
        # Sets the library as a shared library.
        SHARED

        # Provides a relative path to your source file(s).
        OpenNIEx.cpp
        OpenNIEx_jni.cpp
        )

# Searches for a specified prebuilt library and stores the path as a
# variable. Because CMake includes system libraries in the search path by
# default, you only need to specify the name of the public NDK library
# you want to add. CMake verifies that the library exists before
# completing its build.

find_library( # Sets the name of the path variable.
        log-lib

        # Specifies the name of the NDK library that
        # you want CMake to locate.
        log)

# Specifies libraries CMake should link to your target library. You
# can link multiple libraries, such as libraries you define in this
# build script, prebuilt third-party libraries, or system libraries.

target_link_libraries( # Specifies the target library.
        #native-lib
        OpenNIEx
        OpenNI2

        # Links the target library to the log library
        # included in the NDK.
        ${log-lib})
```



OpenNI-Android-2.3.0.63 패키지에서 아래와 같이 assets 디렉토리를 새로 생성한 프로젝트의 ₩app₩src₩main 디렉토리에 복사한다.



위 과정을 모두 완료하면, 새로운 프로젝트에서 color/depth/IR map정보를 읽고 표시할 수 있다.

프로그램 컴파일이 완료되고, 카메라를 플러그인 하고, APK프로그램을 오픈하면, USB사용허가 팝업이 나타날 것이다. 만약 팝업 윈도우가 나타나지 않고, USB 사용권한을 얻지 못하면, 데이터를 읽을 수 없다. 다음은 정상 동작하는 Depth Stream을 Display하는 실행화면이다.

프로그램 실행하고 USB Permission이 획득되면 OnDeviceOpened() 메소드가 호출되는데, MainActivity.java에 onDeviceOpened() 메소드를 오버라이딩하여 재정의하고, 쓰레드를

생성하여 실행한다. 다음은 해당 코드이다.

```
@Override
    public void onDeviceOpened(UsbDevice device) {
        int VID = device.getVendorId();
        int PID = device.getProductId();
        int rc = OpenNIEx.open(VID, PID);
        if(rc < 0){
            Toast.makeText(this, "OpenNI Open failed",
Toast.LENGTH_SHORT).show();
            return;
        }else {
            Log.d(TAG, "open device success "+ Integer.toHexString(VID) +"0x"
+Integer.toHexString(PID));
        }
        startThread();
    }
```

```
void startThread(){
        mInit_Ok = true;
        Log.v(TAG, "start thread");
        m_thread = new Thread(){

            @Override
            public void run() {

                while (!mExit) {
                    if(OpenNIEx.WaitAndUpdate() < 0){
                        continue;
                    }
                    m_glView.update(mWidth, mHeight);
                }
            }
        };

        m_thread.start();
    }
```

위 WaitAndUpdate() 메소드는 아래에 Native로 선언되어 있으며, JNI 라이브러리 코드는 cpp로 구현되어 있다.

```
/* @OpenNIEx.java
 * Native Code 선언 부분
 */

public class OpenNIEx {
    static {
        System.loadLibrary("OpenNIEx");
    }

    public  native static int init();
    public  native static int open(int w, int h);
    public  native static int WaitAndUpdate();
    public  native static int ConvertTORGBA(ByteBuffer dst, int w, int h);
```

```
    public  native static void closedevice();
}
```

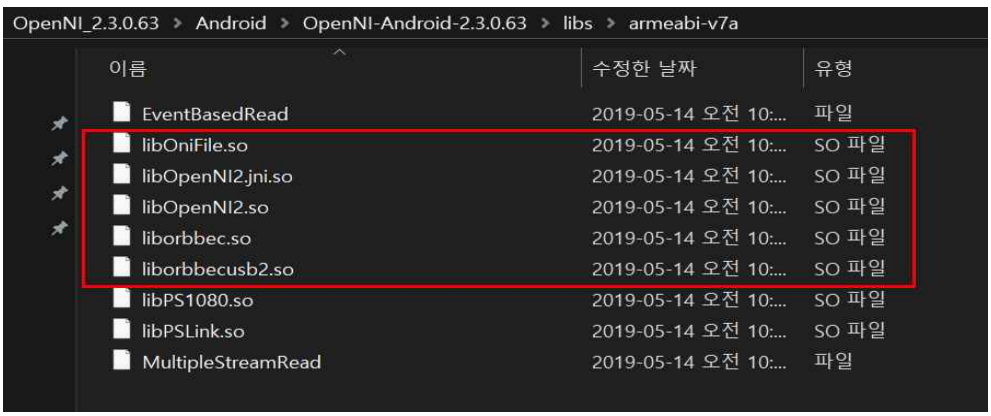
OpenNIex 클래스의 WaitAndUpdate() 코드의 실제 구현은 아래와 같이 cpp로 작성되었다. waitAndUpdate() 메소드부분을 발췌하였다.

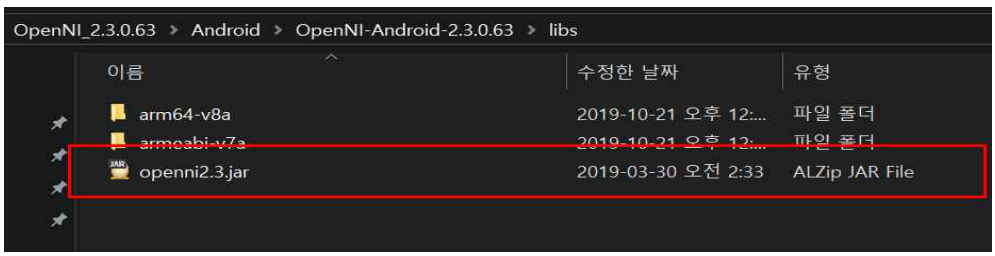
```
/* OpenNIEx 라이브러리 클래스 선언부
 * @ OpenNIEx.h
 */
using namespace openni;

class OpenNIEx {
public:
    OpenNIEx();
    ~OpenNIEx();
    int init();
    int enumerateDevices( int vid, int pid);
    int open(int vid, int pid);
    int waitAndUpdate();
    void close();
    int ConventToRGBA(uint8_t * dst, int w, int h);

private:

    void calcDepthHist(VideoFrameRef& frame);
    VideoStream mDepth;
    VideoFrameRef mFrame;
    Device mDevice;
    uint32_t* m_histogram;
    int mWidth;
    int mHeight;
};

/* OpenNIEx 라이브러리 구현 코드
 * @ OpenNIEx.cpp
 */
int OpenNIEx::waitAndUpdate(){
    VideoStream* pStream = &mDepth;
    int changedStreamDummy;
    Status rc = OpenNI::waitForAnyStream(&pStream, 1, &changedStreamDummy,
READ_WAIT_TIMEOUT);
    if (rc != STATUS_OK)
    {
        LOGE("Wait failed! (timeout is %d ms)\n%s\n", READ_WAIT_TIMEOUT,
OpenNI::getExtendedError());
        return -1;
    }
    rc = mDepth.readFrame(&mFrame);
    if (rc != STATUS_OK)
    {
        LOGD("Read failed!\n%s\n", OpenNI::getExtendedError());
        return 0;
    }
    if (mFrame.getVideoMode().getPixelFormat() != PIXEL_FORMAT_DEPTH_1_MM &&
mFrame.getVideoMode().getPixelFormat() != PIXEL_FORMAT_DEPTH_100_UM)
```

```
    {
        LOGD("Unexpected frame format\n");
        return 0;
    }

    // Depth값을 출력하도록 print code를 추가함
    uint16_t* pDepth = (uint16_t*)mFrame.getData();
    int middleIndex = (mFrame.getHeight()+1)*mFrame.getWidth()/2;
    LOGD("Depth: %8d\n", pDepth[middleIndex]);

    return 0;
}
```

- Access SDK

Orbbec-SDK에 의해 제공되는 Library Files들은 크게 두가지 파트로 5개의 ".so dynamic" 파일과 1개 ".jar" 파일이다.

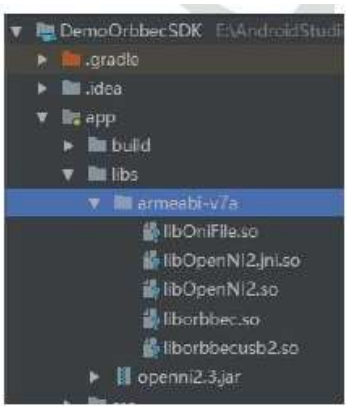


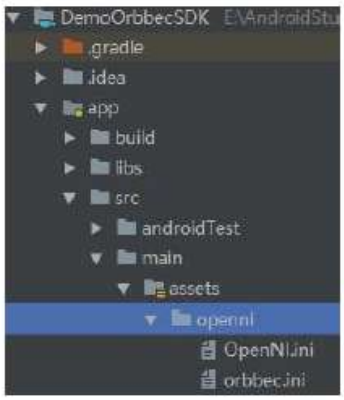


그리고 ".ini" 설정파일 두개(OpenNI.ini, orbbec.ini)가 제공된다.

위 so 라이브러리 파일들은 프로젝트 모듈내 libs 디렉토리 아래에 위치해야 하고, ini 파일은 app₩src₩main₩asset₩openni 디렉토리 아래에 위치해야한다. 과 ini 설정파일들은 프로젝트 모듈내의 libs 디렉토리에 위치하도록 만들어야 한다.

그리고 build.gradle 파일에서 jniLibs 디렉토리를 재설정한 뒤, 프로젝트 전체를 동기화(sync) 시켜야 한다.





- SDK Initialization

Orbbec-SDK API를 호출하기 전 초기화가 꼭 필요하며, 프로그램이 시작할 때 주로 수행된다. 다음과 같이 MainActivity 코드에 onCreate() 메소드에서 직접 호출해도 되고, onDeviceOpened() 메소드를 오버라이딩하여 재정의하여 사용해도 된다.

다음은 onCreate() 메소드에서 직접 호출한 예제이다.

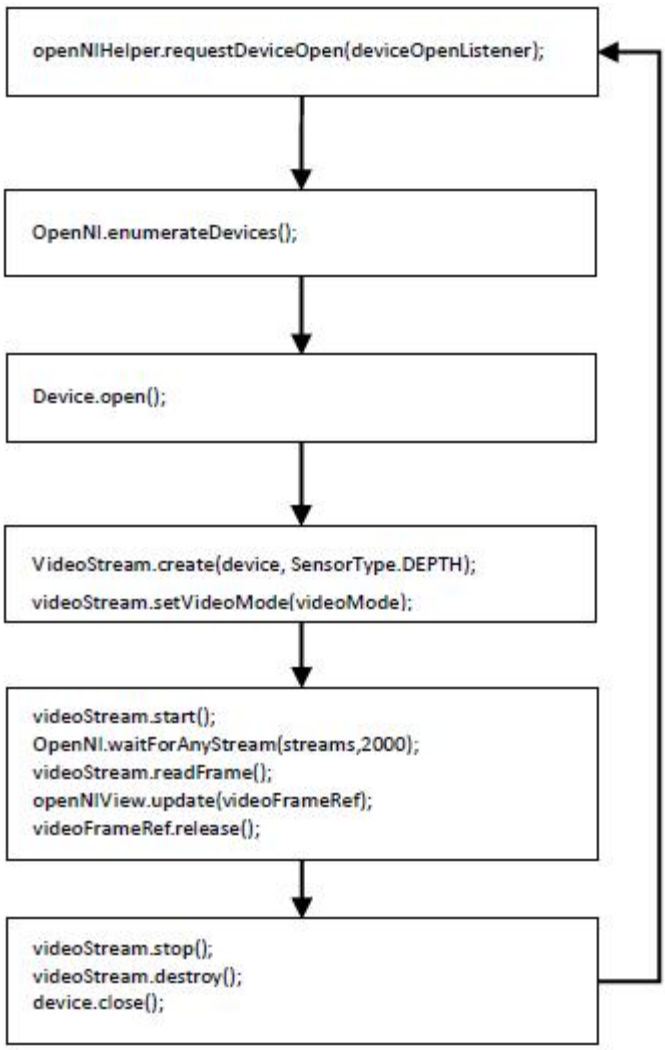
```
import android.app.Application;
import org.openni.OpenNI;

public class MyApplication extends Application {

   @Override
   public void onCreate() {
      super.onCreate();
      initializeSDK();
   }

   private void initializeSDK() {
      OpenNI.setLogAndroidOutput(true);
      OpenNI.setLogMinSeverity(0);
      OpenNI.initialize();
   }
```

Initializing call interface:

Interface	API	Parameter	Description
OpenNI	OpenNI .setLogAndroidOutput(true);	ture: output Android Log false： No output	Setting if SDK Log output or not.
	OpenNI .setLogMinSeverity(0);	0 - Verbose; 1 - Info; 2 - Warning; 3 – Error;	Setting Log output level.
	OpenNI .Initialize();		Initializing SDK(Must be called).

- Interface Collaborative Calling Rules

다음은 순차적으로 API를 호출하는 절차를 언급하였으며, Active Acquirement Mode를 설명한 순서도 중 하나이다. Device가 Closed된 후에, Device를 다시 Require 및 Open하는 순서이다.



우리가 제공하는 "AstraSteresoS_OpenNI 샘플 프로젝트도 위 내용을 반영하였다. 이 밖에도 다양한 방법과 Active Acquirement Mode와 더불어 Callback Mode로 사용이 가능하다. 자세한 내용은 Orbbec SDK의 **"Orbbec-OpenNI2.0 SDK Android Interface.pdf**" 문서의 13장을 숙지하기 바란다.

- Return & Release SDK

안드로이드의 앱 종료시나 Back버튼을 누르고 앱을 백그라운드로 보내면 모든 SDK 리소스를 해제하도록 한다. 이는 initializeSDK()에 상응하는 동작이며, OpenNI.shutdown()을 호출하기전에 모든 VideoStream과 Device Resource는 해제되어야한다. 리소스 해제과정은 "Orbbec-OpenNI2.0 SDK Android Interface.pdf" 문서의 Step8-11을 참조한다.

SDK를 다시 사용하기 위해서는 SDK interfaceans서의 Step3인 initializeSDK()부터 다시 시작한다.

Interface	API	Parameter	Description
OpenNI	shutdown();	N/A	Release Orbbec-SDK resource

```
@Override
Protected void onDestroy() {
    OpenNI.shutdown();
    super.onDestory();
}
```

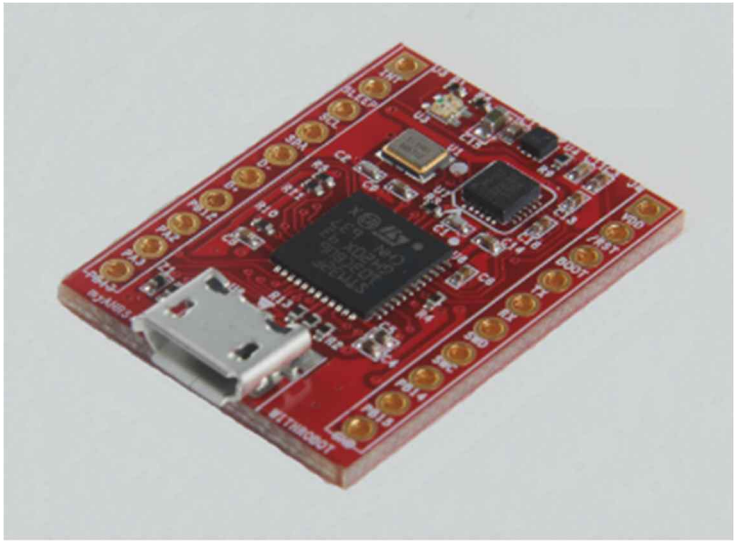
위와 같이 onDestroy() 메소드를 재정의하여 종료하도록 한다.

본 문서는 최소한의 내용을 소개해 드리며, 담당 개발자는 Orbbec SDK문서를 필독하여 숙지해두기를 바란다.

7.3 AHRS IMU Driver Library and Demo Application

9 DOF-IMU (Inertial Measurement Unit)는 자이로스코프/가속도계/지자기센서로 구성된 센서 모듈을 뜻하며, LVRK-3399와 함께 제공된 센서모듈은 WITHROBOT사의 myAHRS+ 제품으로 안정적인 가속도 및 방위각을 제공하며, UART/USB/I2C 통신 인터페이스를 지원하여 다양한 시스템과 연결이 가능하다.

LVRK-3399 보드와는 기본적으로 USB 인터페이스를 통하여 패킷을 전달받는다.



해당 모듈은 3축 자이로스코프, 3축 가속도계, 3축 지자기 센서를 융합하여 얻은 3차원 공간상의 방위각(Heading)과 자세(Attitude) 정보를 출력한다. 출력데이터로 Euler angle 및 Quaternion, 출력 속도는 최대 100 Hz이다.

모듈의 데이터 메시지는 문자열 형식 메시지이고, 다음과 같은 형태로 출력된다.

RIIMU	형식	$RIIMU, sequence number, accel_x, accel_y, accel_z, gyro_x, gyro_y, gyro_z, magnet_x, magnet_y, magnet_z, temperature
	예	$RIIMU,27,25,26,-1901,-16,-7,-11,259,-147,55,158
IMU	형식	$IMU, sequence number, accel_x, accel_y, accel_z, gyro_x, gyro_y, gyro_z, magnet_x, magnet_y, magnet_z, temperature
	예	$IMU,60,0.0297,0.0019,-1.0056,0.0153,-0.0282,0.3487,129.5813,-110.5982,142.4527,35.5
RPY	형식	$RPY, sequence number, roll, pitch, yaw
	예	$RPY,68,0.04,1.56,34.22
QUAT	형식	$QUAT, sequence number, x, y, z, w
	예	$QUAT,55,-0.0037,0.0134,0.2932,0.9560
RPYIMU	형식	$RPYIMU, sequence number, roll, pitch, yaw, accel_x, accel_y, accel_z, gyro_x, gyro_y, gyro_z, magnet_x, magnet_y, magnet_z, temperature
	예	$RPYIMU,82,0.04,1.67,34.07,0.0307,0.0014,-1.0095,-0.0435,0.0919,0.1660,137.2258,-90.1564,134.8918,35.6
QUATIMU	형식	$QUATIMU, sequence number, x, y, z, w, accel_x, accel_y, accel_z, gyro_x, gyro_y, gyro_z, magnet_x, magnet_y, magnet_z, temperature
	예	$QUATIMU,02,-0.0039,0.0135,0.2940,0.9557,0.0238,0.0034,-0.9978,-0.0448,-0.0896,0.2866,136.5006,-86.5058,134.0961,35.8

위 String형식의 출력 포맷 중에서 "RPY" 형식이 기본으로 동작된다.

런타임에서 동적으로 각 출력 포맷을 변경할 수 있다. 하지만 LVRK-3399와 같이 제공한 안드로이드 샘플 예제는 RPY만 지원하고 있으며, 디바이스로 커맨드를 전송하여 모드를 변경할 수 있다.

보통은 RPY 자세 값을 사용하여 렌더링이 가능하지만, 어플리케이션 담당자가 다른 포맷이 필요한 경우 설정 변경을 위한 추가 API를 제공하거나, 모듈의 펌웨어 디폴트 출력을 원하는 출력으로 고정하도록 제조사에 요청한다.

7.3.1 안드로이드 드라이버 및 데모 프로젝트

LVRK-3399와 함께 제공하는 IMU Demo 프로젝트에 사용한 센서 모듈은 USB인터페이스를 사용하여 회로가 연결되어 있으며, 필요 시 UART로 연결도 가능하다.

프로젝트는 드라이버 라이브러리와 해당 라이브러리를 사용한 테스트 앱을 포함한다.

드라이버 라이브러리는 IMU 센서 모듈과 LVRK-3399의 Android USB Host Mode(OTG) 기능을 통하여 데이터 통신이 가능하도록 해준다.

드라이버는 모두 JAVA로 구현되어 있으며, imuDriverForAndroid 라이브러리의 SerialInputOutputManager 클래스의 runnable인터페이스로 구현된 run() 메소드에서 오버라이딩하여 지속적으로 포트를 read(), write() 메소드로 시리얼포트를 접근하여 데이터를 주고 받는다.

```
@Override
    public void run() {
        synchronized (this) {
            if (getState() != State.STOPPED) {
                throw new IllegalStateException("Already running.");
            }
            mState = State.RUNNING;
        }

        Log.i(TAG, "Running ..");
        try {
            while (true) {
                if (getState() != State.RUNNING) {
                    Log.i(TAG, "Stopping mState=" + getState());
                    break;
                }
                step();
            }
        } catch (Exception e) {
            Log.w(TAG, "Run ending due to exception: " + e.getMessage(), e);
            final Listener listener = getListener();
```

```
            if (listener != null) {
              listener.onRunError(e);
            }
        } finally {
            synchronized (this) {
                mState = State.STOPPED;
                Log.i(TAG, "Stopped.");
            }
        }
    }

    private void step() throws IOException {
        // Handle incoming data.
        int len = mDriver.read(mReadBuffer.array(), READ_WAIT_MILLIS);
        if (len > 0) {
            if (DEBUG) Log.d(TAG, "Read data len=" + len);
            final Listener listener = getListener();
            if (listener != null) {
                final byte[] data = new byte[len];
                mReadBuffer.get(data, 0, len);
                listener.onNewData(data);
            }
            mReadBuffer.clear();
        }

        // Handle outgoing data.
        byte[] outBuff = null;
        synchronized (mWriteBuffer) {
            len = mWriteBuffer.position();
            if (len > 0) {
                outBuff = new byte[len];
                mWriteBuffer.rewind();
                mWriteBuffer.get(outBuff, 0, len);
                mWriteBuffer.clear();
            }
        }
        if (outBuff != null) {
            if (DEBUG) {
                Log.d(TAG, "Writing data len=" + len);
            }
            mDriver.write(outBuff, READ_WAIT_MILLIS);
        }
    }
```

USB포트를 통해 읽어온 패킷은 다음의 SerialConsoleActivity.java 클래스의 updateReceiveData() 메소드에서 String형식의 메시지를 UI에 업데이트 하도록 구성한 간단한 데모이다. 다음은 수신한 패킷 메시지를 UI 업데이트하는 코드이다.

```
// @SerialconsoleActivity.java
private void updateReceivedData(byte[] data) {
        final String message = "Read " + data.length + " bytes:  \n"
                + HexDump.dumpHexString(data) + " \n\n";
        mDumpTextView.append(message);
        mScrollView.smoothScrollTo(0, mDumpTextView.getBottom());
```

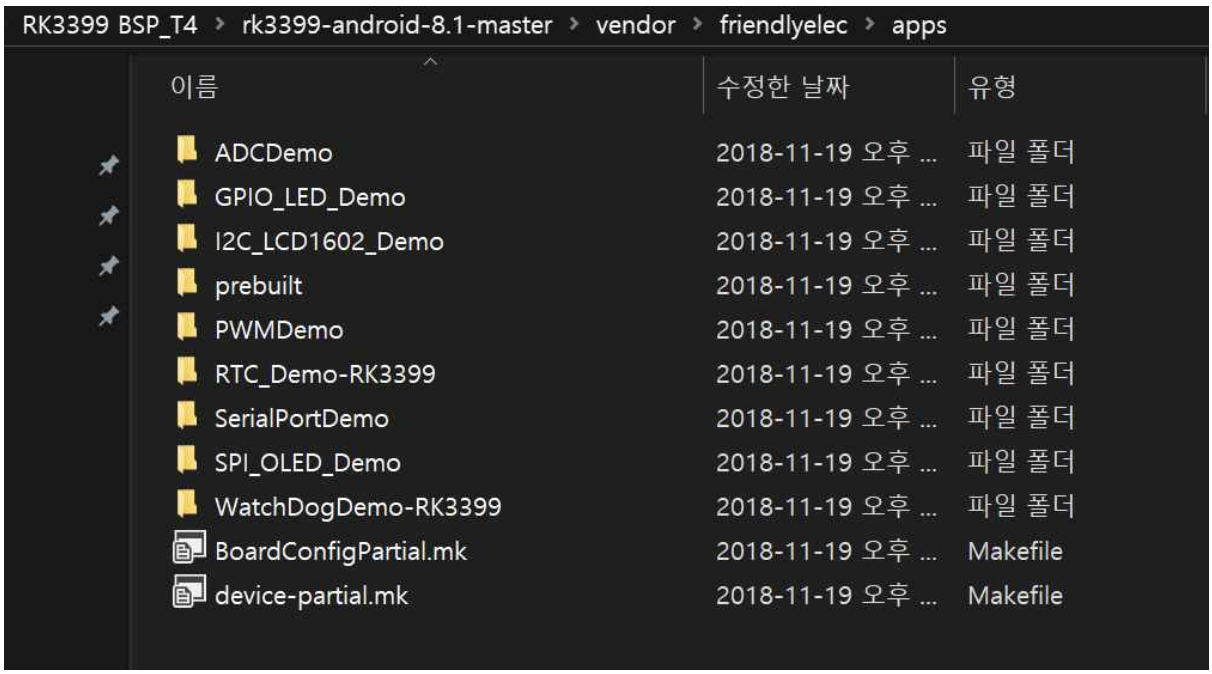
```
}
```

위 String타입의 message값을 디코딩하여 Roll, Pitch, Yaw값을 앱에서 렌더링 등을 위한 필요한 곳에서 사용하면 된다.

만약 앱에서 Blocking 방식의 RPY값 업데이트가 필요하다면, API를 추가 작성하여 제공할 수 있다. 별도의 요청사항이 있지 않는 한 RPY 포즈 스트림 출력 상태로 릴리즈 한다.

8. Factory Reset : Pre-installed Apps

현재 LVRK-3399 패키지를 압축을 풀고, 빌드를 하면 Google앱 이외에 같이 설치된 기본 앱들이 있다.



Rk3399-android-8.1-master₩vendor₩friendlyelec₩apps 디렉토리를 살펴보면 8가지 HW Test Demo 들이 포함되어 있으며, 이들 앱은 Factory Reset을 실행하더라도 초기화이후 다시 설치되는 기본 앱이 된다. 이 디렉토리에 추가 개발한 앱의 소스를 포함시키고, device-partial.mk 파일에 새로 추가한 앱의 디렉토리 이름을 추가한다.

```
# FriendlyThings examples
# http://wiki.friendlyarm.com/wiki/index.php/FriendlyThings_for_RK3399
PRODUCT_PACKAGES += \
        libfriendlyarm-things \
        SerialPortDemo \
        WatchDogDemo \
```

```
        RTCDemo \
        GPIO_LED_Demo

# PRODUCT_PACKAGES += LCD1602
# PRODUCT_PACKAGES += ADCDemo
# PRODUCT_PACKAGES += PWMDemo
# PRODUCT_PACKAGES += SPI-OLED
```

다음의 링크를 방문하여 [4.2 App System Right ~ 4.3 Compile Your App] 관련내용 참조하여 Custom App을 Pre-installed App에 포함시키는 절차를 숙지하도록 한다. AndroidManifest.xml과 Android.mk 파일을 수정하도록 한다.

http://wiki.friendlyarm.com/wiki/index.php/FriendlyThings_for_RK3399

- Modify AndroidManifest.xml

```
android:sharedUserId="android.uid.system"
```

- Modify Android.mk

```
LOCAL_PATH:= $(call my-dir)
include $(CLEAR_VARS)

LOCAL_SRC_FILES := $(call all-subdir-java-files)

LOCAL_PACKAGE_NAME := Project Name

LOCAL_CERTIFICATE := platform
LOCAL_MODULE_TAGS := optional
LOCAL_CFLAGS := -lfriendlyarm-hardware

include $(BUILD_PACKAGE)
```

- Compile Your APP Together with Android Source code

```
cd rk3399-android-8.1
. setenv.sh
cd vendor/friendlyelec/apps/Your_APP_Name
mm
```

9. Media Process Platform (MPP)

Media Process Platform(MPP)은 RK3399 등의 Rockchip 플랫폼이 사용하는 비디오 코덱 파서와 하드웨어 추상 레이어 모듈로서 Rockchip사 Cross Platform Media Process를 위한 미들웨어 라이브러리이다. 이는 비디오 및 이미지 처리에서 고성능, 유연성, 확장성을 보장하기 위한 것이다. vcodec_service/vpu_service/mpp_service 등의 하드웨어 커널드라

이버가 제공되며, 안드로이드 및 리눅스를 지원한다. 본 문서는 하드웨어 디코더 사용을 위한 데모 프로젝트의 자세한 내용은 제외한다.

- 소스코드:
  git clone -b release https://github.com/rockchip-linux/mpp.git
- Android 빌드를 위해서는 NDK 패키지가 필요하며, "android-ndk-r10d" 버전을 사용한다. ndk소스는 google website에서 검색하여 old release에서 다운로드 받는다.
- MPP관련 자세한 내용은 http://opensource.rock-chips.com/wiki_Mpp 사이트를 자세히 참조하도록 한다.
- 4K Video Player 프로그램은 10장의 "RK4K Video Player APK"를 설치해서 테스트 해볼 수 있다.

10. 드라이버 라이브러리 프로젝트 코드 저장소

- Release Note
  https://github.com/jackcha76/DocToElvision
- UVC Camera Driver Library
  https://github.com/jackcha76/UVCCamera-master_release-v0.1
- IMU Monitor Driver Library
  https://github.com/jackcha76/IMUMonitor-master_release-v0.1
- Astra Stereo S Depth Stream Driver Library
  https://github.com/jackcha76/AstraSteresoS_OpenNI
- Rockchip MPP
  https://github.com/rockchip-linux/gstreamer-rockchip
- RK4K Video Player APK
  https://gitlab.com/friendlyelec/rk3399-android-8.1/tree/master/vendor/rockchip/common/apps