

SenXorViewer Android App Note

|  |  |
| --- | --- |
| Version: | 1.2 |
| Date: | 26th Nov 2019 |
| Author: | Jackson LAU |

|  |  |  |
| --- | --- | --- |
| **Revision** | **Date** | **Comment** |
| 1.1 | 25 Feb 2019 | Initial release |
| 1.2 | 27 Nov 2019 | Publish version |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Table of Contents

[Introduction 3](#_Toc25745963)

[Project Setup 3](#_Toc25745964)

[Debugging Over WiFi 4](#_Toc25745965)

[Project Structure 5](#_Toc25745966)

[Color Palette 5](#_Toc25745967)

[libuvccamera & usbCameraCommon 6](#_Toc25745968)

[thermalViewer 7](#_Toc25745969)

[Application 8](#_Toc25745970)

[USB Connect Dialog 8](#_Toc25745971)

[Video Streaming 8](#_Toc25745972)

[UI Components 9](#_Toc25745973)

[Settings 9](#_Toc25745974)

# Introduction

The SenXorViewer APP Note is the release version of the android software from Meridian Innovation Limited. It is including the project structure and settings in order to facilitate users to understand the features, or developers to customize their own application.

# Project Setup

1. Download the Android NDK r14b. Make sure the version is **r14b or older**. It can be found on the following website:

<https://developer.android.com/ndk/downloads/older_releases.html>

1. Extract the downloaded archive file onto some location on your filesystem.
2. Launch Android Studio and click File -> New -> Import Project for SenXorViewer.
3. Modify the file “local.properties”. Determine the path of Android NDK r14b and add the following line:

ndk.dir = *path*

For example, if the path was C:\Users\Lenovo\Documents\android-ndk-r14b, you would type:

ndk.dir = C:\\Users\\Lenovo\\Documents\\android-ndk-r14b

1. Try to build and run the application module. Suggest not using Available Virtual Devices (AVD) because XCAM will not being detected in AVD.
2. Make sure to check the compileSdkVersion, buildToolsVersion, minSdkVersion etc., Gradle and Android Plugin versions match on every gradle file.

compileSdkVersion = 27

buildToolsVersion = “26.0.2”

minSdkVersion = 18

targetSdkVersion = 27

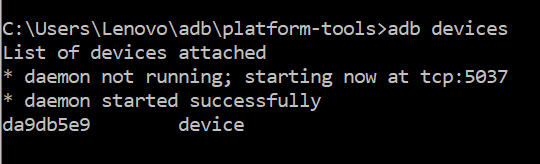
Gradle version = 4.1

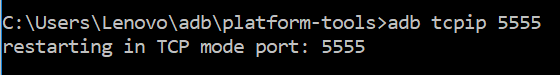
Android Plugin version = 3.0.1

1. To install the application, download the APK file to mobile or get from Google Play Store. Then, grant the permission for installation.
2. Some devices may need to enable OTG when connecting the camera to the application.

# Debugging Over WiFi

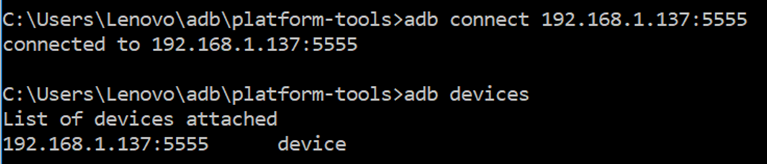
Ref: <https://futurestud.io/tutorials/how-to-debug-your-android-app-over-wifi-without-root>

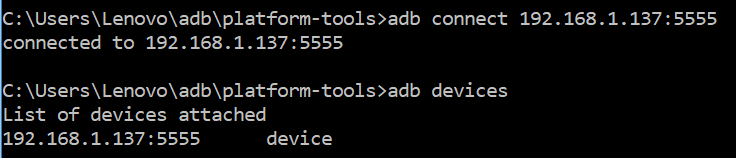
1. ****Connect your device via USB and make sure USB debugging is working. It needs activate Developer options. Check this by seeing if your device is listed after running “adb devices”. On Windows, Android Studio Terminal is an easiest way to do so.
2. Run “adb tcpip 5555”.

****

1. Determine the WIFI IP address assigned to the device, which would be located in Settings -> About phone -> status.

Run “adb connect *IP address of device*:5555”.

****

1. ****The device is now connected. Run “adb devices” to check.

# Project Structure

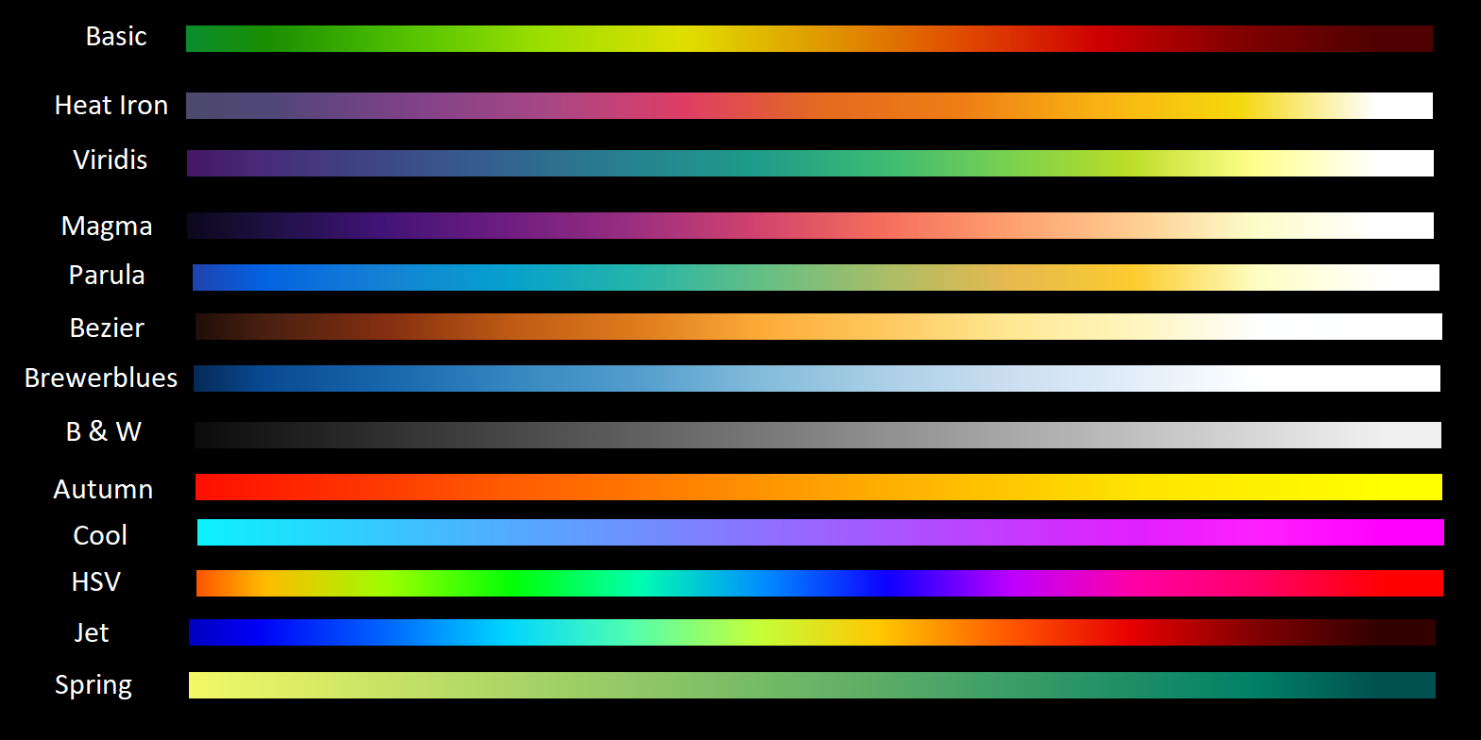
The SenXorViewer application contains 2 library modules: **libuvccamera** and **usbCameraCommon**.

1 Application module: **thermalViewer**

## Color Palette

SenXorViewer is mainly focusing on providing suitable color palette selection in which are suitable for most of the use cases. On application level the color seed is controlled by paletteNum which is one of the preference items, and then pass to the backend mCameraHandler for mapping.

Default SenXorViewer is supporting 13 palettes, interpolated to form color mapping. Details please check thermalViewer/src/main/res/values/colors.xml.



## libuvccamera & usbCameraCommon

These are the customized libraries for accessing UVC camera on non-rooted Android device. libuvccamera is used for controlling the USB event, camera dialog and color mapping on Java Native Interface (JNI) layer; usbCameraCommon is used for video streaming with UVC camera handler and media encoders. It is built upon the UVCCamera open source library. The temperature data array is decoded and manipulate in here, and then update the result through Frame callback function. Details please check UVCCameraHandler::mCameraHandler.

AbstractUVCCameraHandler class handles camera handler, which should be controlled from mCameraHandler in MainActivity. The temperature array is updated from mTemperatureCallback. It contains some customized functions for XCAM:

* public float getTemperature(int posX, int posY)
* public void setCaptureElement(boolean isFlip)
* public void setThermalEnable(boolean enable)
* public void setCmosEnable(boolean enable)
* public void setMeridianParams(int numPalette, int xShift, int yShift){
* public void setIsAutoScale(boolean input)
* public void setPaletteRange(float min, float max)

<https://github.com/saki4510t/UVCCamera>

## thermalViewer

ThermalViewer is the application module to construct the main activity and provide skeleton codes. Also, there are some utilities provided for thermal image preview, including temperature unit conversion, thermal data upsampling, strokedTextView and CrashReport. It is the sample code on how to integrate the library modules for reference.

Temperature unit conversion and helper class are used for calculating the value between Celsius, Fahrenheit and Kelvin.

Thermal data upsampling is used to generate a bigger size of the thermal data that can make equivalent size of thermal image to overlay on the CMOS layer. The thermal data array size from XCAM is 32\*32, which is much smaller to the CMOS image sensor output.

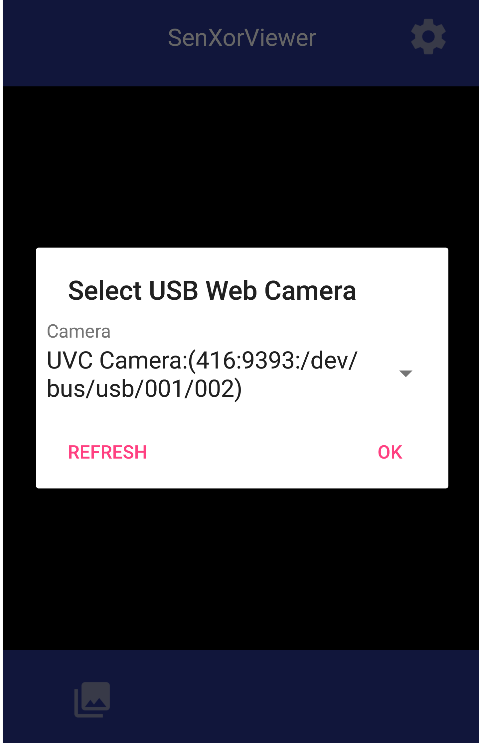
mCameraHandler is extend class of abstract class AbstractUVCCameraHandler. It handles the basic operation of the XCAM and color camera.

* public void startPreview(final Object surface)
* public void captureStill()
* public interface CameraCallback{}
* All functions from AbstractUVCCameraHandler class

FrameCallBack is the handler to receive callback frames. onFrame(ByteBuffer, int[]) will be called when the frame is updated from mCameraHandler. ByteBuffer is the pixel buffer which can be converted into bitmap by copyPixelsFromBuffer(frame). int[] thermal\_data is the thermal value buffer in size of 32\*32 = 1024. TemperatureBroadcastReceiver is the BroadcastReceiver that can also get frame data. However in order to maintain data synchronization, we mainly use the data updated from FrameCallBack. The main difference of these two functions is FrameCallBack is used for handling the full frame such as creating Bitmap, apply face detection etc.; TemperatureBroadcastReceiver is used for handling particular point temperature such as cross hair target and UI stats. User can do temperature checking on the thermal data array or image processing on bitmap buffer depends on the use case.

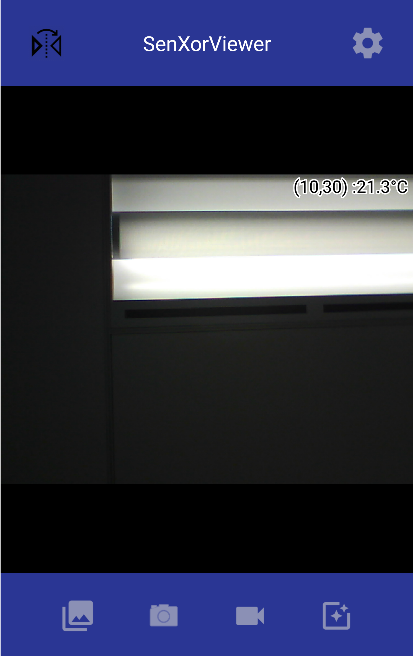
# Application

## USB Connect Dialog

After the XCAM is attached, the USB connect dialog should be pop out and the XCAM UVC Camera device is listed. Press “OK”.

## Video Streaming

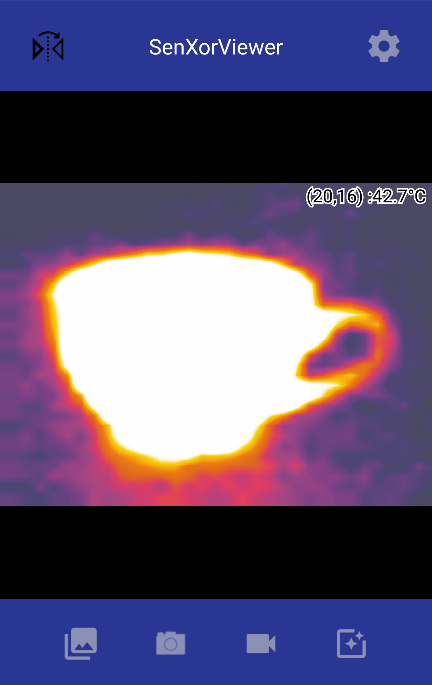
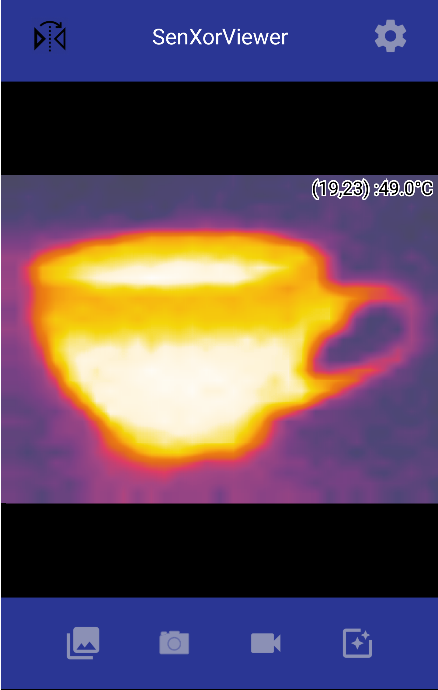
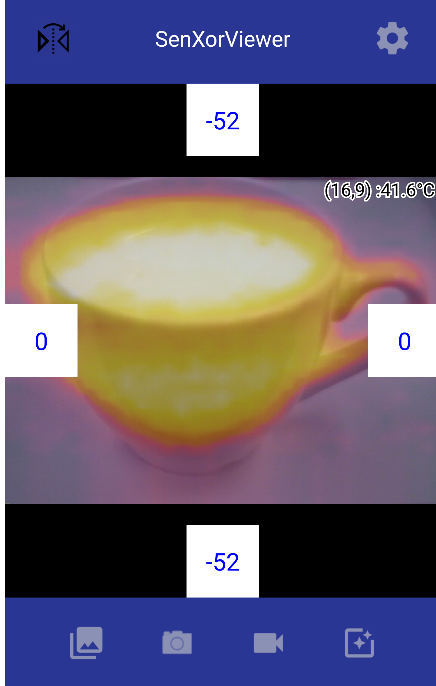
Overlay Mode

CIS Image Mode

Thermal Image Mode

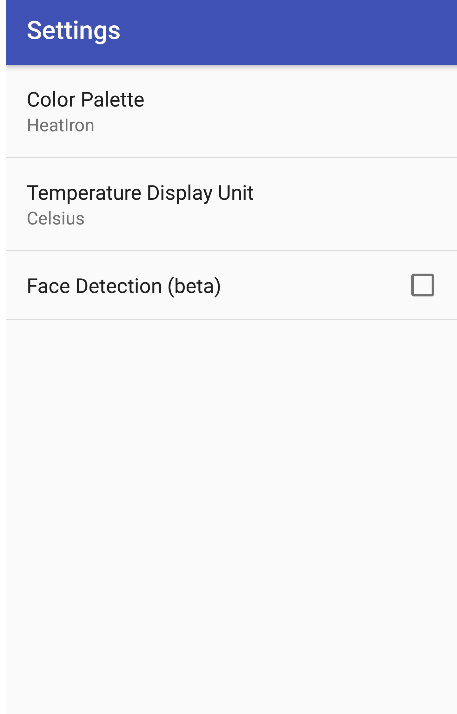
There are 3 preview modes by using XCAM. Press  to switch the mode.

## UI Components

|  |  |
| --- | --- |
| Items | Description |
|  | X-axis / Horizontal mirroring the preview window |
|  | Settings |
|  | Open Gallery |
|  | Capture preview window |
|  | Record video of preview window |
|  | Preview mode |
| SenXorViewer | App title. Under overlay mode, overlayer shifting buttons will pop out to adjust thermal image overlay manually. |

## Settings

****

Select Color Palette for thermal camera

Select Celsius or Fahrenheit