

SenXorViewer Android App Note

|  |  |
| --- | --- |
| Version: | 1.2 |
| Date: | 19th Nov 2019 |
| Author: | Jackson LAU |

|  |  |  |
| --- | --- | --- |
| **Revision** | **Date** | **Comment** |
| 1.1 | 25 Feb 2019 | Initial release |
| 1.2 | 19 Nov 2019 |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Table of Contents

[Introduction 3](#_Toc25054406)

[Project Setup 3](#_Toc25054407)

[Debugging Over WiFi 4](#_Toc25054408)

[Project Structure 5](#_Toc25054409)

[Colorbar 5](#_Toc25054410)

[OpenCV 6](#_Toc25054411)

[FaceDetection 6](#_Toc25054412)

[libuvccamera & usbCameraCommon 7](#_Toc25054413)

[thermalViewer 7](#_Toc25054414)

[Application 8](#_Toc25054415)

[USB Connect Dialog 8](#_Toc25054416)

[UI Components 9](#_Toc25054417)

[Settings 10](#_Toc25054418)

# 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

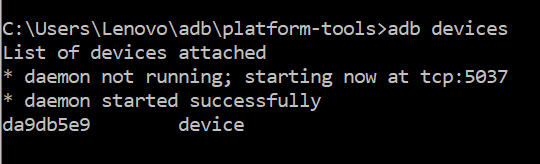
Gradle version = 4.1

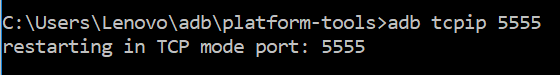
Android Plugin version = 3.0.1

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

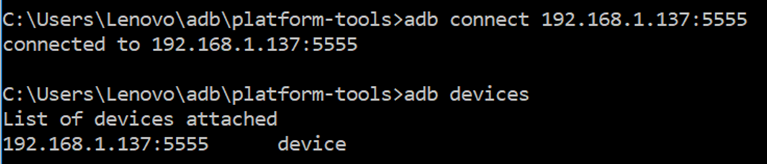
Ref: <https://futurestud.io/tutorials/how-to-debug-your-android-app-over-wifi-without-root>

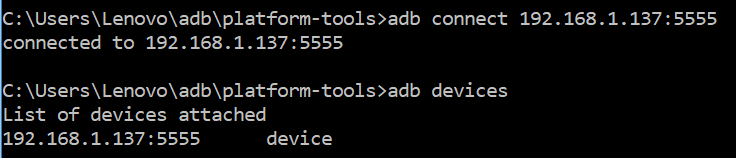
1. ****Connect your device via USB and make sure USB debugging is working. It needs activate Developer options. Check this by seeing if your device is listed after running “adb devices”. On Windows, Android Studio Terminal is an easiest way to do so.
2. Run “adb tcpip 5555”.

****

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 5 library modules: **colorbar**, **facedetection**, **libuvccamera**, **opencv** and **usbCameraCommon**.

1 Application module: **thermalViewer**

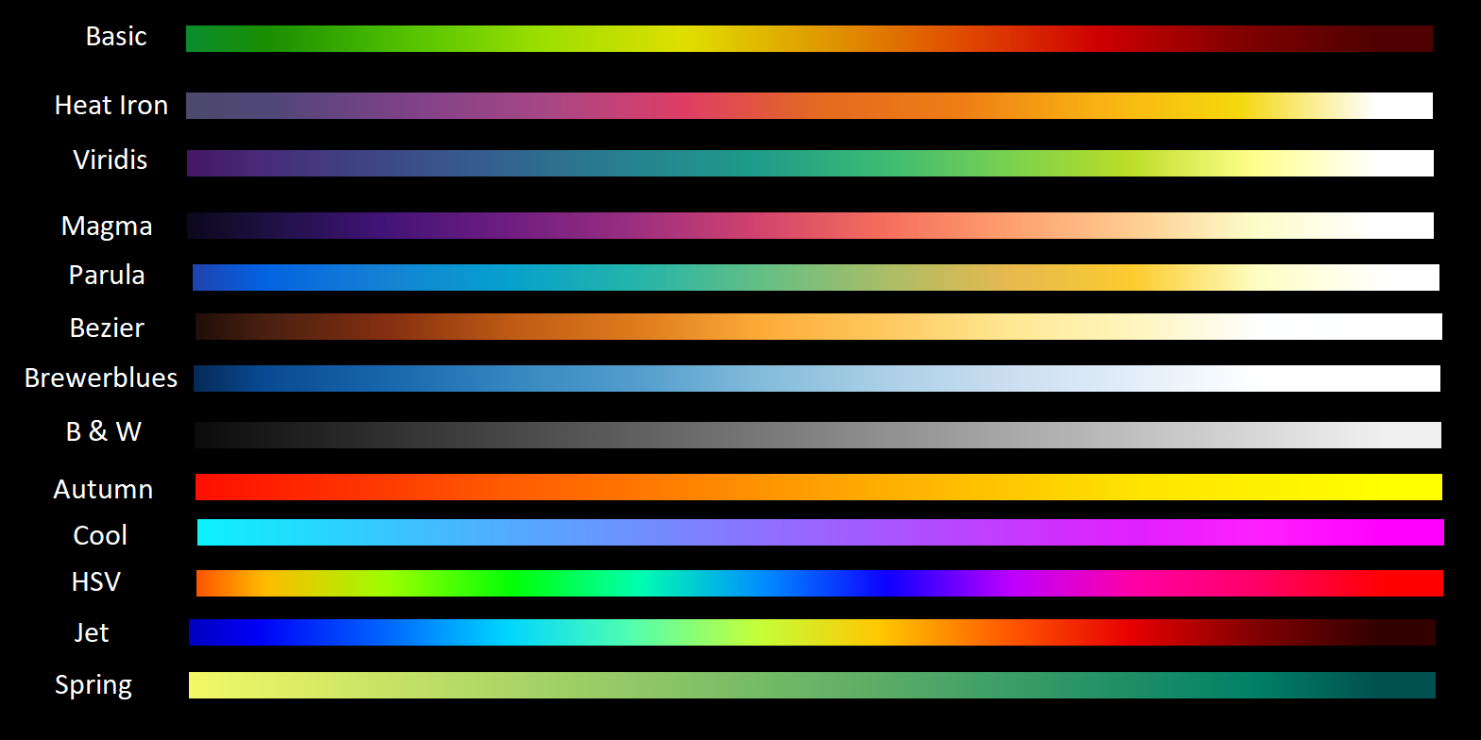
## Colorbar

This is an open source module library comes from <https://github.com/rtugeek/ColorSeekBar>. User can customize the color bar according to the API provided.

SenXorViewer is mainly focusing on providing suitable color palette selection in which are suitable for most of the use cases. However, this module is only controlling the color bar itself, but not the thermal images color mapping which should be on libuvccamera library. Please treat it as a UI module.

The customization including the bar/thumb size, the transparency and click/drag event handler. The color seed should not be modified because it has to match with the color table for data mapping. 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. Details please check thermalViewer/src/main/res/values/colors.xml.



## OpenCV

This is the Open Source Computer Vision Library for computational efficiency and with strong focus on real-time applications. It has C++/Python, Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. Simple image processing can be done with thermal image like edge detection or thresholding for face detection.

For more details, please check <https://opencv.org/>

## FaceDetection

Face detection is the module applying image processing using OpenCV on thermal image. There are various algorithms can be used to detect face and this module is using thresholding for temperature checking and convex hull of contours for checking the face shape on thermal image.

User may also develop their own algorithm to detect anything from CMOS image and/or thermal image. It is just an example to show how to apply intelligence on thermal images, as one of the use cases in security checking under low lighting environment.

**public class** DetectResult {  
**public boolean isFace**; *// True if the contour could be a face* **public** Rect **rect**; *// Rectangle that completely covers the contour. Least granular* **public** MatOfPoint **contour**; *// The contour. (Most granular)* **public** MatOfPoint **approxContour**; *// Approximate contour* **public** MatOfPoint **hull**; *// Polygon cover*}

…

**private static int** *MaxSkinInCBy10* = 400; *// Max detect temp, degrees (40.0) C multiplied by 10***private static int** *MinSkinInCBy10* = 290; *// Min detect temp, degrees (29.0) C multiplied by 10*

*/\*  
Minimum size. (Empirical) - Detected object must be larger than this size. E.g. 10% of the image size  
 \*/***private static int** *minRectSizePct* = 10;  
  
*/\*  
Maximum size. (Empirical) - Detected object must be smaller than this size. E.g. 90% of the image size  
\*/***private static int** *maxRectSizePct* = 90;

…

**public** List<DetectResult> detectFaces(Bitmap bitmap, **int**[] thermalData)

**public** Mat getImage(Context context, **int** resourceId )

**public** Mat[] getMonoChromeImageEx(**int**[] thermalData )

**public int** getUpperDetectTemperature()

**public int** getLowerDetectTemperature()

**public void** setUpperDetectTemperature(**int** input) *// input = degree \* 10 ,e.g: 400*

**public void** setLowerDetectTemperature(**int** input) *// input = degree \* 10 ,e.g: 290*

**public void** setMinRectSizePct(**int** input) *// Should be 0 <= input <= 100*

**public void** setMaxRectSizePct(**int** input)

**public** List<DetectResult> processContours( Mat monoImage )

**public void** displayImage(Context context, ImageView imageView, Mat image)

**public** Bitmap matToBitmap(Mat image)

The range of face detecting temperature is set at 29 – 40 °C by default.

## 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, also we can pass UI elements in setCaptureElement() for showing on screen capturing. The temperature array is updated from mTemperatureCallback. It contains some customized functions for XCAM:

* public float getTemperature(int posX, int posY)
* public void setCaptureElement(String temp, ImageView CrossHair, boolean isFlip)
* public void setThermalEnable(boolean enable)
* public void setCmosEnable(boolean enable)
* public void SetMeridianParams(int numPalette, int xShift, int yShift){
* public void setParaletteHDR(boolean input)
* public void setmIsThermalEnable(boolean enable)

<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.

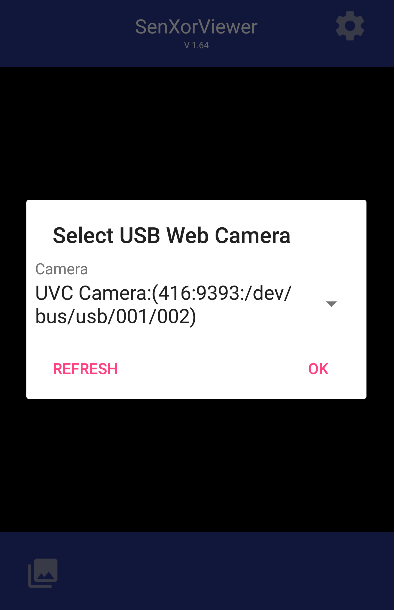
Thermal data upsampling is used to generate a bigger size of the thermal data that can make equivalent size of thermal image to overlay on the CMOS layer. The thermal data array size from XCAM is 32\*32, which is much smaller to the CMOS image sensor output.

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.

setColorBar(int palette) is settings which palettes user want to use.

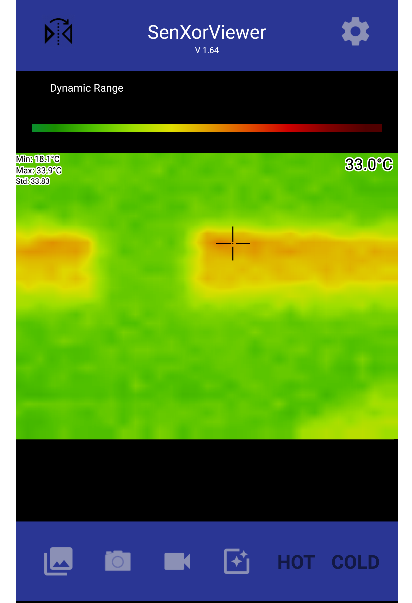
# Application

## USB Connect Dialog

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

Video Streaming



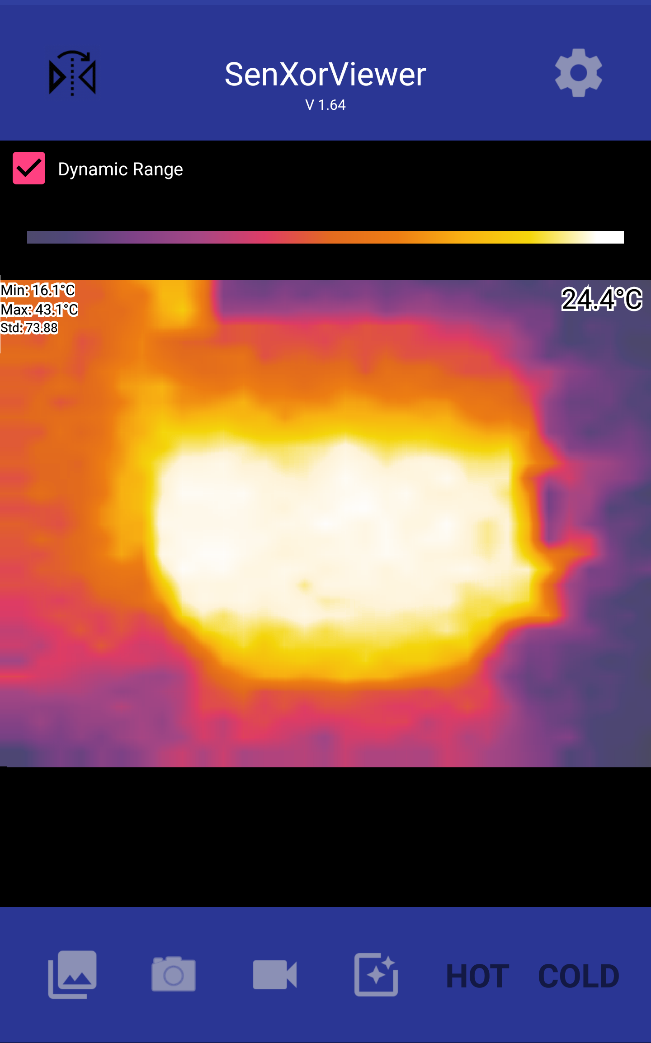
Hybrid Mode

CIS Image Mode

Thermal Image Mode

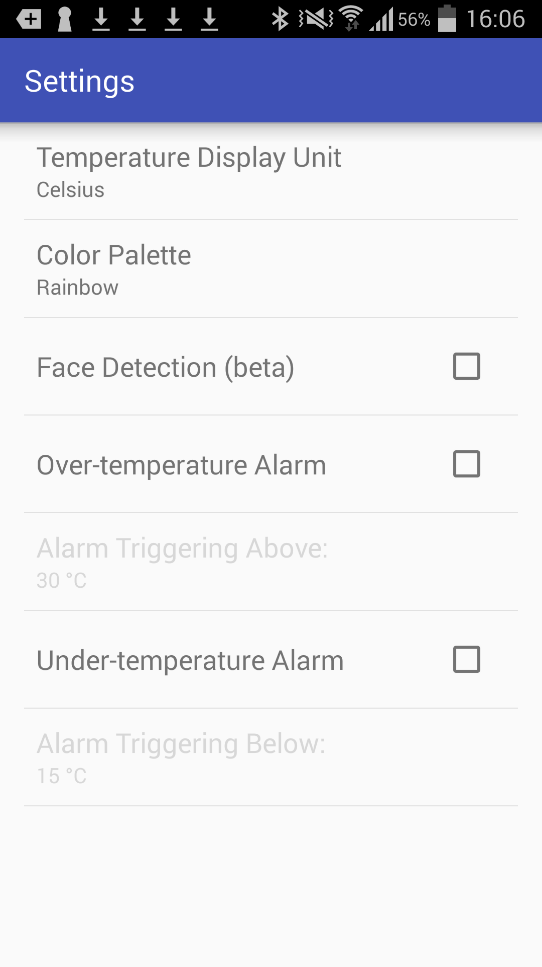
There are 3 preview modes by using XCAM. Press  to switch the mode.

## UI Components

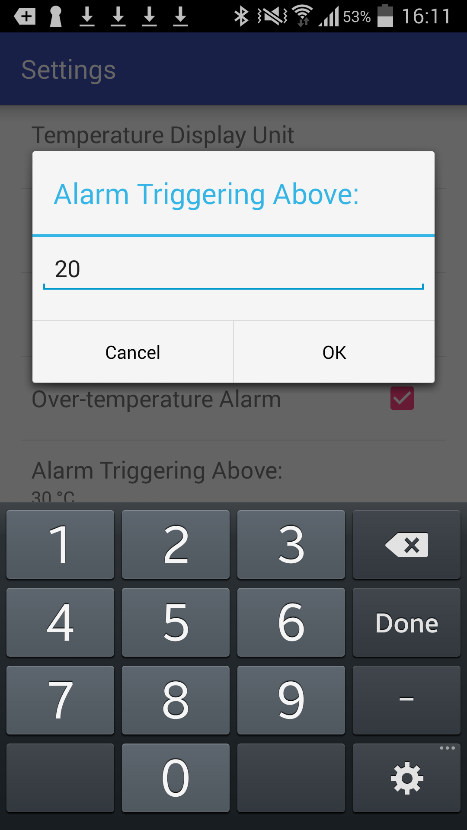


|  |  |
| --- | --- |
| Items | Description |
|  | Crosshair that follows the click event from user in preview window. It is also the position indicator which pin point the pixel and its temperature will be updated at the right corner. |
|  | X-axis / Horizontal mirroring the preview window |
|  | Settings |
|  | Open Gallery |
|  | Capture preview window |
|  | Record video of preview window |
|  | Preview mode |
|  | Temperature alarm indicators |
|  | Dynamic Range of color palette to map with thermal image. When it is ON, the palette will be mapping with the temperature range within the min/max on each frame |
|  | Color bar that is using for thermal image. User can switch the palette by clicking on it or in the setting. |

## Settings



Select Celsius or Fahrenheit



Enter threshold temperature

Set Under-temperature Alarm on or off

If on, set lower threshold temperature

Set Over-temperature Alarm on or off

If on, set upper threshold temperature

Set Face Detection on or off

Select Color Palette for thermal camera