USB Android Screen Manager Documentation and Instructions

Jet Propulsion Laboratory
WFIRST Coronagraph Development

Leo Neat
lneat@ucsc.edu

July 13, 2017

Description

This application is able to control an android's screen remotly over a usb connection to a configured desktop with no android UI.

USB Android Screen Manager was created to implement the new testbed concept of an android phone replacing a laser as a light source for testing a EM-CCD camera in photon counting mode. It is required that pixels go completely dark when their values were set to zero, meaning that only phones with the new Organic Light Emitting Diodes(OLED) pixels should be utilized for this application. We reccomend using the samsung galaxy s7 or s8 because of their high pixel density.

Theses are the base requirments that this application is able to fullfil.

1. Control the android device without a user interface(UI)
2. To have no stray lights or internal LEDs that would disturb the EMCCD.
3. Accept an RGB array as input for phone pixel values
4. Accept a flat field as input for phones pixel values
5. Accept the length of visible time as an input
6. Accept periodic spot matrices as input for phones pixel values
7. Allow the user to control the intensity of individual pixels

1 Enviroment Configuration

1.1 Android Studio Download and Configuration

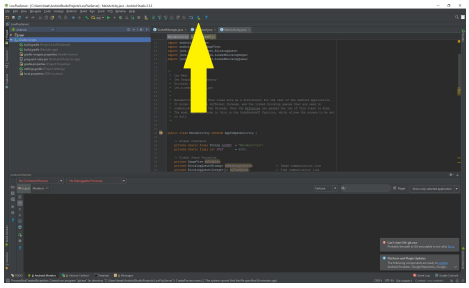
Android Studio is required in order to communicate with the phone remotely over USB. Android Studio is a Free application produced by Google that is an IDE specific to Android Devlopement. When I have time, I will try to programaticly inteface with android studio so that using it is not nessesary, but untill then it is required.

1. Go to the following link and press the DOWNLOAD button: `https://developer.android.com/studio/index.html`.
2. Run the installation executable.
3. Click next till the "Select components to install" box appears, make sure that Android SDK is checked on
4. Click next till the installation begins.
5. When install finsihes, click the box that says "Start Android Studio" and continue.
6. Click "Start a new Android Studio project".
7. Click next untill the IDE Appears, Note this project is just used to set up the enviroment

1.2 Android Debug Bridge(ADB) Download and Configuration

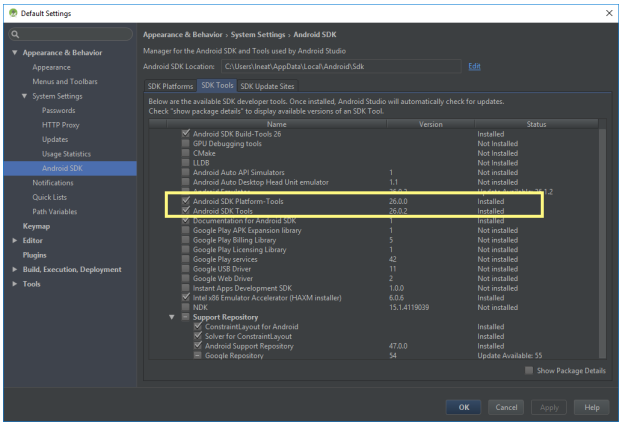
ADB is androids way of comunicating with an android device over USB. ADB is a package that needs to be installed using Androids SDK manager. A link to the ADB documentation can be found here: `https://developer.android.com/studio/command-line/adb.html`

Figure 1: SDK Manager



1. Open Android Studio
2. Click on the Android SDK Manager ICON(Fig. 1).
3. Under the side dropdown list click on "Appearance and Behavior" → "System Settings" → "Android SDK" .
4. Using the tabs at the top, Click on the SDK Tools tab.
5. Check the boxes "Android SDK Platform-Tools" and "Android SDK Tools"(Fig. 2)

Figure 2: Settings



6. Copy the filepath at the top of that window that says SDK Location
7. Click the apply button.
8. Open up Command Prompt.
9. Type "cd " and paste the directory that you copied and press enter.
10. Type "cd platform-tools" and press enter.
11. Type "Dir" and you should see a file named "adb.exe", if you do not then their was an error in your previous steps.
12. Type "echo %cd%" and copy the result file path
13. Close the Command Prompt.
14. Start the Control Panel.
15. Click on "System".
16. Select "Advanced system settings".

17. Click on the "Advanced" tab.
18. Click on "Environment Variables.
19. Click "New...".
20. Name the vairable "ADB".
21. Paste the filepath in to the "Variable value:" section.
22. Click OK.
23. Open up Command Prompt.
24. Type "adb".
25. If you do not get an error then this step is complete.

1.3 Python Client Download and Configuration

1. Click on the following link to the GitHub repository and download it: `https://github.com/Leo-Neat/LowFluxPythonClient`.
2. Extract the code from the downloaded ZIP file into your desired location.
3. Open up "LowFluxCLient.py" in your favorite python IDE or terminal

1.4 Android Server Download and Configuration

1. Click on the following link to the GitHub repository and download it: `https://github.com/Leo-Neat/LowFluxAndroidAppt`.
2. Extract the code from the downloaded ZIP file into your desired location.
3. Start up the Android Studio Application.
4. Your Sample Application should be loaded, Click "File" $\rightarrow$ "Close Project".
5. Click "Open a new Android Studio project".
6. Select the location of where you unzipped the git repository. A green logo should appear next to the file ou should select.
7. If an error pops up click through it, it is possable that some temporary files were replaced, and the error should be bypassed.
8. Now Your all setup to run the System.

2 Running the System

2.1 Android Initialization

1. Open Up the Android Studio Application and Load the LowFluxAndroidApp that was downloaded from GitHub.
2. Attach your device to the computer via USB port.
3. Make sure that your phone is in devloper mode and USB debugging is turned on. If not, here is a link to a tutorial on how to do that: `https://www.howtogeek.com/129728/how-to-access-the-developer-options-menu-and-enable-usb-debugging-on-android-4.2/`
4. Once your phone is attached and USB debuging is turned on, press the green play button at the top of Android Studio.
5. You will then be prompted ot pick a Deployment Target, If all is correct, you should see your phone under "Connected Devices".
6. Click on you phone and press OK
7. Now wait for the build to complete and the application to be running. You will know the application is running if your screen is completely blank.
8. The phone is now ready to be paired with a python client.

2.2 Python Initialization

1. Open up your favorite Python IDE or terminal.
2. Change your path to the directory inwhich the LowFluxPythonClient was downloaded
3. For a simple UI run the python file LowFluxCLient.py. This will give you different options on how to modify the Android Screen
4. For more Control you will want to import the RemoteScreen.py into your own python script.
5. Look at the API bellow for instructions on how to use the RemoteScreen Class.

3 Remote Screen API

3.1 Constants

1. **PORT** - This is an Integer that holds the value for the TCP comucation port between the Android server and the Python Client. If you change this value, you will also need to edit it on the Android Side in the ThreadManager.java file.

2. **NEWDATA** - This is an Integer that the Python client uses to signal the Andorid server when it wants to send a new screen, do not alter.

3. **SUCCESS** - This is an Integer that the Python client uses to signal the Andorid server when a screen is done sending, do not alter.

4. **DISCONNNECT** - This is an Integer that the Python client uses to signal the Andorid server when it wants to disconnect from the server, do not alter.

5. **SETTIME** - This is an Integer that the Python client uses to signal the Andorid server when it wants to set a new time delay for the screen, do not alter.

3.2 Class Variables

1. **sock** - This is a private socket variable that should only be used within the class. It gets initlized in the socketconfig method.

2. **screen** - This is a public 3D numpy arrray that acts as a bridge between the Android Screen and Python.

3. **width** - One of the parameters that is passed in to the constructor of the remote screen class. This specifys how many pixels wide your phone is. You can look that information up online.

4. **height** - One of the parameters that is passed into the constructor of the remote screen class. This specifys how many pixels high your phone is. You can look that information up online.

5. **csdt** - The Current Screen delay time. This is an integer in (ms) that represents how long the screen will be turned off for.

6. **csst** - The Current Screen Show tiem. This is an integer in (ms) that represents how long the screen will be on for.

7. **mode** - This is a String that holds the most recently called function in it. This can be used to tell what the current android screen state is.

3.3 Functions

1. **init** - (w,h) - (See width Class Variable, See height Class Variable) - Initializes the Remote screen class for controlling an android device connected via USB port. Note that adb needs to be added as an environment variable for this to work.

2. **socketconfig** - () - This function initializes the socket connection with the android device. It needs to be called after the ADB forward command or it will raise an error. This function needs to be called before sendscreen is called, or their will be no connection to send the screen to.

3. **sendscreen** - () - This function sends the current state of the screen to the android device screen. It is requried that the socketconfig function is called before this or this function will not work. This should be called as an updater method to sync the screen with the program.

4. **settime** - (See csdt Class Variable, See csst Class Variable) - Sets the time at which the image is shown on the android device. The first argument is the amount of time that is spent between when the image is shown and when the image is not show. The second argument is the length of time that the image is shown.

5. **disconnect** - () - Sends a message to the android client that the user wants to disconnect. Note: What ever screen is left on the android will stay their until another client changes it.

6. **changepixel** - (pixel's x value=INT, pixel's y value=INT, new intensity=INT) - Changes an individual pixel value on the screen. The first argument is the x coordinate of the pixel, the second argument is the y coordinate and the third is the intensity which is [0,255].

7. **turnblack** - () - Turns all of the pixels off effectively resetting the android screen.

8. **turncolor** - (The new intensity value of the full screen=INT) - Sets a flat feild for the screen of the android phone.

9. **useimage** - () - This method allows the user to pick an image for which they want the phone to be converted to. It uses the Tkinter library to open up a file browser in order to select a picture to enter. The picture Dimension must match the given width and height or an error will occur.

10. **evendistro** - (x spacing between pixels=INT, y spacing between pixels=INT, bright pixels intensity=INT, dark pixles intensity=INT) - This function allows the user to put a grid of pixels on to the android screen. The grid is defined by four Parameters. The x and y space are integers that depict the distance in pixels between where you want the bright spots. The bval is the intensity level of the bright spots [0,255] and the dval is the intensity of the dark spots [0,255].

11. **circle** - (radius of circle=INT, bright pixels value=INT, dark pixel value=INT, center x pixel position=INT, center y pixel position=INT) - This allows the user to draw a circle on the screen based upon the given parameters. The bval and dval represent the intensity of the bright and dark pixels [0,255]. The x and y pos are theposition of the center of the circle.

12. **printstate** - () - This class acts as a toString method. It allows the user to see what the current state of the screen is with out the full array being printed.

3.4 Things to Note

1. Even if you change the screen in python, the effect willnot take place on the phone unless you call the sendscreen method.
2. In order for this application to work, Numpy needs to be installed on your computer.
3. If you run into an error on the python side, you can just re-run the appli-action and re connect.
4. If you have any questions email me at address above.