

Wireless-X

Generated by Doxygen 1.9.0



<b>1 Main Page</b>	<b>1</b>
<b>2 WIRELESS-X</b>	<b>3</b>
2.1 All in one android app that wirelessly controls keyboard, mouse of Laptop/Desktop and also acts as a Webcam device. . . . .	3
2.2 Git repository link . . . . .	3
2.3 Team Members . . . . .	3
2.4 Contributions . . . . .	3
2.5 Introduction . . . . .	3
2.6 Motivation . . . . .	4
2.7 Working . . . . .	4
2.8 Installation Setup . . . . .	4
2.9 Running the application (Strictly follow the below order to run it successfully) . . . . .	4
2.9.1 *Extra (Inorder to remove v42loopback devices, use below command): . . . . .	5
2.10 Steps for Debugging (If python code doesn't run after above commands): . . . . .	5
2.11 References . . . . .	5
<b>3 Hierarchical Index</b>	<b>7</b>
3.1 Class Hierarchy . . . . .	7
<b>4 Class Index</b>	<b>9</b>
4.1 Class List . . . . .	9
<b>5 File Index</b>	<b>11</b>
5.1 File List . . . . .	11
<b>6 Class Documentation</b>	<b>13</b>
6.1 com.example.wireless_x.MainActivity Class Reference . . . . .	13
6.1.1 Detailed Description . . . . .	14
6.1.2 Member Function Documentation . . . . .	14
6.1.2.1 camera_switch() . . . . .	15
6.1.2.2 enter_wireless_x() . . . . .	15
6.1.2.3 keyPress() . . . . .	15
6.1.2.4 onBackPressed() . . . . .	15
6.1.2.5 onCameraFrame() . . . . .	15
6.1.2.6 onCreate() . . . . .	16
6.1.2.7 onDestroy() . . . . .	16
6.1.2.8 onPause() . . . . .	16
6.1.2.9 onRequestPermissionsResult() . . . . .	17
6.1.2.10 onResume() . . . . .	17
6.1.2.11 shiftPress() . . . . .	17
6.1.2.12 test_IP() . . . . .	17
<b>7 File Documentation</b>	<b>19</b>
7.1 MainActivity.java File Reference . . . . .	19

7.1.1 Detailed Description . . . . .	19
7.2 Wireless-X_server.py File Reference . . . . .	19
7.2.1 Detailed Description . . . . .	20
7.2.2 Function Documentation . . . . .	21
7.2.2.1 bind_sockets() . . . . .	21
7.2.2.2 camera_stream_connections() . . . . .	21
7.2.2.3 listening_connections() . . . . .	21
7.2.2.4 mouse_keyboard_connections() . . . . .	21
<b>Index</b>	<b>23</b>

## Chapter 1

# Main Page

Wireless-X consists of an android app backed by a python server. This app eliminates the need to buy a wireless mouse, wireless keyboard and a dedicated webcam. Using this app, the user can use his/her android smartphone's screen as mouse, a keyboard layout available on the app as the wireless keyboard, and his/her smartphone's camera as the webcam. A python server running on the target laptop/PC will capture these commands and emulate the effects on laptop.



## Chapter 2

# WIRELESS-X

### 2.1 All in one android app that wirelessly controls keyboard, mouse of Laptop/Desktop and also acts as a Webcam device.

### 2.2 Git repository link

[https://git.cse.iitb.ac.in/rajneeshkatkam/PARA-Site\\_WirelessX](https://git.cse.iitb.ac.in/rajneeshkatkam/PARA-Site_WirelessX)

### 2.3 Team Members

1. Aditya Pradhan (Roll number: 203059006)
2. Ajinkya Jumbad (Roll number: 203050032)
3. Prashant Ravi (Roll number: 203050082)
4. Rajneesh Katkam (Roll number: 203050086)

### 2.4 Contributions

1. Aditya Pradhan worked on python server and keyboard layout
2. Ajinkya Jumbad worked on mouse layout and the networking part of client-side
3. Prashant Ravi worked on python server and the networking part of client-side
4. Rajneesh Katkam worked on camera layout and the networking part of client-side

### 2.5 Introduction

The total package of Wireless-X consist of an android app being backed by a Python server. Using this app, we have tried to eliminate the need to buy a wireless mouse, wireless keyboard and a dedicated web-cam. Instead, using this app, user can use his/her android phone's screen as his mouse, a custom build keyboard layout as his wireless keyboard and his/her smartphone's camera as web-cam. A Python server running on target laptop will capture these commands and emulate the effects on laptop.

## 2.6 Motivation

People come from different economical background and buying a wireless mouse, or keyboard or web-cam may not be easy for everyone. Even if they can manage to buy one, it is not efficient to do so when they have already invested on a smartphone. One thing, that is very common now-a-days across all section of society is having a smartphone, either a basic or a high-end one doesn't matter. The required hardware to emulate a wireless keyboard, mouse or external web-cam is already present in a smartphone, and we just need to tailor it to make it usable for a common end-user.

## 2.7 Working

The Wireless-X app has two components: a python server running on laptop/PC and a client running on the android app. The user acts as a client where he/she sends the mouse and keyboard actions to be performed. If the user has turned on the camera, the camera frames are also sent to the python server. There are two sockets, one socket manages the mouse and keyboard actions, then the python program uses `autopy` and `pynput` libraries to translate these requests into the actual actions. The other socket is responsible for handling the camera frames, the python program uses the `pyfakewebcam` library to set up a camera virtual device on the laptop/PC and uses the `OpenCV` library to decode the camera frames coming from the client-side.

## 2.8 Installation Setup

1. Make sure that you are in the `WirelessX_Source_Code/WirelessX-Python-Server` directory (where `install.sh` file is located).
2. Grant the permission to execute `install.sh` installation script using the following command:  

```
$ sudo chmod a+x install.sh
```
3. Execute the `install.sh` script to install the necessary dependencies using the following command:  

```
$ sudo ./install.sh
```

## 2.9 Running the application (Strictly follow the below order to run it successfully)

NOTE: Before proceeding make sure that the Laptop/Desktop and the android phone are connected to same Wi-Fi/Hotspot.

1. Navigate to the `WirelessX_Source_Code/WirelessX-Python-Server` directory and run the Wireless-X server using the following command: `$ python3 Wireless-X_server.py`
2. Enter your linux system password (the same password you enter while executing a command as "sudo"). This is required in order to set up the virtual webcam device.
3. Application Installation and Setup on Android Smartphone:
  - a. Install the Wireless-X apk (you can copy the apk from the folder `WirelessX_Source` to your smartphone) on Android smartphone and give required permissions.
  - b. Now, enter the IP address displayed in the terminal (on which the server is running) into the android application.
  - c. Click on Test Button to test the connection of smartphone with the server. If failed, Recheck if you have entered correct IP address of Laptop/ Desktop (on which the server is running).



- d. After successful connection, you would be able to control mouse, keyboard of laptop and use smartphones camera as webcam for the laptop/Desktop.
- e. Now you would be able to use this virtual webcam device on Google Chrome for video conferencing. (Tested on Google Chrome for Microsoft Teams and Google Meet).
- f. (Optional) Inorder to test if camera frames are received to the Laptop/ Desktop, use the below command while (Note: camera option should be turned on in the Wireless-X apk on Android): `$ ffmpeg -i /dev/video20`

### 2.9.1 \*Extra (Inorder to remove v4l2loopback devices, use below command):

```
$ sudo modprobe -r v4l2loopback
```

## 2.10 Steps for Debugging (If python code doesn't run after above commands):

1. Check if your virtual device is created

```
$ ls /dev | grep -P '^video\d+$'
OR
$ v4l2-ctl --list-devices      # TO List the virtual devices in detail
```

Output should look somewhat like this:

```
Wireless-X Camera (platform:v4l2loopback-000):
/dev/video20
```

```
Webcam C170: Webcam C170 (usb-0000:00:1a.0-1.2):
/dev/video0
/dev/video1
```

2. Inorder to test if virtual device is working:

Copy the sample code from <https://github.com/jremmons/pyfakewebcam> page and save it as python file and run

```
$ python3 demo.py
```

If everything worked correctly, no error should be displayed and terminal should be blank.

Now, Open another terminal and test if virtual device output is being display by entering below command:

```
$ ffmpeg -i /dev/video20
```

Note: ffmpeg

## 2.11 References

OpenCV Reference:

<https://cmsdk.com/java/android-opencv-tcp-video-streaming.html>

v4l2loopback References:

<https://github.com/jremmons/pyfakewebcam>  
<https://github.com/jremmons/pyfakewebcam/issues/5>  
<https://github.com/umlaeute/v4l2loopback#DKMS>

Android Reference:

<https://stackoverflow.com/questions/23024831/android-shared-preferences-for-creating-one-time-activity-example>



## Chapter 3

# Hierarchical Index

### 3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

CvCameraViewListener2	
com.example.wireless_x.MainActivity . . . . .	<a href="#">13</a>
AppCompatActivity	
com.example.wireless_x.MainActivity . . . . .	<a href="#">13</a>



## Chapter 4

# Class Index

### 4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

[com.example.wireless\\_x.MainActivity](#)

This is where the main code of the Wireless-X android application is written . . . . . 13



## Chapter 5

# File Index

### 5.1 File List

Here is a list of all documented files with brief descriptions:

<a href="#">MainActivity.java</a>	
This is where the main code of the Wireless-X android application is written . . . . .	19
<a href="#">Wireless-X_server.py</a>	
This includes the code for the server-side of Wireless-X . . . . .	19





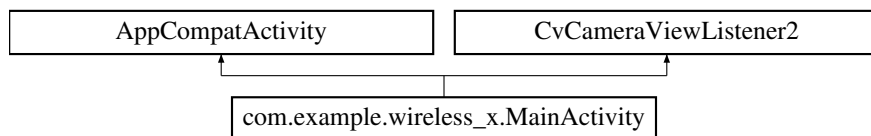
## Chapter 6

# Class Documentation

### 6.1 com.example.wireless\_x.MainActivity Class Reference

This is where the main code of the Wireless-X android application is written.

Inheritance diagram for com.example.wireless\_x.MainActivity:



#### Classes

- class **SendKeyboardPressesThread**  
*Used to send the keyboard events to the server.*
- class **SendMouseClicks**  
*Used to send the mouse click events to the server.*
- class **SendMouseCoordinatesThread**  
*Sends the mouse coordinates to the server.*
- class **TestIP\_Thread**  
*Tests whether the server's IP address is valid or not.*

#### Public Member Functions

- void [shiftPress](#) (View view)  
*Displays the keys which correspond to special characters.*
- void [test\\_IP](#) (View view)  
*Describes the action to be performed when Test IP is clicked on the app.*
- String [getEmojiByUnicode](#) (int unicode)  
*Returns the emoji corresponding to an unicode.*
- void [enter\\_wireless\\_x](#) (View view)  
*Performs the action when the "Enter Wireless-X" button is clicked.*

- void [onBackPressed](#) ()  
*Performs the action when the back button is pressed.*
- void [mouse\\_on\\_off](#) (View view)  
*Enables or disables the visibility of Mouse UI.*
- void [camera\\_on\\_off](#) (View view)  
*Enables or disables the camera layout.*
- void [camera\\_switch](#) (View view)  
*Implementation of the camera switch button functionality.*
- void [onRequestPermissionsResult](#) (int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults)  
*Sets up the camera view if all the permissions are granted.*
- void [onResume](#) ()  
*Handles the onResume state of the app.*
- void [onPause](#) ()  
*Handles the onPause state of the app.*
- void [onDestroy](#) ()  
*Handles the onDestroy state of the app.*
- Mat [onCameraFrame](#) (CameraBridgeViewBase.CvCameraViewFrame inputFrame)  
*Transmits the camera frames to the server.*
- void [layout\\_switch](#) (View view)  
*Sets up the layout as defined in the "activity\_main.xml" file.*
- void [mouse\\_click](#) (View view)  
*Sends the mouse clicks.*
- void [keyPress](#) (View view)  
*Handles the key press event.*

## Protected Member Functions

- void [onCreate](#) (Bundle savedInstanceState)  
*Sets up the app layout and contains the methods to handle various touch-related events.*

### 6.1.1 Detailed Description

This is where the main code of the Wireless-X android application is written.

The [MainActivity](#) consists of the methods that initialize all the required variables and fields when the app starts, methods which keep listening to the mouse and keyboard events such as a mouse click event or a key press event, screen touch events, methods which send the camera frames to the virtual camera device running on the laptop and so on.

### 6.1.2 Member Function Documentation

### 6.1.2.1 camera\_switch()

```
void com.example.wireless_x.MainActivity.camera_switch (
    View view ) [inline]
```

Implementation of the camera switch button functionality.

This method changes the main camera to the front or rear camera of the smartphone depending upon what the user has selected.

### 6.1.2.2 enter\_wireless\_x()

```
void com.example.wireless_x.MainActivity.enter_wireless_x (
    View view ) [inline]
```

Performs the action when the "Enter Wireless-X" button is clicked.

This method displays the mouse layout once the user clicks on "Enter Wireless-X" button.

### 6.1.2.3 keyPress()

```
void com.example.wireless_x.MainActivity.keyPress (
    View view ) [inline]
```

Handles the key press event.

This method handles the key press event and also handles the scroll button available on mouse layout.

### 6.1.2.4 onBackPressed()

```
void com.example.wireless_x.MainActivity.onBackPressed ( ) [inline]
```

Performs the action when the back button is pressed.

It checks whether the back button is pressed twice within 2 seconds, if it is, then it exits the app. It also saves the IP address of the server so that the user doesn't need to re-enter it the next time he/she opens the app.

### 6.1.2.5 onCameraFrame()

```
Mat com.example.wireless_x.MainActivity.onCameraFrame (
    CameraBridgeViewBase.CvCameraViewFrame inputFrame ) [inline]
```

Transmits the camera frames to the server.

On receiving a camera frame, this method encodes that frame and transmits it to the server.

### 6.1.2.6 onCreate()

```
void com.example.wireless_x.MainActivity.onCreate (
    Bundle savedInstanceState ) [inline], [protected]
```

Sets up the app layout and contains the methods to handle various touch-related events.

Initializes all the app components and contains an listener for those events which can occur when the user interacts with the screen by single tap, double tap, scrolling or some gesture on the screen. Method which listens for screen-touch related events.

When the user performs a double tap, it is translated to the double left-click on a physical mouse. Similarly, when the user performs a single tap, it's effect is same as a single click on any physical mouse. There is also an onScroll event which corresponds to the mouse scrolling event. This listener uses the GestureDetector class to handle such events.

Method to handle the double-tap event.

When the user performs a double tap, it is translated to the double left-click on a physical mouse. This is done by starting two threads simultaneously, which product the effect of two single-clicks without much delay, thus corresponding to a double-click.

Method to handle the single-tap event.

When the user performs a single tap, it is translated to the single left-click on a physical mouse. This is done by starting a thread, which sends the event information to the server running on laptop and then the server acts accordingly.

Method to handle the mouse scrolling event.

When the user performs a scroll event, the coordinates are transferred to the server, which translates those coordinates to the position with respect to the laptop screen.

Method to handle the screen-touch event.

This method calls the GestureDetector object to handle the screen-touch event which can be any one of the single-tap, double-tap or scroll events.

### 6.1.2.7 onDestroy()

```
void com.example.wireless_x.MainActivity.onDestroy ( ) [inline]
```

Handles the onDestroy state of the app.

If the app reaches the "onDestroy" state in the lifecycle, then this method disables the camera view.

### 6.1.2.8 onPause()

```
void com.example.wireless_x.MainActivity.onPause ( ) [inline]
```

Handles the onPause state of the app.

If the app reaches the "onPause" state in the lifecycle, then this method disables the camera view. It also saves the server's IP address so that the next time the app is opened, the user doesn't require to enter the same address again.

### 6.1.2.9 onRequestPermissionsResult()

```
void com.example.wireless_x.MainActivity.onRequestPermissionsResult (
    int requestCode,
    @NonNull String[] permissions,
    @NonNull int[] grantResults ) [inline]
```

Sets up the camera view if all the permissions are granted.

This method initializes all the camera parameters subject to the condition that all the required permissions are granted by the user. If this is not the case, then an error message is displayed.

### 6.1.2.10 onResume()

```
void com.example.wireless_x.MainActivity.onResume ( ) [inline]
```

Handles the onResume state of the app.

If the app reaches an "onResume" state in the lifecycle, then this method checks if all the permissions are granted or not, if they are, then it sets up camera parameters otherwise it requests the permissions.

### 6.1.2.11 shiftPress()

```
void com.example.wireless_x.MainActivity.shiftPress (
    View view ) [inline]
```

Displays the keys which correspond to special characters.

When the shift-key is pressed on the keyboard in Wireless-X app, this method changes the layout of some keys to those keys which correspond to special characters such as brackets, '@', etc.

### 6.1.2.12 test\_IP()

```
void com.example.wireless_x.MainActivity.test_IP (
    View view ) [inline]
```

Describes the action to be performed when Test IP is clicked on the app.

This method tries to set-up a connection with the IP address entered in the textfield to check if the IP address entered by the user is valid or not.

The documentation for this class was generated from the following file:

- [MainActivity.java](#)



## Chapter 7

# File Documentation

### 7.1 MainActivity.java File Reference

This is where the main code of the Wireless-X android application is written.

#### Classes

- class [com.example.wireless\\_x.MainActivity](#)  
*This is where the main code of the Wireless-X android application is written.*
- class **com.example.wireless\_x.MainActivity.TestIP\_Thread**  
*Tests whether the server's IP address is valid or not.*
- class **com.example.wireless\_x.MainActivity.SendMouseCoordinatesThread**  
*Sends the mouse coordinates to the server.*
- class **com.example.wireless\_x.MainActivity.SendMouseClicks**  
*Used to send the mouse click events to the server.*
- class **com.example.wireless\_x.MainActivity.SendKeyboardPressesThread**  
*Used to send the keyboard events to the server.*

#### 7.1.1 Detailed Description

This is where the main code of the Wireless-X android application is written.

### 7.2 Wireless-X\_server.py File Reference

This includes the code for the server-side of Wireless-X.

#### Functions

- def [Wireless-X\\_server.bind\\_sockets](#) ()  
*This function establishes two sockets for receiving camera frames as well as mouse and keyboard actions.*
- def [Wireless-X\\_server.mouse\\_keyboard\\_connections](#) ()  
*This function decodes the received mouse and keyboard actions and acts accordingly.*
- def [Wireless-X\\_server.camera\\_stream\\_connections](#) ()  
*This function is responsible for handling the camera frames.*
- def [Wireless-X\\_server.listening\\_connections](#) ()  
*This function is responsible for listening to connections.*

## Variables

- `Wireless-X_server.virtualCamera` = `subprocess.run(["sudo", "modprobe", "v4l2loopback", "devices=1", "video_nr=20", "card_label=Wireless-X Camera", "exclusive_caps=1"])`  
*Creates a virtual camera on the laptop/PC.*
- `Wireless-X_server.width`  
*Stores the width of the screen.*
- `Wireless-X_server.height`  
*Stores the height of the screen.*
- `Wireless-X_server.curr_x`  
*Stores the x-coordinate of the current mouse location.*
- `Wireless-X_server.curr_y`  
*Stores the y-coordinate of the current mouse location.*
- `def Wireless-X_server.remote_x = curr_x/2`  
*Stores the mid of x-coordinate of current mouse location.*
- `def Wireless-X_server.remote_y = curr_y/2`  
*Stores the mid of y-coordinate of current mouse location.*
- `string Wireless-X_server.s = "`  
*Socket used for receiving keyboard and mouse related actions.*
- `string Wireless-X_server.cameraSocket = "`  
*Socket used for receiving the camera frames of user's smartphone.*
- `int Wireless-X_server.img_width = 720`  
*Width of the camera frame.*
- `int Wireless-X_server.img_height = 480`  
*Height of the camera frame.*
- `Wireless-X_server.camera = pyfakewebcam.FakeWebcam('/dev/video20', img_width, img_height)`  
*Virtual webcam device.*
- `bool Wireless-X_server.thread_run = True`  
*The camera and keyboard-mouse sockets receive user requests until this variable is set to 'True'.*
- `Wireless-X_server.keyboard = KeyboardController()`  
*Initializing the KeyboardController object.*
- `Wireless-X_server.mouse = MouseController()`  
*Initializing the MouseController object.*
- `int Wireless-X_server.mouse_speed = 2`  
*Speed of mouse movement.*
- `int Wireless-X_server.screenshot_count = 0`  
*Screenshot counter to keep track of screenshots.*
- dictionary `Wireless-X_server.special_key_android_dictionary` = `{"F1": "F1", "F2": "F2", "F3": "F3", "F4": "F4", "F5": "F5", "F6": "F6", "F7": "F7", "F8": "F8", "F9": "F9", "F10": "F10", "F11": "F11", "F12": "F12", "Alt": "ALT", "Backspace": "BACKSPACE", "Caps\nLock": "CAPS_LOCK", "Ctrl": "CONTROL", "Delete": "DELETE", "↓": "DOWN_ARROW", "End": "END", "Esc": "ESCAPE", "Home": "HOME", "←": "LEFT_ARROW", "META": "META", "Page Down": "PAGE_DOWN", "Page Up": "PAGE_UP", "Enter": "RETURN", "→": "RIGHT_ARROW", "Shift": "SHIFT", "Space": "SPACE", "↑": "UP_ARROW", "Tab": "Tab"}`  
*Maps the keys in keyboard layout to the actual keyboard keys.*

### 7.2.1 Detailed Description

This includes the code for the server-side of Wireless-X.

The server running on laptop or PC is responsible for receiving the actions performed by user on the Wireless-X android app as well as receiving the camera frames of the user's smartphone (if the user has turned) on the camera). Such actions are transmitted to the server in encoded form, the server decodes the received message and instructs the laptop or PC to perform the action described in that message.



## 7.2.2 Function Documentation

### 7.2.2.1 bind\_sockets()

```
def Wireless-X_server.bind_sockets ( )
```

This function establishes two sockets for receiving camera frames as well as mouse and keyboard actions.

This function creates a camera socket which is responsible for receiving the camera frames, and it also creates another socket which is responsible for receiving the keyboard and mouse frames.

### 7.2.2.2 camera\_stream\_connections()

```
def Wireless-X_server.camera_stream_connections ( )
```

This function is responsible for handling the camera frames.

This function uses the 'OpenCV' library to decode and resize the camera frame. Then, this frame is scheduled on the virtual webcam device created using 'pyfakewebcam' library.

### 7.2.2.3 listening\_connections()

```
def Wireless-X_server.listening_connections ( )
```

This function is responsible for listening to connections.

This function creates two threads corresponding to the two sockets, one for handling mouse and keyboard events and the other for handling camera frames received from the user's smartphone.

### 7.2.2.4 mouse\_keyboard\_connections()

```
def Wireless-X_server.mouse_keyboard_connections ( )
```

This function decodes the received mouse and keyboard actions and acts accordingly.

This function checks if the message received corresponds to a mouse action (left-click, scroll, etc.) or a key (such as keypress). It then instructs the laptop to perform these actions, using the 'autopy' and 'pynput' libraries. Characters in keyboard are not supported by 'autopy' library, so the actions corresponding to these special characters are handled by the 'pynput' library, other key actions are handled by the 'autopy' library. In case of mouse, the 'autopy' library is efficient, so we used the 'pynput' library.



# Index

- bind\_sockets
  - Wireless-X\_server.py, [21](#)
- camera\_stream\_connections
  - Wireless-X\_server.py, [21](#)
- camera\_switch
  - com.example.wireless\_x.MainActivity, [14](#)
- com.example.wireless\_x.MainActivity, [13](#)
  - camera\_switch, [14](#)
  - enter\_wireless\_x, [15](#)
  - keyPress, [15](#)
  - onBackPressed, [15](#)
  - onCameraFrame, [15](#)
  - onCreate, [15](#)
  - onDestroy, [16](#)
  - onPause, [16](#)
  - onRequestPermissionsResult, [16](#)
  - onResume, [17](#)
  - shiftPress, [17](#)
  - test\_IP, [17](#)
- enter\_wireless\_x
  - com.example.wireless\_x.MainActivity, [15](#)
- keyPress
  - com.example.wireless\_x.MainActivity, [15](#)
- listening\_connections
  - Wireless-X\_server.py, [21](#)
- MainActivity.java, [19](#)
- mouse\_keyboard\_connections
  - Wireless-X\_server.py, [21](#)
- onBackPressed
  - com.example.wireless\_x.MainActivity, [15](#)
- onCameraFrame
  - com.example.wireless\_x.MainActivity, [15](#)
- onCreate
  - com.example.wireless\_x.MainActivity, [15](#)
- onDestroy
  - com.example.wireless\_x.MainActivity, [16](#)
- onPause
  - com.example.wireless\_x.MainActivity, [16](#)
- onRequestPermissionsResult
  - com.example.wireless\_x.MainActivity, [16](#)
- onResume
  - com.example.wireless\_x.MainActivity, [17](#)
- shiftPress
  - com.example.wireless\_x.MainActivity, [17](#)
- test\_IP
  - com.example.wireless\_x.MainActivity, [17](#)
- Wireless-X\_server.py, [19](#)
  - bind\_sockets, [21](#)
  - camera\_stream\_connections, [21](#)
  - listening\_connections, [21](#)
  - mouse\_keyboard\_connections, [21](#)