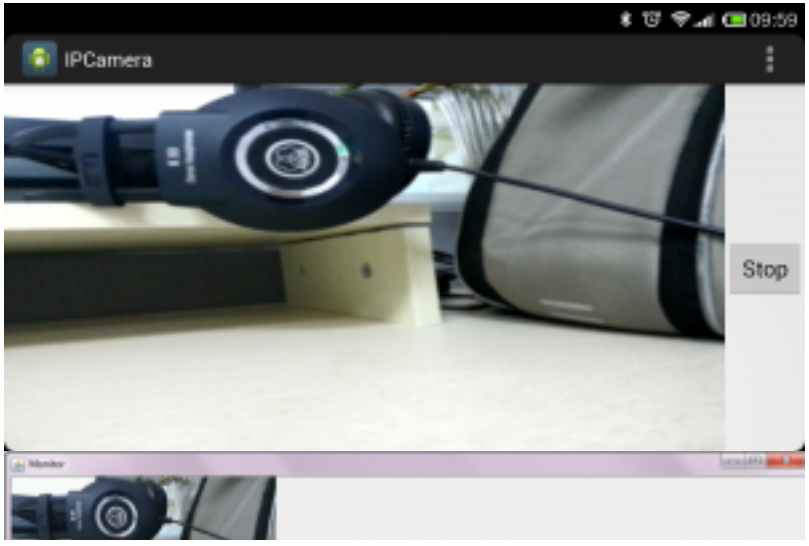
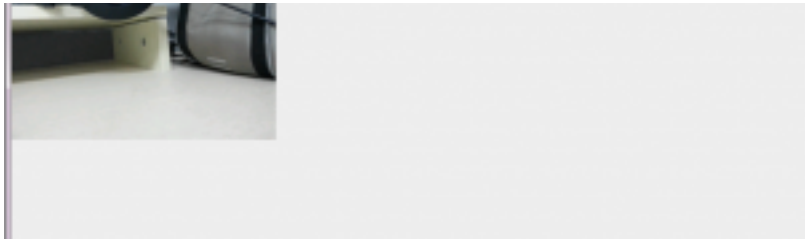
Making Android Smart Phone a Remote IP Camera

Desmond Shaw / September 1, 2014 / Android / Android, camera, socket, Swing 8 Comments

When you are going to purchase a new smart phone like iPhone 6 or Galaxy S5, don't just throw your old devices away. Via socket connection, we can build a remote monitoring system with obsolete mobile devices instead of purchasing expensive Webcams or wireless cameras. In this article, I'd like to share how to connect an Android camera app to a remote server by socket, as well as how to display the camera preview frames of my mobile device on the server Window.





Thinking

The following points helped me figure out the solution:

1. Create a customized Android camera application, and capture preview data from Android camera.
2. Send the preview data to a remote server frame by frame through Socket.
3. Convert the preview data format from NV21 to RGB on server.
4. Draw and display RGB data in Swing component.

Android Camera – Socket Connection – Remote Monitor

To quickly build up a custom Android camera application, we don't need to spend too much time. Google has already provided us an online tutorial Camera. Based on the source code, we just need to make a little bit of improvements.

Create a preview callback to receive copies of preview frames:

```
1  private Camera.PreviewCallback mPreviewCallback = new PreviewCallback() {
2
```

```
        @Override
        public void onPreviewFrame(byte[] data, Camera camera) {
            // TODO Auto-generated method stub
            synchronized (mQueue) {
                if (mQueue.size() == MAX_BUFFER) {
                    mQueue.poll();
                }
                mQueue.add(data);
            }
         }
     };
```

Register preview callback before starting the camera preview:

```
try {
            mCamera.setPreviewCallback(mPreviewCallback);
            mCamera.setPreviewDisplay(mHolder);
            mCamera.startPreview();

        } catch (Exception e){
            Log.d(TAG, "Error starting camera preview: " + e.getMessage());
        }
```

Do not forget to remove preview callback before stopping the camera:

```
public void onPause() {
        if (mCamera != null) {
```

```
            mCamera.setPreviewCallback(null);
            mCamera.stopPreview();
        }
        resetBuff();
    }
```

Use **AlertDialog** to initialize the IP address and port number:

```
private void setting() {
        LayoutInflater factory = LayoutInflater.from(this);
        final View textEntryView = factory.inflate(R.layout.server_setting, null);
        AlertDialog dialog =  new AlertDialog.Builder(IPCamera.this)
            .setIconAttribute(android.R.attr.alertDialogIcon)
            .setTitle(R.string.setting_title)
            .setView(textEntryView)
            .setPositiveButton(R.string.ok, new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int whichButton) {

                    EditText ipEdit = (EditText)textEntryView.findViewById(R.id.ip_edit);
                    EditText portEdit = (EditText)textEntryView.findViewById(R.id.port_edit);
                    mIP = ipEdit.getText().toString();
                    mPort = Integer.parseInt(portEdit.getText().toString());

                    Toast.makeText(IPCamera.this, "New address: " + mIP + ":" + mPort, Toast.LENGT
                }
            })
            .setNegativeButton(R.string.cancel, new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int whichButton) {

```

```
                    /* User clicked cancel so do some stuff */
                }
            })
            .create();
        dialog.show();
    }
```

When clicking the start button, a socket connection will be triggered in a worker thread. We can wrap type, length, width and height in a JSON message for communication. Once the server receives the message, it will make an initialization and allocate memory for preview frames. The worker thread won't stop sending preview frames until the stop button is pressed:

```
mSocket = new Socket();
mSocket.connect(new InetSocketAddress(mIP, mPort), 10000);
BufferedOutputStream outputStream = new BufferedOutputStream(mSocket.getOutputStream());
BufferedInputStream inputStream = new BufferedInputStream(mSocket.getInputStream());

JsonObject jsonObj = new JsonObject();
    jsonObj.addProperty("type", "data");
    jsonObj.addProperty("length", mCameraPreview.getPreviewLength());
    jsonObj.addProperty("width", mCameraPreview.getPreviewWidth());
    jsonObj.addProperty("height", mCameraPreview.getPreviewHeight());

    byte[] buff = new byte[256];
    int len = 0;
    String msg = null;
    outputStream.write(jsonObj.toString().getBytes());
    outputStream.flush();
```

```
17
18      while ((len = inputStream.read(buff)) != -1) {
19          msg = new String(buff, 0, len);
20
21          // JSON analysis
22          JsonParser parser = new JsonParser();
23          boolean isJSON = true;
24          JsonElement element = null;
25          try {
26              element =  parser.parse(msg);
27          }
28          catch (JsonParseException e) {
29              Log.e(TAG, "exception: " + e);
30              isJSON = false;
31          }
32          if (isJSON && element != null) {
33              JsonObject obj = element.getAsJsonObject();
34              element = obj.get("state");
35              if (element != null && element.getAsString().equals("ok")) {
36                  // send data
37                  while (true) {
38                      outputStream.write(mCameraPreview.getImageBuffer());
39                      outputStream.flush();
40
41                      if (Thread.currentThread().isInterrupted())
42                          break;
43                  }
44
45                  break;
46              }
47          }
```

```
            else {
                break;
            }
        }

        outputStream.close();
        inputStream.close();
```

On the server side, use a buffer queue to cache incoming data:

```
public int fillBuffer(byte[] data, int off, int len, LinkedList<byte[]> YUVQueue) {
        mTotalLength += len;
        mByteArrayOutputStream.write(data, off, len);

        if (mTotalLength == mFrameLength) {

            synchronized (YUVQueue) {
                YUVQueue.add(mByteArrayOutputStream.toByteArray());
                mByteArrayOutputStream.reset();
            }

            mTotalLength = 0;
            System.out.println("received file");
        }

        return 0;
    }
```

Referring to StackOverflow, we can use the following code to decode the NV21 data:

```
public static int[] convertYUVtoRGB(byte[] yuv, int width, int height)
            throws NullPointerException, IllegalArgumentException {
        int[] out = new int[width * height];
        int sz = width * height;

        int i, j;
        int Y, Cr = 0, Cb = 0;
        for (j = 0; j < height; j++) {
            int pixPtr = j * width;
            final int jDiv2 = j >> 1;
            for (i = 0; i < width; i++) {
                Y = yuv[pixPtr];
                if (Y < 0)
                    Y += 255;
                if ((i & 0x1) != 1) {
                    final int cOff = sz + jDiv2 * width + (i >> 1) * 2;
                    Cb = yuv[cOff];
                    if (Cb < 0)
                        Cb += 127;
                    else
                        Cb -= 128;
                    Cr = yuv[cOff + 1];
                    if (Cr < 0)
                        Cr += 127;
                    else
                        Cr -= 128;
                }
                int R = Y + Cr + (Cr >> 2) + (Cr >> 3) + (Cr >> 5);
                if (R < 0)
```

```
                    R = 0;
                else if (R > 255)
                    R = 255;
                int G = Y - (Cb >> 2) + (Cb >> 4) + (Cb >> 5) - (Cr >> 1)
                        + (Cr >> 3) + (Cr >> 4) + (Cr >> 5);
                if (G < 0)
                    G = 0;
                else if (G > 255)
                    G = 255;
                int B = Y + Cb + (Cb >> 1) + (Cb >> 2) + (Cb >> 6);
                if (B < 0)
                    B = 0;
                else if (B > 255)
                    B = 255;
                out[pixPtr++] = 0xff000000 + (B << 16) + (G << 8) + R;
            }
        }

        return out;
    }
```

Create **BufferedImage** for rendering in Swing component:

```
BufferedImage bufferedImage = null;
int[] rgbArray = Utils.convertYUVtoRGB(data, mWidth, mHeight);
bufferedImage = new BufferedImage(mWidth, mHeight, BufferedImage.TYPE_USHORT_565_RGB);
bufferedImage.setRGB(0, 0, mWidth, mHeight, rgbArray, 0, mWidth);
```

```
public void paint(Graphics g) {
        synchronized (mQueue) {
            if (mQueue.size() > 0) {
                mLastFrame = mQueue.poll();
            }
        }
        if (mLastFrame != null) {
            g.drawImage(mLastFrame, 0, 0, null);
        }
        else if (mImage != null) {
            g.drawImage(mImage, 0, 0, null);
        }
    }
```

Source Code

https://github.com/DynamsoftRD/Android-IP-Camera

```
git clone https://github.com/DynamsoftRD/Android-IP-Camera.git
```

Desmond Shaw

8 Comments Code Pool

Recommend Share S

Join the discussion…

Alex Smilyanskiy • 6 months ago

Thanks for this tutorial man! Searching for this for about a month and actualy find. Very useful, code is ea understand.

• Reply • Share ›

Vighnesh Bheed • a year ago

Hi,

Learnt A lot from the tutorial. but still having some issues...
The app was running on mobile and the Monitor part on eclipse IDE, the phone was connected to the Pc cable .

I wasnt Able to get any output on my monitor scene . and the log on eclipse was stuck saying servers wa

• Reply • Share ›

Desmond Shaw Mod ➚ Vighnesh Bheed • a year ago

The preview data can't be transfered via USB cable. I think the server is waiting for socket connec

• Reply • Share ›

KakashiHatake • a year ago

Hi,
The tutorial was awesome learnt a lot from it but i cant get one part right.
I started the server side app with eclipse and was running the IPcam on my phone . connected to PC via
cant see any output on the monitor
. I dont know whats wrong i exactly followed the code ...
forgive my bad english..

• Reply • Share ›

Rakkesh Kiren • a year ago

Hi,
I was wondering if it is possible to view the image on another Android phone rather than using your comp
this code be modified to do that?

• Reply • Share ›

Desmond Shaw Mod ➚ Rakkesh Kiren • a year ago

Yes. If you want to view the image on another Android phone, you just need to create a simple Act
view - ImageView or any custom view - for rendering the received image data.

• Reply • Share ›

Holger • 2 years ago

Hi
I am not a programmer, but maybee you could give me a hint, where could find an app, that can simply ta
on request and send it to a predefined mail address.
The request should best be triggered by a sms with some keyword. The idea is, to place my old mobile ir
and get a picture of the environment, whenever I want it. To save energy the device should be simply in s
the time until it receives the SMS.
Does anybody have an idea where I can find such an app or who could program that?
Many thanks
Holger

  • Reply • Share ›

Desmond Shaw Mod Holger • 2 years ago

I'm not sure whether there's such an app. But you can ask some Android developers to make it:
1. the app should have the permissions of SMS, email and camera
2. once the SMS received, analyze the SMS content with your predefined keywords
3. once keywords matched, trigger your customized camera(Don't use camera intent) to take a p
4. send the picture via email (a remote upload server connected through socket is better)
5. That's all

  • Reply • Share ›

 Subscribe  Add Disqus to your site Privacy

Search …

Tutorials

Barcode Programming

DWT with Web Frameworks

WebSocket Programming

Android Programming

OpenCV Programming

Visual SourceSafe

Team Foundation Server

WCF & Java