1. OBJECTIVE

In this tutorial, we will introduce some programming techniques in android studio, which includes the following:

(1) Display of WebView on android studio;
(2) PIPPY motion control by android pad;
(3) Use of multi-interfaces to switch windows for advertisements;
(4) Background music on start-up;
(5) Rotating icons for decoration;

2. COMPONENTS

(1) Window PC;
(2) Pad;
(3) USB Type-C Cable;
(4) Smartphone (Personal Hotspot);
(5) PIPPY;

3. BASIC KNOWLEDGE FOR THE EXPERIMENT

Android Studio is the official integrated development environment (IDE) for Android application development. It is based on IntelliJ IDEA, a Java integrated development environment for software, and incorporates its code editing and developer tools.

To support application development within the Android operating system, Android Studio uses a Gradle-based build system, Android Emulator, code templates and GitHub integration. Every project in Android Studio has one or more modalities with source code and resource files. These modalities include Android app modules, Library modules and Google App Engine modules.

Android Studio uses an Apply Changes feature to push code and resource changes to a running application. A code editor assists the developer with writing code and offering code completion, refraction and analysis. Applications built in Android Studio are then compiled into the APK format for submission to the Google Play Store.

4. EXPERIMENTAL STEPS

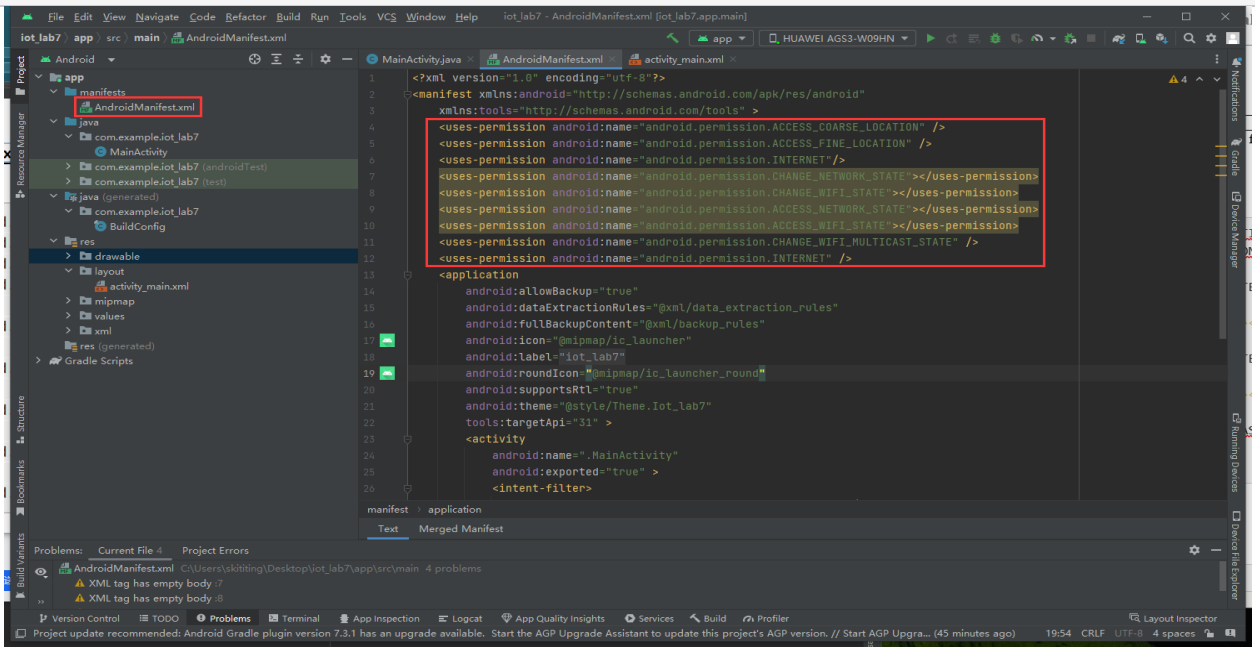
4.1. Camera video display

Notice:

- The followings are based on *5.3* of *Tutorial 1*, the URL of the PIPPY's camera video should be found first.

1) Create a new android project, add the following codes in **`AndroidManifest.xml`**.

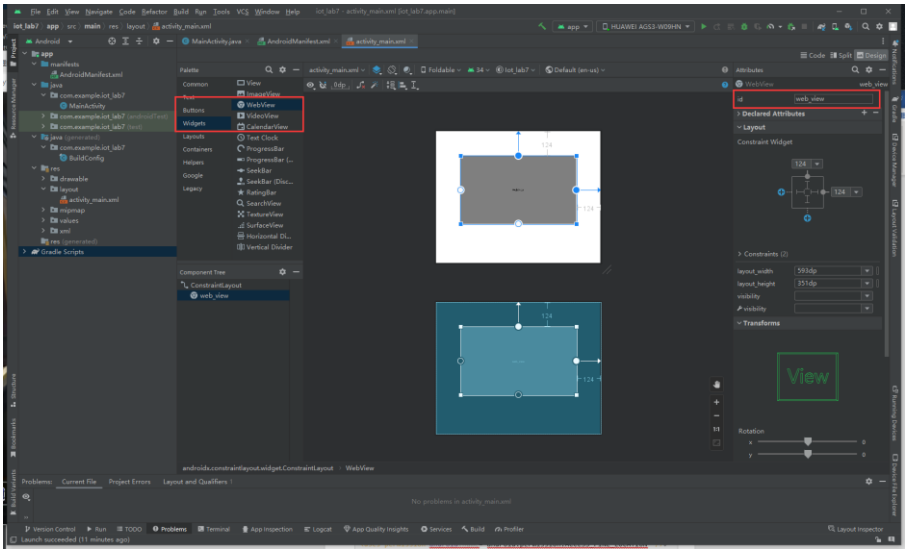
```
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.CHANGE_NETWORK_STATE"></uses-perm
ission>
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE"></uses-permiss
ion>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"></uses-perm
ission>
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"></uses-permiss
ion>
<uses-permission android:name="android.permission.CHANGE_WIFI_MULTICAST_STATE" />
```



2) Open **`MainActivity.java`** file and add the following packages.

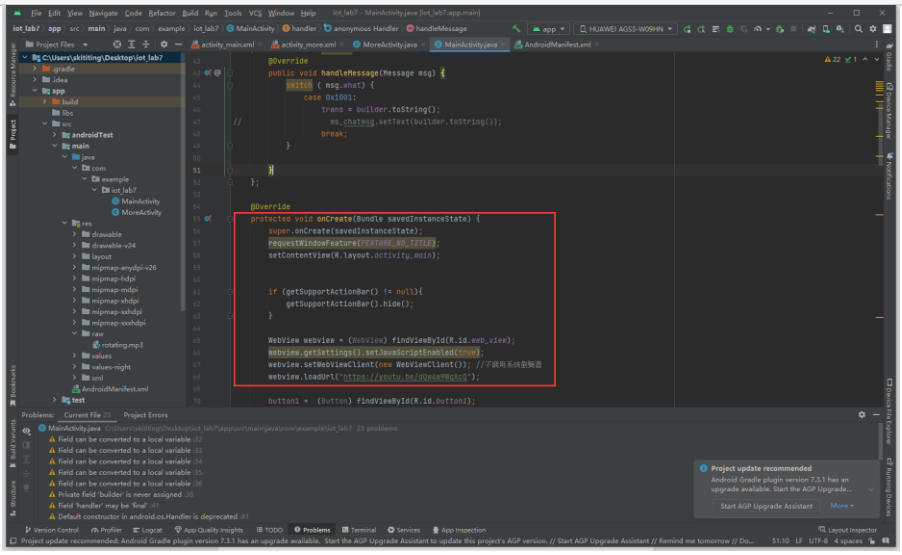
```
import android.webkit.WebViewClient;
import android.webkit.WebView;
import static android.view.Window.FEATURE_NO_TITLE;
```

3) Create a WebView widget in the layout xml file.

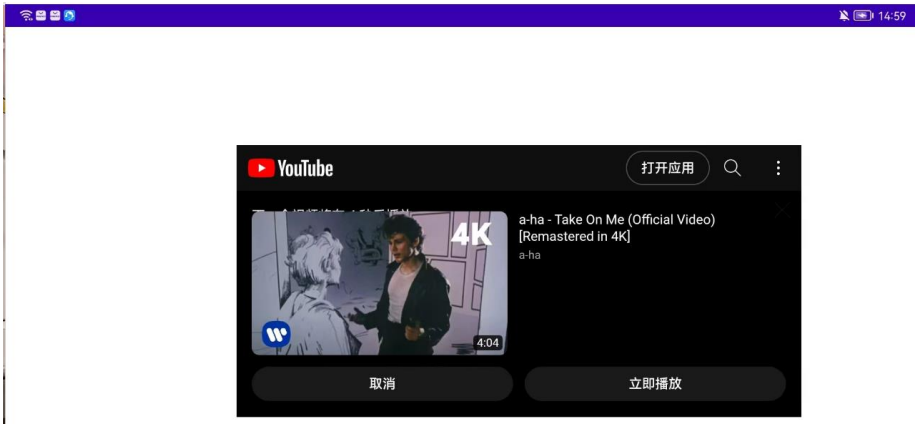


4) Write the following codes under the **`onCreate`** class.

```
requestWindowFeature(FEATURE_NO_TITLE);
setContentView(R.layout.activity_main);
if (getSupportActionBar() != null){
    getSupportActionBar().hide();}
WebView webview = (WebView) findViewById(R.id.web_view);
webview.getSettings().setJavaScriptEnabled(true);
webview.setWebViewClient(new WebViewClient()); //不調用系統瀏覽器
webview.loadUrl("https://youtu.be/dQw4w9WgXcQ");
```



5) Build the project and verify its effectiveness.



Notice:

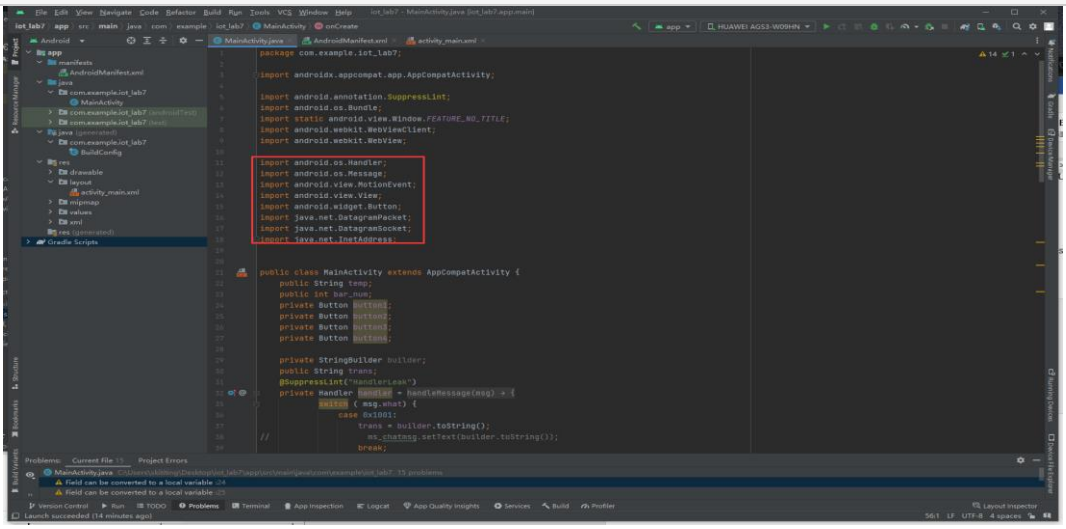
- You should first connect the Android Pad to the Internet.
- The URL in the code indicates the web that will appear. Replace it with the URL of the PIPPY's monitor video, you can stream the camera video (See *5.3* in *Tutorial 1*).

4.2. PIPPY Motion Control

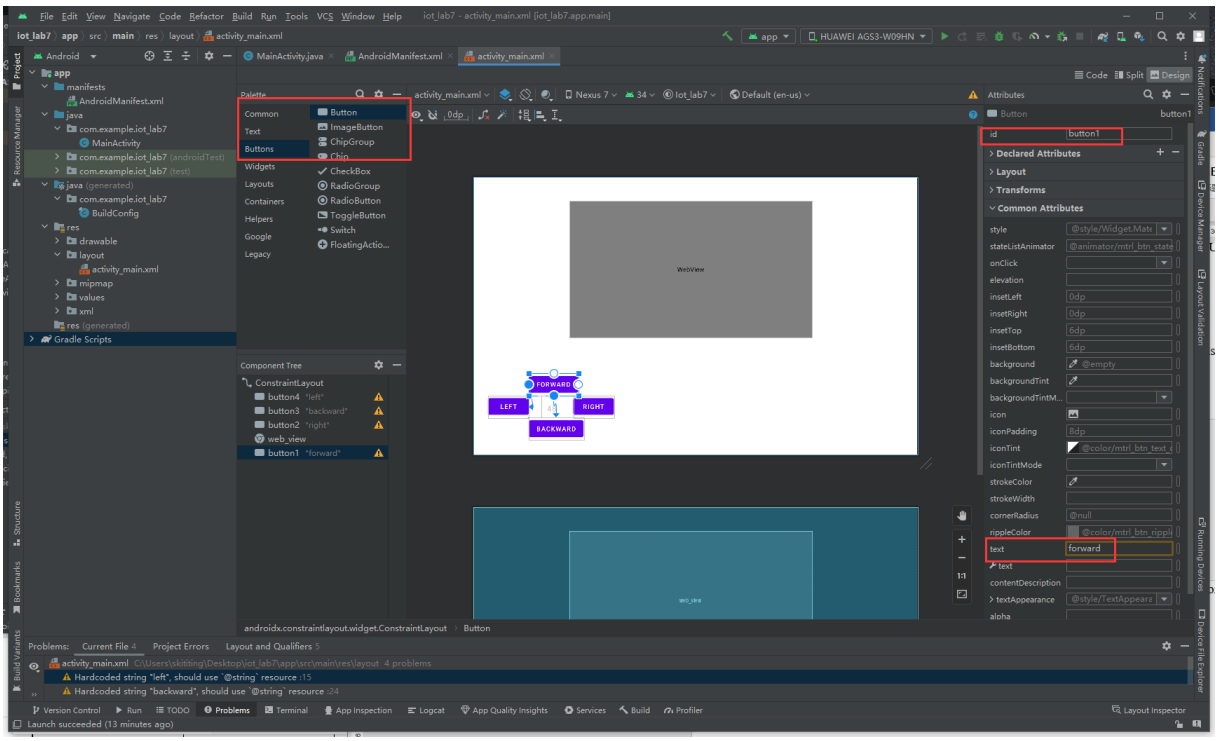
In *Tutorial 1,* we have found the key words that can be used to send the instructions to the PIPPY. The followings will introduce how to use buttons to send those commands.

1) Open **`MainActivity.java`** file and add the following packages.

```
import android.os.Handler;
import android.os.Message;
import android.view.MotionEvent;
import android.view.View;
import android.widget.Button;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
```

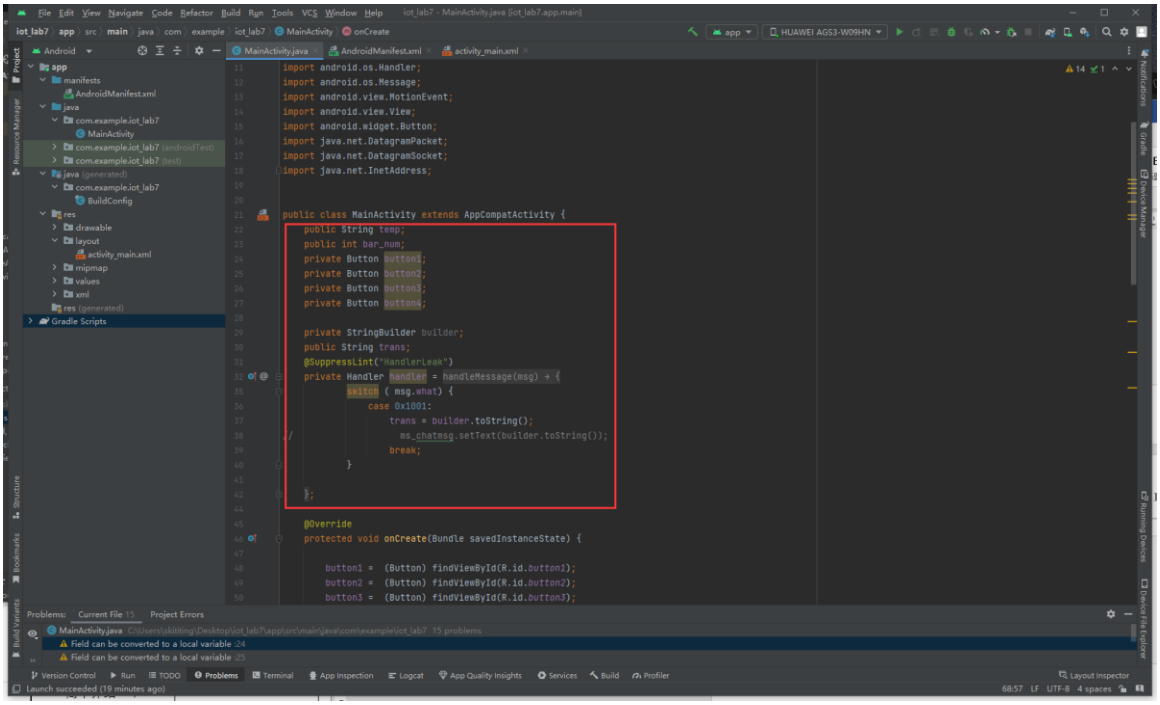


2) Create several buttons in the layout xml file, set their text to the appropriate commands to control the PIPPY.

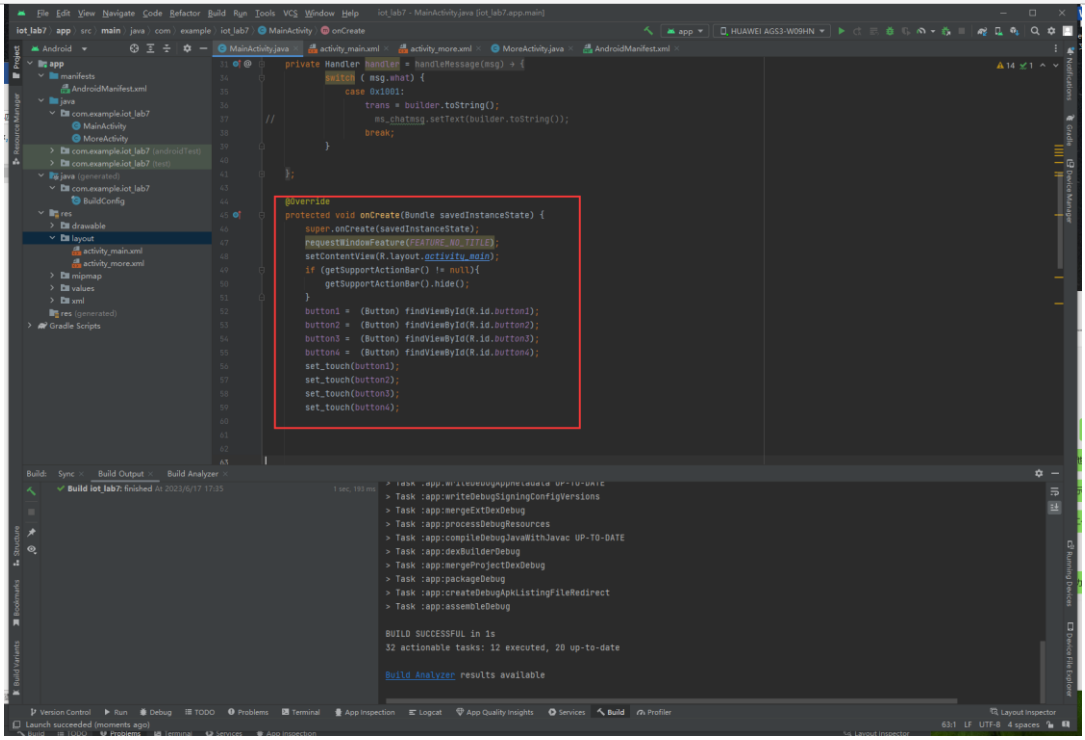


3) Add the following declarations and the initializations.

```
    public String temp;
    public int bar_num;
    private Button button1;
    private Button button2;
    private Button button3;
    private Button button4;
    private StringBuilder builder;
    public String trans;
    @SuppressLint("HandlerLeak")
    private Handler handler = new Handler() {
        @Override
        public void handleMessage(Message msg) {
            switch ( msg.what) {
                case 0x1001:
                    trans = builder.toString();
//                    ms_chatmsg.setText(builder.toString());
                    break;
            }
        }
    };
```



```
button1 =  (Button) findViewById(R.id.button1);
button2 =  (Button) findViewById(R.id.button2);
button3 =  (Button) findViewById(R.id.button3);
button4 =  (Button) findViewById(R.id.button4);
set_touch(button1);
set_touch(button2);
set_touch(button3);
set_touch(button4);
```



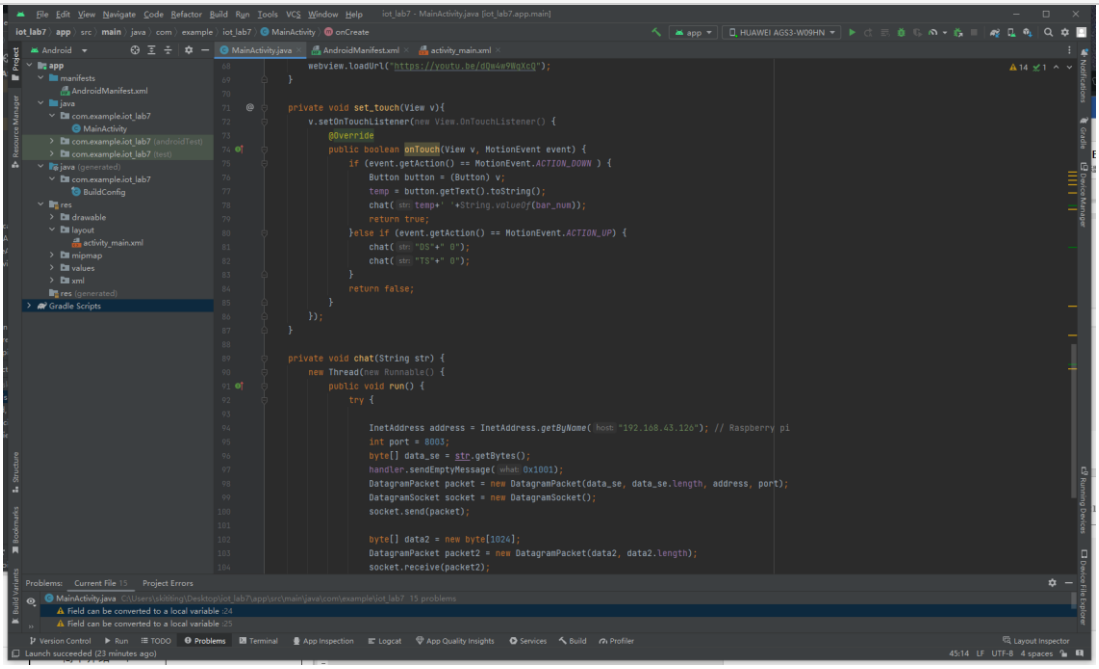
4) Write the following codes for **`chat`** function (Introduced in tutorials in *ECEN3024*):

```
private void chat(String str) {
    new Thread(new Runnable() {
        public void run() {
            try {

                InetAddress address = InetAddress.getByName("192.168.43.126
"); // Raspberry pi
                int port = 8003;
                byte[] data_se = str.getBytes();
                handler.sendEmptyMessage(0x1001);
                DatagramPacket packet = new DatagramPacket(data_se, data_se.
length, address, port);
                DatagramSocket socket = new DatagramSocket();
                socket.send(packet);

                byte[] data2 = new byte[1024];
                DatagramPacket packet2 = new DatagramPacket(data2, data2.len
gth);
                socket.receive(packet2);
                handler.sendEmptyMessage(0x1001);

                socket.close();
            } catch (Exception e )
            {
                e.printStackTrace();
            }
        }
    }).start();
}
```

5) Write the following codes for **`set_touch`** function:

```
private void set_touch(View v){
    v.setOnTouchListener(new View.OnTouchListener() {
        @Override
        public boolean onTouch(View v, MotionEvent event) {
            if (event.getAction() == MotionEvent.ACTION_DOWN ) {
                Button button = (Button) v;
                temp = button.getText().toString();
                chat(temp+' '+String.valueOf(bar_num));
                return true;
```

```
            }else if (event.getAction() == MotionEvent.ACTION_UP) {
                chat("DS"+" 0");
                chat("TS"+" 0");
            }
            return false;
        }
    });
}
```

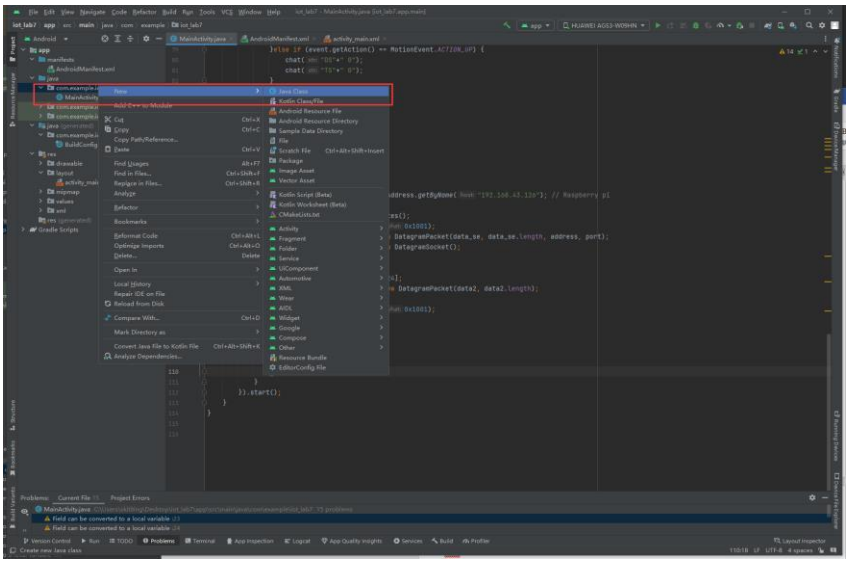


Notice:

- **`set_touch`** will send the text of the buttons in the form of string to the Raspi via **`chat`** function, if you want to send other short instructions, you only need to change the text of the button.
- **`set_touch`** will continuously send the corresponding instructions when the button is being pressed. When released, it will immediately send the **`DS`** and **`TS`** to stop the PIPPY entirely.
- To verify this function, you should run the server on Raspi and monitor the corresponding port.

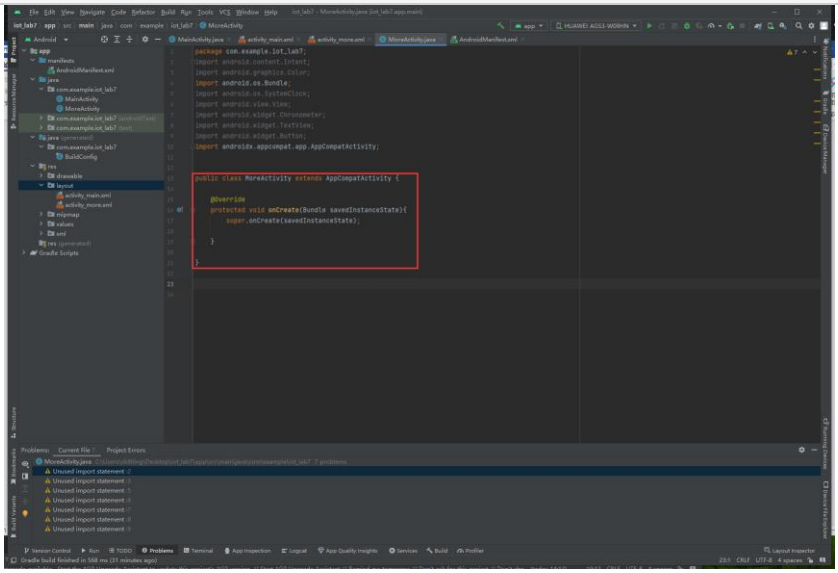
4.3. Window switching

1) Create a new java file named `MoreActivity.java`, and this activity will serve as the start-up advertisement.

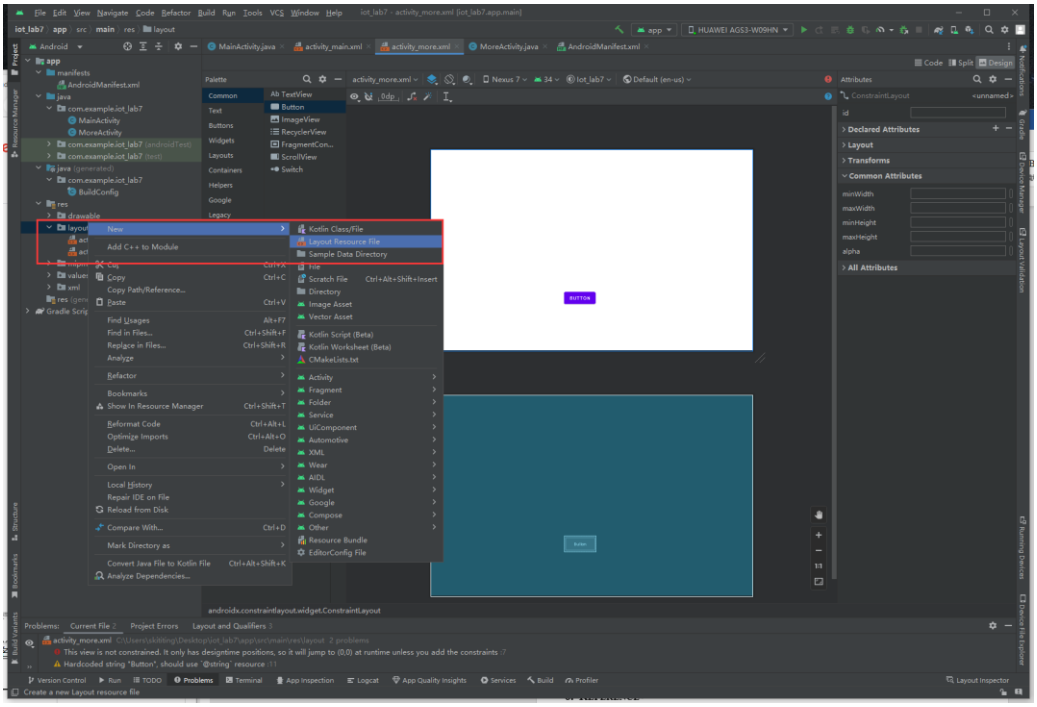


2) Import the followings to `MoreActivity.java`. Also change the class `MoreActivity` into `extends AppcompatActivity` and add the `onCreate` function.

```
import android.content.Intent;
import android.os.Bundle;
import android.os.SystemClock;
import android.view.View;
import android.widget.Chronometer;
import android.widget.TextView;
import android.widget.Button;
import androidx.appcompat.app.AppCompatActivity;
```

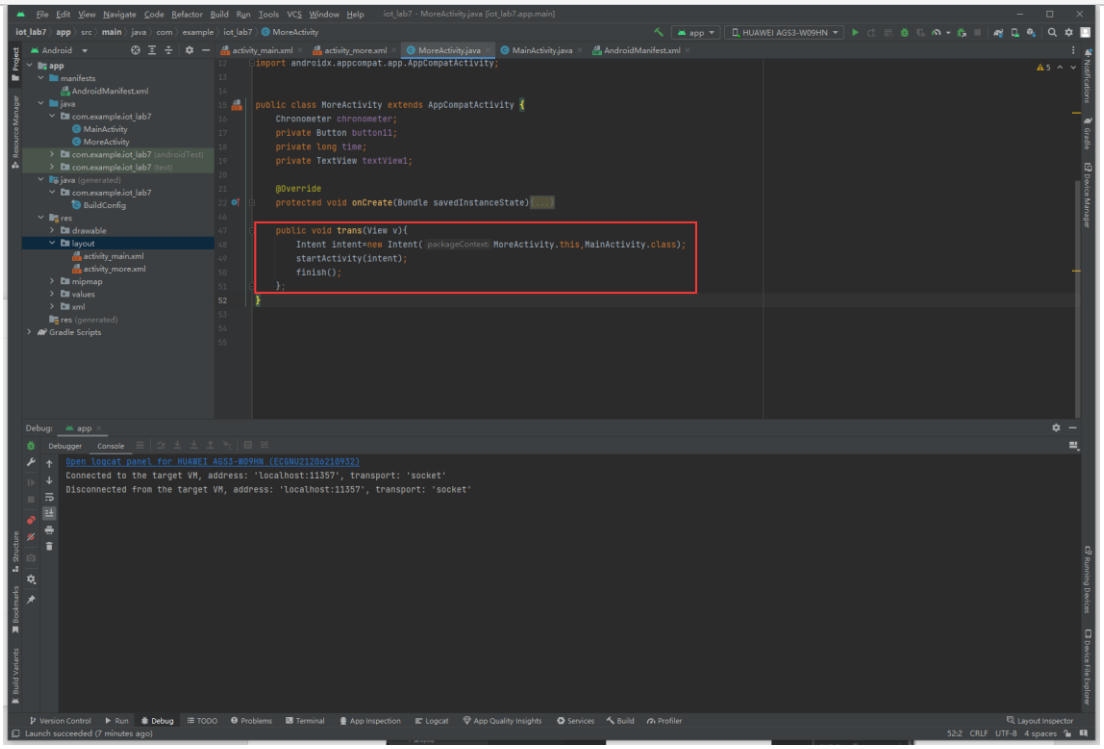


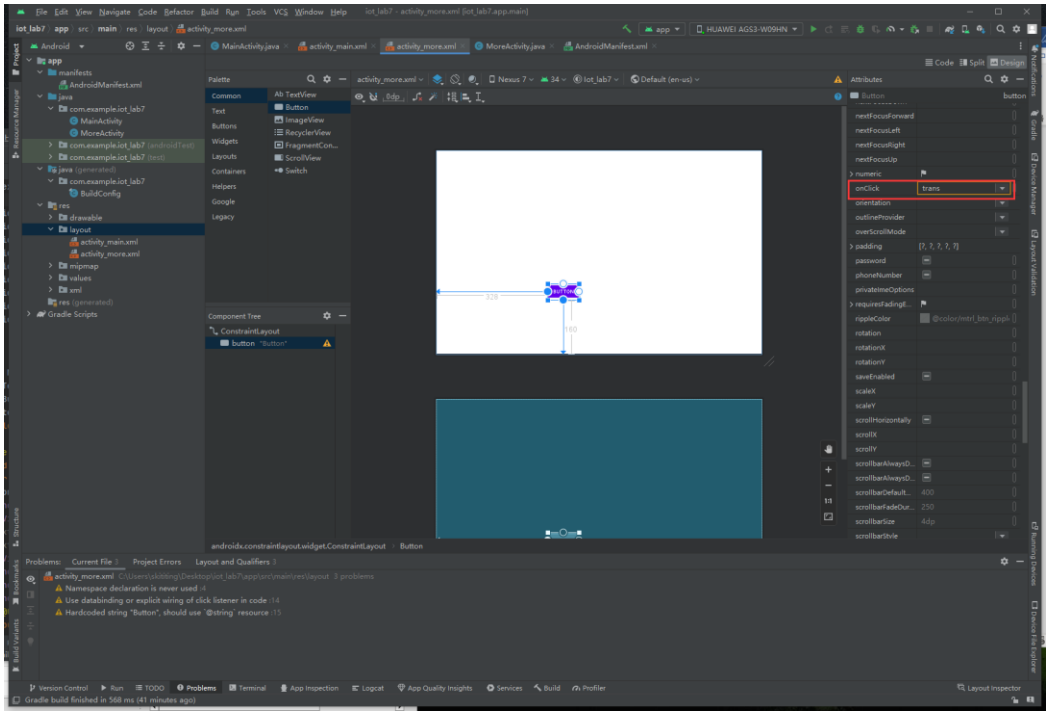
3) Create a new layout xml file named **`activity_more.xml`**.



4) Create a button and write the following code for **`trans`** function and set it to be the onclick function of the button.

```
public void trans(View v){
    Intent intent=new Intent(MoreActivity.this,MainActivity.class);
    startActivity(intent);
    finish();};
```

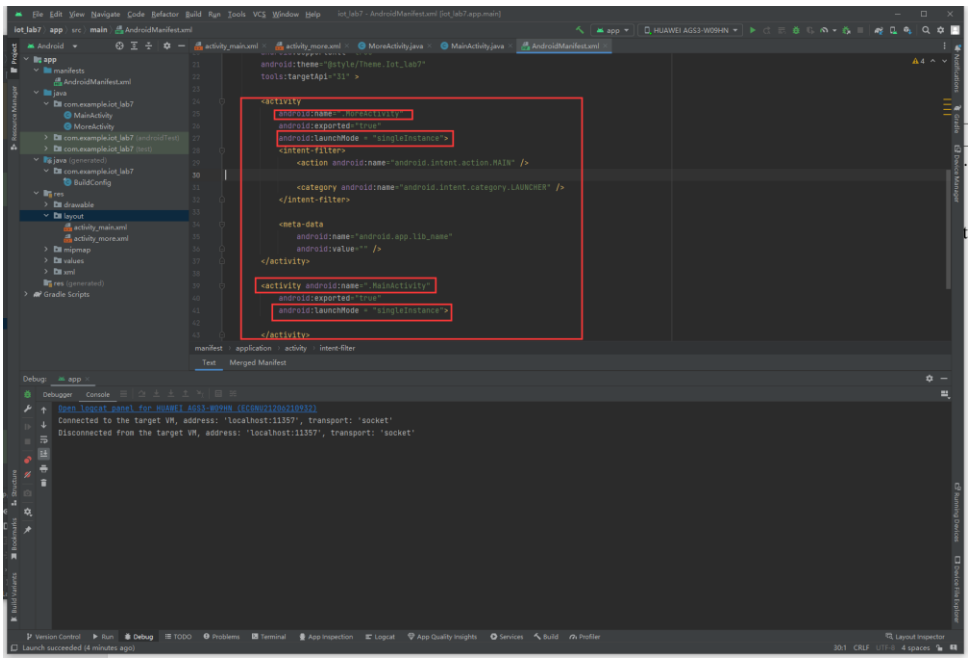




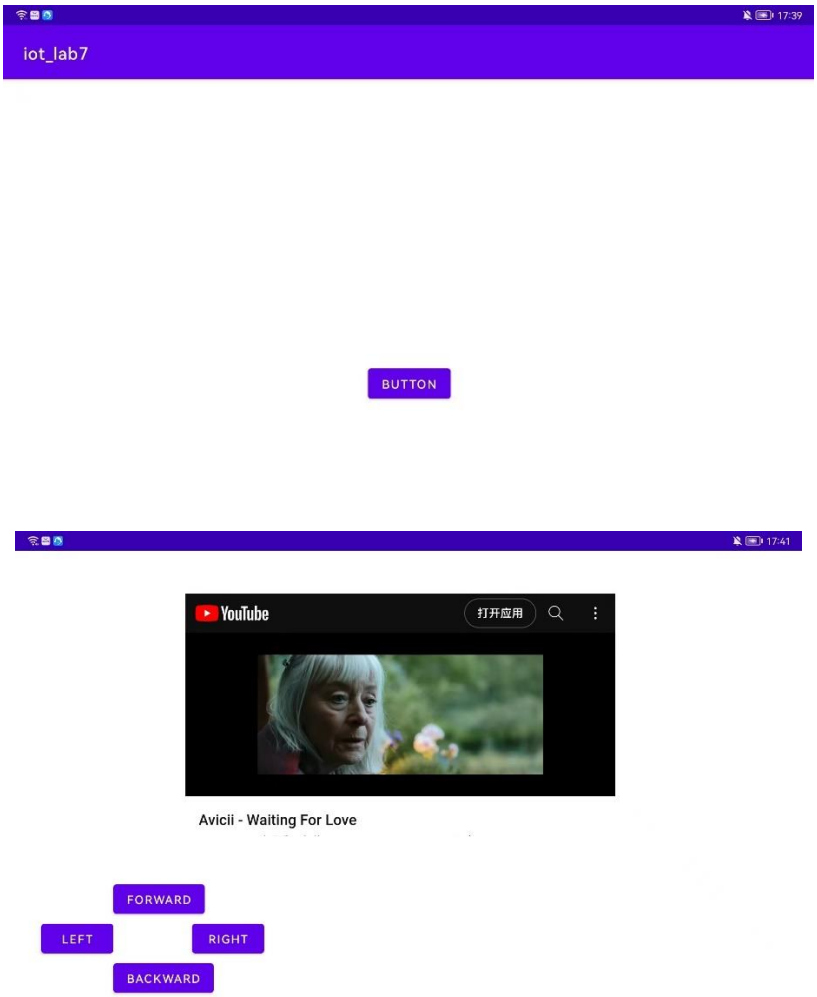
Notice:

- `Intent` switches from `MoreActivity` to `MainActivity` according to its argument sequence.

5) Add the description for `MoreActivity` in `AndroidManifest.xml`, change the start-up priority.

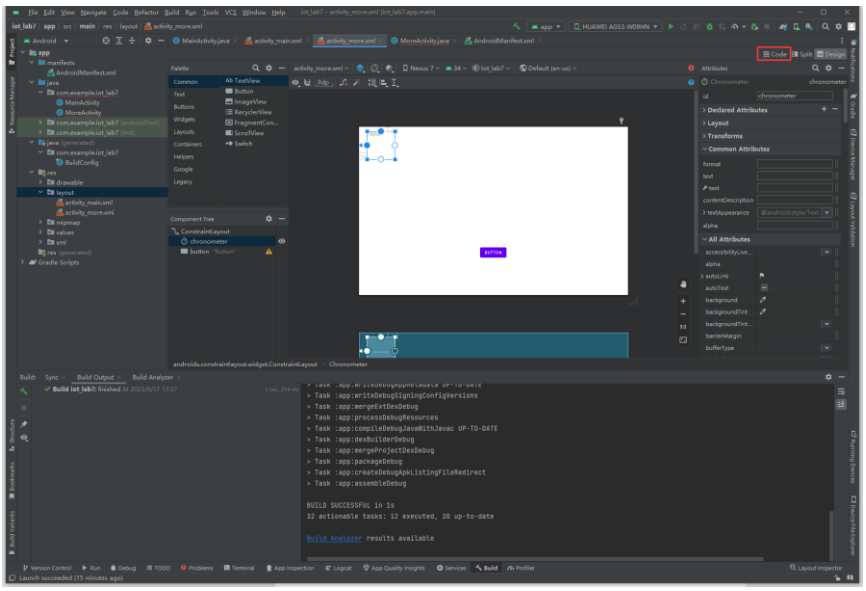


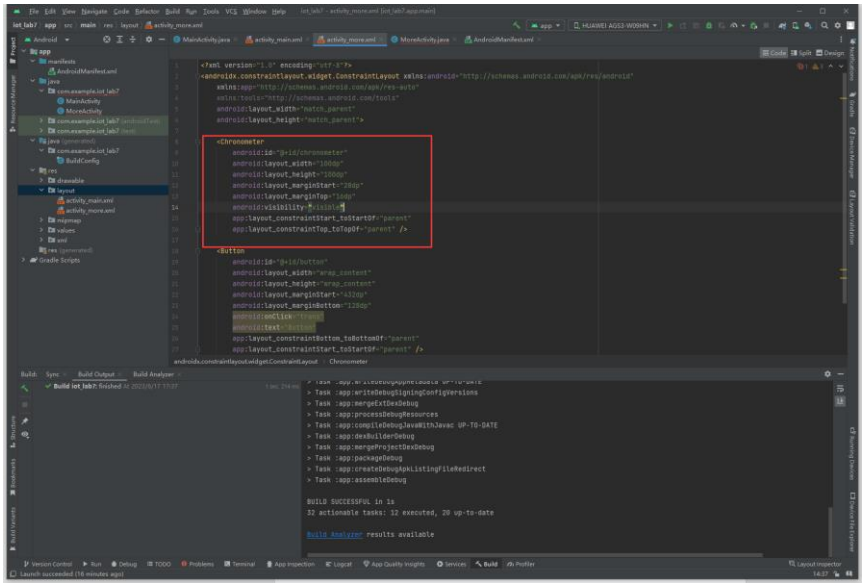
6) Check the validity of the codes.



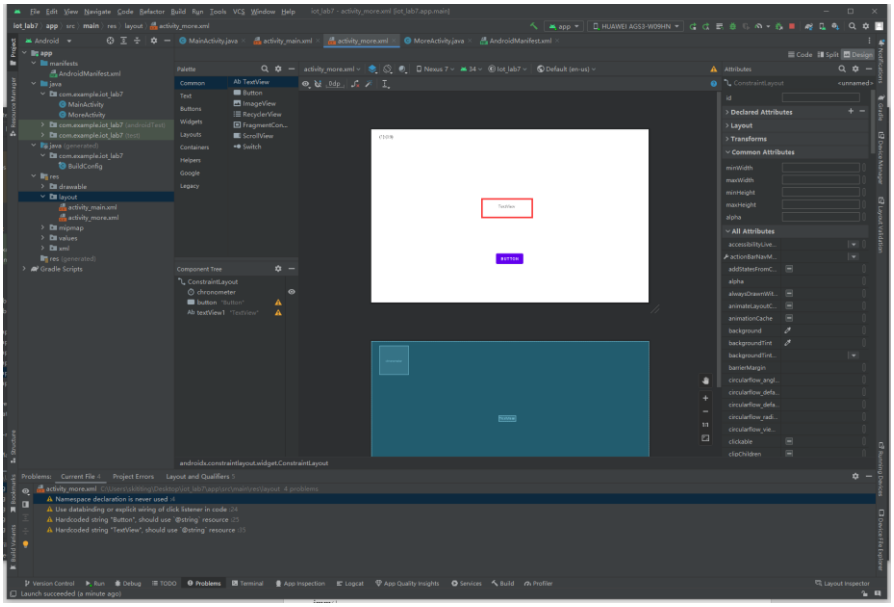
7) To realize auto-switch, `chronometer` can be utilized as a timer. First, add the declaration of the widget in the xml file.

```
<Chronometer
    android:id="@+id/chronometer"
    android:layout_width="100dp"
    android:layout_height="100dp"
    android:layout_marginStart="28dp"
    android:layout_marginTop="16dp"
    android:visibility="visible"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

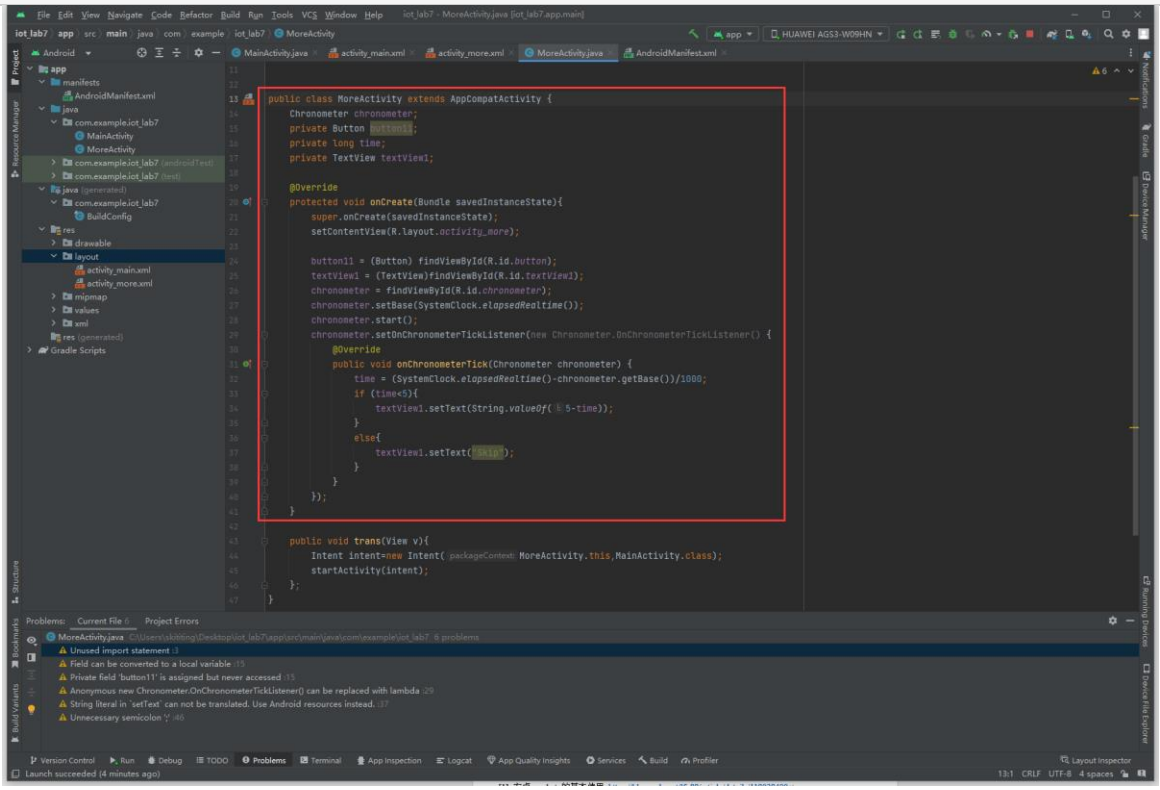




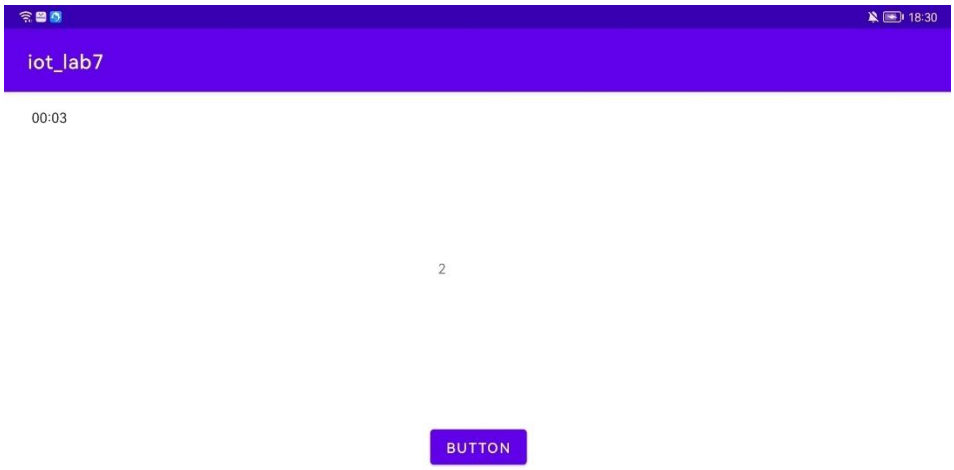
8) Add a **`textView1`**, write the following codes.

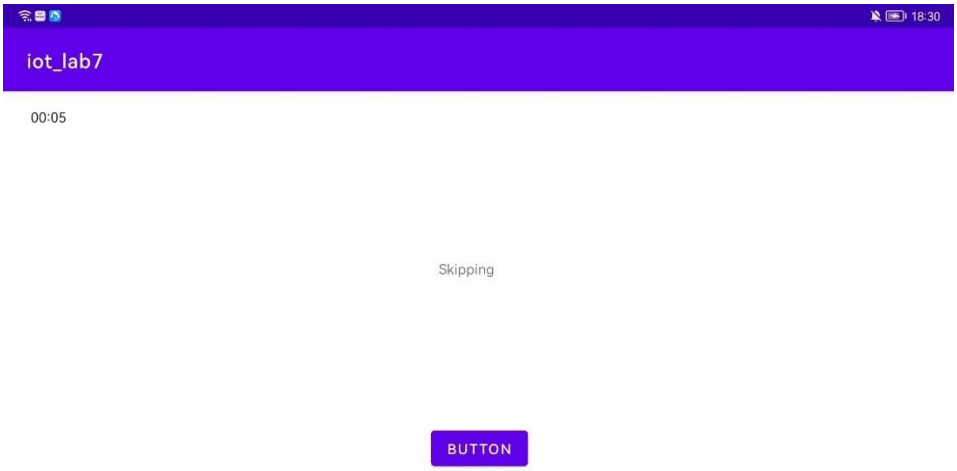


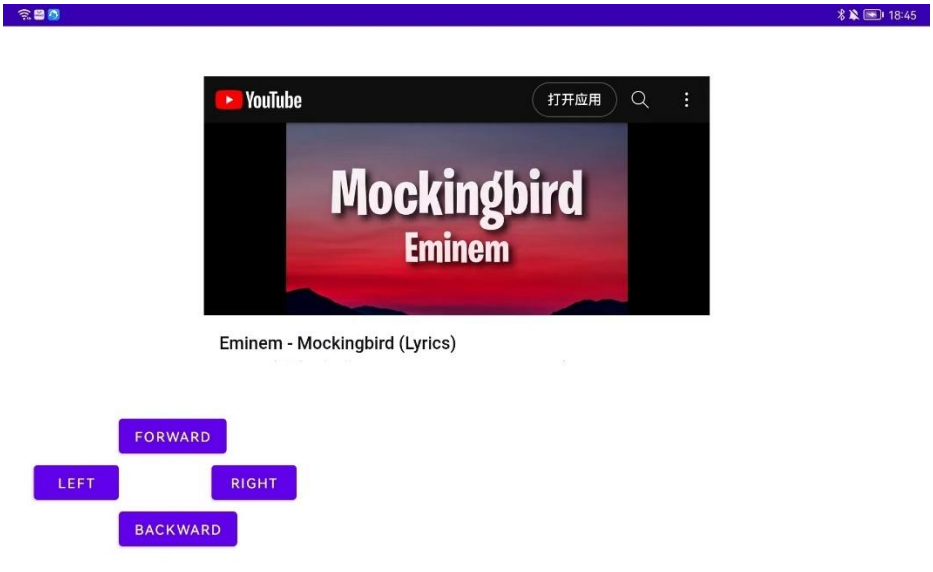
```
Chronometer chronometer;
private Button button11;
private long time;
private TextView textView1;
@Override
protected void onCreate(Bundle savedInstanceState){
    super.onCreate(savedInstanceState);
    requestWindowFeature(FEATURE_NO_TITLE);
    setContentView(R.layout.activity_more);
    button11 = (Button) findViewById(R.id.button);
    textView1 = (TextView)findViewById(R.id.textView1);
    chronometer = findViewById(R.id.chronometer);
    chronometer.setBase(SystemClock.elapsedRealtime());
    chronometer.start();
    chronometer.setOnChronometerTickListener(new Chronometer.OnChronometerTi
ckListener() {
        @Override
        public void onChronometerTick(Chronometer chronometer) {
          time = (SystemClock.elapsedRealtime()-chronometer.getBase())/1000;
            if (time<5){
                textView1.setText(String.valueOf(5-time));}
            else{
                textView1.setText("Skipping");
                button11.callOnClick();}}});}
```



9) Followings are the overall effect of the codes.







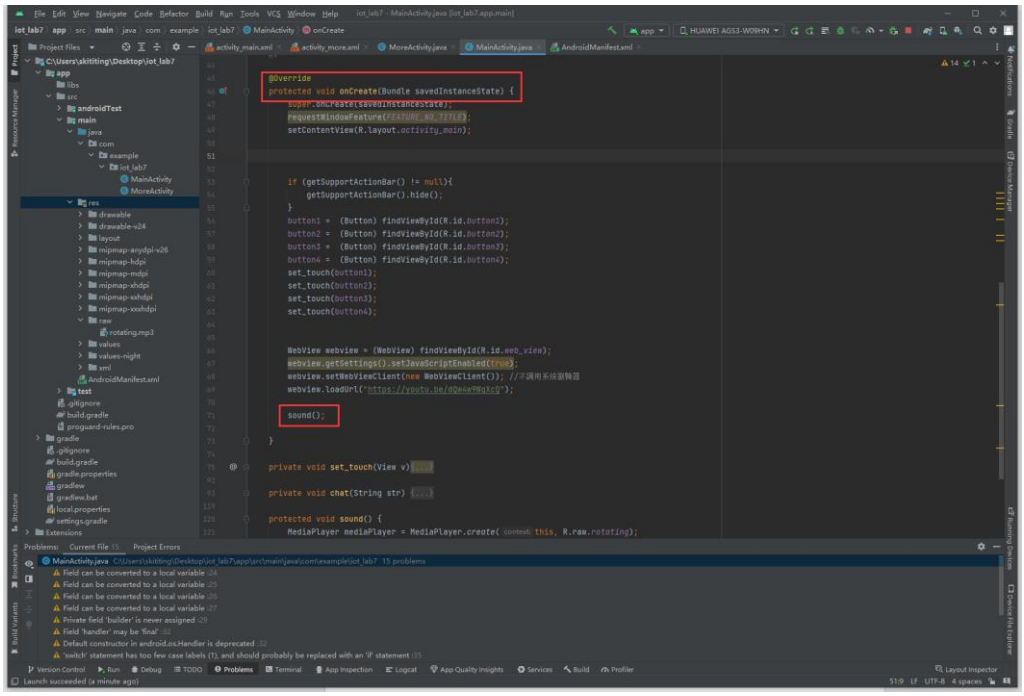
4.4. Background Music

1) Download the music in `mp3` format, create a directory in the project path named "raw" as follows.

2) Write the following code for `sound` function.

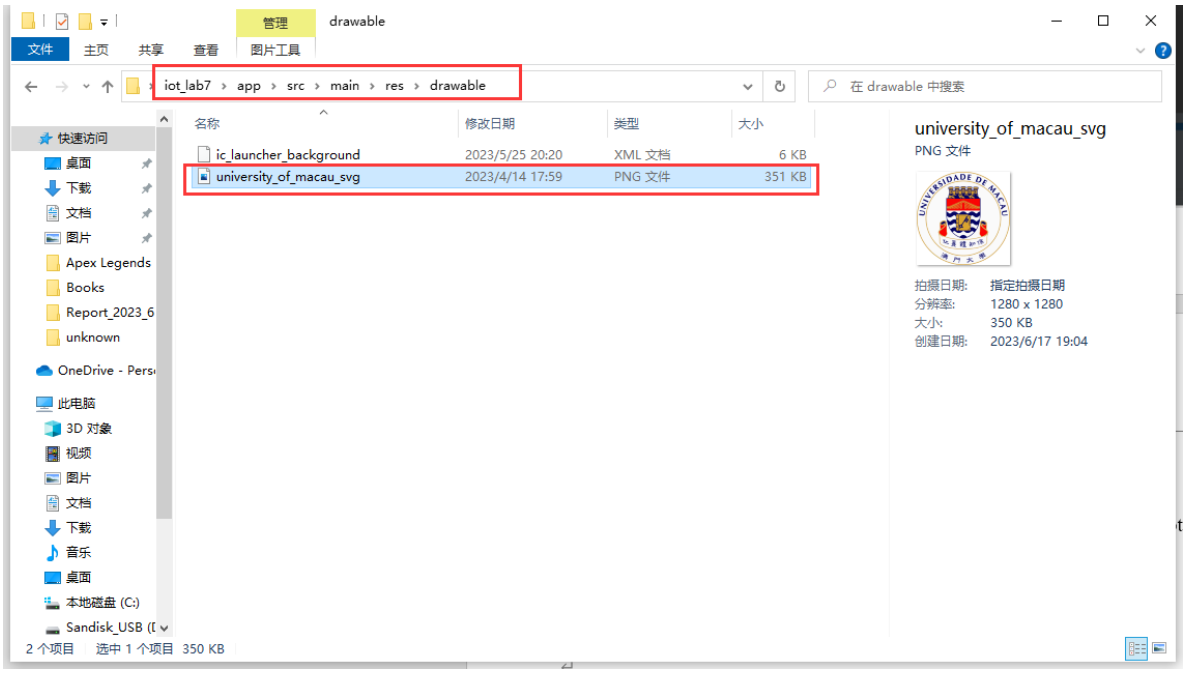
```
protected void sound() {
    MediaPlayer mediaPlayer = MediaPlayer.create(this, R.raw.rotating);
    mediaPlayer.start();
}
```

3) Place it under the `onCreate` function.



4.5. Rotating Icon

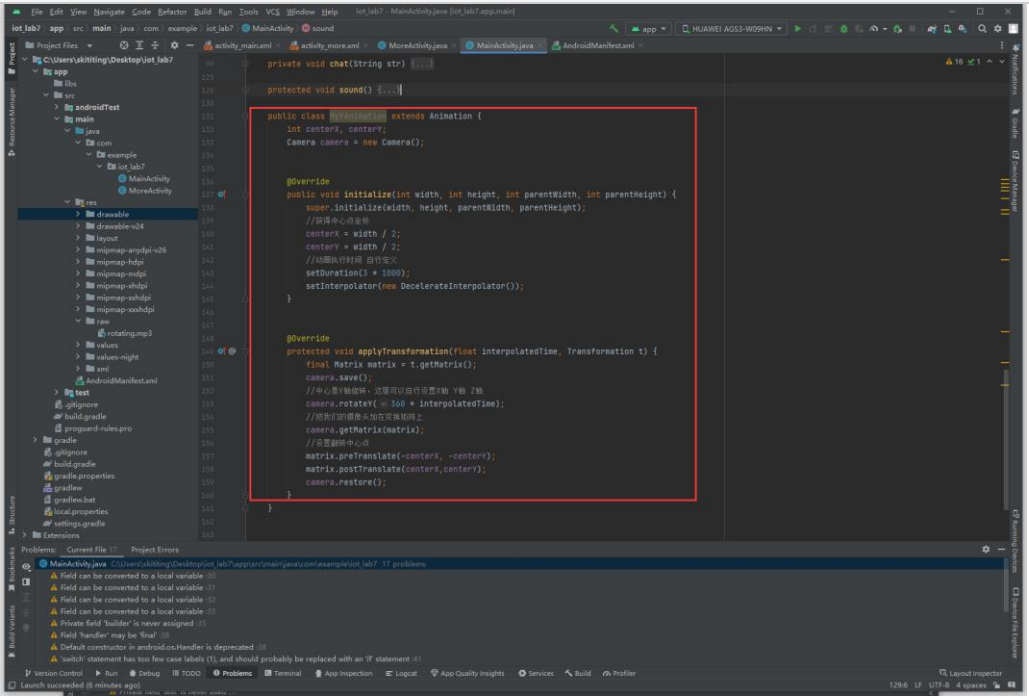
1) Download the icon in **png** format (other formats may applicable but not tested), place it under the "drawable" directory.



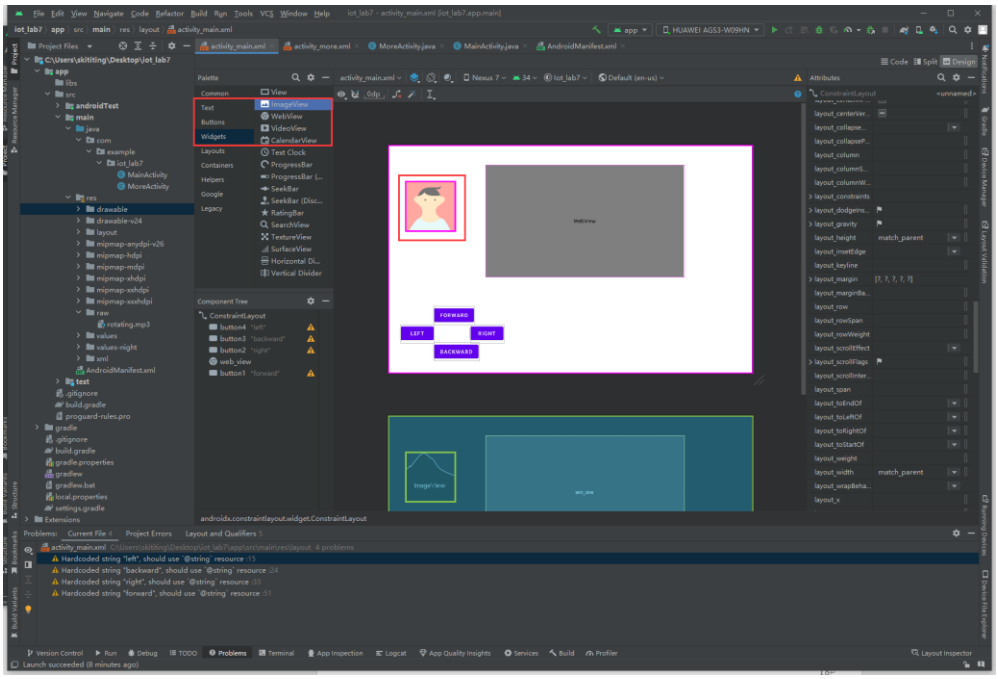
2) Write the following class.

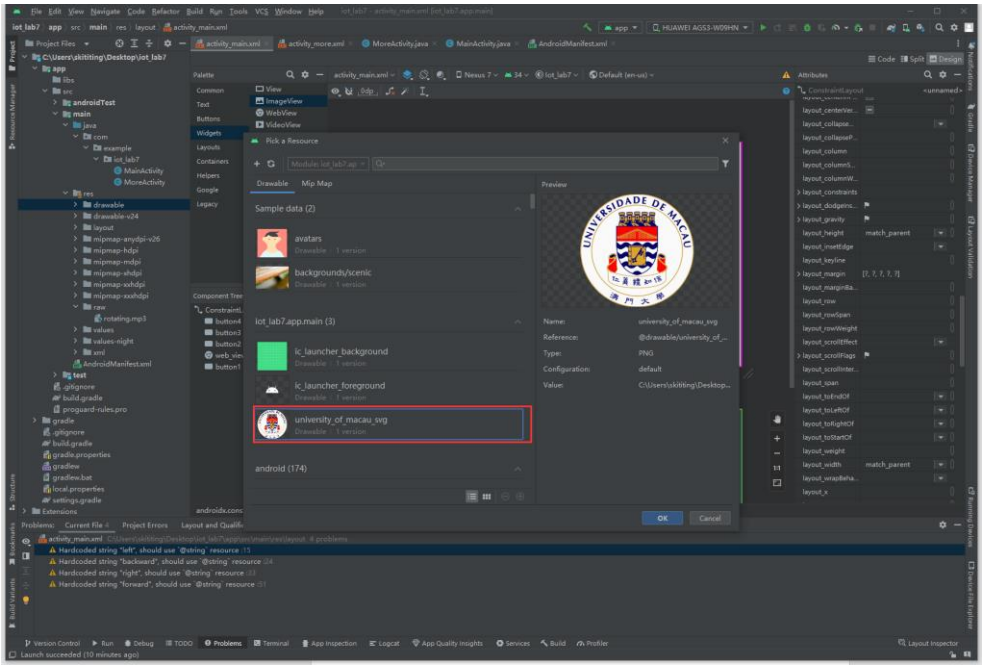
```
public class MyYAnimation extends Animation {
    int centerX, centerY;
    Camera camera = new Camera();
    @Override
    public void initialize(int width, int height, int parentWidth, int paren
tHeight) {
        super.initialize(width, height, parentWidth, parentHeight);
        //获得中心点坐标
        centerX = width / 2;
        centerY = width / 2;
        //动画执行时间 自行定义
        setDuration(3 * 1000);
        setInterpolator(new DecelerateInterpolator());
    }
    @Override
    protected void applyTransformation(float interpolatedTime, Transformatio
n t) {
        final Matrix matrix = t.getMatrix();
        camera.save();
        //中心是Y轴旋转，这里可以自行设置X轴 Y轴 Z轴
```

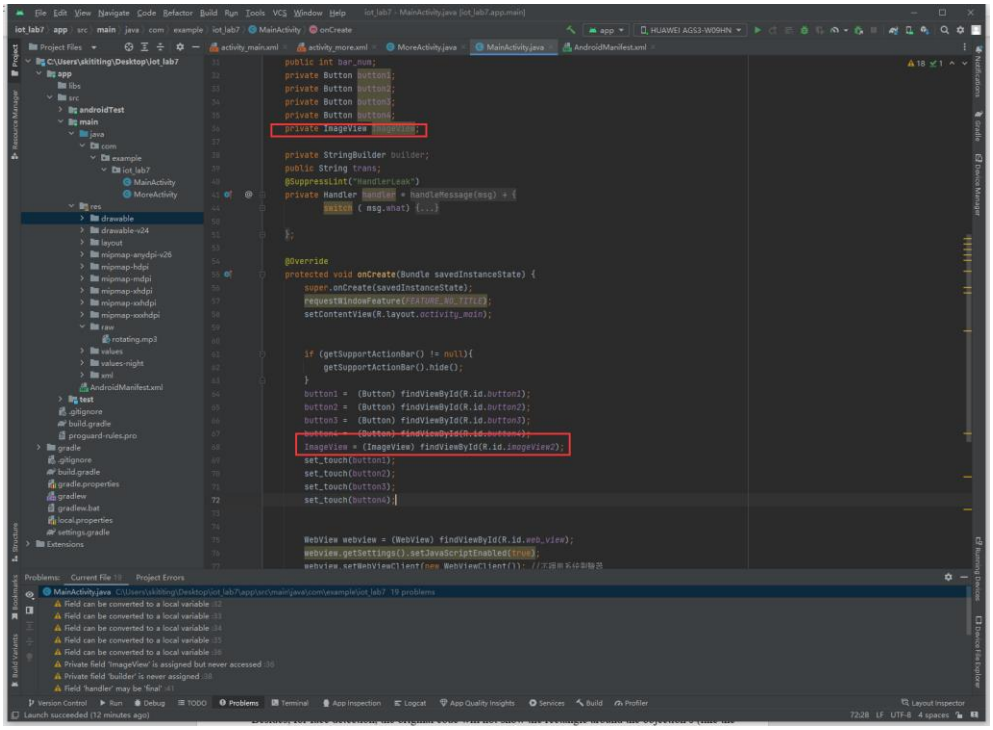
```
            camera.rotateY(360 * interpolatedTime);
            //把我们的摄像头加在变换矩阵上
            camera.getMatrix(matrix);
            //设置翻转中心点
            matrix.preTranslate(-centerX, -centerY);
            matrix.postTranslate(centerX,centerY);
            camera.restore();
        }
}
```



3) Create an **`imageView`** widget.

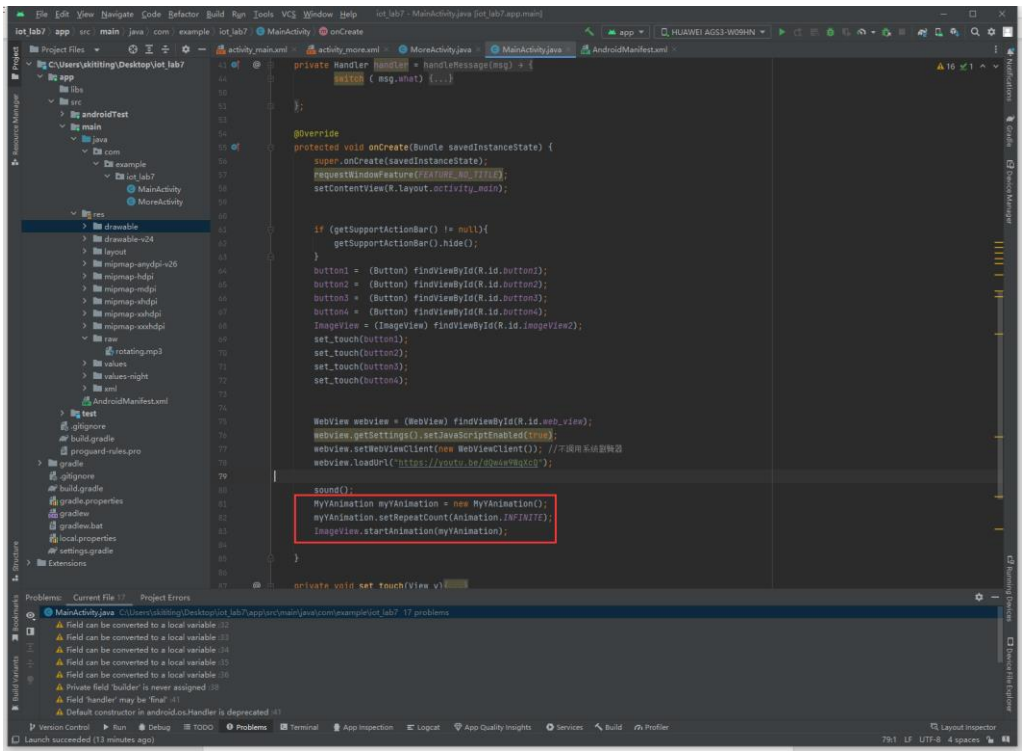






4) Place the below codes in **`onCreate`** .

```
MyYAnimation myYAnimation = new MyYAnimation();
myYAnimation.setRepeatCount(Animation.INFINITE);
ImageView.startAnimation(myYAnimation);
```



5) Followings are the overall effect of the codes.

5. POTENTIAL DIRECTION

5.1. Make the PIPPY generate different state's notification.

While changing the modes of the PIPPY (e.g., PIPPY is doing motion changes or face detection), there is no display to tell the user what is PIPPY's current state. It is recommended that a notification be given so that the user can better control the device.

To achieve this function, you can try to make the button change color when you press it or use **`TextView`** to make the pad show you PIPPY's current state (e.g., QR detection or face tracking).

5.2. Use rectangles to show the contour of face during face detection.

For face detection, the codes in the tutorials do not show the rectangle around the object (as shown in *Tutorial 3*), you can write codes to make the rectangles appear on the screen. The valid original codes are shown below from **`camera_opencv.py`**, just follow its definition and try to add your own functions in it.

```
95      def elementDraw(self,imgInput):
96          if self.CVMode == 'none':
97              pass
98
99          elif self.CVMode == 'findColor':
100             if self.findColorDetection:
101                 cv2.putText(imgInput,'Target Detected',(40,60), CVThread.font, 0.5,(255,255,255),1,cv2.LINE_AA)
102                 self.drawing = 1
103             else:
104                 cv2.putText(imgInput,'Target Detecting',(40,60), CVThread.font, 0.5,(255,255,255),1,cv2.LINE_AA)
105                 self.drawing = 0
106
107             if self.radius > 10 and self.drawing:
108                 cv2.rectangle(imgInput,(int(self.box_x-self.radius),int(self.box_y+self.radius)),(int(self.box_x+self.radius),int(self.box_y-self.radius)),(255,255,255),1)
109
110         elif self.CVMode == 'findlineCV':
111             if frameRender:
112                 imgInput = cv2.cvtColor(imgInput, cv2.COLOR_BGR2GRAY)
113                 retval_bw, imgInput =  cv2.threshold(imgInput, 0, 255, cv2.THRESH_OTSU)
114                 imgInput = cv2.erode(imgInput, None, iterations=6)
115             try:
116                 if lineColorSet == 255:
117                     cv2.putText(imgInput,('Following White Line'),(30,50), cv2.FONT_HERSHEY_SIMPLEX, 0.5,(255,255,255),1,cv2.LINE_AA)
118                     cv2.putText(imgInput,('Following White Line'),(230,50), cv2.FONT_HERSHEY_SIMPLEX, 0.5,(0,0,0),1,cv2.LINE_AA)
119                 else:
120                     cv2.putText(imgInput,('Following Black Line'),(30,50), cv2.FONT_HERSHEY_SIMPLEX, 0.5,(255,255,255),1,cv2.LINE_AA)
121                     cv2.putText(imgInput,('Following Black Line'),(230,50), cv2.FONT_HERSHEY_SIMPLEX, 0.5,(0,0,0),1,cv2.LINE_AA)
122
123                 cv2.putText(imgInput,(self.CVCommand),(30,90), cv2.FONT_HERSHEY_SIMPLEX, 0.5,(255,255,255),1,cv2.LINE_AA)
124                 cv2.putText(imgInput,(self.CVCommand),(230,90), cv2.FONT_HERSHEY_SIMPLEX, 0.5,(0,0,0),1,cv2.LINE_AA)
```

6. REFERENCE

[1] 安卓 socket 的基本使用, https://blog.csdn.net/lfq88/article/details/118028400
[2] Android Studio 中文社区, http://forum.android-studio.org/forum.php
[3] Java 教程, https://www.runoob.com/java/java-tutorial.html

END OF TUTORIAL