APPENDIX

THE CODES FOR ALL THE APPLICATIONS USED IN THIS PROJECT ARE GIVEN BELOW

# ANDROID LIVE STREAMING APP CODE

Manifest File

<?xml version="1.0" encoding="utf-8"?>

<manifest xmlns:android="http://schemas.android.com/apk/res/android"

package="com.project.livestreamapp"

android:versionCode="1"

android:versionName="1.0" >

<uses-sdk

android:minSdkVersion="8"

android:targetSdkVersion="19" />

<uses-feature

android:name="android.hardware.camera"

android:required="true" />

*<!..*

*This is done to alert the user that the app will use the camera.*

*-->*

<uses-feature

android:name="android.hardware.camera.autofocus"

android:required="false" />

<supports-screens

android:largeScreens="true"

android:normalScreens="true"

android:smallScreens="true"

android:xlargeScreens="true" />

<uses-permission android:name="android.permission.INTERNET" />

<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />

<uses-permission android:name="android.permission.RECORD\_AUDIO" />

<uses-permission android:name="android.permission.CAMERA" />

<application

android:allowBackup="true"

android:icon="@drawable/ic\_launcher"

android:label="@string/app\_name"

android:theme="@style/AppTheme" >

<activity

android:name=".MainActivity"

android:label="@string/app\_name" >

<intent-filter>

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />

</intent-filter>

</activity>

</application>

</manifest>

**LAYOUT: activity\_main.xml**

<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:id="@+id/surface\_layout"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:layout\_alignParentLeft="true"

android:layout\_alignParentTop="true"

android:background="@android:color/black" >

<net.majorkernelpanic.streaming.gl.SurfaceView

android:id="@+id/surface"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_gravity="center" />

</FrameLayout>

**MAIN ACTIVITY CODE**

package com.project.livestreamapp;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

import android.app.Activity;

import android.app.AlertDialog;

import android.content.DialogInterface;

import android.os.Bundle;

import android.view.Menu;

import android.view.MenuItem;

import android.view.SurfaceHolder;

import android.view.Window;

import android.view.WindowManager;

import net.majorkernelpanic.streaming.Session;

import net.majorkernelpanic.streaming.SessionBuilder;

import net.majorkernelpanic.streaming.audio.AudioQuality;

import net.majorkernelpanic.streaming.gl.SurfaceView;

import net.majorkernelpanic.streaming.rtsp.RtspClient;

public class MainActivity extends Activity implements RtspClient.Callback, Session.Callback, SurfaceHolder.Callback {

public final static String TAG = MainActivity.class.getSimpleName();

public static final String STREAM\_URL = "rtsp://11c434.entrypoint.cloud.wowza.com:1935/app-b65e/15126c34"; //***THIS IS THE SERVER //ADDRESS FOR OUR STREAM. THE FORMAT IS*** *rtsp://WOWZASERVERADDRESS:PORT/WORKSPACE/STREAM\_NAME*

private static SurfaceView mSurfaceView;

private Session mSession;

private static RtspClient mClient;

@Override

protected void onCreate(Bundle savedInstanceState) { **// *THIS IS THE FIRST CALLBACK THAT OCCURS WHEN USER OPENS OUR APP***

super.onCreate(savedInstanceState);

getWindow().addFlags(WindowManager.LayoutParams.FLAG\_KEEP\_SCREEN\_ON); **//*TO ENSURE THAT SCREEN REMAINS ON***

getWindow().addFlags(WindowManager.LayoutParams.FLAG\_FULLSCREEN);

requestWindowFeature(Window.FEATURE\_NO\_TITLE);

setContentView(R.layout.activity\_main);**//*LINKS THE VIEW OF OUR APP TO OUR activity\_main.xml FILE WE HAVE ALREADY CODED***

mSurfaceView = (SurfaceView) findViewById(R.id.surface); **//*LINKS TO OUR SURFACE VIEW***

mSurfaceView.getHolder().addCallback(this);

initRtspClient(); ***// Initialize RTSP client***

}

@Override

protected void onResume() {

super.onResume();

}

@Override

public void onDestroy(){ **//*CALLED WHEN USER QUITS APPLICATION***

super.onDestroy();

mClient.release();

mSession.release();

mSurfaceView.getHolder().removeCallback(this);

}

@Override

public void surfaceChanged(SurfaceHolder arg0, int arg1, int arg2, int arg3) {

}

@Override

public void surfaceCreated(SurfaceHolder arg0) {

}

@Override

public void surfaceDestroyed(SurfaceHolder arg0) {

}

public void onBitrareUpdate(long bitrate) {

}

@Override

public void onSessionError(int reason, int streamType, Exception e) { **// *CALLED WHEN THERE IS AN ERROR IN THE APPLICATION***

switch (reason) {

case Session.ERROR\_CAMERA\_ALREADY\_IN\_USE:

break;

case Session.ERROR\_CAMERA\_HAS\_NO\_FLASH:

break;

case Session.ERROR\_INVALID\_SURFACE:

break;

case Session.ERROR\_STORAGE\_NOT\_READY:

break;

case Session.ERROR\_CONFIGURATION\_NOT\_SUPPORTED:

break;

case Session.ERROR\_OTHER:

break;

}

if (e != null) {

alertError(e.getMessage());

e.printStackTrace(); **//*THIS WAS USED FOR DEBUGGING THE APPLICATION. THIS IS OPTIONAL.***

}

}

@Override

public void onPreviewStarted() { }

@Override

public void onSessionConfigured() { }

@Override

public void onSessionStarted() { }

@Override

public void onSessionStopped() { }

@Override

public void onRtspUpdate(int message, Exception exception) {

switch (message) {

case RtspClient.ERROR\_CONNECTION\_FAILED:

case RtspClient.ERROR\_WRONG\_CREDENTIALS:

alertError(exception.getMessage()); ***//TO PROVIDE NECESSARY ERROR MESSAGES TO USER***

exception.printStackTrace();

break;

}

}

private void alertError(final String msg) { ***//THIS FUNCTION DISPLAYS ERROR MESSAGE msg TO USER***

final String error = (msg == null) ? "Unknown error: " : msg;

AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);

builder.setMessage(error).setPositiveButton("Ok",

new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog, int id) {

}

});

AlertDialog dialog = builder.create();

dialog.show();

}

private void initRtspClient() {

mSession = SessionBuilder.getInstance() **//THIS LINE CREATES THE RTSP SESSION**

.setContext(getApplicationContext())

.setAudioEncoder(SessionBuilder.AUDIO\_AAC)

.setAudioQuality(new AudioQuality(8000, 16000)) **//*SETS AUDIO TYPE AND QUALITY***

.setVideoEncoder(SessionBuilder.VIDEO\_H264) **//*SETS VIDEO TYPE AND QUALITY***

.setSurfaceView(mSurfaceView)

.setPreviewOrientation(0)

.setCallback(this).build();

mClient = new RtspClient();

mClient.setSession(mSession);

mClient.setCallback(this);

mSurfaceView.setAspectRatioMode(SurfaceView.ASPECT\_RATIO\_PREVIEW);

String ip, port, path; **// THE FOLLOWING CODE SNIPPETS DERIVES THE IP, PORT AND PATH FROM OUR STREAM\_URL**

Pattern uri = Pattern.compile("rtsp://(.+):(\\d\*)/(.+)");

Matcher m = uri.matcher(STREAM\_URL);

m.find();

ip = m.group(1); **// IP =** 11c434.entrypoint.cloud.wowza.com

port = m.group(2); **// PORT =** 1935

path = m.group(3); **//path =** app-b65e/15126c34

// **SINCE WE HAVE DISABLED OUR AUTHENTICATION FROM THE SERVER WE DO NOT NEED TO PROVIDE USERNAME OR PASSWORD. HAD //WE NOT DONE THAT, WE WOULD HAVE TO ADD THIS LINE IN OUR CODE:**

**//**mClient.setCredentials(WowzaParams.PUBLISHER\_USERNAME,WowzaParams.PUBLISHER\_PASSWORD);

mClient.setServerAddress(ip, Integer.parseInt(port));

mClient.setStreamPath("/" + path);

mSession.startPreview(); **// Starts showing user the camera footage**

mClient.startStream(); **// Starts Streaming**

}

}

## PROJECT APP FOR CONTROLLING ROBOT THROUGH MQTT PUBLISHING AND VIEWING LIVE STREAM FROM SERVER

**MANIFEST FILE**

<?xml version="1.0" encoding="utf-8"?>

<manifest xmlns:android="http://schemas.android.com/apk/res/android"

package="com.project.mqttexamples"

android:versionCode="1"

android:versionName="1.0" >

<uses-sdk

android:minSdkVersion="8"

android:targetSdkVersion="21" />

<uses-permission android:name="android.permission.WAKE\_LOCK" />

<uses-permission android:name="android.permission.INTERNET" />

<uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE" />

<uses-permission android:name="android.permission.READ\_PHONE\_STATE" />

<application

android:allowBackup="true"

android:icon="@drawable/ic\_launcher"

android:label="@string/app\_name"

android:theme="@style/AppTheme" >

*<!--*

*We must declare the Vitamio main activity (InitActivity) in our applications tab to allow our app access to the Vitamio classes*

*-->*

<activity

android:name="io.vov.vitamio.activity.InitActivity"

android:configChanges="orientation|screenSize|smallestScreenSize|keyboard|keyboardHidden|navigation"

android:launchMode="singleTop"

android:theme="@android:style/Theme.NoTitleBar"

android:windowSoftInputMode="stateAlwaysHidden" />

<activity

android:name=".MainActivity"

android:label="@string/app\_name" >

<intent-filter>

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />

</intent-filter>

</activity>

*<!..*

*Since MqttService falls under a android service category, it must be specified in the manifest*

*-->*

<service android:name="org.eclipse.paho.android.service.MqttService" >

</service>

</application>

</manifest>

**LAYOUT FILE: activity\_main.xml**

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical"

android:background="#000"

>

<io.vov.vitamio.widget.VideoView

android:id="@+id/vitamio\_videoView"

android:layout\_width="match\_parent"

android:layout\_height="0dp"

android:layout\_weight="0.7"

/>

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="vertical"

android:gravity="center"

>"

<Button

android:id="@+id/buttonForward"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:background="@drawable/up" />

<RelativeLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="horizontal"

>

<Button

android:id="@+id/buttonRight"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignParentRight="true"

android:layout\_alignParentTop="true"

android:background="@drawable/right"

/>

<Button

android:id="@+id/buttonLeft"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignParentLeft="true"

android:layout\_alignParentTop="true"

android:background="@drawable/left"

/>

</RelativeLayout>

<Button

android:id="@+id/buttonBack"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:background="@drawable/down" />

</LinearLayout>

</LinearLayout>

**MAIN ACTIVITY CODE**

package com.project.mqttexamples;

import io.vov.vitamio.LibsChecker; import io.vov.vitamio.MediaPlayer; import io.vov.vitamio.widget.MediaController;

import io.vov.vitamio.widget.VideoView; import java.io.UnsupportedEncodingException;

import org.eclipse.paho.android.service.MqttAndroidClient; import org.eclipse.paho.client.mqttv3.IMqttActionListener;

import org.eclipse.paho.client.mqttv3.IMqttToken; import org.eclipse.paho.client.mqttv3.MqttClient;

import org.eclipse.paho.client.mqttv3.MqttConnectOptions; import org.eclipse.paho.client.mqttv3.MqttException;

import org.eclipse.paho.client.mqttv3.MqttMessage;

import android.app.Activity; import android.os.Bundle; import android.util.Log;

import android.view.MotionEvent;import android.view.View;

import android.view.View.OnClickListener; import android.view.View.OnTouchListener;

import android.widget.Button; import android.widget.Toast;

public class MainActivity extends Activity implements OnTouchListener {

private String mqttServerAddress="tcp://m12.cloudmqtt.com:12879"; **//*MQTT SERVER IP***

private String mqttUserName="pphdjxet"; **//*MQTT USERNAME***

private String mqttPassword="\_ur1NqE1w5gq"; **//*MQTT PASSWORD***

private String path = "rtmp://11c434.entrypoint.cloud.wowza.com:1935/app-b65e/15126c34"; **// *ADDRESS OF OUR SERVER***

VideoView mVideoView; **//*THIS WILL CONTAIN THE INSTANCE OF OUR VIDEO PLAYER***

Button fwd,lft,ryt,bck; **//*THESE WILL KEEP THE INSTANCES OF THE FOUR BUTTONS***

MqttAndroidClient client;

MqttConnectOptions options;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

if (!LibsChecker.checkVitamioLibs(this))

return;

setContentView(R.layout.activity\_main); **// *LINK THE VIEW OF THIS ACTIVITY TO THE XML FILE WE HAVE CREATED***

mVideoView = (VideoView) findViewById(R.id.vitamio\_videoView);

***// FIRST WE WORK OUT THE LIVE STREAMING PART***

mVideoView.setVideoPath(path); **// *WE HAVE SET THE VIDEO PATH TO OUR WOWZA SERVER***

mVideoView.setMediaController(new MediaController(this)); **//*ENABLES MEDIA CONTROLS LIKE PLAY PAUSE AND TIMELINE***

mVideoView.requestFocus();

mVideoView.setOnPreparedListener(new MediaPlayer.OnPreparedListener() {

@Override

public void onPrepared(MediaPlayer mediaPlayer) {

mediaPlayer.setPlaybackSpeed(1.0f);

}

});

**// *THE MQTT PART***

String clientId = MqttClient.generateClientId();

client =new MqttAndroidClient(this.getApplicationContext(), mqttServerAddress, clientId);

**//*Here we create a clientID and initialize it with the parameters of our server address***

options = new MqttConnectOptions();

options.setMqttVersion(MqttConnectOptions.MQTT\_VERSION\_3\_1); **//Since we are using mqttv3**

options.setUserName(mqttUserName); **//Specifying username and password**

options.setPassword(.toCharArray());

**//Linking the Button objects to the respective Buttons and make them responsive to touch**

fwd=(Button)findViewById(R.id.buttonForward);

bck=(Button)findViewById(R.id.buttonBack);

lft=(Button)findViewById(R.id.buttonLeft);

ryt=(Button)findViewById(R.id.buttonRight);

fwd.setOnTouchListener(this);

bck.setOnTouchListener(this);

lft.setOnTouchListener(this);

ryt.setOnTouchListener(this);

}

public void sendMessage(final String message){ **//This function will publish the String message to the topic "test"**

IMqttToken token = null;

try {

token = client.connect(options); **// First, we connect to the MQTT Server**

} catch (MqttException e1) {

// TODO Auto-generated catch block

e1.printStackTrace();

}

token.setActionCallback(new IMqttActionListener() {

@Override

public void onSuccess(IMqttToken asyncActionToken) {

String topic = "test"; **//TOPIC NAME**

**// TO PUBLISH THE PAYLOAD OR MESSAGE, WE NEED TO CONVERT IT INTO A BYTE ARRAY FIRST**

byte[] encodedPayload = new byte[0];

try {

encodedPayload = message.getBytes("UTF-8");

MqttMessage message = new MqttMessage(encodedPayload); **// CONVERTING TO MQTT Message Instance**

client.publish(topic, message); **// PUBLISHING MESSAGE TO TOPIC "TEST"**

} catch (UnsupportedEncodingException | MqttException e) {

e.printStackTrace();

}

}

@Override

public void onFailure(IMqttToken asyncActionToken, Throwable exception) {

**// Something went wrong e.g. connection timeout or firewall problems**

Log.d(TAG, "onFailure");

Toast.makeText(getApplicationContext(), "A Problem Occured", Toast.LENGTH\_SHORT).show();

}

});

}

@Override

public boolean onTouch(View pressed, MotionEvent event) { **// CALLED WHEN A BUTTON IS TOUCHED**

if (event.getAction() == MotionEvent.ACTION\_DOWN) {

if (pressed==fwd){

sendMessage("w"); **// MQTT COMMAND TO GO FORWARD**

}

else if(pressed==lft){

sendMessage("a"); **// MQTT COMMAND TO GO LEFT**

}

else if(pressed==bck){

sendMessage("s"); **// MQTT COMMAND TO GO BACK**

}

else if(pressed==ryt){

sendMessage("d"); **// MQTT COMMAND TO GO RIGHT**

}

} else if (event.getAction() == MotionEvent.ACTION\_UP) {

**// ACTION\_UP occurs when user releases a button**

sendMessage("h"); **// MQTT COMMAND TO STOP**

}

return false;

}

}

**NODE MCU SCRIPT.LUA**

clien1="clitttt"

wifi.setmode(wifi.STATION)

wifi.sta.config("WIFI NAME","WIFI PASSWORD")

gpio.mode (2, gpio.OUTPUT) -- ia1 - GPIO4

gpio.mode (3, gpio.OUTPUT) -- ia2 - GPIO0

gpio.mode (4, gpio.OUTPUT) -- ib2 - GPIO2

gpio.mode (5, gpio.OUTPUT) -- ib1 - GPIO5

gpio.write(2, gpio.LOW)

gpio.write(3, gpio.LOW)

gpio.write(4, gpio.LOW)

gpio.write(5, gpio.LOW)

tmr.delay(1000)

m = mqtt.Client(clien1, 120, "pphdjxet","\_ur1NqE1w5gq")

m:on("connect", function(con) print ("connected") end)

m:on("offline", function(con) print ("offline") end)

m:connect("m12.cloudmqtt.com",12879, 0, function(conn)

print("connected")

m:subscribe("test", 0, function(conn)

print("subscribed")

end)

end)

m:on("message", function(conn, topic, data)

print(topic .. ":" )

if data == "w" then

gpio.write(2, gpio.HIGH)

gpio.write(3, gpio.LOW)

gpio.write(4, gpio.HIGH)

gpio.write(5, gpio.LOW)

tmr.delay(1000)

end)

end

if data == "h" then

gpio.write(2, gpio.LOW)

gpio.write(3, gpio.LOW)

gpio.write(4, gpio.LOW)

gpio.write(5, gpio.LOW)

tmr.delay(1000)

m:publish("topic","h",0,0,function(conn)

print("sent")

end)

end

if data == "s" then

gpio.write(2, gpio.LOW)

gpio.write(3, gpio.HIGH)

gpio.write(4, gpio.LOW)

gpio.write(5, gpio.HIGH)

tmr.delay(1000)

m:publish("topic","s",0,0,function(conn)

print("sent")

end)

end

if data == "a" then

gpio.write(2, gpio.LOW)

gpio.write(3, gpio.HIGH)

gpio.write(4, gpio.HIGH)

gpio.write(5, gpio.LOW)

tmr.delay(1000)

end

if data == "d" then

gpio.write(2, gpio.HIGH)

gpio.write(3, gpio.LOW)

gpio.write(4, gpio.LOW)

gpio.write(5, gpio.HIGH)

tmr.delay(1000)

end)

end

end)