Código de fuente: Aplicación SAPLM

Contenidos:

Archivo - Application.java

```java
package com.example.SAPLM;

import android.app.Activity;
import androidx.lifecycle.LifecycleObserver;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.content.Context;
import android.content.IntentFilter;
import android.os.Bundle;
import android.widget.Toast;

import com.example.SAPLM.bluetoothActivities.BluetoothConnection;
import com.example.SAPLM.settingsActivities.DeveloperConsoleActivity;

import java.io.IOException;

public class Application extends android.app.Application implements LifecycleObserver {

    @Override
    public void onCreate() {
        super.onCreate();

        // ProcessLifecycleOwner.get().getLifecycle().addObserver(this);

        IntentFilter bluetoothAdapterActionChangedFilter = new IntentFilter(BluetoothAdapter.ACTION_STATE_CHANGED);
        registerReceiver(BluetoothConnection.bluetoothAdapterStatusBR, bluetoothAdapterActionChangedFilter);

        IntentFilter bluetoothScanModeStatusChangedFilter = new IntentFilter();
        bluetoothScanModeStatusChangedFilter.addAction(BluetoothAdapter.ACTION_DISCOVERY_STARTED);
        bluetoothScanModeStatusChangedFilter.addAction(BluetoothAdapter.ACTION_DISCOVERY_FINISHED);
        bluetoothScanModeStatusChangedFilter.addAction(BluetoothAdapter.ACTION_SCAN_MODE_CHANGED);
        registerReceiver(BluetoothConnection.bluetoothScanModeStatusBR, bluetoothScanModeStatusChangedFilter);

        IntentFilter bluetoothDeviceStatusChangedFilter = new IntentFilter();
        bluetoothDeviceStatusChangedFilter.addAction(BluetoothDevice.ACTION_ACL_CONNECTED);
        bluetoothDeviceStatusChangedFilter.addAction(BluetoothDevice.ACTION_ACL_DISCONNECTED);
        bluetoothDeviceStatusChangedFilter.addAction(BluetoothDevice.ACTION_ACL_DISCONNECT_REQUESTED);
        bluetoothDeviceStatusChangedFilter.addAction(BluetoothDevice.ACTION_BOND_STATE_CHANGED);
        registerReceiver(BluetoothConnection.bluetoothDeviceStatusBR, bluetoothDeviceStatusChangedFilter);

        Thread internetDetector = new Thread(){
            @Override
            public void run() {
                super.run();
                while(true){
                    try {
                        internetConnectionStatus = isConnected();
                        Thread.sleep(10000);
                        if (internetConnectionStatus) {Application.toastMessage("Internet connection detected");}
                        else {Application.toastMessage("Proper internet connection has not been detected");}
                    } catch (Throwable th) { th.printStackTrace(); Application.toastMessage(
```

Archivo - Application.java

```
th.toString());}
                }
            }
        };

        internetDetector.start();

        Application.persistentDataSave.putString("MAIN_TEXT_KEY", new String());
        Application.persistentDataSave.putString("ONGOING_TEXT_KEY", new String());

    }

    @Override
    public void onTerminate() {
        super.onTerminate();
    }

    private static Activity mActivity;
    private static Context mContext;
    public static Bundle persistentDataSave = new Bundle();


    public static Activity getActivity() { return mActivity; }
    public static void setActivity(Activity mActivitySet) { mActivity = mActivitySet; }

    public static Context getContext() { return mContext; }
    public static void setContext(Context mContextoSet) { mContext = mContextoSet; }

    public static void toastMessage(final String message){
        mActivity.runOnUiThread(new Runnable() {
            @Override
            public void run() {
                if (mActivity != null) {
                    //Toast.makeText(mActivity, message, Toast.LENGTH_SHORT).show();
                }
            }
        });
    }

    private static boolean internetConnectionStatus;

    public static boolean isSystemOnline(){
        return internetConnectionStatus;
    }

    public boolean isConnected() throws InterruptedException, IOException {
        final String command = "ping -c 1 google.com";
        return Runtime.getRuntime().exec(command).waitFor() == 0;
    }



}
```

Archivo - MainActivity.java

```java
package com.example.SAPLM;

import android.Manifest;
import android.content.pm.PackageManager;
import android.os.Bundle;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import androidx.appcompat.app.AppCompatActivity;
import android.util.DisplayMetrics;
import android.view.View;
import android.widget.ImageButton;
import android.widget.TextView;

import com.example.SAPLM.bluetoothActivities.BluetoothConnection;
import com.example.SAPLM.settingsActivities.settings_layout;

public class MainActivity extends AppCompatActivity {

    static ImageButton imgBtn_microphone;
    static ImageButton imgBtn_microphoneBackground;
    static ImageButton imgBtn_commands;
    static ImageButton imgBtn_contacts;
    static ImageButton imgBtn_bluetooth;
    static ImageButton imgBtn_settings;
    static ImageButton imgBtn_menu;
    static MainMenuHandler mainMenuHandler;
    static TextView textView_speechRecognitionResults;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        if(savedInstanceState == null) {

            mainMenuHandler = new MainMenuHandler(this, this);
            Application.setActivity(this);
            Application.setContext(this);
            MainMenuHandler.dpiConstant = ((float) this.getResources().getDisplayMetrics().
densityDpi / DisplayMetrics.DENSITY_DEFAULT);

            imgBtn_microphone = findViewById(R.id.imgBtn_microphone);
            imgBtn_microphoneBackground = findViewById(R.id.imgBtn_microphone_background);
            imgBtn_commands = findViewById(R.id.imgBtn_commands);
            imgBtn_contacts = findViewById(R.id.imgBtn_contacts);
            imgBtn_bluetooth = findViewById(R.id.imgBtn_bluetooth);
            imgBtn_settings = findViewById(R.id.imgBtn_settings);
            imgBtn_menu = findViewById(R.id.imgBtn_menu);
            textView_speechRecognitionResults = findViewById(R.id.
textView_speechRecognitionResults);

            if (ContextCompat.checkSelfPermission(this,
                    Manifest.permission.RECORD_AUDIO)
                    != PackageManager.PERMISSION_GRANTED) {
                if (ActivityCompat.shouldShowRequestPermissionRationale(this,
                        Manifest.permission.RECORD_AUDIO)) {
                } else {
                    ActivityCompat.requestPermissions(this,
                            new String[]{Manifest.permission.RECORD_AUDIO},
                            527);
                }
            } else {
```

```
62                  GoogleSpeechRecognizer.setupRecognizer();
63                  PocketsphinxListener.setupRecognizer();
64                  PocketsphinxListener.startListening();
65
66              }
67
68              settings_layout.setupSettingsList();
69
70              BluetoothConnection.initBTAutoConnectService();
71
72          }
73      }
74
75      @Override
76      public void onRequestPermissionsResult(int requestCode,
77                                             String permissions[], int[] grantResults) {
78          switch (requestCode) {
79              case 527: {
80                  // If request is cancelled, the result arrays are empty.
81                  if (grantResults.length > 0
82                          && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
83
84                      GoogleSpeechRecognizer.setupRecognizer();
85                      PocketsphinxListener.setupRecognizer();
86                      PocketsphinxListener.startListening();
87
88                  } else { }
89              }
90          }
91      }
92
93      public void onClickListener_menu(View view){ mainMenuHandler.menuButtonOnClick(); }
94
95      public void onClickListener_commands(View view){ mainMenuHandler.commandsButtonOnClick(
    ); }
96
97      public void onClickListener_settings(View view){ mainMenuHandler.settingsButtonOnClick(
    ); }
98
99      public void onClickListener_contacts(View view){ mainMenuHandler.contactsButtonOnClick(
    ); }
100
101     public void onClickListener_bluetooth(View view){ mainMenuHandler.
    bluetoothButtonOnClick(); }
102
103     public void onClickListener_microphone(View view){ MicrophoneHandler.microphoneOnClick(
    ); }
104
105     @Override
106     public void onDestroy() {
107
108         PocketsphinxListener.shutdownRecognizer();
109         GoogleSpeechRecognizer.shutdownRecognizer();
110         BluetoothConnection.closeAllConnections();
111         TextToSpeechHandler.shutdownTextSpeech();
112
113         super.onDestroy();
114     }
115
116 }
117
```

Archivo - MainMenuHandler.java

```java
package com.example.SAPLM;

import android.animation.ValueAnimator;
import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.view.View;
import android.view.animation.AccelerateDecelerateInterpolator;

import com.example.SAPLM.bluetoothActivities.BluetoothActivity;
import com.example.SAPLM.commandActivity.CommandsActivity;
import com.example.SAPLM.settingsActivities.settings_layout;

public class MainMenuHandler extends MainActivity{

    public static float dpiConstant;
    public static float radialMenuRadius = 263.0f;
    public static boolean menuUncoiled = false;
    public static boolean isAnimationRunning = false;
    public static Context mainMenuContext;
    public static Activity mainMenuActivity;

    public MainMenuHandler(Context context, Activity activity){
        this.mainMenuActivity = activity;
        this.mainMenuContext = context;
    }

    public static void menuButtonOnClick(){
        long animationDuration = 850;

        if(isAnimationRunning==false) {
            isAnimationRunning = true;

            if (menuUncoiled == false) {
                ValueAnimator animatorCommands = ValueAnimator.ofFloat(360, 72);
                animatorCommands.setDuration(animationDuration);
                animatorCommands.setInterpolator(new AccelerateDecelerateInterpolator());
                animatorCommands.addUpdateListener(new ValueAnimator.AnimatorUpdateListener(
) {
                    @Override
                    public void onAnimationUpdate(ValueAnimator animation) {
                        float value = ((Float) (animation.getAnimatedValue()))
                                .floatValue();

                        float translationX = (float) (Math.cos(value * (Math.PI / 180.0f)) *
 (radialMenuRadius / 2.0f) - (radialMenuRadius / 2.0f));
                        float translationY = (float) (Math.sqrt(-Math.pow(translationX, 2.0d
) - radialMenuRadius * translationX));
                        if (value > 180.0f) {
                            translationY = -translationY;
                        }

                        imgBtn_commands.setTranslationX(convertDpToPixel(translationX));
                        imgBtn_commands.setTranslationY(convertDpToPixel(translationY));

                    }
                });
                imgBtn_commands.setVisibility(View.VISIBLE);
                animatorCommands.start();

                ValueAnimator animatorContacts = ValueAnimator.ofFloat(360, 144);
                animatorContacts.setDuration(animationDuration);
                animatorContacts.setInterpolator(new AccelerateDecelerateInterpolator());
```

Archivo - MainMenuHandler.java

```
61                  animatorContacts.addUpdateListener(new ValueAnimator.AnimatorUpdateListener
    () {
62                      @Override
63                      public void onAnimationUpdate(ValueAnimator animation) {
64                          float value = ((Float) (animation.getAnimatedValue()))
65                                  .floatValue();
66
67                          float translationX = (float) (Math.cos(value * (Math.PI / 180.0f))
    * (radialMenuRadius / 2.0f) - (radialMenuRadius / 2.0f));
68                          float translationY = (float) (Math.sqrt(-Math.pow(translationX, 2.
    0d) - radialMenuRadius * translationX));
69                          if (value > 180.0f) {
70                              translationY = -translationY;
71                          }
72
73                          imgBtn_contacts.setTranslationX(convertDpToPixel(translationX));
74                          imgBtn_contacts.setTranslationY(convertDpToPixel(translationY));
75
76                      }
77                  });
78                  imgBtn_contacts.setVisibility(View.VISIBLE);
79                  animatorContacts.start();
80
81                  ValueAnimator animatorSettings = ValueAnimator.ofFloat(360, 216);
82                  animatorSettings.setDuration(animationDuration);
83                  animatorSettings.setInterpolator(new AccelerateDecelerateInterpolator());
84                  animatorSettings.addUpdateListener(new ValueAnimator.AnimatorUpdateListener
    () {
85                      @Override
86                      public void onAnimationUpdate(ValueAnimator animation) {
87                          float value = ((Float) (animation.getAnimatedValue()))
88                                  .floatValue();
89
90                          float translationX = (float) (Math.cos(value * (Math.PI / 180.0f))
    * (radialMenuRadius / 2.0f) - (radialMenuRadius / 2.0f));
91                          float translationY = (float) (Math.sqrt(-Math.pow(translationX, 2.
    0d) - radialMenuRadius * translationX));
92                          if (value > 180.0f) {
93                              translationY = -translationY;
94                          }
95
96                          imgBtn_settings.setTranslationX(convertDpToPixel(translationX));
97                          imgBtn_settings.setTranslationY(convertDpToPixel(translationY));
98
99                      }
100                 });
101                 imgBtn_settings.setVisibility(View.VISIBLE);
102                 animatorSettings.start();
103
104                 ValueAnimator animatorBluetooth = ValueAnimator.ofFloat(360, 288);
105                 animatorBluetooth.setDuration(animationDuration);
106                 animatorBluetooth.setInterpolator(new AccelerateDecelerateInterpolator());
107                 animatorBluetooth.addUpdateListener(new ValueAnimator.
    AnimatorUpdateListener() {
108                     @Override
109                     public void onAnimationUpdate(ValueAnimator animation) {
110                         float value = ((Float) (animation.getAnimatedValue()))
111                                 .floatValue();
112
113                         float translationX = (float) (Math.cos(value * (Math.PI / 180.0f))
    * (radialMenuRadius / 2.0f) - (radialMenuRadius / 2.0f));
114                         float translationY = (float) (Math.sqrt(-Math.pow(translationX, 2.
    0d) - radialMenuRadius * translationX));
```

```
115                         if (value > 180.0f) {
116                             translationY = -translationY;
117                         }
118
119                         imgBtn_bluetooth.setTranslationX(convertDpToPixel(translationX));
120                         imgBtn_bluetooth.setTranslationY(convertDpToPixel(translationY));
121
122                     }
123                 });
124                 imgBtn_bluetooth.setVisibility(View.VISIBLE);
125                 animatorBluetooth.start();
126
127                 imgBtn_commands.animate().alpha(1.0f).setDuration(animationDuration).
    setInterpolator(new AccelerateDecelerateInterpolator());
128                 imgBtn_contacts.animate().alpha(1.0f).setDuration(animationDuration).
    setInterpolator(new AccelerateDecelerateInterpolator());
129                 imgBtn_settings.animate().alpha(1.0f).setDuration(animationDuration).
    setInterpolator(new AccelerateDecelerateInterpolator());
130                 imgBtn_bluetooth.animate().alpha(1.0f).setDuration(animationDuration).
    setInterpolator(new AccelerateDecelerateInterpolator()).withEndAction(new Runnable() {
131                     @Override
132                     public void run() {
133                         isAnimationRunning = false;
134                     }
135                 });
136
137                 menuUncoiled = true;
138
139
140             } else {
141
142
143                 ValueAnimator animatorCommands = ValueAnimator.ofFloat(72, 360);
144                 animatorCommands.setDuration(animationDuration);
145                 animatorCommands.setInterpolator(new AccelerateDecelerateInterpolator());
146                 animatorCommands.addUpdateListener(new ValueAnimator.AnimatorUpdateListener
    () {
147                     @Override
148                     public void onAnimationUpdate(ValueAnimator animation) {
149                         float value = ((Float) (animation.getAnimatedValue()))
150                                 .floatValue();
151
152                         float translationX = (float) (Math.cos(value * (Math.PI / 180.0f))
    * (radialMenuRadius / 2.0f) - (radialMenuRadius / 2.0f));
153                         float translationY = (float) (Math.sqrt(-Math.pow(translationX, 2.
    0d) - radialMenuRadius * translationX));
154                         if (value > 180.0f) {
155                             translationY = -translationY;
156                         }
157
158                         imgBtn_commands.setTranslationX(convertDpToPixel(translationX));
159                         imgBtn_commands.setTranslationY(convertDpToPixel(translationY));
160
161                     }
162                 });
163                 imgBtn_commands.setVisibility(View.VISIBLE);
164                 animatorCommands.start();
165
166                 ValueAnimator animatorContacts = ValueAnimator.ofFloat(144, 360);
167                 animatorContacts.setDuration(animationDuration);
168                 animatorContacts.setInterpolator(new AccelerateDecelerateInterpolator());
169                 animatorContacts.addUpdateListener(new ValueAnimator.AnimatorUpdateListener
    () {
```

```
170                     @Override
171                     public void onAnimationUpdate(ValueAnimator animation) {
172                         float value = ((Float) (animation.getAnimatedValue()))
173                                 .floatValue();
174 
175                         float translationX = (float) (Math.cos(value * (Math.PI / 180.0f))
    * (radialMenuRadius / 2.0f) - (radialMenuRadius / 2.0f));
176                         float translationY = (float) (Math.sqrt(-Math.pow(translationX, 2.
    0d) - radialMenuRadius * translationX));
177                         if (value > 180.0f) {
178                             translationY = -translationY;
179                         }
180 
181                         imgBtn_contacts.setTranslationX(convertDpToPixel(translationX));
182                         imgBtn_contacts.setTranslationY(convertDpToPixel(translationY));
183 
184                     }
185                 });
186                 imgBtn_contacts.setVisibility(View.VISIBLE);
187                 animatorContacts.start();
188 
189                 ValueAnimator animatorSettings = ValueAnimator.ofFloat(216, 360);
190                 animatorSettings.setDuration(animationDuration);
191                 animatorSettings.setInterpolator(new AccelerateDecelerateInterpolator());
192                 animatorSettings.addUpdateListener(new ValueAnimator.AnimatorUpdateListener
    () {
193                     @Override
194                     public void onAnimationUpdate(ValueAnimator animation) {
195                         float value = ((Float) (animation.getAnimatedValue()))
196                                 .floatValue();
197 
198                         float translationX = (float) (Math.cos(value * (Math.PI / 180.0f))
    * (radialMenuRadius / 2.0f) - (radialMenuRadius / 2.0f));
199                         float translationY = (float) (Math.sqrt(-Math.pow(translationX, 2.
    0d) - radialMenuRadius * translationX));
200                         if (value > 180.0f) {
201                             translationY = -translationY;
202                         }
203 
204                         imgBtn_settings.setTranslationX(convertDpToPixel(translationX));
205                         imgBtn_settings.setTranslationY(convertDpToPixel(translationY));
206 
207                     }
208                 });
209                 imgBtn_settings.setVisibility(View.VISIBLE);
210                 animatorSettings.start();
211 
212                 ValueAnimator animatorBluetooth = ValueAnimator.ofFloat(288, 360);
213                 animatorBluetooth.setDuration(animationDuration);
214                 animatorBluetooth.setInterpolator(new AccelerateDecelerateInterpolator());
215                 animatorBluetooth.addUpdateListener(new ValueAnimator.
    AnimatorUpdateListener() {
216                     @Override
217                     public void onAnimationUpdate(ValueAnimator animation) {
218                         float value = ((Float) (animation.getAnimatedValue()))
219                                 .floatValue();
220 
221                         float translationX = (float) (Math.cos(value * (Math.PI / 180.0f))
    * (radialMenuRadius / 2.0f) - (radialMenuRadius / 2.0f));
222                         float translationY = (float) (Math.sqrt(-Math.pow(translationX, 2.
    0d) - radialMenuRadius * translationX));
223                         if (value > 180.0f) {
224                             translationY = -translationY;
```

```
225                         }
226
227                         imgBtn_bluetooth.setTranslationX(convertDpToPixel(translationX));
228                         imgBtn_bluetooth.setTranslationY(convertDpToPixel(translationY));
229
230                     }
231                 });
232                 imgBtn_bluetooth.setVisibility(View.VISIBLE);
233                 animatorBluetooth.start();
234
235                 imgBtn_commands.animate().alpha(0.0f).setDuration(animationDuration).
    setInterpolator(new AccelerateDecelerateInterpolator());
236                 imgBtn_contacts.animate().alpha(0.0f).setDuration(animationDuration).
    setInterpolator(new AccelerateDecelerateInterpolator());
237                 imgBtn_settings.animate().alpha(0.0f).setDuration(animationDuration).
    setInterpolator(new AccelerateDecelerateInterpolator());
238                 imgBtn_bluetooth.animate().alpha(0.0f).setDuration(animationDuration).
    setInterpolator(new AccelerateDecelerateInterpolator()).withEndAction(new Runnable() {
239                     @Override
240                     public void run() {
241                         isAnimationRunning = false;
242                     }
243                 });
244
245                 menuUncoiled = false;
246             }
247         }
248     }
249
250     public static void commandsButtonOnClick(){
251         Context currentContext = Application.getContext();
252         Activity currentActivity = Application.getActivity();
253         Intent i = new Intent(currentContext, CommandsActivity.class);
254         currentContext.startActivity(i);
255     }
256
257     public static void contactsButtonOnClick(){
258         Context currentContext = Application.getContext();
259         Activity currentActivity = Application.getActivity();
260         Intent i = new Intent(currentContext, ContactsActivity.class);
261         currentContext.startActivity(i);
262     }
263
264     public static void bluetoothButtonOnClick(){
265         Context currentContext = Application.getContext();
266         Activity currentActivity = Application.getActivity();
267         Intent i = new Intent(currentContext, BluetoothActivity.class);
268         currentContext.startActivity(i);
269     }
270
271     public static void settingsButtonOnClick(){
272         Intent i = new Intent(Application.getContext(), settings_layout.class);
273         mainMenuContext.startActivity(i);
274         //mainMenuActivity.overridePendingTransition(R.anim.fade_in, R.anim.fade_out);
275     }
276
277
278
279
280
281
282
283
```

```
284     /**
285      * This method converts dp unit to equivalent pixels, depending on device density.
286      *
287      * @param dp A value in dp (density independent pixels) unit. Which we need to convert
    into pixels
288      * @return A float value to represent px equivalent to dp depending on device density
289      */
290     public static float convertDpToPixel(float dp){ return dp * dpiConstant; }
291
292     /**
293      * This method converts device specific pixels to density independent pixels.
294      *
295      * @param px A value in px (pixels) unit. Which we need to convert into db
296      * @return A float value to represent dp equivalent to px value
297      */
298     public static float convertPixelsToDp(float px){
299         return px / dpiConstant;
300     }
301 }
302
```

Archivo - ContactsActivity.java

```java
package com.example.SAPLM;

import android.graphics.PorterDuff;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import androidx.core.content.ContextCompat;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import android.view.MenuItem;

public class ContactsActivity extends AppCompatActivity {
    private static Toolbar toolbar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.contacts_layout);
        Application.setActivity(this);
        Application.setContext(this);

        toolbar = this.findViewById(R.id.toolbar);

        setSupportActionBar(toolbar);

        // add back arrow to toolbar
        if (getSupportActionBar() != null){
            getSupportActionBar().setDisplayHomeAsUpEnabled(true);
            getSupportActionBar().setDisplayShowHomeEnabled(true);

            final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.abc_ic_ab_back_material);
            upArrow.setColorFilter(getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP);
            getSupportActionBar().setHomeAsUpIndicator(upArrow);
        }

        TextToSpeechHandler.initializeTextToSpeech();
        try {
            Thread.sleep(250);
        }catch (Throwable th) { Application.toastMessage(th.toString());}

        TextToSpeechHandler.addToSpeakingQueue("hola");

    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        if (item.getItemId() == android.R.id.home) // Press Back Icon
        {
            finish();
        }

        return super.onOptionsItemSelected(item);
    }
}

```

Archivo - MicrophoneHandler.java

```java
package com.example.SAPLM;

import android.graphics.PorterDuff;
import android.view.animation.AccelerateDecelerateInterpolator;

import java.util.concurrent.Callable;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.Future;

public class MicrophoneHandler extends MainActivity {

    public enum MicrophoneState { INHIBITED, LISTENING_KEYOWRD, LISTENING_COMMAND}

    static private MicrophoneState currentState = MicrophoneState.LISTENING_KEYOWRD;

    public static void microphoneOnClick(){
        switch (currentState){
            case LISTENING_KEYOWRD:
                PocketsphinxListener.stopRecognizer();
                setState(MicrophoneState.INHIBITED);
                break;
            case LISTENING_COMMAND:
                GoogleSpeechRecognizer.forceStopRecognizer();
                //PocketsphinxListener.startListening();
                setState(MicrophoneState.LISTENING_KEYOWRD);
                break;
            case INHIBITED:
                setState(MicrophoneState.LISTENING_KEYOWRD);
                PocketsphinxListener.startListening();
                break;
        }

    }

    public static MicrophoneState getState(){
        return currentState;
    }

    public static void setState(MicrophoneState desiredState){
        switch (desiredState){
            case INHIBITED:
                imgBtn_microphoneBackground.clearColorFilter();
                imgBtn_microphoneBackground.setColorFilter(Application.getContext().getColor(R.color.ColorBackMicInhibited), PorterDuff.Mode.SRC_ATOP);
                imgBtn_microphone.setImageResource(R.drawable.ic_main_mic_off);
                endAnimation();
                currentState = MicrophoneState.INHIBITED;
                break;
            case LISTENING_KEYOWRD:
                imgBtn_microphoneBackground.clearColorFilter();
                imgBtn_microphoneBackground.setColorFilter(Application.getContext().getColor(R.color.ColorBackMicKeyword), PorterDuff.Mode.SRC_ATOP);
                imgBtn_microphone.setImageResource(R.drawable.ic_main_mic_listening);
                endAnimation();
                currentState = MicrophoneState.LISTENING_KEYOWRD;
                break;
            case LISTENING_COMMAND:
                imgBtn_microphoneBackground.clearColorFilter();
                imgBtn_microphoneBackground.setColorFilter(Application.getContext().getColor(R.color.ColorBackMicCommand), PorterDuff.Mode.SRC_ATOP);
                imgBtn_microphone.setImageResource(R.drawable.ic_main_mic_listening);
                startAnimation();
```

```
61                  currentState = MicrophoneState.LISTENING_COMMAND;
62                  break;
63          }
64      }
65
66      private static ExecutorService executor = Executors.newSingleThreadExecutor();
67      private static boolean terminateAnimation = false;
68      private static boolean animationHasFinished = false;
69
70      public static void startAnimation(){
71          terminateAnimation = false;
72          Future<Void> future = executor.submit(new Callable<Void>() {
73              public Void call() throws Exception {
74                  try {
75                      while (terminateAnimation == false) {
76                          imgBtn_microphoneBackground.animate().scaleX(0.90f).setDuration(350
    ).setInterpolator(new AccelerateDecelerateInterpolator());
77                          imgBtn_microphone.animate().scaleX(0.90f).setDuration(350).
    setInterpolator(new AccelerateDecelerateInterpolator());
78                          imgBtn_microphoneBackground.animate().scaleY(0.90f).setDuration(350
    ).setInterpolator(new AccelerateDecelerateInterpolator());
79                          imgBtn_microphone.animate().scaleY(0.90f).setDuration(350).
    setInterpolator(new AccelerateDecelerateInterpolator()).withEndAction(new Runnable() {
80                              @Override
81                              public void run() {
82                                  animationHasFinished = true;
83                              }
84                          });
85                          while(animationHasFinished==false);
86                          animationHasFinished= false;
87                          imgBtn_microphoneBackground.animate().scaleX(1.00f).setDuration(350
    ).setInterpolator(new AccelerateDecelerateInterpolator());
88                          imgBtn_microphone.animate().scaleX(1.00f).setDuration(350).
    setInterpolator(new AccelerateDecelerateInterpolator());
89                          imgBtn_microphoneBackground.animate().scaleY(1.00f).setDuration(350
    ).setInterpolator(new AccelerateDecelerateInterpolator());
90                          imgBtn_microphone.animate().scaleY(1.00f).setDuration(350).
    setInterpolator(new AccelerateDecelerateInterpolator()).withEndAction(new Runnable() {
91                              @Override
92                              public void run() {
93                                  animationHasFinished = true;
94                              }
95                          });;
96                          while(animationHasFinished==false);
97                          animationHasFinished = false;
98                      }
99                      terminateAnimation = false;
100                 }catch (Throwable th){ ;
101                     th.printStackTrace();
102                 }
103                 return null;
104             }
105         });
106     }
107
108     public static void endAnimation(){
109         terminateAnimation = true;
110     }
111
112 }
113
```

Archivo - TextToSpeechHandler.java

```java
package com.example.SAPLM;

import android.speech.tts.TextToSpeech;
import android.speech.tts.Voice;

import java.util.Locale;
import java.util.Set;

public class TextToSpeechHandler {

    private static TextToSpeech textToSpeech;
    public static boolean isInitialized = false;

    public static void initializeTextToSpeech(){
        if (isInitialized == false) {
            textToSpeech = new TextToSpeech(Application.getContext(), new TextToSpeech.OnInitListener() {
                @Override
                public void onInit(int status) {
                    try {
                        if (status != TextToSpeech.ERROR) {
                            textToSpeech.setLanguage(new Locale("spa"));
                            //textToSpeech.setLanguage(Locale.UK);
                            //textToSpeech.setOnUtteranceProgressListener(utteranceListener);
                            textToSpeech.setPitch(1.0f);
                            textToSpeech.setSpeechRate(1.0f);
                            textToSpeech.getVoices();
                            isInitialized = true;
                        }
                    } catch (Throwable th) {
                        Application.toastMessage(th.toString());
                    }
                }
            });
        }
    }

    public static void addToSpeakingQueue(String text){
        try {
            if (isInitialized) {
                textToSpeech.speak(text, TextToSpeech.QUEUE_ADD, null, "232");
            }
        } catch (Throwable th) {
            th.printStackTrace();
            Application.toastMessage(th.toString());
        }
    }

    public static void shutdownTextSpeech(){
        if(textToSpeech != null) {

            textToSpeech.stop();
            textToSpeech.shutdown();
        }
    }

    public static Set<Voice> getVoicesList(){
        if (isInitialized){
            return textToSpeech.getVoices();
        } else {
            return null;
        }
```

Archivo - TextToSpeechHandler.java

```
62      }
63
64      public static void setVoice(Voice targetVoice){
65          if(isInitialized) {
66              textToSpeech.setVoice(targetVoice);
67          }
68      }
69
70
71
72  }
73
74
75
```

Archivo - PocketsphinxListener.java

```java
package com.example.SAPLM;

import java.io.File;

import edu.cmu.pocketsphinx.Assets;
import edu.cmu.pocketsphinx.Hypothesis;
import edu.cmu.pocketsphinx.RecognitionListener;
import edu.cmu.pocketsphinx.SpeechRecognizer;
import edu.cmu.pocketsphinx.SpeechRecognizerSetup;

public final class PocketsphinxListener implements RecognitionListener {
    private static SpeechRecognizer recognizer;
    private static String KEYPHRASE = "asistente";
    private static final String KWS_SEARCH = "wakeup";


    @Override
    public void onPartialResult(Hypothesis hypothesis) {
        if (hypothesis == null)
            return;

        String text = hypothesis.getHypstr();

        if (text.contains(KEYPHRASE)) {

            Application.toastMessage(KEYPHRASE);
            recognizer.stop();
            //recognizer.cancel();
            //recognizer.shutdown();
            GoogleSpeechRecognizer.startSpeechListening();
            //recognizer.startListening(KWS_SEARCH);
        }

    }


    @Override
    public void onResult(Hypothesis hypothesis) {
        if (hypothesis != null) {
            String text = hypothesis.getHypstr();

        }
    }

    @Override
    public void onBeginningOfSpeech() {
    }

    @Override
    public void onEndOfSpeech() {

    }

    public static void setupRecognizer(){
        try {
            Assets assets = new Assets(Application.getContext());
            File assetDir = assets.syncAssets();
            recognizer = SpeechRecognizerSetup.defaultSetup()
                    .setAcousticModel(new File(assetDir, "es-es-ptm"))
                    .setDictionary(new File(assetDir, "cmudict-es-es.dict"))
                    .setKeywordThreshold(Float.MIN_VALUE
                    )
                    .setRawLogDir(assetDir) // To disable logging of raw audio comment out
```

```
 63 this call (takes a lot of space on the device)
 64
 65                     .getRecognizer();
 66             recognizer.addListener(new PocketsphinxListener());
 67             recognizer.addKeyphraseSearch(KWS_SEARCH, KEYPHRASE);
 68         } catch (Throwable th){
 69             Application.toastMessage(th.toString());
 70         }
 71     }
 72
 73     public static void startListening(){
 74         recognizer.startListening(KWS_SEARCH);
 75     }
 76
 77     public static void shutdownRecognizer(){
 78         if (recognizer != null) {
 79             recognizer.cancel();
 80             recognizer.shutdown();
 81         }
 82     }
 83
 84     public static void stopRecognizer(){
 85         recognizer.stop();
 86     }
 87
 88     @Override
 89     public void onError(Exception error) {
 90         Application.toastMessage("Error: " + error.toString());
 91     }
 92
 93     @Override
 94     public void onTimeout() {
 95         Application.toastMessage("Timeout!");
 96     }
 97
 98
 99
100
101 }
102
```

Archivo - GoogleSpeechRecognizer.java

```java
package com.example.SAPLM;

import android.content.Intent;
import android.os.Bundle;
import android.speech.RecognitionListener;
import android.speech.RecognizerIntent;
import android.speech.SpeechRecognizer;
import android.view.View;

import com.example.SAPLM.bluetoothActivities.BluetoothConnection;
import com.example.SAPLM.bluetoothActivities.UnifiedTransmissionProtocol;

import org.apache.commons.lang3.StringUtils;

import java.text.Normalizer;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Calendar;
import java.util.Date;
import java.util.Locale;

public class GoogleSpeechRecognizer extends MainActivity implements RecognitionListener {

    private static SpeechRecognizer speechRecognizer = null;

    private boolean forceCancel = false;

    public static void startSpeechListening(){


        String language =  "es-ES";
        Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
        intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,language);
        intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE, language);
        intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_PREFERENCE, language);
        intent.putExtra(RecognizerIntent.EXTRA_ONLY_RETURN_LANGUAGE_PREFERENCE, language);
        intent.putExtra(RecognizerIntent.EXTRA_CALLING_PACKAGE, Application.getActivity().
getPackageName());
        intent.putExtra(RecognizerIntent.EXTRA_PARTIAL_RESULTS,true);
        if (Application.isSystemOnline()==false) { intent.putExtra(RecognizerIntent.
EXTRA_PREFER_OFFLINE,true); }
        speechRecognizer.startListening(intent);


        if (Application.isSystemOnline()) {
            Thread autoStop = new Thread() {
                @Override
                public void run() {
                    super.run();
                    try {
                        Thread.sleep(4000);
                        forceStopRecognizer();
                    } catch (Throwable th) {
                        Application.toastMessage(th.toString());
                    }
                }
            };
            autoStop.start();
        }

    }

```

```
62      public static void setupRecognizer(){
63          speechRecognizer = SpeechRecognizer.createSpeechRecognizer(Application.getContext()
    );
64          speechRecognizer.setRecognitionListener(new GoogleSpeechRecognizer());
65      }
66
67      public static void shutdownRecognizer(){
68          speechRecognizer.stopListening();
69          speechRecognizer.cancel();
70          speechRecognizer.destroy();
71          speechRecognizer = null;
72          System.gc();
73      }
74
75      public static void forceStopRecognizer(){
76          Application.getActivity().runOnUiThread(
77                  new Runnable() {
78                      @Override
79                      public void run() {
80                          speechRecognizer.stopListening();
81                          MicrophoneHandler.setState(MicrophoneHandler.MicrophoneState.
    LISTENING_KEYOWRD);
82                      }
83                  }
84          );
85
86      }
87
88
89      public static void analiseListenerResults(ArrayList<String> data, float[]
    confidenceScore){
90          outerloop:
91          for (int i = 0; i < data.size(); i++) {
92              String hypothesis = Normalizer.normalize(((String) data.get(i)).toLowerCase(),
    Normalizer.Form.NFKD);
93              String[] hypothesisSplit = hypothesis.split(" ");
94
95              if (Arrays.asList(hypothesisSplit).contains("encender") && Arrays.asList(
    hypothesisSplit).contains("velador")) {
96                  UnifiedTransmissionProtocol.modulesList.get(2).getDevicesList().get(0).
    setData(Boolean.TRUE);
97                  UnifiedTransmissionProtocol.Module.syncAllDevicesValueToSystem();
98                  break outerloop;
99              }
100
101             if (Arrays.asList(hypothesisSplit).contains("encender") && Arrays.asList(
    hypothesisSplit).contains("estufa")) {
102                 UnifiedTransmissionProtocol.modulesList.get(2).getDevicesList().get(3).
    setData(Boolean.TRUE);
103                 UnifiedTransmissionProtocol.Module.syncAllDevicesValueToSystem();
104                 break outerloop;
105             }
106
107             if (Arrays.asList(hypothesisSplit).contains("encender") && Arrays.asList(
    hypothesisSplit).contains("ventilador")) {
108                 UnifiedTransmissionProtocol.modulesList.get(2).getDevicesList().get(2).
    setData(Boolean.TRUE);
109                 UnifiedTransmissionProtocol.Module.syncAllDevicesValueToSystem();
110                 break outerloop;
111             }
112
113             if (Arrays.asList(hypothesisSplit).contains("encender") && Arrays.asList(
    hypothesisSplit).contains("luz")) {
```

```
114                 UnifiedTransmissionProtocol.modulesList.get(2).getDevicesList().get(1).
    setData(Boolean.TRUE);
115                 UnifiedTransmissionProtocol.Module.syncAllDevicesValueToSystem();
116                 break outerloop;
117             }
118 
119             if (Arrays.asList(hypothesisSplit).contains("apagar") && Arrays.asList(
    hypothesisSplit).contains("velador")) {
120                 UnifiedTransmissionProtocol.modulesList.get(2).getDevicesList().get(0).
    setData(Boolean.FALSE);
121                 UnifiedTransmissionProtocol.Module.syncAllDevicesValueToSystem();
122                 break outerloop;
123             }
124 
125             if (Arrays.asList(hypothesisSplit).contains("apagar") && Arrays.asList(
    hypothesisSplit).contains("estufa")) {
126                 UnifiedTransmissionProtocol.modulesList.get(2).getDevicesList().get(3).
    setData(Boolean.FALSE);
127                 UnifiedTransmissionProtocol.Module.syncAllDevicesValueToSystem();
128                 break outerloop;
129             }
130 
131             if (Arrays.asList(hypothesisSplit).contains("apagar") && Arrays.asList(
    hypothesisSplit).contains("ventilador")) {
132                 UnifiedTransmissionProtocol.modulesList.get(2).getDevicesList().get(2).
    setData(Boolean.FALSE);
133                 UnifiedTransmissionProtocol.Module.syncAllDevicesValueToSystem();
134                 break outerloop;
135             }
136 
137             if (Arrays.asList(hypothesisSplit).contains("apagar") && Arrays.asList(
    hypothesisSplit).contains("luz")) {
138                 UnifiedTransmissionProtocol.modulesList.get(2).getDevicesList().get(1).
    setData(Boolean.FALSE);
139                 UnifiedTransmissionProtocol.Module.syncAllDevicesValueToSystem();
140                 break outerloop;
141             }
142 
143             if (Arrays.asList(hypothesisSplit).contains("encender") && Arrays.asList(
    hypothesisSplit).contains("todo")) {
144                 for(UnifiedTransmissionProtocol.Module.Device dev :
    UnifiedTransmissionProtocol.modulesList.get(2).getDevicesList()) {
145                     dev.setData(Boolean.TRUE);
146                 }
147                 UnifiedTransmissionProtocol.Module.syncAllDevicesValueToSystem();
148                 break outerloop;
149             }
150 
151             if (Arrays.asList(hypothesisSplit).contains("apagar") && Arrays.asList(
    hypothesisSplit).contains("todo")) {
152                 for(UnifiedTransmissionProtocol.Module.Device dev :
    UnifiedTransmissionProtocol.modulesList.get(2).getDevicesList()) {
153                     dev.setData(Boolean.FALSE);
154                 }
155                 UnifiedTransmissionProtocol.Module.syncAllDevicesValueToSystem();
156                 break outerloop;
157             }
158 
159             if (Arrays.asList(hypothesisSplit).contains("llamar") && Arrays.asList(
    hypothesisSplit).contains("enfermera")) {
160                 UnifiedTransmissionProtocol.modulesList.get(3).getDevicesList().get(2).
    setData(Boolean.TRUE);
161                 UnifiedTransmissionProtocol.Module.syncAllDevicesValueToSystem();
```

```
162                 break outerloop;
163             }
164 
165             if (Arrays.asList(hypothesisSplit).contains("llamar") && Arrays.asList(
    hypothesisSplit).contains("enfermeria")) {
166                 UnifiedTransmissionProtocol.modulesList.get(3).getDevicesList().get(2).
    setData(Boolean.TRUE);
167                 UnifiedTransmissionProtocol.Module.syncAllDevicesValueToSystem();
168                 break outerloop;
169             }
170 
171             if (Arrays.asList(hypothesisSplit).contains("subir") && Arrays.asList(
    hypothesisSplit).contains("toda") && Arrays.asList(hypothesisSplit).contains("cortina")) {
172                 UnifiedTransmissionProtocol.modulesList.get(3).getDevicesList().get(1).
    setData((byte)7);
173                 UnifiedTransmissionProtocol.Module.syncAllDevicesValueToSystem();
174                 break outerloop;
175             }
176 
177             if (Arrays.asList(hypothesisSplit).contains("bajar") && Arrays.asList(
    hypothesisSplit).contains("toda") && Arrays.asList(hypothesisSplit).contains("cortina")) {
178                 UnifiedTransmissionProtocol.modulesList.get(3).getDevicesList().get(1).
    setData((byte)0);
179                 UnifiedTransmissionProtocol.Module.syncAllDevicesValueToSystem();
180                 break outerloop;
181             }
182 
183             if (Arrays.asList(hypothesisSplit).contains("subir") && Arrays.asList(
    hypothesisSplit).contains("toda") && Arrays.asList(hypothesisSplit).contains("camilla")) {
184                 UnifiedTransmissionProtocol.modulesList.get(3).getDevicesList().get(0).
    setData((byte)3);
185                 UnifiedTransmissionProtocol.Module.syncAllDevicesValueToSystem();
186                 break outerloop;
187             }
188 
189             if (Arrays.asList(hypothesisSplit).contains("bajar") && Arrays.asList(
    hypothesisSplit).contains("toda") && Arrays.asList(hypothesisSplit).contains("camilla")) {
190                 UnifiedTransmissionProtocol.modulesList.get(3).getDevicesList().get(0).
    setData((byte)0);
191                 UnifiedTransmissionProtocol.Module.syncAllDevicesValueToSystem();
192                 break outerloop;
193             }
194 
195 
196             if (Arrays.asList(hypothesisSplit).contains("encender") && Arrays.asList(
    hypothesisSplit).contains("rele")) {
197                 for (String word : hypothesisSplit) {
198                     if (StringUtils.isNumeric(word) == true) {
199                         int number = Integer.parseInt(word);
200                         if (number<4) {
201                             UnifiedTransmissionProtocol.modulesList.get(2).getDevicesList()
    .get(number).setData(Boolean.TRUE);
202                             UnifiedTransmissionProtocol.Module.syncAllDevicesValueToSystem(
    );
203                         }
204                         break outerloop;
205                     }
206                 }
207             }
208 
209             if (Arrays.asList(hypothesisSplit).contains("apagar") && Arrays.asList(
    hypothesisSplit).contains("rele")) {
210                 for (String word : hypothesisSplit) {
```

```
211                     if (StringUtils.isNumeric(word) == true) {
212                         int number = Integer.parseInt(word);
213                         if (number<4) {
214                             UnifiedTransmissionProtocol.modulesList.get(2).getDevicesList()
    .get(number).setData(Boolean.FALSE);
215                             UnifiedTransmissionProtocol.Module.syncAllDevicesValueToSystem(
    );
216                         }
217                         break outerloop;
218                     }
219                 }
220             }
221 
222             if (Arrays.asList(hypothesisSplit).contains("decir") && Arrays.asList(
    hypothesisSplit).contains("hora")){
223                 Date currentTime = Calendar.getInstance().getTime();
224                 SimpleDateFormat df = new SimpleDateFormat("d-MMMM-yyyy", new Locale("spa")
    );
225                 String todayAsString = df.format(currentTime);
226                 SimpleDateFormat onlyTimeFormar = new SimpleDateFormat("HH:mm:ss", new
    Locale("spa"));
227                 String onlyTimeString = onlyTimeFormar.format(currentTime);
228 
229                 //TextToSpeechHandler.initializeTextToSpeech();
230                 TextToSpeechHandler.addToSpeakingQueue("Hoy es " + todayAsString + "y son
    las" + onlyTimeString);
231                 Application.toastMessage("Decir hora");
232             }
233 
234         }
235     }
236 
237     @Override
238     public void onReadyForSpeech(Bundle params) {
239         MicrophoneHandler.setState(MicrophoneHandler.MicrophoneState.LISTENING_COMMAND);
240     }
241 
242     @Override
243     public void onBeginningOfSpeech() {
244 
245     }
246 
247     @Override
248     public void onRmsChanged(float rmsdB) {
249 
250     }
251 
252     @Override
253     public void onBufferReceived(byte[] buffer) {
254 
255     }
256 
257     @Override
258     public void onEndOfSpeech() {
259 
260         //Application.toastMessage("End of speech");
261     }
262 
263     @Override
264     public void onError(int error) {
265         //Application.toastMessage("On error" + error);
266 
267         if (forceCancel==false) {
```

```
268             PocketsphinxListener.startListening();
269             MicrophoneHandler.setState(MicrophoneHandler.MicrophoneState.LISTENING_KEYOWRD)
    ;
270         }
271     }
272 
273     @Override
274     public void onResults(Bundle results) {
275         String str = new String();
276         ArrayList<String> resultsArrayList = results.getStringArrayList(SpeechRecognizer.
    RESULTS_RECOGNITION);
277         float[] confidenceScore = results.getFloatArray(SpeechRecognizer.CONFIDENCE_SCORES)
    ;
278         analiseListenerResults(resultsArrayList, confidenceScore);
279 
280         StringBuilder builder = new StringBuilder();
281         for (String line : resultsArrayList) {
282             builder.append(line + "\n");
283         }
284 
285         MainActivity.textView_speechRecognitionResults.setText(builder.toString());
286         MainActivity.textView_speechRecognitionResults.setVisibility(View.VISIBLE);
287 
288 
289 
290         MicrophoneHandler.setState(MicrophoneHandler.MicrophoneState.LISTENING_KEYOWRD);
291 
292         PocketsphinxListener.startListening();
293     }
294 
295     @Override
296     public void onPartialResults(Bundle partialResults) {
297         String str = new String();
298         ArrayList<String> resultsArrayList = partialResults.getStringArrayList(
    SpeechRecognizer.RESULTS_RECOGNITION);
299         float[] confidenceScore = partialResults.getFloatArray(SpeechRecognizer.
    CONFIDENCE_SCORES);
300 
301         StringBuilder builder = new StringBuilder();
302         for (String line : resultsArrayList) {
303             builder.append(line + "\n");
304         }
305 
306     }
307 
308     @Override
309     public void onEvent(int eventType, Bundle params) {
310 
311     }
312 }
313 
```

Archivo - CommandItem.java

```java
package com.example.SAPLM.commandActivity;

import com.example.SAPLM.R;
import com.example.SAPLM.bluetoothActivities.UnifiedTransmissionProtocol;

import java.util.List;

public class CommandItem {
    public enum ItemType { CHILD, PARENT}
    public enum BoardGroupState {FOLDED, UNFOLDED}

    private String itemName;
    public ItemType itemType = ItemType.CHILD;
    private int deviceStateIconID;
    private int deviceStateInteger = 0;
    private int moduleIndex = 0;
    private int deviceIndex = 0;
    private List<String> childInternalID;
    private BoardGroupState groupState = BoardGroupState.UNFOLDED;



    public CommandItem( int moduleIndex, int deviceIndex ){
        this.itemName = UnifiedTransmissionProtocol.modulesList.get(moduleIndex).
getDevicesList().get(deviceIndex).getName();
        this.itemType = ItemType.CHILD;
        this.moduleIndex = moduleIndex;
        this.deviceIndex = deviceIndex;
        this.deviceStateIconID = UnifiedTransmissionProtocol.modulesList.get(moduleIndex).
getDevicesList().get(deviceIndex).getIconResourceId();
    }

    public CommandItem(int boardInternalId){
        this.itemName = UnifiedTransmissionProtocol.modulesList.get(boardInternalId).
getModuleName();
        this.itemType = ItemType.PARENT;
        this.deviceStateIconID = UnifiedTransmissionProtocol.modulesList.get(boardInternalId
).getIconResourceId();
    }

    public void onClickHandler(){
        UnifiedTransmissionProtocol.modulesList.get(moduleIndex).getDevicesList().get(
deviceIndex).deviceOnClickHandler();
    }


    //Getters

    public BoardGroupState getGroupState (){
        return groupState;
    }

    public int getDeviceStateIconID() {
        return deviceStateIconID;
    }

    public int getDeviceStateInteger() {
        return deviceStateInteger;
    }

    public int getDeviceIndex() {
        return deviceIndex;
    }
```

```
 59
 60     public String getItemName() {
 61         return itemName;
 62     }
 63
 64     public int getModuleIndex(){
 65         return moduleIndex;
 66     }
 67
 68     public List<String> getChildInternalID() {
 69         return childInternalID;
 70     }
 71
 72     public ItemType getItemType() {
 73         return itemType;
 74     }
 75
 76
 77     public String getDeviceStateSubtitle(){
 78         return UnifiedTransmissionProtocol.modulesList.get(moduleIndex).getDevicesList().
    get(deviceIndex).getDataAsString();
 79     }
 80
 81     public int getProgressBarValue(){
 82         return UnifiedTransmissionProtocol.modulesList.get(moduleIndex).getDevicesList().
    get(deviceIndex).getProgressBarValue();
 83     }
 84
 85     // Setters
 86     public void setChildInternalID(List<String> childInternalID) {
 87         this.childInternalID = childInternalID;
 88     }
 89
 90
 91     public void setDeviceStateIconID(int deviceStateIconID) {
 92         this.deviceStateIconID = deviceStateIconID;
 93     }
 94
 95     public void setDeviceStateInteger(int deviceStateInteger) {
 96         this.deviceStateInteger = deviceStateInteger;
 97     }
 98
 99     public void setModuleIndex(int moduleIndex) {
100         this.moduleIndex = moduleIndex;
101     }
102
103     public void setItemName(String itemName) {
104         this.itemName = itemName;
105     }
106
107     public void setItemType(ItemType itemType) {
108         this.itemType = itemType;
109     }
110
111     public void setDeviceIndex(int deviceIndex) {
112         this.deviceIndex = deviceIndex;
113     }
114
115     public void setGroupState (BoardGroupState groupState){
116         this.groupState = groupState;
117     }
118
119 }
```

Archivo - CommandsActivity.java

```java
package com.example.SAPLM.commandActivity;

import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.graphics.PorterDuff;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import androidx.core.content.ContextCompat;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.recyclerview.widget.DefaultItemAnimator;
import androidx.recyclerview.widget.DividerItemDecoration;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.os.Looper;
import android.view.MenuItem;
import android.view.View;

import com.example.SAPLM.Application;
import com.example.SAPLM.R;
import com.example.SAPLM.bluetoothActivities.BluetoothConnection;
import com.example.SAPLM.settingsActivities.AvailableSetting;
import com.example.SAPLM.settingsActivities.RecyclerTouchListener;
import com.example.SAPLM.settingsActivities.SettingsLayoutAdapter;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class CommandsActivity extends AppCompatActivity {
    private static Toolbar toolbar;

    public static List<CommandItem> commandList = new ArrayList<>();
    private RecyclerView recyclerView;
    public static CommandsListAdapter mAdapter;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.commands_layout);
        Application.setActivity(this);
        Application.setContext(this);

        toolbar = this.findViewById(R.id.toolbar);

        setSupportActionBar(toolbar);

        // add back arrow to toolbar
        if (getSupportActionBar() != null){
            getSupportActionBar().setDisplayHomeAsUpEnabled(true);
            getSupportActionBar().setDisplayShowHomeEnabled(true);

            final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.
abc_ic_ab_back_material);
            upArrow.setColorFilter(getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP)
;
            getSupportActionBar().setHomeAsUpIndicator(upArrow);
        }

        recyclerView = (RecyclerView) findViewById(R.id.commandsRecyclerView);
```

```
62
63          mAdapter = new CommandsListAdapter(commandList);
64
65          recyclerView.setHasFixedSize(true);
66
67          // vertical RecyclerView
68          // keep movie_list_row.xml width to `match_parent`
69          RecyclerView.LayoutManager mLayoutManager = new LinearLayoutManager(
    getApplicationContext());
70
71          // horizontal RecyclerView
72          // keep movie_list_row.xml width to `wrap_content`
73          // RecyclerView.LayoutManager mLayoutManager = new LinearLayoutManager(
    getApplicationContext(), LinearLayoutManager.HORIZONTAL, false);
74
75          recyclerView.setLayoutManager(mLayoutManager);
76
77          // adding inbuilt divider line
78          recyclerView.addItemDecoration(new DividerItemDecoration(this, LinearLayoutManager.
    VERTICAL));
79
80          // adding custom divider line with padding 16dp
81          //recyclerView.addItemDecoration(new MyDividerItemDecoration(this,
    LinearLayoutManager.HORIZONTAL, 16));
82          recyclerView.setItemAnimator(new DefaultItemAnimator());
83
84          recyclerView.setAdapter(mAdapter);
85
86          // row click listener
87          recyclerView.addOnItemTouchListener(new RecyclerTouchListener(getApplicationContext
    (), recyclerView, new RecyclerTouchListener.ClickListener() {
88              @Override
89              public void onClick(View view, int position) {
90                  CommandItem commandItem = commandList.get(position);
91                  if (commandItem.getItemType()== CommandItem.ItemType.PARENT){
92
93                  }else{
94                      commandItem.onClickHandler();
95                  }
96
97
98                  mAdapter.notifyDataSetChanged();
99
100             }
101
102             @Override
103             public void onLongClick(View view, int position) {
104
105             }
106         }));
107
108         if (commandList.isEmpty()) setupCommandsList();
109
110         mAdapter.notifyDataSetChanged();
111     }
112
113
114     public static void updateGUI(){
115         if (mAdapter!=null) {
116             if (Looper.myLooper() == Looper.getMainLooper()) {
117                 mAdapter.notifyDataSetChanged();
118             } else {
119                 Application.getActivity().runOnUiThread(new Runnable() {
```

```
120                     @Override
121                     public void run() {
122                         mAdapter.notifyDataSetChanged();
123                     }
124                 });
125             }
126         }
127
128     }
129
130     @Override
131     public boolean onOptionsItemSelected(MenuItem item) {
132         if (item.getItemId() == android.R.id.home) // Press Back Icon
133         {
134             finish();
135         }
136
137         return super.onOptionsItemSelected(item);
138     }
139
140     public static void setupCommandsList(){
141         //commandList.add(new CommandItem("MODULO PRINCIPAL","MAIN_BOARD", new ArrayList(
    Arrays.asList("LED"))));
142
143         commandList.add(new CommandItem(1));
144         commandList.add(new CommandItem(1,0));
145         commandList.add(new CommandItem(1, 1));
146
147         commandList.add(new CommandItem(2));
148         commandList.add(new CommandItem(2,0));
149         commandList.add(new CommandItem(2, 1));
150         commandList.add(new CommandItem(2, 2));
151         commandList.add(new CommandItem(2, 3));
152
153         commandList.add(new CommandItem(3));
154         commandList.add(new CommandItem(3,0));
155         commandList.add(new CommandItem(3, 1));
156         commandList.add(new CommandItem(3, 2));
157     }
158 }
159
```

Archivo - CommandsListAdapter.java

```java
package com.example.SAPLM.commandActivity;

import android.graphics.Color;
import android.graphics.PorterDuff;
import android.graphics.Typeface;
import android.graphics.drawable.ColorDrawable;
import android.graphics.drawable.Drawable;

import androidx.constraintlayout.widget.ConstraintLayout;
import androidx.constraintlayout.widget.Guideline;
import androidx.recyclerview.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.ProgressBar;
import android.widget.TextView;

import com.example.SAPLM.Application;
import com.example.SAPLM.R;
import com.example.SAPLM.settingsActivities.AvailableSetting;
import com.example.SAPLM.settingsActivities.SettingsLayoutAdapter;

import java.util.List;

public class CommandsListAdapter extends RecyclerView.Adapter<CommandsListAdapter.viewHolder
> {

    private List<CommandItem> commandsList;

    public class viewHolder extends RecyclerView.ViewHolder {
        public TextView command_title, command_state_text;
        public ImageView command_icon, arrow_list;
        public ProgressBar status_progress_bar;
        public Guideline main_guide;

        public viewHolder(View view) {
            super(view);
            command_title = (TextView) view.findViewById(R.id.command_title_textView);
            status_progress_bar = (ProgressBar) view.findViewById(R.id.
command_state_progressBar);
            command_state_text = (TextView) view.findViewById(R.id.command_state_textView);
            command_icon = (ImageView) view.findViewById(R.id.command_icon);
            arrow_list = (ImageView) view.findViewById(R.id.arrow_list_imageButton);
            main_guide = (Guideline) view.findViewById(R.id.guideline_main);
        }
    }


    public CommandsListAdapter(List<CommandItem> cmdList) {
        this.commandsList = cmdList;
    }

    @Override
    public viewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View itemView = LayoutInflater.from(parent.getContext())
                .inflate(R.layout.commands_layout_child_item_style, parent, false);

        return new viewHolder(itemView);
    }

    @Override
    public void onBindViewHolder(viewHolder holder, int position) {
```

```
62          CommandItem item = commandsList.get(position);
63          holder.command_title.setText(item.getItemName());
64
65          if (item.getItemType()== CommandItem.ItemType.PARENT){
66              holder.arrow_list.setVisibility(View.VISIBLE);
67              if (item.getGroupState()== CommandItem.BoardGroupState.UNFOLDED)
68              {
69                  Drawable icon = Application.getContext().getDrawable(R.drawable.
    icons_downward_arrow);
70                  icon.setColorFilter(Application.getContext().getColor(R.color.colorPrimary)
    , PorterDuff.Mode.SRC_ATOP);
71                  holder.arrow_list.setImageDrawable(icon);
72              }else {
73                  Drawable icon = Application.getContext().getDrawable(R.drawable.
    icons_upward_arrow);
74                  icon.setColorFilter(Application.getContext().getColor(R.color.colorPrimary)
    , PorterDuff.Mode.SRC_ATOP);
75                  holder.arrow_list.setImageDrawable(icon);
76              }
77              ConstraintLayout.LayoutParams params = (ConstraintLayout.LayoutParams) holder.
    main_guide.getLayoutParams();
78              params.guidePercent = 0.05f;
79              holder.main_guide.setLayoutParams(params);
80              holder.status_progress_bar.setVisibility(View.GONE);
81              holder.command_state_text.setVisibility(View.GONE);
82              holder.command_title.setTypeface(holder.command_title.getTypeface(), Typeface.
    BOLD);
83          }else {
84              ConstraintLayout.LayoutParams params = (ConstraintLayout.LayoutParams) holder.
    main_guide.getLayoutParams();
85              holder.command_state_text.setText(item.getDeviceStateSubtitle());
86              params.guidePercent = 0.15f;
87              holder.main_guide.setLayoutParams(params);
88              holder.arrow_list.setVisibility(View.GONE);
89              holder.command_state_text.setText(item.getDeviceStateSubtitle());
90              holder.command_state_text.setVisibility(View.VISIBLE);
91              holder.status_progress_bar.setIndeterminate(false);
92              holder.status_progress_bar.setProgress(item.getProgressBarValue());
93              holder.status_progress_bar.setVisibility(View.VISIBLE);
94
95          }
96
97
98          Drawable cmdIcon = Application.getContext().getDrawable(item.getDeviceStateIconID()
    );
99          cmdIcon.setColorFilter(Application.getContext().getColor(R.color.colorPrimary),
    PorterDuff.Mode.SRC_ATOP);
100         holder.command_icon.setImageDrawable(cmdIcon);
101     }
102
103     @Override
104     public int getItemCount() {
105         return commandsList.size();
106     }
107 }
```

Archivo - settings_layout.java

```java
package com.example.SAPLM.settingsActivities;

import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.graphics.PorterDuff;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import androidx.core.content.ContextCompat;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.DefaultItemAnimator;
import androidx.recyclerview.widget.DividerItemDecoration;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
import androidx.appcompat.widget.Toolbar;
import android.view.MenuItem;
import android.view.View;

import com.example.SAPLM.Application;
import com.example.SAPLM.R;

import java.util.ArrayList;
import java.util.List;

public class settings_layout extends AppCompatActivity {

    private static Toolbar toolbar;

    public static List<AvailableSetting> settingsList = new ArrayList<>();
    private RecyclerView recyclerView;
    public static SettingsLayoutAdapter mAdapter;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.settings_layout);
        Application.setActivity(this);
        Application.setContext(this);

        toolbar = findViewById(R.id.toolbar);

        setSupportActionBar(toolbar);

        // add back arrow to toolbar
        if (getSupportActionBar() != null){
            getSupportActionBar().setDisplayHomeAsUpEnabled(true);
            getSupportActionBar().setDisplayShowHomeEnabled(true);


            final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.
abc_ic_ab_back_material);
            upArrow.setColorFilter(getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP)
;
            getSupportActionBar().setHomeAsUpIndicator(upArrow);
        }

        recyclerView = (RecyclerView) findViewById(R.id.settingsRecyclerView);

        mAdapter = new SettingsLayoutAdapter(settingsList);

        recyclerView.setHasFixedSize(true);

```

```
62          // vertical RecyclerView
63          // keep movie_list_row.xml width to `match_parent`
64          RecyclerView.LayoutManager mLayoutManager = new LinearLayoutManager(
    getApplicationContext());
65
66          // horizontal RecyclerView
67          // keep movie_list_row.xml width to `wrap_content`
68          // RecyclerView.LayoutManager mLayoutManager = new LinearLayoutManager(
    getApplicationContext(), LinearLayoutManager.HORIZONTAL, false);
69
70          recyclerView.setLayoutManager(mLayoutManager);
71
72          // adding inbuilt divider line
73          recyclerView.addItemDecoration(new DividerItemDecoration(this, LinearLayoutManager.
    VERTICAL));
74
75          // adding custom divider line with padding 16dp
76          //recyclerView.addItemDecoration(new MyDividerItemDecoration(this,
    LinearLayoutManager.HORIZONTAL, 16));
77          recyclerView.setItemAnimator(new DefaultItemAnimator());
78
79          recyclerView.setAdapter(mAdapter);
80
81          // row click listener
82          recyclerView.addOnItemTouchListener(new RecyclerTouchListener(getApplicationContext
    (), recyclerView, new RecyclerTouchListener.ClickListener() {
83              @Override
84              public void onClick(View view, int position) {
85                  AvailableSetting setting = settingsList.get(position);
86
87                  Context currentContext = Application.getContext();
88                  Activity currentActivity = Application.getActivity();
89                  Intent i = new Intent(currentContext, setting.getTargetActivity());
90                  currentContext.startActivity(i);
91                  //currentActivity.overridePendingTransition(R.anim.fade_in, R.anim.fade_out
    );
92              }
93
94              @Override
95              public void onLongClick(View view, int position) {
96
97              }
98          }));
99
100         mAdapter.notifyDataSetChanged();
101
102     }
103
104     @Override
105     public boolean onOptionsItemSelected(MenuItem item) {
106         if (item.getItemId() == android.R.id.home) // Press Back Icon
107         {
108             finish();
109         }
110
111         return super.onOptionsItemSelected(item);
112     }
113
114     public static void setupSettingsList() {
115
116         settingsList.add(new AvailableSetting(
117             Application.getContext().getString(R.string.internet_conectivity_title),
118                 Application.getContext().getString(R.string.internet_conectivity_subtitle),
```

```
119             R.drawable.ic_setting_internet_connectivity,
120             InternetConectivityActivity.class
121         ));
122 
123         settingsList.add(new AvailableSetting(
124                 Application.getContext().getString(R.string.bluetooth_settings_title),
125                 Application.getContext().getString(R.string.bluetooth_settings_subtitle),
126                 R.drawable.ic_setting_bluetooth,
127                 BluetoothSettingsActivity.class
128         ));
129 
130         settingsList.add(new AvailableSetting(
131                 Application.getContext().getString(R.string.
    voice_recognition_settings_title),
132                 Application.getContext().getString(R.string.
    voice_recognition_settings_subtitle),
133                 R.drawable.ic_setting_voice_recognition,
134                 VoiceRecognitionSettingsActivity.class
135         ));
136 
137         settingsList.add(new AvailableSetting(
138                 Application.getContext().getString(R.string.commands_settings_title),
139                 Application.getContext().getString(R.string.commands_settings_subtitle),
140                 R.drawable.ic_setting_commands_3,
141                 CommandsSettingsActivity.class
142         ));
143 
144         settingsList.add(new AvailableSetting(
145                 Application.getContext().getString(R.string.contacts_communication_title),
146                 Application.getContext().getString(R.string.contacts_communication_subtitle
    ),
147                 R.drawable.ic_setting_contacts_and_communication,
148                 ContactsCommunicationActivity.class
149         ));
150 
151         settingsList.add(new AvailableSetting(
152                 Application.getContext().getString(R.string.appearance_miscellaneous_title)
    ,
153                 Application.getContext().getString(R.string.
    appearance_miscellaneous_subtitle),
154                 R.drawable.ic_setting_appearance_and_miscellaneous,
155                 AppearanceMiscellaneousActivity.class
156         ));
157 
158         settingsList.add(new AvailableSetting(
159                 Application.getContext().getString(R.string.
    developer_console_activity_title),
160                 Application.getContext().getString(R.string.developer_console_subtitle),
161                 R.drawable.ic_setting_developer_console,
162                 DeveloperConsoleActivity.class
163         ));
164 
165         settingsList.add(new AvailableSetting(
166                 Application.getContext().getString(R.string.information_help_title),
167                 Application.getContext().getString(R.string.information_help_subtitle),
168                 R.drawable.ic_setting_information_and_help,
169                 InformationHelpActivity.class
170         ));
171 
172         // notify adapter about data set changes
173         // so that it will render the list with new data
174         //mAdapter.notifyDataSetChanged();
175     }
```

```
176
177
178 }
179
```

Archivo - TerminalManager.java

```java
package com.example.SAPLM.settingsActivities;

import android.graphics.Typeface;
import android.text.SpannableStringBuilder;
import android.text.Spanned;
import android.text.style.StyleSpan;

import com.example.SAPLM.bluetoothActivities.UnifiedTransmissionProtocol;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.regex.Matcher;
import java.util.regex.Pattern;


public class TerminalManager {

    public static void handleTerminalInput(String newTerminalInput){

    }

    public static List<String> currentParameters = new ArrayList<>();

    public static List<TerminalCommand> commandsList = new ArrayList<>(Arrays.asList(
            new TerminalCommand("COMPOSE", new Runnable() {
                @Override
                public void run() {
                    UnifiedTransmissionProtocol.CommandType command = UnifiedTransmissionProtocol.CommandType.definedCommandList.stream().filter(p-> p.getCommandTag().equals((currentParameters.get(0)))).findFirst().orElse(null);
                    UnifiedTransmissionProtocol.Module target = UnifiedTransmissionProtocol.modulesList.stream().filter(p-> p.getModuleAlias().equals((currentParameters.get(1)))).findFirst().orElse(null);

                    if (command==null) {
                        SpannableStringBuilder builder= new SpannableStringBuilder();
                        builder.append("Tipo de comando \"")
                                .append(currentParameters.get(0), new StyleSpan(Typeface.BOLD), Spanned.SPAN_EXCLUSIVE_EXCLUSIVE)
                                .append("\" no reconocido");
                        DeveloperConsoleActivity.sendTextToTerminal(builder);
                        return;
                    }

                    if (target==null) {
                        SpannableStringBuilder builder= new SpannableStringBuilder();
                        builder.append("Alias de modulo \"")
                                .append(currentParameters.get(1), new StyleSpan(Typeface.BOLD), Spanned.SPAN_EXCLUSIVE_EXCLUSIVE)
                                .append("\" no encontrado");
                        DeveloperConsoleActivity.sendTextToTerminal(builder);
                        return;
                    }


                    System.out.println( insertSpaces(UnifiedTransmissionProtocol.bytesToHex( UnifiedTransmissionProtocol.getCommand_buffer() ), " ", 2) );
                }
            })
    ));

    public static class TerminalCommand{
```

```
57          private String commandBaseTag;
58          private Runnable commandHandler;
59
60          TerminalCommand(String commandBaseTag, Runnable commandHandler){
61              this.commandBaseTag = commandBaseTag;
62              this.commandHandler = commandHandler;
63          }
64
65          public void handleCommand(){
66
67          }
68      }
69
70      public static String insertSpaces(String text, String insert, int period) {
71          Pattern p = Pattern.compile("(.{" + period + "})", Pattern.DOTALL);
72          Matcher m = p.matcher(text);
73          return m.replaceAll("$1" + insert);
74      }
75  }
76
```

Archivo - AvailableSetting.java

```java
package com.example.SAPLM.settingsActivities;

public class AvailableSetting {
    private String title, subtitle;
    private int iconResourceID;
    private Class targetActivity;

    public AvailableSetting() {
    }

    public AvailableSetting(String title, String subtitle, int iconResourceID, Class targetActivity) {
        this.title = title;
        this.subtitle = subtitle;
        this.iconResourceID = iconResourceID;
        this.targetActivity = targetActivity;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String name) {
        this.title = name;
    }

    public String getSubtitle() {
        return subtitle;
    }

    public void setSubtitle(String subtitle) {
        this.subtitle = subtitle;
    }

    public int getIconResourceID() {
        return iconResourceID;
    }

    public void setIconResourceID(int iconResourceID) {
        this.iconResourceID = iconResourceID;
    }

    public Class getTargetActivity() { return targetActivity; }

    public void setTargetActivity(Class targetActivity) { this.targetActivity = targetActivity; }
}

```

Archivo - RecyclerTouchListener.java

```java
package com.example.SAPLM.settingsActivities;

import android.content.Context;
import androidx.recyclerview.widget.RecyclerView;
import android.view.GestureDetector;
import android.view.MotionEvent;
import android.view.View;

/**
 * Created by Ravi Tamada on 03/09/16.
 * updated by Ravi on 14/11/17
 * www.androidhive.info
 */
public class RecyclerTouchListener implements RecyclerView.OnItemTouchListener {

    private GestureDetector gestureDetector;
    private ClickListener clickListener;

    public RecyclerTouchListener(Context context, final RecyclerView recyclerView, final ClickListener clickListener) {
        this.clickListener = clickListener;
        gestureDetector = new GestureDetector(context, new GestureDetector.SimpleOnGestureListener() {
            @Override
            public boolean onSingleTapUp(MotionEvent e) {
                return true;
            }

            @Override
            public void onLongPress(MotionEvent e) {
                View child = recyclerView.findChildViewUnder(e.getX(), e.getY());
                if (child != null && clickListener != null) {
                    clickListener.onLongClick(child, recyclerView.getChildAdapterPosition(child));
                }
            }
        });
    }

    @Override
    public boolean onInterceptTouchEvent(RecyclerView rv, MotionEvent e) {

        View child = rv.findChildViewUnder(e.getX(), e.getY());
        if (child != null && clickListener != null && gestureDetector.onTouchEvent(e)) {
            clickListener.onClick(child, rv.getChildAdapterPosition(child));
        }
        return false;
    }

    @Override
    public void onTouchEvent(RecyclerView rv, MotionEvent e) {
    }

    @Override
    public void onRequestDisallowInterceptTouchEvent(boolean disallowIntercept) {

    }

    public interface ClickListener {
        void onClick(View view, int position);

        void onLongClick(View view, int position);
    }
```

```
61 }
```

Archivo - SettingsLayoutAdapter.java

```java
package com.example.SAPLM.settingsActivities;

import android.graphics.PorterDuff;
import android.graphics.drawable.Drawable;
import androidx.recyclerview.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;

import com.example.SAPLM.Application;
import com.example.SAPLM.R;

import java.util.List;

public class SettingsLayoutAdapter extends RecyclerView.Adapter<SettingsLayoutAdapter.viewHolder> {

    private List<AvailableSetting> settingsList;

    public class viewHolder extends RecyclerView.ViewHolder {
        public TextView setting_title, setting_subtitle;
        public ImageView setting_icon;

        public viewHolder(View view) {
            super(view);
            setting_title = (TextView) view.findViewById(R.id.setting_title);
            setting_subtitle = (TextView) view.findViewById(R.id.setting_subtitle);
            setting_icon = (ImageView) view.findViewById(R.id.setting_icon);
        }
    }


    public SettingsLayoutAdapter(List<AvailableSetting> settingsList) {
        this.settingsList = settingsList;
    }

    @Override
    public viewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View itemView = LayoutInflater.from(parent.getContext())
                .inflate(R.layout.settings_layout_style, parent, false);

        return new viewHolder(itemView);
    }

    @Override
    public void onBindViewHolder(viewHolder holder, int position) {
        AvailableSetting setting = settingsList.get(position);
        holder.setting_title.setText(setting.getTitle());
        holder.setting_subtitle.setText(setting.getSubtitle());

        Drawable icon = Application.getContext().getDrawable(setting.getIconResourceID());
        icon.setColorFilter(Application.getContext().getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP);
        holder.setting_icon.setImageDrawable(icon);
    }

    @Override
    public int getItemCount() {
        return settingsList.size();
    }
}
```

Archivo - InformationHelpActivity.java

```java
package com.example.SAPLM.settingsActivities;

import android.graphics.PorterDuff;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import androidx.core.content.ContextCompat;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import android.view.MenuItem;

import com.example.SAPLM.Application;
import com.example.SAPLM.R;

public class InformationHelpActivity extends AppCompatActivity {
    private static Toolbar toolbar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.information_help_layout);
        Application.setActivity(this);
        Application.setContext(this);

        toolbar = this.findViewById(R.id.toolbar);

        setSupportActionBar(toolbar);

        // add back arrow to toolbar
        if (getSupportActionBar() != null){
            getSupportActionBar().setDisplayHomeAsUpEnabled(true);
            getSupportActionBar().setDisplayShowHomeEnabled(true);

            final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.
abc_ic_ab_back_material);
            upArrow.setColorFilter(getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP)
;
            getSupportActionBar().setHomeAsUpIndicator(upArrow);
        }
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        if (item.getItemId() == android.R.id.home) // Press Back Icon
        {
            finish();
        }

        return super.onOptionsItemSelected(item);
    }
}

```

Archivo - MyDividerItemDecoration.java

```java
package com.example.SAPLM.settingsActivities;

import android.content.Context;
import android.content.res.Resources;
import android.content.res.TypedArray;
import android.graphics.Canvas;
import android.graphics.Rect;
import android.graphics.drawable.Drawable;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
import android.util.TypedValue;
import android.view.View;

/**
 * Created by Ravi on 30/10/15.
 * updated by Ravi on 14/11/17
 */
public class MyDividerItemDecoration extends RecyclerView.ItemDecoration {

    private static final int[] ATTRS = new int[]{
            android.R.attr.listDivider
    };

    public static final int HORIZONTAL_LIST = LinearLayoutManager.HORIZONTAL;

    public static final int VERTICAL_LIST = LinearLayoutManager.VERTICAL;

    private Drawable mDivider;
    private int mOrientation;
    private Context context;
    private int margin;

    public MyDividerItemDecoration(Context context, int orientation, int margin) {
        this.context = context;
        this.margin = margin;
        final TypedArray a = context.obtainStyledAttributes(ATTRS);
        mDivider = a.getDrawable(0);
        a.recycle();
        setOrientation(orientation);
    }

    public void setOrientation(int orientation) {
        if (orientation != HORIZONTAL_LIST && orientation != VERTICAL_LIST) {
            throw new IllegalArgumentException("invalid orientation");
        }
        mOrientation = orientation;
    }

    @Override
    public void onDrawOver(Canvas c, RecyclerView parent, RecyclerView.State state) {
        if (mOrientation == VERTICAL_LIST) {
            drawVertical(c, parent);
        } else {
            drawHorizontal(c, parent);
        }
    }

    public void drawVertical(Canvas c, RecyclerView parent) {
        final int left = parent.getPaddingLeft();
        final int right = parent.getWidth() - parent.getPaddingRight();

        final int childCount = parent.getChildCount();
        for (int i = 0; i < childCount; i++) {
```

Archivo - MyDividerItemDecoration.java

```
64              final View child = parent.getChildAt(i);
65              final RecyclerView.LayoutParams params = (RecyclerView.LayoutParams) child
66                      .getLayoutParams();
67              final int top = child.getBottom() + params.bottomMargin;
68              final int bottom = top + mDivider.getIntrinsicHeight();
69              mDivider.setBounds(left + dpToPx(margin), top, right - dpToPx(margin), bottom);
70              mDivider.draw(c);
71          }
72      }
73
74      public void drawHorizontal(Canvas c, RecyclerView parent) {
75          final int top = parent.getPaddingTop();
76          final int bottom = parent.getHeight() - parent.getPaddingBottom();
77
78          final int childCount = parent.getChildCount();
79          for (int i = 0; i < childCount; i++) {
80              final View child = parent.getChildAt(i);
81              final RecyclerView.LayoutParams params = (RecyclerView.LayoutParams) child
82                      .getLayoutParams();
83              final int left = child.getRight() + params.rightMargin;
84              final int right = left + mDivider.getIntrinsicHeight();
85              mDivider.setBounds(left, top + dpToPx(margin), right, bottom - dpToPx(margin));
86              mDivider.draw(c);
87          }
88      }
89
90      @Override
91      public void getItemOffsets(Rect outRect, View view, RecyclerView parent, RecyclerView.
    State state) {
92          if (mOrientation == VERTICAL_LIST) {
93              outRect.set(0, 0, 0, mDivider.getIntrinsicHeight());
94          } else {
95              outRect.set(0, 0, mDivider.getIntrinsicWidth(), 0);
96          }
97      }
98
99      private int dpToPx(int dp) {
100         Resources r = context.getResources();
101         return Math.round(TypedValue.applyDimension(TypedValue.COMPLEX_UNIT_DIP, dp, r.
    getDisplayMetrics()));
102     }
103 }
```

Archivo - CommandsSettingsActivity.java

```java
package com.example.SAPLM.settingsActivities;

import android.graphics.PorterDuff;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import androidx.core.content.ContextCompat;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import android.view.MenuItem;

import com.example.SAPLM.Application;
import com.example.SAPLM.R;

public class CommandsSettingsActivity extends AppCompatActivity {
    private static Toolbar toolbar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.commands_settings_layout);
        Application.setActivity(this);
        Application.setContext(this);

        toolbar = this.findViewById(R.id.toolbar);

        setSupportActionBar(toolbar);

        // add back arrow to toolbar
        if (getSupportActionBar() != null){
            getSupportActionBar().setDisplayHomeAsUpEnabled(true);
            getSupportActionBar().setDisplayShowHomeEnabled(true);

            final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.abc_ic_ab_back_material);
            upArrow.setColorFilter(getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP);
            getSupportActionBar().setHomeAsUpIndicator(upArrow);
        }
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        if (item.getItemId() == android.R.id.home) // Press Back Icon
        {
            finish();
        }

        return super.onOptionsItemSelected(item);
    }
}

```

Archivo - DeveloperConsoleActivity.java

```java
package com.example.SAPLM.settingsActivities;

import android.graphics.Color;
import android.graphics.PorterDuff;
import android.graphics.Typeface;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import androidx.core.content.ContextCompat;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;

import android.os.Looper;
import android.text.Html;
import android.text.Spanned;
import android.text.SpannedString;
import android.util.TypedValue;
import android.view.MenuItem;
import android.view.View;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.LinearLayout;
import android.widget.ScrollView;
import android.widget.TextView;

import com.example.SAPLM.Application;
import com.example.SAPLM.R;

public class DeveloperConsoleActivity extends AppCompatActivity {
    private static Toolbar toolbar;
    private static ImageButton enterButton;
    private static ScrollView scrollViewTerminal;
    private static EditText editTextTerminal;
    private static LinearLayout terminalLinearLayout;
    private static TextView mainTextView;

    private static final String mainText_KEY = "MAIN_TEXT_KEY";
    private static final String ongoingText_KEY = "ONGOING_TEXT_KEY";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.developer_console_layout);
        Application.setActivity(this);
        Application.setContext(this);

        toolbar = this.findViewById(R.id.toolbar);
        enterButton = this.findViewById(R.id.enterButton);
        scrollViewTerminal = this.findViewById(R.id.scrollViewTerminal);
        editTextTerminal = this.findViewById(R.id.editTextTerminal);
        terminalLinearLayout = this.findViewById(R.id.terminalLinearLayout);




        setSupportActionBar(toolbar);

        // add back arrow to toolbar
        if (getSupportActionBar() != null){
            getSupportActionBar().setDisplayHomeAsUpEnabled(true);
            getSupportActionBar().setDisplayShowHomeEnabled(true);

            final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.
abc_ic_ab_back_material);
```

```
63              upArrow.setColorFilter(getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP
    );
64              getSupportActionBar().setHomeAsUpIndicator(upArrow);
65          }
66
67          mainTextView = new TextView(this);
68          mainTextView.setHint("SAPLM TERMINAL");
69          mainTextView.setTextColor(Color.BLACK);
70          mainTextView.setTextSize(TypedValue.COMPLEX_UNIT_SP, 54.0f /getResources().
    getDisplayMetrics().density);
71          mainTextView.setLayoutParams(new LinearLayout.LayoutParams(LinearLayout.
    LayoutParams.WRAP_CONTENT, LinearLayout.LayoutParams.WRAP_CONTENT));
72          mainTextView.setTypeface(Typeface.MONOSPACE);
73          terminalLinearLayout.addView(mainTextView);
74
75
76
77          scrollViewTerminal.fullScroll(View.FOCUS_DOWN);
78
79      }
80
81
82
83      @Override
84      protected void onPause() {
85          if(mainTextView!=null) {
86              try {
87                  Application.persistentDataSave.putString(ongoingText_KEY, editTextTerminal.
    getText().toString());
88              }catch (Throwable th){}
89          }
90          super.onPause();
91      }
92
93      @Override
94      protected void onResume() {
95          if (Application.persistentDataSave.containsKey(mainText_KEY)&&mainTextView!=null) {
96              try {
97                  mainTextView.setText(Html.fromHtml(Application.persistentDataSave.getString
    (mainText_KEY), Html.FROM_HTML_MODE_LEGACY), TextView.BufferType.SPANNABLE);
98                  editTextTerminal.setText(Application.persistentDataSave.getString(
    ongoingText_KEY));
99                  scrollViewTerminal.fullScroll(View.FOCUS_DOWN);
100             }catch (Throwable th){}
101         }
102         super.onResume();
103     }
104
105     @Override
106     public boolean onOptionsItemSelected(MenuItem item) {
107         if (item.getItemId() == android.R.id.home) // Press Back Icon
108         {
109             finish();
110         }
111
112         return super.onOptionsItemSelected(item);
113     }
114
115     public static void sendTextToTerminal(Spanned newText){
116
117         String newString = Application.persistentDataSave.getString(mainText_KEY) + Html.
    toHtml(newText, Html.FROM_HTML_MODE_LEGACY);
118         Application.persistentDataSave.putString(mainText_KEY,newString);
```

```
119
120         if (Looper.myLooper() == Looper.getMainLooper()){
121             mainTextView.setText(Html.fromHtml(newString, Html.FROM_HTML_MODE_LEGACY),
    TextView.BufferType.SPANNABLE);
122             scrollViewTerminal.fullScroll(View.FOCUS_DOWN);
123         } else {
124             Application.getActivity().runOnUiThread(new Runnable() {
125                 @Override
126                 public void run() {
127                     mainTextView.setText(Html.fromHtml(Application.persistentDataSave.
    getString(mainText_KEY), Html.FROM_HTML_MODE_LEGACY), TextView.BufferType.SPANNABLE);
128                     scrollViewTerminal.fullScroll(View.FOCUS_DOWN);
129                 }
130             });
131         }
132     }
133
134     public static void sendTextToTerminal(String newText){
135
136         Spanned spannedConverted = Html.fromHtml(newText,Html.FROM_HTML_MODE_LEGACY);
137
138         String newString = Application.persistentDataSave.getString(mainText_KEY) + Html.
    toHtml(spannedConverted, Html.FROM_HTML_MODE_LEGACY);
139         Application.persistentDataSave.putString(mainText_KEY,newString);
140
141         if (mainTextView!=null) {
142             if (Looper.myLooper() == Looper.getMainLooper()) {
143                 mainTextView.setText(Html.fromHtml(newString, Html.FROM_HTML_MODE_LEGACY),
    TextView.BufferType.SPANNABLE);
144                 scrollViewTerminal.fullScroll(View.FOCUS_DOWN);
145             } else {
146                 Application.getActivity().runOnUiThread(new Runnable() {
147                     @Override
148                     public void run() {
149                         mainTextView.setText(Html.fromHtml(Application.persistentDataSave.
    getString(mainText_KEY), Html.FROM_HTML_MODE_LEGACY), TextView.BufferType.SPANNABLE);
150                         scrollViewTerminal.fullScroll(View.FOCUS_DOWN);
151                     }
152                 });
153             }
154         }
155     }
156
157
158
159     public static void clearTerminalText(){
160         Application.persistentDataSave.putString(mainText_KEY,"");
161
162         if (Looper.myLooper() == Looper.getMainLooper()){
163             mainTextView.setText("");
164             scrollViewTerminal.fullScroll(View.FOCUS_DOWN);
165         } else {
166             Application.getActivity().runOnUiThread(new Runnable() {
167                 @Override
168                 public void run() {
169                     mainTextView.setText("");
170                     scrollViewTerminal.fullScroll(View.FOCUS_DOWN);
171                 }
172             });
173         }
174     }
175
176
```

Archivo - DeveloperConsoleActivity.java

```
177     public void onClickEnterButton(View view){
178         sendTextToTerminal(Html.fromHtml(editTextTerminal.getText().toString(),Html.
    FROM_HTML_MODE_LEGACY));
179         TerminalManager.handleTerminalInput(editTextTerminal.getText().toString());
180     }
181 
182     public void onClickClearButton(View view){
183         clearTerminalText();
184     }
185 
186 
187 }
188 
```

Archivo - BluetoothSettingsActivity.java

```java
package com.example.SAPLM.settingsActivities;

import android.graphics.PorterDuff;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import androidx.core.content.ContextCompat;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import android.view.MenuItem;

import com.example.SAPLM.Application;
import com.example.SAPLM.R;

public class BluetoothSettingsActivity extends AppCompatActivity {
    private static Toolbar toolbar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.bluetooth_settings_layout);
        Application.setActivity(this);
        Application.setContext(this);

        toolbar = this.findViewById(R.id.toolbar);

        setSupportActionBar(toolbar);

        // add back arrow to toolbar
        if (getSupportActionBar() != null){
            getSupportActionBar().setDisplayHomeAsUpEnabled(true);
            getSupportActionBar().setDisplayShowHomeEnabled(true);

            final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.
abc_ic_ab_back_material);
            upArrow.setColorFilter(getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP)
;
            getSupportActionBar().setHomeAsUpIndicator(upArrow);
        }
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        if (item.getItemId() == android.R.id.home) // Press Back Icon
        {
            finish();
        }

        return super.onOptionsItemSelected(item);
    }
}

```

Archivo - InternetConectivityActivity.java

```java
package com.example.SAPLM.settingsActivities;

import android.graphics.PorterDuff;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import androidx.core.content.ContextCompat;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import android.view.MenuItem;

import com.example.SAPLM.Application;
import com.example.SAPLM.R;

public class InternetConectivityActivity extends AppCompatActivity {
    private static Toolbar toolbar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.internet_conectivity_layout);
        Application.setActivity(this);
        Application.setContext(this);

        toolbar = this.findViewById(R.id.toolbar);

        setSupportActionBar(toolbar);

        // add back arrow to toolbar
        if (getSupportActionBar() != null){
            getSupportActionBar().setDisplayHomeAsUpEnabled(true);
            getSupportActionBar().setDisplayShowHomeEnabled(true);

            final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.
abc_ic_ab_back_material);
            upArrow.setColorFilter(getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP)
;
            getSupportActionBar().setHomeAsUpIndicator(upArrow);
        }
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        if (item.getItemId() == android.R.id.home) // Press Back Icon
        {
            finish();
        }

        return super.onOptionsItemSelected(item);
    }
}

```

Archivo - ContactsCommunicationActivity.java

```java
package com.example.SAPLM.settingsActivities;

import android.graphics.PorterDuff;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import androidx.core.content.ContextCompat;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import android.view.MenuItem;

import com.example.SAPLM.Application;
import com.example.SAPLM.R;

public class ContactsCommunicationActivity extends AppCompatActivity {
    private static Toolbar toolbar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.contacts_communication_layout);
        Application.setActivity(this);
        Application.setContext(this);

        toolbar = this.findViewById(R.id.toolbar);

        setSupportActionBar(toolbar);

        // add back arrow to toolbar
        if (getSupportActionBar() != null){
            getSupportActionBar().setDisplayHomeAsUpEnabled(true);
            getSupportActionBar().setDisplayShowHomeEnabled(true);

            final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.
abc_ic_ab_back_material);
            upArrow.setColorFilter(getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP)
;
            getSupportActionBar().setHomeAsUpIndicator(upArrow);
        }
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        if (item.getItemId() == android.R.id.home) // Press Back Icon
        {
            finish();
        }

        return super.onOptionsItemSelected(item);
    }
}

```

Archivo - AppearanceMiscellaneousActivity.java

```java
package com.example.SAPLM.settingsActivities;

import android.graphics.PorterDuff;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import androidx.core.content.ContextCompat;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import android.view.MenuItem;

import com.example.SAPLM.Application;
import com.example.SAPLM.R;

public class AppearanceMiscellaneousActivity extends AppCompatActivity {

    private static Toolbar toolbar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.appearance_miscellaneous_layout);
        Application.setActivity(this);
        Application.setContext(this);

        toolbar = this.findViewById(R.id.toolbar);




        setSupportActionBar(toolbar);

        // add back arrow to toolbar
        if (getSupportActionBar() != null){
            getSupportActionBar().setDisplayHomeAsUpEnabled(true);
            getSupportActionBar().setDisplayShowHomeEnabled(true);

            final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.
abc_ic_ab_back_material);
            upArrow.setColorFilter(getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP)
;
            getSupportActionBar().setHomeAsUpIndicator(upArrow);
        }
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        if (item.getItemId() == android.R.id.home) // Press Back Icon
        {
            finish();
        }

        return super.onOptionsItemSelected(item);
    }
}

```

Archivo - VoiceRecognitionSettingsActivity.java

```java
package com.example.SAPLM.settingsActivities;

import android.content.DialogInterface;
import android.graphics.PorterDuff;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import android.speech.tts.Voice;
import androidx.core.content.ContextCompat;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ProgressBar;

import com.example.SAPLM.Application;
import com.example.SAPLM.R;
import com.example.SAPLM.TextToSpeechHandler;
import com.github.ybq.android.spinkit.sprite.Sprite;
import com.github.ybq.android.spinkit.style.FadingCircle;

import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.Locale;

public class VoiceRecognitionSettingsActivity extends AppCompatActivity {
    private static Toolbar toolbar;

    private Button speech_test_btn;
    private Button set_voice_btn;
    private EditText phrase_editText;
    private static ProgressBar progressBar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.voice_recognition_settings_layout);
        Application.setActivity(this);
        Application.setContext(this);

        toolbar = this.findViewById(R.id.toolbar);
        speech_test_btn = this.findViewById(R.id.speech_test_btn);
        set_voice_btn = this.findViewById(R.id.set_voice_btn);
        phrase_editText = this.findViewById(R.id.phrase_editText);

        setSupportActionBar(toolbar);

        // add back arrow to toolbar
        if (getSupportActionBar() != null){
            getSupportActionBar().setDisplayHomeAsUpEnabled(true);
            getSupportActionBar().setDisplayShowHomeEnabled(true);

            final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.
abc_ic_ab_back_material);
            upArrow.setColorFilter(getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP)
;
            getSupportActionBar().setHomeAsUpIndicator(upArrow);
        }

```

```
62          TextToSpeechHandler.initializeTextToSpeech();
63
64
65          progressBar = (ProgressBar)findViewById(R.id.progressBar);
66          Sprite fadingCircle = new FadingCircle();
67          fadingCircle.setColor(R.color.colorPrimary);
68          progressBar.setIndeterminateDrawable(fadingCircle);
69
70
71      }
72
73      @Override
74      public boolean onOptionsItemSelected(MenuItem item) {
75          if (item.getItemId() == android.R.id.home) // Press Back Icon
76          {
77              finish();
78          }
79
80          return super.onOptionsItemSelected(item);
81      }
82
83      public void speechTestOnClick(View view){
84          if (TextToSpeechHandler.isInitialized){
85              //TextToSpeechHandler.addToSpeakingQueue(phrase_editText.getText().toString());
86
87              Date currentTime = Calendar.getInstance().getTime();
88              SimpleDateFormat df = new SimpleDateFormat("d-MMMM-yyyy", new Locale("spa"));
89              String todayAsString = df.format(currentTime);
90              SimpleDateFormat onlyTimeFormar = new SimpleDateFormat("HH:mm:ss", new Locale("
    spa"));
91              String onlyTimeString = onlyTimeFormar.format(currentTime);
92
93              TextToSpeechHandler.addToSpeakingQueue("Hoy es " + todayAsString + "y son las"
    + onlyTimeString);
94
95          }else{
96              Application.toastMessage("Not initialized");
97          }
98      }
99
100     public static ArrayList<String> voiceName= new ArrayList<>();
101     public static Voice[] voiceArray;
102
103     public void setVoiceOnClick(View view){
104         AlertDialog.Builder builder = new AlertDialog.Builder(Application.getContext());
105
106         voiceArray = TextToSpeechHandler.getVoicesList().toArray(new Voice[
    TextToSpeechHandler.getVoicesList().size()]);
107
108         for(Voice voice : voiceArray){
109             voiceName.add(voice.getName());
110         }
111
112         builder.setTitle("Seleccionar voz");
113         builder.setItems(voiceName.toArray(new String[voiceName.size()]), new
    DialogInterface.OnClickListener() {
114             @Override
115             public void onClick(DialogInterface dialog, int which) {
116                 TextToSpeechHandler.setVoice(voiceArray[which]);
117             }
118         });
119         builder.show();
120     }
```

```
121
122     @Override
123     protected void onPause() {
124         if (TextToSpeechHandler.isInitialized){
125             //TextToSpeechHandler.shutdownTextSpeech();
126         }
127
128         super.onPause();
129     }
130 }
131
```

Archivo - CRC8_CCITT.java

```
 1 package com.example.SAPLM.bluetoothActivities;
 2
 3
 4 public class CRC8_CCITT {
 5     public static byte[] crc_table = {
 6             (byte) 0x00, (byte) 0x07, (byte) 0x0e, (byte) 0x09, (byte) 0x1c, (byte) 0x1b, (
   byte) 0x12, (byte) 0x15, (byte) 0x38, (byte) 0x3f, (byte) 0x36, (byte) 0x31, (byte) 0x24, (
   byte) 0x23, (byte) 0x2a, (byte) 0x2d,
 7             (byte) 0x70, (byte) 0x77, (byte) 0x7e, (byte) 0x79, (byte) 0x6c, (byte) 0x6b, (
   byte) 0x62, (byte) 0x65, (byte) 0x48, (byte) 0x4f, (byte) 0x46, (byte) 0x41, (byte) 0x54, (
   byte) 0x53, (byte) 0x5a, (byte) 0x5d,
 8             (byte) 0xe0, (byte) 0xe7, (byte) 0xee, (byte) 0xe9, (byte) 0xfc, (byte) 0xfb, (
   byte) 0xf2, (byte) 0xf5, (byte) 0xd8, (byte) 0xdf, (byte) 0xd6, (byte) 0xd1, (byte) 0xc4, (
   byte) 0xc3, (byte) 0xca, (byte) 0xcd,
 9             (byte) 0x90, (byte) 0x97, (byte) 0x9e, (byte) 0x99, (byte) 0x8c, (byte) 0x8b, (
   byte) 0x82, (byte) 0x85, (byte) 0xa8, (byte) 0xaf, (byte) 0xa6, (byte) 0xa1, (byte) 0xb4, (
   byte) 0xb3, (byte) 0xba, (byte) 0xbd,
10             (byte) 0xc7, (byte) 0xc0, (byte) 0xc9, (byte) 0xce, (byte) 0xdb, (byte) 0xdc, (
   byte) 0xd5, (byte) 0xd2, (byte) 0xff, (byte) 0xf8, (byte) 0xf1, (byte) 0xf6, (byte) 0xe3, (
   byte) 0xe4, (byte) 0xed, (byte) 0xea,
11             (byte) 0xb7, (byte) 0xb0, (byte) 0xb9, (byte) 0xbe, (byte) 0xab, (byte) 0xac, (
   byte) 0xa5, (byte) 0xa2, (byte) 0x8f, (byte) 0x88, (byte) 0x81, (byte) 0x86, (byte) 0x93, (
   byte) 0x94, (byte) 0x9d, (byte) 0x9a,
12             (byte) 0x27, (byte) 0x20, (byte) 0x29, (byte) 0x2e, (byte) 0x3b, (byte) 0x3c, (
   byte) 0x35, (byte) 0x32, (byte) 0x1f, (byte) 0x18, (byte) 0x11, (byte) 0x16, (byte) 0x03, (
   byte) 0x04, (byte) 0x0d, (byte) 0x0a,
13             (byte) 0x57, (byte) 0x50, (byte) 0x59, (byte) 0x5e, (byte) 0x4b, (byte) 0x4c, (
   byte) 0x45, (byte) 0x42, (byte) 0x6f, (byte) 0x68, (byte) 0x61, (byte) 0x66, (byte) 0x73, (
   byte) 0x74, (byte) 0x7d, (byte) 0x7a,
14             (byte) 0x89, (byte) 0x8e, (byte) 0x87, (byte) 0x80, (byte) 0x95, (byte) 0x92, (
   byte) 0x9b, (byte) 0x9c, (byte) 0xb1, (byte) 0xb6, (byte) 0xbf, (byte) 0xb8, (byte) 0xad, (
   byte) 0xaa, (byte) 0xa3, (byte) 0xa4,
15             (byte) 0xf9, (byte) 0xfe, (byte) 0xf7, (byte) 0xf0, (byte) 0xe5, (byte) 0xe2, (
   byte) 0xeb, (byte) 0xec, (byte) 0xc1, (byte) 0xc6, (byte) 0xcf, (byte) 0xc8, (byte) 0xdd, (
   byte) 0xda, (byte) 0xd3, (byte) 0xd4,
16             (byte) 0x69, (byte) 0x6e, (byte) 0x67, (byte) 0x60, (byte) 0x75, (byte) 0x72, (
   byte) 0x7b, (byte) 0x7c, (byte) 0x51, (byte) 0x56, (byte) 0x5f, (byte) 0x58, (byte) 0x4d, (
   byte) 0x4a, (byte) 0x43, (byte) 0x44,
17             (byte) 0x19, (byte) 0x1e, (byte) 0x17, (byte) 0x10, (byte) 0x05, (byte) 0x02, (
   byte) 0x0b, (byte) 0x0c, (byte) 0x21, (byte) 0x26, (byte) 0x2f, (byte) 0x28, (byte) 0x3d, (
   byte) 0x3a, (byte) 0x33, (byte) 0x34,
18             (byte) 0x4e, (byte) 0x49, (byte) 0x40, (byte) 0x47, (byte) 0x52, (byte) 0x55, (
   byte) 0x5c, (byte) 0x5b, (byte) 0x76, (byte) 0x71, (byte) 0x78, (byte) 0x7f, (byte) 0x6a, (
   byte) 0x6d, (byte) 0x64, (byte) 0x63,
19             (byte) 0x3e, (byte) 0x39, (byte) 0x30, (byte) 0x37, (byte) 0x22, (byte) 0x25, (
   byte) 0x2c, (byte) 0x2b, (byte) 0x06, (byte) 0x01, (byte) 0x08, (byte) 0x0f, (byte) 0x1a, (
   byte) 0x1d, (byte) 0x14, (byte) 0x13,
20             (byte) 0xae, (byte) 0xa9, (byte) 0xa0, (byte) 0xa7, (byte) 0xb2, (byte) 0xb5, (
   byte) 0xbc, (byte) 0xbb, (byte) 0x96, (byte) 0x91, (byte) 0x98, (byte) 0x9f, (byte) 0x8a, (
   byte) 0x8d, (byte) 0x84, (byte) 0x83,
21             (byte) 0xde, (byte) 0xd9, (byte) 0xd0, (byte) 0xd7, (byte) 0xc2, (byte) 0xc5, (
   byte) 0xcc, (byte) 0xcb, (byte) 0xe6, (byte) 0xe1, (byte) 0xe8, (byte) 0xef, (byte) 0xfa, (
   byte) 0xfd, (byte) 0xf4, (byte) 0xf3
22     };
23
24     public static byte crc_CCITT(byte[] data, int data_len)
25     {
26         byte crc = 0;
27         byte[] d = data;
28         int dataIndex = 0;
29         byte tbl_idx = 0;
30
31         while (data_len!=0) {
```

```
32              tbl_idx = (byte) (crc ^ d[dataIndex]);
33              crc = (byte) (crc_table[tbl_idx & 0xFF] & (byte)0xff);
34              dataIndex++;
35              data_len--;
36          }
37          return (byte) (crc & (byte)0xff);
38      }
39  }
40
```

Archivo - AvailableDevice.java

```java
package com.example.SAPLM.bluetoothActivities;

import com.example.SAPLM.R;

public class AvailableDevice {

    public enum DeviceState { DEFAULT, DISCONNECTED, CONNECTED, LOADING}
    private String deviceName;
    private DeviceState currentState;
    private int deviceStateIconID;

    public AvailableDevice() {
    }

    public AvailableDevice(String deviceName, DeviceState defaultState) {
        this.deviceName = deviceName;
        this.currentState = defaultState;
    }

    public void setCurrentState(DeviceState currentState) {
        this.currentState = currentState;
    }

    public DeviceState getCurrentState() {
        return currentState;
    }

    public void setDeviceName(String deviceName) {
        this.deviceName = deviceName;
    }

    public String getDeviceName() {
        return deviceName;
    }

    public int getDeviceStateIconID() {
        switch (currentState){
            case DEFAULT:
                return 0;
            case CONNECTED:
                return R.drawable.ic_done_icon;
            case LOADING:
                return R.drawable.loading_animation;
            case DISCONNECTED:
                return R.drawable.ic_close_icon;
        }
        return 0;
    }

    public int getTextAppearance(){
        switch (currentState){
            case DEFAULT:
                return R.style.btDeviceListAppearance;
            case CONNECTED:
                return R.style.btDeviceListSelectedAppearance;
            case LOADING:
                return R.style.btDeviceListAppearance;
            case DISCONNECTED:
                return R.style.btDeviceListAppearance;
        }
        return R.style.btDeviceListAppearance;
    }
}
```

Archivo - BluetoothActivity.java

```java
package com.example.SAPLM.bluetoothActivities;

import android.content.Context;
import android.content.Intent;
import android.graphics.Color;
import android.graphics.PorterDuff;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import android.os.Looper;
import androidx.core.content.ContextCompat;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.DefaultItemAnimator;
import androidx.recyclerview.widget.DividerItemDecoration;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
import androidx.appcompat.widget.Toolbar;
import android.view.MenuItem;
import android.view.View;
import android.widget.ImageView;
import android.widget.ProgressBar;
import android.widget.TextView;

import com.example.SAPLM.Application;
import com.example.SAPLM.R;
import com.example.SAPLM.settingsActivities.BluetoothSettingsActivity;
import com.example.SAPLM.settingsActivities.RecyclerTouchListener;
import com.github.ybq.android.spinkit.sprite.Sprite;
import com.github.ybq.android.spinkit.style.Circle;

import java.util.ArrayList;
import java.util.List;

public class BluetoothActivity extends AppCompatActivity {
    private static Toolbar toolbar;


    public static List<AvailableDevice> deviceList = new ArrayList<>();
    private RecyclerView recyclerView;
    public static DeviceListAdapter mAdapter;

    public static ImageView imgView_bluetoothStatusIcon;
    public static TextView textView_bluetoothState;
    public static ProgressBar loading_progressBar;

    public static Boolean guiInitialized = false;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.bluetooth_layout);
        Application.setActivity(this);
        Application.setContext(this);

        toolbar = this.findViewById(R.id.toolbar);
        textView_bluetoothState = this.findViewById(R.id.textView_btStatus);
        imgView_bluetoothStatusIcon = this.findViewById(R.id.imgView_btStatusIcon);
        loading_progressBar = this.findViewById(R.id.loading_progressBar);

        Sprite fadingCircle = new Circle();
        fadingCircle.setColor(Color.WHITE);
        loading_progressBar.setIndeterminateDrawable(fadingCircle);

```

```
64          setSupportActionBar(toolbar);
65
66          // add back arrow to toolbar
67          if (getSupportActionBar() != null){
68              getSupportActionBar().setDisplayHomeAsUpEnabled(true);
69              getSupportActionBar().setDisplayShowHomeEnabled(true);
70
71              final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.
    abc_ic_ab_back_material);
72              upArrow.setColorFilter(getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP
    );
73              getSupportActionBar().setHomeAsUpIndicator(upArrow);
74          }
75
76          recyclerView = (RecyclerView) findViewById(R.id.deviceListRecyclerView);
77
78          mAdapter = new DeviceListAdapter(deviceList);
79
80          recyclerView.setHasFixedSize(true);
81
82          // vertical RecyclerView
83          // keep movie_list_row.xml width to `match_parent`
84          RecyclerView.LayoutManager mLayoutManager = new LinearLayoutManager(
    getApplicationContext());
85
86          // horizontal RecyclerView
87          // keep movie_list_row.xml width to `wrap_content`
88          // RecyclerView.LayoutManager mLayoutManager = new LinearLayoutManager(
    getApplicationContext(), LinearLayoutManager.HORIZONTAL, false);
89
90          recyclerView.setLayoutManager(mLayoutManager);
91
92          // adding inbuilt divider line
93          recyclerView.addItemDecoration(new DividerItemDecoration(this, LinearLayoutManager.
    VERTICAL));
94
95          // adding custom divider line with padding 16dp
96          //recyclerView.addItemDecoration(new MyDividerItemDecoration(this,
    LinearLayoutManager.HORIZONTAL, 16));
97          recyclerView.setItemAnimator(new DefaultItemAnimator());
98
99          recyclerView.setAdapter(mAdapter);
100
101         // row click listener
102         recyclerView.addOnItemTouchListener(new RecyclerTouchListener(getApplicationContext
    (), recyclerView, new RecyclerTouchListener.ClickListener() {
103             @Override
104             public void onClick(View view, int position) {
105                 AvailableDevice device = deviceList.get(position);
106
107                 BluetoothConnection.connectBluetoothDeviceAsync(position);
108             }
109
110             @Override
111             public void onLongClick(View view, int position) {
112
113             }
114         }));
115
116         guiInitialized = true;
117
118         BluetoothConnection.listAvailableBluetoothDevices();
119
```

```
120     }
121
122     @Override
123     public boolean onOptionsItemSelected(MenuItem item) {
124         if (item.getItemId() == android.R.id.home) // Press Back Icon
125         {
126             finish();
127         }
128
129         return super.onOptionsItemSelected(item);
130     }
131
132     public static void updateDevicesList(){
133         if (guiInitialized) {
134             Application.getActivity().runOnUiThread(new Runnable() {
135                 @Override
136                 public void run() {
137                     BluetoothActivity.mAdapter.notifyDataSetChanged();
138                 }
139             });
140         }
141     }
142
143     public void configBtn_onClick(View view){
144         Context currentContext = Application.getContext();
145         Intent i = new Intent(currentContext, BluetoothSettingsActivity.class);
146         currentContext.startActivity(i);
147     }
148
149
150     public enum bluetoothConnectionState { SYSTEM_DISCONNECTED, SYSTEM_CONNECTED,
    BLUETOOTH_DISABLED, WAITING, BT_TURNING_OFF, BT_TURNING_ON, UNDEFINED}
151     public static bluetoothConnectionState currentBluetoothState = bluetoothConnectionState
    .UNDEFINED;
152
153     public static void setBluetoothStateGui(bluetoothConnectionState state){
154         if (guiInitialized) {
155             currentBluetoothState = state;
156
157             if (Looper.myLooper() == Looper.getMainLooper()) {
158                 switch (currentBluetoothState) {
159                     case SYSTEM_CONNECTED:
160                         textView_bluetoothState.setText(R.string.system_connected_message);
161                         imgView_bluetoothStatusIcon.setImageResource(R.drawable.
    ic_done_icon);
162                         imgView_bluetoothStatusIcon.setVisibility(View.VISIBLE);
163                         loading_progressBar.setVisibility(View.GONE);
164                         break;
165                     case BLUETOOTH_DISABLED:
166                         textView_bluetoothState.setText(R.string.bluetooth_disabled_message
    );
167                         imgView_bluetoothStatusIcon.setImageResource(R.drawable.
    ic_bluetooth_disabled);
168                         imgView_bluetoothStatusIcon.setVisibility(View.VISIBLE);
169                         loading_progressBar.setVisibility(View.GONE);
170                         break;
171                     case BT_TURNING_ON:
172                         textView_bluetoothState.setText(R.string.
    bluetooth_initializing_adapater);
173                         imgView_bluetoothStatusIcon.setVisibility(View.GONE);
174                         loading_progressBar.setVisibility(View.VISIBLE);
175                         break;
176                     case BT_TURNING_OFF:
```

```
177                         textView_bluetoothState.setText(R.string.bluetooth_closing_adapter)
    ;
178                         imgView_bluetoothStatusIcon.setVisibility(View.GONE);
179                         loading_progressBar.setVisibility(View.VISIBLE);
180                         break;
181                     case SYSTEM_DISCONNECTED:
182                         textView_bluetoothState.setText(R.string.
    system_disconnected_message);
183                         imgView_bluetoothStatusIcon.setImageResource(R.drawable.
    ic_close_icon);
184                         imgView_bluetoothStatusIcon.setVisibility(View.VISIBLE);
185                         loading_progressBar.setVisibility(View.GONE);
186                         break;
187                     case WAITING:
188                         textView_bluetoothState.setText(R.string.
    bluetooth_attempting_connection_message);
189                         imgView_bluetoothStatusIcon.setVisibility(View.GONE);
190                         loading_progressBar.setVisibility(View.VISIBLE);
191                         break;
192                 }
193             } else {
194                 Application.getActivity().runOnUiThread(new Runnable() {
195                     @Override
196                     public void run() {
197                         switch (currentBluetoothState) {
198                             case SYSTEM_CONNECTED:
199                                 textView_bluetoothState.setText(R.string.
    system_connected_message);
200                                 imgView_bluetoothStatusIcon.setImageResource(R.drawable.
    ic_done_icon);
201                                 imgView_bluetoothStatusIcon.setVisibility(View.VISIBLE);
202                                 loading_progressBar.setVisibility(View.GONE);
203                                 break;
204                             case BLUETOOTH_DISABLED:
205                                 textView_bluetoothState.setText(R.string.
    bluetooth_disabled_message);
206                                 imgView_bluetoothStatusIcon.setImageResource(R.drawable.
    ic_bluetooth_disabled);
207                                 imgView_bluetoothStatusIcon.setVisibility(View.VISIBLE);
208                                 loading_progressBar.setVisibility(View.GONE);
209                                 break;
210                             case BT_TURNING_ON:
211                                 textView_bluetoothState.setText(R.string.
    bluetooth_initializing_adapater);
212                                 imgView_bluetoothStatusIcon.setVisibility(View.GONE);
213                                 loading_progressBar.setVisibility(View.VISIBLE);
214                                 break;
215                             case BT_TURNING_OFF:
216                                 textView_bluetoothState.setText(R.string.
    bluetooth_closing_adapter);
217                                 imgView_bluetoothStatusIcon.setVisibility(View.GONE);
218                                 loading_progressBar.setVisibility(View.GONE);
219                                 break;
220                             case SYSTEM_DISCONNECTED:
221                                 textView_bluetoothState.setText(R.string.
    system_disconnected_message);
222                                 imgView_bluetoothStatusIcon.setImageResource(R.drawable.
    ic_close_icon);
223                                 imgView_bluetoothStatusIcon.setVisibility(View.VISIBLE);
224                                 loading_progressBar.setVisibility(View.GONE);
225                                 break;
226                             case WAITING:
227                                 textView_bluetoothState.setText(R.string.
```

Archivo - BluetoothActivity.java

```
227 bluetooth_attempting_connection_message);
228                                 imgView_bluetoothStatusIcon.setVisibility(View.GONE);
229                                 loading_progressBar.setVisibility(View.VISIBLE);
230                                 break;
231                         }
232                     }
233                 });
234             }
235         }
236     }
237
238
239 }
240
```

Archivo - DeviceListAdapter.java

```java
package com.example.SAPLM.bluetoothActivities;

import android.graphics.Color;
import android.graphics.PorterDuff;
import android.graphics.drawable.ColorDrawable;
import android.graphics.drawable.Drawable;
import androidx.recyclerview.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;

import com.example.SAPLM.Application;
import com.example.SAPLM.R;

import java.util.List;

public class DeviceListAdapter extends RecyclerView.Adapter<DeviceListAdapter.viewHolder> {

    private List<AvailableDevice> deviceList;

    public class viewHolder extends RecyclerView.ViewHolder {
        public TextView textView_device_name;
        public ImageView imgView_device_state_icon;

        public viewHolder(View view) {
            super(view);
            textView_device_name = (TextView) view.findViewById(R.id.
textView_bluetooth_device_name);
            imgView_device_state_icon = (ImageView) view.findViewById(R.id.
imageView_bluetooth_device_state_icon);
        }
    }


    public DeviceListAdapter(List<AvailableDevice> deviceList) {
        this.deviceList = deviceList;
    }

    @Override
    public viewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View itemView = LayoutInflater.from(parent.getContext())
                .inflate(R.layout.bluetooth_list_layout_style, parent, false);

        return new viewHolder(itemView);
    }

    @Override
    public void onBindViewHolder(viewHolder holder, int position) {
        AvailableDevice device = deviceList.get(position);
        holder.textView_device_name.setText(device.getDeviceName());
        holder.textView_device_name.setTextAppearance(device.getTextAppearance());

        int iconID = device.getDeviceStateIconID();

        if (iconID!=0) {
            Drawable icon = Application.getContext().getDrawable(iconID);
            icon.setColorFilter(Application.getContext().getColor(R.color.colorPrimary),
PorterDuff.Mode.SRC_ATOP);
            holder.imgView_device_state_icon.setImageDrawable(icon);
        } else {
            holder.imgView_device_state_icon.clearColorFilter();
```

Archivo - DeviceListAdapter.java

```
61              Drawable transparentDrawable = new ColorDrawable(Color.TRANSPARENT);
62              holder.imgView_device_state_icon.setImageDrawable(transparentDrawable);
63          }
64      }
65
66      @Override
67      public int getItemCount() {
68          return deviceList.size();
69      }
70  }
71
```

Archivo - BluetoothConnection.java

```java
package com.example.SAPLM.bluetoothActivities;

import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothServerSocket;
import android.bluetooth.BluetoothSocket;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.widget.TabHost;

import com.example.SAPLM.Application;
import com.example.SAPLM.settingsActivities.DeveloperConsoleActivity;
import com.example.SAPLM.settingsActivities.TerminalManager;


import java.io.InputStream;
import java.io.OutputStream;
import java.nio.charset.StandardCharsets;
import java.util.ArrayList;
import java.util.List;
import java.util.Set;
import java.util.UUID;
import java.util.concurrent.Executor;
import java.util.concurrent.Executors;
import java.util.concurrent.ScheduledExecutorService;
import java.util.concurrent.ScheduledFuture;


public class BluetoothConnection extends BluetoothActivity {

    static ScheduledExecutorService bluetoothThreadPool= Executors.newScheduledThreadPool(1);
    static ScheduledFuture<?> bluetoothComThread;
    public static OutputStream outputStream;
    public static InputStream inputStream;
    public static BluetoothAdapter btAdapter = BluetoothAdapter.getDefaultAdapter();
    public static BluetoothDevice targetDevice;
    public static BluetoothDevice[] bluetoothDevicesArray;
    public static List<BluetoothDevice> bluetoothDevicesList;
    public static BluetoothSocket btSocket;
    public static BluetoothServerSocket btServerSocket;
    public static UUID btUUID = UUID.fromString("00001101-0000-1000-8000-00805f9b34fb");
    public static String btString = "HC-05";
    public static Boolean bluetoothConnected = false;

    private static Executor bluetoothConnectorExecutor = Executors.newSingleThreadExecutor();
    private static Boolean isListAvailableBluetoothDevicesRunning = false;

    public static void listAvailableBluetoothDevices(){
        if ((!isListAvailableBluetoothDevicesRunning)) {
            try {
                BluetoothActivity.setBluetoothStateGui(bluetoothConnectionState.WAITING);
                isListAvailableBluetoothDevicesRunning = true;
                bluetoothConnectorExecutor.execute(new Runnable() {
                    @Override
                    public void run() {
                        if (btAdapter != null) {
                            if (btAdapter.isEnabled()) {
                                Set<BluetoothDevice> bondedDevices = btAdapter.getBondedDevices();

```

```
61                                  bluetoothDevicesList = new ArrayList<>();
62                                  bluetoothDevicesList.addAll(bondedDevices);
63
64                                  if (bondedDevices.size() > 0) {
65                                      bluetoothDevicesArray = bondedDevices.toArray(new
   BluetoothDevice[bondedDevices.size()]);
66                                  }
67                                  updateBluetoothListGUI();
68                              } else {
69                                  BluetoothActivity.setBluetoothStateGui(
   bluetoothConnectionState.BLUETOOTH_DISABLED);
70                              }
71                          }
72                          isListAvailableBluetoothDevicesRunning = false;
73                      }
74                  });
75              }catch (Throwable th)
76              {
77                  Application.toastMessage(th.toString()); th.printStackTrace();
78              }
79          }
80      }
81
82      public static Boolean isBluetoothAdapterReady(){
83          return ((btAdapter != null)&&(btAdapter.isEnabled()));
84      }
85
86      public static void updateBluetoothListGUI(){
87          if (BluetoothActivity.mAdapter!=null) {
88              BluetoothActivity.deviceList.clear();
89              for (int x = 0; x < bluetoothDevicesArray.length; x++) {
90                  BluetoothActivity.deviceList.add(new AvailableDevice(bluetoothDevicesArray[
   x].getName(), AvailableDevice.DeviceState.DEFAULT));
91              }
92
93              if (bluetoothConnected) {
94                  BluetoothActivity.setBluetoothStateGui(bluetoothConnectionState.
   SYSTEM_CONNECTED);
95                  deviceList.get(bluetoothDevicesList.indexOf(targetDevice)).setCurrentState(
   AvailableDevice.DeviceState.CONNECTED);
96              }
97              else {
98                  BluetoothActivity.setBluetoothStateGui(bluetoothConnectionState.
   SYSTEM_DISCONNECTED);
99              }
100
101             BluetoothActivity.updateDevicesList();
102         }
103     }
104
105     private static Boolean attemptingBluetoothConnection = false;
106
107
108     public static void connectBluetoothDeviceSync(int deviceIndex){
109         if ((!isListAvailableBluetoothDevicesRunning)&&(!attemptingBluetoothConnection)&&
   isBluetoothAdapterReady()) {
110             try {
111                 attemptingBluetoothConnection = true;
112                 if (bluetoothConnected) closeAllConnections();
113                 BluetoothActivity.setBluetoothStateGui(bluetoothConnectionState.WAITING);
114                 targetDevice = bluetoothDevicesArray[deviceIndex];
115                 btSocket = targetDevice.createRfcommSocketToServiceRecord(btUUID);
116                 btSocket.connect();
```

```
117                 if (btSocket.isConnected()) {
118                     inputStream = btSocket.getInputStream();
119                     outputStream = btSocket.getOutputStream();
120                     bluetoothConnected = true;
121                     BluetoothActivity.setBluetoothStateGui(bluetoothConnectionState.
    SYSTEM_CONNECTED);
122                     BluetoothActivity.updateDevicesList();
123                 } else {
124                     bluetoothConnected = false;
125                     BluetoothActivity.setBluetoothStateGui(bluetoothConnectionState.
    SYSTEM_DISCONNECTED);
126                 }
127                 attemptingBluetoothConnection = false;
128             } catch (Throwable th) {
129                 bluetoothConnected = false;
130                 Application.toastMessage(th.toString());
131                 BluetoothActivity.setBluetoothStateGui(bluetoothConnectionState.
    SYSTEM_DISCONNECTED);
132                 attemptingBluetoothConnection = false;
133             }
134         }
135     }
136 
137     private static int attemptingConnectionDeviceIndex;
138 
139     public static void connectBluetoothDeviceAsync(int deviceIndex){
140         attemptingConnectionDeviceIndex = deviceIndex;
141         if ((!isListAvailableBluetoothDevicesRunning)&&(!attemptingBluetoothConnection)&&
    isBluetoothAdapterReady()) {
142             bluetoothConnectorExecutor.execute(new Runnable() {
143                 @Override
144                 public void run() {
145                     connectBluetoothDeviceSync(attemptingConnectionDeviceIndex);
146                 }
147             });
148         }
149     }
150 
151     public static void sendCommandBuffer(byte[] data){
152         try {
153             outputStream.write(new byte[] {(byte)0x7F, (byte)0x7F, (byte)0x7F, (byte)0x7F})
    ;
154             outputStream.write(data,0, 20);
155         }catch (Throwable th) {
156             Application.toastMessage(th.toString());
157         }
158     }
159 
160     public static byte[] receiveCommandBuffer(int timeout){
161         long startTimeMillis = System.currentTimeMillis();
162         while((System.currentTimeMillis()-startTimeMillis)<500L){
163             try {
164                 if (inputStream.available() >= 32) {
165                     byte[] inputBuffer = new byte[32];
166                     inputStream.read(inputBuffer, 0, 32);
167                     return inputBuffer;
168                 }
169             }catch (Throwable th){
170                 Application.toastMessage(th.toString());
171             }
172         }
173 
174         return null;
```

```
175     }
176
177
178
179     public static class BluetoothReceiverService{
180
181         private static Executor receiverExecutor = Executors.newSingleThreadExecutor();
182         private static Boolean terminate = false;
183         private static Boolean isAutoConnectRunning = false;
184
185         public static void initialize(){
186             if (isAutoConnectRunning==false) {
187                 receiverExecutor.execute(new Runnable() {
188                     @Override
189                     public void run() {
190                         isAutoConnectRunning = true;
191                         while (terminate == false) {
192                             try {
193                                 if (inputStream.available()>= 32) {   //  >= 32
194                                     byte[] inputBuffer = new byte[32];
195                                     inputStream.read(inputBuffer, 0, 32);
196                                     UnifiedTransmissionProtocol.command_buffer =
    inputBuffer;
197                                     DeveloperConsoleActivity.sendTextToTerminal("Received
    from: " + UnifiedTransmissionProtocol.lastMessageTransmitterModule.getModuleName() + " : "
    + TerminalManager.insertSpaces(UnifiedTransmissionProtocol.bytesToHex(inputBuffer), " ", 2)
    );
198                                     if (UnifiedTransmissionProtocol.
    DecomposeMessageFromBuffer()== UnifiedTransmissionProtocol.CommandStatus.
    SUCCESSFUL_DECOMPOSITION) {
199                                         UnifiedTransmissionProtocol.handleReceivedCommand()
    ;
200                                     } else { Application.toastMessage("UNSUCCESSFUL
    DECOMPOSITION");}
201                                 }
202                             } catch (Throwable th) {
203                                 Application.toastMessage(th.toString());
204                             }
205                         }
206                         isAutoConnectRunning = false;
207                     }
208                 });
209             }
210         }
211
212         public static void terminateSync(){
213             terminate = true;
214             while(isAutoConnectRunning==true) { }
215             terminate = false;
216         }
217
218     }
219
220
221
222
223
224
225
226     private static Executor autoConnectExecutor = Executors.newSingleThreadExecutor();
227     private static Boolean terminateAutoConnect = false;
228     private static Boolean isAutoConnectRunning = false;
229
```

```
230     public static void initBTAutoConnectService(){
231         if (isAutoConnectRunning() == false) {
232             autoConnectExecutor.execute(new Runnable() {
233                 @Override
234                 public void run() {
235                     isAutoConnectRunning = true;
236                     try { Thread.sleep(5000);} catch (Throwable th) { th.printStackTrace();
    }
237                     while (!terminateAutoConnect && isBluetoothAdapterReady()) {
238                         listAvailableBluetoothDevices();
239                         while (isListAvailableBluetoothDevicesRunning) {}
240                         for (int x = 0; x < bluetoothDevicesArray.length; x++) {
241                             // MAC ADDRESS HC-05 : 98:D3:91:FD:38:05
242                             // MAC ADDRESS XT1068 : 5C:51:88:A2:38:97
243                             if (bluetoothDevicesArray[x].getAddress().compareToIgnoreCase("
    98:D3:91:FD:38:05") == 0) {
244                                 if (!terminateAutoConnect&&!bluetoothConnected)
    connectBluetoothDeviceSync(x);
245                             }
246                         }
247                         try { Thread.sleep(10000);} catch (Throwable th) { th.
    printStackTrace(); }
248                     }
249                     isAutoConnectRunning = false;
250                     terminateAutoConnect = false;
251                 }
252             });
253         }else{
254             terminateAutoConnect = false;
255         }
256     }
257
258     public static void terminateBTAutoConnect(){
259         terminateAutoConnect = true;
260     }
261
262     public static Boolean isAutoConnectRunning(){
263         return isAutoConnectRunning;
264     }
265
266
267
268
269     public static void connectionLost(){
270
271     }
272
273     public static void closeAllConnections(){
274         try {
275             bluetoothConnected = false;
276             if (bluetoothComThread != null) bluetoothComThread.cancel(true);
277             if (btSocket != null) btSocket.close();
278             if (outputStream != null) outputStream.close();
279             if (inputStream != null) inputStream.close();
280
281             btSocket = null;
282             outputStream = null;
283             inputStream = null;
284         } catch (Throwable th){
285             th.printStackTrace();
286         }
287     }
288
```

```
289
290
291     public static BroadcastReceiver bluetoothAdapterStatusBR = new BroadcastReceiver() {
292         @Override
293         public void onReceive(Context context, Intent intent) {
294             final String action = intent.getAction();
295             if (action.equals(BluetoothAdapter.ACTION_STATE_CHANGED)){
296                 final int stateCode = intent.getIntExtra(BluetoothAdapter.EXTRA_STATE,
    BluetoothAdapter.ERROR);
297
298                 switch (stateCode){
299                     case BluetoothAdapter.STATE_OFF:
300                         //Bluetooth adapter turned off handler
301                         BluetoothActivity.setBluetoothStateGui(bluetoothConnectionState.
    BLUETOOTH_DISABLED);
302                         break;
303                     case BluetoothAdapter.STATE_TURNING_OFF:
304                         //Bluetooth adapter turning off handler
305                         terminateBTAutoConnect();
306                         BluetoothActivity.setBluetoothStateGui(bluetoothConnectionState.
    BT_TURNING_OFF);
307                         break;
308                     case BluetoothAdapter.STATE_ON:
309                         BluetoothActivity.setBluetoothStateGui(bluetoothConnectionState.
    SYSTEM_DISCONNECTED);
310                         updateDevicesList();
311                         initBTAutoConnectService();
312
313                         break;
314                     case BluetoothAdapter.STATE_TURNING_ON:
315                         //Bluetooth adapter turning on handler
316                         BluetoothActivity.setBluetoothStateGui(bluetoothConnectionState.
    BT_TURNING_ON);
317                         break;
318                 }
319             }
320         }
321     };
322
323     public static BroadcastReceiver bluetoothScanModeStatusBR = new BroadcastReceiver() {
324
325         @Override
326         public void onReceive(Context context, Intent intent) {
327             final String action = intent.getAction();
328
329             if (action.equals(BluetoothAdapter.ACTION_SCAN_MODE_CHANGED)) {
330
331                 int mode = intent.getIntExtra(BluetoothAdapter.EXTRA_SCAN_MODE,
    BluetoothAdapter.ERROR);
332
333                 switch (mode) {
334                     case BluetoothAdapter.SCAN_MODE_CONNECTABLE_DISCOVERABLE:
335
336                         break;
337                     case BluetoothAdapter.SCAN_MODE_CONNECTABLE:
338
339                         break;
340                     case BluetoothAdapter.SCAN_MODE_NONE:
341
342                         break;
343                 }
344             }
345         }
```

```
346     };
347
348     public static BroadcastReceiver bluetoothDeviceStatusBR = new BroadcastReceiver() {
349         @Override
350         public void onReceive(Context context, Intent intent) {
351             String action = intent.getAction();
352
353             switch (action){
354                 case BluetoothDevice.ACTION_ACL_CONNECTED:
355                     terminateBTAutoConnect();
356                     BluetoothReceiverService.initialize();
357                     updateDevicesList();
358                     break;
359                 case BluetoothDevice.ACTION_ACL_DISCONNECTED:
360                     BluetoothReceiverService.terminateSync();
361                     closeAllConnections();
362                     BluetoothActivity.setBluetoothStateGui(bluetoothConnectionState.
    SYSTEM_DISCONNECTED);
363                     initBTAutoConnectService();
364                     updateBluetoothListGUI();
365                     //Application.toastMessage("Dispositivo desconectado!");
366                     break;
367                 case BluetoothDevice.ACTION_ACL_DISCONNECT_REQUESTED:
368
369                     break;
370                 case BluetoothDevice.ACTION_BOND_STATE_CHANGED:
371
372                     break;
373             }
374         }
375     };
376
377 }
378
```

Archivo - UnifiedTransmissionProtocol.java

```java
package com.example.SAPLM.bluetoothActivities;

import com.example.SAPLM.R;
import com.example.SAPLM.commandActivity.CommandsActivity;
import com.example.SAPLM.settingsActivities.DeveloperConsoleActivity;
import com.example.SAPLM.settingsActivities.TerminalManager;


import java.io.UnsupportedEncodingException;
import java.nio.ByteBuffer;
import java.nio.ByteOrder;
import java.nio.charset.StandardCharsets;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.concurrent.Executor;
import java.util.concurrent.Executors;
import java.util.regex.Matcher;
import java.util.regex.Pattern;



public class UnifiedTransmissionProtocol {

    public static List<Module> modulesList = new ArrayList<UnifiedTransmissionProtocol.Module>(Arrays.asList(
            new Module("Dispositivo Movil", "PHONE", (byte)0x00, R.drawable.ic_icon_phone_module, new ArrayList<Module.Device>(Arrays.asList(
                    new Module.Device("Módulo principal", new Boolean(false), R.drawable.ic_icon_main_board, (byte)0x00, new Boolean(true))
            ))),
            new Module("Módulo principal","MAIN", (byte)0x01, R.drawable.ic_icon_main_board, new ArrayList<Module.Device>(Arrays.asList(
                    new Module.Device("Módulo de potencia", new Boolean(false), R.drawable.icons_board, (byte)0x00, new Boolean(true)),
                    new Module.Device("Módulo motriz", new Boolean(false), R.drawable.icons_board, (byte)0x01, new Boolean(true))
            ))),
            new Module("Módulo de potencia","POWER", (byte)0x02, R.drawable.icons_board, new ArrayList<Module.Device>(Arrays.asList(
                    new Module.Device("Velador", new Boolean(false), R.drawable.icons_desk_lamp, (byte)0x00, new Boolean(false)),
                    new Module.Device("Luz", new Boolean(false), R.drawable.icons_bulb, (byte)0x01, new Boolean(false)),
                    new Module.Device("Ventilador", new Boolean(false), R.drawable.icons_fan, (byte)0x02, new Boolean(false)),
                    new Module.Device("Estufa", new Boolean(false), R.drawable.icons_heater, (byte)0x03, new Boolean(false))
            ))),
            new Module("Módulo motriz","MOTOR", (byte)0x03, R.drawable.icons_board, new ArrayList<Module.Device>(Arrays.asList(
                    new Module.Device("Camilla", new Byte((byte)0), R.drawable.icons_stretcher, (byte)0x00, new Boolean(false)),
                    new Module.Device("Cortina", new Byte((byte)0), R.drawable.icons_curtain, (byte)0x01, new Boolean(false)),
                    new Module.Device("Llamar enfermera", new Boolean(false), R.drawable.icons_nurse, (byte)0x02, new Boolean(false))
            )))
    ));

    protected static byte[] command_buffer = new byte[32];

    private static Module currentModuleID = Module.getLocalModule();
```

```
 49     public static Module lastMessageTransmitterModule = new Module();
 50     public static byte lastMessagePID;
 51     private static CommandType lastCommandType;
 52     private static List<Parameter> lastParameterList;
 53     private static boolean isAvailableCommand;
 54
 55
 56     public static byte[] getCommand_buffer() {
 57         return command_buffer;
 58     }
 59
 60
 61
 62
 63
 64
 65     public static class Parameter {
 66         private byte[] parameterData;
 67
 68         Parameter(byte[] parameterData){
 69             this.parameterData = parameterData;
 70         }
 71
 72         public byte[] getParameterData() {
 73             return parameterData;
 74         }
 75
 76         public byte getParameterLength() {
 77             return (byte)parameterData.length;
 78         }
 79     }
 80
 81
 82
 83     public static CommandStatus ComposeMessageToBuffer(CommandType targetType, Module
    targetModule, List<Parameter> parameterList) {
 84         command_buffer = new byte[32];
 85         command_buffer[0] = SectionDividersChar.SOH.getCode();
 86         if (lastMessagePID!=0xFF) { lastMessagePID++; } else { lastMessagePID=(byte)0x00; }
 87         command_buffer[1] =  lastMessagePID;
 88         command_buffer[2] =  targetModule.getInternalCode(); // Final receiver
 89         command_buffer[3] =  currentModuleID.getInternalCode(); // Original transmitter
 90         command_buffer[4] = SectionDividersChar.STX.getCode();
 91         command_buffer[5] = targetType.getCommandCode();
 92
 93         if (parameterList.size()>12) return CommandStatus.PARAMETER_COUNT_OVERSIZE;
 94         int currentIndexWriting = 6;
 95         for (Parameter parameter : parameterList) {
 96             command_buffer[currentIndexWriting] = parameter.getParameterLength();
 97             currentIndexWriting++;
 98             System.arraycopy(parameter.getParameterData(), 0, command_buffer,
    currentIndexWriting, parameter.getParameterLength());
 99             currentIndexWriting+=parameter.getParameterLength();
100         }
101
102         byte CRC = CRC8_CCITT.crc_CCITT(command_buffer, currentIndexWriting);
103         int footerStartIndex = currentIndexWriting;
104
105         command_buffer[footerStartIndex] = SectionDividersChar.ETX.getCode();
106         command_buffer[footerStartIndex+1] = CRC;
107         command_buffer[footerStartIndex+2] = SectionDividersChar.ETB.getCode();
108         return CommandStatus.SUCCESSFUL_COMPOSITION;
109     }
```

```
110
111
112     public static CommandStatus DecomposeMessageFromBuffer() {
113         try {
114             String commandString = new String(command_buffer, "ISO-8859-1");
115             Pattern headerPattern = Pattern.compile("\\u0001(.{3}?)\\u0002", Pattern.DOTALL
    );
116             Pattern footerPattern = Pattern.compile("\\u0003(.{1}?)\\u0017", Pattern.DOTALL
    );
117             Matcher headerMatcher = headerPattern.matcher(commandString);
118             Matcher footerMatcher = footerPattern.matcher(commandString);
119
120             if (headerMatcher.find()==false) return CommandStatus.WRONG_HEADER_SEGMENTATION
    ;
121             if (footerMatcher.find()==false) return CommandStatus.WRONG_FOOTER_SEGMENTATION
    ;
122
123             int headerStartIndex = headerMatcher.start();
124             int parameterStartIndex = headerMatcher.end()+1;
125             int footerStartIndex = footerMatcher.start();
126
127             lastMessagePID = command_buffer[headerStartIndex+1];    //verificar si hay que
    convertir a desde unsigned
128             if (command_buffer[headerStartIndex+2]!=currentModuleID.getInternalCode())
    return CommandStatus.WRONG_MODULE_ID;
129
130             lastMessageTransmitterModule = Module.getModule(command_buffer[headerStartIndex
    +3]);
131
132             lastCommandType = CommandType.definedCommandList.stream().filter(p-> p.
    getCommandCode()==command_buffer[headerMatcher.end()]).findFirst().orElse(null);
133
134             lastParameterList = new ArrayList<Parameter>();
135
136             while(parameterStartIndex!=footerStartIndex) {
137                 int parameterLength = command_buffer[parameterStartIndex];
138                 byte[] parameterData = new byte[parameterLength];
139                 System.arraycopy(command_buffer, parameterStartIndex+1, parameterData, 0,
    parameterLength);
140                 lastParameterList.add(new Parameter(parameterData));
141                 parameterStartIndex+=(1+parameterLength);
142             }
143
144             if (command_buffer[footerStartIndex+1]!=CRC8_CCITT.crc_CCITT(command_buffer,
    footerStartIndex)) return CommandStatus.WRONG_CHECKSUM_CONSISTENCY;
145
146         } catch (UnsupportedEncodingException e) {
147             e.printStackTrace();
148             return CommandStatus.FAILED_DECOMPOSITION;
149         }
150
151
152         isAvailableCommand = true;
153         return CommandStatus.SUCCESSFUL_DECOMPOSITION;
154     }
155
156     public static void handleReceivedCommand() {
157         if (isAvailableCommand) {
158             if (lastCommandType!=null) lastCommandType.handleCommand();
159             isAvailableCommand = false;
160         }
161     }
162
```

```
163
164     enum CommandStatus {
165         SUCCESSFUL_DECOMPOSITION((byte)0x00),
166         SUCCESSFUL_COMPOSITION((byte)0x01),
167         WRONG_HEADER_SEGMENTATION((byte)0x02),
168         WRONG_FOOTER_SEGMENTATION((byte)0x03),
169         WRONG_CHECKSUM_CONSISTENCY((byte)0x04),
170         WRONG_MODULE_ID((byte)0x05),
171         UNDEFINED_COMMAND_CODE((byte)0x06),
172         PARAMETER_DATA_OVERFLOW((byte)0x07),
173         PARAMETER_COUNT_OVERSIZE((byte)0x08),
174         FAILED_DECOMPOSITION((byte)0x09),
175         FAILED_COMPOSITION((byte)0x0A);
176
177         private final byte code;
178         CommandStatus(byte code){
179             this.code = code;
180         }
181         public byte getCode() {
182             return code;
183         }
184
185     }
186
187     public static class Module extends UnifiedTransmissionProtocol{
188         private String moduleName;
189         private String moduleAlias;
190         private byte internalCode;
191         private int iconResourceId;
192         private List<Device> devicesList;
193
194         Module(String moduleName,String moduleAlias, byte internalCode, int iconResourceId,
    List<Device> devicesList){
195             this.moduleName = moduleName;
196             this.internalCode = internalCode;
197             this.iconResourceId = iconResourceId;
198             this.devicesList = devicesList;
199         }
200
201         Module(){
202
203         }
204
205         public List<Device> getDevicesList() {
206             return devicesList;
207         }
208
209         public int getIconResourceId() {
210             return iconResourceId;
211         }
212
213         public byte getInternalCode() {
214             return internalCode;
215         }
216
217         public String getModuleName() {
218             return moduleName;
219         }
220
221         public String getModuleAlias() {
222             return moduleAlias;
223         }
224
```

```
225         public List<Parameter> exportDataToParameterList(){
226             //Desarrollar esta funcion
227             // AVR-GCC USES LITTLE-ENDIAN
228             // AVR-GCC USES STRICTLY IEEE 754 32-BIT FLOAT POINT
229 
230             List<Parameter> exportList = new ArrayList<UnifiedTransmissionProtocol.
    Parameter>();
231 
232             for (int x = 0; x < devicesList.size(); x++) {
233                 for (Module.Device device : devicesList) {
234                     if (x==byteToUnsignedInt(device.getInternalIndex())) {
235                         Parameter newParameter = null;
236 
237                         if (device.getData().getClass()==Byte.class) {
238                             ByteBuffer byteBuffer = ByteBuffer.allocate(Byte.BYTES);
239                             byteBuffer.order(ByteOrder.LITTLE_ENDIAN);
240                             byteBuffer.put((byte) device.getData());
241                             newParameter = new Parameter(byteBuffer.array());
242                         } else if (device.getData().getClass()==Short.class) {
243                             ByteBuffer shortBuffer = ByteBuffer.allocate(Short.BYTES);
244                             shortBuffer.order(ByteOrder.LITTLE_ENDIAN);
245                             shortBuffer.putShort((short) device.getData());
246                             newParameter = new Parameter(shortBuffer.array());
247                         } else if (device.getData().getClass()==Integer.class) {
248                             ByteBuffer intBuffer = ByteBuffer.allocate(Integer.BYTES);
249                             intBuffer.order(ByteOrder.LITTLE_ENDIAN);
250                             intBuffer.putInt((int) device.getData());
251                             newParameter = new Parameter(intBuffer.array());
252                         } else if (device.getData().getClass()==Float.class) {
253                             ByteBuffer floatBuffer = ByteBuffer.allocate(Float.BYTES);
254                             floatBuffer.order(ByteOrder.LITTLE_ENDIAN);
255                             floatBuffer.putFloat((float) device.getData());
256                             newParameter = new Parameter(floatBuffer.array());
257                         } else if (device.getData().getClass()==Double.class) {
258                             ByteBuffer doubleBuffer = ByteBuffer.allocate(Float.BYTES);
259                             doubleBuffer.order(ByteOrder.LITTLE_ENDIAN);
260                             doubleBuffer.putFloat((float)((double)device.getData()));
261                             newParameter = new Parameter(doubleBuffer.array());
262                         } else if (device.getData().getClass()==String.class) {
263                             String data = (String) device.getData();
264                             newParameter = new Parameter(data.getBytes(StandardCharsets.
    US_ASCII));
265                         } else if (device.getData().getClass()==Boolean.class) {
266                             byte data = (byte) 0x00;
267                             if ((Boolean)device.getData()==true) data = (byte)0xFF;
268                             ByteBuffer booleanBuffer = ByteBuffer.allocate(Byte.BYTES);
269                             booleanBuffer.order(ByteOrder.LITTLE_ENDIAN);
270                             booleanBuffer.put(data);
271                             newParameter = new Parameter(booleanBuffer.array());
272                         }
273 
274                         if (newParameter!=null) exportList.add(newParameter);
275 
276                         break;
277                     }
278 
279                 }
280             }
281 
282             return exportList;
283         }
284 
285         public void updateDataFromParameterList(List<Parameter> parameterList){
```

```
286
287             for (int x = 0; x < parameterList.size(); x++) {
288                 Device deviceToUpdate = devicesList.get(x);
289                 ByteBuffer buffer = ByteBuffer.wrap(parameterList.get(x).getParameterData()
    );
290                 buffer.order(ByteOrder.LITTLE_ENDIAN);
291
292                 if (deviceToUpdate.getData().getClass()==Byte.class) {
293                     deviceToUpdate.setData(new Byte(buffer.get()));
294                 } else if (deviceToUpdate.getData().getClass()==Short.class) {
295                     deviceToUpdate.setData(new Short(buffer.getShort()));
296                 } else if (deviceToUpdate.getData().getClass()==Integer.class) {
297                     deviceToUpdate.setData(new Integer(buffer.getInt()));
298                 } else if (deviceToUpdate.getData().getClass()==Float.class) {
299                     deviceToUpdate.setData(new Float(buffer.getFloat()));
300                 } else if (deviceToUpdate.getData().getClass()==Double.class) {
301                     deviceToUpdate.setData(new Double(buffer.getFloat()));
302                 } else if (deviceToUpdate.getData().getClass()==String.class) {
303                     deviceToUpdate.setData(new String(parameterList.get(x).getParameterData
    (),StandardCharsets.US_ASCII));
304                 } else if (deviceToUpdate.getData().getClass()==Boolean.class) {
305                     if (buffer.get()==(byte)0xFF) {
306                         deviceToUpdate.setData(new Boolean(true));
307                     } else {
308                         deviceToUpdate.setData(new Boolean(false));
309                     }
310                 }
311
312
313                 devicesList.set( x, deviceToUpdate );
314             }
315         }
316
317         public void updateDeviceValue(byte internalIndex, byte[] data) {
318             for (int x = 0; x < devicesList.size(); x++) {
319                 if (devicesList.get(x).getInternalIndex()==internalIndex) {
320                     Device deviceToUpdate = devicesList.get(x);
321                     ByteBuffer buffer = ByteBuffer.wrap(data);
322                     buffer.order(ByteOrder.LITTLE_ENDIAN);
323
324                     if (deviceToUpdate.getData().getClass()==Byte.class) {
325                         deviceToUpdate.setData(new Byte(buffer.get()));
326                     } else if (deviceToUpdate.getData().getClass()==Short.class) {
327                         deviceToUpdate.setData(new Short(buffer.getShort()));
328                     } else if (deviceToUpdate.getData().getClass()==Integer.class) {
329                         deviceToUpdate.setData(new Integer(buffer.getInt()));
330                     } else if (deviceToUpdate.getData().getClass()==Float.class) {
331                         deviceToUpdate.setData(new Float(buffer.getFloat()));
332                     } else if (deviceToUpdate.getData().getClass()==Double.class) {
333                         deviceToUpdate.setData(new Double(buffer.getFloat()));
334                     } else if (deviceToUpdate.getData().getClass()==String.class) {
335                         deviceToUpdate.setData(new String(data,StandardCharsets.US_ASCII));
336                     } else if (deviceToUpdate.getData().getClass()==Boolean.class) {
337                         if (buffer.get()==(byte)0xFF) {
338                             deviceToUpdate.setData(new Boolean(true));
339                         } else {
340                             deviceToUpdate.setData(new Boolean(false));
341                         }
342                     }
343                 }
344             }
345         }
346
```

```
347
348         public byte[] exportDeviceValue(byte internalIndex) {
349             for (int x = 0; x < devicesList.size(); x++) {
350                 if (devicesList.get(x).getInternalIndex()==internalIndex) {
351
352                     if (devicesList.get(x).getData().getClass()==Byte.class) {
353                         ByteBuffer byteBuffer = ByteBuffer.allocate(Byte.BYTES);
354                         byteBuffer.order(ByteOrder.LITTLE_ENDIAN);
355                         byteBuffer.put((byte) devicesList.get(x).getData());
356                         return byteBuffer.array();
357                     } else if (devicesList.get(x).getData().getClass()==Short.class) {
358                         ByteBuffer shortBuffer = ByteBuffer.allocate(Short.BYTES);
359                         shortBuffer.order(ByteOrder.LITTLE_ENDIAN);
360                         shortBuffer.putShort((short) devicesList.get(x).getData());
361                         return shortBuffer.array();
362                     } else if (devicesList.get(x).getData().getClass()==Integer.class) {
363                         ByteBuffer intBuffer = ByteBuffer.allocate(Integer.BYTES);
364                         intBuffer.order(ByteOrder.LITTLE_ENDIAN);
365                         intBuffer.putInt((int) devicesList.get(x).getData());
366                         return intBuffer.array();
367                     } else if (devicesList.get(x).getData().getClass()==Float.class) {
368                         ByteBuffer floatBuffer = ByteBuffer.allocate(Float.BYTES);
369                         floatBuffer.order(ByteOrder.LITTLE_ENDIAN);
370                         floatBuffer.putFloat((float) devicesList.get(x).getData());
371                         return floatBuffer.array();
372                     } else if (devicesList.get(x).getData().getClass()==Double.class) {
373                         ByteBuffer doubleBuffer = ByteBuffer.allocate(Float.BYTES);
374                         doubleBuffer.order(ByteOrder.LITTLE_ENDIAN);
375                         doubleBuffer.putFloat((float)((double)devicesList.get(x).getData())
    );
376                         return doubleBuffer.array();
377                     } else if (devicesList.get(x).getData().getClass()==String.class) {
378                         String data = (String) devicesList.get(x).getData();
379                         return data.getBytes(StandardCharsets.US_ASCII);
380                     } else if (devicesList.get(x).getData().getClass()==Boolean.class) {
381                         byte data = (byte) 0x00;
382                         if ((Boolean)devicesList.get(x).getData()==true) data = (byte)0xFF;
383                         ByteBuffer booleanBuffer = ByteBuffer.allocate(Byte.BYTES);
384                         booleanBuffer.order(ByteOrder.LITTLE_ENDIAN);
385                         booleanBuffer.put(data);
386                         return booleanBuffer.array();
387                     }
388                 }
389             }
390
391             return null;
392         }
393
394
395         private static Executor synchronizationExecutor = Executors.newSingleThreadExecutor
    ();
396         private static boolean synchronizationRunning = false;
397
398         public static boolean awaitingModuleResponse = false;
399         public static byte awaitingModuleResponse_ModuleInternalCode = (byte)0x00;
400         public static boolean awaitingSynchronization = false;
401
402         public static void syncAllDevicesValueToSystem(){
403             // Send data to system from internal data
404
405
406             if (synchronizationRunning==false) {
407                 synchronizationExecutor.execute(new Runnable() {
```

```
408                     @Override
409                     public void run() {
410                         synchronizationRunning = true;
411                         do {
412                             awaitingSynchronization=false;
413                             for (Module mod : modulesList) {
414                                 for (Module.Device dev : mod.getDevicesList()) {
415                                     if (dev.isSynchronizableOnlyFromSystem() == false &&
    dev.isSynchronizedData() == false) {
416                                         List<Parameter> export = mod.
    exportDataToParameterList();
417 
418                                         ComposeMessageToBuffer(CommandType.
    UPDATE_ALL_DEVICES_VALUE, mod, export);
419                                         DeveloperConsoleActivity.sendTextToTerminal("Sent
    to: " + mod.getModuleName() + " : " + TerminalManager.insertSpaces(bytesToHex(
    command_buffer), " ", 2));
420                                         awaitingModuleResponse = true;
421                                         awaitingModuleResponse_ModuleInternalCode = mod.
    internalCode;
422                                         BluetoothConnection.sendCommandBuffer(
    command_buffer);
423                                         long startTimeMillis = System.currentTimeMillis();
424                                         while ((System.currentTimeMillis() -
    startTimeMillis) < 1000L) {
425                                             if (awaitingModuleResponse == false) {
426                                                 for (Module.Device devSync : mod.
    getDevicesList()) {
427                                                     devSync.setSynchronizedState(true);
428                                                 }
429                                                 break;
430                                             }
431                                         }
432                                         awaitingModuleResponse = false;
433                                         break; //Only send one message per module
434                                     }
435                                 }
436                             }
437                         } while (awaitingSynchronization==true);
438                         synchronizationRunning = false;
439                     }
440                 });
441             } else {
442                 awaitingSynchronization = true;
443             }
444 
445         }
446 
447         public static void syncAllDevicesValueFromSystem(){
448             // Retrieve data from system to internal data
449             // Metodo estatico
450 
451 
452             synchronizationRunning = true;
453 
454             synchronizationRunning = false;
455         }
456 
457 
458 
459 
460         public static Module getModule(byte internalCode) {
461             for (Module mod : modulesList) {
```

```
462                 if (mod.getInternalCode()==internalCode) return mod;
463             }
464             return null;
465         }
466 
467         public static Module getLocalModule() {
468             byte localModuleInternalCode = (byte) 0x00;
469             for (Module mod : modulesList) {
470                 if (mod.getInternalCode()==localModuleInternalCode) return mod;
471             }
472             return null;
473         }
474 
475         public String getModuleStatusTerminal() {
476 
477             String composedString = new String("");
478 
479 
480 
481             for (Device dev : devicesList) {
482                 composedString += dev.getName() + " : " + dev.getData().toString() + System
    .lineSeparator();
483             }
484 
485 
486 
487             return composedString;
488         }
489 
490 
491         public static class Device{
492             private String name;
493             public Object data;
494             private int iconResourceId;
495             private byte internalIndex;     // Index or internal code of the device
496             private boolean synchronizedData = false;
497             private boolean synchronizableOnlyFromSystem = false;   // Only synchronize
    from system
498 
499             Device(String name, Object data, int iconResourceId, byte internalIndex,
    boolean synchronizableOnlyFromSystem){
500                 this.name = name;
501                 this.data = data;
502                 this.iconResourceId = iconResourceId;
503                 this.internalIndex = internalIndex;
504                 this.synchronizableOnlyFromSystem = synchronizableOnlyFromSystem;
505             }
506 
507             public void setData(Object data) {
508                 if (synchronizableOnlyFromSystem == false) {
509                     CommandsActivity.updateGUI();
510                     this.data = data;
511                     synchronizedData = false;
512                 }
513             }
514 
515             public Object getData() {
516                 return data;
517             }
518 
519             public int getIconResourceId() {
520                 return iconResourceId;
521             }
```

```
522
523             public byte getInternalIndex() {
524                 return internalIndex;
525             }
526
527             public String getName() {
528                 return name;
529             }
530
531             public boolean isSynchronizableOnlyFromSystem() {
532                 return synchronizableOnlyFromSystem;
533             }
534
535             public boolean isSynchronizedData() {
536                 return synchronizedData;
537             }
538
539             public void setSynchronizedState(boolean synchronizedData) {
540                 this.synchronizedData = synchronizedData;
541             }
542
543             public void deviceOnClickHandler(){
544                 switch (name){
545                     case "Módulo principal":
546                         break;
547                     case "Módulo de potencia":
548                         break;
549                     case "Módulo motriz":
550                         break;
551
552                     case "Velador":
553                         if ((boolean)data == false) { setData(Boolean.TRUE); } else {
    setData(Boolean.FALSE); }
554                         break;
555                     case "Luz":
556                         if ((boolean)data == false) { setData(Boolean.TRUE); } else {
    setData(Boolean.FALSE); }
557                         break;
558                     case "Ventilador":
559                         if ((boolean)data == false) { setData(Boolean.TRUE); } else {
    setData(Boolean.FALSE); }
560                         break;
561                     case "Estufa":
562                         if ((boolean)data == false) { setData(Boolean.TRUE); } else {
    setData(Boolean.FALSE); }
563                         break;
564                     case "Camilla":
565                         if (byteToUnsignedInt((byte)data)<3) { setData((byte)(((byte)data)+
    1)); } else { setData((byte)0x00); }
566                         break;
567                     case "Cortina":
568                         if (byteToUnsignedInt((byte)data)<7) { setData((byte)(((byte)data)+
    1)); } else { setData((byte)0x00); }
569                         break;
570                     case "Llamar enfermera":
571                         if ((boolean)data == false) {
572                             setData(Boolean.TRUE);
573
574                             autoToggleNurseStateThread = new Thread(
    autoToggleNurseStateRunnable);
575                             autoToggleNurseStateThread.start();
576                         } else { setData(Boolean.FALSE); }
577                         break;
```

```
578                 }
579                 syncAllDevicesValueToSystem();
580             }
581 
582             public int getProgressBarValue(){
583                 int returnValue = 0;
584                 switch (name){
585                     case "Módulo principal":
586                         if ((boolean)data == false) { returnValue = 0; } else { returnValue
    = 100; }
587                         break;
588                     case "Módulo de potencia":
589                         if ((boolean)data == false) { returnValue = 0; } else { returnValue
    = 100; }
590                         break;
591                     case "Módulo motriz":
592                         if ((boolean)data == false) { returnValue = 0; } else { returnValue
    = 100; }
593                         break;
594 
595                     case "Velador":
596                         if ((boolean)data == false) { returnValue = 0; } else { returnValue
    = 100; }
597                         break;
598                     case "Luz":
599                         if ((boolean)data == false) { returnValue = 0; } else { returnValue
    = 100; }
600                         break;
601                     case "Ventilador":
602                         if ((boolean)data == false) { returnValue = 0; } else { returnValue
    = 100; }
603                         break;
604                     case "Estufa":
605                         if ((boolean)data == false) { returnValue = 0; } else { returnValue
    = 100; }
606                         break;
607 
608                     case "Camilla":
609                         returnValue = Math.round((float)(((byte)data)*33.333f));
610                         break;
611                     case "Cortina":
612                         returnValue = Math.round((float)(((byte)data)*14.286f));;
613                         break;
614                     case "Llamar enfermera":
615                         if ((boolean)data == false) { returnValue = 0; } else { returnValue
    = 100; }
616                         break;
617                 }
618                 return returnValue;
619             }
620 
621             public String getDataAsString(){
622                 String returnString = "";
623                 switch (name){
624                     case "Módulo principal":
625                         if ((boolean)data == false) { returnString = "Desconectado"; } else
    { returnString = "En linea"; }
626                         break;
627                     case "Módulo de potencia":
628                         if ((boolean)data == false) { returnString = "Desconectado"; } else
    { returnString = "En linea"; }
629                         break;
630                     case "Módulo motriz":
```

```
631                         if ((boolean)data == false) { returnString = "Desconectado"; } else
     { returnString = "En linea"; }
632                         break;
633
634                     case "Velador":
635                         if ((boolean)data == false) { returnString = "Apagado"; } else {
    returnString = "Encendido"; }
636                         break;
637                     case "Luz":
638                         if ((boolean)data == false) { returnString = "Apagado"; } else {
    returnString = "Encendido"; }
639                         break;
640                     case "Ventilador":
641                         if ((boolean)data == false) { returnString = "Apagado"; } else {
    returnString = "Encendido"; }
642                         break;
643                     case "Estufa":
644                         if ((boolean)data == false) { returnString = "Apagado"; } else {
    returnString = "Encendido"; }
645                         break;
646
647                     case "Camilla":
648                         returnString = "Posición " + byteToUnsignedInt((byte)data);
649                         break;
650                     case "Cortina":
651                         returnString = "Posición " + byteToUnsignedInt((byte)data);
652                         break;
653                     case "Llamar enfermera":
654                         if ((boolean)data == false) { returnString = "Disponible"; } else {
     returnString = "Llamando"; }
655                         break;
656                 }
657                 return returnString;
658             }
659
660             public static Thread autoToggleNurseStateThread;
661             Runnable autoToggleNurseStateRunnable = new Runnable() {
662                 @Override
663                 public void run() {
664                     try {
665                         Thread.sleep(3000);
666                         modulesList.get(3).getDevicesList().get(2).setData(new Boolean(
    false));
667                         modulesList.get(3).getDevicesList().get(2).setSynchronizedState(
    true);
668                         CommandsActivity.updateGUI();
669                     } catch (InterruptedException e) {
670                         e.printStackTrace();
671                     }
672                 }
673             };
674
675         }
676
677     }
678
679     enum SectionDividersChar {
680         SOH((byte)0x01),
681         STX((byte)0x02),
682         ETX((byte)0x03),
683         ETB((byte)0x17);
684
685         private final byte code;
```

```
686         SectionDividersChar(byte code){
687             this.code = code;
688         }
689         public byte getCode() {
690             return code;
691         }
692     }
693 
694     public static class CommandType{
695         private byte commandCode;
696         private String commandTag;
697         private Runnable handlerRunnable;
698 
699 
700         CommandType(byte cmdCode, String tag, Runnable handlerCommand){
701             this.commandCode = cmdCode;
702             this.commandTag = tag;
703             this.handlerRunnable = handlerCommand;
704         }
705 
706         public byte getCommandCode() {
707             return commandCode;
708         }
709 
710         public String getCommandTag() {
711             return commandTag;
712         }
713 
714         public void setCommandCode(byte commandCode) {
715             this.commandCode = commandCode;
716         }
717 
718         public void setCommandTag(String commandTag) {
719             this.commandTag = commandTag;
720         }
721 
722         public void handleCommand() {
723             handlerRunnable.run();
724         }
725 
726 
727         public static ArrayList<CommandType> definedCommandList = new ArrayList<CommandType
    >(Arrays.asList(
728                 new CommandType((byte)0x00, "UPDATE_ALL_DEVICES_VALUE", new Runnable() {
729                     @Override
730                     public void run() {
731 
732                         modulesList.get(byteToUnsignedInt(lastMessageTransmitterModule.
    getInternalCode())).updateDataFromParameterList(lastParameterList);
733 
734                         System.out.println("Running UPDATE_ALL_DEVICES_VALUE!");
735 
736                     }}),
737                 new CommandType((byte)0x01, "UPDATE_DEVICE_VALUE", new Runnable() {
738                     @Override
739                     public void run() {
740 
741 
742                         ByteBuffer buffer = ByteBuffer.wrap(lastParameterList.get(0).
    getParameterData());
743 
744                         modulesList.get(byteToUnsignedInt(lastMessageTransmitterModule.
    getInternalCode())).updateDeviceValue(buffer.get(),lastParameterList.get(1).
```

```
744 getParameterData());
745
746                         System.out.println("Running UPDATE_DEVICE_VALUE!");
747
748
749
750                     }}),
751                 new CommandType((byte)0x02, "GET_ALL_DEVICES_VALUE", new Runnable() {
752                     @Override
753                     public void run() {
754                         ComposeMessageToBuffer(definedCommandList.stream().filter(p-> p.
    getCommandTag().equals(("UPDATE_ALL_DEVICES_VALUE"))).findFirst().orElse(null),
755                                 lastMessageTransmitterModule,
756                                 modulesList.get(byteToUnsignedInt(
    lastMessageTransmitterModule.getInternalCode())).exportDataToParameterList());
757
758                         // SEND COMMAND_BUFFER VIA BLUETOOTH
759
760
761                         System.out.println("Running GET_ALL_DEVICES_VALUE!");
762                     }}),
763                 new CommandType((byte)0x03, "GET_DEVICE_VALUE", new Runnable() {
764                     @Override
765                     public void run() {
766
767                         ByteBuffer buffer = ByteBuffer.wrap(lastParameterList.get(0).
    getParameterData());
768                         byte[] deviceData = modulesList.get(byteToUnsignedInt(
    lastMessageTransmitterModule.getInternalCode())).exportDeviceValue(buffer.get());
769
770                         ComposeMessageToBuffer(definedCommandList.stream().filter(p-> p.
    getCommandTag().equals(("UPDATE_DEVICE_VALUE"))).findFirst().orElse(null),
771                                 lastMessageTransmitterModule,
772                                 new ArrayList<Parameter>(Arrays.asList(new Parameter(new
    byte[] {buffer.get()}), new Parameter(deviceData))));
773
774                         // SEND COMMAND_BUFFER VIA BLUETOOTH
775
776                         System.out.println("Running GET_DEVICE_VALUE!");
777                     }}),
778                 new CommandType((byte)0x04, "MESSAGE_STATUS", new Runnable() {
779                     @Override
780                     public void run() {
781
782                         ByteBuffer buffer = ByteBuffer.wrap(lastParameterList.get(0).
    getParameterData());
783                         if (buffer.get()==(byte)0x09){
784                             Module.awaitingModuleResponse = false;
785                         }
786
787
788
789
790                     }})
791         ));
792
793         public static CommandType UPDATE_ALL_DEVICES_VALUE = definedCommandList.get(0);
794         public static CommandType UPDATE_DEVICE_VALUE = definedCommandList.get(1);
795         public static CommandType GET_ALL_DEVICES_VALUE = definedCommandList.get(2);
796         public static CommandType GET_DEVICE_VALUE = definedCommandList.get(3);
797         public static CommandType MESSAGE_STATUS = definedCommandList.get(4);
798
799     }
```

```
800
801     public static int byteToUnsignedInt(byte x){
802         return ((int) x) & 0xff;
803     }
804
805
806     private static final char[] HEX_ARRAY = "0123456789ABCDEF".toCharArray();
807     public static String bytesToHex(byte[] bytes) {
808         char[] hexChars = new char[bytes.length * 2];
809         for (int j = 0; j < bytes.length; j++) {
810             int v = bytes[j] & 0xFF;
811             hexChars[j * 2] = HEX_ARRAY[v >>> 4];
812             hexChars[j * 2 + 1] = HEX_ARRAY[v & 0x0F];
813         }
814         return new String(hexChars);
815     }
816
817
818
819
820 }
821
```