

Archivo - Application.java

```
1 package com.example.SAPLM;
2
3 import android.app.Activity;
4 import androidx.lifecycle.LifecycleObserver;
5 import android.bluetooth.BluetoothAdapter;
6 import android.bluetooth.BluetoothDevice;
7 import android.content.Context;
8 import android.content.IntentFilter;
9 import android.os.Bundle;
10 import android.widget.Toast;
11
12 import com.example.SAPLM.bluetoothActivities.BluetoothConnection;
13 import com.example.SAPLM.settingsActivities.DeveloperConsoleActivity;
14
15 import java.io.IOException;
16
17 public class Application extends android.app.Application implements LifecycleObserver {
18
19     @Override
20     public void onCreate() {
21         super.onCreate();
22
23         // ProcessLifecycleOwner.get().getLifecycle().addObserver(this);
24
25         IntentFilter bluetoothAdapterActionChangedFilter = new IntentFilter(BluetoothAdapter
26             .ACTION_STATE_CHANGED);
26         registerReceiver(BluetoothConnection.bluetoothAdapterStatusBR,
27             bluetoothAdapterActionChangedFilter);
28
29         IntentFilter bluetoothScanModeStatusChangedFilter = new IntentFilter();
30         bluetoothScanModeStatusChangedFilter.addAction(BluetoothAdapter.
31             ACTION_DISCOVERY_STARTED);
30         bluetoothScanModeStatusChangedFilter.addAction(BluetoothAdapter.
32             ACTION_DISCOVERY_FINISHED);
31         bluetoothScanModeStatusChangedFilter.addAction(BluetoothAdapter.
32             ACTION_SCAN_MODE_CHANGED);
32         registerReceiver(BluetoothConnection.bluetoothScanModeStatusBR,
33             bluetoothScanModeStatusChangedFilter);
33
34         IntentFilter bluetoothDeviceStatusChangedFilter = new IntentFilter();
35         bluetoothDeviceStatusChangedFilter.addAction(BluetoothDevice.ACTION_ACL_CONNECTED);
36         bluetoothDeviceStatusChangedFilter.addAction(BluetoothDevice.ACTION_ACL_DISCONNECTED
37 );
37         bluetoothDeviceStatusChangedFilter.addAction(BluetoothDevice.
38             ACTION_ACL_DISCONNECT_REQUESTED);
38         bluetoothDeviceStatusChangedFilter.addAction(BluetoothDevice.
39             ACTION_BOND_STATE_CHANGED);
39         registerReceiver(BluetoothConnection.bluetoothDeviceStatusBR,
40             bluetoothDeviceStatusChangedFilter);
40
41         Thread internetDetector = new Thread(){
42             @Override
43             public void run() {
44                 super.run();
45                 while(true){
46                     try {
47                         internetConnectionStatus = isConnected();
48                         Thread.sleep(10000);
49                         if (internetConnectionStatus) {Application.toastMessage("Internet
50 connection detected");}
50                         else {Application.toastMessage("Proper internet connection has not
51 been detected");}
51                     } catch (Throwable th) { th.printStackTrace(); Application.toastMessage(
```

Archivo - Application.java

```
51 th.toString());}
52 }
53 }
54 };
55
56 internetDetector.start();
57
58 Application.persistentDataSave.putString("MAIN_TEXT_KEY", new String());
59 Application.persistentDataSave.putString("ONGOING_TEXT_KEY", new String());
60
61 }
62
63 @Override
64 public void onTerminate() {
65     super.onTerminate();
66 }
67
68 private static Activity mActivity;
69 private static Context mContext;
70 public static Bundle persistentDataSave = new Bundle();
71
72
73 public static Activity getActivity() { return mActivity; }
74 public static void setActivity(Activity mActivitySet) { mActivity = mActivitySet; }
75
76 public static Context getContext() { return mContext; }
77 public static void setContext(Context mContexttoSet) { mContext = mContexttoSet; }
78
79 public static void toastMessage(final String message){
80     mActivity.runOnUiThread(new Runnable() {
81         @Override
82         public void run() {
83             if (mActivity != null) {
84                 //Toast.makeText(mActivity, message, Toast.LENGTH_SHORT).show();
85             }
86         }
87     });
88 }
89
90 private static boolean internetConnectionStatus;
91
92 public static boolean isSystemOnline(){
93     return internetConnectionStatus;
94 }
95
96 public boolean isConnected() throws InterruptedException, IOException {
97     final String command = "ping -c 1 google.com";
98     return Runtime.getRuntime().exec(command).waitFor() == 0;
99 }
100
101
102
103
104
105
106 }
107
```

## Archivo - MainActivity.java

```
1 package com.example.SAPLM;
2
3 import android.Manifest;
4 import android.content.pm.PackageManager;
5 import android.os.Bundle;
6 import androidx.core.app.ActivityCompat;
7 import androidx.core.content.ContextCompat;
8 import androidx.appcompat.app.AppCompatActivity;
9 import android.util.DisplayMetrics;
10 import android.view.View;
11 import android.widget.ImageButton;
12 import android.widget.TextView;
13
14 import com.example.SAPLM.bluetoothActivities.BluetoothConnection;
15 import com.example.SAPLM.settingsActivities.settings_layout;
16
17 public class MainActivity extends AppCompatActivity {
18
19     static ImageButton imgBtn_microphone;
20     static ImageButton imgBtn_microphoneBackground;
21     static ImageButton imgBtn_commands;
22     static ImageButton imgBtn_contacts;
23     static ImageButton imgBtn_bluetooth;
24     static ImageButton imgBtn_settings;
25     static ImageButton imgBtn_menu;
26     static MainMenuHandler mainMenuHandler;
27     static TextView textView_speechRecognitionResults;
28
29
30     @Override
31     protected void onCreate(Bundle savedInstanceState) {
32         super.onCreate(savedInstanceState);
33         setContentView(R.layout.activity_main);
34
35         if(savedInstanceState == null) {
36
37             mainMenuHandler = new MainMenuHandler(this, this);
38             Application.setActivity(this);
39             Application.setContext(this);
40             MainMenuHandler.dpiConstant = ((float) this.getResources().getDisplayMetrics().densityDpi / DisplayMetrics.DENSITY_DEFAULT);
41
42             imgBtn_microphone = findViewById(R.id.imgBtn_microphone);
43             imgBtn_microphoneBackground = findViewById(R.id.imgBtn_microphone_background);
44             imgBtn_commands = findViewById(R.id.imgBtn_commands);
45             imgBtn_contacts = findViewById(R.id.imgBtn_contacts);
46             imgBtn_bluetooth = findViewById(R.id.imgBtn_bluetooth);
47             imgBtn_settings = findViewById(R.id.imgBtn_settings);
48             imgBtn_menu = findViewById(R.id.imgBtn_menu);
49             textView_speechRecognitionResults = findViewById(R.id.textView_speechRecognitionResults);
50
51             if (ContextCompat.checkSelfPermission(this,
52                 Manifest.permission.RECORD_AUDIO)
53                 != PackageManager.PERMISSION_GRANTED) {
54                 if (ActivityCompat.shouldShowRequestPermissionRationale(this,
55                     Manifest.permission.RECORD_AUDIO)) {
56                 } else {
57                     ActivityCompat.requestPermissions(this,
58                         new String[]{Manifest.permission.RECORD_AUDIO},
59                         527);
60                 }
61             } else {
```

Archivo - MainActivity.java

```
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
98 public void onClickListener_settings(View view){ mainMenuHandler.settingsButtonOnClick();
99 )
100
101 public void onClickListener_contacts(View view){ mainMenuHandler.contactsButtonOnClick();
102 )
103
104 public void onClickListener_bluetooth(View view){ mainMenuHandler.
105 bluetoothButtonOnClick(); }
106
107
108 public void onDestroy() {
109
110     PocketsphinxListener.shutdownRecognizer();
111     GoogleSpeechRecognizer.shutdownRecognizer();
112     BluetoothConnection.closeAllConnections();
113     TextToSpeechHandler.shutdownTextSpeech();
114
115     super.onDestroy();
116 }
117
```

```

1 package com.example.SAPLM;
2
3 import android.animation.ValueAnimator;
4 import android.app.Activity;
5 import android.content.Context;
6 import android.content.Intent;
7 import android.view.View;
8 import android.view.animation.AccelerateDecelerateInterpolator;
9
10 import com.example.SAPLM.bluetoothActivities.BluetoothActivity;
11 import com.example.SAPLM.commandActivity.CommandsActivity;
12 import com.example.SAPLM.settingsActivities.settings_layout;
13
14 public class MainMenuHandler extends MainActivity{
15
16     public static float dpiConstant;
17     public static float radialMenuRadius = 263.0f;
18     public static boolean menuUncoiled = false;
19     public static boolean isAnimationRunning = false;
20     public static Context mainMenuContext;
21     public static Activity mainMenuActivity;
22
23     public MainMenuHandler(Context context, Activity activity){
24         this.mainMenuActivity = activity;
25         this.mainMenuContext = context;
26     }
27
28     public static void menuButtonOnClick(){
29         long animationDuration = 850;
30
31         if(isAnimationRunning==false) {
32             isAnimationRunning = true;
33
34             if (menuUncoiled == false) {
35                 ValueAnimator animatorCommands = ValueAnimator.ofFloat(360, 72);
36                 animatorCommands.setDuration(animationDuration);
37                 animatorCommands.setInterpolator(new AccelerateDecelerateInterpolator());
38                 animatorCommands.addUpdateListener(new ValueAnimator.AnimatorUpdateListener(
39                 {
40                     @Override
41                     public void onAnimationUpdate(ValueAnimator animation) {
42                         float value = ((Float) (animation.getAnimatedValue()))
43                             .floatValue();
44
45                         float translationX = (float) (Math.cos(value * (Math.PI / 180.0f)) *
46 (radialMenuRadius / 2.0f) - (radialMenuRadius / 2.0f));
47                         float translationY = (float) (Math.sqrt(-Math.pow(translationX, 2.0d
48 ) - radialMenuRadius * translationX));
49                         if (value > 180.0f) {
50                             translationY = -translationY;
51                         }
52
53                         imgBtn_commands.setTranslationX(convertDpToPixel(translationX));
54                         imgBtn_commands.setTranslationY(convertDpToPixel(translationY));
55
56                     });
57
58                     ValueAnimator animatorContacts = ValueAnimator.ofFloat(360, 144);
59                     animatorContacts.setDuration(animationDuration);
60                     animatorContacts.setInterpolator(new AccelerateDecelerateInterpolator());
61                 }
62             }
63         }
64     }
65
66     imgBtn_commands.setVisibility(View.VISIBLE);
67     animatorCommands.start();
68
69     ValueAnimator animatorContacts = ValueAnimator.ofFloat(360, 144);
70     animatorContacts.setDuration(animationDuration);
71     animatorContacts.setInterpolator(new AccelerateDecelerateInterpolator());
72
73 }

```

```

61             animatorContacts.addUpdateListener(new ValueAnimator.AnimatorUpdateListener
62     () {
63         @Override
64         public void onAnimationUpdate(ValueAnimator animation) {
65             float value = ((Float) (animation.getAnimatedValue()))
66                 .floatValue();
67
68             float translationX = (float) (Math.cos(value * (Math.PI / 180.0f))
69             * (radialMenuRadius / 2.0f) - (radialMenuRadius / 2.0f));
70             float translationY = (float) (Math.sqrt(-Math.pow(translationX, 2.
71             0d) - radialMenuRadius * translationX));
72             if (value > 180.0f) {
73                 translationY = -translationY;
74             }
75
76             imgBtn_contacts.setTranslationX(convertDpToPixel(translationX));
77             imgBtn_contacts.setTranslationY(convertDpToPixel(translationY));
78
79         }
80
81     });
82     imgBtn_contacts.setVisibility(View.VISIBLE);
83     animatorContacts.start();
84
85     ValueAnimator animatorSettings = ValueAnimator.ofFloat(360, 216);
86     animatorSettings.setDuration(animationDuration);
87     animatorSettings.setInterpolator(new AccelerateDecelerateInterpolator());
88     animatorSettings.addUpdateListener(new ValueAnimator.AnimatorUpdateListener
89     () {
90         @Override
91         public void onAnimationUpdate(ValueAnimator animation) {
92             float value = ((Float) (animation.getAnimatedValue()))
93                 .floatValue();
94
95             float translationX = (float) (Math.cos(value * (Math.PI / 180.0f))
96             * (radialMenuRadius / 2.0f) - (radialMenuRadius / 2.0f));
97             float translationY = (float) (Math.sqrt(-Math.pow(translationX, 2.
98             0d) - radialMenuRadius * translationX));
99             if (value > 180.0f) {
100                 translationY = -translationY;
101             }
102
103             imgBtn_settings.setTranslationX(convertDpToPixel(translationX));
104             imgBtn_settings.setTranslationY(convertDpToPixel(translationY));
105
106         }
107     });
108     imgBtn_settings.setVisibility(View.VISIBLE);
109     animatorSettings.start();
110
111     ValueAnimator animatorBluetooth = ValueAnimator.ofFloat(360, 288);
112     animatorBluetooth.setDuration(animationDuration);
113     animatorBluetooth.setInterpolator(new AccelerateDecelerateInterpolator());
114     animatorBluetooth.addUpdateListener(new ValueAnimator.
115         AnimatorUpdateListener() {
116             @Override
117             public void onAnimationUpdate(ValueAnimator animation) {
118                 float value = ((Float) (animation.getAnimatedValue()))
119                     .floatValue();
120
121                 float translationX = (float) (Math.cos(value * (Math.PI / 180.0f))
122                 * (radialMenuRadius / 2.0f) - (radialMenuRadius / 2.0f));
123                 float translationY = (float) (Math.sqrt(-Math.pow(translationX, 2.
124                 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).
128             setInterpolator(new AccelerateDecelerateInterpolator());
129             imgBtn_contacts.animate().alpha(1.0f).setDuration(animationDuration).
130             setInterpolator(new AccelerateDecelerateInterpolator());
131             imgBtn_settings.animate().alpha(1.0f).setDuration(animationDuration).
132             setInterpolator(new AccelerateDecelerateInterpolator());
133             imgBtn_bluetooth.animate().alpha(1.0f).setDuration(animationDuration).
134             setInterpolator(new AccelerateDecelerateInterpolator()).withEndAction(new Runnable() {
135
136                 @Override
137                 public void run() {
138                     isAnimationRunning = false;
139                 }
140             });
141
142             menuUncoiled = true;
143
144         } else {
145
146             ValueAnimator animatorCommands = ValueAnimator.ofFloat(72, 360);
147             animatorCommands.setDuration(animationDuration);
148             animatorCommands.setInterpolator(new AccelerateDecelerateInterpolator());
149             animatorCommands.addUpdateListener(new ValueAnimator.AnimatorUpdateListener
150             () {
151
152                 @Override
153                 public void onAnimationUpdate(ValueAnimator animation) {
154                     float value = ((Float) (animation.getAnimatedValue()))
155                         .floatValue();
156
157                     float translationX = (float) (Math.cos(value * (Math.PI / 180.0f))
158 * (radialMenuRadius / 2.0f) - (radialMenuRadius / 2.0f));
159                     float translationY = (float) (Math.sqrt(-Math.pow(translationX, 2.
160 0d) - radialMenuRadius * translationX));
161                     if (value > 180.0f) {
162                         translationY = -translationY;
163                     }
164
165                     imgBtn_commands.setTranslationX(convertDpToPixel(translationX));
166                     imgBtn_commands.setTranslationY(convertDpToPixel(translationY));
167
168                 }
169             });
170             imgBtn_commands.setVisibility(View.VISIBLE);
171             animatorCommands.start();
172
173             ValueAnimator animatorContacts = ValueAnimator.ofFloat(144, 360);
174             animatorContacts.setDuration(animationDuration);
175             animatorContacts.setInterpolator(new AccelerateDecelerateInterpolator());
176             animatorContacts.addUpdateListener(new ValueAnimator.AnimatorUpdateListener
177             () {

```

## Archivo - MainMenuHandler.java

```
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))
176             * (radialMenuRadius / 2.0f) - (radialMenuRadius / 2.0f));
177         float translationY = (float) (Math.sqrt(-Math.pow(translationX, 2.
178             0d) - radialMenuRadius * translationX));
179         if (value > 180.0f) {
180             translationY = -translationY;
181         }
182
183         imgBtn_contacts.setTranslationX(convertDpToPixel(translationX));
184         imgBtn_contacts.setTranslationY(convertDpToPixel(translationY));
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     () {
194         @Override
195         public void onAnimationUpdate(ValueAnimator animation) {
196             float value = ((Float) (animation.getAnimatedValue()))
197                 .floatValue();
198
199             float translationX = (float) (Math.cos(value * (Math.PI / 180.0f))
200                 * (radialMenuRadius / 2.0f) - (radialMenuRadius / 2.0f));
201             float translationY = (float) (Math.sqrt(-Math.pow(translationX, 2.
202                 0d) - radialMenuRadius * translationX));
203             if (value > 180.0f) {
204                 translationY = -translationY;
205             }
206
207             imgBtn_settings.setTranslationX(convertDpToPixel(translationX));
208             imgBtn_settings.setTranslationY(convertDpToPixel(translationY));
209
210         }
211     });
212     imgBtn_settings.setVisibility(View.VISIBLE);
213     animatorSettings.start();
214
215     ValueAnimator animatorBluetooth = ValueAnimator.ofFloat(288, 360);
216     animatorBluetooth.setDuration(animationDuration);
217     animatorBluetooth.setInterpolator(new AccelerateDecelerateInterpolator());
218     animatorBluetooth.addUpdateListener(new ValueAnimator.
219     AnimatorUpdateListener() {
220         @Override
221         public void onAnimationUpdate(ValueAnimator animation) {
222             float value = ((Float) (animation.getAnimatedValue()))
223                 .floatValue();
224
225             float translationX = (float) (Math.cos(value * (Math.PI / 180.0f))
226                 * (radialMenuRadius / 2.0f) - (radialMenuRadius / 2.0f));
227             float translationY = (float) (Math.sqrt(-Math.pow(translationX, 2.
228                 0d) - radialMenuRadius * translationX));
229             if (value > 180.0f) {
230                 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).
236     setInterpolator(new AccelerateDecelerateInterpolator());
237     imgBtn_contacts.animate().alpha(0.0f).setDuration(animationDuration).
238     setInterpolator(new AccelerateDecelerateInterpolator());
239     imgBtn_settings.animate().alpha(0.0f).setDuration(animationDuration).
240     setInterpolator(new AccelerateDecelerateInterpolator());
241     imgBtn_bluetooth.animate().alpha(0.0f).setDuration(animationDuration).
242     setInterpolator(new AccelerateDecelerateInterpolator()).withEndAction(new Runnable() {
243         @Override
244         public void run() {
245             isAnimationRunning = 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
```

Archivo - MainMenuHandler.java

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

Archivo - ContactsActivity.java

```
1 package com.example.SAPLM;
2
3 import android.graphics.PorterDuff;
4 import android.graphics.drawable.Drawable;
5 import android.os.Bundle;
6 import androidx.core.content.ContextCompat;
7 import androidx.appcompat.app.AppCompatActivity;
8 import androidx.appcompat.widget.Toolbar;
9 import android.view.MenuItem;
10
11 public class ContactsActivity extends AppCompatActivity {
12     private static Toolbar toolbar;
13
14     @Override
15     protected void onCreate(Bundle savedInstanceState) {
16         super.onCreate(savedInstanceState);
17         setContentView(R.layout.contacts_layout);
18         Application.setActivity(this);
19         Application.setContext(this);
20
21         toolbar = this.findViewById(R.id.toolbar);
22
23         setSupportActionBar(toolbar);
24
25         // add back arrow to toolbar
26         if (getSupportActionBar() != null){
27             getSupportActionBar().setDisplayHomeAsUpEnabled(true);
28             getSupportActionBar().setDisplayShowHomeEnabled(true);
29
30             final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.
abc_ic_ab_back_material);
31             upArrow.setColorFilter(getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP)
32             getSupportActionBar().setHomeAsUpIndicator(upArrow);
33         }
34
35         TextToSpeechHandler.initializeTextToSpeech();
36         try {
37             Thread.sleep(250);
38         }catch (Throwable th) { Application.toastMessage(th.toString());}
39
40         TextToSpeechHandler.addToSpeakingQueue("holá");
41     }
42
43     @Override
44     public boolean onOptionsItemSelected(MenuItem item) {
45         if (item.getItemId() == android.R.id.home) // Press Back Icon
46         {
47             finish();
48         }
49
50         return super.onOptionsItemSelected(item);
51     }
52 }
53 }
```

```
1 package com.example.SAPLM;
2
3 import android.graphics.PorterDuff;
4 import android.view.animation.AccelerateDecelerateInterpolator;
5
6 import java.util.concurrent.Callable;
7 import java.util.concurrent.ExecutorService;
8 import java.util.concurrent.Executors;
9 import java.util.concurrent.Future;
10
11 public class MicrophoneHandler extends MainActivity {
12
13     public enum MicrophoneState { INHIBITED, LISTENING_KEYOWRD, LISTENING_COMMAND}
14
15     static private MicrophoneState currentState = MicrophoneState.LISTENING_KEYOWRD;
16
17     public static void microphoneOnClick(){
18         switch (currentState){
19             case LISTENING_KEYOWRD:
20                 PocketsphinxListener.stopRecognizer();
21                 setState(MicrophoneState.INHIBITED);
22                 break;
23             case LISTENING_COMMAND:
24                 GoogleSpeechRecognizer.forceStopRecognizer();
25                 //PocketsphinxListener.startListening();
26                 setState(MicrophoneState.LISTENING_KEYOWRD);
27                 break;
28             case INHIBITED:
29                 setState(MicrophoneState.LISTENING_KEYOWRD);
30                 PocketsphinxListener.startListening();
31                 break;
32         }
33     }
34 }
35
36     public static MicrophoneState getState(){
37         return currentState;
38     }
39
40     public static void setState(MicrophoneState desiredState){
41         switch (desiredState){
42             case INHIBITED:
43                 imgBtn_microphoneBackground.clearColorFilter();
44                 imgBtn_microphoneBackground.setColorFilter(Application.getContext().getColor
(R.color.ColorBackMicInhibited), PorterDuff.Mode.SRC_ATOP);
45                 imgBtn_microphone.setImageResource(R.drawable.ic_main_mic_off);
46                 endAnimation();
47                 currentState = MicrophoneState.INHIBITED;
48                 break;
49             case LISTENING_KEYOWRD:
50                 imgBtn_microphoneBackground.clearColorFilter();
51                 imgBtn_microphoneBackground.setColorFilter(Application.getContext().getColor
(R.color.ColorBackMicKeyword), PorterDuff.Mode.SRC_ATOP);
52                 imgBtn_microphone.setImageResource(R.drawable.ic_main_mic_listening);
53                 endAnimation();
54                 currentState = MicrophoneState.LISTENING_KEYOWRD;
55                 break;
56             case LISTENING_COMMAND:
57                 imgBtn_microphoneBackground.clearColorFilter();
58                 imgBtn_microphoneBackground.setColorFilter(Application.getContext().getColor
(R.color.ColorBackMicCommand), PorterDuff.Mode.SRC_ATOP);
59                 imgBtn_microphone.setImageResource(R.drawable.ic_main_mic_listening);
60                 startAnimation();
61         }
62     }
63 }
```

```

61             currentState = MicrophoneState.LISTENING_COMMAND;
62         }
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)
77                         .setInterpolator(new AccelerateDecelerateInterpolator());
78                     imgBtn_microphone.animate().scaleX(0.90f).setDuration(350).
79                         setInterpolator(new AccelerateDecelerateInterpolator());
80                     imgBtn_microphoneBackground.animate().scaleY(0.90f).setDuration(350)
81                         .setInterpolator(new AccelerateDecelerateInterpolator());
82                     imgBtn_microphone.animate().scaleY(0.90f).setDuration(350).
83                         setInterpolator(new AccelerateDecelerateInterpolator()).withEndAction(new Runnable() {
84                             @Override
85                             public void run() {
86                                 animationHasFinished = true;
87                             }
88                         });
89                     while(animationHasFinished==false);
90                     animationHasFinished= false;
91                     imgBtn_microphoneBackground.animate().scaleX(1.00f).setDuration(350)
92                         .setInterpolator(new AccelerateDecelerateInterpolator());
93                     imgBtn_microphone.animate().scaleX(1.00f).setDuration(350).
94                         setInterpolator(new AccelerateDecelerateInterpolator());
95                     imgBtn_microphoneBackground.animate().scaleY(1.00f).setDuration(350)
96                         .setInterpolator(new AccelerateDecelerateInterpolator());
97                     imgBtn_microphone.animate().scaleY(1.00f).setDuration(350).
98                         setInterpolator(new AccelerateDecelerateInterpolator()).withEndAction(new Runnable() {
99                             @Override
100                            public void run() {
101                                animationHasFinished = true;
102                            }
103                        });
104                    });
105                });
106            }
107
108        public static void endAnimation(){
109            terminateAnimation = true;
110        }
111    }
112 }
113

```

```
1 package com.example.SAPLM;
2
3 import android.speech.tts.TextToSpeech;
4 import android.speech.tts.Voice;
5
6 import java.util.Locale;
7 import java.util.Set;
8
9 public class TextToSpeechHandler {
10
11     private static TextToSpeech textToSpeech;
12     public static boolean isInitialized = false;
13
14     public static void initializeTextToSpeech() {
15         if (isInitialized == false) {
16             textToSpeech = new TextToSpeech(Application.getContext(), new TextToSpeech.
17                 OnInitListener() {
18                     @Override
19                     public void onInit(int status) {
20                         try {
21                             if (status != TextToSpeech.ERROR) {
22                                 textToSpeech.setLanguage(new Locale("spa"));
23                                 //textToSpeech.setLanguage(Locale.UK);
24                                 //textToSpeech.setOnUtteranceProgressListener(utteranceListener
25                             );
26
27                             textToSpeech.setPitch(1.0f);
28                             textToSpeech.setSpeechRate(1.0f);
29                             textToSpeech.getVoices();
30                             isInitialized = true;
31                         }
32                     } catch (Throwable th) {
33                         Application.toastMessage(th.toString());
34                     }
35                 });
36         }
37
38         public static void addToSpeakingQueue(String text) {
39             try {
40                 if (isInitialized) {
41                     textToSpeech.speak(text, TextToSpeech.QUEUE_ADD, null, "232");
42                 }
43             } catch (Throwable th) {
44                 th.printStackTrace();
45                 Application.toastMessage(th.toString());
46             }
47
48         public static void shutdownTextSpeech() {
49             if (textToSpeech != null) {
50
51                 textToSpeech.stop();
52                 textToSpeech.shutdown();
53             }
54         }
55
56         public static Set<Voice> getVoicesList() {
57             if (isInitialized) {
58                 return textToSpeech.getVoices();
59             } else {
60                 return null;
61             }
62         }
63     }
64 }
```

```
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
```

```

1 package com.example.SAPLM;
2
3 import java.io.File;
4
5 import edu.cmu.pocketsphinx.Assets;
6 import edu.cmu.pocketsphinx.Hypothesis;
7 import edu.cmu.pocketsphinx.RecognitionListener;
8 import edu.cmu.pocketsphinx.SpeechRecognizer;
9 import edu.cmu.pocketsphinx.SpeechRecognizerSetup;
10
11 public final class PocketsphinxListener implements RecognitionListener {
12     private static SpeechRecognizer recognizer;
13     private static String KEYPHRASE = "asistente";
14     private static final String KWS_SEARCH = "wakeup";
15
16
17     @Override
18     public void onPartialResult(Hypothesis hypothesis) {
19         if (hypothesis == null)
20             return;
21
22         String text = hypothesis.getHypstr();
23
24         if (text.contains(KEYPHRASE)) {
25
26             Application.toastMessage(KEYPHRASE);
27             recognizer.stop();
28             //recognizer.cancel();
29             //recognizer.shutdown();
30             GoogleSpeechRecognizer.startSpeechListening();
31             //recognizer.startListening(KWS_SEARCH);
32         }
33
34     }
35
36
37     @Override
38     public void onResult(Hypothesis hypothesis) {
39         if (hypothesis != null) {
40             String text = hypothesis.getHypstr();
41
42         }
43     }
44
45     @Override
46     public void onBeginningOfSpeech() {
47     }
48
49     @Override
50     public void onEndOfSpeech() {
51
52     }
53
54     public static void setupRecognizer(){
55         try {
56             Assets assets = new Assets(Application.getContext());
57             File assetDir = assets.syncAssets();
58             recognizer = SpeechRecognizerSetup.defaultSetup()
59                 .setAcousticModel(new File(assetDir, "es-es-ptm"))
60                 .setDictionary(new File(assetDir, "cmudict-es-es.dict"))
61                 .setKeywordThreshold(Float.MIN_VALUE
62                 )
63                 .setRawLogDir(assetDir) // To disable logging of raw audio comment out

```

Archivo - PocketsphinxListener.java

```
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
```

```
1 package com.example.SAPLM;
2
3 import android.content.Intent;
4 import android.os.Bundle;
5 import android.speech.RecognitionListener;
6 import android.speech.RecognizerIntent;
7 import android.speech.SpeechRecognizer;
8 import android.view.View;
9
10 import com.example.SAPLM.bluetoothActivities.BluetoothConnection;
11 import com.example.SAPLM.bluetoothActivities.UnifiedTransmissionProtocol;
12
13 import org.apache.commons.lang3.StringUtils;
14
15 import java.text.Normalizer;
16 import java.text.SimpleDateFormat;
17 import java.util.ArrayList;
18 import java.util.Arrays;
19 import java.util.Calendar;
20 import java.util.Date;
21 import java.util.Locale;
22
23 public class GoogleSpeechRecognizer extends MainActivity implements RecognitionListener {
24
25     private static SpeechRecognizer speechRecognizer = null;
26
27     private boolean forceCancel = false;
28
29     public static void startSpeechListening() {
30
31
32         String language = "es-ES";
33         Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
34         intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,language);
35         intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE, language);
36         intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_PREFERENCE, language);
37         intent.putExtra(RecognizerIntent.EXTRA_ONLY_RETURN_LANGUAGE_PREFERENCE, language);
38         intent.putExtra(RecognizerIntent.EXTRA_CALLING_PACKAGE, Application.getActivity().
39             getPackageName());
40         intent.putExtra(RecognizerIntent.EXTRA_PARTIAL_RESULTS,true);
41         if (Application.isSystemOnline()==false) { intent.putExtra(RecognizerIntent.
42             EXTRA_PREFER_OFFLINE,true); }
43         speechRecognizer.startListening(intent);
44
45         if (Application.isSystemOnline()) {
46             Thread autoStop = new Thread() {
47                 @Override
48                 public void run() {
49                     super.run();
50                     try {
51                         Thread.sleep(4000);
52                         forceStopRecognizer();
53                     } catch (Throwable th) {
54                         Application.toastMessage(th.toString());
55                     }
56                 };
57             autoStop.start();
58         }
59     }
60 }
```

```

62     public static void setupRecognizer(){
63         speechRecognizer = SpeechRecognizer.createSpeechRecognizer(Application.getContext()
64     );
65     speechRecognizer.setRecognitionListener(new GoogleSpeechRecognizer());
66 }
67
68     public static void shutdownRecognizer(){
69         speechRecognizer.stopListening();
70         speechRecognizer.cancel();
71         speechRecognizer.destroy();
72         speechRecognizer = null;
73         System.gc();
74 }
75
76     public static void forceStopRecognizer(){
77         Application.getActivity().runOnUiThread(
78             new Runnable() {
79                 @Override
80                 public void run() {
81                     speechRecognizer.stopListening();
82                     MicrophoneHandler.setState(MicrophoneHandler.MicrophoneState.
83 LISTENING_KEYWORD);
84                 }
85             }
86         );
87 }
88
89     public static void analyseListenerResults(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")) {

```

Archivo - GoogleSpeechRecognizer.java

```
114             UnifiedTransmissionProtocol.modulesList.get(2).getDevicesList().get(1) .  
115             setData(Boolean.TRUE);  
116             UnifiedTransmissionProtocol.Module.syncAllDevicesValueToSystem();  
117             break outerloop;  
118         }  
119         if (Arrays.asList(hypothesisSplit).contains("apagar") && Arrays.asList(  
120             hypothesisSplit).contains("velador")) {  
121             UnifiedTransmissionProtocol.modulesList.get(2).getDevicesList().get(0) .  
122             setData(Boolean.FALSE);  
123             UnifiedTransmissionProtocol.Module.syncAllDevicesValueToSystem();  
124             break outerloop;  
125         }  
126         if (Arrays.asList(hypothesisSplit).contains("apagar") && Arrays.asList(  
127             hypothesisSplit).contains("estufa")) {  
128             UnifiedTransmissionProtocol.modulesList.get(2).getDevicesList().get(3) .  
129             setData(Boolean.FALSE);  
130             UnifiedTransmissionProtocol.Module.syncAllDevicesValueToSystem();  
131             break outerloop;  
132         }  
133         if (Arrays.asList(hypothesisSplit).contains("apagar") && Arrays.asList(  
134             hypothesisSplit).contains("ventilador")) {  
135             UnifiedTransmissionProtocol.modulesList.get(2).getDevicesList().get(2) .  
136             setData(Boolean.FALSE);  
137             UnifiedTransmissionProtocol.Module.syncAllDevicesValueToSystem();  
138             break outerloop;  
139         }  
140         if (Arrays.asList(hypothesisSplit).contains("apagar") && Arrays.asList(  
141             hypothesisSplit).contains("luz")) {  
142             UnifiedTransmissionProtocol.modulesList.get(2).getDevicesList().get(1) .  
143             setData(Boolean.FALSE);  
144             UnifiedTransmissionProtocol.Module.syncAllDevicesValueToSystem();  
145             break outerloop;  
146         }  
147         if (Arrays.asList(hypothesisSplit).contains("encender") && Arrays.asList(  
148             hypothesisSplit).contains("todo")) {  
149             for(UnifiedTransmissionProtocol.Module.Device dev :  
150                 UnifiedTransmissionProtocol.modulesList.get(2).getDevicesList()) {  
151                 dev.setData(Boolean.TRUE);  
152             }  
153             UnifiedTransmissionProtocol.Module.syncAllDevicesValueToSystem();  
154             break outerloop;  
155         }  
156         if (Arrays.asList(hypothesisSplit).contains("apagar") && Arrays.asList(  
157             hypothesisSplit).contains("todo")) {  
158             for(UnifiedTransmissionProtocol.Module.Device dev :  
159                 UnifiedTransmissionProtocol.modulesList.get(2).getDevicesList()) {  
160                 dev.setData(Boolean.FALSE);  
161             }  
162             UnifiedTransmissionProtocol.Module.syncAllDevicesValueToSystem();  
163             break outerloop;  
164         }  
165         if (Arrays.asList(hypothesisSplit).contains("llamar") && Arrays.asList(  
166             hypothesisSplit).contains("enfermera")) {  
167             UnifiedTransmissionProtocol.modulesList.get(3).getDevicesList().get(2) .  
168             setData(Boolean.TRUE);  
169             UnifiedTransmissionProtocol.Module.syncAllDevicesValueToSystem();  
170         }
```

```

162             break outerloop;
163         }
164
165         if (Arrays.asList(hypothesisSplit).contains("llamar") && Arrays.asList(
166             hypothesisSplit).contains("enfermeria")) {
167             UnifiedTransmissionProtocol.modulesList.get(3).getDevicesList().get(2).
168             setData(Boolean.TRUE);
169             UnifiedTransmissionProtocol.Module.syncAllDevicesValueToSystem();
170             break outerloop;
171         }
172
173         if (Arrays.asList(hypothesisSplit).contains("subir") && Arrays.asList(
174             hypothesisSplit).contains("toda") && Arrays.asList(hypothesisSplit).contains("cortina")) {
175             UnifiedTransmissionProtocol.modulesList.get(3).getDevicesList().get(1).
176             setData((byte)7);
177             UnifiedTransmissionProtocol.Module.syncAllDevicesValueToSystem();
178             break outerloop;
179         }
180
181         if (Arrays.asList(hypothesisSplit).contains("bajar") && Arrays.asList(
182             hypothesisSplit).contains("toda") && Arrays.asList(hypothesisSplit).contains("cortina")) {
183             UnifiedTransmissionProtocol.modulesList.get(3).getDevicesList().get(1).
184             setData((byte)0);
185             UnifiedTransmissionProtocol.Module.syncAllDevicesValueToSystem();
186             break outerloop;
187         }
188
189         if (Arrays.asList(hypothesisSplit).contains("bajar") && Arrays.asList(
190             hypothesisSplit).contains("toda") && Arrays.asList(hypothesisSplit).contains("camilla")) {
191             UnifiedTransmissionProtocol.modulesList.get(3).getDevicesList().get(0).
192             setData((byte)3);
193             UnifiedTransmissionProtocol.Module.syncAllDevicesValueToSystem();
194             break outerloop;
195         }
196
197         if (Arrays.asList(hypothesisSplit).contains("encender") && Arrays.asList(
198             hypothesisSplit).contains("rele")) {
199             for (String word : hypothesisSplit) {
200                 if (StringUtils.isNumeric(word) == true) {
201                     int number = Integer.parseInt(word);
202                     if (number<4) {
203                         UnifiedTransmissionProtocol.modulesList.get(2).getDevicesList().
204                         get(number).setData(Boolean.TRUE);
205                         UnifiedTransmissionProtocol.Module.syncAllDevicesValueToSystem(
206                         );
207                     }
208                 }
209             if (Arrays.asList(hypothesisSplit).contains("apagar") && Arrays.asList(
210                 hypothesisSplit).contains("rele")) {
211                 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()
215                         .get(number).setData(Boolean.FALSE);
216                     UnifiedTransmissionProtocol.Module.syncAllDevicesValueToSystem(
217                         );
218                 }
219             }
220         }
221     }
222     if (Arrays.asList(hypothesisSplit).contains("decir") && Arrays.asList(
223         hypothesisSplit).contains("hora")){
224         Date currentTime = Calendar.getInstance().getTime();
225         SimpleDateFormat df = new SimpleDateFormat("d-MMM-yyyy", new Locale("spa"))
226         ;
227         String todayAsString = df.format(currentTime);
228         SimpleDateFormat onlyTimeFormar = new SimpleDateFormat("HH:mm:ss", new
229             Locale("spa")));
230         String onlyTimeString = onlyTimeFormar.format(currentTime);
231         //TextToSpeechHandler.initializeTextToSpeech();
232         TextToSpeechHandler.addToSpeakingQueue("Hoy es " + todayAsString + " y son
233         las" + onlyTimeString);
234         Application.toastMessage("Decir hora");
235     }
236 }
237 @Override
238 public void onReadyForSpeech(Bundle params) {
239     MicrophoneHandler.setState(MicrophoneHandler.MicrophoneState.LISTENING_COMMAND);
240 }
241 @Override
242 public void onBeginningOfSpeech() {
243 }
244 @Override
245 public void onRmsChanged(float rmsdB) {
246 }
247 @Override
248 public void onBufferReceived(byte[] buffer) {
249 }
250 @Override
251 public void onEndOfSpeech() {
252     //Application.toastMessage("End of speech");
253 }
254 @Override
255 public void onError(int error) {
256     //Application.toastMessage("On error" + error);
257     if (forceCancel==false) {
258
259
260
261
262
263
264
265
266
267

```

Archivo - GoogleSpeechRecognizer.java

```
268         PocketsphinxListener.startListening();
269         MicrophoneHandler.setState(MicrophoneHandler.MicrophoneState.LISTENING_KEYWORD)
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     ;
279     analyseListenerResults(resultsArrayList, confidenceScore);
280
281     StringBuilder builder = new StringBuilder();
282     for (String line : resultsArrayList) {
283         builder.append(line + "\n");
284     }
285
286     MainActivity.textView_speechRecognitionResults.setText(builder.toString());
287     MainActivity.textView_speechRecognitionResults.setVisibility(View.VISIBLE);
288
289
290     MicrophoneHandler.setState(MicrophoneHandler.MicrophoneState.LISTENING_KEYWORD);
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

```
1 package com.example.SAPLM.commandActivity;
2
3 import com.example.SAPLM.R;
4 import com.example.SAPLM.bluetoothActivities.UnifiedTransmissionProtocol;
5
6 import java.util.List;
7
8 public class CommandItem {
9     public enum ItemType { CHILD, PARENT}
10    public enum BoardGroupState {FOLDED, UNFOLDED}
11
12    private String itemName;
13    public ItemType itemType = ItemType.CHILD;
14    private int deviceStateIconID;
15    private int deviceStateInteger = 0;
16    private int moduleIndex = 0;
17    private int deviceIndex = 0;
18    private List<String> childInternalID;
19    private BoardGroupState groupState = BoardGroupState.UNFOLDED;
20
21
22
23    public CommandItem( int moduleIndex, int deviceIndex ){
24        this.itemName = UnifiedTransmissionProtocol.modulesList.get(moduleIndex) .
25            getDevicesList().get(deviceIndex).getName();
26        this.itemType = ItemType.CHILD;
27        this.moduleIndex = moduleIndex;
28        this.deviceIndex = deviceIndex;
29        this.deviceStateIconID = UnifiedTransmissionProtocol.modulesList.get(moduleIndex) .
30            getDevicesList().get(deviceIndex).getIconResourceId();
31    }
32
33    public CommandItem(int boardInternalId){
34        this.itemName = UnifiedTransmissionProtocol.modulesList.get(boardInternalId) .
35            getModuleName();
36        this.itemType = ItemType.PARENT;
37        this.deviceStateIconID = UnifiedTransmissionProtocol.modulesList.get(boardInternalId)
38            .getIconResourceId();
39    }
40
41
42    //Getters
43
44    public BoardGroupState getGroupState (){
45        return groupState;
46    }
47
48    public int getDeviceStateIconID() {
49        return deviceStateIconID;
50    }
51
52    public int getDeviceStateInteger() {
53        return deviceStateInteger;
54    }
55
56    public int getDeviceIndex() {
57        return deviceIndex;
58    }
```

```
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().
79             get(deviceIndex).getDataAsString();
80     }
81
82     public int getProgressBarValue() {
83         return UnifiedTransmissionProtocol.modulesList.get(moduleIndex).getDevicesList().
84             get(deviceIndex).getProgressBarValue();
85     }
86
87     // Setters
88     public void setChildInternalID(List<String> childInternalID) {
89         this.childInternalID = childInternalID;
90     }
91
92     public void setDeviceStateIconID(int deviceStateIconID) {
93         this.deviceStateIconID = deviceStateIconID;
94     }
95
96     public void setDeviceStateInteger(int deviceStateInteger) {
97         this.deviceStateInteger = deviceStateInteger;
98     }
99
100    public void setModuleIndex(int moduleIndex) {
101        this.moduleIndex = moduleIndex;
102    }
103
104    public void setItemName(String itemName) {
105        this.itemName = itemName;
106    }
107
108    public void setItemType(ItemType itemType) {
109        this.itemType = itemType;
110    }
111
112    public void setDeviceIndex(int deviceIndex) {
113        this.deviceIndex = deviceIndex;
114    }
115
116    public void setGroupState (BoardGroupState groupState) {
117        this.groupState = groupState;
118    }
119 }
```

Archivo - CommandsActivity.java

```
1 package com.example.SAPLM.commandActivity;
2
3 import android.app.Activity;
4 import android.content.Context;
5 import android.content.Intent;
6 import android.graphics.PorterDuff;
7 import android.graphics.drawable.Drawable;
8 import android.os.Bundle;
9 import androidx.core.content.ContextCompat;
10 import androidx.appcompat.app.AppCompatActivity;
11 import androidx.appcompat.widget.Toolbar;
12 import androidx.recyclerview.widget.DefaultItemAnimator;
13 import androidx.recyclerview.widget.DividerItemDecoration;
14 import androidx.recyclerview.widget.LinearLayoutManager;
15 import androidx.recyclerview.widget.RecyclerView;
16
17 import android.os.Looper;
18 import android.view.MenuItem;
19 import android.view.View;
20
21 import com.example.SAPLM.Application;
22 import com.example.SAPLM.R;
23 import com.example.SAPLM.bluetoothActivities.BluetoothConnection;
24 import com.example.SAPLM.settingsActivities.AvailableSetting;
25 import com.example.SAPLM.settingsActivities.RecyclerTouchListener;
26 import com.example.SAPLM.settingsActivities.SettingsLayoutAdapter;
27
28 import java.util.ArrayList;
29 import java.util.Arrays;
30 import java.util.List;
31
32 public class CommandsActivity extends AppCompatActivity {
33     private static Toolbar toolbar;
34
35     public static List<CommandItem> commandList = new ArrayList<>();
36     private RecyclerView recyclerView;
37     public static CommandsListAdapter mAdapter;
38
39
40     @Override
41     protected void onCreate(Bundle savedInstanceState) {
42         super.onCreate(savedInstanceState);
43         setContentView(R.layout.commands_layout);
44         Application.setActivity(this);
45         Application.setContext(this);
46
47         toolbar = this.findViewById(R.id.toolbar);
48
49         setSupportActionBar(toolbar);
50
51         // add back arrow to toolbar
52         if (getSupportActionBar() != null){
53             getSupportActionBar().setDisplayHomeAsUpEnabled(true);
54             getSupportActionBar().setDisplayShowHomeEnabled(true);
55
56             final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.
57                 abc_ic_ab_back_material);
58             upArrow.setColorFilter(getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP)
59             ;
60             getSupportActionBar().setHomeAsUpIndicator(upArrow);
61         }
62
63         recyclerView = (RecyclerView) findViewById(R.id.commandsRecyclerView);
```

```
62         mAdapter = new CommandsListAdapter(commandList);
63
64         recyclerView.setHasFixedSize(true);
65
66         // vertical RecyclerView
67         // keep movie_list_row.xml width to `match_parent`
68         RecyclerView.LayoutManager mLayoutManager = new LinearLayoutManager(
69             getApplicationContext());
70
71         // horizontal RecyclerView
72         // keep movie_list_row.xml width to `wrap_content`
73         // RecyclerView.LayoutManager mLayoutManager = new LinearLayoutManager(
74             getApplicationContext(), LinearLayoutManager.HORIZONTAL, false);
75
76         recyclerView.setLayoutManager(mLayoutManager);
77
78         // adding inbuilt divider line
79         recyclerView.addItemDecoration(new DividerItemDecoration(this, LinearLayoutManager.VERTICAL));
80
81         // adding custom divider line with padding 16dp
82         //recyclerView.addItemDecoration(new MyDividerItemDecoration(this,
83             LinearLayoutManager.HORIZONTAL, 16));
84         recyclerView.setItemAnimator(new DefaultItemAnimator());
85
86         recyclerView.setAdapter(mAdapter);
87
88         // row click listener
89         recyclerView.addOnItemTouchListener(new RecyclerTouchListener(getApplicationContext(),
90             recyclerView, new RecyclerTouchListener.ClickListener() {
91
92             @Override
93             public void onClick(View view, int position) {
94                 CommandItem commandItem = commandList.get(position);
95                 if (commandItem.getItemType() == CommandItem.ItemType.PARENT) {
96
97                     }else{
98                         commandItem.onClickHandler();
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() {
```

Archivo - CommandsActivity.java

```
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(
142     Arrays.asList("LED")));
143
144     commandList.add(new CommandItem(1));
145     commandList.add(new CommandItem(1,0));
146     commandList.add(new CommandItem(1, 1));
147
148     commandList.add(new CommandItem(2));
149     commandList.add(new CommandItem(2,0));
150     commandList.add(new CommandItem(2, 1));
151     commandList.add(new CommandItem(2, 2));
152     commandList.add(new CommandItem(2, 3));
153
154     commandList.add(new CommandItem(3));
155     commandList.add(new CommandItem(3,0));
156     commandList.add(new CommandItem(3, 1));
157     commandList.add(new CommandItem(3, 2));
158 }
159 }
```

```
1 package com.example.SAPLM.commandActivity;
2
3 import android.graphics.Color;
4 import android.graphics.PorterDuff;
5 import android.graphics.Typeface;
6 import android.graphics.drawable.ColorDrawable;
7 import android.graphics.drawable.Drawable;
8
9 import androidx.constraintlayout.widget.ConstraintLayout;
10 import androidx.constraintlayout.widget.Guideline;
11 import androidx.recyclerview.widget.RecyclerView;
12 import android.view.LayoutInflater;
13 import android.view.View;
14 import android.view.ViewGroup;
15 import android.widget.ImageView;
16 import android.widget.ProgressBar;
17 import android.widget.TextView;
18
19 import com.example.SAPLM.Application;
20 import com.example.SAPLM.R;
21 import com.example.SAPLM.settingsActivities.AvailableSetting;
22 import com.example.SAPLM.settingsActivities.SettingsLayoutAdapter;
23
24 import java.util.List;
25
26 public class CommandsListAdapter extends RecyclerView.Adapter<CommandsListAdapter.ViewHolder> {
27
28     private List<CommandItem> commandsList;
29
30     public class ViewHolder extends RecyclerView.ViewHolder {
31         public TextView command_title, command_state_text;
32         public ImageView command_icon, arrow_list;
33         public ProgressBar status_progress_bar;
34         public Guideline main_guide;
35
36         public viewHolder(View view) {
37             super(view);
38             command_title = (TextView) view.findViewById(R.id.command_title_textView);
39             status_progress_bar = (ProgressBar) view.findViewById(R.id.
40             command_state_progressBar);
41             command_state_text = (TextView) view.findViewById(R.id.command_state_textView);
42             command_icon = (ImageView) view.findViewById(R.id.command_icon);
43             arrow_list = (ImageView) view.findViewById(R.id.arrow_list_imageButton);
44             main_guide = (Guideline) view.findViewById(R.id.guideline_main);
45         }
46     }
47
48     public CommandsListAdapter(List<CommandItem> cmdList) {
49         this.commandsList = cmdList;
50     }
51
52     @Override
53     public viewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
54         View itemView = LayoutInflater.from(parent.getContext())
55             .inflate(R.layout.commands_layout_child_item_style, parent, false);
56
57         return new viewHolder(itemView);
58     }
59
60     @Override
61     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
100    cmdIcon.setColorFilter(Application.getContext().getColor(R.color.colorPrimary),
    PorterDuff.Mode.SRC_ATOP);
101    holder.command_icon.setImageDrawable(cmdIcon);
102
103    @Override
104    public int getItemCount() {
105        return commandsList.size();
106    }
107 }

```

Archivo - settings\_layout.java

```
1 package com.example.SAPLM.settingsActivities;
2
3 import android.app.Activity;
4 import android.content.Context;
5 import android.content.Intent;
6 import android.graphics.PorterDuff;
7 import android.graphics.drawable.Drawable;
8 import android.os.Bundle;
9 import androidx.core.content.ContextCompat;
10 import androidx.appcompat.app.AppCompatActivity;
11 import androidx.recyclerview.widget.DefaultItemAnimator;
12 import androidx.recyclerview.widget.DividerItemDecoration;
13 import androidx.recyclerview.widget.LinearLayoutManager;
14 import androidx.recyclerview.widget.RecyclerView;
15 import androidx.appcompat.widget.Toolbar;
16 import android.view.MenuItem;
17 import android.view.View;
18
19 import com.example.SAPLM.Application;
20 import com.example.SAPLM.R;
21
22 import java.util.ArrayList;
23 import java.util.List;
24
25 public class settings_layout extends AppCompatActivity {
26
27     private static Toolbar toolbar;
28
29     public static List<AvailableSetting> settingsList = new ArrayList<>();
30     private RecyclerView recyclerView;
31     public static SettingsLayoutAdapter mAdapter;
32
33
34     @Override
35     protected void onCreate(Bundle savedInstanceState) {
36         super.onCreate(savedInstanceState);
37         setContentView(R.layout.settings_layout);
38         Application.setActivity(this);
39         Application.setContext(this);
40
41         toolbar = findViewById(R.id.toolbar);
42
43         setSupportActionBar(toolbar);
44
45         // add back arrow to toolbar
46         if (getSupportActionBar() != null){
47             getSupportActionBar().setDisplayHomeAsUpEnabled(true);
48             getSupportActionBar().setDisplayShowHomeEnabled(true);
49
50
51             final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.
52             abc_ic_ab_back_material);
53             upArrow.setColorFilter(getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP)
54             ;
55             getSupportActionBar().setHomeAsUpIndicator(upArrow);
56
57         }
58         recyclerView = (RecyclerView) findViewById(R.id.settingsRecyclerView);
59         mAdapter = new SettingsLayoutAdapter(settingsList);
60         recyclerView.setHasFixedSize(true);
61 }
```

Archivo - settings\_layout.java

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

Archivo - settings\_layout.java

```
119         R.drawable.ic_setting_internet_connectivity,
120         InternetConnectivityActivity.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.
132             voice_recognition_settings_title),
133         Application.getContext().getString(R.string.
134             voice_recognition_settings_subtitle),
135         R.drawable.ic_setting_voice_recognition,
136         VoiceRecognitionSettingsActivity.class
137     ));
138
139     settingsList.add(new AvailableSetting(
140         Application.getContext().getString(R.string.commands_settings_title),
141         Application.getContext().getString(R.string.commands_settings_subtitle),
142         R.drawable.ic_setting_commands_3,
143         CommandsSettingsActivity.class
144     ));
145
146     settingsList.add(new AvailableSetting(
147         Application.getContext().getString(R.string.contacts_communication_title),
148         Application.getContext().getString(R.string.contacts_communication_subtitle
149     ),
150         R.drawable.ic_setting_contacts_and_communication,
151         ContactsCommunicationActivity.class
152     ));
153
154     settingsList.add(new AvailableSetting(
155         Application.getContext().getString(R.string.appearance_miscellaneous_title)
156
157         Application.getContext().getString(R.string.
158             appearance_miscellaneous_subtitle),
159         R.drawable.ic_setting_appearance_and_miscellaneous,
160         AppearanceMiscellaneousActivity.class
161     ));
162
163     settingsList.add(new AvailableSetting(
164         Application.getContext().getString(R.string.
165             developer_console_activity_title),
166         Application.getContext().getString(R.string.developer_console_subtitle),
167         R.drawable.ic_setting_developer_console,
168         DeveloperConsoleActivity.class
169     ));
170
171     settingsList.add(new AvailableSetting(
172         Application.getContext().getString(R.string.information_help_title),
173         Application.getContext().getString(R.string.information_help_subtitle),
174         R.drawable.ic_setting_information_and_help,
175         InformationHelpActivity.class
176     ));
177
178     // notify adapter about data set changes
179     // so that it will render the list with new data
180     // mAdapter.notifyDataSetChanged();
181 }
```

Archivo - settings\_layout.java

```
176  
177  
178 }  
179
```

Archivo - TerminalManager.java

```
1 package com.example.SAPLM.settingsActivities;
2
3 import android.graphics.Typeface;
4 import android.text.SpannableStringBuilder;
5 import android.text.Spanned;
6 import android.text.style.StyleSpan;
7
8 import com.example.SAPLM.bluetoothActivities.UnifiedTransmissionProtocol;
9
10 import java.util.ArrayList;
11 import java.util.Arrays;
12 import java.util.List;
13 import java.util.regex.Matcher;
14 import java.util.regex.Pattern;
15
16
17 public class TerminalManager {
18
19     public static void handleTerminalInput(String newTerminalInput) {
20
21     }
22
23     public static List<String> currentParameters = new ArrayList<>();
24
25     public static List<TerminalCommand> commandsList = new ArrayList<>(Arrays.asList(
26         new TerminalCommand("COMPOSE", new Runnable() {
27             @Override
28             public void run() {
29                 UnifiedTransmissionProtocol.CommandType command =
30                     UnifiedTransmissionProtocol.CommandType.definedCommandList.stream().filter(p-> p.
31                     getCommandTag().equals((currentParameters.get(0)))).findFirst().orElse(null);
32                 UnifiedTransmissionProtocol.Module target = UnifiedTransmissionProtocol.
33                     modulesList.stream().filter(p-> p.getModuleAlias().equals((currentParameters.get(1)))).
34                     findFirst().orElse(null);
35
36                 if (command==null) {
37                     SpannableStringBuilder builder= new SpannableStringBuilder();
38                     builder.append("Tipo de comando \")")
39                         .append(currentParameters.get(0), new StyleSpan(Typeface.
40                         BOLD), Spanned.SPAN_EXCLUSIVE_EXCLUSIVE)
41                             .append("\\" no reconocido");
42                     DeveloperConsoleActivity.sendTextToTerminal(builder);
43                     return;
44                 }
45
46                 if (target==null) {
47                     SpannableStringBuilder builder= new SpannableStringBuilder();
48                     builder.append("Alias de modulo \")")
49                         .append(currentParameters.get(1), new StyleSpan(Typeface.
50                         BOLD), Spanned.SPAN_EXCLUSIVE_EXCLUSIVE)
51                             .append("\\" no encontrado");
52                     DeveloperConsoleActivity.sendTextToTerminal(builder);
53                     return;
54                 }
55
56                 System.out.println( insertSpaces(UnifiedTransmissionProtocol.bytesToHex(
57                     UnifiedTransmissionProtocol.getCommand_buffer() ), " ", 2 ) );
58             }
59         }));
60
61     public static class TerminalCommand{
```

Archivo - TerminalManager.java

```
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

```
1 package com.example.SAPLM.settingsActivities;
2
3 public class AvailableSetting {
4     private String title, subtitle;
5     private int iconResourceID;
6     private Class targetActivity;
7
8     public AvailableSetting() {
9     }
10
11    public AvailableSetting(String title, String subtitle, int iconResourceID, Class
targetActivity) {
12        this.title = title;
13        this.subtitle = subtitle;
14        this.iconResourceID = iconResourceID;
15        this.targetActivity = targetActivity;
16    }
17
18    public String getTitle() {
19        return title;
20    }
21
22    public void setTitle(String name) {
23        this.title = name;
24    }
25
26    public String getSubtitle() {
27        return subtitle;
28    }
29
30    public void setSubtitle(String subtitle) {
31        this.subtitle = subtitle;
32    }
33
34    public int getIconResourceID() {
35        return iconResourceID;
36    }
37
38    public void setIconResourceID(int iconResourceID) {
39        this.iconResourceID = iconResourceID;
40    }
41
42    public Class getTargetActivity() { return targetActivity; }
43
44    public void setTargetActivity(Class targetActivity) { this.targetActivity =
targetActivity; }
45 }
46
```

Archivo - RecyclerTouchListener.java

```
1 package com.example.SAPLM.settingsActivities;
2
3 import android.content.Context;
4 import androidx.recyclerview.widget.RecyclerView;
5 import android.view.GestureDetector;
6 import android.view.MotionEvent;
7 import android.view.View;
8
9 /**
10  * Created by Ravi Tamada on 03/09/16.
11  * updated by Ravi on 14/11/17
12  * www.androidhive.info
13 */
14 public class RecyclerTouchListener implements RecyclerView.OnItemTouchListener {
15
16     private GestureDetector gestureDetector;
17     private ClickListener clickListener;
18
19     public RecyclerTouchListener(Context context, final RecyclerView recyclerView, final
20         ClickListener clickListener) {
21         this.clickListener = clickListener;
22         gestureDetector = new GestureDetector(context, new GestureDetector.
23             SimpleOnGestureListener() {
24                 @Override
25                 public boolean onSingleTapUp(MotionEvent e) {
26                     return true;
27                 }
28
29                 @Override
30                 public void onLongPress(MotionEvent e) {
31                     View child = recyclerView.findChildViewUnder(e.getX(), e.getY());
32                     if (child != null && clickListener != null) {
33                         clickListener.onLongClick(child, recyclerView.getChildAdapterPosition(
34                             child));
35                     }
36                 }
37             });
38     }
39
40     @Override
41     public boolean onInterceptTouchEvent(RecyclerView rv, MotionEvent e) {
42
43         View child = rv.findChildViewUnder(e.getX(), e.getY());
44         if (child != null && clickListener != null && gestureDetector.onTouchEvent(e)) {
45             clickListener.onClick(child, rv.getChildAdapterPosition(child));
46         }
47         return false;
48     }
49
50     @Override
51     public void onTouchEvent(RecyclerView rv, MotionEvent e) {
52
53     }
54
55     @Override
56     public void onRequestDisallowInterceptTouchEvent(boolean disallowIntercept) {
57
58     }
59
60     public interface ClickListener {
61         void onClick(View view, int position);
62         void onLongClick(View view, int position);
63     }
64 }
```

```
61 }
```

Archivo - SettingsLayoutAdapter.java

```
1 package com.example.SAPLM.settingsActivities;
2
3 import android.graphics.PorterDuff;
4 import android.graphics.drawable.Drawable;
5 import androidx.recyclerview.widget.RecyclerView;
6 import android.view.LayoutInflater;
7 import android.view.View;
8 import android.view.ViewGroup;
9 import android.widget.ImageView;
10 import android.widget.TextView;
11
12 import com.example.SAPLM.Application;
13 import com.example.SAPLM.R;
14
15 import java.util.List;
16
17 public class SettingsLayoutAdapter extends RecyclerView.Adapter<SettingsLayoutAdapter.
viewHolder> {
18
19     private List<AvailableSetting> settingsList;
20
21     public class viewHolder extends RecyclerView.ViewHolder {
22         public TextView setting_title, setting_subtitle;
23         public ImageView setting_icon;
24
25         public viewHolder(View view) {
26             super(view);
27             setting_title = (TextView) view.findViewById(R.id.setting_title);
28             setting_subtitle = (TextView) view.findViewById(R.id.setting_subtitle);
29             setting_icon = (ImageView) view.findViewById(R.id.setting_icon);
30         }
31     }
32
33
34     public SettingsLayoutAdapter(List<AvailableSetting> settingsList) {
35         this.settingsList = settingsList;
36     }
37
38     @Override
39     public viewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
40         View itemView = LayoutInflater.from(parent.getContext())
41             .inflate(R.layout.settings_layout_style, parent, false);
42
43         return new viewHolder(itemView);
44     }
45
46     @Override
47     public void onBindViewHolder(viewHolder holder, int position) {
48         AvailableSetting setting = settingsList.get(position);
49         holder.setting_title.setText(setting.getTitle());
50         holder.setting_subtitle.setText(setting.getSubtitle());
51
52         Drawable icon = Application.getContext().getDrawable(setting.getIconResourceID());
53         icon.setColorFilter(Application.getContext().getColor(R.color.colorPrimary),
PorterDuff.Mode.SRC_ATOP);
54         holder.setting_icon.setImageDrawable(icon);
55     }
56
57     @Override
58     public int getItemCount() {
59         return settingsList.size();
60     }
61 }
```

Archivo - InformationHelpActivity.java

```
1 package com.example.SAPLM.settingsActivities;
2
3 import android.graphics.PorterDuff;
4 import android.graphics.drawable.Drawable;
5 import android.os.Bundle;
6 import androidx.core.content.ContextCompat;
7 import androidx.appcompat.app.AppCompatActivity;
8 import androidx.appcompat.widget.Toolbar;
9 import android.view.MenuItem;
10
11 import com.example.SAPLM.Application;
12 import com.example.SAPLM.R;
13
14 public class InformationHelpActivity extends AppCompatActivity {
15     private static Toolbar toolbar;
16
17     @Override
18     protected void onCreate(Bundle savedInstanceState) {
19         super.onCreate(savedInstanceState);
20         setContentView(R.layout.information_help_layout);
21         Application.setActivity(this);
22         Application.setContext(this);
23
24         toolbar = this.findViewById(R.id.toolbar);
25
26         setSupportActionBar(toolbar);
27
28         // add back arrow to toolbar
29         if (getSupportActionBar() != null){
30             getSupportActionBar().setDisplayHomeAsUpEnabled(true);
31             getSupportActionBar().setDisplayShowHomeEnabled(true);
32
33             final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.
34             abc_ic_ab_back_material);
35             upArrow.setColorFilter(getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP)
36             ;
37             getSupportActionBar().setHomeAsUpIndicator(upArrow);
38         }
39     }
40
41     @Override
42     public boolean onOptionsItemSelected(MenuItem item) {
43         if (item.getItemId() == android.R.id.home) // Press Back Icon
44         {
45             finish();
46         }
47
48     }
49 }
```

```
1 package com.example.SAPLM.settingsActivities;
2
3 import android.content.Context;
4 import android.content.res.Resources;
5 import android.content.res.TypedArray;
6 import android.graphics.Canvas;
7 import android.graphics.Rect;
8 import android.graphics.drawable.Drawable;
9 import androidx.recyclerview.widget.LinearLayoutManager;
10 import androidx.recyclerview.widget.RecyclerView;
11 import android.util.TypedValue;
12 import android.view.View;
13
14 /**
15  * Created by Ravi on 30/10/15.
16  * updated by Ravi on 14/11/17
17  */
18 public class MyDividerItemDecoration extends RecyclerView.ItemDecoration {
19
20     private static final int[] ATTRS = new int[]{
21         android.R.attr.listDivider
22     };
23
24     public static final int HORIZONTAL_LIST = LinearLayoutManager.HORIZONTAL;
25
26     public static final int VERTICAL_LIST = LinearLayoutManager.VERTICAL;
27
28     private Drawable mDivider;
29     private int mOrientation;
30     private Context context;
31     private int margin;
32
33     public MyDividerItemDecoration(Context context, int orientation, int margin) {
34         this.context = context;
35         this.margin = margin;
36         final TypedArray a = context.obtainStyledAttributes(ATTRS);
37         mDivider = a.getDrawable(0);
38         a.recycle();
39         setOrientation(orientation);
40     }
41
42     public void setOrientation(int orientation) {
43         if (orientation != HORIZONTAL_LIST && orientation != VERTICAL_LIST) {
44             throw new IllegalArgumentException("invalid orientation");
45         }
46         mOrientation = orientation;
47     }
48
49     @Override
50     public void onDrawOver(Canvas c, RecyclerView parent, RecyclerView.State state) {
51         if (mOrientation == VERTICAL_LIST) {
52             drawVertical(c, parent);
53         } else {
54             drawHorizontal(c, parent);
55         }
56     }
57
58     public void drawVertical(Canvas c, RecyclerView parent) {
59         final int left = parent.getPaddingLeft();
60         final int right = parent.getWidth() - parent.getPaddingRight();
61
62         final int childCount = parent.getChildCount();
63         for (int i = 0; i < childCount; i++) {
```

```
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

```
1 package com.example.SAPLM.settingsActivities;
2
3 import android.graphics.PorterDuff;
4 import android.graphics.drawable.Drawable;
5 import android.os.Bundle;
6 import androidx.core.content.ContextCompat;
7 import androidx.appcompat.app.AppCompatActivity;
8 import androidx.appcompat.widget.Toolbar;
9 import android.view.MenuItem;
10
11 import com.example.SAPLM.Application;
12 import com.example.SAPLM.R;
13
14 public class CommandsSettingsActivity extends AppCompatActivity {
15     private static Toolbar toolbar;
16
17     @Override
18     protected void onCreate(Bundle savedInstanceState) {
19         super.onCreate(savedInstanceState);
20         setContentView(R.layout.commands_settings_layout);
21         Application.setActivity(this);
22         Application.setContext(this);
23
24         toolbar = this.findViewById(R.id.toolbar);
25
26         setSupportActionBar(toolbar);
27
28         // add back arrow to toolbar
29         if (getSupportActionBar() != null){
30             getSupportActionBar().setDisplayHomeAsUpEnabled(true);
31             getSupportActionBar().setDisplayShowHomeEnabled(true);
32
33             final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.
34             abc_ic_ab_back_material);
35             upArrow.setColorFilter(getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP)
36             ;
37             getSupportActionBar().setHomeAsUpIndicator(upArrow);
38         }
39     }
40
41     @Override
42     public boolean onOptionsItemSelected(MenuItem item) {
43         if (item.getItemId() == android.R.id.home) // Press Back Icon
44         {
45             finish();
46         }
47
48     }
49 }
```

Archivo - DeveloperConsoleActivity.java

```
1 package com.example.SAPLM.settingsActivities;
2
3 import android.graphics.Color;
4 import android.graphics.PorterDuff;
5 import android.graphics.Typeface;
6 import android.graphics.drawable.Drawable;
7 import android.os.Bundle;
8 import androidx.core.content.ContextCompat;
9 import androidx.appcompat.app.AppCompatActivity;
10 import androidx.appcompat.widget.Toolbar;
11
12 import android.os.Looper;
13 import android.text.Html;
14 import android.text.Spanned;
15 import android.text.SpannedString;
16 import android.util.TypedValue;
17 import android.view.MenuItem;
18 import android.view.View;
19 import android.widget.EditText;
20 import android.widget.ImageButton;
21 import android.widget.LinearLayout;
22 import android.widget ScrollView;
23 import android.widget.TextView;
24
25 import com.example.SAPLM.Application;
26 import com.example.SAPLM.R;
27
28 public class DeveloperConsoleActivity extends AppCompatActivity {
29     private static Toolbar toolbar;
30     private static ImageButton enterButton;
31     private static ScrollView scrollViewTerminal;
32     private static EditText editTextTerminal;
33     private static LinearLayout terminalLinearLayout;
34     private static TextView mainTextView;
35
36     private static final String mainText_KEY = "MAIN_TEXT_KEY";
37     private static final String ongoingText_KEY = "ONGOING_TEXT_KEY";
38
39     @Override
40     protected void onCreate(Bundle savedInstanceState) {
41         super.onCreate(savedInstanceState);
42         setContentView(R.layout.developer_console_layout);
43         Application.setActivity(this);
44         Application.setContext(this);
45
46         toolbar = this.findViewById(R.id.toolbar);
47         enterButton = this.findViewById(R.id.enterButton);
48         scrollViewTerminal = this.findViewById(R.id.scrollViewTerminal);
49         editTextTerminal = this.findViewById(R.id.editTextTerminal);
50         terminalLinearLayout = this.findViewById(R.id.terminalLinearLayout);
51
52
53
54         setSupportActionBar(toolbar);
55
56
57         // add back arrow to toolbar
58         if (getSupportActionBar() != null){
59             getSupportActionBar().setDisplayHomeAsUpEnabled(true);
60             getSupportActionBar().setDisplayShowHomeEnabled(true);
61
62             final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.
abc_ic_ab_back_material);
```

Archivo - DeveloperConsoleActivity.java

```
63         upArrow.setColorFilter(getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP
64     );
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().
71     getDisplayMetrics().density);
72     mainTextView.setLayoutParams(new LinearLayout.LayoutParams(LinearLayout.
73     LayoutParams.WRAP_CONTENT, LinearLayout.LayoutParams.WRAP_CONTENT));
73     mainTextView.setTypeface(Typeface.MONOSPACE);
74     terminalLinearLayout.addView(mainTextView);
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.
88             getText().toString());
89         }catch (Throwable th){}
90     }
91     super.onPause();
92 }
93
94 @Override
95 protected void onResume() {
96     if (Application.persistentDataSave.containsKey(mainText_KEY)&&mainTextView!=null) {
97         try {
98             mainTextView.setText(Html.fromHtml(Application.persistentDataSave.getString(
99                 mainText_KEY), Html.FROM_HTML_MODE_LEGACY), TextView.BufferType.SPANNABLE);
100            editTextTerminal.setText(Application.persistentDataSave.getString(
101                ongoingText_KEY));
102            scrollViewTerminal.fullScroll(View.FOCUS_DOWN);
103        }catch (Throwable th){}
104    }
105    super.onResume();
106 }
107
108
109
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.
118     toHtml(newText, Html.FROM_HTML_MODE_LEGACY);
119     Application.persistentDataSave.putString(mainText_KEY,newString);
```

Archivo - DeveloperConsoleActivity.java

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

Archivo - DeveloperConsoleActivity.java

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

Archivo - BluetoothSettingsActivity.java

```
1 package com.example.SAPLM.settingsActivities;
2
3 import android.graphics.PorterDuff;
4 import android.graphics.drawable.Drawable;
5 import android.os.Bundle;
6 import androidx.core.content.ContextCompat;
7 import androidx.appcompat.app.AppCompatActivity;
8 import androidx.appcompat.widget.Toolbar;
9 import android.view.MenuItem;
10
11 import com.example.SAPLM.Application;
12 import com.example.SAPLM.R;
13
14 public class BluetoothSettingsActivity extends AppCompatActivity {
15     private static Toolbar toolbar;
16
17     @Override
18     protected void onCreate(Bundle savedInstanceState) {
19         super.onCreate(savedInstanceState);
20         setContentView(R.layout.bluetooth_settings_layout);
21         Application.setActivity(this);
22         Application.setContext(this);
23
24         toolbar = this.findViewById(R.id.toolbar);
25
26         setSupportActionBar(toolbar);
27
28         // add back arrow to toolbar
29         if (getSupportActionBar() != null){
30             getSupportActionBar().setDisplayHomeAsUpEnabled(true);
31             getSupportActionBar().setDisplayShowHomeEnabled(true);
32
33             final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.
34             abc_ic_ab_back_material);
35             upArrow.setColorFilter(getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP)
36             ;
37             getSupportActionBar().setHomeAsUpIndicator(upArrow);
38         }
39     }
40
41     @Override
42     public boolean onOptionsItemSelected(MenuItem item) {
43         if (item.getItemId() == android.R.id.home) // Press Back Icon
44         {
45             finish();
46         }
47     }
48 }
49
```

Archivo - InternetConectivityActivity.java

```
1 package com.example.SAPLM.settingsActivities;
2
3 import android.graphics.PorterDuff;
4 import android.graphics.drawable.Drawable;
5 import android.os.Bundle;
6 import androidx.core.content.ContextCompat;
7 import androidx.appcompat.app.AppCompatActivity;
8 import androidx.appcompat.widget.Toolbar;
9 import android.view.MenuItem;
10
11 import com.example.SAPLM.Application;
12 import com.example.SAPLM.R;
13
14 public class InternetConectivityActivity extends AppCompatActivity {
15     private static Toolbar toolbar;
16
17     @Override
18     protected void onCreate(Bundle savedInstanceState) {
19         super.onCreate(savedInstanceState);
20         setContentView(R.layout.internet_conectivity_layout);
21         Application.setActivity(this);
22         Application.setContext(this);
23
24         toolbar = this.findViewById(R.id.toolbar);
25
26         setSupportActionBar(toolbar);
27
28         // add back arrow to toolbar
29         if (getSupportActionBar() != null){
30             getSupportActionBar().setDisplayHomeAsUpEnabled(true);
31             getSupportActionBar().setDisplayShowHomeEnabled(true);
32
33             final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.
34             abc_ic_ab_back_material);
35             upArrow.setColorFilter(getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP)
36             ;
37             getSupportActionBar().setHomeAsUpIndicator(upArrow);
38         }
39     }
40
41     @Override
42     public boolean onOptionsItemSelected(MenuItem item) {
43         if (item.getItemId() == android.R.id.home) // Press Back Icon
44         {
45             finish();
46         }
47
48     }
49 }
```

Archivo - ContactsCommunicationActivity.java

```
1 package com.example.SAPLM.settingsActivities;
2
3 import android.graphics.PorterDuff;
4 import android.graphics.drawable.Drawable;
5 import android.os.Bundle;
6 import androidx.core.content.ContextCompat;
7 import androidx.appcompat.app.AppCompatActivity;
8 import androidx.appcompat.widget.Toolbar;
9 import android.view.MenuItem;
10
11 import com.example.SAPLM.Application;
12 import com.example.SAPLM.R;
13
14 public class ContactsCommunicationActivity extends AppCompatActivity {
15     private static Toolbar toolbar;
16
17     @Override
18     protected void onCreate(Bundle savedInstanceState) {
19         super.onCreate(savedInstanceState);
20         setContentView(R.layout.contacts_communication_layout);
21         Application.setActivity(this);
22         Application.setContext(this);
23
24         toolbar = this.findViewById(R.id.toolbar);
25
26         setSupportActionBar(toolbar);
27
28         // add back arrow to toolbar
29         if (getSupportActionBar() != null){
30             getSupportActionBar().setDisplayHomeAsUpEnabled(true);
31             getSupportActionBar().setDisplayShowHomeEnabled(true);
32
33             final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.
34             abc_ic_ab_back_material);
35             upArrow.setColorFilter(getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP)
36             ;
37             getSupportActionBar().setHomeAsUpIndicator(upArrow);
38         }
39     }
40
41     @Override
42     public boolean onOptionsItemSelected(MenuItem item) {
43         if (item.getItemId() == android.R.id.home) // Press Back Icon
44         {
45             finish();
46         }
47
48     }
49 }
```

```
1 package com.example.SAPLM.settingsActivities;
2
3 import android.graphics.PorterDuff;
4 import android.graphics.drawable.Drawable;
5 import android.os.Bundle;
6 import androidx.core.content.ContextCompat;
7 import androidx.appcompat.app.AppCompatActivity;
8 import androidx.appcompat.widget.Toolbar;
9 import android.view.MenuItem;
10
11 import com.example.SAPLM.Application;
12 import com.example.SAPLM.R;
13
14 public class AppearanceMiscellaneousActivity extends AppCompatActivity {
15
16     private static Toolbar toolbar;
17
18     @Override
19     protected void onCreate(Bundle savedInstanceState) {
20         super.onCreate(savedInstanceState);
21         setContentView(R.layout.appearance_miscellaneous_layout);
22         Application.setActivity(this);
23         Application.setContext(this);
24
25         toolbar = this.findViewById(R.id.toolbar);
26
27
28         setSupportActionBar(toolbar);
29
30         // add back arrow to toolbar
31         if (getSupportActionBar() != null){
32             getSupportActionBar().setDisplayHomeAsUpEnabled(true);
33             getSupportActionBar().setDisplayShowHomeEnabled(true);
34
35             final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.
36             abc_ic_ab_back_material);
37             upArrow.setColorFilter(getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP)
38             ;
39             getSupportActionBar().setHomeAsUpIndicator(upArrow);
40         }
41     }
42
43     @Override
44     public boolean onOptionsItemSelected(MenuItem item) {
45         if (item.getItemId() == android.R.id.home) // Press Back Icon
46         {
47             finish();
48         }
49
50         return super.onOptionsItemSelected(item);
51     }
52 }
```

Archivo - VoiceRecognitionSettingsActivity.java

```
1 package com.example.SAPLM.settingsActivities;
2
3 import android.content.DialogInterface;
4 import android.graphics.PorterDuff;
5 import android.graphics.drawable.Drawable;
6 import android.os.Bundle;
7 import android.speech.tts.Voice;
8 import androidx.core.content.ContextCompat;
9 import androidx.appcompat.app.AlertDialog;
10 import androidx.appcompat.app.AppCompatActivity;
11 import androidx.appcompat.widget.Toolbar;
12 import android.view.MenuItem;
13 import android.view.View;
14 import android.widget.Button;
15 import android.widget.EditText;
16 import android.widget.ProgressBar;
17
18 import com.example.SAPLM.Application;
19 import com.example.SAPLM.R;
20 import com.example.SAPLM.TextToSpeechHandler;
21 import com.github.ybq.android.spinkit.sprite.Sprite;
22 import com.github.ybq.android.spinkit.style.FadingCircle;
23
24 import java.text.SimpleDateFormat;
25 import java.util.ArrayList;
26 import java.util.Calendar;
27 import java.util.Date;
28 import java.util.Locale;
29
30 public class VoiceRecognitionSettingsActivity extends AppCompatActivity {
31     private static Toolbar toolbar;
32
33     private Button speech_test_btn;
34     private Button set_voice_btn;
35     private EditText phrase_editText;
36     private static ProgressBar progressBar;
37
38     @Override
39     protected void onCreate(Bundle savedInstanceState) {
40         super.onCreate(savedInstanceState);
41         setContentView(R.layout.voice_recognition_settings_layout);
42         Application.setActivity(this);
43         Application.setContext(this);
44
45         toolbar = this.findViewById(R.id.toolbar);
46         speech_test_btn = this.findViewById(R.id.speech_test_btn);
47         set_voice_btn = this.findViewById(R.id.set_voice_btn);
48         phrase_editText = this.findViewById(R.id.phrase_editText);
49
50         setSupportActionBar(toolbar);
51
52         // add back arrow to toolbar
53         if (getSupportActionBar() != null){
54             getSupportActionBar().setDisplayHomeAsUpEnabled(true);
55             getSupportActionBar().setDisplayShowHomeEnabled(true);
56
57             final Drawable upArrow = ContextCompat.getDrawable(this, R.drawable.
58             abc_ic_ab_back_material);
59             upArrow.setColorFilter(getColor(R.color.colorPrimary), PorterDuff.Mode.SRC_ATOP)
59             ;
59             getSupportActionBar().setHomeAsUpIndicator(upArrow);
60         }
61     }
}
```

Archivo - VoiceRecognitionSettingsActivity.java

```
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-MMM-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"
94             + onlyTimeString);
95
96         }else{
97             Application.toastMessage("Not initialized");
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[
107            TextToSpeechHandler.getVoicesList().size()]);
108
109        for(Voice voice : voiceArray){
110            voiceName.add(voice.getName());
111        }
112
113        builder.setTitle("Seleccionar voz");
114        builder.setItems(voiceName.toArray(new String[voiceName.size()]), new
115        DialogInterface.OnClickListener() {
116            @Override
117            public void onClick(DialogInterface dialog, int which) {
118                TextToSpeechHandler.setVoice(voiceArray[which]);
119            }
120        });
121        builder.show();
122    }
```

Archivo - VoiceRecognitionSettingsActivity.java

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

```

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, (
7             byte) 0x12, (byte) 0x15, (byte) 0x38, (byte) 0x3f, (byte) 0x36, (byte) 0x31, (byte) 0x24, (
8                 byte) 0x23, (byte) 0x2a, (byte) 0x2d,
9                 (byte) 0x70, (byte) 0x77, (byte) 0x7e, (byte) 0x79, (byte) 0x6c, (byte) 0x6b, (
10            byte) 0x62, (byte) 0x65, (byte) 0x48, (byte) 0x4f, (byte) 0x46, (byte) 0x41, (byte) 0x54, (
11                byte) 0x53, (byte) 0x5a, (byte) 0x5d,
12                (byte) 0xe0, (byte) 0xe7, (byte) 0xee, (byte) 0xe9, (byte) 0xfc, (byte) 0xfb, (
13                    byte) 0xf2, (byte) 0xf5, (byte) 0xd8, (byte) 0xdf, (byte) 0xd6, (byte) 0xd1, (byte) 0xc4, (
14                        byte) 0xc3, (byte) 0xca, (byte) 0xcd,
15                        (byte) 0x90, (byte) 0x97, (byte) 0x9e, (byte) 0x99, (byte) 0x8c, (byte) 0x8b, (
16                            byte) 0x82, (byte) 0x85, (byte) 0xa8, (byte) 0xaf, (byte) 0xa6, (byte) 0xa1, (byte) 0xb4, (
17                                byte) 0xb3, (byte) 0xba, (byte) 0xbd,
18                                (byte) 0xc7, (byte) 0xc0, (byte) 0xc9, (byte) 0xce, (byte) 0xdb, (byte) 0xdc, (
19                                    byte) 0xd5, (byte) 0xd2, (byte) 0xff, (byte) 0xf8, (byte) 0xf1, (byte) 0xf6, (byte) 0xe3, (
20                                        byte) 0xe4, (byte) 0xed, (byte) 0xea,
21                                        (byte) 0xb7, (byte) 0xb0, (byte) 0xb9, (byte) 0xbe, (byte) 0xab, (byte) 0xac, (
22                                            byte) 0xa5, (byte) 0xa2, (byte) 0x8f, (byte) 0x88, (byte) 0x81, (byte) 0x86, (byte) 0x93, (
23                                                byte) 0x94, (byte) 0x9d, (byte) 0x9a,
24                                                (byte) 0x27, (byte) 0x20, (byte) 0x29, (byte) 0x2e, (byte) 0x3b, (byte) 0x3c, (
25                                                    byte) 0x35, (byte) 0x32, (byte) 0x1f, (byte) 0x18, (byte) 0x11, (byte) 0x16, (byte) 0x03, (
26                                                        byte) 0x04, (byte) 0x0d, (byte) 0xa,
27                                                        (byte) 0x57, (byte) 0x50, (byte) 0x59, (byte) 0x5e, (byte) 0x4b, (byte) 0x4c, (
28                                                            byte) 0x45, (byte) 0x42, (byte) 0x6f, (byte) 0x68, (byte) 0x61, (byte) 0x66, (byte) 0x73, (
29                                                                byte) 0x74, (byte) 0x7d, (byte) 0x7a,
30                                                                (byte) 0x89, (byte) 0x8e, (byte) 0x87, (byte) 0x80, (byte) 0x95, (byte) 0x92, (
31                                                                    byte) 0x9b, (byte) 0x9c, (byte) 0xb1, (byte) 0xb6, (byte) 0xbf, (byte) 0xb8, (byte) 0xad, (
32                                                                        byte) 0xaa, (byte) 0xa3, (byte) 0xa4,
33                                                                        (byte) 0xf9, (byte) 0xfe, (byte) 0xf7, (byte) 0xf0, (byte) 0xe5, (byte) 0xe2, (
34                                                                            byte) 0xeb, (byte) 0xec, (byte) 0xc1, (byte) 0xc6, (byte) 0xcf, (byte) 0xc8, (byte) 0xdd, (
35                                                                                byte) 0xda, (byte) 0xd3, (byte) 0xd4,
36                                                                                (byte) 0x69, (byte) 0x6e, (byte) 0x67, (byte) 0x60, (byte) 0x75, (byte) 0x72, (
37                                                                                    byte) 0x7b, (byte) 0x7c, (byte) 0x51, (byte) 0x56, (byte) 0x5f, (byte) 0x58, (byte) 0x4d, (
38                                                                                        byte) 0x4a, (byte) 0x43, (byte) 0x44,
39                                                                                        (byte) 0x19, (byte) 0x1e, (byte) 0x17, (byte) 0x10, (byte) 0x05, (byte) 0x02, (
40                                                                                            byte) 0x0b, (byte) 0x0c, (byte) 0x21, (byte) 0x26, (byte) 0x2f, (byte) 0x28, (byte) 0x3d, (
41                                                                                                byte) 0x3a, (byte) 0x33, (byte) 0x34,
42                                                                                                (byte) 0x4e, (byte) 0x49, (byte) 0x40, (byte) 0x47, (byte) 0x52, (byte) 0x55, (
43                                                                                      byte) 0x5c, (byte) 0x5b, (byte) 0x76, (byte) 0x71, (byte) 0x78, (byte) 0x7f, (byte) 0x6a, (
44                          byte) 0x6d, (byte) 0x64, (byte) 0x63,
45                          (byte) 0x3e, (byte) 0x39, (byte) 0x30, (byte) 0x37, (byte) 0x22, (byte) 0x25, (
46                              byte) 0x2c, (byte) 0x2b, (byte) 0x06, (byte) 0x01, (byte) 0x08, (byte) 0x0f, (byte) 0x1a, (
47                                  byte) 0x1d, (byte) 0x14, (byte) 0x13,
48                                  (byte) 0xae, (byte) 0xa9, (byte) 0xa0, (byte) 0xa7, (byte) 0xb2, (byte) 0xb5, (
49                                      byte) 0xbc, (byte) 0xbb, (byte) 0x96, (byte) 0x91, (byte) 0x98, (byte) 0x9f, (byte) 0x8a, (
50                                          byte) 0x8d, (byte) 0x84, (byte) 0x83,
51                                          (byte) 0xde, (byte) 0xd9, (byte) 0xd0, (byte) 0xd7, (byte) 0xc2, (byte) 0xc5, (
52                                              byte) 0xcc, (byte) 0xcb, (byte) 0xe6, (byte) 0xe1, (byte) 0xe8, (byte) 0xef, (byte) 0xfa, (
53                                              byte) 0xfd, (byte) 0xf4, (byte) 0xf3
54      };
55
56
57     public static byte crc_CCITT(byte[] data, int data_len)
58     {
59         byte crc = 0;
60         byte[] d = data;
61         int dataIndex = 0;
62         byte tbl_idx = 0;
63
64         while (data_len!=0) {
65
66             byte val = d[dataIndex];
67             dataIndex++;
68
69             if (tbl_idx == 0) {
70                 byte val2 = d[dataIndex];
71                 dataIndex++;
72
73                 if (val2 >= 0x80) {
74                     val = val ^ 0x80;
75                 }
76
77                 if (val2 >= 0x40) {
78                     val = val ^ 0x40;
79                 }
80
81                 if (val2 >= 0x20) {
82                     val = val ^ 0x20;
83                 }
84
85                 if (val2 >= 0x10) {
86                     val = val ^ 0x10;
87                 }
88
89                 if (val2 >= 0x08) {
90                     val = val ^ 0x08;
91                 }
92
93                 if (val2 >= 0x04) {
94                     val = val ^ 0x04;
95                 }
96
97                 if (val2 >= 0x02) {
98                     val = val ^ 0x02;
99                 }
100
101                 if (val2 >= 0x01) {
102                     val = val ^ 0x01;
103                 }
104
105                 if (val2 >= 0x00) {
106                     val = val ^ 0x00;
107                 }
108
109                 if (val2 >= 0x08) {
110                     val = val ^ 0x08;
111                 }
112
113                 if (val2 >= 0x04) {
114                     val = val ^ 0x04;
115                 }
116
117                 if (val2 >= 0x02) {
118                     val = val ^ 0x02;
119                 }
120
121                 if (val2 >= 0x01) {
122                     val = val ^ 0x01;
123                 }
124
125                 if (val2 >= 0x00) {
126                     val = val ^ 0x00;
127                 }
128
129                 if (val2 >= 0x08) {
130                     val = val ^ 0x08;
131                 }
132
133                 if (val2 >= 0x04) {
134                     val = val ^ 0x04;
135                 }
136
137                 if (val2 >= 0x02) {
138                     val = val ^ 0x02;
139                 }
140
141                 if (val2 >= 0x01) {
142                     val = val ^ 0x01;
143                 }
144
145                 if (val2 >= 0x00) {
146                     val = val ^ 0x00;
147                 }
148
149                 if (val2 >= 0x08) {
150                     val = val ^ 0x08;
151                 }
152
153                 if (val2 >= 0x04) {
154                     val = val ^ 0x04;
155                 }
156
157                 if (val2 >= 0x02) {
158                     val = val ^ 0x02;
159                 }
160
161                 if (val2 >= 0x01) {
162                     val = val ^ 0x01;
163                 }
164
165                 if (val2 >= 0x00) {
166                     val = val ^ 0x00;
167                 }
168
169                 if (val2 >= 0x08) {
170                     val = val ^ 0x08;
171                 }
172
173                 if (val2 >= 0x04) {
174                     val = val ^ 0x04;
175                 }
176
177                 if (val2 >= 0x02) {
178                     val = val ^ 0x02;
179                 }
180
181                 if (val2 >= 0x01) {
182                     val = val ^ 0x01;
183                 }
184
185                 if (val2 >= 0x00) {
186                     val = val ^ 0x00;
187                 }
188
189                 if (val2 >= 0x08) {
190                     val = val ^ 0x08;
191                 }
192
193                 if (val2 >= 0x04) {
194                     val = val ^ 0x04;
195                 }
196
197                 if (val2 >= 0x02) {
198                     val = val ^ 0x02;
199                 }
200
201                 if (val2 >= 0x01) {
202                     val = val ^ 0x01;
203                 }
204
205                 if (val2 >= 0x00) {
206                     val = val ^ 0x00;
207                 }
208
209                 if (val2 >= 0x08) {
210                     val = val ^ 0x08;
211                 }
212
213                 if (val2 >= 0x04) {
214                     val = val ^ 0x04;
215                 }
216
217                 if (val2 >= 0x02) {
218                     val = val ^ 0x02;
219                 }
220
221                 if (val2 >= 0x01) {
222                     val = val ^ 0x01;
223                 }
224
225                 if (val2 >= 0x00) {
226                     val = val ^ 0x00;
227                 }
228
229                 if (val2 >= 0x08) {
230                     val = val ^ 0x08;
231                 }
232
233                 if (val2 >= 0x04) {
234                     val = val ^ 0x04;
235                 }
236
237                 if (val2 >= 0x02) {
238                     val = val ^ 0x02;
239                 }
240
241                 if (val2 >= 0x01) {
242                     val = val ^ 0x01;
243                 }
244
245                 if (val2 >= 0x00) {
246                     val = val ^ 0x00;
247                 }
248
249                 if (val2 >= 0x08) {
250                     val = val ^ 0x08;
251                 }
252
253                 if (val2 >= 0x04) {
254                     val = val ^ 0x04;
255                 }
256
257                 if (val2 >= 0x02) {
258                     val = val ^ 0x02;
259                 }
260
261                 if (val2 >= 0x01) {
262                     val = val ^ 0x01;
263                 }
264
265                 if (val2 >= 0x00) {
266                     val = val ^ 0x00;
267                 }
268
269                 if (val2 >= 0x08) {
270                     val = val ^ 0x08;
271                 }
272
273                 if (val2 >= 0x04) {
274                     val = val ^ 0x04;
275                 }
276
277                 if (val2 >= 0x02) {
278                     val = val ^ 0x02;
279                 }
280
281                 if (val2 >= 0x01) {
282                     val = val ^ 0x01;
283                 }
284
285                 if (val2 >= 0x00) {
286                     val = val ^ 0x00;
287                 }
288
289                 if (val2 >= 0x08) {
290                     val = val ^ 0x08;
291                 }
292
293                 if (val2 >= 0x04) {
294                     val = val ^ 0x04;
295                 }
296
297                 if (val2 >= 0x02) {
298                     val = val ^ 0x02;
299                 }
300
301                 if (val2 >= 0x01) {
302                     val = val ^ 0x01;
303                 }
304
305                 if (val2 >= 0x00) {
306                     val = val ^ 0x00;
307                 }
308
309                 if (val2 >= 0x08) {
310                     val = val ^ 0x08;
311                 }
312
313                 if (val2 >= 0x04) {
314                     val = val ^ 0x04;
315                 }
316
317                 if (val2 >= 0x02) {
318                     val = val ^ 0x02;
319                 }
320
321                 if (val2 >= 0x01) {
322                     val = val ^ 0x01;
323                 }
324
325                 if (val2 >= 0x00) {
326                     val = val ^ 0x00;
327                 }
328
329                 if (val2 >= 0x08) {
330                     val = val ^ 0x08;
331                 }
332
333                 if (val2 >= 0x04) {
334                     val = val ^ 0x04;
335                 }
336
337                 if (val2 >= 0x02) {
338                     val = val ^ 0x02;
339                 }
340
341                 if (val2 >= 0x01) {
342                     val = val ^ 0x01;
343                 }
344
345                 if (val2 >= 0x00) {
346                     val = val ^ 0x00;
347                 }
348
349                 if (val2 >= 0x08) {
350                     val = val ^ 0x08;
351                 }
352
353                 if (val2 >= 0x04) {
354                     val = val ^ 0x04;
355                 }
356
357                 if (val2 >= 0x02) {
358                     val = val ^ 0x02;
359                 }
360
361                 if (val2 >= 0x01) {
362                     val = val ^ 0x01;
363                 }
364
365                 if (val2 >= 0x00) {
366                     val = val ^ 0x00;
367                 }
368
369                 if (val2 >= 0x08) {
370                     val = val ^ 0x08;
371                 }
372
373                 if (val2 >= 0x04) {
374                     val = val ^ 0x04;
375                 }
376
377                 if (val2 >= 0x02) {
378                     val = val ^ 0x02;
379                 }
380
381                 if (val2 >= 0x01) {
382                     val = val ^ 0x01;
383                 }
384
385                 if (val2 >= 0x00) {
386                     val = val ^ 0x00;
387                 }
388
389                 if (val2 >= 0x08) {
390                     val = val ^ 0x08;
391                 }
392
393                 if (val2 >= 0x04) {
394                     val = val ^ 0x04;
395                 }
396
397                 if (val2 >= 0x02) {
398                     val = val ^ 0x02;
399                 }
400
401                 if (val2 >= 0x01) {
402                     val = val ^ 0x01;
403                 }
404
405                 if (val2 >= 0x00) {
406                     val = val ^ 0x00;
407                 }
408
409                 if (val2 >= 0x08) {
410                     val = val ^ 0x08;
411                 }
412
413                 if (val2 >= 0x04) {
414                     val = val ^ 0x04;
415                 }
416
417                 if (val2 >= 0x02) {
418                     val = val ^ 0x02;
419                 }
420
421                 if (val2 >= 0x01) {
422                     val = val ^ 0x01;
423                 }
424
425                 if (val2 >= 0x00) {
426                     val = val ^ 0x00;
427                 }
428
429                 if (val2 >= 0x08) {
430                     val = val ^ 0x08;
431                 }
432
433                 if (val2 >= 0x04) {
434                     val = val ^ 0x04;
435                 }
436
437                 if (val2 >= 0x02) {
438                     val = val ^ 0x02;
439                 }
440
441                 if (val2 >= 0x01) {
442                     val = val ^ 0x01;
443                 }
444
445                 if (val2 >= 0x00) {
446                     val = val ^ 0x00;
447                 }
448
449                 if (val2 >= 0x08) {
450                     val = val ^ 0x08;
451                 }
452
453                 if (val2 >= 0x04) {
454                     val = val ^ 0x04;
455                 }
456
457                 if (val2 >= 0x02) {
458                     val = val ^ 0x02;
459                 }
460
461                 if (val2 >= 0x01) {
462                     val = val ^ 0x01;
463                 }
464
465                 if (val2 >= 0x00) {
466                     val = val ^ 0x00;
467                 }
468
469                 if (val2 >= 0x08) {
470                     val = val ^ 0x08;
471                 }
472
473                 if (val2 >= 0x04) {
474                     val = val ^ 0x04;
475                 }
476
477                 if (val2 >= 0x02) {
478                     val = val ^ 0x02;
479                 }
480
481                 if (val2 >= 0x01) {
482                     val = val ^ 0x01;
483                 }
484
485                 if (val2 >= 0x00) {
486                     val = val ^ 0x00;
487                 }
488
489                 if (val2 >= 0x08) {
490                     val = val ^ 0x08;
491                 }
492
493                 if (val2 >= 0x04) {
494                     val = val ^ 0x04;
495                 }
496
497                 if (val2 >= 0x02) {
498                     val = val ^ 0x02;
499                 }
500
501                 if (val2 >= 0x01) {
502                     val = val ^ 0x01;
503                 }
504
505                 if (val2 >= 0x00) {
506                     val = val ^ 0x00;
507                 }
508
509                 if (val2 >= 0x08) {
510                     val = val ^ 0x08;
511                 }
512
513                 if (val2 >= 0x04) {
514                     val = val ^ 0x04;
515                 }
516
517                 if (val2 >= 0x02) {
518                     val = val ^ 0x02;
519                 }
520
521                 if (val2 >= 0x01) {
522                     val = val ^ 0x01;
523                 }
524
525                 if (val2 >= 0x00) {
526                     val = val ^ 0x00;
527                 }
528
529                 if (val2 >= 0x08) {
530                     val = val ^ 0x08;
531                 }
532
533                 if (val2 >= 0x04) {
534                     val = val ^ 0x04;
535                 }
536
537                 if (val2 >= 0x02) {
538                     val = val ^ 0x02;
539                 }
540
541                 if (val2 >= 0x01) {
542                     val = val ^ 0x01;
543                 }
544
545                 if (val2 >= 0x00) {
546                     val = val ^ 0x00;
547                 }
548
549                 if (val2 >= 0x08) {
550                     val = val ^ 0x08;
551                 }
552
553                 if (val2 >= 0x04) {
554                     val = val ^ 0x04;
555                 }
556
557                 if (val2 >= 0x02) {
558                     val = val ^ 0x02;
559                 }
560
561                 if (val2 >= 0x01) {
562                     val = val ^ 0x01;
563                 }
564
565                 if (val2 >= 0x00) {
566                     val = val ^ 0x00;
567                 }
568
569                 if (val2 >= 0x08) {
570                     val = val ^ 0x08;
571                 }
572
573                 if (val2 >= 0x04) {
574                     val = val ^ 0x04;
575                 }
576
577                 if (val2 >= 0x02) {
578                     val = val ^ 0x02;
579                 }
580
581                 if (val2 >= 0x01) {
582                     val = val ^ 0x01;
583                 }
584
585                 if (val2 >= 0x00) {
586                     val = val ^ 0x00;
587                 }
588
589                 if (val2 >= 0x08) {
590                     val = val ^ 0x08;
591                 }
592
593                 if (val2 >= 0x04) {
594                     val = val ^ 0x04;
595                 }
596
597                 if (val2 >= 0x02) {
598                     val = val ^ 0x02;
599                 }
600
601                 if (val2 >= 0x01) {
602                     val = val ^ 0x01;
603                 }
604
605                 if (val2 >= 0x00) {
606                     val = val ^ 0x00;
607                 }
608
609                 if (val2 >= 0x08) {
610                     val = val ^ 0x08;
611                 }
612
613                 if (val2 >= 0x04) {
614                     val = val ^ 0x04;
615                 }
616
617                 if (val2 >= 0x02) {
618                     val = val ^ 0x02;
619                 }
620
621                 if (val2 >= 0x01) {
622                     val = val ^ 0x01;
623                 }
624
625                 if (val2 >= 0x00) {
626                     val = val ^ 0x00;
627                 }
628
629                 if (val2 >= 0x08) {
630                     val = val ^ 0x08;
631                 }
632
633                 if (val2 >= 0x04) {
634                     val = val ^ 0x04;
635                 }
636
637                 if (val2 >= 0x02) {
638                     val = val ^ 0x02;
639                 }
640
641                 if (val2 >= 0x01) {
642                     val = val ^ 0x01;
643                 }
644
645                 if (val2 >= 0x00) {
646                     val = val ^ 0x00;
647                 }
648
649                 if (val2 >= 0x08) {
650                     val = val ^ 0x08;
651                 }
652
653                 if (val2 >= 0x04) {
654                     val = val ^ 0x04;
655                 }
656
657                 if (val2 >= 0x02) {
658                     val = val ^ 0x02;
659                 }
660
661                 if (val2 >= 0x01) {
662                     val = val ^ 0x01;
663                 }
664
665                 if (val2 >= 0x00) {
666                     val = val ^ 0x00;
667                 }
668
669                 if (val2 >= 0x08) {
670                     val = val ^ 0x08;
671                 }
672
673                 if (val2 >= 0x04) {
674                     val = val ^ 0x04;
675                 }
676
677                 if (val2 >= 0x02) {
678                     val = val ^ 0x02;
679                 }
680
681                 if (val2 >= 0x01) {
682                     val = val ^ 0x01;
683                 }
684
685                 if (val2 >= 0x00) {
686                     val = val ^ 0x00;
687                 }
688
689                 if (val2 >= 0x08) {
690                     val = val ^ 0x08;
691                 }
692
693                 if (val2 >= 0x04) {
694                     val = val ^ 0x04;
695                 }
696
697                 if (val2 >= 0x02) {
698                     val = val ^ 0x02;
699                 }
700
701                 if (val2 >= 0x01) {
702                     val = val ^ 0x01;
703                 }
704
705                 if (val2 >= 0x00) {
706                     val = val ^ 0x00;
707                 }
708
709                 if (val2 >= 0x08) {
710                     val = val ^ 0x08;
711                 }
712
713                 if (val2 >= 0x04) {
714                     val = val ^ 0x04;
715                 }
716
717                 if (val2 >= 0x02) {
718                     val = val ^ 0x02;
719                 }
720
721                 if (val2 >= 0x01) {
722                     val = val ^ 0x01;
723                 }
724
725                 if (val2 >= 0x00) {
726                     val = val ^ 0x00;
727                 }
728
729                 if (val2 >= 0x08) {
730                     val = val ^ 0x08;
731                 }
732
733                 if (val2 >= 0x04) {
734                     val = val ^ 0x04;
735                 }
736
737                 if (val2 >= 0x02) {
738                     val = val ^ 0x02;
739                 }
740
741                 if (val2 >= 0x01) {
742                     val = val ^ 0x01;
743                 }
744
745                 if (val2 >= 0x00) {
746                     val = val ^ 0x00;
747                 }
748
749                 if (val2 >= 0x08) {
750                     val = val ^ 0x08;
751                 }
752
753                 if (val2 >= 0x04) {
754                     val = val ^ 0x04;
755                 }
756
757                 if (val2 >= 0x02) {
758                     val = val ^ 0x02;
759                 }
760
761                 if (val2 >= 0x01) {
762                     val = val ^ 0x01;
763                 }
764
765                 if (val2 >= 0x00) {
766                     val = val ^ 0x00;
767                 }
768
769                 if (val2 >= 0x08) {
770                     val = val ^ 0x08;
771                 }
772
773                 if (val2 >= 0x04) {
774                     val = val ^ 0x04;
775                 }
776
777                 if (val2 >= 0x02) {
778                     val = val ^ 0x02;
779                 }
780
781                 if (val2 >= 0x01) {
782                     val = val ^ 0x01;
783                 }
784
785                 if (val2 >= 0x00) {
786                     val = val ^ 0x00;
787                 }
788
789                 if (val2 >= 0x08) {
790                     val = val ^ 0x08;
791                 }
792
793                 if (val2 >= 0x04) {
794                     val = val ^ 0x04;
795                 }
796
797                 if (val2 >= 0x02) {
798                     val = val ^ 0x02;
799                 }
800
801                 if (val2 >= 0x01) {
802                     val = val ^ 0x01;
803                 }
804
805                 if (val2 >= 0x00) {
806                     val = val ^ 0x00;
807                 }
808
809                 if (val2 >= 0x08) {
810                     val = val ^ 0x08;
811                 }
812
813                 if (val2 >= 0x04) {
814                     val = val ^ 0x04;
815                 }
816
817                 if (val2 >= 0x02) {
818                     val = val ^ 0x02;
819                 }
820
821                 if (val2 >= 0x01) {
822                     val = val ^ 0x01;
823                 }
824
825                 if (val2 >= 0x00) {
826                     val = val ^ 0x00;
827                 }
828
829                 if (val2 >= 0x08) {
830                     val = val ^ 0x08;
831                 }
832
833                 if (val2 >= 0x04) {
834                     val = val ^ 0x04;
835                 }
836
837                 if (val2 >= 0x02) {
838                     val = val ^ 0x02;
839                 }
840
841                 if (val2 >= 0x01) {
842                     val = val ^ 0x01;
843                 }
844
845                 if (val2 >= 0x00) {
846                     val = val ^ 0x00;
847                 }
848
849                 if (val2 >= 0x08) {
850                     val = val ^ 0x08;
851                 }
852
853                 if (val2 >= 0x04) {
854                     val = val ^ 0x04;
855                 }
856
857                 if (val2 >= 0x02) {
858                     val = val ^ 0x02;
859                 }
860
861                 if (val2 >= 0x01) {
862                     val = val ^ 0x01;
863                 }
864
865                 if (val2 >= 0x00) {
866                     val = val ^ 0x00;
867                 }
868
869                 if (val2 >= 0x08) {
870                     val = val ^ 0x08;
871                 }
872
873                 if (val2 >= 0x04) {
874                     val = val ^ 0x04;
875                 }
876
877                 if (val2 >= 0x02) {
878                     val = val ^ 0x02;
879                 }
880
881                 if (val2 >= 0x01) {
882                     val = val ^ 0x01;
883                 }
884
885                 if (val2 >= 0x00) {
886                     val = val ^ 0x00;
887                 }
888
889                 if (val2 >= 0x08) {
890                     val = val ^ 0x08;
891                 }
892
893                 if (val2 >= 0x04) {
894                     val = val ^ 0x04;
895                 }
896
897                 if (val2 >= 0x02) {

```

Archivo - CRC8\_CCITT.java

```
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

```
1 package com.example.SAPLM.bluetoothActivities;
2
3 import com.example.SAPLM.R;
4
5 public class AvailableDevice {
6
7     public enum DeviceState { DEFAULT, DISCONNECTED, CONNECTED, LOADING}
8     private String deviceName;
9     private DeviceState currentState;
10    private int deviceStateIconID;
11
12    public AvailableDevice() {
13    }
14
15    public AvailableDevice(String deviceName, DeviceState defaultState) {
16        this.deviceName = deviceName;
17        this.currentState = defaultState;
18    }
19
20    public void setCurrentState(DeviceState currentState) {
21        this.currentState = currentState;
22    }
23
24    public DeviceState getCurrentState() {
25        return currentState;
26    }
27
28    public void setDeviceName(String deviceName) {
29        this.deviceName = deviceName;
30    }
31
32    public String getDeviceName() {
33        return deviceName;
34    }
35
36    public int getDeviceStateIconID() {
37        switch (currentState){
38            case DEFAULT:
39                return 0;
40            case CONNECTED:
41                return R.drawable.ic_done_icon;
42            case LOADING:
43                return R.drawable.loading_animation;
44            case DISCONNECTED:
45                return R.drawable.ic_close_icon;
46        }
47        return 0;
48    }
49
50    public int getTextAppearance(){
51        switch (currentState){
52            case DEFAULT:
53                return R.style.btDeviceListAppearance;
54            case CONNECTED:
55                return R.style.btDeviceListSelectedAppearance;
56            case LOADING:
57                return R.style.btDeviceListAppearance;
58            case DISCONNECTED:
59                return R.style.btDeviceListAppearance;
60        }
61        return R.style.btDeviceListAppearance;
62    }
63 }
```

```
1 package com.example.SAPLM.bluetoothActivities;
2
3 import android.content.Context;
4 import android.content.Intent;
5 import android.graphics.Color;
6 import android.graphics.PorterDuff;
7 import android.graphics.drawable.Drawable;
8 import android.os.Bundle;
9 import android.os.Looper;
10 import androidx.core.content.ContextCompat;
11 import androidx.appcompat.app.AppCompatActivity;
12 import androidx.recyclerview.widget.DefaultItemAnimator;
13 import androidx.recyclerview.widget.DividerItemDecoration;
14 import androidx.recyclerview.widget.LinearLayoutManager;
15 import androidx.recyclerview.widget.RecyclerView;
16 import androidx.appcompat.widget.Toolbar;
17 import android.view.MenuItem;
18 import android.view.View;
19 import android.widget.ImageView;
20 import android.widget.ProgressBar;
21 import android.widget.TextView;
22
23 import com.example.SAPLM.Application;
24 import com.example.SAPLM.R;
25 import com.example.SAPLM.settingsActivities.BluetoothSettingsActivity;
26 import com.example.SAPLM.settingsActivities.RecyclerTouchListener;
27 import com.github.ybq.android.spinkit.sprite.Sprite;
28 import com.github.ybq.android.spinkit.style.Circle;
29
30 import java.util.ArrayList;
31 import java.util.List;
32
33 public class BluetoothActivity extends AppCompatActivity {
34     private static Toolbar toolbar;
35
36
37     public static List<AvailableDevice> deviceList = new ArrayList<>();
38     private RecyclerView recyclerView;
39     public static DeviceListAdapter mAdapter;
40
41     public static ImageView imgView_bluetoothStatusIcon;
42     public static TextView textView_bluetoothState;
43     public static ProgressBar loading_progressBar;
44
45     public static Boolean guiInitialized = false;
46
47
48     @Override
49     protected void onCreate(Bundle savedInstanceState) {
50         super.onCreate(savedInstanceState);
51         setContentView(R.layout.bluetooth_layout);
52         Application.setActivity(this);
53         Application.setContext(this);
54
55         toolbar = this.findViewById(R.id.toolbar);
56         textView_bluetoothState = this.findViewById(R.id.textView_btStatus);
57         imgView_bluetoothStatusIcon = this.findViewById(R.id.imgView_btStatusIcon);
58         loading_progressBar = this.findViewById(R.id.loading_progressBar);
59
60         Sprite fadingCircle = new Circle();
61         fadingCircle.setColor(Color.WHITE);
62         loading_progressBar.setIndeterminateDrawable(fadingCircle);
63 }
```

## Archivo - BluetoothActivity.java

```
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,
151     BLUETOOTH_DISABLED, WAITING, BT_TURNING_OFF, BT_TURNING_ON, UNDEFINED}
152     public static bluetoothConnectionState currentBluetoothState = bluetoothConnectionState
153     .UNDEFINED;
154
155     public static void setBluetoothStateGui(bluetoothConnectionState state) {
156         if (guiInitialized) {
157             currentBluetoothState = state;
158
159             if (Looper.myLooper() == Looper.getMainLooper()) {
160                 switch (currentBluetoothState) {
161                     case SYSTEM_CONNECTED:
162                         textView_bluetoothState.setText(R.string.system_connected_message);
163                         imgView_bluetoothStatusIcon.setImageResource(R.drawable.
164                         ic_done_icon);
165                         imgView_bluetoothStatusIcon.setVisibility(View.VISIBLE);
166                         loading_progressBar.setVisibility(View.GONE);
167                         break;
168                     case BLUETOOTH_DISABLED:
169                         textView_bluetoothState.setText(R.string.bluetooth_disabled_message
170                         );
171                         imgView_bluetoothStatusIcon.setImageResource(R.drawable.
172                         ic_bluetooth_disabled);
173                         imgView_bluetoothStatusIcon.setVisibility(View.VISIBLE);
174                         loading_progressBar.setVisibility(View.GONE);
175                         break;
176                     case BT_TURNING_ON:
177                         textView_bluetoothState.setText(R.string.
178                         bluetooth_initializing_adapater);
179                         imgView_bluetoothStatusIcon.setVisibility(View.GONE);
180                         loading_progressBar.setVisibility(View.VISIBLE);
181                         break;
182                     case BT_TURNING_OFF:
```

```

177                     textView_bluetoothState.setText(R.string.bluetooth_closing_adapter)
178                 ;
179                 imgView_bluetoothStatusIcon.setVisibility(View.GONE);
180                 loading_progressBar.setVisibility(View.VISIBLE);
181                 break;
182             case SYSTEM_DISCONNECTED:
183                 textView_bluetoothState.setText(R.string.
184                     system_disconnected_message);
185                 imgView_bluetoothStatusIcon.setImageResource(R.drawable.
186                     ic_close_icon);
187                 imgView_bluetoothStatusIcon.setVisibility(View.VISIBLE);
188                 loading_progressBar.setVisibility(View.GONE);
189                 break;
190             case WAITING:
191                 textView_bluetoothState.setText(R.string.
192                     bluetooth_attempting_connection_message);
193                 imgView_bluetoothStatusIcon.setVisibility(View.GONE);
194                 loading_progressBar.setVisibility(View.VISIBLE);
195                 break;
196         }
197     } else {
198         Application.getActivity().runOnUiThread(new Runnable() {
199             @Override
200             public void run() {
201                 switch (currentBluetoothState) {
202                     case SYSTEM_CONNECTED:
203                         textView_bluetoothState.setText(R.string.
204                             system_connected_message);
205                         imgView_bluetoothStatusIcon.setImageResource(R.drawable.
206                             ic_done_icon);
207                         imgView_bluetoothStatusIcon.setVisibility(View.VISIBLE);
208                         loading_progressBar.setVisibility(View.GONE);
209                         break;
210                     case BLUETOOTH_DISABLED:
211                         textView_bluetoothState.setText(R.string.
212                             bluetooth_disabled_message);
213                         imgView_bluetoothStatusIcon.setImageResource(R.drawable.
214                             ic_bluetooth_disabled);
215                         imgView_bluetoothStatusIcon.setVisibility(View.VISIBLE);
216                         loading_progressBar.setVisibility(View.GONE);
217                         break;
218                     case BT_TURNING_ON:
219                         textView_bluetoothState.setText(R.string.
220                             bluetooth_initializing_adapater);
221                         imgView_bluetoothStatusIcon.setVisibility(View.GONE);
222                         loading_progressBar.setVisibility(View.VISIBLE);
223                         break;
224                     case BT_TURNING_OFF:
225                         textView_bluetoothState.setText(R.string.
226                             bluetooth_closing_adapter);
227                         imgView_bluetoothStatusIcon.setVisibility(View.GONE);
228                         loading_progressBar.setVisibility(View.GONE);
229                         break;
230                     case SYSTEM_DISCONNECTED:
231                         textView_bluetoothState.setText(R.string.
232                             system_disconnected_message);
233                         imgView_bluetoothStatusIcon.setImageResource(R.drawable.
234                             ic_close_icon);
235                         imgView_bluetoothStatusIcon.setVisibility(View.VISIBLE);
236                         loading_progressBar.setVisibility(View.GONE);
237                         break;
238                     case WAITING:
239                         textView_bluetoothState.setText(R.string.
240                             bluetooth_attempting_connection_message);
241                         imgView_bluetoothStatusIcon.setVisibility(View.GONE);
242                         loading_progressBar.setVisibility(View.VISIBLE);
243                         break;
244                 }
245             }
246         });
247     }
248 }

```

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

```
1 package com.example.SAPLM.bluetoothActivities;
2
3 import android.graphics.Color;
4 import android.graphics.PorterDuff;
5 import android.graphics.drawable.ColorDrawable;
6 import android.graphics.drawable.Drawable;
7 import androidx.recyclerview.widget.RecyclerView;
8 import android.view.LayoutInflater;
9 import android.view.View;
10 import android.view.ViewGroup;
11 import android.widget.ImageView;
12 import android.widget.TextView;
13
14 import com.example.SAPLM.Application;
15 import com.example.SAPLM.R;
16
17 import java.util.List;
18
19 public class DeviceListAdapter extends RecyclerView.Adapter<DeviceListAdapter.ViewHolder> {
20
21     private List<AvailableDevice> deviceList;
22
23     public class ViewHolder extends RecyclerView.ViewHolder {
24         public TextView textView_device_name;
25         public ImageView imgView_device_state_icon;
26
27         public viewHolder(View view) {
28             super(view);
29             textView_device_name = (TextView) view.findViewById(R.id.
30             textView_bluetooth_device_name);
31             imgView_device_state_icon = (ImageView) view.findViewById(R.id.
32             imageView_bluetooth_device_state_icon);
33         }
34     }
35
36     public DeviceListAdapter(List<AvailableDevice> deviceList) {
37         this.deviceList = deviceList;
38     }
39
40     @Override
41     public viewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
42         View itemView = LayoutInflater.from(parent.getContext())
43             .inflate(R.layout.bluetooth_list_layout_style, parent, false);
44
45         return new viewHolder(itemView);
46     }
47
48     @Override
49     public void onBindViewHolder(viewHolder holder, int position) {
50         AvailableDevice device = deviceList.get(position);
51         holder.textView_device_name.setText(device.getDeviceName());
52         holder.textView_device_name.setTextAppearance(device.getTextAppearance());
53
54         int iconID = device.getDeviceStateIconID();
55
56         if (iconID!=0) {
57             Drawable icon = Application.getContext().getDrawable(iconID);
58             icon.setColorFilter(Application.getContext().getColor(R.color.colorPrimary),
59             PorterDuff.Mode.SRC_ATOP);
60             holder.imgView_device_state_icon.setImageDrawable(icon);
61         } else {
62             holder.imgView_device_state_icon.clearColorFilter();
63         }
64     }
65
66     @Override
67     public int getItemCount() {
68         return deviceList.size();
69     }
70 }
```

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

```

61                         bluetoothDevicesList = new ArrayList<>();
62                         bluetoothDevicesList.addAll(bondedDevices);
63
64                         if (bondedDevices.size() > 0) {
65                             bluetoothDevicesArray = bondedDevices.toArray(new
66                             BluetoothDevice[bondedDevices.size()]);
67                             }
68                         updateBluetoothListGUI();
69                         } else {
70                             BluetoothActivity.setBluetoothStateGui(
71                             bluetoothConnectionState.BLUETOOTH_DISABLED);
72                         }
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[
91                 x].getName(), AvailableDevice.DeviceState.DEFAULT));
92             }
93
94             if (bluetoothConnected) {
95                 BluetoothActivity.setBluetoothStateGui(blueoothConnectionState.
96                 SYSTEM_CONNECTED);
97                 deviceList.get(blueoothDevicesList.indexOf(targetDevice)).setCurrentState(
98                 AvailableDevice.DeviceState.CONNECTED);
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)&&
110         isBluetoothAdapterReady() {
111             try {
112                 attemptingBluetoothConnection = true;
113                 if (bluetoothConnected) closeAllConnections();
114                 BluetoothActivity.setBluetoothStateGui(blueoothConnectionState.WAITING);
115                 targetDevice = bluetoothDevicesArray[deviceIndex];
116                 btSocket = targetDevice.createRfcommSocketToServiceRecord(btUUID);
117                 btSocket.connect();
118             }
119         }
120     }

```

```

117             if (btSocket.isConnected()) {
118                 inputStream = btSocket.getInputStream();
119                 outputStream = btSocket.getOutputStream();
120                 bluetoothConnected = true;
121                 BluetoothActivity.setBluetoothStateGui(blueoothConnectionState.
122                     SYSTEM_CONNECTED);
122                 BluetoothActivity.updateDevicesList();
123             } else {
124                 bluetoothConnected = false;
125                 BluetoothActivity.setBluetoothStateGui(blueoothConnectionState.
126                     SYSTEM_DISCONNECTED);
126             }
127             attemptingBluetoothConnection = false;
128         } catch (Throwable th) {
129             bluetoothConnected = false;
130             Application.toastMessage(th.toString());
131             BluetoothActivity.setBluetoothStateGui(blueoothConnectionState.
132                     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) &&
142     isBluetoothAdapterReady() {
143         bluetoothConnectorExecutor.execute(new Runnable() {
144             @Override
145             public void run() {
146                 connectBluetoothDeviceSync(attemptingConnectionDeviceIndex);
147             }
148         });
149     }
150 }
151
152 public static void sendCommandBuffer(byte[] data) {
153     try {
154         outputStream.write(new byte[] {(byte) 0x7F, (byte) 0x7F, (byte) 0x7F, (byte) 0x7F});
155         outputStream.write(data, 0, 20);
156     } catch (Throwable th) {
157         Application.toastMessage(th.toString());
158     }
159 }
160
161 public static byte[] receiveCommandBuffer(int timeout) {
162     long startTimeMillis = System.currentTimeMillis();
163     while((System.currentTimeMillis()-startTimeMillis)<500L) {
164         try {
165             if (inputStream.available() >= 32) {
166                 byte[] inputBuffer = new byte[32];
167                 inputStream.read(inputBuffer, 0, 32);
168                 return inputBuffer;
169             }
170         } catch (Throwable th){
171             Application.toastMessage(th.toString());
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 =
197                                         inputBuffer;
198                                     DeveloperConsoleActivity.sendTextToTerminal("Received
199                                     from: " + UnifiedTransmissionProtocol.lastMessageTransmitterModule.getModuleName() + " : "
200                                     + TerminalManager.insertSpaces(UnifiedTransmissionProtocol.bytesToHex(inputBuffer), " ", 2)
201                                     );
202                                     if (UnifiedTransmissionProtocol.
203                                         DecomposeMessageFromBuffer() == UnifiedTransmissionProtocol.CommandStatus.
204                                         SUCCESSFUL_DECOMPOSITION) {
205                                         UnifiedTransmissionProtocol.handleReceivedCommand()
206                                         ;
207                                         } else {
208                                             Application.toastMessage("UNSUCCESSFUL
209                                             DECOMPOSITION");
210                                         }
211                                         }
212                                         } catch (Throwable th) {
213                                             Application.toastMessage(th.toString());
214                                         }
215                                         isAutoConnectRunning = 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                 }
238                 while (!terminateAutoConnect && isBluetoothAdapterReady()) {
239                     listAvailableBluetoothDevices();
240                     while (isListAvailableBluetoothDevicesRunning) {}
241                     for (int x = 0; x < bluetoothDevicesArray.length; x++) {
242                         // MAC ADDRESS HC-05 : 98:D3:91:FD:38:05
243                         // MAC ADDRESS XT1068 : 5C:51:88:A2:38:97
244                         if (bluetoothDevicesArray[x].getAddress().compareToIgnoreCase("98:D3:91:FD:38:05") == 0) {
245                             if (!terminateAutoConnect&&!bluetoothConnected)
246                                 connectBluetoothDeviceSync(x);
247                             }
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,
297                     BluetoothAdapter.ERROR);
298
299                 switch (stateCode){
300                     case BluetoothAdapter.STATE_OFF:
301                         //Bluetooth adapter turned off handler
302                         BluetoothActivity.setBluetoothStateGui(blueoothConnectionState.
303                             BLUETOOTH_DISABLED);
304                         break;
305                     case BluetoothAdapter.STATE_TURNING_OFF:
306                         //Bluetooth adapter turning off handler
307                         terminateBTAutoConnect();
308                         BluetoothActivity.setBluetoothStateGui(blueoothConnectionState.
309                             BT_TURNING_OFF);
310                         break;
311                     case BluetoothAdapter.STATE_ON:
312                         BluetoothActivity.setBluetoothStateGui(blueoothConnectionState.
313                             SYSTEM_DISCONNECTED);
314                         updateDevicesList();
315                         initBTAutoConnectService();
316
317                         break;
318                     case BluetoothAdapter.STATE_TURNING_ON:
319                         //Bluetooth adapter turning on handler
320                         BluetoothActivity.setBluetoothStateGui(blueoothConnectionState.
321                             BT_TURNING_ON);
322                         break;
323
324         public static BroadcastReceiver bluetoothScanModeStatusBR = new BroadcastReceiver() {
325
326             @Override
327             public void onReceive(Context context, Intent intent) {
328                 final String action = intent.getAction();
329
330                 if (action.equals(BluetoothAdapter.ACTION_SCAN_MODE_CHANGED)) {
331
332                     int mode = intent.getIntExtra(BluetoothAdapter.EXTRA_SCAN_MODE,
333                         BluetoothAdapter.ERROR);
334
335                     switch (mode) {
336                         case BluetoothAdapter.SCAN_MODE_CONNECTABLE_DISCOVERABLE:
337
338                             break;
339                         case BluetoothAdapter.SCAN_MODE_CONNECTABLE:
340
341                             break;
342                         case BluetoothAdapter.SCAN_MODE_NONE:
343
344                             break;
345
346                     }
347
348                 }
349             }
350         }

```

```
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(blueoothConnectionState.
363 SYSTEM_DISCONNECTED);
364                     initBTAutoConnectService();
365                     updateBluetoothListGUI();
366                     //Application.toastMessage("Dispositivo desconectado!");
367                     break;
368                 case BluetoothDevice.ACTION_ACL_DISCONNECT_REQUESTED:
369                     break;
370                 case BluetoothDevice.ACTION_BOND_STATE_CHANGED:
371
372                     break;
373             }
374         }
375     };
376
377 }
378
```

```

1 package com.example.SAPLM.bluetoothActivities;
2
3 import com.example.SAPLM.R;
4 import com.example.SAPLM.commandActivity.CommandsActivity;
5 import com.example.SAPLM.settingsActivities.DeveloperConsoleActivity;
6 import com.example.SAPLM.settingsActivities.TerminalManager;
7
8
9 import java.io.UnsupportedEncodingException;
10 import java.nio.ByteBuffer;
11 import java.nio.ByteOrder;
12 import java.nio.charset.StandardCharsets;
13 import java.util.ArrayList;
14 import java.util.Arrays;
15 import java.util.List;
16 import java.util.concurrent.Executor;
17 import java.util.concurrent.Executors;
18 import java.util.regex.Matcher;
19 import java.util.regex.Pattern;
20
21
22
23 public class UnifiedTransmissionProtocol {
24
25     public static List<Module> modulesList = new ArrayList<UnifiedTransmissionProtocol.
26     Module>(Arrays.asList(
27         new Module("Dispositivo Móvil", "PHONE", (byte) 0x00, R.drawable.
28         ic_icon_phone_module, new ArrayList<Module.Device>(Arrays.asList(
29             new Module.Device("Módulo principal", new Boolean(false), R.drawable.
30             ic_icon_main_board, (byte) 0x00, new Boolean(true))
31             )),
32             new Module("Módulo principal", "MAIN", (byte) 0x01, R.drawable.ic_icon_main_board,
33             new ArrayList<Module.Device>(Arrays.asList(
34                 new Module.Device("Módulo de potencia", new Boolean(false), R.drawable.
35                 icons_board, (byte) 0x00, new Boolean(true)),
36                 new Module.Device("Módulo motriz", new Boolean(false), R.drawable.
37                 icons_board, (byte) 0x01, new Boolean(true))
38             )),
39             new Module("Módulo de potencia", "POWER", (byte) 0x02, R.drawable.icons_board, new
40             ArrayList<Module.Device>(Arrays.asList(
41                 new Module.Device("Velador", new Boolean(false), R.drawable.
42                 icons_desk_lamp, (byte) 0x00, new Boolean(false)),
43                 new Module.Device("Luz", new Boolean(false), R.drawable.icons_bulb, (
44                     byte) 0x01, new Boolean(false)),
45                 new Module.Device("Ventilador", new Boolean(false), R.drawable.icons_fan
46             , (byte) 0x02, new Boolean(false)),
47                 new Module.Device("Estufa", new Boolean(false), R.drawable.icons_heater,
48                 (byte) 0x03, new Boolean(false))
49             )),
50             new Module("Módulo motriz", "MOTOR", (byte) 0x03, R.drawable.icons_board, new
51             ArrayList<Module.Device>(Arrays.asList(
52                 new Module.Device("Camilla", new Byte((byte) 0), R.drawable.
53                 icons_stretcher, (byte) 0x00, new Boolean(false)),
54                 new Module.Device("Cortina", new Byte((byte) 0), R.drawable.icons_curtain
55             , (byte) 0x01, new Boolean(false)),
56                 new Module.Device("Llamar enfermera", new Boolean(false), R.drawable.
57                 icons_nurse, (byte) 0x02, new Boolean(false))
58             )));
59
60     protected static byte[] command_buffer = new byte[32];
61
62     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    }

```

Archivo - UnifiedTransmissionProtocol.java

```

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         );
117             Pattern footerPattern = Pattern.compile("\u0003.{1}\u0017", Pattern.DOTALL
118         );
119             Matcher headerMatcher = headerPattern.matcher(commandString);
120             Matcher footerMatcher = footerPattern.matcher(commandString);
121             if (headerMatcher.find()==false) return CommandStatus.WRONG_HEADER_SEGMENTATION
122         ;
123             if (footerMatcher.find()==false) return CommandStatus.WRONG_FOOTER_SEGMENTATION
124         ;
125             int headerstartIndex = headerMatcher.start();
126             int parameterstartIndex = headerMatcher.end()+1;
127             int footerstartIndex = footerMatcher.start();
128             lastMessagePID = command_buffer[headerstartIndex+1]; //verificar si hay que
convertir a desde unsigned
129             if (command_buffer[headerstartIndex+2]!=currentModuleID.getInternalCode())
return CommandStatus.WRONG_MODULE_ID;
130             lastMessageTransmitterModule = Module.getModule(command_buffer[headerstartIndex
+3]);
131             lastCommandType = CommandType.definedCommandList.stream().filter(p-> p.
getCommandCode()==command_buffer[headerMatcher.end()]).findFirst().orElse(null);
132             lastParameterList = new ArrayList<Parameter>();
133             while(parameterstartIndex!=footerstartIndex) {
134                 int parameterLength = command_buffer[parameterstartIndex];
135                 byte[] parameterData = new byte[parameterLength];
136                 System.arraycopy(command_buffer, parameterstartIndex+1, parameterData, 0,
parameterLength);
137                 lastParameterList.add(new Parameter(parameterData));
138                 parameterstartIndex+=(1+parameterLength);
139             }
140             if (command_buffer[footerstartIndex+1]!=CRC8_CCITT.crc_CCITT(command_buffer,
footerstartIndex)) return CommandStatus.WRONG_CHECKSUM_CONSISTENCY;
141         } catch (UnsupportedEncodingException e) {
142             e.printStackTrace();
143             return CommandStatus.FAILED_DECOMPOSITION;
144         }
145
146         isAvailableCommand = true;
147         return CommandStatus.SUCCESSFUL_DECOMPOSITION;
148     }
149
150
151     isAvailableCommand = true;
152     return CommandStatus.SUCCESSFUL_DECOMPOSITION;
153 }
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,
195     List<Device> devicesList) {
196         this.moduleName = moduleName;
197         this.internalCode = internalCode;
198         this.iconResourceId = iconResourceId;
199         this.devicesList = devicesList;
200     }
201
202     Module() {
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.
231             Parameter>();
232
233         for (int x = 0; x < devicesList.size(); x++) {
234             for (Module.Device device : devicesList) {
235                 if (x==byteToUnsignedInt(device.getInternalIndex())) {
236                     Parameter newParameter = null;
237
238                     if (device.getData().getClass()==Byte.class) {
239                         ByteBuffer byteBuffer = ByteBuffer.allocate(Byte.BYTES);
240                         byteBuffer.order(ByteOrder.LITTLE_ENDIAN);
241                         byteBuffer.put((byte) device.getData());
242                         newParameter = new Parameter(byteBuffer.array());
243                     } else if (device.getData().getClass()==Short.class) {
244                         ByteBuffer shortBuffer = ByteBuffer.allocate(Short.BYTES);
245                         shortBuffer.order(ByteOrder.LITTLE_ENDIAN);
246                         shortBuffer.putShort((short) device.getData());
247                         newParameter = new Parameter(shortBuffer.array());
248                     } else if (device.getData().getClass()==Integer.class) {
249                         ByteBuffer intBuffer = ByteBuffer.allocate(Integer.BYTES);
250                         intBuffer.order(ByteOrder.LITTLE_ENDIAN);
251                         intBuffer.putInt((int) device.getData());
252                         newParameter = new Parameter(intBuffer.array());
253                     } else if (device.getData().getClass()==Float.class) {
254                         ByteBuffer floatBuffer = ByteBuffer.allocate(Float.BYTES);
255                         floatBuffer.order(ByteOrder.LITTLE_ENDIAN);
256                         floatBuffer.putFloat((float) (double)device.getData());
257                         newParameter = new Parameter(floatBuffer.array());
258                     } else if (device.getData().getClass()==Double.class) {
259                         ByteBuffer doubleBuffer = ByteBuffer.allocate(Double.BYTES);
260                         doubleBuffer.order(ByteOrder.LITTLE_ENDIAN);
261                         doubleBuffer.putFloat((float) (double)device.getData());
262                         newParameter = new Parameter(doubleBuffer.array());
263                     } else if (device.getData().getClass()==String.class) {
264                         String data = (String) device.getData();
265                         newParameter = new Parameter(data.getBytes(StandardCharsets.
266                             US_ASCII));
267                     } else if (device.getData().getClass()==Boolean.class) {
268                         byte data = (byte) 0x00;
269                         if ((Boolean)device.getData()==true) data = (byte) 0xFF;
270                         ByteBuffer booleanBuffer = ByteBuffer.allocate(Byte.BYTES);
271                         booleanBuffer.order(ByteOrder.LITTLE_ENDIAN);
272                         booleanBuffer.put(data);
273                         newParameter = new Parameter(booleanBuffer.array());
274                     }
275
276                     if (newParameter!=null) exportList.add(newParameter);
277                     break;
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     );
291         buffer.order(ByteOrder.LITTLE_ENDIAN);
292
293         if (deviceToUpdate.getData().getClass() == Byte.class) {
294             deviceToUpdate.setData(new Byte(buffer.get()));
295         } else if (deviceToUpdate.getData().getClass() == Short.class) {
296             deviceToUpdate.setData(new Short(buffer.getShort()));
297         } else if (deviceToUpdate.getData().getClass() == Integer.class) {
298             deviceToUpdate.setData(new Integer(buffer.getInt()));
299         } else if (deviceToUpdate.getData().getClass() == Float.class) {
300             deviceToUpdate.setData(new Float(buffer.getFloat()));
301         } else if (deviceToUpdate.getData().getClass() == Double.class) {
302             deviceToUpdate.setData(new Double(buffer.getFloat()));
303         } else if (deviceToUpdate.getData().getClass() == String.class) {
304             deviceToUpdate.setData(new String(parameterList.get(x).getParameterData()
305             (), StandardCharsets.US_ASCII));
306         } else if (deviceToUpdate.getData().getClass() == Boolean.class) {
307             if (buffer.get() == (byte) 0xFF) {
308                 deviceToUpdate.setData(new Boolean(true));
309             } else {
310                 deviceToUpdate.setData(new Boolean(false));
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(Double.BYTES);
374                     doubleBuffer.order(ByteOrder.LITTLE_ENDIAN);
375                     doubleBuffer.putFloat((float) ((double) devicesList.get(x).getData()))
376
377                     return doubleBuffer.array();
378                 } else if (devicesList.get(x).getData().getClass() == String.class) {
379                     String data = (String) devicesList.get(x).getData();
380                     return data.getBytes(StandardCharsets.US_ASCII);
381                 } else if (devicesList.get(x).getData().getClass() == Boolean.class) {
382                     byte data = (byte) 0x00;
383                     if ((Boolean) devicesList.get(x).getData() == true) data = (byte) 0xFF;
384                     ByteBuffer booleanBuffer = ByteBuffer.allocate(Byte.BYTES);
385                     booleanBuffer.order(ByteOrder.LITTLE_ENDIAN);
386                     booleanBuffer.put(data);
387                     return booleanBuffer.array();
388                 }
389             }
390
391             return null;
392         }
393
394
395         private static Executor synchronizationExecutor = Executors.newSingleThreadExecutor
396         ();
397         private static boolean synchronizationRunning = false;
398
399         public static boolean awaitingModuleResponse = false;
400         public static byte awaitingModuleResponse_ModuleInternalCode = (byte) 0x00;
401         public static boolean awaitingSynchronization = false;
402
403         public static void syncAllDevicesValueToSystem() {
404             // Send data to system from internal data
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 &&
416                                 dev.isSynchronizedData() == false) {
417                                 List<Parameter> export = mod.
418                                     exportDataToParameterList();
419                                 ComposeMessageToBuffer(CommandType.
420                                     UPDATE_ALL_DEVICES_VALUE, mod, export);
421                                 DeveloperConsoleActivity.sendTextToTerminal("Sent
422                                     to: " + mod.getModuleName() + " : " + TerminalManager.insertSpaces(bytesToHex(
423                                     command_buffer), " ", 2));
424                                 awaitingModuleResponse = true;
425                                 awaitingModuleResponse_ModuleInternalCode = mod.
426                                     internalCode;
427                                 BluetoothConnection.sendCommandBuffer(
428                                     command_buffer);
429                                 long startTimeMillis = System.currentTimeMillis();
430                                 while ((System.currentTimeMillis() -
431                                     startTimeMillis) < 1000L) {
432                                     if (awaitingModuleResponse == false) {
433                                         for (Module.Device devSync : mod.
434                                             getDevicesList()) {
435                                             devSync.setSynchronizedState(true);
436                                         }
437                                     }
438                                     awaitingModuleResponse = false;
439                                     break; //Only send one message per module
440                                 }
441                             });
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                 synchronizationRunning = true;
452
453                 synchronizationRunning = false;
454             }
455
456
457
458
459             public static Module getModule(byte internalCode) {
460                 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
483             .lineSeparator();
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 {
554                     setData(Boolean.FALSE);
555                 }
556                 break;
557             case "Luz":
558                 if ((boolean) data == false) { setData(Boolean.TRUE); } else {
559                     setData(Boolean.FALSE);
560                 }
561                 break;
562             case "Ventilador":
563                 if ((boolean) data == false) { setData(Boolean.TRUE); } else {
564                     setData(Boolean.FALSE);
565                 }
566                 break;
567             case "Estufa":
568                 if ((boolean) data == false) { setData(Boolean.TRUE); } else {
569                     setData(Boolean.FALSE);
570                 }
571                 break;
572             case "Camilla":
573                 if (byteToUnsignedInt((byte) data) < 3) { setData((byte) (((byte) data) +
574 1)); } else { setData((byte) 0x00); }
575                 break;
576             case "Cortina":
577                 if (byteToUnsignedInt((byte) data) < 7) { setData((byte) (((byte) data) +
578 1)); } else { setData((byte) 0x00); }
579                 break;
580             case "Llamar enfermera":
581                 if ((boolean) data == false) {
582                     setData(Boolean.TRUE);
583
584                     autoToggleNurseStateThread = new Thread(
585                         autoToggleNurseStateRunnable);
586                     autoToggleNurseStateThread.start();
587                 } else { setData(Boolean.FALSE); }
588                 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
587 = 100; }
588                 break;
589             case "Módulo de potencia":
590                 if ((boolean) data == false) { returnValue = 0; } else { returnValue
591 = 100; }
592                 break;
593             case "Módulo motriz":
594                 if ((boolean) data == false) { returnValue = 0; } else { returnValue
595 = 100; }
596                 break;
597             case "Velador":
598                 if ((boolean) data == false) { returnValue = 0; } else { returnValue
599 = 100; }
600                 break;
601             case "Luz":
602                 if ((boolean) data == false) { returnValue = 0; } else { returnValue
603 = 100; }
604                 break;
605             case "Ventilador":
606                 if ((boolean) data == false) { returnValue = 0; } else { returnValue
607 = 100; }
608                 break;
609             case "Estufa":
610                 if ((boolean) data == false) { returnValue = 0; } else { returnValue
611 = 100; }
612                 break;
613             case "Camilla":
614                 returnValue = Math.round((float)((byte) data)*33.333f));
615                 break;
616             case "Cortina":
617                 returnValue = Math.round((float)((byte) data)*14.286f));
618                 break;
619             case "Llamar enfermera":
620                 if ((boolean) data == false) { returnValue = 0; } else { returnValue
621 = 100; }
622                 break;
623         }
624         return returnValue;
625     }
626
627     public String getDataAsString(){
628         String returnString = "";
629         switch (name){
630             case "Módulo principal":
631                 if ((boolean) data == false) { returnString = "Desconectado"; } else
632 { returnString = "En linea"; }
633                 break;
634             case "Módulo de potencia":
635                 if ((boolean) data == false) { returnString = "Desconectado"; } else
636 { returnString = "En linea"; }
637                 break;
638             case "Módulo motriz":
639

```

```

631                     if ((boolean) data == false) { returnString = "Desconectado"; } else
632             { returnString = "En linea"; }
633             break;
634
635         case "Velador":
636             if ((boolean) data == false) { returnString = "Apagado"; } else {
637                 returnString = "Encendido"; }
638                 break;
639             case "Luz":
640                 if ((boolean) data == false) { returnString = "Apagado"; } else {
641                     returnString = "Encendido"; }
642                     break;
643                 case "Ventilador":
644                     if ((boolean) data == false) { returnString = "Apagado"; } else {
645                         returnString = "Encendido"; }
646                         break;
647
648         case "Camilla":
649             returnString = "Posición " + byteToUnsignedInt((byte) data);
650             break;
651         case "Cortina":
652             returnString = "Posición " + byteToUnsignedInt((byte) data);
653             break;
654         case "Llamar enfermera":
655             if ((boolean) data == false) { returnString = "Disponible"; } else {
656                 returnString = "Llamando"; }
657                 break;
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(
667                     false));
668                 modulesList.get(3).getDevicesList().get(2).setSynchronizedState(
669                     true);
670                 CommandsActivity.updateGUI();
671             } catch (InterruptedException e) {
672                 e.printStackTrace();
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         CommandType(byte cmdCode, String tag, Runnable handlerCommand) {
700             this.commandCode = cmdCode;
701             this.commandTag = tag;
702             this.handlerRunnable = handlerCommand;
703         }
704
705         public byte getCommandCode() {
706             return commandCode;
707         }
708
709         public String getCommandTag() {
710             return commandTag;
711         }
712
713         public void setCommandCode(byte commandCode) {
714             this.commandCode = commandCode;
715         }
716
717         public void setCommandTag(String commandTag) {
718             this.commandTag = commandTag;
719         }
720
721         public void handleCommand() {
722             handlerRunnable.run();
723         }
724
725
726
727         public static ArrayList<CommandType> definedCommandList = new ArrayList<CommandType>(
    >(Arrays.asList(
        new CommandType((byte)0x00, "UPDATE_ALL_DEVICES_VALUE", new Runnable() {
            @Override
            public void run() {
                modulesList.get(byteToUnsignedInt(lastMessageTransmitterModule.
                    getIntInternalCode())) .updateDataFromParameterList(lastParameterList);
            }
            System.out.println("Running UPDATE_ALL_DEVICES_VALUE!");
        }),
        new CommandType((byte)0x01, "UPDATE_DEVICE_VALUE", new Runnable() {
            @Override
            public void run() {
                ByteBuffer buffer = ByteBuffer.wrap(lastParameterList.get(0) .
                    getParameterData());
                modulesList.get(byteToUnsignedInt(lastMessageTransmitterModule.
                    getIntInternalCode())) .updateDeviceValue(buffer.get(), lastParameterList.get(1));
            }
        })
    ))
)

```

```

744     getParameterData();
745
746             System.out.println("Running UPDATE_DEVICE_VALUE!");
747
748
749         }}}},
750     new CommandType((byte)0x02, "GET_ALL_DEVICES_VALUE", new Runnable() {
751         @Override
752         public void run() {
753             ComposeMessageToBuffer(defineCommandList.stream().filter(p-> p.
754             getCommandTag().equals(("UPDATE_ALL_DEVICES_VALUE"))).findFirst().orElse(null),
755                     lastMessageTransmitterModule,
756                     modulesList.get(byteToUnsignedInt(
757                         lastMessageTransmitterModule.getInternalCode()))).exportDataToParameterList());
758
759             // SEND COMMAND_BUFFER VIA BLUETOOTH
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).
768             getParameterData());
769             byte[] deviceData = modulesList.get(byteToUnsignedInt(
770                 lastMessageTransmitterModule.getInternalCode())).exportDeviceValue(buffer.get());
771
772             ComposeMessageToBuffer(defineCommandList.stream().filter(p-> p.
773             getCommandTag().equals(("UPDATE_DEVICE_VALUE"))).findFirst().orElse(null),
774                     lastMessageTransmitterModule,
775                     new ArrayList<Parameter>(Arrays.asList(new Parameter(new
776                     byte[] {buffer.get()}), new Parameter(deviceData))));
777
778             // SEND COMMAND_BUFFER VIA BLUETOOTH
779
780             System.out.println("Running GET_DEVICE_VALUE!");
781         }}),
782     new CommandType((byte)0x04, "MESSAGE_STATUS", new Runnable() {
783         @Override
784         public void run() {
785
786             ByteBuffer buffer = ByteBuffer.wrap(lastParameterList.get(0).
787             getParameterData());
788             if (buffer.get()==(byte)0x09){
789                 Module.awaitingModuleResponse = false;
790             }
791
792
793         }});
794
795         public static CommandType UPDATE_ALL_DEVICES_VALUE = defineCommandList.get(0);
796         public static CommandType UPDATE_DEVICE_VALUE = defineCommandList.get(1);
797         public static CommandType GET_ALL_DEVICES_VALUE = defineCommandList.get(2);
798         public static CommandType GET_DEVICE_VALUE = defineCommandList.get(3);
799         public static CommandType MESSAGE_STATUS = defineCommandList.get(4);
800     }

```

```
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
```