**SIMULACIONES**

**MÓDULO #16: IV + IoT + ESP32 + ANDROID**

**Diego Luis Aristizábal R, Profesor asociado con tenencia del cargo de la Escuela de Física de la Universidad Nacional de Colombia Sede Medellín**

**2023-01**

|  |
| --- |
| Temas |
| * INTRODUCCIÓN * APP CON ESP32 IoT: RECIBIENDO DATOS * APP CON ESP32 IoT: ENVIANDO DATOS * TAREA * REFERENCIAS |

**INTRODUCCIÓN**

En este módulo se diseñarán e implementarán tres apps que comunican vía **IoT con protocolo MQTT** un dispositivo móvil **ANDROID** con la tarjeta **ESP32** a la cual están conectados sensores o leds. El patrón de comunicación a usar es el **PUBLICADOR-SUSCRIPTOR** en donde tanto el dispositivo móvil como la tarjeta **ESP32** son CLIENTES.

En la primera app, el **dispositivo móvil Android** controla **vía IoT** el color desplegado por un **led RGB** el cual está conectado a las **ESP32**. El **móvil** **PUBLICA**.

En la segunda app el **dispositivo móvil Android** recibe datos **vía IoT** de la **ESP32** obtenidos de la medida de la iluminacia que realiza el **sensor de iluminancia** **GY-30 BH1750** que está conectado a ella. Estos datos los despliega en gauge, tabla y gráfica. La **ESP 32** **PUBLICA**.

En la tercera app el **dispositivo móvil Android** recibe datos **vía IoT** de la **ESP32** obtenidos de la medida de las distancias de obstáculos que realiza el **sensor ultrasónico** **HC-SR04** que está conectado a ella. Estos datos los despliega en gauge, tabla y gráfica. Adicionalmente con bases a rangos de esta distancia el **dispoitivo móvil** controla **vía IoT** el color de un **led RGB** conectado a la **ESP32**. Aquí ambos **PUBLICAN**.

**APP CON ESP32 IoT: RECIBIENDO DATOS**

En este primer ejemplo, se tiene un **led RGB** acoplado a la **ESP32**. Mediante un **dispositivo móvil Android** se controla **vía IoT** el color de la luz emitida por el **led RGB**. Aquí el **dispositivo móvil** **Android** **PUBLICA**.

**La app**

Crear un proyecto denominado **MiCuadragesimaTerceraApp** con la jerarquía de paquetes y clases de la Figura 1.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Figura 1

Para implementar el IoT se usará la librería **Eclipse Paho**. Para esto es necesario seguir los siguientes pasos:

* Bajar los siguientes .jar:

[org.eclipse.paho.android.service-1.1.1.jar](https://repo1.maven.org/maven2/org/eclipse/paho/org.eclipse.paho.android.service/1.1.1/org.eclipse.paho.android.service-1.1.1.jar)

[org.eclipse.paho.client.mqttv3-1.2.0.jar](https://repo1.maven.org/maven2/org/eclipse/paho/org.eclipse.paho.client.mqttv3/1.2.0/org.eclipse.paho.client.mqttv3-1.2.0.jar)

y agegarlos al la carpeta **libs** del proyecto.

* En el archivo manifiesto agregar:

. . .

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

* En el arcchivo build.gardle (module: …) agregar:

dependencies **{** . . .

implementation files('libs/org.eclipse.paho.android.service-1.1.1.jar')  
 implementation files('libs/org.eclipse.paho.client.mqttv3-1.2.0.jar')  
  
**}**

* Por último **sincronizar** el proyecto con Gradles Files, Figura 2

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Figura 2

Las siguientes imágenes se deben incorporar al proyecto, Figura 3.

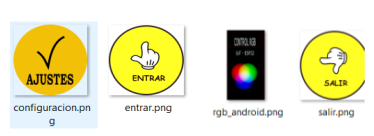


Figura 3

El código de las clases es el siguiente.

La clase **AlmacenDatosRAM**.

public class AlmacenDatosRAM {  
  
  
 public static int *ancho*, *alto*, *dimensionReferencia*, *tamanoLetraResolucionIncluida*;  
  
 public static int *estado\_conexion\_nube*=1;  
  
 public static String *MQTTHOST*;  
 public static String *USERNAME*;  
 public static String *PASSWORD*;  
 public static String *topicStr*;  
  
 public static String *conectado\_PubSub*=" ";  
  
 public static boolean *conectado*=false;  
}

La clase **DialogoSalir**:

public class DialogoSalir {  
  
 private Activity actividad;  
 private PopupWindow popup\_salir;  
 private Button si,no;  
 private TextView salir;  
 private int tamano\_letra\_25;  
 private int tamano\_letra\_20;  
 private LinearLayout linear\_layout\_principal;  
  
  
 public DialogoSalir(Activity actividad){  
  
 this.actividad=actividad;  
  
 gestionarResolucion();  
  
 crearGUI();  
  
 }  
  
  
 private void gestionarResolucion( ){  
  
 tamano\_letra\_25 = (int)(0.8f\* AlmacenDatosRAM.*tamanoLetraResolucionIncluida*);  
 tamano\_letra\_20 = (int)(1.2f\*AlmacenDatosRAM.*tamanoLetraResolucionIncluida*);  
  
 }  
  
 private void crearGUI(){  
  
 linear\_layout\_principal = new LinearLayout(actividad);  
 linear\_layout\_principal.setBackgroundColor(Color.*rgb*(183,216,199));  
 linear\_layout\_principal.setOrientation(LinearLayout.*VERTICAL*);  
 linear\_layout\_principal.setWeightSum(2);*//3  
  
 //LinearLayoutArriba* LinearLayout linear\_layout\_arriba = new LinearLayout(actividad);  
 linear\_layout\_arriba.setOrientation(LinearLayout.*HORIZONTAL*);  
  
 *//LinearLayoutMedio* LinearLayout linear\_layout\_medio = new LinearLayout(actividad);  
 linear\_layout\_medio.setOrientation(LinearLayout.*HORIZONTAL*);  
  
 *//LinearLayoutAbajo* LinearLayout linear\_layout\_abajo = new LinearLayout(actividad);  
 linear\_layout\_abajo.setOrientation(LinearLayout.*HORIZONTAL*);  
  
  
 *//parametro para pegar elementos a cada linearyout* LinearLayout.LayoutParams parametros = new LinearLayout.LayoutParams(0, ViewGroup.LayoutParams.*MATCH\_PARENT*);  
 parametros.weight = 1.0f;  
  
 *//elementos* salir=new TextView(actividad);  
 salir.setTextSize(tamano\_letra\_20);  
 salir.setText(" ¿SALIR? ");  
 salir.setTextColor(Color.*BLACK*);  
  
 si=new Button(actividad);  
 si.setTextSize(tamano\_letra\_25);  
 si.setText("SI");  
 si.getBackground().setColorFilter(Color.*rgb*(183, 216, 199), PorterDuff.Mode.*MULTIPLY*);  
  
 no=new Button(actividad);  
 no.setTextSize(tamano\_letra\_25);  
 no.setText("NO");  
 no.getBackground().setColorFilter(Color.*rgb*(183, 216, 199), PorterDuff.Mode.*MULTIPLY*);  
  
  
 linear\_layout\_arriba.addView(salir, parametros);  
 linear\_layout\_medio.addView(si,parametros);  
 linear\_layout\_medio.addView(no,parametros);  
 *//linear\_layout\_abajo.addView(no,parametros);* linear\_layout\_principal.addView(linear\_layout\_arriba);  
 linear\_layout\_principal.addView(linear\_layout\_medio);  
 *//linear\_layout\_principal.addView(linear\_layout\_abajo);* eventosPopMenu();  
  
  
 }  
  
  
 public void mostrarPopMenuCoeficientes() {  
  
 popup\_salir = new PopupWindow(actividad);  
  
 popup\_salir.setWidth(WindowManager.LayoutParams.*WRAP\_CONTENT*);  
 popup\_salir.setHeight(WindowManager.LayoutParams.*WRAP\_CONTENT*);  
  
 popup\_salir.setContentView(linear\_layout\_principal);  
 popup\_salir.setFocusable(true);  
 popup\_salir.showAtLocation(linear\_layout\_principal, Gravity.*CENTER*, 0, 0);  
  
  
 }  
  
 private void eventosPopMenu() {  
  
 si.setOnClickListener(new View.OnClickListener() {  
  
 public void onClick(View v) {  
  
 actividad.finish();  
 popup\_salir.dismiss();  
  
 }  
 });  
  
 no.setOnClickListener(new View.OnClickListener() {  
  
 public void onClick(View v) {  
  
 popup\_salir.dismiss();  
  
 }  
 });  
  
 }  
  
  
}

La clase **Boton**.

public class Boton extends ImageView {  
  
 private Bitmap imagen;  
 Bitmap imagen\_escalada = null;  
 private float tamamano\_letra;  
 private String cadena = "";  
  
  
 public Boton(Context context) {  
  
 super(context);  
  
  
 }  
  
  
 public void setText(String cadena) {  
  
 this.cadena = cadena;  
  
  
 }  
  
  
 public String getText() {  
  
 this.cadena = cadena;  
  
 return cadena;  
  
  
 }  
  
  
 public void setImagen(int imagen\_importada) {  
  
  
 imagen = BitmapFactory.*decodeResource*(getResources(), imagen\_importada);  
  
  
 }  
  
  
 protected void onDraw(Canvas canvas) {  
  
  
 Paint pincel = new Paint();  
 pincel.setAntiAlias(true);  
 pincel.setLinearText(true);  
  
 float ancho=getWidth();  
 float alto=getHeight();  
 float escala=1;  
 if(ancho>alto){  
 escala=alto;  
 } else {escala=ancho;}  
  
 if (imagen != null) {  
  
 imagen\_escalada = escalarImagen(imagen, (int) (0.9f \* escala), (int) (0.9f \* escala));  
  
 }  
  
 int ex = (getWidth() - imagen\_escalada.getWidth()) / 2;  
 int ey = (getHeight() - imagen\_escalada.getHeight()) / 2;  
  
 tamamano\_letra = 0.08f \* getWidth();  
  
 if (imagen != null) {  
 canvas.drawBitmap(imagen\_escalada, ex, ey, null);*//dimension,dimension, null);* }  
  
  
 pincel.setTextSize(tamamano\_letra);  
 float anchoCadenaUnidades = pincel.measureText(cadena);  
 float posicion\_x\_letra = (getWidth() - anchoCadenaUnidades) / 2;  
 float posicion\_y\_letra = ey + imagen\_escalada.getHeight() + tamamano\_letra;  
  
 pincel.setColor(Color.*BLACK*);  
 canvas.drawText(cadena, posicion\_x\_letra, posicion\_y\_letra, pincel);  
  
  
 invalidate();  
  
 }  
  
 */\*  
 Se escala con Matrix para no dañar la resolución  
 \*/* public Bitmap escalarImagen(Bitmap bitmap, int newWidth, int newHeight) {  
 Bitmap scaledBitmap = Bitmap.*createBitmap*(newWidth, newHeight, bitmap.getConfig());  
  
 float scaleX = newWidth / (float) bitmap.getWidth();  
 float scaleY = newHeight / (float) bitmap.getHeight();  
  
 Matrix scaleMatrix = new Matrix();  
 scaleMatrix.setScale(scaleX, scaleY, 0, 0);  
  
 Canvas canvas = new Canvas(scaledBitmap);  
 canvas.setMatrix(scaleMatrix);  
 Paint paint = new Paint(Paint.*FILTER\_BITMAP\_FLAG*);  
 paint.setAntiAlias(true);  
 paint.setDither(true);  
 paint.setFilterBitmap(true);  
 canvas.drawBitmap(bitmap, 0, 0, paint);  
  
 return scaledBitmap;  
  
 }  
  
  
}

La clase **TableroColor**.

public class TableroColor extends View {  
  
 private Context context;  
 private int rojo=100;  
 private int verde=200;  
 private int azul=60;  
 private int color\_rgb, color\_azul, color\_rojo,color\_verde;  
  
 public TableroColor(Context context) {  
  
 super(context);  
  
 this.context = context;  
  
  
 eventos();  
 }  
  
  
 private void eventos(){  
  
 this.setOnTouchListener(new View.OnTouchListener() {  
 *//@Override* public boolean onTouch(View v, MotionEvent event) {  
  
 int action = event.getAction();  
 *//x = event.getX();//en pixeles  
 // y = event.getY();//en pixeles* switch (action) {  
 case MotionEvent.*ACTION\_DOWN*:  
  
  
 break;  
  
 case MotionEvent.*ACTION\_MOVE*:  
  
  
 break;  
  
 case MotionEvent.*ACTION\_UP*:  
  
  
 break;  
  
  
 }  
  
  
 return true;  
 }  
 });  
  
  
 }  
  
  
 public void setColoRojo(int rojo){  
  
 this.rojo=rojo;  
  
 }  
  
 public void setColoVerde(int verde){  
  
 this.verde=verde;  
  
 }  
  
 public void setColoAzul(int azul){  
  
 this.azul=azul;  
  
 }  
  
  
 *//Método para dibujar* protected void onDraw(Canvas canvas) {  
  
  
 Paint pincel = new Paint();  
 pincel.setAntiAlias(true);  
 pincel.setLinearText(true);  
  
 pincel.setFilterBitmap(true);*//si se escala* pincel.setDither(true);*//si se escala* float ancho = this.getWidth();  
 float alto = this.getHeight();  
  
 float a=0;  
  
 if( ancho>alto){  
 a=alto;  
 } else{  
  
 a=ancho;  
  
 }  
  
  
  
 float indent=0.05f\*alto;  
  
 color\_rgb= Color.*rgb*(rojo,verde,azul);  
 pincel.setColor(color\_rgb);  
 pincel.setStyle(Paint.Style.*FILL*);  
 canvas.drawRect(indent, indent, 0.5f \* ancho - indent, alto - indent, pincel);  
 pincel.setColor(Color.*BLACK*);  
 pincel.setStyle(Paint.Style.*STROKE*);  
 canvas.drawRect(indent, indent, 0.5f \* ancho - indent, alto - indent, pincel);  
  
 pincel.setStyle(Paint.Style.*FILL*);  
 color\_rojo= Color.*rgb*(rojo, 0, 0);  
 pincel.setColor(color\_rojo);  
 canvas.drawRect(indent + 0.5f \* ancho, indent, ancho - indent, indent + 0.1f \* alto, pincel);  
  
  
  
 pincel.setTextSize(0.05f \* a);  
 pincel.setColor(Color.*RED*);  
 canvas.drawText("Rojo= " + rojo, indent + 0.5f \* ancho, indent + 0.2f \* alto, pincel);  
  
  
 color\_verde = Color.*rgb*(0, verde, 0);  
 pincel.setColor(color\_verde);  
 canvas.drawRect(indent + 0.5f \* ancho, indent + 0.3f \* alto, ancho - indent, indent + 0.4f \* alto, pincel);  
  
 pincel.setColor(Color.*rgb*(0,128,0));  
 canvas.drawText("Verde= " + verde, indent + 0.5f \* ancho, indent + 0.5f \* alto, pincel);  
  
 color\_azul = Color.*rgb*(0, 0, azul);  
 pincel.setColor(color\_azul);  
 canvas.drawRect(indent + 0.5f \* ancho, indent + 0.6f \* alto, ancho - indent, indent + 0.7f \* alto, pincel);  
  
 pincel.setColor(Color.*BLUE*);  
 canvas.drawText("Azul= " + azul, indent + 0.5f \* ancho, indent + 0.8f \* alto, pincel);  
  
 pincel.setTextSize(0.03f \* a);  
 pincel.setColor(Color.*LTGRAY*);  
 canvas.drawText("Copyright 2022 para Diego L. Aristizábal Ramírez", indent,alto-0.3f\*indent, pincel);  
  
  
 *//Para efectos de animación* invalidate();  
  
  
 }  
  
}

La clase **ClientePubSubMQTT**.

public class ClientePubSubMQTT implements MqttCallback, IMqttActionListener {  
  
 private Activity actividad;  
  
  
 private static String *MQTTHOST*;  
 private static String *USERNAME*;  
 private static String *PASSWORD*;  
 private String topicStr;  
  
  
 private MqttAndroidClient client;  
 private MqttConnectOptions options;  
 private IMqttToken token;  
 private JSONObject obj;  
 private String dato;  
 private String datoString;  
  
  
 public ClientePubSubMQTT(Activity actividad) {  
  
 this.actividad = actividad;  
  
 *MQTTHOST* = AlmacenDatosRAM.*MQTTHOST*;  
 *USERNAME* = AlmacenDatosRAM.*USERNAME*;  
 *PASSWORD* = AlmacenDatosRAM.*PASSWORD*;  
 topicStr = AlmacenDatosRAM.*topicStr*;  
  
 }  
  
  
 */\*  
 1. Establecer conexión  
 \*/* public void conectar() {  
  
 String clientId = MqttClient.*generateClientId*();  
 client = new MqttAndroidClient(actividad.getApplicationContext(), *MQTTHOST*, clientId);  
 client.setCallback(this);  
 options = new MqttConnectOptions();  
 options.setUserName(*USERNAME*);  
 options.setPassword(*PASSWORD*.toCharArray());  
  
 *//no estaba  
 // options.setCleanSession(true);  
  
 //hacer conexión* try {  
 token = client.connect(options, null, this);  
 *//Log.d(TAG, "Conectado...");* AlmacenDatosRAM.*conectado\_PubSub* = "Conectado con el broker...";  
 AlmacenDatosRAM.*conectado* = true;  
 } catch (MqttException e) {  
 *//Log.d(TAG, "Falla la conexión...");* AlmacenDatosRAM.*conectado\_PubSub* = "Falla conexión con el broker...";  
 AlmacenDatosRAM.*conectado* = false;  
 e.printStackTrace();  
 }  
  
 }  
  
  
 *//método automático  
 //suscripción del tópico* @Override  
 public void onSuccess(IMqttToken asyncActionToken) {  
  
 *//Log.d(TAG, "onSuccess: ");* try {  
  
 client.subscribe(topicStr, 0);  
 AlmacenDatosRAM.*conectado\_PubSub* = "Se hizo la suscripción al tópico...";  
 AlmacenDatosRAM.*conectado* = true;  
 *//Log.d(TAG, "Se hizo la suscripción al tópico...");* } catch (MqttException e) {  
 *//Log.d(TAG, "Falla la suscripción al tópico...");* AlmacenDatosRAM.*conectado\_PubSub* = "Falla la suscripción al tópico...";  
 AlmacenDatosRAM.*conectado* = false;  
 e.printStackTrace();  
 }  
  
 }  
  
  
 *//método automático* @Override  
 public void onFailure(IMqttToken asyncActionToken, Throwable exception) {  
 Log.*d*(*TAG*, "falla conexión");  
 AlmacenDatosRAM.*conectado\_PubSub* = "Falla conexión con el broker...";  
 AlmacenDatosRAM.*conectado* = false;  
 }  
  
  
 *//método automático* @Override  
 public void connectionLost(Throwable throwable) {  
  
 *//Log.d(TAG, "entro a lost");* }  
  
  
 *//método automático* @Override  
 public void messageArrived(String s, MqttMessage mqttMessage) throws Exception {  
  
 if (AlmacenDatosRAM.*conectado* == true) {  
 datoString = new String(mqttMessage.getPayload());  
 AlmacenDatosRAM.*conectado\_PubSub* = "Recibiendo datos...";  
 }  
  
  
 }  
  
  
 *//método automático* @Override  
 public void deliveryComplete(IMqttDeliveryToken token) {  
 Log.*d*(*TAG*, "deliveryComplete: ");  
 }  
  
  
 */\*  
 2. Recibir mensajes  
 \*/* public String leerString() {  
  
 return datoString;  
  
 }  
  
  
 */\*  
 2. Enviar mensajes  
 \*/* public void setEnviarMensajes(byte[] datoBytesEnviar) {  
  
  
 Log.*d*(*TAG*, "enviando datos: ");  
  
  
 if (AlmacenDatosRAM.*conectado* == true) {  
 AlmacenDatosRAM.*conectado\_PubSub* = "Enviando datos... ";  
 MqttMessage message = new MqttMessage();  
  
 message.setQos(2);*//estaba en 0* message.setRetained(true);*//false  
  
 //message.setQos(0);//estaba en 0  
 //message.setRetained(false);//false* message.setPayload(datoBytesEnviar);  
  
 try {  
 client.publish(topicStr, message);  
 } catch (MqttException e) {  
  
 e.printStackTrace();  
 }  
  
 }  
 }  
  
  
 */\*  
 3. Mantener conexión  
 \*/  
  
 /\*  
 4. Desconectarse  
 \*/* public void desconectar() {  
  
 try {  
 client.unsubscribe(topicStr);  
 options.setCleanSession(true);*//no estaba* IMqttToken token = client.disconnect();  
 } catch (MqttException e) {  
 e.printStackTrace();  
 }  
  
  
 }  
  
}

La clase **ActividadComoClientePubMQTT**.

public class ActividadComoClientePubMQTT extends Activity implements Runnable {  
  
  
 private int tamanoLetraResolucionIncluida;  
  
 *//Objetos GUI necesarios* private TextView textRojo, textVerde, textAzul;  
 private TextView textviewAviso;  
 private Button botonConectar;  
 private SeekBar seekBarRojo, seekBarVerde, seekBarAzul;  
 private TableroColor tablero;  
  
 *//valores de las variables* private int rojo, verde, azul;  
  
 private ClientePubSubMQTT cliente;  
 private Thread hilo;  
  
  
 public void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
  
 gestionarResolucion();  
  
 *//para crear elementos de la GUI* crearElementosGUI();  
  
 */\*  
 Para informar cómo se debe pegar el administrador de  
 diseño LinearLayout obtenido con el método crearGui()  
 \*/* ViewGroup.LayoutParams parametro\_layout\_principal = new ViewGroup.LayoutParams(ViewGroup.LayoutParams.*MATCH\_PARENT*, ViewGroup.LayoutParams.*MATCH\_PARENT*);  
  
  
 *//pegar el contenedor con la GUI* this.setContentView(crearGUI(), parametro\_layout\_principal);  
  
 eventos();  
  
 crearCliente();  
  
 hilo = new Thread(this);  
  
  
 }*//fin del método onCreate* private void gestionarResolucion() {  
  
 *//tamano de letra para usar acomodado a la resolución de pantalla* tamanoLetraResolucionIncluida = (int) (0.8 \* AlmacenDatosRAM.*tamanoLetraResolucionIncluida*);  
  
 }  
  
 */\*método responsable de la creación de los elementos de la GUI\*/* private void crearElementosGUI() {  
  
 botonConectar = new Button(this);  
 botonConectar.setTextSize(TypedValue.*COMPLEX\_UNIT\_SP*, tamanoLetraResolucionIncluida);  
 botonConectar.setText("CONECTAR");  
 botonConectar.getBackground().setColorFilter(Color.*rgb*(183, 216, 199), PorterDuff.Mode.*MULTIPLY*);  
  
 textRojo = new TextView(this);  
 textRojo.setTextSize(TypedValue.*COMPLEX\_UNIT\_SP*, tamanoLetraResolucionIncluida);  
 textRojo.setGravity(Gravity.*CENTER*);  
 textRojo.setBackgroundColor(Color.*BLACK*);  
 String marca\_frecancia\_x = "ROJO\n 0 A 255";*//con salto de línea* textRojo.setText(marca\_frecancia\_x);  
  
 textVerde = new TextView(this);  
 textVerde.setTextSize(TypedValue.*COMPLEX\_UNIT\_SP*, tamanoLetraResolucionIncluida);  
 textVerde.setGravity(Gravity.*CENTER*);  
 textVerde.setBackgroundColor(Color.*BLACK*);  
 String marca\_frecuencia\_y = "VERDE\n 0 A 255";*//con salto de línea* textVerde.setText(marca\_frecuencia\_y);  
  
 textAzul = new TextView(this);  
 textAzul.setTextSize(TypedValue.*COMPLEX\_UNIT\_SP*, tamanoLetraResolucionIncluida);  
 textAzul.setGravity(Gravity.*CENTER*);  
 textAzul.setBackgroundColor(Color.*BLACK*);  
 String marca\_angulo = "AZUL\n 0 A 255";*//con salto de línea* textAzul.setText(marca\_angulo);  
  
 seekBarRojo = new SeekBar(this);  
 seekBarRojo.setMax(255);  
 seekBarRojo.setScaleY(0.2f);  
 seekBarRojo.setProgress(100);  
 rojo = seekBarRojo.getProgress();  
  
 seekBarVerde = new SeekBar(this);  
 seekBarVerde.setMax(255);  
 seekBarVerde.setScaleY(0.2f);  
 seekBarVerde.setProgress(200);  
 verde = seekBarVerde.getProgress();  
  
 seekBarAzul = new SeekBar(this);  
 seekBarAzul.setMax(255);  
 seekBarAzul.setScaleY(0.2f);  
 seekBarAzul.setProgress(60);  
 azul = seekBarAzul.getProgress();  
  
 textviewAviso = new TextView(this);  
 textviewAviso.setGravity(Gravity.*FILL\_VERTICAL*);  
 textviewAviso.setBackgroundColor(Color.*rgb*(183,216,199));  
 textviewAviso.setTextSize(0.8f\*tamanoLetraResolucionIncluida);  
 textviewAviso.setText(" ");  
 textviewAviso.setTextColor(Color.*BLACK*);  
  
 tablero = new TableroColor(this);  
  
 }*//fin crearElemnetosGUI  
  
  
  
  
 /\*método responsable de administrar el diseño de la GUI\*/* private LinearLayout crearGUI() {  
  
 LinearLayout linearPrincipal = new LinearLayout(this);  
 linearPrincipal.setOrientation(LinearLayout.*VERTICAL*);  
 linearPrincipal.setBackgroundColor(Color.*YELLOW*);  
 linearPrincipal.setWeightSum(10.0f);  
  
 *//el linear principal arriba* LinearLayout linearPrincipalArriba = new LinearLayout(this);  
 linearPrincipalArriba.setOrientation(LinearLayout.*HORIZONTAL*);  
 linearPrincipalArriba.setBackgroundColor(Color.*YELLOW*);  
  
 *//el linear principal abajo* LinearLayout linearPrincipalAbajo = new LinearLayout(this);  
 linearPrincipalAbajo.setOrientation(LinearLayout.*HORIZONTAL*);  
 linearPrincipalAbajo.setBackgroundColor(Color.*rgb*(183,216,199));  
 linearPrincipalAbajo.setWeightSum(1.0f);  
  
  
 *//linear secundario izquierda* LinearLayout linear\_izquierda = new LinearLayout(this);  
 linear\_izquierda.setBackgroundColor(Color.*WHITE*);  
 linear\_izquierda.setWeightSum(1.0f);  
  
 *//linear secundario derecha* LinearLayout linear\_derecha = new LinearLayout(this);  
 linear\_derecha.setBackgroundColor(Color.*YELLOW*);  
 linear\_derecha.setOrientation(LinearLayout.*VERTICAL*);  
 linear\_derecha.setWeightSum(6.0f);  
  
 *//pegar estos secundarios al princial* LinearLayout.LayoutParams parametros\_pegado\_izquierdo = new LinearLayout.LayoutParams(0, android.view.ViewGroup.LayoutParams.*MATCH\_PARENT*);  
 *//ocupará el 80% de linear\_principal* parametros\_pegado\_izquierdo.weight = 8.0f;  
 parametros\_pegado\_izquierdo.setMargins(5, 5, 5, 5);  
 linearPrincipalArriba.addView(linear\_izquierda, parametros\_pegado\_izquierdo);  
  
 *//pegar estos secundarios al princial* LinearLayout.LayoutParams parametros\_pegado\_derecho = new LinearLayout.LayoutParams(0, android.view.ViewGroup.LayoutParams.*MATCH\_PARENT*);  
 *//ocupará el 20% de linear\_principal* parametros\_pegado\_derecho.weight = 2.0f;  
 parametros\_pegado\_derecho.setMargins(5, 5, 5, 5);  
 linearPrincipalArriba.addView(linear\_derecha, parametros\_pegado\_derecho);  
  
 *//linear secundario derecha* LinearLayout linear\_1 = new LinearLayout(this);  
 linear\_1.setBackgroundColor(Color.*BLUE*);  
 linear\_1.setOrientation(LinearLayout.*VERTICAL*);  
 linear\_1.setWeightSum(1.0f);  
  
 *//linear secundario derecha* LinearLayout linear\_2 = new LinearLayout(this);  
 linear\_2.setBackgroundColor(Color.*YELLOW*);  
 linear\_2.setOrientation(LinearLayout.*VERTICAL*);  
 linear\_2.setWeightSum(1.0f);  
  
 *//linear secundario derecha* LinearLayout linear\_3 = new LinearLayout(this);  
 linear\_3.setBackgroundColor(Color.*YELLOW*);  
 linear\_3.setOrientation(LinearLayout.*VERTICAL*);  
 linear\_3.setWeightSum(1.0f);  
  
 *//linear secundario derecha* LinearLayout linear\_4 = new LinearLayout(this);  
 linear\_4.setBackgroundColor(Color.*YELLOW*);  
 linear\_4.setOrientation(LinearLayout.*VERTICAL*);  
 linear\_4.setWeightSum(1.0f);  
  
 *//linear secundario derecha* LinearLayout linear\_5 = new LinearLayout(this);  
 linear\_5.setBackgroundColor(Color.*YELLOW*);  
 linear\_5.setOrientation(LinearLayout.*VERTICAL*);  
 linear\_5.setWeightSum(1.0f);  
  
 *//linear secundario derecha* LinearLayout linear\_6 = new LinearLayout(this);  
 linear\_6.setBackgroundColor(Color.*YELLOW*);  
 linear\_6.setOrientation(LinearLayout.*VERTICAL*);  
 linear\_6.setWeightSum(1.0f);  
  
 LinearLayout.LayoutParams parametros\_pegado\_linears = new LinearLayout.LayoutParams(android.view.ViewGroup.LayoutParams.*MATCH\_PARENT*, 0);  
 parametros\_pegado\_linears.weight = 1.0f;  
 parametros\_pegado\_linears.setMargins(5, 5, 5, 5);  
  
 linear\_derecha.addView(linear\_1, parametros\_pegado\_linears);  
 linear\_derecha.addView(linear\_2, parametros\_pegado\_linears);  
 linear\_derecha.addView(linear\_3, parametros\_pegado\_linears);  
 linear\_derecha.addView(linear\_4, parametros\_pegado\_linears);  
 linear\_derecha.addView(linear\_5, parametros\_pegado\_linears);  
 linear\_derecha.addView(linear\_6, parametros\_pegado\_linears);  
  
 LinearLayout.LayoutParams parametros\_pegado\_componentes\_text = new LinearLayout.LayoutParams(android.view.ViewGroup.LayoutParams.*MATCH\_PARENT*, 0);  
 parametros\_pegado\_componentes\_text.weight = 1.0f;  
 parametros\_pegado\_componentes\_text.setMargins(5, 5, 5, 5);  
  
 int indent\_y = (int) (0.4f \* linear\_1.getHeight());  
 LinearLayout.LayoutParams parametros\_pegado\_componentes\_seek = new LinearLayout.LayoutParams(android.view.ViewGroup.LayoutParams.*MATCH\_PARENT*, 0);  
 parametros\_pegado\_componentes\_seek.weight = 1.0f;  
 parametros\_pegado\_componentes\_seek.setMargins(5, (int) indent\_y, 5, (int) indent\_y);  
  
 linear\_1.addView(textRojo, parametros\_pegado\_componentes\_text);  
 linear\_2.addView(seekBarRojo, parametros\_pegado\_componentes\_seek);  
  
 linear\_3.addView(textVerde, parametros\_pegado\_componentes\_text);  
 linear\_4.addView(seekBarVerde, parametros\_pegado\_componentes\_seek);  
  
 linear\_5.addView(textAzul, parametros\_pegado\_componentes\_text);  
 linear\_6.addView(seekBarAzul, parametros\_pegado\_componentes\_seek);  
  
 *//pegar la pizarra al linear izquierdo* linear\_izquierda.setOrientation(LinearLayout.*VERTICAL*);  
 linear\_izquierda.addView(tablero, parametros\_pegado\_componentes\_text);*//pegado igual al text  
  
  
 //pegar liear arriba y abajo* LinearLayout.LayoutParams parametrosPegadoBotonesAbajo = new LinearLayout.LayoutParams(0, android.view.ViewGroup.LayoutParams.*MATCH\_PARENT*);  
 parametrosPegadoBotonesAbajo.weight = 1.0f;  
 linearPrincipalAbajo.addView(botonConectar, parametrosPegadoBotonesAbajo);  
  
 *//linear intermedio que da el estado de la conexxión bluetooth* LinearLayout linearPrincipalIntermedia = new LinearLayout(this);  
 LinearLayout.LayoutParams parametrosPegadoTextView = new LinearLayout.LayoutParams(0, ViewGroup.LayoutParams.*MATCH\_PARENT*);  
 parametrosPegadoTextView.weight = 1.0f;  
 linearPrincipalIntermedia.addView(textviewAviso, parametrosPegadoTextView);  
  
 *//pegar linear arriba, intermedia y abajo* LinearLayout.LayoutParams parametrosPegadoArriba = new LinearLayout.LayoutParams(android.view.ViewGroup.LayoutParams.*MATCH\_PARENT*, 0);  
 parametrosPegadoArriba.weight = 8.6f;  
  
 LinearLayout.LayoutParams parametrosPegadoIntermedia = new LinearLayout.LayoutParams(android.view.ViewGroup.LayoutParams.*MATCH\_PARENT*, 0);  
 parametrosPegadoIntermedia.weight = 0.4f;  
  
 LinearLayout.LayoutParams parametrosPegadoAbajo = new LinearLayout.LayoutParams(android.view.ViewGroup.LayoutParams.*MATCH\_PARENT*, 0);  
 parametrosPegadoAbajo.weight = 1.0f;  
  
 linearPrincipal.addView(linearPrincipalArriba, parametrosPegadoArriba);  
 linearPrincipal.addView(linearPrincipalIntermedia, parametrosPegadoIntermedia);  
 linearPrincipal.addView(linearPrincipalAbajo, parametrosPegadoAbajo);  
  
 return linearPrincipal;  
  
 }*//fin gui* private void eventos() {  
  
  
 *//evento cliente* botonConectar.setOnClickListener(new View.OnClickListener() {  
  
 public void onClick(View v) {  
  
 if (botonConectar.getText() == "CONECTAR") {  
 botonConectar.setText("EMPEZAR");  
 *//conectar cliente* cliente.conectar();  
 AlmacenDatosRAM.*estado\_conexion\_nube*=2;  
 actualizarAviso();  
  
  
 } else {  
  
  
 hilo.start();  
 botonConectar.setEnabled(false);  
  
 }  
  
 }  
 });  
  
  
  
 *//eventos seekbar* seekBarRojo.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {  
 int progressChanged = 0;  
  
 public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {  
  
 progressChanged = progress;  
 rojo = progressChanged;  
 tablero.setColoRojo(rojo);  
  
 }  
  
 public void onStartTrackingTouch(SeekBar seekBar) {  
 *// TODO Auto-generated method stub* }  
  
 public void onStopTrackingTouch(SeekBar seekBar) {  
  
 }  
  
 });  
  
  
 seekBarVerde.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {  
 int progressChanged = 0;  
  
 public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {  
  
 progressChanged = progress;  
 verde = progressChanged;  
 tablero.setColoVerde(verde);  
  
 }  
  
 public void onStartTrackingTouch(SeekBar seekBar) {  
 *// TODO Auto-generated method stub* }  
  
 public void onStopTrackingTouch(SeekBar seekBar) {  
  
 }  
  
 });  
  
  
 seekBarAzul.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {  
 int progressChanged = 0;  
  
 public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {  
  
 progressChanged = progress;  
 azul = (progressChanged);  
 tablero.setColoAzul(azul);  
  
 }  
  
 public void onStartTrackingTouch(SeekBar seekBar) {  
 *// TODO Auto-generated method stub* }  
  
 public void onStopTrackingTouch(SeekBar seekBar) {  
  
 }  
  
 });  
  
 }*//fin eventos* public void crearCliente() {  
  
 cliente = new ClientePubSubMQTT(this);  
  
  
 }  
  
  
 protected void onResume() {  
 super.onResume();  
 AlmacenDatosRAM.*estado\_conexion\_nube*=1;  
 actualizarAviso();  
  
 }  
  
  
  
  
 public boolean onKeyDown(int keyCode, KeyEvent event) {  
 if (keyCode == KeyEvent.*KEYCODE\_BACK*) {  
 *// Esto es lo que hace mi botón al pulsar ir a atrás  
 //alerta();* DialogoSalir dialogo\_salir=new DialogoSalir(this);  
 dialogo\_salir.mostrarPopMenuCoeficientes();  
 return true;  
 }  
 return super.onKeyDown(keyCode, event);  
 }  
  
  
 @RequiresApi(api = Build.VERSION\_CODES.*KITKAT*)  
 public void run() {  
  
 while (true) {  
  
 try {  
 Thread.*sleep*(200);  
  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
  
  
 *//escribir IoT* escribir();  
  
  
 }  
  
  
 }  
  
  
 */\*  
 comunicaciones IoT PUB  
 \*/  
  
  
 //lo cambie por este* private void escribir(){  
  
 *//escribir (enviar) datos al cliente* String dato = getStringJSON();  
 byte[] dato\_en\_byte = dato.getBytes();  
 if (dato\_en\_byte != null) {  
 cliente.setEnviarMensajes(dato\_en\_byte);  
 AlmacenDatosRAM.*estado\_conexion\_nube*=4;  
 actualizarAviso();  
  
 } else {  
  
 AlmacenDatosRAM.*estado\_conexion\_nube*=3;  
 actualizarAviso();  
  
 }  
 }  
  
  
 private String getStringJSON(){  
  
 JSONObject obj = new JSONObject();  
  
 try {  
  
 obj.put("r", new Integer(rojo));  
 obj.put("g", new Integer(verde));  
 obj.put("b", new Integer(azul));  
  
 } catch (JSONException e) {  
 *// TODO Auto-generated catch block* e.printStackTrace();  
 }  
  
 *//convertir a String* return obj.toString();  
  
 }  
  
  
 private void actualizarAviso(){  
  
 *//text\_aviso.setText(" Estado conexión IoT:" + AlmacenDatosRAM.conectado\_PubSub)* if(AlmacenDatosRAM.*estado\_conexion\_nube*==1){  
  
 AlmacenDatosRAM.*conectado\_PubSub*= " Hacer clic en CONECTAR para acceder al BROKER...";  
 String aviso = AlmacenDatosRAM.*conectado\_PubSub*;  
 textviewAviso.setText(aviso);  
  
 }  
  
 if(AlmacenDatosRAM.*estado\_conexion\_nube*==2){  
  
 AlmacenDatosRAM.*conectado\_PubSub*= " Hacer clic en EMPEZAR para publicar datos del BROKER...";  
 String aviso = AlmacenDatosRAM.*conectado\_PubSub*;  
 textviewAviso.setText(aviso);  
  
 }  
  
  
 if(AlmacenDatosRAM.*estado\_conexion\_nube*==3){  
  
 AlmacenDatosRAM.*conectado\_PubSub*= " No se están enviando datos ...";  
 String aviso = AlmacenDatosRAM.*conectado\_PubSub*;  
 textviewAviso.setText(aviso);  
  
 }  
  
 if(AlmacenDatosRAM.*estado\_conexion\_nube*==4){  
  
 AlmacenDatosRAM.*conectado\_PubSub*= " Enviando datos ...";  
 String aviso = AlmacenDatosRAM.*conectado\_PubSub*;  
 textviewAviso.setText(aviso);  
  
 }  
  
 }  
  
  
}

La clase **ActividadConfiguracion**.

public class ActividadConfiguracion extends Activity {  
  
 private EditText edit\_text\_broker, edit\_text\_usuario, edit\_text\_topico,  
 edit\_text\_contrasena;;  
 private TextView text\_broker, text\_usuario, text\_contrasena, text\_topico;  
 private int tamano\_letra;  
 private Button boton\_grabar;  
  
 public void onCreate(Bundle savedInstanceState){  
 super.onCreate(savedInstanceState);  
  
  
 gestionarResolucion();  
  
  
  
 *//para crear elementos de la GUI* crearElementosGUI();  
  
 *//para informar cómo se debe pegar el adminitrador de  
 //diseño obtenido con el método GUI* ViewGroup.LayoutParams parametro\_layout\_principal = new ViewGroup.LayoutParams(ViewGroup.LayoutParams.*MATCH\_PARENT*, ViewGroup.LayoutParams.*MATCH\_PARENT*);  
  
 *//pegar el contenedor con la GUI* this.setContentView(crearGUI(), parametro\_layout\_principal);  
  
 *//Eventos GUI* eventosGui();  
  
 } *//fin del método onCreate* private void gestionarResolucion(){  
  
 tamano\_letra = (int)(0.5f\* AlmacenDatosRAM.*tamanoLetraResolucionIncluida*);  
  
 }  
  
 private void eventosGui() {  
  
 boton\_grabar.setOnClickListener(new View.OnClickListener() {  
  
 @RequiresApi(api = Build.VERSION\_CODES.*KITKAT*)  
 public void onClick(View v) {  
  
 guardar();  
 *//regresar a la actividad anterior* onBackPressed();  
  
 }  
 });  
 }  
  
  
  
 */\*método responsable de la creación de los elementos de la GUI\*/* private void crearElementosGUI(){  
  
 text\_broker = new TextView(this);  
 text\_broker.setGravity(Gravity.*FILL\_VERTICAL*);  
 text\_broker.setBackgroundColor(Color.*YELLOW*);  
 text\_broker.setTextSize(tamano\_letra);  
 text\_broker.setText(" URL DEL BROKER");  
 text\_broker.setTextColor(Color.*BLACK*);  
  
 edit\_text\_broker = new EditText(this);  
 edit\_text\_broker.setTextSize(tamano\_letra);  
 edit\_text\_broker.setText("" + AlmacenDatosRAM.*MQTTHOST*);  
  
 text\_usuario = new TextView(this);  
 text\_usuario.setGravity(Gravity.*FILL\_VERTICAL*);  
 text\_usuario.setBackgroundColor(Color.*YELLOW*);  
 text\_usuario.setTextSize(tamano\_letra);  
 text\_usuario.setText(" USUARIO");  
 text\_usuario.setTextColor(Color.*BLACK*);  
  
 edit\_text\_usuario = new EditText(this);  
 edit\_text\_usuario.setTextSize(tamano\_letra);  
 edit\_text\_usuario.setText("" + AlmacenDatosRAM.*USERNAME*);  
  
 text\_contrasena = new TextView(this);  
 text\_contrasena.setGravity(Gravity.*FILL\_VERTICAL*);  
 text\_contrasena.setBackgroundColor(Color.*YELLOW*);  
 text\_contrasena.setTextSize(tamano\_letra);  
 text\_contrasena.setText(" CONTRASEÑA");  
 text\_contrasena.setTextColor(Color.*BLACK*);  
  
 edit\_text\_contrasena = new EditText(this);  
 *// edit\_text\_contrasena.setInputType(InputType.TYPE\_TEXT\_VARIATION\_PASSWORD);* edit\_text\_contrasena.setInputType(InputType.*TYPE\_CLASS\_TEXT* | InputType.*TYPE\_TEXT\_VARIATION\_PASSWORD* );  
 edit\_text\_contrasena.setTransformationMethod(PasswordTransformationMethod.*getInstance*());  
 edit\_text\_contrasena.setTextSize(tamano\_letra);  
 edit\_text\_contrasena.setText("" + AlmacenDatosRAM.*PASSWORD*);  
  
 text\_topico = new TextView(this);  
 text\_topico.setGravity(Gravity.*FILL\_VERTICAL*);  
 text\_topico.setBackgroundColor(Color.*YELLOW*);  
 text\_topico.setTextSize(tamano\_letra);  
 text\_topico.setText(" TOPICO");  
 text\_topico.setTextColor(Color.*BLACK*);  
  
 edit\_text\_topico = new EditText(this);  
 edit\_text\_topico.setTextSize(tamano\_letra);  
 edit\_text\_topico.setText("" + AlmacenDatosRAM.*topicStr*);  
  
 boton\_grabar = new Button(this);  
 boton\_grabar.setTextSize(TypedValue.*COMPLEX\_UNIT\_SP*, tamano\_letra);  
 boton\_grabar.setText("GUARDAR");  
 boton\_grabar.getBackground().setColorFilter(Color.*rgb*(255, 255, 100), PorterDuff.Mode.*MULTIPLY*);  
  
 }  
  
  
 */\*método responsable de administrar el diseño de la GUI\*/* private LinearLayout crearGUI() {  
  
 *//LinearLayoutPrincipal* LinearLayout linear\_principal = new LinearLayout(this);  
 linear\_principal.setOrientation(LinearLayout.*VERTICAL*);  
 linear\_principal.setBackgroundColor(Color.*WHITE*);  
  
 *//LinearLayout primera fila* LinearLayout linearLayoutPrimeraFila = new LinearLayout(this);  
 linearLayoutPrimeraFila.setBackgroundColor(Color.*YELLOW*);  
 linearLayoutPrimeraFila.setWeightSum(2f);  
 linearLayoutPrimeraFila.setOrientation(LinearLayout.*HORIZONTAL*);  
  
 *//LinearLayout segunda fila* LinearLayout linearLayoutSegundaFila = new LinearLayout(this);  
 linearLayoutSegundaFila.setBackgroundColor(Color.*YELLOW*);  
 linearLayoutSegundaFila.setWeightSum(2f);  
 linearLayoutSegundaFila.setOrientation(LinearLayout.*HORIZONTAL*);  
  
 *//LinearLayout tercera fila* LinearLayout linearLayoutTerceraFila = new LinearLayout(this);  
 linearLayoutTerceraFila.setBackgroundColor(Color.*YELLOW*);  
 linearLayoutTerceraFila.setWeightSum(2f);  
 linearLayoutTerceraFila.setOrientation(LinearLayout.*HORIZONTAL*);  
  
 *//LinearLayout cuarta fila* LinearLayout linearLayoutCuartaFila = new LinearLayout(this);  
 linearLayoutCuartaFila.setBackgroundColor(Color.*YELLOW*);  
 linearLayoutCuartaFila.setWeightSum(2f);  
 linearLayoutCuartaFila.setOrientation(LinearLayout.*HORIZONTAL*);  
  
 *//LinearLayout quinta fila* LinearLayout linearLayoutQuintaFila = new LinearLayout(this);  
 linearLayoutQuintaFila.setBackgroundColor(Color.*YELLOW*);  
 linearLayoutQuintaFila.setWeightSum(2f);  
 linearLayoutQuintaFila.setOrientation(LinearLayout.*HORIZONTAL*);  
  
 *//LinearLayout sexta fila* LinearLayout linearLayoutSextaFila = new LinearLayout(this);  
 linearLayoutSextaFila.setBackgroundColor(Color.*YELLOW*);  
 linearLayoutSextaFila.setWeightSum(1f);  
 linearLayoutSextaFila.setOrientation(LinearLayout.*HORIZONTAL*);  
  
 *//LinearLayout septima fila y pegar panel deslizable* LinearLayout linearLayoutSeptimaFila = new LinearLayout(this);  
 linearLayoutSeptimaFila.setBackgroundColor(Color.*BLACK*);  
 linearLayoutSeptimaFila.setOrientation(LinearLayout.*VERTICAL*);  
  
 *//pegar las filas en el principal* LinearLayout.LayoutParams parametrosPrimerasCincoFilas = new LinearLayout.LayoutParams(ViewGroup.LayoutParams.*MATCH\_PARENT*,0);  
 parametrosPrimerasCincoFilas.weight = 0.5f;  
 linearLayoutPrimeraFila.setLayoutParams(parametrosPrimerasCincoFilas);  
 linearLayoutSegundaFila.setLayoutParams(parametrosPrimerasCincoFilas);  
 linearLayoutTerceraFila.setLayoutParams(parametrosPrimerasCincoFilas);  
 linearLayoutCuartaFila.setLayoutParams(parametrosPrimerasCincoFilas);  
 linearLayoutQuintaFila.setLayoutParams(parametrosPrimerasCincoFilas);  
 linearLayoutSextaFila.setLayoutParams(parametrosPrimerasCincoFilas);  
  
 LinearLayout.LayoutParams parametrosSeptimaFila = new LinearLayout.LayoutParams(ViewGroup.LayoutParams.*MATCH\_PARENT*,0);  
 parametrosSeptimaFila.weight = 7.5f;  
 linearLayoutSeptimaFila.setLayoutParams(parametrosSeptimaFila);  
  
 linear\_principal.addView(linearLayoutPrimeraFila);  
 linear\_principal.addView(linearLayoutSegundaFila);  
 linear\_principal.addView(linearLayoutTerceraFila);  
 linear\_principal.addView(linearLayoutCuartaFila);  
 linear\_principal.addView(linearLayoutQuintaFila);  
 linear\_principal.addView(linearLayoutSextaFila);  
 linear\_principal.addView(linearLayoutSeptimaFila);  
  
 *//pegar elementos en cinco primeras filas* LinearLayout.LayoutParams parametros\_pegado\_elementos\_cinco\_primeras\_filas\_I = new LinearLayout.LayoutParams(0, android.view.ViewGroup.LayoutParams.*MATCH\_PARENT*);  
 parametros\_pegado\_elementos\_cinco\_primeras\_filas\_I .weight = 1.0f;  
 LinearLayout.LayoutParams parametros\_pegado\_elementos\_cinco\_primeras\_filas\_D = new LinearLayout.LayoutParams(0, android.view.ViewGroup.LayoutParams.*MATCH\_PARENT*);  
 parametros\_pegado\_elementos\_cinco\_primeras\_filas\_D .weight = 1.0f;  
 *//primera fila* text\_broker.setLayoutParams(parametros\_pegado\_elementos\_cinco\_primeras\_filas\_I);  
 edit\_text\_broker.setLayoutParams(parametros\_pegado\_elementos\_cinco\_primeras\_filas\_D);  
 linearLayoutPrimeraFila.addView(text\_broker);  
 linearLayoutPrimeraFila.addView(edit\_text\_broker);  
  
 *//segunda fila* text\_usuario.setLayoutParams(parametros\_pegado\_elementos\_cinco\_primeras\_filas\_I);  
 edit\_text\_usuario.setLayoutParams(parametros\_pegado\_elementos\_cinco\_primeras\_filas\_D);  
 linearLayoutSegundaFila.addView(text\_usuario);  
 linearLayoutSegundaFila.addView(edit\_text\_usuario);  
 *//tercera fila* text\_contrasena.setLayoutParams(parametros\_pegado\_elementos\_cinco\_primeras\_filas\_I);  
 edit\_text\_contrasena.setLayoutParams(parametros\_pegado\_elementos\_cinco\_primeras\_filas\_D);  
 linearLayoutTerceraFila.addView(text\_contrasena);  
 linearLayoutTerceraFila.addView(edit\_text\_contrasena);  
  
 *//cuarta fila* text\_topico.setLayoutParams(parametros\_pegado\_elementos\_cinco\_primeras\_filas\_I);  
 edit\_text\_topico.setLayoutParams(parametros\_pegado\_elementos\_cinco\_primeras\_filas\_D);  
 linearLayoutCuartaFila.addView(text\_topico);  
 linearLayoutCuartaFila.addView(edit\_text\_topico);  
  
 *//pegar elemento quinta fila* LinearLayout.LayoutParams parametros\_pegado\_elemento\_quinta\_fila = new LinearLayout.LayoutParams(0, android.view.ViewGroup.LayoutParams.*MATCH\_PARENT*);  
 parametros\_pegado\_elemento\_quinta\_fila .weight = 1.0f;  
 linearLayoutSextaFila.addView(boton\_grabar,parametros\_pegado\_elemento\_quinta\_fila);  
  
  
 return linear\_principal;  
 }  
  
  
 */\*  
 Guardar el estado automáticamente cuando se salga  
 de esta pantalla  
 \*/* protected void onPause() {  
 super.onPause();  
  
 }  
  
 private void guardar(){  
  
 String broker = edit\_text\_broker.getText().toString();  
 AlmacenDatosRAM.*MQTTHOST* = broker;  
  
 String usuario = edit\_text\_usuario.getText().toString();  
 AlmacenDatosRAM.*USERNAME* = usuario;  
  
 String pasword = edit\_text\_contrasena.getText().toString();  
 AlmacenDatosRAM.*PASSWORD* = pasword;  
  
 String topico= edit\_text\_topico.getText().toString();  
 AlmacenDatosRAM.*topicStr* = topico;  
  
  
 SharedPreferences prefs = getSharedPreferences("MisPreferencias", Context.*MODE\_PRIVATE*);  
 SharedPreferences.Editor editor = prefs.edit();  
 editor.putString("broker", AlmacenDatosRAM.*MQTTHOST*);  
 editor.putString("usuario", AlmacenDatosRAM.*USERNAME*);  
 editor.putString("pasword", AlmacenDatosRAM.*PASSWORD*);  
 editor.putString("topico", AlmacenDatosRAM.*topicStr*);  
 editor.commit();  
  
 }  
  
}

La clase **ActividadPrincipalMiCuadragesimaTerceraApp**.

public class ActividadPrincipalMiCuadragesimaTerceraApp extends Activity {  
  
  
 private Boton entrar, salir, ajustes;  
 LinearLayout linear\_layout\_segunda\_fila;  
  
  
 public void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
  
 gestionarResolucion();  
  
 *//para crear elementos de la GUI* crearElementosGUI();  
  
 */\*  
 Para informar cómo se debe pegar el administrador de  
 diseño LinearLayout obtenido con el método crearGui()  
 \*/* ViewGroup.LayoutParams parametro\_layout\_principal = new ViewGroup.LayoutParams(ViewGroup.LayoutParams.*MATCH\_PARENT*, ViewGroup.LayoutParams.*MATCH\_PARENT*);  
  
  
 *//pegar el contenedor con la GUI* this.setContentView(crearGUI(), parametro\_layout\_principal);  
  
 *//actualizar preferencias* actualizarPreferenciasMQTT();  
  
  
 eventos();  
  
  
  
 }*//fin del método onCreate  
  
 /\*Método auxiliar para asuntos de resolución\*/* private void gestionarResolucion() {  
  
 *//independencia de la resolución de la pantalla* DisplayMetrics displayMetrics = this.getApplicationContext().getResources().getDisplayMetrics();  
 int alto = displayMetrics.heightPixels;  
 int ancho = displayMetrics.widthPixels;  
 AlmacenDatosRAM.*ancho*=ancho;  
 AlmacenDatosRAM.*alto*=alto;  
 int dimensionReferencia;  
  
 *//tomar el menor valor entre alto y ancho de pantalla* if (alto > ancho) {  
  
 dimensionReferencia = ancho;  
 } else {  
  
 dimensionReferencia = alto;  
 }  
  
 AlmacenDatosRAM.*dimensionReferencia* =dimensionReferencia;  
  
 *//una estimación de un buen tamaño* int tamanoLetra = dimensionReferencia / 20;  
  
  
 *//tamano de letra para usar acomodado a la resolución de pantalla* int tamanoLetraResolucionIncluida = (int) (tamanoLetra / displayMetrics.scaledDensity);  
  
 *//guardar en el almacen de datos para que otras clases la accedan fácilmente* AlmacenDatosRAM.*tamanoLetraResolucionIncluida* = tamanoLetraResolucionIncluida;  
  
 }  
  
  
 private void crearElementosGUI() {  
  
 entrar = new Boton(this);  
 entrar.setImagen(R.drawable.*entrar*);  
  
 salir = new Boton(this);  
 salir.setImagen(R.drawable.*salir*);  
  
 ajustes = new Boton(this);  
 ajustes.setImagen(R.drawable.*configuracion*);  
  
 }  
  
  
 */\*método responsable de administrar el diseño de la GUI\*/* private LinearLayout crearGUI() {  
  
  
 LinearLayout linear\_layout\_principal = new LinearLayout(this);  
 linear\_layout\_principal.setOrientation(LinearLayout.*VERTICAL*);  
 linear\_layout\_principal.setGravity(Gravity.*CENTER\_HORIZONTAL*);  
 linear\_layout\_principal.setGravity(Gravity.*FILL*);  
 linear\_layout\_principal.setBackgroundColor(Color.*WHITE*);  
 linear\_layout\_principal.setWeightSum(10);  
  
  
 *//LinearLayout primera fila* LinearLayout linear\_layout\_primera\_fila = new LinearLayout(this);  
 linear\_layout\_primera\_fila.setOrientation(LinearLayout.*HORIZONTAL*);  
 linear\_layout\_primera\_fila.setGravity(Gravity.*FILL*);  
 linear\_layout\_primera\_fila.setBackgroundColor(Color.*BLACK*);  
 LinearLayout.LayoutParams parametros\_primera\_fila = new LinearLayout.LayoutParams(ViewGroup.LayoutParams.*MATCH\_PARENT*, 0);*//ViewGroup.LayoutParams.WRAP\_CONTENT);* parametros\_primera\_fila.weight = 9.0f;  
 linear\_layout\_primera\_fila.setLayoutParams(parametros\_primera\_fila);  
  
  
 *//fondo primera fila* Drawable fondo = getResources().getDrawable(R.drawable.*rgb\_android*);  
 linear\_layout\_primera\_fila.setBackgroundDrawable(fondo);  
  
 *//LinearLayout segunda fila* linear\_layout\_segunda\_fila = new LinearLayout(this);  
 linear\_layout\_segunda\_fila.setOrientation(LinearLayout.*HORIZONTAL*);  
 linear\_layout\_segunda\_fila.setGravity(Gravity.*FILL*);  
 LinearLayout.LayoutParams parametros\_segunda\_fila = new LinearLayout.LayoutParams(ViewGroup.LayoutParams.*MATCH\_PARENT*, 0);  
 parametros\_segunda\_fila.weight = 1.0f;  
 linear\_layout\_segunda\_fila.setWeightSum(2.0f);  
 linear\_layout\_segunda\_fila.setLayoutParams(parametros\_segunda\_fila);  
  
  
 LinearLayout.LayoutParams parametros\_pegado\_boton = new LinearLayout.LayoutParams(0, ViewGroup.LayoutParams.*MATCH\_PARENT*);  
 parametros\_pegado\_boton.weight = 1.0f;  
 entrar.setLayoutParams(parametros\_pegado\_boton);  
 salir.setLayoutParams(parametros\_pegado\_boton);  
 ajustes.setLayoutParams(parametros\_pegado\_boton);  
 linear\_layout\_segunda\_fila.addView(entrar);  
 linear\_layout\_segunda\_fila.addView(ajustes);  
 *//linear\_layout\_segunda\_fila.addView(salir);* linear\_layout\_principal.addView(linear\_layout\_primera\_fila);  
 linear\_layout\_principal.addView(linear\_layout\_segunda\_fila);  
  
  
 return linear\_layout\_principal;  
  
  
 }  
  
  
 private void eventos() {  
  
 entrar.setOnClickListener(new View.OnClickListener() {  
  
 public void onClick(View v) {  
  
 lanzarActividadComunicacion();  
 linear\_layout\_segunda\_fila.removeAllViews();  
 linear\_layout\_segunda\_fila.addView(salir);  
 linear\_layout\_segunda\_fila.addView(ajustes);  
  
 }  
 });  
  
 salir.setOnClickListener(new View.OnClickListener() {  
  
 public void onClick(View v) {  
  
 finish();  
 }  
 });  
  
 ajustes.setOnClickListener(new View.OnClickListener() {  
  
 public void onClick(View v) {  
  
 lanzarActividadConfiguracion();  
 }  
 });  
  
  
 }  
  
  
  
  
  
 private void lanzarActividadComunicacion() {  
  
 Intent intent = new Intent(this, ActividadComoClientePubMQTT.class);  
 startActivity(intent);  
  
 }  
  
  
 private void lanzarActividadConfiguracion(){  
  
 Intent intent = new Intent(this, ActividadConfiguracion.class);  
 startActivity(intent);  
  
 }  
  
  
  
 protected void onDestroy() {  
 super.onDestroy();  
 finish();  
  
 }  
  
  
 private void actualizarPreferenciasMQTT(){  
  
 SharedPreferences prefs = getSharedPreferences("MisPreferencias", Context.*MODE\_PRIVATE*);  
  
 AlmacenDatosRAM.*MQTTHOST* = prefs.getString("broker", "");  
 AlmacenDatosRAM.*USERNAME* = prefs.getString("usuario", "");  
 AlmacenDatosRAM.*PASSWORD* = prefs.getString("pasword","");*///no lo tenía* AlmacenDatosRAM.*topicStr* = prefs.getString("topico", "");  
  
 }  
  
  
  
}

El **archivo manifiesto**. Observar que hubo que agregar los permisos de acceso a la red Internet.

*<?*xml version="1.0" encoding="utf-8"*?>*<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 package="com.curso\_simulaciones.micuadragesimaterceraapp">  
  
 <application  
 android:allowBackup="true"  
 android:dataExtractionRules="@xml/data\_extraction\_rules"  
 android:fullBackupContent="@xml/backup\_rules"  
 android:icon="@mipmap/ic\_launcher"  
 android:label="@string/app\_name"  
 android:roundIcon="@mipmap/ic\_launcher\_round"  
 android:supportsRtl="true"  
 android:theme="@style/Theme.MiCuadragesimaTerceraApp"  
 tools:targetApi="31">  
  
 <activity android:name=".ActividadPrincipalMiCuadragesimaTerceraApp"  
 android:theme="@android:style/Theme.NoTitleBar.Fullscreen"  
 android:screenOrientation="portrait"  
 android:exported="true">  
  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN" />  
 <category android:name="android.intent.category.LAUNCHER" />  
 </intent-filter>  
  
  
 </activity>  
  
  
  
 *<!--declarando otra actividad-->* <activity android:name=".actividades\_secundarias.ActividadComoClientePubMQTT"  
 android:theme="@android:style/Theme.NoTitleBar.Fullscreen"  
 android:screenOrientation="portrait">  
 </activity>  
  
  
 *<!--declarando otra actividad-->* <activity android:name=".actividades\_secundarias.ActividadConfiguracion"  
 android:theme="@android:style/Theme.NoTitleBar.Fullscreen"  
 android:screenOrientation="portrait">  
 </activity>  
  
  
 <service android:name="org.eclipse.paho.android.service.MqttService"/>  
  
 </application>  
  
 <uses-permission android:name="android.permission.WAKE\_LOCK" />  
 <uses-permission android:name="android.permission.INTERNET" />  
 <uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE" />  
  
  
</manifest>

**Nota importante**:

Recordar del módulo anterior que en el semestre anterior 2022.2 se encontró que las librerías MQTT usadas en este curso no están actualizadas para dispositivos móviles Android 12 o mayores y por lo tanto la aplicación fallaba en éstos. Este impase se logró salvar cambiando en el archivo **build.gradle (Module:app)** la línea de código resaltada con amarillo (se cambió a valor 30, es decir una api anterior). Se debe **sincronizar**:

plugins **{** id 'com.android.application'  
**}**android **{** namespace 'com.curso\_simulaciones.micuadragesimaterceraapp'  
 compileSdk 33  
  
 defaultConfig **{** applicationId "com.curso\_simulaciones.micuadragesimaterceraapp"  
 minSdk 21  
 targetSdk 30  
 versionCode 1  
 versionName "1.0"  
  
 testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"  
 **}**

**El programa para ESP32**

Conectar la **ESP32** al computador y programarla desde el **IDE** de **ARDUINO** con el siguiente código (**led\_rgb\_mqtt\_esp32.ino**).

//para comunicación WiFi

#include <WiFi.h>

//para el protocolo MQTT de IoT

#include <PubSubClient.h>

//para manejar JSON

#include <ArduinoJson.hpp>

#include <ArduinoJson.h>

StaticJsonDocument<300> doc;//300 bytes

int pinRed=23; //GPIO 19

int pinGreen= 22; //GPIO 21

int pinBlue= 21; //GPIO 22

int frecuencia = 5000;

int canal\_1 = 0;

int canal\_2 = 1;

int canal\_3 = 2;

int resolucion = 8;

int r,g,b;

//Variables para conexiones WiFi

const char\* ssid = "xxxxx";//reemplazar SSID;

const char\* password = "xxxxx";//reemplazar pasword;

//URL del Broker MQTT de ESCUELA DE FÍSICA UNALMED

const char\* mqtt\_server = "168.176.136.61";

const int mqttPort = 1883;

const char\* mqttUser = " ";//reemplazar usuario

const char\* mqttPassword = " ";//reemplazar pasword

const char\* topico =" ";//reemplazar topico

WiFiClient espCliente;

PubSubClient mqttCliente(espCliente);

void setup() {

Serial.begin(115200);

configurarPines();

//conectar a WiFi

conectarToWiFi();

//configurar MQTT

setupMQTT();

}

//administrar conexión wiFi

void conectarToWiFi() {

delay(10);

//Comenzar conexión a red WiFI

Serial.println();

Serial.print("Conectando a...");

Serial.println(ssid);

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println("");

Serial.println("WiFi conectado");

Serial.println("IP address: ");

Serial.println(WiFi.localIP());

}

//administrar configuración de conexión al Broker MQTT

void setupMQTT() {

mqttCliente.setServer(mqtt\_server, mqttPort);

//establecer la función de devolución de llamada

mqttCliente.setCallback(callback);

}

//conectar el cliente ESP32 MQTT al Broker

void reconnect() {

Serial.println("Conectando a Broker MQTT...");

//loop hasta lograr conexión

while (!mqttCliente.connected()) {

Serial.println("Reconectando al Broker MQTT..");

String clientId = "ESP32Client-";

clientId += String(random(0xffff), HEX);

if (mqttCliente.connect(clientId.c\_str())) {

Serial.println("Conectado");

//suscripción al tópico: el que determinen

mqttCliente.subscribe(topico);

}

}

}

/\*

Ahora se especifica una función de devolución de llamada.

Primero se imprime el nombre del tema y luego se

recibe el mensaje.

\*/

void callback(char\* topic, byte\* payload, unsigned int length) {

Serial.print("Callback - ");

Serial.print("Message:");

//se recibe el mensaje

for (int i = 0; i < length; i++) {

(char)payload[i];

}

DeserializeObject(payload);

}

void loop() {

if (!mqttCliente.connected())

reconnect();

mqttCliente.loop();

establecerColor(r, g, b);

} //fin loop

void configurarPines(){

//configuración de los canales

//rojo

ledcSetup(canal\_1, frecuencia, resolucion);

ledcAttachPin(pinRed,canal\_1);

//azul

ledcSetup(canal\_2, frecuencia, resolucion);

ledcAttachPin(pinGreen,canal\_2);

//verde

ledcSetup(canal\_3, frecuencia, resolucion);

ledcAttachPin(pinBlue,canal\_3);

}

void establecerColor(int R, int G, int B) {

ledcWrite(canal\_1,R);

ledcWrite(canal\_2,G);

ledcWrite(canal\_3,B);

}

void DeserializeObject(byte\* payload){

//void DeserializeObject(String mensajeRGB){

//char buffer[200];

deserializeJson(doc, payload);

r = doc["r"];

g = doc["g"];

b = doc["b"];

}

**El montaje ESP32**

Desconectar la ESP32 del computador y proceder a realizar el montaje ilustrado en la Figura 4.

Imagen que contiene electrónica, circuito

Descripción generada automáticamente

Figura 4

**Ejecutar**

Ejecutar la app. Se despliega la GUI de la Figura 5 izquierda. Hacer clic en el botón AJUSTE y se despliega la GUI de la Figura 6 derecha. En esta GUI se configura la app: URL del BROKER (escribir el URL del BROKER de la Escuela de Física: tcp://168.176.136.61:1883), USUARIO (cualquiera), CONTRASEÑA (no es necesario en esta app), TOPICO (cualquiera: **cada grupo pone su tópico**). Todos los dispositivos móviles que quieran recibir los datos deben tener el mismo tópico.

 Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

Figura 5

Una vez configurada la app (esta configuración será persistente) regresar y hacer clic en el botón ENTRAR. Se despliega la GUI de la Figura 6 arriba izquierda. Comenzar la comunicación haciendo clic en el botón EMPEZAR.

Forma, Cuadrado

Descripción generada automáticamente Forma, Cuadrado

Descripción generada automáticamente

Figura 6

Proceder a cambiar el color en el tablero y observar el comportamiento del led RGB, Figura 17.

**APP CON ESP32 IoT: ENVIANDO DATOS**

En este segundo ejemplo, se tiene un módulo **GY-30 BH1750**, sensor de iluminación digital para medición de **iluminancia** (**lux**), Figura 8 izquierda, acoplado a la **ESP32**, Figura 7 derecha. Un **dispositivo móvil Android** recibe los datos de ilumniación que son enviados por la **ESP32** **vía IoT**. Aquí la **ESP32** **PUBLICA**.

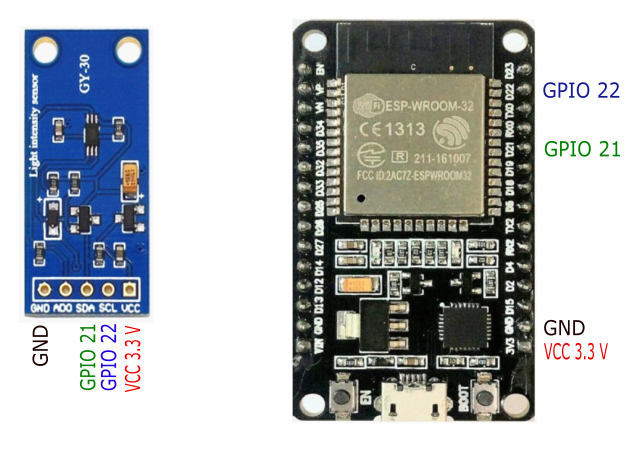


Figura 7

**La app**

Crear un proyecto denominado **MiCuadragesimaCuartaApp** con la jerarquía de paquetes y clases de la Figura 8.

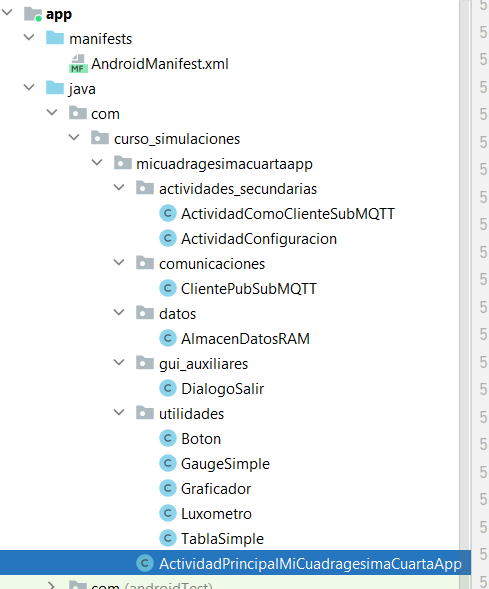


Figura 8

Para implementar el IoT se usará la librería **Eclipse Paho** tal como se hizo en la app anterior. Proceder tal cual se hizo en esa app.

Adicionalmente se usa para graficar la librería **MPAndroidChart**. Proceder a obtenerla tal y como se indicó en módulos anteriores.

Las siguientes imágenes se deben incorporar al proyecto, Figura 9.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Figura 9

La clase **Boton**, **DialogoSalir** y **ClientePubSubMQTT** tienen el mismo código de la app anterior.

El código de las otras clases es el siguiente.

La clase **AlmacenDatosRAM**.

public class AlmacenDatosRAM {  
  
  
 public static int *ancho*, *alto*, *dimensionReferencia*, *tamanoLetraResolucionIncluida*;  
  
 public static int *nDatos* = 50;  
  
 public static int *nDatosGraficar*= 20;  
  
 public static int *estado\_conexion\_nube*=1;  
  
 public static String *MQTTHOST*;  
 public static String *USERNAME*;  
 public static String *PASSWORD*;  
 public static String *topicStr*;  
  
 public static String *conectado\_PubSub*="Hacer clic en CONECTAR para acceder al BROKER...";  
  
 public static boolean *conectado*=false;  
  
 *//public static int periodo\_muestreo= 1000; // 1 segundo* public static String *unidades* = "unidades";  
  
 public static String *tiempo*;  
  
 public static double[] *x* = new double[1000];  
  
 public static double[] *y* = new double[1000];  
  
 */\*  
 ArrayList es una clase que permite almacenar  
 objetos con la diferencia respecto a los  
 arreglos [], que ellla misma va cambiando  
 dinámicamente su tamaño a medida que se le  
 agregan elementos  
 \*/* public static ArrayList *datos* = new ArrayList<>();  
}

La clase **GaugeSimple**.

public class GaugeSimple extends View {  
  
 private float largo;  
 private float minimo = 0;  
 private float maximo = 100f;  
 private float medida = 0.0f;*//tomar como medida inicial* private String unidades = "UNIDADES";  
  
 *//color de los sectores* private int colorPrimerTercio = Color.*rgb*(200, 200, 0);  
 private int colorSegundoTercio = Color.*rgb*(0, 180, 0);  
 private int colorTercerTercio = Color.*RED*;  
  
 *//color del marco* private int colorFondoTacometro = Color.*rgb*(240,240,240);  
 private int colorBordeTacometro= Color.*BLACK*;  
  
 *//color franja dinámica* private int colorFranjaDinamica = Color.*RED*;  
  
  
 private int angPrimertercio = 100;  
 private int angSegundoTercio = 100;  
 private int angTercerTercio = 40;  
  
 private int colorLineas = Color.*BLACK*;  
 private int colorNumeros =Color.*BLACK*;  
  
 private int colorNumerosDesplieggue = Color.*BLACK*;  
  
 private int numeroDivisiones= 25;  
 private int separacionDivisionesGrandes = 5;  
  
  
 */\*\*  
 \* Constructor de GaugeSimple  
 \*/* public GaugeSimple(Context context) {  
  
 super(context);  
  
 if (android.os.Build.VERSION.*SDK\_INT* >= android.os.Build.VERSION\_CODES.*HONEYCOMB*) {  
 this.setLayerType(View.*LAYER\_TYPE\_SOFTWARE*, null);  
  
 }  
 }  
  
 */\*\*  
 \* Modifica el rango de medicion  
 \* desde minimo hasta maximo  
 \*  
 \* @param minimo  
 \* @param maximo  
 \*/* public void setRango(float minimo, float maximo) {  
  
 this.minimo = minimo;  
 this.maximo = maximo;  
  
 }  
  
 public void setSeparacionesDivisionesGrandes(int separacionDivisionesGrandes){  
  
 this.separacionDivisionesGrandes = separacionDivisionesGrandes;  
  
  
 }  
  
  
 private void setNumeroDivisiones(int numeroDivisiones){  
  
 this.numeroDivisiones =numeroDivisiones;  
  
  
 }  
  
  
 */\*\*  
 \* Modifica el valor medido  
 \*  
 \* @param medida  
 \*/* public void setMedida(float medida) {  
  
 this.medida = medida;  
  
 }  
  
  
 */\*\*  
 \* Regresa el valor medido  
 \*  
 \* @return medida  
 \*/* public float getMedida() {  
  
 return medida;  
 }  
  
  
 */\*\*  
 \* Modifica las unidades del instrumento virtual  
 \*  
 \* @param unidades  
 \*/* public void setUnidades(String unidades) {  
  
 this.unidades = unidades;  
  
 }  
  
 */\*\*  
 \* Modifica el color del borde del marco  
 \* @param colorBordeMarco  
 \*/* public void setColorBordeTacometro(int colorBordeMarco){  
  
 this.colorBordeTacometro = colorBordeMarco;  
  
 }  
  
  
 */\*\*  
 \* Modifica los colores de los sectores circulares  
 \*  
 \* @param colorPrimerTercio  
 \* @param colorSegundoTercio  
 \* @param colorTercerTercio  
 \*/* public void setColorSectores(int colorPrimerTercio, int colorSegundoTercio, int colorTercerTercio) {  
  
 this.colorPrimerTercio = colorPrimerTercio;  
 this.colorSegundoTercio = colorSegundoTercio;  
 this.colorTercerTercio = colorTercerTercio;  
  
 }  
  
 */\*\*  
 \* Modifica los angulos de los sectores circulares  
 \* Deben sumar 250 grados  
 \*  
 \* @param angPrimerTercio  
 \* @param angSegundoTercio  
 \* @param angTercerTercio  
 \*/* public void setAngulosSectores(int angPrimerTercio, int angSegundoTercio, int angTercerTercio) {  
 this.angPrimertercio = angPrimerTercio;  
 this.angSegundoTercio = angSegundoTercio;  
 this.angTercerTercio = angTercerTercio;  
  
 }  
  
  
  
 */\*\*  
 \* Modifica el color de fondo del tacometro  
 \*  
 \* @param colorFondoTacometro  
 \*/* public void setColorFondoTacometro(int colorFondoTacometro) {  
  
 this.colorFondoTacometro = colorFondoTacometro;  
  
  
 }  
  
  
 */\*\*  
 \* Modifica el color de las lineas del tacometro  
 \*  
 \* @param color\_lineas  
 \*/* public void setColorLineasTacometro(int color\_lineas) {  
  
 this.colorLineas = color\_lineas;  
  
  
 }  
  
  
 public void setColorNumeros(int colorNumeros){  
  
 this.colorNumeros = colorNumeros;  
  
 }  
  
  
 */\*\*  
 \* Modifica el color del numero que se despliega  
 \*  
 \* @param colorNumerosDesplieggue  
 \*/* public void setColorNumeroDespliegue(int colorNumerosDesplieggue) {  
  
 this.colorNumerosDesplieggue = colorNumerosDesplieggue;  
  
 }  
  
  
 */\*\*  
 \* Modifica el color de la franja dinámica  
 \* @param colorFranjaDinamica  
 \*/* public void setColorFranjaDinámica(int colorFranjaDinamica) {  
  
 this.colorFranjaDinamica = colorFranjaDinamica;  
  
  
 }  
  
  
 */\*\*  
 \* @param canvas  
 \*/  
  
 //método para dibujar* protected void onDraw(Canvas canvas) {  
 super.onDraw(canvas);  
  
 */\*  
 se graba el estado actual del canvas  
 para al final restaurarlo  
 \*/* canvas.save();  
  
  
 */\*  
 La vista tendra las mismas dimensiones de su  
 contenedor  
 \*/* float ancho = this.getWidth();*//ancho de la vista* float alto = this.getHeight();*//alto de la vista  
  
 /\*  
 Se define la variable largo como el 80%  
 del menor valor entre alto y largo del  
 contenedor  
 \*/* if (ancho > alto) {  
  
 largo = 0.8f \* alto;  
  
 } else {  
  
 largo = 0.8f \* ancho;  
  
  
 }  
  
 */\*  
 se hace tralación del (0,0) al centro  
 del contenedor  
 \*/* canvas.translate(0.5f \* ancho, 0.5f \* alto);  
  
 *//configurando el pincel* Paint pincel = new Paint();  
 *//evita efecto sierra* pincel.setAntiAlias(true);  
 *//tamaño texto* pincel.setTextSize(0.05f \* largo);  
 *//para mejor manejo de la métrica de texto* pincel.setLinearText(true);  
 *//para efectos de buen escalado de bitmaps* pincel.setFilterBitmap(true);  
 *//para buen manejo de gradientes de color* pincel.setDither(true);  
  
  
  
 *//dibujar fondo del tacómetro  
 //marco borde: circulo no relleno* pincel.setStyle(Paint.Style.*STROKE*);  
 *//grosor y color* pincel.setStrokeWidth(0.02f\*largo);  
 pincel.setColor(colorBordeTacometro);  
 canvas.drawCircle(0, 0, 0.5f \* largo, pincel);  
 *//marco fondo: círculo relleno* pincel.setStyle(Paint.Style.*FILL*);  
 pincel.setColor(colorFondoTacometro);  
 canvas.drawCircle(0, 0, 0.48f \* largo, pincel);  
  
  
  
 *//dibujar los tres segementos circulares* float esquinaSuperiorIzquierdaX = -0.45f \* largo;  
 float esquinaSuperiorIzquierdaY = -0.45f \* largo;  
 float esquinaInferiorDerechaX = 0.45f \* largo;  
 float esquinaInferiorDerechaY = 0.45f \* largo;  
  
 RectF rect = new RectF(esquinaSuperiorIzquierdaX, esquinaSuperiorIzquierdaY,  
 esquinaInferiorDerechaX, esquinaInferiorDerechaY);  
  
 *//grosor líneas* pincel.setStrokeWidth(0.02f\*largo);  
 *//arcos* pincel.setStyle(Paint.Style.*STROKE*);  
 pincel.setColor(colorPrimerTercio);  
 *//con argumento false solo dibuja el arco y no el sector circular* canvas.drawArc(rect, 150, angPrimertercio, false, pincel);  
 pincel.setColor(colorSegundoTercio);  
 canvas.drawArc(rect, 150 + angPrimertercio, angSegundoTercio, false, pincel);  
 pincel.setColor(colorTercerTercio);  
 canvas.drawArc(rect, 150 + angPrimertercio + angSegundoTercio, angTercerTercio, false, pincel);  
  
  
 *//dibujar la escala* float indent = (float) (0.05 \* largo);  
 float posicionY = (float) (0.5 \* largo);  
  
  
  
 */\*  
 Divisiones grandes, pequeñas y números  
 Se dibuja primero la división vertical.  
 Luego se repite rotando de a 50 grados comenzando  
 en 235 grados.  
 \*/* pincel.setStyle(Paint.Style.*FILL*);  
  
  
 for (int i = 0; i < numeroDivisiones +1; i = i + 1) {*//6  
 // float anguloRotacion = 235 + 50 \* i;* float salto = 240f/numeroDivisiones;  
 *//float anguloRotacion = 235 + 50 \* i;* float anguloRotacion = 240 + salto \* i;  
 canvas.save();  
 canvas.rotate(anguloRotacion, 0, 0);  
 pincel.setColor(colorLineas);  
  
 if(i%separacionDivisionesGrandes ==0) {  
 *//dibujar líneas grandes* pincel.setStrokeWidth(0.01f \* largo);  
 canvas.drawLine(0, -posicionY, 0, -posicionY + indent, pincel);  
  
 *//dibujar los números* float valorIncrementoMarcas = (maximo - minimo) /numeroDivisiones;  
 int valorMarca = (int) (minimo + valorIncrementoMarcas \* i);  
 String numero = "" + valorMarca;  
  
 *//ancho de la cadena del número* float anchoCadenaNumero = pincel.measureText(numero);  
  
 *//dibuja números rotados  
 //endereza los números a orientación horizontal* canvas.rotate(-anguloRotacion, 0, -posicionY + 2.5f \* indent);  
 pincel.setColor(colorNumeros);  
 canvas.drawText(numero, -0.5f \* anchoCadenaNumero, -posicionY + 2.5f \* indent, pincel);  
 } else {  
 *//divisiones pequeñas* pincel.setStrokeWidth(0.005f \* largo);  
 canvas.drawLine(0, -posicionY, 0, -posicionY + (float) (0.6 \* indent), pincel);  
  
 }  
  
 canvas.restore();  
  
 }  
  
 */\*  
 dibujar la aguja  
 \*/  
 //aqui empieza dibujo de la aguja  
 //calcular angulo para ubicar la aguja de acuerdo al valor medido* float angulo\_rotacion\_medida = 240 + (240f / (maximo - minimo)) \* (medida - minimo);  
 *//Dibujar aguja* pincel.setStrokeWidth(0.005f \* largo);  
 pincel.setColor(Color.*RED*);  
 canvas.rotate(angulo\_rotacion\_medida, 0, 0);  
 float b = (float) (1.5f\* indent);  
 canvas.drawLine(0, -posicionY, 0, b, pincel);  
 canvas.rotate(-angulo\_rotacion\_medida, 0, 0);  
 pincel.setStyle(Paint.Style.*FILL*);  
 pincel.setColor(colorFondoTacometro);  
 canvas.drawCircle(0, 0, (float) (0.4 \* indent), pincel);  
 pincel.setColor(Color.*RED*);  
 pincel.setStyle(Paint.Style.*STROKE*);  
 canvas.drawCircle(0, 0, (float) (0.4 \* indent), pincel);  
 *//aquí termina dibujo de la aguja  
  
  
  
  
 //franja dinámica* float a = (float) 0.03\* largo;  
 rect = new RectF(esquinaSuperiorIzquierdaX - a, esquinaSuperiorIzquierdaY - a,  
 esquinaInferiorDerechaX + a, esquinaInferiorDerechaY + a);  
 pincel.setColor(colorFranjaDinamica);  
 pincel.setStrokeWidth(0.01f\*largo);  
 canvas.drawArc(rect, 150, angulo\_rotacion\_medida - 240, false, pincel);  
  
  
 *//Dibujar las unidades* pincel.setStyle(Paint.Style.*FILL*);  
 pincel.setColor(colorLineas);  
 pincel.setTextSize(0.08f \* largo);  
 float anchoCadenaUnidades = pincel.measureText(unidades);  
 canvas.drawText(unidades, -0.5f \* anchoCadenaUnidades,- 0.15f \* largo, pincel);  
 *//aqui termina dibujo de las unidades  
  
 //aqui despliegue de la medida* pincel.setTextSize(0.1f \* largo);  
 float anchoCadenaNumero = pincel.measureText("" + medida);  
 pincel.setColor(colorNumerosDesplieggue);  
 canvas.drawText("" + medida, -0.5f \* anchoCadenaNumero, 0.2f \* largo, pincel);  
  
 *//marcar empresa* String empresa = "IoT.PhysicsSensor";  
 pincel.setTextSize(0.05f \* largo);  
 float anchoCadenaNombreEmpresa = pincel.measureText(empresa);  
 canvas.drawText(empresa, -0.5f \* anchoCadenaNombreEmpresa, 0.35f \* largo, pincel);  
  
  
  
  
  
  
  
  
 *//se restaura el canvas al estado incial  
 //el que se garbó al principio de este método* canvas.restore();  
  
 *//para efectos de animación* invalidate();  
  
 }*//fin onDraw*}

La clase **Luxometro**.

public class Luxometro extends GaugeSimple {  
  
  
  
 public Luxometro(Context context){  
 super(context);  
  
  
 }  
  
  
 public void cambiarEscala(float medida){  
  
 float maximo=100f;  
 float minimo=0f;  
  
 if(medida>0 && ((medida < 100)^(medida==100f ))){  
  
 maximo=100f;  
 minimo=0f;  
  
 }  
  
 if(medida>100 && ((medida < 1000)^(medida==500f ))){  
  
 maximo=500f;  
 minimo=0f;  
  
 }  
  
 if(medida>500 && ((medida < 1000)^(medida==1000f ))){  
  
 maximo=1000f;  
 minimo=0f;  
  
 }  
  
  
 if(medida>1000 && ((medida < 5000)^(medida==5000f ))){  
  
 maximo=5000f;  
 minimo=0f;  
  
 }  
  
 if(medida>5000 && ((medida < 10000)^(medida==10000f ))){  
  
 maximo=10000f;  
 minimo=0f;  
  
 }  
  
 if(medida>10000 && ((medida < 50000)^(medida==50000f ))){  
  
 maximo=50000f;  
 minimo=0f;  
  
 }  
  
  
 this.setRango(minimo,maximo);  
  
 }  
  
  
}

La clase **Graficador**.

public class Graficador extends LineChart {  
  
  
 private LineDataSet lineDataSet;  
 private String tituloEjeY="";  
 private int colorLinea;  
 private float grosorLinea=1f;  
 private int colorValores;  
 private int colorMarcadores;  
  
  
  
 public Graficador(Context context) {  
  
 super(context);  
  
 if (android.os.Build.VERSION.*SDK\_INT* >= android.os.Build.VERSION\_CODES.*HONEYCOMB*) {  
 this.setLayerType(View.*LAYER\_TYPE\_SOFTWARE*, null);  
  
 }  
  
  
 }  
  
  
 public void setDatos(ArrayList datos) {  
  
 LineDataSet lineDataSet = new LineDataSet(datos, tituloEjeY);  
 lineDataSet.setLineWidth(grosorLinea);  
 lineDataSet.setColor(colorLinea);  
 lineDataSet.setValueTextColor(colorValores);  
 lineDataSet.setCircleColor(colorMarcadores);  
 LineData lineData = new LineData(lineDataSet);  
 this.setData(lineData);  
  
 */\*  
 Esta instrucción es de la librería y es necesaria para  
 que el eje se vaya desplazando a medida que entran datos  
 \*/* this.moveViewTo(lineData.getEntryCount() - 7, 50f, YAxis.AxisDependency.*LEFT*);  
  
  
 }  
  
  
 */\*  
 Modifica el grosr del trazo de la gráfica  
 \*/* public void setGrosorLinea(float grosorLinea){  
  
 this.grosorLinea=grosorLinea;  
 }  
  
 public void setColorLinea(int colorLinea){  
  
 this.colorLinea=colorLinea;  
  
 }  
  
  
 public void setColorFondo(int colorFondo){  
  
 this.setBackgroundColor(colorFondo);  
  
 }  
  
  
 public void setColorTextoEjes(int colorTextoEjes){  
  
 *//texto eje y izquierda* this.getAxisLeft().setTextColor(colorTextoEjes);  
 *//texto eje y derecha* this.getAxisRight().setTextColor(colorTextoEjes);  
 *//texto eje x* this.getXAxis().setTextColor(colorTextoEjes);  
 *//texto título eje y* this.getLegend().setTextColor(colorTextoEjes);  
 *//texto título eje x* this.getDescription().setTextColor(colorTextoEjes);  
  
 }  
  
 public void setColorValores(int colorValores){  
  
 this.colorValores=colorValores;  
  
  
 }  
  
  
 public void setTituloEjeX(String tituloEjeX){  
  
  
 this.getDescription().setText(tituloEjeX);  
  
 }  
  
  
 public void setTituloEjeY(String tituloEjeY){  
  
 this.tituloEjeY=tituloEjeY;  
  
 }  
  
  
 public void setColorMarcadores(int colorMarcadores){  
  
 this.colorMarcadores=colorMarcadores;  
 }  
  
  
}

La clase **TablaSimple**.

public class TablaSimple extends LinearLayout {  
  
 private ScrollView panelScroll;  
 private TableLayout table;  
 private int tamanoLetraResolucionIncluida;  
 private int dimensionReferencia;  
 private Context context;  
 private int contador = -1;  
 private String etiquetaX="x";  
 private String etiquetaY="y";  
 private float x,y;  
 private int colorColumna1= Color.*YELLOW*;  
 private int colorColumna2=Color.*BLUE*;  
 private int colorColumna3=Color.*RED*;  
 *// private int n=0;  
  
  
 /\*\*  
 \* Cosntructor de TablaSimple  
 \* @param context  
 \*/* public TablaSimple(Context context) {  
 super(context);  
 this.context=context;  
  
 gestionarResolucion();  
  
  
  
 this.setLayoutParams(new LayoutParams(LayoutParams.*MATCH\_PARENT*, LayoutParams.*MATCH\_PARENT*));  
  
 gui();  
  
 }  
  
  
 private void gestionarResolucion(){  
  
 */\*  
 El alto en la actividad principal (PORTRAI)  
 corresponde al ancho aquí (LANSCAPE  
 \*/* dimensionReferencia = (int)(0.4f\* AlmacenDatosRAM.*alto*);  
  
 tamanoLetraResolucionIncluida = (int)(0.6\* AlmacenDatosRAM.*tamanoLetraResolucionIncluida*);  
  
  
 }*//fin método gestionarResolucion()* private void gui(){  
  
 panelScroll = new ScrollView(context);  
 table=new TableLayout(context);  
  
 LinearLayout linearLayoutPrincipal = new LinearLayout(context);  
 linearLayoutPrincipal.setOrientation(LinearLayout.*VERTICAL*);  
 linearLayoutPrincipal.setBackgroundColor(Color.*BLACK*);  
  
 LinearLayout.LayoutParams parametroPegado = new LinearLayout.LayoutParams(LayoutParams.*MATCH\_PARENT*,LayoutParams.*MATCH\_PARENT*);  
  
 panelScroll.addView(table);  
 linearLayoutPrincipal.addView(panelScroll);  
  
 this.addView(linearLayoutPrincipal, parametroPegado);  
  
  
 }  
  
 */\*\*  
 \* Modifica las etiquetas de las columnas  
 \* @param etiquetaX  
 \* @param etiquetaY  
 \*/* public void setEtiquetaColumnas(String etiquetaX, String etiquetaY){  
  
 this.etiquetaX=etiquetaX;  
 this.etiquetaY=etiquetaY;  
  
 }  
  
 */\*\*  
 \* Envía los datos a la tabla  
 \* @param x  
 \* @param y  
 \*/* public void enviarDatos(float x, float y){  
  
 this.x=x;  
 this.y=y;  
 contador=contador+1;  
 incrementarFila();  
  
 }  
  
  
 */\*\*  
 \* Borra los datos enviados a la tabla  
 \*/* public void borrar(){  
 removerFilas();  
 }  
  
 */\*\*  
 \* Modifica los colores de las columnas  
 \* @param colorColumna1  
 \* @param colorColumna2  
 \* @param colorColumna3  
 \*/* public void setColorColumnas(int colorColumna1, int colorColumna2,int colorColumna3){  
  
 this.colorColumna1=colorColumna1;  
 this.colorColumna2=colorColumna2;  
 this.colorColumna3=colorColumna3;  
  
 }  
  
  
 private void incrementarFila() {  
  
 *//crear nueva TableRow* TableRow fila = new TableRow(context);  
  
 TableRow.LayoutParams layoutTexto = new TableRow.LayoutParams((int)(0.33\*dimensionReferencia), TableRow.LayoutParams.*WRAP\_CONTENT*);  
  
  
 *//columna uno* TextView textNumeroDato = new TextView(context);  
 textNumeroDato.setTextSize(TypedValue.*COMPLEX\_UNIT\_SP*, tamanoLetraResolucionIncluida);  
 textNumeroDato.setLayoutParams(layoutTexto);  
  
 if(contador==0){  
  
 textNumeroDato.setText("# de DATO");  
  
 } else {  
  
 textNumeroDato.setText("DATO " + contador);  
 }  
  
 *//columna 2* textNumeroDato.setTextSize(tamanoLetraResolucionIncluida);  
 textNumeroDato.setTextColor(colorColumna1);  
 textNumeroDato.setGravity(Gravity.*CENTER\_HORIZONTAL*);  
  
 TextView textValorX = new TextView(context);  
 textValorX.setTextSize(TypedValue.*COMPLEX\_UNIT\_SP*, tamanoLetraResolucionIncluida);  
 textValorX.setLayoutParams(layoutTexto);  
  
 if(contador==0)  
 {  
 textValorX.setText(etiquetaX);  
  
 } else {  
  
 textValorX.setText("" + x);  
  
 }  
  
 *//columna 3* textValorX.setTextSize(tamanoLetraResolucionIncluida);  
 textValorX.setTextColor(colorColumna2);  
 textValorX.setGravity(Gravity.*CENTER\_HORIZONTAL*);  
  
  
 TextView textValorY = new TextView(context);  
 textValorY.setTextSize(TypedValue.*COMPLEX\_UNIT\_SP*, tamanoLetraResolucionIncluida);  
 textValorY.setLayoutParams(layoutTexto);  
  
 if(contador==0)  
 {  
 textValorY.setText(etiquetaY);  
  
 } else {  
  
 textValorY.setText("" + y);  
  
 }  
  
 textValorY.setTextSize(tamanoLetraResolucionIncluida);  
 textValorY.setTextColor(colorColumna3);  
 textValorY.setGravity(Gravity.*CENTER\_HORIZONTAL*);  
  
  
 fila.setGravity(Gravity.*CENTER\_HORIZONTAL*);  
 *//adicionar las tres columnas a la fila* fila.addView(textNumeroDato);  
 fila.addView(textValorX);  
 fila.addView(textValorY);  
  
 *//Adicionar TabRow a la Tabla* table.addView( fila, new TableLayout.LayoutParams(LayoutParams.*WRAP\_CONTENT*, LayoutParams.*WRAP\_CONTENT*));  
  
  
 }  
  
  
 private void removerFilas() {  
  
  
 if (contador > 1) {  
  
 table.removeAllViews();  
 contador=-1;  
  
 }  
  
 }  
}

La clase **ActividadConfiguracion**.

public class ActividadConfiguracion extends Activity {  
  
 private EditText edit\_text\_numero\_datos, edit\_text\_numero\_datos\_graficar , edit\_text\_broker, edit\_text\_usuario, edit\_text\_topico,  
 edit\_text\_contrasena;;  
 private TextView text\_numero\_datos, text\_numero\_datos\_graficar, text\_broker, text\_usuario, text\_contrasena, text\_topico;  
 private int tamano\_letra;  
 private Button boton\_grabar;  
  
 public void onCreate(Bundle savedInstanceState){  
 super.onCreate(savedInstanceState);  
  
  
 gestionarResolucion();  
  
  
  
 *//para crear elementos de la GUI* crearElementosGUI();  
  
 *//para informar cómo se debe pegar el adminitrador de  
 //diseño obtenido con el método GUI* ViewGroup.LayoutParams parametro\_layout\_principal = new ViewGroup.LayoutParams(ViewGroup.LayoutParams.*MATCH\_PARENT*, ViewGroup.LayoutParams.*MATCH\_PARENT*);  
  
 *//pegar el contenedor con la GUI* this.setContentView(crearGUI(), parametro\_layout\_principal);  
  
 *//Eventos GUI* eventosGui();  
  
 } *//fin del método onCreate* private void gestionarResolucion(){  
  
 tamano\_letra = (int)(0.5f\* AlmacenDatosRAM.*tamanoLetraResolucionIncluida*);  
  
 }  
  
 private void eventosGui() {  
  
 boton\_grabar.setOnClickListener(new View.OnClickListener() {  
  
 @RequiresApi(api = Build.VERSION\_CODES.*KITKAT*)  
 public void onClick(View v) {  
  
 guardar();  
 *//regresar a la actividad anterior* onBackPressed();  
  
 }  
 });  
 }  
  
  
  
 */\*método responsable de la creación de los elementos de la GUI\*/* private void crearElementosGUI(){  
  
 text\_numero\_datos = new TextView(this);  
 text\_numero\_datos.setGravity(Gravity.*FILL\_VERTICAL*);  
 text\_numero\_datos.setTextSize(tamano\_letra);  
 text\_numero\_datos.setText(" NÚMERO DE DATOS (máximo 1000)");  
 text\_numero\_datos.setTextColor(Color.*BLACK*);  
  
 edit\_text\_numero\_datos = new EditText(this);  
 *//Despliega teclado solo con números enteros positivos.* edit\_text\_numero\_datos.setKeyListener(DigitsKeyListener.*getInstance*(false, false));  
 edit\_text\_numero\_datos.setTextSize(tamano\_letra);  
 edit\_text\_numero\_datos.setText("" + AlmacenDatosRAM.*nDatos*);  
  
 text\_numero\_datos\_graficar = new TextView(this);  
 text\_numero\_datos\_graficar.setGravity(Gravity.*FILL\_VERTICAL*);  
 text\_numero\_datos\_graficar.setTextSize(tamano\_letra);  
 text\_numero\_datos\_graficar.setText(" NÚMERO DE DATOS A GRAFICAR");  
 text\_numero\_datos\_graficar.setTextColor(Color.*BLACK*);  
  
 edit\_text\_numero\_datos\_graficar = new EditText(this);  
 *//Despliega teclado solo con números enteros positivos.* edit\_text\_numero\_datos\_graficar.setKeyListener(DigitsKeyListener.*getInstance*(false, false));  
 edit\_text\_numero\_datos\_graficar.setTextSize(tamano\_letra);  
 edit\_text\_numero\_datos\_graficar.setText("" + AlmacenDatosRAM.*nDatosGraficar*);  
  
 text\_broker = new TextView(this);  
 text\_broker.setGravity(Gravity.*FILL\_VERTICAL*);  
 text\_broker.setBackgroundColor(Color.*YELLOW*);  
 text\_broker.setTextSize(tamano\_letra);  
 text\_broker.setText(" URL DEL BROKER");  
 text\_broker.setTextColor(Color.*BLACK*);  
  
 edit\_text\_broker = new EditText(this);  
 edit\_text\_broker.setTextSize(tamano\_letra);  
 edit\_text\_broker.setText("" + AlmacenDatosRAM.*MQTTHOST*);  
  
 text\_usuario = new TextView(this);  
 text\_usuario.setGravity(Gravity.*FILL\_VERTICAL*);  
 text\_usuario.setBackgroundColor(Color.*YELLOW*);  
 text\_usuario.setTextSize(tamano\_letra);  
 text\_usuario.setText(" USUARIO");  
 text\_usuario.setTextColor(Color.*BLACK*);  
  
 edit\_text\_usuario = new EditText(this);  
 edit\_text\_usuario.setTextSize(tamano\_letra);  
 edit\_text\_usuario.setText("" + AlmacenDatosRAM.*USERNAME*);  
  
 text\_contrasena = new TextView(this);  
 text\_contrasena.setGravity(Gravity.*FILL\_VERTICAL*);  
 text\_contrasena.setBackgroundColor(Color.*YELLOW*);  
 text\_contrasena.setTextSize(tamano\_letra);  
 text\_contrasena.setText(" CONTRASEÑA");  
 text\_contrasena.setTextColor(Color.*BLACK*);  
  
 edit\_text\_contrasena = new EditText(this);  
 *// edit\_text\_contrasena.setInputType(InputType.TYPE\_TEXT\_VARIATION\_PASSWORD);* edit\_text\_contrasena.setInputType(InputType.*TYPE\_CLASS\_TEXT* | InputType.*TYPE\_TEXT\_VARIATION\_PASSWORD* );  
 edit\_text\_contrasena.setTransformationMethod(PasswordTransformationMethod.*getInstance*());  
 edit\_text\_contrasena.setTextSize(tamano\_letra);  
 edit\_text\_contrasena.setText("" + AlmacenDatosRAM.*PASSWORD*);  
  
 text\_topico = new TextView(this);  
 text\_topico.setGravity(Gravity.*FILL\_VERTICAL*);  
 text\_topico.setBackgroundColor(Color.*YELLOW*);  
 text\_topico.setTextSize(tamano\_letra);  
 text\_topico.setText(" TOPICO");  
 text\_topico.setTextColor(Color.*BLACK*);  
  
 edit\_text\_topico = new EditText(this);  
 edit\_text\_topico.setTextSize(tamano\_letra);  
 edit\_text\_topico.setText("" + AlmacenDatosRAM.*topicStr*);  
  
 boton\_grabar = new Button(this);  
 boton\_grabar.setTextSize(TypedValue.*COMPLEX\_UNIT\_SP*, tamano\_letra);  
 boton\_grabar.setText("GUARDAR");  
 boton\_grabar.getBackground().setColorFilter(Color.*rgb*(255, 255, 100), PorterDuff.Mode.*MULTIPLY*);  
  
 }  
  
  
 */\*método responsable de administrar el diseño de la GUI\*/* private LinearLayout crearGUI() {  
  
 *//LinearLayoutPrincipal* LinearLayout linear\_principal = new LinearLayout(this);  
 linear\_principal.setOrientation(LinearLayout.*VERTICAL*);  
 linear\_principal.setBackgroundColor(Color.*WHITE*);  
 linear\_principal.setWeightSum(10f);  
  
 *//LinearLayout primera fila* LinearLayout linearLayoutPrimeraFila = new LinearLayout(this);  
 linearLayoutPrimeraFila.setBackgroundColor(Color.*YELLOW*);  
 linearLayoutPrimeraFila.setWeightSum(2f);  
 linearLayoutPrimeraFila.setOrientation(LinearLayout.*HORIZONTAL*);  
  
 *//LinearLayout segunda fila* LinearLayout linearLayoutSegundaFila = new LinearLayout(this);  
 linearLayoutSegundaFila.setBackgroundColor(Color.*YELLOW*);  
 linearLayoutSegundaFila.setWeightSum(2f);  
 linearLayoutSegundaFila.setOrientation(LinearLayout.*HORIZONTAL*);  
  
 *//LinearLayout tercera fila* LinearLayout linearLayoutTerceraFila = new LinearLayout(this);  
 linearLayoutTerceraFila.setBackgroundColor(Color.*YELLOW*);  
 linearLayoutTerceraFila.setWeightSum(2f);  
 linearLayoutTerceraFila.setOrientation(LinearLayout.*HORIZONTAL*);  
  
 *//LinearLayout cuarta fila* LinearLayout linearLayoutCuartaFila = new LinearLayout(this);  
 linearLayoutCuartaFila.setBackgroundColor(Color.*YELLOW*);  
 linearLayoutCuartaFila.setWeightSum(2f);  
 linearLayoutCuartaFila.setOrientation(LinearLayout.*HORIZONTAL*);  
  
 *//LinearLayout quinta fila* LinearLayout linearLayoutQuintaFila = new LinearLayout(this);  
 linearLayoutQuintaFila.setBackgroundColor(Color.*YELLOW*);  
 linearLayoutQuintaFila.setWeightSum(2f);  
 linearLayoutQuintaFila.setOrientation(LinearLayout.*HORIZONTAL*);  
  
 *//LinearLayout sexta fila* LinearLayout linearLayoutSextaFila = new LinearLayout(this);  
 linearLayoutSextaFila.setBackgroundColor(Color.*YELLOW*);  
 linearLayoutSextaFila.setWeightSum(2f);  
 linearLayoutSextaFila.setOrientation(LinearLayout.*HORIZONTAL*);  
  
 *//LinearLayout septima fila* LinearLayout linearLayoutSeptimaFila = new LinearLayout(this);  
 linearLayoutSeptimaFila.setBackgroundColor(Color.*YELLOW*);  
 linearLayoutSeptimaFila.setWeightSum(1f);  
 linearLayoutSeptimaFila.setOrientation(LinearLayout.*HORIZONTAL*);  
  
 *//LinearLayout octava fila* LinearLayout linearLayoutOctavaFila = new LinearLayout(this);  
 linearLayoutOctavaFila.setBackgroundColor(Color.*BLACK*);  
 linearLayoutOctavaFila.setOrientation(LinearLayout.*VERTICAL*);  
  
 *//pegar las filas en el principal* LinearLayout.LayoutParams parametrosPrimerasSieteFilas = new LinearLayout.LayoutParams(ViewGroup.LayoutParams.*MATCH\_PARENT*,0);  
 parametrosPrimerasSieteFilas.weight = 0.5f;  
 linearLayoutPrimeraFila.setLayoutParams(parametrosPrimerasSieteFilas);  
 linearLayoutSegundaFila.setLayoutParams(parametrosPrimerasSieteFilas);  
 linearLayoutTerceraFila.setLayoutParams(parametrosPrimerasSieteFilas);  
 linearLayoutCuartaFila.setLayoutParams(parametrosPrimerasSieteFilas);  
 linearLayoutQuintaFila.setLayoutParams(parametrosPrimerasSieteFilas);  
 linearLayoutSextaFila.setLayoutParams(parametrosPrimerasSieteFilas);  
 linearLayoutSeptimaFila.setLayoutParams(parametrosPrimerasSieteFilas);  
  
 LinearLayout.LayoutParams parametrosOctavaFila = new LinearLayout.LayoutParams(ViewGroup.LayoutParams.*MATCH\_PARENT*,0);  
 parametrosOctavaFila.weight = 6.5f;  
 linearLayoutOctavaFila.setLayoutParams(parametrosOctavaFila);  
  
 linear\_principal.addView(linearLayoutPrimeraFila);  
 linear\_principal.addView(linearLayoutSegundaFila);  
 linear\_principal.addView(linearLayoutTerceraFila);  
 linear\_principal.addView(linearLayoutCuartaFila);  
 linear\_principal.addView(linearLayoutQuintaFila);  
 linear\_principal.addView(linearLayoutSextaFila);  
 linear\_principal.addView(linearLayoutSeptimaFila);  
 linear\_principal.addView(linearLayoutOctavaFila);  
  
 *//pegar elementos en seis filas* LinearLayout.LayoutParams parametros\_pegado\_elementos\_cinco\_primeras\_filas\_I = new LinearLayout.LayoutParams(0, android.view.ViewGroup.LayoutParams.*MATCH\_PARENT*);  
 parametros\_pegado\_elementos\_cinco\_primeras\_filas\_I .weight = 1.0f;  
 LinearLayout.LayoutParams parametros\_pegado\_elementos\_cinco\_primeras\_filas\_D = new LinearLayout.LayoutParams(0, android.view.ViewGroup.LayoutParams.*MATCH\_PARENT*);  
 parametros\_pegado\_elementos\_cinco\_primeras\_filas\_D .weight = 1.0f;  
 *//primera fila* text\_broker.setLayoutParams(parametros\_pegado\_elementos\_cinco\_primeras\_filas\_I);  
 edit\_text\_broker.setLayoutParams(parametros\_pegado\_elementos\_cinco\_primeras\_filas\_D);  
 linearLayoutPrimeraFila.addView(text\_broker);  
 linearLayoutPrimeraFila.addView(edit\_text\_broker);  
 *//segunda fila* text\_usuario.setLayoutParams(parametros\_pegado\_elementos\_cinco\_primeras\_filas\_I);  
 edit\_text\_usuario.setLayoutParams(parametros\_pegado\_elementos\_cinco\_primeras\_filas\_D);  
 linearLayoutSegundaFila.addView(text\_usuario);  
 linearLayoutSegundaFila.addView(edit\_text\_usuario);  
 *//tercera fila* text\_contrasena.setLayoutParams(parametros\_pegado\_elementos\_cinco\_primeras\_filas\_I);  
 edit\_text\_contrasena.setLayoutParams(parametros\_pegado\_elementos\_cinco\_primeras\_filas\_D);  
 linearLayoutTerceraFila.addView(text\_contrasena);  
 linearLayoutTerceraFila.addView(edit\_text\_contrasena);  
 *//cuarta fila* text\_topico.setLayoutParams(parametros\_pegado\_elementos\_cinco\_primeras\_filas\_I);  
 edit\_text\_topico.setLayoutParams(parametros\_pegado\_elementos\_cinco\_primeras\_filas\_D);  
 linearLayoutCuartaFila.addView(text\_topico);  
 linearLayoutCuartaFila.addView(edit\_text\_topico);  
 *//quinta fila* text\_numero\_datos.setLayoutParams(parametros\_pegado\_elementos\_cinco\_primeras\_filas\_I);  
 edit\_text\_numero\_datos.setLayoutParams(parametros\_pegado\_elementos\_cinco\_primeras\_filas\_D);  
 linearLayoutQuintaFila.addView(text\_numero\_datos);  
 linearLayoutQuintaFila.addView(edit\_text\_numero\_datos);  
 *//sexta fila* text\_numero\_datos\_graficar.setLayoutParams(parametros\_pegado\_elementos\_cinco\_primeras\_filas\_I);  
 edit\_text\_numero\_datos\_graficar.setLayoutParams(parametros\_pegado\_elementos\_cinco\_primeras\_filas\_D);  
 linearLayoutSextaFila.addView(text\_numero\_datos\_graficar);  
 linearLayoutSextaFila.addView(edit\_text\_numero\_datos\_graficar);  
  
  
 *//pegar elemento septima fila* LinearLayout.LayoutParams parametros\_pegado\_elemento\_septima\_fila = new LinearLayout.LayoutParams(0, android.view.ViewGroup.LayoutParams.*MATCH\_PARENT*);  
 parametros\_pegado\_elemento\_septima\_fila .weight = 1.0f;  
 linearLayoutSeptimaFila.addView(boton\_grabar,parametros\_pegado\_elemento\_septima\_fila);  
  
  
 return linear\_principal;  
 }  
  
  
 */\*  
 Guardar el estado automáticamente cuando se salga  
 de esta pantalla  
 \*/* protected void onPause() {  
 super.onPause();  
  
 }  
  
 private void guardar(){  
  
 String broker = edit\_text\_broker.getText().toString();  
 AlmacenDatosRAM.*MQTTHOST* = broker;  
  
 String usuario = edit\_text\_usuario.getText().toString();  
 AlmacenDatosRAM.*USERNAME* = usuario;  
  
 String pasword = edit\_text\_contrasena.getText().toString();  
 AlmacenDatosRAM.*PASSWORD* = pasword;  
  
 String topico= edit\_text\_topico.getText().toString();  
 AlmacenDatosRAM.*topicStr* = topico;  
  
 String numero\_datos= edit\_text\_numero\_datos.getText().toString();  
 AlmacenDatosRAM.*nDatos* = Integer.*parseInt*(numero\_datos);  
  
 String numero\_datos\_graficar= edit\_text\_numero\_datos\_graficar.getText().toString();  
 AlmacenDatosRAM.*nDatosGraficar* = Integer.*parseInt*(numero\_datos\_graficar);  
  
  
 SharedPreferences prefs = getSharedPreferences("MisPreferencias", Context.*MODE\_PRIVATE*);  
 SharedPreferences.Editor editor = prefs.edit();  
 editor.putString("broker", AlmacenDatosRAM.*MQTTHOST*);  
 editor.putString("usuario", AlmacenDatosRAM.*USERNAME*);  
 editor.putString("pasword", AlmacenDatosRAM.*PASSWORD*);  
 editor.putString("topico", AlmacenDatosRAM.*topicStr*);  
 editor.putInt("n\_datos", AlmacenDatosRAM.*nDatos*);  
 editor.putInt("n\_datos\_graficar", AlmacenDatosRAM.*nDatosGraficar*);  
 editor.commit();  
  
 }  
  
}

La clase **ActividadComoClienteSubMQTT**.

public class ActividadComoClienteSubMQTT extends Activity implements Runnable {  
  
  
 private int tamanoLetraResolucionIncluida;  
 private LinearLayout linear\_layout\_segunda\_fila;  
 private LinearLayout.LayoutParams parametros\_tabla\_grafica;  
 private Button botonConectar, botonTablaGrafica;  
 private TextView textviewAviso;  
  
 private Luxometro luxometro;  
 private TablaSimple tabla;  
 private Graficador graficador;  
  
 private ClientePubSubMQTT cliente;  
  
 private JSONObject obj;  
  
 private Thread hilo;  
  
 private int periodo\_muestreo;  
 private int contador = 0;  
 private int numero\_datos=0;  
 private int n=-1;  
 private float medida;  
 private int tiempo\_base=0;  
 private int tiempo\_anterior;  
 private int tiempo\_real;  
  
 *//hilo para actualizar tabla* private final Handler myHandler = new Handler();  
  
  
 public void onCreate(Bundle icicle) {  
 super.onCreate(icicle);  
  
 gestionarResolucion();  
  
 crearElementosGUI();  
  
 */\*  
 Para informar cómo se debe pegar el administrador de  
 diseño LinearLayout obtenido con el método crearGui()  
 \*/* ViewGroup.LayoutParams parametro\_layout\_principal = new ViewGroup.LayoutParams(ViewGroup.LayoutParams.*MATCH\_PARENT*, ViewGroup.LayoutParams.*MATCH\_PARENT*);  
  
  
 *//pegar el contenedor con la GUI* this.setContentView(crearGUI(), parametro\_layout\_principal);  
  
  
 eventos();  
  
 crearCliente();  
  
 hilo = new Thread(this);  
  
 }  
  
 private void gestionarResolucion() {  
  
 *//tamano de letra para usar acomodado a la resolución de pantalla* tamanoLetraResolucionIncluida = (int) (0.8 \* AlmacenDatosRAM.*tamanoLetraResolucionIncluida*);  
  
 }  
  
 */\*método responsable de la creación de los elementos de la GUI\*/* private void crearElementosGUI() {  
  
 *//tacometro* luxometro = new Luxometro(this);  
 luxometro.setRango(0,100);  
 luxometro.setUnidades("lx");  
  
 *//tabla* tabla = new TablaSimple(this);  
 tabla.setEtiquetaColumnas("Tiempo (s)", AlmacenDatosRAM.*unidades*);*//"Medida");  
  
  
 //graficador* graficador = new Graficador(this);  
 *//se está muestreando cada segundo (1000 ms)* graficador.setTituloEjeX("Tiempo (s)");  
 graficador.setTituloEjeY(AlmacenDatosRAM.*unidades*);  
 graficador.setGrosorLinea(2f);  
 graficador.setColorLinea(Color.*RED*);  
 graficador.setColorValores(Color.*YELLOW*);  
 graficador.setColorMarcadores(Color.*GREEN*);  
 graficador.setColorFondo(Color.*BLACK*);  
 graficador.setColorTextoEjes(Color.*WHITE*);  
  
 botonConectar = new Button(this);  
 botonConectar.setTextSize(TypedValue.*COMPLEX\_UNIT\_SP*, tamanoLetraResolucionIncluida);  
 botonConectar.setText("CONECTAR");  
 botonConectar.getBackground().setColorFilter(Color.*rgb*(183, 216, 199), PorterDuff.Mode.*MULTIPLY*);  
  
  
 botonTablaGrafica = new Button(this);  
 botonTablaGrafica.setTextSize(TypedValue.*COMPLEX\_UNIT\_SP*, tamanoLetraResolucionIncluida);  
 botonTablaGrafica.setText("GRAFICA");  
 botonTablaGrafica.getBackground().setColorFilter(Color.*rgb*(183, 216, 199), PorterDuff.Mode.*MULTIPLY*);  
 botonTablaGrafica.setEnabled(false);  
  
 textviewAviso = new TextView(this);  
 textviewAviso.setGravity(Gravity.*FILL\_VERTICAL*);  
 textviewAviso.setBackgroundColor(Color.*rgb*(183,216,199));  
 textviewAviso.setTextSize(0.8f\*tamanoLetraResolucionIncluida);  
 textviewAviso.setText(AlmacenDatosRAM.*conectado\_PubSub*);  
 textviewAviso.setTextColor(Color.*BLACK*);  
  
  
 }*//fin crearElemnetosGUI  
  
  
  
  
 /\*método responsable de administrar el diseño de la GUI\*/* private LinearLayout crearGUI() {  
  
 LinearLayout linear\_layout\_principal = new LinearLayout(this);  
 linear\_layout\_principal.setOrientation(LinearLayout.*VERTICAL*);  
 linear\_layout\_principal.setBackgroundColor(Color.*rgb*(183,216,199));  
 linear\_layout\_principal.setWeightSum(10.0f);  
  
 *//LinearLayout primera fila* LinearLayout linear\_layout\_primera\_fila = new LinearLayout(this);  
 linear\_layout\_primera\_fila.setOrientation(LinearLayout.*VERTICAL*);  
 linear\_layout\_primera\_fila.setGravity(Gravity.*FILL*);  
 linear\_layout\_primera\_fila.setBackgroundColor(Color.*rgb*(245, 245, 245));  
  
 *//LinearLayout segunda fila* linear\_layout\_segunda\_fila = new LinearLayout(this);  
 linear\_layout\_segunda\_fila.setOrientation(LinearLayout.*VERTICAL*);  
 linear\_layout\_segunda\_fila.setGravity(Gravity.*FILL*);  
 linear\_layout\_segunda\_fila.setBackgroundColor(Color.*rgb*(245, 245, 245));  
 linear\_layout\_segunda\_fila.setWeightSum(1.0f);  
  
 *//LinearLayout tercera fila* LinearLayout linear\_layout\_tercera\_fila = new LinearLayout(this);  
 linear\_layout\_tercera\_fila.setBackgroundColor(Color.*RED*);  
  
 *//LinearLayout cuarta fila* LinearLayout linear\_layout\_cuarta\_fila = new LinearLayout(this);  
 linear\_layout\_cuarta\_fila.setOrientation(LinearLayout.*HORIZONTAL*);  
 linear\_layout\_cuarta\_fila.setBackgroundColor(Color.*rgb*(183,216,199));  
 linear\_layout\_cuarta\_fila.setWeightSum(2.0f);  
  
 *//pegar primera fila al principal* LinearLayout.LayoutParams parametros\_primera\_fila = new LinearLayout.LayoutParams(ViewGroup.LayoutParams.*MATCH\_PARENT*, 0);  
 parametros\_primera\_fila.weight = 4.25f;  
 parametros\_primera\_fila.setMargins(20, 20, 20, 20);  
 linear\_layout\_primera\_fila.setLayoutParams(parametros\_primera\_fila);  
 linear\_layout\_principal.addView(linear\_layout\_primera\_fila);  
  
 *//pegar segunda fila al principal* LinearLayout.LayoutParams parametros\_segunda\_fila = new LinearLayout.LayoutParams(ViewGroup.LayoutParams.*MATCH\_PARENT*, 0);  
 parametros\_segunda\_fila.weight = 4.25f;  
 parametros\_segunda\_fila.setMargins(20, 20, 20, 20);  
 linear\_layout\_segunda\_fila.setLayoutParams(parametros\_segunda\_fila);  
 linear\_layout\_principal.addView(linear\_layout\_segunda\_fila);  
  
 *//pegar tercera fila al principal* LinearLayout.LayoutParams parametros\_tercera\_fila = new LinearLayout.LayoutParams(ViewGroup.LayoutParams.*MATCH\_PARENT*, 0);  
 parametros\_tercera\_fila.weight =0.5f;  
 parametros\_tercera\_fila.setMargins(20, 0, 20, 0);  
 linear\_layout\_tercera\_fila.setLayoutParams(parametros\_tercera\_fila);  
 linear\_layout\_principal.addView(linear\_layout\_tercera\_fila);  
  
 *//pegar cuarta fila al principal* LinearLayout.LayoutParams parametros\_cuarta\_fila = new LinearLayout.LayoutParams(ViewGroup.LayoutParams.*MATCH\_PARENT*, 0);  
 parametros\_cuarta\_fila.weight = 1.0f;  
 parametros\_cuarta\_fila.setMargins(20, 20, 20, 20);  
 linear\_layout\_cuarta\_fila.setLayoutParams(parametros\_cuarta\_fila);  
 linear\_layout\_principal.addView(linear\_layout\_cuarta\_fila);  
  
 *//pegar gauge en primera fila* linear\_layout\_primera\_fila.addView(luxometro);  
 *//pegar tabla en segunda fila* parametros\_tabla\_grafica = new LinearLayout.LayoutParams(ViewGroup.LayoutParams.*MATCH\_PARENT*,0);  
 parametros\_tabla\_grafica.weight = 1.0f;  
 linear\_layout\_segunda\_fila.addView(tabla,parametros\_tabla\_grafica);  
  
 *//pegado textView en tercera fila* LinearLayout.LayoutParams parametrosPegadoTextView = new LinearLayout.LayoutParams(0, ViewGroup.LayoutParams.*MATCH\_PARENT*);  
 parametrosPegadoTextView.weight = 1.0f;  
 linear\_layout\_tercera\_fila.addView(textviewAviso , parametrosPegadoTextView);  
  
 *//pegar los botones en la cuarta fila* LinearLayout.LayoutParams parametros\_botones\_cuarta\_fila = new LinearLayout.LayoutParams(0, ViewGroup.LayoutParams.*MATCH\_PARENT*);  
 parametros\_botones\_cuarta\_fila.weight = 1.0f;  
 botonConectar.setLayoutParams(parametros\_botones\_cuarta\_fila);  
 botonTablaGrafica.setLayoutParams(parametros\_botones\_cuarta\_fila);  
  
 linear\_layout\_cuarta\_fila.addView(botonConectar);  
 linear\_layout\_cuarta\_fila.addView(botonTablaGrafica);  
  
  
 return linear\_layout\_principal;  
 }*//fin gui* protected void eventos(){  
  
 *//evento cliente* botonConectar.setOnClickListener(new View.OnClickListener() {  
  
 public void onClick(View v) {  
  
 if (botonConectar.getText() == "CONECTAR") {  
 botonConectar.setText("EMPEZAR");  
 *//conectar cliente* cliente.conectar();  
 AlmacenDatosRAM.*estado\_conexion\_nube*=2;  
 actualizarAviso();  
  
  
 } else {  
  
 borrarDatos();  
 botonTablaGrafica.setEnabled(true);  
 empezarHilo();  
 botonConectar.setEnabled(false);  
  
 }  
  
 }  
 });  
  
 *//eventos botones* botonTablaGrafica.setOnClickListener(new View.OnClickListener() {  
  
 public void onClick(View v) {  
  
 if (botonTablaGrafica.getText() == "TABLA") {  
 botonTablaGrafica.setText("GRAFICA");  
 linear\_layout\_segunda\_fila.removeView(graficador);  
 *//pegar tabla en segunda fila* linear\_layout\_segunda\_fila.addView(tabla,parametros\_tabla\_grafica);  
  
  
 } else {  
  
 botonTablaGrafica.setText("TABLA");  
 linear\_layout\_segunda\_fila.removeView(tabla);  
 *//pegar graficador en segunda fila* linear\_layout\_segunda\_fila.addView(graficador,parametros\_tabla\_grafica);  
  
 }  
  
 }  
 });  
  
  
 }  
  
  
  
 public void crearCliente() {  
  
 cliente = new ClientePubSubMQTT(this);  
  
  
 }  
  
 public void empezarHilo() {  
  
 hilo = new Thread(this);  
 hilo.start();  
  
 }  
  
  
 private void actualizarTacometroTabla(){  
  
 luxometro.setUnidades(AlmacenDatosRAM.*unidades*);  
 luxometro.cambiarEscala(medida);  
 tabla.setEtiquetaColumnas("Tiempo en s", "Medida en " + AlmacenDatosRAM.*unidades*);  
  
 }  
  
  
 *//no estaba* private void borrarDatos(){  
 contador =0;  
 AlmacenDatosRAM.*datos*.clear();  
 n=-1;  
 numero\_datos=0;  
 tabla.borrar();  
  
 }  
  
  
 public boolean onKeyDown(int keyCode, KeyEvent event) {  
 if (keyCode == KeyEvent.*KEYCODE\_BACK*) {  
 *// Esto es lo que hace mi botón al pulsar ir a atrás  
 //alerta();* DialogoSalir dialogo\_salir=new DialogoSalir(this);  
 dialogo\_salir.mostrarPopMenuCoeficientes();  
 return true;  
 }  
 return super.onKeyDown(keyCode, event);  
 }  
  
  
  
  
 @RequiresApi(api = Build.VERSION\_CODES.*KITKAT*)  
 public void run() {  
  
 while (true) {  
  
 try {  
 Thread.*sleep*(periodo\_muestreo);  
  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
  
 *//leer IoT* leer();  
  
  
 }  
  
  
 }  
  
  
  
 */\*  
 comunicaciones IoT SUB  
 \*/* @RequiresApi(api = Build.VERSION\_CODES.*KITKAT*)  
 private void leer() {  
  
 *//String JSON* String nuevo\_dato\_string = cliente.leerString();  
  
 if (nuevo\_dato\_string != null) {  
 convertirStrigJson(nuevo\_dato\_string);  
 hacerTrabajoDuro();  
 AlmacenDatosRAM.*estado\_conexion\_nube*=4;  
  
  
  
 } else {  
  
 AlmacenDatosRAM.*estado\_conexion\_nube*=3;  
 actualizarAviso();  
  
 }  
  
 }  
  
 *//obtener la información del JSON* public void convertirStrigJson(String datoString) {  
  
 *//convertir String a JSON* try {  
 obj = new JSONObject(datoString);  
 } catch (JSONException e) {  
 e.printStackTrace();  
 }  
  
 *//obtener la información  
 //la medida* try {  
 AlmacenDatosRAM.*unidades*= obj.getString("unidad");  
  
 } catch (JSONException e) {  
 e.printStackTrace();  
 }  
  
 try {  
 AlmacenDatosRAM.*tiempo*= ""+obj.getInt("tiempo");  
 *//muestrear a más del doble de frecuencia  
 //con la que se envían los datos* periodo\_muestreo = (int)(0.4\*obj.getInt("periodo"));  
 medida =(float) obj.getDouble("valor");  
  
 } catch (JSONException e) {  
 e.printStackTrace();  
 }  
  
 }  
  
  
 */\*  
 Para el asunto de la table es mejor usar un manejador  
 para que no se reviente la aplicación  
 \*/* private void hacerTrabajoDuro() {  
 *//.... realizar el trabajo duro  
  
 //Actualiza la UI usando el handler y el runnable* myHandler.post(updateRunnable);  
  
 }  
  
  
  
 final Runnable updateRunnable = new Runnable() {  
 public void run() {  
  
 actualizarAviso();  
  
 n=n+1;  
  
 *//el instante donde envió el primer dato desde la ESP32  
 //despúes de la conexión* if(n==0){  
 *//convertir a entero (ms)* tiempo\_base=Integer.*parseInt*(AlmacenDatosRAM.*tiempo*);  
 }  
  
  
 *//el tiempo es el real recibido en donde se envío el dato n* tiempo\_real = Integer.*parseInt*(AlmacenDatosRAM.*tiempo*)-tiempo\_base;  
  
  
 if(tiempo\_real!=tiempo\_anterior & numero\_datos<AlmacenDatosRAM.*nDatos*) {  
  
  
 *//ya redeinido de nuevo el cero procedemos a procesar los datos* numero\_datos = numero\_datos + 1;  
 *//convertir a segundos con dos decimales* float t\_r = (float) Math.*round*(tiempo\_real \* 0.001 \* 100f) / 100f;  
  
 *//actualizar la tabla* tabla.enviarDatos(t\_r, (medida));  
  
 *//acualizar luxometro* luxometro.cambiarEscala(medida);  
 luxometro.setMedida(medida);  
  
 *//actualizar la gráfica* graficador.setTituloEjeY(AlmacenDatosRAM.*unidades*);  
 *//graficar los últimos 20 puntos* AlmacenDatosRAM.*datos*.add((new Entry(t\_r, medida)));  
  
 *//graficar sólo un intervalo de datos* graficarIntervaloDatos(AlmacenDatosRAM.*nDatosGraficar*);  
  
  
 tiempo\_anterior = tiempo\_real;  
  
 }  
  
  
 if (numero\_datos+1 > AlmacenDatosRAM.*nDatos*) {  
  
 botonConectar.setEnabled(true);  
  
  
 }  
  
 }  
 };  
  
  
 private void graficarIntervaloDatos(int ancho\_intervalo){  
  
 contador = contador + 1;  
  
 if(contador > ancho\_intervalo) {  
 AlmacenDatosRAM.*datos*.remove(0);  
 }  
 graficador.setDatos(AlmacenDatosRAM.*datos*);  
  
  
 }  
  
  
 private void actualizarAviso(){  
  
  
 if(AlmacenDatosRAM.*estado\_conexion\_nube*==1){  
  
 AlmacenDatosRAM.*conectado\_PubSub*= "Hacer clic en CONECTAR para acceder al BROKER...";  
 String aviso = AlmacenDatosRAM.*conectado\_PubSub*;  
 textviewAviso.setText(aviso);  
  
 }  
  
 if(AlmacenDatosRAM.*estado\_conexion\_nube*==2){  
  
 AlmacenDatosRAM.*conectado\_PubSub*= "Hacer clic en EMPEZAR para solictar datos del BROKER...";  
 String aviso = AlmacenDatosRAM.*conectado\_PubSub*;  
 textviewAviso.setText(aviso);  
  
 }  
  
  
 if(AlmacenDatosRAM.*estado\_conexion\_nube*==3){  
  
 AlmacenDatosRAM.*conectado\_PubSub*= " No se están recibiendo datos ...";  
 String aviso = AlmacenDatosRAM.*conectado\_PubSub*;  
 textviewAviso.setText(aviso);  
  
 }  
  
 if(AlmacenDatosRAM.*estado\_conexion\_nube*==4){  
  
 AlmacenDatosRAM.*conectado\_PubSub*= " Recibiendo datos ...";  
 String aviso = AlmacenDatosRAM.*conectado\_PubSub*;  
 textviewAviso.setText(aviso);  
  
 }  
  
 }  
  
 @Override  
 protected void onResume() {  
 super.onResume();  
 AlmacenDatosRAM.*estado\_conexion\_nube*=1;  
 actualizarAviso();  
  
 }  
  
 protected void onPause() {  
 super.onPause();  
   
 }  
  
}

La clase **ActividadPrincipalMiCuadragesimaCuartaApp**.

public class ActividadPrincipalMiCuadragesimaCuartaApp extends Activity {  
  
  
 private Boton entrar, salir, ajustes;  
 LinearLayout linear\_layout\_segunda\_fila;  
  
  
 public void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
  
 gestionarResolucion();  
  
 *//para crear elementos de la GUI* crearElementosGUI();  
  
 */\*  
 Para informar cómo se debe pegar el administrador de  
 diseño LinearLayout obtenido con el método crearGui()  
 \*/* ViewGroup.LayoutParams parametro\_layout\_principal = new ViewGroup.LayoutParams(ViewGroup.LayoutParams.*MATCH\_PARENT*, ViewGroup.LayoutParams.*MATCH\_PARENT*);  
  
  
 *//pegar el contenedor con la GUI* this.setContentView(crearGUI(), parametro\_layout\_principal);  
  
 *//actualizar preferencias* actualizarPreferenciasMQTT();  
  
  
 eventos();  
  
  
 }*//fin del método onCreate  
  
 /\*Método auxiliar para asuntos de resolución\*/* private void gestionarResolucion() {  
  
 *//independencia de la resolución de la pantalla* DisplayMetrics displayMetrics = this.getApplicationContext().getResources().getDisplayMetrics();  
 int alto = displayMetrics.heightPixels;  
 int ancho = displayMetrics.widthPixels;  
 AlmacenDatosRAM.*ancho*=ancho;  
 AlmacenDatosRAM.*alto*=alto;  
 int dimensionReferencia;  
  
 *//tomar el menor valor entre alto y ancho de pantalla* if (alto > ancho) {  
  
 dimensionReferencia = ancho;  
 } else {  
  
 dimensionReferencia = alto;  
 }  
  
 AlmacenDatosRAM.*dimensionReferencia* =dimensionReferencia;  
  
 *//una estimación de un buen tamaño* int tamanoLetra = dimensionReferencia / 20;  
  
  
 *//tamano de letra para usar acomodado a la resolución de pantalla* int tamanoLetraResolucionIncluida = (int) (tamanoLetra / displayMetrics.scaledDensity);  
  
 *//guardar en el almacen de datos para que otras clases la accedan fácilmente* AlmacenDatosRAM.*tamanoLetraResolucionIncluida* = tamanoLetraResolucionIncluida;  
  
 }  
  
  
 private void crearElementosGUI() {  
  
 entrar = new Boton(this);  
 entrar.setImagen(R.drawable.*entrar*);  
  
 salir = new Boton(this);  
 salir.setImagen(R.drawable.*salir*);  
  
 ajustes = new Boton(this);  
 ajustes.setImagen(R.drawable.*configuracion*);  
  
 }  
  
  
 */\*método responsable de administrar el diseño de la GUI\*/* private LinearLayout crearGUI() {  
  
  
 LinearLayout linear\_layout\_principal = new LinearLayout(this);  
 linear\_layout\_principal.setOrientation(LinearLayout.*VERTICAL*);  
 linear\_layout\_principal.setGravity(Gravity.*CENTER\_HORIZONTAL*);  
 linear\_layout\_principal.setGravity(Gravity.*FILL*);  
 linear\_layout\_principal.setBackgroundColor(Color.*WHITE*);  
 linear\_layout\_principal.setWeightSum(10);  
  
  
 *//LinearLayout primera fila* LinearLayout linear\_layout\_primera\_fila = new LinearLayout(this);  
 linear\_layout\_primera\_fila.setOrientation(LinearLayout.*HORIZONTAL*);  
 linear\_layout\_primera\_fila.setGravity(Gravity.*FILL*);  
 linear\_layout\_primera\_fila.setBackgroundColor(Color.*BLACK*);  
 LinearLayout.LayoutParams parametros\_primera\_fila = new LinearLayout.LayoutParams(ViewGroup.LayoutParams.*MATCH\_PARENT*, 0);*//ViewGroup.LayoutParams.WRAP\_CONTENT);* parametros\_primera\_fila.weight = 9.0f;  
 linear\_layout\_primera\_fila.setLayoutParams(parametros\_primera\_fila);  
  
  
 *//fondo primera fila* Drawable fondo = getResources().getDrawable(R.drawable.*luxometro\_android*);  
 linear\_layout\_primera\_fila.setBackgroundDrawable(fondo);  
  
 *//LinearLayout segunda fila* linear\_layout\_segunda\_fila = new LinearLayout(this);  
 linear\_layout\_segunda\_fila.setOrientation(LinearLayout.*HORIZONTAL*);  
 linear\_layout\_segunda\_fila.setGravity(Gravity.*FILL*);  
 LinearLayout.LayoutParams parametros\_segunda\_fila = new LinearLayout.LayoutParams(ViewGroup.LayoutParams.*MATCH\_PARENT*, 0);  
 parametros\_segunda\_fila.weight = 1.0f;  
 linear\_layout\_segunda\_fila.setWeightSum(2.0f);  
 linear\_layout\_segunda\_fila.setLayoutParams(parametros\_segunda\_fila);  
  
  
 LinearLayout.LayoutParams parametros\_pegado\_boton = new LinearLayout.LayoutParams(0, ViewGroup.LayoutParams.*MATCH\_PARENT*);  
 parametros\_pegado\_boton.weight = 1.0f;  
 entrar.setLayoutParams(parametros\_pegado\_boton);  
 salir.setLayoutParams(parametros\_pegado\_boton);  
 ajustes.setLayoutParams(parametros\_pegado\_boton);  
 linear\_layout\_segunda\_fila.addView(entrar);  
 linear\_layout\_segunda\_fila.addView(ajustes);  
  
  
 linear\_layout\_principal.addView(linear\_layout\_primera\_fila);  
 linear\_layout\_principal.addView(linear\_layout\_segunda\_fila);  
  
  
 return linear\_layout\_principal;  
  
  
 }  
  
  
 private void eventos() {  
  
 entrar.setOnClickListener(new View.OnClickListener() {  
  
 public void onClick(View v) {  
  
 lanzarActividadComunicacion();  
 linear\_layout\_segunda\_fila.removeAllViews();  
 linear\_layout\_segunda\_fila.addView(salir);  
 linear\_layout\_segunda\_fila.addView(ajustes);  
  
 }  
 });  
  
 salir.setOnClickListener(new View.OnClickListener() {  
  
 public void onClick(View v) {  
  
 finish();  
 }  
 });  
  
 ajustes.setOnClickListener(new View.OnClickListener() {  
  
 public void onClick(View v) {  
  
 lanzarActividadConfiguracion();  
 }  
 });  
  
  
 }  
  
  
  
 private void lanzarActividadComunicacion() {  
  
 Intent intent = new Intent(this, ActividadComoClienteSubMQTT.class);  
 startActivity(intent);  
  
 }  
  
  
 private void lanzarActividadConfiguracion(){  
  
 Intent intent = new Intent(this, ActividadConfiguracion.class);  
 startActivity(intent);  
  
 }  
  
  
  
 protected void onDestroy() {  
 super.onDestroy();  
 finish();  
  
 }  
  
  
 private void actualizarPreferenciasMQTT(){  
  
 SharedPreferences prefs = getSharedPreferences("MisPreferencias", Context.*MODE\_PRIVATE*);  
  
 AlmacenDatosRAM.*MQTTHOST* = prefs.getString("broker", "");  
 AlmacenDatosRAM.*USERNAME* = prefs.getString("usuario", "");  
 AlmacenDatosRAM.*PASSWORD* = prefs.getString("pasword","");*///no lo tenía* AlmacenDatosRAM.*topicStr* = prefs.getString("topico", "");  
 AlmacenDatosRAM.*nDatos* = prefs.getInt("n\_datos", 50);  
 AlmacenDatosRAM.*nDatosGraficar* = prefs.getInt("n\_datos\_graficar", 10);  
  
 }  
  
  
  
}

El **archivo manifiesto**.

*<?*xml version="1.0" encoding="utf-8"*?>*<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 package="com.curso\_simulaciones.micuadragesimacuartaapp">  
  
 <application  
 android:allowBackup="true"  
 android:dataExtractionRules="@xml/data\_extraction\_rules"  
 android:fullBackupContent="@xml/backup\_rules"  
 android:icon="@mipmap/ic\_launcher"  
 android:label="@string/app\_name"  
 android:roundIcon="@mipmap/ic\_launcher\_round"  
 android:supportsRtl="true"  
 android:theme="@style/Theme.MiCuadragesimaCuartaApp"  
 tools:targetApi="31" >  
  
 <activity android:name=".ActividadPrincipalMiCuadragesimaCuartaApp"  
 android:theme="@android:style/Theme.NoTitleBar.Fullscreen"  
 android:screenOrientation="portrait"  
 android:exported="true">  
  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN" />  
 <category android:name="android.intent.category.LAUNCHER" />  
 </intent-filter>  
  
  
 </activity>  
  
 *<!--declarando otra actividad-->* <activity  
 android:name=".actividades\_secundarias.ActividadComoClienteSubMQTT"  
 android:screenOrientation="portrait"  
 android:theme="@android:style/Theme.NoTitleBar.Fullscreen"></activity>  
  
 *<!--declarando otra actividad-->* <activity  
 android:name=".actividades\_secundarias.ActividadConfiguracion"  
 android:screenOrientation="portrait"  
 android:theme="@android:style/Theme.NoTitleBar.Fullscreen"></activity>  
  
  
 <service android:name="org.eclipse.paho.android.service.MqttService"/>  
  
 </application>  
  
 <uses-permission android:name="android.permission.WAKE\_LOCK" />  
 <uses-permission android:name="android.permission.INTERNET" />  
 <uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE" />  
  
  
</manifest>

**Nota importante**:

Recordar que para lograr compatibilidad a partir de Android 12 se debe hacer el cambio en el archivo **build.gradle (Module:app)** en la línea de código resaltada con amarillo (se cambió a valor 30, es decir una api anterior). Se debe **sincronizar**:

plugins **{** id 'com.android.application'  
**}**android **{** namespace 'com.curso\_simulaciones.micuadragesimacuartaapp'  
 compileSdk 33  
  
 defaultConfig **{** applicationId "com.curso\_simulaciones.micuadragesimacuartaapp"  
 minSdk 21  
 targetSdk 30  
 versionCode 1  
 versionName "1.0"  
  
 testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"  
 **}**

**El programa para ESP32**

Conectar la **ESP32** al computador y programarla desde el **IDE** de **ARDUINO** con el siguiente código (**luxometro\_mqtt\_esp32.ino**).

//para comunicación WiFi

#include <WiFi.h>

//para el protocolo MQTT de IoT

#include <PubSubClient.h>

//para comunicación del sensor

#include <Wire.h>

#include <BH1750.h>

//para manejar JSON

#include <ArduinoJson.hpp>

#include <ArduinoJson.h>

BH1750 lightMeter(0x23);

StaticJsonDocument<300> doc;//300 bytes

uint16\_t valor;

int tiempo; //en ms

int periodo = 100; //en ms

int minimo=0;

int maximo=100;

//Variables para conexiones WiFi

const char\* ssid = "xxxxxx";//reemplazar SSID;

const char\* password = "xxxxxxx";//reemplazar pasword;

//datos del Broker MQTT:

const char\* mqtt\_server = "168.176.136.61";

const int mqttPort = 1883;

const char\* mqttUser = "diego";//reemplazar usuario

const char\* mqttPassword = " ";//reemplazar pasword

const char\* topico = "iot/simulaciones/luxometro/equipo\_0";//reemplazar pasword

WiFiClient espCliente;

PubSubClient mqttCliente(espCliente);

void setup() {

Serial.begin(115200);

Wire.begin();

if (lightMeter.begin()) {

Serial.println(F("BH1750 inicializado"));

}

else {

Serial.println(F("Error inicializando BH1750"));

}

Serial.println(F("BH1750 Test begin"));

//conectar a WiFi

conectarToWiFi();

//configurar MQTT

setupMQTT();

}

//administrar conexión wiFi

void conectarToWiFi() {

delay(10);

//Comenzar conexión a red WiFI

Serial.println();

Serial.print("Conectando a...");

Serial.println(ssid);

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println("");

Serial.println("WiFi conectado");

Serial.println("IP address: ");

Serial.println(WiFi.localIP());

}

//administrar configuración de conexión al Broker MQTT

void setupMQTT() {

mqttCliente.setServer(mqtt\_server, mqttPort);

//establecer la función de devolución de llamada

mqttCliente.setCallback(callback);

}

//conectar el cliente ESP32 MQTT al Broker

void reconnect() {

Serial.println("Conectando a Broker MQTT...");

//loop hasta lograr conexión

while (!mqttCliente.connected()) {

Serial.println("Reconectando al Broker MQTT..");

String clientId = "ESP32Client-";

clientId += String(random(0xffff), HEX);

if (mqttCliente.connect(clientId.c\_str())) {

Serial.println("Conectado");

//suscripción al tópico

mqttCliente.subscribe(topico);

}

}

}

/\*

Ahora se especifica una función de devolución de llamada.

Primero se imprime el nombre del tema y luego se

recibe el mensaje.

\*/

void callback(char\* topic, byte\* payload, unsigned int length) {

Serial.print("Callback - ");

Serial.print("Message:");

for (int i = 0; i < length; i++) {

Serial.print((char)payload[i]);

}

}

void loop() {

if (!mqttCliente.connected())

reconnect();

mqttCliente.loop();

//leer luxometro

valor = lightMeter.readLightLevel();

//serializar datos JSON para enviarlos

SerializeObject();

//cada periodo de muestreo

delay(periodo);

tiempo = tiempo + periodo;

} //fin loop

void SerializeObject(){

doc["maximo"] = maximo;

doc["minimo"] = minimo;

doc["unidad"] = "lx";

doc["periodo"] = periodo;

doc["tiempo"] = tiempo;

doc["valor"] = valor;

char buffer[200];

serializeJsonPretty(doc, buffer);

Serial.println("Enviando mensaje a MQTT tópico...");

Serial.println(buffer);

if (mqttCliente.publish(topico, buffer) == true) {

Serial.println("Envío de mensaje exitoso");

} else {

Serial.println("Error en el envío del mensaje");

}

mqttCliente.loop();

Serial.println("-------------");

}

**El montaje ESP32**

El montaje es el de la Figura 8.

**Ejecutar**

Ejecutar la app. Se despliega la GUI de la Figura 10 izquierda. Hacer clic en el botón AJUSTES y se despliega la GUI de la Figura 10 derecha. En esta GUI se configura la app: URL del BROKER (escribir el URL del BROKER de la Escuela de Física: tcp://168.176.136.61:1883), USUARIO (cualquiera), CONTRASEÑA (no es necesario en esta app), TOPICO (cualquiera: **cada grupo pone su tópico**). Todos los dispositivos móviles que quieran recibir los datos deben tener el mismo tópico. También se introduce el número de datos que se desea recoger y el numero de datos que se desplegarán gráficamente (esto último para evitar la congestión en la gráfica).

Imagen que contiene Texto

Descripción generada automáticamente Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Figura 10

Una vez configurada la app (esta configuración será persistente) regresar y hacer clic en el botón ENTRAR. Se despliega la GUI de la Figura 11 izquierda. Comenzar la comunicación haciendo clic en el botón EMPEZAR.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamenteInterfaz de usuario gráfica, Aplicación

Descripción generada automáticamenteInterfaz de usuario gráfica, Texto

Descripción generada automáticamenteGráfico, Gráfico de líneas

Descripción generada automáticamente

Figura 11

**TAREA**

Implementar un proyecto en el que la ESP32 envia los datos de los sensores acoplados a ella: el sensor DHT11 (humedad relativa y temperatura) y la distancia a un obstáculo medida con el semsor ultrasónico SR04 (en donde se hace la corrección de la velocidad del sonido en el aire debido a la temperatura y la humedad relativa). La aplicación android debe comunicarse con la ESP32 vía el IoT en donde el móvil hace de SUB y la ESP32 de PUB y los datos intercambiados deben tener el **formato JSON**

* La jerarquía de clases y paquetes debe ser la de la Figura 12.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Figura 12

* Las GUIs desplegadas deben tener el aspecto de las GUI de la Figura 13A, 13B, 14A, 14B y 15. **Observar que en el Gauge** se debe desplegar la información de la distancia (con escala dinámica), la temperatura y la humedad relativa Figuras 16A y 16B. La tabla y la gráfica despliegan la información de la distancia y el tiempo.
* El tacómetro debe ser construido con código propio y la gráfica se apoya en la librería librería **MPAndroidChart** y la comunicación IoT se apoya en la librería **Eclipse Paho** tal como se hizo en la app anterior.

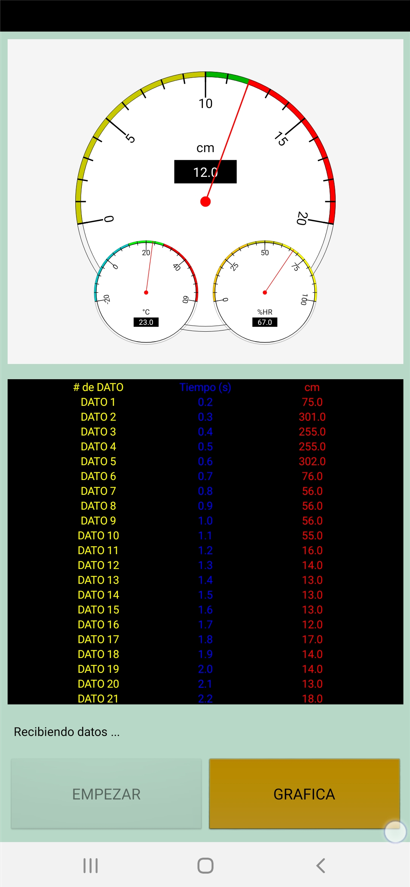
 Interfaz de usuario gráfica

Descripción generada automáticamente

A B

Figra 13

Captura de pantalla de un celular

Descripción generada automáticamente 

A B

Figura 14

Gráfico

Descripción generada automáticamente

Figura 15

* La app debe ejecutarse simultáneamente en **mínimo tres** dispositivos android.
* La app debe instalarse con **icono personal**.
* Debe entregarse en un archivo .zip los siguiente:
  + Código fuente de la app
  + El apk de la app con icono personal
  + Código fuente del programa de la ESP32.

**REFERENCIAS**

* Developers. Introducción general a Bluetooth. <https://developer.android.com/guide/topics/connectivity/bluetooth?hl=es-419>. Consultada [Abril 16 de 2022].