androidthings

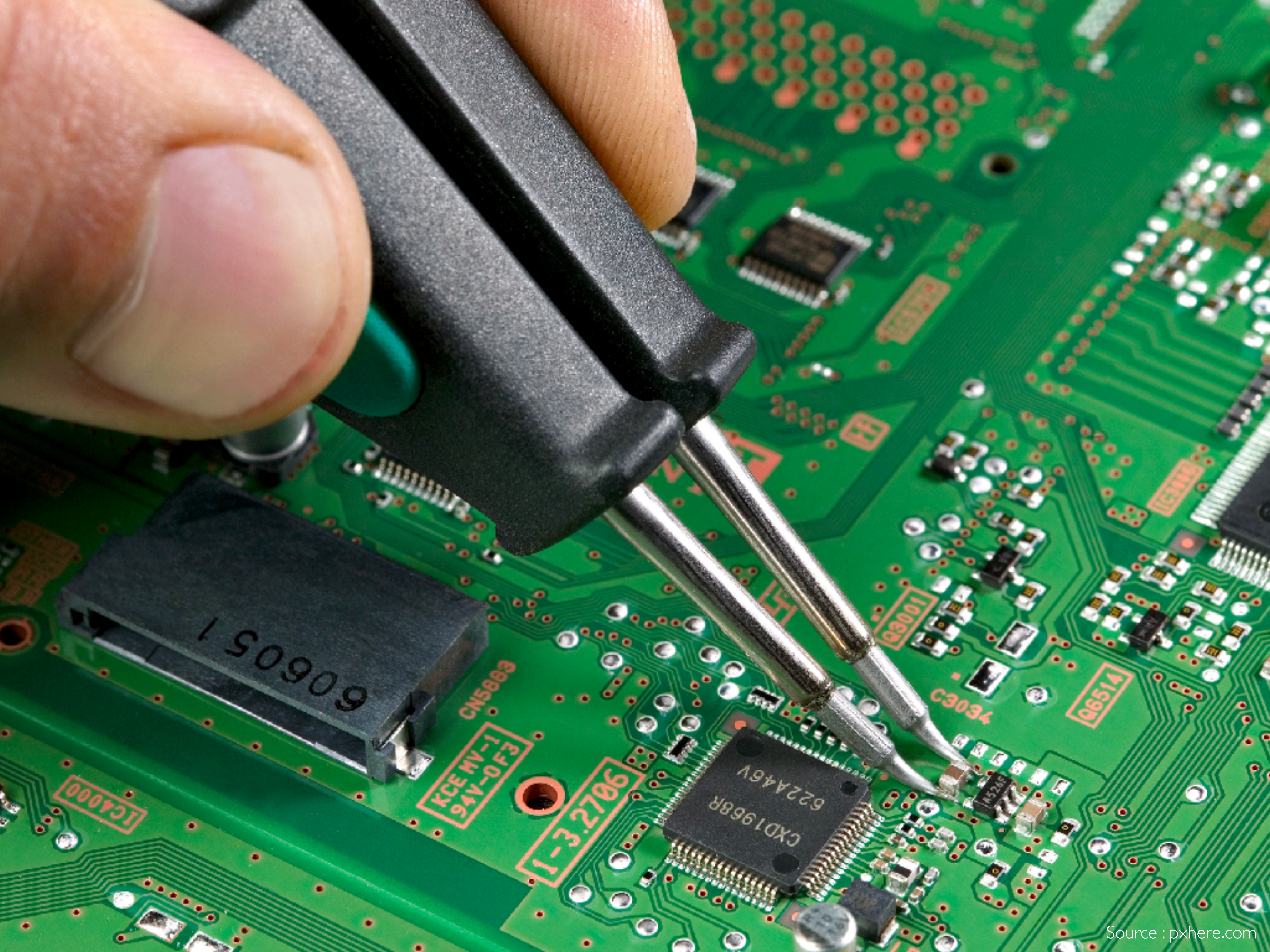
Comment faire sa domotique DIY
sans jamais toucher à un fer à souder

Code d'Armor

Source : pxhere.com

PROBLEME

Source : pxhere.com

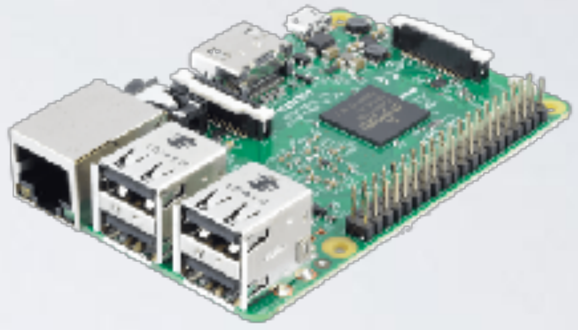


Source : pxhere.com

HARDWARE

HARDWARE

- Raspberry Pi 3 Model B
- RFXCom RFXtrx433 USB
- Oregon Scientific THGR122
  LaCrosse TX3



SOFTWARE

- Domoticz
- openHAB
- …

PROBLÈME

(bis)

Source : pxhere.com

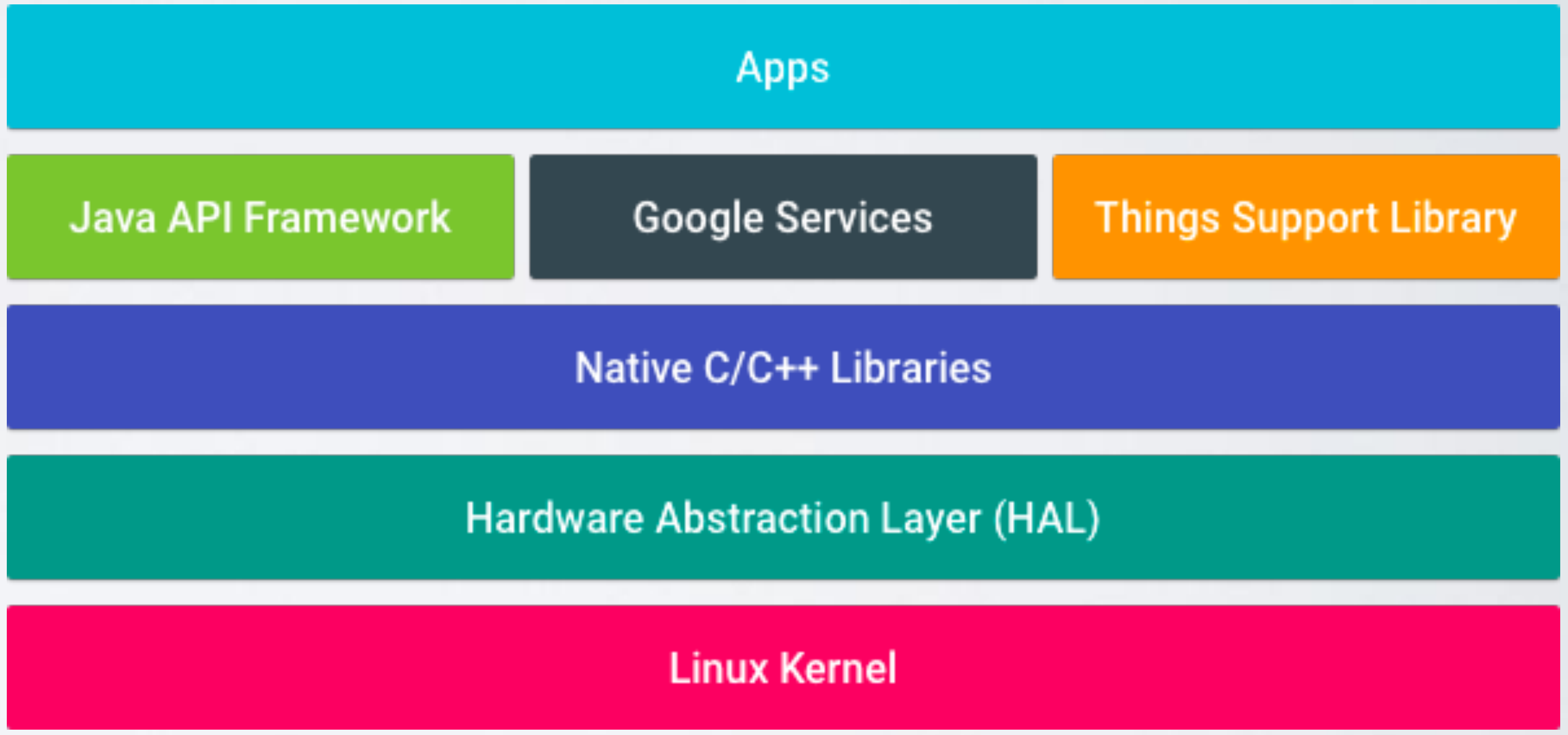
Source : pxhere.com

androidthings

Code d'Armor

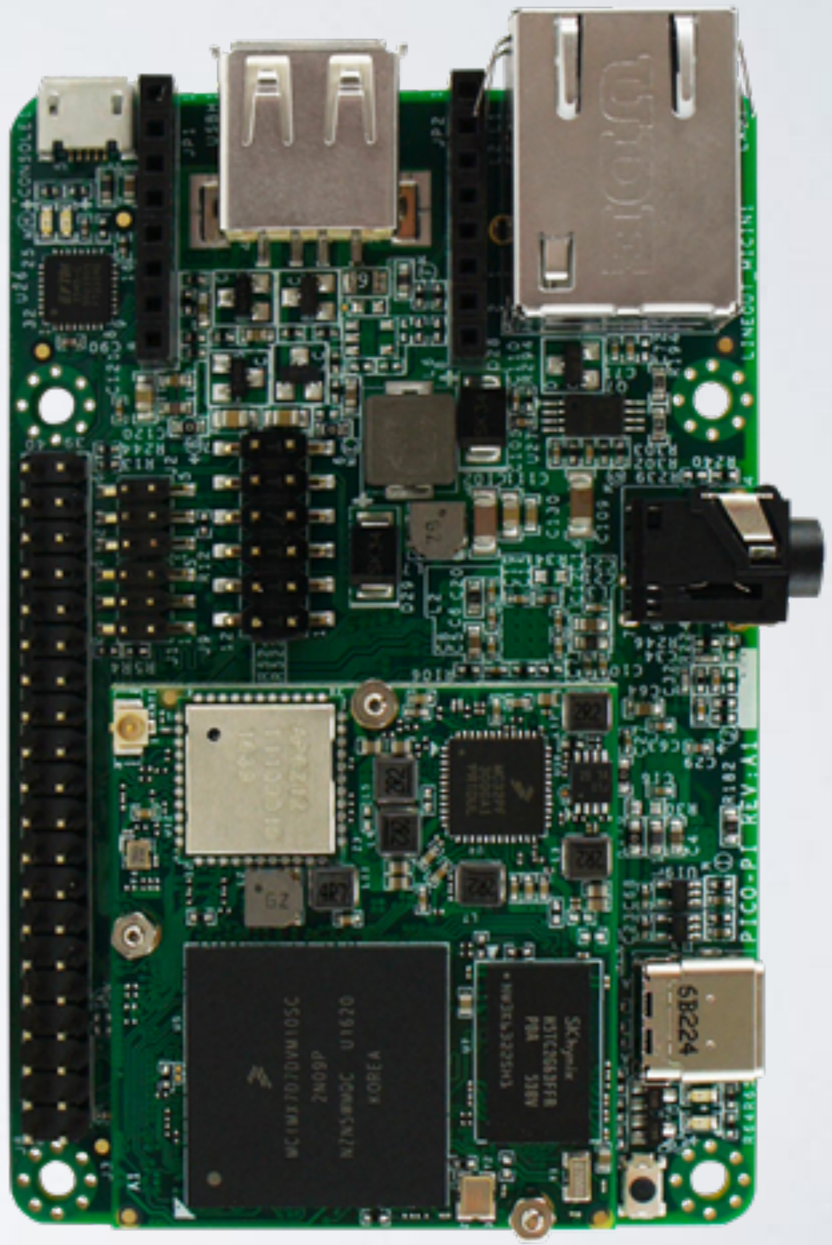
ANDROID THINGS

- Core Android
- Things support lib



ANDROID THING

- NXP Pico (i.MX7D & i.MX6UL)
- NXP Argon (i.MX6UL)
- Raspberry Pi 3 (Model B)



ANDROID THING

Welcome to Android Things

Build hardware that harnesses the power of
Android and the scale of Google services.

Learn more

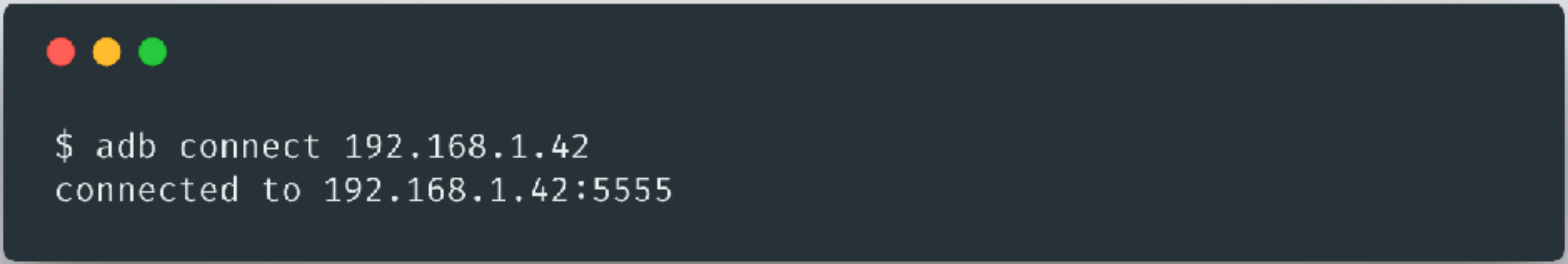
https://partner.android.com/things/console

DÉVELOPPER POUR THINGS

DÉVELOPPER

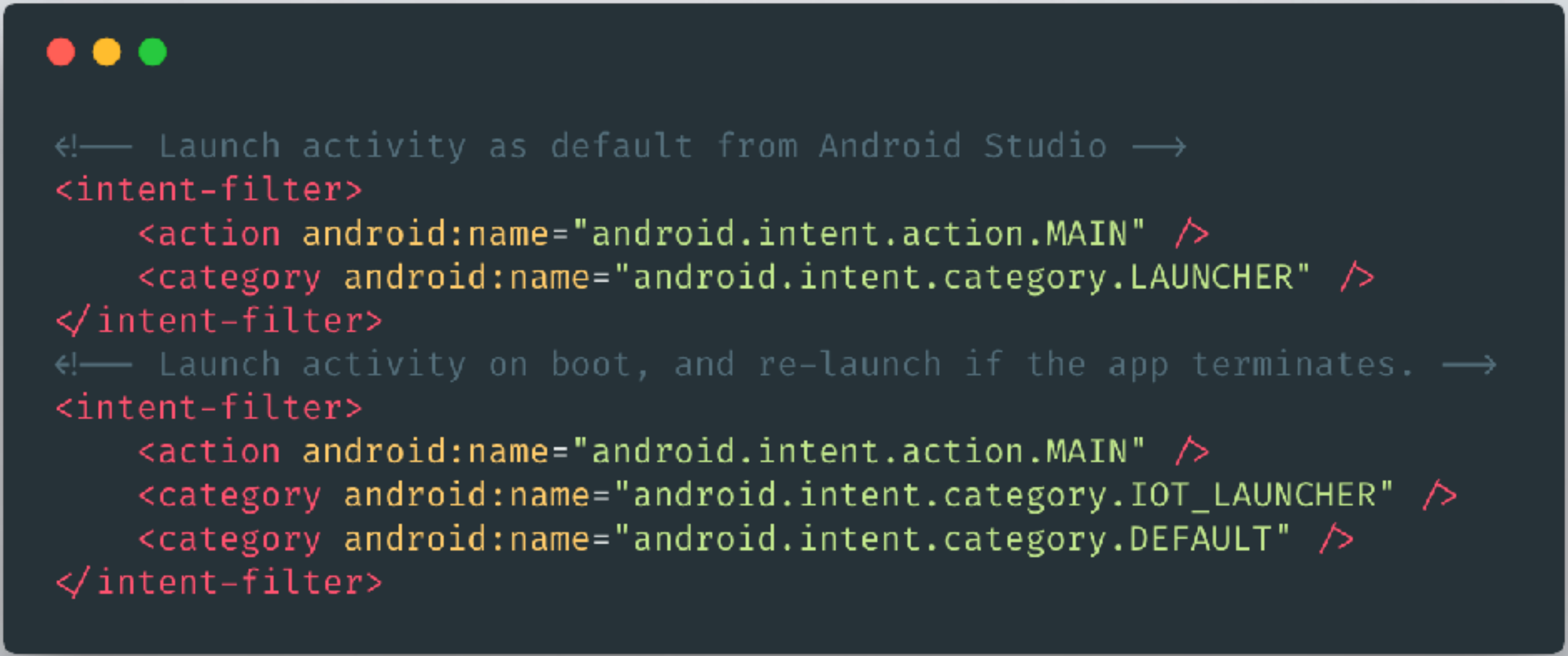
- Android Studio
- c’est tout

DÉVELOPPER



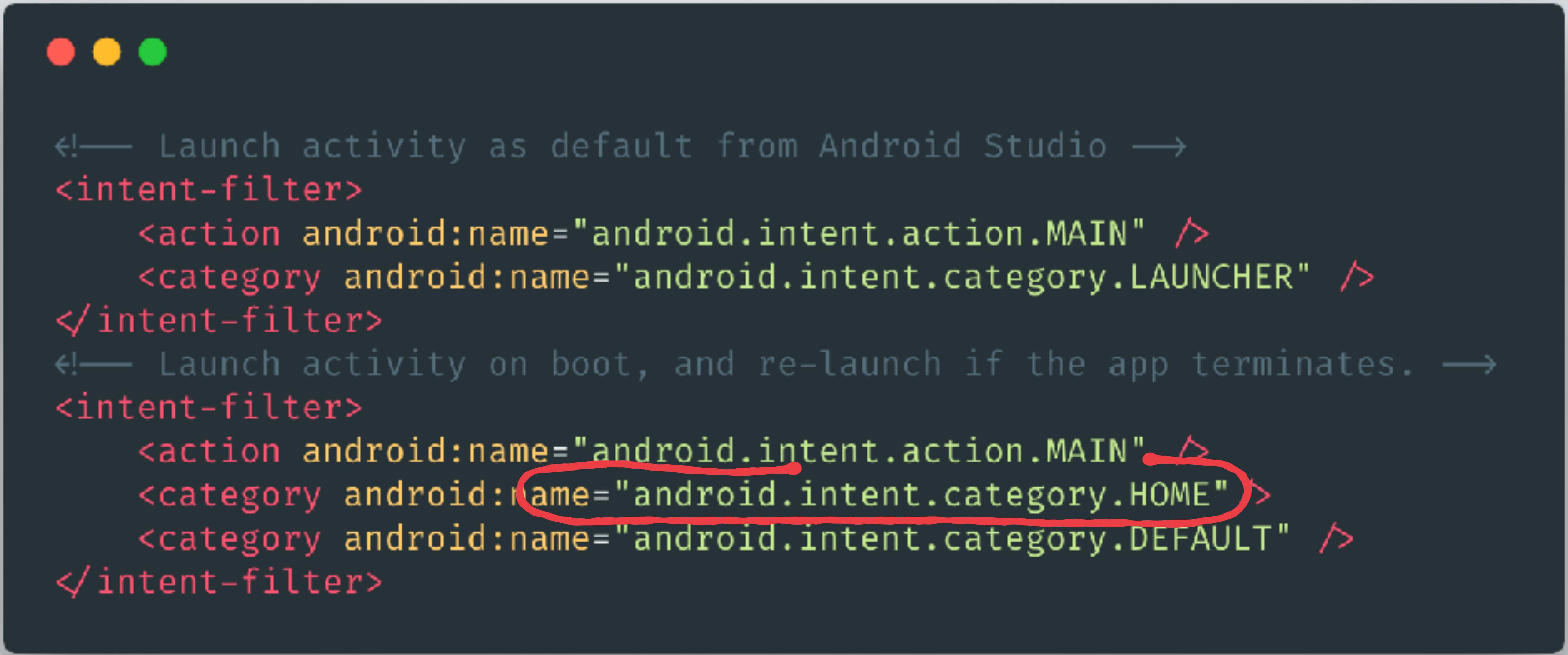
```
$ adb connect 192.168.1.42
connected to 192.168.1.42:5555
```

DÉVELOPPER



```xml
<!-- Launch activity as default from Android Studio -->
<intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
<!-- Launch activity on boot, and re-launch if the app terminates. -->
<intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.IOT_LAUNCHER" />
    <category android:name="android.intent.category.DEFAULT" />
</intent-filter>
```

DÉVELOPPER



```
<!-- Launch activity as default from Android Studio -->
<intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
<!-- Launch activity on boot, and re-launch if the app terminates. -->
<intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.HOME" />
    <category android:name="android.intent.category.DEFAULT" />
</intent-filter>
```

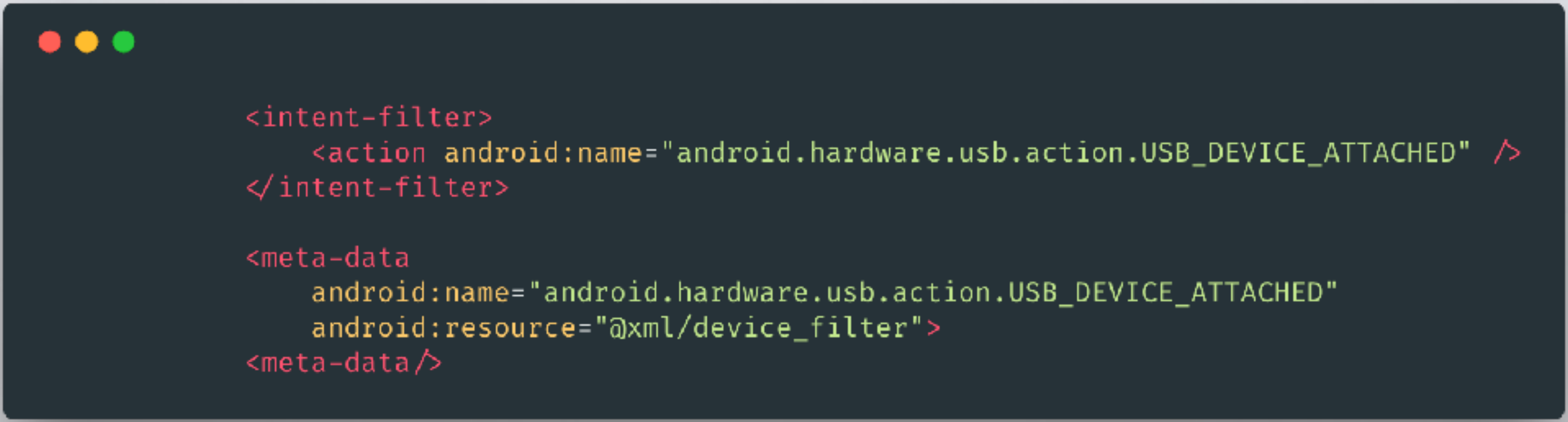
LIRE LES DONNÉES

LIRE LES DONNÉES

- UART API
- USB HOST API

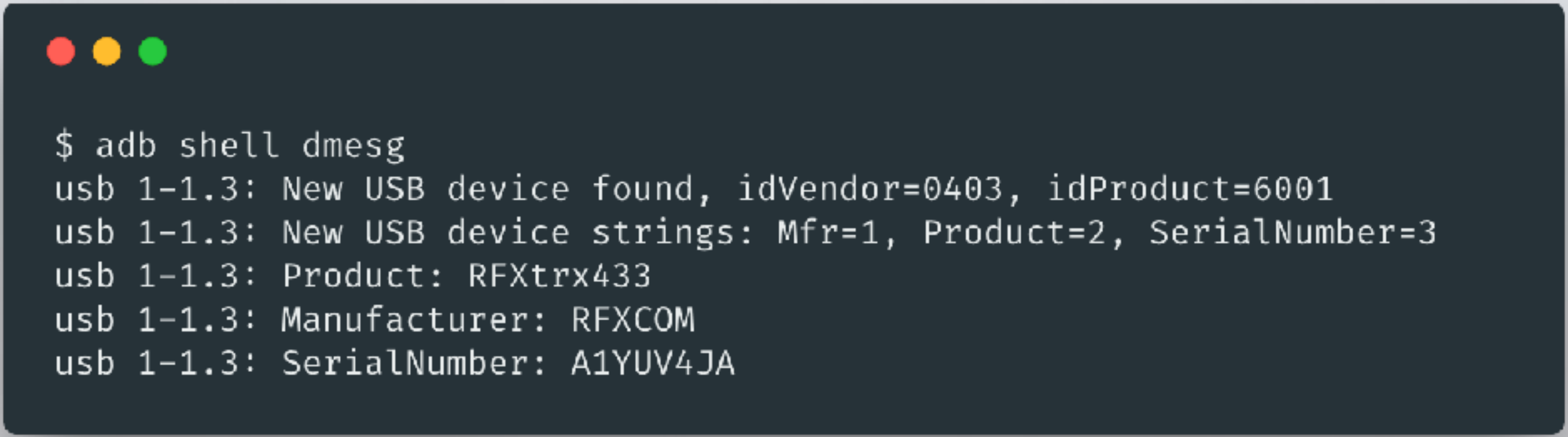
Code d'Armor

LIRE LES DONNÉES



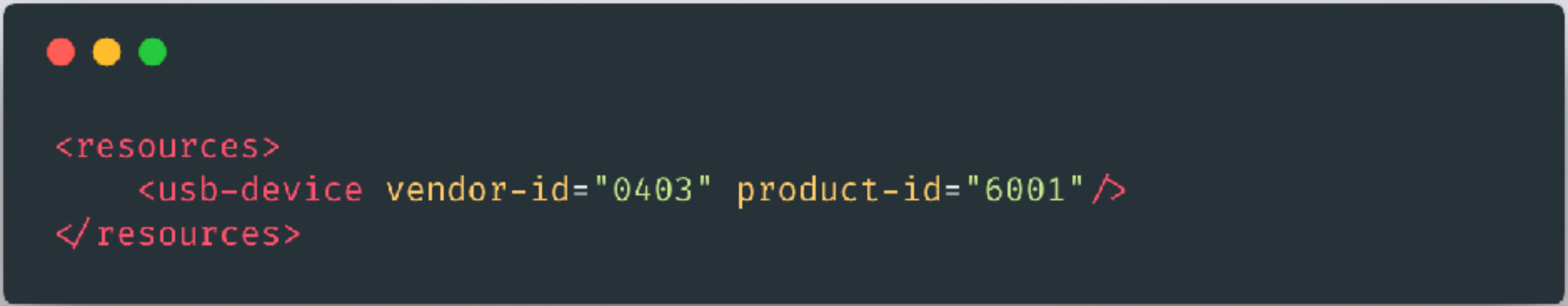
```
<intent-filter>
    <action android:name="android.hardware.usb.action.USB_DEVICE_ATTACHED" />
</intent-filter>

<meta-data
    android:name="android.hardware.usb.action.USB_DEVICE_ATTACHED"
    android:resource="@xml/device_filter">
<meta-data/>
```

LIRE LES DONNÉES



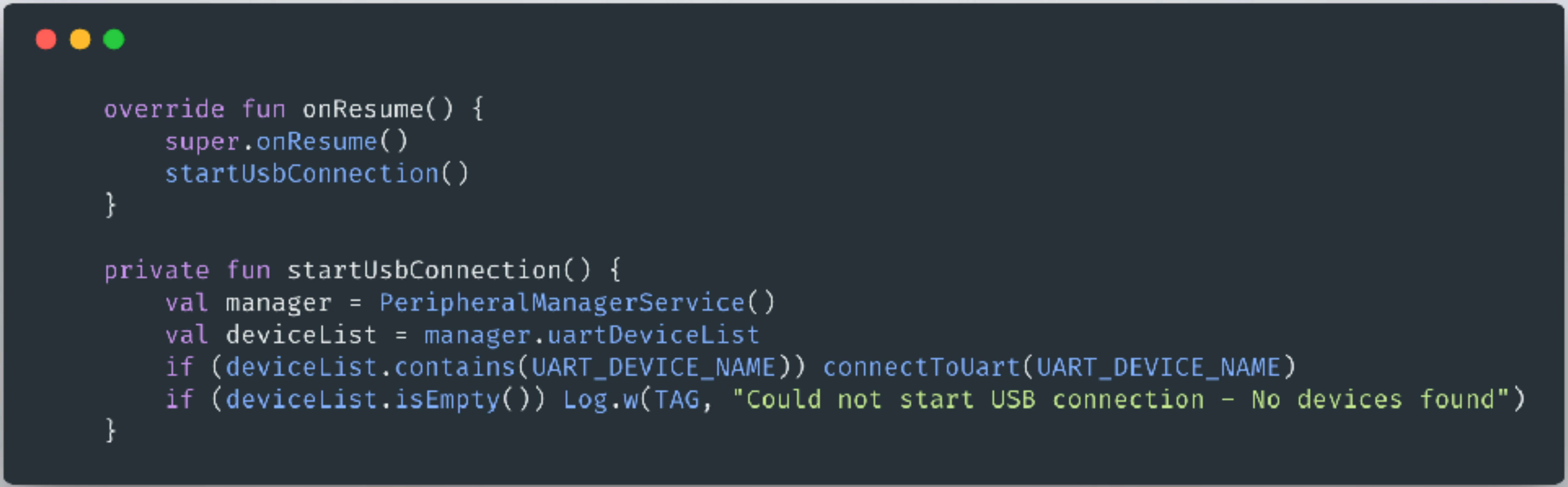
```
$ adb shell dmesg
usb 1-1.3: New USB device found, idVendor=0403, idProduct=6001
usb 1-1.3: New USB device strings: Mfr=1, Product=2, SerialNumber=3
usb 1-1.3: Product: RFXtrx433
usb 1-1.3: Manufacturer: RFXCOM
usb 1-1.3: SerialNumber: A1YUV4JA
```

LIRE LES DONNÉES



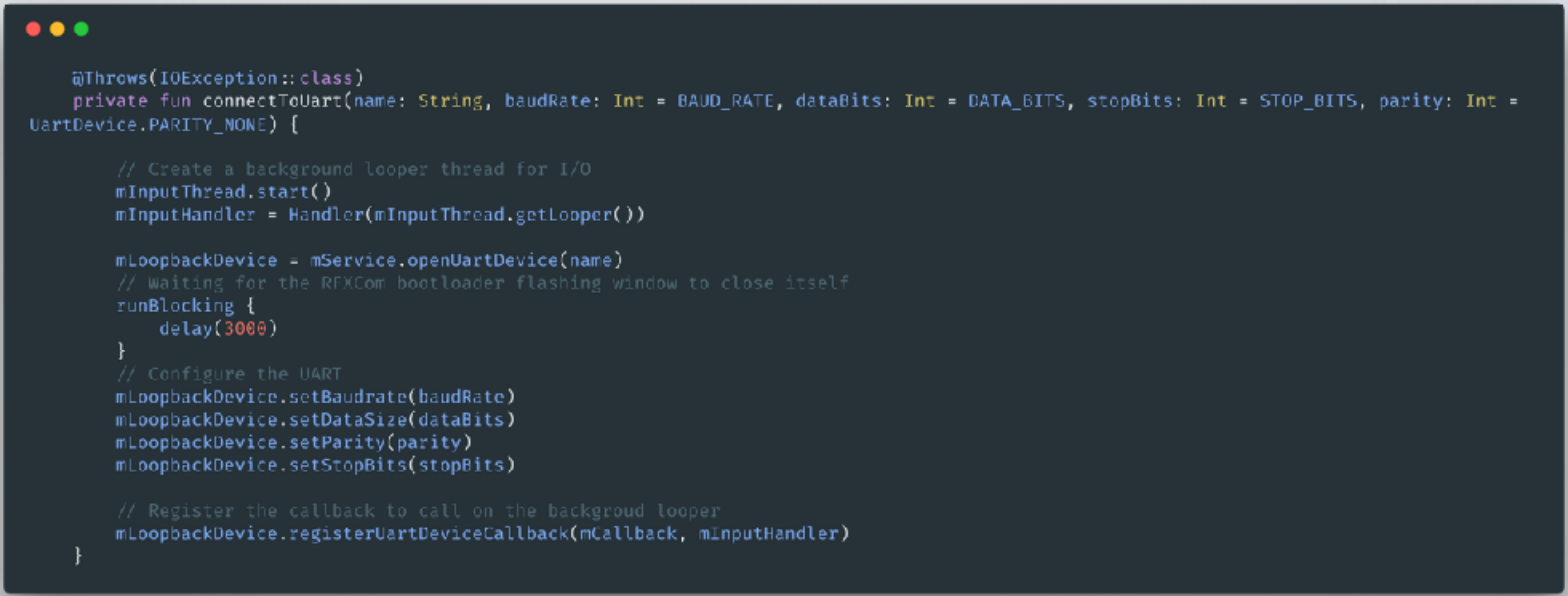
```
<resources>
    <usb-device vendor-id="0403" product-id="6001"/>
</resources>
```

LIRE LES DONNÉES



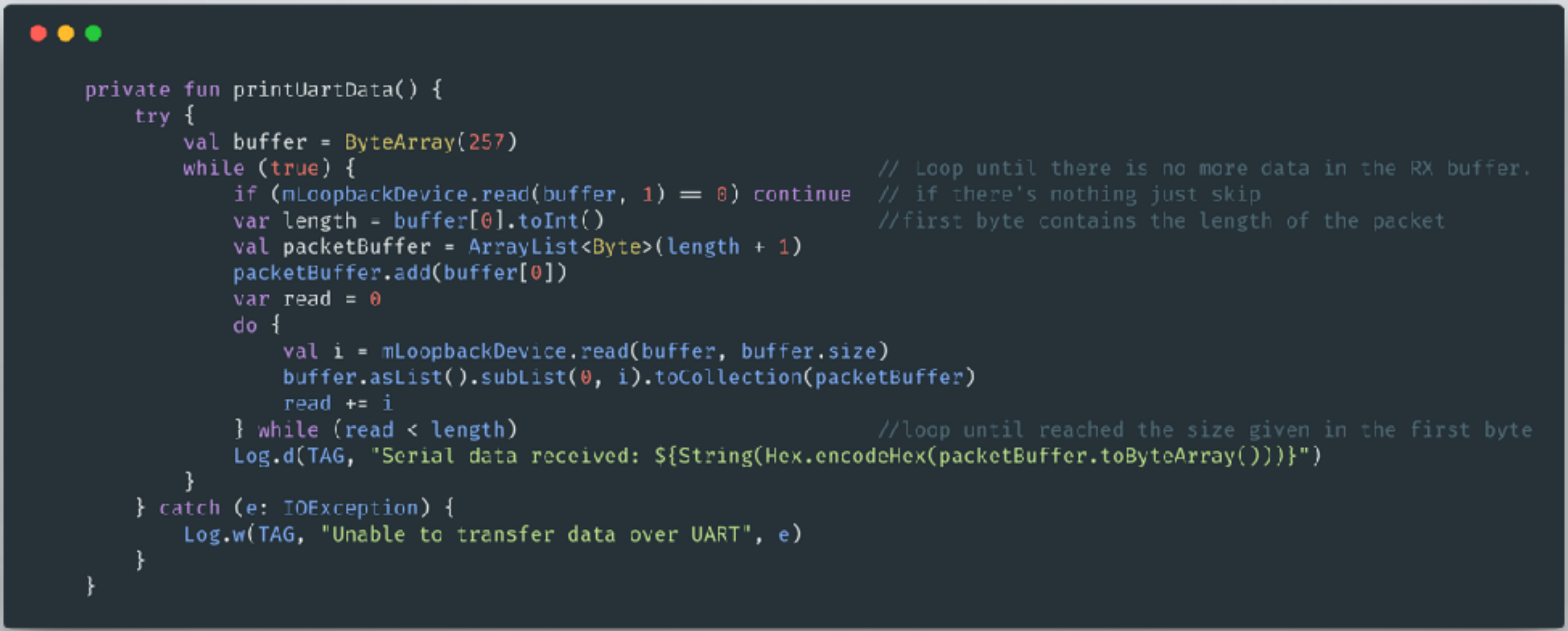
```
override fun onResume() {
    super.onResume()
    startUsbConnection()
}

private fun startUsbConnection() {
    val manager = PeripheralManagerService()
    val deviceList = manager.uartDeviceList
    if (deviceList.contains(UART_DEVICE_NAME)) connectToUart(UART_DEVICE_NAME)
    if (deviceList.isEmpty()) Log.w(TAG, "Could not start USB connection - No devices found")
}
```

LIRE LES DONNÉES



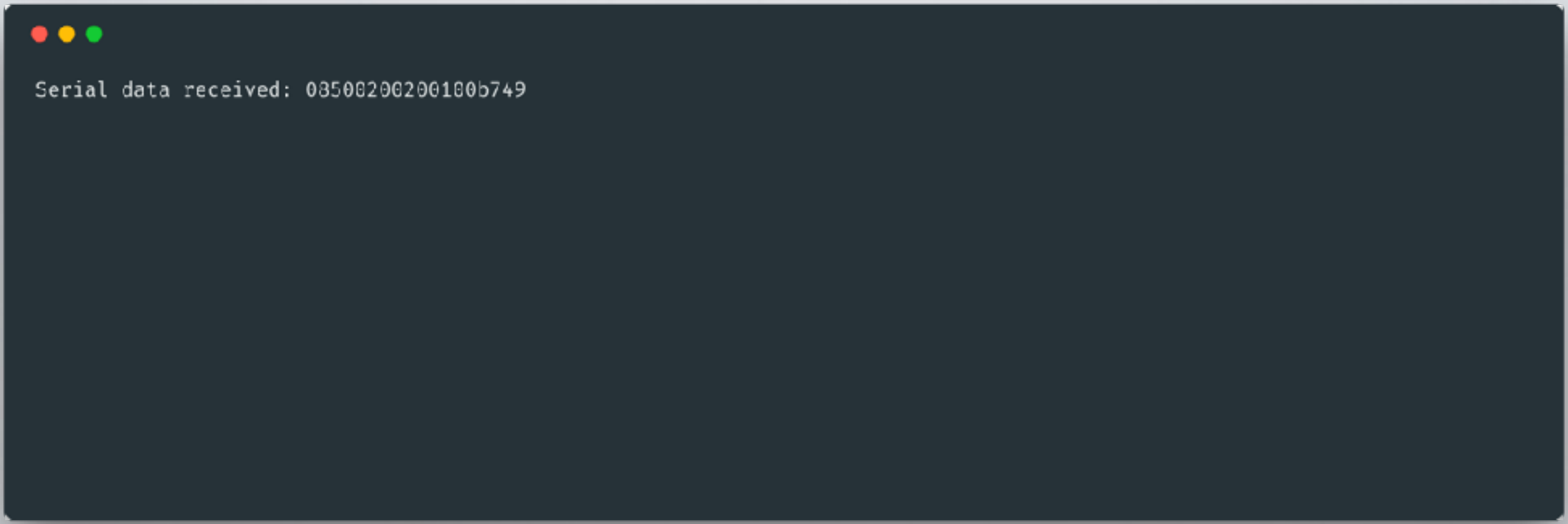
```kotlin
    @Throws(IOException::class)
    private fun connectToUart(name: String, baudRate: Int = BAUD_RATE, dataBits: Int = DATA_BITS, stopBits: Int = STOP_BITS, parity: Int =
UartDevice.PARITY_NONE) {

        // Create a background looper thread for I/O
        mInputThread.start()
        mInputHandler = Handler(mInputThread.getLooper())

        mLoopbackDevice = mService.openUartDevice(name)
        // Waiting for the RFXCom bootloader flashing window to close itself
        runBlocking {
            delay(3000)
        }
        // Configure the UART
        mLoopbackDevice.setBaudrate(baudRate)
        mLoopbackDevice.setDataSize(dataBits)
        mLoopbackDevice.setParity(parity)
        mLoopbackDevice.setStopBits(stopBits)

        // Register the callback to call on the backgroud looper
        mLoopbackDevice.registerUartDeviceCallback(mCallback, mInputHandler)
    }
```

LIRE LES DONNÉES



```
private fun printUartData() {
    try {
        val buffer = ByteArray(257)
        while (true) {                                          // Loop until there is no more data in the RX buffer.
            if (mLoopbackDevice.read(buffer, 1) == 0) continue  // if there's nothing just skip
            var length = buffer[0].toInt()                      //first byte contains the length of the packet
            val packetBuffer = ArrayList<Byte>(length + 1)
            packetBuffer.add(buffer[0])
            var read = 0
            do {
                val i = mLoopbackDevice.read(buffer, buffer.size)
                buffer.asList().subList(0, i).toCollection(packetBuffer)
                read += i
            } while (read < length)                             //loop until reached the size given in the first byte
            Log.d(TAG, "Serial data received: ${String(Hex.encodeHex(packetBuffer.toByteArray()))}")
        }
    } catch (e: IOException) {
        Log.w(TAG, "Unable to transfer data over UART", e)
    }
}
```

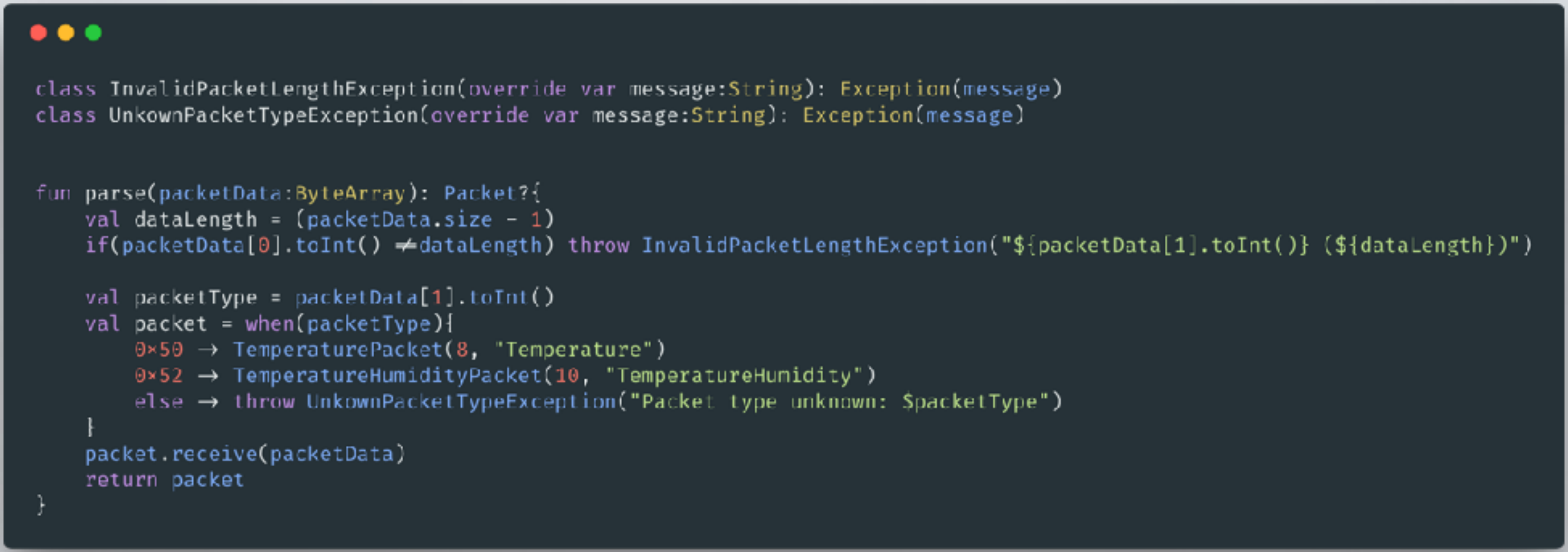
LIRE LES DONNÉES



```
Serial data received: 08500200200100b749
```

DÉCHIFFRER LES TRAMES

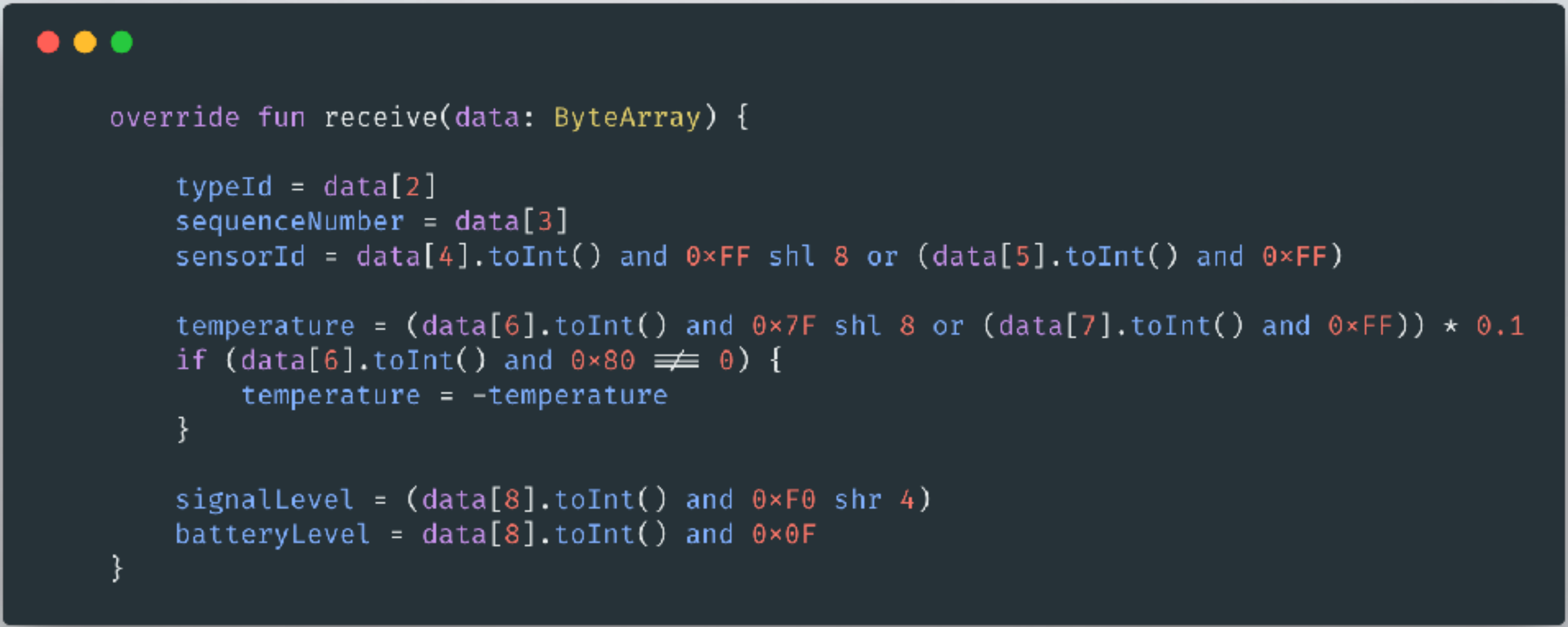
DÉCHIFFRER LES TRAMES



```kotlin
class InvalidPacketLengthException(override var message:String): Exception(message)
class UnkownPacketTypeException(override var message:String): Exception(message)


fun parse(packetData:ByteArray): Packet?{
    val dataLength = (packetData.size - 1)
    if(packetData[0].toInt() ≠dataLength) throw InvalidPacketLengthException("${packetData[1].toInt()} (${dataLength})")

    val packetType = packetData[1].toInt()
    val packet = when(packetType){
        0x50 → TemperaturePacket(8, "Temperature")
        0x52 → TemperatureHumidityPacket(10, "TemperatureHumidity")
        else → throw UnkownPacketTypeException("Packet type unknown: $packetType")
    }
    packet.receive(packetData)
    return packet
}
```

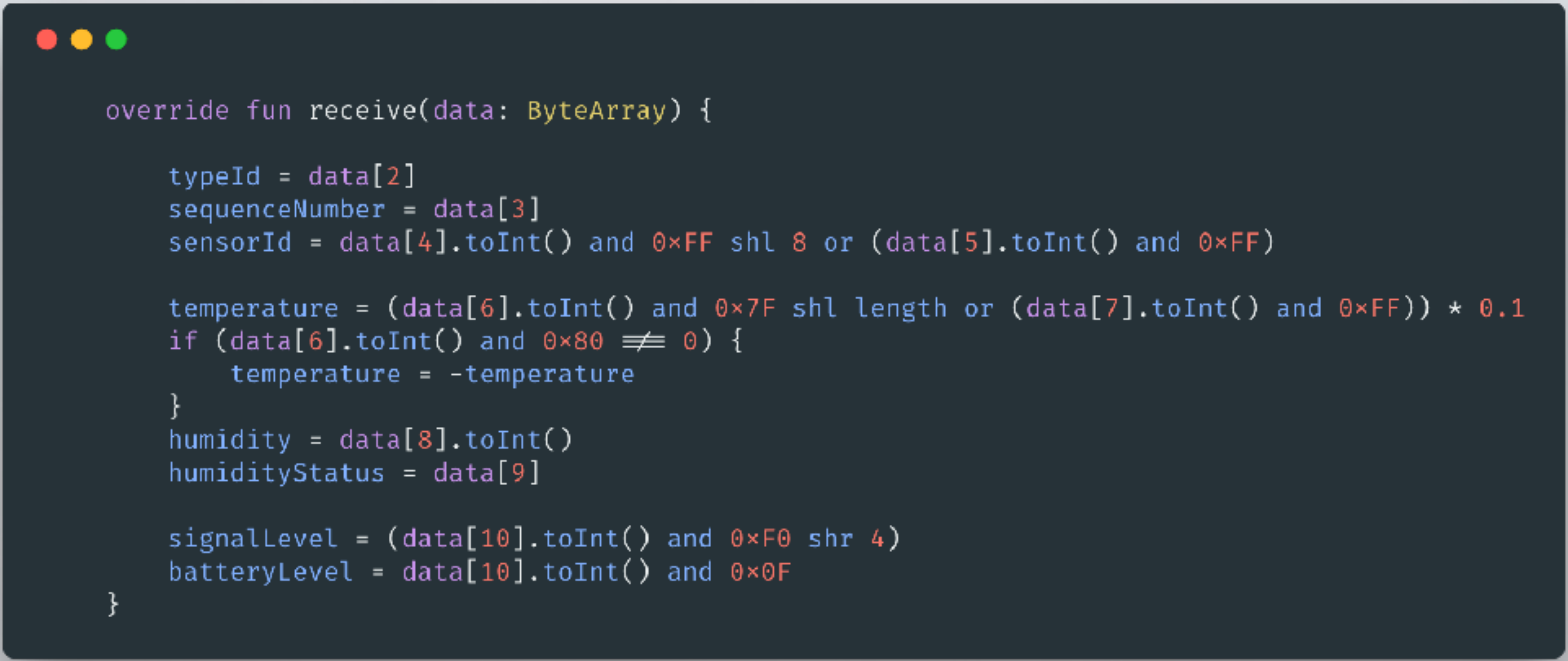
Code d'Armor

DÉCHIFFRER LES TRAMES



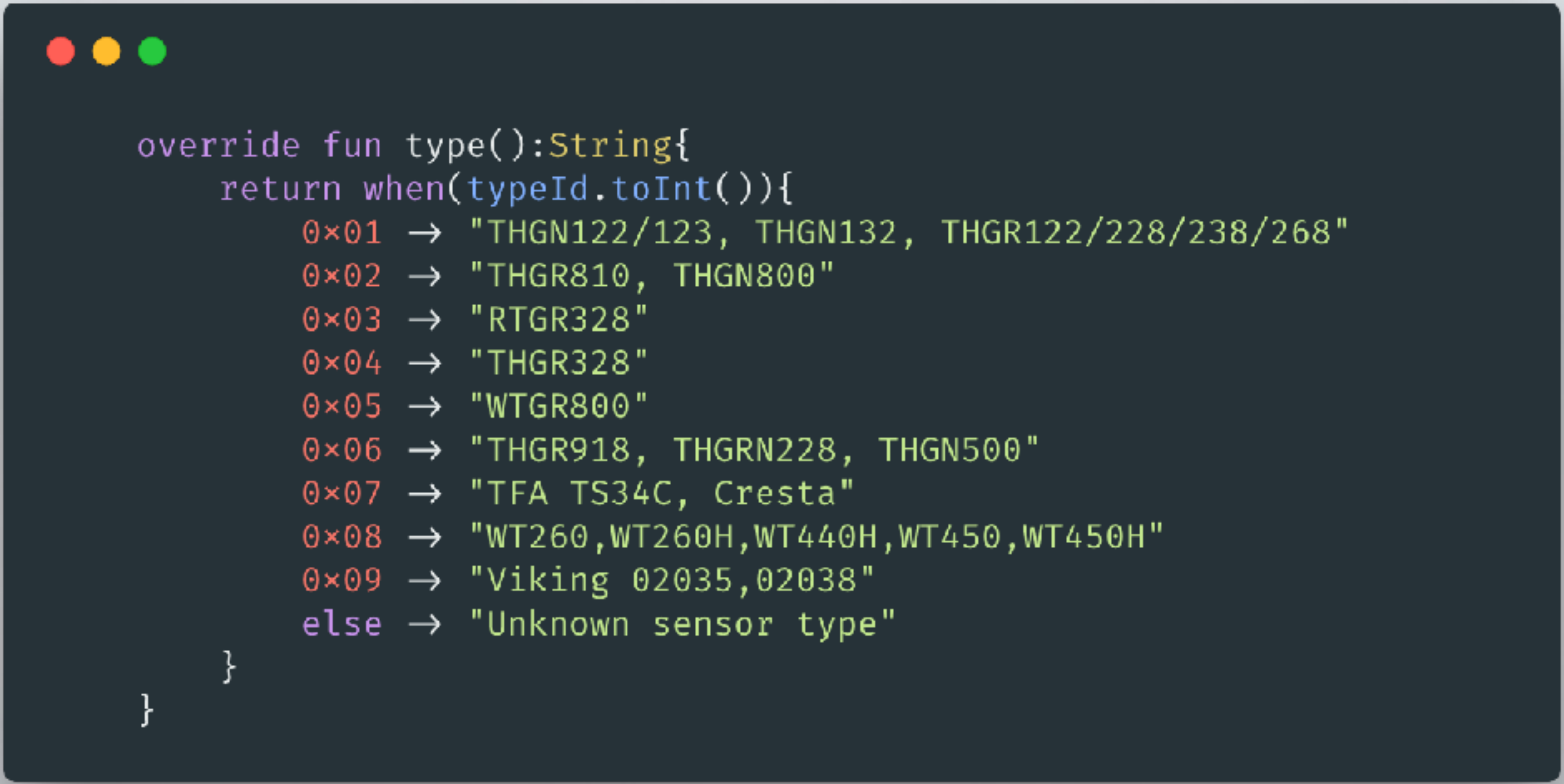
```kotlin
override fun receive(data: ByteArray) {

    typeId = data[2]
    sequenceNumber = data[3]
    sensorId = data[4].toInt() and 0xFF shl 8 or (data[5].toInt() and 0xFF)

    temperature = (data[6].toInt() and 0x7F shl 8 or (data[7].toInt() and 0xFF)) * 0.1
    if (data[6].toInt() and 0x80 !== 0) {
        temperature = -temperature
    }

    signalLevel = (data[8].toInt() and 0xF0 shr 4)
    batteryLevel = data[8].toInt() and 0x0F
}
```

DÉCHIFFRER LES TRAMES



```kotlin
override fun receive(data: ByteArray) {

    typeId = data[2]
    sequenceNumber = data[3]
    sensorId = data[4].toInt() and 0xFF shl 8 or (data[5].toInt() and 0xFF)

    temperature = (data[6].toInt() and 0x7F shl length or (data[7].toInt() and 0xFF)) * 0.1
    if (data[6].toInt() and 0x80 !== 0) {
        temperature = -temperature
    }
    humidity = data[8].toInt()
    humidityStatus = data[9]

    signalLevel = (data[10].toInt() and 0xF0 shr 4)
    batteryLevel = data[10].toInt() and 0x0F
}
```

DÉCHIFFRER LES TRAMES

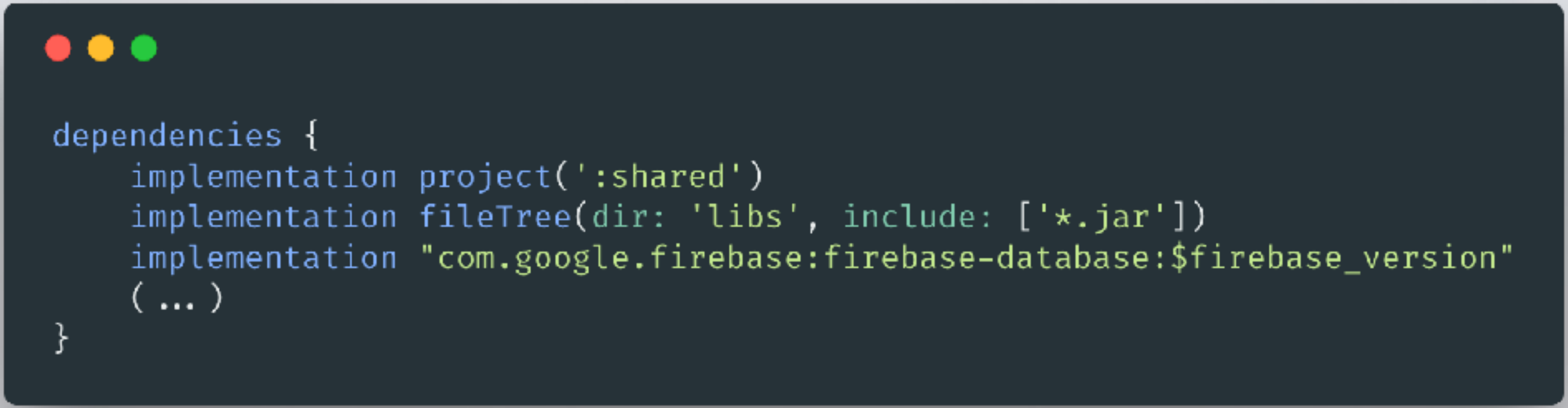


```
override fun type():String{
    return when(typeId.toInt()){
        0×01 → "THGN122/123, THGN132, THGR122/228/238/268"
        0×02 → "THGR810, THGN800"
        0×03 → "RTGR328"
        0×04 → "THGR328"
        0×05 → "WTGR800"
        0×06 → "THGR918, THGRN228, THGN500"
        0×07 → "TFA TS34C, Cresta"
        0×08 → "WT260,WT260H,WT440H,WT450,WT450H"
        0×09 → "Viking 02035,02038"
        else → "Unknown sensor type"
    }
}
```

DÉCHIFFRER LES TRAMES

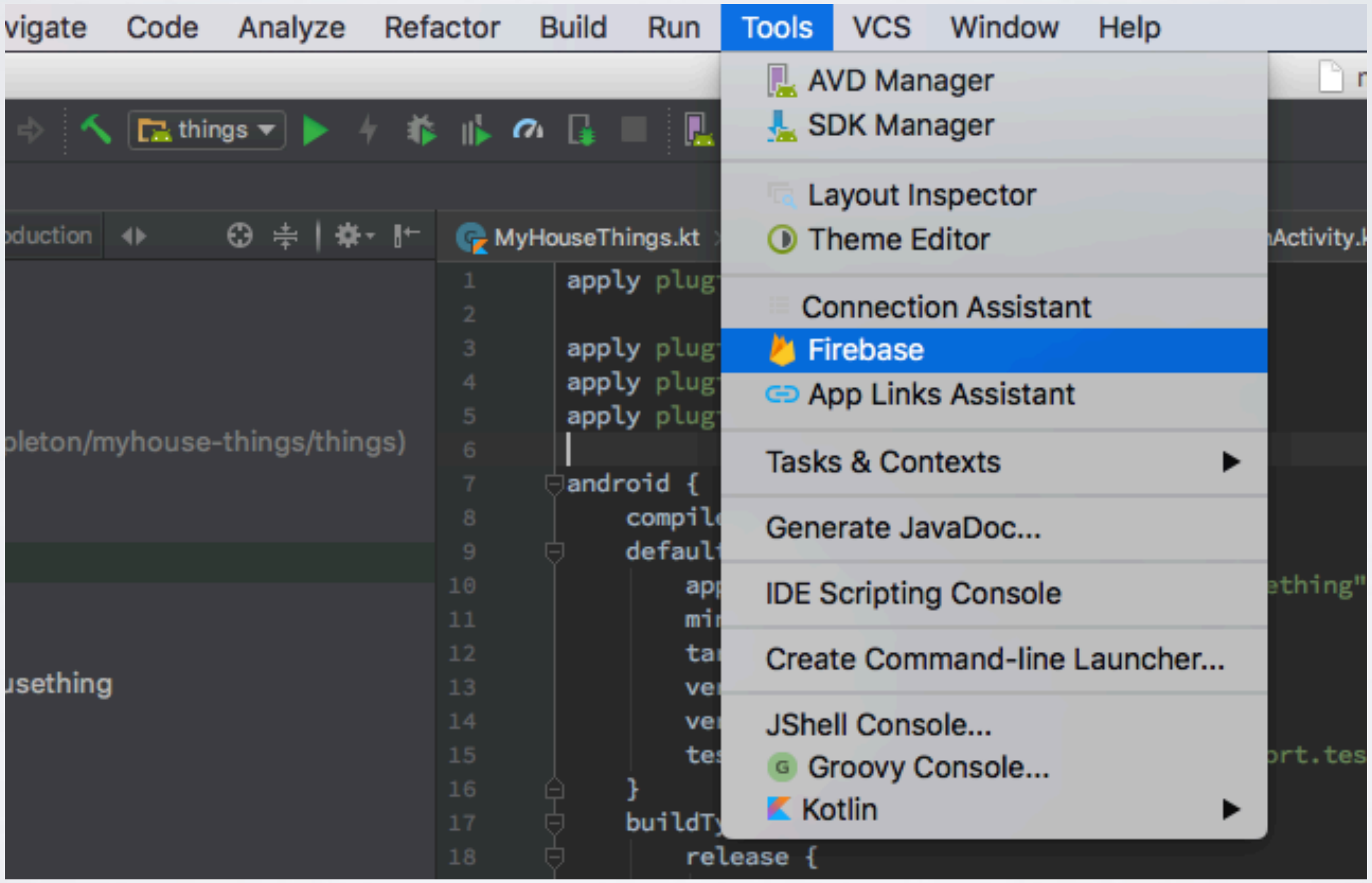
PUBLIER LES DONNÉES

PUBLIER LES DONNÉES



```
dependencies {
    implementation project(':shared')
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation "com.google.firebase:firebase-database:$firebase_version"
    ( ... )
}
```

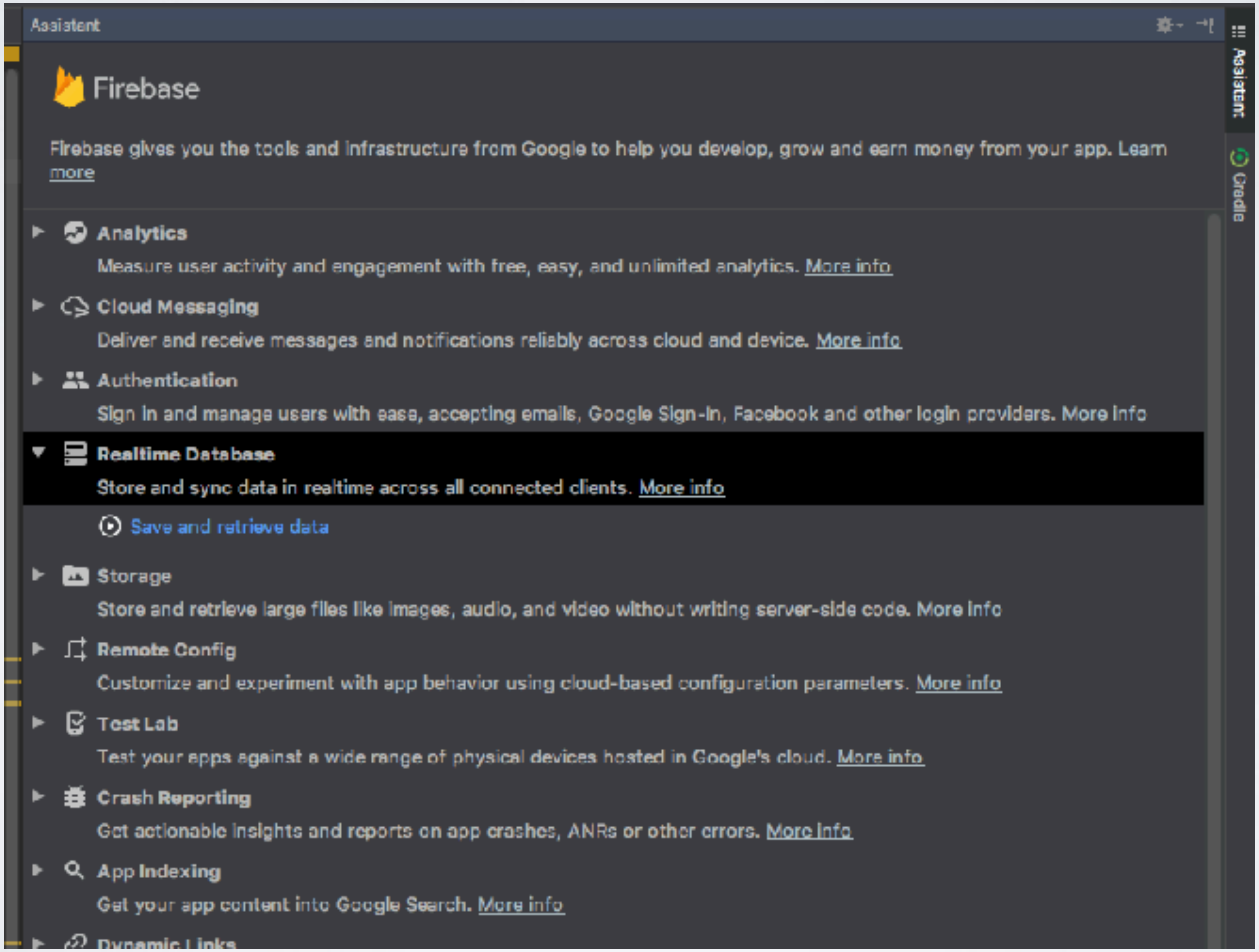
PUBLIER LES DONNÉES



```
apply plug

apply plug
apply plug
apply plug

android {
    compil
    defaul
        app
        mi
        ta
        ve
        ve
        te
    }
    buildTy
        release {
```

Code d'Armor

PUBLIER LES DONNÉES



Firebase

Firebase gives you the tools and infrastructure from Google to help you develop, grow and earn money from your app. Learn more

▶ Analytics
Measure user activity and engagement with free, easy, and unlimited analytics. More info

▶ Cloud Messaging
Deliver and receive messages and notifications reliably across cloud and device. More info

▶ Authentication
Sign in and manage users with ease, accepting emails, Google Sign-In, Facebook and other login providers. More info

▼ Realtime Database
Store and sync data in realtime across all connected clients. More info

Save and retrieve data

▶ Storage
Store and retrieve large files like images, audio, and video without writing server-side code. More info

▶ Remote Config
Customize and experiment with app behavior using cloud-based configuration parameters. More info

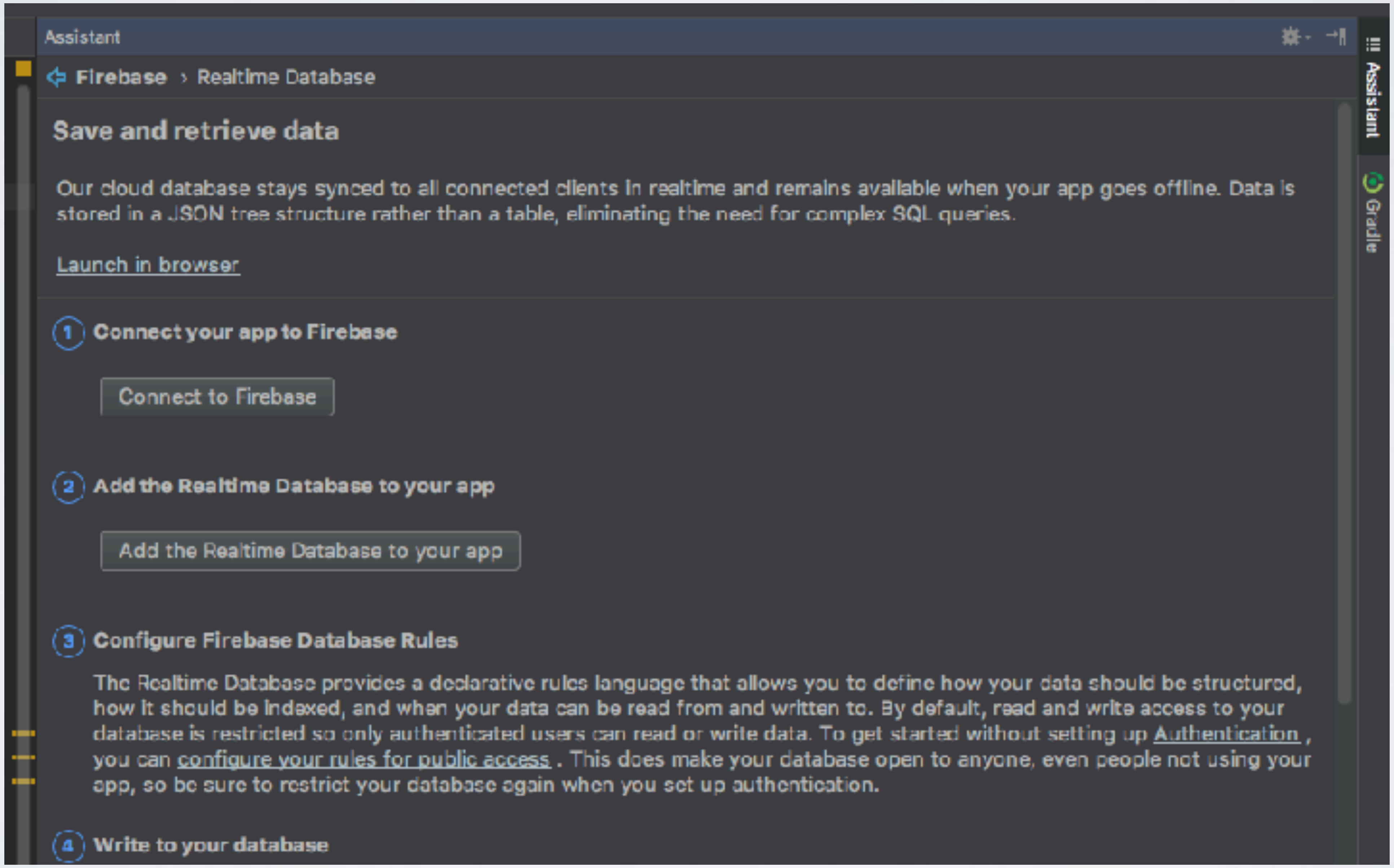
▶ Test Lab
Test your apps against a wide range of physical devices hosted in Google's cloud. More info

▶ Crash Reporting
Get actionable insights and reports on app crashes, ANRs or other errors. More info

▶ App Indexing
Get your app content into Google Search. More info

▶ Dynamic Links

PUBLIER LES DONNÉES



Firebase › Realtime Database

Save and retrieve data

Our cloud database stays synced to all connected clients in realtime and remains available when your app goes offline. Data is stored in a JSON tree structure rather than a table, eliminating the need for complex SQL queries.

Launch in browser

1. **Connect your app to Firebase**

   Connect to Firebase

2. **Add the Realtime Database to your app**

   Add the Realtime Database to your app

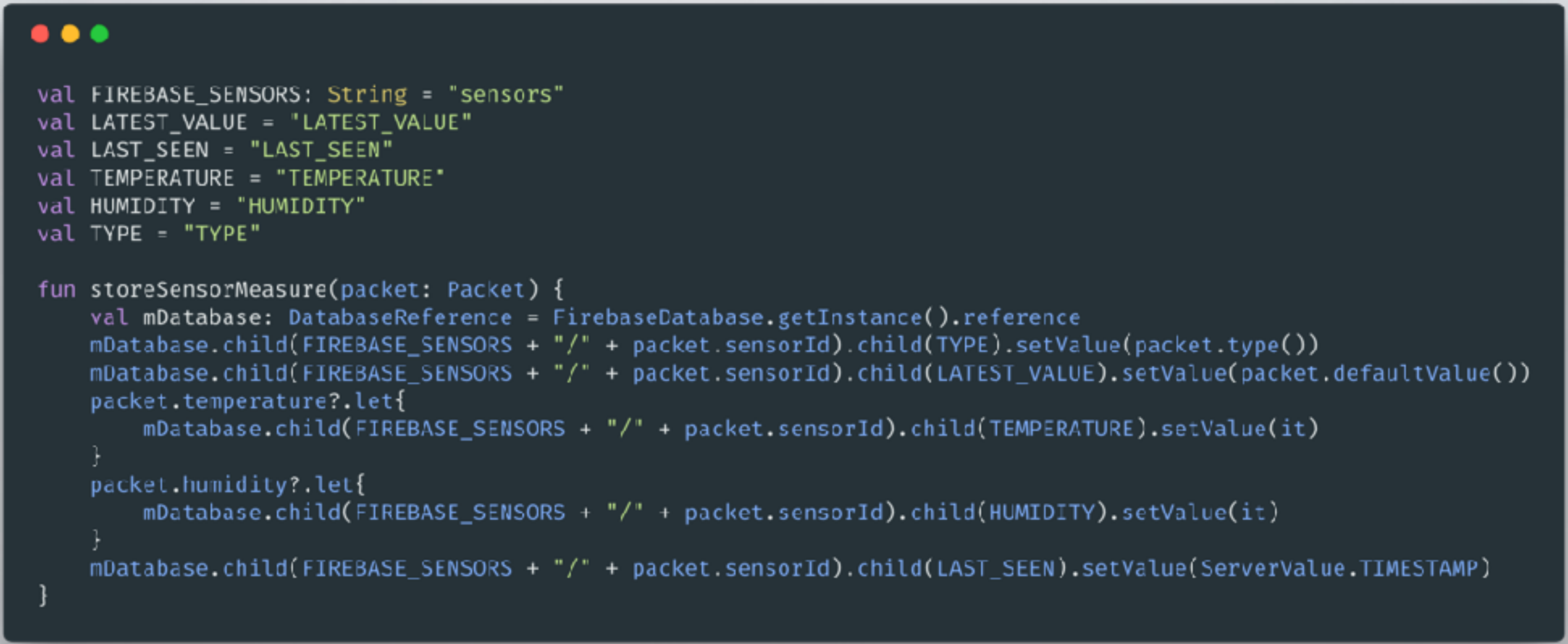
3. **Configure Firebase Database Rules**

   The Realtime Database provides a declarative rules language that allows you to define how your data should be structured, how it should be indexed, and when your data can be read from and written to. By default, read and write access to your database is restricted so only authenticated users can read or write data. To get started without setting up Authentication , you can configure your rules for public access . This does make your database open to anyone, even people not using your app, so be sure to restrict your database again when you set up authentication.

4. **Write to your database**

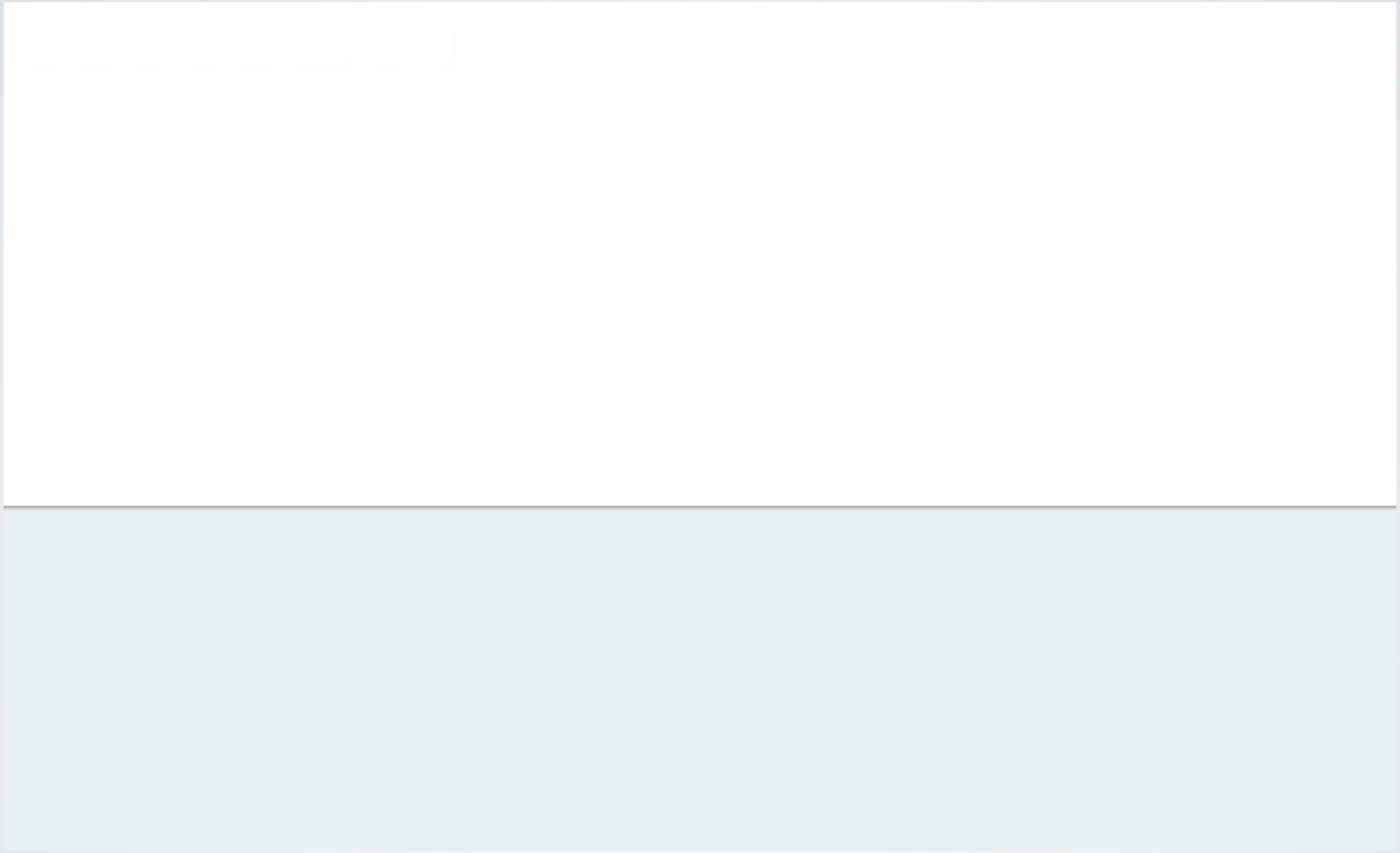
Code d'Armor

PUBLIER LES DONNÉES



```kotlin
val FIREBASE_SENSORS: String = "sensors"
val LATEST_VALUE = "LATEST_VALUE"
val LAST_SEEN = "LAST_SEEN"
val TEMPERATURE = "TEMPERATURE"
val HUMIDITY = "HUMIDITY"
val TYPE = "TYPE"

fun storeSensorMeasure(packet: Packet) {
    val mDatabase: DatabaseReference = FirebaseDatabase.getInstance().reference
    mDatabase.child(FIREBASE_SENSORS + "/" + packet.sensorId).child(TYPE).setValue(packet.type())
    mDatabase.child(FIREBASE_SENSORS + "/" + packet.sensorId).child(LATEST_VALUE).setValue(packet.defaultValue())
    packet.temperature?.let{
        mDatabase.child(FIREBASE_SENSORS + "/" + packet.sensorId).child(TEMPERATURE).setValue(it)
    }
    packet.humidity?.let{
        mDatabase.child(FIREBASE_SENSORS + "/" + packet.sensorId).child(HUMIDITY).setValue(it)
    }
    mDatabase.child(FIREBASE_SENSORS + "/" + packet.sensorId).child(LAST_SEEN).setValue(ServerValue.TIMESTAMP)
}
```

PUBLIER LES DONNÉES



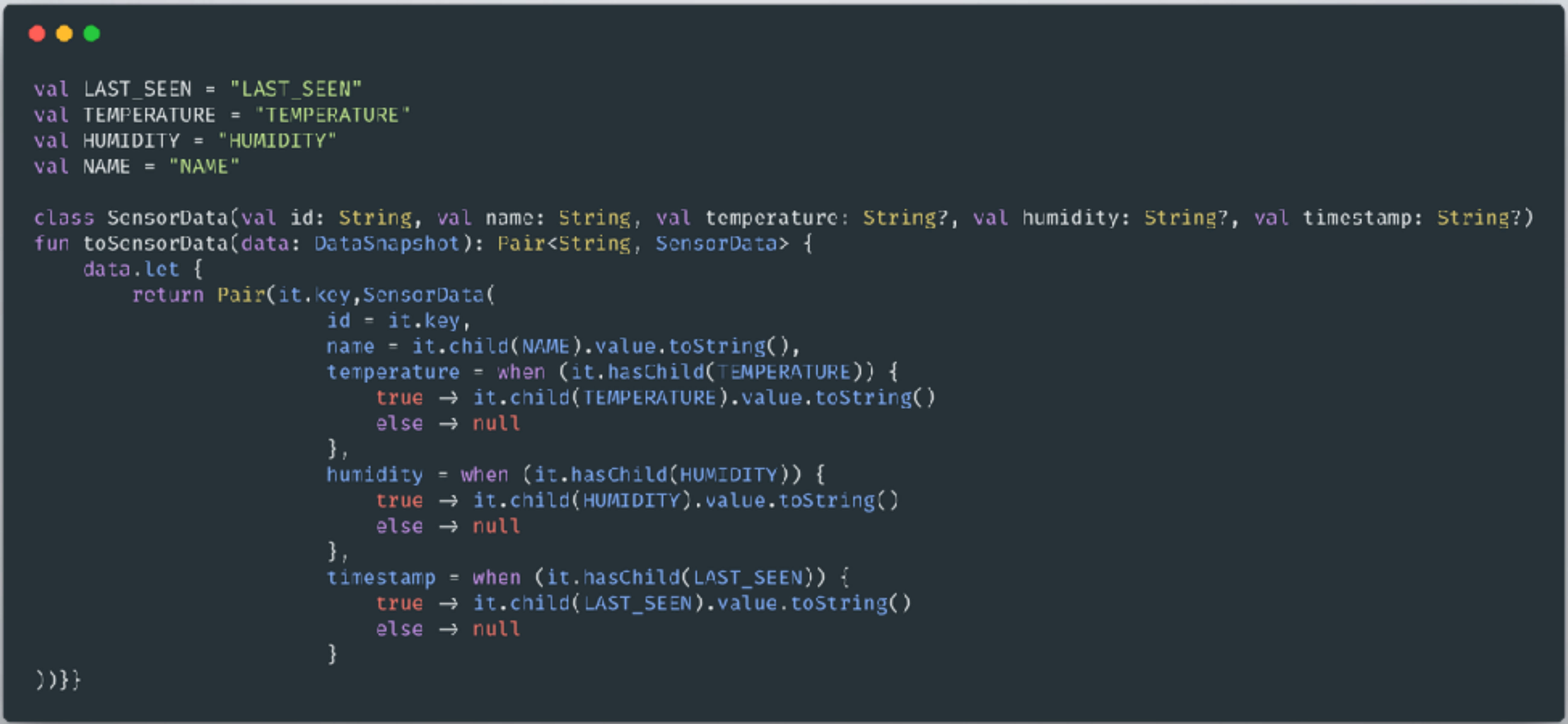
COMPANION

Code
d'Armor

COMPANION

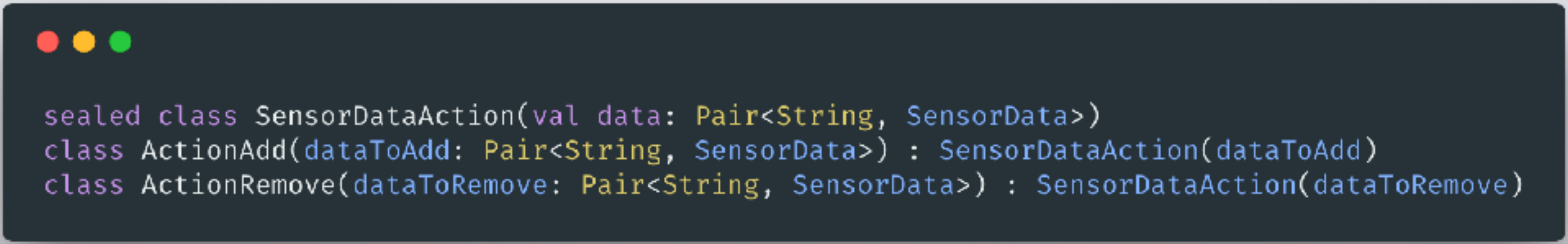
- LiveData
- ViewModel

COMPANION



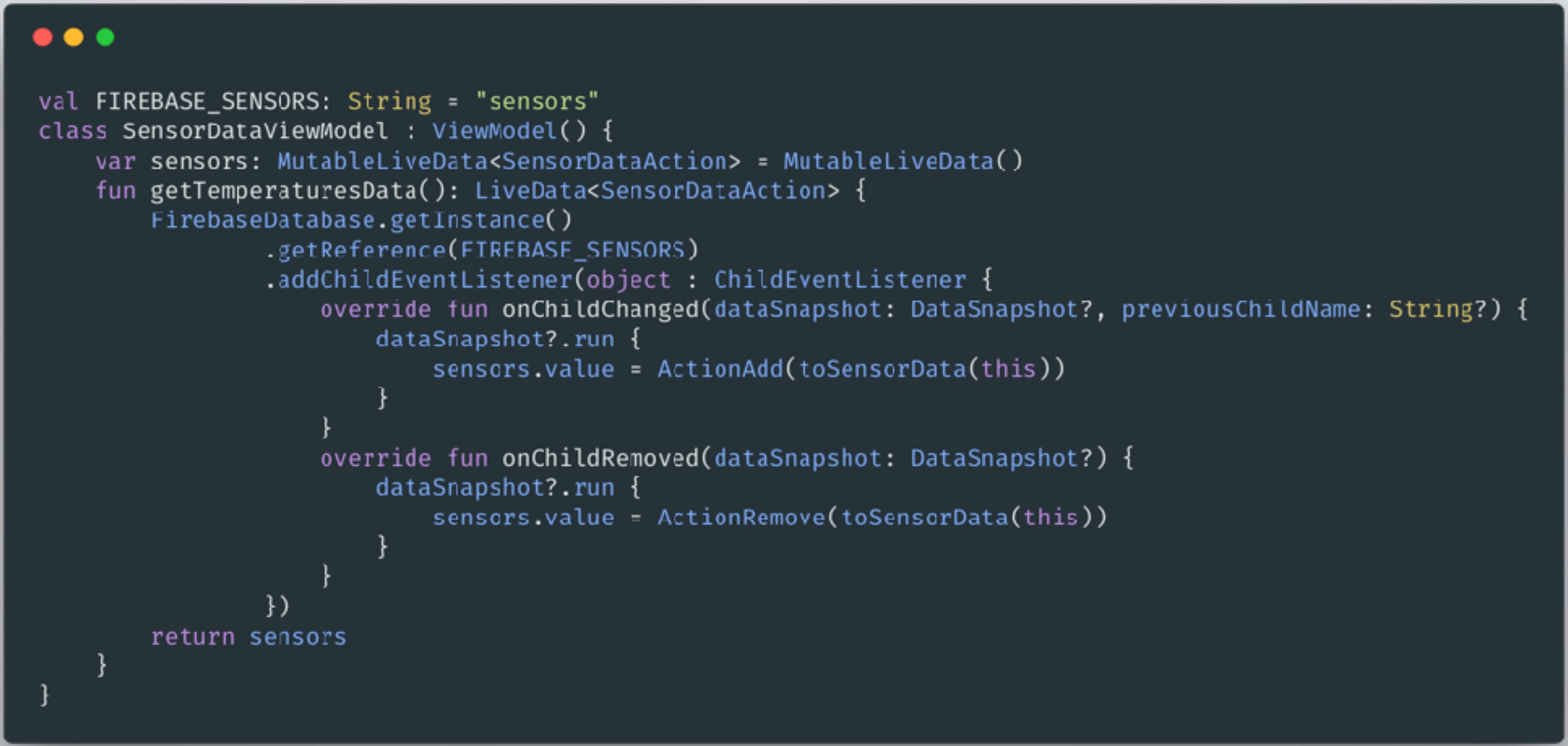
```kotlin
val LAST_SEEN = "LAST_SEEN"
val TEMPERATURE = "TEMPERATURE"
val HUMIDITY = "HUMIDITY"
val NAME = "NAME"

class SensorData(val id: String, val name: String, val temperature: String?, val humidity: String?, val timestamp: String?)
fun toSensorData(data: DataSnapshot): Pair<String, SensorData> {
    data.let {
        return Pair(it.key,SensorData(
                        id = it.key,
                        name = it.child(NAME).value.toString(),
                        temperature = when (it.hasChild(TEMPERATURE)) {
                            true → it.child(TEMPERATURE).value.toString()
                            else → null
                        },
                        humidity = when (it.hasChild(HUMIDITY)) {
                            true → it.child(HUMIDITY).value.toString()
                            else → null
                        },
                        timestamp = when (it.hasChild(LAST_SEEN)) {
                            true → it.child(LAST_SEEN).value.toString()
                            else → null
                        }
))}}
```

COMPANION



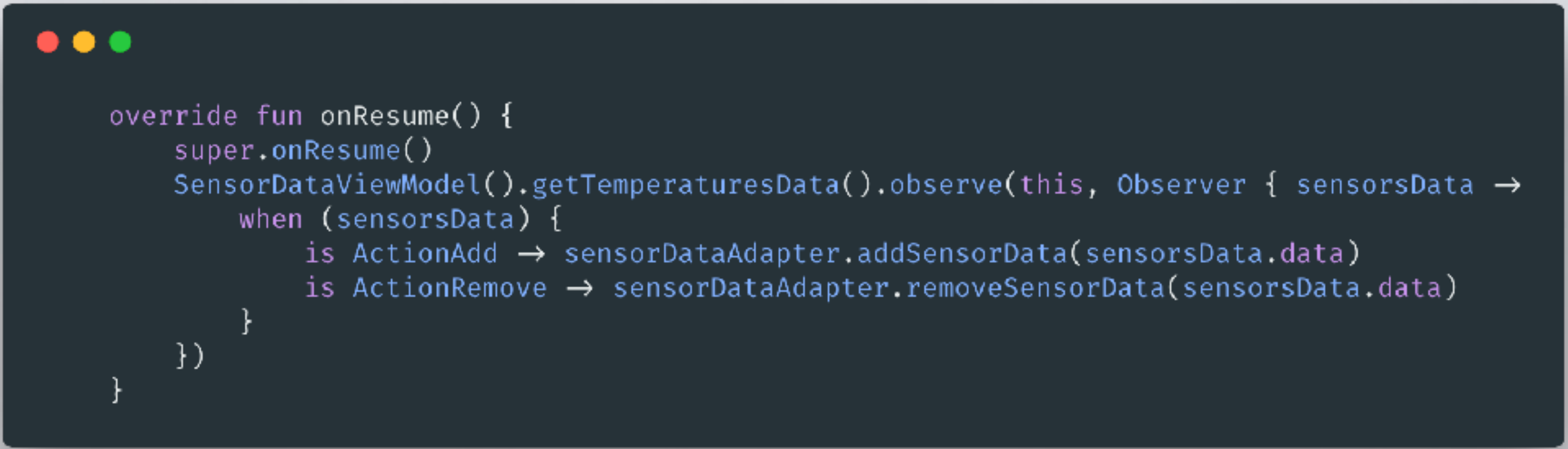
```kotlin
sealed class SensorDataAction(val data: Pair<String, SensorData>)
class ActionAdd(dataToAdd: Pair<String, SensorData>) : SensorDataAction(dataToAdd)
class ActionRemove(dataToRemove: Pair<String, SensorData>) : SensorDataAction(dataToRemove)
```

COMPANION



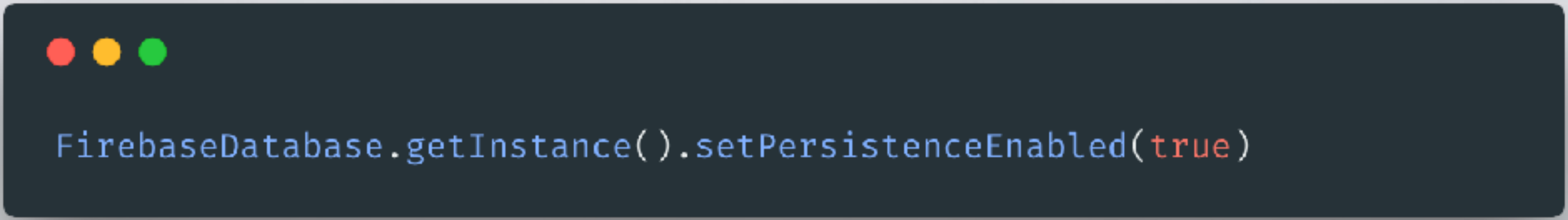
```kotlin
val FIREBASE_SENSORS: String = "sensors"
class SensorDataViewModel : ViewModel() {
    var sensors: MutableLiveData<SensorDataAction> = MutableLiveData()
    fun getTemperaturesData(): LiveData<SensorDataAction> {
        FirebaseDatabase.getInstance()
                .getReference(FIREBASE_SENSORS)
                .addChildEventListener(object : ChildEventListener {
                    override fun onChildChanged(dataSnapshot: DataSnapshot?, previousChildName: String?) {
                        dataSnapshot?.run {
                            sensors.value = ActionAdd(toSensorData(this))
                        }
                    }
                    override fun onChildRemoved(dataSnapshot: DataSnapshot?) {
                        dataSnapshot?.run {
                            sensors.value = ActionRemove(toSensorData(this))
                        }
                    }
                })
        return sensors
    }
}
```

COMPANION



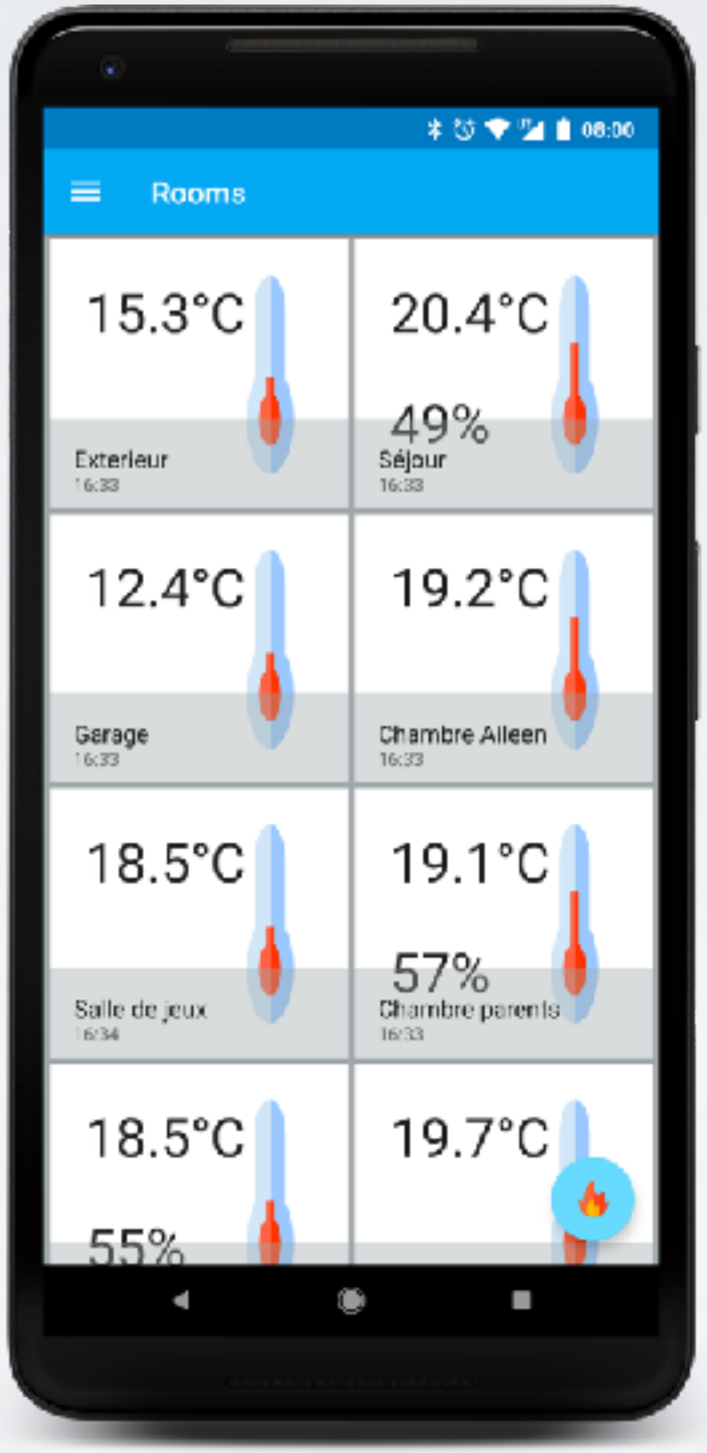
```
override fun onResume() {
    super.onResume()
    SensorDataViewModel().getTemperaturesData().observe(this, Observer { sensorsData →
        when (sensorsData) {
            is ActionAdd → sensorDataAdapter.addSensorData(sensorsData.data)
            is ActionRemove → sensorDataAdapter.removeSensorData(sensorsData.data)
        }
    })
}
```

COMPANION



```
FirebaseDatabase.getInstance().setPersistenceEnabled(true)
```

COMPANION


08:00

Rooms

15.3°C
Exterieur
16:33

20.4°C
49%
Séjour
16:33

12.4°C
Garage
16:33

19.2°C
Chambre Aileen
16:33

18.5°C
Salle de jeux
16:34

19.1°C
57%
Chambre parents
16:33

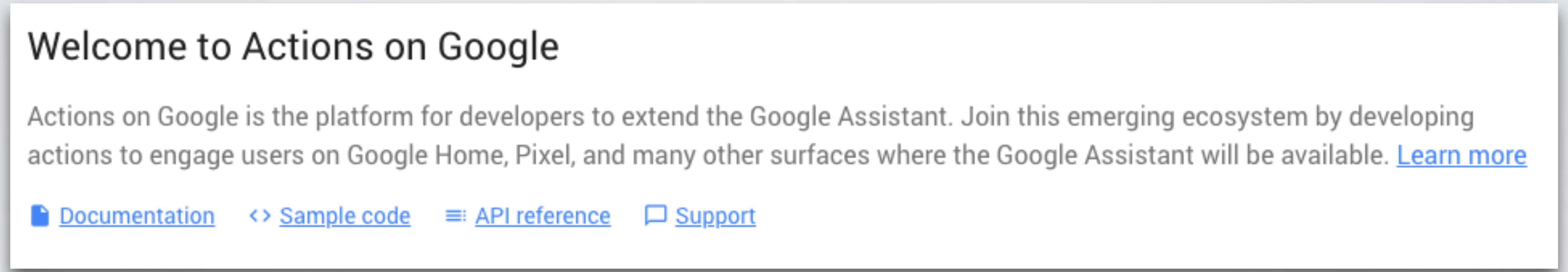
18.5°C
55%

19.7°C

OK GOOGLE?

Code d'Armor

OK GOOGLE?

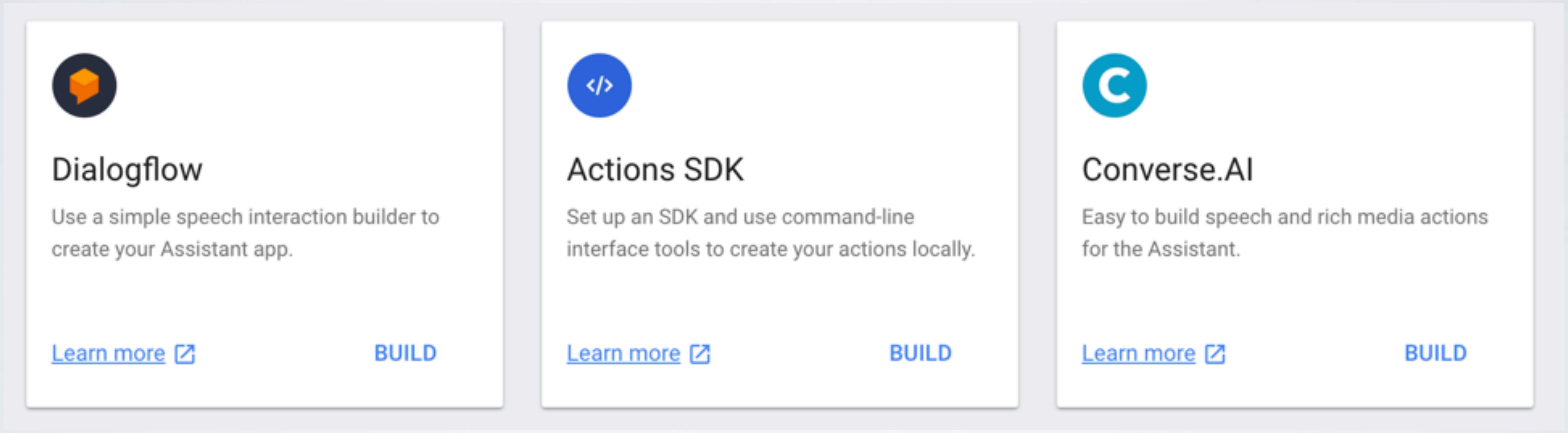


Welcome to Actions on Google

Actions on Google is the platform for developers to extend the Google Assistant. Join this emerging ecosystem by developing actions to engage users on Google Home, Pixel, and many other surfaces where the Google Assistant will be available. Learn more

https://console.actions.google.com

OK GOOGLE?



Dialogflow

Use a simple speech interaction builder to create your Assistant app.

Learn more BUILD

Actions SDK

Set up an SDK and use command-line interface tools to create your actions locally.

Learn more BUILD

Converse.AI

Easy to build speech and rich media actions for the Assistant.

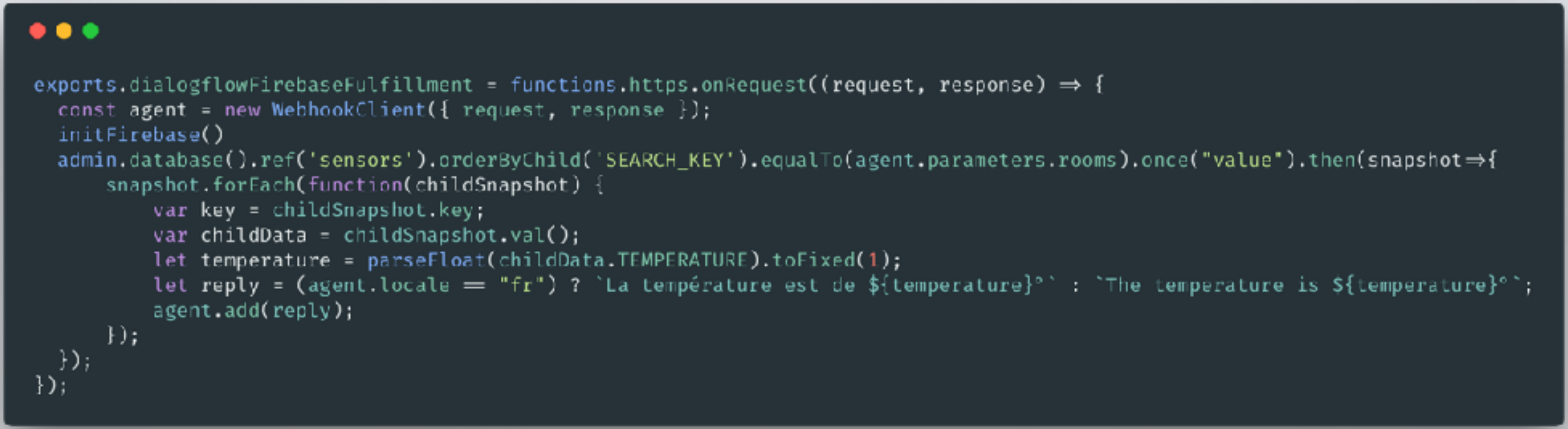
Learn more BUILD

OK GOOGLE?

- Fullfilment Webhooks
- Firebase Cloud Functions

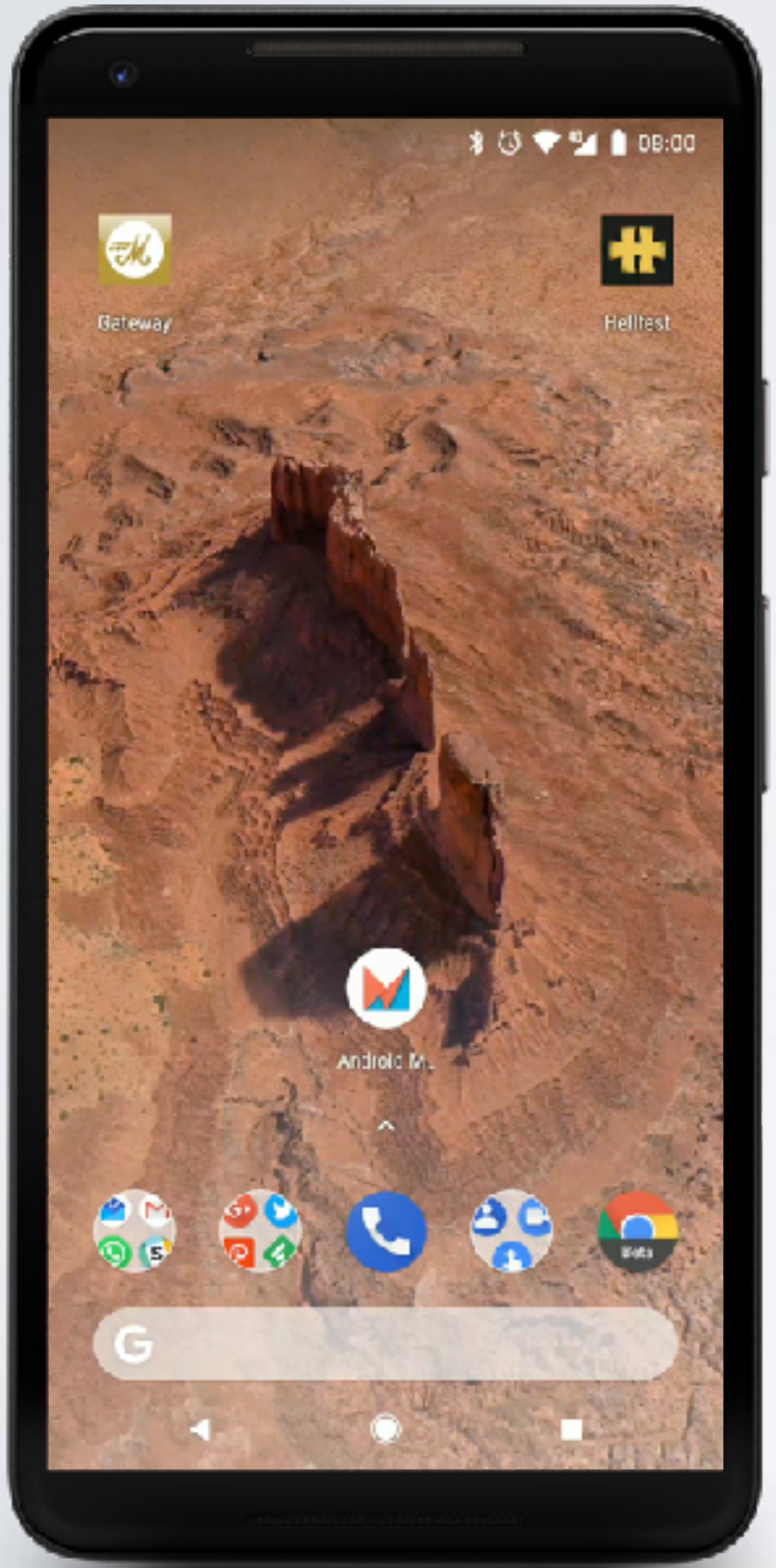
Code d'Armor

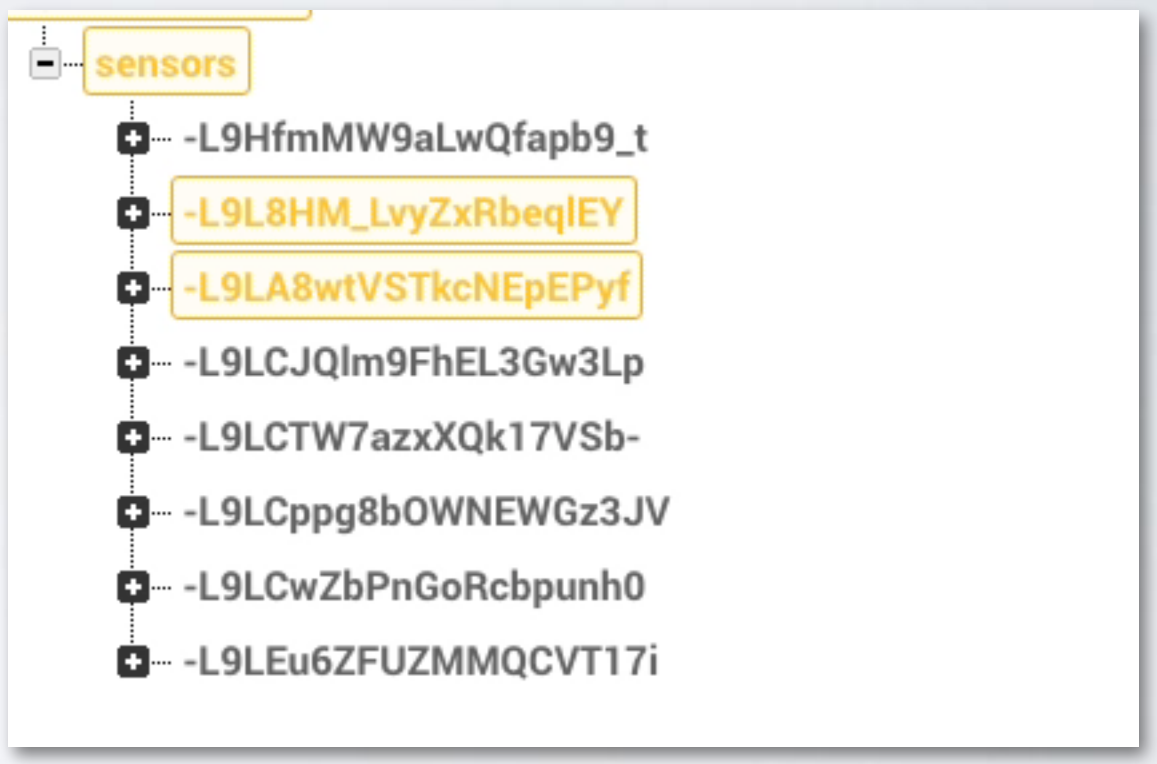
OK GOOGLE?



```
exports.dialogflowFirebaseFulfillment = functions.https.onRequest((request, response) ⇒ {
  const agent = new WebhookClient({ request, response });
  initFirebase()
  admin.database().ref('sensors').orderByChild('SEARCH_KEY').equalTo(agent.parameters.rooms).once("value").then(snapshot⇒{
      snapshot.forEach(function(childSnapshot) {
          var key = childSnapshot.key;
          var childData = childSnapshot.val();
          let temperature = parseFloat(childData.TEMPERATURE).toFixed(1);
          let reply = (agent.locale == "fr") ? `La température est de ${temperature}°` : `The temperature is ${temperature}°`;
          agent.add(reply);
      });
  });
});
```

OK GOOGLE?

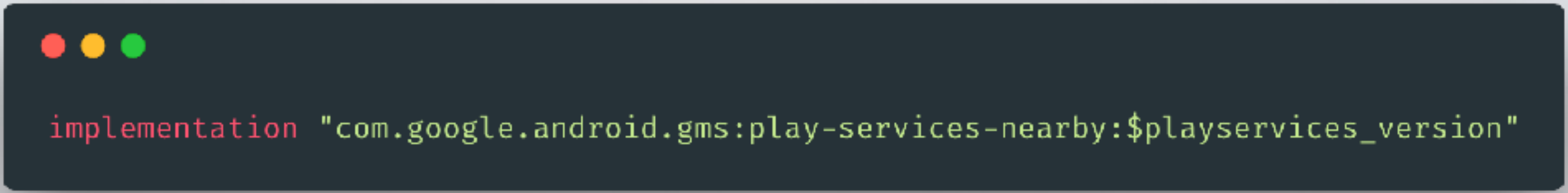




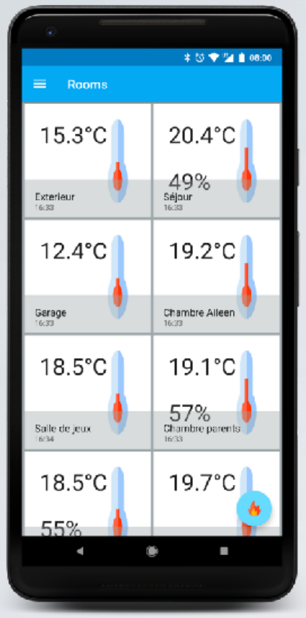
```
sensors
  -L9HfmMW9aLwQfapb9_t
  -L9L8HM_LvyZxRbeqlEY
  -L9LA8wtVSTkcNEpEPyf
  -L9LCJQlm9FhEL3Gw3Lp
  -L9LCTW7azxXQk17VSb-
  -L9LCppg8bOWNEWGz3JV
  -L9LCwZbPnGoRcbpunh0
  -L9LEu6ZFUZMMQCVT17i
```

SETTINGS

NEARBY API



```
implementation "com.google.android.gms:play-services-nearby:$playservices_version"
```



08:00

Rooms

15.3°C Exterieur 16:33	20.4°C 49% Séjour 16:33
12.4°C Garage 16:33	19.2°C Chambre Aileen 16:33
18.5°C Salle de jeux 16:34	19.1°C 57% Chambre parents 16:33
18.5°C 55%	19.7°C

Firebase

Code d'Armor

Code
d'Armor