Guillem Frisach Pedrola
Francesc Ferré Tarrés

# Aplicacions Mòbils i Encastades

Documentació Pràctica 2 i 3

# Índex

# 1 Estructura de la pràctica ANDROID

Aquesta pràctica consisteix en la realització del joc de memòria popular anomenat *Simon Says* per Android.

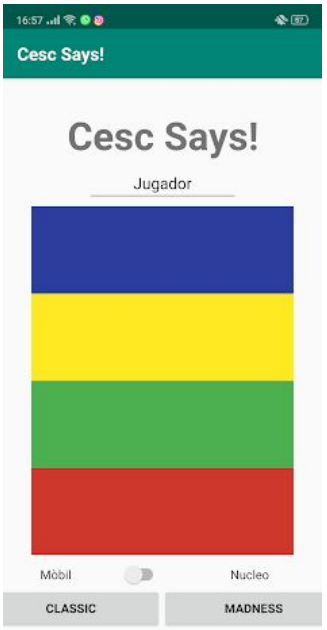
Un cop ja s'ha creat el joc, aquest ha de connectar-se per Bluetooth a la placa i ha de il·luminar el LED que indica el programa.

A més a més, s'inclou un top 5 dels millor jugadors que han jugat en aquell mòbil corresponent, independentment de si l'aplicació roman activa o no.

Els enllaços als projectes són els següents:

| **Nom projecte** | **Enllaç al projecte** |
|---|---|
| Practica2_AME (Android) | https://github.com/CescFT/Practica2_AME/ |
| Practica3AME (Android + MBED) | https://github.com/CescFT/Practica3AME/ |

## 1.1 Decisions de disseny ANDROID




El nostre joc està generat per dos layouts diferents, controlats per dos activitats diferents, el primer layout et dóna l'opció d'escriure el nom del jugador actual i et permet escollir entre dos modes de joc:

- **Clàssic**: En aquest mode el joc es comporta com sempre ha funcionat. Va passant de nivells i en cadascun dels nivells s'ha de repetir la seqüència del nivell anterior, i així successivament.
- **Mode "Madness"**: Aquest mode es divideix en tres nivells diferents. Aquest mode, en trets generals, va donant seqüències de colors, sense dependència entre fases, és a dir, que cada seqüència es genera nova i de manera aleatòria al principi de cada fase. Depenent del nivell escollit (1, 2 o 3), apareixen més o menys colors en la seqüència.

Hem decidit implementar dos modes de joc, per tal de que la nostra pràctica comptés amb un valor afegit. No és una funcionalitat complexa, ni "revolucionària" però clarament és diferent de la manera més clàssica de jugar al *Simon Says.*


En cas que el jugador selecciona l'opció de joc "**Madness**", et permet escollir els diferents nivells, i l'aparença és la següent (Veure imatge al marge esquerra del paràgraf).

Cada nivell, és la representació del nombre d'increments que hi ha per ronda, és a dir que al nivell 1, cada ronda s'afegeix un color a la seqüència, al 2, se n'afegeixen dos, i al nivell 3, s'afegeixen de tres en tres.

La decisió d'implementar nivells en "madness" i no en el classic, és purament per a mantenir el mode clàssic tal com el seu nom indica.

En el mode clàssic, si el jugador s'equivoca, es redirecciona directament a la segona activitat, que és el Ranking de millors jugadors. Si l'usuari no indica el seu nom a la pantalla principal del joc, el nom predeterminat, serà "Jugador".

S'ha decidit implementar el rànquing només en el mode classic ja que el mode "madness" compta amb diferents nivells i diferents modes de puntuació. Això hagués requerit fer un rànquing amb cada una de les diferents dificultats, i vam considerar que seria un embolic tant per a la programació de l'aplicació com per a l'enteniment del jugador, ja que hauria de ser conscient de en quin ranking està.

Aquesta activitat és la que mostra el ranking, l'aparença que té aquest layout és el següent:


Per tal de mantenir persistida la informació dins de l'aplicació, es fa ús de les ***Shared Preferences***. Un objecte SharedPreferences apunta a un fitxer que conté parells de valor de clau i proporciona mètodes senzills per llegir-los i escriure-los. Cada fitxer SharedPreferences pot ser privat o compartit.

Una decisió de disseny presa en aquest punt, va ser procurar reduir al màxim l'espai que usem dins de les Shared Prefernces. Si no ho haguessim controlat, la mida del rànquig hagués anat augmentant incontroladament amb els resultats de les partides jugades. Per tal de controlar la mida d'aquesta llista on es desen les puntuacions, el que hem fet ha estat desar només els 5 millors resultats. D'aquesta manera sempre tindrem acotat el nombre de resultats que guardem, i la mida de la llista. Així doncs ens preocupem de la optimització de la memòria del dispositiu Android del jugador final.

El problema de les ***Shared Preferences*** és que en aquest és complicat emmagatzemar una llista de valors.

Per tal de mantenir emmagatzemada la llista, s'ha fet ús del llenguatge interoperable JSON (JavaScript Object Notation). Al acabar la partida, el que es fa és emmagatzemar el nou jugador que acaba de jugar.

Per mantenir el top 5 ben ordenat, el que s'ha fet és generar un *custom object* que representa un jugador (conté el nom i la puntuació obtinguda).

Aquesta classe implementa la Serialització (per tal que funcioni el JSON) i la interfície Comparable<T>. D'aquesta forma, s'ha d'implementar el mètode compareTo() i podem mantenir la llista ordenada fent ús de la funció sort() de Collections:

```
Collections.sort(List<T>);
```

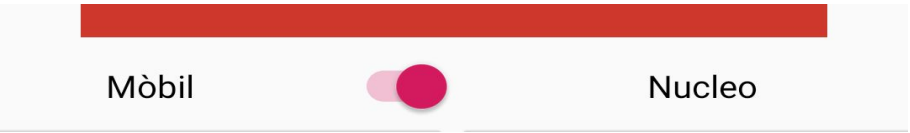
Tot i que la llista queda ordenada, el sentit d'aquesta, és descendent (la puntuació més petita és la primera). Per tal de capgirar-ho, i ordenar-ho com a nosaltres més ens convé, el que es fa és cridar de nou a Collections, amb el mètode reverse():

```
Collections.reverse(List<T>);
```

Així doncs, el ranking queda ben ordenat.

Finalment, el botó de tornar, retorna a la pantalla principal i permet iniciar una nova partida, conservant el nom de jugador.

En cas que abans d'iniciar la partida, l'usuari activi el switch que hi ha en el layout principal, ens permet que la placa reprodueixi el mateix que va passant al joc (encenent LEDS). Per a que funcioni, el dispositiu amb el que juguem, haurà d'estar prèviament vinculat amb el bluetooth de la placa.




D'aquesta forma i tenint el Bluetooth activat, la placa anirà repetint les seqüències de colors i quan s'apreti un color també s'encendrà el LED.

## 1.2 Elements Addicionals implementats

Per tal que sigui un joc més animat, i com a simple valor afegit, s'ha decidit:

- Assignar a cada botó (en representació d'un color) un so diferent. S'ha optat per assignar una nota musical.
- En els botons d'escollir un mode de joc o un altre, també tenen assignat un so. Aquest so és un senzill "clic".
- El canvi de fase també té un so distintiu, s'ha decidit incloure un so al canvi de fase (una campaneta) per tal de que l'usuari tingui constància de que ha superat el nivell.
- Quan el jugador juga en mode clàssic i perd, el mòbil vibra per tal d'indicar que s'ha acabat la partida.

# 2 Comunicació Bluetooth amb la placa

Per tal de comunicar-nos amb la placa, cal seguir els següents passos:

1. **Activar el Bluetooth des dels ajustaments del sistema**: Abans d'engegar l'aplicació, si es vol jugar amb la placa, cal que s'activi el Bluetooth des dels ajustaments, just abans d'entrar en l'aplicació.
2. **Entrar al joc i activar el switch**: D'aquesta forma, permetrem que els missatges Bluetooth arribin a la placa.
3. **Jugar en qualsevol dels dos modes de joc**: Mentre s'estigui jugant, s'aniran obrint i apagant els LEDs, segons la seqüència de colors que vagi sorgint i també quan el jugador premi un botó.

A més a més, la placa també és coneixedora dels següents events:

- ***Game Over***: Informa al jugador que ha perdut, atès que s'ha equivocat.
- ***Canvi de fase***: Informa al jugador quan ha superat una fase.
- ***Puc continuar jugant***: Fa referència a que el jugador no s'ha confós en pitjar el color que tocava.

## 2.1 Implementació de la comunicació Bluetooth

El codi font que s'ha utilitzat per la comunicació Bluetooth, en ambdós casos s'ha cercat per Internet:

- Bluetooth a Android: https://stackoverflow.com/questions/22899475/android-sample-bluetooth-code-to-send-a-simple-string-via-bluetooth
- Bluetooth per a la placa: https://os.mbed.com/questions/83079/Send-data-between-PC-Monitor-and-Bluetoo/

Bàsicament el codi que hi ha en l'Android és ben senzill, simplement recupera els dispositius aparellats amb el dispositiu mòbil i a través de codi es connecta. En cas que no hi hagi dispositius connectats, escriu als Logs que no ha trobat Bluetooth o que no hi ha dispositius aparellats.

Els missatges Bluetooth que enviem són els següents:

- **"1"**: Botó ***vermell*** pitjat o està en la seqüència i per tant ha de engegar-se el LED.
- **"2"**: Botó ***verd*** pitjat o està en la seqüència i per tant ha de engegar-se el LED.
- **"3"**: Botó ***groc*** pitjat o està en la seqüència i per tant ha de engegar-se el LED.
- **"4"**: Botó ***blau*** pitjat o està en la seqüència i per tant ha de engegar-se el LED.
- **"5" o "0"**: Informa al **jugador** que no s'ha confós en pitjar el botó i que **pot seguir jugant**.
- **"7"**: Informa al **jugador** que ha **passat de nivell** o fase.
- **"8"**: Informa al **jugador** que s'ha equivocat i què **ha perdut**.

Aquests codis són els que envia l'aplicació Android a la placa mitjançant la comunicació sense fil Bluetooth.

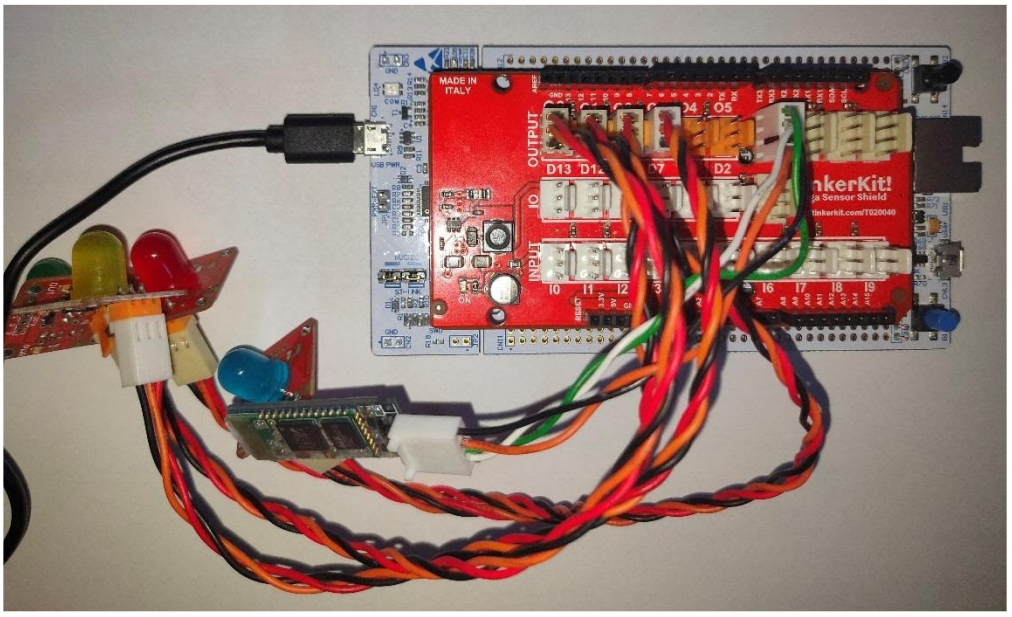
La placa els rep per mitjà del port Serial preparat pel Bluetooth (D1, D0). En aquest ho rep en *chars*. Llavors, per tal que la comunicació sigui efectiva, cal veure els codis en quin valor ASCII estan representats:

| **Caràcter o string** | **Número en ASCII** |
|---|---|
| "0" | 48 |
| "1" | 49 |
| "2" | 50 |
| "3" | 51 |
| "4" | 52 |
| "5" | 53 |
| "7" | 55 |
| "8" | 56 |

En funció del que rep, la placa té un comportament o un altre, segons el que s'ha descrit amb anterioritat.

## 2.2 Manual de connexions

Per tal que el nostre software de la placa funcioni correctament, cal connectar els LEDs i el circuit Bluetooth en els pins corresponents. Ha de quedar connectat com en la següent imatge:




De forma més concreta, els pins connectats són:

| | | |
|---|---|---|
| O0 | D11 | Verd |
| O1 | D10 | Blau |
| O2 | D9 | Vermell |
| O3 | D6 | Groc |
| Serial 1 | (D1,D0) | Circuit Bluetooth |

## 2.3 Implementacions no aconseguides

Es va intentar permetre la jugabilitat des de la placa, és a dir, que l'usuari tingués la opció d'escollir si volia jugar des del dispositiu mòbil o introduint els valors per consola.

Es va arribar a implementar, i funcionava mitjançant el pas de codis entre android i la placa. Cada codi tenia un significat, i aquests són els mateixos que s'ha exposat amb anterioritat.

Per cada valor que s'introdueix per consola, la placa comunicava a android amb un char que, la mateixa aplicació comprovava seguint el mateix procés que se segueix si es juga des del mòbil.

En cas de que l'input de l'usuari fos correcte, android enviava un missatge de "tot ok" a la placa, la que tornava a demanar input a l'usuari.

L'inconvenient que ens vam trobar, va ser un problema greu de concurrència. Com que ambdues funcions (llegir per teclat i escoltar a android) de la placa es trobaven al while que hi ha dins del main, la funció que escoltava i encenia els LEDS no tenia temps de mostrar la seqüència a l'usuari, i la placa es quedava bloquejada esperant l'input de l'usuari.

Som conscients que hauríem d'haver aplicat multithreading, però no vàrem tenir els coneixements necessaris.