CUBO DE ADRIK

ADRIÁN MORALES ALFONSO 11 / 2024

SISTEMAS ELECTRÓNICOS PARA AUTOMATIZACIÓN

ÍNDICE

1. Introducción 2
2. Desarrollo 3
3. Código completo 8
4. Manual de Usuario 35
5. Conclusiones 37

1. Introducción

1.1. Idea

Cubo de Adrik es un proyecto, basado en el cubo de Rubik tradicional, que combina el desafío lógico de los rompecabezas clásicos con tecnologías modernas de interfaces táctiles y controladores electrónicos.

1.2. Descripción del proyecto

En el proyecto se pretende llevar a cabo una simulación digital del cubo de Rubik a una pantalla táctil, donde los usuarios puedan interactuar directamente con las caras del cubo para rotar, resolver y disfrutar de una experiencia visual e inmersiva.

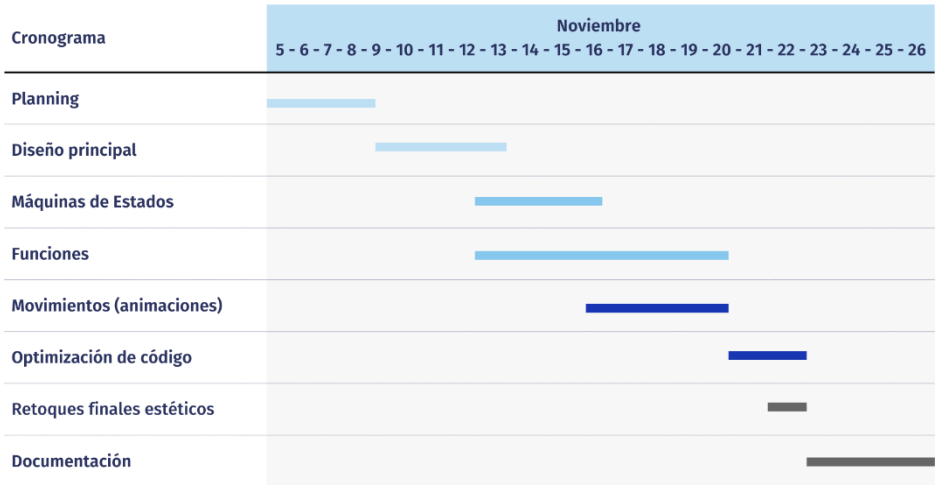
1.3. Justificación de uso del microcontrolador

El proyecto fue desarrollado en el lenguaje de programación C y utilizó el microcontrolador TM4C1294 de la familia ARM Cortex M4F como base para controlar la lógica del juego y la interacción del usuario.

Se ha decidido el uso de este microcontrolador de Texas Instruments por su "gran" memoria flash interna de 1Mb, su gran potencia y coste económico.

1.4. Gestión de conflictos y cronograma

Se observa que la mayor parte del tiempo se empleó en las funciones. Más adelante se explicará el porqué.



2. Desarrollo

2.1. Periféricos utilizados

- **BMI160**
  Usado principalmente como acelerómetro. Es un sensor que, como su propio nombre indica, mide la aceleración. En nuestro caso, lo hemos utilizado para los ejes x e y (Movimientos de vista del cubo).
- **Pantalla VM800**
  La pantalla es el elemento principal utilizado en el desarrollo de este juego, pues es donde visualizamos todo lo que ocurre en él, y a su vez, interactuamos con su tecnología táctil resistiva. Mediante las funciones gráficas se ha dibujado todo lo que compone el juego y sus distintos escenarios e interacciones. Nos hemos apoyado de la librería ft800.c y de FT800 programmer guide.
- **Botones de la placa**
  Uno de ellos se ha usado para el cambio de perspectivas, para que resulte más fácil la resolución del cubo.
  El otro verifica si se ha realizado correctamente, o no el cubo.
- **UART**
  Mediante la UART se ha configurado la transmisión de datos seriales, que en nuestro caso nos ha servido para ver en el ordenador la solución del cubo aleatorio generado.

2.2. Librerías

```
#include <stdint.h>
#include <stdbool.h>
#include <stdio.h>
#include <math.h> //Para operaciones matemáticas más complejas
#include "driverlib2.h"//Unión de varias cabeceras
#include "FT800_TIVA.h" //Funciones principales de la pantalla
#include "utils/uartstdio.h" //Para el uso de la UART
#include "HAL_I2C.h"//Capa Hal. Cabecera y definiciones
#include "sensorlib2.h" //Unión de varias cabeceras
#include <time.h> //Usada para el número random
```

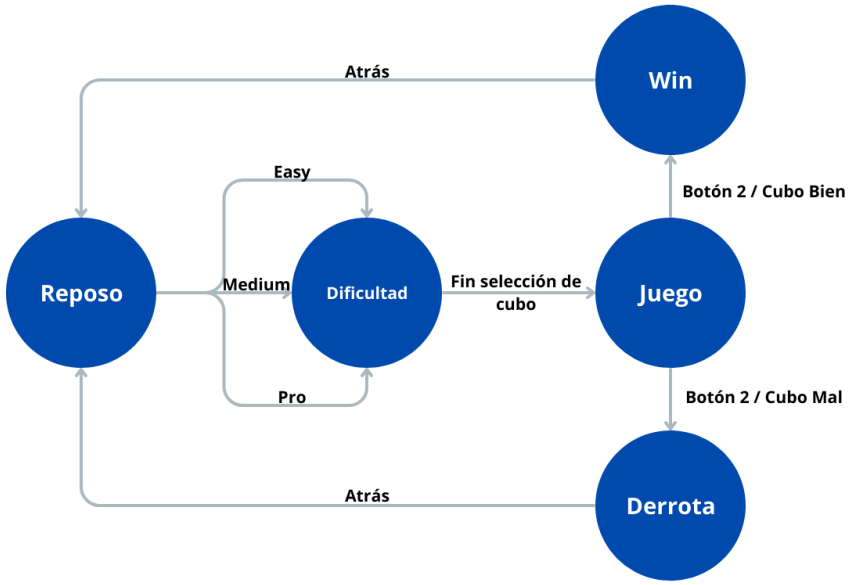
2.3. Máquinas de estado

La primera de las máquinas de estado es el corazón del funcionamiento del juego. Parte de un estado de reposo en el que se reinician las variables empleadas en el cronómetro y las animaciones. Una vez elegido el modo de juego nos movemos al estado *DIFICULTAD,* en el que, según la opción elegida anteriormente, deshace un cubo, ya resuelto, en tantos movimientos. Al deshacerlo nos delata por el puerto serie los movimientos empleados.

Una vez empezado el estado *JUEGO* se activa el cronómetro y se empiezan a contar los movimientos. Hay dos perspectivas disponibles, la de una única cara principal, y la típica representación en alzado planta y perfil. Nos moveremos entre ellas pulsando uno de los botones de la propia placa. Cabe decir que sólo puede haber movimientos en la primera de las perspectivas.

Con el otro de los botones iniciaremos el proceso de verificación del cubo, con el que obtendremos, o no, la victoria en el juego (estado *Win* o *derrota*). En los dos estados tenemos la opción de volver al menú principal, *REPOSO*.

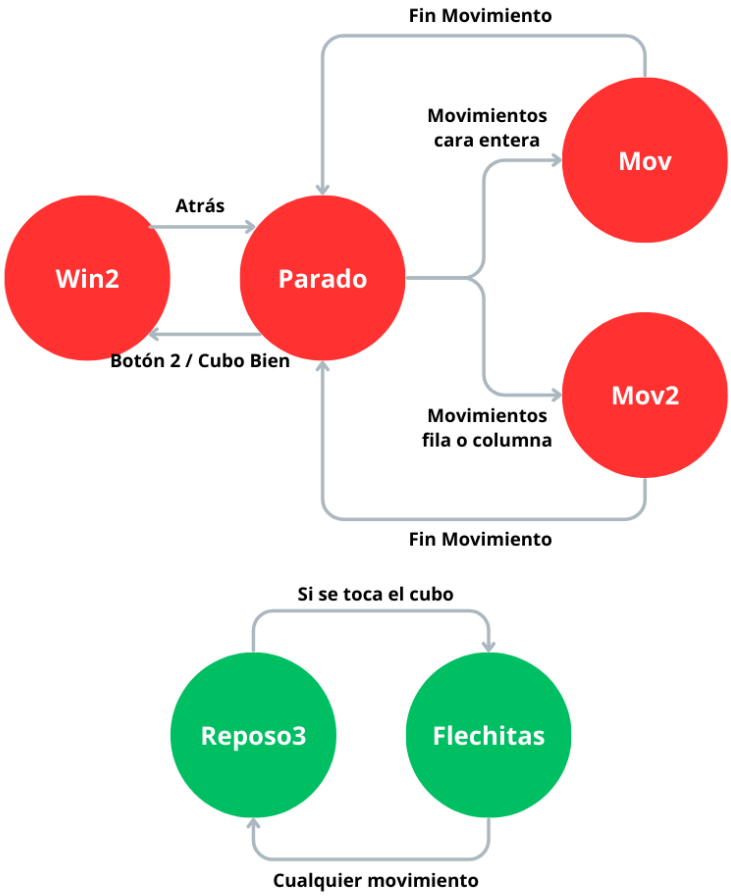
A continuación, podemos observar la representación de la máquina de estados, que es bastante sencilla.



Las otras dos máquinas de estados son aún menos complejas y son para anular la representación, o no, de las diferentes partes de las animaciones.

La de las flechas representan, o no, unas flechas con las que moveremos las filas o columnas deseadas.

La de los movimientos, tiene dos bloques importantes. *Mov* incluye los cuatro movimientos que giran caras completas, y *Mov2* incluye los doce movimientos posibles de filas o columnas que se le puede hacer a una cara. *Win2* es una animación que sólo terminaría volviendo al menú principal. A continuación, vemos estas máquinas de estado representadas.

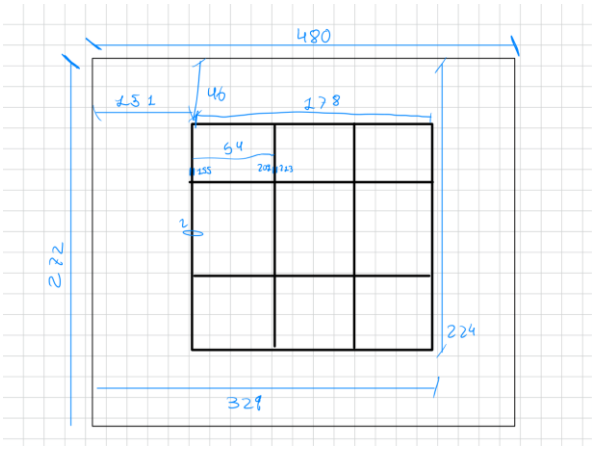


2.4. Optimización de código y funciones

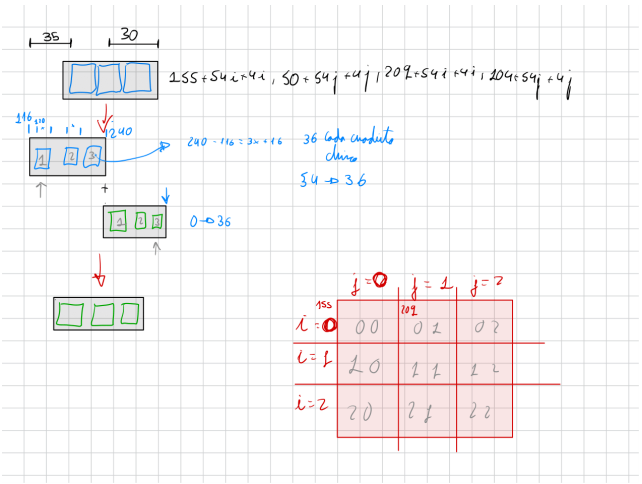
Podemos llegar a pensar que el cubo de Rubik es algo muy abstracto y complejo, pero en realidad es bastante matemático (y complejo). Es por ello, que, para ahorrar bastantes recursos y líneas de código repetidas, he empleado casi unas 30 funciones algunas más complejas que otras. Entre ellas destacamos las de actualizar las caras, filas o columnas como *CopiaFila, Espejo* o *RotacionHoraria* (Explicaremos el concepto más adelante), las de animaciones como *movs123* o *izq* y el resto como *victoria* o *Color*.

2.5. Conceptos

Para empezar, la pantalla utilizada nos daba una resolución de 480x272, es por eso por lo que hice unos bocetos para tener claro a la hora de la programación donde iba cada cosa, en las dos vistas. Se debían de tener en cuenta los márgenes en negro de la primera vista.

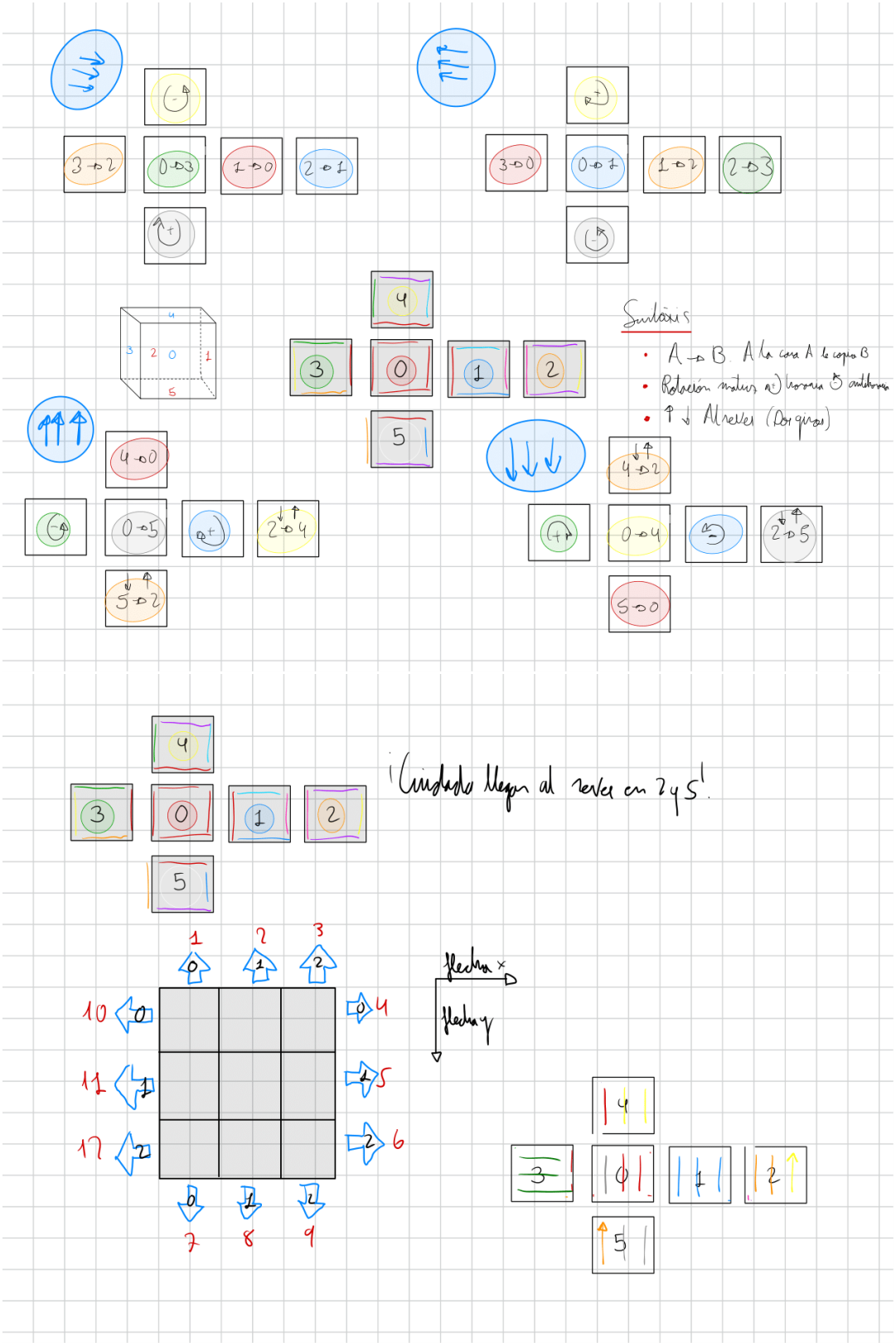


A la hora de la animación tenía que hacer un falso 3D en dos dimensiones. Dividí el movimiento en tres partes: la primera de ellas, es decir, cuando empezaba el movimiento era exactamente igual que cuando el cubo estaba parado. En la segunda, es el momento más ancho (la diagonal del cubo), en el que se ve la mitad de lo que tenía y otra mitad de lo que se va a quedar. La última parte es finalmente los cuadrados que se van a quedar. Linealizando el modelo en dos ciclos 0 a 17 y 17 a 34 (por temas de rapidez y píxeles de anchura de diagonal) obtenemos unas ecuaciones en función de una variable p (0 – 17 – 0) en la que se basa el movimiento.

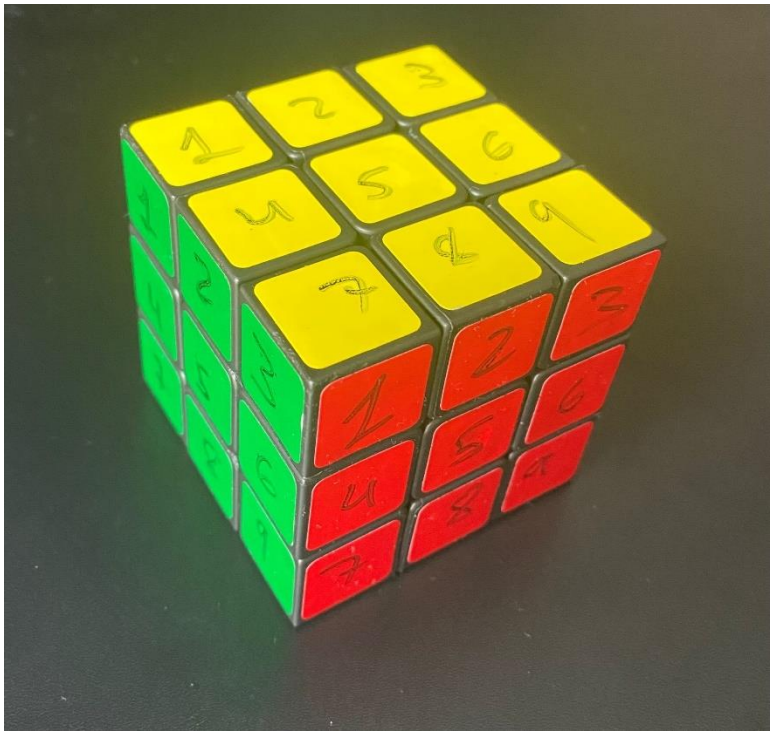


A la hora de asociar los movimientos que se hacían a la matriz principal de cubos seguía el siguiente proceso. Copiaba en una matriz auxiliar todo el cubo de 6x3x3, hacía la animación que fuera necesaria y actualizaba toda la matriz de cubos con ayuda de la auxiliar en torno a la vista de la cara 0. Es decir, siempre se representa la misma cara 0 y se actualiza junto a las demás caras dado un **único sistema de referencia**. Al resolver el sistema de esta manera, sobre una sola cara, tengo sólo 4 movimientos de caras: arriba, derecha, izquierda y abajo, en vez de 24 (que tendría al no usar un sistema de referencia). Al igual pasaría con los movimientos de filas y columnas, que con

una sola cara tengo 12, en vez de 72. He seguido los siguientes esquemas, que han resultado realmente útiles a la hora de hacer las funciones, y que a su vez, han sido creados con la ayuda de un cubo físico numerado en forma de matriz que ha servido para depurar errores espaciales en el planteamiento.



Observamos que al realizar un movimiento cualquiera hay que actualizar cada cara del cubo copiando, rotando o espejando.



Para determinar el color de cada cuadrito empleamos la función *color()* cuya entrada es la posición de tal cuadrito. La matriz está compuesta por las iniciales de cada color, y según cual sea establece un color u otro.

La función que comprueba la victoria podría haberse hecho con una comparación con otra matriz solución, pero utilizaría muchos recursos a la hora de que puede estar correcto, pero tendría que comprobar cada cara al no estar en la misma disposición u orden. Es por ello que la función finalmente realizada, comprueba el color que esta en el primer cuadro de cada cara y verifica que los demás sean iguales, devolviendo un “false” cuando no sea igual, y un “true” cuando termina la función sin encontrar fallo.

3. Código completo

```
/* PROYECTO RUBIK - SEPA
 * Adrián Morales 11/2024
 */

//LIBRERÍAS---------------------
#include <stdint.h>
#include <stdbool.h>
#include <stdio.h>
#include <math.h>
```

```
#include "driverlib2.h"
#include "FT800_TIVA.h"
#include "utils/uartstdio.h"
#include "HAL_I2C.h"
#include "sensorlib2.h"
#include <time.h>
//------------------------------

//DEFINES-------------------------------------------------
#define dword long
#define byte char

#define MSEC 40000
#define MaxEst 10

//Botones on/off
#define B1_OFF GPIOPinRead(GPIO_PORTJ_BASE,GPIO_PIN_0)
#define B1_ON !(GPIOPinRead(GPIO_PORTJ_BASE,GPIO_PIN_0))
#define B2_OFF GPIOPinRead(GPIO_PORTJ_BASE,GPIO_PIN_1)
#define B2_ON !(GPIOPinRead(GPIO_PORTJ_BASE,GPIO_PIN_1))

//#define SLEEP SysCtlSleep()
#define SLEEP SysCtlSleepFake()

#define PosMin 750
#define PosMax 1000

#define XpMax 286
#define XpMin 224
#define YpMax 186
#define YpMin 54

#define NUM_SSI_DATA   3
//-------------------------------------------------------

char chipid = 0;                        // Holds value of Chip ID read from the FT800
unsigned long cmdBufferRd = 0x00000000;         // Store the value read from the REG_CMD_READ
register
unsigned long cmdBufferWr = 0x00000000;         // Store the value read from the REG_CMD_WRITE
register
unsigned int t=0;

// User Application - Initialization of MCU / FT800 / Display
unsigned long POSX, POSY, BufferXY;
unsigned long POSYANT=0;
unsigned int CMD_Offset = 0;
unsigned long REG_TT[6];
```

```
const int32_t REG_CAL[6]= {CAL_DEFAULTS};
volatile int posicion;
int Flag_ints = 0;
//Bmi
char Cambia=0;
char string[80];
int DevID=0;
// BMI160/BMM150
int8_t returnValue;
struct bmi160_gyro_t        s_gyroXYZ;
struct bmi160_accel_t       s_accelXYZ;
struct bmi160_mag_xyz_s32_t s_magcompXYZ;
//Calibration off-sets
int8_t accel_off_x;
int8_t accel_off_y;
int8_t accel_off_z;
int16_t gyro_off_x;
int16_t gyro_off_y;
int16_t gyro_off_z;
float T_act,P_act,H_act;
char mode;
long int inicio, tiempo;

volatile long int ticks=0;
uint8_t Sensor_OK=0;
#define BP 2

//VARIABLES PROPIAS

int hora=0,min=0,seg=0;
int nmovs=0;
enum {REPOSO,DIFICULTAD,JUEGO,Win,derrota};
enum {PARADO,MOV,MOV2,win2};
enum {REPOSO2,Cara,Vistas};
enum {REPOSO3,Flechitas};
char estado=0;
char estado_mov=0;
char estado_flecha=0;
char fila_mov=0;

char CuboSol[6][3][3] = {
                      { // Cara 0 (ROJO)
                        {'R', 'R', 'R'},
                        {'R', 'R', 'R'},
                        {'R', 'R', 'R'}
                      },
                      { // Cara 1 (AZUL)
```

```
                        {'B', 'B', 'B'},
                        {'B', 'B', 'B'},
                        {'B', 'B', 'B'}
                      },
                      { // Cara 2 (NARANJA)
                        {'O', 'O', 'O'},
                        {'O', 'O', 'O'},
                        {'O', 'O', 'O'}
                      },
                      { // Cara 3 (VERDE)
                        {'G', 'G', 'G'},
                        {'G', 'G', 'G'},
                        {'G', 'G', 'G'}
                      },
                      { // Cara 4 (AMARILLO)
                        {'Y', 'Y', 'Y'},
                        {'Y', 'Y', 'Y'},
                        {'Y', 'Y', 'Y'}
                      },
                      { // Cara 5 (BLANCO)
                        {'W', 'W', 'W'},
                        {'W', 'W', 'W'},
                        {'W', 'W', 'W'}
                      }
};
char CuboAux[6][3][3];Cubo[6][3][3];
char cara=0,cara2=2;
int ciclo=0;
int dir=0,vista=0;
char onecycle=0, reset=0, block=1;
int i=0,j=0,k=0,p=0,escala=0,escala2=0,posx1=0,posy1=0;
int numero=0, contador;
char flechax,flechay;
int ebotonA=0,ebotonB=0;
char cadena[30];
int r=160,g=208,b=255;


uint32_t RELOJ, PeriodoPWM, periodo2;


//FUNCIONES-----------------------------
//Función simulación sleep
void SysCtlSleepFake(void)
{
    while(!Flag_ints);
    Flag_ints=0;
```

```
}

void IntTick(void){
    ticks++;
}




//Función despertar con reloj
void IntTimer1(void)
{
    TimerIntClear(TIMER1_BASE, TIMER_TIMA_TIMEOUT); // Borra flag
    SysCtlDelay(10); //Conveniente, Mirar Driverlib.
    Flag_ints++; //Señalizar FlagInts
    t++; //Incrementar variable de tiempo
}
void Timer0IntHandler(void)
{
    TimerIntClear(TIMER0_BASE, TIMER_TIMA_TIMEOUT); //Borra flag
    Cambia=1;
    SysCtlDelay(100);
}

int nAleatorio(int min, int max) {
    return rand() % (max - min + 1) + min;
}

void crono(){

    if(t>=20) {
        seg++;
        t=0;
    }
    if(seg==60){
        min++;
        seg=0;
    }
    if(min==60){
        hora++;
        min=0;
    }
    if(reset){
        reset=0;
        t=0;seg=0;min=0;hora=0;
    }
}
char Color(cara,i,j){
```

```
    if(Cubo[cara][i][j]=='W') ComColor(255,255,255);
    if(Cubo[cara][i][j]=='R') ComColor(255,0,0);
    if(Cubo[cara][i][j]=='B') ComColor(0, 73, 255);
    if(Cubo[cara][i][j]=='Y') ComColor(255,255,0);
    if(Cubo[cara][i][j]=='O') ComColor(255, 162, 0);
    if(Cubo[cara][i][j]=='G') ComColor(0,255,0);
    return 0;
}

//Función que evalua el estado del botón
char eval_bot(char est, char bot)
{
    switch(est)
    {
    case 0:
        if(bot){ est=1;
        }
        break;
    case 1:
        est++;
        break;
    case 2:
        if(!bot) est=0;
        break;
    }
    return est;
}

//Función victoria
bool victoria() {
    char color;
    int i,j,cara;
    for (cara=0;cara<6;cara++) {
        color=Cubo[cara][0][0];
        for (i=0;i<3;i++) {
            for (j=0;j<3;j++) {
                if (Cubo[cara][i][j] != color) {
                    return false;
                }
            }
        }
    }
    return true;
}

//FUNCIONES COPIAS Y ROTACIONES CUBO
void CopiaCara(char cara, char cara2) {
```

```
    int i,j;
    for (i=0;i<3;i++) {
        for (j=0;j<3;j++) {
            Cubo[cara][i][j]=CuboAux[cara2][i][j];
        }
    }
}

void CopiaCol(char cara, char cara2, char Col) {
    int i;
    for (i=0;i<3;i++) {
        Cubo[cara][i][Col]=CuboAux[cara2][i][Col];
    }
}


void CopiaFila(char cara, char cara2, char Fila) {
    int j;
    for (j=0;j<3;j++) {
        Cubo[cara][Fila][j]=CuboAux[cara2][Fila][j];
    }
}


void RotacionHoraria(char cara) {
    //    Cubo[cara][0][0] = CuboAux[cara][2][0];
    //    Cubo[cara][0][1] = CuboAux[cara][1][0];
    //    Cubo[cara][0][2] = CuboAux[cara][0][0];
    //    Cubo[cara][1][0] = CuboAux[cara][2][1];
    //    Cubo[cara][1][1] = CuboAux[cara][1][1];
    //    Cubo[cara][1][2] = CuboAux[cara][0][1];
    //    Cubo[cara][2][0] = CuboAux[cara][2][2];
    //    Cubo[cara][2][1] = CuboAux[cara][1][2];
    //    Cubo[cara][2][2] = CuboAux[cara][0][2];
    int i,j;
    for (i=0;i<3;i++) {
        for (j=0;j<3;j++) {
            Cubo[cara][i][j] = CuboAux[cara][2-j][i];
        }
    }
}

void RotacionAntiHoraria(char cara) {
    //    Cubo[cara][0][0] = CuboAux[cara][0][2];
    //    Cubo[cara][0][1] = CuboAux[cara][1][2];
    //    Cubo[cara][0][2] = CuboAux[cara][2][2];
    //    Cubo[cara][1][0] = CuboAux[cara][0][1];
    //    Cubo[cara][1][1] = CuboAux[cara][1][1];
```

```
    //    Cubo[cara][1][2] = CuboAux[cara][2][1];
    //    Cubo[cara][2][0] = CuboAux[cara][0][0];
    //    Cubo[cara][2][1] = CuboAux[cara][1][0];
    //    Cubo[cara][2][2] = CuboAux[cara][2][0];
    int i,j;
    for (i=0;i<3;i++) {
        for (j=0;j<3;j++) {
            Cubo[cara][i][j] = CuboAux[cara][j][2-i];
        }
    }
}

void Espejo(char cara, char cara2){
    //    Cubo[cara][0][0] = CuboAux[cara2][2][2];
    //    Cubo[cara][0][1] = CuboAux[cara2][2][1];
    //    Cubo[cara][0][2] = CuboAux[cara2][2][0];
    //    Cubo[cara][1][0] = CuboAux[cara2][1][2];
    //    Cubo[cara][1][1] = CuboAux[cara2][1][1];
    //    Cubo[cara][1][2] = CuboAux[cara2][1][0];
    //    Cubo[cara][2][0] = CuboAux[cara2][0][2];
    //    Cubo[cara][2][1] = CuboAux[cara2][0][1];
    //    Cubo[cara][2][2] = CuboAux[cara2][0][0];
    int i,j;
    for (i=0;i<3;i++) {
        for (j=0;j<3;j++) {
            Cubo[cara][i][j]=CuboAux[cara2][2-i][2-j];
        }
    }
}

void CopiaCaraAux(char cara, char cara2) {
    int i,j;
    for (i=0;i<3;i++) {
        for (j=0;j<3;j++) {
            CuboAux[cara2][i][j] = Cubo[cara][i][j];
        }
    }
}

void CopiaCuboAux() {
    int c;
    for (c=0;c<6;c++) {
        CopiaCaraAux(c,c);
    }
}

void CopiaCuboSol() {
```

```
    int c,i,j;
    for (c=0;c<6;c++) {
        for (i=0;i<3;i++) {
            for (j=0;j<3;j++) {
                Cubo[c][i][j] = CuboSol[c][i][j];
            }
        }
    }
}



//Funciones declaradas más abajo
void periodo();
void movs123();
void movs456();
void movs789();
void movsabc();
void drch();
void abaj();
void arr();
void izq();
void movsaleatorio();
//---------------------------------------


int main(void){


    RELOJ=SysCtlClockFreqSet((SYSCTL_XTAL_25MHZ | SYSCTL_OSC_MAIN | SYSCTL_USE_PLL |
SYSCTL_CFG_VCO_480), 120000000);
    //Reloj Interrupciones
    SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER1); //Habilita T1
    TimerClockSourceSet(TIMER1_BASE, TIMER_CLOCK_SYSTEM); //T1 a 120MHz
    TimerConfigure(TIMER1_BASE, TIMER_CFG_PERIODIC); //T1 igual...
    periodo2 = RELOJ/20; //0.05s = 50 ms
    TimerLoadSet(TIMER1_BASE, TIMER_A, periodo2 -1);
    TimerIntRegister(TIMER1_BASE,TIMER_A,IntTimer1);
    IntEnable(INT_TIMER1A);
    TimerIntEnable(TIMER1_BASE, TIMER_TIMA_TIMEOUT);
    IntMasterEnable(); //Habilitacion global de interrupciones
    TimerEnable(TIMER1_BASE, TIMER_A);
    //reloj2
    SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER0);
    TimerClockSourceSet(TIMER0_BASE, TIMER_CLOCK_SYSTEM);
    TimerConfigure(TIMER0_BASE, TIMER_CFG_PERIODIC);
    TimerLoadSet(TIMER0_BASE, TIMER_A, RELOJ/2 -1);
    TimerIntRegister(TIMER0_BASE, TIMER_A,Timer0IntHandler);
```

```
    IntEnable(INT_TIMER0A);
    TimerIntEnable(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
    IntMasterEnable();
    TimerEnable(TIMER0_BASE, TIMER_A);

    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_UART0);
    GPIOPinConfigure(GPIO_PA0_U0RX);
    GPIOPinConfigure(GPIO_PA1_U0TX);
    GPIOPinTypeUART(GPIO_PORTA_BASE, GPIO_PIN_0 | GPIO_PIN_1);

    UARTStdioConfig(0, 115200, RELOJ);

    //LEDS
    //Habilitar los periféricos implicados
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOK);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOJ);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPION);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_0 |GPIO_PIN_4); //F0 y F4: salidas
    GPIOPinTypeGPIOOutput(GPIO_PORTN_BASE, GPIO_PIN_0 |GPIO_PIN_1); //N0 y N1: salidas
    //Servo y botones
    SysCtlPeripheralEnable(SYSCTL_PERIPH_PWM0);
    GPIOPinTypeGPIOInput(GPIO_PORTJ_BASE, GPIO_PIN_0|GPIO_PIN_1);
    GPIOPadConfigSet(GPIO_PORTJ_BASE,GPIO_PIN_0|GPIO_PIN_1,GPIO_STRENGTH_2MA,GPIO_PIN_TYPE_STD
_WPU);
    //
    GPIOPinTypeGPIOInput(GPIO_PORTJ_BASE, GPIO_PIN_0|GPIO_PIN_1);   //J0 y J1: entradas
    GPIOPadConfigSet(GPIO_PORTJ_BASE,GPIO_PIN_0|GPIO_PIN_1,GPIO_STRENGTH_2MA,GPIO_PIN_TYPE_STD
_WPU); //Pullup en J0 y J1


    SysCtlPeripheralSleepEnable(SYSCTL_PERIPH_GPIOJ);
    SysCtlPeripheralClockGating(true);


    HAL_Init_SPI(1, RELOJ);  //Boosterpack a usar, Velocidad del MC
    Inicia_pantalla();       //Arranque de la pantalla

    // Delay antes de empezar a representar

    SysCtlDelay(RELOJ/3);


    // PANTALLA INICIAL----------------------------------------------------------------
    Nueva_pantalla(r,g,b); //borron de pantalla por el color indicado
    ComColor(255,0,0);
    ComTXT(HSIZE/2,VSIZE/2-22, 31, OPT_CENTERX,"Cubo de Adrik"); //cadena inmutable
```

```
    ComTXT(HSIZE/2,VSIZE/2+22, 20, OPT_CENTERX,"Adrian Morales"); //cadena inmutable
    Dibuja(); //Aqui es cuando se hace realidad el borrado
    Espera_pant();
    //--------------------------------------------------------------------------------


    for(i=0;i<6;i++)    Esc_Reg(REG_TOUCH_TRANSFORM_A+4*i, REG_CAL[i]); //calibrasion de
pantalla
    Conf_Boosterpack(2, RELOJ);
    //UARTprintf("Inicializando BMI160, modo NAVIGATION... ");
    bmi160_initialize_sensor();
    bmi160_config_running_mode(APPLICATION_NAVIGATION);
    //UARTprintf("Hecho! \nLeyendo DevID... ");
    readI2C(BMI160_I2C_ADDR2,BMI160_USER_CHIP_ID_ADDR, &DevID, 1);
    //UARTprintf("DevID= 0X%x \n", DevID);


    SysTickIntRegister(IntTick);
    SysTickPeriodSet(12000);
    SysTickIntEnable();
    SysTickEnable();

    while(1){
        SLEEP;
        //Para ver la posición del acelerómetro
        if(Cambia==1){
            Cambia=0;ticks=0;
            bmi160_read_accel_xyz(&s_accelXYZ);
            //UARTprintf("-----------------------------------------------\n");
            //sprintf(string, "  ACCL:
X:%6d\tY:%6d\tZ:%6d  \n",s_accelXYZ.x,s_accelXYZ.y,s_accelXYZ.z);
            //UARTprintf(string);


        }
        srand(time(0)); //Para el número aleatorio

        Lee_pantalla();
        Nueva_pantalla(r, g, b); //Actualizo la pantalla y la pongo a un color
        //MÁQUINA DE ESTADOS DE LOS MODOS DE JUEGO
        switch (estado) {
        case REPOSO:
            //reposo();
            reset=1;
            nmovs=0;
            ciclo=0;
            p=0;
            onecycle=0;
```

```
            ComColor(255,255,255);
            ComTXT(240,50, 29, OPT_CENTERX,"Elige un modo de juego");
            ComColor(255,255,255);
            ComFgcolor(0,0,255);
            if (Boton(80, 90, 300, 40, 30, "Easy")){
estado=DIFICULTAD;contador=5;CopiaCuboSol();}
            ComFgcolor(255, 0, 0);
            if (Boton(80, 140, 300, 40, 30, "Medium")){
estado=DIFICULTAD;contador=15;CopiaCuboSol();}
            ComFgcolor(0, 0, 0);
            if (Boton(80, 190, 300, 40, 30, "PRO")){
estado=DIFICULTAD;contador=25;CopiaCuboSol();}

            break;
        case DIFICULTAD:
            estado_mov=PARADO;estado_flecha=REPOSO3;
            UARTprintf("\n-------------------------- \n");
            while(contador){
                CopiaCuboAux();
                contador--;
                numero = nAleatorio(0, 15);
                sprintf(cadena, "%d:  ",numero);
                UARTprintf(cadena);
                movsaleatorio(numero);

            }
            UARTprintf("\n-------------------------- \n");
            estado=JUEGO;
            break;

        case JUEGO:
            crono();
            //Perspectiva juego
            ebotonA=eval_bot(ebotonA, B1_ON);
            ebotonB=eval_bot(ebotonB, B2_ON);
            if(ebotonB==1) vista++;
            if(vista>1) vista=0;
            if(ebotonA==1) {
                if (victoria()) {estado=Win;estado_mov=win2; CopiaCuboAux();}
                else {estado=derrota; r=0;g=0;b=0;}
            }
            //Cronómetro
            ComColor(255,255,255);
            sprintf(cadena,"%02d:%02d:%02d", hora,min,seg);
            ComTXT(77,VSIZE/2, 29, OPT_CENTERX,cadena);
            //Cubo vista principal
```

```
            if(vista==0){
                //Representación del cubo
                ComColor(0,0,0);
                ComRect(151, 46, 329, 224, true);
                for (i=0;i<3;i++){
                    for (j=0;j<3;j++){
                        Color(0,i,j);
                        ComRect(155+58*j, 50+58*i, 209+58*j, 104+58*i, true); //Cada cuadrito
                    }
                }

                //Aparecer flechas
                for (j=0;j<3;j++){
                    for (i=0;i<3;i++){
                        if(POSX>155+i*58 && POSX<209+i*58 && POSY>50+j*58 && POSY<104+j*58)
//Pulsando 1er cuadro
                        {
                            flechax=i;
                            flechay=j;
                            estado_flecha=1;
                        }
                    }
                }
                //GIROS TOTALES
                if((s_accelXYZ.x<-10000) && (estado_mov!=MOV)){
                    CopiaCuboAux();
                    estado_mov=MOV;
                    dir=3;

                }
                if((s_accelXYZ.x>10000) && (estado_mov!=MOV)){
                    CopiaCuboAux();
                    estado_mov=MOV;
                    dir=4;

                }
                if((s_accelXYZ.y<-10000) && (estado_mov!=MOV)){
                    CopiaCuboAux();
                    estado_mov=MOV;
                    dir=2;

                }
                if((s_accelXYZ.y>10000) && (estado_mov!=MOV)){
                    CopiaCuboAux();
                    estado_mov=MOV;
                    dir=1;
```

```
                }

            }
            //Cubo vistas
            if(vista==1){
                estado_flecha=0;
                cara=3;
                for (k=0;k<4;k++){
                    for (i=0;i<3;i++){
                        for (j=0;j<3;j++){
                            if(k==1) cara=0; if(k==2) cara=1; if(k==3) cara=2;
                            Color(cara,i,j);
                            ComRect(135+21*j+74*k, 104+21*i, 156+21*j+74*k, 125+21*i, true);
//Cada cuadrito
                        }
                    }
                }
                for (k=0;k<2;k++){
                    for (i=0;i<3;i++){
                        for (j=0;j<3;j++){
                            if(k==0) cara=4; if(k==1) cara=5;
                            Color(cara,i,j);
                            ComRect(209+21*j, 30+21*i+148*k, 230+21*j, 51+21*i+148*k, true);
//Cada cuadrito
                        }
                    }
                }
            }

            break;
        case Win:
            estado_flecha=0;
            ComColor(255,255,255);
            ComFgcolor(255, 0, 0);
            if (Boton(30, 30, 40, 40, 16, "<")){
estado=REPOSO;estado_mov=PARADO;estado_flecha=REPOSO3;
            }


            break;

        case derrota:
            estado_flecha=0;
            ComColor(255,0,0);
            ComTXT(HSIZE/2,VSIZE/2-22, 31, OPT_CENTERX,"DERROTA :("); //cadena inmutable
            ComColor(255,255,255);
            ComFgcolor(255, 0, 0);
```

```
            if (Boton(30, 30, 40, 40, 16, "<")){
estado=REPOSO;estado_mov=PARADO;estado_flecha=REPOSO3;r=160;g=208;b=255;
            }

            break;
        }
        //MÁQUINA DE ESTADOS FLECHAS
        switch (estado_flecha) {
        case REPOSO3:
            break;
        case Flechitas:
            ComColor(0,0,0);
            //izq
            ComLine(149,77+flechay*58,109,77+flechay*58,1);
            ComLine(109,77+flechay*58,129,62+flechay*58,1);
            ComLine(109,77+flechay*58,129,92+flechay*58,1);
            //derecha
            ComLine(330,77+flechay*58,370,77+flechay*58,1);
            ComLine(370,77+flechay*58,350,62+flechay*58,1);
            ComLine(370,77+flechay*58,350,92+flechay*58,1);
            //arriba
            ComLine(182+flechax*58,45,182+flechax*58,5,1);
            ComLine(167+flechax*58,25,182+flechax*58,5,1);
            ComLine(197+flechax*58,25,182+flechax*58,5,1);
            //abajo
            ComLine(182+flechax*58,225,182+flechax*58,265,1);
            ComLine(167+flechax*58,245,182+flechax*58,265,1);
            ComLine(197+flechax*58,245,182+flechax*58,265,1);

            //Tenemos en cuenta que no se pueden detectar dos posiciones a la vez por el tipo
de pantalla
            for (i=0;i<3;i++){
                if(flechax==i){
                    if(POSX>155+i*58 && POSX<209+i*58 && POSY>0 && POSY<46) {fila_mov=i+1;
CopiaCuboAux(); estado_mov=MOV2;estado_flecha=0;}
                    else if (POSX>155+i*58 && POSX<209+i*58 && POSY>224 && POSY<272){
fila_mov=i+7;CopiaCuboAux(); estado_mov=MOV2;estado_flecha=0;}
                }
                if(flechay==i){
                    if(POSX>330 && POSX<480 && POSY>50+i*58 && POSY<104+i*58)
{fila_mov=i+4;CopiaCuboAux(); estado_mov=MOV2;estado_flecha=0;}
                    else if (POSX>0 && POSX<150 && POSY>50+i*58 && POSY<104+i*58)
{fila_mov=i+10;CopiaCuboAux(); estado_mov=MOV2;estado_flecha=0;}
                }
            }

            break;
```

```
}

//MÁQUINA DE ESTADOS DE LAS ANIMACIONES
switch (estado_mov) {
case PARADO:

    break;
case MOV:
    periodo();

    if(dir==1){
        ComRect(151-p*2, 46, 329+p*2, 224, true); //Cuadro negro encima FILAS
        //DERECHA
        for (i=0;i<3;i++){
            drch();
        }
        if(onecycle){
            onecycle=0;
            //ACTUALIZAR CARAS
            CopiaCara(0,3);
            CopiaCara(1,0);
            CopiaCara(2,1);
            CopiaCara(3,2);
            RotacionHoraria(5);
            RotacionAntiHoraria(4);
        }
    }
    else if(dir==2){
        ComRect(151-p*2, 46, 329+p*2, 224, true); //Cuadro negro encima FILAS
        //IZQUIERDA
        for (i=0;i<3;i++){
            izq();
        }
        if(onecycle){
            onecycle=0;
            //ACTUALIZAR CARAS
            CopiaCara(0,1);
            CopiaCara(1,2);
            CopiaCara(2,3);
            CopiaCara(3,0);
            RotacionHoraria(4);
            RotacionAntiHoraria(5);
        }
    }
    else if(dir==3){
        ComRect(151, 46-p*2, 329, 224+p*2, true); //Cuadro negro encima COLS
        //ARRIBA
```

```
                for (j=0;j<3;j++){
                    arr();
                }
                if(onecycle){
                    onecycle=0;
                    //ACTUALIZAR CARAS
                    CopiaCara(0,5);
                    CopiaCara(4,0);
                    Espejo(5,2);
                    Espejo(2,4);
                    RotacionHoraria(1);
                    RotacionAntiHoraria(3);
                }
            }
            else if(dir==4){
                ComRect(151, 46-p*2, 329, 224+p*2, true); //Cuadro negro encima COLS
                //ABAJO

                for (j=0;j<3;j++){
                    abaj();
                }
                if(onecycle){
                    onecycle=0;
                    //ACTUALIZAR CARAS
                    CopiaCara(0,4);
                    CopiaCara(5,0);
                    Espejo(4,2);
                    Espejo(2,5);
                    RotacionAntiHoraria(1);
                    RotacionHoraria(3);
                }
            }

            //  ComRect(151-p*2, 46+54*fil, 329+p*2, 108+54*fil, true); //Cuadro negro encima
PRIMERA FILA
            // ComRect(151+54*col, 46-p*2, 213+54*col, 224+p*2, true); //Cuadro negro encima
PRIMERA COL

            break;

        case MOV2:

            periodo();

            if(fila_mov==1){
                movs123();
                if(onecycle){
```

```
            onecycle=0;
            //ACTUALIZAR CARAS
            CopiaCol(0,5,0);
            CopiaCol(4,0,0);
            Espejo(2,4);
            CopiaCol(2,2,0);
            CopiaCol(2,2,1);
            Espejo(5,2);
            CopiaCol(5,5,1);
            CopiaCol(5,5,2);
            RotacionAntiHoraria(3);

        }
    }

    else if(fila_mov==2){
        movs123();
        if(onecycle){
            onecycle=0;
            //ACTUALIZAR CARAS
            CopiaCol(0,5,1);
            CopiaCol(4,0,1);
            Espejo(2,4);
            CopiaCol(2,2,0);
            CopiaCol(2,2,2);
            Espejo(5,2);
            CopiaCol(5,5,0);
            CopiaCol(5,5,2);

        }
    }

    else if(fila_mov==3){
        movs123();
        if(onecycle){
            onecycle=0;
            //ACTUALIZAR CARAS
            CopiaCol(0,5,2);
            CopiaCol(4,0,2);
            Espejo(2,4);
            CopiaCol(2,2,1);
            CopiaCol(2,2,2);
            Espejo(5,2);
            CopiaCol(5,5,0);
            CopiaCol(5,5,1);
            RotacionHoraria(1);
```

```
        }
    }
    else if(fila_mov==4){
        movs456();
        if(onecycle){
            onecycle=0;
            //ACTUALIZAR CARAS
            CopiaFila(0,3,0);
            CopiaFila(3,2,0);
            CopiaFila(1,0,0);
            CopiaFila(2,1,0);
            RotacionAntiHoraria(4);
        }
    }
    else if(fila_mov==5){
        movs456();
        if(onecycle){
            onecycle=0;
            //ACTUALIZAR CARAS
            CopiaFila(0,3,1);
            CopiaFila(3,2,1);
            CopiaFila(1,0,1);
            CopiaFila(2,1,1);
        }
    }

    else if(fila_mov==6){
        movs456();
        if(onecycle){
            onecycle=0;
            //ACTUALIZAR CARAS
            CopiaFila(0,3,2);
            CopiaFila(3,2,2);
            CopiaFila(1,0,2);
            CopiaFila(2,1,2);
            RotacionHoraria(5);
        }
    }
    else if(fila_mov==7){
        movs789();
        if(onecycle){
            onecycle=0;
            //ACTUALIZAR CARAS
            CopiaCol(0,4,0);
            CopiaCol(5,0,0);
            Espejo(4,2);
            CopiaCol(4,4,1);
```

```
            CopiaCol(4,4,2);
            Espejo(2,5);
            CopiaCol(2,2,0);
            CopiaCol(2,2,1);
            RotacionHoraria(3);

        }
    }

    else if(fila_mov==8){
        movs789();
        if(onecycle){
            onecycle=0;
            //ACTUALIZAR CARAS
            CopiaCol(0,4,1);
            CopiaCol(5,0,1);
            Espejo(4,2);
            CopiaCol(4,4,0);
            CopiaCol(4,4,2);
            Espejo(2,5);
            CopiaCol(2,2,0);
            CopiaCol(2,2,2);

        }
    }

    else if(fila_mov==9){
        movs789();
        if(onecycle){
            onecycle=0;
            //ACTUALIZAR CARAS
            CopiaCol(0,4,2);
            CopiaCol(5,0,2);
            Espejo(4,2);
            CopiaCol(4,4,0);
            CopiaCol(4,4,1);
            Espejo(2,5);
            CopiaCol(2,2,1);
            CopiaCol(2,2,2);
            RotacionAntiHoraria(1);

        }
    }
    else if(fila_mov==10){
        movsabc();
        if(onecycle){
            onecycle=0;
```

```
                //ACTUALIZAR CARAS
                CopiaFila(0,1,0);
                CopiaFila(3,0,0);
                CopiaFila(1,2,0);
                CopiaFila(2,3,0);
                RotacionHoraria(4);
            }
        }
        else if(fila_mov==11){
            movsabc();
            if(onecycle){
                onecycle=0;
                //ACTUALIZAR CARAS
                CopiaFila(0,1,1);
                CopiaFila(3,0,1);
                CopiaFila(1,2,1);
                CopiaFila(2,3,1);
            }
        }

        else if(fila_mov==12){
            movsabc();
            if(onecycle){
                onecycle=0;
                //ACTUALIZAR CARAS
                CopiaFila(0,1,2);
                CopiaFila(3,0,2);
                CopiaFila(1,2,2);
                CopiaFila(2,3,2);
                RotacionAntiHoraria(5);
            }
        }
        break;
    case win2:
        periodo();
        ComRect(151-p*2, 46, 329+p*2, 224, true); //Cuadro negro encima FILAS
        for (i=0;i<3;i++){
            drch();
        }
        if(onecycle){
            onecycle=0;
            //ACTUALIZAR CARAS
            CopiaCara(0,3);
            CopiaCara(1,0);
            CopiaCara(2,1);
            CopiaCara(3,2);
            CopiaCuboAux();
```

```
                estado_mov=win2;
                nmovs--;
            }
            ComColor(0,0,0);
            ComTXT(HSIZE/2,VSIZE/2-32, 31, OPT_CENTERX,"VICTORIA!"); //cadena inmutable
            sprintf(cadena,"Movimientos realizados: %d", nmovs);
            ComTXT(HSIZE/2,VSIZE/2+15, 23, OPT_CENTERX,cadena);
            sprintf(cadena,"Tiempo: %02d:%02d:%02d", hora,min,seg);
            ComTXT(HSIZE/2,VSIZE/2+35, 23, OPT_CENTERX,cadena);
            ComColor(255,255,255);
            ComTXT(HSIZE/2+2,VSIZE/2-32+2, 31, OPT_CENTERX,"VICTORIA!"); //cadena inmutable
            sprintf(cadena,"Movimientos realizados: %d", nmovs);
            ComTXT(HSIZE/2+1,VSIZE/2+15+1, 23, OPT_CENTERX,cadena);
            sprintf(cadena,"Tiempo: %02d:%02d:%02d", hora,min,seg);
            ComTXT(HSIZE/2+1,VSIZE/2+35+1, 23, OPT_CENTERX,cadena);

            break;
        }
        Dibuja();
    }
}

//FUNCIONES de movimientos no principales-------------------------
//Movimiento de monofilas/monocolumnas
void periodo(){

    ciclo++;
    ComColor(0,0,0);

    if(ciclo<=17){
        p++;//Hasta 17
        escala=54-p;
        escala2=p*2;
    }
    else if(ciclo<35){
        p--;
        escala=p*2;
        escala2=54-p;
    }
    else {
        estado_mov=PARADO;
        ciclo=0;
        onecycle=1;
        nmovs++;
    }
}
```

```
void movs123(){
    j=fila_mov-1;
    ComRect(151+58*j, 46-p*2, 213+58*j, 224+p*2, true); //Cuadro negro encima PRIMERA COL
    arr();
}

void movs456(){
    i=fila_mov-4;
    ComRect(151-p*2, 46+58*i, 329+p*2, 108+58*i, true); //Cuadro negro encima PRIMERA COL
    drch();
}

void movs789(){
    j=fila_mov-7;
    ComRect(151+58*j, 46-p*2, 213+58*j, 224+p*2, true); //Cuadro negro encima PRIMERA COL
    abaj();
}

void movsabc(){
    i=fila_mov-10;
    ComRect(151-p*2, 46+58*i, 329+p*2, 108+58*i, true); //Cuadro negro encima PRIMERA COL
    izq();
}

// Movimiento de lados arriba, izquierda, derecha y abajo
void izq(){
    for (j=0;j<3;j++){
        posx1=(155-p*2)+(escala)*j+4*j;
        Color(0,i,j);
        ComRect(posx1, 50+58*i, posx1+escala, 104+58*i, true); //Cada cuadrito
        posx1=(325+p*2)-(escala2)*j-4*j;
        Color(1,i,2-j);
        ComRect(posx1, 50+58*i, posx1-escala2, 104+58*i, true); //Cada cuadrito
    }
}
void drch(){
    for (j=0;j<3;j++){
        posx1=(325+p*2)-(escala)*j-4*j;
        Color(0,i,2-j);
        ComRect(posx1, 50+58*i, posx1-escala, 104+58*i, true); //Cada cuadrito
        posx1=(155-p*2)+(escala2)*j+4*j;
        Color(3,i,j);
        ComRect(posx1, 50+58*i, posx1+escala2, 104+58*i, true); //Cada cuadrito
    }
}
void arr(){
    for (i=0;i<3;i++){
```

```
        posx1=(50-p*2)+(escala)*i+4*i;
        Color(0,i,j);
        ComRect(155+58*j,posx1,209+58*j,posx1+escala, true); //Cada cuadrito
        posx1=(220+p*2)-(escala2)*i-4*i;
        Color(5,2-i,j);
        ComRect(155+58*j,posx1,209+58*j,posx1-escala2, true); //Cada cuadrito
    }
}
void abaj(){
    for (i=0;i<3;i++){
        posx1=(220+p*2)-(escala)*i-4*i;
        Color(0,2-i,j);
        ComRect(155+58*j,posx1,209+58*j,posx1-escala, true); //Cada cuadrito
        posx1=(50-p*2)+(escala2)*i+4*i;
        Color(4,i,j);
        ComRect(155+58*j,posx1,209+58*j,posx1+escala2, true); //Cada cuadrito
    }
}

void movsaleatorio(int random){
    if(random==0){
        //MOV DERECHA
        CopiaCara(0,3);
        CopiaCara(1,0);
        CopiaCara(2,1);
        CopiaCara(3,2);
        RotacionHoraria(5);
        RotacionAntiHoraria(4);
        UARTprintf("Movimiento Derecha <-----> Movimiento Izquierda\n");
    }
    else if(random==1){ //MOV IZQUIERDA
        CopiaCara(0,1);
        CopiaCara(1,2);
        CopiaCara(2,3);
        CopiaCara(3,0);
        RotacionHoraria(4);
        RotacionAntiHoraria(5);
        UARTprintf("Movimiento Izquierda <-----> Movimiento Derecha\n");
    }
    else if(random==2){ //MOV Arriba
        CopiaCara(0,5);
        CopiaCara(4,0);
        Espejo(5,2);
        Espejo(2,4);
        RotacionHoraria(1);
        RotacionAntiHoraria(3);
        UARTprintf("Movimiento Arriba <-----> Movimiento Abajo\n");
```

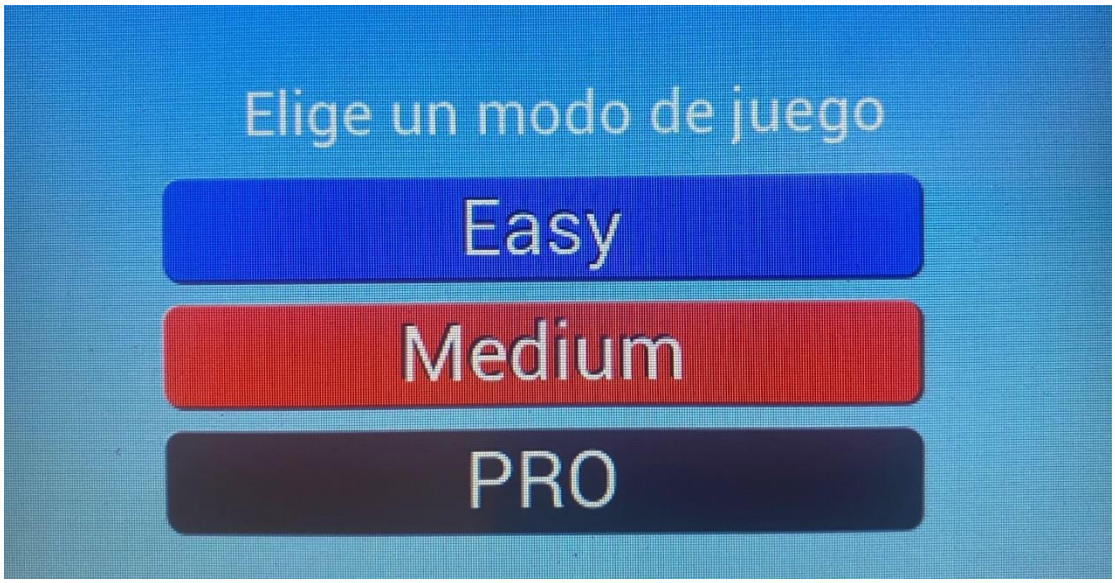
```
    }
    else if(random==3){ //MOV ABAJO
        CopiaCara(0,4);
        CopiaCara(5,0);
        Espejo(4,2);
        Espejo(2,5);
        RotacionAntiHoraria(1);
        RotacionHoraria(3);
        UARTprintf("Movimiento Abajo <-----> Movimiento Arriba\n");
    }
    else if(random==4){ //MOV 1
        CopiaCol(0,5,0);
        CopiaCol(4,0,0);
        Espejo(2,4);
        CopiaCol(2,2,0);
        CopiaCol(2,2,1);
        Espejo(5,2);
        CopiaCol(5,5,1);
        CopiaCol(5,5,2);
        RotacionAntiHoraria(3);
        UARTprintf("Movimiento 1 <-----> Movimiento 7\n");
    }
    else if(random==5){ //MOV 2
        CopiaCol(0,5,1);
        CopiaCol(4,0,1);
        Espejo(2,4);
        CopiaCol(2,2,0);
        CopiaCol(2,2,2);
        Espejo(5,2);
        CopiaCol(5,5,0);
        CopiaCol(5,5,2);
        UARTprintf("Movimiento 2 <-----> Movimiento 8\n");
    }
    else if(random==6){ //MOV 3
        CopiaCol(0,5,2);
        CopiaCol(4,0,2);
        Espejo(2,4);
        CopiaCol(2,2,1);
        CopiaCol(2,2,2);
        Espejo(5,2);
        CopiaCol(5,5,0);
        CopiaCol(5,5,1);
        RotacionHoraria(1);
        UARTprintf("Movimiento 3 <-----> Movimiento 9\n");
    }
    else if(random==7){ //MOV 4
        CopiaFila(0,3,0);
```

```
        CopiaFila(3,2,0);
        CopiaFila(1,0,0);
        CopiaFila(2,1,0);
        RotacionAntiHoraria(4);
        UARTprintf("Movimiento 4 <-----> Movimiento 10\n");
    }
    else if(random==8){ //MOV 5
        CopiaFila(0,3,1);
        CopiaFila(3,2,1);
        CopiaFila(1,0,1);
        CopiaFila(2,1,1);
        UARTprintf("Movimiento 5 <-----> Movimiento 11\n");
    }
    else if(random==9){ //MOV 6
        CopiaFila(0,3,2);
        CopiaFila(3,2,2);
        CopiaFila(1,0,2);
        CopiaFila(2,1,2);
        RotacionHoraria(5);
        UARTprintf("Movimiento 6 <-----> Movimiento 12\n");
    }
    else if(random==10){ //MOV 7
        CopiaCol(0,4,0);
        CopiaCol(5,0,0);
        Espejo(4,2);
        CopiaCol(4,4,1);
        CopiaCol(4,4,2);
        Espejo(2,5);
        CopiaCol(2,2,0);
        CopiaCol(2,2,1);
        RotacionHoraria(3);
        UARTprintf("Movimiento 7 <-----> Movimiento 1\n");
    }
    else if(random==11){ //MOV 8
        CopiaCol(0,4,1);
        CopiaCol(5,0,1);
        Espejo(4,2);
        CopiaCol(4,4,0);
        CopiaCol(4,4,2);
        Espejo(2,5);
        CopiaCol(2,2,0);
        CopiaCol(2,2,2);
        UARTprintf("Movimiento 8 <-----> Movimiento 2\n");
    }
    else if(random==12){ //MOV 9
        CopiaCol(0,4,2);
        CopiaCol(5,0,2);
```

```
        Espejo(4,2);
        CopiaCol(4,4,0);
        CopiaCol(4,4,1);
        Espejo(2,5);
        CopiaCol(2,2,1);
        CopiaCol(2,2,2);
        RotacionAntiHoraria(1);
        UARTprintf("Movimiento 9 <-----> Movimiento 3\n");
    }
    else if(random==13){ //MOV 10
        CopiaFila(0,1,0);
        CopiaFila(3,0,0);
        CopiaFila(1,2,0);
        CopiaFila(2,3,0);
        RotacionHoraria(4);
        UARTprintf("Movimiento 10 <-----> Movimiento 4\n");
    }
    else if(random==14){ //MOV 11
        CopiaFila(0,1,1);
        CopiaFila(3,0,1);
        CopiaFila(1,2,1);
        CopiaFila(2,3,1);
        UARTprintf("Movimiento 11 <-----> Movimiento 5\n");
    }
    else if(random==15){ //MOV 12
        CopiaFila(0,1,2);
        CopiaFila(3,0,2);
        CopiaFila(1,2,2);
        CopiaFila(2,3,2);
        RotacionAntiHoraria(5);
        UARTprintf("Movimiento 12 <-----> Movimiento 6\n");
    }
}
```

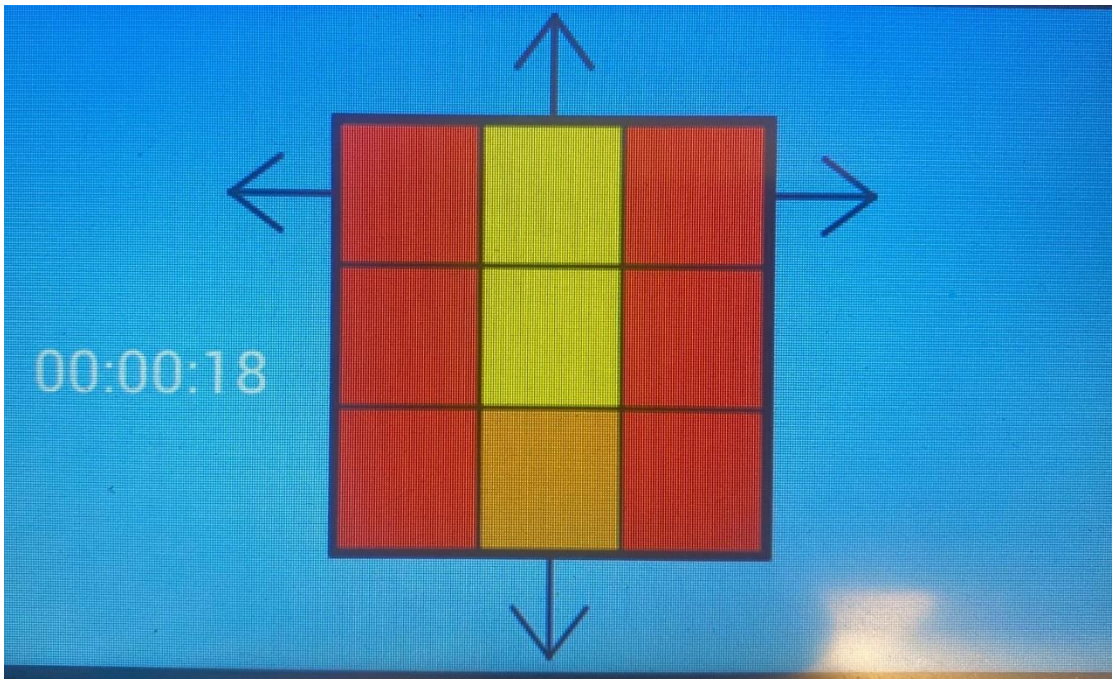
4. Manual de Usuario

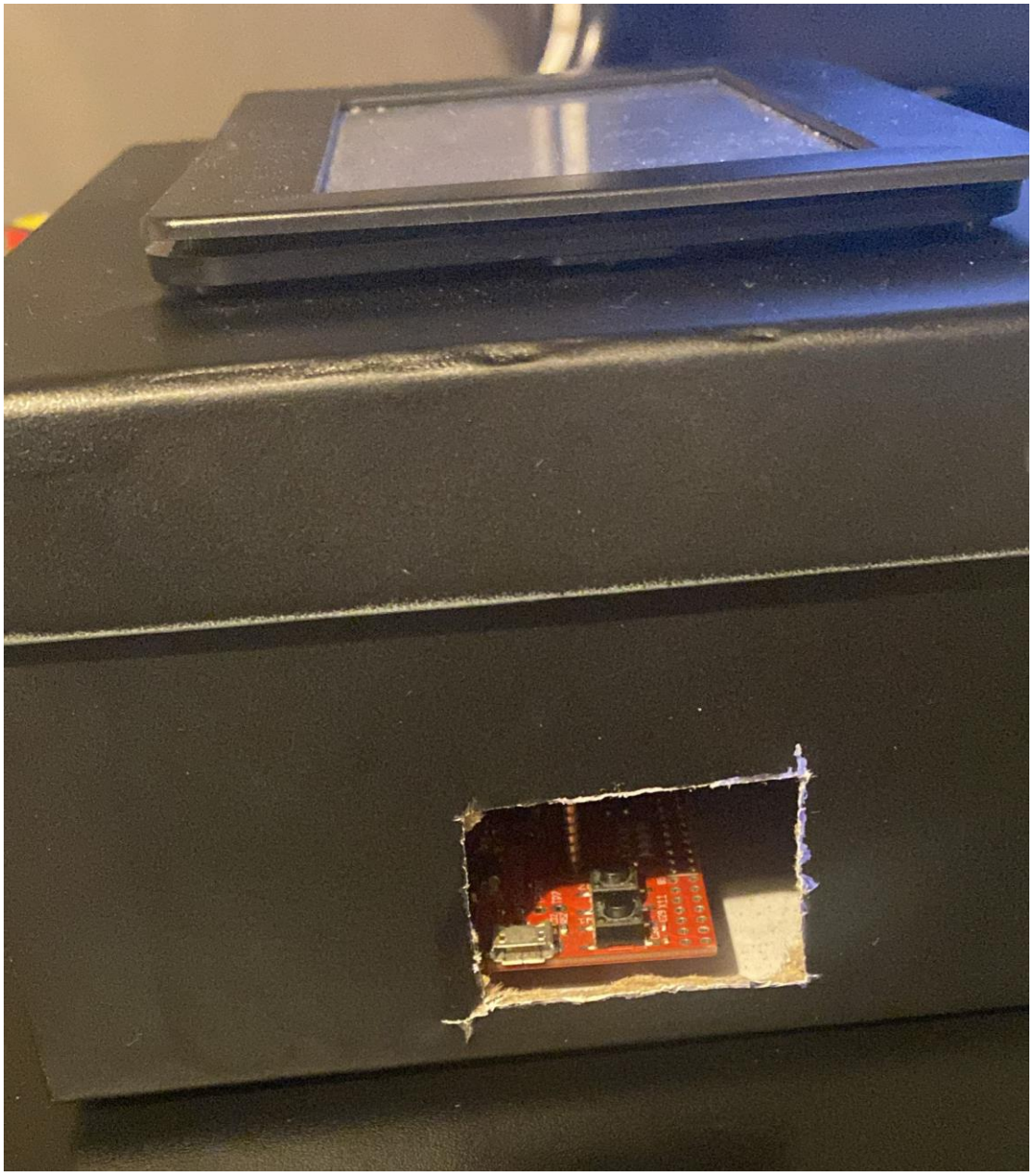
En la primera pantalla observamos lo siguiente:



Simplemente seleccionamos el modo de juego deseado pulsando el botón correspondiente.

Dentro del juego podemos cambiar las caras seleccionadas inclinando la caja hacia un lado u otro, o también girar filas o columnas seleccionando primero el cuadrito correspondiente y después pulsando la flecha del giro deseado.





Aquí tenemos dos botones a la vista. El más lejano, nos alternará las perspectivas del juego entre dos, la cara principal, y las de alzado, planta y perfil. El más cercano es el que hace la comprobación del cubo. Si está bien realizado llegaremos a la pantalla de la victoria.

5. Conclusiones

Desarrollar un proyecto en C sobre un cubo de Rubik permite fortalecer habilidades en programación estructurada y manejo de estructuras de datos, además de profundizar en la implementación de algoritmos para resolver problemas. Impulsa la eficiencia en el uso de memoria y rendimiento, al tiempo que refuerza el pensamiento lógico y abstracto. También fomenta la creatividad al modelar y simular un objeto físico. Completar este proyecto demuestra dedicación, capacidad técnica y resolución de problemas complejos, siendo un valioso aprendizaje personal y profesional.