E.E.S.T.N°5:

Tecnicatura en informática personal y profesional.

Informe final de: Console Labyrinth

Curso: 7mo 1ra.

Taller: Praticas profecionales.

Profesor: Daniel Beltrami.

Alumnos:

Andre Alvarez.

Sebastian Almiron.

Sheila Baigorri.

Nahuel Espinosa .

INDICE:

Presentación del proyecto

1.1 INTRODUCCIÓN

En este documento se explaya el proceso de desarrollo y concepción del proyecto de videoconsola portátil diseñada desde cero, tanto a nivel de Hardware como de Software.

Dicho proyecto cumple con un propósito similar a una "Prueba final" para el último año de tecnicatura en informática profesional y personal con el objetivo de evaluar al aprendizaje y conocimientos adquiridos a lo largo de anteriores años de carrera cursados. Sin embargo, la planificación será pensada y organizada para un producto el cual tenga un alcance mayor y pueda ser comercializado.

1.2 PROBLEMATICA PROPUESTA

Se pretende diseñar una videoconsola cual pueda ser vendida y comercializada como un producto orientado a niños en etapa de crecimiento. Con el objetivo de contribuir a su buen desarrollo motriz, dado que los juegos como el clásico laberinto, logran captar toda la atención y concentración del usuario. Además, con ayuda del sistema sensorizado de la consola se pone en práctica el dominio del propio cuerpo, estimulando los movimientos armoniosos de los músculos, la agilidad y el equilibrio.

1.3 SOLUCIÓN DE PROBLEMATICA

Dicha consola soportará un juego de laberinto en dos dimensiones, el cual podrá ser controlado con ayuda de un sistema de sensor de movimientos, permitiendo al usuario ejecutar simples movimientos de inclinación que serán procesados por la consola para cambiar la dirección del jugador.

Los laberintos estarán programados para generarse de forma aleatoria cada vez que se inicie una nueva partida.

A su vez, se elabora una memoria de proyecto en donde se desglosa el proceso de confección de este, para que posteriormente pueda ser consultada por terceras personas o un equipo de trabajo.

Historial de actividades:

2.1 Diagrama de Gantt:

Actividad:	Fechas de inicio:	Fechas de finalización:
Programación en C++.	15/5/2020	4/6/2020
Diseño del prototipo en arduino.	15/5/2020	11/6/2020
Composición de sonidos.	29/5/2020	25/6/2020
Programación en Arduino.	5/6/2020	11/6/2020
Testeo digital del prototipo en Arduino.	12/6/2020	16/7/2020
Creación de la carcasa para la consola.	11/7/2020	19/8/2020
Creación del manual de instrucciones para el prototipo final.	26/8/2020	2/9/2020
Creación de la documentación final.	26/8/2020	4/11/2020
Impresión de la carcasa.	2/9/2020	9/9/2020
Creación de texturas.	21/10/2020	28/10/2020
Implementación de texturas en Arduino.	29/10/2020	11/11/2020
Adquisición de los materiales.	17/2/2021	3/3/2021
Armado del prototipo	4/3/2021	31/3/2021

Componentes y herramientas utilizadas:

3.1 Software:

- ### 3.1.1 Visual Studio:

  Entorno de desarrollo integrado (IDE, por sus siglas en inglés) para Windows y macOS. Es compatible con múltiples lenguajes de programación, tales como C++, C#, Visual Basic .NET, F#, Java, Python, Ruby y PHP, al igual que entornos de desarrollo web, como ASP.NET MVC, Django, etc., a lo cual hay que sumarle las nuevas capacidades en línea bajo Windows Azure en forma del editor Mónaco.

  Visual Studio permite a los desarrolladores crear sitios y aplicaciones web, así como servicios web en cualquier entorno compatible con la plataforma .NET (a partir de la versión .NET 2002).

- ### 3.1.2 Arduino IDE:

  El entorno de desarrollo integrado (**IDE**) de Arduino es una aplicación multiplataforma (para Windows, macOS, Linux) que está escrita en el lenguaje de programación Java. Se utiliza para escribir y cargar programas en placas compatibles con Arduino, pero también, con la ayuda de núcleos de terceros, se puede usar con placas de desarrollo de otros proveedores.

- ### 3.1.3 FL studio:

  FL Studio es una estación de trabajo de audio digital con las características de editor de audio, secuenciador con soporte multipista y MIDI utilizado para la producción musical y desarrollado por la compañía belga Image-line Software.

- ### 3.1.4 Photoshop:

  Adobe Photoshop es un editor de fotografías desarrollado por Adobe Systems Incorporated. Usado principalmente para el retoque de fotografías y gráficos. Fue creado en 1986 por los hermanos Thomas Knoll y John Knoll. Photoshop puede editar y componer imágenes rasterizadas y soporta varios modelos de colores: RGB, CMYK, CIELAB, colores sólidos y semitonos. Photoshop usa sus propios formatos de archivo PSD y PSB para soportar estas características**.**

- ### 3.1.5 Proteus Design Suite:

  Proteus Design Suite es software de automatización de diseño electrónico, desarrollado por Labcenter Electronics Ltd., que consta de los dos programas principales: Ares e Isis, y los módulos VSM y Electra.

- ### 3.1.6 Tinkercad 3D:

  TinkerCAD es ofrecer un software gratuito online de diseño e impresión 3D creado por la empresa Autodesk. Simple y destinado para todos los públicos independientemente de su nivel de conocimientos, su edad y para qué quieran destinar sus creaciones**.**

3.2 Hardware:

- ### 3.2.1 Placa Arduino MEGA:

  Arduino Mega es una placa de microcontrolador de código abierto basado en el microchip ATmega2560.La placa está equipada con conjuntos de pines de E/S digitales y analógicas que pueden conectarse a varias placas de expansión y otros circuitos."

  ESPECIFICACIONES TÉCNICAS:

  Microcontrolador: Microchip ATmega2560.

  Voltaje de funcionamiento: 5 voltios

  Voltaje de entrada: 7 a 20 voltios

  Pines de E/S digitales: 54 (de los cuales 15 proporcionan salida PWM)

  Pines de entrada analógica: 16

  Corriente DC por Pin de E/S: 20 mA

  Corriente CC para Pin de 3.3V: 50 mA

  Memoria Flash: 256 KB de los cuales 8 KB utilizados por el gestor de arranque

  SRAM: 8 KB

  EEPROM: 4 KB

  Velocidad del reloj: 16 MHz

  Longitud: 101,52 mm

  Ancho: 53,3 mm

  Peso: 37 g

Cable Arduino:

USB Para conectar Arduino al equipo.

Voltaje de funcionamiento: 5V

- **3.2.2 Pantalla LCD TFT 2.8:**

  Este Shield te permite agregar una interfaz hombre-máquina moderna y profesional, su pantalla Lcd a colores y táctil son ideales para la visualización e ingreso de datos para tus proyectos. El panel tiene integrado un controlador gráfico ILI9341 y un controlador táctil. El LCD se controla mediante 8 pines de datos (2-9) y 5 pines de control (A0-A4), la pantalla táctil comparte el bus de datos del Lcd para enviar la posición. El módulo dispone también de una ranura de conexión para una tarjeta SD con interfaz SPI.

  ESPECIFICACIONES TÉCNICAS:

  Voltaje de Operación: 5V DC

  Interfaz LCD: Data (8) Control (5)

  Interface MicroSD: SPI (SS, DI, DO, SCK)

  Nivel lógico de SPI: 5V

  Controlador de pantalla: ILI9341

  Controlador de pantalla con buffer de video incluido

  Resolución de la pantalla: 240×320 pixeles.

  Regulador de 3.3 Volts incluido

  Pantalla LCD TFT de 2,8”

  65536 colores (16 bit: R5G6B5)

- ## 3.2.3 Sensor Giroscopio MPU 6050:

"El MPU6050 es un dispositivo que incluye un acelerómetro de 3 ejes y un giroscopio de 3 ejes en un solo circuito integrado. Además, incluye la tecnología Digital Motion Processor™ (DMP™), un “coprocesador” de movimiento que permite ejecutar algunos algoritmos esenciales y reducir la carga en el procesador principal, todo esto en un pequeño encapsulado tipo QFN"

ESPECIFICACIONES TÉCNICAS:

• Basado en el circuito integrado MPU-6050

• Alimentación: 3 a 5 VDC

• Rango del Giroscopio: 250, 500, 1000, 2000 °/s

• Rango del Acelerómetro: ±2, ±4, ±8, ±16 g

• Comunicación: I2C

• Conversor ADC 16 Bits (salida digital)

• Regulador Integrado

• Tamaño: 20 x 16 x 3 mm

- **3.2.4 DF Player Mini:**

  Placa con un pequeño reproductor de audio MP3 con amplificador integrado y que puede funcionar por si sólo simplemente conectando unos pulsadores que permiten la reproducción de archivos directamente cargados en una tarjeta de memoria Micro SD. Para su alimentación simplemente se debe conectar a una fuente de alimentación externa de 5V, poniendo el Gnd en común con Arduino. Por otro lado, se conectan los pines RX y TX del puerto serie a Arduino. Por último, el altavoz al DFPlayer Mini a los pines Spk_1 y Spk_2.

  ESPECIFICACIONES TÉCNICAS:

  Alimentación: 3 a 5 VDC

  Dimensiones: 21x21x12 mm.

  Peso: 4 g.

  Marca: DFRobot.

  Salida: Soporte para rango dinámico de 90 db y 48 kHz.

  Lector de memoria: tarjetas Micro SD de hasta 32 GB.

- **3.2.5 Altavoz:**

  Parlante CICLOS YD50 de 4 Ohm

  ESPECIFICACIONES TÉCNICAS:

  Diámetro: 50mm

  Resistencia: 4 ohm ± 15%

  Frecuencia de resonancia: 500 ± 20Hz

  Potencia de salida: 5 W

- **3.2.6 Memoria micro SD:**

  Memoria flash de tamaño reducido que permite la lectura y escritura de múltiples posiciones de memoria en la misma operación.

  ESPECIFICACIONES TÉCNICAS:

  Dimensiones: 15×11×1 milímetros

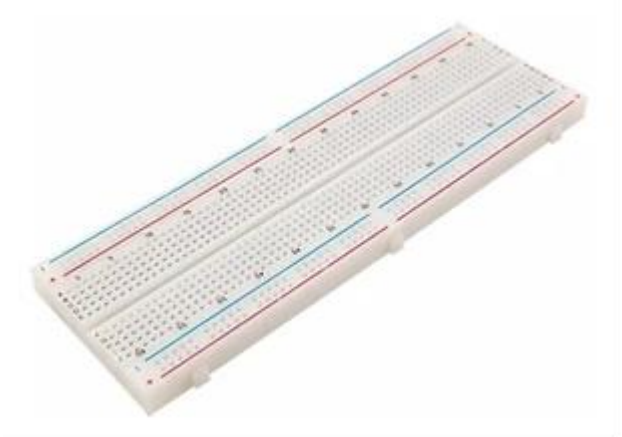
  Marca: Kingston

  Capacidad: 32 GB

  Tasa de transferencia: 10 Mbit/s

- **3.2.7 Protoboard:**

  Experimentador de circuitos electrónicos, para probar circuitos sin tener que armar la placa común de cobre.



- **3.2.8 Cables Macho-Macho:**

  Cables tipo jumper macho-macho de 20cm c/u, diferentes colores.

  Para uso en protoboard y prototipos en general.

Historial de tareas:

4.1 Creación de la idea:

La idea principal proviene de poder hacer un videojuego que no fuera lineal y que, a su vez, al terminarlo, se pudiera jugar otra vez ya que sería diferente, así el usuario no se aburriría de siempre tener los mismos niveles.

Con la idea principal en mente, se optó, por hacer una recreación de un clásico juego de empezar en una salida y llegar a una meta atravesando un laberinto, pero cada partida nueva que se empezara el laberinto sería diferente al anterior.

Tras esto, al pensar como poder unir este concepto con el hecho del uso de hardware unido con el uso de microcontroladores, se procedió a investigar sobre recursos para poder crear movimiento en el personaje, pero que no fuera los típicos que se usan en el mercado tales como joysticks o teclados y mouse. Con un poco más de investigación se llegó a la conclusión de que el personaje podría moverse con el movimiento de un giroscopio.

El giroscopio tomaría entonces, la orientación así el lado al que fuera inclinado y movería al personaje.

Por el ultimo se siguió investigando acerca de pantallas y fuentes para la reproducción de canciones y se optó porque el juego se incluyera dentro de una consola propia que pudiera cumplir con todas las necesidades a la vez.

4.2 Implementación de la idea:

Con la idea en mente, lo siguiente fue llevarlo a cabo.

Debido a las circunstancias sanitarias mundiales, que impidieron la compra de materiales físicos, se procedió por hacer planificación previa, y realizar todos los esquemas necesarios, así como todos los testeos que pudiéramos realizar de manera digital para que al poder armar el prototipo hubiera menos complicaciones.

Como primera implementación del prototipo inicial, se usaría Arduino, un microcontrolador que permite construir dispositivos digitales y dispositivos interactivos que puedan detectar y controlar objetos del mundo real, lo que nos permitiría tener una idea de cómo funcionarían los componentes antes de cambiar a otro microcontrolador.

4.3 Programación en C++:

Para esta parte antes de hacer uso de Arduino, se creó un programa de ejemplo en C++ para planificar la lógica de como funcionaria el juego.

Para que el juego pudiera crear un laberinto aleatorio satisfactoriamente, del cual se pueda salir, se crearon pequeñas celdas prediseñadas con paredes y caminos, que luego serán seleccionadas aleatoriamente algunas de ellas, y se ubicarán una tras otra y crearán un laberinto.

Con este concepto, obviamente, no se asume que la selección de paredes y caminos sea la ideal para que exista una conexión entre la salida y la meta. Por lo que luego de esta selección de celdas, el programa intentara recorrer el laberinto desde la salida hasta la meta. Cuando el programa se dé cuenta de que no puede seguir avanzando por ningún camino, optara por crear un camino entre la celda más cercana a la que haya llegado de la meta, y una celda adyacente. Este proceso se repetirá hasta llegar a la meta, asegurando que el jugador cada vez que juegue un nuevo nivel podrá completarlo.

4.4 Investigación de las conexiones de los componentes a usar:

Al mismo tiempo que se realizaba el programa de ejemplo en C++, y con los componentes ya seleccionados se procedio a investigar sobre sus conexiones con respecto a Arduino.

PANTALLA LCD TFT DE 2.8”:

El controlador posee un buffer **RAM**, disminuyendo los requerimientos del microcontrolador a usar. Este Shield puede trabajar perfectamente con Arduino Uno, Mega, Due, Leonardo.

En la mayoría de los casos la comunicación con la pantalla TFT se realiza a través del bus SPI debido al alto volumen de datos que requieren.

Para la conexión de este componente deben ocuparse los pines **5V** y **GND** para la alimentación desde la placa Arduino, también se usarán pines de datos, en base a la tabla sobre el bus **SPI** con designaciones habituales para los pines en distintos dispositivos, la siguiente tabla muestra los distintos pines que encontraremos en los distintos modelos de TFT:

Nombre	Alias	Pin(en Arduino Mega)	Descripcion
VCC			+3.3 ... 5 Volt
GND		Ground	Ground
SCLK	CLK/SCK/SCLK	D52(hardware)	Clock (SPI)
MISO	MISO/SDO/DOUT	D50(hardware)	Master In Slave Out (SPI)
MOSI	MOSI/SDI/DIN	D51 (hardware)	Master Out Slave In (SPI)
SS	SS/CS/SDA	D53(hardware, solo en esclavo)	Slave/Chip Select (SPI)
RES	RST/RES/REST	D42(variable, se fija por software)	Controller Reset
RS	RS/DC	D43 (variable, se fija por software)	Mode: Command/Data

Sensor Giroscopio MPU 6050:

El módulo Acelerómetro **MPU** tiene un giroscopio de tres ejes con el que podemos medir velocidad angular y un acelerómetro también de 3 ejes con el que medimos los componentes **X**, **Y** y **Z** de la aceleración.

La comunicación del módulo es por **I2C**, esto le permite trabajar con la mayoría de microcontroladores. Los pines **SCL** y **SDA** tienen una resistencia pull-up en placa para una conexión directa al microcontrolador o Arduino.

Conexiones entre Arduino y el MPU6050

Las conexiones son del modo I2C estándar:

MPU6050	Arduino Uno, Nano, Mini
VCC	5V
GND	GND
SCL	D21 (SCL)
SDA	D20 (SDA)
INT	D2

DFPLAYER Mini:

El DFPlayer Mini es un reproductor de audio de bajo coste y pequeño que podemos conectar a un procesador como Arduino para reproducir audio en formato MP3, WMA, WAV.

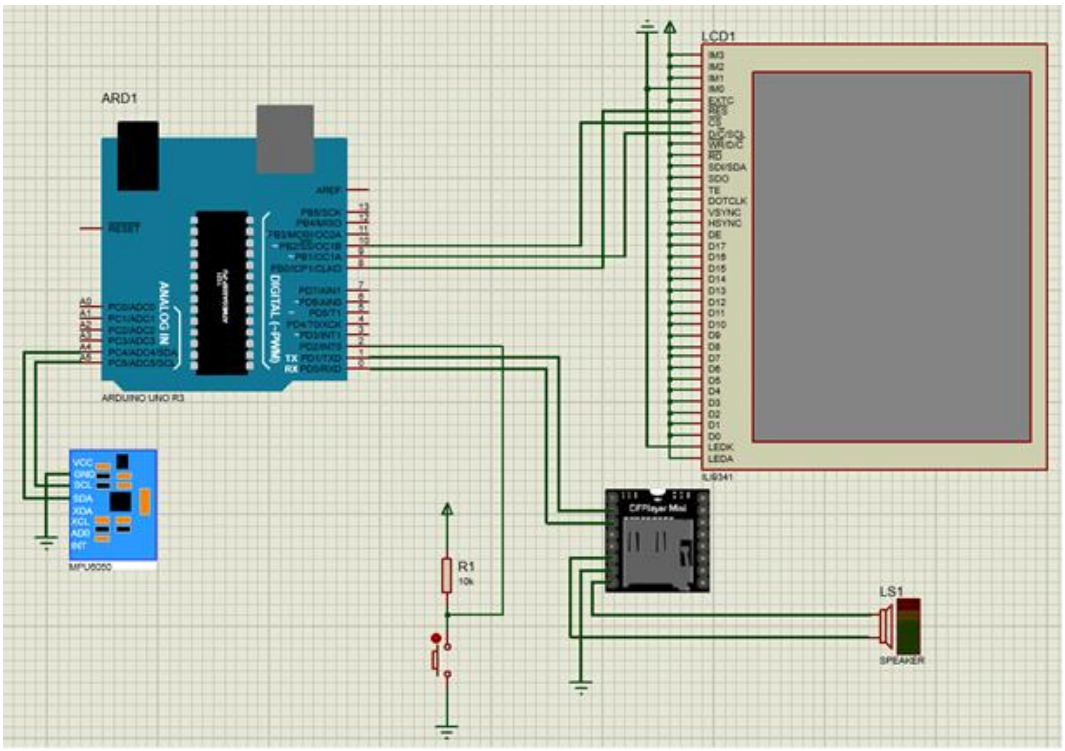
Arduino no tiene potencia potente para reproducir un archivo comprimido como un MP3. Por mucho que lo intentéis, o incluso si empleáis un procesador más potente, no es función de un autómata realizar estas tareas. Es mucho mejor delegar en un subprocesador específico.

Conexiones entre Arduino y el DFPLAYER MINI

DFPLAYER MINI	Arduino Uno, Nano, Mini
VCC	5V
RX	D11(TX)
TX	D10(RX)
SPK_1	CONECTOR IZQ DEL PARLANTE
GND	GND
SPK_2	CONECTOR DER DEL PARLANTE

4.5 Elaboración de un esquema electrónico inicial:

Con las conexiones de los componentes al Arduino lo siguiente fue armar un esquema funcional de los mismo en Proteus, una herramienta diseñada para lo mismo y que a su vez, permite simularlos luego de cargarles el código del Arduino.



4.6 Creación y composición de música:

Para esta tarea se ha empleado el Programa FL Studio, y se han creado varios efectos de sonido y temas musicales para el juego.

4.7 Programación en Arduino:

Con la implementación lógica que se creó previamente en C++, lo único restante fue, crear ese mismo programa, pero usando el lenguaje de programación Arduino. A su vez para el correcto funcionamiento del siguiente código se utilizaron bibliotecas externas al IDE de Arduino standard de Arduino.

Bibliotecas:

DF Player Mini.h

MPU6050.h

Time.h

Función setup: La función setup es la primera función que el Arduino ejecuta. En ella se guardan los assets o celdas con caminos y paredes prediseñados y luego generará el laberinto mediante strings elegiendo de manera aleatoria estos assets, concatenándolos de manera que cada string simulé una línea completa. Luego llamara a distintas funciones del programa para que estos strings creen un laberinto que sea posible de completar y pasara toda la información previamente creada y procesada a la pantalla TFT.

Función generar salida: Esta función se encarga de generar una entrada y una salida en paredes opuestas.

Función DFS: Ejecuta el algoritmo de búsqueda DFS (Búsqueda en Profundidad o Depth First Search), que busca dentro del laberinto si existe un camino entre la salida y la meta. Si lo encuentra termina su ejecución y el programa sigue. En caso de no hacerlo ejecutara la función destruir paredes.

Función Destruir paredes: Esta función se encargará de crear caminos entre assets adyacentes entre sí, para que el jugador pueda llegar a la meta.

Función loop: La función loop de Arduino, será la encargada de registrar cada movimiento que el usuario haga, mediante el sensor de MPU y representarlo en la pantalla moviendo al personaje.

Función terminar: Por último, si el jugador llega a la meta o no se ejecutará esta función, que dependiendo del resultado mostrará si ganó o perdió.

Funciones de dibujado: son funciones que se encargarán de dibujar las texturas que se mostrarán en la pantalla, tales como arbol, pasto1, pasto2, flor1, flor2, personaje.

Código:

```
#include "SPI.h"
#include "Adafruit_GFX.h"
#include "Adafruit_ILI9341.h"
#include <math.h>
#include "MPU6050.h"
#include <SoftwareSerial.h>
#include "DFRobotDFPlayerMini.h"

const int Rx = 10; // Pin10 -> RX,
const int Tx = 11; // Pin11 -> TX
SoftwareSerial mySerial(Rx,Tx);

#define TFT_cs   53 //SS
#define TFT_rst  42 //RESET
#define TFT_dc   43 //D/C
#define TFT_mosi 51 //SDI(MOSI)
#define TFT_sclk 52 //SCK
#define TFT_miso 50 //SDO(MISO)

Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_cs, TFT_dc, TFT_mosi, TFT_sclk, TFT_rst, TFT_miso);
DFRobotDFPlayerMini myMP3;

int y = 15, op;
int dv = y + 2;
int tamx, tamy,dotx,doty,salix,saliy,startx,starty;
int matriz[17][17];
int dibux = tft.width() / dv, dibuy = tft.height() / dv;
bool termine=false;
int cancion,cantcan=2;

void arbol(int lx, int ly);
void pasto1(int lx, int ly);
void pasto2(int lx, int ly);
void flor1(int lx, int ly);
void flor2(int lx, int ly);
void personaje (int lx, int ly);
void terminar ();
void destruirparedes(int x,int y);
void generarsalida();

double mini=0;
int xbreak,ybreak;
bool findfs=false, unavez=false;

long randNumber;
```

```
struct assets {

  String linea1;
  String linea2;
  String linea3;
};

assets celdas[16];
int nivel=1;
String adyacencia[17];

MPU6050 mpu;

void setup() {

  int f,i;

  for(f=0;f<17;f++)
    adyacencia[f]="";

  for(f=0;f<17;f++)
    for(i=0;i<17;i++)
      matriz[f][i]=-1;

  if(unavez==false)
  {
  tft.begin();
  tft.fillScreen(ILI9341_GREEN);
  tft.setCursor(tft.width() /6 ,tft.height()/2);
  tft.setTextColor(ILI9341_BLACK);
  tft.setTextSize(2);
  tft.println("CREANDO NIVEL");

  Serial.begin(9600);
  mySerial.begin(9600);
  myMP3.begin(mySerial);

  randomSeed(analogRead(A0));
  mpu.initialize();

  myMP3.setTimeOut(500);
  myMP3.volume(20);
  myMP3.EQ(DFPLAYER_EQ_NORMAL);
  myMP3.outputDevice(DFPLAYER_DEVICE_SD);

  pinMode(2,INPUT);
```

```
tamx = dibux / 5;
tamy = dibuy / 5;

celdas[0].linea1= "o o";
celdas[0].linea2= "   ";
celdas[0].linea3= "o o";

celdas[1].linea1= "o o";
celdas[1].linea2= "o o";
celdas[1].linea3= "o o";

celdas[2].linea1= "ooo";
celdas[2].linea2= "   ";
celdas[2].linea3= "ooo";

celdas[3].linea1= "ooo";
celdas[3].linea2= "o  ";
celdas[3].linea3= "o o";

celdas[4].linea1= "o o";
celdas[4].linea2= "o  ";
celdas[4].linea3= "ooo";

celdas[5].linea1= "o o";
celdas[5].linea2= "  o";
celdas[5].linea3= "ooo";

celdas[6].linea1= "ooo";
celdas[6].linea2= "  o";
celdas[6].linea3= "o o";

celdas[7].linea1= "o o";
celdas[7].linea2= "   ";
celdas[7].linea3= "ooo";

celdas[8].linea1= "o o";
celdas[8].linea2= "  o";
celdas[8].linea3= "o o";

celdas[9].linea1= "ooo";
celdas[9].linea2= "   ";
celdas[9].linea3= "o o";

celdas[10].linea1= "o o";
celdas[10].linea2= "o  ";
celdas[10].linea3= "o o";
```

```
celdas[11].linea1= " o ";
celdas[11].linea2= " o ";
celdas[11].linea3= " o ";

celdas[12].linea1= "ooo";
celdas[12].linea2= " o ";
celdas[12].linea3= "   ";

celdas[13].linea1= "   ";
celdas[13].linea2= " o ";
celdas[13].linea3= "ooo";

celdas[14].linea1= "  o";
celdas[14].linea2= " oo";
celdas[14].linea3= "  o";

celdas[15].linea1= "o  ";
celdas[15].linea2= "oo ";
celdas[15].linea3= "o  ";

unavez=true;

}

adyacencia[0]= "oooooooooooooooo";
adyacencia[16]= "oooooooooooooooo";

for(nivel=1;nivel<=15;nivel+=3)
 {
 adyacencia[nivel]+="o";
 adyacencia[nivel+1]+="o";
 adyacencia[nivel+2]+="o";

 for(int f=0;f<5;f++)
    {
     randNumber = random(16);
     adyacencia[nivel]+=celdas[randNumber].linea1;
     adyacencia[nivel+1]+=celdas[randNumber].linea2;
     adyacencia[nivel+2]+=celdas[randNumber].linea3;}

  adyacencia[nivel]+="o";
  adyacencia[nivel+1]+="o";
  adyacencia[nivel+2]+="o";
 }

 generarsalida();
```

```
 int cont=0;

for(f=0;f<y+2;f++)
 {for(i=0;i<y+2;i++)
  {if(adyacencia[f][i]=='o')
    matriz[f][i]=2;
   else
  if(adyacencia[f][i]==' ')
    matriz[f][i]=0;
   else
  if(adyacencia[f][i]=='E')
    {matriz[f][i]=3;
    startx=dotx=f;
    starty=doty=i;}
  else
  if(adyacencia[f][i]=='S')
    {matriz[f][i]=4;
    salix=f;
    saliy=i;}

  }
 }

 termine=false;
 findfs=false;

while(termine==false){

mini=0;

dfs(dotx,doty);
int distanciay,distanciax,movimiento;

distanciax=max(xbreak,salix)-min(xbreak,salix);
distanciay=max(ybreak,saliy)-min(ybreak,saliy);

if(distanciay <= distanciax)
    {if(salix-xbreak <= 0)
       movimiento=4;
    else
       movimiento=3;
    }
 else
    {if(saliy-ybreak <= 0)
       movimiento=2;
    else
```

```
        movimiento=1;
      }

 dotx=xbreak;
 doty=ybreak;

 destruirparedes(xbreak,ybreak,movimiento);
}

  tft.fillRect(tft.width() /6 ,tft.height()/2,tamx * 13*6,tamy * 5,ILI9341_GREEN);

 for (f = 0; f < y + 2; f++)
 {for (i = 0; i < y + 2; i++)
  {if (adyacencia[f][i] == 'o')
   {
     cont=random(5);

     if (cont == 0)
       arbol(i * dibux, f * dibuy);
     else
     if (cont == 1)
       pasto1(i * dibux, f * dibuy);
     else
     if (cont == 2)
       flor2(i * dibux, f * dibuy);
     else
     if (cont == 3)
       flor1(i * dibux, f * dibuy);
     else
     if (cont == 4)
       pasto2(i * dibux, f * dibuy);
   }
   else if (adyacencia[f][i] == ' ' || adyacencia[f][i] == 'E')
     {tft.fillRect((i * dibux)-tamx, (f * dibuy)-tamy, tamx * 8, tamy * 6 + 1, tft.color565(98, 98,
71));
     }
   else if (adyacencia[f][i] == 'S')
     {meta(i * dibux, f * dibuy);}
  }
 }

 cont=0;

 dotx=startx;
 doty=starty;

 personaje((doty * dibux),(dotx * dibuy));
```

```
 cancion=random(1,cantcan);

 myMP3.play (cancion);
}

void dfs (int x,int y){

 if(findfs==true)
  return;

 int cont=0;

 int posx[]={1,0,-1,0};
 int posy[]={0,1,0,-1};
 int auxy=17;

 for(int f=0;f<4;f++)
    {if(x+posx[f]<auxy && x+posx[f]>=0 && y+posy[f]<auxy && y+posy[f]>=0 )
       {if(matriz[x+posx[f]][y+posy[f]]==0)
          {matriz[x+posx[f]][y+posy[f]]=1;
           cont++;
           dfs(x+posx[f],y+posy[f]);}
       }

     if(x+posx[f]==salix && y+posy[f]==saliy)
         {findfs=true;
         termine=true;
         return;}
    }

    if(findfs==true)
      return;

    if(cont==0)
      {int x2=salix,y2=saliy,f;
      double res;

      res=sqrt((pow((x2-x),2)+pow((y2-y),2)));

      if(mini==0 || res<mini)
       {mini=res;
       xbreak=x;
       ybreak=y;}
      }
}
```

```
void destruirparedes(int xbreak, int ybreak, int movimiento){

  for(int f=0;f<4;f++)
      {switch (movimiento)
        {
           case 1:{
                 if(ybreak+f<y+1)
                    {adyacencia[xbreak][ybreak+f]=' ';
                    matriz[xbreak][ybreak+f]=0;
                    }
                 break;}

           case 2:{if(ybreak-f>0)
                    {adyacencia[xbreak][ybreak-f]=' ';
                    matriz[xbreak][ybreak-f]=0;
                    }
                 break;}

           case 3:{if(xbreak+f<y+1)
                   {adyacencia[xbreak+f][ybreak]=' ';
                    matriz[xbreak+f][ybreak]=0;
                    }
                 break;}

           case 4:{if(xbreak-f>0)
                    {adyacencia[xbreak-f][ybreak]=' ';
                    matriz[xbreak-f][ybreak]=0;
                    }
                 break;}
        }
      }
}

void loop() {

   int ax,ay,az;
   int gx,gy,gz;
   String op;

   mpu.getAcceleration(&ax, &ay, &az);
   mpu.getRotation(&gx, &gy, &gz);

   int numax,numay,numaz;
   int numgx,numgy,numgz;

   numax=ax/100;
   numay=ay/100;
```

```
  numaz=az/100;
  numgx=gx/100;
  numgy=gy/100;
  numgz=gz/100;

  if(numax>=60)
   op="A";
  else
  if(numay>=60)
   op="S";
  else
  if(numay<=-1)
   op="W";
  else
  if(numax<=-75)
   op="D";
  else
   op="Q";

  if(op=="A")
    { if(doty-1>=0)
          if(adyacencia[dotx][doty-1]==' '||adyacencia[dotx][doty-
1]=='E'||adyacencia[dotx][doty-1]=='S')
            {tft.fillRect((doty * dibux)-tamx, (dotx * dibuy)-tamy, tamx * 8, tamy * 6 + 1,
tft.color565(98, 98, 71));
              adyacencia[dotx][doty]=' ';
              doty--;
              adyacencia[dotx][doty]='P';
              personaje((doty * dibux),(dotx * dibuy));
            }

    delay(1000);}

  if(op=="W")
    { if(dotx-1>=0)
       if(adyacencia[dotx-1][doty]==' '||adyacencia[dotx-1][doty]=='E'||adyacencia[dotx-
1][doty]=='S')
                  { tft.fillRect((doty * dibux)-tamx, (dotx * dibuy)-tamy, tamx * 8, tamy * 6 + 1,
tft.color565(98, 98, 71));
                    adyacencia[dotx][doty]=' ';
                    dotx--;
                    adyacencia[dotx][doty]='P';
                    personaje((doty * dibux),(dotx * dibuy));
                  }
    delay(1000);}

  if(op=="D")
```

```
     {if(doty+1<y+2)
       if(adyacencia[dotx][doty+1]==' '||adyacencia[dotx][doty+1]=='E'||adyacencia[dotx][doty+1]=='S')
                    { tft.fillRect((doty * dibux)-tamx, (dotx * dibuy)-tamy, tamx * 8, tamy * 6 + 1, tft.color565(98, 98, 71));
                       adyacencia[dotx][doty]=' ';
                       doty++;
                       adyacencia[dotx][doty]='P';
                       personaje((doty * dibux),(dotx * dibuy));
                    }
     delay(1000);}

   if(op=="S")
     {if(dotx+1<y+2)
          if(adyacencia[dotx+1][doty]==' '||adyacencia[dotx+1][doty]=='E'||adyacencia[dotx+1][doty]=='S')
                 {   tft.fillRect((doty * dibux)-tamx, (dotx * dibuy)-tamy, tamx * 8, tamy * 6 + 1, tft.color565(98, 98, 71));
                     adyacencia[dotx][doty]=' ';
                     dotx++;
                     adyacencia[dotx][doty]='P';
                     personaje((doty * dibux),(dotx * dibuy));
                     }
     delay(1000);}

   if(op=="Q")
    {delay(1000);}

         if(dotx==salix && doty==saliy)
          {terminar();}

}

void generarsalida(){

   int pared1,pared2,entrada,salida,f;

   entrada=random(1,16);
   salida=random(1,16);

   pared1=random(1,5);

   for(f=0;f<4;f++)
      {switch (pared1)
        {
          case 1:{adyacencia[entrada][f]=' ';
                adyacencia[entrada][0]='E';
```

```
                pared2=2;
                break;}

          case 2:{adyacencia[entrada][y+1]='E';
                adyacencia[entrada][y+1-f]=' ';
                pared2=1;
                break;}

          case 3:{adyacencia[f][entrada]=' ';
                adyacencia[0][entrada]='E';
                pared2=4;
                break;}

          case 4:{adyacencia[y+1-f][entrada]=' ';
                adyacencia[y+1][entrada]='E';
                pared2=3;
                break;}
          }
     }

   for(f=0;f<4;f++)
     {switch (pared2)
       {
          case 1:{adyacencia[salida][f]=' ';
                adyacencia[salida][0]='S';
                break;}

          case 2:{adyacencia[salida][y+1-f]=' ';
                adyacencia[salida][y+1]='S';
                break;}

          case 3:{adyacencia[f][salida]=' ';
                adyacencia[0][salida]='S';
                break;}

          case 4:{adyacencia[y+1-f][salida]=' ';
                adyacencia[y+1][salida]='S';
                break;}
          }
     }
 }

void arbol(int lx, int ly) {
 tft.fillRect(lx + (tamx * 1), ly + 0, tamx * 4, tamy * 1, tft.color565(40, 100, 0));
 tft.fillRect(lx + (tamx * 0), ly + (tamy * 1), tamx * 6, tamy * 2, tft.color565(40, 100, 0));
 tft.fillRect(lx + (tamx * 2), ly + (tamy * 3), tamx * 2, tamy * 2, tft.color565(53, 33, 4));
}
```

```
void pasto1(int lx, int ly) {
  tft.fillRect(lx + 0, ly + (tamy * 1), tamx * 1, tamy * 4, tft.color565(0, 250, 0));
  tft.fillRect(lx + (tamx * 1), ly + (tamy * 2), tamx * 1, tamy * 3, tft.color565(159, 250, 21));
  tft.fillRect(lx + (tamx * 2), ly + (tamy * 4), tamx * 1, tamy * 1, tft.color565(159, 250, 21));
  tft.fillRect(lx + (tamx * 3), ly + (tamy * 3), tamx * 2, tamy * 2, tft.color565(159, 250, 21));
  tft.fillRect(lx + (tamx * 5), ly + (tamy * 2), tamx * 1, tamy * 3, tft.color565(159, 250, 21));
}

void pasto2(int lx, int ly) {
  tft.fillRect(lx + 0, ly + (tamy * 2), tamx * 5, tamy * 3, tft.color565(39, 193, 49));
  tft.fillRect(lx + (tamx * 1), ly + (tamy * 2), tamx * 3, tamy * 1, ILI9341_GREEN);
  tft.fillRect(lx + (tamx * 2), ly + (tamy * 3), tamx * 1, tamy * 1, ILI9341_GREEN);
}

void flor1(int lx, int ly) {
  int col1 = random(101);
  int col2 = random(101);
  int col3 = random(101);
  tft.fillRect(lx + (tamx * 2), ly + 0, tamx * 1, tamy * 3, tft.color565(175, col2, col3));
  tft.fillRect(lx + (tamx * 1), ly + (tamy * 1), tamx * 3, tamy * 1, tft.color565(175, col2, col3));
  tft.fillRect(lx + (tamx * 2), ly + (tamy * 1), tamx * 1, tamy * 1, ILI9341_YELLOW);
  tft.fillRect(lx + (tamx * 2), ly + (tamy * 3), tamx * 1, tamy * 2, tft.color565(20,50,0));
}


void flor2(int lx, int ly) {
  int col1 = random(101);
  int col2 = random(101);
  int col3 = random(101);
  tft.fillRect(lx + (tamx * 1), ly + 0, tamx * 1, tamy * 3, tft.color565(col1, col2, col3));
  tft.fillRect(lx + 0, ly + (tamy * 1), tamx * 3, tamy * 1, tft.color565(col1, col2, col3));
  tft.fillRect(lx + (tamx * 3), ly + (tamy * 2), tamx * 1, tamy * 3, tft.color565(col1, col2, col3));
  tft.fillRect(lx + (tamx * 2), ly + (tamy * 3), tamx * 3, tamy * 1, tft.color565(col1, col2, col3));
  tft.fillRect(lx + (tamx * 1), ly + (tamy * 1), tamx * 1, tamy * 1, ILI9341_YELLOW);
  tft.fillRect(lx + (tamx * 3), ly + (tamy * 3), tamx * 1, tamy * 1, ILI9341_YELLOW);
}

void meta (int lx, int ly) {
  lx-=2;
  ly-=2;
  tft.fillRect(lx + 0, ly + 0, tamx * 8, tamy * 6, tft.color565(100, 75, 0));
  tft.fillRect(lx + 0, ly + 0, tamx * 8, tamy * 3, ILI9341_BLACK);
  tft.fillRect(lx + (tamx * 1), ly + 0, tamx * 1, tamy * 1, ILI9341_WHITE);
  tft.fillRect(lx + (tamx * 3), ly + 0, tamx * 1, tamy * 1, ILI9341_WHITE);
  tft.fillRect(lx + (tamx * 5), ly + 0, tamx * 1, tamy * 1, ILI9341_WHITE);
  tft.fillRect(lx + (tamx * 7), ly + 0, tamx * 1, tamy * 1, ILI9341_WHITE);
```

```
  tft.fillRect(lx + 0, ly + (tamy * 1), tamx * 1, tamy * 1, ILI9341_WHITE);
  tft.fillRect(lx + (tamx * 2), ly + (tamy * 1), tamx * 1, tamy * 1, ILI9341_WHITE);
  tft.fillRect(lx + (tamx * 4), ly + (tamy * 1), tamx * 1, tamy * 1, ILI9341_WHITE);
  tft.fillRect(lx + (tamx * 6), ly + (tamy * 1), tamx * 1, tamy * 1, ILI9341_WHITE);
  tft.fillRect(lx + (tamx * 1), ly + (tamy * 2), tamx * 1, tamy * 1, ILI9341_WHITE);
  tft.fillRect(lx + (tamx * 3), ly + (tamy * 2), tamx * 1, tamy * 1, ILI9341_WHITE);
  tft.fillRect(lx + (tamx * 5), ly + (tamy * 2), tamx * 1, tamy * 1, ILI9341_WHITE);
  tft.fillRect(lx + (tamx * 7), ly + (tamy * 2), tamx * 1, tamy * 1, ILI9341_WHITE);
  tft.fillRect(lx + 0, ly + (tamy * 3), tamx * 1, tamy * 3, tft.color565(53, 33, 4));
  tft.fillRect(lx + (tamx * 7), ly + (tamy * 3), tamx * 1, tamy * 3, tft.color565(53, 33, 4));
}


void personaje(int lx,int ly){
   tft.fillRect(lx+(tamx * 1),ly,tamx * 4,tamy * 5,ILI9341_BLUE);
   tft.fillRect(lx,ly+(tamy * 1),tamx * 6,tamy * 3,ILI9341_BLUE);
   tft.fillRect(lx+(tamx * 1),ly+(tamy*1),tamx * 1,tamy * 1,ILI9341_WHITE);
   tft.fillRect(lx+(tamx * 4),ly+(tamy*1),tamx * 1,tamy * 1,ILI9341_WHITE);
   tft.fillRect(lx+(tamx * 2),ly+(tamy*3),tamx * 2,tamy * 1,ILI9341_WHITE);
}

void terminar (){
  tft.fillScreen(ILI9341_BLACK);
  tft.setCursor(tft.width() / 10-5,tft.height()/2-10);
  tft.setTextColor(ILI9341_WHITE);
  tft.setTextSize(2);
  tft.println("GRACIAS POR JUGAR");
  myMP3.play(4);
  delay(5000);
  tft.fillScreen(ILI9341_GREEN);
  tft.setCursor(tft.width() /6 ,tft.height()/2);
  tft.setTextColor(ILI9341_BLACK);
  tft.setTextSize(2);
  tft.println("CREANDO NIVEL");

  setup();
}
```

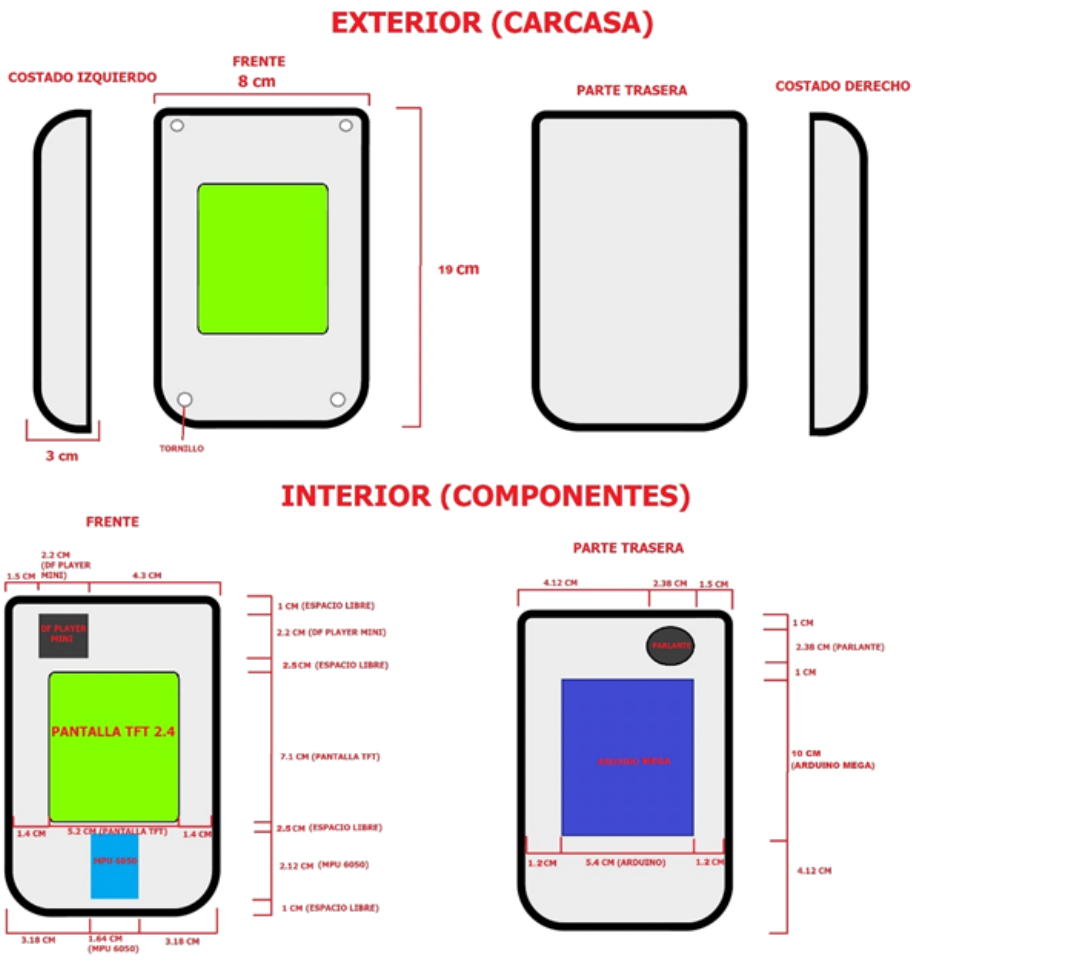
4.8 Creación de una carcasa:

Para revestir el circuito electrónico es imprescindible la implementación de una carcasa.

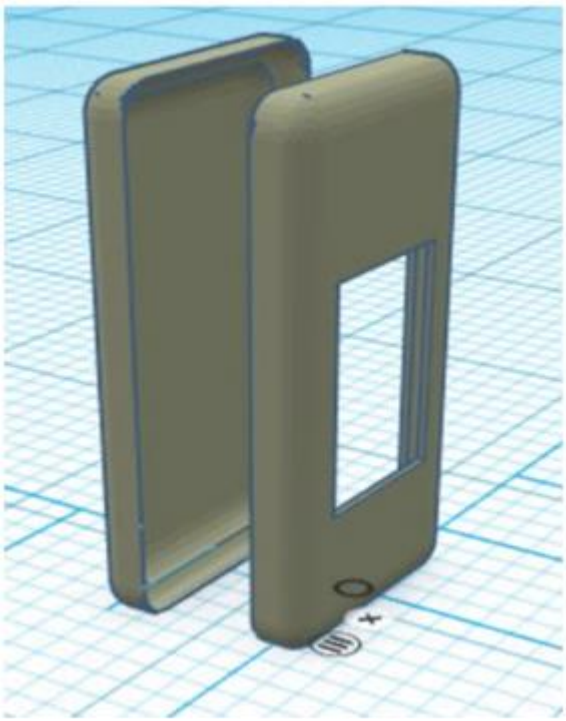
Para la creación de la misma, se ha utilizado como referencia el modelo de una Game Boy.

Game Boy pertenece a una serie de videoconsolas portátiles alimentadas con pilas y comercializadas por Nintendo (1989).

Teniendo en cuenta las dimensiones de los componentes y respetando el modelo de videoconsola portátil, se han esbozado planos en donde cada elemento cuenta con su respectiva ranura, además de varios orificios para tornillos:



Por último, se elaboró un modelo 3D de la carcasa con ayuda de los planos, utilizando la herramienta TinkerCAD. Para luego imprimirla en una impresora 3D.



4.9 Testeos digitales:

Como última tarea digital, se llevó a cabo testeos con el esquema electrónico hecho en Proteus, y el código de Arduino.

Tras arreglar varios bugs y mejorar el código para que fuera más eficiente, se obtuvo el código final, que se mostró previamente.

4.10 Compra de materiales y armado del prototipo:

Con la situación sanitaria mundial un poco mejor, se procedió a comprar, así como pedir prestado a algunos conocidos y docentes, los materiales y componentes necesarios para poder armar el prototipo.

Finalmente, con todos los materiales y componentes, se procedió a armar un prototipo funcional.

Presupuestos y comercialización

5.1 Presupuesto de Software

Tarea	Horas de trabajo totales	Precio
Programación en C++	32hs	$7500
Programación en Arduino.	16hs	$2500
Composición de música virtual.	12hs	$3000
Elaboración de esquema electrónico	4hs	$400
Creación del modelo 3D de la carcasa	3hs	$300
Creación de texturas	5hs	$250
Precio total del Software:		$16950

5.2 Presupuesto de Hardware

Componente	Precio
Kit placa Arduino MEGA y Cable	$2000
Sensor Giroscopio MPU 6050	$425
Protoboard	$250
Pantalla LCD TFT 2.8	$1990
Kit de 10 resistencias de 10k OHM	$187
Parlante	$400
Cables Macho-Macho	$187
Carcasa plástica	$300
Precio total del Hardware:	$5739

Precio total de productos y armado= $27689 ($184)

5.3 Comercialización del código:

Gracias a Internet, una forma de comercialización, que será semi gratuita, es el hecho de subir el código de Arduino, junto al plano de las conexiones, a una página desde la cual cualquier persona podría descargarlo y subir a su propia placa Arduino.

Lo único restante al usuario, sería que adquiera por sus medios cada uno de los componentes y materiales y pudiera armarlo.

Para poder realizar esta opción se requiere de 3 factores sumamente fundamentales:

1. Saber Arduino y poder conectar cada uno de los componentes correctamente.
2. Crear circuitos impresos.
3. Poder adquirir cada uno de los materiales.

5.4 Comercialización de la consola armada:

Debido a que, para poder concretar la comercialización anterior, se requieren de conocimientos previos, no todas las personas podrían adquirir dicho producto.

Por lo que la manera que podría llegar a más personas, sería, la producción en masa, lo que permitiría fabricar grandes cantidades del producto con maquinaría, facilitando su producción y a su vez minimizando posibles errores.

Se dispondrá de fábricas alrededor del mundo y la venta se haría a través de la página, y en conjunto con páginas de venta de productos online, se podría llegar a un acuerdo para utilizar sus servicios de distribución, para poder llegar a tiendas y jugueterías. También, se podría vender desde estas páginas.

Para poder vender en los países de Latinoamérica, se ubicaría una fábrica en Brasil, y se llegaría a un acuerdo de distribución con MercadoLibre.

En Norteamérica, la fábrica se ubicaría en Estados Unidos y su manera de distribución seria a través de Amazon NA.

Para poder vender en Europa, se ubicaría una fábrica en Alemania, y su manera de distribución seria a través de Amazon EU.

En Asia, se ubicaría una fábrica en China, y su distribución la realizaría Amazon CN.

Debido a la superficie extensa de Rusia, una manera más eficiente de poder vender el producto sin tener que cobrar grandes impuestos de exportación e importación, se optó porque tuviera su propia fabrica, y se llegaría a un acuerdo con WildBerries

Para África, KalahariNet se encargaría de la venta y distribución de la misma y su fábrica se ubicaría en Sudáfrica.

5.5 Precio y aclaración final de venta del producto:

El producto se vendería al precio de $250 dólares estadounidenses o $37500 pesos argentinos.

Aclaración:

Debido a la distribución del producto, en varios países el producto no incluye impuestos y su precio podría variar al hacer la conversión a su moneda local.

Impuestos y una variación del precio en su moneda local podrían y serán aplicados dependiendo el país y si el distribuidor así lo desea.

5.6 Ganancias:

El producto tiene un coste total de armado de $184 dólares y un margen de venta de $250 dólares, lo que generaría una ganancia de $66 dólares por consola vendida.

Sin embargo, si el producto fuese, producido en fábricas en grandes cantidades, la compra de productos se haría al mayor, el armado del mismo sería automatizado por maquinarias reduciendo su coste de armado en gran manera, y no se requeriría de escribir códigos personalizados por videoconsola ya que solo fue necesario escribir una vez.

De esta manera, el producto, tendría un costo de producción de $60 dólares, lo que generaría una ganancia de $190 dólares por consola vendida.

Manual de la consola:

Console Labyrinth

Manual de instrucciones

Español

INFORMACION DE SALUD Y SEGURIDAD:

Por favor, lea atentamente el contenido de salud y seguridad que se describe a continuación antes de usar el producto.

⚠ ADVERTENCIA - General

• No exponga el producto a temperaturas elevadas.

• No deje que el producto entre en contacto con líquidos o humedad. Evite tomar bebidas mientras utiliza el producto.

• No ejerza demasiada fuerza sobre el producto ni coloque objetos pesados sobre él.

⚠ ADVERTENCIA-Baterías

Si se produce una fuga de líquido de la batería, evite el contacto, por el contrario, si toca el líquido de la fuga con las manos, lávese con agua y jabón.

Si el líquido entra en contacto con los ojos, lávese inmediatamente con abundante agua y busque asistencia médica.

Para evitar la fuga de las baterías:

• No desarme, o intente reparar las baterías.

• No toque los terminales de la batería, ni provoque un corto entre los terminales con un objeto metálico.

•No remueva las baterías a menos que necesiten ser reemplazadas.

⚠ ADVERTENCIA-Mantenimiento

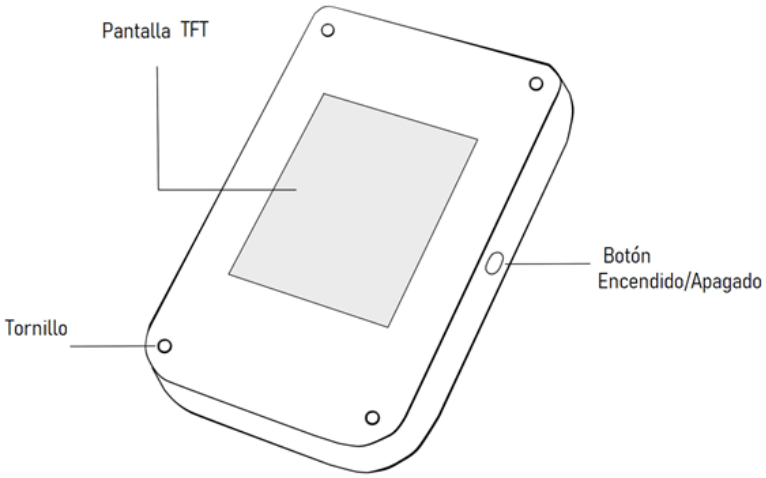
•No desarme o intente reparar el producto si se daña.

•Utilice un paño seco para limpiar el producto si se ensucia. No utilice agua, alcohol u otros químicos.

•No se deshaga del producto o de sus componentes tirándolos a la basura doméstica.

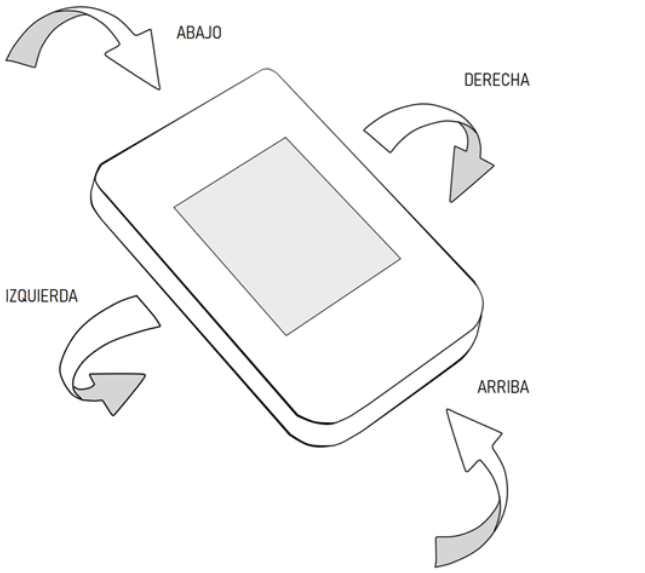
INSTRUCCIONES DE USO

1. Presione el botón ubicado en el lateral izquierdo de la consola para encenderla. Se mostrará en pantalla un laberinto y podrá comenzar a jugar



.

2. Estará ubicado en la entrada ("**E**"). Gire ligeramente la consola como se observa en la siguiente imagen ilustrativa para moverse dentro del juego y llegar a la meta ("**S**").



CÓMO REEMPLAZAR LAS BATERÍAS:

1. Desatornille cuidadosamente extrayendo todos los tornillos ubicados en la parte frontal de la carcasa para apartarla.

2. Retire la bateria de 9V y remplácelas por otras del mismo tipo.

3. Vuelva a colocar los tornillos para cerrar la carcasa.

SERVICIO DE ATENCIÓN AL CONSUMIDOR:

Para dudas o consultas, póngase en contacto con:

4nR34lvarez@gmail.com

o llame al: +54 115951-0696

ENGLISH

HEALTH AND SAFETY INFORMATION:

Please read the health and safety content described below carefully before using the product.

⚠ WARNING - General

• Do not expose the product to high temperatures.

• Do not let the product come into contact with liquid or moisture. Avoid drinking beverages while using the product.

• Do not use excessive force on the product or place heavy objects on it.

⚠ WARNING- Batteries

If liquid leaks from the battery, avoid contact, on the contrary, if you touch the leaking liquid with your hands, wash with soap and water.

If the liquid comes into contact with the eyes, rinse immediately with plenty of water and seek medical assistance.

To avoid battery leakage:

• Do not disassemble, or attempt to repair the batteries.

• Do not touch the battery terminals, or cause a short between the terminals with a metal object.

• Do not remove batteries unless they need to be replaced.

⚠ WARNING-Maintenance

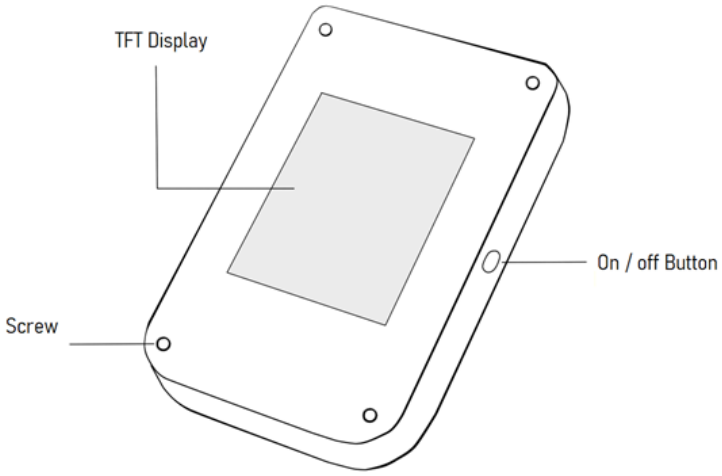
• Do not disassemble or attempt to repair the product if it is damaged.

• Use a dry cloth to clean the product if it gets dirty. Do not use water, alcohol or other chemicals.

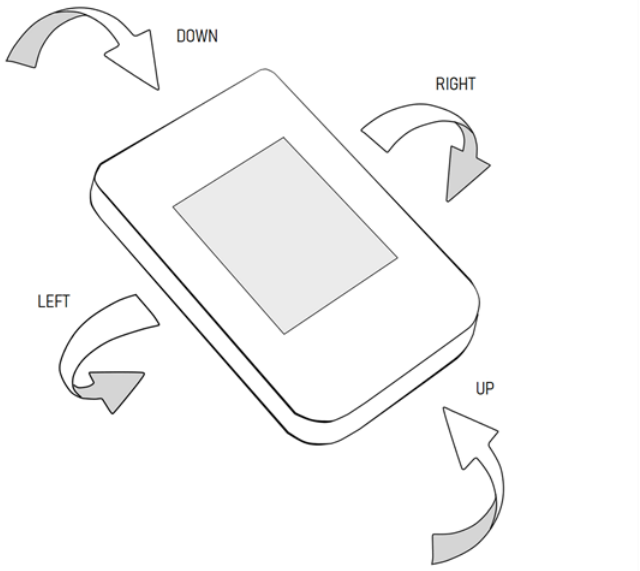
• Do not dispose of the product or its components by throwing them away with household waste.

INSTRUCTIONS FOR USE

1. Press the button located on the left side of the console to turn it on. A maze will be displayed on the screen and you can start playing.



2. You will be located at the entrance. Rotate the console slightly as shown in the following illustrative image to move within the game and reach the objetive.



HOW TO REPLACE THE BATTERIES:

1. Unscrew carefully by removing all screws located on the front of the case to set it aside.

2. Remove the 9V battery and replace them with others of the same type.

3. Replace the screws to close the case.

CONSUMER ATTENTION SERVICE:

For questions or queries, please contact:

4nR34lvarez@gmail.com

or call: +54 115951-0696

Bibliografía utilizada:

Especificaciones pantalla TFT 2.8: https://www.amazon.com/-/es/HiLetgo-ILI9341-Pantalla-t%C3%A1ctil-STM32/dp/B073R7BH1B

Especificaciones Arduino mega: https://proyectoarduino.com/arduino-mega-2560/

Especificaciones del giroscopio MPU 6050: https://naylampmechatronics.com/sensores-posicion-inerciales-gps/33-modulo-mpu6050-acelerometro-giroscopio-i2c.html

Especificaciones DFPlayer mini: https://hetpro-store.com/dfplayer-mini-reproductor-de-mp3/

Características de parlante de 4 OHMS: https://articulo.mercadolibre.com.ar/MLA-861961892-parlante-2-pulgadas-3w-4-ohms-ciclos-_JM#position=1&type=item&tracking_id=f83be169-1987-4feb-ae4a-8491c821ca60

MicroSD: https://es.wikipedia.org/wiki/Memoria_flash y https://es.wikipedia.org/wiki/MicroSD

Visual Studio Code: https://code.visualstudio.com/

Bibliotecas para C++:

WinBGim: https://winbgim.codecutter.org/

Arduino: https://www.arduino.cc/

Bibliotecas para Arduino:

DF Player Mini: https://github.com/DFRobot/DFPlayer-Mini-mp3

MPU6050: https://www.arduinolibraries.info/libraries/mpu6050

Time: https://www.arduinolibraries.info/libraries/time

FL Studio: https://www.image-line.com/

Adobe Photoshop: https://www.adobe.com/la/products/photoshop.html

Proteus: https://www.labcenter.com/

Tinkercad 3D: https://www.tinkercad.com/

Manual de usuario: basado en la guía de usuario de la serie de videoconsolas desarrolladas por Nintendo: GameBoy Micro:

https://www.nintendo.com/consumer/downloads/micro_spanish.pdf

Tomando solo los aspectos relevantes como la sección de:

- Información de Salud y Seguridad 66-69
- Componentes 70-71
- Cargando el Arreglo de Batería del GameBoy micro 76-78
- Uso del Sistema GameBoy micro 79

El manual de instrucciones cuenta con los lenguajes español-inglés para respaldar un alcance mayor.

Agradecimientos:

- Pablo Pereyra: Impresión 3D de la carcasa del prototipo y prestación de dispositivos fundamentales para el armado del prototipo.
- Gastón Fontela: Profesor de programación.
- Javier Monje: Prestación de dispositivos fundamentales para el armado del prototipo.
- Silvia Lingiardi: Profesora. Enseñanza del Diagrama de Gantt
- Pablo Mileti: Ayuda con temas fundamentales de Arduino.
- Fernando Galarza: Prestación de elementos fundamentales.