UTN Facultad Regional Mendoza
Informática II

Trabajo Final

Proyecto final: Space Invaders

Profesor Titular: Ing. Marcelo Ledda
Profesora JTP: Ing. Diedrichs, Ana Laura

Nombres: Nieves Brandon
Legajo: 47313
Fecha: 21/11/2023
Curso: 2R5

INDICE

1. Introducción

El proyecto consiste en una simplificación del juego “Space invaders”, utilizando dos softwares de programación (processing y arduino) comunicados por un puerto serial al hardware correspondiente.

El código de Processing se encarga del entorno gráfico del programa mostrado por pantalla, esto consiste en un artillero que se desplaza por pantalla a voluntad del usuario, un conjunto de atacantes con movimiento arbitrario, como la interacción entre el artillero y los atacantes. Arduino toma el papel de comunicar a processing con el usuario por medio del hardware arduino uno y un cable USB.

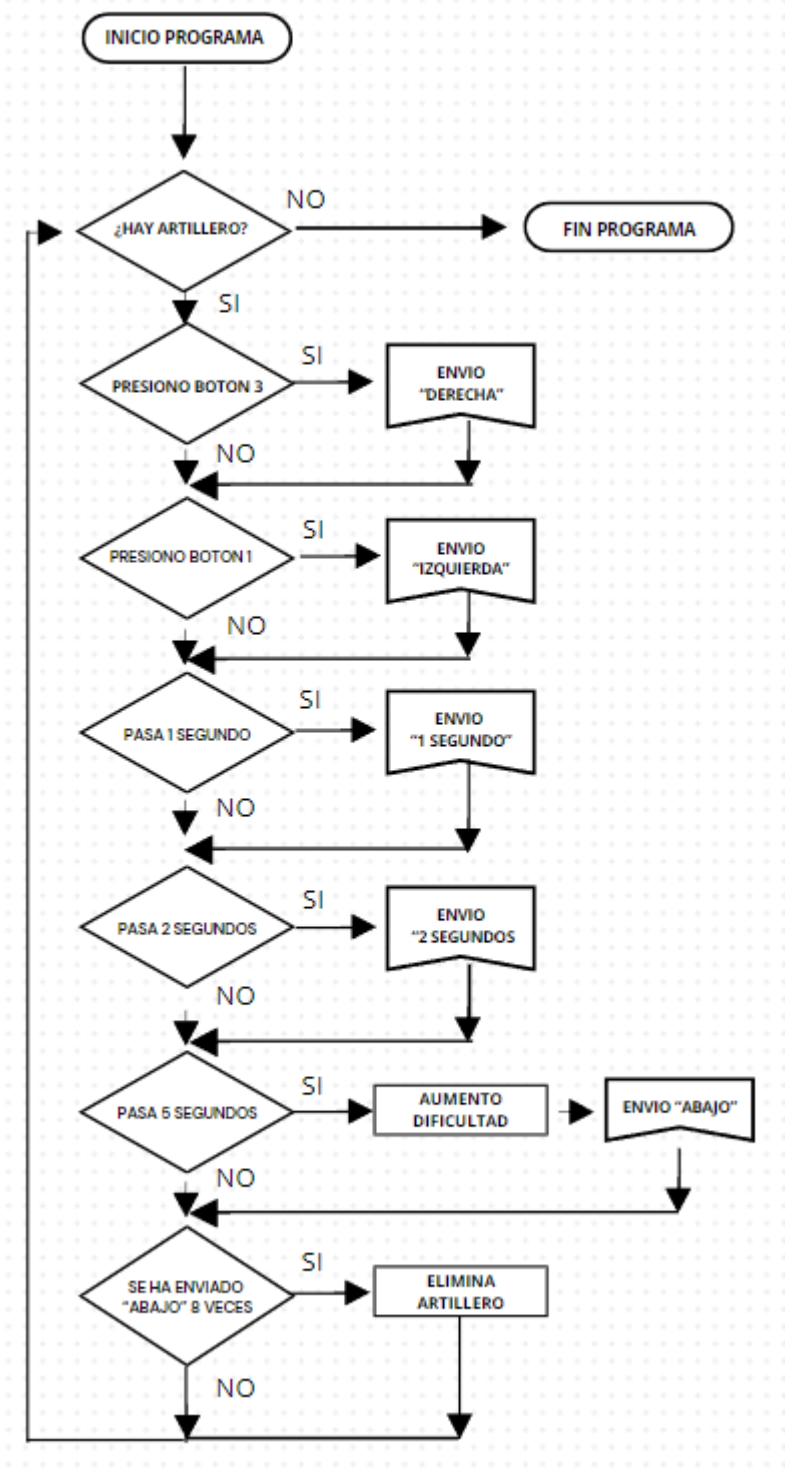
Se utiliza un shield externo Multi-Function junto a su librería, en donde se obtuvo esta última se detalla en la sección 5.3-Referencias y las especificaciones del shield se detallan en la sección 5.4-Otros anexos.

El juego consiste en eliminar a todos los atacantes por medio de los proyectiles del artillero mientras esquivas los proyectiles enemigos, perderás si tocas un proyectil o si dejas que los atacantes bajen lo suficiente como para tocarte. Los atacantes se moverán más rápido conforme pase el tiempo y soltarán menos proyectiles mientras menos atacantes queden.

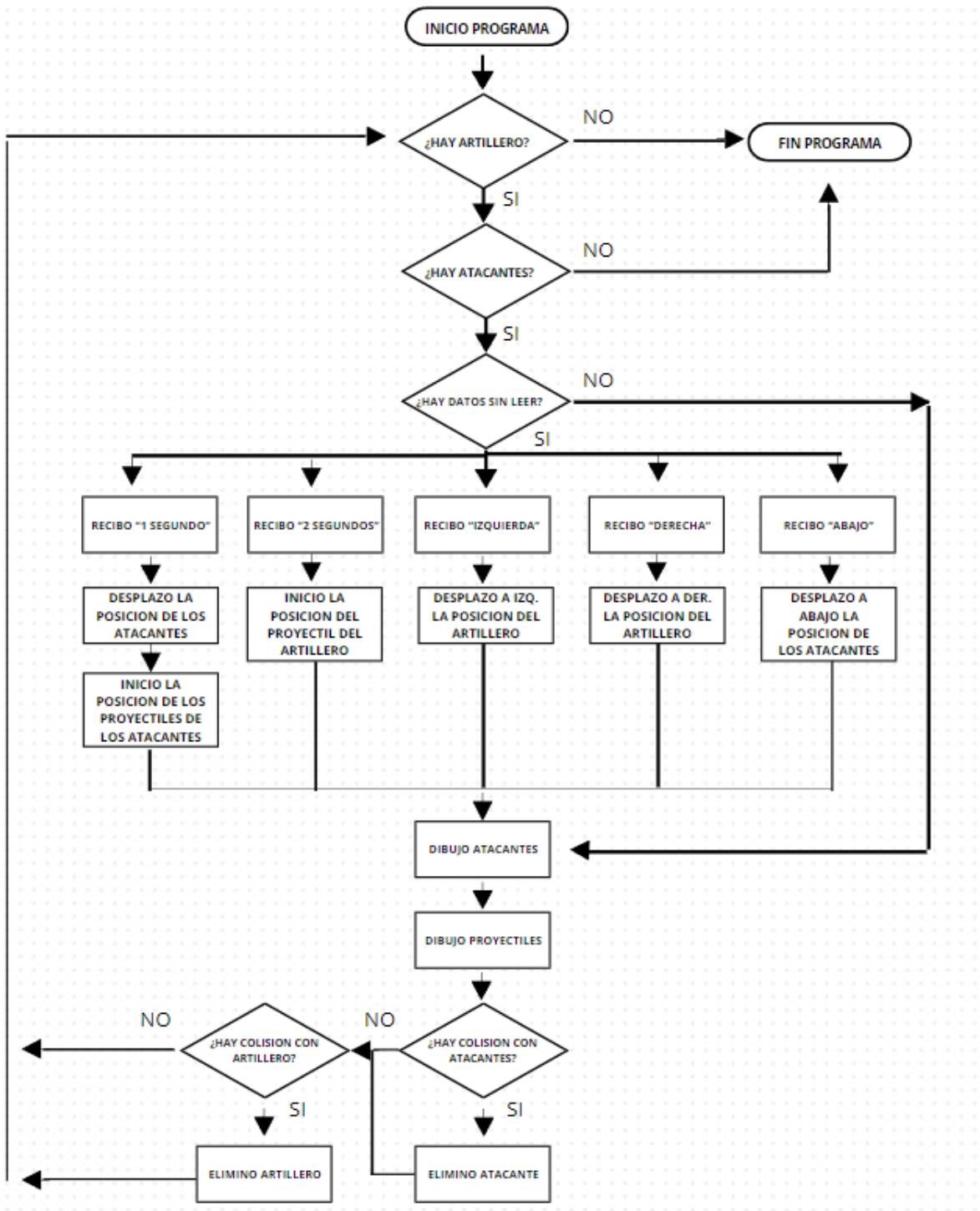
El programa está definido a una velocidad de comunicación de 9600, los proyectiles arbitrarios lanzados por el artillero se limitan a uno cada 2 segundos mientras que los proyectiles de los atacantes se limitan a 3 en pantalla a la vez. La cantidad de intentos para terminar el juego es solamente una y la única interacción del usuario con el programa es la provenientes de los botones del shield de arduino.

El código de arduino como el código de processing usa la clase Game para declarar variables y métodos utilizados para el funcionamiento del programa. Además se usa en processing funciones de entorno gráfico para el dibujo del artillero, atacantes y proyectiles mostrados por pantalla. En arduino se utilizan variables globales con el fin de reducir dificultad en código, además se usan librerías propias del shield utilizado. Las variables utilizadas en ambos códigos son booleanas, de tipo entero, de tipo byte y arrays.

2. Diagrama funcional



Descripción: Diagrama del código arduino.



Descripción: Diagrama del código processing.

3. Código fuente de los Programas

Código fuente de arduino:

SECCIÓN 1.1: Declaración de variables

Descripción: Declaración de variables de la clase como su inicialización.

SECCIÓN 1.2:Métodos de la clase

1.2.1:
int MoveAlienShips(int , int){...}

Descripción: Cada 1 segundo envía un mensaje distinto por puerto serial que indica la nueva posición del conjunto atacante, sea hacia la izq, der o abajo y, si es hacia abajo también aumenta la dificultad del juego al llamar a la función "IncressDifficulty()". Parámetros: Tiempo transcurrido y dificultad. Retorna:Tiempo actualizado.

1.2.2:

int GunnerShots(int , int){...}

Descripción: Al pasar 2 segundos envía un mensaje por puerto serie indicando a processing que dibuje un proyectil del artillero. Parámetros: Tiempo transcurrido y dificultad. Retorno: Tiempo actualizado.

1.2.3:
int ShipsShot(int , int){...}

Descripción: Al pasar 1 segundo envía un mensaje por puerto serie que da inicio a processing para que dibuje un proyectil del atacante. Parámetros: Tiempo transcurrido y dificultad. Retorno: Tiempo actualizado.

1.2.4:
void IncressDifficulty(){...}

Descripción: Aumenta la dificultad por medio de reducir el tiempo de comparación. Parámetros: Ninguno. Retorno: Ninguno.

1.2.5:
void GameOver(int){...}

Descripción:Termina el juego tras enviar 10 veces "abajo" y elige la música “loser” tras llamar a la función sound. Parámetros: Ninguno. Retorno: Ninguno.

1.2.6:
int DoBotton(int){...}

Descripción:Lee los botones del shield cuando son presionados, luego envía información por puerto o pausa el juego en función del botón presionado. Parámetros: El valor de pausa del programa. Retorna:El valor de pausa del programa.
1.2.7:
void Sound(char){...}

Descripción: Hace sonar el beeper del shield en función de la opción recibida por puerto serie. Las opciones disponibles son:
1. Suena el beeper cuando se presiona el botón que indica derecha del shield.
2. Suena el beeper cuando se presiona el botón que indica izquierda del shield.
3. Suena el beeper cuando se recibe la información por puerto serie que indica que se ganó el juego.
4.Suena el beeper cuando se recibe la información por puerto serie que indica que se perdió el juego.

1.2.8:
int onPause1(int){...}

int onPause2(int){...}

int onPause3(int){...}

Descripción: El conjunto de funciones se encargan de mantener las distancias del tiempo de arduino y uno artificial cuando el programa está en pausa.

SECCIÓN 3: Función void setup() {...}

Descripción: Inicializa el tiempo transcurrido del shield, los pulsadores del shield y la velocidad de comunicación del puerto serie.

SECCIÓN 4: Función loop() {...}

Descripción: Función en donde se ejecutan en forma repetitiva todas las funciones antes descritas. También es aquí donde se ejecuta el programa en modo en pausa o no.

Código fuente processing**:**

SECCIÓN 1.1: Declaración e inicialización de variables.

Descripción: Declaro e inicializo variables de la clase y variables globales que se utilizarán a posterior.

SECCIÓN 1.2: Métodos de la clase.
1.2.1:
void PositionGunner(int, int){...}

Descripción:Dibuja el artillero.

1.2.2:
void MoveGunnerRight(){...}

void MoveGunnerLeft(){...}

Descripción: Conjunto de funciones que desplazan la posición del artillero en función del dato recibido por puerto serie.

1.2.3: int Ships(int, int, int, int, int, int){...}

Descripción: Dibuja a cada atacante siempre que el estado del mismo sea verdadero y cambia su posición. En la misma función se llama a la función collision con el fin de modificar el estado del atacante. Parámetros: Posición (X,Y) de proyectil, posición (X,Y) de atacantes y su estado de desplazamiento a derecha o izquierda. Retorna: Nueva posición X de atacantes.

1.2.4:
int GunnerShot(int, int, boolean){...}

Descripción: Dibuja el proyectil del artillero y desplaza su posición hacia arriba siempre que su estado sea verdadero, de lo contrario no se hace

nada. Parámetros: Posición (X,Y) del proyectil del artillero y su estado. Retorna: Nueva posición Y del proyectil.

1.2.5:
int ShipsShot(int, int, int, int, int, int){...}

Descripción: Dibuja el proyectil de un atacante al azar siempre que el estado del atacante sea verdadero y desplaza la posiciónY del proyectil. Parámetros: Posición (X,Y) del proyectil, posición (X,Y) del artillero, un número aleatorio y un entero. Retorna: Nueva posición Y del proyectil.

1.2.6:
void Collision(int, int, int, int, int , int , int){...}

Descripción: Cambia de estado de verdadero a falso del artillero o del atacante si alguno fue golpeado por un proyectil. Parámetros: Posición (X,Y) del proyectil, posición (X,Y) del atacante y un estado booleano. Retorna: Nada.

1.2.7:
void WinGame(){...}

Descripción: Cuenta el número de atacantes derribados, sí la cuenta es igual a 21 se gana el juego. Parámetros: Estado booleano. Retorna: Estado booleano.

SECCIÓN 3: Función setup() {...}

Descripción: Inicializa la fuente usada en los textos mostrados por pantalla, la clase de tipo Game, las imágenes utilizadas, el modo de dibujo de los rectángulos y textos y la velocidad de comunicación del puerto serie.

SECCIÓN 4: Función Draw() {...}

Descripción: Función que se ejecuta permanentemente, ésta ejecuta todas las funciones antes descritas. Está dividida en 5 partes:{0,1,2,3,4}.
0: Muestra por pantalla el texto “PAUSE” y pausa el juego.
1: Corre el juego en su normalidad.
2: Desarrollo futuro.
3: Muestra por pantalla el texto “WIN” y termina el juego.

4: Muestra por pantalla el texto “DEFEAT” y termina el juego.

SECCIÓN 5: Función serialEvent()

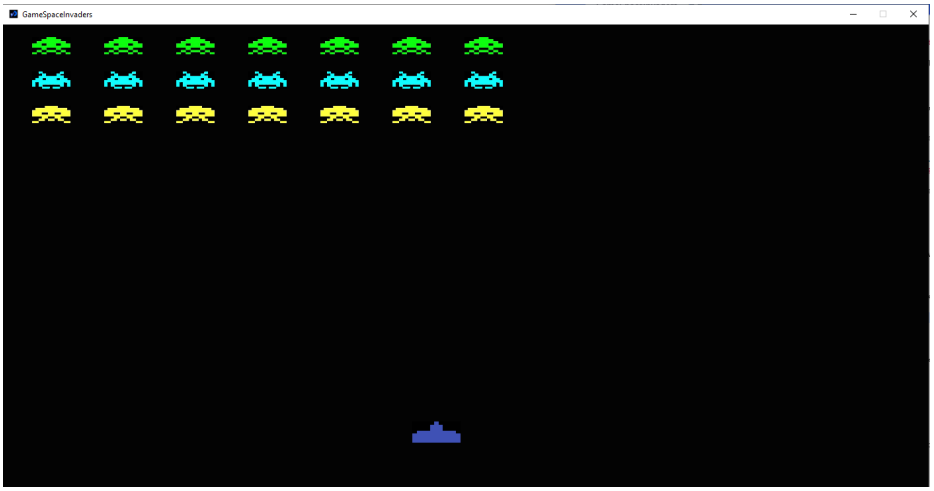
void serialEvent (Serial MiPuerto){...}

Descripción: Esta función ejecutada constantemente, cumple el papel de leer el buffer del puerto serie y ejecutar funciones en base a la información allí leída. Se encuentra divida en 9 partes:{r,l,s,z,c,R,L,P,G}
r: Desplaza al conjunto atacante a derecha y elige al azar una de dos imágenes disponibles para el conjunto atacante.
l: Desplaza al conjunto atacante a izquierda y elige al azar una de dos imágenes disponibles para el conjunto atacante.
s: Modifica la posiciónY del conjunto atacante.
z: Guarda las posiciones (X,Y) del artillero para utilizarlas al dibujar su proyectil.
c: Guarda las posiciones (X,Y) del atacante para utilizarlas al dibujar su proyectil, con un máximo de 3 proyectiles.
R: Desplaza el artillero a derecha ejecutando la función “MoveGunnerRight()”.
L: Desplaza el artillero a izquierda ejecutando la función “MoveGunnerLeft()”.
P: Pausa el juego tras pasar a la sección 0 de la función Draw().
G: Termina el juego tras pasar a la sección 4 de la función Draw().

(Juego desactualizado) Enlace de video en youtube (breve funcionamiento del juego):
https://youtu.be/pnj7x0A_ABU

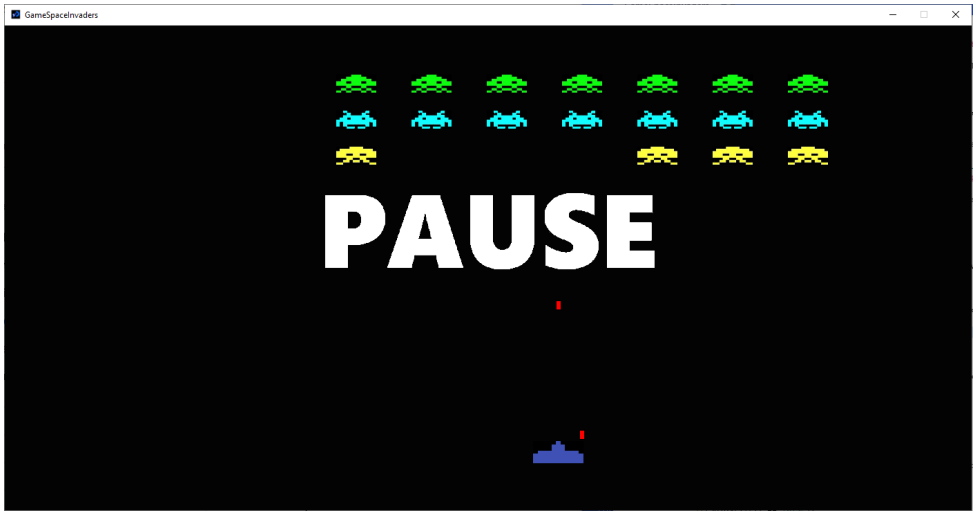
Enlace código arduino:
https://github.com/Nieves433/Proyecto-Final-Inform-tica-II/blob/main/Código/Arduino/Arduino%20Code.ino
Enlace código processing:
https://github.com/Nieves433/Proyecto-Final-Inform-tica-II/blob/main/Código/Processing/Processing%20Code.pde

4. Salidas del Programa



Descripción: Pantalla mostrada al iniciar el juego.

Descripción: Pantalla mostrada al ganar el juego tras derribar a todos los atacantes.

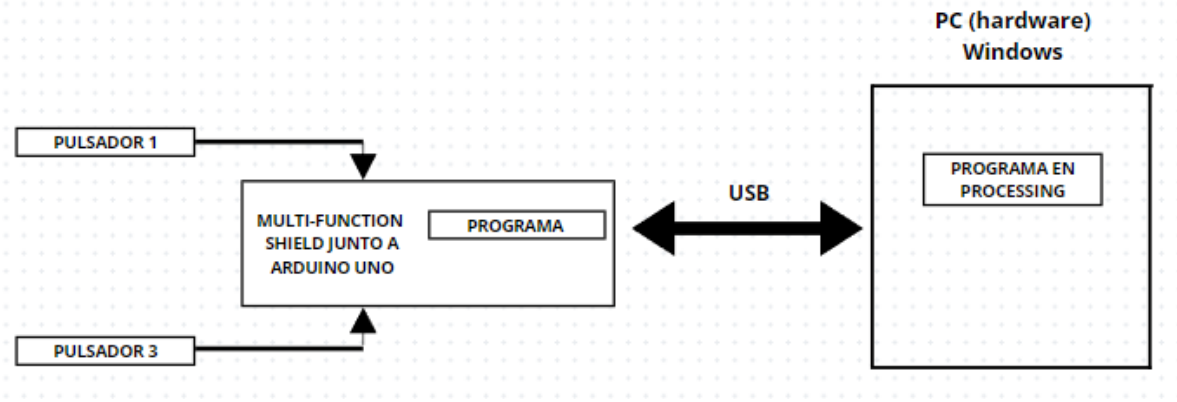


Descripción: Pantalla mostrada al presionar botón del medio del shield.

Descripción: Pantalla mostrada al perder por tocar un proyectil del atacante o por tocar a algún atacante.

5. Anexos

5. 1 Esquemático



5.2 Listado de componentes

-Multi-function shield compatible con arduino uno
-2 interruptores tipo pulsadores color rojo sin luz con 19 mm de diámetro (adheridos al shield)
-Un Arduino tipo Arduino uno
-Un cable USB tipo A USB tipo B de 50 cm
-Software: Arduino V1.8.15
-Software: Processing V4.0b1
-Software: Windows 10

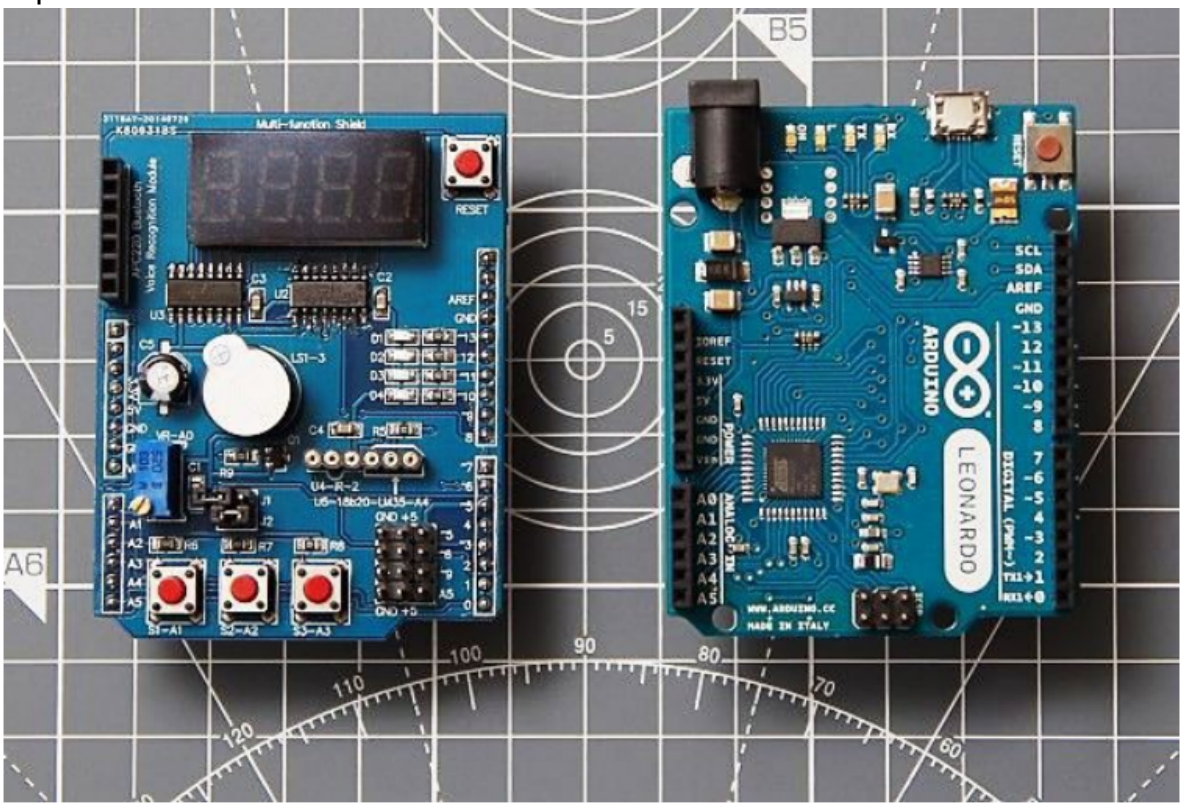
5.3 Referencias

-Librería usada: MultiFuncShield-Library
-Video de referencia utilizado en imágenes: https://www.youtube.com/watch?v=7BoJBYh16CQ&list=PLRqwX-V7Uu6YB9x6f23CBftiyx0u_5sO9&index=2
-Referencias de arduino:https://www.arduino.cc/reference
-Referencias de processing:https://processing.org/reference
-Cómo instalar librerías en arduino:Installing Libraries | Arduino Documentation
-Archivo de librería:https://github.com/DireCat/MFShield
-Lugar en donde se obtuvieron las imágenes:

https://spaceinvaders.fandom.com/wiki/UFO
-PDF de referencias para funciones del shield:
https://www.mpja.com/download/hackatronics-arduino-multi-function-shield.pdf
-Especificaciones del fabricante para el shield:
https://taillieu.info/index.php/articles/category-blog/424-arduino-r3-multifunctional-expansion-board-shield

5.4 Otros anexos que deba incluir

Shield multi-function a izquierda y arduino uno a derecha, únicos soportes de hardware utilizados.



Especificaciones del shield:

- An LED display module (four-digit, seven-segment)
- Two serial LED display driver ICs(74HC595)
- Four LEDs
- A multi-turn trimpot (10KOhm)
- Three pushbuttons
- A piezo-buzzer

A port for DS18B20 temperature sensor
A port for TSOP1838 infrared receiver module
A APC220 wireless module interface (serial/UART interface)
A reset button

Diseño de pines del shield:

Serial LED display driver (74HC595) - Latch 4, Clock 7, Data 8
Four LEDs - Pin 10,11,12,13
Trimpot (10KOhm)-Pin A0
Three pushbuttons - Pin A1,A2,A3
Piezo-buzzer - Pin 3 (digital on/off)
DS18B20/LM35 temperature sensor - Pin A4
Infrared receiver module (remote control) - Pin 2
Header for APC220 shield - Gnd, 5V , 0, 1 (rx/tx)
Reset pushbutton - Gnd
Free pins header (PWM) - 5, 6, 9, A5

Circuit diagram del shield: