

Trabajo de Laboratorio Final

ARQUITECTURA DE SISTEMAS DE ELABORACIÓN DE DATOS II

Alumnos:

- Bacich, Sofía
 - López, Agustín Iván
 - Morán, Iván
-

Profesor:

- Loiseau, Matías
-

Índice

Objetivo	2
Introducción	3
Marco Teórico	4
Arduino	4
Arduino UNO	4
Arduino Mega 2560	5
Display LCD 2.4" TFT 240x320 v2.1	8
UART	9
ITEAD Joystick Shield	10
Botonera	11
WeMOS D1 E1 Mini - ESP8266	13
Desarrollo	15
Conclusión	20
Mejoras a Desarrollar	21
Apéndice 1: Códigos	22
Apéndice 2: Videos	42
Bibliografía	43

Objetivo

El objetivo del presente trabajo de laboratorio es diseñar una versión propia del juego PONG, mostrándolo mediante un display LCD y controlandolo con un joystick y una botonera. También nos proponemos mostrar los puntajes de cada partido en una página web que hosteamos mediante un módulo WeMOS D1 E1 Mini - ESP8266.

Introducción

Inspirados en el Arcade, se plantea realizar una versión propia del juego PONG que permita jugar tanto en modo de un sólo jugador (single player) contra la máquina como en modo de dos jugadores (two player mode) donde ambos se enfrenten entre sí.

Para eso se requerirá usar un microcontrolador para diseñar la lógica del juego y el control del mismo. En nuestro caso, elegimos usar un Arduino Mega 2560 como nuestro microcontrolador principal; un display LCD 2.4" TFT de 240x320 pixeles para poder mostrar el juego; y un joystick shield montado sobre un Arduino UNO para poder controlar el juego mediante un joystick y una botonera.

También se mostrarán los puntajes en una página web que hosteamos mediante un módulo WeMOS D1 E1 Mini - ESP8266 y que se comunicará con el Arduino Mega 2560 mediante comunicación serial.

Marco Teórico

A continuación vamos a detallar los conceptos teóricos detrás de los componentes que utilizaremos en la etapa de desarrollo del trabajo de laboratorio final.

Arduino

Arduino es una plataforma de creación electrónica de código abierto basada en hardware y software fáciles de utilizar. De esta manera ofrece las bases para que cualquier persona o empresa pueda crear sus propias placas, o que puedan programar las que arduino ya ofrece.

Arduino ofrece un entorno de desarrollo Arduino IDE, donde se puede crear programas para las placas de Arduino y donde se ofrecen varias utilidades, entre ellas la posibilidad de buscar y descargar librerías, la posibilidad de compilar el código, el control de los puertos de la computadora para subir el código por medio de conexión USB a las placas, etc.

También posee varios modelos de placas, entre los cuales se encuentran:

Arduino UNO

Arduino Uno es una placa de microcontrolador basada en ATmega328P, la cual puede ser vista en la Figura 1. Posee 14 pines de entrada/salida digital (6 de los cuales pueden usarse como salidas de PWM), 6 pines de entrada analógica, conexión USB, etc. Su diagrama completo se puede ver en la Figura 2.

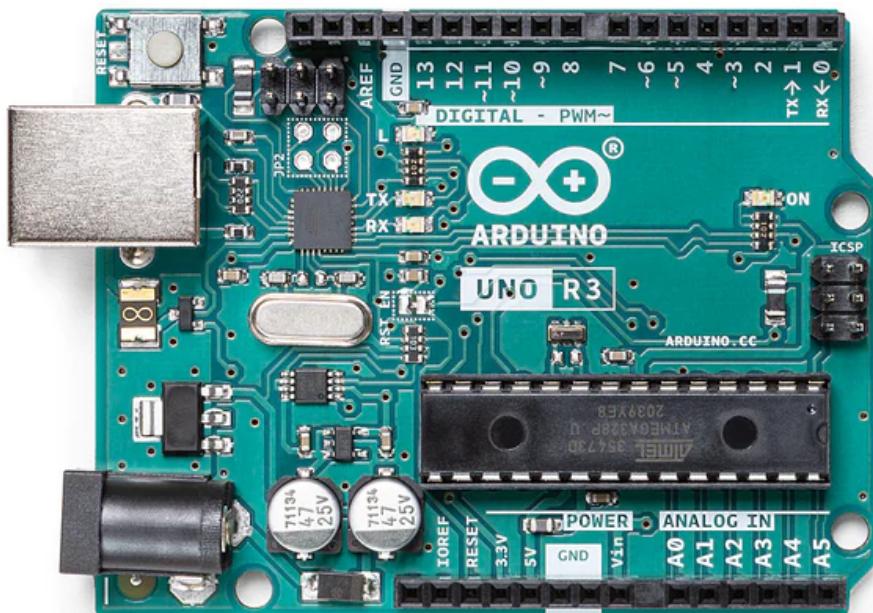


Figura 1

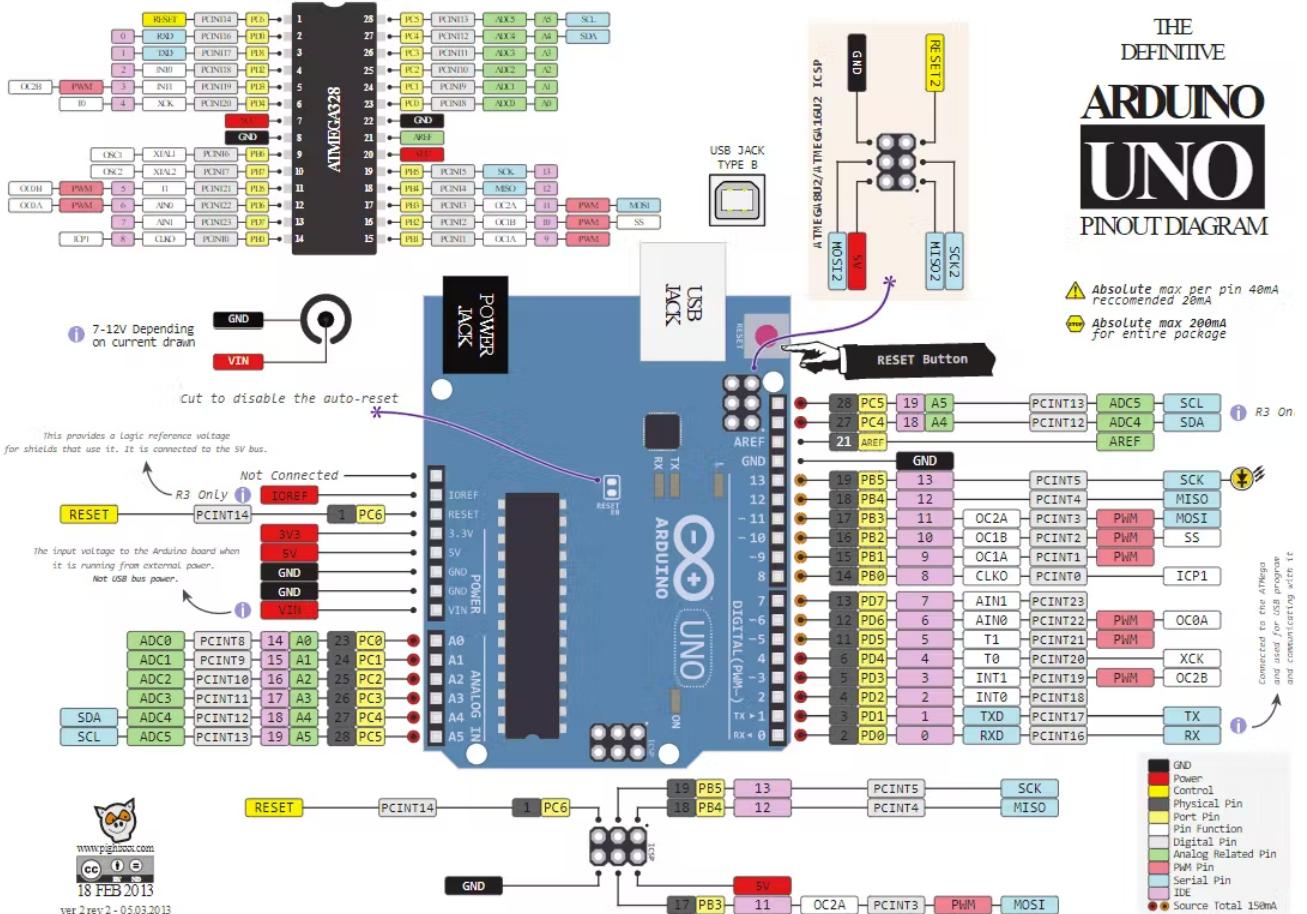


Figura 2

Arduino Mega 2560

Arduino Mega 2560 también es una placa de microcontrolador basada en ATmega328P, la cual puede ser observada en la Figura 3. Pero, a diferencia del Arduino UNO, posee muchos más pines y prestaciones: 54 pines de entrada/salida digital (15 de los cuales pueden usarse como salidas de PWM), 16 pines de entrada analógica, 4 UARTs, conexión USB, etc. Su diagrama completo se puede ver en las Figuras 4 y 5.

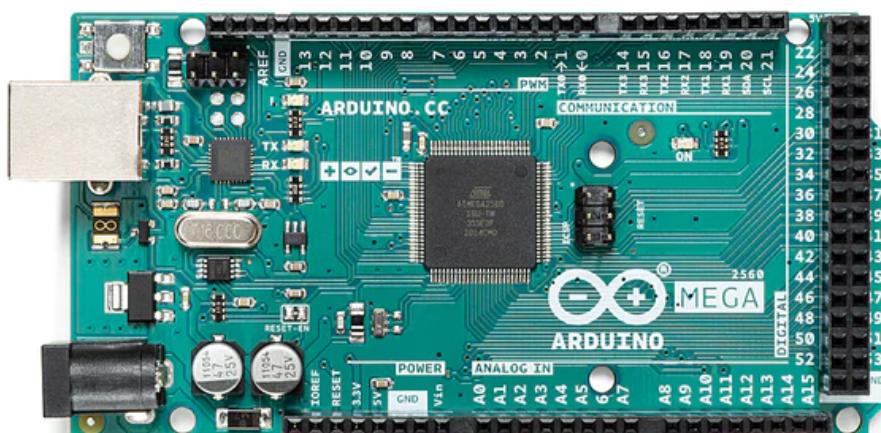


Figura 3

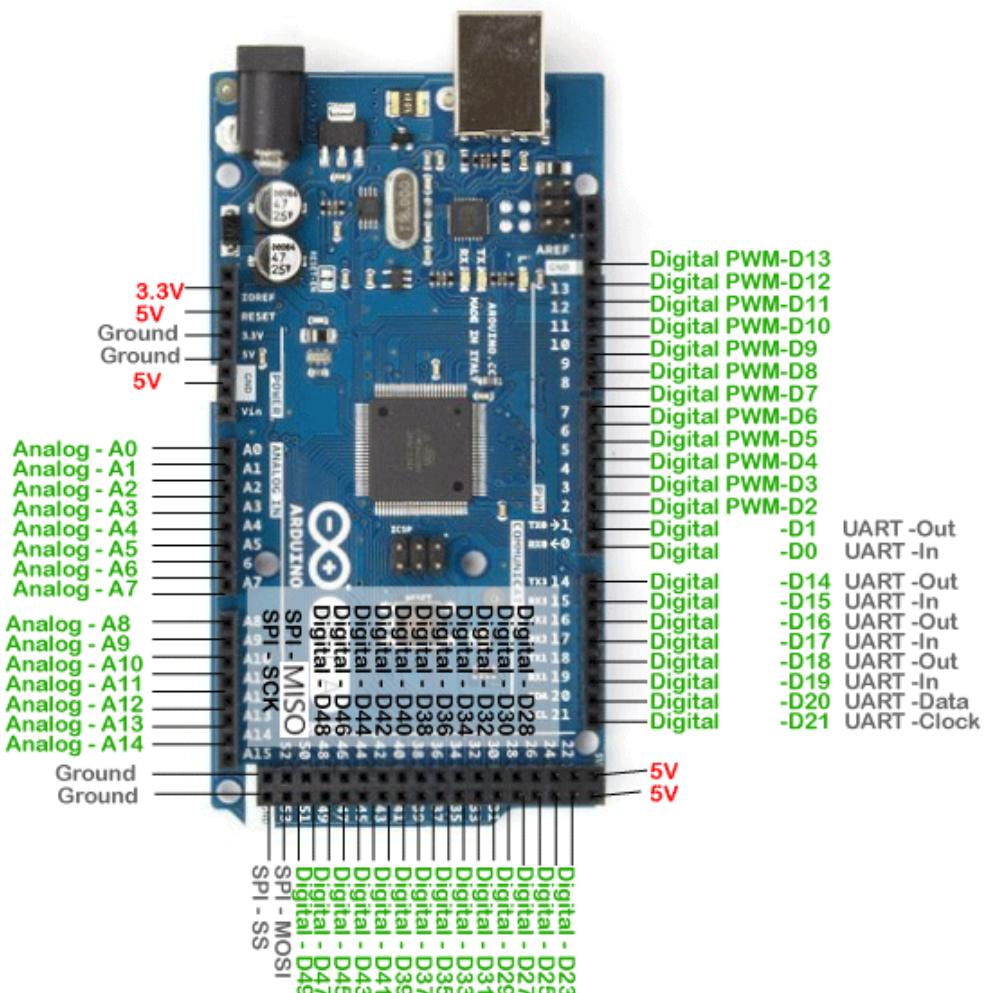
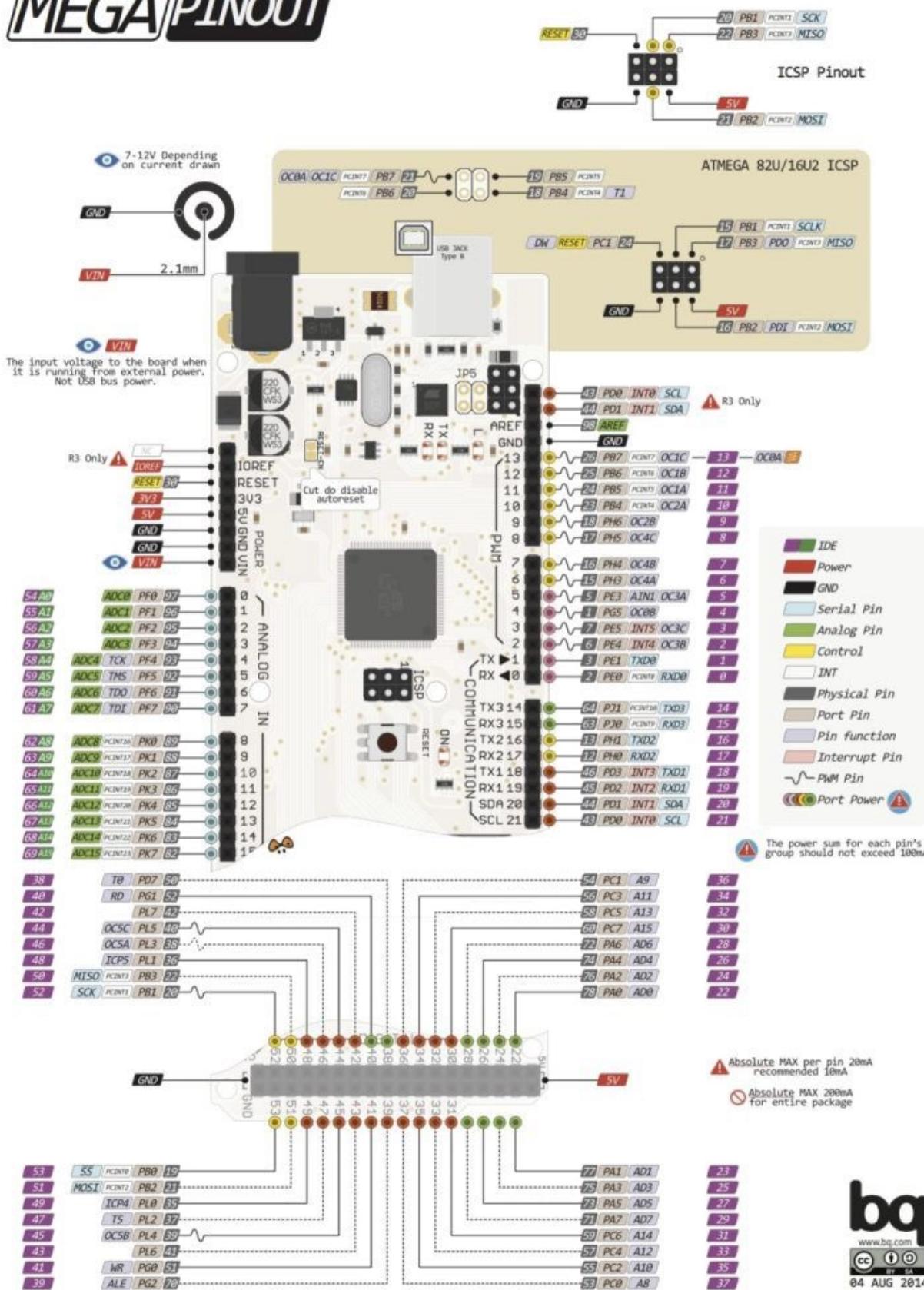


Figura 4

MEGA PINOUT



Display LCD 2.4" TFT 240x320 v2.1

Las pantallas o displays LCD permiten mostrar texto, gráficos e imágenes en los proyectos con microcontroladores como Arduino, Pic o Raspberry Pi. Pueden ser utilizadas para mostrar la lectura de sensores, implementar una interfaz hombre máquina (HMI) y ayudarnos a realizar depuración de código (debugging) en nuestros proyectos. La forma en que funcionan los LCD es utilizando una retroiluminación o luz de fondo (backlight) que proporciona una fuente de luz a los píxeles individuales dispuestos en una cuadrícula rectangular. Cada píxel tiene un subpíxel RGB (rojo, verde y azul) que se puede activar o desactivar. Cuando todos los subpíxeles de un píxel están apagados, el mismo aparece en negro. Cuando todos los subpíxeles están encendidos al 100 %, aparece en blanco. Al ajustar los niveles individuales de luz roja, verde y azul, se obtienen millones de combinaciones de colores.

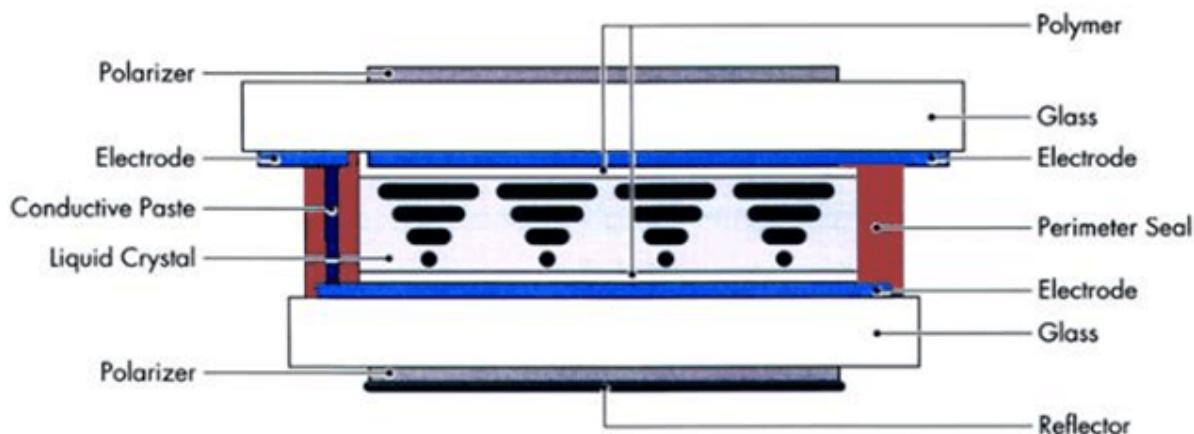


Figura 6

En la Figura 6 se puede ver la estructura de un LCD. La misma incluye una capa delgada de material de cristal líquido intercalada entre dos electrodos sobre sustratos de vidrio, con dos polarizadores en cada lado. Un polarizador es un filtro óptico que deja pasar las ondas de luz de una polarización específica mientras bloquea las ondas de luz de otras polarizaciones. Los electrodos deben ser transparentes, por lo que el material más popular es el ITO (óxido de indio y estaño). Como la pantalla LCD no puede emitir luz por sí misma, normalmente se coloca una retroiluminación o luz de fondo (backlight) detrás de una pantalla LCD para poder verla en un entorno oscuro. Las fuentes de luz para la retroiluminación pueden ser LED (diodo emisor de luz) o CCFL (lámparas fluorescentes de catodo frío). La retroiluminación LED es la más popular. Por supuesto, si se desea tener una pantalla a color, se puede convertir una capa de filtro de color (RGB) en una celda LCD. También se puede agregar un panel táctil frente a una pantalla LCD para hacer que la pantalla sea táctil.

Existen distintos tipos de LCD, en nuestro caso el utilizado es un TFT, es decir que utiliza transistores de película delgada TFT (thin-film transistors). Estos transistores son pequeños transistores de commutación, así como condensadores que se colocan dentro de una matriz sobre un sustrato de vidrio. Cuando se activa la fila adecuada, se puede transmitir una carga por la columna exacta para que se pueda direccionar un píxel específico. Porque

todas las filas adicionales con las que se cruza la columna se apagan, simplemente el capacitor al lado del píxel designado recibe una carga. Esto permite que el display tenga colores vivos y un rápido refresco.

En nuestro caso, el display LCD TFT 2.4" v2.1 que utilizamos es una pantalla a colores táctil con una resolución de 240x320 píxeles, controlador gráfico ILI9325, controlador táctil XPT2046, comunicación SPI y que puede mostrar hasta 262144 colores RGB distintos. El display incluye además un socket para memorias SD, útil para almacenar imágenes en formato mapa de bits (bmp). El adaptador de tarjetas se conecta por un puerto SPI al Arduino. El display y la memoria SD pueden compartir el bus de comunicación SPI (SCK, MISO, MOSI) pero deben utilizar pines de CS (chip select) separados. El controlador gráfico además posee un buffer RAM, disminuyendo los requerimientos del microcontrolador a usar. En la Figura 7 se puede ver un ejemplo de este display LCD TFT.

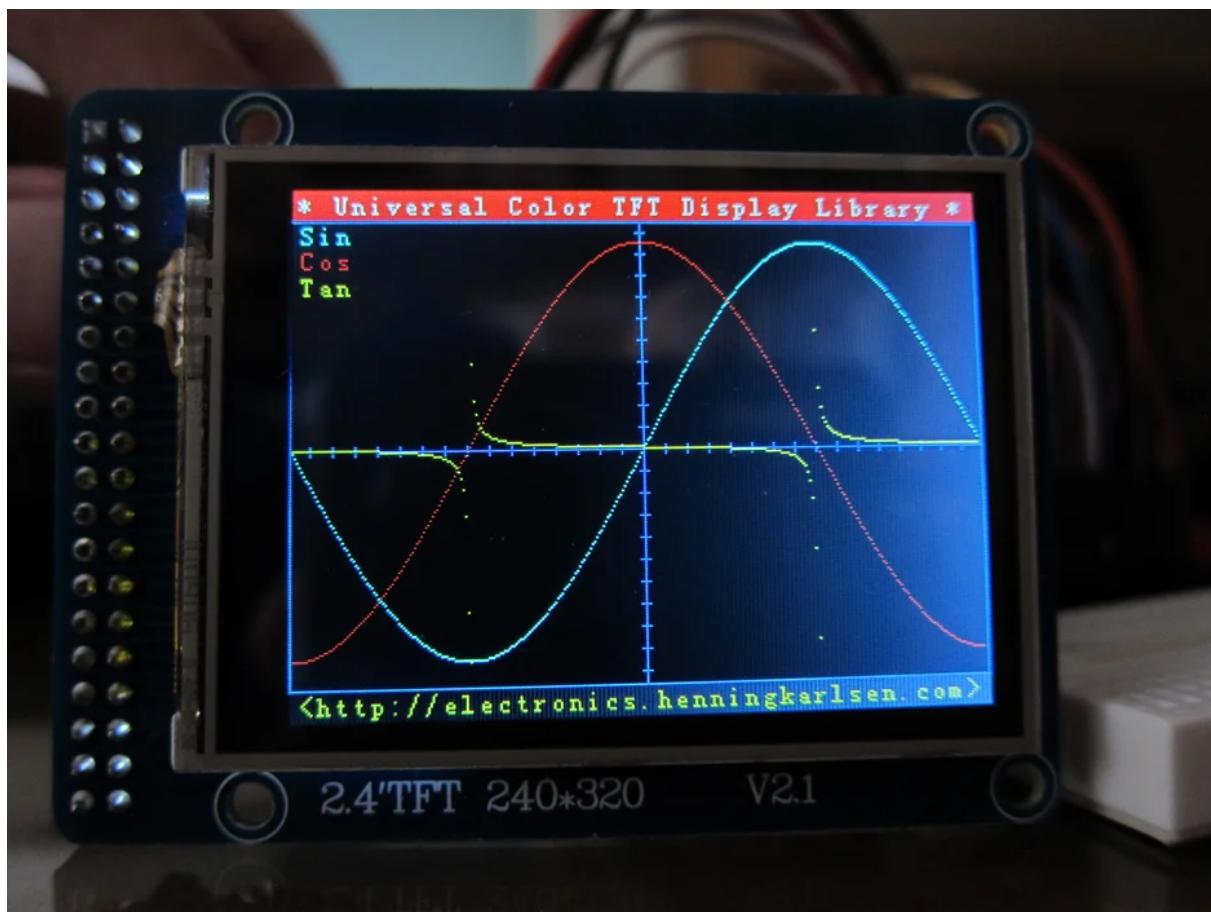


Figura 7

UART

UART (universal asynchronous receiver / transmitter, por sus siglas en inglés), define un protocolo o un conjunto de normas para el intercambio de datos en serie entre dos dispositivos. UART es sumamente simple, como se puede ver en la Figura 8, y utiliza solo dos hilos entre el transmisor y el receptor para transmitir y recibir en ambas direcciones. Ambos extremos tienen una conexión a masa. La comunicación en UART puede ser simplex (los datos se envían en una sola dirección), semidúplex (cada extremo se comunica, pero solo uno al mismo tiempo), o dúplex completo (ambos extremos pueden transmitir simultáneamente). En UART, los datos se transmiten en forma de tramas.

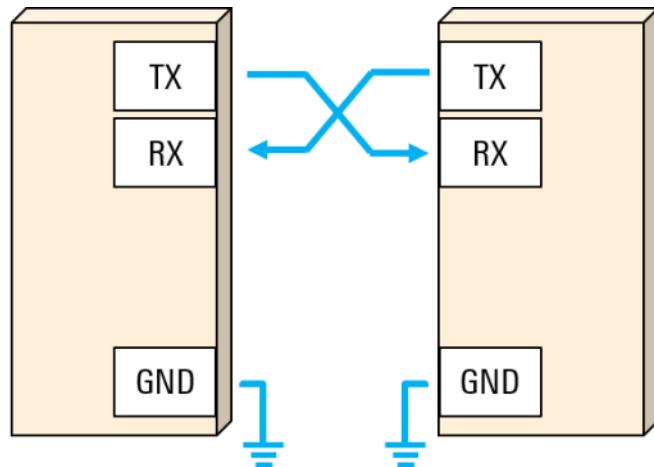


Figura 8

UART fue uno de los primeros protocolos en serie. Los puertos en serie, que en su día proliferaron a gran escala, se basan casi siempre en el protocolo UART, y los dispositivos que utilizan interfaces RS-232, módems externos, etc. son ejemplos típicos de la aplicación de UART. En los últimos años la popularidad de UART ha disminuido, y se ha ido sustituyendo por protocolos como SPI e I2C para la comunicación entre chips y componentes. En lugar de comunicarse por un puerto en serie, la mayoría de los ordenadores y periféricos modernos utilizan hoy tecnologías como Ethernet y USB. Sin embargo, UART se sigue utilizando para aplicaciones de baja velocidad y bajo rendimiento, puesto que es muy simple, económico y fácil de integrar.

ITEAD Joystick Shield

El Joystick Shield es un kit que contiene todas las partes necesarias para permitir usar un controlador similar a los de Nintendo a través de Arduino. El mismo se puede visualizar en la Figura 9. El shield contiene cinco botones push (los botones A, B, G y F, más el botón C asociado a presionar el joystick), dos botones adicionales de control (los botones D y E) y un joystick para pulgar de doble eje (permitiendo movimiento horizontal y vertical).



Figura 9

El funcionamiento de los botones es similar a los de cualquier botón pull-up, tal como describiremos en la brevedad, y están conectados a los pines digitales D3 al D9; mientras tanto, el joystick controla el movimiento vertical y horizontal a través de voltajes analógicos, mapeados a los pines analógicos A0 y A1. El joystick shield se puede montar directamente encima de un Arduino UNO, y además provee pines de salida para cada uno de los botones y para los movimientos analógicos del joystick. En la Figura 10 se puede ver la descripción de estos pines.

Pin	Output type	Description
D0	NA	No Connection
D1	NA	No Connection
D2	NA	No Connection
D3	Digital	Button E Output
D4	Digital	Button D Output
D5	Digital	Button C Output
D6	Digital	Button B Output
D7	Digital	Button A Output
D8	Digital	Button F Output
D9	Digital	Button G Output
D10	NA	No Connection
D11	NA	No Connection
D12	NA	No Connection
D13	NA	No Connection
A0	Analog	Joystick Y Ouput
A1	Analog	Joystick X Ouput
A2	NA	No Connection
A3	NA	No Connection
A4	NA	No Connection
A5	NA	No Connection

Figura 10

Botonera

Un botón o pulsador es un dispositivo utilizado para realizar cierta función. Los botones son de diversas formas y tamaños y se encuentran en todo tipo de dispositivos, aunque principalmente en aparatos eléctricos y electrónicos. Los botones son por lo general activados, al ser pulsados con un dedo. Permiten el flujo de corriente mientras son accionados. Cuando ya no se presiona sobre él vuelve a su posición de reposo. Puede ser un contacto normalmente abierto en reposo NA (Normalmente abierto), o con un contacto NC (normalmente cerrado) en reposo. Existen dos formas de conectar un pulsador, como se puede ver en la Figura 11, y a pesar de que la función principal es la misma, a la hora de programar es necesario tener en cuenta dicha conexión. Estas dos formas son: Pull Up (En este caso cuando se presiona el pulsador, el microcontrolador "ve" o lee un cero en ese pin)

y Pull Down ((En este caso cuando se presiona el pulsador, el microcontrolador "ve" o lee un uno en ese pin). En general, la forma más usada en dispositivos electrónicos es Pull Up.

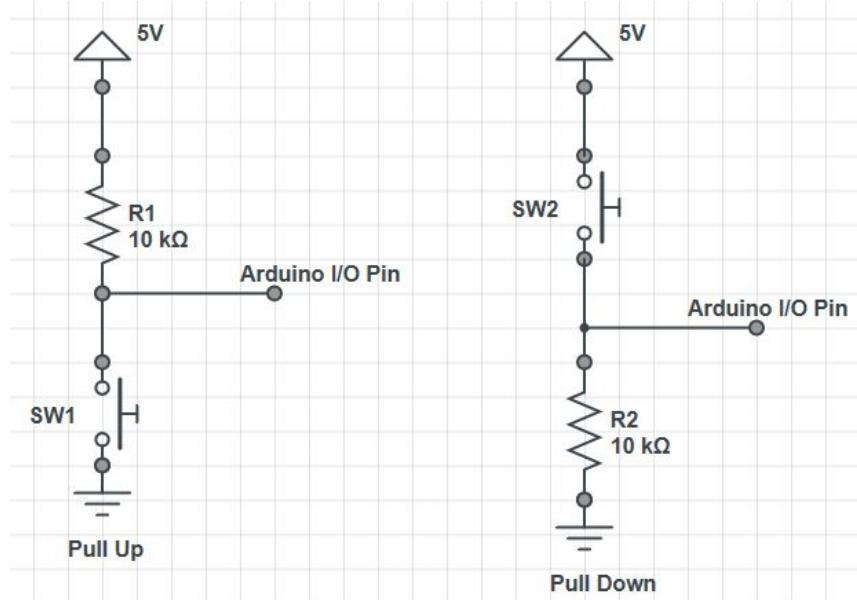


Figura 11

Es necesario aclarar, sin embargo, que los pulsadores son dispositivos que tienen un defecto, el cual se llama rebote. Cuando se presiona o se suelta el pulsador, se produce una fluctuación entre sus contactos internos, por lo tanto cuando se va a pasar de un 1 (HIGH) a un 0 (LOW) o viceversa, esas fluctuaciones son también leídas por el microcontrolador y se produce un comportamiento inesperado en el funcionamiento de nuestros proyectos. El efecto de este fenómeno puede ser visto con claridad en la Figura 12, y puede ser solucionado mediante un antirrebote, el cual puede realizarse tanto por software como por hardware.

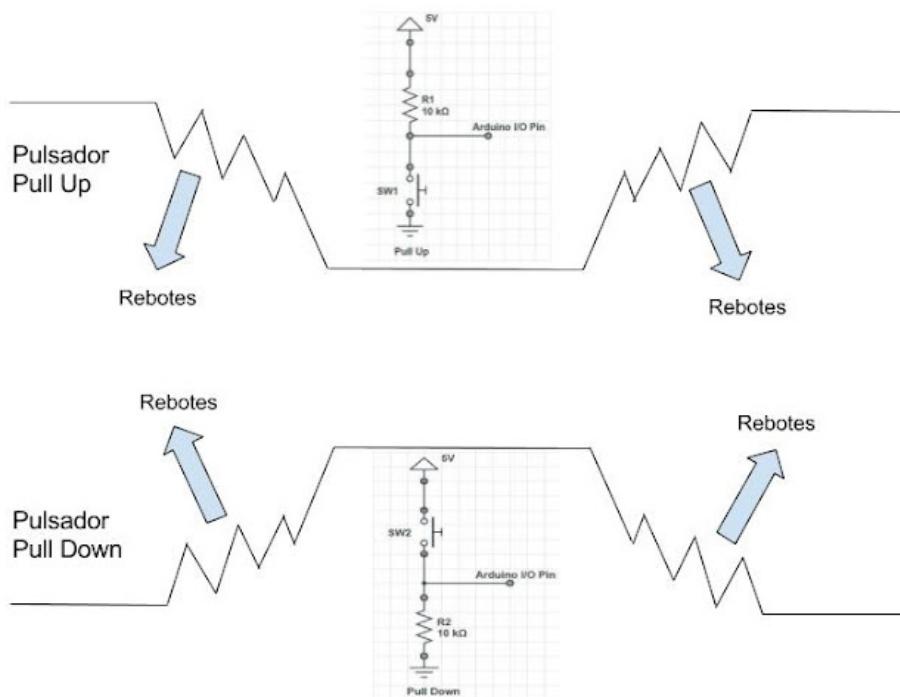


Figura 12

Por software, se coloca un condicional en donde una variable cambia su valor cuando el pulsador se presiona, es decir, cuando se encuentra en LOW. Luego de que la variable tenga su nuevo valor y cuando se suelte el pulsador, se realiza entonces la acción deseada en el proyecto. Por hardware, sin embargo, es muy efectivo poner un condensador de 10uF en el pulsador, como se puede ver en la Figura 13.

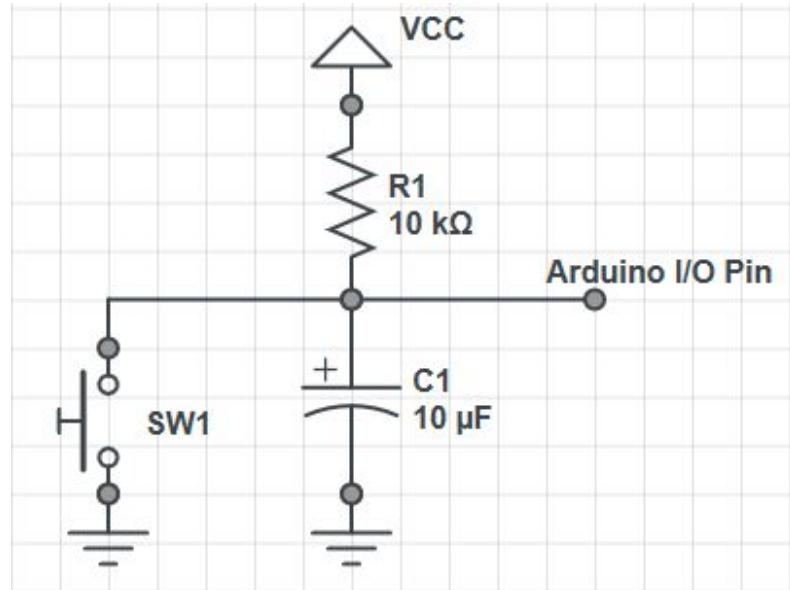


Figura 13

WeMOS D1 E1 Mini - ESP8266

Wemos D1 Mini ESP8266 es una plataforma de desarrollo similar a Arduino especialmente orientada al Internet de las cosas (IoT) ya que es un módulo Wi-Fi que permite la conexión por este medio. Un ejemplo de ella se puede ver en la Figura 14.

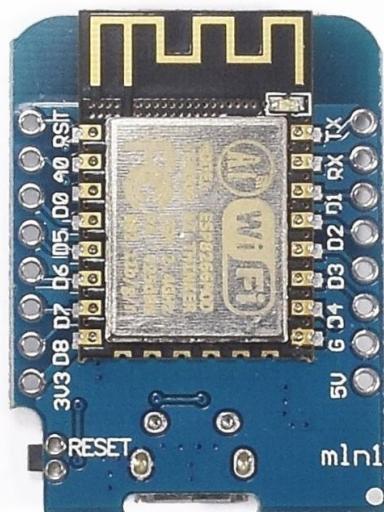


Figura 14

Wemos D1 mini está diseñado especialmente para trabajar montado en protoboard o soldado sobre una placa. Su diagrama puede ser visto en la Figura 15. En nuestro caso

realizamos una conexión por cable entre este módulo y la EDU-CIAA-NXP, y después con este nos conectamos una red Wi-Fi y hosteamos una página web con los datos del sensado del agua del tanque.

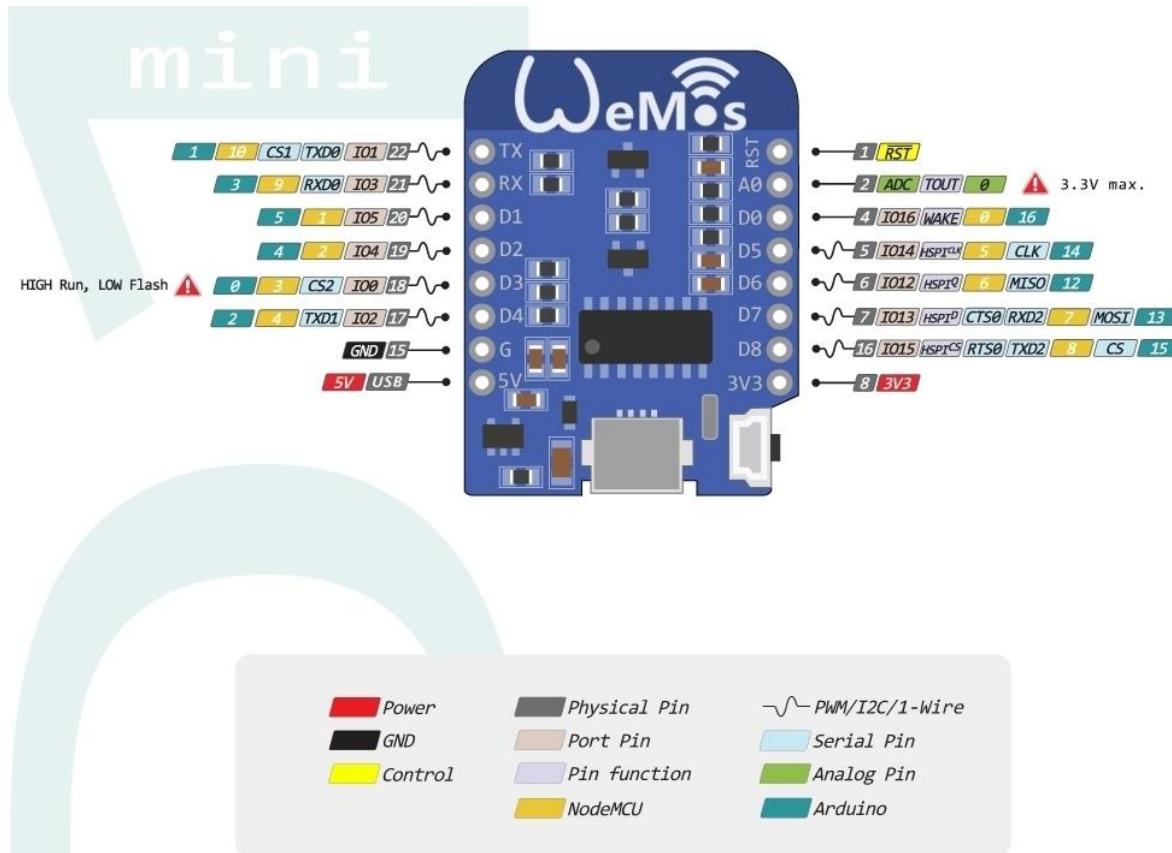


Figura 15

La placa Wemos D1 Mini ESP8266 tiene como núcleo al SoM ESP-12E que a su vez está basado en el SoC Wi-Fi ESP8266, integra además el conversor USB-Serial TTL CH340G y conector micro-USB necesario para la programación y comunicación a PC. Posee un regulador de voltaje de 3.3V en placa, esto permite alimentar la placa directamente del puerto micro-USB o por los pinos 5V y GND. Los pinos de entradas/salidas (GPIO) trabajan a 3.3V por lo que para conexión a sistemas de 5V es necesario utilizar conversores de nivel. La plataforma ESP8266 permite el desarrollo de aplicaciones en diferentes lenguajes como: Arduino, Lua, MicroPython, C/C++, Scratch.

Desarrollo

Como se puede inferir a partir de la introducción, este es un proyecto bastante complejo, que requiere usar tres microcontroladores distintos, un display y un joystick shield. Es por ello, que comenzamos el desarrollo de este trabajo con una fase de experimentación, para probar que el joystick shield montado sobre un Arduino UNO funcionaba correctamente y de la manera que nosotros esperamos. Utilizando los pines de salida del joystick shield, se deseaba que directamente pudiéramos enviar las respuestas de los botones utilizados (A, B y C) y de los movimientos analógicos (X e Y) a nuestro dispositivo principal donde se corre el código del juego, de esta manera no teniendo la necesidad de programar el Arduino UNO. Esto lo probamos primero que nada utilizando un LED, conectándole a la pata positiva el pin de salida de uno de los botones y a la pata negativa un pin GND; y en efecto, al presionar el botón correspondiente con el joystick shield montado sobre un Arduino UNO que no corría ningún código pudimos notar que el LED se prendía.

Lo siguiente que hicimos fue elegir el display que íbamos a utilizar, conectado a nuestro Arduino Mega 2560, en el cual se iba a correr el juego. El número de pines de este display está lejos de ser el ideal, pero nos tomamos nuestro tiempo para conectarlo y luego comenzamos a buscar ejemplos de uso de este display. Por desgracia, no encontramos ninguno, y sólo después de una búsqueda exhaustiva pudimos encontrar un link¹ a un proyecto que utilizaba este display con un Arduino Mega 2560. Afortunadamente, el mismo nos proveía de un link desde el cual pudimos descargar la librería oficial para controlar este módulo... pero el mismo no funcionaba, así que tuvimos que realizar otra búsqueda exhaustiva para poder encontrar la librería². Al menos, estábamos armados con el conocimiento de cómo se llamaba la misma, por lo que finalmente pudimos encontrar la librería UTFT y descargarla. De esta manera, pudimos probar que nuestro display funcionaba correctamente al ejecutar el ejemplo “UTFT_Demo_320x240” de la carpeta “Arduino (ARM) + Teensy” (lo único que tuvimos que cambiar del ejemplo es que nuestro display usa ILI9325D_8 y sus correspondientes pines de RS, WR, CS y RST).

Habiendo comprobado que todos nuestros dispositivos funcionan correctamente y que estaban bien conectados, procedimos a comenzar con el desarrollo de nuestro juego PONG. De esta forma, el diagrama de nuestro sistema quedó de la manera que se puede ver en la Figura 16. Para el juego, nos basamos en un ejemplo³ de un Arcade Retro utilizando componentes similares pero distintos, del cual pudimos extraer un conocimiento de cómo es que la lógica del juego y del joystick debería funcionar. Sin embargo, el uso de distintos componentes nos forzó a tener que modificar esta lógica, especialmente porque para nuestro display se debe utilizar otra librería, la cual tanto nos costó encontrar, que tiene funcionalidades distintas e incluso algunas ausentes. Una de esas funcionalidades ausentes que resultó ser crítica es que la librería UTFT no permite rotar el sentido del display, algo que era usado numerosas veces en el ejemplo original. Esto nos forzó a tener que usar exclusivamente una orientación horizontal para nuestro juego, a diferencia de la orientación vertical que se usaba en el ejemplo. Por lo tanto, después de una depuración muy importante, cambiamos suficientemente la lógica del juego para que termine siendo muy distinta de la original.

¹ <https://www.instructables.com/My-Second-Project-Arduino-24-TFT-LCD/>

² <http://www.rinkydinkelectronics.com/library.php?id=51>

³

https://create.arduino.cc/projecthub/bmcage/ingegno-retro-games-console-4188d9?ref=tag&ref_id=arcade&offset=12

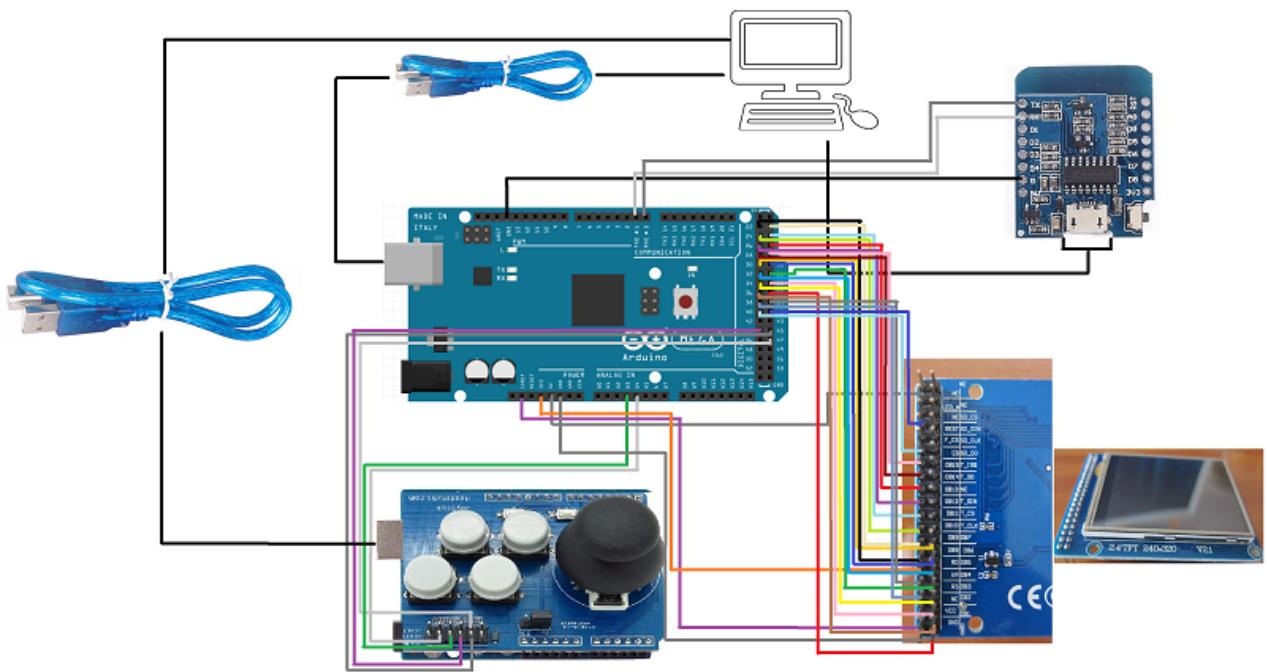


Figura 16

El principio de funcionamiento del juego es una pelota o ball, representada por un círculo llenado en el display, que en nuestro loop va a ir modificando su posición dinámicamente a partir de unos modificadores de posición vertical y horizontal. A partir de estos cambios, la pelota se redibuja, dando así el efecto de que se va moviendo por la pantalla. La misma, cuando choque con las paredes, también tiene que modificar su trayectoria en Y (al chocar con la pared superior la pelota tiene que ir para abajo y al chocar con la pared inferior la pelota tiene que ir para arriba), lo cual hace simplemente cambiando el valor de un modificador (A) de 1 a -1 y viceversa.

El otro elemento fundamental de nuestro juego son las paletas o paddle, que estarán controladas por los jugadores y la computadora. Su principio de funcionamiento es que las paletas se irán moviendo (redibujándose con distintos valores de posición, tal como la paleta) en un eje vertical de acuerdo a las entradas recibidas por parte del joystick (botones A y B para el jugador uno; y movimiento analógico en X para el jugador dos). Mientras tanto, para la computadora se calcula dinámicamente la trayectoria en la que se tiene que mover a partir de una inteligencia artificial muy básica. La gran importancia de las paletas en el juego PONG es que al impactar con la pelota, la pelota tiene que cambiar su trayectoria en X (si está yendo para la derecha tiene que rebotar hacia la izquierda y viceversa). Esto también lo hacemos simplemente cambiando el valor de un modificador (bx) de 1 a -1 y viceversa. Otros elementos de la lógica que diseñamos incluyen mostrar el resultado o score de manera actualizada en el partido; una vez uno de los dos jugadores llegue a cinco goles ganará el partido. Para anotar un gol, simplemente la pelota tiene que llegar a los extremos del display sin que sea interceptada por las paletas. Otro elemento que realizamos fue dibujar la cancha o court utilizando líneas verticales y horizontales de acuerdo a una cancha de tenis; y además, diseñamos un menú que aparecerá cuando se inicie el arcade el cual nos permite elegir si queremos jugar un partido en modo de un jugador o de dos jugadores. De esta manera, tenemos nuestro PONG completo, como se puede ver en la Figura 17.

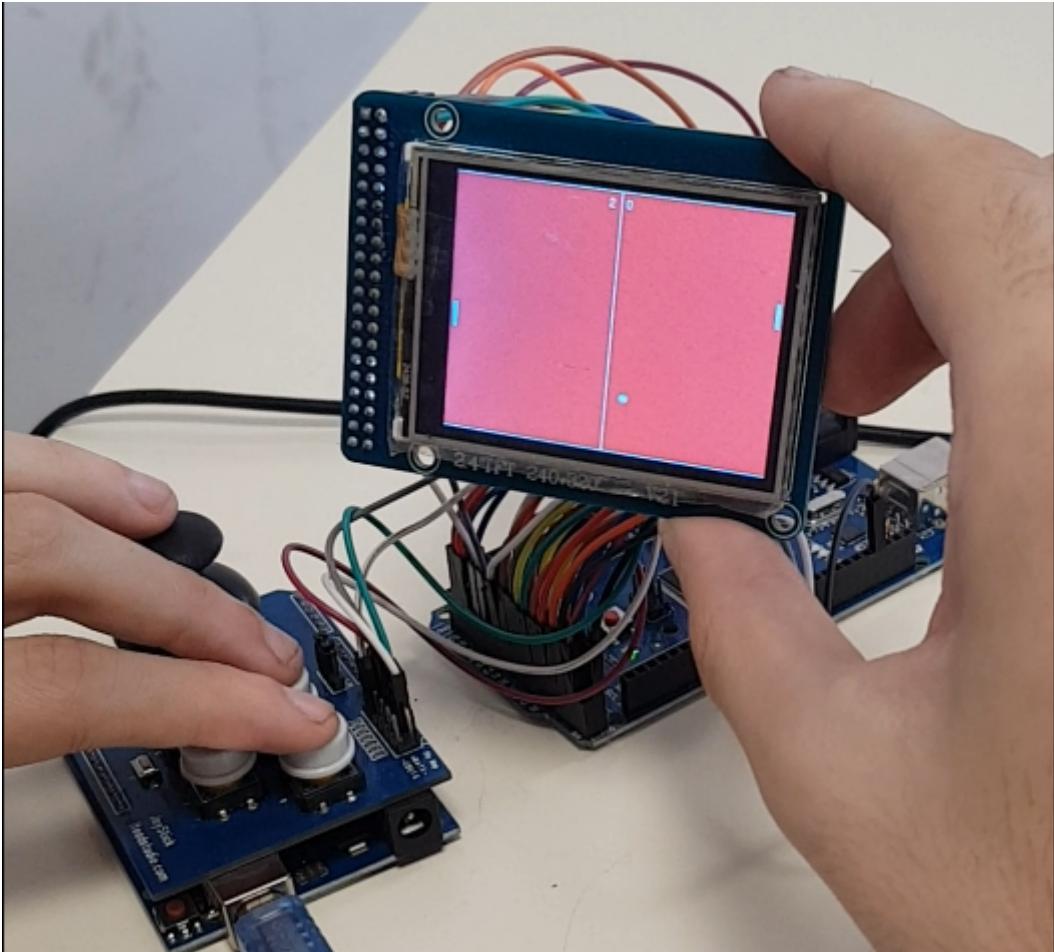


Figura 17

Finalmente, hemos hablado que diseñamos una inteligencia artificial muy rudimentaria. La misma calcula un número random en cuatro momentos fundamentales que corresponden a cuando la pelota pasa por cuatro puntos en X específicos desde un extremo a otro de la “cancha” (estos puntos son X igual a 50, 100, 150 y 200). De esta manera, desde que el jugador devolvía la pelota a la máquina, la máquina tiene la posibilidad de adoptar en cuatro puntos distintos una de los siguientes comportamientos: seguir la trayectoria de la pelota con precisión; detenerse y no realizar ningún movimiento; o moverse en dirección contraria a la pelota. La elección de este comportamiento se realiza a partir del número random, con un 10% de posibilidades de que se quede quieta, un 20% de posibilidades de que vaya en dirección contraria y un 70% de posibilidades de que devuelva la pelota. En el Algoritmo 1 se puede ver el funcionamiento de nuestra inteligencia artificial.

```

if (oneplayer) {
    if ((BPX==200) || (BPX==150) || (BPX==100) || (BPX==50)){
        Random = random(1, 10);
    }
    if (Random<=7){

        if (( bx == 1)||((BPX > 100) && ( bx == -1))) {
            if ((A == -1) && (BPy < (computerPaddle+PADDLESIZE/2))) {
                U = 1;
                D = 0;
            } //Computer simulation
            if ((A == 1) && (BPy > (computerPaddle+PADDLESIZE/2))) {
                D = 1;
                U = 0;
            }
        }
        else {
            D = 0;
            U = 0;
        }
    }

    if ((Random>7) && (Random<=9)){
        if (( bx == 1)||((BPX > 100) && ( bx == -1))) {
            if ((A == -1) && (BPy < (computerPaddle+PADDLESIZE/2))) {
                U = 0;
                D = 1;
            } //Computer simulation
            if ((A == 1) && (BPy > (computerPaddle+PADDLESIZE/2))) {
                D = 0;
                U = 1;
            }
        }
        else {
            D = 0;
            U = 0;
        }
    }

    if (Random>9){

        if (( A == 1)||((BPy > 100) && ( A == -1))) {
            if ((bx == -1) && (BPX < (computerPaddle-PADDLESIZE/2))) {
                U = 1;
                D = 0;
            } //Computer simulation
            if ((bx == 1) && (BPX > (computerPaddle-PADDLESIZE/2))) {
                D = 1;
                U = 0;
            }
        }
        else {
            D = 0;
            U = 0;
        }
    }
}

```

Algoritmo 1

Ahora bien, tuvimos que además diseñar un nuevo programa que estará alojado en el ESP8266 para poder hostear una página web que se recargue dinámicamente cada dos segundos con los resultados o scores de las partidas jugadas en el PONG. Para esto, nos basamos bastante en el programa que habíamos hecho en el trabajo de laboratorio anterior, cambiando simplemente algunos detalles. Para la comunicación serial, en vez de usar la librería SoftwareSerial, probamos una teoría de que en realidad no es necesaria si lo único que se busca es comunicar dos dispositivos. Esto es así porque los Arduino directamente tienen integrado la librería Serial utilizando su Serial0 (que además está asociado a la

comunicación por USB), en el caso del Arduino Mega con los pines TX_0 y RX_0 y en el caso del ESP8266 con sus únicos pines de TX y RX. Además, empleamos una función específica de Serial llamada serialEvent, la cual nos permitió identificar que ha habido un evento serial y poder responder al mismo almacenando la entrada que recibimos por comunicación serial en una variable de tipo String.

Los valores enviados además tienen que tener un carácter de corte (_) para poder hacer que nos demos cuenta que ya no hay más valores por recibir. De esta manera, cada uno de los resultados o scores recibidos serán almacenados temporalmente y luego se pasarán a la página web en formato html. Desde el lado del Arduino Mega 2560 también tuvimos que agregar una funcionalidad para que por comunicación serial sean enviados los resultados calculados a partir de los goles del ganador y del rival (con algoritmos distintos para los casos de que ganes contra otro jugador o contra la máquina), siempre agregándole el carácter de corte al final de la transmisión. De esta manera, logramos mostrar en la página web los resultados o scores a partir de los partidos jugados en el PONG, como se puede ver en la Figura 18. Una vez termina el partido, finalmente, se vuelve a mostrar el menú inicial para poder seleccionar si se quiere volver a jugar otra partida y en qué modo.

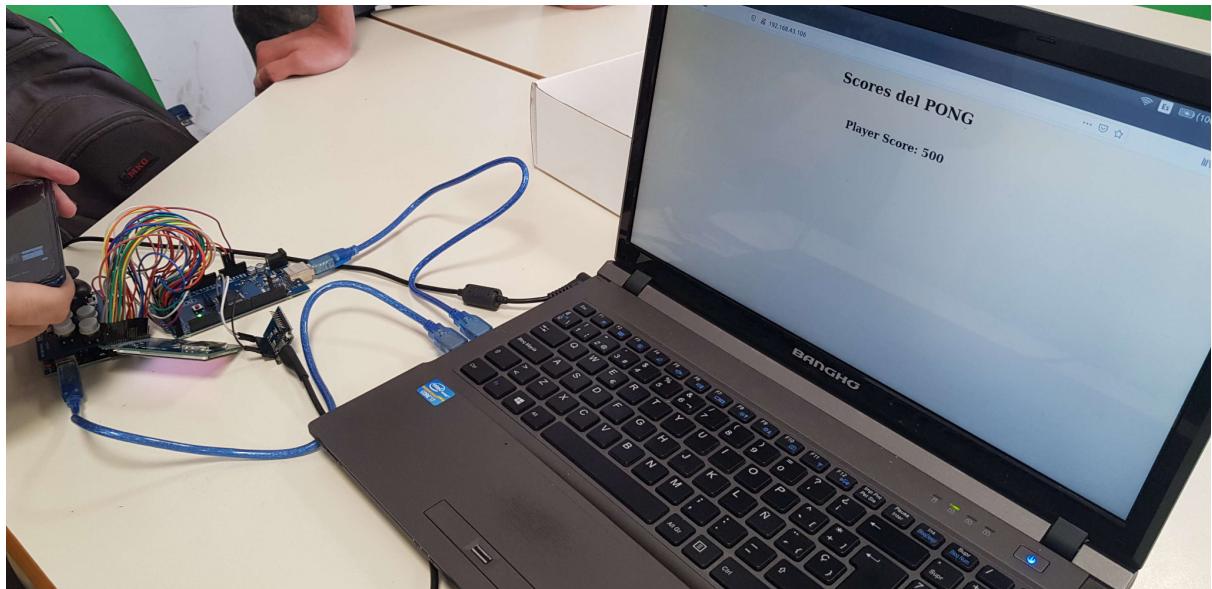


Figura 18

Conclusión

Pudimos lograr satisfactoriamente nuestros objetivos. Hasta el momento, fue el trabajo más complejo que realizamos, ya que utilizamos cinco módulos diferentes (dos arduinos, el display, el joystick shield, y el ESP8266) como base para el desarrollo del mismo. Lo cual implicó tener que investigar sobre estos, usar conocimientos previos y adquirir los nuevos necesarios, muchas pruebas, errores y aciertos, etc. Especialmente fue la primera vez que trabajamos con un display y un joystick, aprendiendo muchísimo de estos.

Consideramos que toda la lógica del PONG y el manejo del display fueron mucho más profundas y difíciles que lo que nos esperábamos, pero el lograr entenderlas y hacerlas funcionar, fue realmente gratificante. Teniendo en cuenta también que nosotros mismos elegimos este proyecto, ya que compartimos un gusto especial por los videojuegos, esta experiencia nos permitió aprender también más de ellos y de su funcionamiento interno. Profundizar en aquello de lo que no éramos tan conscientes. Por último, queremos despedirnos con una toma muy buena del sistema que hicimos, mostrando la pantalla de victoria una vez que finaliza la partida, la cual se puede ver en la Figura 19.

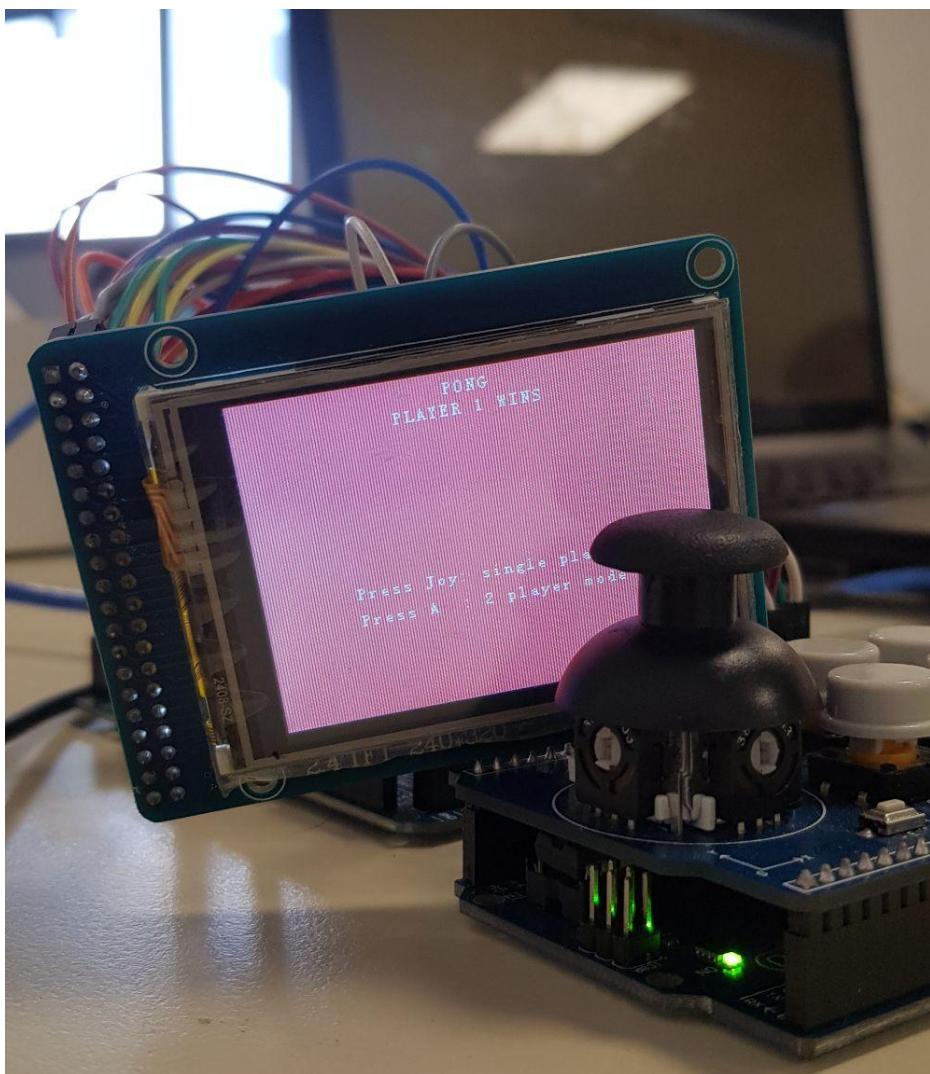


Figura 19

Mejoras a Desarrollar

Hay varias mejoras y posibles ideas futuras que se nos ocurrieron para este proyecto.

Una de ellas es mostrar los puntajes con mayor detalle: poder identificar a los distintos jugadores mediante una interfaz que mostrariamos por el display y que controlaríamos con el Joystick Shield (como en los Arcade antiguos, donde te salen cinco caracteres, y vos podes mover hacia abajo o hacia arriba para cambiar el valor de cada carácter, mientras que movimiento hacia la derecha o hacia la izquierda cambias de posición de carácter). Estos nombre de jugador y resultados además podrían persistir dentro del Arduino Mega 2560, almacenando estos valores en la memoria EEPROM del microcontrolador y luego enviándolos apenas se inicie el sistema al ESP8266. De esta manera, podríamos mostrar un listado con un historial de todos los jugadores y su puntaje total, similar a los de las máquinas arcade, y así lograr una mayor competitividad.

Otra posible mejora es incrementar la dificultad del juego, ya sea mediante el aumento de la velocidad y reducción del tamaño de las paletas (paddles) que se realice de forma dinámica a medida que transcurre el juego y/o seleccionando un nivel de dificultad en el menú inicial de forma estática. Sin embargo, para esto es necesario adaptar el algoritmo que controla la inteligencia de la computadora, ya que cuando intentamos realizar este cambio el algoritmo es muy lento para seguir la velocidad del juego y no funciona correctamente.

Apuntando más alto, también una posible idea es hacer un arcade más ambicioso que posea varios juegos (entre estos el PONG que se logró crear en este proyecto), cuya interfaz permita elegir entre estos y jugarlos, con puntajes para cada juego individual y también uno global.

Apéndice 1: Códigos

Adjuntamos los códigos del Arduino Mega 2560 (*ArduinoPong.ino* y *joystick.cpp*) y del módulo WeMOS D1 E1 Mini - ESP8266 (*arduinoWebServer.ino*)

ArduinoPong.ino

```
#include <UTFT.h>

#include "joystick.cpp"

#define TFT_RST 41 // Reset line for TFT (or connect to +5V)
#define TFT_CS 40 // Chip select line for TFT display
#define TFT_WR 39 // Write/Read
#define TFT_RS 38 // RS

//TFT resolution 320*240

#define MIN_X 0
#define MIN_Y 0
#define MAX_X 240
#define MAX_Y 320

#define PADDLESIZE 22 //size of the paddle
#define BALLSIZE 4 //radius size of the ball
#define PADDLEREGION 3 //pixels around paddle that still hit ball

#define slowdownspeed 5 //game too fast, increase this to slow it down. A delay in ms

extern uint8_t SmallFont[];

UTFT myGLCD(ILI9325D_8, TFT_RS, TFT_WR, TFT_CS, TFT_RST);
```

```

//obtain a joystick

Joystick js;

float BPX, BPX_OLD; // Ball Position X

float BPY, BPY_OLD; // Ball Position Y

int byx; // Ball position in y after receiving input from A

int bx; // Ball direction in x

int A; // Ball Direction in y (1 = left to right, -1 = right to left)

bool oneplayer = true;

int playerScore; // These variables hold the player & computer score.

int computerScore;

int sendScore;

#define WINSCORE 5 // When you reach this value you win


int R, L, D, U; // These values adjust the paddles L & R from player
buttons, U & D from computer algorithm

int playerPaddle; // Player Paddle Position

int computerPaddle; // Computer's Paddle Position

int Random;

void DrawCourt(boolean onlycenter) // Draw the play court lines (pass 1 to
only have the center line drawn for speed)

{

if(!onlycenter) {

    myGLCD.setColor(0, 255, 255);

    myGLCD.drawHLine(0,0,319);

    myGLCD.drawHLine(0,239,319);
}

```

```

    }

    myGLCD.drawVLine(160, 0, 239); // Center Line

}

void DisplayScore(int playerScore, int computerScore)
{
    myGLCD.setColor(0, 255, 255);

    myGLCD.setBackColor(VGA_TRANSPARENT);

    myGLCD.print( String(playerScore) + " " + String(computerScore), CENTER, 5);
}

void EraseScore(int playerScore, int computerScore){
    myGLCD.setColor(255, 0, 0);

    myGLCD.setBackColor(VGA_TRANSPARENT);

    myGLCD.print( String(playerScore) + " " + String(computerScore), CENTER, 5);
}

void erasePaddle() {
    myGLCD.setColor(255, 0, 0);

    myGLCD.drawVLine(3, pLayerPaddle-1, PADDLESIZE+2);

    myGLCD.drawVLine(4, pLayerPaddle-1, PADDLESIZE+2);

    myGLCD.drawVLine(5, pLayerPaddle-1, PADDLESIZE+2);

    myGLCD.drawVLine(6, pLayerPaddle-1, PADDLESIZE+2);

    myGLCD.drawVLine(314, computerPaddle-1, PADDLESIZE+2);

    myGLCD.drawVLine(315, computerPaddle-1, PADDLESIZE+2);

    myGLCD.drawVLine(316, computerPaddle-1, PADDLESIZE+2);
}

```

```

myGLCD.drawLine(317, computerPaddle-1, PADDLESIZE+2);

}

void drawPaddle() {
    myGLCD.setColor(255, 0, 0);
    myGLCD.drawHLine(3, playerPaddle-1, 3);
    myGLCD.drawHLine(3, playerPaddle, 3);
    myGLCD.drawHLine(3, playerPaddle+PADDLESIZE+1, 3);
    myGLCD.drawHLine(3, playerPaddle+PADDLESIZE+2, 3);
    myGLCD.setColor(0, 255, 255);
    myGLCD.fillRect(3, playerPaddle, 6, playerPaddle + PADDLESIZE);

    if (playerPaddle==1) // This is so the paddle does not move off the screen
on the bottom side of the screen
        playerPaddle=2;
    if (playerPaddle==MAX_X-PADDLESIZE-1) // This is so the paddle does not move
off the screen on the top side of the screen
        playerPaddle=MAX_X-PADDLESIZE-2;

    myGLCD.fillRect(314, computerPaddle, 317, computerPaddle + PADDLESIZE);
    myGLCD.setColor(255, 0, 0);
    myGLCD.drawHLine(314, computerPaddle-1, 3);
    myGLCD.drawHLine(314, computerPaddle, 3);
    myGLCD.drawHLine(314, computerPaddle+PADDLESIZE+1, 3);
    myGLCD.drawHLine(314, computerPaddle+PADDLESIZE+2, 3);

    if (computerPaddle==1)
        computerPaddle=2;
}

```

```

if (computerPaddle==MAX_X-PADDLESIZE-1)

    (computerPaddle=MAX_X-PADDLESIZE-2);

}

void setup(){

Serial.begin(115200);

myGLCD.InitLCD();

myGLCD.setFont(SmallFont);

myGLCD.fillRect(0, 0, 0); // clear display

js.init();

// Draw splash screen text

myGLCD.setColor(255, 255, 255);

myGLCD.print("PONG", CENTER, 5);

myGLCD.print("Press Joy: single player", CENTER, 140);

myGLCD.print("Press A : 2 player mode", CENTER, 160);

while( ! Joystick::JoystickPressed() && ! js.A()); // New game when

switch 4 is pressed

if (Joystick::JoystickPressed())

oneplayer = true;

else

oneplayer = false;

```

```

myGLCD.fillScr(255, 0, 0); // clear screen again

DrawCourt(0); // Draw court lines

playerScore=0;
computerScore=0;

DisplayScore(playerScore, computerScore);

BPX = 160;
BPY = 120;
byx=120;
bx=1;
A=1;

playerPaddle=MAX_X/2-PADDLESIZE/2;
computerPaddle=MAX_X/2-PADDLESIZE/2;

randomSeed(analogRead(2));

}

void Loop() {

if (oneplayer) {

if ((BPX==200) || (BPX==150) || (BPX==100) || (BPX==50)){ //The
probability of the computer making an error

Random = random(1, 10);

}

if (Random<=7){

}
}

```

```

if (( bx == 1) ||((BPX > 100) && ( bx == -1))) {

if ((A == -1) && (BPY < (computerPaddle+PADDOLESIZE/2))) {

U = 1;

D = 0;

} //Computer simulation

if ((A == 1) && (BPY > (computerPaddle+PADDOLESIZE/2))) {

D = 1;

U = 0;

}

}

else {

D = 0;

U = 0;

}

}

if ((Random>7) && (Random<=9)) {

if (( bx == 1) ||((BPX > 100) && ( bx == -1))) {

if ((A == -1) && (BPY < (computerPaddle+PADDOLESIZE/2))) {

U = 0;

D = 1;

} //Computer simulation

if ((A == 1) && (BPY > (computerPaddle+PADDOLESIZE/2))) {

D = 0;

U = 1;

}

}

}

```

```

else {
    D = 0;
    U = 0;
}

}

if (Random>9){

    if (( A == 1)||((BPY > 100) && ( A == -1))) {

        if ((bx == -1) && (BPX < (computerPaddle-PADDLESIZE/2))) {

            U = 1;
            D = 0;

        } //Computer simulation

        if ((bx == 1) && (BPX > (computerPaddle-PADDLESIZE/2))) {

            D = 1;
            U = 0;

        }

    }

    else {

        D = 0;
        U = 0;
    }

}

} else {

    // twoplayer mode, bottom player with Joystick

    int side = js.getX();

    if (side > 0) {U=0; D=1;}
}

```

```

    else if (side < 0) {U=1; D=0;}

    else {U=0; D=0;}

}

DrawCourt(0); // Draw court line(s)

DisplayScore(playerScore, computerScore);

// see if player is using A & B buttons to signal they wish to move the player
paddle

R = js.B();

L = js.A();

playerPaddle=playerPaddle+R; //These equations are for the movement of the
paddle, R, L, D, and U are all boolean. paddles initially equal 48. This is
so

//at startup the paddles are at center.

playerPaddle=playerPaddle-L;

computerPaddle=computerPaddle+D; //I used D and U because i use the buttons
for other applications but they can be defined as player2 R and L

computerPaddle=computerPaddle-U;

drawPaddle();

byx=(A+byx);

BPX_OLD = BPX;

BPY_OLD = BPY;

BPY=((byx));

BPX=(bx+BPX);

//choque con pared

if ((BPY == 236) || (BPY == 2)){

```

```

(A=(-1*A));
}

else {
};

//bounce ball back if we hit it with paddle

if ((BPY<=(computerPaddle+PADDLESIZE+BALLSIZE+PADDLEREGION))
&& (BPY>=(computerPaddle-BALLSIZE-PADDLEREGION)) && (BPX == 314)){
(bx=(-1*bx));
}

else{
};

if ((BPY<=(playerPaddle+PADDLESIZE+BALLSIZE+PADDLEREGION))
&& (BPY>=(playerPaddle-BALLSIZE-PADDLEREGION)) && (BPX==6))){
(bx=(-1*bx));
}

else{
};

delay(slowdownspeed);

if (BPX >= MAX_Y - 2 || BPX <= 2){      // someone scored!
EraseScore(playerScore, computerScore);

if (BPX >= MAX_Y - 2)
playerScore = playerScore+1;

else
computerScore = computerScore+1;

// Reset:                                // reset court after score

```

```

DisplayScore(playerScore, computerScore);

DrawCourt(0);

myLCD.setColor(255, 0, 0);

myLCD.fillCircle(BPX, BPY, 7); // erase ball in Last known Location

BPX=160; // set ball to center of screen

BPY=120;

//center the paddles

erasePaddle();

playerPaddle=MAX_X/2-PADDLESIZE/2; // set paddles in the center of the display

computerPaddle=MAX_X/2-PADDLESIZE/2;

drawPaddle();

myLCD.setColor(0, 255, 255);

myLCD.fillCircle(BPX, BPY, 4); // draw ball in center

delay(3000); // delay 3 seconds

myLCD.setColor(255, 0, 0);

myLCD.fillCircle(BPX, BPY, 7);

byx=120;

}

//we allow score to disappear as rotating screen + drawing is SLOW !!

//DisplayScore(playerScore, computerScore);

```

```

if(playerScore==WINScore || computerScore==WINScore) { // if someone hit the
winning score then game over - print who one and reset game

myLCD.fillScr(255, 0, 0);

myLCD.setColor(255, 255, 255);

}

if (playerScore==WINScore){ // player wins

myLCD.print("PLAYER 1 WINS", CENTER, 20);

sendScore = playerScore * 100 - computerScore * 50;

}

else {

if (oneplayer){

myLCD.print("COMPUTER WINS", CENTER, 20); // computer wins

sendScore = 50 + playerScore * 50;

}

else{

myLCD.print("PLAYER 2 WINS", CENTER, 20); // player 2 wins

sendScore = computerScore * 100 - playerScore * 50;

}

}

Serial.print(sendScore);

Serial.print("_");

//NEWGAME: //Resets the screen for a new game

}

//center the paddles

playerPaddle=MAX_X/2-PADDLESIZE/2; // set paddles in the center of the
display

computerPaddle=MAX_X/2-PADDLESIZE/2;

```

```

// Draw splash screen text

myLCD.setColor(255, 255, 255);

myLCD.print("PONG", CENTER, 5);

myLCD.print("Press Joy: single player", CENTER, 140);

myLCD.print("Press A : 2 player mode", CENTER, 160);

while( !(Joystick::JoystickPressed()) && !(js.A()))); // New game when
joystick is pressed

if (Joystick::JoystickPressed())

onePlayer = true;

else

onePlayer = false;

myLCD.fillScr(255, 0, 0);

myLCD.setColor(255, 0, 0);

myLCD.fillCircle(BPX, BPY, 7);

BPX = 160;

BPY = 120;

byx=120;

bx=1;

A=1;

myLCD.setColor(0, 255, 255);

myLCD.fillCircle(BPX, BPY, 4);

myLCD.setColor(255, 0, 0);

myLCD.fillCircle(BPX, BPY, 7);

myLCD.setColor(0, 255, 255);

myLCD.fillCircle(BPX, BPY, 4);

myLCD.setColor(255, 0, 0);

myLCD.fillCircle(BPX, BPY, 7);

```

```

computerScore=0;

playerScore=0;

DrawCourt(0);

DisplayScore(playerScore, computerScore);

delay(2000);      // wait 4 seconds to start new game

}

myGLCD.setColor(255, 0, 0);

myGLCD.fillCircle(BPX_OLD, BPY_OLD, BALLSIZE); //Erase ball

myGLCD.setColor(0, 255, 255);

myGLCD.fillCircle(BPX, BPY, BALLSIZE); //This is the actual ball

}

```

joystick.cpp

```

#ifndef JOYSTICKCPP
#define JOYSTICKCPP

#include <Arduino.h>

// analog pins

#define XPIN      A4
#define YPIN      A3

// digital pin

#define FIREPIN    44 // D3 (A) to start and drop block
#define ROTATEPIN  45 // D2 (B) to rotate block
#define JOYSTICKPIN 47 // D4 press joystick

// joystick JOYCENTER for both axis

```

```

#define JOYCENTER      512

class Joystick
{
public:

    // joystick position constants

    static const int NEUTRAL = 0;
    static const int SOFT = 1;
    static const int HARD = 2;
    static const int HARDER = 3;

    int FIREPINPRESSED;    //HIGH
    int ROTATEPINPRESSED; //HIGH

    void init ()
    {
        FIREPINPRESSED = HIGH;
        ROTATEPINPRESSED = HIGH;
        pinMode ( FIREPIN, INPUT_PULLUP ); //if no pullup resistor
        //pinMode ( FIREPIN, INPUT );
        pinMode ( ROTATEPIN, INPUT_PULLUP );
        //pinMode ( ROTATEPIN, INPUT );
        //we test if LOW when unpressed
        if (digitalRead(FIREPIN)==HIGH && digitalRead(ROTATEPIN)==HIGH) {
            FIREPINPRESSED = LOW;
            ROTATEPINPRESSED = LOW;
        }
        pinMode ( JOYSTICKPIN, INPUT_PULLUP );
    }

    // X positie als waarde tussen -4 en 4
    static int getX()
    {
        return getPosition(XPIN) * -1;
    }

    // Y positie als waarde tussen -4 en 4
    static int getY()
    {

```

```

        return getPosition(YPIN) * +1;
    }

boolean A()
{
    return digitalRead(FIREPIN) == FIREPINPRESSED;
}

boolean fire()
{
    return digitalRead(FIREPIN) == FIREPINPRESSED;
}

boolean RotatePushed()
{
    return digitalRead(ROTAPEPIN) == ROTATEPINPRESSED;
}

boolean B()
{
    return digitalRead(ROTAPEPIN) == ROTATEPINPRESSED;
}

static boolean JoystickPressed()
{
    return digitalRead(JOYSTICKPIN) == LOW;
}

void waitForRelease()
{
    while (fire());
}

void waitForRelease(int howLong)
{
    int c = 0;
    do
    {
        delay (10);
        c += 10;
    }
    while ((fire() || getY() != 0 || getX() != 0) && c < howLong);
}

```

```

}

void waitForClick()
{
    while (!fire());
}

private:

static int getPosition (int pin)
{
    const int n = analogRead(pin) - JOYCENTER;

    return n / 128;
}

};

#endif

```

arduinoWebServer.ino

```

#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>

const char* ssid = "AndroidAP0137";
const char* password = "wendy12345";

ESP8266WebServer server(80);

String scores = "";
String input = "";
bool receivedData = false;

const char webpage[] PROGMEM = R"=====(
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Scores del PONG</title>
</head>
<body style="background-color: #f9e79f ">

```

```

<center>
<div>
<h1>Scores del PONG</h1>
</div>
<br>
<div><h2>
    <span id="scoresData"></span><br><br>
</h2>
</div>
<script>
setInterval(function()
{
    getData();
}, 2000);
function getData() {
    var ajax = new XMLHttpRequest();
    ajax.onreadystatechange = function() {
        if (this.readyState == 4 && this.status == 200) {
            document.getElementById("scoresData").innerHTML =
            this.responseText;
        }
    };
    ajax.open("GET", "dataRead", true);
    ajax.send();
}
</script>
</center>
</body>
</html>
)=====;

void scoresData(){
    server.send(200, "text/plain", scores);
}

void handleRoot()
{
    String s = webpage;
    server.send(200, "text/html", s);
}

void setup(void)
{

```

```

Serial.begin(115200);
input.reserve(200);

WiFi.begin(ssid, password);
Serial.println("");

while (WiFi.status() != WL_CONNECTED)
{
    Serial.print("Connecting...");
}

Serial.println("");
Serial.print("Connected to ");
Serial.println(ssid);
Serial.print("IP address: ");
Serial.println(WiFi.localIP());

server.on("/", handleRoot);
server.on("/dataRead", scoresData);
server.begin();
}

void loop(void)

{
    server.handleClient();
    if(receivedData){
        Serial.print(input);
        scores = scores + "Player Score: " + input + "<br>";
        scoresData();
        receivedData = false;
        input = "";
    }
    delay(2000);
}

void serialEvent(){
    while(Serial.available()){
        char inChar = (char)Serial.read();
        if(inChar == '_'){
            receivedData = true;
        }else{

```

```
    input += inChar;
}
}
f
```

Apéndice 2: Videos

Video de PONG en Modo de Dos Jugadores:

https://drive.google.com/file/d/11tML6UBiAqRU2cHJGuPmkExDAih5L_Eb/view?usp=sharing

Video de PONG en Modo de Un Jugador:

<https://drive.google.com/file/d/120944ynjpH3fSMnNABt07xA02rQWyxcc/view?usp=sharing>

Bibliografía

AJAX with ESP8266: Dynamic Web Page Update Without Reloading. Circuit Digest.

Recuperado el 26/10/2022 de:

<https://circuitdigest.com/microcontroller-projects/ajax-with-esp8266-dynamic-web-page-update-without-reloading>

Arduino docs. Arduino. Recuperado 4/12/2022:

<https://www.arduino.cc/>

Arduino Mega 2560 DataSheet. Robot Shop. Recuperado el 4/12/2022:

<http://eprints.polsri.ac.id/4598/8/File%20VIII%20%28Lampiran%29.pdf>

Arduino Mega Pinout. Javatpoint. Recuperado el 4/12/2022:

<https://www.javatpoint.com/arduino-mega-pinout>

Arduino Mega 2560 rev3. Store Arduino. Recuperado el 4/12/2022:

<https://store.arduino.cc/products/arduino-mega-2560-rev3>

Arduino UNO Pinout Guide. Circuito.io Blog. Recuperado el 4/12/2022:

<https://www.circuito.io/blog/arduino-uno-pinout/>

Arduino UNO rev3. Store Arduino. Recuperado el 4/12/2022:

<https://store.arduino.cc/products/arduino-uno-rev3>

Botones. Factor Evolución. Recuperado el 26/10/2022 de:

<https://www.factor.mx/portal/base-de-conocimiento/botones/>

Comunicación Serial. Código Facilito. Recuperado el 4/12/2022:

<https://codigofacilito.com/articulos/comunicacion-serial>

Comunicación Serial con Arduino. Control automático educación. Recuperado el 4/12/2022:

<https://controlautomaticoeducacion.com/arduino/comunicacion-serial-con-arduino/#:~:text=El%20TX%20y%20RX%20del,por%20medio%20del%20protocolo%20serial.>

Comunicación serie arduino. Aprendiendo Arduino. Recuperado el 4/12/2022:

<https://aprendiendoarduino.wordpress.com/2016/07/02/comunicacion-serie-arduino/>

Display LCD tft 24 RGB Spi 240x320 táctil ili9341. Naylamp Mechatronics. Recuperado el 4/12/2022:

<https://naylampmechatronics.com/lcd-color/144-display-lcd-tft-24-rbg-spi-240x320-tactil-ili9341.html>

El Serial.Write() Y El Serial.Print(). JOOBER Technologies. Recuperado el 1/11/2022 de:

<https://www.joober.eu/el-serial-write-y-el-serial-print-en-arduino/>

Ingegno Retro Games Console. Create Arduino. Recuperado el 4/12/2022:

https://create.arduino.cc/projecthub/bmcage/ingegno-retro-games-console-4188d9?ref=tag&ref_id=arcade&offset=12

ITEAD Joystick Shield.Wiki Iteadstudio. Recuperado el 4/12/2022:

https://wiki.iteadstudio.com/ITEAD_Joystick_Shield

Library UTFT. Rinky Dink Electronics. Recuperado el 4/12/2022:

<http://www.rinkydinklelectronics.com/library.php?id=51>

My Second Project: Arduino 2.4 TFT LCD. Instructables. Recuperado el 4/12/2022:

<https://www.instructables.com/My-Second-Project-Arduino-24-TFT-LCD/>

Pantalla LCD TFT 2.4 pulgadas 240x320 Lector SD. Solectro Shop. Recuperado el

4/12/2022:

<https://solectroshop.com/es/pantalla-tft/49-pantalla-lcd-tft-24-pulgadas-240x320-lector-sd.html>

Pulsadores y antirrebote con Arduino. Arduino. Recuperado el 26/10/2022 de:

<http://arduparatodos.blogspot.com/2017/01/pulsadores-y-antirrebote-con-arduino.html>

Qué es UART. Rohde & Schwarz. Recuperado el 26/10/2022 de:

https://www.rohde-schwarz.com/es/productos/test-y-medida/osciloscopios/educational-content/que-es-uart_254524.html

Usando ESP8266 con el IDE de Arduino. Naylamp Mechatronics. Recuperado el

26/10/2022:

https://naylampmechatronics.com/blog/56_usando-esp8266-con-el-ide-de-arduino.html

WEMOS D1 MINI ESP8266 WIFI. Naylamp Mechatronics. Recuperado el 26/10/2022 de:

<https://naylampmechatronics.com/espressif-esp/291-wemos-d1-mini-esp8266-wifi.html>

What is an LCD Display. Orient Display. Recuperado el 4/12/2022:

<https://www.orientdisplay.com/knowledge-base/lcd-basics/what-is-lcd-liquid-crystal-display/>