

# Informe TP Final Arduino

**Proyecto:** Juego de Adivinanza y Medición con Arduino

**Grupo:** 6

**Integrantes:**

- Alcoceba Manuel
- Mokorel Valentín
- Miremont Matías
- Arbelo Ramiro

### **Resumen del proyecto:**

Nuestro proyecto consiste en un juego interactivo que combina adivinanzas, luces, sonidos y un sensor ultrasónico.

La primera parte del juego consiste en adivinar un número entre 10 y 20 usando un teclado matricial. El sistema va dando pistas visuales con los distintos leds (rojo, amarillo y azul) para indicar qué tan cerca está la respuesta, y además muestra los mensajes en una pantalla LCD.

Cuando el jugador acierta el número, el sistema reproduce la melodía "la marcha de boca" con el buzzer y pasa a una segunda fase, donde el jugador tiene que colocar un objeto a una distancia del sensor ultrasónico, dicha distancia debe coincidir con el número adivinado.

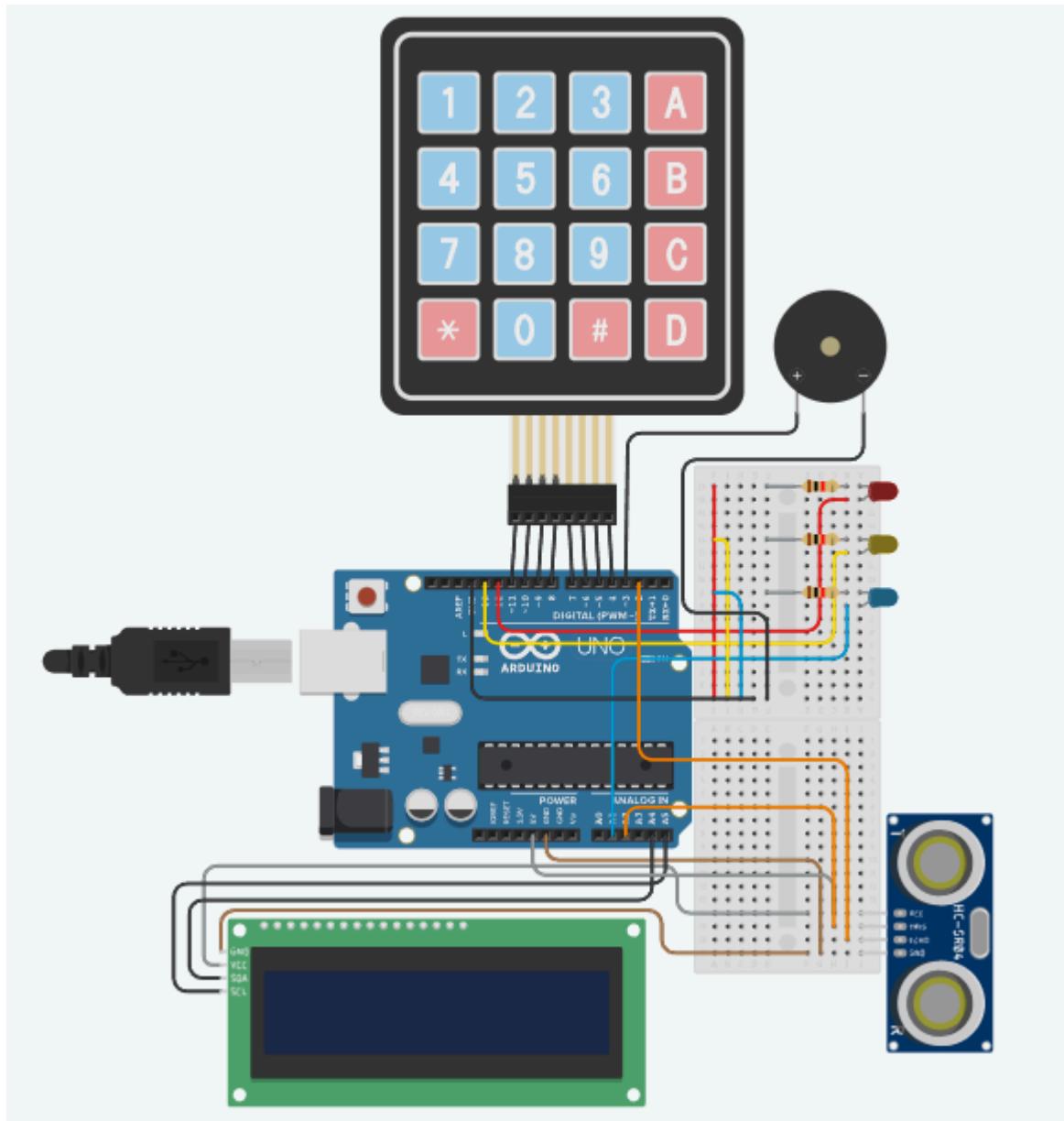
En resumen, el proyecto mezcla juego, electrónica y programación, y nos permitió poner en práctica varios temas vistos en clase, como el uso de periféricos, manejo de entradas y salidas digitales, control por tiempo y lógica de estados.

### **Descripción técnica del hardware:**

#### **Componentes utilizados:**

- **Microcontrolador Arduino UNO**
- **Pantalla LCD 16x2 con interfaz I2C (dirección 0x27)**
- **Keypad 4x4**
- pin 4, pin 5, pin 6, pin 7, pin 8, pin 9, pin 10, pin 11
- **Leds:**
- rojo (pin 12), amarillo (pin 13), azul (pin A1)
- **Buzzer**
- pin 3, GND
- **Sensor ultrasónico HC-SR04**
- Trigger(A2), Echo (pin 2), GND, 5V
- **3 resistencias, 1 para cada Led**
- **Cables y protoboard**
- **Alimentación: 5V (VCC)**

Diagrama esquemático o conexiones:



Descripción del Software y su estructura:

El programa fue desarrollado en Arduino IDE, utilizando el lenguaje C++ y las librerías Wire.h, LiquidCrystal\_I2C.h y Keypad.h.

**Función principal del software:**

El código permite jugar a “Adivinar el número”, usando un teclado, una pantalla LCD, tres LEDs, un buzzer y un sensor ultrasónico.

El jugador debe ingresar un número entre 10 y 20; el sistema indica con luces si está cerca o lejos, y al acertar pasa a una segunda etapa donde debe ubicar un objeto a una distancia específica.

## Estructura general del programa:

1. Inicio (setup):
  - Se configuran los pines de entrada y salida.
  - Se inicia la pantalla LCD.
  - Se genera un número aleatorio entre 10 y 20.
  - Se muestra el mensaje inicial del juego.
2. Bucle principal (loop):
  - Lee las teclas presionadas.
  - Permite ingresar, borrar y confirmar números.
  - Compara el número ingresado con el generado:
    - Si acierta, reproduce una melodía y se prenden los leds; Se pasa a la etapa del sensor ultrasónico.
    - Si no acierta, indica con los leds y mensajes si el número es mayor o menor. Si está a +/- 1, se prende el led azul, si está a +/- 3 el led amarillo y si está a más se prende el led rojo.
    - Si se acaban los intentos, muestra el número correcto y reinicia el juego.
3. Etapa con sensor ultrasónico:
  - El jugador debe colocar un objeto a la distancia igual al número acertado.
  - En la pantalla se muestran mensajes de “acercate” o “alejate”.
  - Cuando se logra la distancia correcta, el juego se reinicia.

## El programa está dividido en bloques funcionales:

1. Inicialización **setup()**: configuración de pines, pantalla LCD, teclado y generación del número aleatorio.
2. Lectura del teclado **getKey()** y **getKeyOnce()**: detección de teclas presionadas sin repeticiones.
3. Control de leds **apagarTodosLosLeds()**, **mostrarProximidad()**: indica cuán cerca está la respuesta.
4. Gestión de juego **siguienteIntento()**, **reiniciarJuego()**: administra los intentos y reinicia el juego.
5. Reproducción musical **reproducirCancion()**: ejecuta una melodía en caso de acierto.
6. Medición ultrasónica **medirDistancia()**: calcula la distancia en cm para la segunda fase.
7. Lógica principal **loop()**: controla la interacción general entre el jugador y el sistema.

## Diagrama de máquina de estados

Estados principales del sistema:

1. **Inicio**: muestra pantalla de bienvenida y genera un número aleatorio a adivinar.
2. **Esperando ingreso**: usuario introduce número con el teclado.

3. **Verificación:**
  - Si el número es inválido, vuelve a “esperando ingreso”.
  - Si acierta, pasa al juego con el sensor ultrasónico.
  - Si falla, muestra pistas y pasa a “nuevo intento” o “game over”.
4. **Sensor ultrasónico:** el jugador se aleja o acerca hasta que la distancia coincide con el número generado.
5. **Reinicio:** vuelve al estado de inicio.

#### **Descripción del contador de flancos y control por tiempo**

##### **Contador de flancos:**

El sensor ultrasónico (HC-SR04) funciona enviando un pulso de sonido muy corto a través del pin TRIGGER. Ese pulso viaja por el aire, rebota en algún objeto, y vuelve al sensor. El pin ECHO del sensor se pone en estado alto (HIGH) mientras el sonido está viajando de ida y vuelta.

En el programa se usa la función: **pulseIn(ECHO\_PIN, HIGH);** Esta función mide cuánto tiempo el pin **ECHO\_PIN** permanece en estado HIGH, osea, cuánto dura el pulso.

Internamente, Arduino detecta el flanco ascendente, que sería cuando la señal pasa de LOW a HIGH, y luego espera al flanco descendente, esto sería cuando vuelve a LOW. La diferencia entre esos dos momentos es el tiempo total del recorrido del sonido.

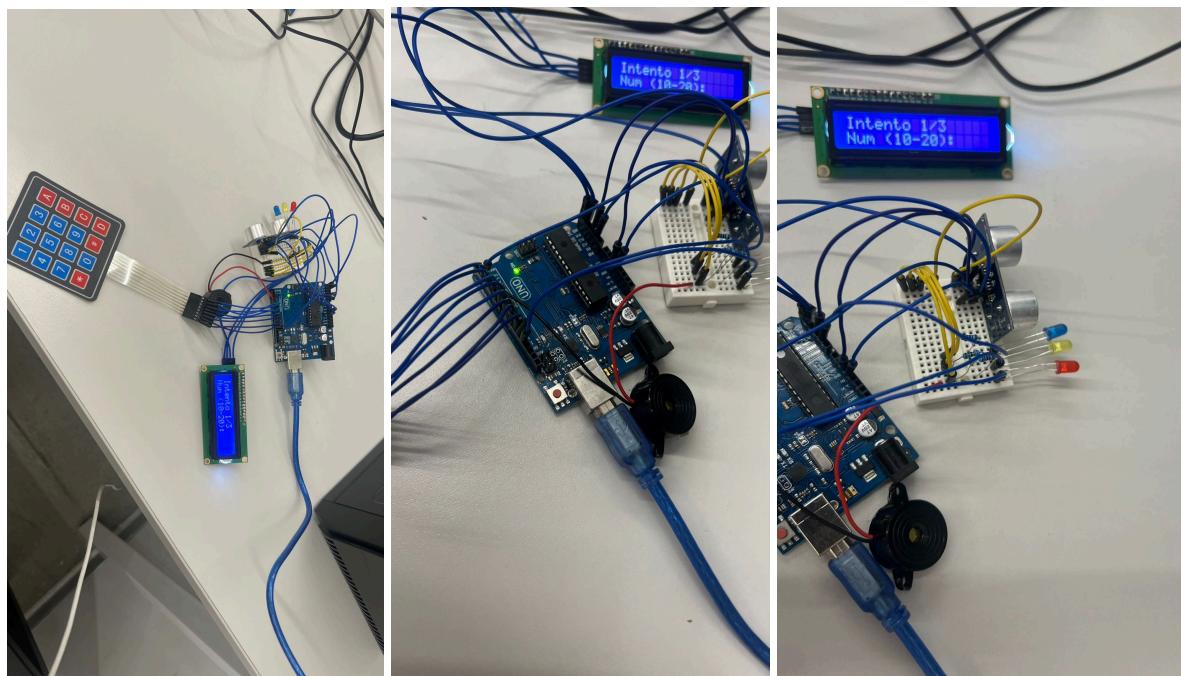
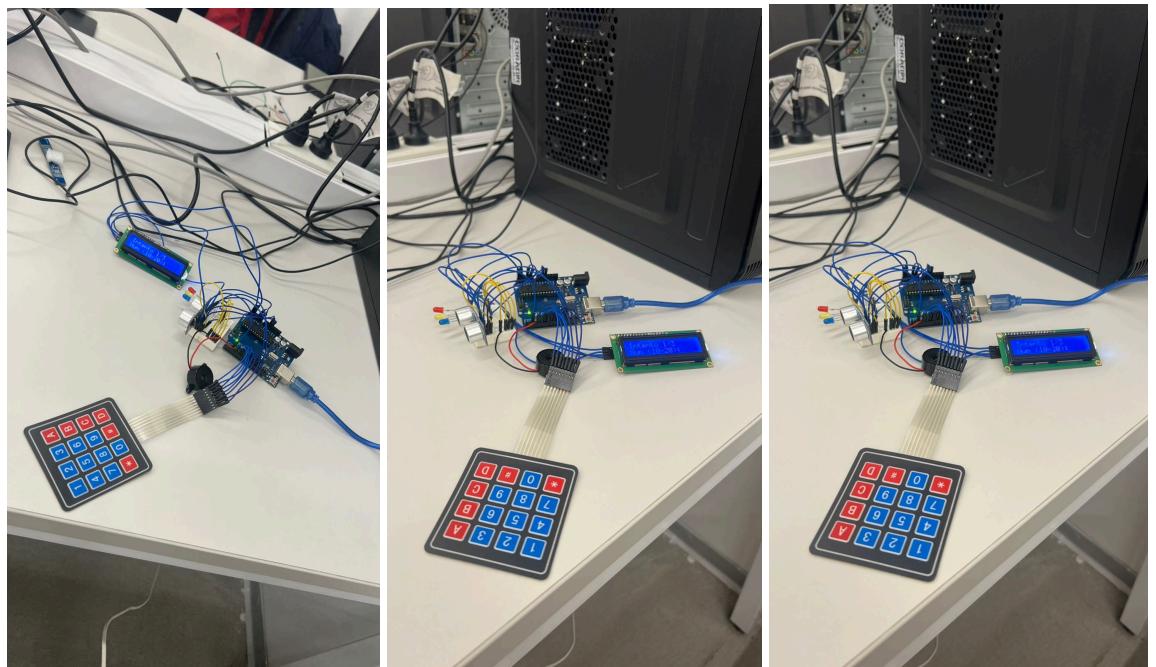
Ese tiempo que está medido en microsegundos se guarda en una variable, y después se convierte a distancia en centímetros con la fórmula: **distancia = duración \* 0.034 / 2;** donde el número 0.034 representa la velocidad del sonido y el número 2 es porque hace un viaje de ida y vuelta.

##### **Control por tiempo:**

El keypad tiene un problema típico que se llama rebote, esto sucede cuando se presiona una tecla, los contactos metálicos no hacen una conexión limpia, sino que generan una serie de microgolpes eléctricos en pocos milisegundos. Entonces el microcontrolador lee directamente ese valor e interpreta que se presionó varias veces la misma tecla, aunque en realidad solo se tocó una vez.

Para evitar eso, en nuestro programa usamos un control por tiempo dentro de la función **getKeyOnce( ).** Esta función aprovecha la variable **millis( ),** que cuenta el tiempo que lleva funcionando el Arduino, y establece un pequeño retardo de unos 120 milisegundos entre una lectura y la que le sigue. Entonces lo que hace el programa es ignorar los microgolpes y solo acepta una nueva tecla si pasó ese tiempo mínimo desde la última pulsación válida, haciendo que el sistema ahora responda de manera más estable.

**Capturas del sistema funcionando:**



### **Conclusión grupal:**

Este proyecto nos sirvió mucho para poner en práctica todo lo que fuimos aprendiendo de Arduino. Pudimos combinar varias cosas distintas: leer un teclado, usar una pantalla LCD, controlar leds y un buzzer, y además trabajar con un sensor ultrasónico. Al principio parecía complicado coordinar todo, pero una vez que encontramos las librerías con los códigos fue tomando forma. También aprendimos a manejar temas como los rebotes del teclado y los tiempos de respuesta del sensor, que son detalles que en la teoría parecen simples pero en la práctica te hacen pensar.

Nos gustó que el resultado final no sea solo un circuito que prende luces, sino un juego interactivo al que le fuimos incorporando funciones sobre la marcha, con mensajes, sonidos y una parte de distancia real. Nos dejó una buena experiencia tanto técnica como de trabajo en grupo.