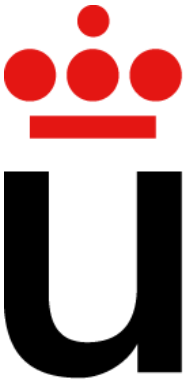


Universidad
Rey Juan Carlos

Proyecto Final:

La RuleTrago

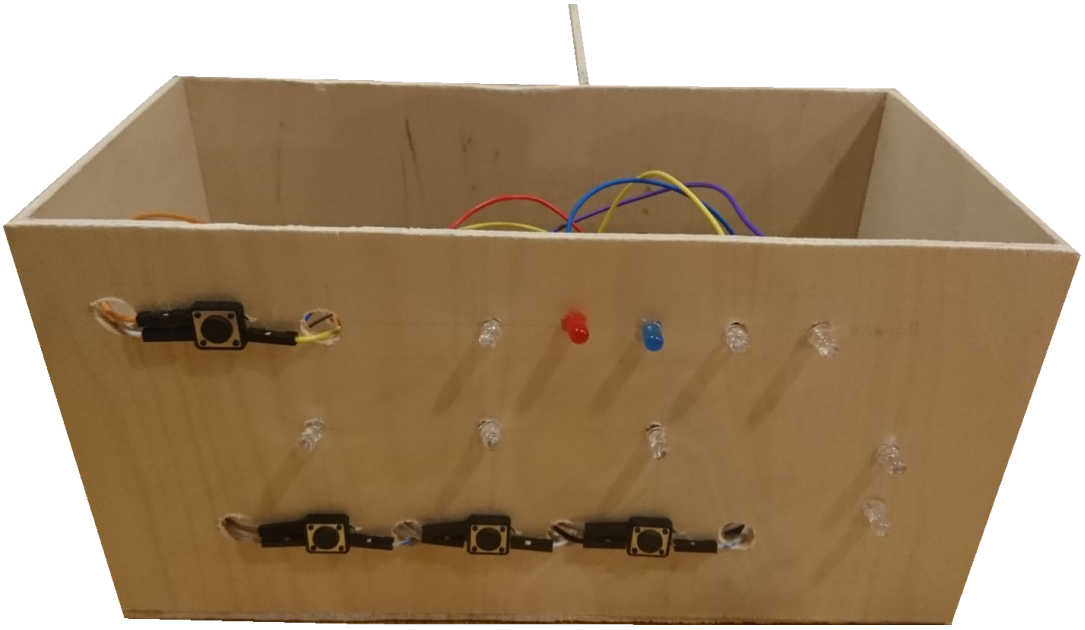
Diseño de Sistemas Empotrados

Grupo 6:
Raúl Llona Cabeza
Iker Suarez Amutxastegi
Juan de Carlos López

INDICE

Introducción

En este proyecto de Arduino se va a crear una máquina para jugar por turnos a diversos juegos en grupo (**mediante botones y LEDs**), cuyo resultado causa que se dispensen chupitos (**utilizando una bomba de agua**).



Reglas del juego

En cada turno, al jugador que le toque pulsa el botón situado en la parte superior izquierda, que activa la ruleta de LEDs de colores situada en la parte superior, la cual gira hasta detenerse aleatoriamente en uno de los 5 LEDs. Cada color indica que se debe hacer:

LED Verde: (*1 jugador*) No ocurre nada, se pasa al siguiente jugador.

LED Rojo: (*1 jugador*) La máquina dispensa un chupito que el jugador debe tomar.

LED Azul: (*1 jugador*) Se inicia una secuencia de memoria en los LEDs azules de la parte inferior de la máquina, el jugador debe memorizar la secuencia y pulsar en orden los botones a los que corresponden cada LED. Hay 2 resultados posibles:

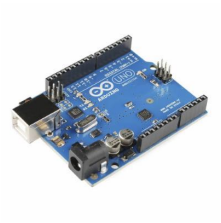
- Falla la secuencia: se enciende un LED Rojo, y la máquina dispensa un chupito.
- Acierta la secuencia: se enciende un LED Verde, y el jugador no bebe.

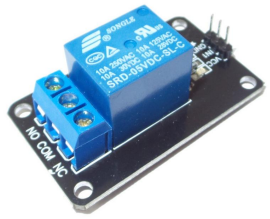
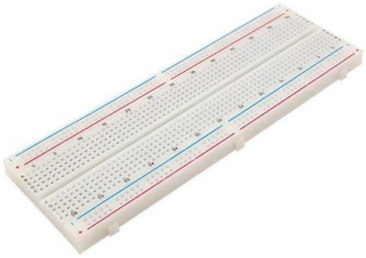
LED Amarillo: (*2 jugadores*) El jugador que ha tirado de la ruleta debe pulsar uno de los tres botones; posteriormente tiene que retar a un segundo jugador, el cual tiene que adivinar que botón ha sido presionado y pulsarlo también. Hay 2 resultados posibles:

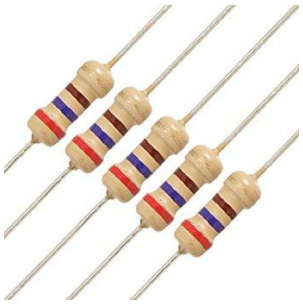
- Falla el botón: se enciende un LED Rojo junto a un LED azul para indicar que botón era el correcto; y la máquina sirve un chupito que el jugador 2 debe beber.
- Acierta el botón: se enciende un LED Verde; y la máquina sirve un chupito que el jugador 1 debe beber.

LED Morado: (*2 jugadores*) Al igual que en el caso anterior, el jugador que ha tirado de la ruleta reta a otro. Este caso es un juego de reacción, cada jugador escoge 1 botón (izquierdo o derecho) y espera a que se iluminen los LEDs azules; en cuanto esto ocurra, ambos jugadores deben tratar de pulsar su botón lo antes posible. La máquina iluminará el LED del ganador. La máquina sirve un chupito que el perdedor debe beber.

Componentes

Nombre Componente	Unidades	Precio
 *Arduino Uno*	1	Provista por la URJC
 *Bomba Sumergible*	1	3’5€
 *Pila de 9V*	1	2’5€
 *Acoplador de Pila 9V*	1	0’4€

 *Relé de 5V*	1	3’5€
 *Pulsador Arduino*	4	0’4€
 *Tabla de Prototipado*	1	2’5€
 *LEDs de Colores*	10	0’10€ - 0’50€ Depende del color

 *Cables Puente*	~40	3’10€
 *Resistencias*	10	0’15€
 *Tubo de Plástico*	1	Entre 0,3 y 0,8€
 *Planchas de Madera*	2	10€

Software

La parte software se divide en tres partes, la declaración de variables, el `setup` y el `loop`.

Declaración de variables

Primero de todo se han declarado todas las variables que han sido asociadas a pines, es decir, todas las variables relacionadas a los leds, los botones y al relé. Además, los leds y los botones se han guardado en arrays para facilitar su uso a lo largo del programa. Es importante resaltar que debido a la falta de pines digitales ha sido necesario el uso de los pines analógicos, los cuales han sido tratados como digitales.

```
//Leds Ruleta
int led1 = A0;
int led2 = A1;
int led3 = 11;
int led4 = 10;
int led5 = 5;
int leds[] = {led1, led2, led3, led4, led5};

//Leds botones
int ledB0 = 2;
int ledB1 = 3;
int ledB2 = 4;
int ledsB[] = {ledB0, ledB1, ledB2};

//Leds Acierto / Error
int ledV = 12;
int ledR = 13;

//Botones
int botonInicio = A5;
int boton0 = A4;
int boton1 = A3;
int boton2 = A2;
int boton[] = {boton0, boton1, boton2};

//Relay
int relay = 9;
```

Por otro lado, también se han declarado variables para el desarrollo de la ejecución de los diversos juegos. Para empezar, se declaran variables para guardar el estado de los botones, y después se declaran elementos para su posterior uso en los juegos específicos.

```
//Variables para guardar el estado de los botones
int botonEstadoInicio = 0;
int botonEstado0 = 0;
int botonEstado1 = 0;
int botonEstado2 = 0;
int botonEstado[] = {botonEstado0, botonEstado1, botonEstado2};

//Variables juego reto
int botonEstado0a = 0;
int botonEstado1a = 0;
int botonEstado2a = 0;

//Variables juego memoria
int turno;
int numTurnos = 5;
int orden[5];
boolean parar;

//Variables juego reaccion
int tiempoReaccion;

//Estado general
int estado = 0;
int juego = 0;

//Variables
int t;
int c;
int j;
int aux;

//Variables para controlar los tiempos de ejecución
unsigned long tiempoAnterior;
unsigned long tiempo;

//Valor aleatorio
int num;
```

Setup

A continuación, en el apartado de `setup()`, se inicializa el monitor para facilitar el testeo y la resolución de errores y se crea la semilla para generar poder generar números aleatorios correctamente. Por otra parte, se activan los pines correspondientes, aquellas que están asociadas a leds se declaran como `input`, y aquellas que se asocian a botones o al relé se declaran como `output`.

Por último, también se inicializa el pin del delay en *HIGH* debido a que está configurado para que el circuito se cierre cuando le llega una señal *LOW*.

```
void setup() {
  Serial.begin(9600);
  randomSeed(analogRead(6));
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
  pinMode(led3, OUTPUT);
  pinMode(led4, OUTPUT);
  pinMode(led5, OUTPUT);
  pinMode(ledB0, OUTPUT);
  pinMode(ledB1, OUTPUT);
  pinMode(ledB2, OUTPUT);
  pinMode(ledV, OUTPUT);
  pinMode(ledR, OUTPUT);
  pinMode(botonInicio, INPUT);
  pinMode(boton0, INPUT);
  pinMode(boton1, INPUT);
  pinMode(boton2, INPUT);
  pinMode(relay, OUTPUT);
  digitalWrite(relay, HIGH);
}
```

Loop

La parte del `loop` se divide en tres secciones principales, la espera del botón inicial (estado = 0, fuera del switch), la selección del juego (estado = 1) y cada juego (estado = 2,3,4).

Botón de inicio

Primero de todo, como en la declaración de variables la variable “estado” ha sido inicializada a 0, el programa simplemente esperará y no accederá a ningún apartado del switch. Cuando se pulsa el botón de inicio, el estado se modifica a 1.

```
botonEstadoInicio = digitalRead(botonInicio);
```

```
  if (botonEstadoInicio == HIGH) {
    estado = 1;

  }
```

La ruleta

Una vez que el estado vale 1, el programa puede acceder al apartado correspondiente de la switch, donde genera un número aleatorio entre 25 y 50. Este número indica cuanto gira la ruleta, y su módulo entre 5 indica el juego que se realiza.

```
num = random(25, 50);
```

Además, se inicializa la variable t, que indica cuanto tiempo dura el *delay* entre cada cambio de luz en la ruleta. Esto sirve para aumentar dicho tiempo en los últimos cambios. Y se apagan todos los leds, para cuando se vuelva a pulsar el botón.

```
t = 100;
Serial.println(num);
digitalWrite(leds[0], LOW);
digitalWrite(leds[1], LOW);
digitalWrite(leds[2], LOW);
digitalWrite(leds[3], LOW);
digitalWrite(leds[4], LOW);
```

Después, se realiza la parte más importante de está sección, la ruleta. Consta de un *for* que se encarga de iluminar la luz correspondiente en cada iteración y de sumar tiempo a la variable t en las últimas iteraciones.

```
for (int i = 0; i < num; i++) {
        digitalWrite(leds[i % 5], HIGH);
        if ((num - i) < 8) {
          t = t + 100;
        }
        delay(t);
        if (i != (num - 1)) {
          digitalWrite(leds[i % 5], LOW);
        }
}
```

Por último, existe un switch, que mediante el módulo entre cinco del número generado decide cuál será el juego seleccionado, y cambia el estado al número correspondiente.

```
switch ((num - 1) % 5) {
        case 0:
          //no bebe
```

```
        estado = 0;
        break;
      case 1:
        //bebe
        dispensar();
        delay(500);
        estado = 0;
        break;
      case 2:
        //juego memoria
        estado = 2;
        break;
      case 3:
        //reto a alguien
        estado = 3;
        break;
      case 4:
        //juego reacción
        estado = 4;
        break;
      default:
        break;
}
```

Los juegos

Funciones auxiliares

Primero de todo es importante mencionar que a lo largo de toda la programación de los juegos se han utilizado funciones para evitar la repetición de código. Estas funciones son las siguientes:

- La espera a que se pulse algún botón de juego. Esto se realiza mediante un bucle infinito hasta que los valores de los botones sean diferentes.

```
void esperarBotones() {
  while (digitalRead(boton0) == digitalRead(boton1) &&
digitalRead(boton0) == digitalRead(boton2)) {
  }
}
```

- Lectura de los botones, que sirve para guardar el estado de todos los botones. Los estados se guardan en las variables normales y en el array, porque se les da un uso diferente.

```
void leerBotones() {
  botonEstado0 = digitalRead(boton0);
  botonEstado1 = digitalRead(boton1);
```

```
  botonEstado2 = digitalRead(boton2);
  botonEstado[0] = digitalRead(boton0);
  botonEstado[1] = digitalRead(boton1);
  botonEstado[2] = digitalRead(boton2);
  Serial.println(botonEstado[0]);
  Serial.println(botonEstado[1]);
  Serial.println(botonEstado[2]);
  delay(500);
}
```

- Encender luz verde o roja, que sirve para indicar si algo es correcto o se ha realizado mal.

```
void luzVerde() {
  digitalWrite(ledV, HIGH);
  delay(1000);
  digitalWrite(ledV, LOW);
}
void luzRoja() {
  digitalWrite(ledR, HIGH);
  delay(1000);
  digitalWrite(ledR, LOW);
}
```

- Dispensar la bebida, que desactiva el relé, para cerrar el circuito que unen la pila y la bomba, y así poder activar está última.

```
void dispensar() {
  digitalWrite(relay, LOW);
  delay(800);
  digitalWrite(relay, HIGH);
  delay(500);
}
```

Juego de la memoria

En el caso de que el resultado del módulo sea dos, el estado se establece en dos, es decir el juego de la memoria.

Primero de todo se encienden todas las luces durante tres segundos para preparar al jugador, y se crea y se guarda la secuencia de luces, para después poder comparar con la que introducirá el jugador.

Una vez se emita toda la secuencia a realizar se esperará a que el jugador pulse algún botón, si el botón pulsado es el correcto se iluminará la luz verde y le permitirá pulsar otro botón otra vez, en el caso de que se confunda y pulse un botón que no corresponda se iluminará la luz roja y dispensará la bebida.

```
case 2:
```

```
//Juego memoria
digitalWrite(ledsB[0], HIGH);
digitalWrite(ledsB[1], HIGH);
digitalWrite(ledsB[2], HIGH);
delay(3000);
digitalWrite(ledsB[0], LOW);
digitalWrite(ledsB[1], LOW);
digitalWrite(ledsB[2], LOW);
turno = 0;
parar = false;
for (int i = 0; i < numTurnos; i++) {
  //Se genera la secuencia a memorizar
  orden[i] = random(0, 3);
  digitalWrite(ledsB[orden[i]], HIGH);
  delay(1000);
  digitalWrite(ledsB[orden[i]], LOW);
  delay(200);
}
while (turno < numTurnos && !parar) {
  Serial.println(parar);
  esperarBotones();
  leerBotones();
  j = 0;
  while (j < 3 && !parar) {
    if (botonEstado[j] == 1) {
      if (orden[turno] == j) {
        luzVerde();
      } else {
        parar = true;
      }
    }
    j++;
  }
  turno++;
}
delay (1000);
if (!parar) {
  for (int j = 0; j < 5; j++) {
    delay(200);
    digitalWrite(ledV, HIGH);
    delay (200);
    digitalWrite(ledV, LOW);
  }
} else {
  for (int j = 0; j < 5; j++) {
    delay(200);
    digitalWrite(ledR, HIGH);
```

```
        delay (200);
        digitalWrite(ledR, LOW);
      }
      dispensar();
    }
    estado = 0;
    break;
```

Juego del reto

En el caso de que el resultado del módulo sea tres, el estado se establece en tres, es decir el juego del reto.

El programa entra en un bucle infinito esperando que se pulse algún botón, cuando se pulse el valor de todos los botones y vuelve a esperar. Cuando se pulse por segunda vez si se compara el valor de los datos almacenados con los nuevos valores. Si coinciden, se ilumina el led verde, por el contrario se ilumina el led rojo. En ambos casos se dispensa la bebida.

```
    case 3:
    //Juego reto
    esperarBotones();
    leerBotones();
    esperarBotones();
    botonEstado0a = digitalRead(boton0);
    botonEstado1a = digitalRead(boton1);
    botonEstado2a = digitalRead(boton2);
    Serial.println(botonEstado0a);
    Serial.println(botonEstado1a);
    Serial.println(botonEstado2a);

    delay(500);
    for (int i = 0; i < 3; i++) {
      if (botonEstado[i] == 1) {
        aux = i;
      }
    }
       if (botonEstado[0] == botonEstado0a && botonEstado[1] ==
botonEstado1a && botonEstado[2] == botonEstado2a) {
      luzVerde();
    } else {
      luzRoja();
    }
    for (int j = 0; j < 3; j++) {
      delay(400);
      digitalWrite(ledsB[aux], HIGH);
      delay (400);
      digitalWrite(ledsB[aux], LOW);
```

```
}
delay(1000);
dispensar();
estado = 0;
break;
```

Juego de reacción

En el caso de que el resultado del módulo sea cuatro, el estado se establece en cuatro, es decir el juego de reacción.

Se encienden los leds de los botones durante cinco segundos, para preparar al jugador. Después se apagan, y se genera un valor que varía entre 2 y 12 segundos (el valor se genera aleatoriamente, 2000-12000). También se utiliza la función *millis()* para guardar el valor del tiempo actual. A continuación, la ejecución entra en un bucle en el cual permanecerá hasta que pasen los segundos correspondientes (actualizando la variable tiempo mediante *millis()*) o hasta que se pulse algún botón. Si sucede el segundo caso, se encenderá el led del botón correspondiente, el led rojo. Esto se debe a que alguien ha pulsado el botón antes de tiempo. Si el caso es el primero, y el tiempo pasa sin que nadie pulse ningún botón, se encienden los leds y el programa entra en un bucle que espera a que se pulse algún botón. El primer botón pulsado ilumina su led y enciende el led verde.

En ambos casos se dispensa la bebida.

```
case 4:
//juego reacción
parar = false;
digitalWrite(ledsB[0], HIGH);
digitalWrite(ledsB[2], HIGH);
delay(5000);
digitalWrite(ledsB[0], LOW);
digitalWrite(ledsB[2], LOW);
tiempoReaccion = (random(2000, 12000));
tiempo = millis();
tiempoAnterior = tiempo;
while (tiempo - tiempoAnterior < tiempoReaccion && !parar) {
  tiempo = millis();
  botonEstado[0] = digitalRead(boton0);
  botonEstado[2] = digitalRead(boton2);
  for (int i = 0; i < 3; i++) {
    botonEstado[i] = digitalRead(boton[i]);
    if (botonEstado[i] == 1) {
      digitalWrite(ledsB[i], HIGH);
      parar = true;
      aux = i;
    }
  }
```

```
}
if (!parar) {
  for (int i = 0; i < 3; i++) {
    digitalWrite(ledsB[i], HIGH);
  }
  esperarBotones();
  leerBotones();
  for (int i = 0; i < 3; i++) {
    digitalWrite(ledsB[i], LOW);
  }
  if (botonEstado[0] == 1) {
    digitalWrite(ledsB[0], HIGH);
    for (int j = 0; j < 5; j++) {
      delay(200);
      digitalWrite(ledV, HIGH);
      delay (200);
      digitalWrite(ledV, LOW);
    }
    delay(500);
    digitalWrite(ledsB[0], LOW);
  } else {
    digitalWrite(ledsB[2], HIGH);
    for (int j = 0; j < 5; j++) {
      delay(200);
      digitalWrite(ledV, HIGH);
      delay (200);
      digitalWrite(ledV, LOW);
    }
    delay(500);
    digitalWrite(ledsB[2], LOW);
  }
} else {
  for (int j = 0; j < 5; j++) {
    delay(200);
    digitalWrite(ledR, HIGH);
    delay (200);
    digitalWrite(ledR, LOW);
  }
  delay(500);
  digitalWrite(ledsB[aux], LOW);
}
estado = 0;
dispensar();
break;
```

Hardware

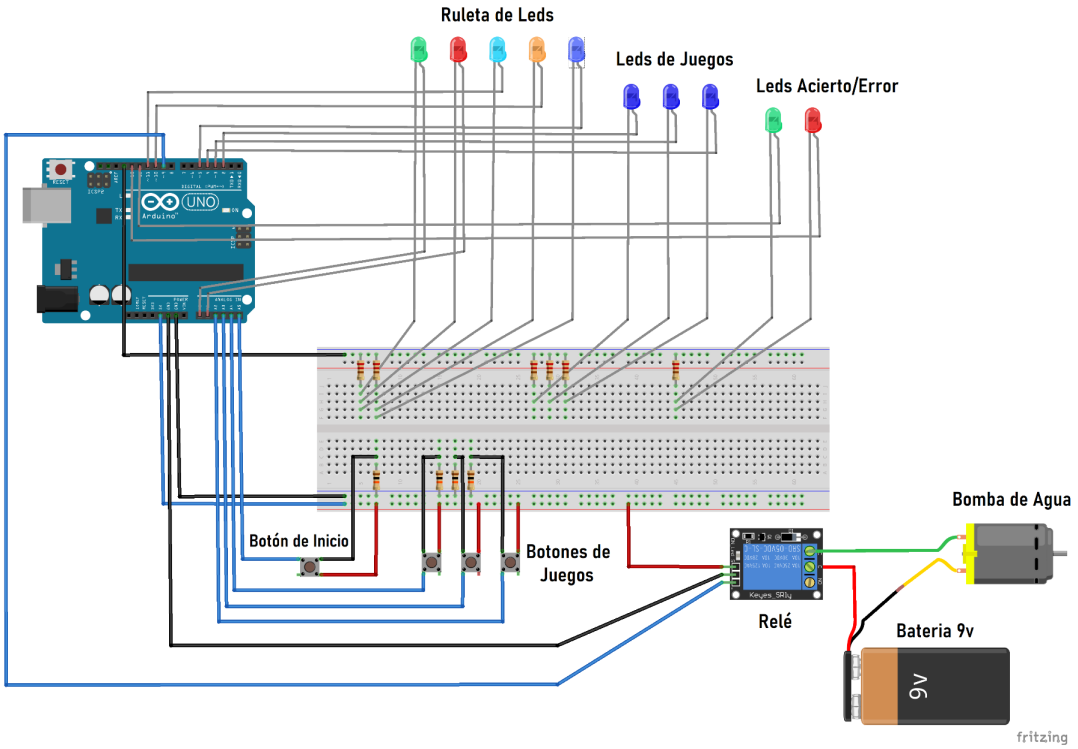
La implementación hardware del proyecto se podría dividir en tres partes: conexiones de LEDs, conexiones de botones y conexión de la bomba de agua.

- Para conectar los 10 diodos LED se han utilizado tanto pines digitales como analógicos de la placa Arduino, cerrando el circuito con la ayuda de la placa de prototipado y resistencias de 220Ω a través del pin *Ground*. Destacar que los 3 LEDs azules de los juegos se han conectado de forma separada, cada uno con una resistencia propia, ya que son los únicos que se llegan a encender simultáneamente durante el uso de la máquina.

- En cuanto a los 4 pulsadores se han utilizado pines analógicos de entrada, el pin 5v y un pin *Ground* conectados a través de la tabla de prototipado y resistencias de 10kΩ.

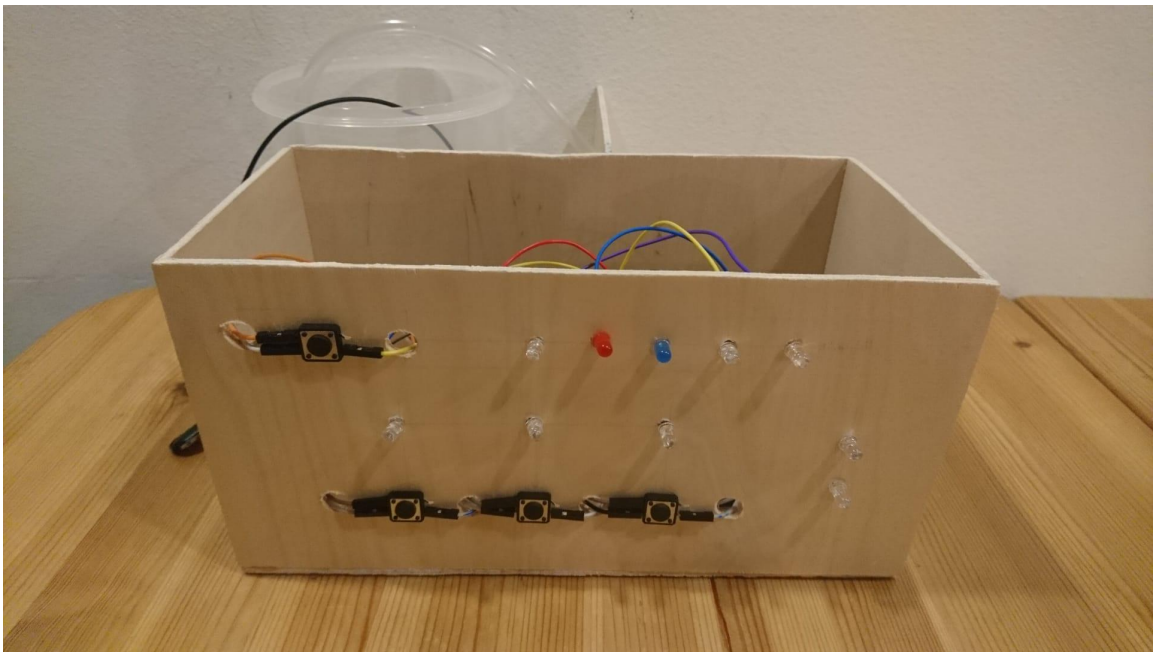
- Para lograr la correcta implementación de la bomba de agua es necesario emplear una batería de 9v y un relé. El relé actúa en el circuito como una puerta que da paso y cierra la señal a la bomba de agua. En las 3 conexiones de la parte izquierda del relé se conectan los pines 5v, un *Ground* y un pin digital. Por otro lado, atornillados en la parte derecha del relé, se conecta el cable positivo de la batería al punto de fijación C (Contacto Común) y el cable positivo de la bomba de agua al punto de fijación NC (Normalmente Cerrado). Adicionalmente, hay que empalmar ambos cables negativos de la bomba y la batería.

Esquema de Conexiones

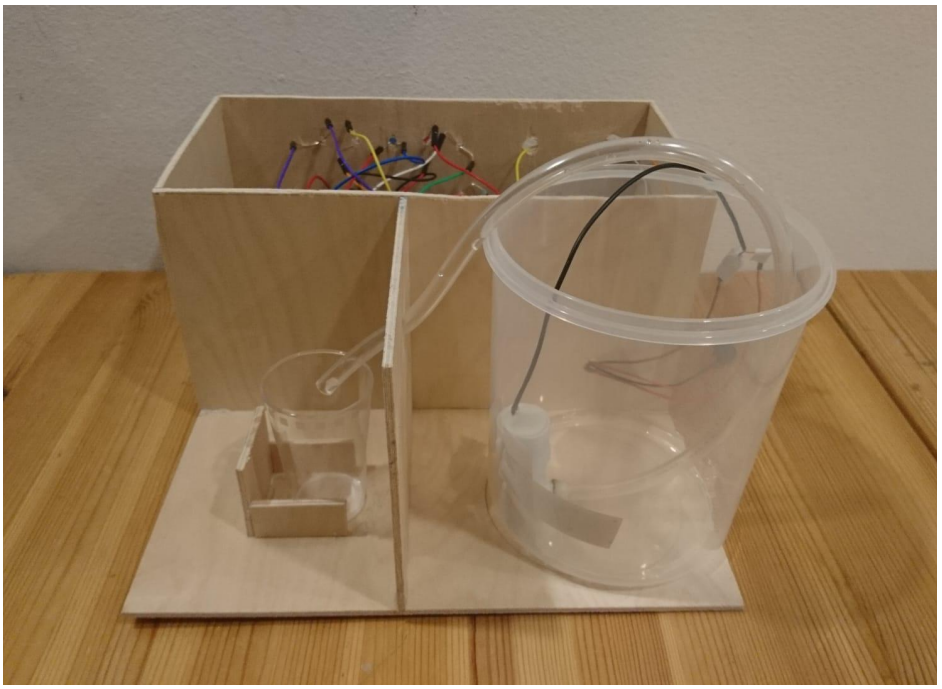


Montaje

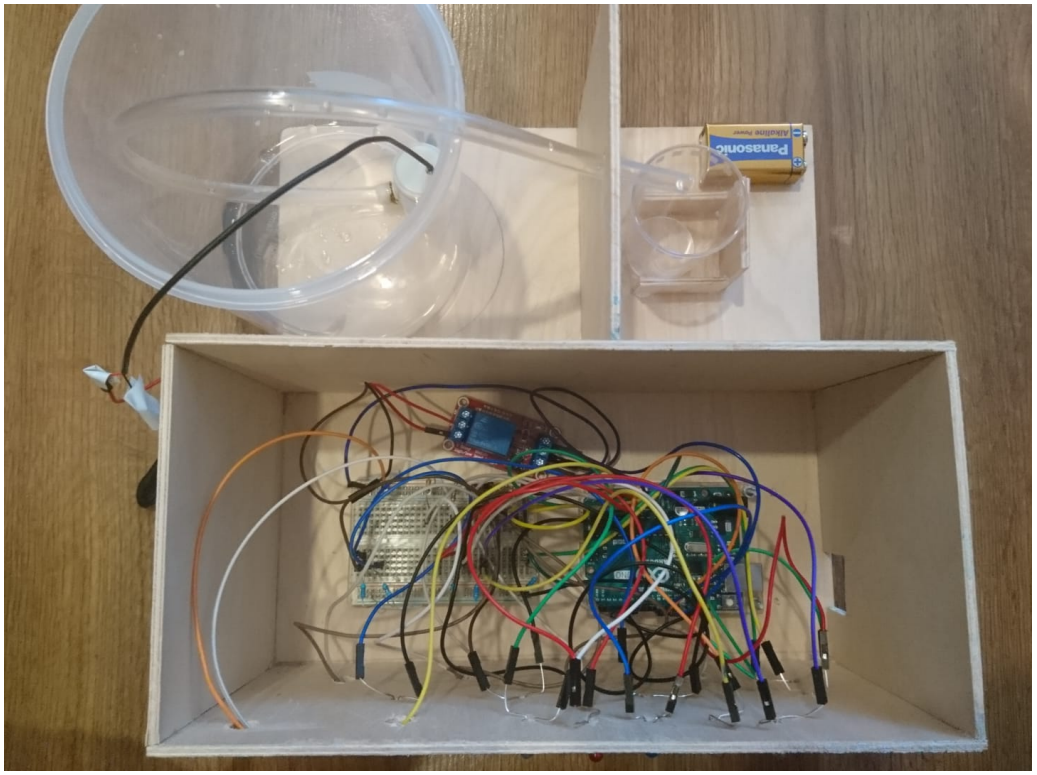
Todo el proyecto se ha construido sobre una caja de madera montada a medida para encajar todos los elementos que necesita la máquina. En la parte frontal de la caja se encuentran dispuestos todos los botones y leds para jugar:



En la parte posterior, se encuentra el recipiente de donde se extrae la bebida a servir, en dicho recipiente se encuentra la bomba que está conectada a la batería y al relé. A su vez, en la boca de la bomba está puesto el tubo de plástico por donde fluye la bebida hasta el vaso de chupito.



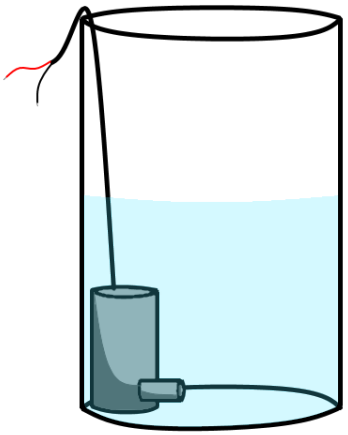
En el interior de la caja se encuentran todos los componentes electrónicos y todas las conexiones. De tal forma que quedan aislados de posibles vertimientos de la bebida.



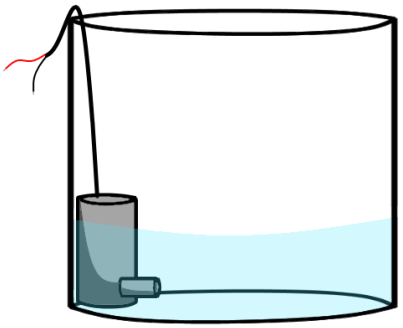
Problemas y conclusiones

Durante el desarrollo del proyecto se han encontrado diversos problemas que impedían en funcionamiento correcto de la máquina:

Se tuvo que comprar 2 bombas de agua, la primera bomba utilizada se ha estropeado a causa de que un poco de agua ha entrado en un componente sensible (la zona para evitar que no entre el agua está mal sellada).



Para evitar repetir este problema, se ha utilizado un recipiente mucho más grande para instalar la bomba; y se ha evitado que el nivel de agua del recipiente alcance la parte superior de la bomba.



Por otra parte, también ha habido problemas con el uso del relé debido a que desconocíamos cómo hacer un uso correcto del componente. Utilizándolo y haciendo múltiples pruebas al final conseguimos entender cómo funcionaba y cómo utilizarlo de manera eficaz.

Por último la gran cantidad de cables utilizados para conectar todos los elementos necesarios dificultaron el proceso de montaje debido ya que nos costaba reconocer en qué parte estaban los errores

Anexo de Código

```
//Leds Ruleta
int led1 = A0;
int led2 = A1;
int led3 = 11;
int led4 = 10;
int led5 = 5;
int leds[] = {led1, led2, led3, led4, led5};

//Leds botones
int ledB0 = 2;
int ledB1 = 3;
int ledB2 = 4;
int ledsB[] = {ledB0, ledB1, ledB2};

//Leds Acierto / Error
int ledV = 12;
int ledR = 13;

//Botones
int botonInicio = A5;
int boton0 = A4;
int boton1 = A3;
int boton2 = A2;
int boton[] = {boton0, boton1, boton2};

//Variables para guardar el estado de los botones
int botonEstadoInicio = 0;
int botonEstado0 = 0;
int botonEstado1 = 0;
int botonEstado2 = 0;
int botonEstado[] = {botonEstado0, botonEstado1, botonEstado2};

//Variables juego reto
int botonEstado0a = 0;
int botonEstado1a = 0;
int botonEstado2a = 0;

//Variables juego memoria
int turno;
int numTurnos = 5;
int orden[5];
boolean parar;

//Variables juego reaccion
int tiempoReaccion;
```

```
//Estado general
int estado = 0;
int juego = 0;

//Variables
int t;
int c;
int j;
int aux;

//Variables para controlar los tiempos de ejecución
unsigned long tiempoAnterior;
unsigned long tiempo;

//Valor aleatorio
int num;

//Relay
int relay = 9;

void setup() {
  Serial.begin(9600);
  randomSeed(analogRead(6));
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
  pinMode(led3, OUTPUT);
  pinMode(led4, OUTPUT);
  pinMode(led5, OUTPUT);
  pinMode(ledB0, OUTPUT);
  pinMode(ledB1, OUTPUT);
  pinMode(ledB2, OUTPUT);
  pinMode(ledV, OUTPUT);
  pinMode(ledR, OUTPUT);
  pinMode(botonInicio, INPUT);
  pinMode(boton0, INPUT);
  pinMode(boton1, INPUT);
  pinMode(boton2, INPUT);
  pinMode(relay, OUTPUT);
  digitalWrite(relay, HIGH);
}

void loop() {

  botonEstadoInicio = digitalRead(botonInicio);
  if (botonEstadoInicio == HIGH) {
    estado = 1;
```

```
}
switch (estado) {
  case 1:
    num = random(25, 50);
    t = 100;
    Serial.println(num);
    digitalWrite(leds[0], LOW);
    digitalWrite(leds[1], LOW);
    digitalWrite(leds[2], LOW);
    digitalWrite(leds[3], LOW);
    digitalWrite(leds[4], LOW);
    for (int i = 0; i < num; i++) {
      digitalWrite(leds[i % 5], HIGH);
      if ((num - i) < 8) {
        t = t + 100;
      }
      delay(t);
      if (i != (num - 1)) {
        digitalWrite(leds[i % 5], LOW);
      }
    }
    switch ((num - 1) % 5) {
      case 0:
        //no bebe
        estado = 0;
        break;
      case 1:
        //bebe
        dispensar();
        delay(500);
        estado = 0;
        break;
      case 2:
        //juego memoria
        estado = 2;
        break;
      case 3:
        //reto a alguien
        estado = 3;
        break;
      case 4:
        //juego reacción
        estado = 4;
        break;
      default:
        break;
```

```
      }
      break;
    case 3:
      //Juego reto
      esperarBotones();
      leerBotones();
      esperarBotones();
      botonEstado0a = digitalRead(boton0);
      botonEstado1a = digitalRead(boton1);
      botonEstado2a = digitalRead(boton2);
      Serial.println(botonEstado0a);
      Serial.println(botonEstado1a);
      Serial.println(botonEstado2a);

      delay(500);
      for (int i = 0; i < 3; i++) {
        if (botonEstado[i] == 1) {
          aux = i;
        }
      }
      if (botonEstado[0] == botonEstado0a && botonEstado[1] ==
botonEstado1a && botonEstado[2] == botonEstado2a) {
        luzVerde();
      } else {
        luzRoja();
      }
      for (int j = 0; j < 3; j++) {
        delay(400);
        digitalWrite(ledsB[aux], HIGH);
        delay (400);
        digitalWrite(ledsB[aux], LOW);

      }
      delay(1000);
      dispensar();
      estado = 0;
      break;
    case 2:
      //Juego memoria
      digitalWrite(ledsB[0], HIGH);
      digitalWrite(ledsB[1], HIGH);
      digitalWrite(ledsB[2], HIGH);
      delay(3000);
      digitalWrite(ledsB[0], LOW);
      digitalWrite(ledsB[1], LOW);
      digitalWrite(ledsB[2], LOW);
      turno = 0;
```

```
  parar = false;
  for (int i = 0; i < numTurnos; i++) {
    //Se genera la secuencia a memorizar
    orden[i] = random(0, 3);
    digitalWrite(ledsB[orden[i]], HIGH);
    delay(1000);
    digitalWrite(ledsB[orden[i]], LOW);
    delay(200);
  }
  while (turno < numTurnos && !parar) {
    Serial.println(parar);
    esperarBotones();
    leerBotones();
    j = 0;
    while (j < 3 && !parar) {
      if (botonEstado[j] == 1) {
        if (orden[turno] == j) {
          luzVerde();
        } else {
          parar = true;
        }
      }
      j++;
    }
    turno++;
  }
  delay (1000);
  if (!parar) {
    for (int j = 0; j < 5; j++) {
      delay(200);
      digitalWrite(ledV, HIGH);
      delay (200);
      digitalWrite(ledV, LOW);
    }
  } else {
    for (int j = 0; j < 5; j++) {
      delay(200);
      digitalWrite(ledR, HIGH);
      delay (200);
      digitalWrite(ledR, LOW);
    }
    dispensar();
  }
  estado = 0;
  break;
case 4:
  //juego reacción
```

```
parar = false;
digitalWrite(ledsB[0], HIGH);
digitalWrite(ledsB[2], HIGH);
delay(5000);
digitalWrite(ledsB[0], LOW);
digitalWrite(ledsB[2], LOW);
tiempoReaccion = (random(2000, 12000));
tiempo = millis();
tiempoAnterior = tiempo;
while (tiempo - tiempoAnterior < tiempoReaccion && !parar) {
  tiempo = millis();
  botonEstado[0] = digitalRead(boton0);
  botonEstado[2] = digitalRead(boton2);
  for (int i = 0; i < 3; i++) {
    botonEstado[i] = digitalRead(boton[i]);
    if (botonEstado[i] == 1) {
      digitalWrite(ledsB[i], HIGH);
      parar = true;
      aux = i;
    }
  }
}
if (!parar) {
  for (int i = 0; i < 3; i++) {
    digitalWrite(ledsB[i], HIGH);
  }
  esperarBotones();
  leerBotones();
  for (int i = 0; i < 3; i++) {
    digitalWrite(ledsB[i], LOW);
  }
  if (botonEstado[0] == 1) {
    digitalWrite(ledsB[0], HIGH);
    for (int j = 0; j < 5; j++) {
      delay(200);
      digitalWrite(ledV, HIGH);
      delay (200);
      digitalWrite(ledV, LOW);
    }
    delay(500);
    digitalWrite(ledsB[0], LOW);
  } else {
    digitalWrite(ledsB[2], HIGH);
    for (int j = 0; j < 5; j++) {
      delay(200);
      digitalWrite(ledV, HIGH);
      delay (200);
```

```
            digitalWrite(ledV, LOW);
          }
          delay(500);
          digitalWrite(ledsB[2], LOW);
        }
      } else {
        for (int j = 0; j < 5; j++) {
          delay(200);
          digitalWrite(ledR, HIGH);
          delay (200);
          digitalWrite(ledR, LOW);
        }
        delay(500);
        digitalWrite(ledsB[aux], LOW);
      }
      estado = 0;
      dispensar();
      break;
    default:
      break;
  }
}

void esperarBotones() {
  while (digitalRead(boton0) == digitalRead(boton1) &&
digitalRead(boton0) == digitalRead(boton2)) {
  }
}

void leerBotones() {
  botonEstado0 = digitalRead(boton0);
  botonEstado1 = digitalRead(boton1);
  botonEstado2 = digitalRead(boton2);
  botonEstado[0] = digitalRead(boton0);
  botonEstado[1] = digitalRead(boton1);
  botonEstado[2] = digitalRead(boton2);
  Serial.println(botonEstado[0]);
  Serial.println(botonEstado[1]);
  Serial.println(botonEstado[2]);
  delay(500);
}

void luzVerde() {
  digitalWrite(ledV, HIGH);
  delay(1000);
  digitalWrite(ledV, LOW);
}
```

```
void luzRoja() {
  digitalWrite(ledR, HIGH);
  delay(1000);
  digitalWrite(ledR, LOW);
}

void dispensar() {
  digitalWrite(relay, LOW);
  delay(800);
  digitalWrite(relay, HIGH);
  delay(500);
}
```