Proyecto: Cart-O-Matic

dispensador automático de cartas

Memoria

Grupo 6

Jorge	Diego	Alberto
Adame Prudencio	**Guerrero Carrasco**	**Pérez Pérez**

viernes, 13 de **mayo de 2022**

Sistemas Empotrados y de Tiempo Real

Grado en Ingeniería Informática

Índice de **contenidos**

Introducción 3
Diseño y ejecución del proyecto: cronología 4
Presupuesto 5
Problemas enfrentados y resueltos durante el proyecto 7
Producto final 9
Hardware: esquema de componentes electrónicos 9
Software desarrollado 10
Casos de uso 14
Vídeo resumen del proyecto 14
Recursos asociados y contacto 14

Índice de **imágenes**

Imagen 1. Visión general del proyecto. 3
Imagen 2. Vista aérea del proyecto (desacoplando las bases). 3
Imagen 3. Controladora L298N con *shield*. 8
Imagen 4. Esquema de componentes electrónicos. 9
Imagen 5. Aspecto físico de la electrónica del proyecto. 9
Imagen 6. Información sobre vías de divulgación acerca del proyecto. 14

Introducción

Cart-O-Matic es un dispensador automático de cartas. Este producto permite realizar, con una interacción rápida y sencilla, realizar un reparto de tantas cartas como se desee a una serie de jugadores dispuestos en posición de juego.

El proyecto ha sido realizado por el grupo 6 de Sistemas Empotrados y de Tiempo Real en el Grado en Ingeniería Informática (Campus de Móstoles, curso 2021-22), compuesto por tres alumnos del Doble Grado en Ingeniería Informática e Ingeniería del *Software*:

- Jorge Adame Prudencio (j.adame.2018@alumnos.urjc.es).
- Diego Guerrero Carrasco (d.guerrero.2018@alumnos.urjc.es).
- Alberto Pérez Pérez (a.perezpe.2018@alumnos.urjc.es).

El presente artículo expone toda la información relativa al proyecto desde su concepción hasta su exposición en el aula en formato cronológico. Como consecuencia, se introduce en primer lugar el diseño y ejecución del proyecto en formato cronológico, introduciendo cuestiones como el presupuesto dedicado o la descripción de los problemas aparecidos y resueltos durante la ejecución del proceso; concluyendo con la exhibición del proyecto en su estado final, dividiéndolo en tres partes esenciales: *hardware*, *software* y casos de uso reales. Una vez finalizada, se introduce un vídeo explicativo del proyecto completo y una descripción de los recursos adicionales dispuestos.

El resultado final del proyecto es:

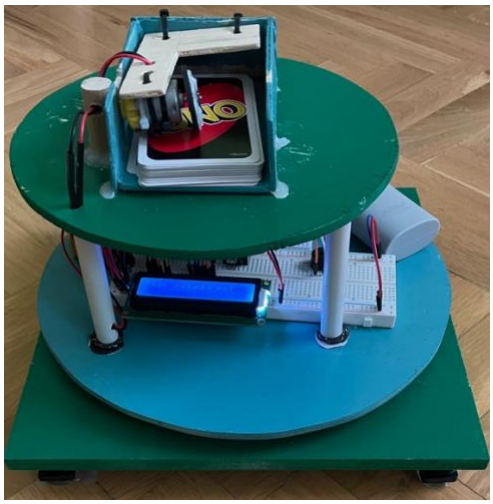

Imagen 1. Visión general del proyecto.

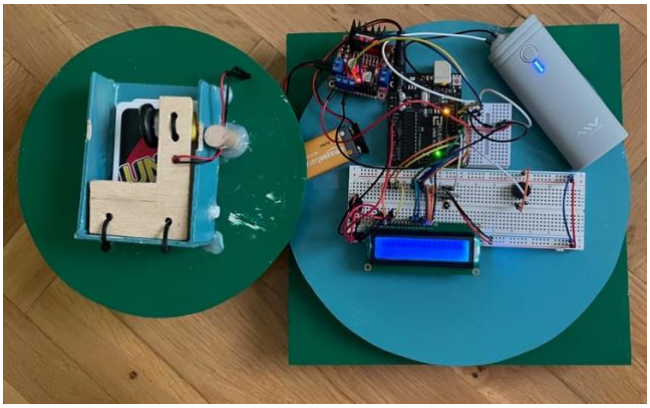

Imagen 2. Vista aérea del proyecto (desacoplando las bases).

El presente documento es una memoria que extrae el contenido introducido en la entrada publicada en el *blog* oficial de la asignatura. Se puede acceder haciendo *click* en este enlace o accediendo a través del vínculo: bit.ly/SETR-G6-Post.

Diseño y ejecución del proyecto: cronología

El proceso de diseño y desarrollo del proyecto se ha ejecutado fundamentalmente durante las horas lectivas dedicadas al proyecto en el aula, introduciendo algunas sesiones adicionales de trabajo fuera del aula para realizar los procesos más difícilmente realizables en el aula —como, por ejemplo, todas las tareas relacionadas con el corte de materiales—. La cronología asociada al mencionado proceso queda resumida en los siguientes apartados:

- Durante las tres primeras sesiones en el aula, el equipo procedió a iniciar la planificación temporal y a realizar un primer diseño del objeto físico, tomando las primeras decisiones y analizando todas las posibles opciones en materia de diseño. Se realizaron diversos prototipos de baja fidelidad para reunir las mejores ideas de cada integrante y así decidir de forma común cuál sería el aspecto y la funcionalidad completa del dispositivo antes de proceder a la compra de materiales físicos y al inicio de la implementación *software* de su funcionalidad.
- En este punto se tomaron diversas decisiones de diseño relativas al *hardware* —acerca de cuestiones como, por ejemplo, los materiales empleados, las dimensiones de las superficies empleadas en el proyecto, el aspecto final del mismo y cómo dividir el trabajo y planificarlo temporalmente de forma adecuada a las necesidades del proyecto y de cada uno de los integrantes— y *software*, valorando cuestiones sobre la división del trabajo en bloques independientes y la planificación temporal del proceso.
- La cuarta sesión se dedicó a hacer la compra conjuntamente de los componentes básicos acordados que se utilizarían posteriormente para la construcción y el ensamblado del dispositivo.
- Una vez comprados prácticamente todos los materiales empleados para construir el sistema, se procedió a la división de tareas a realizar durante el periodo vacacional de Semana Santa:
    - En materia de *hardware*, cada uno de los integrantes se ocupó de una tarea, siendo las más complejas de realizar y las que debían realizarse fuera de la Universidad las siguientes tres: la confección manual de un recipiente para albergar las cartas a repartir, el corte y preparación de las superficies de madera y los soportes de plástico; y el pintado y acabado de las bases empleadas para una mejor apariencia. Se aprovechó la coincidencia de la realización del proyecto con las vacaciones de Semana Santa para realizar estas tareas fuera de la universidad, de modo que todos los materiales estarían preparados para su ensamblaje al finalizar este periodo.
    - Por otro lado, en paralelo, se procedió a la implementación del *software* que se empleó para «dar vida» al proyecto. Para ello, se dispuso un proceso de desarrollo iterativo e incremental en el que cada integrante se ocupó de la implementación de cada una de las partes del ciclo de utilización del producto: la interacción inicial con el usuario —esto es, la interacción con las librerías externas empleadas para el manejo de la pantalla LCD y el mando de control por infrarrojos—, la validación de los datos introducidos y la realización de los cálculos pertinentes para determinar los giros adecuados de los motores introducidos y la implementación del algoritmo

general para la ejecución ordenada y por pasos de los distintos componentes *software* dispuestos en el proceso.

- Una vez finalizada la fase principal de implementación de los componentes *hardware* y *software* del proyecto, se procedió a ensamblar las distintas partes para la integración del producto final. Con este fin, se dispusieron tres sesiones presenciales de larga duración en las que se procedió a la fusión de los componentes *software* implementados y a la creación del algoritmo final que permitiría ejecutar toda la funcionalidad del proyecto y coordinar los distintos componentes *hardware*.
- Por otro lado, estas sesiones también se emplearon para el ensamblaje de todas las partes que conformaban el sistema físico, empleando una sala de trabajo grupal de la Universidad para proceder a unir todas las piezas preparadas y a preparar la circuitería electrónica que integra el proyecto resolviendo los últimos problemas acontecidos y finalizando con éxito el montaje del proyecto.
- La última de las sesiones de ensamblaje del proyecto se destinó, además, a la preparación de la exposición del producto elaborado, dividiéndola en tres bloques esenciales: *hardware* —incluyendo una visión superficial de las componentes técnica-electrónica y de los materiales y su disposición en el producto final—, *software* —realizando un repaso por todo el código implementado y aprovechando su explicación en orden secuencial para revisar simultáneamente el funcionamiento del proyecto— y los casos de uso, introduciendo algunas grabaciones realizadas sobre el funcionamiento del producto.

Presupuesto

El presupuesto dispuesto se ha dividido en dos categorías: componentes electrónicos y componentes no electrónicos. Se introducen a continuación los totales de ambas categorías y el total conjunto. A continuación, se sitúan dos tablas que detallan los materiales introducidos en cada una de las categorías dispuestas, especificando el importe de todos los productos (salvo aquellos proporcionados por la Universidad).

Tabla 1. Presupuesto final: suma de totales.

TOTAL	**51,16 €**
Componentes electrónicos adquiridos	23,71 €
Componentes no electróncios adquiridos	27,45 €

Tabla 2. Presupuesto para componentes electrónicos.

Cantidad	Producto	Procedencia	Importe
1	Arduino UNO R3	Caja de componentes	–
1	Placa de prototipado	Caja de componentes	–
1	Placa de prototipado miniatura	Caja de componentes	–
1	Pantalla LCD1602	Caja de componentes	–
1	Motor tipo Microservo de rotación continua	Compra aparte	9,62 €

1	Motor estándar de corriente continua	Caja de componentes	–
1	Controlador de motores de puente H L298N	Compra aparte	7,10 €
1	Receptor de señales infrarrojas	Caja de componentes	–
1	Mando de control remoto (IR)	Caja de componentes	–
1	Resistencia 220Ω	Caja de componentes	–
1	Potenciómetro	Caja de componentes	
1	Conjunto de cables conectores tipo jumper	Caja de componentes	–
1	Emisor de sonido piezoeléctrico	Caja de componentes	–
1	Fuente de potencia 5V 2,1A (PowerBank)	Compra aparte	6,99 €
	TOTAL		**23,71 €**

Tabla 3. Presupuesto para componentes no electrónicos.

Cantidad	Producto	Importe
1	Varilla de madera lisa	2,59 €
1	Tablón de contrachapado 60x30x10 mm	5,19 €
1	Tablón de contrachapado 60x30x5 mm	3,69 €
1	Tubo redondo de PVC 12 mm (1 m.)	1,32 €
1	Varilla cilíndrica de aluminio 4 mm (1 m.)	0,92 €
1	Papel de lija para madera	0,40 €
4	Conteras para embutir 20×20 mm	0,93 €
4	Muelles de compresión 0,8×80 mm	1,77 €
1	Paquete de tornillos y tuercas de zinc ø4 mm.	1,95 €
4	Gotas adhesivas de silicona transparente antideslizamiento	2,99 €
1	Paquete de gomas pequeñas antideslizantes	0,60 €
2	Pinturas en spray para madera	3,90 €
1	Caja de cartón para desplazamientos	1,20 €
	TOTAL	**27,45 €**

Problemas enfrentados y resueltos durante el proyecto

Los principales problemas aparecidos durante el proyecto se han concentrado en la fase de ensamblaje y montaje del producto físico. Los más destacados han sido:

- En las primeras fases del diseño, a la hora de tomar la decisión sobre qué motores emplear para la base rotatoria de la parte inferior del proyecto —el encargado de la rotación del sistema completo—, se dispusieron varias opciones. La caja de componentes incluía un Microservo de 180 grados de rotación que, si bien se ajustaba en dimensiones al tamaño ideado para este motor y permitía controlar con bastante precisión el ángulo en el que se debía colocar el dispositivo, no permitía la rotación completa, hecho por el que se descartó esta idea.

  Por otro lado, la caja también incluía un motor de pasos o stepper, así como la controladora ULN2003A necesaria para manejar el tiempo y velocidad de giro de estos motores. Se realizaron varias pruebas reales con este motor, experimentando a variar su velocidad e introduciendo algunas funciones para poder calcular el ángulo de giro, pero se terminó descartando por su gran tamaño —ya que es notablemente más grande que el Microservo— y por su lógica de manejo compleja en código, que añadía una complejidad adicional a un código que ya era complejo por la gran cantidad de componentes que orquestaba.

  Como consecuencia, se tomó como decisión final la adquisición y utilización de un motor de tipo Microservo de rotación continua, motor que aúna los requisitos de tamaño del Microservo y la capacidad de girar de forma continua en ambos sentidos.
- Uno de los principales problemas para realizar este trabajo ha sido la ausencia de un lugar adecuado para trabajar con los materiales empleados. Para poder trabajar con maderas y plásticos de cierto grosor, ha sido necesario buscar medios y lugares físicos adecuados para ello fuera de la Universidad, ya que en esta no se ha facilitado acceso a ningún taller ni lugar habilitado a tal efecto. Además, ninguno de nosotros disponíamos de medios para el corte de la madera en casa, aunque sí se disponía de un par de pistolas termoencoladoras que nos han permitido realizar todas las uniones entre materiales necesarias.
- En materia de electrónica, el principal problema apareció cuando, una vez ensamblados los motores empleados en la práctica, se descubrió que no eran capaces de realizar los movimientos necesarios. La lectura de las datasheet de ambos motores nos permitió descubrir que la corriente enviada por la placa Arduino —de 5V y 40 mA— no era suficiente, sino que era necesario emplear una corriente de mayor intensidad. Como consecuencia, se procedió a comprar una fuente de energía externa que proporciona 5V y 2,1A, de modo que llegarían 1,05A a cada uno de los motores dispuestos en paralelo. Esta corriente resultó ser suficiente para efectuar con éxito los movimientos de ambos motores, logrando que el dispositivo rotase sobre su eje y que el motor de la parte superior disparase cartas de forma adecuada.

  Sin embargo, cuando se quiso implementar el giro hacia atrás del motor de la parte superior —para evitar expulsar más de una carta en cada disparo—, se descubrió que el controlador de puente H de tipo L293D incluido en la caja de componentes no era adecuado, ya que la intensidad máxima soportada por este componente ronda los 600 mA

(según fabricante). Como consecuencia, fue necesario acudir a comprar un componente de igual funcionalidad y especificaciones superiores, adquiriendo el controlador L298N con interfaz para conexiones de cables (tal como se ve en la imagen adyacente).

Imagen 3. Controladora L298N con *shield*.

Producto final

En este punto se muestra el producto integral obtenido tras la ejecución del proceso de desarrollo anteriormente descrito en tres subsecciones: *hardware*, *software* y casos de uso.

Hardware: esquema de componentes electrónicos

El esquema que muestra los componentes empleados y las interconexiones establecidas entre ellos es el siguiente:

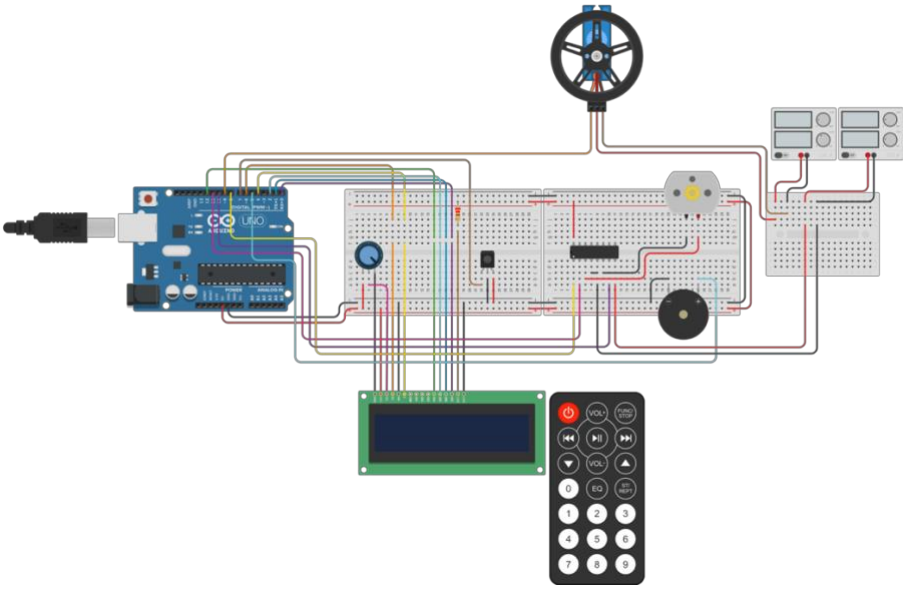

Imagen 4. Esquema de componentes electrónicos.

Los componentes empleados son: placa Arduino UNO, potenciómetro, pantalla LCD, sensor de infrarrojos y mando emisor de señales infrarrojas, *controladora de puente H tipo L298N*, motor de corriente continua, zumbador piezoeléctrico, *motor tipo Servo de rotación continua*, *fuente de potencia de 5V 2,1A* y resistencia de 220Ω. Los únicos componentes externos a la caja proporcionada por la Universidad son aquellos que se destacan en tipografía cursiva.

El aspecto físico del sistema en la realidad queda de la siguiente forma:

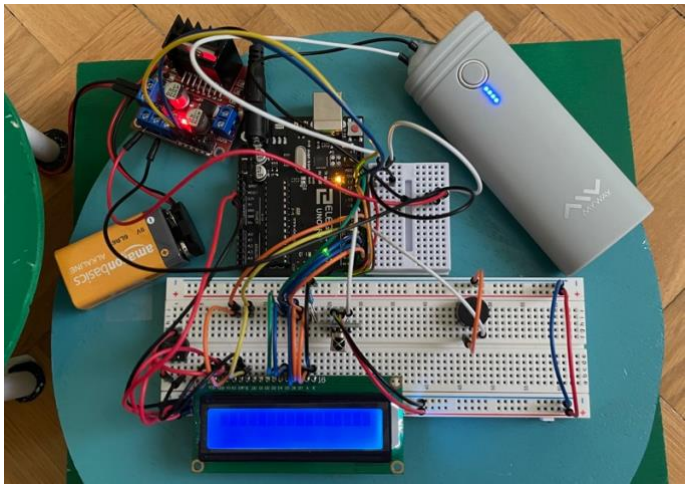

Imagen 5. Aspecto físico de la electrónica del proyecto.

Se observan en la fotografía anterior, de forma destacada, el controlador de puente H tipo L298N (superior izquierda), la fuente de energía externa de 5V 2,1A (superior derecha), la placa de prototipado por la que circula la corriente del Arduino (centro), la placa de prototipado miniatura empleada para hacer circular la energía de la fuente externa (superior derecha), la placa Arduino UNO (superior central), la pila de 9V para alimentar el Arduino (izquierda) y algunos componentes como el LCD (inferior), el receptor de señales infrarrojas (centro) o el emisor de sonidos tipo Piezo (derecha).

Software desarrollado

Por otro lado, se introduce a continuación el código final empleado para orquestar el funcionamiento de los componentes *hardware* anteriormente descritos. Se han introducido comentarios adicionales para facilitar su comprensión:

```
// Librerías
#include <LiquidCrystal.h>
#include <IRremote.hpp>
#include <Servo.h>

// Convenio de pines
// Sensor infrarrojos
int infrarrojo = 7;

// Pantalla
int db7 = 0;
int db6 = 1;
int db5 = 2;
int db4 = 12;
int screenEnable = 4;
int screenRS = 6;
LiquidCrystal lcd(screenRS, screenEnable, db4, db5, db6, db7);

// Motor de abajo
int motorMicroservo = 9;
Servo microservo;

// Motor arriba
int rodilloEnable = 8;
int rodilloDirA = 10;
int rodilloDirB = 11;
// Piezo
int piezo = 5;

// Variables globales
// Nº de jugadores
int numJugadores = 0;
// Nº de cartas disponibles
int numCartas = 0;
// Cuántas a cada uno
int cartasCadaUno = 0;
// Cuando el número de personas no es compatible con 360, es necesario saber a
// cuántas personas hay que repartir con más ángulo
int personasMasAngulo = 0;
// Número de cartas sobrantes tras realizar el reparto (si sobran)
int cartasSobrantes = 0;

// Valores prefijados para giro de microservo
int frecuenciaMicroservoRpm = 15;
int velocidadSentidoCW = 30;
int velocidadSentidoCCW = 150;


// 1 -> Interfaz de gestión de la pantalla LCD
void imprimirYPosicionarLCD(String texto,
                            int posCursorColumnaInicial,
                            int posCursorFilaInicial,
                            int posCursorColumnaFinal,
                            int posCursorFilaFinal) {
    lcd.setCursor(posCursorColumnaInicial, posCursorFilaInicial);
    lcd.print(texto);
    lcd.setCursor(posCursorColumnaFinal, posCursorFilaFinal);
    lcd.cursor();
}

void inicializarLCD() {
    lcd.begin(16, 2);
    lcd.print("Bienvenido...");
    lcd.setCursor(0, 1);
    for (int posicionCursor = 0; posicionCursor < 16; posicionCursor++) {
        lcd.print("*");
        delay(50);
    }
```

```
    delay(500);
    lcd.clear();
}

int escribirNumeros(int valorMaximo, String rotulo, bool obligatorio) {
    int valorNumeroFinal = 0;
    int posColumnaCursor = 0;
    bool pulsacionPlay = false;
    imprimirYPosicionarLCD(rotulo, 0, 0, 0, 1);
    IrReceiver.begin(infrarrojo, DISABLE_LED_FEEDBACK);
    while (!pulsacionPlay) {
        if (IrReceiver.decode()) {
            int ultimoValorIntroducido = decodificarNumeros((unsigned long) IrReceiver.decodedIRData.decodedRawData);
            if (ultimoValorIntroducido != -3) {
                if (ultimoValorIntroducido != -1 && ultimoValorIntroducido != -2) {
                    if (!(ultimoValorIntroducido == 0 && posColumnaCursor == 0)) {
                        int nuevoValor = valorNumeroFinal * 10 + ultimoValorIntroducido;
                        if (nuevoValor <= valorMaximo) {
                            valorNumeroFinal = valorNumeroFinal * 10 + ultimoValorIntroducido;
                            posColumnaCursor = posColumnaCursor + 1;
                            imprimirYPosicionarLCD((String) valorNumeroFinal, 0, 1, posColumnaCursor, 1);
                        }
                    }
                    IrReceiver.resume();
                } else if (ultimoValorIntroducido != -2) {
                    if (obligatorio) {
                        if (valorNumeroFinal != 0) {
                            pulsacionPlay = true;
                            lcd.clear();
                            return valorNumeroFinal;
                        } else {
                            IrReceiver.resume();
                        }
                    } else {
                        pulsacionPlay = true;
                        lcd.clear();
                        return valorNumeroFinal;
                    }
                } else {
                    if (posColumnaCursor > 0) {
                        valorNumeroFinal = valorNumeroFinal / 10;
                        posColumnaCursor = posColumnaCursor - 1;
                        imprimirYPosicionarLCD(" ", posColumnaCursor, 1, posColumnaCursor, 1);
                    }
                    IrReceiver.resume();
                }
            } else {
                IrReceiver.resume();
            }
        }
    }
}


// 2 -> Interfaz de gestión de receptor de señales infrarrojas
int decodificarNumeros(unsigned long numeroIR) {
    switch (numeroIR) {
        case 3910598400:
            return 0;
            break;
        case 4077715200:
            return 1;
            break;
        case 3877175040:
            return 2;
            break;
        case 2707357440:
            return 3;
            break;
        case 4144561920:
            return 4;
            break;
        case 3810328320:
            return 5;
            break;
        case 2774204160:
            return 6;
            break;
        case 3175284480:
            return 7;
            break;
        case 2907897600:
            return 8;
            break;
        case 3041591040:
            return 9;
            break;
        case 3208707840: // PLAY
            return -1;
            break;
        case 3141861120: // BACK
            return -2;
            break;
        default:         // OTHER VALUES
            return -3;
```

```
            break;
    }
}


// 3 -> Interfaz de control del rodillo
void girarRodillo(int tiempoGiroOriginal) {
    // Se activa giro hacia delante
    digitalWrite(rodilloDirA, HIGH);
    delay(tiempoGiroOriginal);

    // Giro hacia atrás
    digitalWrite(rodilloDirA, LOW);
    digitalWrite(rodilloDirB, HIGH);
    delay(tiempoGiroOriginal + 300);

    // Se desactiva el motor
    digitalWrite(rodilloDirA, LOW);
    digitalWrite(rodilloDirB, LOW);
}


// 4 -> Interfaz de control del Piezo
void sonar(int sonido) {
    // Sonido = 0 => Sonido de éxito
    // Sonido = 1 => Sonido de fallo
    if (sonido == 0) {
        int melodia[] = {659, 8, 659, 8, 0, 8, 659, 8, 0, 8, 523, 8, 659, 8, 784, 4};
        int numNotas = sizeof(melodia) / sizeof(melodia[0]) / 2;
        int tempo = 140;
        for (int notaActual = 0; notaActual < numNotas * 2; notaActual = notaActual + 2) {
            tone(piezo, melodia[notaActual], tempo * 0.9 * melodia[notaActual + 1]);
            delay(tempo);
        }
        noTone(piezo);
    } else if (sonido == 1) {
        int melodia[] = {440, 4, 392, 4, 349, 4};
        int numNotas = sizeof(melodia) / sizeof(melodia[0]) / 2;
        int tempo = 50;
        for (int notaActual = 0; notaActual < numNotas * 2; notaActual = notaActual + 2) {
            tone(piezo, melodia[notaActual], tempo * 0.9 * melodia[notaActual + 1]);
            delay(tempo);
        }
        noTone(piezo);
    }
}


// 5 -> Lógica del reparto
bool combinacionValida() {
    if (numJugadores < 2 || numJugadores > 10) {
        return false;
    }

    if (cartasCadaUno == 0) {
      cartasCadaUno = (int) numCartas / numJugadores;
    }
    int cartasRepartidas = numJugadores * cartasCadaUno;
    cartasSobrantes = numCartas - cartasRepartidas;
    return cartasSobrantes >= 0;
}

// Cálculo del ángulo según el número de jugadores
// Tiene en cuenta el posible desfase producido al tener un número de jugadores
// no divisible por 360 grados (p. ej. 7 jugadores).
int calculoAngulo() {
    float anguloDecimales = 360 / numJugadores;
    float anguloEntero = floor(anguloDecimales);
    personasMasAngulo = (int) (anguloDecimales - anguloEntero) * numJugadores;
    return anguloEntero;
}

int anguloProximoGiro(int jugadorActual) {
    int angulo = calculoAngulo();
    if (jugadorActual <= personasMasAngulo) {
        return angulo + 1;
    } else {
        return angulo;
    }
}

int tiempoParaAngulo(int angulo) {
    float tiempoVueltaMilisegundos = 1.0 / frecuenciaMicroservoRpm * 60000;
    float tiempoGiroAngulo = (tiempoVueltaMilisegundos * angulo) / 360;
    return round(tiempoGiroAngulo);
}

// Obtiene el ángulo para el jugador actual y metiendo el ángulo estándar
void girarBase(int jugadorActual, int valorServo) {
    if (valorServo > 90){
        microservo.write(velocidadSentidoCCW);
    } else {
        microservo.write(velocidadSentidoCW);
    }
    delay(tiempoParaAngulo(anguloProximoGiro(jugadorActual)));
```

```
    microservo.write(90);
}


// 6 -> Gestión de errores (UI)
void imprimirPopupPlayContinuar(String mensaje, int sonidoAsociado) {
    imprimirYPosicionarLCD(mensaje, 0, 0, 0, 1);
    imprimirYPosicionarLCD("Play continuar", 0, 1, 14, 1);
    if (sonidoAsociado >= 0) {
        sonar(sonidoAsociado);
    }
    IrReceiver.begin(infrarrojo, DISABLE_LED_FEEDBACK);
    bool pulsacionPlay = false;
    while (!pulsacionPlay) {
        if (IrReceiver.decode()) {
            int ValorIntroducido = decodificarNumeros((unsigned long) IrReceiver.decodedIRData.decodedRawData);
            if (ValorIntroducido == -1) {
                pulsacionPlay = true;
            } else {
                IrReceiver.resume();
            }
        }
    }
    lcd.clear();
}


// 7 -> Inicialización
void inicializarComprobarParametros() {
    bool valido = false;
    do {
        numJugadores = escribirNumeros(99, "Num jugadores:", true);
        numCartas = escribirNumeros(999, "Num cartas:", true);
        cartasCadaUno = escribirNumeros(99, "Cartas x jug:", false);

        valido = combinacionValida();
        if (!valido) imprimirPopupPlayContinuar("Comb imposible", 1);
        lcd.clear();
    } while (!valido);

    // cartasSobrantes se inicializa al validar
    if (cartasCadaUno == 0) cartasCadaUno = (numCartas - cartasSobrantes) / numJugadores;
}

void setup() {
    // Configuración de pines que no dependen de librerías
    pinMode(rodilloEnable, OUTPUT);
    pinMode(rodilloDirA, OUTPUT);
    pinMode(rodilloDirB, OUTPUT);

    // Configuración microservo
    microservo.attach(motorMicroservo);

    // Inicialización de pantalla y de interacción con usuario
    inicializarLCD();
}


// 8 -> Algoritmo cíclico (loop)
void loop() {
    // 1 -> Inicializar parámetros
    // Interacción con el usuario para inicializar valores
    inicializarComprobarParametros();

    // 2 -> Solo se continúa cuando todo ha ido bien
    // Para empezar a repartir tengo que saber con qué angulo voy a hacerlo
    // Puedo tener ángulos diferentes según el nº de jugadores, importante usar bien el personasMasAngulo
    // Jugadores numerados de 1 a n, donde n es el nº de jugadores total
    int jugadorActual = 1;

    // Una vuelta en cada sentido, así que controlo el sentido
    // Empiezo en horario (0) y luego antihorario (180)
    // El servo para con 90
    int valorServo = 0;

    // He dado x vueltas
    int vueltasCompletas = 0;

    // While hasta llegar al nº de cartas para cada uno
    while (vueltasCompletas < cartasCadaUno) {
        // Lanzo carta
        girarRodillo(260);
        cartasSobrantes -= 1;
        // Sujeto repartido, sumo/resto y a otra cosa
        if (valorServo == 0) jugadorActual += 1;
        else if (valorServo == 180) jugadorActual -= 1;
        // Controlo límites
        // Si he terminado una vuelta en sentido horario cambio valor de servo y vamos a restar, pero a este le reparto
carta de nuevo
        if (jugadorActual == numJugadores + 1) {
            valorServo = 180;
            vueltasCompletas += 1;
            jugadorActual -= 1;
        } else if (jugadorActual == 0) {
            valorServo = 0;
```

```
            vueltasCompletas += 1;
            jugadorActual += 1;
        } else {
            girarBase(jugadorActual, valorServo);
        }
    }
    // Reparto terminado
    imprimirPopupPlayContinuar("Fin", 0);
}
```

Este código se puede visualizar al completo en un formato más legible haciendo *click* aquí.

Casos de uso

Se introducen a continuación dos vídeos cortos que permiten ver el dispositivo en funcionamiento:

Caso 1: inicialización de valores	Caso 2: funcionamiento del sistema
Vídeo en YouTube	Vídeo en YouTube

Vídeo resumen del proyecto

En este vídeo-resumen del proyecto, los miembros del grupo revisan el producto final elaborado en tres secciones principales: *hardware*, *software* y casos de uso.

Este vídeo puede ser visualizado en YouTube haciendo *click* aquí o entrando en el enlace:

bit.ly/SETR-G6-Video

Recursos asociados y contacto

Imagen 6. Información sobre vías de divulgación acerca del proyecto.

Todos los recursos empleados para la ejecución de la práctica están disponibles en un repositorio de acceso público para su libre visualización y reutilización bajo licencia Apache 2.0. El repositorio puede ser accedido haciendo *click* aquí. Los archivos que se introducen son:

- Presentación empleada para la exposición realizada en el aula el viernes 29 de abril de 2022.
- Código implementado y cargado en la placa Arduino UNO en la fecha de la exposición y de la grabación de los vídeos introducidos en este post.
- Modelo 3D desarrollado en formato SCAD exportado como STL del aspecto físico del producto.
- Esquema en máxima resolución del plano de componentes electrónicos empleado en la presentación adjunta.
- Memoria ejecutiva sobre la realización del proyecto (el presente documento) en formato PDF. Contiene la misma información que el *post* del *blog* oficial de la asignatura.