UNIVERSITAT POLITÈCNICA DE VALÈNCIA

# UNIVERSITAT POLITÈCNICA DE VALÈNCIA

## Escuela Técnica Superior de Ingeniería Industrial

# Diseño e implementación de un controlador y secuenciador para un emisor laser RGB basado en microcontrolador Arduino.

Trabajo Fin de Grado

Grado en Ingeniería en Tecnologías Industriales

AUTOR/A: Li Zheng, Xiaohui

Tutor/a: Pérez Jiménez, Alberto José

CURSO ACADÉMICO: 2021/2022

## AGRADECIMIENTOS

En primer lugar me gustaría agradecer a Alberto, por aceptar ser mi tutor, por toda la ayuda ofrecida, y por su simpatía y paciencia durante todo el desarrollo del trabajo.

También quería agradecer a Chiña, por su compañía y apoyo constante.

A Nancy y Felipe, por los valores emocionales recibidos en los momentos de fatiga y estrés.

Y por último quería dedicar mi agradecimiento a mi familia, por estar siempre a mi lado, y seguir juntos adelante.

## RESUMEN

En los últimos tiempos, los láseres RGB llegan a formar un componente importante en muchas técnicas y maquinarias, desde las más elementales hasta las más vanguardias.

El siguiente proyecto tiene como objetivo el diseño e implementación de un circuito secuenciador de láser RGB mediante placa de Arduino Nano, el dispositivo debe desempeñar funciones de calibración del láser, reproducción de secuencias de colores de forma automática y permite la programación de las secuencias desde una computadora. El código de programa para el funcionamiento se ha desarrollado con la ayuda del entorno de desarrollo Arduino IDE, que está basado en el lenguaje Java.

El trabajo termina con un manual de usuario explicando el funcionamiento del sistema.



Palabras Clave: Arduino Nano; Láser RGB; microcontrolador

# RESUM

En els darrers temps, els làsers RGB arriben a formar un component important en moltes tècniques i maquinàries, des de les més elementals fins a les més avantguardes.

El següent projecte té com a objectiu el disseny i la implementació d'un circuit seqüenciador de làser RGB mitjançant placa d'Arduino Nano, el dispositiu ha d'exercir funcions de calibratge del làser, reproducció de seqüències de colors de forma automàtica i permet la programació de les seqüències des d'una ordinador. El codi de programa per al funcionament s'ha desenvolupat amb l'ajuda de l'entorn de desenvolupament Arduino IDE, basat en el llenguatge Java.

El treball acaba amb un manual dusuari explicant el funcionament del sistema.

Paraules Clau: Arduino Nano; Làser RGB; microcontrolador

# ABSTRACT

In recent times, RGB lasers have become an important component in many techniques and machinery, from the most elementary to the most avant-garde.

The following project aims to design and implement an RGB laser sequencer circuit using an Arduino Nano board, the device must perform functions of laser calibration, reproduction of color sequences automatically and allows the programming of sequences from a computer. computer. The program code for operation has been developed with the help of the Arduino IDE development environment, which is based on the Java language.

The work ends with a user manual explaining the operation of the system.



Keywords: Arduino Nano; RGB Laser; microcontroller

# INDICE DE CONTENIDOS

# Documento I

# Memoria

# 1 Introducción

## 1.1 Motivación, justificación y objetivos

### 1.1.1 Motivación

Láser es un acrónimo de "amplificación de la luz por emisión estimulada de radiación". Un Láser es un dispositivo que estimula átomos o moléculas para que emitan luz a determinadas longitudes de onda y amplifica esa luz, produciendo normalmente un haz de radiación muy estrecho. La emisión suele abarcar una gama muy limitada de longitudes de onda visibles, infrarrojas o ultravioletas. Se han desarrollado muchos tipos diferentes de láseres, con características muy variadas.

Una gama importante de los láser sería el sistema de visualización de luz láser RGB (diodos rojos, verdes y azules). Mediante un sofisticado sistema de modulación y escáneres, las pantallas de los proyectores láser pueden mezclar colores y cubrir la mayor parte del espectro visible de la luz (también llamado longitud de onda). Un proyector láser RGB es capaz de proyectar un color blanco puro (mezclando los tres colores correctamente), por lo que los sistemas de visualización láser de luz RGB también se denominan sistemas de visualización láser a todo color o láser de luz blanca.

Gracias a la expansión continua de los campos de aplicación, los láseres RGB forman un componente importante en muchas maquinarias y técnicas tanto en ámbito científico, tecnológico como industrial.

Como la mayoría de los dispositivos electrónicos, el láser RGB requiere un sistema de control para su configuración y puesta en funcionamiento, un sistema de Arduino puede ser una buena alternativa, tiene la ventaja de tener una reducida dimensión, lo cual consigue que la maquinaria no dependa de una computadora adicional impidiendo así su portabilidad. Y además es económica, por lo que permite reducir el coste del conjunto de equipo.

### 1.1.2 Justificación

Existen numerosas aplicaciones de láseres RGB en los diferentes campos, en este apartado se presenta algunos ejemplos de su uso más vanguardia.

**La secuenciación de ADN**

La secuenciación de ADN es el proceso de determinar la secuencia de nucleótidos (As, Ts, Cs y Gs) de un fragmento de ADN. En la secuenciación (Sanger), el ADN blanco es copiado muchas veces y se hacen fragmentos de diferentes longitudes. Los nucleótidos fluorescentes que actúan como "terminadores de cadena" marcan los extremos de los fragmentos y son irradiadas por un láser y captadas por un detector que permiten la determinación de la secuencia.

Las técnicas de secuenciación de nueva generación son estrategias nuevas a gran escala que aumentan la velocidad y reducen el costo de la secuenciación del ADN. Las longitudes características de láser en esta aplicación son de 473 nm hasta 639 nm.

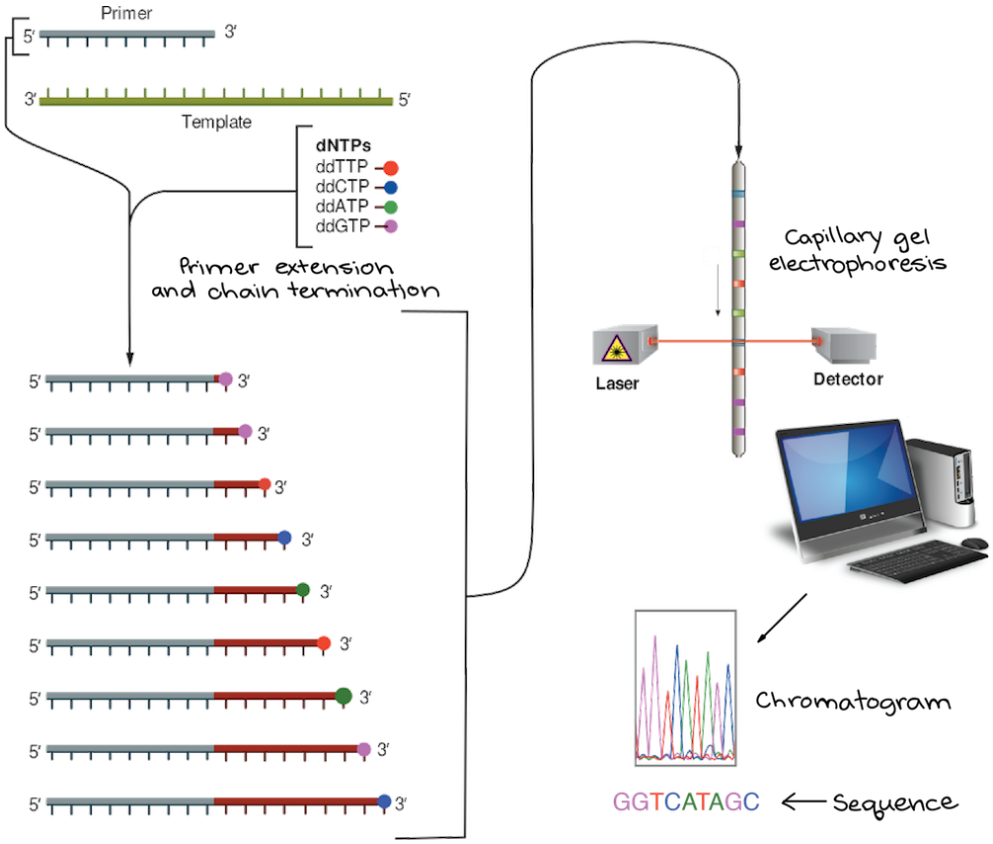


*Figura 1. Imagen modificada de "Secuenciación de Sanger" de Estevezj [1]*

## Holografía

La holografía o visión gráfica es una técnica avanzada de fotografía que consiste en crear imágenes tridimensionales basada en el empleo de la luz. Para esto se utiliza un rayo láser que graba microscópicamente una película fotosensible. La interferencia que se produce entre dos haces de luz coherentes hace posible que la luz de uno de estos se reflecte en el objeto. Esta, al recibir una luz puntual desde la perspectiva adecuada, proyecta una imagen en tres dimensiones.

Además, procesadas e iluminadas de manera precisa, las imágenes pueden aparecer saliéndose de sus límites, hacia fuera o hacia dentro del marco, y el observador, sin tener la necesidad de ningún accesorio, las puede ver sin discontinuidades y variando las perspectivas dependiendo de su posición.

La utilización de las técnicas holográficas en sistemas de vídeo es un proceso complejo que supone un gran reto a nivel tecnológico. Si se pueden resolver estos retos, se podría convertir en el sistema que se utilizaría en una futura televisión tridimensional.

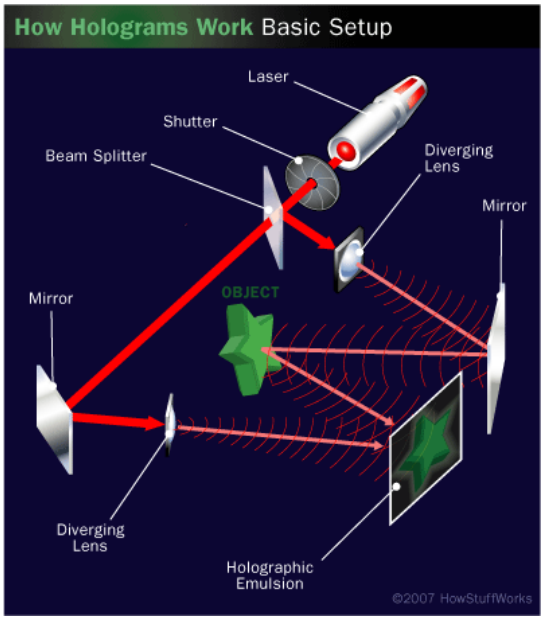


*Figura 2. Esquema láser en holografía [2]*

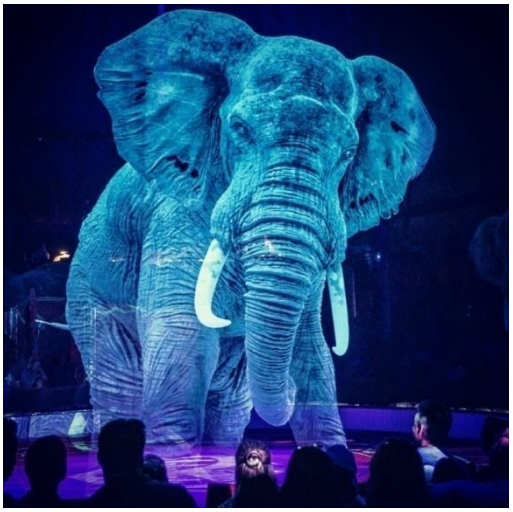

*Figura 3. Efecto láser holografía*

**La inspección visual de alta precisión**

La inspección visual de alta precisión es una rama de la inteligencia artificial en rápido desarrollo. En pocas palabras, es utilizar máquinas en lugar de ojos humanos para realizar mediciones y toma de decisiones.


*Figura 4. Máquina De Inspección Visual Ccd De Alta Precisión*

El núcleo de un sistema de inspección visual de alta precisión es la adquisición y el procesamiento de imágenes. Toda la información proviene de la imagen, y la calidad de la imagen en sí es extremadamente crítica para todo el sistema visual. La fuente de luz es un factor importante que afecta el nivel de imagen de todo el sistema, ya que afecta directamente la calidad de los datos de entrada y el efecto de la aplicación.

Mediante el diseño apropiado de la fuente de luz y el sistema óptico, la información del objetivo y la información de fondo en la imagen se pueden separar de manera óptima, lo que puede reducir en gran medida la dificultad de la segmentación e identificación del algoritmo de procesamiento de imágenes y, al mismo tiempo, mejorar la precisión de posicionamiento y medición de la imagen. Lo que hace que el sistema sea confiable y estable, mejorando el rendimiento general. Las longitudes de onda características del láser abarcan rangos del 405 nm al 980 nm.

En la actualidad, la aplicación de inspección visual de alta precisión es bastante popular, principalmente en electrónica, automóvil, metalurgia, industria de alimentación, etc.

**Técnica de control para alimentos líquidos mediante Imágenes de retrodispersión láser**

El enfoque de esta técnica es implementar un sistema de control que recopile rápidamente grandes cantidades de datos de las operaciones de la cadena de procesos de la industria alimentaria para luego mejorar la toma de decisiones sobre las modificaciones que se requieran en cada momento.

Otra mayor ventaja de esta técnica es que se trata de una técnica no destructiva, es decir, utiliza principio físico-químico para recopilar datos sin entrar en contacto con las muestras ni modificarlas. En la industria alimentaria, esto representa no solo la optimización de recursos para el procesamiento, sino también importantes avances en términos de control de calidad/seguridad que van desde la recepción de la materia prima hasta el almacenamiento del producto final dada la reducción de puntos posibles de contaminación.

La Figura 5A muestra la configuración del dispositivo de la técnica, consiste en un diodo láser rojo y una cámara conectado a una computadora. El rayo láser apuntó a través una placa que contiene la muestra de la crema, una imagen del rayo láser disperso se forma en la superficie del alimento y fue capturado por la cámara.

Un alto contraste y las zonas de color bien diferenciadas son fundamentales en esta técnica para obtener correctos perfiles de color, a partir de los cuales se generan los descriptores (Fig. 5C). Y como se ve en la Fig. 5B, se obtuvo un perfil seleccionando una línea de la imagen cruzando el patrón láser. Posteriormente, mediante un software específico se extrae los descriptores de imágenes para caracterizar las propiedades físico-químicas de los alimentos líquidos.

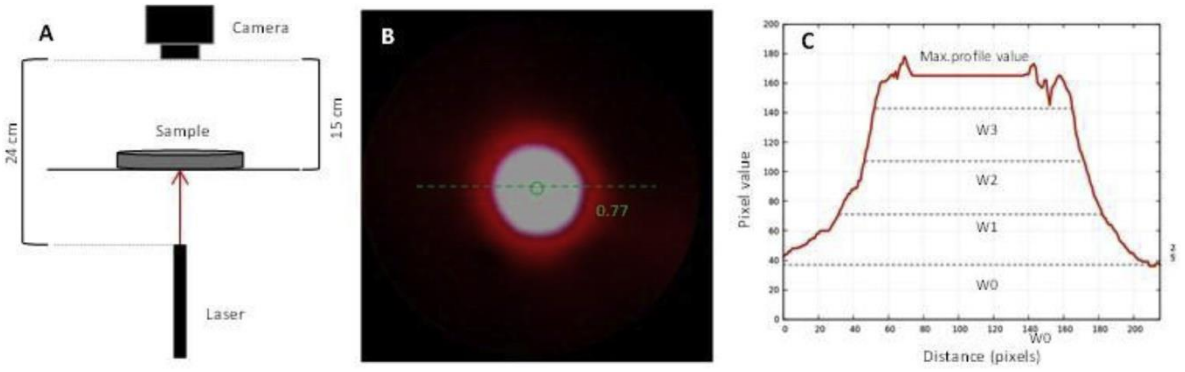


*Figura 5. A: esquema de configuración del dispositivo (la flecha roja representa el rayo láser). B: La imagen de muestra obtenida por el dispositivo C: Anchos de perfil a las alturas seleccionadas [3]*

### 1.1.3 Objetivos

Una vez explicadas la motivación y justificación que llevan a la realización del trabajo, se plantean una serie de objetivos que marcarán el camino a seguir a lo largo del desarrollo del mismo. Estos son los siguientes:

- Adquirir conocimientos sobre el sistema Arduino y la placa de Arduino Nano.
- Diseñar un circuito electrónico que permita configurar los parámetros del proyector láser RGB, tales como intensidad, color de haz, secuencia de luces, etc. Entender los conceptos teóricos de los diferentes componentes electrónicos.
- Desarrollar el código de programación para el funcionamiento del láser mediante la ejecución de los comandos desde la placa. Manejar las funciones y sintaxis básicas de programación en Arduino IDE.
- Comprobar y validar la exactitud y repetitividad del programa en el circuito secuenciador después de ser transferido a la placa Arduino Nano.

### 1.1.4 Metodología

Para cumplir el objetivo marcado en el epígrafe anterior, el trabajo realizado ha sido dividido en diferentes fases:

- Fase 1: Adquisición de los conocimientos y materiales necesarios e instalación de la herramienta de desarrollo. Durante esta fase se han investigado los principios básicos de la programación del Arduino y de los componentes electrónicos que forman parte del montaje.

- Fase 2: Realización del montaje en el protoboard, integrando los elementos de inputs y outputs al circuito, pineando de forma correcta con la placa de Arduino. Para facilitar el diseño y poder dar una referencia al usuario, se emplea un LED RGB en lugar del láser RGB en el montaje.

- Fase 3: Escribir el código de programa mediante Arduino IDE, el entorno de desarrollo basado en lenguaje C++.

- Fase 4: Redacción de la memoria.

## 1.2 Estructura de la memoria

Este proyecto consta de dos documentos: memoria y presupuesto. En el documento Memoria se describen la base teórica y los componentes empleados para el diseño y desarrollo del proyecto, así como una explicación detallada del código de programación y el manual de usuario para el manejo correcto del sistema. Así, el documento memoria queda estructurado en los siguientes capítulos:

**Capítulo 1: Introducción.** Presenta la motivación y justificación de la realización de este trabajo y enumera los objetivos a conseguir.

**Capítulo 2: Marco teórico: conceptos y componentes.** Explica en qué consiste el sistema Arduino, así como explicaciones sobre los conocimientos necesarios sobre los hardwares para la comprensión del mismo.

**Capítulo 3: Montaje de circuito electrónico.** Trata de explicar los distintos elementos que componen la instalación y su forma de integrar en el montaje.

**Capítulo 4: Código programa.** Este capítulo recoge la explicación detallada del código.

**Capítulo 5: Manual de usuario.** Valida el funcionamiento del sistema, y explica las maniobras a seguir para realizar distintas tareas.

**Capítulo 6: Conclusiones y trabajos futuros.** Se valorará el grado de consecución de los objetivos planteados y se plantearán los trabajos futuros que sería interesante realizar.

**Anexo: Código de programación completo**

Asimismo este Trabajo Final de Grado contiene también el documento Presupuesto en el que se recogen los costes derivados del desarrollo del mismo.

# 2 Marco teórico: conceptos y componentes

## 2.1 Arduino

### 2.1.1 Introducción

Arduino es una plataforma electrónica de código abierto basada en hardware y software de manejo sencillo. Las placas Arduino pueden leer entradas y convertirlo en una salida: activar un motor, encender un LED, etc. mediante transmisión de un conjunto de instrucciones al microcontrolador en la placa. Para ello se utiliza el lenguaje de programación Arduino (Wiring), y el Software Arduino (IDE), basado en Processing.


Arduino nació en el Instituto de Diseño de Interacción Ivrea como una herramienta fácil para la creación rápida de prototipos, dirigida a estudiantes sin experiencia en electrónica y programación. Con el avance del tiempo llegó a una comunidad más amplia, la placa Arduino comenzó a cambiar para adaptarse a las nuevas necesidades y desafíos, diferenciando su oferta de simples placas de 8 bits a productos para aplicaciones IoT, wearables, impresión 3D y entornos integrados.

El gran éxito de las placas de Arduino en plataformas de microcontroladores se debe principalmente a los siguientes factores:

- Económicas: las placas Arduino son relativamente económicas en comparación con otras placas.
- Multiplataforma: el software Arduino (IDE) se ejecuta en los sistemas operativos Windows, Macintosh OSX y Linux. La mayoría de los sistemas de microcontroladores están limitados a Windows.
- Software extensible y de código abierto: el software Arduino se publica como herramientas de código abierto, disponibles para que los programadores experimentados lo extiendan. El lenguaje se puede expandir a través de librerías C++.

- Código abierto y hardware extensible: los planos de las placas Arduino se publican bajo una licencia Creativa Común, por lo que los diseñadores de circuitos pueden hacer su propia versión del módulo, ampliándolo y mejorándolo.

Para este trabajo se utiliza la placa Arduino Nano, una placa de desarrollo basada en el microcontrolador ATmega328P con una dimensión relativamente reducida dentro de las gamas de las placas de Arduino.

### 2.1.2 Software: Arduino IDE

El entorno de desarrollo integrado Arduino IDE representa el software básico ideal para poder crear, copiar o modificar código Arduino. Esta herramienta permite programar hardware Arduino de forma intuitiva. Su mayor potencial se encuentra en la compatibilidad, Arduino IDE permite programar la gran mayoría de placas existentes en el mercado. Y también por su simplicidad de ejecución.

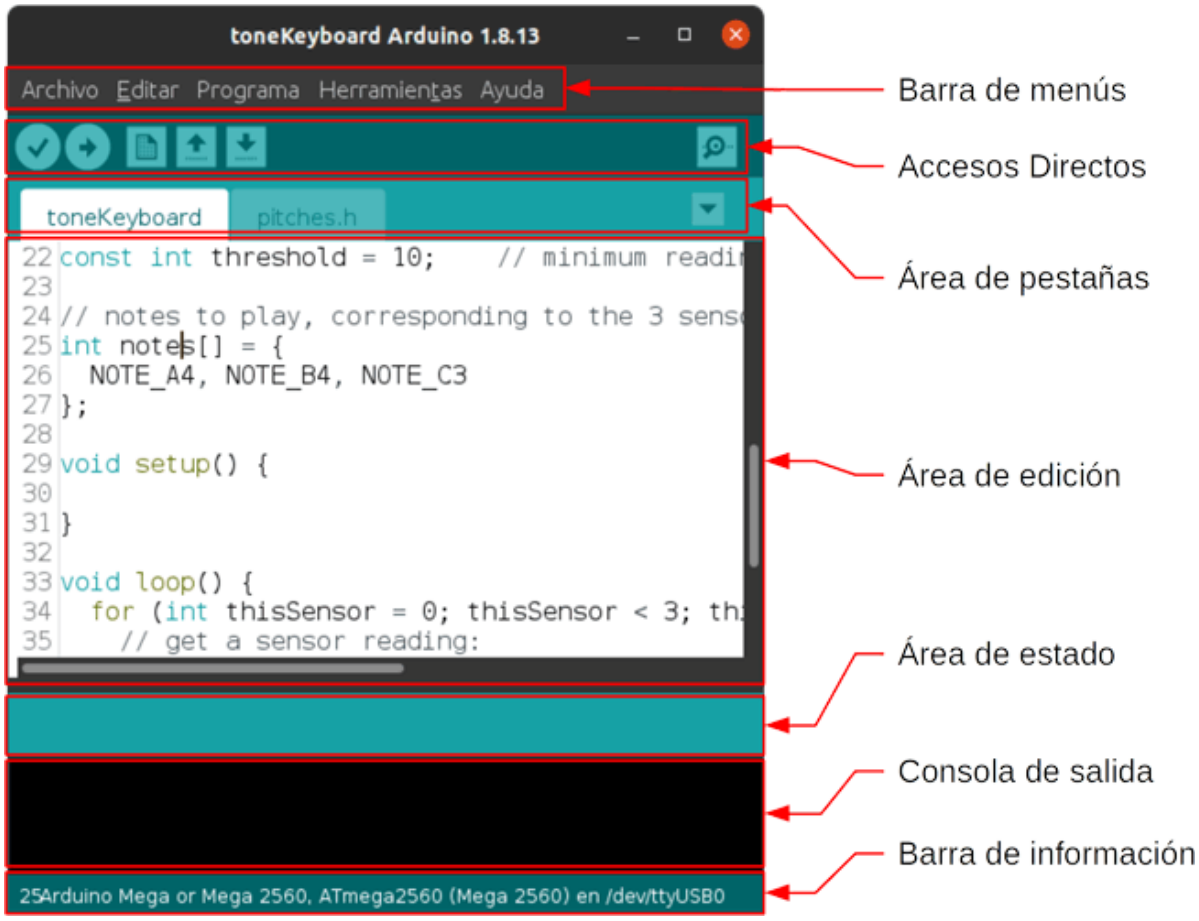


*Figura 6. Interfaz Arduino IDE*

Como se puede ver en la figura siguiente el software cuenta con varias secciones:

- **Barra de menús**: en esta barra se encuentran varios menús que permiten acceder a muchas de las acciones disponibles en el entorno de desarrollo.
- **Barra de accesos directos**: en esta barra se encuentra un grupo de botones con las acciones más comúnmente realizadas (como compilar y cargar el código al Arduino).
- **Área de pestañas**: en esta área se muestran los archivos correspondientes al proyecto abierto.
- **Área de edición**: en esta área es donde se escribe el código.
- **Área de estado**: en esta área se muestra una barra de progreso cuando se compila o carga el sketch al Arduino.
- **Consola de salida**: en esta se muestran las salidas generadas por el compilador.
- **Barra de información**: se muestra información sobre la configuración del IDE.

A pesar de las ventajas de Arduino IDE mencionado arriba, este software no es la única alternativa para programar las placas de Arduino. Existen otros entornos de desarrollo para programar, tales como Visual Studio (Visual Micro); UECIDE (Universal Embebed IDE); Programino IDE; etc.

### 2.1.3 Placa Arduino NANO

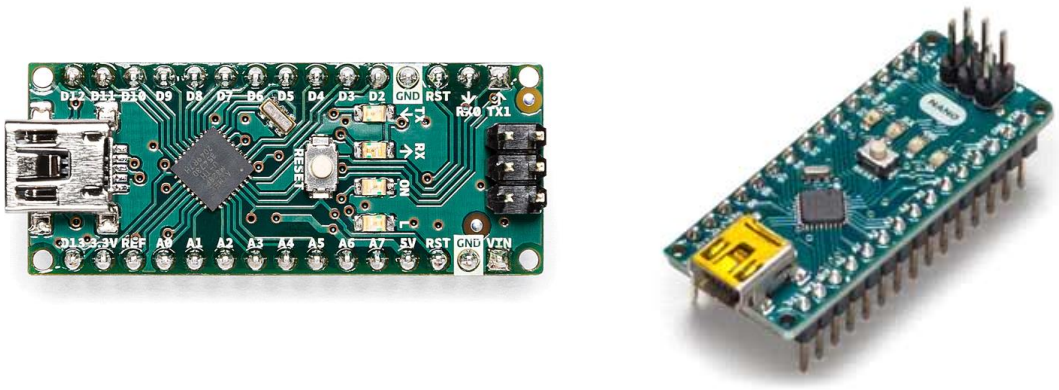


*Figura 7. Placa Arduino Nano [4]*

#### 2.1.3.1 Introducción

Arduino Nano es una placa de desarrollo basada en el microcontrolador ATmega328P caracterizada por su reducido tamaño (45 x 18 mm). El hecho de ser de menor tamaño lo hace especialmente útil cuando el proyecto a desarrollar requiere dimensiones reducidas. Además, la posición de sus pines facilita su uso en placas de prototipo.

#### 2.1.3.2 Características

| CARACTERÍSTICAS | DESCRIPCIÓN |
|---|---|
| Microcontrolador | ATmega328P |
| Voltaje operativo | 5V |
| Alimentación | 7-12V |
| Pines digitales I/O | 22 (6 Pines de PWM) |
| Entradas analógicas | 8 |
| Corriente Continua pines I/O | 40 mA |
| Memoria Flash | 32 KB (de los cuales 2 KB para bootloader) |
| Sram | 2 KB |
| EEPROM | 1 KB |
| Velocidad de reloj | 16 MHz |

*Tabla 1. Características placa Arduino Nano*

#### 2.1.3.3 Distribución de pines Arduino Nano (pineado)

- **Pines digitales y analógicos**

Arduino Nano cuenta con un total de 14 pines digitales de entrada/salida, de los cuales 6 pueden ser utilizados como salidas analógicas (utilizando señales PWM).

Y tiene 8 entradas analógicas con una resolución de 10 bits (1024 posibles valores). Estos pines pueden ser utilizados como digitales en caso de no ser necesarios como analógicos.

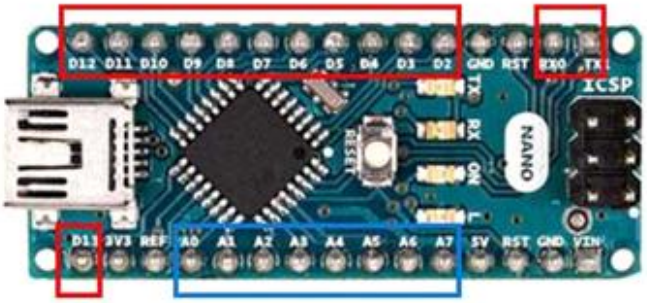


*Figura 8. Pines analógicas y digitales en Arduino Nano*

- **Puerto serie**

Los pines D0(TX) y D1(RX) corresponden al puerto Serial. Estos están conectados a la interfaz USB-Serie para permitir la comunicación entre la placa y el ordenador.

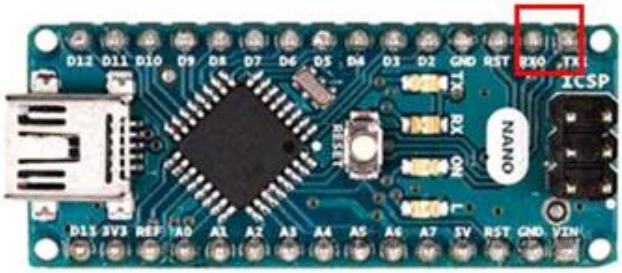


*Figura 9. Pines puerto serie en Arduino Nano*

- **Bus I2C**

También cuenta con un bus I2C en los pines A4 (SDA) y A5 (SCL), en el capítulo de montaje se verá que son utilizados para la conexión entre placa y la pantalla OLED.

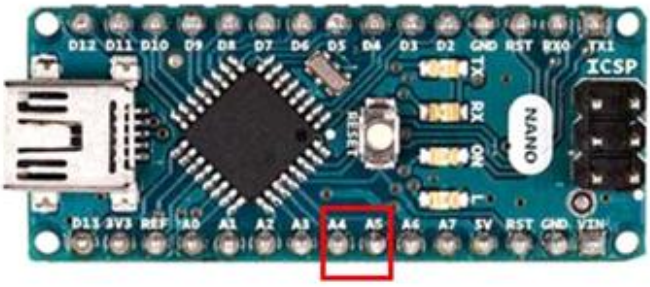


*Figura 10. Pines Bus I2C en Arduino Nano*

- **Pines de alimentación**

  El Arduino Nano cuenta con pines de alimentación:

  - **5V**: este pin ofrece los 5 voltios estables con que es energizado el microcontrolador y el resto de los componentes en la placa.
  - **3V3**: ofrece un voltaje de 3.3 voltios. (La corriente máxima que se puede extraer de este pin es de 50 mA).
  - **VIN**: permite alimentar la placa con un voltaje entre 6 y 20 voltios.
  - **GND**: tierra de la placa.

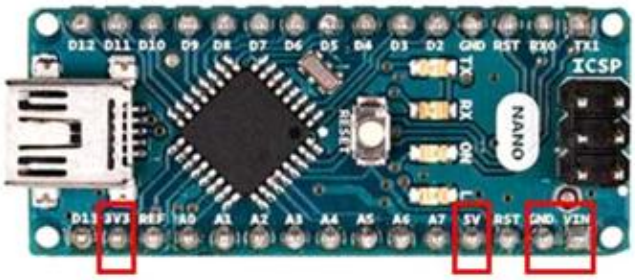


*Figura 11. Pines alimentación en Arduino Nano*

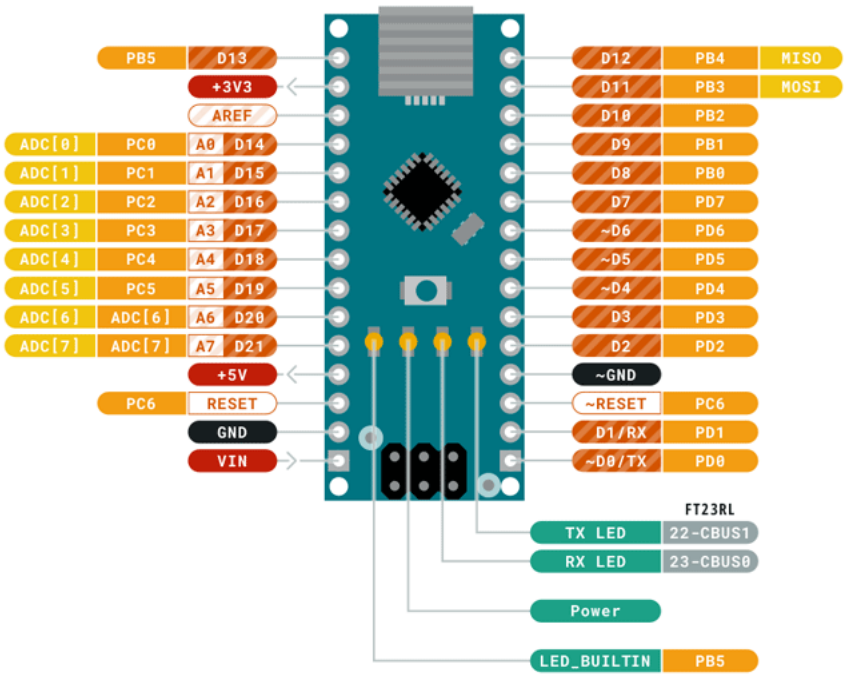


*Figura 12. Esquema pineado completo Arduino Nano [4]*

#### 2.1.3.4 Otros elementos en la placa

- **Botón de Reinicio**

El botón RESET está conectado al pin RESET del microcontrolador y por tanto permite reiniciar el microcontrolador. Esto significa que todo el código programado será ejecutado nuevamente, tal y como si el sistema se acabara de conectar a la alimentación.

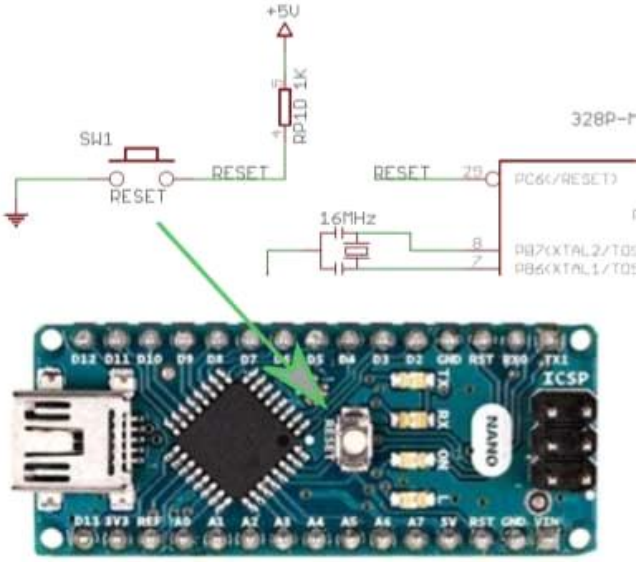


*Figura 13. Botón reset en Arduino Nano*

- **LEDs**

En la zona central derecha de la placa integran 4 leds que desempeña funciones diferentes:

- **LED ON**: este LED está conectado a la salida del selector de voltaje, por lo tanto se enciende cuando la placa está siendo alimentada correctamente.
- **LED Tx**: pestañea cuando el Arduino transmite información al ordenador. Esto permite comprobar de forma simple si la placa está realmente transmitiendo información.
- **LED Rx**: pestañea cuando la placa recibe información del ordenador. De este modo se puede comprobar si realmente se está efectuando la comunicación.
- **LED Integrado**: este LED está conectado al pin digital 13 y es denotado como LED_BUILTIN. Para encenderlo/apagarlo es necesario poner este pin en un estado alto (HIGH)/bajo (LOW).

## 2.2 Pantalla OLED

### 2.2.1 Introducción

OLED son las siglas en inglés de Organic Light-Emitting Diode que traducido sería diodo orgánico de emisión de luz.

Se trata de una tecnología de emisión de luz plana, hecha colocando una serie de películas delgadas orgánicas entre dos conductores. Cuando se aplica corriente eléctrica, se emite una luz brillante. Los OLED son pantallas emisoras que no requieren retroiluminación y, por lo tanto, son más delgadas y eficientes que las pantallas LCD (que sí requieren retroiluminación blanca).

Y a día de hoy, las pantallas OLED se utilizan en muchos dispositivos, y como objetivo de este apartado se centra en las pantallas OLED con Arduino, en el montaje del circuito se emplea una pantalla de tamaño de 0,96” que contiene 128 x 32 pixeles.

### 2.2.2 Controlador SSD1306 para pantalla OLED con Arduino

Las pantallas OLED deben estar ensamblado con un driver o controlador, junto con toda la electrónica necesaria en un mismo módulo para poder mostrar datos e información en ella. Normalmente estos módulos utilizan un controlador bastante conocido que se llama SSD1306.

La comunicación entre el SSD1306 y el microcontrolador, en nuestro caso, con un Arduino Nano se realiza mediante I2C.

### 2.2.3 Visualización de gráficos en pantalla OLED

Para mostrar los datos en la pantalla, el controlador SSD1306 tiene una memoria RAM gráfica que ocupa 1 KB.

Esto equivale a 1.024 bytes o 8.192 bits que se distribuyen en la pantalla en una matriz de filas (páginas) y columnas (segmentos). En total hay 8 páginas (filas) y cada página tiene 128 segmentos (columnas) que, a su vez, cada segmento almacena 1 byte. Cada bit representa un píxel en la pantalla OLED y mediante la programación, se puede encender o apagar para que muestre cualquier información.

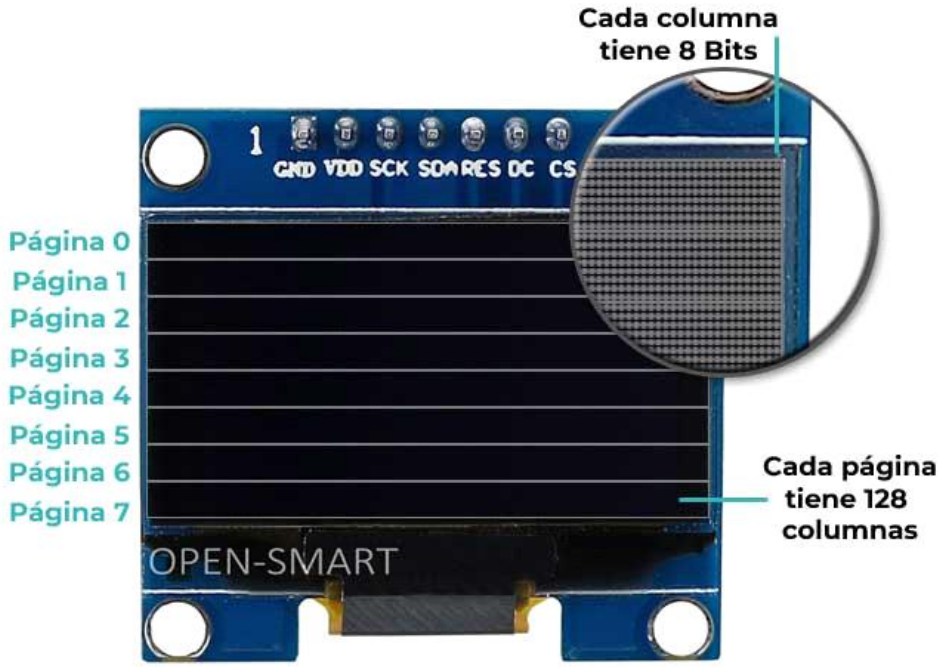


*Figura 14. Detalle Pantalla OLED 0,96" [5]*

## 2.3 Diodos LED

### 2.3.1 LED simple

Un LED es un diodo emisor de luz, es decir, un tipo particular de diodo que emite luz al ser atravesado por una corriente eléctrica. Los diodos (emisor de luz, o no) son unos de los dispositivos electrónicos fundamentales.

Un diodo es una unión de dos materiales semiconductores con dopados distintos, esta diferencia de dopado hace que genere una barrera de potencial, que como primera consecuencia hace que el paso de corriente en uno de los sentidos no sea posible, es llamado polaridad a esta característica del led. Por tanto, se tiene que conectar correctamente la tensión al dispositivo.

La patilla larga debe ser conectada al voltaje positivo (ánodo), y la corta al voltaje negativo (cátodo).

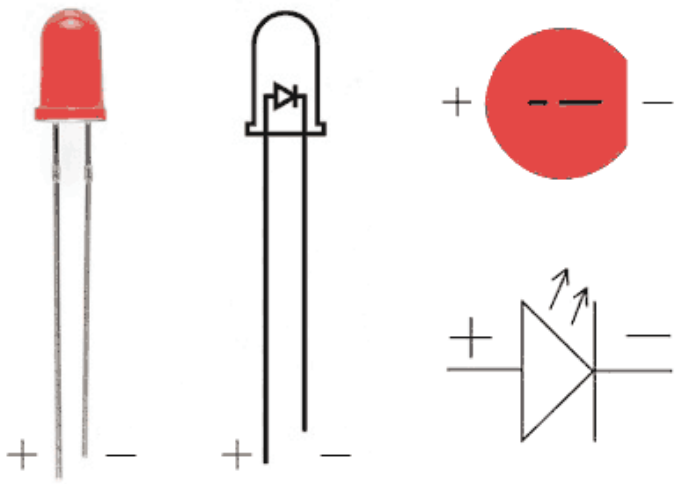

*Figura 15. Esquema LED simple*

### 2.3.2 LED RGB

El LED RGB es un tipo especial de diodo LED que se compone por varias matrices LEDs simples como las que se encuentran en otros LEDs monocolor, empaquetados en un mismo encapsulado. De esa forma, pueden emitir en estos tres colores primarios, generando así todo tipo de efectos y colores diferentes (incluso el blanco combinando el rojo, verde y azul a la vez) tan solo controlando una de las patillas de estos componentes.

Tiene un pinout algo diferente a los LEDs convencionales, ya que incluyen 3 pines, uno por cada color y otro más adicional común a todos, el cátodo (-).

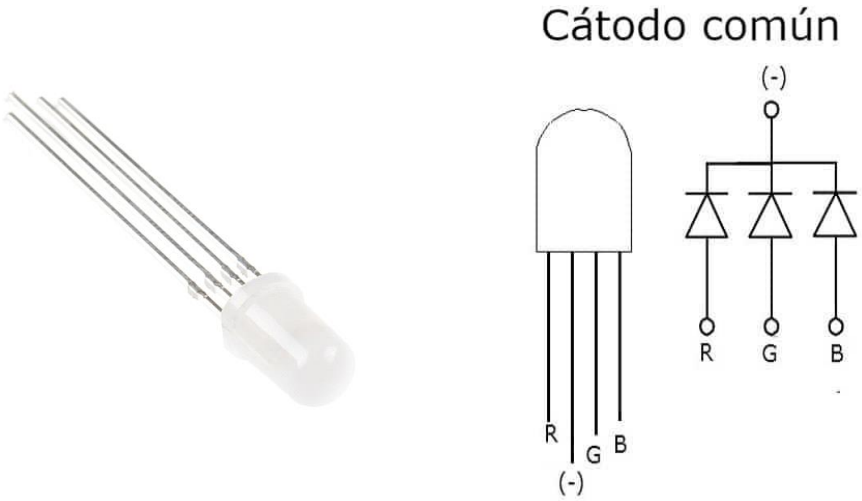


*Figura 16. Esquema LED RGB*

## 2.4 Láser RGB

Los láseres RGB son sistemas de láser con módulos de tres colores exactos: rojo, verde y azul. No importa si las fuentes de color son módulos láser DPSS, OPSL o Diodo. Combinando los tres colores, es posible obtener efectos de láser de luz blanca.

Se puede mostrar la mayoría de los colores con un sistema láser RGB que permite la modulación analógica mezclando tonos rojos, verdes y azules. Un proyector RGB típico con modulación analógica tiene 256 niveles de brillo para cada láser. Esto da (256 x 256 x 256) 16.777.216 colores diferentes disponibles.


*Figura 17. Láser RGB*

## 2.5 Pulsador

Un pulsador es un componente eléctrico presente en la mayoría de los aparatos electrónicos, consiste en un interruptor que funciona al mantener una presión sobre él, por norma general, los pulsadores tienen un estado inicial donde no permiten pasar la corriente.

El pulsador tiene cuatro patillas que están conectadas a pares como se ve en el siguiente esquema.

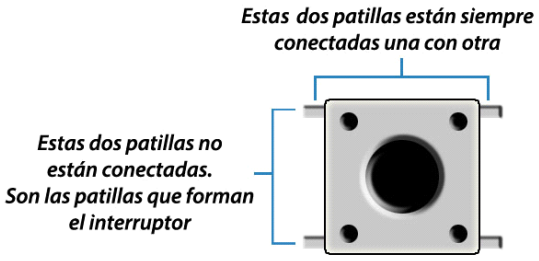


*Figura 18. Interruptor Pulsador*

Cuando se pulsa el interruptor se cierra el circuito y deja pasar la corriente. Esto permite, por ejemplo, controlar un LED, un motor o cualquier otro elemento, en los siguientes capítulos se explicará cómo se integran y su funcionamiento en nuestro circuito.

## 2.6 Protoboard

Una protoboard, o breadboard, es prácticamente una PCB temporal con una forma y tamaño generalizados. Utilizada comúnmente para pruebas y prototipos temporales de circuitos. Se usa insertando las terminales de los dispositivos electrónicos en los orificios de la protoboard de la forma en que tengan continuidad.

Básicamente un protoboard se divide en tres zonas:

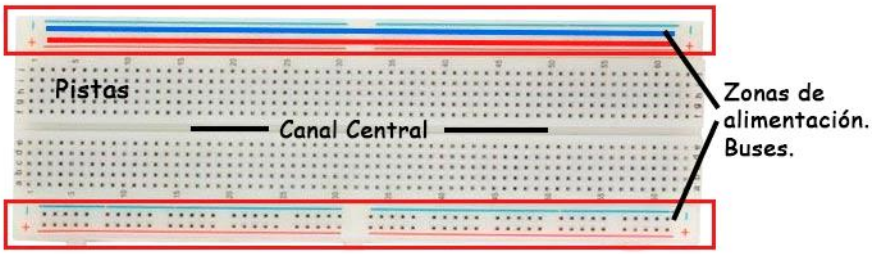


*Figura 19. Protoboard o tabla de prototipo*

- Buses: se localizan a ambos lados de la placa. Se representan por las líneas rojas (buses positivos o de voltaje) y azules (buses negativos o de tierra), no hay conexión física entre ellas. La fuente de alimentación generalmente se conecta aquí.
- Canal central: está ubicado en la zona central, y se utiliza para la conexión de los circuitos integrados manteniendo aislados los pines de ambos lados del circuito integrado.

- Pistas: el resto de los orificios de la protoboard pertenecen a las pistas. En las pistas, los agujeros de una columna, generalmente 5, están internamente conectados por una varilla de metal de resorte, pero no con los agujeros de las columnas adyacentes o la columna simétrica al surco. Así, es posible insertar los circuitos integrados a horcajadas; por lo tanto, para cada clavija quedan cuatro orificios disponibles para las conexiones con otros componentes.

# 3 Montaje de circuito electrónico

## 3.1 OUTPUTS

En informática, se conoce como dispositivos de salida (output) a aquellos aparatos que permiten la extracción o recuperación de información proveniente del sistema informático.

Los dispositivos de salida son también llamados periféricos de salida y traducen la información de una computadora a formatos visuales, sonoros, impresos o de cualquier otra naturaleza, que puedan ser comprendidos por el usuario.

En este apartado se explicará cómo se conectan los distintos componentes de Outputs utilizados en nuestro circuito con la placa Arduino.

### 3.1.1 Pineado módulo pantalla OLED con Arduino

La pantalla OLED utilizado en el montaje es una pantalla que utiliza la interfaz I2C. En este tipo de pantallas se puede encontrar 4 pines:

- **GND**: pin de tierra.
- **VCC**: es el pin de alimentación. Se puede alimentar la pantalla entre 1,8V y 6V.
- **SCL**: es el pin de la señal de reloj de la interfaz I2C.
- **SDA**: es el pin de la señal de datos de la interfaz I2C.

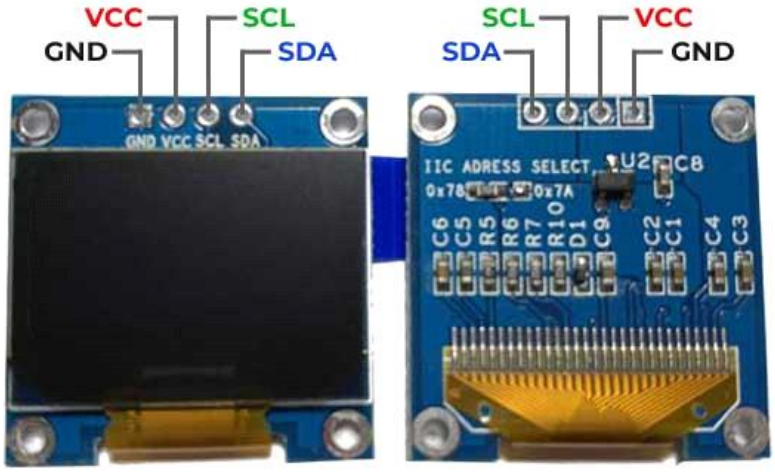


*Figura 20. Pineado módulo pantalla OLED 0,96" [5]*

El siguiente esquema se muestra cómo se conecta la pantalla OLED con un Arduino Nano. Se conecta el pin A4 al SDA, el pin A5 al SCL (o SCK), el pin de alimentación de 5V al VCC y el pin de GND al pin de tierra correspondiente del módulo de la pantalla.

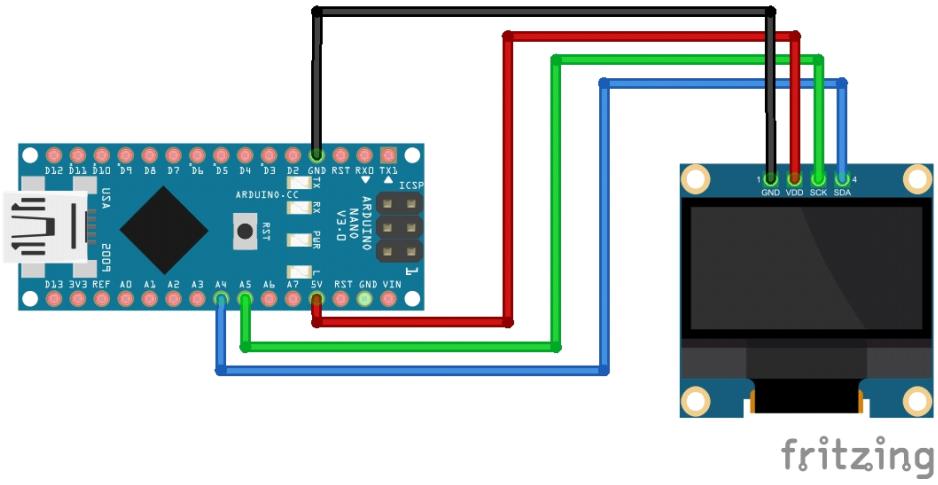


*Figura 21. Esquema conexión Arduino Nano con OLED*

### 3.1.2 Conexión Arduino con LED RGB

Para facilitar el diseño y montaje del secuenciador, se ha utilizado una LED RGB en lugar del láser RGB para simular el funcionamiento del circuito.

Un LED RGB es en realidad la unión de tres LEDs de los colores básicos, en un encapsulado común, compartiendo el Ground (cátodo es otro nombre más para el negativo).

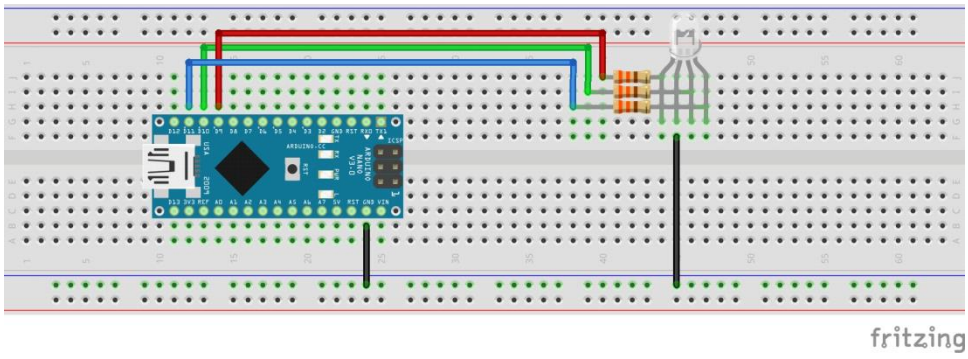


*Figura 22. Esquema conexión Arduino Nano con LED RGB*

El montaje consiste en sencillamente conectar el negativo (el pin más largo) a Ground, y luego identificar los pines de colores: el pin más largo en estos LED es el GND, al lado de GND hay dos pines a un lado y uno solitario al otro, por lo normal el solitario es el rojo R. Por lo que, el pin out (patillaje) de un RGB LED suele ser R, GND, G, B.

Los pines R, G y B se los conecta a los pines de salida de la placa mediante una resistencia que límite las intensidades.

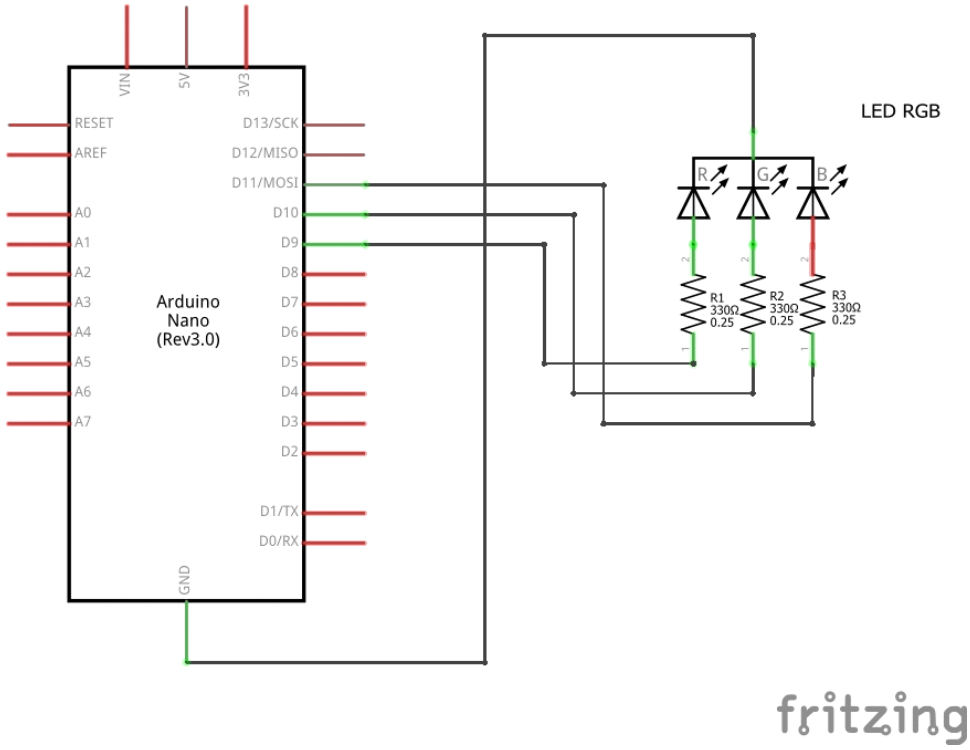


*Figura 23. Esquema eléctrico conexión Arduino Nano con LED RGB*

En este esquema se ha utilizado los pines 9, 10 y 11. El motivo de esta operación es porque en Arduino Nano, estos pines corresponden a las de salidas de señales PWM (Pulse Width Modulation), mediante los cuales se puede poner distintas intensidades mediante función *analogWrite*.

### 3.1.3 Conexión Arduino con Láser RGB

Aunque en el desarrollo del montaje no se va a proceder la conexión del circuito con el láser, ya que se utiliza en su lugar una LED RGB para la simulación, sí que convendría explicar el modo de conexión para la futura aplicación del dispositivo.

La conexión de la placa Arduino Nano con el láser se hace de manera similar a la conexión con la LED, se utilizan los pines 9, 10 y 11 de la placa, que son las correspondientes a las de PWM.

Los pines del Arduino se conectan a las entradas de control del laser, de manera que la entrada R se conecta con el pin 9, la G con el pin 10 y la B con el pin 11. Las entradas de control funcionan en el rango de 0 voltios a 5 voltios, iluminando el laser correspondiente entre 0% y 100% de su potencia. Este rango coincide con el rango de voltaje de los pines de salida del Arduino.


*Figura 24. Entradas de control de láser RGB*

## 3.2 INPUTS

A diferencia de los Outputs, los Inputs se definen como punto, puerto o dispositivo a través del cual una señal, energía, potencia o información es recibida por un dispositivo o un equipo.

En nuestro circuito se utiliza dos tipos de inputs, los pulsadores y el monitor serie del software Arduino IDE.

### 3.2.1 Conexión Arduino con pulsadores

Para conectar un interruptor pulsador o también llamado como botón al Arduino conviene tener en cuenta varios puntos: un pulsador puede conectarse a cualquier pin de Arduino (digital o analógico, ya que los analógicos usualmente funcionan también como digitales). Y se debe configurar el pin seleccionado como entrada digital. Además, en el montaje es necesario utilizar una resistencia pull-up o pull-down, ya sea interna o externa, estas dos resistencias son un mecanismo básico, muy habitual dentro del mundo de la electrónica y automatización.

Las dos formas de configuración que existen para conectar un botón al Arduino se visualizan en los siguientes diagramas:

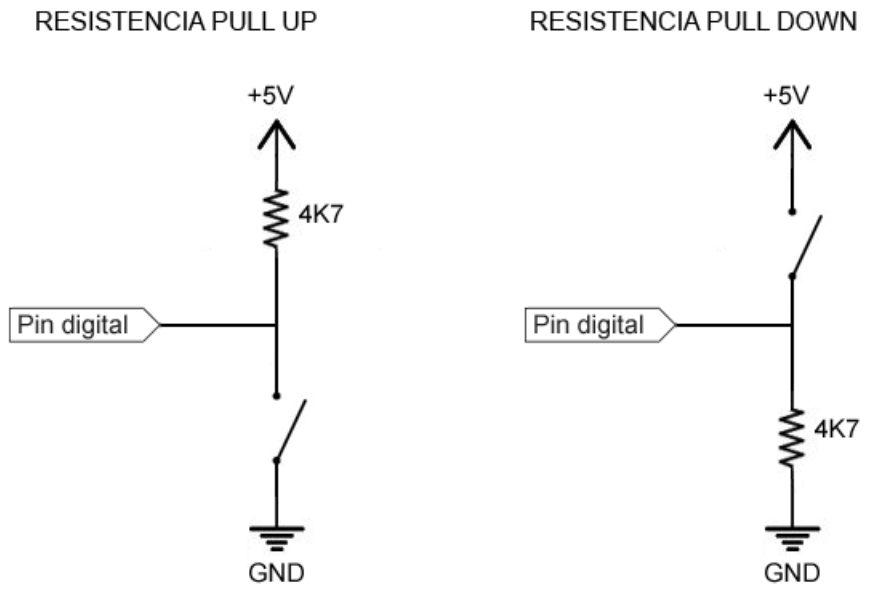


*Figura 25. Circuito Resistencia PULL UP y PULL DOWN*

Las resistencias de Pull-Down y Pull-Up se conectan entre el PIN digital y una de las tensiones de referencia (0V o 5V) y "fuerzan" el valor de la tensión a LOW o HIGH, respectivamente. La resistencia de Pull-Up fuerza HIGH cuando el pulsador está abierto. Cuando está cerrado el PIN se pone a LOW, la intensidad que circula se ve limitada por esta resistencia. Mientras que la resistencia de Pull-Down fuerza LOW cuando el pulsador está abierto. Cuando está cerrado el PIN se pone a HIGH, y la intensidad que circula se ve limitada por esta resistencia.

En nuestro montaje se utiliza la configuración Pull-Down, por lo que se mantiene en estado LOW mientras se deja el pulsador sin presionar. Y una vez pulsado, mandaría una señal HIGH a la placa Arduino.

En los siguientes esquemas se puede visualizar las conexiones de la placa con los tres pulsadores, de los cuales los dos que están situados en la zona central funcionan como cursores y al de la derecha se le da el uso para confirmar la selección. Que se verá con detalle el desarrollo en el siguiente capítulo.

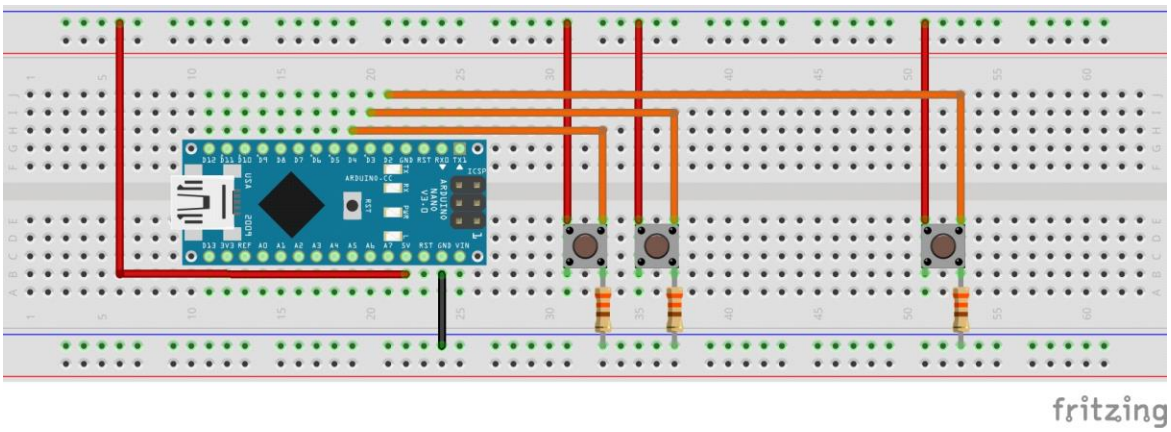


*Figura 26. Esquema conexión Arduino Nano con pulsadores*

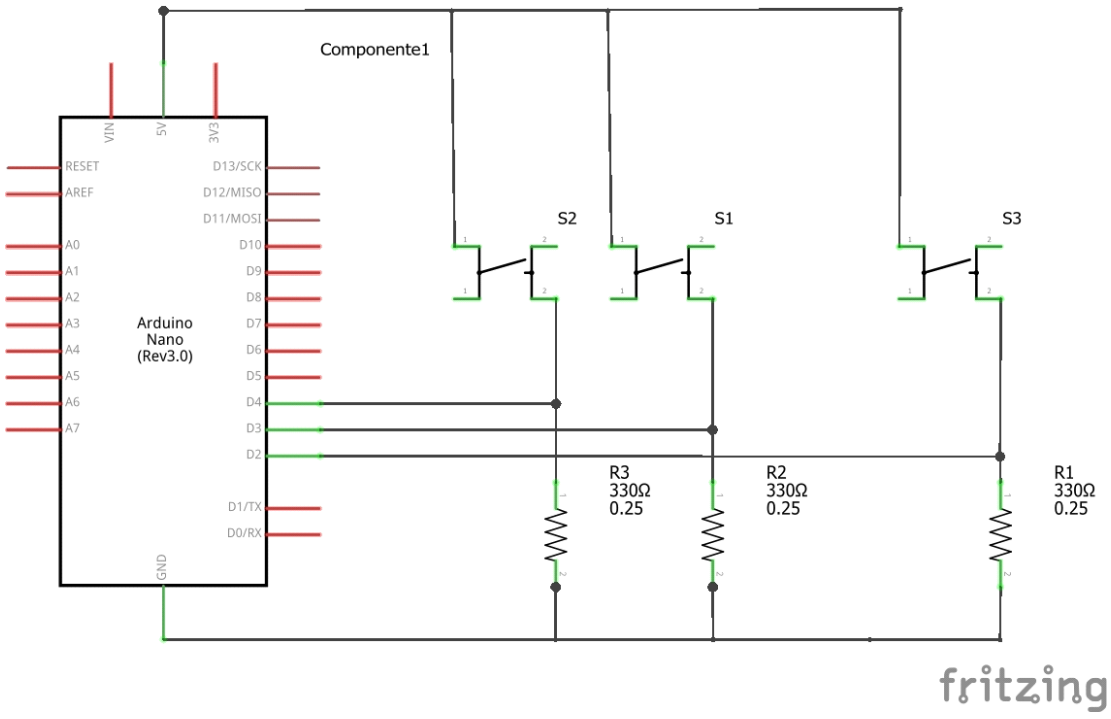


*Figura 27. Circuito conexión Arduino Nano con pulsadores*

El valor de las resistencias en este caso no es crítico, pero se debe considerar que valores menores causarán que circule una corriente mayor cuando se oprime el botón, mientras que los valores mayores pueden permitir que el ruido se introduzca al pin de entrada.

Los valores típicos para resistencia pull-up o pull-down son de 1 a 10 KOhms, en el montaje se utiliza 3 resistencias de valor 330 ohmios.

### 3.2.2 Monitor Serial

En el circuito de trabajo, se diseña una función para poder programar las secuencias por el usuario en el momento necesario, para este fin se va a necesitar enviar información desde el ordenador a la placa de Arduino Nano mediante el uso del monitor serie.

El monitor de puerto serie es una pequeña utilidad integrada dentro de IDE Standard que nos permite enviar y recibir fácilmente información a través del puerto serie, útil para el control de Arduino. Su uso es muy sencillo, y dispone de dos zonas, una que muestra los datos recibidos, y otra para enviarlos.

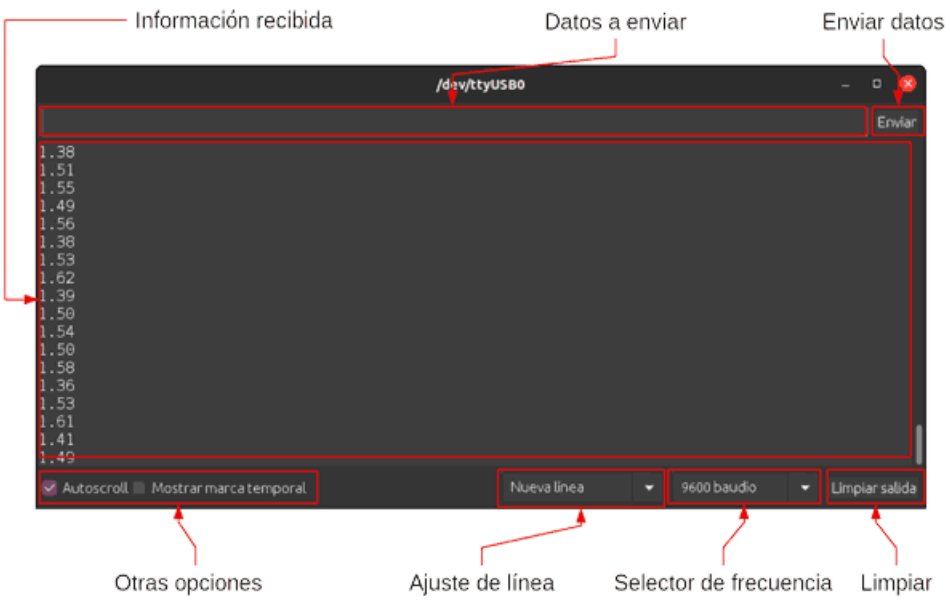


*Figura 28. Interfaz Monitor Serie de Arduino IDE*

### 3.2.3 Montaje completo del circuito prototipo

Teniendo definido las configuraciones de conexión de cada componente con la placa Arduino, se obtiene el montaje completo del circuito prototipo, tal como demuestra en la figura siguiente.

Se ha sustituido los cables de conexión a alimentación de 5V por cables de color amarillo, para no confundirse con los cables conectados a los pines del LED RGB. Lo mismo para la pantalla OLED, que se ha reemplazado los cables conectados a los pines de salida A4 y A5 por los de color blanco.

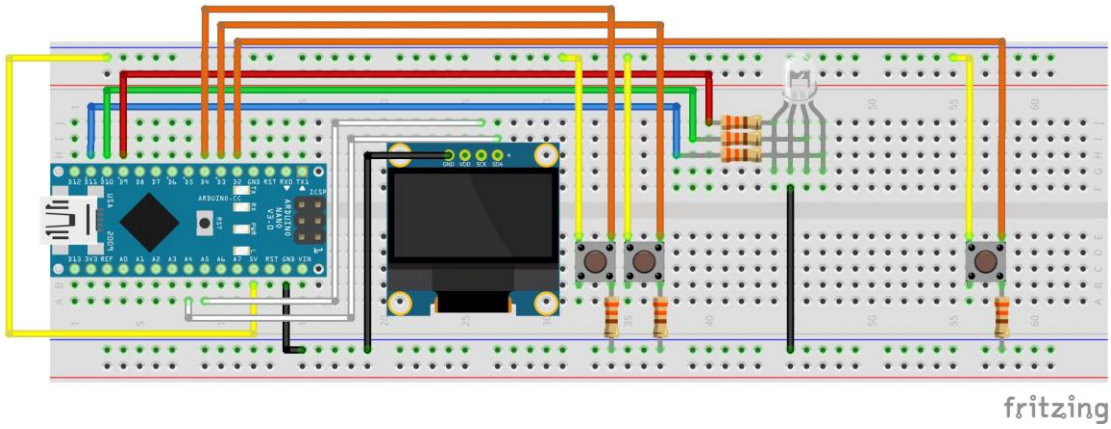


*Figura 29. Esquema conexión Arduino Nano completo*

El esquema eléctrico del circuito es el siguiente:

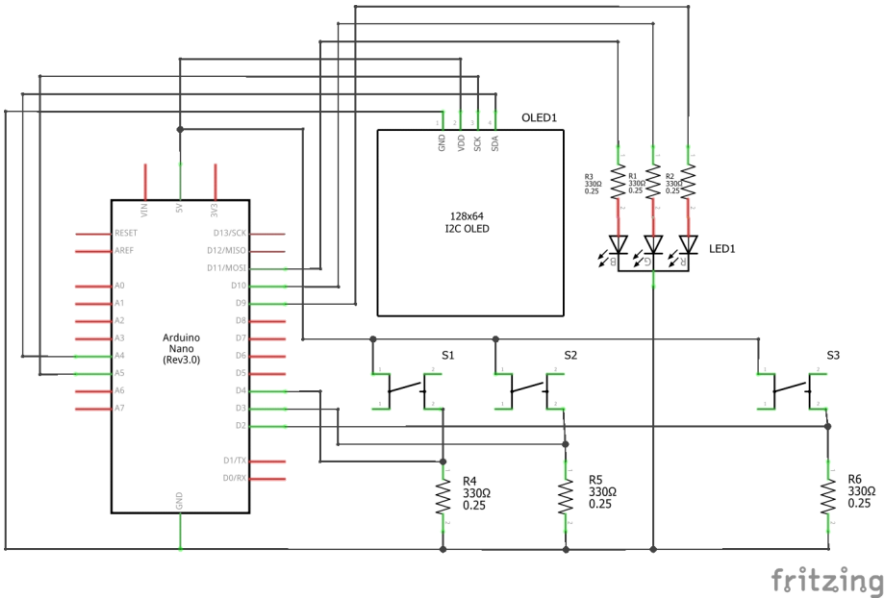


*Figura 30. Circuito eléctrico conexión Arduino Nano completo*

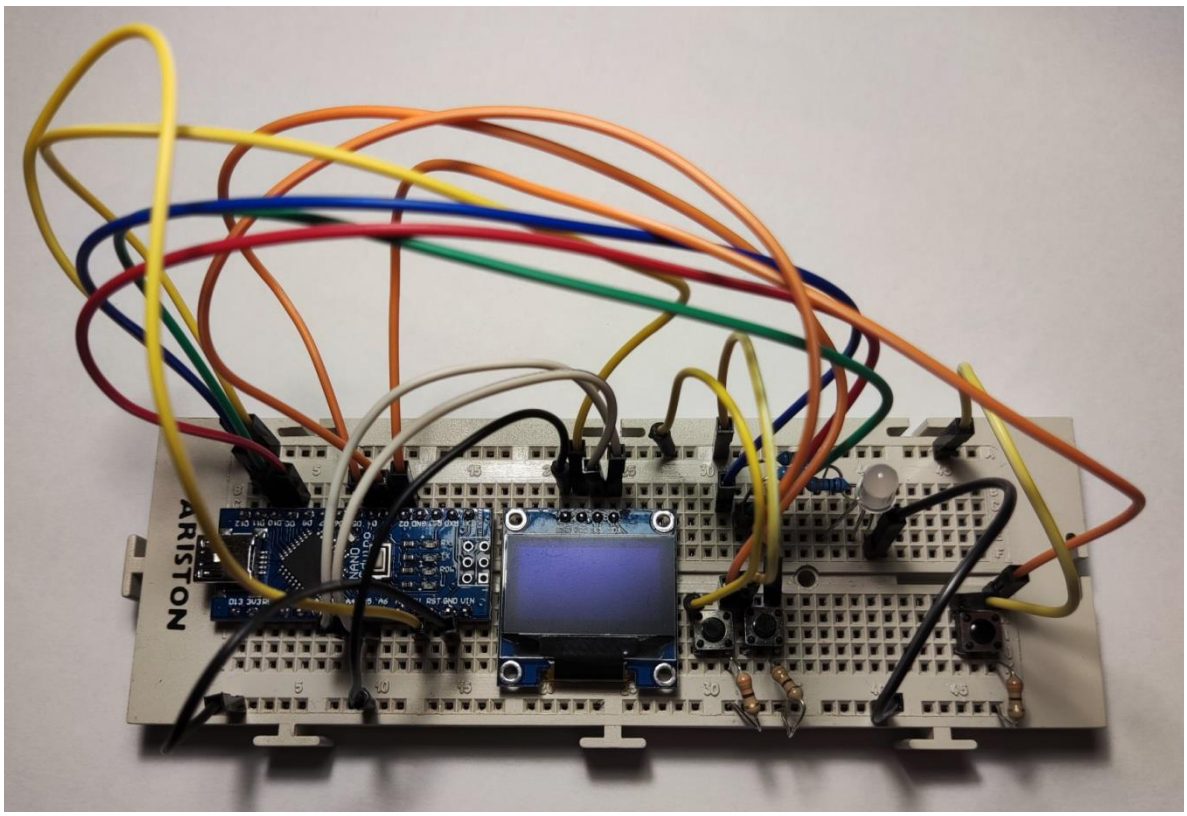


*Figura 31. Montaje circuito secuenciador*

# 4 Códigos programa

En esta sección se va a explicar cómo se ha realizado la implementación del código para llevar a cabo las distintas funciones de diseño. Se explica primero el funcionamiento del sistema desde punto de vista global mediante un diagrama de flujos de acciones, posteriormente se explica la estructura del sketch o proyecto. Y por último se intenta comentar la función *void setup()* y la función *loop().* Este último apartado se va a dividir en diferentes secciones para simplificar la explicación.

Cada una de las secciones se centra en una de las tareas del sistema. En cada sección se va a intentar explicar cuál es el objetivo que se quiere lograr y cómo se ha realizado, mostrando los fragmentos más relevantes del código cuando estos sean necesarios para la explicación.

## 4.1 Diagrama de flujo

El siguiente diagrama es una representación gráfica que muestra las variaciones y relaciones de una serie de acciones con un objetivo en común.

Una vez iniciado el sistema, se dan tres opciones para selección: calibrar la LED mediante asignación de la intensidad expresada en tanto por ciento a cada una de los LEDs rojo, verde y azul; reproducir una secuencia de LED preprogramada; o reprogramar una secuencia con una nueva combinación de colores, diferente tiempo de encendido y apagado, etc.

Mediante el cursor se puede seleccionar la opción deseada, y con la confirmación mediante apretar el pulsador OK se pasa a la rama de la opción escogida.

En la tarea de calibración, el sistema solicita al usuario los tres valores de intensidades de LED, una vez introducida, los valores serán guardados por la memoria EEPROM, se trata de una memoria no volátil, es decir, no se desaparece la información después de reiniciar el sistema. Y para comprobar la correcta lectura de los datos parpadea la LED en color blanco.

En la tarea de reproducción de secuencia, simplemente reproduce la secuencia de LEDs seleccionada, las secuencias son preprogramadas y debe ser leída de la EEPROM.

Y la última tarea consiste en programar la secuencia, después de seleccionar la secuencia objetivo, se introduce los datos mediante el monitor serie y serán guardados en la memoria no volátil para su recuperación la tarea de reproducción.

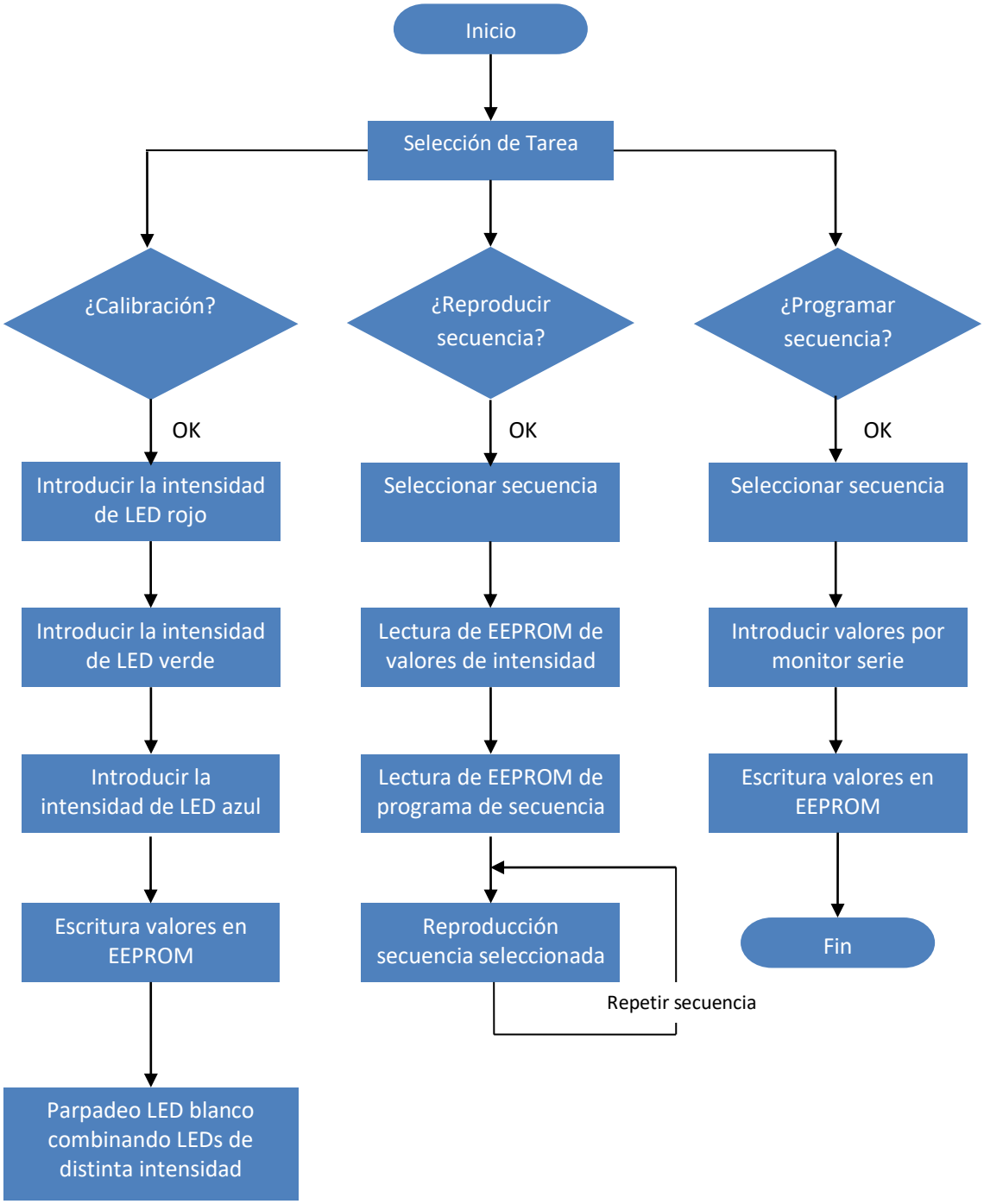


*Figura 32. Diagrama de acciones del sistema*

## 4.2 Estructura sketch

Un programa de Arduino se denomina sketch o proyecto y tiene la extensión .ino. La estructura básica de un sketch de Arduino es bastante simple y se compone de al menos tres partes. Estas tres partes son obligatorias y encierran bloques que contienen declaraciones, estamentos o instrucciones.

El programa empieza con la declaración de las variables y llamadas a librerías, las variables declaradas en esta sección son variables globales, es decir, puede ser empleada en cualquier función del programa.

setup() es la parte encargada de recoger la configuración y loop() es la que contiene el programa que se ejecuta cíclicamente (de ahí el término loop –bucle-). Ambas funciones son necesarias para que el programa trabaje.

La función de configuración (setup) debe contener la inicialización de los elementos y esta función sólo se ejecuta una vez justo después de hacer el reset y no se vuelve a ejecutar hasta que no haya otro reset.

La función bucle (loop) contiene el código que se ejecutará continuamente (lectura de entradas, activación de salidas, etc.). Esta función es el núcleo de todos los programas de Arduino y se usa para el control activo de la placa. La función loop se ejecuta justo después de setup.

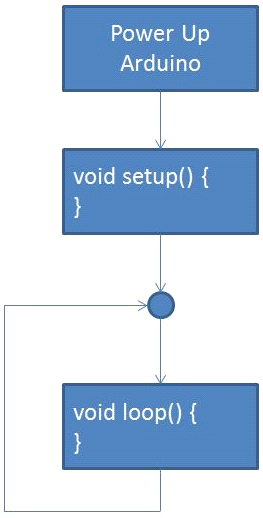


*Figura 33. Diagrama de acciones del Arduino*

## 4.3 Funciones básicas de Arduino

Una función es un conjunto de instrucciones que realizan una acción determinada. Esta función tendrá un nombre y podrá ser llamada desde otras partes del código tantas veces como sea necesario. Las funciones pueden recibir datos, denominados parámetros, que pueden ser manipulados en su interior. En este apartado se explicarán las funciones básicas de Arduino que se ha hecho uso en el sketch del trabajo.

Funciones de entrada/salida digital (I/O: Input/Output):

- **pinMode():** configura el pin especificado para actuar como entrada o salida digital. Sintaxis: *pinMode(pin, modo)*
- **digitalWrite():**activa (HIGH) o desactiva (LOW) un pin digital. Sintaxis: *digitalWrite(pin, valor)*
- **digitalRead()**: lee el valor (HIGH o LOW) del pin digital especificado. Sintaxis: *digitalRead(pin)*

Funciones de entrada/salida analógicas:

- **analogRead()**: lee el valor presente en el pin analógico especificado. Sintaxis: *analogRead(pin)*
- **analogWrite()**: escribe en el pin indicado un valor analógico (mediante un pulso cuya duración determina el valor analógico transmitido, también llamado PWM –"Pulse Width Modulation"). Sintaxis: *analogWrite(pin, valor)*

Funciones de tiempo:

- **millis()**: devuelve el número de milisegundos transcurridos desde que el Arduino comenzó a ejecutar el programa en curso. Sintaxis: *millis()*
- **delay()**: introduce una pausa en el programa de una duración especificada (en milisegundos) como parámetro. Sintaxis: *delay(ms)*

Funciones de comunicaciones en serie (serial):

- **available()**: devuelve el número de bytes (caracteres u octetos) disponibles para su lectura desde el puerto serie. Sintaxis: *Serial.available()*
- **print() y println():** manda por el puerto serial la información y lo imprime en el monitor serie. Sintaxis: *Serial.println(valor)* o también *Serial.println(valor, formato)*
- **read():** lee datos provenientes del puerto serie. Sintaxis: *Serial.read()*

Funciones de comunicaciones con display OLED:

- **display.setCursor():** sitúa el cursor en las coordenadas específicas de la pantalla del display. Sintaxis: *display.setCursor( valorX, valorY)*
- **display.print() y display.println():** escribe los datos en la pantalla. Sintaxis: *display.print(valor)*
- **display.clearDisplay():** Limpia el contenido del buffer de la pantalla, es decir, poner todos los píxeles a off. Sintaxis: *display.clearDisplay()*
- **display.display():** envía los datos a imprimir al RAM de la pantalla. Sintaxis: *display.display()*
- **display.drawLine():** dibuja una línea en el display. Sintaxis: *display.drawLine(x0, y0, x1, y1, color)*
- **display.drawCircle():** dibuja un círculo en el display. Sintaxis: *display.drawCircle(x0, y0, radio, color)*

Funciones de comunicación con memoria EEPROM:

- **EEPROM.put():** escribe datos en una dirección de la memoria EEPROM. Sintaxis: *EEPROM.put(dirección, valor)*
- **EEPROM.get():** lee datos de una dirección de la memoria EEPROM. Sintaxis: *EEPROM.get(dirección, valor)*

## 4.4 Función setup()

### 4.4.1 Iniciación pantalla

En el principio de la función setup iniciamos tanto el monitor serie como el display OLED, en caso de fallo, escribirá un mensaje de aviso en el monitor serie.

```
#ifdef __DEBUG__
    Serial.begin(9600);
    delay(100);
    Serial.println("Iniciando pantalla OLED");
    #endif

      if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C))
        {
        #ifdef __DEBUG__
        Serial.println("No se encuentra la pantalla OLED");
        #endif
        while (true);
        }
```

*Código 1. Iniciación pantalla*

### 4.4.2 Mensaje de inicio: título trabajo

Mediante un corto fragmento de código se escribe en la pantalla del display el nombre del proyecto, y se mantiene durante tres segundos mediante la función delay().

```
display.clearDisplay();
display.setTextSize(1);
display.setTextColor(SSD1306_WHITE);
display.setCursor(40, 20);
display.println(F("Circuito"));
display.println(F("  Secuenciador de un       Laser RGB"));
display.drawRect(2, 2, 126, 62, SSD1306_WHITE);
display.display();
delay(3000);
```

*Código 2. Escritura de título en display*

Cabe destacar que en todo el sketch se usa el macro *F()* para optimizar el consumo de memoria RAM, que es la memoria más limitada del microcontrolador Atmega328p (1 kB).

El uso de *F()* para cadenas constantes permite instruir al compilador para que almacene las cadenas constantes solo en FLASH(32 kB). De esta manera evitamos sobrecargar la función *display.print().*

### 4.4.3 Selección tarea

El objetivo de este código es mostrar en pantalla las tres opciones de tareas posibles, y mediante los dos pulsadores que sirven de cursor junto con el botón de confirmación (OK), poder desplazar un círculo en la pantalla, seleccionar la tarea deseada y dar confirmación para entrar en el siguiente paso.

Para ello se ha utilizado un bucle condicionado tipo while, que repite el código mientras el variable OK sea en estado LOW, como el variable se ha declarado valor 0 al principio del sketch, el bucle ejecuta al menos una vez. En cada bucle se leerá el valor de la entrada del pulsador OK, cuando detecta una señal HIGH (una pulsación en el botón OK) desde el pin de entrada conectado, incumple la condición y termina el bucle.

Al terminar el bucle, se obtendría el valor de la variable opción, que se limita en 0, 1 ó 2 que corresponde a las tres alternativas de opciones. Esto se consigue de la siguiente forma: en cada bucle leerá los inputs de los pulsadores centrales, si detecta un HIGH en el pulsador izquierdo el valor de opción disminuye en una unidad, y si lo detecta en el pulsador derecho el valor opción aumenta en una unidad.

```
while(OK==LOW)
        {
        valueLeft = digitalRead(inputLeftPin);
        valueRight = digitalRead(inputRightPin);
          if (valueLeft == HIGH)
            opcion = opcion - 1;

          if (valueRight == HIGH)
            opcion = opcion + 1;

          if (opcion < 0)
            opcion = 0;

          else if (opcion > 2)
            opcion = 2;

          else if (opcion == 0)
            {
            display.clearDisplay();
            display.setCursor(2, 8);
            display.println(F("MENU:"));
            display.drawLine(0, 20, display.width(), 20, SSD1306_WHITE);
            display.setCursor(0, 30);
            display.println(F(" 1 Calibracion"));
            display.setCursor(0, 40);
            display.println(F(" 2 Emision secuencia"));
            display.setCursor(0, 50);
            display.println(F(" 3 Reprogramar sec"));
            display.drawCircle(8, 33, 5, SSD1306_WHITE);
            display.display();
            }

          else if (opcion == 1)
            {

            […];

            display.drawCircle(8, 43, 5, SSD1306_WHITE);
            display.display();
            }

          else if (opcion == 2)
            {

            […];

            display.drawCircle(8, 53, 5, SSD1306_WHITE);
            display.display();
            }
        OK = digitalRead(inputOkPin);
        delay(100);
        }
    OK = 0;
```

*Código 3. Selección tarea*

Para que sea visual la selección, cada vez que el valor opción cambie, se cambiará las coordenadas de un pequeño círculo en la pantalla OLED, por lo que el segmento de código correspondientes al texto a mostrar en el display sería el mismo para las distintas opciones excepto la línea de `display.drawCircle(x, y, radio, color)`.

### 4.4.4 Opción 1: Calibración

El concepto de calibrar se refiere a que mediante configuración de la intensidad de cada una de los LEDs elementales, conseguir que la mezcla de colores sea más exacta (se pude emplear un colorímetro). Ya que en la práctica, un láser RGB puede ser empleados en maquinarias sofisticadas que requiere una mayor precisión en el color que emite.

#### 4.4.4.1 Configuración de intensidades

```
while(OK==LOW)
            {
            valueLeft = digitalRead(inputLeftPin);
            valueRight = digitalRead(inputRightPin);
              if (valueLeft == HIGH)
                intenR = intenR - 5;

              if (valueRight == HIGH)
                intenR = intenR + 5;

              if (intenR < 0)
                intenR = 0;

              else if (intenR > 100)
                intenR = 100;

              else
                {
                display.clearDisplay();
                display.setCursor(8, 20);
                display.drawLine(0, 40, display.width(), 40,
SSD1306_WHITE);
                display.println(F("Intensidad Led R:"));
                display.setCursor(50, 50);
                display.print(intenR);
                display.print(F("%"));
                display.display();
                Color(intenR*2.55, 0, 0);
                }

            OK = digitalRead(inputOkPin);
            delay(100);
            }
        OK = 0;
        Color(0, 0, 0);
```

*Código 4. Configuración intensidad LED rojo*

Este código tiene una estructura similar al código para seleccionar tareas, mediante lectura de valores provenientes de los pulsados centrales se le da el valor a la variable intenR, puede ir incrementando y disminuyendo de 5 en 5. Al mismo tiempo emite el comando de encender el LED rojo con la intensidad definida. La configuración termina con la lectura de un HIGH en el pin de input del pulsador OK.

El encendido de LED se lleva a cabo mediante la llamada a función *color(),* que está predefinida antes de la función setup, que sería el siguiente:

```
void Color(int R, int G, int B)
    {
    analogWrite(9 , R) ;   // Red    - Rojo
    analogWrite(10, G) ;   // Green - Verde
    analogWrite(11, B) ;   // Blue - Azul
    }
```

*Código 5. Función encendido LED*

Los valores 9, 10 y 11 corresponden a los pines de salida PWM, lo que significa que el voltaje de salida, que serían los valores R, G y B, puede ser de 0V a 5V en una escala de 0 a 255. Combinando los diferentes valores de R, G y B se puede emitir el color deseado.

Al terminar este segmento, se repite el mismo código para obtener intenG e intenB que corresponden a las intensidades de los LEDs verde y azul.

#### 4.4.4.2 Visualización de los valores de intensidades introducidas y guardado de datos

Una vez terminada la configuración de valores de intensidades se imprimirán en la pantalla mediante funciones de comunicación con display.

```
display.clearDisplay();
display.setCursor(14, 20);
display.print(F("Red"));
display.setCursor(14, 32);
display.print(intenR);
display.println(F("%"));
display.drawRect(12, 10, 30, 40, SSD1306_WHITE);

[…];
[…];

MyStruct intensidades =
  {
  intenR,
  intenG,
  intenB
  };

eeAddress = 1000;
EEPROM.put(eeAddress, intensidades);
```

*Código 6. Escritura datos en EEPROM*

La función EEPROM de la placa Arduino Nano tiene un tamaño de 1KB, que equivale a 1024 bytes, para la escritura del dato se necesita la dirección del byte, es decir, la posición de la memoria donde se almacena la información, 1024 bytes significa 1024 posiciones de almacenaje, en el caso del código de arriba, se ha escogido la posición 1000.

Se ha creado una estructura que recoge los tres variables llamado intensidades para que sea más fácil la escritura con la función *EEPROM.put().* También se puede escribir los variables de uno en uno con la función *EEPROM.write().*

### 4.4.5 Opción 2: Reproducción secuencia preprogramada

Como el nombre indica, este fragmento corresponde a la de tarea de reproducción de secuencias de LEDs, sin embargo, en este apartado se explica el fragmento dentro de la función setup, que consiste en la lectura de datos de intensidad almacenada y selección de secuencia, ya que se procesa solo una sola vez.

En el apartado correspondiente de la función loop se comentará el control de encendido y apagado de LEDs.

#### 4.4.5.1 Lectura de valores intensidad

En esta sección de código se lee los valores de intensidad guardadas en la EEPROM mediante función *EEPROM.get(),* como se ha programado, se ha guardado en la posición 1000 de la memoria. Como pasa con la escritura, también se puede leer dato por dato empleando la función *EEPROM.read().*

Posteriormente, se escribirá en pantalla del display los valores leídos. Aquí el código se ha simplificado para una intensidad.

```
if(opcion==1)
        {
        MyStruct intensidades;
        eeAddress = 1000;
        EEPROM.get(eeAddress, intensidades);
        intenR = intensidades.a;
        intenG = intensidades.b;
        intenB = intensidades.c;

        display.clearDisplay();
        display.setCursor(14, 20);
        display.print(F("Red"));
        display.setCursor(14, 32);
        display.print(intenR);
        display.println(F("%"));
```

```
display.drawRect(12, 10, 30, 40, SSD1306_WHITE);

[];
[];

}
```

*Código 7. Escritura datos en EEPROM*

#### 4.4.5.2 Selección de la secuencia a reproducir

La estructura de este fragmento es igual al de selección de tareas, por lo que no se verá con detalle, simplemente se ha sustituido la variable *opcion* por la variable *sec.*

### 4.4.6 Opción 3: Programar las secuencia vía monitor serie

Para que el dispositivo de diseño tenga un mayor ámbito de aplicación, conviene que las secuencias de láser puedan ser programables por el usuario, para tal fin, el sistema se ha diseñado para poder introducir datos por monitor serie creando secuencias nuevas.

El código empieza con la selección de secuencia a reprogramar, que está explicada en apartados anteriores, posteriormente se solicita los datos por monitor serie, y termina con la escritura de datos introducidos en la memoria no volátil.

```
          if (sec == 0)
            eeAddress=0;

          if (sec == 1)
            eeAddress=300;

          if (sec == 2)
            eeAddress=600;

        Serial.println(F("Introduce el número de emisiones en una
repetición de secuencia (Max 26):"));
          while (Serial.available()==0)
            {
            }
        NumEmis = Serial.parseInt();
        EEPROM.put(eeAddress, NumEmis);
        eeAddress += sizeof(int);
        Serial.print(F("Número de emisiones: "));
        Serial.println(NumEmis);

        Serial.println(F("Introduce el tiempo de reproducción de
secuencia(desde inicio de una secuencia hasta el inicio del siguiente
repetición):"));
```

```
          while (Serial.available()==0)
            {
            }

            int Treprod = Serial.parseInt();
            EEPROM.put(eeAddress, Treprod);
            eeAddress += sizeof(int);
            Serial.print(F("Tiempo de reproducción: "));
            Serial.println(Treprod);



          for (byte i=0; i<NumEmis; i++)
            {
            Serial.print(F("Introduce los valores del color "));
            Serial.println(i+1);
              while (Serial.available()==0)
                {
                }
            int Valor1 = Serial.parseInt();

              while (Serial.available()==0)
                {
                }
            int Valor2 = Serial.parseInt();

              while (Serial.available()==0)
                {
                }
            int Valor3 = Serial.parseInt();

            Serial.print(F("Color "));
            Serial.println(i+1);
            Serial.println(Valor1);
            Serial.println(Valor2);
            Serial.println(Valor3);

            Serial.print(F("Introduce los valores del tiempo on y tiempo
off del color "));
            Serial.println(i+1);
              while (Serial.available()==0)
                {
                }
            int Ton = Serial.parseInt();

              while (Serial.available()==0)
                {
                }
            int Toff = Serial.parseInt();

            Serial.print(F("Tiempo on del color "));
            Serial.println(i+1);
            Serial.println(Ton);
            Serial.print(F("Tiempo off del color "));
            Serial.println(i+1);
            Serial.println(Toff);
```

```
            Serial.print(F("Introduce el inicial del nombre del color
"));
            Serial.println(i+1);
              while (Serial.available()==0)
                {
                }
            Name = Serial.read();
            Serial.println(Name);

            ColorStruct MyColor =
              {
              Valor1,
              Valor2,
              Valor3,
              Ton,
              Toff,
              Name
              };
            EEPROM.put(eeAddress, MyColor);
            eeAddress += sizeof(ColorStruct);
            }
```

*Código 8. Reprogramación secuencia*

En el código de arriba, después de escribir en monitor serie mensajes de solicitación de datos, y antes de la función de lectura tipo serial se pone siempre un bucle vacío: `while (Serial.available()==0){}`, para que el sistema permanece sin acción mientras no detecta envío de dato desde puerto serie.

Los primeros valores solicitados son números de emisiones (NumEmis), una emisión consiste en un encendido y un apagado durante la secuencia; tiempo de reproducción (Trep) que sería el tiempo que transcurre desde el inicio de reproducción hasta el inicio de la siguiente repetición. Posteriormente se ha creado un bucle para leer información de cada una de las emisiones, que sería el parámetro de color(Valor1, 2 y 3); Tiempo que el led permanece encendida(Ton); tiempo que el led permanece apagado(Toff) y el inicial del nombre de color definido(Name).

Depende del tipo de variable, se usa un tipo de función de lectura serial distinto, para variables tipo int se emplea la función *Serial.parseInt(),* y para las de tipo char se usa *Serial.read().*

Después de cada lectura de dato de serial, se procede el almacenaje de los datos en la EEPROM, se ha creado una estructura que contiene los datos de cada emisión, y se almacena mediante la función `EEPROM.put(eeAddress, MyColor)` siendo *eeAddress* la posición de la memoria, y *MyColo*r la estructura que contiene datos de cada emisión.

El valor de *eeAddress* es un dato muy crítico en este apartado, ya que es fundamental saber en qué posición se encuentra cada dato. Se ha definido posición 0 como posición inicial para la secuencia 1; posición 300 para la secuencia 2 y posición 600 para la secuencia 3. Por lo cual para la secuencia 1 tiene reservada en la memoria 300 bytes de espacio para almacenaje (de posición 0 al posición 299), lo mismo para la secuencia 2 (300-599) y secuencia 3 (600-899), considerando el tamaño de los variables de cada emisión, se debe limitar el número de emisiones de una secuencia en 26.

Al empezar a grabar dato desde posición inicial, después de cada escritura de dato, hay que actualizar la posición de *eeAddress* para la siguiente escritura, esto se hace mediante sumando el tamaño del variable que se acaba de grabar al valor de posición antes de la escritura. Para saber el tamaño de las variables se utiliza la función *sizeof(tipo variable),* que retorna el valor de bytes que ocupa la variable.

## 4.5 Función loop()

### 4.5.1 Opción 1: Calibración

En este segmento simplemente llama la función color para parpadear una luz blanca combinando los LEDs RGB a las intensidades definidas con el fin de ver el resultado de calibración. En caso necesario se puede cambiar el color blanco por cualquier otro color para hacer medición con el colorímetro.

```
if(opcion==0)
   {
   intenR*=2.5;
   intenG*=2.5;
   intenB*=2.5;
   Color(intenR, intenG, intenB);
   }
```

*Código 9. Parpadeo LED color blanco*

### 4.5.2 Opción 2: Reproducción secuencia

La función de este último fragmento del sketch es para reproducir secuencias preprogramadas, empieza con la selección de secuencia, una vez obtenido el valor de sec se leerá todos los datos almacenados en la EEPROM tanto de las intensidades RGB guardados en las posiciones 1000, como los de dicha secuencia a reproducir, y envía comandos para el encendido y apagado de LEDs según definido previamente.

```
if(opcion==1)
   {
     if (sec == 0)
         eeAddress=0;

     if (sec == 1)
       eeAddress=300;

     if (sec == 2)
       eeAddress=600;

   display.clearDisplay();
   display.setCursor(2, 20);
   display.println(F("Secuencia Color:"));
   display.setCursor(10, 40);

   EEPROM.get(eeAddress, NumEmis);
   eeAddress+= sizeof(int);
   EEPROM.get(eeAddress, Treprod);
   eeAddress+= sizeof(int);
   ColorStruct MyColor;
     for (int i=0; i<NumEmis; i++)
       {
       T0 = millis();
       EEPROM.get(eeAddress, MyColor);
       Valor1 = MyColor.Valor1 * intenR * 0.1;
       Valor2 = MyColor.Valor2 * intenG * 0.1;
       Valor3 = MyColor.Valor3 * intenB * 0.1;
       Color (Valor1, Valor2, Valor3);

       Serial.print(MyColor.Name);
       Serial.print(F(" - "));
       display.print(MyColor.Name);
       display.print(F(" - "));
       display.display();
       delay(MyColor.Ton);
       Color (0, 0, 0);
       delay(MyColor.Toff);
       eeAddress += sizeof(ColorStruct);
       }
   Tfin = millis();
   Tdelay = Treprod - (Tfin - T0);
   delay(Tdelay);
   }
```

*Código 10. Reproducción secuencia*

Se ha definido T0 al tiempo transcurrido desde el comienzo de reset del Arduino hasta el inicio de parpadeo de los LEDs de la secuencia, y el Tfin al tiempo desde reset hasta cuando termina la última emisión de la secuencia, restando el segundo al primero se obtiene el tiempo de la secuencia en emisión. Y restando el tiempo de reproducción a éste se obtiene el tiempo que hay que mantener apagado los LEDs hasta la siguiente repetición en milisegundos.

# 5 Manual de usuario

## 5.1 Esquema sistema circuito secuenciador

Para ilustrar la funcionalidad del módulo del circuito secuenciador se presenta el esquema del montaje.

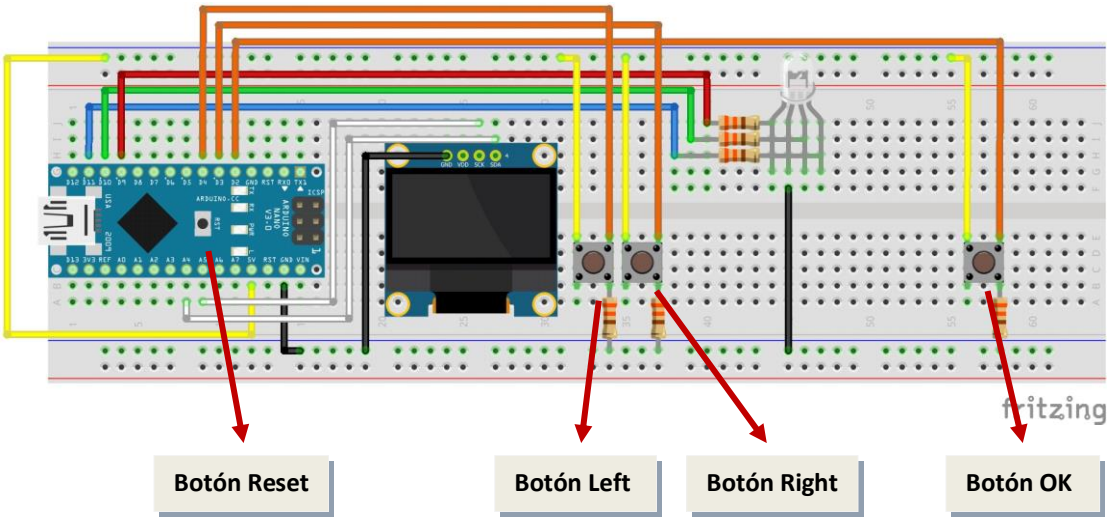


*Figura 32. Esquema módulo circuito secuenciador*

## 5.2 Selección opción tarea

1. Al conectar el módulo a la fuente de alimentación, se enciende el display mostrando el nombre del dispositivo. A tres segundos pasa al panel de menú.
2. En panel de menú, utilizar los botones “Left” y “right” para mover el cursor a la tarea deseada.

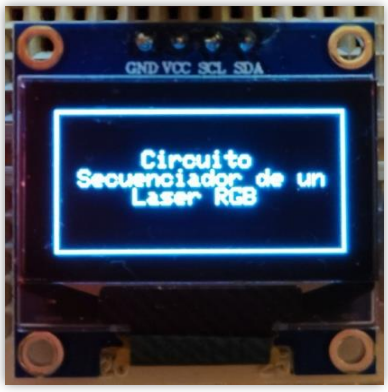


*Figura 33. Inicio display*


*Figura 34. Panel menú*

3. Pulsar botón “OK” para confirmar la selección.

## 5.3 Calibración

1. Muestra en el display el mensaje de solicitación de dato de intensidad.

2. Mediante los botones “Left” y “Right” para fijar el valor de intensidad desde 0% al 100%, cuando presiona “Left” el valor disminuye, y cuando se presiona “Right” el valor aumenta.

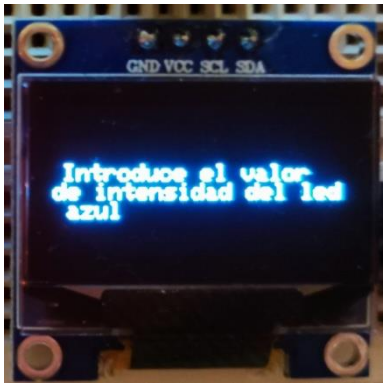


*Figura 35. Panel intensidad*


*Figura 36. Panel introducción datos*

3. Presionar el botón “OK” para confirmar la configuración del valor de intensidad.

4. Repetir el mismo proceso hasta tener los tres valores de intensidad del LED rojo, LED verde y LED azul.

5. El display muestra los valores introducidos.

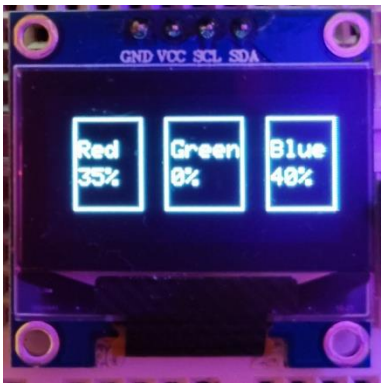


*Figura 37. Panel valores intensidad*

## 5.4 Reproducción de secuencia

1. Usar los botones “Left” y “Right” para seleccionar la secuencia a reproducir.


*Figura 38. Display selección secuencia*

2. Pulsar botón “OK” para confirmar la selección.
3. El dispositivo reproduce la secuencia seleccionada y muestra en pantalla el color de la LED emitida.


*Figura 39. Display secuencia color*

## 5.5 Reprogramación de secuencia

1. Usar los botones “Left” y “Right” para seleccionar la secuencia a reproducir.
2. Pulsar botón “OK” para confirmar la selección.
3. Introducir valor que se solicita el monitor serie del software IDE hasta que la pantalla de monitor escriba el texto de “proceso completo”.

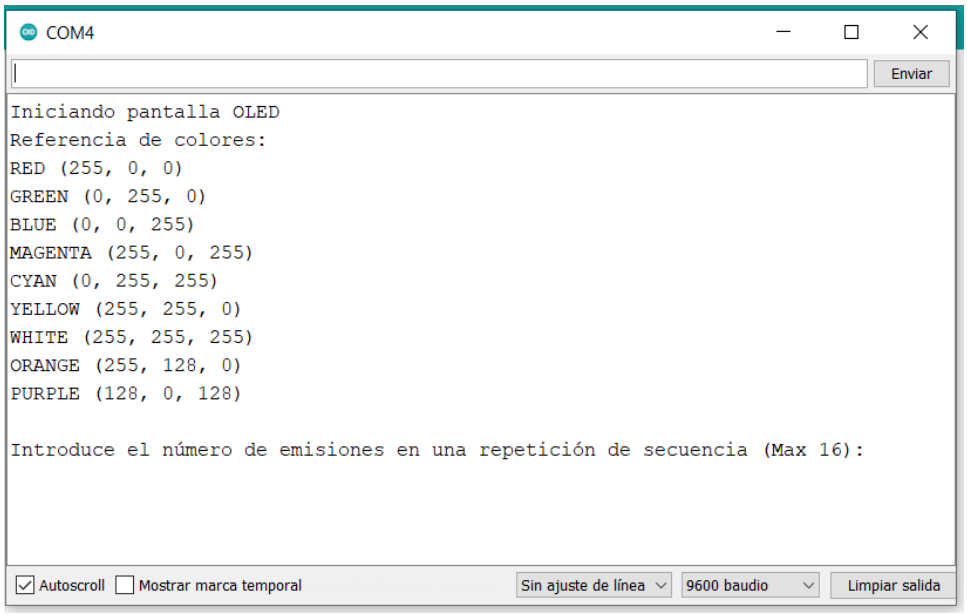


*Figura 40. Interfaz monitor serie Arduino IDE*

4. El display de OLED indica el finalizo de proceso.
5. Pulsar botón “reset” para volver al menú principal.

# 6 Conclusiones y trabajos futuros

## 6.1 Conclusiones

Una vez concluido el trabajo, puede decirse que los objetivos iniciales planteados han sido resueltos satisfactoriamente, y se pueden aportar las siguientes conclusiones:

- Se ha diseñado el circuito electrónico que asimila a un módulo de láser RGB que permite configurar los parámetros pertinentes.
- Se ha desarrollado el Software controlador del circuito anterior, de forma que ambos en conjunto forman la unidad de control y permiten comandar el láser para crear la secuencia de colores deseada por el usuario. Este Software cumple con todos los requisitos demandados en el planteamiento del mismo.
- La validación del programa ha resultado exitosa, y se ha comprobado que la repetitividad del mismo es muy buena.

El desarrollo del presente Trabajo fin de Grados resulta ser una oportunidad para poner en práctica los conocimientos sobre electrónica e informática adquiridos durante la carrera en la solución de una necesidad real.

Durante el proceso del diseño, instalación del circuito y el posterior proceso de codificación del programa se ha entendido mejor el funcionamiento de un sistema electrónico, de cómo un sistema toma las señales del mundo físico y las convierten en corriente o voltaje (Inputs) y también de cómo convierte la corriente o voltaje en señales físicamente útiles (Outputs) mediante procesadores y el lenguaje informático.

Mediante la realización del proyecto se ha aprendido la configuración del hardware del Arduino Nano, se ha comprendido la estructura básica de sketch de programación Arduino y los fundamentos de su lenguaje de programación, trabajándose con las entradas I/O digitales y analógicas del mismo y su puerto serie.

Y también se ha visto la amplia posibilidad de aplicación de la placa Arduino, conectando los diferentes tipos de periféricos a la placa, puede desempeñar múltiples funciones, por lo que es ideal para desarrollar elementos autóctonos. Por lo que en mi futura carrera profesional como ingeniera sería una alternativa para satisfacer ciertas necesidades en proyectos creativos.

## 6.2 Trabajos futuros

Concluido el Trabajo Fin de Grado, se detecta la necesidad inherente de realizar unos trabajos futuros de cara a poner en pleno funcionamiento definitivo el conjunto desarrollado. Los más destacados son:

Desde perspectiva estética, se puede buscar un receptáculo adecuado y estético para el conjunto de la placa Arduino Nano y sus extensiones, de tal forma que quede encapsulado, dejando únicamente el conector mini USB, y unas pequeñas ventanas para mantener a los pulsadores y el botón reset de la placa accesibles al usuario.

Otra manera de conseguir una mejor presentación del conjunto sería una impresión del circuito en una placa PBC, de esta manera se pretende reducir la dimensión del dispositivo, y también le daría más estabilidad y resistencia al sistema.

Y en cuanto al aspecto funcional se sustituiría la conexión de la placa Arduino con el LED RGB por conexión de un láser RGB para el uso real, ya que el LED sirve para simular el funcionamiento del láser en el montaje prototipo.

# ANEXO: CÓDIGO SKETCH COMPLETO

```
#define __DEBUG__

#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <EEPROM.h>

#define ANCHO_PANTALLA 128
#define ALTO_PANTALLA 64
#define DEBUG(a) Serial.println(a);

// Objeto de la clase Adafruit_SSD1306
Adafruit_SSD1306 display(ANCHO_PANTALLA, ALTO_PANTALLA, &Wire, -1);

  const int inputLeftPin = 4;
  const int inputRightPin = 3;
  const int inputOkPin = 2;

  byte OK = 0;
  byte valueLeft = 0;
  byte valueRight = 0;
  byte intenR = 0;
  byte intenG = 0;
  byte intenB = 0;
  byte opcion = 0;
  int eeAddress = 0;
  int NumEmis;
  byte sec = 0;
  int Valor1, Valor2, Valor3;
  int Ton, Toff, Treprod, T0, Tfin, Tdelay;
  char Name;


  struct MyStruct
    {
    byte a;
    byte b;
    byte c;
    };

   struct ColorStruct
    {
    int Valor1;
    int Valor2;
    int Valor3;
    int Ton;
    int Toff;
    char Name;
    };
```

```
void Color(int R, int G, int B)
    {
    analogWrite(9 , R) ;   // Red    - Rojo
    analogWrite(10, G) ;   // Green - Verde
    analogWrite(11, B) ;   // Blue - Azul
    }

void setup()
    {
    #ifdef __DEBUG__
    Serial.begin(9600);
    delay(100);
    Serial.println("Iniciando pantalla OLED");
    #endif

    // Iniciar pantalla OLED en la dirección 0x3C
      if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C))
        {
        #ifdef __DEBUG__
        Serial.println("No se encuentra la pantalla OLED");
        #endif
        while (true);
        }

    pinMode(inputLeftPin, INPUT);
    pinMode(inputRightPin, INPUT);
    pinMode(inputOkPin, INPUT);

      for (int i =9 ; i<12 ; i++)
        pinMode(i, OUTPUT);

    //Título de trabajo
    display.clearDisplay();
    display.setTextSize(1);
    display.setTextColor(SSD1306_WHITE);
    display.setCursor(40, 20);
    display.println(F("Circuito"));
    display.println(F("  Secuenciador de un       Laser RGB"));
    display.drawRect(2, 2, 126, 62, SSD1306_WHITE);
    display.display();
    delay(3000);

    //Menú opciones
       while(OK==LOW)
        {
        valueLeft = digitalRead(inputLeftPin);
        valueRight = digitalRead(inputRightPin);
          if (valueLeft == HIGH)
            opcion = opcion - 1;

          if (valueRight == HIGH)
            opcion = opcion + 1;

          if (opcion < 0)
            opcion = 0;

          else if (opcion > 2)
```

```
          opcion = 2;

        else if (opcion == 0)
          {
          display.clearDisplay();
          display.setCursor(2, 8);
          display.println(F("MENU:"));
          display.drawLine(0, 20, display.width(), 20, SSD1306_WHITE);
          display.setCursor(0, 30);
          display.println(F(" 1 Calibracion"));
          display.setCursor(0, 40);
          display.println(F(" 2 Emision secuencia"));
          display.setCursor(0, 50);
          display.println(F(" 3 Reprogramar sec"));
          display.drawCircle(8, 33, 5, SSD1306_WHITE);
          display.display();
          }

        else if (opcion == 1)
          {
          display.clearDisplay();
          display.setCursor(2, 8);
          display.println(F("MENU:"));
          display.drawLine(0, 20, display.width(), 20, SSD1306_WHITE);
          display.setCursor(0, 30);
          display.println(F(" 1 Calibracion"));
          display.setCursor(0, 40);
          display.println(F(" 2 Emision secuencia"));
          display.setCursor(0, 50);
          display.println(F(" 3 Reprogramar sec"));
          display.drawCircle(8, 43, 5, SSD1306_WHITE);
          display.display();
          }

        else if (opcion == 2)
          {
          display.clearDisplay();
          display.setCursor(2, 8);
          display.println(F("MENU:"));
          display.drawLine(0, 20, display.width(), 20, SSD1306_WHITE);
          display.setCursor(0, 30);
          display.println(F(" 1 Calibracion"));
          display.setCursor(0, 40);
          display.println(F(" 2 Emision secuencia"));
          display.setCursor(0, 50);
          display.println(F(" 3 Reprogramar sec"));
          display.drawCircle(8, 53, 5, SSD1306_WHITE);
          display.display();
          }
      OK = digitalRead(inputOkPin);
      delay(100);
      }
  OK = 0;

    if(opcion==0)
      {
      display.clearDisplay();
```

```
        display.setCursor(4, 25);
        display.println(F("Introduce el valor  de intensidad del led
rojo"));
        display.display();
        delay(2000);

        // Selección de intensidad del led roja
          while(OK==LOW)
            {
            valueLeft = digitalRead(inputLeftPin);
            valueRight = digitalRead(inputRightPin);
              if (valueLeft == HIGH)
                intenR = intenR - 5;

              if (valueRight == HIGH)
                intenR = intenR + 5;

              if (intenR < 0)
                intenR = 0;

              else if (intenR > 100)
                intenR = 100;

              else
                {
                display.clearDisplay();
                display.setCursor(8, 20);
                display.drawLine(0, 40, display.width(), 40,
SSD1306_WHITE);
                display.println(F("Intensidad Led R:"));
                display.setCursor(50, 50);
                display.print(intenR);
                display.print(F("%"));
                display.display();
                Color(intenR*2.55, 0, 0);
                }

            OK = digitalRead(inputOkPin);
            delay(100);
            }
        OK = 0;
        Color(0, 0, 0);

        // Selección de intensidad del led verde
        display.clearDisplay();
        display.setCursor(4, 25);
        display.println(F("Introduce el valor  de intensidad del led
verde"));
        display.display();
        delay(2000);

          while(OK==LOW)
            {
            valueLeft = digitalRead(inputLeftPin);
            valueRight = digitalRead(inputRightPin);
              if (valueLeft == HIGH)
                intenG = intenG - 5;
```

```
              if (valueRight == HIGH)
                intenG = intenG + 5;

              if (intenG < 0)
                intenG = 0;

              else if (intenG > 100)
                intenG = 100;

              else
                {
                display.clearDisplay();
                display.setCursor(8, 20);
                display.drawLine(0, 40, display.width(), 40,
SSD1306_WHITE);
                display.println(F("Intensidad Led G:"));
                display.setCursor(50, 50);
                display.print(intenG);
                display.print(F("%"));
                display.display();
                Color(0, intenG*2.55, 0);
                }

            OK = digitalRead(inputOkPin);
            delay(100);
            }
        OK = 0;
        Color(0, 0, 0);

        // Selección de intensidad del led azul
        display.clearDisplay();
        display.setCursor(4, 25);
        display.println(F("Introduce el valor  de intensidad del led
azul"));
        display.display();
        delay(2000);

          while(OK==LOW)
            {
            valueLeft = digitalRead(inputLeftPin);
            valueRight = digitalRead(inputRightPin);
              if (valueLeft == HIGH)
                intenB = intenB - 5;

              if (valueRight == HIGH)
                intenB = intenB + 5;

              if (intenB < 0)
                intenB = 0;

              else if (intenB > 100)
                intenB = 100;

              else
                {
                display.clearDisplay();
```

```
                display.setCursor(8, 20);
                display.drawLine(0, 40, display.width(), 40,
SSD1306_WHITE);
                display.println(F("Intensidad Led B:"));
                display.setCursor(50, 50);
                display.print(intenB);
                display.print(F("%"));
                display.display();
                Color(0, 0, intenB*2.55);
                }

            OK = digitalRead(inputOkPin);
            delay(100);
            }
        OK = 0;
        Color(0, 0, 0);

        display.clearDisplay();
        display.setCursor(14, 20);
        display.print(F("Red"));
        display.setCursor(14, 32);
        display.print(intenR);
        display.println(F("%"));
        display.drawRect(12, 10, 30, 40, SSD1306_WHITE);

        display.setCursor(55, 20);
        display.print(F("Green"));
        display.setCursor(55, 32);
        display.print(intenG);
        display.println(F("%"));
        display.drawRect(52, 10, 34, 40, SSD1306_WHITE);

        display.setCursor(98, 20);
        display.print(F("Blue"));
        display.setCursor(98, 32);
        display.print(intenB);
        display.println(F("%"));
        display.drawRect(96, 10, 30, 40, SSD1306_WHITE);
        display.display();

        MyStruct intensidades =
          {
          intenR,
          intenG,
          intenB
          };

        eeAddress = 1000;
        EEPROM.put(eeAddress, intensidades);
        }

      if(opcion==1)
        {
        MyStruct intensidades;
        eeAddress = 1000;
        EEPROM.get(eeAddress, intensidades);
        intenR = intensidades.a;
```

```
        intenG = intensidades.b;
        intenB = intensidades.c;

        display.clearDisplay();
        display.setCursor(14, 20);
        display.print(F("Red"));
        display.setCursor(14, 32);
        display.print(intenR);
        display.println(F("%"));
        display.drawRect(12, 10, 30, 40, SSD1306_WHITE);

        display.setCursor(55, 20);
        display.print(F("Green"));
        display.setCursor(55, 32);
        display.print(intenG);
        display.println(F("%"));
        display.drawRect(52, 10, 34, 40, SSD1306_WHITE);

        display.setCursor(98, 20);
        display.print(F("Blue"));
        display.setCursor(98, 32);
        display.print(intenB);
        display.println(F("%"));
        display.drawRect(96, 10, 30, 40, SSD1306_WHITE);
        display.display();
        delay(2000);

        //Menú seleccion secuencia a reproducir
          while(OK==LOW)
            {
            valueLeft = digitalRead(inputLeftPin);
            valueRight = digitalRead(inputRightPin);
              if (valueLeft == HIGH)
                sec = sec - 1;

              if (valueRight == HIGH)
                sec = sec + 1;

              if (sec < 0)
                sec = 0;

              else if (sec > 2)
                sec = 2;

              else if (sec == 0)
                {
                display.clearDisplay();
                display.setCursor(2, 8);
                display.println(F("Elige la sec a emitir"));
                display.drawLine(0, 20, display.width(), 20,
SSD1306_WHITE);
                display.setCursor(0, 30);
                display.println(F(" 1 Secuencia A"));
                display.setCursor(0, 40);
                display.println(F(" 2 secuencia B"));
                display.setCursor(0, 50);
                display.println(F(" 3 Secuencia C"));
```

```
                display.drawCircle(8, 33, 5, SSD1306_WHITE);
                display.display();
                }

              else if (sec == 1)
                {
                display.clearDisplay();
                display.setCursor(2, 8);
                display.println(F("Elige la sec a emitir"));
                display.drawLine(0, 20, display.width(), 20,
SSD1306_WHITE);
                display.setCursor(0, 30);
                display.println(F(" 1 Secuencia A"));
                display.setCursor(0, 40);
                display.println(F(" 2 secuencia B"));
                display.setCursor(0, 50);
                display.println(F(" 3 Secuencia C"));
                display.drawCircle(8, 43, 5, SSD1306_WHITE);
                display.display();
                }

              else if (sec == 2)
                {
                display.clearDisplay();
                display.setCursor(2, 8);
                display.println(F("Elige la sec a emitir"));
                display.drawLine(0, 20, display.width(), 20,
SSD1306_WHITE);
                display.setCursor(0, 30);
                display.println(F(" 1 Secuencia A"));
                display.setCursor(0, 40);
                display.println(F(" 2 secuencia B"));
                display.setCursor(0, 50);
                display.println(F(" 3 Secuencia C"));
                display.drawCircle(8, 53, 5, SSD1306_WHITE);
                display.display();
                }
            OK = digitalRead(inputOkPin);
            delay(100);
            }
        OK = 0;

        }

      if(opcion==2)
        {
        //Menú seleccion secuencia a reprogramar
          while(OK==LOW)
            {
            valueLeft = digitalRead(inputLeftPin);
            valueRight = digitalRead(inputRightPin);
              if (valueLeft == HIGH)
                sec = sec - 1;

              if (valueRight == HIGH)
                sec = sec + 1;
```

```
              if (sec < 0)
                sec = 0;

              else if (sec > 2)
                sec = 2;

              else if (sec == 0)
                {
                display.clearDisplay();
                display.setCursor(2, 8);
                display.println(F("Elige la secuencia:"));
                display.drawLine(0, 20, display.width(), 20,
SSD1306_WHITE);
                display.setCursor(0, 30);
                display.println(F(" 1 Secuencia A"));
                display.setCursor(0, 40);
                display.println(F(" 2 secuencia B"));
                display.setCursor(0, 50);
                display.println(F(" 3 Secuencia C"));
                display.drawCircle(8, 33, 5, SSD1306_WHITE);
                display.display();
                }

              else if (sec == 1)
                {
                display.clearDisplay();
                display.setCursor(2, 8);
                display.println(F("Elige la secuencia:"));
                display.drawLine(0, 20, display.width(), 20,
SSD1306_WHITE);
                display.setCursor(0, 30);
                display.println(F(" 1 Secuencia A"));
                display.setCursor(0, 40);
                display.println(F(" 2 secuencia B"));
                display.setCursor(0, 50);
                display.println(F(" 3 Secuencia C"));
                display.drawCircle(8, 43, 5, SSD1306_WHITE);
                display.display();
                }

              else if (sec == 2)
                {
                display.clearDisplay();
                display.setCursor(2, 8);
                display.println(F("Elige la secuencia:"));
                display.drawLine(0, 20, display.width(), 20,
SSD1306_WHITE);
                display.setCursor(0, 30);
                display.println(F(" 1 Secuencia A"));
                display.setCursor(0, 40);
                display.println(F(" 2 secuencia B"));
                display.setCursor(0, 50);
                display.println(F(" 3 Secuencia C"));
                display.drawCircle(8, 53, 5, SSD1306_WHITE);
                display.display();
                }
            OK = digitalRead(inputOkPin);
```

```
            delay(100);
            }
        OK = 0;
        display.clearDisplay();
        display.setCursor(2, 20);
        display.println(F("Introduce los valores mediante monitor
serie"));
        display.display();

        Serial.println(F("Referencia de colores:"));
        Serial.println(F("RED (255, 0, 0)"));
        Serial.println(F("GREEN (0, 255, 0)"));
        Serial.println(F("BLUE (0, 0, 255)"));
        Serial.println(F("MAGENTA (255, 0, 255)"));
        Serial.println(F("CYAN (0, 255, 255)"));
        Serial.println(F("YELLOW (255, 255, 0)"));
        Serial.println(F("WHITE (255, 255, 255)"));
        Serial.println(F("ORANGE (255, 128, 0)"));
        Serial.println(F("PURPLE (128, 0, 128)"));
        Serial.println(F(" "));

          if (sec == 0)
            eeAddress=0;

          if (sec == 1)
            eeAddress=300;

          if (sec == 2)
            eeAddress=600;

        Serial.println(F("Introduce el número de emisiones en una
repetición de secuencia (Max 16):"));
          while (Serial.available()==0)
            {
            }
        NumEmis = Serial.parseInt();
        EEPROM.put(eeAddress, NumEmis);
        eeAddress += sizeof(int);
        Serial.print(F("Número de emisiones: "));
        Serial.println(NumEmis);

        Serial.println(F("Introduce el tiempo de reproducción de
secuencia(desde inicio de una secuencia hasta el inicio del siguiente
repetición):"));
          while (Serial.available()==0)
            {
            }

            int Treprod = Serial.parseInt();
            EEPROM.put(eeAddress, Treprod);
            eeAddress += sizeof(int);
            Serial.print(F("Tiempo de reproducción: "));
            Serial.println(Treprod);



          for (byte i=0; i<NumEmis; i++)
```

```
            {
            Serial.print(F("Introduce los valores del color "));
            Serial.println(i+1);
              while (Serial.available()==0)
                {
                }
            int Valor1 = Serial.parseInt();

              while (Serial.available()==0)
                {
                }
            int Valor2 = Serial.parseInt();

              while (Serial.available()==0)
                {
                }
            int Valor3 = Serial.parseInt();

            Serial.print(F("Color "));
            Serial.println(i+1);
            Serial.println(Valor1);
            Serial.println(Valor2);
            Serial.println(Valor3);

            Serial.print(F("Introduce los valores del tiempo on y tiempo
off del color "));
            Serial.println(i+1);
              while (Serial.available()==0)
                {
                }
            int Ton = Serial.parseInt();

              while (Serial.available()==0)
                {
                }
            int Toff = Serial.parseInt();

            Serial.print(F("Tiempo on del color "));
            Serial.println(i+1);
            Serial.println(Ton);
            Serial.print(F("Tiempo off del color "));
            Serial.println(i+1);
            Serial.println(Toff);

            Serial.print(F("Introduce el inicial del nombre del color
"));
            Serial.println(i+1);
              while (Serial.available()==0)
                {
                }
            Name = Serial.read();
            Serial.println(Name);

            ColorStruct MyColor =
              {
              Valor1,
              Valor2,
```

```
              Valor3,
              Ton,
              Toff,
              Name
              };
            EEPROM.put(eeAddress, MyColor);
            eeAddress += sizeof(ColorStruct);
            }
            Serial.println(F("Proceso completo"));
            display.clearDisplay();
            display.setCursor(8, 20);
            display.println(F("Proceso completo"));
            display.display();
        }

    }

void loop()
    {
      if(opcion==0)
        {
        intenR*=2.5;
        intenG*=2.5;
        intenB*=2.5;
        Color(intenR, intenG, intenB);
        }

      if(opcion==1)
        {
          if (sec == 0)
              eeAddress=0;

          if (sec == 1)
            eeAddress=300;

          if (sec == 2)
            eeAddress=600;

        display.clearDisplay();
        display.setCursor(2, 20);
        display.println(F("Secuencia Color:"));
        display.setCursor(10, 40);

        EEPROM.get(eeAddress, NumEmis);
        eeAddress+= sizeof(int);
        EEPROM.get(eeAddress, Treprod);
        eeAddress+= sizeof(int);
        ColorStruct MyColor;
          for (int i=0; i<NumEmis; i++)
            {
            T0 = millis();
            EEPROM.get(eeAddress, MyColor);
            Valor1 = MyColor.Valor1 * intenR * 0.1;
            Valor2 = MyColor.Valor2 * intenG * 0.1;
            Valor3 = MyColor.Valor3 * intenB * 0.1;
            Color (Valor1, Valor2, Valor3);
```

```
            Serial.print(MyColor.Name);
            Serial.print(F(" - "));
            display.print(MyColor.Name);
            display.print(F(" - "));
            display.display();
            delay(MyColor.Ton);
            Color (0, 0, 0);
            delay(MyColor.Toff);
            eeAddress += sizeof(ColorStruct);
            }
        Tfin = millis();
        Tdelay = Treprod - (Tfin - T0);
        delay(Tdelay);
        }

}
```

# Bibliografía


[1] *Secuenciación del ADN*. <https://es.khanacademy.org/science/ap-biology/gene-expression-and-regulation/biotechnology/a/dna-sequencing>

[2] *Holografía*. <https://es.wikipedia.org/wiki/Holograf%C3%ADa>

[3] Samuel Verdú, Alberto J. Pérez, José M. Barat, Raúl Grau. *Laser backscattering imaging as a control technique for fluid foods: Application to vegetable-based creams processing.* Journal of Food Engineering (2019)

[4] Arduino Nano. <https://store.arduino.cc/products/arduino-nano>

[5] Luis del Valle Hernández. *SSD1306 pantalla OLED con Arduino y ESP8266 I2C.* <https://programarfacil.com/blog/arduino-blog/ssd1306-pantalla-oled-con-arduino/>

[6] *Language Reference*. https://www.arduino.cc/reference/en/

[7] José Guerra Carmenate. *Arduino IDE entorno de desarrollo oficial.* <https://programarfacil.com/blog/arduino-blog/arduino-ide/#Funciones_principales_de_Arduino_IDE>

[8] Luis Llamas. *Leer un pulsador con Arduino* (2014) <https://www.luisllamas.es/leer-un-pulsador-con-arduino/>

# Documento II

# Presupuesto

# Capítulo 1 Presupuesto

## 1.1 General

En el siguiente documento se muestran los costes derivados de la realización del presente trabajo; en él, se presentan los gastos de material, de software, de personal y de oficina. Los precios aplicados se corresponden con las tarifas legales vigentes y realizando estimaciones coherentes en caso de que haya sido necesario.

El coste de amortización se calcula como:

$$a = \frac{VC - VR}{n} \tag{1.1}$$

$$th = \frac{a}{\mathrm{h}} \tag{1.2}$$

Donde:

- $a$: amortización en euros/año.
- $VC$: valor de compra.
- $VR$: valor residual al cabo del periodo de amortización.
- $n$: periodo de amortización.
- $th$ : tasa horaria en euros/hora.
- $h$: horas trabajadas a lo largo del año.

Los precios unitarios correspondientes al personal se calculan de la siguiente manera:

$$\frac{horas\ trabajadas}{año} = \left(\frac{sem}{año} - \frac{sem\ vac/fest}{año}\right)\left(\frac{horas\ trabajadas}{sem}\right) \tag{1.3}$$

$$Coste\ horario = \frac{salario\ bruto\ anual}{horas\ trabajadas/año} \tag{1.4}$$

## 1.2 Recursos empleados

Se definen a continuación los distintos recursos empleados para la correcta realización de este trabajo. Una vez se tengan claros estos elementos, podrá procederse a calcular el precio unitario de cada uno para posteriormente calcular el presupuesto total del trabajo.

- **Recursos informáticos**

  Para la realización del trabajo se ha requerido el uso de equipos informáticos y recursos varios, para así poder llevar a cabo el diseño del Hardware y el desarrollo del Software entre otros.

| Descripción | Unidades |
|---|---|
| Ordenador portátil | 1 |
| Licencia Microsoft Office | 1 |
| Licencia Arduino IDE | 1 |

- **Recursos de personal**

| Descripción | Unidades |
|---|---|
| Ingeniero técnico en practica | 1 |
| Profesor Titular de Universidad | 1 |

- **Recursos materiales**

  Para la construcción del Hardware, se han empleado una serie de materiales que se recogen en este apartado.

| Descripción | Unidades |
|---|---|
| Placa Arduino Nano | 1 |
| Pantalla OLED 0,96" | 1 |
| LED RGB | 1 |
| Cable de acoplamiento | 16 |
| Pulsador | 3 |
| Resisitencia 330 Ω | 6 |
| Cable mini USB | 1 |

- **Recurso de instalaciones**

| Descripción | Unidades |
|---|---|
| Oficina | 1 |
| Permiso de oficina | 1 |
| Seguro de oficina | 1 |

## 1.3 Desglose de costes unitarios

A continuación se presenta un desglose de los precios unitarios, amortización y tasa horaria de los recursos mostrados en el apartado anterior.

### 1.3.1 Coste de material y software

- Equipo informático.
  - Portátil: se estima un valor residual del 10% y un periode de amortización de 5 años.

$$a = \frac{600-60}{5} = 108\ €/año \qquad (1.5)$$

$$th = \frac{108}{1840} = 0{,}059\ €/hora \qquad (1.6)$$

- Software
  - Licencia Microsoft Office: el valor de la licencia anual es de 80 euros.

$$th = \frac{80}{1840} = 0{,}043\ €/hora \qquad (1.7)$$

  - Licencia Arduino IDE: Arduino es una plataforma Open Source que aporta su software de manera gratuita y sin licencia, por lo que su tasa horaria es nula.

### 1.3.2 Coste de personal cualificado

- Ingeniero técnico prácticas.

$$Salario\ bruto\ mensual = 600\ €/mes \quad (1.8)$$

$$4\,\frac{sem}{mes} \cdot 40\,\frac{horas}{mes} = 160\,\frac{horas}{mes} \quad (1.9)$$

$$Coste\ horario = \frac{600}{160} = 3{,}75\ €/hora \quad (1.10)$$

- Profesor universidad.

$$Salario\ bruto\ mensual = 32000\ € \quad (1.11)$$

$$Coste\ horario = \frac{32000}{1840} = 17{,}39\ €/hora \quad (1.12)$$

### 1.3.3 Coste de oficina

El coste medio de alquiler de oficinas en la ciudad de Valencia es de 300€/mes, incluyendo gastos de luz, agua, calefacción, material de oficina, permisos y seguro correspondientes.

## 1.4 Desglose de costes totales

A continuación se muestra el presupuesto total. Este está formado por dos tablas en las que se muestran los precios desglosados de los recursos presentados anteriormente y los precios finales teniendo en cuenta los impuestos .

| Descripción | Cantidad | Precio | Importe |
|---|---|---|---|
| **Mano de obra** | | | |
| Ingeniero técnico prácticas | 300 | 3,75 | 1125,00 |
| Profesor universidad | 30 | 17,39 | 521,70 |
| | | **Total** | **1.646,70 €** |
| **Recurso informático** | | | |
| Ordenador portátil | 300 | 0,059 | 17,70 |
| Microsoft office | 100 | 0,043 | 4,30 |
| Arduino IDE | 100 | 0 | 0,00 |
| | | **Total** | **22,00 €** |
| **Material** | | | |
| Placa Arduino Nano | 1 | 21,6 | 21,60 |
| Pantalla OLED 0,96" | 1 | 4,01 | 4,01 |
| LED RGB | 1 | 0,56 | 0,56 |
| Cable de acoplamiento | 16 | 0,12 | 1,92 |
| Pulsador | 3 | 0,56 | 1,68 |
| Resisitencia 330 Ω | 6 | 0,48 | 2,88 |
| Cable mini USB | 1 | 1,27 | 1,27 |
| | | **Total** | **33,92 €** |
| **Instalaciones** | | | |
| Oficina | 2 | 300 | 600 |
| | | **Total** | **600,00 €** |
| | | **Total recursos** | **2.302,62 €** |

| | |
|---|---|
| **TOTAL EJECUCIÓN RECURSOS** | **2.302,62 €** |
| 13% Gastos generales | 299,34 € |
| 6% Beneficio industrial | 138,16 € |
| | |
| **TOTAL EJECUCIÓN POR CONTRATA** | **2.740,12 €** |
| 21% I.V.A. | 575,42 € |
| | |
| **TOTAL PRESUPUESTO** | **3.315,54 €** |

El presupuesto del proyecto asciende a:

**#TRES MIL TRESCIENTOS QUINCE EUROS CON CINCUENTA Y CUATRO CENTIMOS#**