Caramelera

Grupo 3

Adrian Mamani. - Thiago Casiano - Lucca Martinez - Valentina Palacios

Especialidad en Computación, Escuela Técnica N°32 D.E. 14

4to 2da - TM: Proyecto Informático

Gonzalo N. Consorti

Desde el 10 de septiembre hasta el 6 de diciembre de 2024

### Introducción

El propósito del proyecto no fue muy claro en el grupo, sin embargo había que realizarlo si queríamos aprobar la materia pero también para aprender cómo usar un arduino físico ya que si bien el TinkerCAD nos brindaba un simulador muy óptimo no era lo mismo que un verdadero arduino asi que por eso yo supongo también que el profe no dio este proyecto a elección confiando en que nosotros podiamos realizarlos sin ningún problema, entre tantos bueno proyectos que había pero tan complicados también que sonaba al decir su nombre solamente elegimos el de la Caramelera no sonaba tan difícil y podía llegar hasta ser divertido -que ingenuos que seriamos pensado eso al principio-, en varios días daban ganas de no querer seguir con el proyecto y eso que no era tan complicado comparado a los demás proyectos pero con tener al menos un parte del proyecto hecho podría ser nuestro logro.

En cuanto a mis compañeros yo desde el principio ni un poco de fe les tuve, los conocía mucho con hablar tan poco, pero sin embargo no fue tanta la desilusión por de ellos y hasta la verda puedo decir que uno por lo menos cumplió, se que fue con ayuda externa pero no hay que sacarle merito por eso, siguiendo con los otros la verdad que puedo llegar a decir que pueden haber hecho lo mismo y de manera diferente pero sin modificar o hacer cambios para el proyecto, la verdad tenía ciertas expectativas por uno que al final se quedó en eso, expectativas, esto porque si no estaba yo pegado a él o yo dando indicaciones no hubiésemos arrancado el proyecto si quiera, si bien algunos días me decía que trabaje en el proyecto, si yo no hacía nada él ni siquiera podía continuar solo, está bien es en grupo pero dale no tengo que estar pegado a vos para que funciones.

Tambien agradecer al Profesor consorti por ayudarnos hasta el ultimo dia y darnos incluso días extra fuera de nuestro horario siendo que nos dio mese para terminarlo.

### Investigación

La idea de clasificar caramelos no es solo una curiosidad técnica; tiene aplicaciones prácticas en entornos comerciales y educativos. Por ejemplo, en una tienda de golosinas, un selector automático puede ayudar a agrupar productos por color o tipo, facilitando el trabajo del personal y mejorando la experiencia del cliente. En un entorno educativo, este proyecto puede servir como una herramienta didáctica para enseñar conceptos básicos de programación y electrónica.

El trabajo en equipo fue fundamental para el éxito del proyecto. Se establecieron roles claros entre los integrantes, lo que permitió una distribución eficiente de las tareas. Cada miembro aportó su experiencia en áreas específicas: algunos se encargaron del diseño del circuito, otros se enfocaron en la programación y otros en la construcción física del dispositivo. Para mantener una comunicación efectiva, se utilizó un documento compartido donde se registraron todas las ideas, avances y problemas encontrados. se estableció que el sistema requerirá los siguientes elementos:

* **Sensor GY-TCS3200**: Permite la detección y clasificación de colores en los caramelos.
* **Pantalla LCD I2C**: Utilizada para mostrar el conteo en tiempo real de los caramelos clasificados.
* **Arduino UNO**: Unidad de control que procesa las señales del sensor y controla el sistema.
* **Dos servomotores**: Encargados de redirigir los caramelos hacia diferentes compartimentos según su color.

LInks

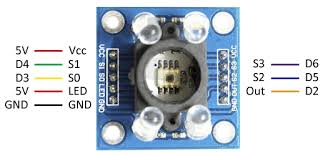
* <https://www.youtube.com/watch?v=g3i51hdfLaw>
* <https://es.gizmodo.com/como-fabricar-una-maquina-arduino-que-separa-los-carame-1489906706>

#### 

El video titulado *How to build a candy sorter using Arduino* proporciona una introducción al uso del sensor GY-TCS3200, detallando tanto las conexiones eléctricas como las funciones principales del código necesario para procesar colores, el sistema consta de el sensor que va ir detectando los colores de los caramelos con forme le llegue, la información pasa y lo lee el arduino transformando esa señal para los servos, logrando que estos giren a un angulo deseado. Además, presenta un modelo 3D de una estructura para la separación de caramelos, lo cual fue particularmente útil, ya que permitió optimizar el tiempo al no tener que diseñar una estructura desde cero, con esto se puede llegar a dar un idea de lo que se quiere lograr además de explicar las conexiones y de mas cosas acerca del proyecto, mencionado antes se puede ver mínimamente cómo se hace la conexión de cada componente que pasare a detallar eso más adelante.

El segundo enlace lleva a un pagina donde muestra el mismo proyecto de manera diferente describiendo un proyecto similar con un enfoque práctico y cotidiano: clasificar caramelos según el gusto de los usuarios y especifica que puede ser empleado para aquellas personas que no les gusta comer ciertos colores de caramelos en su caso los hace con M&Ms, se puede ver un proyecto mas espacioso y amplio pero no es distinto de lo que se espera.

### **Componentes: Sensor TCS3200**



### 

El sensor **TCS3200** es un módulo avanzado que permite la detección de colores mediante la conversión de luz a señales digitales. Este componente es fundamental para el proyecto, ya que se encarga de identificar los colores de los caramelos de manera precisa. A continuación, se detalla su funcionamiento, conexión y características clave:

#### **Estructura y Pines del Sensor**

El **TCS3200** está compuesto por una matriz de fotodiodos y un filtro de color RGB (rojo, verde y azul). Estos filtros permiten medir la intensidad de luz reflejada en diferentes longitudes de onda. El módulo cuenta con cinco pines principales, además de los de alimentación (**VCC** y **GND**), que son esenciales para su funcionamiento:

1. **VCC**:
   * Este pin se conecta al positivo de 5V del Arduino UNO.
   * Proporciona la alimentación necesaria para el funcionamiento del sensor.
2. **GND**:
   * Se conecta al pin GND (tierra) del Arduino.
   * Establece el circuito de referencia eléctrica.
3. **S0 y S1**:
   * Estos pines controlan la frecuencia de salida del sensor.
   * Configuran la escala de salida en Hz:
     + **00**: Apagado.
     + **01**: Escala baja (2%).
     + **10**: Escala media (20%).
     + **11**: Escala alta (100%).
4. **S2 y S3**:
   * Seleccionan el filtro de color que será activado para la lectura.
   * Configuración típica:
     + **S2=0, S3=0**: Rojo.
     + **S2=1, S3=0**: Azul.
     + **S2=0, S3=1**: Verde.
     + **S2=1, S3=1**: Sin filtro (detecta luz blanca).
5. **OUT**:
   * Este pin emite una señal de pulsos que corresponde a la intensidad de luz detectada.
   * Se conecta a un pin digital del Arduino para realizar la lectura de datos.

#### **Ventajas del TCS3200**

* **Precisión**: Es capaz de detectar colores con alta exactitud, lo que lo hace ideal para aplicaciones como clasificación de objetos.
* **Configurabilidad**: Los filtros de colores y las escalas de frecuencia permiten adaptarse a diferentes condiciones de luz y objetos.
* **Fácil integración con Arduino**: Su compatibilidad con señales digitales simplifica la programación.

#### **Conexión al Arduino UNO**

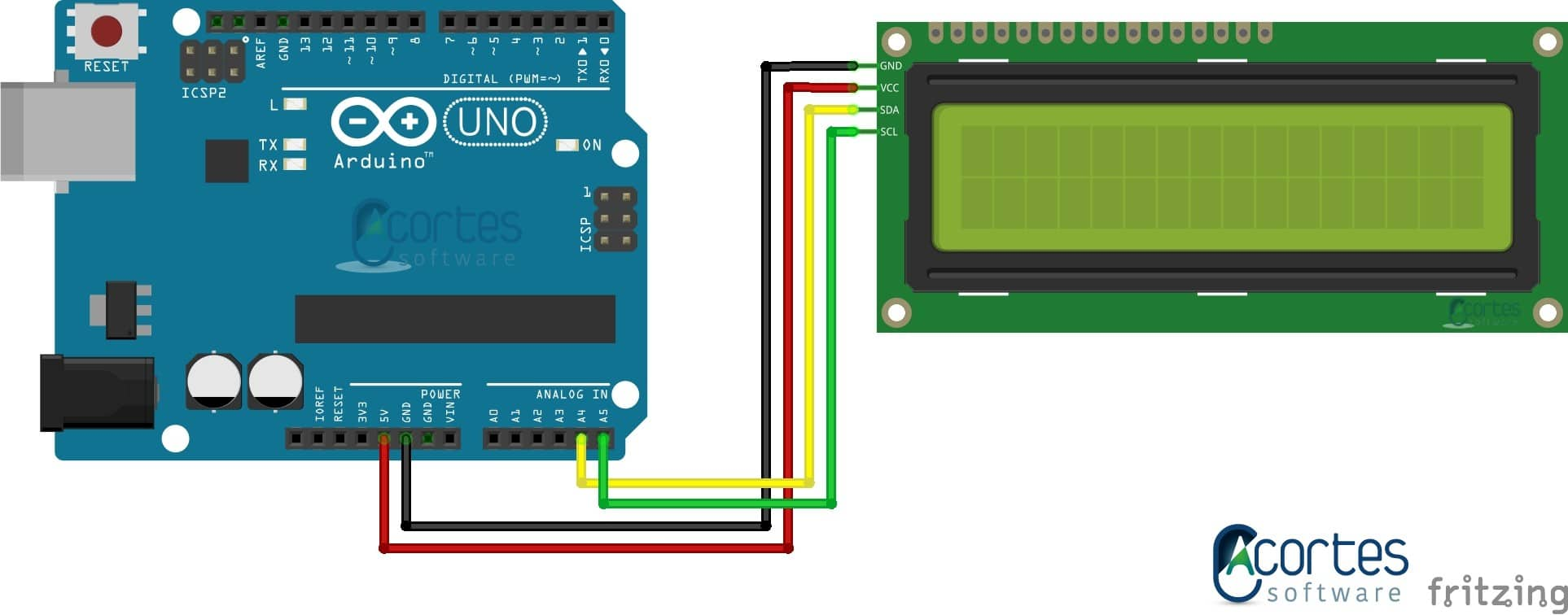
La conexión del TCS3200 al Arduino UNO se realiza de la siguiente manera:

* **VCC** a **5V**.
* **GND** a **GND**.
* **S0** y **S1** a pines digitales configurados para seleccionar la escala de frecuencia.
* **S2** y **S3** a pines digitales para el filtro de colores.
* **OUT** a un pin de entrada digital del Arduino, que se programa para leer los pulsos.

#### **Aplicaciones Comunes**

* Clasificación de objetos por color (como caramelos, botones, o pelotas).
* Sistemas de calibración de color para impresión.
* Automatización industrial para verificar calidad de productos.

### **Componente: Pantalla LCD I2C**



La **pantalla LCD I2C** es un componente esencial para la visualización de datos en proyectos con Arduino. En el caso del selector de caramelos, esta pantalla sirve para mostrar el conteo de caramelos clasificados por colores en tiempo real. Utilizar una pantalla con interfaz I2C permite simplificar la conexión y ahorrar pines en el microcontrolador, lo que es especialmente útil en proyectos con muchos componentes.

#### **Descripción de la Pantalla LCD I2C**

Las pantallas LCD (Liquid Crystal Display) son una de las formas más comunes de mostrar información en proyectos electrónicos. La interfaz **I2C** (Inter-Integrated Circuit) es un protocolo de comunicación que permite la conexión de varios dispositivos a través de solo dos cables, **SDA** (Data Line) y **SCL** (Clock Line). Esto hace que las pantallas LCD con I2C sean más fáciles de usar que las tradicionales, que requieren más pines de conexión.

La pantalla LCD I2C comúnmente utilizada tiene una resolución de **16x2**, lo que significa que tiene 16 columnas y 2 filas para mostrar texto. Sin embargo, también existen versiones con más filas o columnas.

#### **Pines de la Pantalla LCD I2C**

La pantalla LCD I2C tiene típicamente 4 pines, que son los siguientes:

1. **VCC**:
   * Se conecta al pin **5V** del Arduino para alimentar la pantalla.
2. **GND**:
   * Se conecta al pin **GND** del Arduino para establecer la referencia de tierra.
3. **SDA (Serial Data)**:
   * Es el pin de datos que se conecta al pin **A4** en el Arduino UNO (o al pin SDA en otras placas Arduino).
   * Este pin permite la transmisión de datos entre el Arduino y la pantalla.
4. **SCL (Serial Clock)**:
   * Es el pin de reloj que se conecta al pin **A5** en el Arduino UNO (o al pin SCL en otras placas Arduino).
   * El pin SCL sincroniza los datos que se transmiten a través del pin SDA.

#### **Principio de Funcionamiento**

La pantalla LCD I2C funciona mediante el protocolo I2C, que permite la comunicación entre el Arduino y la pantalla usando solo dos pines, lo que simplifica el cableado y permite la conexión de varios dispositivos en un solo bus I2C. El Arduino envía comandos a la pantalla para realizar tareas como mostrar texto, ajustar el brillo y mover el cursor.

El controlador I2C que se encuentra en la pantalla LCD se encarga de traducir los datos enviados por el Arduino en señales que la pantalla puede mostrar, utilizando una matriz de píxeles para generar los caracteres y símbolos que se ven en la pantalla.

**Ejemplo de uso:**

* El Arduino puede enviar el texto "Cantidad de caramelos: 5" a la pantalla LCD I2C, y la pantalla lo mostrará en sus dos filas de manera legible.

#### **Ventajas de Usar LCD I2C**

1. **Reducción de pines**: El uso de I2C reduce el número de pines necesarios para conectar la pantalla, permitiendo usar otros pines para diferentes componentes.
2. **Fácil control**: Gracias a bibliotecas como LiquidCrystal\_I2C, es fácil controlar la pantalla desde el Arduino sin tener que preocuparnos por los detalles de la comunicación I2C.
3. **Ahorro de espacio**: Las pantallas LCD I2C son compactas, lo que las hace ideales para proyectos pequeños y medianos.

#### **Conexión al Arduino UNO**

La conexión del **LCD I2C** al Arduino UNO se realiza de la siguiente manera:

* **VCC** a **5V** del Arduino.
* **GND** a **GND** del Arduino.
* **SDA** a **A4** (en Arduino UNO).
* **SCL** a **A5** (en Arduino UNO).

Una vez conectados estos pines, se utiliza una biblioteca en Arduino (como LiquidCrystal\_I2C) para controlar la pantalla y enviar datos desde el microcontrolador.

#### **Ejemplo de Código para LCD I2C**

Un ejemplo básico de código para mostrar un mensaje en la pantalla LCD I2C sería el siguiente:

| #include <Wire.h> #include <LiquidCrystal\_I2C.h>  // Dirección I2C de la pantalla LCD LiquidCrystal\_I2C lcd(0x27, 16, 2); // Dirección, número de columnas y filas  void setup() {  lcd.begin(16, 2); // Inicializa la pantalla LCD  lcd.print("Caramelos: "); // Muestra texto en la primera línea }  void loop() {  lcd.setCursor(0, 1); // Mueve el cursor a la segunda línea  lcd.print("Cantidad: 5"); // Muestra el número de caramelos en la segunda línea  delay(1000); // Espera 1 segundo } |
| --- |

1. inclusión de la librería

**#include <Wire.h>**: Esta línea incluye la librería **Wire** que permite la comunicación I2C (Inter-Integrated Circuit). Esencialmente, esta librería habilita al Arduino a enviar y recibir datos a través de los pines **SDA** (Data Line) y **SCL** (Clock Line), necesarios para interactuar con dispositivos I2C, como la pantalla LCD.

**#include <LiquidCrystal\_I2C.h>**: Esta es la librería que específicamente maneja la pantalla LCD con interfaz I2C. Permite inicializar y controlar la pantalla con funciones simplificadas, ya que los detalles de la comunicación I2C están abstraídos.

1. configuración del void Setup

**lcd.init();**: Esta línea inicializa la pantalla LCD configurando el número de columnas y filas. Aunque en nuestro caso estos valores ya están establecidos cuando se crea el objeto, se utiliza este comando para asegurarse de que la pantalla esté lista para recibir comandos.

**lcd.print("Caramelos: ");**: Aquí, el mensaje **"Caramelos: "** se imprime en la primera fila de la pantalla LCD. El comando **lcd.print()** permite escribir texto o números en la pantalla. Si hay espacio disponible en la pantalla, el texto se irá mostrando de izquierda a derecha.

1. funcionalidad en el void Loop

**lcd.setCursor(0, 1);**: Este comando mueve el cursor a la posición **(0, 1)**, lo que significa que se sitúa en la **primera columna** de la **segunda fila** de la pantalla LCD. Las coordenadas de la pantalla comienzan en **0**, por lo que **(0, 0)** sería la primera columna de la primera fila, y **(0, 1)** sería la primera columna de la segunda fila.

**lcd.print("Cantidad: 5");**: Este comando muestra el texto **"Cantidad: 5"** en la segunda línea de la pantalla LCD. El número **5** es solo un valor de ejemplo, que podría ser actualizado para mostrar el conteo real de caramelos clasificados según el sensor de colores.

**delay(1000);**: El comando **delay()** detiene el programa durante la cantidad de milisegundos que se le indique. En este caso, **1000** equivale a 1 segundo. Esto hace que el mensaje en la pantalla permanezca durante 1 segundo antes de que el ciclo se repita.

### Componentes: Servo Motor

Los **servomotores** son componentes electromecánicos esenciales en muchos proyectos de robótica y automatización, y en este caso, son utilizados para controlar el movimiento y la clasificación de los caramelos según su color. Un servomotor permite girar un objeto conectado a él en un rango determinado de ángulos, típicamente entre **0° y 180°**.

#### **Descripción del Servomotor**

Un servomotor consta de un motor de corriente continua (DC), una caja de engranajes y un circuito controlador interno. Este sistema permite que el motor gire solo en los ángulos deseados con gran precisión. Los servos generalmente se controlan mediante una señal de **modulación por ancho de pulso** (PWM), en la cual el ancho del pulso determina el ángulo de rotación.

El servomotor tiene tres pines importantes:

1. **Pin de señal (PWM)**: Este pin recibe la señal PWM desde el microcontrolador, como el Arduino. El ancho de pulso en esta señal determinará el ángulo de rotación del motor. Cuanto mayor sea el ancho del pulso, mayor será el ángulo.
2. **Pin de alimentación (VCC)**: Se conecta al pin de 5V del Arduino o a una fuente de alimentación externa.
3. **Pin de tierra (GND)**: Se conecta al pin GND del Arduino o a la tierra de la fuente de alimentación externa.

#### **Principio de Funcionamiento**

Los servomotores se controlan utilizando una señal de control que consiste en una serie de pulsos eléctricos. La duración de cada pulso determina el ángulo de rotación del servo. Por ejemplo:

* Un pulso de 1.5 ms generalmente hace que el servo se coloque a **90°**.
* Un pulso de 1 ms coloca el servo a **0°**.
* Un pulso de 2 ms coloca el servo a **180°**.

El ciclo de trabajo (duty cycle) y la frecuencia de la señal PWM son los parámetros que definen cómo se mueve el servomotor.

En el proyecto de clasificación de caramelos, el Arduino envía una señal PWM al servomotor, indicando la posición exacta en la que debe colocar el caramelo dependiendo de su color. Los servos giran a un ángulo específico para mover los caramelos hacia el compartimento correspondiente.

#### **Conexión de Servomotores al Arduino**

Los servomotores se conectan al Arduino de manera sencilla:

* El **pin de señal (PWM)** del servo se conecta a un pin digital de salida en el Arduino, por ejemplo, **D9**.
* El **pin de alimentación (VCC)** se conecta al pin **5V** del Arduino o a una fuente de alimentación externa.
* El **pin de tierra (GND)** se conecta al pin **GND** del Arduino.

Es importante asegurarse de que el Arduino pueda suministrar suficiente corriente al servomotor o usar una fuente de alimentación externa para evitar daños al microcontrolador.

#### **Uso de la Biblioteca Servo**

Para controlar los servomotores desde el Arduino, se utiliza la **biblioteca Servo**. Esta biblioteca facilita el control de hasta 12 servos en una placa Arduino UNO. A continuación se muestra un ejemplo básico de cómo controlar un servomotor:

| #include <Servo.h> |
| --- |

| Servo myServo; // Creación del objeto servo  void setup() {  myServo.attach(9); // Conectar el servomotor al pin 9 }  void loop() {  myServo.write(0); // Mueve el servo a 0 grados  delay(1000); // Espera 1 segundo  myServo.write(90); // Mueve el servo a 90 grados  delay(1000); // Espera 1 segundo  myServo.write(180); // Mueve el servo a 180 grados  delay(1000); // Espera 1 segundo } |
| --- |

1. inclusión y creación

**#include <Servo.h>**: Esta línea incluye la librería Servo.h, que es fundamental para controlar los servomotores con Arduino. Esta biblioteca proporciona funciones fáciles de usar para mover servos y establecer su posición en grados. La librería maneja automáticamente la señal PWM que se envía al pin del servomotor.

**Servo miServo;**: Aquí se crea un objeto llamado miServo de tipo **Servo**. Este objeto permitirá controlar un servomotor, y lo usaremos para mover el servo conectado al pin 9 del Arduino. Un microcontrolador Arduino puede controlar hasta 12 servos a la vez si se está utilizando la biblioteca Servo.h, cada uno mediante un pin de señal distinto.

1. void setup

**miServo.attach(9);**: En la función setup(), se utiliza el método attach() para asociar el servomotor con un pin específico del Arduino. En este caso, el servomotor está conectado al pin **D9** del Arduino. Este pin digital es el que enviará la señal PWM al servomotor, que determinará su posición.

**setup()** se ejecuta solo una vez al inicio del programa y es donde se configuran las inicializaciones del hardware y las variables, en este caso, asociando el pin con el servomotor.

1. void loop
2. **miServo.write(0);**: La función write() se utiliza para establecer la posición del servomotor en un ángulo específico. El valor que se pasa a esta función está en grados, y en este caso **0** hace que el servo se mueva a la posición más baja posible (generalmente 0 grados).
3. **delay(1000);**: La función delay() introduce una pausa en la ejecución del código. El parámetro 1000 significa que el código se detendrá durante 1000 milisegundos, o **1 segundo**. Esto permite observar claramente el movimiento del servomotor en cada posición antes de que cambie.
4. **miServo.write(90);**: El siguiente comando mueve el servo a la **posición intermedia**, es decir, a **90 grados**. Esto es útil cuando se necesita un ángulo central o una posición intermedia en el servo.
5. **miServo.write(180);**: Finalmente, el servo se mueve a **180 grados**, que es el ángulo máximo en muchos servos estándar. Esto puede representar el extremo opuesto del movimiento del servo.

Este ciclo se repite indefinidamente porque el código se encuentra en la función loop(), la cual se ejecuta una y otra vez mientras el Arduino esté encendido.

Teniendo en cuenta lo mencionado, el desarrollo del proyecto siempre se enfrenta a ciertos obstáculos, especialmente cuando se trabaja con componentes electrónicos y programación. En este caso, al comenzar a escribir el código para el sistema de selección de caramelos utilizando Arduino, surgieron varias dificultades que complicaron la implementación de manera definitiva. Estas fallas, aunque comunes en proyectos de electrónica y programación, fueron parte del proceso de aprendizaje y optimización.

### **Proceso de Desarrollo del Código**

Para comenzar con el desarrollo de nuestro proyecto, seguimos una serie de pasos bien definidos que nos permitieron entender en detalle lo que necesitábamos hacer y cómo podríamos abordar las distintas fases del proyecto de manera eficiente.

Lo primero que hicimos fue visitar la página web que nos fue proporcionada por Consorti, que resultó ser una excelente fuente de información. La página web en cuestión era el siguiente enlace:

* <https://diyodemag.com/projects/lolly_sorter>.

Esta página nos ofreció una visión general del proyecto que íbamos a realizar, proporcionándonos no solo el diseño general, sino también los detalles técnicos necesarios para llevar a cabo todo el proceso de manera correcta.

La página incluía una descripción detallada del proyecto "Lolly Sorter", el cual es un sistema diseñado para clasificar caramelos de diferentes colores utilizando un sensor de color. Esto nos dio la idea de que el proyecto tenía aplicaciones prácticas, y podría utilizarse no solo como una actividad educativa, sino también como un modelo funcional para procesos automatizados de clasificación. El proyecto estaba orientado a un público que busca aprender y aplicar la electrónica básica, el uso de sensores, servos y programación con Arduino, lo cual nos dio un marco adecuado para comenzar con el desarrollo.

### **Especificaciones y detalles clave del proyecto**

En la página también se presentaban las especificaciones exactas del proyecto, las cuales eran esenciales para entender el tipo de componentes que debíamos utilizar, así como las conexiones y las configuraciones necesarias para lograr que el sistema funcionara correctamente. Las especificaciones incluían información sobre los componentes electrónicos necesarios, como el sensor de color TCS3200, servomotores, y una pantalla LCD, entre otros. Estos detalles fueron cruciales, ya que nos permitieron hacer un análisis detallado de lo que ya teníamos y lo que necesitábamos adquirir.

Además de los componentes electrónicos, la página también proporcionaba diagramas de conexión y ejemplos de código. Esto fue particularmente útil, ya que no solo podíamos ver cómo se conectaban los componentes, sino también cómo debía estructurarse el código para que el sistema pudiera detectar el color de los caramelos y, en función de ello, accionar los servos para clasificar los caramelos correctamente.

**El proyecto se basaba principalmente en el uso de un Arduino para controlar los servos y el sensor de color, y para comunicar los resultados al usuario mediante una pantalla LCD.** A través de este diseño, se lograba automatizar la clasificación de los caramelos de forma precisa y rápida, lo cual era el objetivo principal de la idea inicial.

### **La importancia del sensor de color**

Una de las partes más interesantes y clave del proyecto es el sensor de color TCS3200, que es capaz de detectar colores específicos y generar señales de frecuencia proporcionales a la intensidad de los colores detectados. Este sensor es crucial para que el sistema funcione correctamente, ya que es el encargado de identificar el color de los caramelos que se colocan frente a él.

La página proporcionaba instrucciones detalladas sobre cómo conectar este sensor al Arduino, y cómo configurarlo para que pudiera leer los colores de los objetos correctamente. A través del uso de diferentes filtros de color, se podía optimizar la detección y hacer que el sistema fuera más preciso al momento de clasificar los caramelos en función de su color.

### **Implementación del código y la programación**

Una vez que comprendimos las especificaciones del hardware y cómo se conectaban los componentes, pasamos a la fase de programación del sistema. El código proporcionado en la página fue un punto de partida esencial. Sin embargo, decidimos modificarlo ligeramente para adaptarlo mejor a nuestras necesidades y para incorporar algunas mejoras que pensábamos que podían optimizar el rendimiento del sistema.

El código estaba basado en la plataforma de Arduino, lo cual nos permitió aprovechar su facilidad de uso y las bibliotecas que ya estaban disponibles para controlar los servos, leer los datos del sensor de color, y manejar la pantalla LCD. Gracias a esto, pudimos centrarnos en la lógica del proyecto sin tener que preocuparnos demasiado por detalles técnicos complejos de programación o integración de hardware.

En el código, se realizaba la lectura de los valores de color de los caramelos, y luego se procesaba esta información para identificar el color más probable. Dependiendo de la detección, se accionaban los servos para mover los caramelos hacia el compartimiento correspondiente. Además, el sistema mostraba información sobre el color detectado y el conteo de caramelos en una pantalla LCD, lo que hacía el sistema más interactivo y fácil de usar.

**Este proceso de programación fue fundamental para que el proyecto fuera funcional.** A medida que íbamos probando el código, ajustábamos algunos parámetros y corregíamos errores menores que surgían, lo que nos permitió lograr que el sistema fuera cada vez más preciso y eficiente.

El desarrollo de un proyecto como este implica una combinación de pruebas, errores y ajustes en el código y en las conexiones de hardware. Inicialmente, el equipo intentó implementar el código para controlar tanto el sensor de colores **TCS3200** como los **servomotores** y la **pantalla LCD I2C**. Sin embargo, nos encontramos con varios fallos en el camino:

1. **Problemas con la Lectura del Sensor TCS3200**: Uno de los primeros obstáculos fue configurar correctamente el sensor TCS3200 para que detectara los colores de los caramelos de manera precisa. Este sensor funciona mediante la variación en la frecuencia de salida según los colores que detecta, pero en algunos casos, la calibración de los valores no fue la esperada. La información recibida por el Arduino no siempre coincidía con el color real del caramelo, lo que impedía una correcta clasificación.
   * **Posible Solución**: Para superar este problema, se investigó sobre técnicas de filtrado de la señal, como el uso de un filtro de paso bajo para suavizar la salida de frecuencias y así obtener lecturas más confiables. Además, se probó con diferentes configuraciones de luz para garantizar que el sensor pudiera distinguir correctamente los colores bajo diversas condiciones.
2. **Desajustes en el Control de los Servomotores**: Otro reto importante fue el control de los servos para mover los caramelos hacia los compartimentos correspondientes. Al principio, los servomotores no respondían de manera precisa a las señales enviadas por el Arduino. Esto se debió principalmente a una mala interpretación de los valores PWM enviados al servo, lo que provocaba movimientos erráticos o imprecisos.
   * **Posible Solución**: Para resolver esto, se revisaron las conexiones de los servos y se ajustaron los valores de los pulsos PWM, utilizando un código de prueba para calibrar el rango de movimiento de los servos. Esto implicó enviar señales de diferente duración y observar cómo respondían los servos, ajustando posteriormente el código para reflejar estos cambios.
3. **Conexión y Visualización de la Pantalla LCD I2C**: En algunos momentos, la pantalla LCD I2C no mostraba la información correcta o no se mostraba nada en absoluto. Esto podría deberse a varias causas, como una mala conexión de los cables o un problema con la dirección I2C del dispositivo.
   * **Posible Solución**: Se verificó la conexión de la pantalla con el Arduino y se usó una herramienta para detectar la dirección I2C del dispositivo, como el escáner I2C disponible en la comunidad de Arduino. Después de confirmar la dirección correcta, el código fue ajustado para reflejar la información precisa que debía mostrarse en la pantalla LCD.
4. **Falta de Sincronización entre los Componentes**: Un desafío adicional fue la sincronización entre los componentes del sistema. El sensor de color debía detectar el caramelo, luego enviar esa información al Arduino, que a su vez activaba los servos para mover el caramelo y actualizaba la pantalla LCD con la cantidad de caramelos clasificados. El sistema no estaba funcionando de manera fluida, y en muchos casos, los movimientos de los servos no coincidían con la lectura del sensor.
   * **Posible Solución**: Para mejorar la sincronización, se implementaron delays en el código para dar tiempo suficiente entre cada paso del proceso (lectura del sensor, activación del servo y actualización de la pantalla). Además, se consideró la implementación de un sistema de interrupciones para manejar tareas de forma más eficiente y evitar que los diferentes componentes se bloquearan unos a otros.
5. **Manejo de Errores y Depuración**: Durante el proceso de desarrollo, otro desafío fue el manejo de errores. En muchos casos, el Arduino no podía procesar la señal del sensor o los servomotores no funcionaban como se esperaba debido a errores de programación o fallos en las conexiones. Esto nos llevó a realizar múltiples ciclos de **depuración**.
   * **Posible Solución**: Se utilizó la función **Serial.print()** para imprimir valores en el monitor serial y poder verificar qué valores estaban recibiendo los sensores y servos. Esto permitió identificar errores en tiempo real y realizar ajustes sobre la marcha. Además, se implementaron chequeos de estado en el código para garantizar que el sistema pudiera recuperarse de ciertos fallos, como la desconexión de los servos o la pérdida de lectura del sensor.

### **Lecciones Aprendidas**

A través de este proceso de desarrollo, se aprendió lo siguiente:

1. **Importancia de la Calibración**: La correcta calibración de los sensores y servos es crucial para el éxito del proyecto. Cada componente debe ser ajustado cuidadosamente para que funcione dentro de los parámetros adecuados.
2. **Pruebas Continuas**: Las pruebas constantes son esenciales en proyectos de hardware y software. Un cambio en la configuración de un componente puede tener un impacto en todo el sistema, por lo que es necesario realizar pruebas aisladas antes de integrar todos los elementos.
3. **Manejo de Errores**: La implementación de un sistema de depuración efectivo facilita mucho el proceso de diagnóstico y corrección de errores. Herramientas como el monitor serial de Arduino son muy útiles para verificar el comportamiento de cada componente en tiempo real.
4. **Adaptación de los Códigos**: La flexibilidad es clave. No siempre el primer intento de código será el adecuado, y en la mayoría de los casos, se necesitará modificarlo a medida que surgen nuevos problemas o se agregan más funcionalidades al sistema.

### Partes fundamentales del codigo

| void readColor() {  digitalWrite(S2, LOW);  digitalWrite(S3, LOW);  delay(100); // Esperar un poco para que el sensor estabilice  red = pulseIn(OUT, HIGH);   digitalWrite(S2, HIGH);  digitalWrite(S3, HIGH);  delay(100);  green = pulseIn(OUT, HIGH);   digitalWrite(S2, LOW);  digitalWrite(S3, HIGH);  delay(100);  blue = pulseIn(OUT, HIGH); }  Explicación:   * **Esta función lee los valores de color del sensor TCS3200.** * Primero, se configura el sensor para leer el color rojo. Para esto, se configuran los pines S2 y S3 en LOW (lo que activa el filtro de color rojo), y se mide el tiempo en que la señal de salida (OUT) está en HIGH utilizando pulseIn(OUT, HIGH). Esto da un valor proporcional a la intensidad del color rojo. * Luego, se configura para leer el color verde, y después el azul, cambiando las configuraciones de S2 y S3 para activar los filtros correspondientes. * La función pulseIn() mide la duración de una señal de pulso (en este caso, la señal de salida del sensor), l   link a un video explicativo   * [Arduino desde cero en Español - Capítulo 61 - TCS3200 Sensor de color RGB automatización 🔴🟢🔵 - YouTube](https://www.youtube.com/watch?v=GjEEzKfXfik) (se separa por tiempo, se toma la parte inicial de la explicacion)     El **TCS3200** es un sensor de color basado en una matriz de fotodiodos que mide la intensidad de diferentes colores de luz. El sensor TCS3200 utiliza una **matriz de fotodiodos** dispuesta en una cuadrícula de 8x8, donde cada fotodiodo está diseñado para detectar un color específico (rojo, verde o azul) según el filtro colocado sobre ellos.  El sensor TCS3200 tiene un **filtro de color** incorporado, que permite medir la cantidad de luz de cada color (rojo, verde, azul, o incluso la luz sin filtrar) que incide sobre los fotodiodos. Los filtros S2 y S3 (que son pines del TCS3200) se utilizan para seleccionar qué filtro de color aplicar a los fotodiodos.  Fotodiodos y Filtros de Color  El sensor TCS3200 está compuesto por una matriz de fotodiodos que están cubiertos por filtros de color rojo, verde, azul y sin filtro (transparente). La salida de estos fotodiodos se convierte en una señal de frecuencia que es proporcional a la intensidad de la luz que incide sobre ellos.   * **Filtros de color**: El sensor tiene filtros de color rojo, verde y azul, y un filtro sin color. Estos filtros determinan qué tipo de luz llega a cada fotodiodo en la matriz. * **Fotodiodos**: Los fotodiodos están distribuidos en la matriz de manera que algunos de ellos están expuestos a luz roja, otros a luz verde, otros a luz azul, y otros a luz sin filtrar. Cada fotodiodo responde a la intensidad de luz que incide sobre él, produciendo una corriente proporcional a esa intensidad.   La **configuración de los pines S2 y S3** (y la señal de control de los pines S0 y S1) determina qué filtro de color está activo en un momento dado:   * **S2 = LOW, S3 = LOW**: Filtro rojo. * **S2 = HIGH, S3 = HIGH**: Filtro verde. * **S2 = LOW, S3 = HIGH**: Filtro azul.   Cada vez que se cambian los valores de los pines S2 y S3, el sensor cambia el filtro de color que aplica a los fotodiodos y, por lo tanto, la luz que mide.  Los pines **S0** y **S1** en el sensor TCS3200 son utilizados para controlar la frecuencia de salida (en Hz) del sensor, que está relacionada con la intensidad de luz medida por los fotodiodos. Esta frecuencia de salida es clave para obtener lecturas precisas de la intensidad de los colores (rojo, verde, azul) en función de la cantidad de luz reflejada en el objeto que se está midiendo.  Frecuencia de Salida del Sensor  El TCS3200 genera una señal de salida que es una onda cuadrada, cuya frecuencia está directamente proporcional a la cantidad de luz de color detectada por los fotodiodos del sensor. Al controlar los pines **S0** y **S1**, podemos ajustar la escala de esa frecuencia, es decir, cómo rápida o lentamente se envían las señales de pulso. Esto se hace para controlar la resolución y la sensibilidad de las mediciones de color.  Modos de Salida Controlados por S0 y S1  La frecuencia de salida del TCS3200 puede ser ajustada a diferentes niveles usando los pines **S0** y **S1**. Hay cuatro combinaciones posibles para estos dos pines, y cada combinación controla una escala diferente de la frecuencia de salida.  Configuraciones de los Pinos S0 y S1  **S0 = HIGH, S1 = LOW**:  En este caso, la frecuencia de salida está configurada para su **máxima resolución**. Esto significa que el sensor proporcionará una salida de frecuencia más alta y, por lo tanto, una mayor sensibilidad.  Este modo es útil cuando se desea medir con precisión los colores, ya que la frecuencia de salida está directamente relacionada con la intensidad de la luz y permite una medición más detallada.  **S0 = LOW, S1 = LOW**:  Cuando ambos pines **S0 y S1** están configurados en LOW, la frecuencia de salida está configurada para la **escala de salida más baja**.  Este modo se utiliza cuando se quiere reducir la cantidad de señales de salida generadas, lo que puede ser útil para aplicaciones que no requieren alta precisión en la medición de color, o cuando se necesita ahorrar energía.  **S0 = HIGH, S1 = HIGH**:  Con esta configuración, la frecuencia de salida está configurada para una **escala de salida intermedia**.  Este ajuste también puede ser útil dependiendo de las necesidades del proyecto, proporcionando un equilibrio entre precisión y consumo de recursos.  **S0 = LOW, S1 = HIGH**:  Este ajuste también configura una frecuencia intermedia, aunque con una diferente combinación de pines, que puede ser útil en situaciones específicas.  ¿Por Qué Son Importantes los Pinos S0 y S1?  La configuración de los pines **S0** y **S1** influye directamente en la **resolución** y **precisión** con la que se leen los colores. Al ajustar estos pines, puedes equilibrar entre obtener una mayor resolución (más detallada) de la medición del color o, si el rendimiento y la velocidad son más importantes, puedes optar por una menor resolución para reducir la cantidad de datos que el sensor necesita generar.  En el código que proporcionaste, se utilizan los pines **S0** y **S1** para establecer la **escala máxima de frecuencia**, que se logra configurando **S0 en HIGH** y **S1 en LOW**:  Teniendo esto en cuenta se pudo hacer el siguiente codigo:   | // Función para detectar el color basado en los valores de rojo, verde y azul int detectColor(int red, int green, int blue) {  if (red > 150 && green > 1400 && blue > 500) {  return Rojo; // Rojo  }  if (red > 300 && green > 1500 && green < 1600 && blue > 550) {  return Naranja; // Naranja  }  if (red > 2000 && green > 300 && blue > 300) {  return Amarillo; // Amarillo  }  if (red > 1150 && green > 550 && blue > 550 ) {  return Verde; // Verde  }  if (red > 200 && green > 100 && blue < 100) {  return Azul; // Azul  } | | --- |   al sensor se le fueron acercando colores como se puede ver ahí y cada que valor que leia según sus fotodiodos se los fue anotando para después poder hacer que los servos se moviesen, fue un poco tarde igual para empezar con los servos ya que habia que calcular los angulos en los que tenia que estar los recipientes. |
| --- | --- |