Universidad de Salamanca

Escuela Politécnica Superior de Zamora

Trabajo de fin de grado

DISEÑO Y AUTOMATIZACIÓN DE UNA CINTA TRANSPORTADORA CLASIFICADORA

Grado en Ingeniería Mecánica
Departamento de Física Aplicada

Autora
Verónica Morán García

Tutora
Elena Pascual Corral
Cotutor
Miguel Ángel Rabanillo de la Fuente

Fecha de adjudicación: 22 de noviembre de 2022
Fecha de entrega: 20 de junio de 2023

Agradecimientos
A mis tutores Elena y Miguel por su constante atención y apoyo.
A mi familia, Jaime y su familia por su ayuda.

DISEÑO Y AUTOMATIZACIÓN DE UNA CINTA TRANSPORTADORA CLASIFICADORA

"DESIGN AND AUTOMATION OF A SORTING CONVEYOR BELT"

Autora: Verónica Morán García
Tutora: Elena Pascual Corral
Cotutor: Miguel Ángel Rabanillo de la Fuente

Departamento de Física Aplicada
Escuela Politécnica Superior de Zamora
Universidad de Salamanca

Fecha de entrega: 20 de junio de 2023

Certificado de los profesores

Dña. Elena Pascual Corral y D. Miguel Ángel Rabanillo de la Fuente del Departamento de Física Aplicada de la Universidad de Salamanca.

CERTIFICAN:

Que el trabajo titulado “Diseño y automatización de una cinta transportadora” ha sido realizado por Verónica Morán García bajo su dirección.

Y para que conste a todos los efectos oportunos.

En Zamora a 20 de junio de 2023

FIRMAS

Firmado por MORAN GARCIA VERONICA 71043929B el día 22/06/2023 con un certificado emitido por AC FNMT Usuarios

Resumen

En el presente proyecto, se ha propuesto recrear a pequeña escala un proceso industrial común: una cinta transportadora clasificadora. Estos sistemas de transporte continua desempeñan un papel importante en la industria, permitiendo el movimiento eficiente de objetos y materiales a lo largo de una línea de producción.

En este caso, la clasificación de objetos se ha enfocado en base al color, ya que se trata de una característica versátil y adaptable a diferentes tipos de industrias en las cuales es necesario detectar diferentes colores con el fin de clasificar, separar y organizar productos de manera automatizada, reduciendo errores y optimizando el flujo de trabajo.

Para llevar a cabo esta tarea, hemos utilizado un microcontrolador Arduino. Este microcontrolador es ampliamente reconocido y utilizado en el ámbito educativo e industrial. Es accesible en términos económicos, lo que facilita su implementación en proyectos de prototipado y desarrollo. Además, cuenta con una gran comunidad de usuarios, lo que facilita el aprendizaje y la resolución de problemas.

Mediante el empleo del microcontrolador Arduino y otros componentes electrónicos, se ha logrado diseñar un prototipo de una cinta transportadora clasificadora. Esta recreación ha permitido estimar la viabilidad de aplicar este estudio a una escala mayor en la industria. Los resultados obtenidos ofrecen una visión general de los desafíos que podrían surgir al implementar un sistema similar a gran escala, así como propuestas de mejoras a los problemas como ha sido la alimentación o el funcionamiento simultáneo de los sensores.

El objetivo ha sido por tanto explorar la aplicación de Arduino en la automatización industrial, centrándose en la clasificación de objetos mediante el color y apoyándose el proyecto en la maquetación de la cinta. Se demuestra también el potencial de utilizar el microcontrolador Arduino como una herramienta educativa y económica para diseñar prototipos y estimar la viabilidad de proyectos industriales a mayor escala.

Palabras clave: cinta transportadora clasificadora, microcontrolador Arduino, sensor de color.

Abstract

In the present project, the aim has been to recreate on a small scale a common industrial process: a sorting conveyor belt. These continuous transport systems play a crucial role in the industry, enabling the efficient movement of objects and materials along a production line.

In this case, the object classification has been focused on color, as it is a versatile characteristic adaptable to different types of industries where the detection of different colors is necessary for automated sorting, separating, and organizing of products, reducing errors and optimizing workflow.

To carry out this task, we have used an Arduino microcontroller. This microcontroller is widely recognized and used in both educational and industrial settings. It is cost-effective, making it easily implementable in prototyping and development projects. Additionally, it has a large user community, facilitating learning and problem-solving.

By employing the Arduino microcontroller and other electronic components, a prototype of a sorting conveyor belt has been designed. This recreation has allowed us to assess the feasibility of applying this study on a larger scale in the industry. The obtained results provide an overview of the challenges that may arise when implementing a similar system on a larger scale, as well as proposals for improvements, such as addressing power supply or simultaneous sensor operation issues.

The objective has been, therefore, to explore the application of Arduino in industrial automation, focusing on object sorting by color and utilizing the conveyor belt prototype. It also demonstrates the potential of using the Arduino microcontroller as an educational and cost-effective tool for designing prototypes and assessing the feasibility of larger-scale industrial projects.

Keywords: sorting conveyor belt, Arduino microcontroller, color sensor.

Listado de acrónimos

- **ITC**: Las ITC son documentos normativos que complementan y proporcionan instrucciones adicionales sobre cómo cumplir con los requisitos establecidos en la legislación vigente en determinados ámbitos
- **UNE-EN**: Normas técnicas europeas adoptadas por la Asociación Española de Normalización (UNE). La sigla UNE-EN indica que la norma es una adopción de una norma europea.
- **AENOR**: Asociación Española de Normalización y Certificación. Es la entidad encargada de establecer las normas técnicas y certificaciones en España.
- **NTP**: Notas técnicas de prevención. Manual de consulta indispensable para todo prevencionista y obedece al propósito del Instituto de facilitar a los agentes sociales y a los profesionales de la prevención de riesgos laborales herramientas técnicas de consulta.
- **INSST**: Instituto Nacional de Seguridad y Salud en el Trabajo. Es el órgano científico técnico especializado en prevención de riesgos laborales (PRL) de la Administración General del Estado.
- **BOE**: diario oficial del Estado español, es el medio de publicación de las leyes, disposiciones y actos de inserción obligatoria.
- **RGBW**: “Red”, “Green”, “Blue”, “White”. Se refiere a una combinación de colores en un sistema de iluminación que incluye los colores rojo, verde, azul y blanco.
- **I2C**: “*Inter-Integrated Circuit*”. Es un protocolo de comunicación serial utilizado para la comunicación entre microcontroladores y dispositivos periféricos.
- **SPI**: “*Serial Peripheral Interface*”. Es otro protocolo de comunicación serial utilizado para la comunicación entre dispositivos electrónicos.
- **MODBUS**: Protocolo de comunicación utilizado para la comunicación entre dispositivos electrónicos en sistemas de control industrial.
- **PWM**: “*Pulse Width Modulation*”. Es una técnica utilizada para controlar la energía entregada a dispositivos como motores o luces, mediante la variación del ancho de los pulsos de señal.
- **SDA**: “*Serial Data Line*”. Es una línea de comunicación en el protocolo I2C utilizada para la transmisión de datos.
- **SCL**: “*Serial Clock Line*”. Es una línea de comunicación en el protocolo I2C utilizada para la sincronización de los datos.

- **MOSI**: "*Master Out Slave In*". Es una línea de comunicación en el protocolo SPI utilizada para la transmisión de datos del maestro al esclavo.
- **SCK**: "*Serial Clock*". Es una línea de comunicación en el protocolo SPI utilizada para la sincronización de los datos.
- **SS**: "*Slave Select*". Es una línea de comunicación en el protocolo SPI utilizada para seleccionar el esclavo con el que se desea comunicar el maestro.
- **MISO**: "*Master In Slave Out*". Es una línea de comunicación en el protocolo SPI utilizada para la transmisión de datos del esclavo al maestro.
- **Shields**: Son placas o módulos que se conectan directamente a una placa Arduino para agregar funcionalidades adicionales, como sensores, actuadores, comunicación, etc.
- **CNC**: "*Control Numérico por Computadora*". Se refiere a sistemas de control utilizados en máquinas herramienta y robots industriales.
- **UART**: "*Universal Asynchronous Receiver-Transmitter*". Es un circuito utilizado para la comunicación serial asíncrona entre dispositivos.
- **TTL**: "*Transistor-Transistor Logic*". Es una familia de circuitos electrónicos digitales que utiliza transistores bipolares para la lógica digital.
- **C++:** Lenguaje de programación de alto nivel y propósito general. Es ampliamente utilizado en el desarrollo de software y también en programación de microcontroladores como Arduino.
- **PLC**: "*Controlador Lógico Programable*". Es un dispositivo electrónico utilizado para controlar y automatizar procesos en la industria.

Índice general

ANEXOS

Índice de figuras

Índice de tablas

Índice de diagramas

1. Introducción

Las cintas transportadoras han sido una herramienta importante en la industria y la logística durante décadas. Su invención se remonta a finales del siglo XIX, cuando surgieron como una solución eficiente para el transporte de materiales en la industria minera. Desde entonces, han evolucionado y se han utilizado en una amplia gama de sectores, desde la agricultura hasta la fabricación y la distribución (Zrnić, N., Đorđević, M., & Gašić, V., 2022).

La historia de las cintas transportadoras está marcada por avances tecnológicos que han mejorado su rendimiento y capacidad. En sus inicios, eran operadas manualmente y se utilizaban principalmente en minas y canteras. Sin embargo, con el paso del tiempo, se implementaron motores eléctricos y sistemas automatizados, lo que permitió una mayor eficiencia y capacidad de transporte.

Hoy en día, desempeñan un papel crucial en la cadena de suministro global. Su objetivo principal es facilitar el movimiento eficiente de materiales y productos a lo largo de una línea de producción o en un entorno de almacenamiento. Al utilizar tecnologías avanzadas, como sensores y sistemas de control automatizados, las cintas transportadoras pueden clasificar, separar y transportar productos de manera precisa y confiable.

Los beneficios son evidentes: aumentan la productividad, reducen los costos operativos, minimizan los errores y mejoran la seguridad laboral. Además, al permitir un flujo continuo de materiales, las cintas transportadoras contribuyen a optimizar los tiempos de producción y la eficiencia general de los procesos industriales.

En definitiva, se ha recorrido un largo camino desde sus inicios en la industria minera hasta convertirse en una herramienta esencial en diversas industrias. Su historia de innovación y desarrollo continúa, impulsada por la búsqueda de soluciones logísticas más eficientes y rentables. Con objetivos centrados en la automatización, la optimización del flujo de materiales y la mejora de la productividad, las cintas transportadoras siguen siendo un componente fundamental en el mundo industrial actual.

1.1. Legislación sobre cintas transportadoras

La legislación aplicada a las cintas transportadoras incluye diversas normas y regulaciones que buscan garantizar la seguridad en su uso. Entre estas normas se encuentran, Orden de 23 de febrero de 1996 por la que se modifica la "*ITC 04.6.03*" (BOE, 7 de marzo de 1996), norma "*UNE-EN 12882:2008*" (AENOR, 2016) y la "*NTP 89*" (INSST, 2003).

La Orden de 23 de febrero de 1996 modifica el punto 6.º, denominado "*Cintas transportadoras*", de la "*ITC 04.6.03 del Reglamento General de Normas Básicas de Seguridad Minera*" (BOE, 7 de marzo de 1996). Esta modificación tiene como objetivo establecer precauciones contra incendios en las labores subterráneas relacionadas con las cintas transportadoras.

Por su parte, la norma "*UNE-EN 12882:2008*" establece requisitos de seguridad eléctricos y de protección contra la inflamabilidad para cintas transportadoras de usos generales (AENOR, 2016). Esta norma se enfoca en aspectos como la descarga de electricidad estática y los peligros asociados con la presencia de pequeñas llamas en el recubrimiento de la cinta transportadora.

La "*NTP 89*" (INSST, 2003), por su parte, se centra específicamente en las cintas transportadoras de materiales a granel. Esta norma aborda diversos puntos de seguridad, incluyendo los riesgos asociados con la inmovilización de una cinta mientras el mecanismo impulsor continúa funcionando, lo que puede generar calentamiento en la zona de contacto con el tambor o cilindro en movimiento. Además, también se aborda el peligro de propagación de llamas a lo largo de la cinta transportadora si ha estado expuesta a una fuente de energía alta, como un fuego.

En resumen, estas normas y regulaciones proporcionan pautas y requisitos para prevenir peligros relacionados con la electricidad estática, pequeñas llamas, atascos o inmovilizaciones, y propagación de fuego en las cintas transportadoras. Su objetivo principal es salvaguardar la seguridad de los trabajadores y minimizar los riesgos asociados con estas instalaciones.

1.2. Objetivos fijados en este proyecto

El objetivo de este proyecto consiste en el diseño y simulación de un prototipo de cinta transportadora clasificadora automatizada empleando para ello un microcontrolador y diferentes sensores y actuadores que permiten seleccionar y clasificar objetos a partir de diferentes características. En particular, se emplea un microcontrolador Arduino para automatizar y controlar el funcionamiento de la cinta, este se encarga de recopilar la información de los sensores, procesarla y utilizarla para controlar los actuadores. Mediante la programación del microcontrolador, es posible establecer reglas y criterios para la clasificación de los objetos según sus características.

La característica principal que estudiar en este proyecto para la clasificación de objetos será el color, dado que los sensores encargados de detectar variaciones de color se encuentran presentes en diversas industrias y ofrecen amplías posibilidades de control y organización. Los diversos sensores que se emplean en este proyecto desempeñan el papel de detectar objetos en la cinta transportadora y recopilar información sobre características como posición, además de color. Se utilizan por tanto en este proyecto, sensores de detección de obstáculos y sensores de detección de color para realizar la clasificación.

Por otro lado, los actuadores se encargan de controlar la velocidad y el movimiento de los elementos clasificadores, como ruedas o brazos mecánicos, así como de interactuar con los objetos para clasificarlos. En este caso, se emplean motores como actuadores.

Empleando un microcontrolador de tipo Arduino en conjunto con sensores y actuadores en el prototipo de la cinta se busca automatizar y controlar el proceso de clasificación, mejorando la eficiencia y precisión del sistema con el fin de estudiar la viabilidad de implementar este sistema en proyectos de mayor escala.

1.2.1. Elección del sensor de color

En este proyecto de Arduino, se empleará un sensor de color para mejorar la capacidad de detección y clasificación de colores en aplicaciones específicas.

Un sensor de color es un dispositivo que permite detectar y medir los componentes de luz de diferentes colores presentes en un objeto o entorno. Estos sensores han sido utilizados a lo largo de la historia en la industria para realizar tareas de clasificación y control de calidad basadas en colores, desde la industria textil hasta la robótica. En este proyecto se explora cómo utilizar el sensor de color en combinación con Arduino para realizar tareas de clasificación y control de manera precisa y eficiente.

Los sensores de color se clasifican en dos tipos principales: "*sensores RGB*" y sensores de luz tricolor. Cada tipo tiene su propio método de detección y ofrece diferentes aplicaciones (Electronic, 2020).

Los "*sensores RGB*" utilizan una fuente de luz de longitud de onda amplia para iluminar el objeto y luego distinguen los tres colores primarios (rojo, verde y azul) en el receptor. Estos sensores son capaces de medir el nivel de intensidad y composición de cada color. Son utilizados en aplicaciones como la calibración de colores en impresión, control de iluminación y sistemas de visión artificial.

Por otro lado, los sensores de luz tricolor emplean fuentes de luz independientes para iluminar el objeto con luz roja, verde y azul de forma individual. Estos sensores miden la cantidad de luz reflejada en cada componente de color y permiten obtener información precisa sobre el espectro de colores. Son comúnmente utilizados en aplicaciones de clasificación de colores en la industria, como la detección de productos defectuosos en una línea de producción, la clasificación de objetos por su color en sistemas de automatización, entre otros.

Algunos ejemplos de sensores de color basados en este último principio de funcionamiento son los siguientes y en los que se describirán diversas aplicaciones en las que se pueden implementar este tipo de sensores.

- **Sensor de color ADJD-S371**: Este sensor utiliza una combinación de fotodiodos y filtros de color para medir con precisión los componentes de color rojo, verde y azul de la luz incidente. Proporciona salidas analógicas que representan la intensidad de cada componente de color.

  Es adecuado para aplicaciones en las que se requiere una detección precisa del color, como en sistemas de análisis de color en la industria alimentaria para clasificar frutas o vegetales según su madurez o calidad. También se puede utilizar en sistemas de monitoreo de calidad del agua para detectar cambios en la composición química basados en el color.
- **Sensor de color VEML6040**: Este sensor de luz ambiental y color permite medir la intensidad de la luz ambiental, así como la información de color en formato "*RGBW*" (rojo, verde, azul, blanco). Se comunica con Arduino a través de un "*protocolo I2C*".

  Puede ser empleado en aplicaciones de detección de luz ambiental en dispositivos móviles y relojes inteligentes para ajustar automáticamente el brillo de la pantalla según las condiciones de iluminación. También se puede utilizar en sistemas de iluminación automática en hogares o edificios, donde se ajusta el color y la intensidad de la luz según las necesidades de los ocupantes.
- **Sensor de color AS726x**: Esta serie de sensores proporciona mediciones de color en formato "*XYZ*" (coordenadas tricromáticas) y se comunica con Arduino a través de un "*protocolo I2C*". Cada sensor de la serie AS726x se especializa en diferentes rangos de longitud de onda, lo que permite mediciones más precisas en diferentes espectros de color.

  Son ideales para aplicaciones en la industria textil, donde se requiere una medición precisa del color de los tejidos. También pueden utilizarse en sistemas de control de calidad de pinturas y tintas, clasificación de productos químicos basados en su composición cromática, y en la creación de dispositivos de análisis de color portátiles.
- **Sensor de color TCS3200**: Este sensor utiliza una matriz de fotodiodos para detectar la intensidad de luz en diferentes canales de color (rojo, verde, azul). Proporciona salidas digitales que indican la presencia y la intensidad de los colores detectados.

  Puede ser utilizado en proyectos de clasificación de objetos por colores, como en sistemas de conteo de piezas en una línea de producción, clasificación de productos agrícolas según su madurez, control de calidad, robótica y seguimiento de líneas, entre otros.

Cada tipo de sensor de color ofrece ventajas específicas según las necesidades del proyecto. La elección del sensor adecuado dependerá de los requisitos de precisión, velocidad y aplicaciones específicas (Logicbus, 2019).

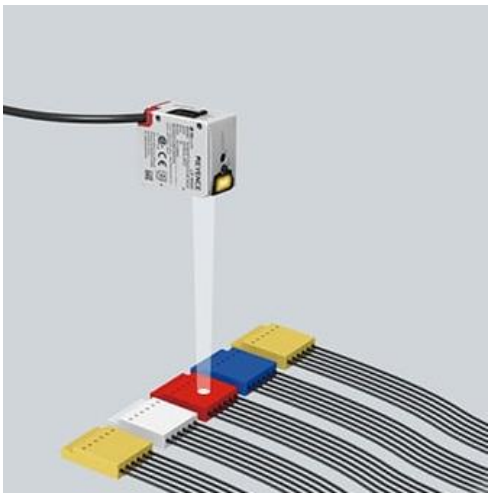


Figura 1. Sensor de color en la industria (Keyence, 2023)

En conclusión, la elección del "*sensor de color TCS3200*" se basa en una serie de factores que lo hacen altamente favorable para su uso con Arduino. En primer lugar, su fácil integración con la plataforma Arduino lo convierte en una opción accesible para los aficionados y profesionales que desean desarrollar proyectos relacionados con la detección de colores.

Además, ofrece un amplio rango de detección de colores, lo que le permite identificar y distinguir una gran variedad de tonos y matices. Esto lo hace adecuado para aplicaciones donde se requiere una medición precisa y confiable del color.

Otro aspecto destacable es su amplio rango de aplicaciones. Desde el control de iluminación y la clasificación de objetos hasta el control de calidad y la robótica, se adapta a diversas industrias y proyectos, brindando flexibilidad y versatilidad en su implementación.

Adicionalmente, este sensor es ampliamente comercializado y se encuentra disponible en el mercado a un bajo costo. Esto facilita su adquisición y lo convierte en una opción atractiva para aquellos que buscan soluciones económicas sin comprometer la calidad y la funcionalidad.

2. Conceptos y herramientas empleadas

2.1. Funcionamiento de Arduino: sensores y actuadores

En este apartado, se abordará el funcionamiento de Arduino, así como la comunicación entre sensores y actuadores. Se estudiará el papel de Arduino como dispositivo central que permite la interacción con el entorno, mediante la captura de datos a través de los sensores. Asimismo, se analizará la función de los actuadores en la ejecución de acciones concretas basadas en dichos datos. Se analizará la selección y aplicación adecuada de los sensores y actuadores para cumplir con los objetivos establecidos en el proyecto.

2.1.1. Arduino MEGA 2560 rev3

Arduino es una plataforma de desarrollo de código abierto que ha experimentado una evolución constante desde su creación. A lo largo de su historia, se han desarrollado diferentes placas de Arduino con características y funcionalidades específicas para adaptarse a las necesidades cambiantes de los usuarios. A continuación, se presenta una breve descripción de algunos de los tipos más destacados de placas de Arduino:

- **Arduino Uno**: Fue la primera placa de Arduino lanzada en 2005 y se convirtió en una de las placas más populares y ampliamente utilizadas en la comunidad Arduino.
  Está equipado con un "*microcontrolador ATmega328P*", que ofrece un rendimiento confiable. Cuenta con 14 pines digitales de entrada/salida, de los cuales 6 pueden generar señales "*PWM*" para el control de motores y luces. Además, dispone de 6 entradas analógicas para la lectura de sensores. El Arduino Uno es perfecto para principiantes debido a su facilidad de uso, amplia disponibilidad de "*shields*" (módulos de expansión) y abundantes recursos educativos.
- **Arduino Mega**: El Arduino Mega es una placa de mayor tamaño y más potente en comparación con el Arduino Uno. Está basado en el "*microcontrolador ATmega2560*", que ofrece un mayor número de pines digitales y entradas analógicas. Con 54 pines digitales de entrada/salida (15 de ellos con capacidades PWM, "Pulse Width Modulation") y 16 entradas analógicas, el Arduino Mega es ideal para proyectos complejos que requieren una mayor cantidad de conexiones y recursos.

- **Arduino Nano**: El Arduino Nano es una versión compacta de la placa Arduino. Aunque es pequeño en tamaño, sigue siendo una placa potente con un "*microcontrolador ATmega328P*". Ofrece 22 pines digitales de entrada/salida (6 de ellos con capacidades PWM) y 8 entradas analógicas. Su tamaño reducido lo hace perfecto para proyectos con limitaciones de espacio, y es compatible con muchos "shields" diseñados para el Arduino Uno.
- **Arduino Leonardo**: Se diferencia de otras placas por su capacidad de emular dispositivos USB, lo que permite una fácil integración con teclados, ratones y otros periféricos. Está basado en el "*microcontrolador ATmega32U4*", que permite la emulación USB directamente desde la placa. Además de sus 20 pines digitales de entrada/salida (7 de ellos con capacidades PWM) y 12 entradas analógicas, el Arduino Leonardo es ideal para proyectos que requieren interacción con dispositivos USB.
- **Arduino Due**: Esta placa se destaca por su mayor potencia de procesamiento y capacidad de ejecutar aplicaciones más complejas.
  Utiliza el "*microcontrolador ARM Cortex-M3*" de 32 "bits", que ofrece un mayor rendimiento, velocidad de reloj y capacidad de memoria en comparación con los microcontroladores utilizados en las placas anteriores. Con 54 pines digitales de entrada/salida (12 de ellos con capacidades PWM) y 12 entradas analógicas, el Arduino Due ofrece una mayor capacidad de procesamiento y memoria para proyectos más complejos.
- **Arduino Mega ADK**: Es similar a Arduino Mega, pero incluye características adicionales para permitir la comunicación con dispositivos Android.
  Combina las características del Arduino Mega con capacidades adicionales para la comunicación con teléfonos móviles o tablets Android. Es especialmente útil para proyectos de "*Internet de las cosas*" (IoT) que requieren interacción con teléfonos móviles o tablets Android.

Se ha elegido para este proyecto la placa "*Arduino Mega 2560 Rev3*" debido a sus aplicaciones en educación y su elevado número de pines destinados a conectar los diferentes sensores necesarios en este ejemplo.

Figura 2. Arduino Mega 2560 Rev3 (Bricogeek, Arduino MEGA 2560 rev3, 2023)

La principal función de la placa "*Arduino Mega 2560 Rev3*", la cual está basada en el "*microcontrolador ATmega2560*", es actuar como el cerebro central de un sistema electrónico, permitiendo la interacción con sensores, actuadores y otros componentes (Arduino, 2023). A continuación, se detallarán algunos de sus componentes principales.

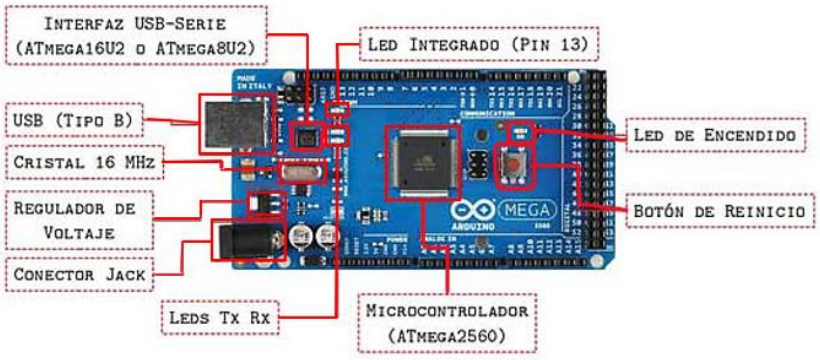


Figura 3. Componentes principales de la placa Arduino Mega Rev3 (Guerra Carmenate, Arduino Mega 2560 el hermano mayor de Arduino UNO, 2022)

El "**microcontrolador ATmega2560"** es el núcleo encargado de grabar, almacenar y ejecutar el código programado (Guerra Carmenate, Arduino Mega 2560 el hermano mayor de Arduino UNO, 2022). Cuenta con una "*memoria EEPROM*" de "*4KB*" (4096 bytes), que es una memoria no volátil. Esta memoria es una forma de almacenamiento de datos que retiene su información incluso cuando el microcontrolador se apaga o pierde energía. Esto permite almacenar datos importantes de forma permanente, como configuraciones, calibraciones o datos de estado, que se conservarán incluso después de reiniciar o apagar el microcontrolador (Arduino D. , 2023).

Los **pines digitales** son puntos de conexión que pueden ser utilizados para enviar o recibir señales digitales. Estos pines pueden operar en dos estados: “HIGH” (alto) o “LOW” (bajo), representando valores binarios 1 y 0, respectivamente. Los pines digitales se utilizan para leer y escribir datos digitales, como el estado de un interruptor o el encendido/apagado de un “*LED*”. En una placa Arduino típica, los pines digitales se numeran y se identifican con un número, por ejemplo, “D0”, “D1”, “D2”, etc.

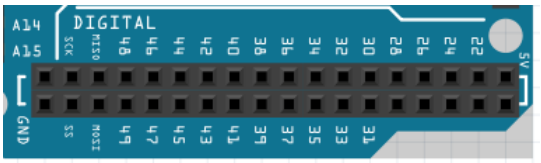


Figura 4. Pines digitales de la placa Arduino Mega 2560 Rev3 (*imágen de elaboración propia*)

Algunos de los pines de esta placa cuentan con modulación por ancho de pulso (PWM, por sus siglas en inglés) es una técnica utilizada en Arduino para generar señales analógicas utilizando pines digitales. Aunque los pines digitales son binarios (HIGH o LOW), el “*PWM*” permite simular señales analógicas generando una onda cuadrada con un ciclo de trabajo variable. Esto significa que se puede ajustar el tiempo en que la señal está en estado alto (HIGH) y en estado bajo (LOW), creando así un nivel de voltaje promedio entre 0 y 5 voltios. Los “*pines PWM*” en una placa Arduino están etiquetados con "~" y tienen un número asociado, por ejemplo, “~3”, “~5”, “~9”, etc. Estos pines se utilizan comúnmente para controlar la intensidad luminosa de un “*LED*”, la velocidad de un “*motor DC*” o para generar señales de audio (Guerra Carmenate, Señal PWM con Arduino y analogWrite, 2022).

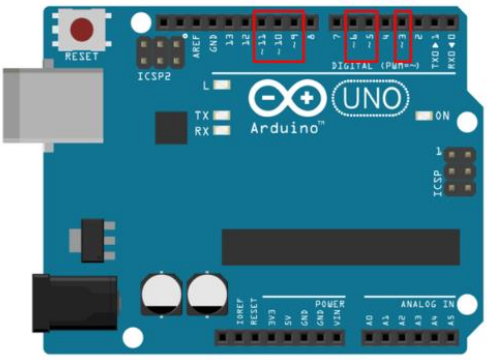


Figura 5. Representación de la placa Arduino señalando las entradas digitales con modulación PWM (Guerra Carmenate, Señal PWM con Arduino y analogWrite, 2022)

Las “**entradas analógicas**” en Arduino permiten medir y leer valores de voltaje continuo en un rango específico. A diferencia de los pines digitales que solo pueden tener dos estados, las entradas analógicas pueden tomar valores en un rango continuo, generalmente entre 0 y 5 voltios en placas Arduino convencionales. Estos pines se utilizan para leer sensores analógicos, como sensores de luz, temperatura o humedad, que proporcionan una salida proporcional a la cantidad medida. En una placa Arduino, las entradas analógicas se numeran y se identifican con la letra "A" seguida de un número, por ejemplo, “A0”, “A1”, “A2”, etc.

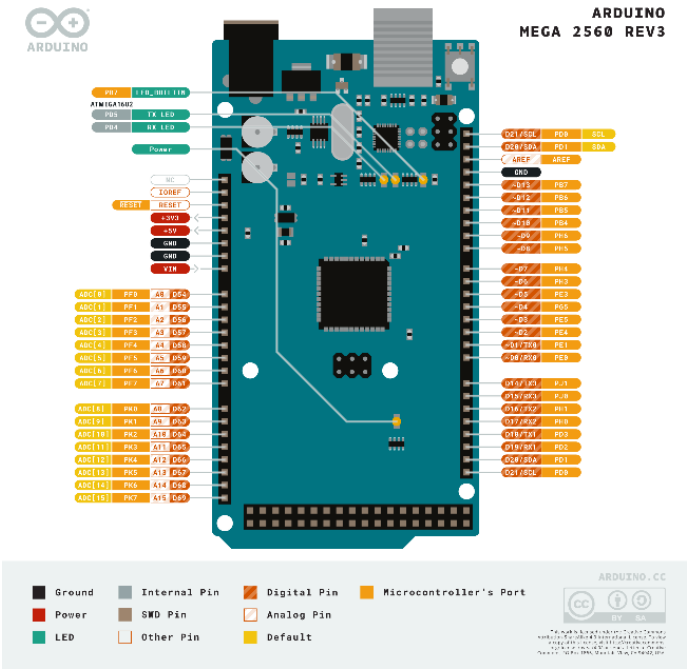


Figura 6. Diagrama de pines de la placa Arduino MEGA 2560 Rev3 (Arduino D. , 2023)

Esta placa cuenta también con un “**conector USB**” que es empleado tanto para alimentar la placa con los 5 voltios suministrados por el ordenador como para cargar los programas al microcontrolador. La “*interfaz USB-Serie*” señalada en la “*Figura 3*” es un microcontrolador adicional de la placa que se encarga de actuar como intermediario entre el “conector USB” y el microcontrolador principal de la placa (Guerra Carmenate, Arduino Mega 2560 el hermano mayor de Arduino UNO, 2022).

El “**conector Jack**”, señalado en la “*Figura 3*” se emplea para suministrar alimentación adicional a la placa y los sensores conectados cuando es necesario emplear más de 5 voltios o no se encuentra conectada la placa al ordenador. Se detallarán más formas de alimentación en el punto “*2.3*”.

Los "**reguladores de voltaje**" se encargan de suministrar con un voltaje adecuado a todos los componentes de un sistema. En el caso del Arduino MEGA, se pueden identificar dos reguladores de voltaje en su esquema. El regulador principal es el "*LD1117S50*", encargado de proporcionar alimentación al microcontrolador y a los pines "5V" de la placa. Este regulador toma un voltaje de entrada desde el "*conector Jack*" o el "*pin VIN*" ("Voltage Input") y suministra una salida de "5V". El segundo regulador es el "*LP2985-33DBVR*", este ofrece un voltaje que ofrece una salida de "3.3V" y está conectado al "pin 3.3V".

En cuanto al "**cristal oscilador**" y al "**resonador cerámico**", son componentes utilizados para generar señales de voltaje con una frecuencia de "16MHz". Estos componentes son empleados por el "*ATmega2560*" para generar las señales de reloj las cuales son necesarias para su funcionamiento adecuado.

Los cuatro "**leds**" presentes en la placa Arduino se encargan de proporcionar retroalimentación visual sobre el estado y la actividad de la placa durante el desarrollo y la depuración de proyectos.

Por último, la placa cuenta con un "**botón de reinicio**" el cual permite reiniciar el sistema, permitiendo ejecutar nuevamente el código guardado.

Para consultar más datos técnicos sobre esta placa se adjunta en el "*ANEXO 3. HOJA DE DATOS DE LOS COMPONENTES*" la correspondiente ficha técnica.

2.1.2. Protoboard

Una "*protoboard*", también conocida como "*placa de pruebas*" o "*breadboard*". Se trata de una placa rectangular con una matriz de orificios conectados eléctricamente entre sí en patrones preestablecidos. Estos orificios permiten insertar y conectar de manera temporal componentes electrónicos, como resistencias, capacitores, cables y microcontroladores como Arduino.

Para utilizar una "*breadboard*", se insertan las clavijas de los componentes eléctricos o los cables de puente en los agujeros correspondientes. Los resortes de los agujeros evitan que los componentes se suelten. Algunos de los agujeros de las placas de circuito están conectados entre sí, y es importante conocer estas conexiones para obtener el circuito deseado. En la "*Figura 7*" se muestra un ejemplo de cómo se realiza esta conexión típicamente.

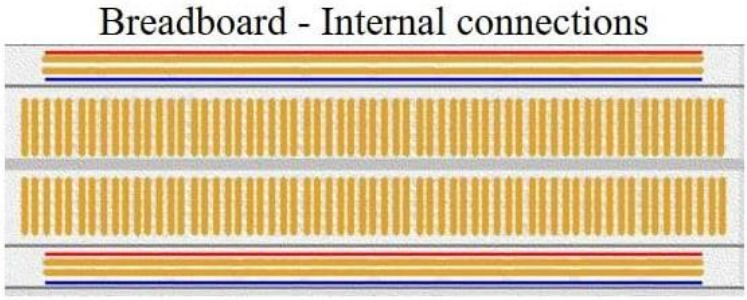


Figura 7. Protoboard y su esquema de conexionado interno
(José, Protoboard, ¿Qué es y cómo se usa?, 2023)

Los pines rojo y azul se utilizan como alimentación "*VCC*" (por ejemplo, +5 V) y "*GND*" (tierra) respectivamente. Dado que muchas partes tienen que estar conectadas a dos cables, es fácil hacer conexiones en estas líneas si es necesario. Como se muestra en la "*Figura 7*", la parte central del tablero divide las conexiones entre los agujeros centrales (José, Protoboard, ¿Qué es y cómo se usa?, 2023).

La placa de pruebas es compatible con los pines y conectores estándar de Arduino, lo que facilita la conexión de los componentes al microcontrolador. Además, proporciona una plataforma segura para realizar pruebas y experimentos, evitando daños tanto a los componentes como a la placa de Arduino.

2.1.3. Motor paso a paso 28BYJ-48 con driver ULN2003

Arduino es compatible con una amplia gama de motores que permiten controlar el movimiento en diversos proyectos. Algunos de los motores más comunes son:

- **Motores de corriente continua (DC)**: Estos motores funcionan mediante la aplicación de una corriente continua, lo que les permite girar en ambas direcciones. Son versátiles y se utilizan en una amplia variedad de aplicaciones, como robótica, vehículos controlados por Arduino y sistemas de automatización.
- **Motores paso a paso**: Estos motores se utilizan cuando se requiere un control preciso del movimiento, como en impresoras 3D, "máquinas CNC" y sistemas de posicionamiento. Los motores paso a paso se mueven en incrementos discretos (pasos) y pueden girar en una dirección específica.
- **Servomotores**: Los servomotores son motores de corriente continua que incluyen un sistema de control de posición y velocidad. Se utilizan comúnmente en aplicaciones que requieren movimientos precisos, como en brazos robóticos, drones y mecanismos de dirección de vehículos controlados por Arduino.
- **Motores de engranajes**: Estos motores proporcionan un mayor torque y se utilizan en proyectos que requieren una mayor fuerza de salida. Se pueden encontrar en robots de combate, vehículos todoterreno y otros dispositivos que necesitan superar obstáculos o realizar tareas que requieren más potencia.

Se selecciona el motor paso a paso para el proyecto debido a sus características y funcionalidades. Algunas de las ventajas que ofrecen estos motores que resultan adecuadas son:

- **Precisión en el control de posición**. Los motores paso a paso permiten un control preciso del movimiento, lo que es importante en una cinta transportadora. Pueden moverse en incrementos discretos, lo que facilita la sincronización con otros componentes del sistema, como sensores y actuadores.
- **Permiten girar en ambas direcciones**, lo que es esencial para una cinta transportadora. Esto permite el avance y retroceso de la cinta según sea necesario para el transporte de objetos en diferentes direcciones.

- **Capacidad de carga**. Estos motores tienen una alta capacidad de carga, lo que significa que pueden mover objetos pesados en la cinta transportadora sin dificultad. Esto es especialmente importante si se requiere transportar objetos de gran tamaño o peso.
- **Control de velocidad**: Los motores paso a paso permiten controlar la velocidad de rotación de manera precisa. Esto es beneficioso para ajustar la velocidad de la cinta transportadora según las necesidades del proyecto, ya sea para un transporte más rápido o lento.
- **Integración con Arduino**. Los motores paso a paso se pueden controlar fácilmente con Arduino utilizando controladores específicos, como los controladores de motor paso a paso basados en drivers. Estos controladores proporcionan una interfaz sencilla para enviar las señales adecuadas desde Arduino y controlar el motor de manera eficiente.

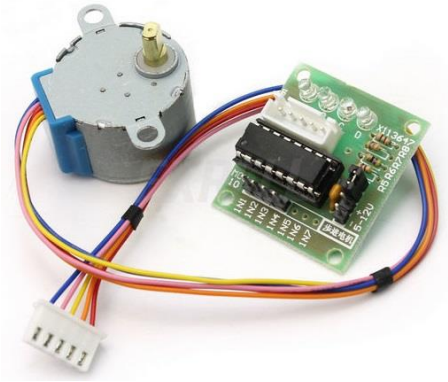


Figura 8. Motor paso a paso 28BYJ con driver ULN2003

A continuación, se detallan algunas especificaciones del motor seleccionado para el proyecto. El "*motor paso a paso 28BYJ-48*" es un motor unipolar de imán permanente, lo que lo hace un motor ampliamente utilizado debido a su eficiencia, simplicidad y bajo costo, empleándose por ello en proyectos de electrónica y robótica.

El funcionamiento del motor paso a paso se basa en el principio de electromagnetismo. El motor cuenta con cuatro pares de bobinas y un imán permanente. Al enviar pulsos eléctricos secuenciales a las bobinas, se crea un campo magnético que interactúa con el imán, provocando el movimiento del eje en incrementos paso a paso. La secuencia de los pulsos determina la dirección y velocidad del movimiento (Programarfacil, 2023).

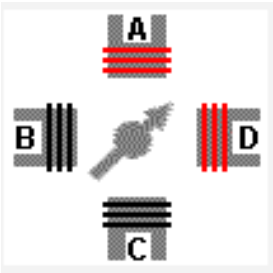


Figura 9. Alimentación de las fases en un motor paso a paso
(Prometec, Prometec: motores paso a paso, 2015)

Además, se trata de un motor paso a paso unipolar, el cual cuenta con dos bobinas por fase, y cada bobina se puede activar por separado mediante una secuencia específica de señales eléctricas. Estas señales hacen que el motor avance en pasos discretos. La configuración unipolar permite un control más sencillo y una mayor facilidad de uso, ya que solo se requiere una corriente en cada bobina para su funcionamiento. Cada uno de los colores indicados en la "*Figura 10*" corresponden con los cables mostrados en la "*Figura 8*".

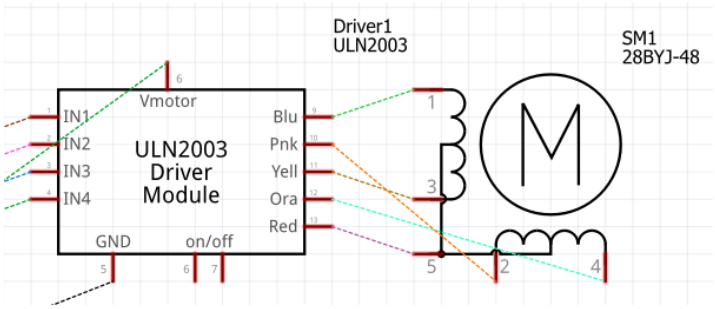


Figura 10. Esquemático del motor paso a paso y driver que representa la disposición unipolar del cableado de las bobinas del motor paso a paso (*imagen de elaboración propia*)

Este motor posee un ángulo de paso de "5.625 grados", lo que significa que se requieren 64 pasos para dar una vuelta completa (360 grados). Tiene una relación de reducción de "1/64", lo que significa que se necesita un impulso eléctrico de 64 pulsos para que el eje del motor gire una vuelta completa. Además, tiene una velocidad de rotación relativamente baja, lo que lo hace adecuado para aplicaciones que requieren movimientos precisos y controlados.

Se muestra en la siguiente "*Figura 11*" una representación de la caja de engranajes reductores (Bitwiser, Paso a Paso (unipolar) 28BYJ-48 con Driver ULN2003, 2017).

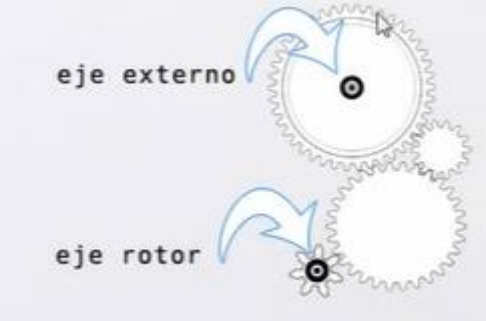


Figura 11. Representación del conjunto de engranajes reductores del interior del motor paso a paso (Bitwiser, Paso a Paso (unipolar) 28BYJ-48 con Driver ULN2003, 2017)

Como se muestra en la "*Figura 11*", el eje del rotor se encuentra conectado mediante una serie de engranajes reductores con el fin de reducir la velocidad de giro y aumentar el torque o fuerza aplicada para el giro. La salida de la caja de reducción es el eje externo accesible para el uso en los diferentes proyectos (Bitwiser, Paso a Paso (unipolar) 28BYJ-48 con Driver ULN2003, 2017).

Para controlar el "*motor paso a paso 28BYJ-48*", se utiliza el "*driver ULN2003*". Este driver actúa como un controlador de potencia y proporciona los pulsos de excitación necesarios para el movimiento del motor. Además, el "*driver ULN2003*" protege al microcontrolador, como Arduino, de los picos de corriente generados por el motor. El driver también simplifica la conexión del motor al microcontrolador, ya que ofrece una interfaz de entrada y salida fácil de usar (Prometec, Prometec: motores paso a paso, 2015).

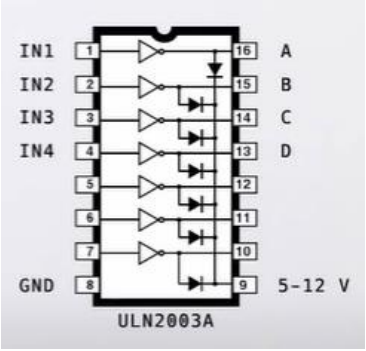


Figura 12. Representación de las conexiones del driver ULN2003 (Bitwiser, Paso a Paso (unipolar) 28BYJ-48 con Driver ULN2003, 2017)

Mediante los pines de entrada (de "IN1" a "IN4") representados en la "*Figura 12*" controlar las salidas que permiten la circulación de corriente con el fin de energizar cada una de las bobinas. De esta manera desde Arduino se envía baja corriente y este circuito integrado se encarga de aumentar dicha corriente con el fin de alimentar adecuadamente al motor. Dicha corriente de salida alcanza los "500 mA" (miliamperios).

2.1.4. OLED 1.3’ SSH1106

La pantalla “*OLED 1.3' SSH1106 (128x64)*” es una pantalla pequeña y compacta que utiliza la tecnología “*OLED*” (“Organic Light-Emitting Diode”) para mostrar gráficos y texto. Tiene una resolución de “128x64 píxeles”, lo que permite una representación clara y nítida de la información.

Figura 13. Oled 1.3' SSH1106 (Bricogeek, Bricogeek: Oled 1.3', 2023)

El funcionamiento de la “pantalla OLED” se basa en la emisión de luz por parte de los diodos orgánicos presentes en cada píxel. Estos diodos se activan individualmente para crear los patrones y textos deseados en la pantalla. Debido a la tecnología “OLED”, la pantalla ofrece un alto contraste y un amplio ángulo de visión, lo que la hace adecuada para diversas aplicaciones.

La “*pantalla OLED 1.3' SSH1106*” se puede comunicar con Arduino a través del “*protocolo I2C*” (“Inter-Integrated Circuit”) o el “*protocolo SPI*” (“Serial Peripheral Interface”). El “*protocolo I2C*” emplea las líneas de comunicación “*SDA*” (cable de comunicación) y “*SCL*” (cable del reloj).

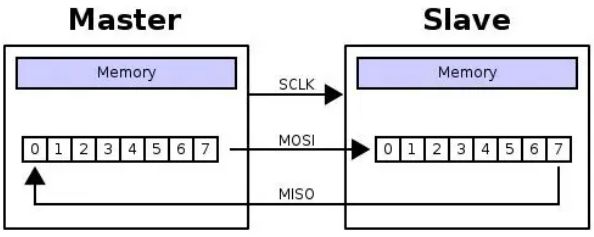


Figura 14. Estructura general del protocolo SPI (García González, ¿Cómo funciona el protocolo SPI?, 2014)

En este proyecto por recomendaciones de fabricante se empleará "*SPI*" (Bricogeek, Bricogeek: Oled 1.3', 2023). Esta interfaz permite una comunicación de datos bidireccional y de alta velocidad entre el Arduino y la pantalla "OLED" mediante cuatro cables: el cable de datos (MOSI), el cable de reloj (SCK), el cable de selección de dispositivo (SS) y el cable de datos de retorno (MISO).

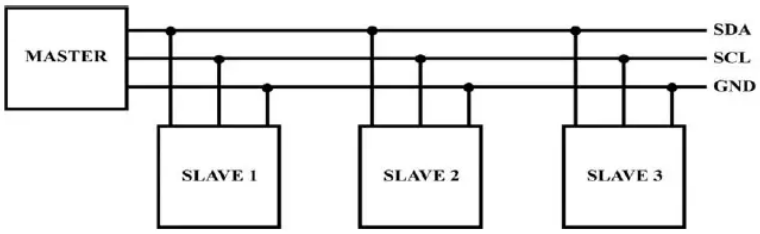


Figura 15. Representación gráfica de bus de comunicación I2C (García González, Comunicación entre Arduino y Digispark por medio de I2C, 2017)

Algunas características destacadas de la "*pantalla OLED 1.3' SSH1106*" incluyen su bajo consumo de energía, lo cual es ideal para proyectos alimentados por baterías, y su rápido tiempo de respuesta, lo que permite una actualización suave de la información en pantalla. Además, la "pantalla OLED" es compatible con una amplia variedad de bibliotecas y librerías, lo que facilita su integración y programación en proyectos de Arduino.

2.1.5. Sensor de colores TCS3200

El "*sensor de color TCS3200*" es un sensor que se utiliza para detectar y medir la intensidad de luz de diferentes longitudes de onda en el rango de los colores del espectro visible. Este sensor se compone de una matriz de sensores fotosensibles un circuito integrado de procesamiento de señal (Llamas, Luis Llamas: Sensor de colores TCS3200, 2016).

Figura 16. Sensor de colores TCS3200 (Bricogeek, Bricogeek: Sensor de colores TCS3200, 2023)

El funcionamiento del "sensor TCS3200" se basa en la detección de luz mediante los fotodiodos y la conversión de esta luz en una señal eléctrica proporcional a la intensidad del color. El sensor cuenta con cuatro filtros de colores: rojo, verde, azul y blanco, que permiten seleccionar la longitud de onda de luz que se desea medir. La salida del sensor devuelve un pulso de duración proporcional a la intensidad del color detectado (Bricogeek, Bricogeek: Sensor de colores TCS3200, 2023).

La comunicación entre el sensor y Arduino se realiza a través de conexiones digitales. El sensor tiene pines de salida que proporcionan señales digitales para la frecuencia de la señal y la salida de datos. Estas señales se conectan a los pines de entrada digital del Arduino para la lectura y procesamiento de los datos.

Entre las características destacadas se encuentra su amplio rango de detección de colores, capacidad para trabajar en diferentes condiciones de iluminación y su facilidad de integración con Arduino y otros microcontroladores. Además, se puede programar para adaptarse a diferentes aplicaciones, como sistemas de clasificación de colores, control de iluminación ambiental o robótica.

2.1.6. Detector de obstáculos con sensor infrarrojo FC-51

El detector de obstáculos con sensor infrarrojo es un dispositivo utilizado para detectar la presencia de objetos en su entorno. Este sensor se basa en la tecnología infrarroja, emitiendo un haz de luz infrarroja y midiendo la cantidad de luz reflejada.

Figura 17. Detector de obstáculos con sensor infrarrojo

El sensor está compuesto por un emisor infrarrojo y un receptor. El emisor emite pulsos de luz infrarroja hacia el área circundante, y el receptor recibe la luz reflejada por los objetos cercanos. Cuando un objeto se encuentra dentro del rango de detección, la cantidad de luz reflejada cambia, lo que indica la presencia de un obstáculo (Llamas, Luis Llamas: detector de obstáculos con sensor infrarrojos, 2016).

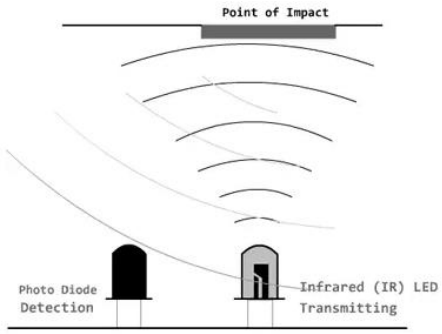


Figura 18. Esquema de funcionamiento de emisor infrarrojo y receptor (Llamas, Luis Llamas: detector de obstáculos con sensor infrarrojos, 2016)

La comunicación del detector de obstáculos con sensor infrarrojo con Arduino u otros microcontroladores se realiza a través de puertos digitales. El sensor emite una señal digital cuando detecta un obstáculo, lo que permite al microcontrolador realizar acciones en respuesta a dicha detección. La sensibilidad y el rango de detección del sensor se pueden ajustar mediante potenciómetros integrados, lo que facilita su adaptación a diferentes entornos y requisitos específicos del proyecto.

Este tipo de sensor es ampliamente utilizado en aplicaciones de robótica, domótica y sistemas de seguridad. Se puede emplear para evitar colisiones, controlar la velocidad de movimiento de un robot, activar o desactivar dispositivos en función de la presencia de objetos, entre otros usos.

2.2. Comunicación con Arduino

La comunicación entre Arduino y un ordenador se realiza a través de la comunicación serial. En esta forma de comunicación, los datos se envían y reciben bit a bit empleando tres líneas: la línea de recepción de datos (RX), la línea de transmisión de datos (TX) y la línea común (GND). Esta comunicación se basa en los puertos seriales conocidos como "*UART*" (receptor/transmisor universal asincrónico), los cuales convierten los datos en una secuencia de bits para su transmisión o recepción a una velocidad determinada para la comunicación serie entre dos dispositivos (MCI, 2023).

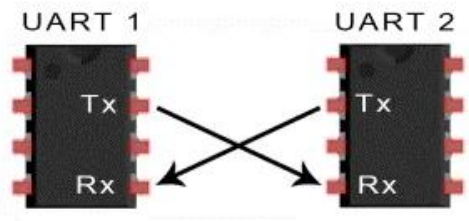


Figura 19. Esquema de comunicación serie entre dos dispositivos (MCI, 2023)

Las placas de Arduino cuentan con "*unidades UART*" que operan a "*nivel TTL*", "Transistor-Transistor Logic" (empleando el rango de voltajes 0-5V), lo que las hace compatibles con la "conexión USB". Además, la mayoría de los Arduinos disponen de un conector USB que permite una conexión instantánea con el ordenador. Esto facilita la transferencia de datos y la programación de Arduino, permitiendo enviar comandos, recibir información y realizar diversas tareas de automatización y control desde el ordenador.

Una vez se conoce la forma de comunicación entre la placa Arduino y los ordenadores serán necesarias otras dos condiciones para que dicha comunicación sea legible por ambas partes. El lenguaje de comunicación y la velocidad de transmisión de los datos.

El lenguaje de comunicación se realiza empleando el "*código ASCII*", que es una codificación estándar para representar caracteres mediante números. Sin embargo, en la actualidad también es posible emplear una extensión llamada "*Unicode*". Esta extensión permite el uso de caracteres más amplios y diversos, incluyendo aquellos que no están presentes en la tabla "*ASCII*" original, como letras acentuadas, caracteres especiales y alfabetos diferentes. Esta extensión facilita la comunicación y la representación de diversos idiomas y caracteres en la interacción entre Arduino y el ordenador.

En la comunicación serial entre la placa Arduino y el ordenador, la velocidad de comunicación juega un papel fundamental. Esta velocidad se mide en "*baudios*" y determina la cantidad de "*bits*" que se transmiten por segundo. Es importante establecer una velocidad de comunicación adecuada y coincidente tanto en el Arduino como en el software del ordenador para garantizar una transferencia de datos eficiente y confiable. Los baudios se configuran mediante el código en el programa Arduino y en el software de comunicación en el ordenador, asegurando que ambas partes estén sincronizadas y puedan interpretar correctamente los datos transmitidos. Una configuración incorrecta de la velocidad de baudios puede resultar en errores de comunicación o en la recepción de datos incorrectos.

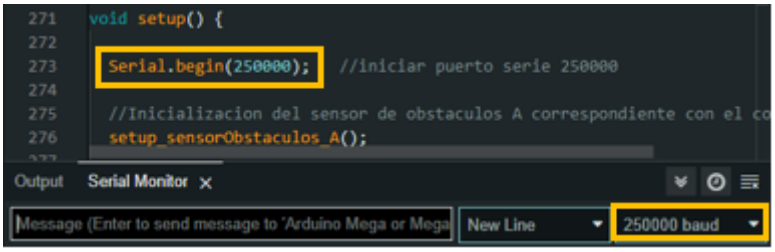


Figura 20. Velocidad de comunicación en el código del proyecto (*imagen de elaboración propia*)

Dicha velocidad de comunicación se puede ajustar dentro de un rango amplio, generalmente entre "300" y "115200 baudios". Por defecto, se suele utilizar una velocidad de "9600 baudios" debido a su compatibilidad generalizada. La velocidad de baudios seleccionada afecta directamente la transmisión de datos entre Arduino, los sensores y los actuadores. Una velocidad más baja, como "300 baudios", implica una transmisión más lenta pero más estable, lo cual puede ser útil en situaciones donde se requiere una alta fiabilidad, pero no una transferencia de datos rápida. Por otro lado, una velocidad más alta, como "115200 baudios", permite una transmisión más rápida de datos, lo que puede ser beneficioso en aplicaciones que requieren una respuesta rápida o una transferencia de datos eficiente. Hace unos años surgían más problemas al emplear velocidades de comunicación más elevadas debido a la compatibilidad con los sensores, pero debido a la tecnología actual se emplean velocidades elevadas (Prometec, Comunicación con el exterior, 2023).

Otro tipo de comunicación se da entre la placa de Arduino y los sensores. Dicha programación se realiza mediante el "*software Arduino IDE*" (IDE, 2023), una plataforma de desarrollo accesible y de código abierto, empleada para escribir y cargar programas en Arduino.

A través de la programación en lenguaje *"C++"* o en el entorno de desarrollo de Arduino, se puede interactuar con los sensores utilizando bibliotecas específicas que facilitan la lectura de datos y el control de los actuadores. El código define cómo se realizan las lecturas de los sensores, cómo se procesan los datos y cómo se realizan las acciones correspondientes según las condiciones especificadas.

Es necesario conocer las funciones empleadas por dicho software a la hora de programar.

Figura 21. Interfaz Arduino IDE 2.0.4 (imagen propia)

La función "*void setup*" es utilizada para realizar configuraciones iniciales y establecer parámetros necesarios antes de que el programa principal comience a ejecutarse. Aquí es donde se definen y configuran los pines de entrada y salida, se establecen las velocidades de comunicación serial, se inicializan variables y se realizan otras tareas de configuración.

Por otro lado, "*void loop*" es la función principal que se ejecuta de manera continua después del "void setup". Aquí es donde se coloca el código que se repite en bucle, como la lectura de sensores, el control de actuadores o cualquier otra lógica de programación que se deba ejecutar de forma repetitiva.

2.3. Alimentación del circuito

Arduino puede ser alimentado de diferentes formas, como a través de un adaptador de corriente o mediante una conexión USB. La elección de la fuente de alimentación dependerá de las necesidades específicas del proyecto y de los componentes conectados.

Un adaptador de corriente para Arduino es un dispositivo utilizado para suministrar la energía eléctrica necesaria para alimentar la placa. El uso de un adaptador de corriente adecuado proporciona una fuente de energía estable y confiable para Arduino, evitando problemas de voltaje insuficiente o inestable que podrían afectar el rendimiento del sistema. Además, proporciona una forma segura de suministrar energía eléctrica sin depender únicamente de baterías u otras fuentes de alimentación temporales.

El Arduino Mega puede aceptar un rango de voltaje de alimentación de “6” a “20 voltios”, siendo estos valores mínimo y máximo. Este rango se recomienda para garantizar un funcionamiento óptimo de la placa. Sin embargo, el Arduino Mega también puede funcionar con voltajes de alimentación más bajos, como “6 voltios”, aunque es posible que algunas funcionalidades se vean afectadas.

En cuanto a los pines de entrada/salida, el Arduino Mega opera con una lógica de “5 voltios”. Esto significa que los pines digitales pueden proporcionar o recibir señales de “5 voltios”. Los pines analógicos también aceptan voltajes en el rango de “0” a “5 voltios”, lo que permite la lectura de sensores analógicos y la generación de señales analógicas mediante la modulación por ancho de pulso (PWM).

En el caso de los motores paso a paso (stepper motors), empleados en este proyecto, suelen requerir una fuente de alimentación externa de mayor voltaje y corriente que la proporcionada por Arduino. Esto se debe a que los motores paso a paso requieren una potencia considerable para funcionar correctamente y realizar los movimientos precisos.

En estos casos que las fuentes de alimentación anteriormente mencionadas son insuficientes, se puede emplear un “*power supply module*” o “*módulo de alimentación*”, es un dispositivo utilizado en proyectos de Arduino para suministrar una fuente de alimentación regulada y controlada. A diferencia de un adaptador de corriente estándar, un módulo de alimentación ofrece una mayor flexibilidad y opciones de configuración en términos de voltaje y corriente de salida.

Figura 22. Módulo de alimentación externa (Bricogeek, Modulo alimentación para protoboard MB-102, 2023)

Este módulo de alimentación generalmente se conecta al Arduino a través de sus pines de entrada de alimentación, como "*VIN*" o "*VCC*", y permite ajustar la tensión de salida según las necesidades del proyecto. Algunos módulos también ofrecen protección contra cortocircuitos, sobrecorriente y sobrecalentamiento, lo que garantiza una alimentación segura para el Arduino y los componentes conectados.

Es importante tener en cuenta los rangos de voltaje al trabajar con el Arduino Mega y al conectar dispositivos externos. Asegurarse de que los dispositivos y sensores utilizados sean compatibles con la lógica de voltaje de "5 voltios" y que los niveles de señal estén dentro de los rangos admitidos para un correcto funcionamiento del sistema.

3. Aspectos relevantes del desarrollo

Una vez conocidas las herramientas empleadas para el desarrollo del proyecto se procede a explicar los aspectos relevantes relacionados con el prototipo. Se explicará la forma de conexión de cada uno de los sensores con la placa Arduino y los conceptos básicos empleados en el código de programación de cada uno de los sensores para posteriormente poder explicar el código final, punto "*4.1*".

Como complemento a este punto se amplía el paso a paso y los problemas enfrentados en el montaje y programación de cada uno de los sensores del prototipo en el "*punto 4.4*" hasta el resultado final.

3.1. Motor paso a paso 28BYJ-48 con driver ULN2003

Para conectar los "*motores paso a paso 28BYJ-48*" con el "*driver ULN2003*" a la placa Arduino, se siguen los siguientes pasos.

En primer lugar, se conectan los cables del motor a los pines correspondientes del "driver ULN2003". Los motores actuales traen una pestaña que permite conectar directamente y sin necesidad de soldaduras los cables de alimentación del motor paso a paso al driver. Dichos pines son los explicados en la "*Figura 10*".

Una vez conectado el motor al controlador, se establece la conexión entre el driver ULN2003 y la placa Arduino. Para ello, se conecta el pin "VCC" (voltaje de corriente continua, VCC a partir de ahora) del driver al pin de alimentación de 5V de Arduino, el pin "GND" (ground o tierra, GND a partir de ahora) del driver al pin "GND" de Arduino, y los pines "IN1", "IN2", "IN3" y "IN4" del driver a pines digitales de salida de Arduino.

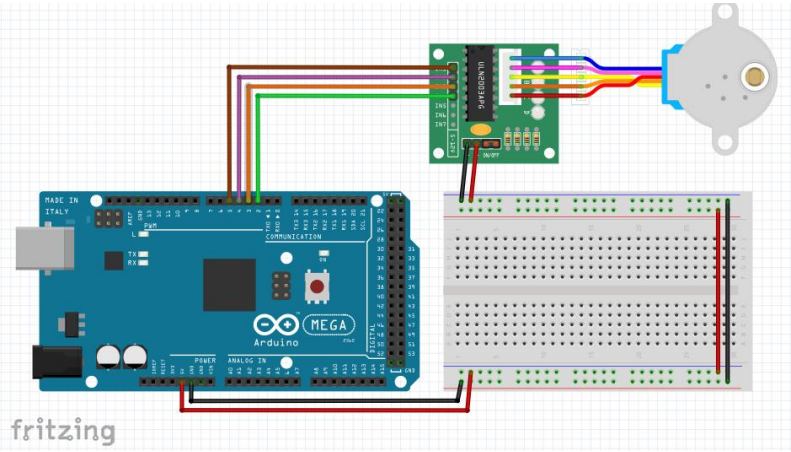


Figura 23. Representación del conexionado del motor paso a paso y driver a la placa Arduino (*imagen de elaboración propia*)

Una vez se hayan establecido las conexiones físicas entre el driver y la placa Arduino, es necesario definir el código adecuado para controlar los motores paso a paso. A continuación, se presenta un ejemplo básico de cómo se puede configurar el código:

- **Incluir la biblioteca "Stepper"**:

Para utilizar las funciones y métodos necesarios para controlar los motores paso a paso, es necesario incluir la biblioteca "`Stepper`" en el Arduino IDE. Esto se logra mediante la instrucción "`#include <Stepper.h>`" al comienzo del código.

```
11  /* Librería motor stepper */
12   #include <Stepper.h>      // incluye libreria stepper
```

Figura 24. Llamamiento a la libreria Stepper (*imagen propia*)

- **Definir las configuraciones iniciales del motor**:

Antes de crear una instancia del objeto "`Stepper`", se deben establecer algunas configuraciones específicas del motor. Esto incluye el número de pasos por revolución del motor y los pines de salida utilizados en el "driver ULN2003" para cada una de las bobinas del motor.

```
73  /* PINES STEP MOTOR CINTA */
74
75   const int Cinta_IN1 = 22;
76   const int Cinta_IN2 = 23;
77   const int Cinta_IN3 = 24;
78   const int Cinta_IN4 = 25;
```

Figura 25. Definición de los pines de salida conectados a Arduino *(imagen propia*)

Los "`const int`" definen los pines de Arduino que se van a conectar a cada uno de los terminales de las bobinas del motor.

Se elige el método de energización de medios pasos, también conocido como "half-step" para todos los motores del proyecto, esta es una técnica utilizada para controlar los motores paso a paso con mayor precisión y menor tamaño de paso. En lugar de energizar las bobinas del motor en su totalidad, como se hace en el método de paso completo, en el método de medios pasos se alternan las secuencias de encendido de las bobinas (Prometec, Prometec: motores paso a paso, 2015).

En este método, cada paso del motor se divide en dos subpasos. En el primer subpaso, se energiza una bobina, y en el segundo subpaso se energizan dos bobinas adyacentes. Esto permite que el motor realice movimientos más suaves y precisos, ya que los pasos son más pequeños y el consumo también es intermedio.

Para dar más claridad a esta explicación se muestra la tabla siguiente, donde "ON" representa que la bobina está siendo energizada y "OFF", su desenergización (Prometec, Prometec: motores paso a paso, 2015).

Tabla 1. Secuencia de energización de bobinas de los motores del proyecto

PASO	BOBINA A	BOBINA B	BOBINA C	BOBINA D
1	**ON**	OFF	OFF	OFF
2	**ON**	**ON**	OFF	OFF
3	OFF	**ON**	OFF	OFF
4	OFF	**ON**	**ON**	OFF
5	OFF	OFF	**ON**	OFF
6	OFF	OFF	**ON**	**ON**
7	OFF	OFF	OFF	**ON**
8	**ON**	OFF	OFF	**ON**

Se definen en el código del proyecto mediante arrays para secuenciar el movimiento.

```
int Paso [ 8 ][ 4 ] =
    {   {1, 0, 0, 0},
        {1, 1, 0, 0},
        {0, 1, 0, 0},
        {0, 1, 1, 0},
        {0, 0, 1, 0},
        {0, 0, 1, 1},
        {0, 0, 0, 1},
        {1, 0, 0, 1}
    };
```

Figura 26. Array de definición de medios pasos (*imagen propia*)

Para la definición de los pasos por revolución se tienen en cuenta los dos tipos de funcionamiento que van a seguir los diferentes motores. En el caso de este proyecto serán necesarios 4 motores, el que se muestra en la "*Figura 25*" corresponde al stepper encargado del movimiento de la cinta, los otros tres se encargarán del movimiento que empujara los bloques de colores hacia sus correspondientes cajas de colores, por ello se han empleado dos tipos de códigos diferentes.

El número total de pasos para una revolución se calcula de la siguiente manera. Cómo se comentó en el punto "*2.1.3*" para conseguir una vuelta completa del eje exterior se requieren 64 vueltas o revoluciones del rotor, un giro completo del rotor requiere de 8 ciclos y cada ciclo requiere de 4 pasos.

Por tanto, para el motor correspondiente al movimiento de la cinta se define una constante (empleando "`int`") que determina el total de pasos que debe recorrer siendo en este caso de 4095. En el caso de los motores que empujan las cajas a clasificar se emplea la mitad de los ciclos, relacionado a lo explicado en el anterior párrafo, para que este grupo de motores únicamente recorra media vuelta en un sentido y media vuelta en el sentido contrario, por ello en este caso la constante de pasos restantes.

```
int steps_left_A = 2048;
```

Figura 27. Constante de pasos restantes correspondiente a uno de los motores de empuje de bloques (*imagen propia*)

En el código proporcionado, la función "`void setup ()`" se encarga de configurar los pines de salida de la placa Arduino. Cada línea de código que comienza con "`pinMode ()`" establece el modo de funcionamiento de un pin específico.

En este caso, se están configurando cuatro pines, denominados "IN1", "IN2", "IN3" y "IN4", como salidas utilizando la función "`pinMode ()`" con el argumento "OUTPUT". Esto significa que estos pines se utilizarán para enviar señales de control al driver o módulo conectado a los motores paso a paso.

```
/* FUNCION STEP MOTOR CINTA */

void setup_stepCinta(){

  //Inicialización del motor step de la cinta
  pinMode(Cinta_IN1, OUTPUT);
  pinMode(Cinta_IN2, OUTPUT);
  pinMode(Cinta_IN3, OUTPUT);
  pinMode(Cinta_IN4, OUTPUT);

  // Establece la dirección como verdadera (solo hacia adelante)
  Direction = true;

}
```

Figura 28. Configuración del funcionamiento de cada uno de los pines del motor cinta (*imagen propia*)

Como se muestra en la "*Figura 28*", dicha definición de los pines se encuentra dentro de una función "`void setup_stepCinta ()`", diferente al "`void setup ()`" mencionado, se debe a que la función de la figura se llamará posteriormente dentro de "`void setup ()`" de esta forma se logra un código más claro y ordenado. Esta configuración se repite para el resto de los motores del código.

- **Definir código bucle del funcionamiento del motor:**

Dentro del "`void loop ()`" se determina dos tipos de funcionamiento como se mencionó en el punto anterior.

La función "`void movimientoCinta ()`", "`void step_A_empuje ()`", "`void step_B_empuje ()`" y "`void step_C_empuje ()`" avanzan un medio paso cada vez que se las invoca debido a la matriz de excitación y controla el punto de la secuencia en el que se encuentra el programa hasta completar el número de pasos definidos. Para ello se emplea "`steps_left`" como se explicó en la "*Figura 27*" para que mientras queden pasos pendientes siga ejecutándose el bucle.

En el caso del motor de la cinta, siempre realizará el movimiento en un único sentido, en el caso de los motores que se encargan de empujar los bloques se pretende realizar media vuelta en un sentido y otra media en el contrario, para ello se añade la variable "`bool direction_A`", "`bool direction_B`" y "`bool direction_C`" la cual invierte el sentido de giro del motor una vez completados los pasos definidos.

```
Direction_B = !Direction_B;
steps_left_B = 2048;
```

Figura 29. Variables definidas dentro de la función "`void step_B_empuje ()`" que indica el cambio de sentido en la dirección de giro hasta completar los pasos definidos (*imagen propia*)

Otra función importante dentro de la programación de los motores es la de desenergización de las bobinas cada vez que termina su funcionamiento. Para ello se cambia el estado de los pines a "LOW" con el fin de que no le llegue más energía.

```
void paradaCinta(){

  digitalWrite(Cinta_IN1, LOW); // Apaga todos los pines
  digitalWrite(Cinta_IN2, LOW);
  digitalWrite(Cinta_IN3, LOW);
  digitalWrite(Cinta_IN4, LOW);

}
```

Figura 30. Desenergizar bobinas del motor cinta *(imagen propia)*

Esto se aplica a cada uno de los motores del código cuando finalizan su movimiento.

3.2. OLED 1.3’ SSH1106

Para el conexionado de la pantalla OLED se emplea como se mencionó en el punto “*2.1.4*” el protocolo de conexionado “SPI” (330ohms, 2019). Para ello se conectan cada uno de los pines a las salidas digitales de Arduino, el pin “GND” a tierra y el pin “VCC” al pin de alimentación de 5V de la placa Arduino.

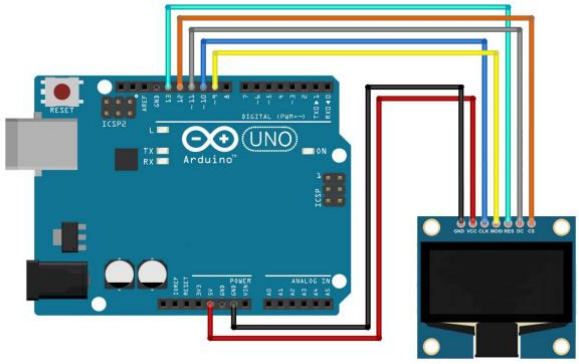


Figura 31. Esquema de conexionado de la pantalla OLED mediante SPI a la placa Arduino (electronica.uy, 2018)

Siguiendo el esquema del anterior actuador se definen los siguientes puntos necesarios para su programación (José, DescubreArduino, 2019).

- **Incluir librerías**:

Las siguientes librerías son comúnmente utilizadas en proyectos de Arduino para interactuar con dispositivos y periféricos específicos, en concreto la pantalla OLED empleada en este caso:

“**`SPI.h`**”: Esta librería proporciona funciones y métodos para la comunicación utilizando el protocolo SPI (Serial Peripheral Interface). Permite la transferencia de datos de manera síncrona entre el Arduino y dispositivos externos, como sensores, pantallas, tarjetas de memoria, entre otros.

“**`Adafruit_GFX.h`**”: Esta librería proporciona una serie de funciones y clases para la creación de gráficos en dispositivos como pantallas LCD y OLED. Permite dibujar formas geométricas básicas, texto, iconos y realizar operaciones de manipulación de píxeles.

“**`Adafruit_SH110X.h`**”: Esta librería es específica para el control de pantallas OLED basadas en el “controlador SSH110X”. Proporciona funciones y métodos para inicializar la pantalla, enviar comandos y datos, así como dibujar gráficos y texto en la pantalla OLED.

- **Definir las configuraciones iniciales de la pantalla**:

En este punto se incluye la configuración de los pines correspondientes para inicializar la comunicación SPI.

```
/* PINES OLED */

const int OLED_CLK  = 13;
const int OLED_MOSI = 12;
const int OLED_RST  = 11;
const int OLED_DC   = 10;
const int OLED_CS   = 9;
```

Figura 32. Asignación de los pines de la OLED a los pines de Arduino (*imagen propia*)

En la siguiente línea de código se crea un objeto de la clase "*Adafruit_SH1106G*" llamado "`display`", este define un conjunto de propiedades y métodos que describen las características y comportamientos de un tipo de objeto en particular. El objeto se inicializa con los parámetros necesarios para configurar y comunicarse con la pantalla "*OLED 1.3' SSH1106*". Los parámetros que se pasan son la definición de las dimensiones de la pantalla OLED y llama a los pines de Arduino empleados en cada uno de los pines de la pantalla para poder comunicarse mediante "*SPI*".

```
//Se define la pantalla OLED

Adafruit_SH1106G display = Adafruit_SH1106G(128, 64,OLED_MOSI, OLED_CLK, OLED_DC, OLED_RST, OLED_CS);
```

Figura 33. Definición de la pantalla OLED para la comunicación SPI con Arduino (*imagen propia*)

Para definir las configuraciones iniciales de la pantalla OLED se define una función con todo lo necesario a declarar en el "`void setup ()`" del código principal.

En esto se incluye la primera línea que inicia la comunicación con la pantalla OLED. El parámetro "0" indica que no se está utilizando una dirección I2C específica para la comunicación, y true indica que se realizará un reinicio de la pantalla (`display.begin(0, true);`).

A continuación, se define una línea que esta línea muestra en la pantalla lo que se ha dibujado o escrito hasta el momento. Es útil para asegurarse de que la pantalla está actualizada correctamente (`display.display();`). Como complemento a esto una línea que borra todo lo que se haya dibujado en la pantalla. Es una forma de asegurarse de que la pantalla esté en blanco antes de comenzar a mostrar nuevos datos (`display.clearDisplay();`).

- Definir código bucle del funcionamiento de la pantalla OLED:

Dado que la pantalla va a mostrar un contador de los objetos de cada color que caen a las cajas clasificadoras se necesitarán dos funciones para lograr el acometido. Ambas se llamarán posteriormente desde la función "`void loop ()`" del código principal.

La primera de las funciones creadas "`void encenderPantalla ()`", consiste en mostrar de forma continua en la pantalla los datos que no variarían a lo largo del funcionamiento de cada bucle, es decir, el texto de los colores y su contador, el cual en esta primera función se guardará el último dato leído. Llamando a la posterior función se actualiza el dato del contador "`void actualizarPantalla ()`", un ejemplo de su funcionamiento se muestra en la "*Figura 34*".

Figura 34. Texto mostrado por la pantalla tras ejecutarse el bucle de las funciones nombradas (*imagen propia*)

3.3. Sensor de colores TCS3200

Para conectar el sensor TCS3200 a Arduino, se necesitan cuatro pines: “S0”, “S1”, “S2” y “S3”, que se utilizan para configurar la frecuencia de muestreo del sensor y el modo de salida, estos se conectan a los pines digitales de la placa Arduino. Además, se requieren dos pines más para la entrada y salida de datos: “OUT” y “OE”, ambos conectados a tierra de la placa.

El sensor TCS3200 también necesita una fuente de luz adecuada para iluminar el objeto que se va a analizar. Esto se puede lograr utilizando LEDs de colores o una fuente de luz blanca. En este caso se emplean los leds incluidos en el sensor, por lo que el pin “LED” se conectará a 5V de la placa.

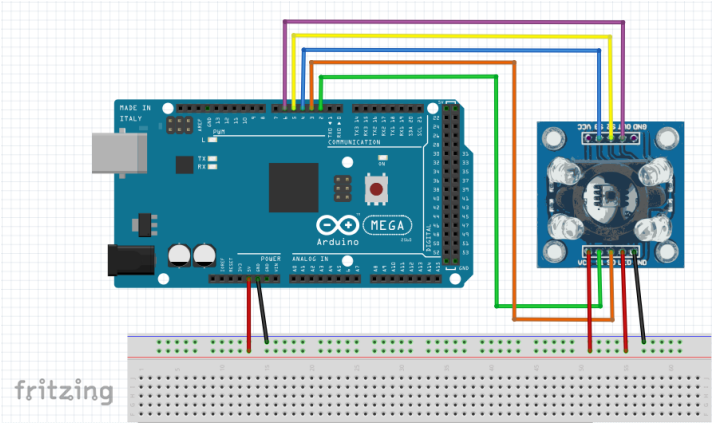


Figura 35. Conexionado del sensor de color a la placa Arduino (*imagen propia*)

La programación del sensor se realiza siguiendo las instrucciones del siguiente sitio web (Bitwiser, Arduino desde cero en Español, 2020), una vez realizadas las conexiones el código se compone de las siguientes partes.

- **Incluir librerías**: Para este sensor en concreto y debido a su programación no ha sido necesario incluir ninguna librería.
- **Definir configuraciones iniciales del sensor de colores**:

Una vez que el sensor está conectado físicamente a Arduino, se puede proceder con la programación. Primero, es necesario configurar los pines correspondientes como salidas utilizando las funciones “`pinMode ()`”. Luego, se deben establecer las frecuencias de muestreo y los modos de salida adecuados mediante la configuración de los pines S0 y S1. Todo esto se recoge en la función creada para albergar todos estos datos llamada “`void setup_sensorColores ()`”, la cual es llamada en “`void setup ()`” del código principal como se mencionó anteriormente.

- **Calibración previa al funcionamiento en la cinta**:

Mediante este primer programa de calibración (Bitwiser, Arduino desde cero en Español, 2020) contiene la programación de Arduino para obtener los valores de los colores y calibrar el sensor de color.

En primer lugar, se configuran los pines "S2" y "S3" del sensor para establecer los fotodiodos con los respectivos filtros de color. En este caso, se comienza con el filtro rojo.

```
void loop() {
  digitalWrite(S2,LOW);     // establece fotodiodos
  digitalWrite(S3,LOW);     // con filtro rojo
  int rojo = pulseIn(salidaTCS, LOW); // obtiene duracion de pulso de salida del sensor
  delay(200);       // demora de 200 mseg
```

Figura 36. Ejemplo de calibración y establecer fotodiodos del color rojo

A continuación, se utiliza la función "`pulseIn()`" para medir la duración del pulso de salida del sensor cuando se aplica el filtro rojo. Este valor se guarda en la variable "`rojo`". Luego, se realiza una demora de 200 milisegundos antes de continuar.

Después, se repite el proceso para el filtro verde y el filtro azul, obteniendo los valores de duración de pulso y almacenándolos en las variables "`verde`" y "`azul`", respectivamente.

A continuación, se utilizan las funciones "`Serial.print ()`" y "`Serial.println ()`" para mostrar los valores de los colores en el Monitor Serial. Se muestra el valor de 'rojo', se agrega un espacio de tabulación ("`\t`"), se muestra el valor de "`verde`", se agrega otro espacio de tabulación y finalmente se muestra el valor de 'azul' junto con un salto de línea.

Este ciclo se repetirá continuamente en la función "`void loop ()`", proporcionando información en tiempo real sobre los valores de los colores detectados por el sensor. Esta información puede ser utilizada para calibrar el sensor y ajustar los umbrales de detección según las necesidades del proyecto.

Tabla 2. Composición RGB de cada uno de los colores a detectar en la cinta

	R	V	A
ROJO	28	44	33
VERDE	37	39	33
AZUL	36	38	23

En la anterior "*Tabla 2*" se muestra que nivel de rojo, verde y azul (fila superior de la tabla representado mediante "R", "V" y "A") que contienen los colores que la cinta va a clasificar, en este caso "ROJO", "VERDE" y "AZUL", representado en la columna izquierda.

Dichos valores se introducirán en el siguiente código, el cual forma parte del código principal de la cinta transportadora.

- **Definir código bucle del sensor de colores**:

Una vez calibrado el valor de los colores se emplean dos funciones en el código principal para cumplir con los requerimientos de clasificación de colores en el proyecto.

La función "`int obtenerValorConFiltro`" recibe dos argumentos de tipo entero: "s2" y "s3", que corresponden a los pines utilizados para configurar los filtros del sensor de color.

Dentro de la función se configuran los filtros del sensor de color según los parámetros recibidos, realiza una lectura de color y devuelve el valor obtenido en forma de entero.

```
/* FUNCIÓN SENSOR COLORES 1: Encargada de obtener el valor del sensor con un filtro específico*/

int obtenerValorConFiltro(int s2, int s3) {

  digitalWrite(s2, LOW);
  digitalWrite(s3, LOW);
  int valor = pulseIn(salidaTCS, LOW);
  delay(100);

  return valor;

}
```

Figura 37. Función encargada de obtener el valor del color detectado mediante el sensor de colores (*imagen propia*)

Esta función, "*Figura 37*", se emplea dentro de la siguiente función "`int identificarColor ()`" empleada para identificar el color específico basado en los valores obtenidos mediante la función "`int obtenerValorConFiltro ()`".

Luego, se definen variables booleanas para cada posible color. Estas variables, como "`colorRojo`", "`colorVerde`" y "`colorAzul`", se asignarán en función de las comparaciones realizadas utilizando los valores de rojo, verde y azul obtenidos mediante el programa de calibración. Por último, se utilizan variables de retorno para indicar que color se ha identificado y escribirlo por pantalla.

Figura 38. Se muestra por pantalla el color identificado mediante la función "`identificarColor ()`" (*imagen propia*)

3.4. Detector de obstáculos con sensor infrarrojo

El conexionado del sensor de obstáculos con sensor infrarrojo con la placa de Arduino se realiza de la siguiente manera: el pin "VCC" del sensor se conecta a una salida de voltaje de 5V de Arduino, el pin "GND" se conecta a un pin de tierra ("GND") de Arduino, y finalmente, el pin de salida del sensor se conecta a un pin digital de entrada de Arduino.

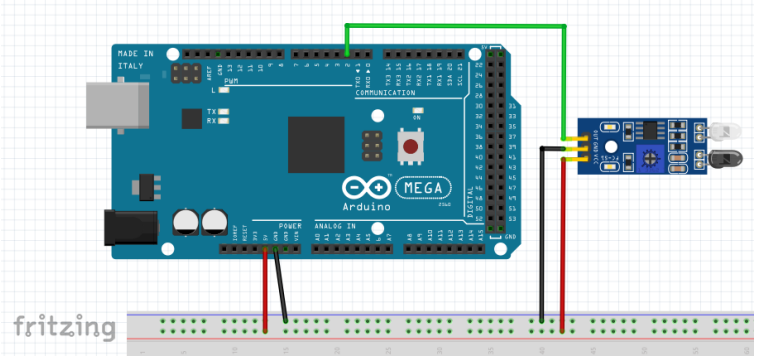


Figura 39. Esquema de conexionado del sensor infrarrojo de obstáculos *(imagen propia)*

Una vez que se ha realizado el conexionado, se puede programar Arduino para utilizar el sensor infrarrojo de obstáculos.

- **Incluir librerías**:

Para este sensor no han sido necesarias incluir ninguna librería para su programación-

- **Definir configuraciones iniciales del sensor de obstáculos**:

En primer lugar y como ya se realizó con sensores anteriores se definen los pines de Arduino que se utilizarán para conectar el sensor, utilizando la función "`pinMode ()`" y estableciendo los pines como entradas dentro de la función "`void setup_sensorObstaculos ()`" la cual se llamará posteriormente desde "`void setup ()`" del código principal.

```
/* INICIALIZAR SENSOR OBSTÁCULOS A */

void setup_sensorObstaculos_A(){
  pinMode(sensorObstaculo_A , INPUT);  //definir pin como entrada
}
```

Figura 40. Definición de los pines del sensor de obstáculos A asociado al funcionamiento de los bloques rojos *(imagen propia)*

Serán necesarias tantas funciones igual a la mostrada en la "Figura 40", como sensores se utilicen en el prototipo, en el caso de este proyecto serán tres sensores diferenciados mediante las letras "A", "B" y "C" encargados de detectar los bloques rojo, verde y azul correspondientemente.

- **Definición de la condición empleada en el bucle del código principal**:

En este caso se trabaja directamente dentro del bucle "`void loop ()`" debido a la simplicidad de funcionamiento requerido para este sensor.

Se emplea la condición "`digitalRead(sensorObstaculo_C) == LOW`", la cual se utiliza para verificar el estado del pin de entrada del sensor de obstáculos conectado a Arduino.

La función "`digitalRead ()`" lee el estado del pin y devuelve el valor leído, que puede ser "HIGH" o "LOW". En este caso, se compara si el valor leído del pin del sensor de obstáculos es igual a "LOW".

Cuando el sensor de obstáculos detecta un obstáculo, el pin de entrada se pone en un estado "LOW", lo que significa que hay una interrupción en el haz infrarrojo emitido por el sensor. Por lo tanto, si la condición "`digitalRead(sensorObstaculo_C) == LOW`" se cumple, significa que se ha detectado un obstáculo.

Esta condición se empleará entre otras para regular el funcionamiento de la OLED y los diferentes motores de la cinta. El funcionamiento conjunto de todas estas especificaciones de programación se detalla en el punto "*4.1*".

```
} else if (tiempoTranscurrido > 7000 && digitalRead(sensorObstaculo_C) == LOW && colorLeido == COLOR_AZUL )
```

Figura 41. Se incluye la condición del sensor de obstáculos dentro del código principal (*imagen propia*)

4. Funcionalidad e implementación del prototipo

En este punto se explicará el código final, la maqueta y el presupuesto del proyecto. Se analizará en conjunto cómo se ha desarrollado el código para garantizar el correcto funcionamiento del sistema. Además, se describirá la maqueta física utilizada para probar y visualizar el proyecto, explicando los componentes empleados y su disposición. Por último, presentaremos un desglose del presupuesto necesario, incluyendo los costos de los materiales y componentes del proyecto.

4.1. Descripción del código final

El código comienza presentando una serie de librerías y definiciones de pines utilizados en el proyecto. La función de las librerías viene detallada en el punto "*3*".

- **Librería pantalla OLED**: Esta librería permite el control de una pantalla OLED utilizando el protocolo SPI y la comunicación I2C. Se utiliza para mostrar información en la pantalla OLED conectada al sistema.
- **Librería "motor stepper"**: Esta librería permite controlar un motor paso a paso. Se utiliza para controlar los motores que accionan la cinta transportadora y los motores de empuje de los bloques.

A continuación, se definen los pines utilizados por los diferentes componentes:

- **Pines OLED**: Se definen los pines utilizados para la comunicación con la pantalla OLED, incluyendo los pines de reloj, datos, reinicio y control.
- **Pines sensor de obstáculos**: Se definen los pines utilizados para la lectura del sensor de obstáculos A, B y C.
- **Pines step motor cinta**: Se definen los pines utilizados para controlar el motor paso a paso de la cinta transportadora.
- **Pines sensor de colores**: Se definen los pines utilizados para la comunicación con el sensor de colores.
- **Pines step motor A, B y C**: Se definen los pines utilizados para controlar los motores paso a paso de los bloques rojos, verdes y azules, respectivamente.

A continuación, se declaran una serie de funciones utilizadas en el código:

- **Funciones OLED**: Se definen funciones relacionadas con la inicialización y actualización de la pantalla OLED.
- **Funciones sensor de obstáculos**: Se definen funciones para la inicialización de los sensores de obstáculos A, B y C.
- **Funciones step cinta**: Se definen funciones relacionadas con el control del motor paso a paso de la cinta transportadora.
- **Funciones step A, B y C**: Se definen funciones relacionadas con el control de los motores paso a paso de los bloques rojos, verdes y azules, respectivamente.
- **Funciones sensor de colores**: Se definen funciones relacionadas con la inicialización y lectura del sensor de colores.

Otras variables que se definen necesarias para el control de la cinta son:

- **Contadores obstáculos rojo, verde y azul**: Se definen variables y contadores relacionados con la detección y conteo de obstáculos de cada color.
- **Contador tiempo**: Se definen variables y funciones relacionadas con el tiempo transcurrido desde el inicio del movimiento de la cinta transportadora.

Dentro de la función "`void setup ()`", se realizan las configuraciones iniciales del sistema antes de que el programa entre en el bucle principal "`void loop ()`" como ya se ha mencionado en el punto "*2.2*". En el código, se llevan a cabo las siguientes acciones:

Se establece la velocidad de comunicación en el puerto serie utilizando la función "`Serial.begin ()`". En este caso, se configura la velocidad a "25000 baudios", para conocer el fundamento de esta elevada velocidad se puede consultar el punto "*2.2*", con fin de conocer cómo se llegó al empleo de dicha velocidad se puede consultar el punto "*4.4*".

Se configuran los pines del Arduino que se utilizarán para controlar la cinta transportadora. Se llama a todas las funciones que inicializan cada uno de los sensores y actuadores para activar su comunicación y funcionamiento.

Dentro del bucle "`void loop ()`" se encuentran varias condiciones que controlan el funcionamiento del sistema.

En primer lugar, se verifica si la pantalla OLED está apagada. Si es así, se enciende llamando a la función "`encenderPantalla ()`" y se establece la variable "`pantallaEncendida`" como true, indicando que efectivamente se encuentra encendida.

A continuación, se calcula el tiempo transcurrido desde el inicio del movimiento de la cinta transportadora utilizando la función "`millis ()`". Este valor se almacena en la variable "`tiempoTranscurrido`". Luego, se imprime el tiempo transcurrido por el puerto serie para propósitos de depuración.

Posteriormente, se verifica si los contadores de objetos rojos, verdes y azules son menores que el número máximo de objetos definido. Esta comprobación asegura que no se hayan detectado el número máximo de objetos de cada color.

Luego, se verifica si el sensor de colores está activo. En caso afirmativo, se llama a la función "`identificarColor ()`" para determinar el color actual detectado por el sensor. El valor del color se almacena en la variable "`colorLeido`".

Si se ha detectado un color válido (diferente de 0), se desactiva el sensor de colores para evitar que se siga leyendo mientras el motor de la cinta transportadora está en movimiento, con esto se evita que todos los sensores estén recibiendo alimentación al mismo tiempo y se consiga una mayor efectividad en el funcionamiento de cada uno de los sensores.

Si el sensor de colores está desactivado, es decir, ha detectado alguno de los colores programados" se procede a mover la cinta transportadora mediante la función "`movimientoCinta ()`".

A continuación, se verifica si se han cumplido todas las condiciones programadas para cada color, entre las que se encuentran, si el tiempo transcurrido es el definido, si el sensor de obstáculos detecta un obstáculo y si el color leído coincide con el color programado. En caso afirmativo, se llama a la función "`paradaCinta ()`" para detener el movimiento de la cinta, ejecutar el movimiento de cada uno de los motores encargados de empujar el bloque correspondiente y realizar, por último, una nueva lectura del sensor de colores.

Estas condiciones se evalúan repetidamente en el bucle "`void loop ()`", permitiendo que el sistema funcione de manera continua y realice las acciones correspondientes según las condiciones específicas.

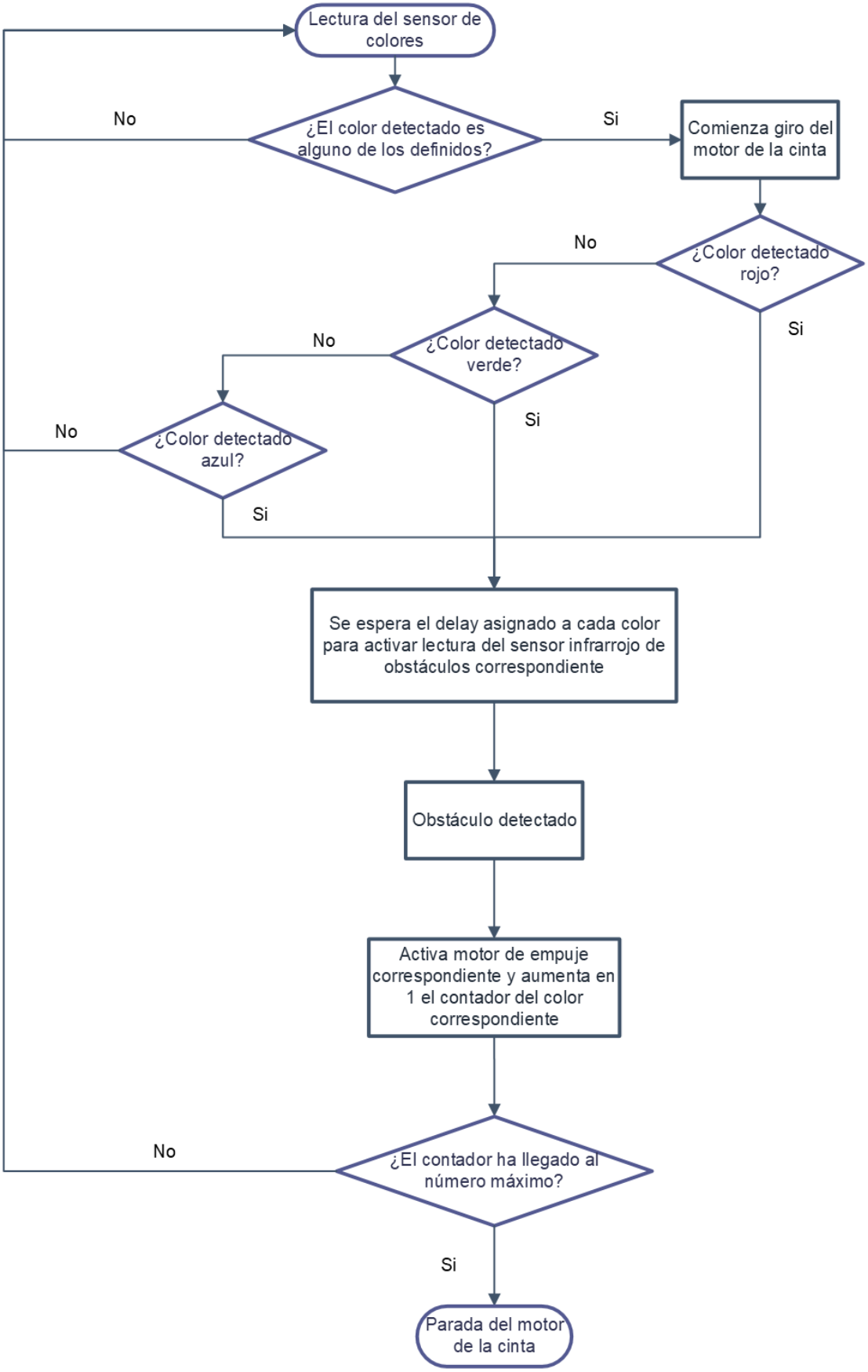


Diagrama 1. Diagrama de flujo del funcionamiento de la cinta transportadora clasificadora

4.2. Descripción de la maqueta del prototipo

Para darle sentido físico al proyecto de la cinta transportadora y colocar todos los sensores de Arduino de manera adecuada, se llevará a cabo la construcción de una maqueta utilizando diversos materiales. Uno de los materiales principales que se utilizará es poliestireno en forma de placa transparente, un tipo de plástico transparente y resistente.

La placa de poliestireno se emplea para crear la estructura base de la maqueta, proporcionando estabilidad y permitiendo la visualización de los componentes internos del sistema. Se cortan láminas de metacrilato en diferentes formas y tamaños para construir los niveles de la cinta transportadora y las plataformas donde se ubicarán los sensores.

Figura 42. Disposición final de la estructura de poliestireno

El primer paso para llegar a la estructura que se muestra en la "*Figura 42*" ha sido la realización de unos planos iniciales mostrados en el "*ANEXO 2. PLANOS*", en los que se reflejan la idea inicial para luego dibujarlos sobre la placa de poliestireno.

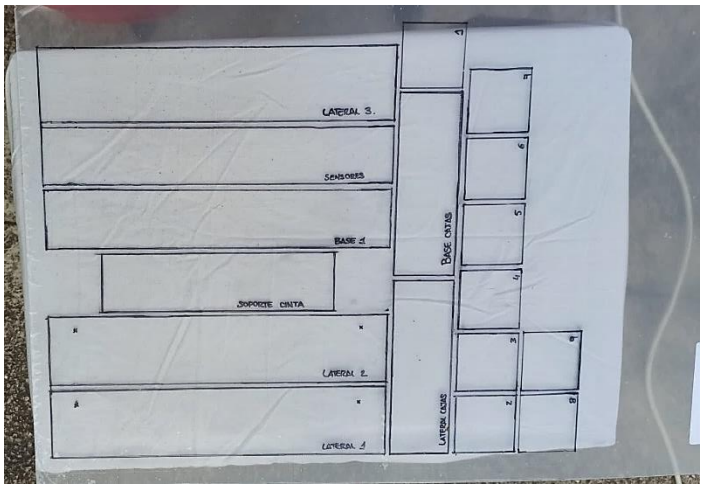


Figura 43. Dibujo de las diferentes piezas sobre la placa de poliestireno

Para cortar el polivinilo se emplea una sierra de calar y una lija para limar los bordes irregulares.

Figura 44. Proceso de cortar las figuras con una sierra de calar

Como unión de las diferentes piezas y para facilitar su pegado se utilizaron perfiles en “L” de PVC y un pegamento adecuado para estos materiales.

Figura 45. Perfil en "L" de PVC empleado para unir las diferentes piezas de polivinilo

Para recrear la propia cinta transportadora, se emplea como material final goma EVA (donde EVA son las siglas de Etileno Vinil Acetato), ya que proporciona una superficie antideslizante la cual transmite de forma adecuada el movimiento de los rodillos que impulsan la cinta. Para unir ambos extremos se cose la cinta.

Figura 47. Cinta de goma EVA en su posición final en la maqueta

Se emplean diferentes engranajes para lograr los diferentes movimientos giratorios de la cinta transportadora. El engranaje encargado de transmitir el movimiento a la cinta transportadora se ubica en uno de los extremos de uno de los rodillos, cuyo engranaje complementario se colocó en el motor asignado al giro de la cinta. De esta manera, al activar el motor, los engranajes se acoplarán y transmitirán el movimiento a la cinta transportadora, haciendo que se desplace de forma continua.

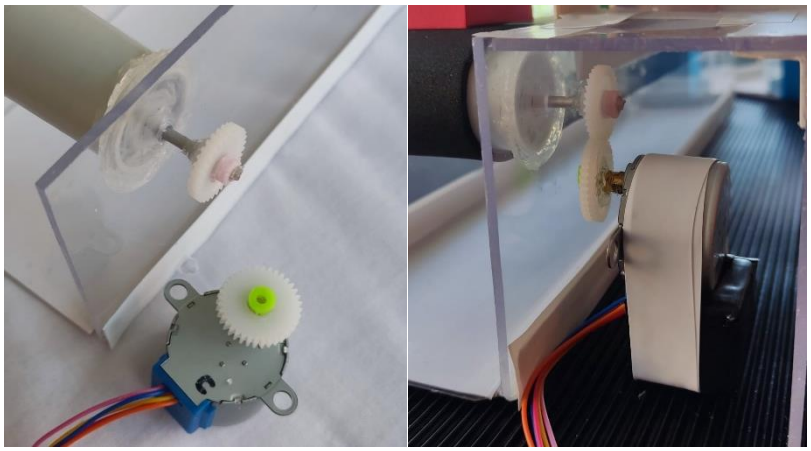


Figura 46. Montaje de los engranajes encargados del movimiento de la cinta transportadora

El resto de los engranajes se encuentran en la plataforma destinada a colocar los motores que empujan las cajas de colores y los detectores de obstáculos. Este juego de engranajes consiste en tres mecanismos de cremallera, estos son una combinación de una cremallera dentada (barra dentada con una serie de dientes alineados que se extienden a lo largo de su longitud) y un piñón que permite la conversión del movimiento rotativo al movimiento lineal.

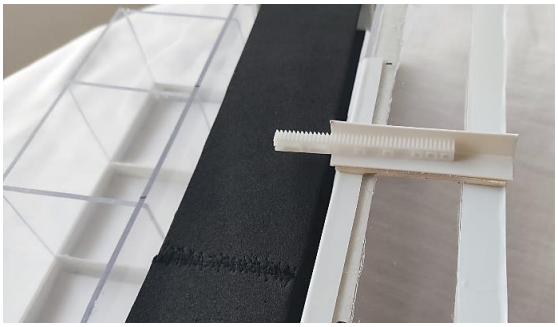


Figura 48. Disposición del engranaje cremallera el cual se moverá sobre un perfil de PVC que actúa de guía

Adicionalmente, se utilizan otros materiales de maquetación reciclados, como alambres, tubos de plástico y pegamento, para asegurar los componentes, mantener la estructura estable y facilitar el ensamblaje de la maqueta.

Figura 49. Cilindros conductores del movimiento de la cinta transportadora

La construcción de esta maqueta permite tener una representación física aproximada del sistema de cinta transportadora controlado por Arduino. Esto no solo brindará una mejor comprensión del proyecto, sino que también facilitará la visualización y el testeo de los sensores en un entorno físico simulado.

A continuación, en la "*Figura 50*" se señalan algunos de los componentes para aportar otra vista de la maqueta mostrada en la "*Figura 51*". Para más detalles consultar "*ANEXO 2. PLANOS*".

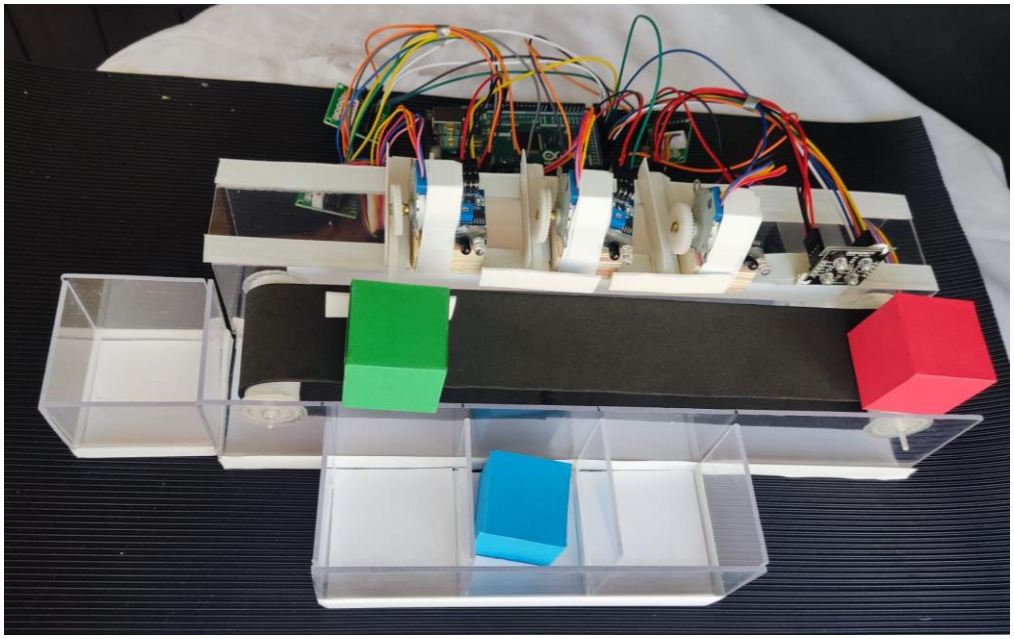


Figura 51. Montaje final

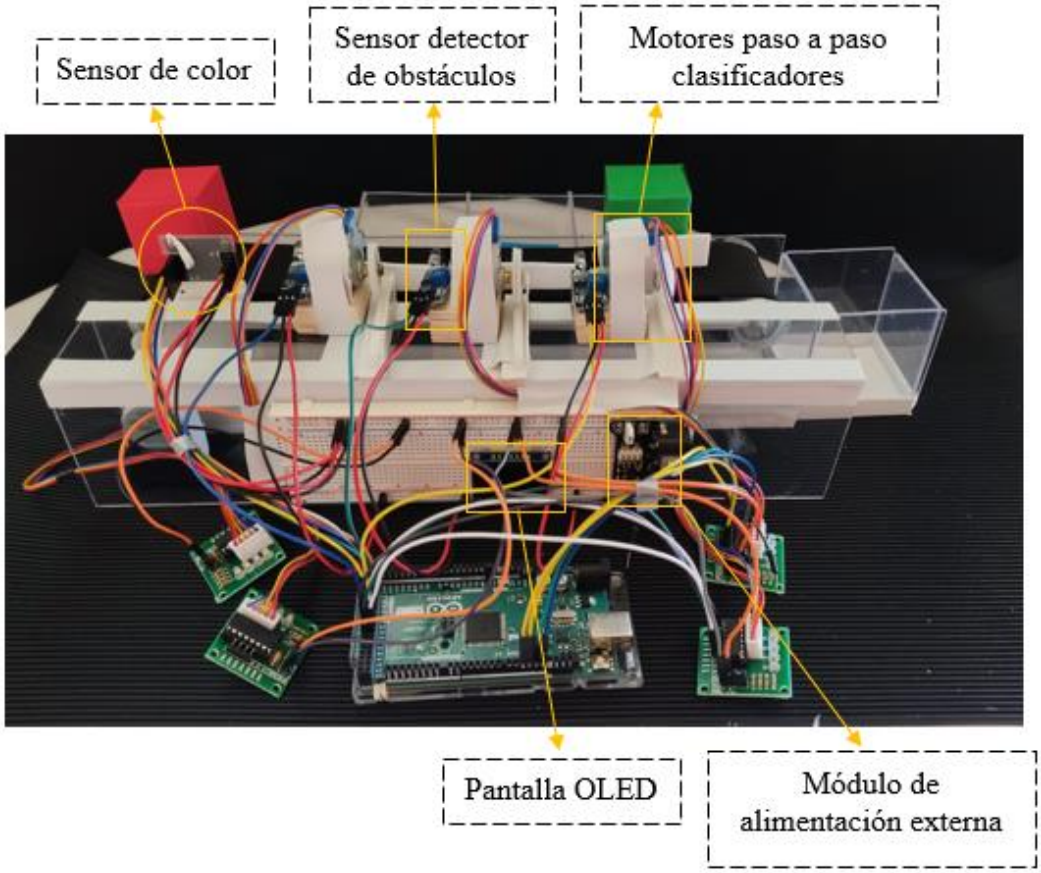


Figura 50. Detalles de la maqueta

4.3. Funcionamiento del sistema

Como complemento del "*Diagrama 1*" se redacta el funcionamiento de la cinta para aportar mayor claridad.

La cinta transportadora funciona de la siguiente manera: inicialmente, colocamos un bloque delante del sensor de colores. Este sensor realizará lecturas continuas hasta que el patrón de colores coincida con alguno de los colores programados. Cuando esto sucede, se activa el funcionamiento del motor paso a paso, encargado de mover la cinta transportadora.

Para optimizar la efectividad de los sensores, se han establecido tiempos de espera. Después de detectar un color específico, se espera un número programado de segundos antes de activar la lectura del primer sensor de obstáculos si el color detectado es rojo. De manera similar, se esperan otros segundos para activar la lectura del segundo sensor de colores si el color detectado es verde, y otros segundos para activar la lectura del tercer sensor de colores si el color detectado es azul.

Cuando el sensor de obstáculos asignado al color correspondiente detecta un obstáculo, el contador en la pantalla OLED asociada aumenta en 1. Si alguno de los contadores alcanza un número determinado de objetos, el motor de la cinta se detiene y no se realizan más lecturas del sensor de colores. En caso de que no se alcance el número determinado, se continúan realizando lecturas del sensor para seguir el proceso de detección.

4.4. Resolución de problemas surgidos en el código

Durante el proceso de programación del código para el funcionamiento de la cinta clasificadora, se presentan varios desafíos que afectan al correcto funcionamiento y eficiencia del sistema. A continuación, se explican los problemas más significativos y las medidas que se aplican para intentar solucionarlos.

En un principio, se escribe todo el código en una secuencia continua, lo cual dificulta la identificación de errores y no permite una visión clara del problema relacionado con el funcionamiento simultáneo de los sensores.

Para abordar este problema y lograr un funcionamiento simultáneo de los componentes, se investiga sobre cómo realizar procesos en paralelo con Arduino, descubriendo que esto no era posible. Esto se debe a que Arduino trabaja secuencialmente, ejecutando una instrucción tras otra. Sin embargo, se encuentra una posible alternativa: trabajar con tiempos de funcionamiento. Esto implica que mientras un sensor esté en funcionamiento, los demás sensores no recibieran alimentación. De esta manera, se puede coordinar correctamente la alimentación de los sensores y lograr un funcionamiento eficiente (José, Cómo hacer multitarea con Arduino, 2023).

Además, para mejorar la claridad del código, se implementan funciones que recogen las partes básicas de programación de cada componente. Esto permite llamar fácilmente a estas funciones desde el código principal y crear condiciones más simples y claras en el bucle principal.

```
/* FUNCIÓN SENSOR COLORES 1: Encargada de obtener el valor del sensor con un filtro especifico*/

int obtenerValorConFiltro(int s2, int s3) {

  digitalWrite(s2, LOW);
  digitalWrite(s3, LOW);
  int valor = pulseIn(salidaTCS, LOW);
  delay(100);

  return valor;

}
```

Figura 52. Ejemplo de función encargada de obtener el valor de los colores con el sensor de colores (*imagen propia*)

Otro problema residía en el rendimiento de los motores, los cuales funcionaban lentamente y sin la suficiente fuerza, a pesar de haber probado todos los métodos de energización disponibles como se explica en el punto "*3.1*". Investigando otros recursos, se descubre un código de control de motores que utiliza una velocidad de transmisión de datos de "25000 baudios" en lugar de los "9600" habituales.

Esta mayor velocidad de transferencia de datos permitió un funcionamiento más fluido y mejorado de los motores (Prometec, Prometec: motores paso a paso, 2015).

Posteriormente, al conectar múltiples motores al sistema, se comprueba que, a pesar de que los "*LEDs*" que indican que las bobinas se encuentran energizadas estaban encendidos (como se puede observar en la "*Figura 53*"), los motores funcionaban de manera lenta o no giraban en absoluto. En este punto, se introduce una alimentación adicional a través del conector Jack para asegurar una alimentación adecuada a los motores (se explica el funcionamiento de este conector en el punto "*2.1.1*").

Figura 53. Ejemplo de LEDs encendidos del driver del motor indicando la energización de las bobinas (Bitwiser, Paso a Paso (unipolar) 28BYJ-48 con Driver ULN2003, 2017)

Al combinar los motores, el sensor de color y los sensores de obstáculos, se comprueba que, al ejecutar el motor de la cinta junto con los sensores simultáneamente, los sensores se apagaban progresivamente. Los LEDs del sensor de color se volvían más tenues y los sensores de obstáculos perdían precisión, llegando incluso a detectar obstáculos continuamente. Para solucionar este problema, se aplican tiempos de funcionamiento. Es decir, los sensores no reciben alimentación hasta que el sensor de color detecte uno de los colores definidos. Luego, se alimentaba el motor paso a paso de la cinta y, pasados unos segundos, se permite la lectura de los sensores de obstáculos. Esto aseguraba que los sensores estén alimentados en el momento adecuado y así evitar interferencias.

Además, se incluye una función para cada motor que desenergiza las bobinas cuando dejan de ser necesarios. Esto se debió a que, incluso cuando los motores no estaban en funcionamiento, los LEDs indicadores permanecían encendidos, lo que indicaba que seguían energizados y por tanto impidiendo que el resto de los sensores funcionen de manera adecuada ("*Figura 55*").

```
digitalWrite(C_IN1, LOW);     // desenergiza todas las bobinas
digitalWrite(C_IN2, LOW);
digitalWrite(C_IN3, LOW);
digitalWrite(C_IN4, LOW);
```

Figura 55. Líneas de código encargadas de desenergizar las bobinas de los motores

Finalmente, el último problema surge cuando todos los sensores están conectados. Se descubre que, debido a que las líneas de alimentación de la breadboard se conectan a los pines de alimentación de la OLED solo en un extremo, como se muestra en la, los sensores ubicados lejos de ese punto de alimentación recibían menos energía y su funcionamiento no era tan eficiente.

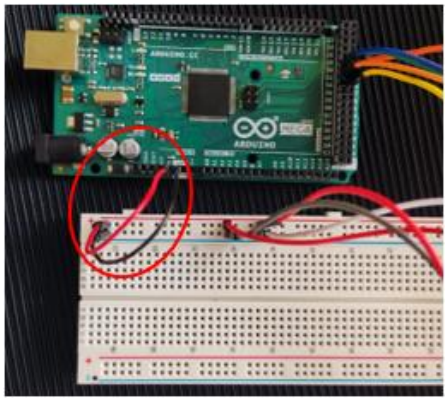


Figura 54. Modo de conexión de la placa Arduino para suministrar alimentación a la “protoboard”

Para resolver esto, se agrega un módulo de alimentación externa (explicado en el punto “*2.3*”) en el extremo de la “protoboard”, lo cual proporcionó una alimentación más homogénea a todos los sensores, solucionando así los problemas de alimentación requeridos en este proyecto.

Otra de las cosas a tener en cuenta en el montaje de los sensores detectores de obstáculos es que, debido a la naturaleza de su funcionamiento, si se encuentran ante luz solar intensa o una superficie reflectiva, su comportamiento puede verse alterado y detectar obstáculos cuando no los hay, por ello se debe reducir la luz de la estancia o cubrirlos en lo posible. El sensor de colores también se ve afectado de esta forma, necesitando ser recalibrado con cada cambio de entorno o condiciones de trabajo.

La programación del código para el funcionamiento de la cinta clasificadora enfrentó varios desafíos, pero mediante la implementación de soluciones como la sincronización mediante tiempos, la optimización de la velocidad de comunicación y la adecuada alimentación de los sensores, se logra obtener un funcionamiento eficiente del sistema.

4.5. Presupuesto

En este punto se elaborará un presupuesto general del proyecto teniendo en cuenta desde los componentes de Arduino hasta el material necesario para la maquetación.

Con este presupuesto lo que se pretende, es que cualquier empresa o particular interesado en implementar el prototipo dispondrá de una estimación aproximada de los costos involucrados. Esto les permitirá tener una idea de la inversión inicial necesaria para llevar a cabo el proyecto, facilitando la toma de decisiones y la planificación financiera.

Tabla 3. Tabla de presupuestos

Material	Cantidad	Coste unitario (€)	Coste total (€)
Arduino Mega 2560 R3	1	44,50	44,50
Protoboard	1	2,80	2,80
Motor paso a paso 28BYJ-48 con driver ULN2003	4	2,95	11,80
OLED 1.3’ SSH1106	1	5,95	5,95
Sensor de color TCS3200	1	8,70	8,70
Detector de obstáculos con sensor de infrarrojos	3	2,13	6,39
Adaptador de corriente	1	7,99	7,99
Placa poliestireno	1	13,99	13,99
Goma cinta	1	17,50	17,50
Engranajes	1	7,59	7,59
		Total	127,21

5. Conclusiones y líneas de trabajo futuras

En conclusión, en este trabajo se ha conseguido un prototipo funcional a nivel educativo, el cual es aplicable para trabajos a pequeña escala y para entender el funcionamiento físico de los sensores y aprender la lógica de la programación en este lenguaje.

Sin embargo, se ha observado que algunos sensores carecen de la fuerza necesaria para soportar un trabajo a gran escala, como son los motores paso a paso. Esto limita su uso en aplicaciones donde se requiere una mayor resistencia y durabilidad.

Además, se ha encontrado que Arduino tiene limitaciones en cuanto a la capacidad de suministrar energía a múltiples sensores simultáneamente. Esto implica que la programación para aplicaciones que involucran muchos sensores funcionando a la vez no es eficiente, ya que se requiere una gestión cuidadosa de la alimentación y la asignación de recursos.

En términos de programación, es necesario implementar técnicas de optimización para mejorar la eficiencia en la gestión de múltiples sensores. Esto incluye el uso de interrupciones y la programación asincrónica lo que resulta en un funcionamiento muy escalonado si se busca utilizar muchos sensores, lo que acaba siendo ineficiente.

Para mejorar estas limitaciones, se podrían considerar las siguientes mejoras. En primer lugar, se podría investigar y utilizar sensores de mayor calidad y resistencia, capaces de soportar trabajos a gran escala o de uso industrial. Además, se podría utilizar una fuente de alimentación externa para asegurar una alimentación adecuada y constante de los sensores.

Otra solución para adaptar el actual proyecto a entornos industriales sería implementar un "*controlador PLC*" ("Controlador Lógico Programable"), el cual está diseñado para resistir condiciones adversas y cuenta con interfaces específicas para la conexión con componentes y sensores industriales. Una de las principales ventajas de utilizar un "controlador PLC" es su capacidad para realizar tareas en tiempo real y manejar múltiples entradas y salidas de forma simultánea. Esto permite una mayor flexibilidad en la programación y control del sistema, ya que se pueden ejecutar diversas acciones de manera coordinada y sincronizada.

El controlador PLC se puede complementar con la placa Arduino Mega al utilizarla como una interfaz de entrada y salida adicional. La placa Arduino Mega puede comunicarse con

el controlador a través de protocolos de comunicación como "*I2C*", "*SPI*", "*RS485*", "*RS232*" o "*MODBUS*", lo que amplía las capacidades de conexión y permite una integración más completa (Shields, 2023).

Aunque Arduino ofrece una solución versátil y accesible para la automatización, es importante considerar las limitaciones de los sensores y la capacidad de alimentación al diseñar aplicaciones a gran escala. Mediante mejoras en la selección de sensores y la optimización de la programación, es posible superar estas limitaciones y lograr una mayor eficiencia en proyectos de automatización más complejos.

Para finalizar, este proyecto ha sido un desafío constante que ha requerido una resolución continua de problemas. Partiendo de un conocimiento prácticamente nulo en programación de Arduino y otros lenguajes, así como en el manejo de sensores, ha implicado un proceso de aprendizaje constante. Además, la construcción de la maqueta, siendo de fabricación casera, ha implicado enfrentarse a problemas y desafíos en su desarrollo para dar sentido al funcionamiento de los sensores. A pesar de los obstáculos, esta experiencia ha sido enriquecedora y me ha permitido adquirir conocimientos en programación, electrónica y resolución de problemas.

Referencias

330ohms. (16 de octubre de 2019). *Mejora tu programación con Arduino: comunicación SPI.* Recuperado el mayo de 2023, de https://blog.330ohms.com/2019/10/16/mejora-tu-programacion-con-arduino-comunicacion-spi/

AENOR. (Marzo de 2016). *aenor.com.* Obtenido de Norma UNE-EN 12882:2016. Cintas transportadoras para usos generales: https://www.en.aenor.com/_layouts/15/r.aspx?c=N0027417

Arduino. (2023). *store.arduino.cc*. Obtenido de https://store.arduino.cc/products/arduino-mega-2560-rev3

Arduino, D. (2023). *Mega 2560 Rev3*. Recuperado el mayo de 2023, de https://docs.arduino.cc/hardware/mega-2560

Az-Delivery. (Junio de 2023). *Módulo infrarrojo IR para sensor de distancia de detección de obstáculos para Raspberry Pi.* Recuperado el Junio de 2023, de https://www.az-delivery.de/es/products/ir-abstand-sensor-modul

Bitwiser, A. (26 de noviembre de 2017). *Paso a Paso (unipolar) 28BYJ-48 con Driver ULN2003*. Recuperado el mayo de 2023, de https://www.youtube.com/watch?v=2-nVV9S7leM&t=736s&ab_channel=BitwiseAr

Bitwiser, A. (12 de septiembre de 2020). *Arduino desde cero en Español*. Recuperado el mayo de 2023, de Capítulo 61 - TCS3200 Sensor de color RGB automatización: https://github.com/bitwiseAr/Curso-Arduino-desde-cero/tree/master/Capitulo61

BOE. (7 de marzo de 1996). *https://www.boe.es/buscar/doc.php?id=BOE-A-1996-5281*. (M. d. Energía, Ed.) Recuperado el mayo de 2023, de https://www.boe.es/buscar/doc.php?id=BOE-A-1996-5281

Bricogeek. (2023). *Arduino MEGA 2560 rev3*. Recuperado el mayo de 2023, de Bricogeek: https://tienda.bricogeek.com/arduino/306-arduino-mega-2560.html?gad=1&gclid=Cj0KCQjw7aqkBhDPARIsAKGa0oLG3SbktOtMi6OECoTaV2aWtRcjqmvdL3KcUiGY9CFD6jj8Vw7ukFsaAveZEALw_wcB

Bricogeek. (mayo de 2023). *Bricogeek: Oled 1.3'*. Obtenido de https://tienda.bricogeek.com/pantallas-oled/905-pantalla-oled-13-ssh1106-128x64.html

Bricogeek. (mayo de 2023). *Bricogeek: Sensor de colores TCS3200*. Obtenido de https://tienda.bricogeek.com/sensores-imagen/652-sensor-de-color-tcs3200.html

Bricogeek. (2023). *Modulo alimentación para protoboard MB-102*. Recuperado el mayo de 2023, de https://tienda.bricogeek.com/fuentes-de-alimentacion/1366-modulo-alimentacion-para-protoboard-mb-102.html?gad=1&gclid=Cj0KCQjw7aqkBhDPARIsAKGa0oJcHy3owBoG40-uuJst2_0MuIRHVQDYoYZSBpcnAt_zR4i9CETDnAwaAv1AEALw_wcB

Crespo, E. (17 de octubre de 2018). *Aprendiendo Arduino*. Recuperado el mayo de 2023, de Motor Paso a Paso con Arduino: https://aprendiendoarduino.wordpress.com/2018/10/17/motor-paso-a-paso-con-arduino/

Datasheet. (2023). *PDF 28BYJ-48 Data sheet.* Obtenido de http://www.datasheet.es/PDF/1006817/28BYJ-48-pdf.html#google_vignette

Electronic, u. (23 de diciembre de 2020). *Basic Introduction to Color Sensor*. Recuperado el mayo de 2023, de https://www.utmel.com/blog/categories/sensors/basic-introduction-to-color-sensor

electronica.uy. (20 de septiembre de 2018). *electronica.uy.* Recuperado el mayo de 2023, de https://www.electronica.uy/images/oled.pdf

García González, A. (15 de octubre de 2014). *¿Cómo funciona el protocolo SPI?* Recuperado el mayo de 2023, de panamahitek: https://panamahitek.com/como-funciona-el-protocolo-spi/#:~:text=El%20SPI%20es%20un%20protocolo,diferentes%20en%20el%20mismo%20cable.

García González, A. (23 de noviembre de 2017). *Comunicación entre Arduino y Digispark por medio de I2C*. Recuperado el mayo de 2023, de panamahitek: https://panamahitek.com/comunicacion-entre-arduino-y-digispark/

Guerra Carmenate, J. (13 de enero de 2022). *Arduino Mega 2560 el hermano mayor de Arduino UNO*. Recuperado el mayo de 2023, de Programarfacil: https://programarfacil.com/blog/arduino-blog/arduino-mega-2560/

Guerra Carmenate, J. (13 de enero de 2022). *Señal PWM con Arduino y analogWrite*. Recuperado el mayo de 2023, de Programarfacil: https://programarfacil.com/blog/arduino-blog/pwm-con-arduino-analogico/

IDE, A. (mayo de 2023). *Arduino IDE: descarga software*. Recuperado el 2023, de https://www.arduino.cc/en/software

INSST. (04 de junio de 2003). *www.insst.es*. (I. N. Trabajo, Ed.) Recuperado el mayo de 2023, de https://www.insst.es/documents/94886/326853/ntp_089.pdf/bb65df44-5894-4e42-a454-328c696b3712?version=1.0&t=1528460890982

José, F. (12 de 2019). *DescubreArduino*. Recuperado el mayo de 2023, de Cómo programar pantallas oled con Arduino: https://descubrearduino.com/como-programar-pantallas-oled-con-arduino/

José, F. (03 de 2023). *Cómo hacer multitarea con Arduino*. Recuperado el mayo de 2023, de DescubreArduino: https://descubrearduino.com/como-hacer-multitarea-con-arduino/

José, F. (17 de mayo de 2023). *Protoboard, ¿Qué es y cómo se usa?* Recuperado el mayo de 2023, de DescrubreArduino: https://descubrearduino.com/protoboard/

Keyence. (2023). *Detección basada en la luz*. Recuperado el mayo de 2023, de https://www.keyence.com.mx/ss/products/sensor/sensorbasics/color/feature/

Llamas, L. (2 de junio de 2016). *Luis Llamas: detector de obstáculos con sensor infrarrojos*. Recuperado el mayo de 2023, de https://www.luisllamas.es/detectar-obstaculos-con-sensor-infrarrojo-y-arduino/

Llamas, L. (26 de octubre de 2016). *Luis Llamas: Sensor de colores TCS3200*. Recuperado el mayo de 2023, de https://www.luisllamas.es/medir-color-arduino-colorimetro-tcs3200/

Logicbus. (10 de junio de 2019). *Ejemplos de aplicaciones de sensores de color*. Recuperado el mayo de 2023, de https://www.logicbus.com.mx/blog/los-beneficios-de-usar-sensores-de-color-sobre-sensores-estandar/

MCI, E. (2023). *arduino.cl*. Recuperado el mayo de 2023, de https://arduino.cl/como-configurar-la-comunicacion-uart-en-arduino/

Programarfacil. (2023). *Motor Paso a Paso 28BYJ-48 con Arduino*. (D. M. Jaramillo, Editor) Recuperado el mayo de 2023, de https://programarfacil.com/blog/arduino-blog/motor-paso-a-paso-uln2003-l298n/

Prometec. (16 de junio de 2015). *Prometec: motores paso a paso*. Recuperado el mayo de 2023, de https://www.prometec.net/motor-28byj-48/

Prometec. (2023). *Comunicación con el exterior*. Recuperado el mayo de 2023, de El puerto serie en Arduino y los tipos String: https://www.prometec.net/comunicacion-exterior/

Robotlandia. (2023). *Pantalla OLED azul para Arduino de 1.3" (driver SH1106, SPI).* Recuperado el junio de 2023, de https://www.velleman.eu/downloads/29/infosheets/sh1106_datasheet.pdf

RS. (junio de 2023). *Sensor de color RGB TCS3200 para Arduino.* Recuperado el junio de 2023, de https://es.rs-online.com/web/p/placas-y-kits-compatibles-con-arduino/2163777?cm_mmc=ES-PLA-DS3A-_-google-_-CSS_ES_ES_Pmax_Test-_--_-2163777&matchtype=&gclid=CjwKCAjwv8qkBhAnEiwAkY-ahmxcH-PX1XdzfHW7mJGu2aQfvdNrlbnJQ7za3k1kD32J7C0cZM-fIxoCB6QQAvD_BwE&gclsrc

Shields, I. (mayo de 2023). *Controlador PLC Arduino*. Obtenido de https://www.industrialshields.com/es_ES/industrial-plc-based-on-arduino-original-boards-automation-solutions-202209-lp?gclid=CjwKCAjwkLCkBhA9EiwAka9QRuOXPey0EI7yZy6tLcmC67B9pNfqdRuQn1cEEM_K0P9gtKn4mqZ4LBoCIZkQAvD_BwE

Zrnić, N., Đorđević, M., & Gašić, V. (2022). History of belt conveyors until the end of the 19th century. En N. Zrnić, M. Đorđević, & V. Gašić, *Explorations in the History and Heritage of Machines and Mechanisms* (págs. 210-223). Springer International Publishing.

ANEXOS

ANEXOS

ANEXO 1. CÓDIGO

1. Código principal
2. Código de calibración del sensor de colores TCS3200

ANEXO 2. PLANOS

PLANO 1. Vista isométrica conjunto maqueta

PLANO 2. Vistas conjunto maqueta

PLANO 3. Descomposición piezas estructura principal

PLANO 4. Plano componentes estructura cinta

PLANO 5. Plano componentes sensores

PLANO 6. Plano de los elementos de las cajas clasificadoras

PLANO 7. Plano detalles cilindros conductores de movimiento

ANEXO 3. HOJA DE DATOS DE LOS COMPONENTES

A. Hoja de datos Arduino Mega 2560 Rev3
B. Hoja de datos del motor paso a paso 28BYJ-48 con driver ULN2003
C. Hoja de datos detecto de obstáculos con sensor infrarrojo FC51
D. Hoja de datos OLED 1.3’ SSH1106
E. Hoja de datos sensor de colores TCS3200

ANEXO 1. CÓDIGO

1. Código principal

Se presenta el código principal del funcionamiento de la cinta transportadora. Su explicación más detallada se redacta en los puntos "*4.1*" y "*4.3*".

```
/* ANEXO 1: PROGRAMA

DISEÑO Y AUTOMATIZACIÓN DE UNA CINTA TRANSPORTADORA CLASIFICADORA

AUTORA: Verónica Morán García

BIBLIOGRAFÍA: Algunas partes del código fueron tomadas de los siguientes
enlaces

    http://www.prometec.net/motor-28byj-48

    https://www.ardu-badge.com/Adafruit%20SH110X

    https://www.luisllamas.es/detectar-obstaculos-con-sensor-infrarrojo-y-
arduino/

*/



/*****************************************************************/



/* LIBRERIAS DE LOS SENSORES */

  // Se incluyen las librerías necesarias de todos los sensores

 /* Libreria pantalla OLED */

  #include <SPI.h>              // Librería que permite la comunicación
serial síncrona (SPI) entre dispositivos
  #include <Wire.h>             // Librería que permite la comunicación de
dispositivos mediante el protocolo I2C
  #include <Adafruit_GFX.h>     // Librería gráfica con funciones y
herramientas para el manejo de gráficos en pantallas
  #include <Adafruit_SH110X.h>  // Librería que controla pantallas OLED
basadas en el controlador SH110X, fabricadas por Adafruit



 /* Librería motor stepper */
```

```
  #include <Stepper.h>      // Permite controlar motores stepper



/*****************************************************************/



/* PINES SENSORES */

  // Se definen los pines y parámetros necesarios para el uso de los
sensores

/* PINES OLED */

 /* Se declaran constantes para asignar los pines de Arduino que se
utilizarán para la comunicación con una pantalla OLED */

  const int OLED_CLK   = 13;  // Pin de reloj para la comunicación SPI
  const int OLED_MOSI = 12;  // Pin de datos (MOSI) para la comunicación
SPI
  const int OLED_RST   = 11;  // Pin para controlar el reinicio (reset) de
la pantalla OLED
  const int OLED_DC    = 10;  // Pin para controlar si los datos enviados
son comandos o información a mostrar en la pantalla OLED
  const int OLED_CS    = 9;   // Pin de selección del chip (chip select)
para la comunicación SPI con la pantalla OLED

 /* Se crea un objeto de la clase Adafruit_SH1106G que representa la
pantalla OLED y se pasan los valores de configuración necesarios para
inicializarla correctamente*/

  Adafruit_SH1106G display = Adafruit_SH1106G(128, 64,OLED_MOSI, OLED_CLK,
OLED_DC, OLED_RST, OLED_CS);

 /* Se crea un logo de inicio de la pantalla OLED */

  #define LOGO16_GLCD_HEIGHT 16
  #define LOGO16_GLCD_WIDTH  16

  static const unsigned char PROGMEM logo16_glcd_bmp[] =
  { B00000000, B11000000,
    B00000001, B11000000,
    B00000001, B11000000,
    B00000011, B11100000,
    B11110011, B11100000,
    B11111110, B11111000,
```

```
    B01111110, B11111111,
    B00110011, B10011111,
    B00011111, B11111100,
    B00001101, B01110000,
    B00011011, B10100000,
    B00111111, B11100000,
    B00111111, B11110000,
    B01111100, B11110000,
    B01110000, B01110000,
    B00000000, B00110000
  };



/* PINES SENSOR DE OBSTÁCULOS */

  const int sensorObstaculo_A = 8;  // Pin correspondiente al sensor
obstáculos de control de bloques rojos

  const int sensorObstaculo_B = 7;  // Pin correspondiente al sensor de
obstáculos obstáculos de control de bloques verdes

  const int sensorObstaculo_C = 6;  // Pin correspondiente al sensor de
obstáculos obstáculos de control de bloques verdes



/* PINES STEP MOTOR CINTA */

  // Se declaran los pines del motor correspondiente al giro de la cinta

  const int Cinta_IN1 = 22;
  const int Cinta_IN2 = 23;
  const int Cinta_IN3 = 24;
  const int Cinta_IN4 = 25;



/* PINES SENSOR DE COLORES */

  // Se declaran los diferentes pines del sensor de colores

  const int S0        = 29;
  const int S1        = 28;
  const int S2        = 31;
  const int S3        = 30;
  const int salidaTCS = 33;   // salidaTCS se utilizará para leer los datos
de color detectados por el sensor de colores
```

```
/* PINES STEP MOTOR A */

  // Se declaran los pines del motor que empuja las cajas rojas

  const int A_IN1 = 36;
  const int A_IN2 = 37;
  const int A_IN3 = 38;
  const int A_IN4 = 39;



/* PINES STEP MOTOR B */

  // Se declaran los pines del motor que empuja las cajas verdes

  const int B_IN1 = 44;
  const int B_IN2 = 45;
  const int B_IN3 = 46;
  const int B_IN4 = 47;



/* PINES STEP MOTOR C */

  // Se declaran los pines del motor que empuja las cajas azules

  const int C_IN1 = 48;
  const int C_IN2 = 49;
  const int C_IN3 = 50;
  const int C_IN4 = 51;


/*****************************************************************/



/* DECLARACIÓN DE LAS FUNCIONES EMPLEADAS EN EL CÓDIGO */


/* FUNCIONES OLED */

  void setup_OLED ();              // Función que inicializa la oled

  void encenderPantalla();         // Función que escribe por pantalla el
texto mostrado de forma continua
```

```
  void actualizarPantalla();        // Función que actualiza el texto del
contador

  bool pantallaEncendida = false;   // Comprueba si la oled está encendida
o no



/* FUNCIONES SENSOR OBSTÁCULOS */

  void setup_sensorObstaculos_A(); // Funcion que inicializa el sensor de
obstáculos del motor A rojo

  void setup_sensorObstaculos_B(); // Funcion que inicializa el sensor de
obstáculos del motor B verde

  void setup_sensorObstaculos_C(); //funcion que inicializa el sensor de
obstáculos del motor C azul



/* FUNCIONES STEP CINTA */

  // Se declaran las funciones relacionadas con el control del motor giro
de la cinta

  int steps_left = 4095;  // Variable se utiliza para realizar un
seguimiento del número de pasos restantes que el motor debe realizar antes
de detenerse
  bool Direction = true;  // Variable para controlar la dirección de giro
del motor paso a paso
  int Steps = 0;          // Variable que se utiliza para llevar un
registro de los pasos que se han realizado hasta el momento

  int Paso [ 8 ][ 4 ] =   // Matriz que contiene el patrón de activación de
las bobinas del motor paso a paso para lograr el movimiento
      {   {1, 0, 0, 0},
          {1, 1, 0, 0},
          {0, 1, 0, 0},
          {0, 1, 1, 0},
          {0, 0, 1, 0},
          {0, 0, 1, 1},
          {0, 0, 0, 1},
          {1, 0, 0, 1}
      };

  void paradaCinta();     // Declaranción de la función que se utiliza para
detener el movimiento de la cinta transportadora
```

```
  void movimientoCinta(); // Declaración de la función que permite
controlar el movimiento de la cinta transportadora
  void setup_stepCinta(); // Declaración de la función que se utiliza para
configurar el entorno y los parámetros iniciales del motor paso a paso



/* FUNCIONES STEP A */

  // Se declaran las funciones relacionadas con el control del motor de
empuje de los bloques rojos

  int steps_left_A = 2048;
  bool Direction_A = true;
  int Steps_A = 0;

  int Paso_A [ 8 ][ 4 ] =
      {   {1, 0, 0, 0},
          {1, 1, 0, 0},
          {0, 1, 0, 0},
          {0, 1, 1, 0},
          {0, 0, 1, 0},
          {0, 0, 1, 1},
          {0, 0, 0, 1},
          {1, 0, 0, 1}
      };

  void setup_stepA();         // Función inicialización motor step A
  void step_A_empuje();       // Función de movimiento de avance y retroceso
del motor A
  void stepper_A();           // Función que define la ejecución de los
pasos del motor
  void SetDirection_A();      // Función que invierte la dirección de giro



/* FUNCIONES STEP B */

  // Se declaran las funciones relacionadas con el control del motor de
empuje de los bloques verdes

  int steps_left_B = 2048;
  bool Direction_B = true;
  int Steps_B = 0;

  int Paso_B [ 8 ][ 4 ] =
      {   {1, 0, 0, 0},
          {1, 1, 0, 0},
          {0, 1, 0, 0},
```

```
          {0, 1, 1, 0},
          {0, 0, 1, 0},
          {0, 0, 1, 1},
          {0, 0, 0, 1},
          {1, 0, 0, 1}
      };

  void setup_stepB();         //Función inicialización motor step B
  void step_B_empuje();       //Función de movimiento de avance y retroceso
del motor B
  void stepper_B();           //Función que define la ejecución de los
pasos del motor
  void SetDirection_B();      //Función que invierte la dirección de giro



/* FUNCIONES STEP C */

  // Se declaran las funciones relacionadas con el control del motor de
empuje de los bloques azules

  int steps_left_C = 2048;
  bool Direction_C = true;
  int Steps_C = 0;

  int Paso_C [ 8 ][ 4 ] =
      {   {1, 0, 0, 0},
          {1, 1, 0, 0},
          {0, 1, 0, 0},
          {0, 1, 1, 0},
          {0, 0, 1, 0},
          {0, 0, 1, 1},
          {0, 0, 0, 1},
          {1, 0, 0, 1}
      };

  void setup_stepC();         //Función inicialización motor step C
  void step_C_empuje();       //Función de movimiento de avance y retroceso
del motor C
  void stepper_C();           //Función que define la ejecución de los
pasos del motor
  void SetDirection_C();      //Función que invierte la dirección de giro



/* FUNCIONES SENSOR DE COLORES */

  bool sensorColores_activo = true;           // Variable que se utiliza
para indicar si el sensor de colores está activo o no
```

```
  void setup_sensorColores();                 // Declaración de la función
encargada de configurar el entorno y los parámetros iniciales del sensor de
colores

  int identificarColor();                     // Declaración de la función
que se utiliza para identificar el color detectado por el sensor de colores
  int obtenerValorConFiltro(int s2, int s3);  // Declaración de la función
que recibe dos parámetros s2 y s3 y se utiliza para obtener el valor del
sensor de colores aplicando un filtro
  int colorLeido = 0;                         // Variable que se utiliza
para almacenar el valor del color detectado por el sensor de colores

  #define COLOR_ROJO  1                       // Se definen macros que
representan los diferentes colores que pueden ser detectados por el sensor
de colores
  #define COLOR_VERDE  2
  #define COLOR_AZUL  3



/* CONTADOR OBSTÁCULOS ROJO */

  int numObjetos = 2;               //numero de objetos a detectar, esta
variable la empleo para todos los sensores de obstaculos

  int contador_rojo = 0;            // Contador de objetos rojos


/* CONTADOR OBSTÁCULOS VERDE */

  int contador_verde = 0;           // Contador de obstaculos verdes


/* CONTADOR OBSTÁCULOS AZUL */

  int contador_azul = 0;            // Contador de obstaculos verdes



/* CONTADOR TIEMPO */

  unsigned long tiempoInicio = 0;           // Variable para almacenar el
tiempo en que se inicia el movimiento

  unsigned long tiempoTranscurrido = 0;     // Variable para almacenar el
tiempo transcurrido desde el inicio del movimiento
```

```
  const unsigned long tiempoEspera = 500;   // Variable para definir el
tiempo de espera deseado en milisegundos

  long tiempo = 0;                          //Inicio de la variable tiempo
en 0s



/*****************************************************************/



/* CÓDIGO PRINCIPAL */


void setup() {

  Serial.begin(250000);   //iniciar puerto serie 250000 baudios

  //Inicializacion del sensor de obstaculos A correspondiente con el color
rojo
  setup_sensorObstaculos_A();

  //Inicialización del sensor de obstáculos B correspondiente con el color
verde
  setup_sensorObstaculos_B();

  //Inicialización del sensor de obstáculos C correspondiente con el color
azul
  setup_sensorObstaculos_C();

  //Inicialización de la pantalla OLED del contador
  setup_OLED();

  //Inicialización del motor de la cinta
  setup_stepCinta();

  //Inicialización del motor de empuje de los bloques rojos
  setup_stepA();

  //Inicialización del motor de empuje de los bloques verdes
  setup_stepB();

  //Inicialización del motor de empuje de los bloques azules
  setup_stepC();

  //Inicialización del sensor de colores
```

```
  setup_sensorColores();

}


  /* Se define el código del funcionamiento principal de la cinta*/

void loop() {

  if(pantallaEncendida == false){      // Se comprueba si la pantalla esta
encendida, si no lo esta se enciende y NO SE REFRESCA

    encenderPantalla();                // Se enciende la pantalla

    pantallaEncendida = true;          // Se declara que la pantalla está
encendida

  }

  tiempoTranscurrido = millis() - tiempoInicio;    // Se crea un contador
de tiempo que se va actualizando en tiempo real

  Serial.println(tiempoTranscurrido);              // Se escribe el tiempo
por pantalla de ordenador para comprobar el correcto funcionamiento


  //Condición de movimiento del cinta de manera continua

  if(contador_rojo < numObjetos && contador_verde < numObjetos &&
contador_azul < numObjetos){    // Cuando el contador de objetos no ha
llegado a los definidos

    if (sensorColores_activo == true) {           // Si el sensor de
colores está activo (true)

      colorLeido = identificarColor();            // Comienza a identificar
colores

      if (colorLeido != 0){                       // Si el color leído es
alguno de los declarados

        sensorColores_activo = false;             // Cambia el estado del
sensor de colores para que deje de leer

      }

    } else if (sensorColores_activo == false ){   //Cuando el estado del
sensor de colores esté desactivado
```

```
      movimientoCinta();                          //Gira el motor de la
cinta de manera continua

    }


    //Condición de funcionamiento de sensor A y correspondiente contador de
objetos rojos

    if (tiempoTranscurrido > 3000 && digitalRead(sensorObstaculo_A) == LOW
&& colorLeido == COLOR_ROJO ){    //Se establece que si han pasado 3
segundos y el sensor de infrarrojos detecta obstáculos

      paradaCinta();                       // Se para la cinta

      tiempoInicio = millis();             // La variable tiempo vuelve a 0

      contador_rojo ++;                    // Se suma el contador
correspondiente

      actualizarPantalla();                // Se refresca pantalla para
añadir este nuevo dato

      Serial.println("Obstaculo rojo");    // Se escribe por pantalla de
ordenador también para comprobar

      sensorColores_activo = true;         // Cambia el estado del sensor de
colores para que deje de leer

      step_A_empuje();                     // Se activa el funcionamiento
del motor de empuje de bloques rojos

      colorLeido = 0;                      // Se vuelve a dejar limpia la
variable que almacena el color leído


    } else if (tiempoTranscurrido > 5000 && digitalRead(sensorObstaculo_B)
== LOW && colorLeido == COLOR_VERDE ){  // Se establece que si han pasado 5
segundos y el sensor de infrarrojos detecta obstáculos

      paradaCinta();                        // Se para la cinta

      tiempoInicio = millis();              // La variable tiempo vuelve a 0

      contador_verde ++;                    // Se suma el contador
correspondiente

      actualizarPantalla();                 // Se refresca pantalla para
añadir este nuevo dato
```

```
      Serial.println("Obstaculo verde");    // Se escribe por pantalla de
ordenador también para comprobar

      sensorColores_activo = true;         // Cambia el estado del sensor
de colores para que deje de leer

      step_B_empuje();                     // Se activa el funcionamiento
del motor de empuje de bloques verdes

      colorLeido = 0;                      // Se vuelve a dejar limpia la
variable que almacena el color leído

    } else if (tiempoTranscurrido > 7000 && digitalRead(sensorObstaculo_C)
== LOW && colorLeido == COLOR_AZUL ){  // Se establece que si han pasado 7
segundos y el sensor de infrarrojos detecta obstáculos

      paradaCinta();                       // Se para la cinta

      tiempoInicio = millis();             // La variable tiempo vuelve a 0

      contador_azul ++;                    // Se suma el contador
correspondiente

      actualizarPantalla();                // Se refresca pantalla para
añadir este nuevo dato

      Serial.println("Obstaculo azul");    // Se escribe por pantalla de
ordenador también para comprobar

      sensorColores_activo = true;         // Cambia el estado del sensor
de colores para que deje de leer

      step_C_empuje();                     // Se activa el funcionamiento
del motor de empuje de bloques azules

      colorLeido = 0;                      // Se vuelve a dejar limpia la
variable que almacena el color leído

    }

  }

}
```

```
/*****************************************************************/
```

```
/* FUNCIONES */


/* OLED */

  /* INICIALIZAR OLED */

  // Función de inicialización de la pantalla OLED llamada en void setup

  void setup_OLED (){

    display.begin(0, true); // Inicia la comunicación con la pantalla OLED
    display.display();      // Muestra en la pantalla OLED los cambios
realizados desde la última actualización
    display.clearDisplay(); // Borra cualquier contenido previo en la
pantalla, dejándola en blanco y lista para mostrar nueva información

  }


  /* ESCRIBIR LOS VALORES DE LOS COLORES POR PANTALLA*/

  // La siguiente función escribe por pantalla todos los valores y se
mantiene, no se refresca

  void encenderPantalla(){

    display.setTextSize(1.5);                 // Se define el tamaño del
texto
    display.setTextColor(SH110X_WHITE);       // Se define el color del
texto
    display.setCursor(0, 12);                 // Se define la posición del
texto
    display.println("---------------------"); // Se muestran los simbolos
por pantalla

    display.setTextSize(1.5);
    display.setTextColor(SH110X_WHITE);
    display.setCursor(0, 24);
    display.println("Rojo:");                 // Se escribe el texto en
pantalla

    display.setTextSize(1.5);
    display.setTextColor(SH110X_WHITE);
```

```
    display.setCursor(0, 36);
    display.println("Verde:");

    display.setTextSize(1.5);
    display.setTextColor(SH110X_WHITE);
    display.setCursor(0, 48);
    display.println("Azul:");

    display.setTextSize(1.5);
    display.setTextColor(SH110X_WHITE);
    display.setCursor(48, 24);
    display.println(contador_rojo);      // Se escribe por pantalla el
valor del contador de objetos rojos

    display.setTextSize(1.5);
    display.setTextColor(SH110X_WHITE);
    display.setCursor(48, 36);
    display.println(contador_verde);    // Se escribe por pantalla el valor
del contador de objetos verdes

    display.setTextSize(1.5);
    display.setTextColor(SH110X_WHITE);
    display.setCursor(48, 48);
    display.println(contador_azul);     // Se escribe por pantalla el valor
del contador de objetos azules

    display.display();                  // Se muestra por pantalla todo lo
anterior
  }


  /* REFRESCAR PANTALLA */

  // La siguiente función se encarga de refrescar la pantalla para incluir
los nuevos valores leídos por el sensor

  void actualizarPantalla() {

    display.clearDisplay();

    display.setTextSize(1.5);
    display.setTextColor(SH110X_WHITE);
    display.setCursor(0, 12);
    display.println("---------------------");

    display.setTextSize(1.5);
    display.setTextColor(SH110X_WHITE);
    display.setCursor(0, 24);
    display.println("Rojo:");
```

```
    display.setTextSize(1.5);
    display.setTextColor(SH110X_WHITE);
    display.setCursor(0, 36);
    display.println("Verde:");

    display.setTextSize(1.5);
    display.setTextColor(SH110X_WHITE);
    display.setCursor(0, 48);
    display.println("Azul:");

    display.setTextSize(1.5);
    display.setTextColor(SH110X_WHITE);
    display.setCursor(48, 24);
    display.println(contador_rojo);
    display.display();

    display.setTextSize(1.5);
    display.setTextColor(SH110X_WHITE);
    display.setCursor(48, 36);
    display.println(contador_verde);

    display.setTextSize(1.5);
    display.setTextColor(SH110X_WHITE);
    display.setCursor(48, 48);
    display.println(contador_azul);

    display.display();

  }



/* SENSOR DE OBSTÁCULOS */

  /* INICIALIZAR SENSOR OBSTÁCULOS A */

  // Se definen todas las funciones de los sensores de obstáculos para
inicializarlos y se llaman desde void setup

  void setup_sensorObstaculos_A(){
    pinMode(sensorObstaculo_A , INPUT);  //definir pin como entrada
  }

  /* INICIALIZAR SENSOR OBSTÁCULOS B */

  void setup_sensorObstaculos_B(){
    pinMode(sensorObstaculo_B , INPUT);  //definir pin como entrada
  }
```

```
  /* INICIALIZAR SENSOR OBSTÁCULOS B */

  void setup_sensorObstaculos_C(){
    pinMode(sensorObstaculo_C , INPUT);  //definir pin como entrada
  }



/* FUNCION STEP MOTOR CINTA */

  void setup_stepCinta(){

    //Inicialización del motor step de la cinta
    pinMode(Cinta_IN1, OUTPUT);
    pinMode(Cinta_IN2, OUTPUT);
    pinMode(Cinta_IN3, OUTPUT);
    pinMode(Cinta_IN4, OUTPUT);

    // Establece la dirección como verdadera (solo hacia adelante)
    Direction = true;

  }

  void paradaCinta(){             // Función para apagar y desenergizar el
motor

    digitalWrite(Cinta_IN1, LOW); // Apaga todos los pines
    digitalWrite(Cinta_IN2, LOW);
    digitalWrite(Cinta_IN3, LOW);
    digitalWrite(Cinta_IN4, LOW);

  }

  void movimientoCinta(){        // Función que define el movimiento del
motor de la cinta

      digitalWrite( Cinta_IN1, Paso[Steps][ 0] );  // Establece el estado
del pin correspondiente según el valor almacenado en la matriz Paso
      digitalWrite( Cinta_IN2, Paso[Steps][ 1] );
      digitalWrite( Cinta_IN3, Paso[Steps][ 2] );
      digitalWrite( Cinta_IN4, Paso[Steps][ 3] );

      Steps = ( Steps + 1 ) % 8 ;  // Incrementa el valor de la variable
Steps en 1 y lo mantiene dentro del rango de 0 a 7. Permite que el índice
de la matriz Paso avance a la siguiente secuencia de pasos.

      steps_left-- ;               // Un paso menos
```

```
      delay(1);                     // Se hace un delay de 1 microsegundo
necesario para el correcto funcionamiento del motor

    steps_left=4095;                // Reestablece el valor de la variable
que almacena el número de pasos del motor

  }



/* FUNCION STEP MOTOR A*/

  void setup_stepA(){

    pinMode(A_IN1, OUTPUT);
    pinMode(A_IN2, OUTPUT);
    pinMode(A_IN3, OUTPUT);
    pinMode(A_IN4, OUTPUT);

  }

  void step_A_empuje(){

    for(int j = 0; j<2;j++){   // Se ejecuta dos veces para realizar el
movimiento de empuje dos veces en la cinta

      while(steps_left_A>0 ){  // Mientras aún quedan pasos por realizar
para completar el movimiento de empuje

        stepper_A() ;          // Avanza un paso
        steps_left_A-- ;       // Un paso menos
        delay (1) ;

      }

      delay(300);                   // Pausa cuando completa el primer
movimiento
      Direction_A = !Direction_A;  // Se invierta la dirección de giro
      steps_left_A = 2048;         // Se vuelven a restaurar los pasos a
dar

    }

    digitalWrite(A_IN1, LOW);     // Desenergiza todas las bobinas
    digitalWrite(A_IN2, LOW);
    digitalWrite(A_IN3, LOW);
    digitalWrite(A_IN4, LOW);
```

```
  }

  void stepper_A() {           //Función de movimiento del motor

    digitalWrite( A_IN1, Paso[Steps_A][ 0] );
    digitalWrite( A_IN2, Paso[Steps_A][ 1] );
    digitalWrite( A_IN3, Paso[Steps_A][ 2] );
    digitalWrite( A_IN4, Paso[Steps_A][ 3] );

    SetDirection_A();

  }

  void SetDirection_A() {

    if(Direction_A){

      Steps_A++;

    } else {

      Steps_A--;

    }

    Steps_A = ( Steps_A + 7 ) % 7 ;

  }



/* FUNCION STEP MOTOR B*/

  void setup_stepB(){

    pinMode(B_IN1, OUTPUT);
    pinMode(B_IN2, OUTPUT);
    pinMode(B_IN3, OUTPUT);
    pinMode(B_IN4, OUTPUT);

  }

  void step_B_empuje(){

    for(int m = 0; m<2;m++){

      while(steps_left_B>0 ){

        stepper_B() ;              // Avanza un paso
```

```
        steps_left_B-- ;        // Un paso menos
        delay (1) ;

      }

      delay(300);
      Direction_B = !Direction_B;
      steps_left_B = 2048;

    }

    digitalWrite(B_IN1, LOW);     // desenergiza todas las bobinas
    digitalWrite(B_IN2, LOW);
    digitalWrite(B_IN3, LOW);
    digitalWrite(B_IN4, LOW);

  }

  void stepper_B() {           //Avanza un paso

    digitalWrite( B_IN1, Paso[Steps_B][ 0] );
    digitalWrite( B_IN2, Paso[Steps_B][ 1] );
    digitalWrite( B_IN3, Paso[Steps_B][ 2] );
    digitalWrite( B_IN4, Paso[Steps_B][ 3] );

    SetDirection_B();

  }

  void SetDirection_B() {

    if(Direction_B){

      Steps_B++;

    } else {

      Steps_B--;

    }

    Steps_B = ( Steps_B + 7 ) % 7 ;

  }


/* FUNCION STEP MOTOR C*/
```

```
void setup_stepC(){

  pinMode(C_IN1, OUTPUT);
  pinMode(C_IN2, OUTPUT);
  pinMode(C_IN3, OUTPUT);
  pinMode(C_IN4, OUTPUT);

}

void step_C_empuje(){

  for(int n = 0; n<2;n++){

    while(steps_left_C>0 ){

      stepper_C() ;             // Avanza un paso
      steps_left_C-- ;        // Un paso menos
      delay (1) ;

    }

    delay(300);
    Direction_C = !Direction_C;
    steps_left_C = 2048;

  }

  digitalWrite(C_IN1, LOW);     // desenergiza todas las bobinas
  digitalWrite(C_IN2, LOW);
  digitalWrite(C_IN3, LOW);
  digitalWrite(C_IN4, LOW);

}

void stepper_C() {           //Avanza un paso

  digitalWrite( C_IN1, Paso[Steps_C][ 0] );
  digitalWrite( C_IN2, Paso[Steps_C][ 1] );
  digitalWrite( C_IN3, Paso[Steps_C][ 2] );
  digitalWrite( C_IN4, Paso[Steps_C][ 3] );

  SetDirection_C();

}

void SetDirection_C() {

  if(Direction_C){
```

```
      Steps_C++;

    } else {

      Steps_C--;

    }

    Steps_C = ( Steps_C + 7 ) % 7 ;

  }



/* FUNCIÓN SENSOR DE COLORES */

  /* INICIALIZACIÓN SENSOR DE COLORES */

  void setup_sensorColores(){

   //Se establecen los pines como salidas y las frecuencias
   pinMode(S0, OUTPUT);          // pin 4 como salida
   pinMode(S1, OUTPUT);          // pin 5 como salida
   pinMode(S2, OUTPUT);          // pin 6 como salida
   pinMode(S3, OUTPUT);          // pin 7 como salida
   pinMode(salidaTCS, INPUT);    // pin 8 como salida

   digitalWrite(S0,HIGH);        // establece frecuencia de salida
   digitalWrite(S1,LOW);

  }


  /* FUNCIÓN SENSOR COLORES 1: Encargada de obtener el valor del sensor con
un filtro especifico*/

  int obtenerValorConFiltro(int s2, int s3) {

    digitalWrite(s2, LOW);
    digitalWrite(s3, LOW);
    int valor = pulseIn(salidaTCS, LOW);
    delay(100);

    return valor;

  }
```

```
  /* FUNCIÓN SENSOR COLORES 2: Encargada de llamar a la función
obtenerValorConFiltro mostrando los valores por pantalla y determinando el
color correspondiente */

  int identificarColor() {

    int rojo  = obtenerValorConFiltro(S2, S3);
    int verde = obtenerValorConFiltro(S2, S3);
    int azul  = obtenerValorConFiltro(S2, S3);

    int returnRojo = 1;
    int returnVerde = 2;
    int returnAzul = 3;

    int colorRojo  =  rojo < 27 && verde > 32 && azul > 23;
    int colorVerde =  verde < 40 && rojo < 41 && azul > 33;
    int colorAzul  =  azul < 21 && rojo > 27 && verde > 28;

    Serial.print("R:");
    Serial.print(rojo);

    Serial.print("\t");

    Serial.print("V:");
    Serial.print(verde);

    Serial.print("\t");

    Serial.print("A:");
    Serial.println(azul);

    if (colorRojo) {

      Serial.println("ROJO");

      //Mostrar en la oled el color rojo detectado
      display.clearDisplay();

      display.setTextSize(1.5);
      display.setTextColor(SH110X_WHITE);
      display.setCursor(0, 0);
      display.println("Objeto ROJO");

      display.display();

      return returnRojo;
```

```
    } else if (colorVerde) {

      Serial.println("VERDE");

      //Mostrar en la oled el color rojo detectado
      display.clearDisplay();

      display.setTextSize(1.5);
      display.setTextColor(SH110X_WHITE);
      display.setCursor(0, 0);
      display.println("Objeto VERDE");

      display.display();

      return returnVerde;

    } else if (colorAzul) {

      Serial.println("AZUL");

      //Mostrar en la oled el color rojo detectado
      display.clearDisplay();

      display.setTextSize(1.5);
      display.setTextColor(SH110X_WHITE);
      display.setCursor(0, 0);
      display.println("Objeto AZUL");

      display.display();

      return returnAzul;

    }else{

      return 0;

    }
  }
```

2. Código de calibración del sensor de colores TCS3200

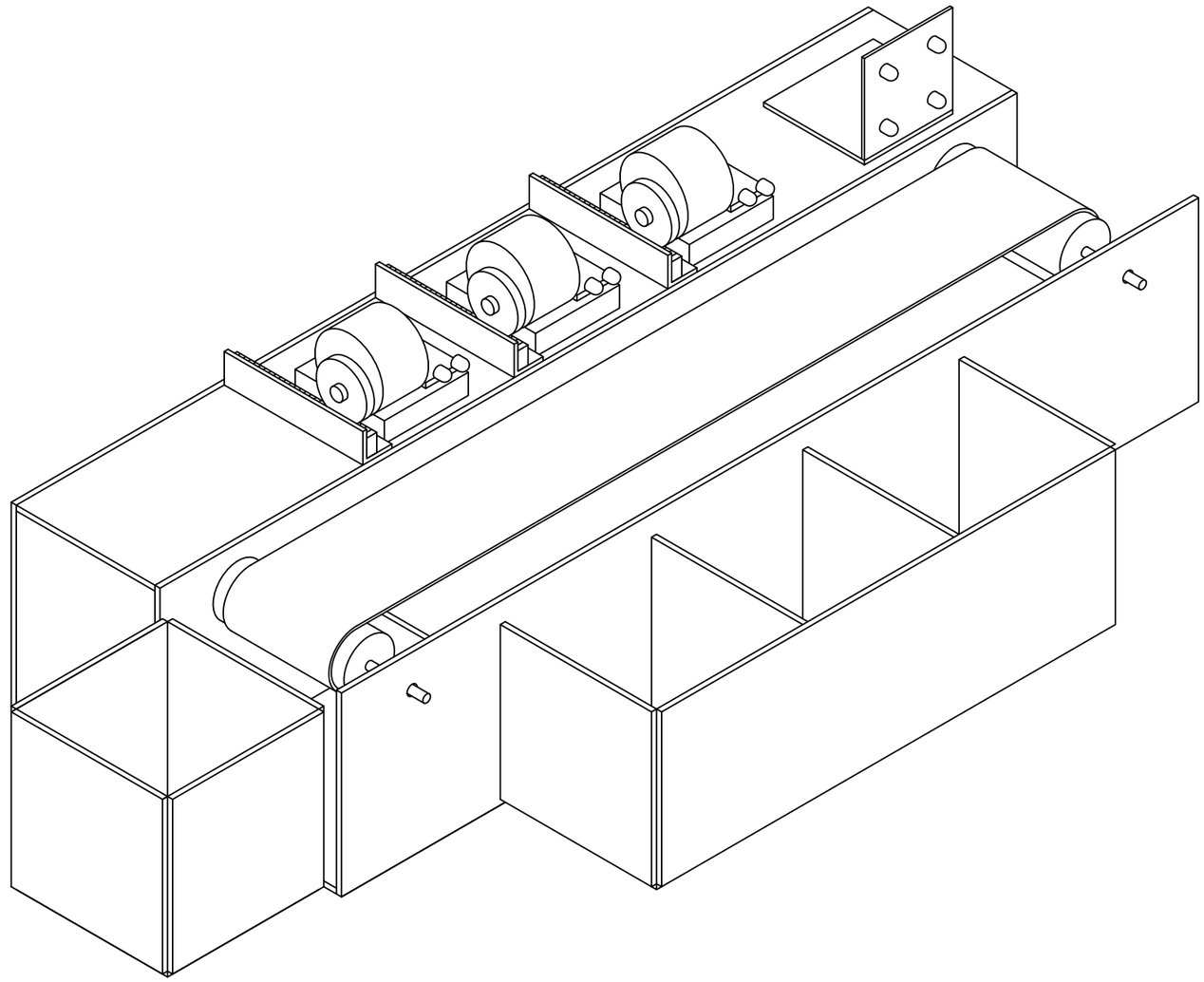
A continuación, se presenta el código de calibración del sensor de colores, este se emplea para determinar el valor de cada componente rojo, azul y verde detectada y definir el color de cada una de las cajas a clasificar. Estos valores se presentan en la "Tabla 2".

```
/*
  Capitulo 61 de Arduino desde cero en Español.
  Visualizacion por monitor serie de las lecturas del sensor de color
  TCS3200 en microsegundos

  https://github.com/bitwiseAr/Curso-Arduino-desde-
cero/blob/master/Capitulo61/Capitulo61-Programa1.txt

*/

#define S0 29   // S0 a pin 4
#define S1 28   // S1 a pin 5
#define S2 31   // S2 a pin 6
#define S3 30   // S3 a pin 7
#define salidaTCS 33  // salidaTCS a pin 8


void setup() {
  pinMode(S0, OUTPUT);    // pin 4 como salida
  pinMode(S1, OUTPUT);    // pin 5 como salida
  pinMode(S2, OUTPUT);    // pin 6 como salida
  pinMode(S3, OUTPUT);    // pin 7 como salida
  pinMode(salidaTCS, INPUT);  // pin 8 como salida

  digitalWrite(S0,HIGH);  // establece frecuencia de salida
  digitalWrite(S1,LOW);   // del modulo al 20 por ciento

  Serial.begin(9600);   // inicializa monitor serie a 9600 bps
}

void loop() {
  digitalWrite(S2,LOW);     // establece fotodiodos
  digitalWrite(S3,LOW);     // con filtro rojo
  int rojo = pulseIn(salidaTCS, LOW); // obtiene duracion de pulso de
salida del sensor
  delay(200);       // demora de 200 mseg

  digitalWrite(S2,HIGH);    // establece fotodiodos
  digitalWrite(S3,HIGH);    // con filtro verde
  int verde = pulseIn(salidaTCS, LOW);  // obtiene duracion de pulso de
salida del sensor
  delay(200);       // demora de 200 mseg
```

```
  digitalWrite(S2,LOW);     // establece fotodiodos
  digitalWrite(S3,HIGH);    // con filtro azul
  int azul = pulseIn(salidaTCS, LOW); // obtiene duracion de pulso de
salida del sensor
  delay(200);       // demora de 200 mseg

  Serial.print("R:");     // muestra texto
  Serial.print(rojo);     // muestra valor de variable rojo

  Serial.print("\t");     // espacio de tabulacion

  Serial.print("V:");     // muestra texto
  Serial.print(verde);      // muestra valor de variable verde

  Serial.print("\t");     // espacio de tabulacion

  Serial.print("A:");     // muestra texto
  Serial.println(azul);     // muestra valor de variable azul

}
```

ANEXO 2. PLANOS

PLANO 1. Vista isométrica conjunto maqueta



Escuela politécnica superior de zamora

VNiVERSiDAD D SALAMANCA

PROYECTO:

DISEÑO Y AUTOMATIZACIÓN DE UNA CINTA CLASIFICADORA

DESIGNACION PLANO:

Plano vista isométrica

conjunto maqueta

FIRMA:

Firmado por MORAN GARCIA VERONICA 71043929B el día 22/06/2023 con un certificado emitido por AC FNMT Usuarios

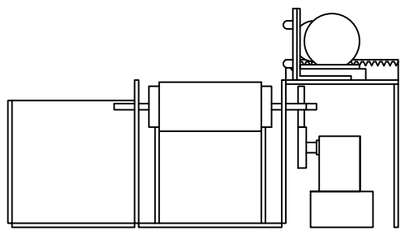
FDO.: VERÓNICA MORÁN GARCÍA
GRADO: INGENIERÍA MECÁNICA

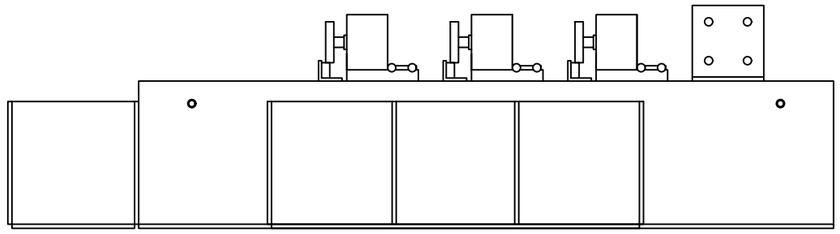
FECHA: 09/06/2023

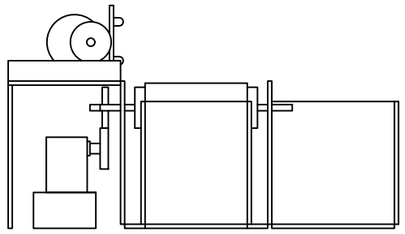
ESCALA: 1/2

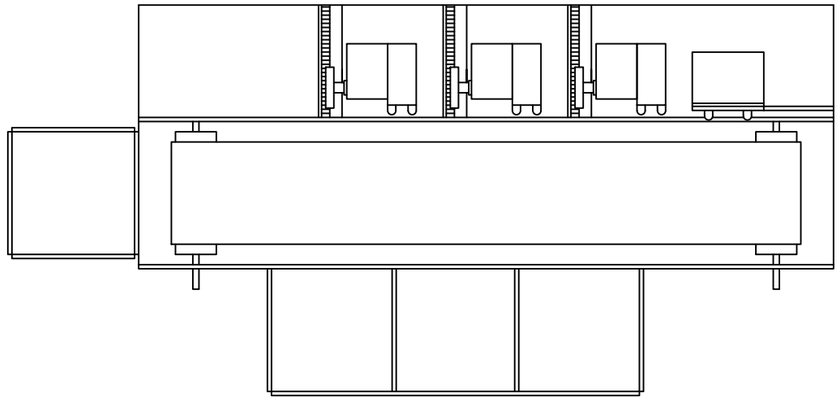
Nº PLANO: 01/07

PLANO 2. Vistas conjunto maqueta









Escuela politécnica superior de zamora

VNiVERSiDAD D SALAMANCA

PROYECTO:

DISEÑO Y AUTOMATIZACIÓN DE UNA CINTA CLASIFICADORA

DESIGNACION PLANO:

Vistas conjunto maqueta

FIRMA:

Firmado por MORAN GARCIA VERONICA 71043929B el día 22/06/2023 con un certificado emitido

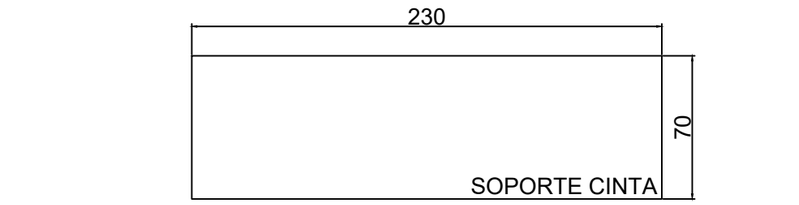
FDO.: VERÓNICA MORÁN GARCÍA
GRADO: INGENIERÍA MECÁNICA

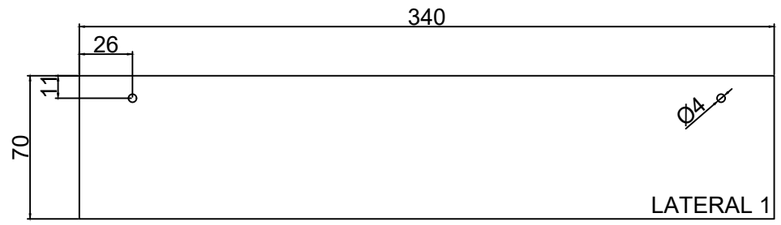
FECHA: 09/06/2023

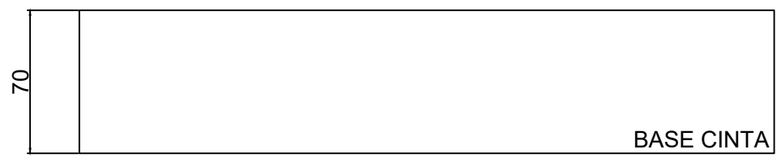
ESCALA: 1/3

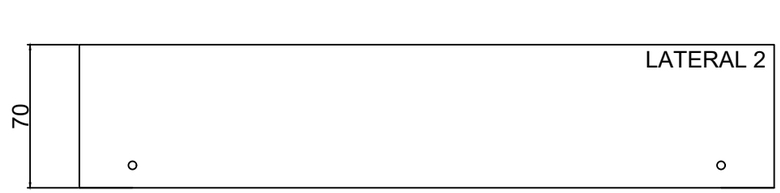
Nº PLANO: 02/07

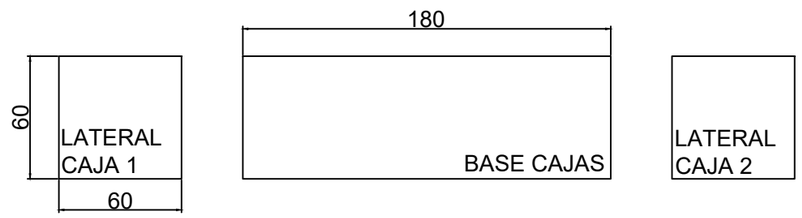
PLANO 3. Descomposición piezas estructura principal

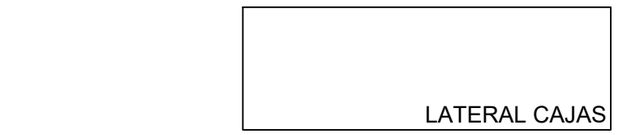


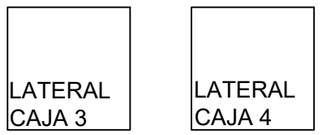


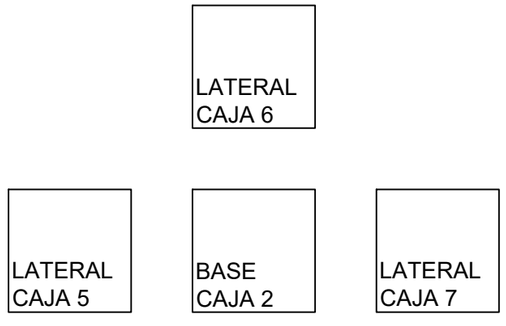












Escuela politécnica superior de zamora

VNiVERSiDAD D SALAMANCA

PROYECTO:

DISEÑO Y AUTOMATIZACIÓN DE UNA CINTA CLASIFICADORA

DESIGNACION PLANO:

Descomposición piezas

estructura principal

FIRMA:

Firmado por MORAN GARCIA VERONICA 71043929B el día 22/06/2023 con un certificado emitido por AC FNMT Usuarios

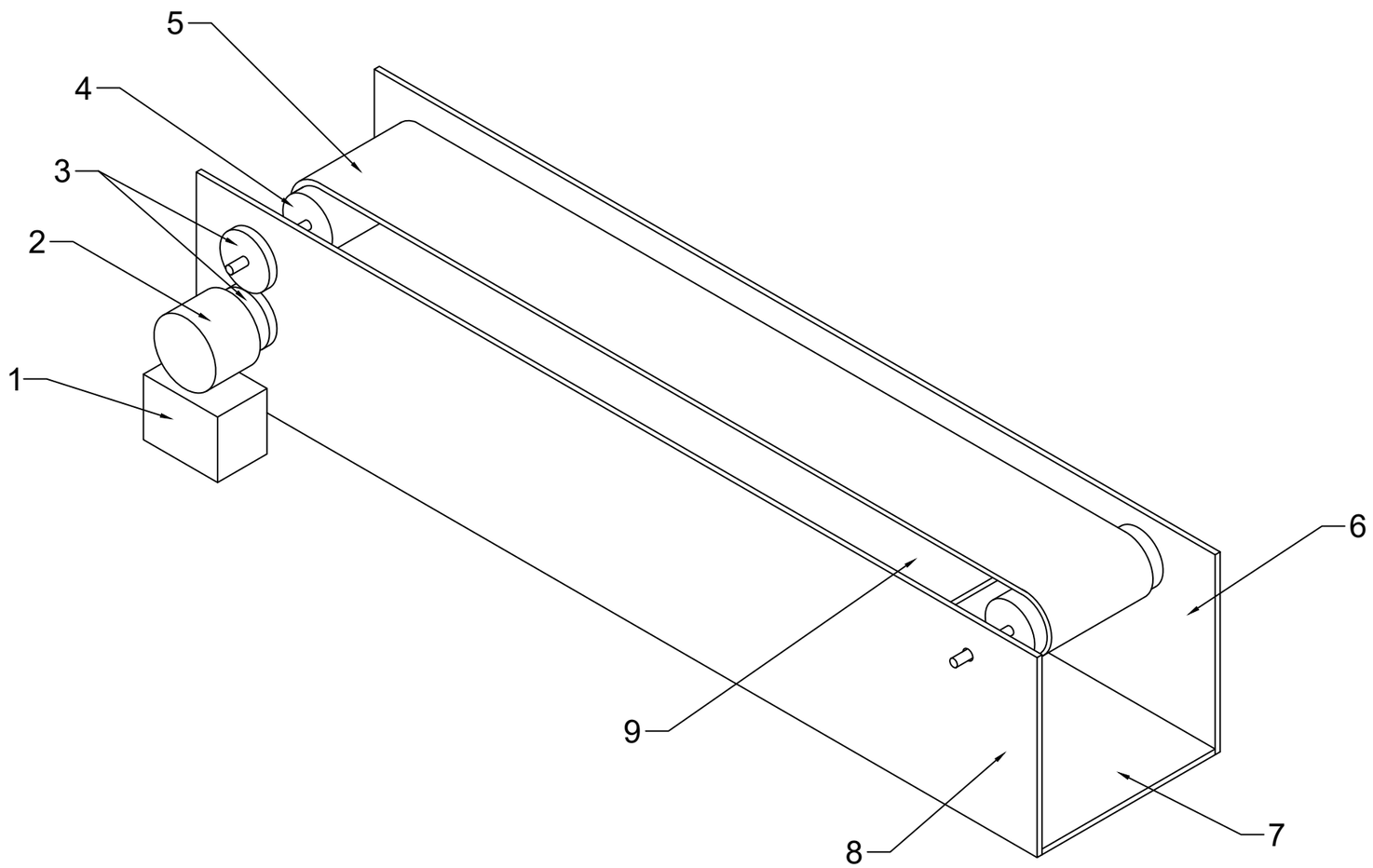
FDO.: VERÓNICA MORÁN GARCÍA
GRADO: INGENIERÍA MECÁNICA

FECHA: 09/06/2023

ESCALA: 1/3

Nº PLANO: 03/07

PLANO 4. Plano componentes estructura cinta



IDENTIFICADOR	CANTIDAD	ELEMENTO
1	1	Soporte motor
2	1	Motor stepper
3	2	Ruedas dentadas
4	2	Cilindros
5	1	Cinta
6	1	Lateral cinta 1
7	1	Base cinta
8	1	Lateral cinta 2
9	1	Soporte cinta

Escuela politécnica superior de zamora

VNiVERSiDAD D SALAMANCA

PROYECTO:

DISEÑO Y AUTOMATIZACIÓN DE UNA CINTA CLASIFICADORA

DESIGNACION PLANO:

Plano componentes

estructura cinta

FIRMA:

Firmado por MORAN GARCIA VERONICA 71043929B el día 22/06/2023 con un certificado emitido por AC FNMT Usuarios

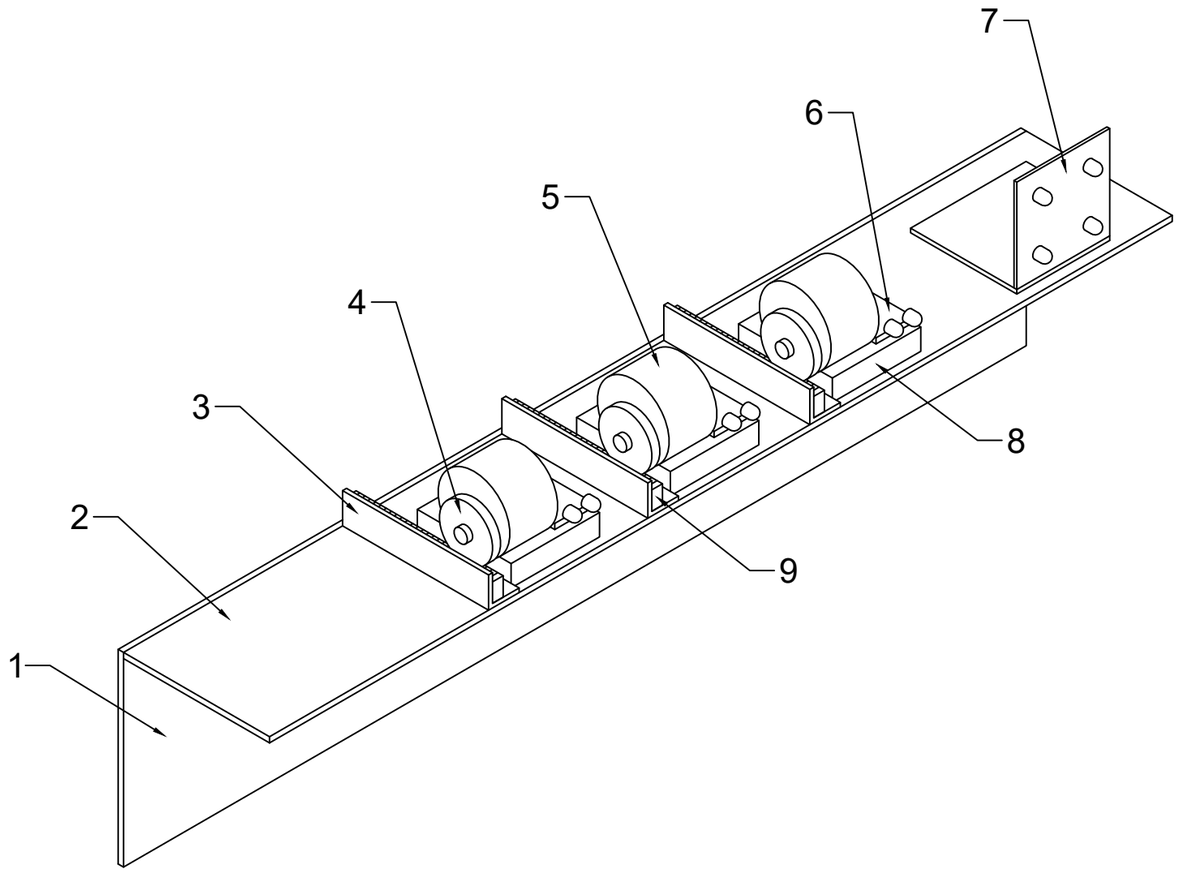
FDO.: VERÓNICA MORÁN GARCÍA
GRADO: INGENIERÍA MECÁNICA

FECHA: 09/06/2023

ESCALA: 1/2

Nº PLANO: 04/07

PLANO 5. Plano componentes sensores



IDENTIFICADOR	CANTIDAD	ELEMENTO
1	1	Placa lateral
2	2	Placa soporte
3	3	Perfil L PVC
4	3	Rueda dentada
5	3	Motor stepper
6	3	Sensor obstáculo
7	1	Sensor color
8	3	Soporte sensores
9	3	Cremallera

Escuela politécnica superior de zamora

VNiVERSiDAD D SALAMANCA

PROYECTO:

DISEÑO Y AUTOMATIZACIÓN DE UNA CINTA CLASIFICADORA

DESIGNACION PLANO:

Plano componentes sensores

FIRMA:

Firmado por MORAN GARCIA VERONICA 71043929B el día 22/06/2023 con un certificado emitido por AC FNMT Usuarios

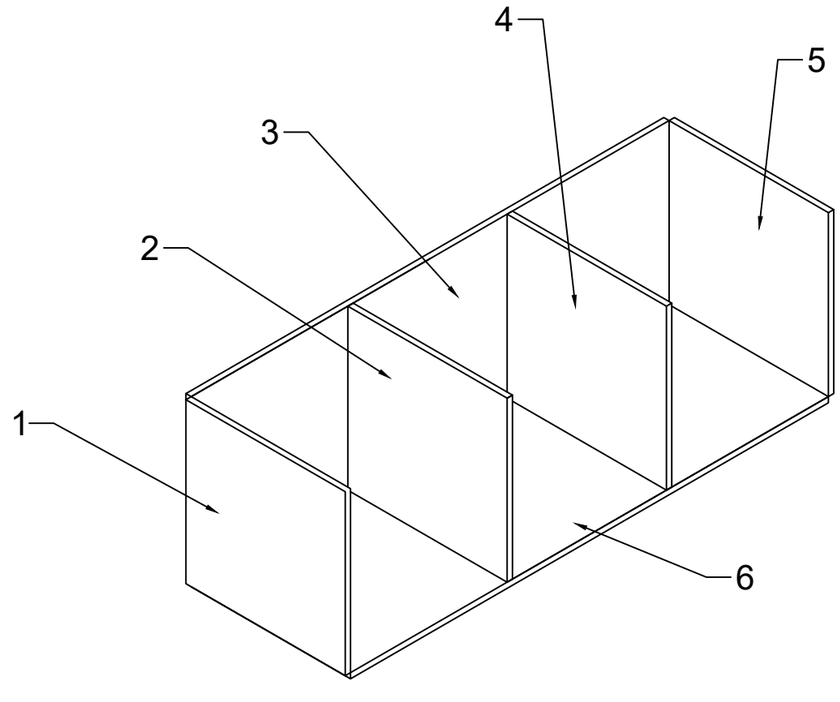
FDO.: VERÓNICA MORÁN GARCÍA
GRADO: INGENIERÍA MECÁNICA

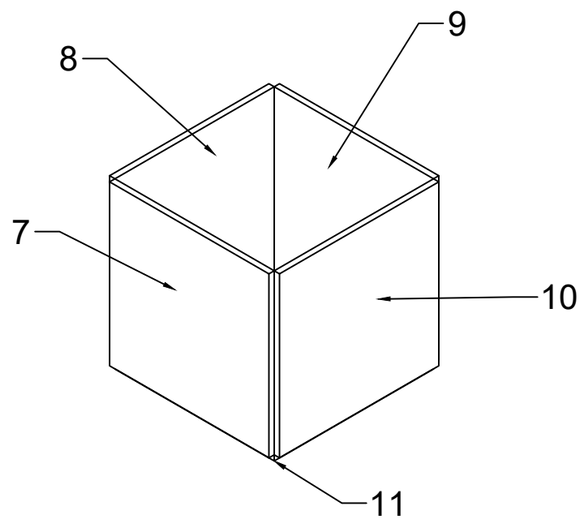
FECHA: 09/06/2023

ESCALA: 1/2

Nº PLANO: 05/07

PLANO 6. Plano de los elementos de las cajas clasificadoras





IDENTIFICADOR	CANTIDAD	ELEMENTO
1	1	Lateral caja 1
2	1	Lateral caja 3
3	1	Lateral cajas
4	1	Lateral caja 4
5	1	Lateral caja 2
6	1	Base cajas
7	1	Lateral caja 5
8	1	Lateral caja 6
9	1	Lateral caja 7
10	1	Lateral caja 8
11	1	Base caja 2

Escuela politécnica superior de zamora

VNiVERSiDAD D SALAMANCA

PROYECTO:

DISEÑO Y AUTOMATIZACIÓN DE UNA CINTA CLASIFICADORA

DESIGNACION PLANO:

Plano de los elementos de las cajas clasificadoras

FIRMA:

Firmado por MORAN GARCIA VERONICA 71043929B el día 22/06/2023 con un certificado emitido por AC FNMT Usuarios

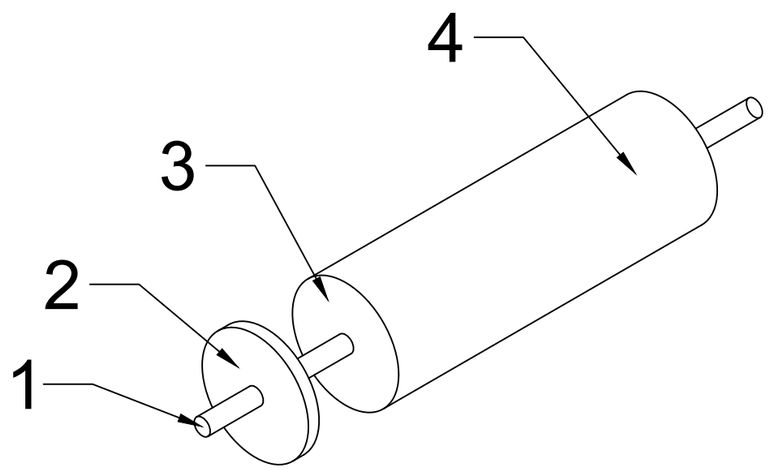
FDO.: VERÓNICA MORÁN GARCÍA
GRADO: INGENIERÍA MECÁNICA

FECHA: 09/06/2023

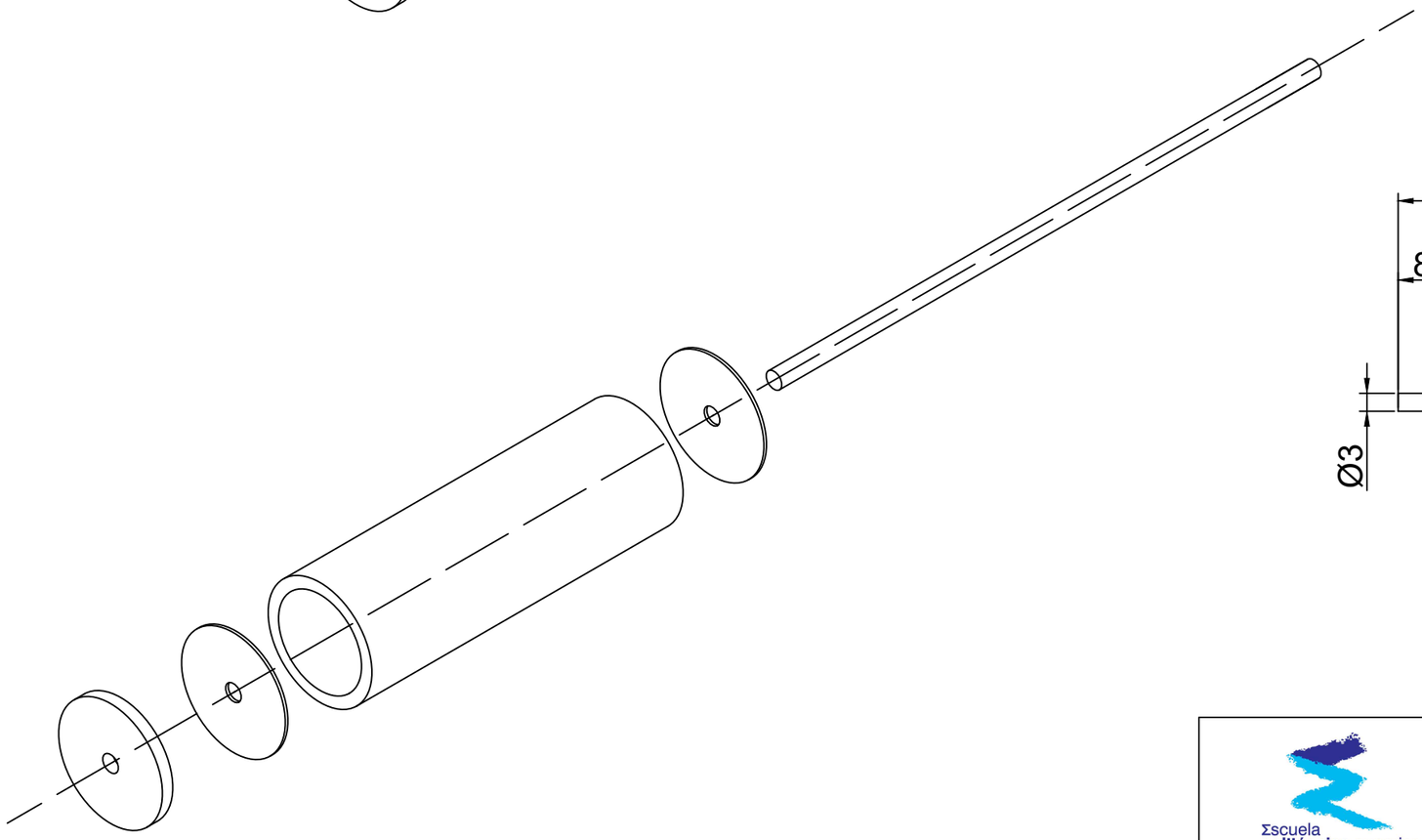
ESCALA: 1/2

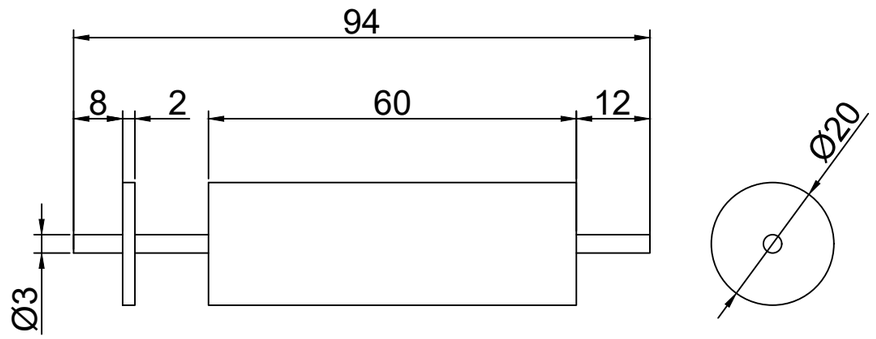
Nº PLANO: 06/07

PLANO 7. Plano detalles cilindros conductores de movimiento



IDENTIFICADOR	CANTIDAD	ELEMENTO
1	2	Varilla cilindro
2	1	Rueda dentada
3	2	Tapaderas
4	1	Cilindros





Escuela politécnica superior de zamora

VNiVERSiDAD D SALAMANCA

PROYECTO:

DISEÑO Y AUTOMATIZACIÓN DE UNA CINTA CLASIFICADORA

DESIGNACION PLANO:

Plano detalles cilindros

conductores de movimiento

FIRMA:

Firmado por MORAN GARCIA VERONICA 71043929B el día 22/06/2023 con un certificado emitido por AC FNMT

FDO.: VERÓNICA MORÁN GARCÍA
GRADO: INGENIERÍA MECÁNICA

FECHA: 09/06/2023

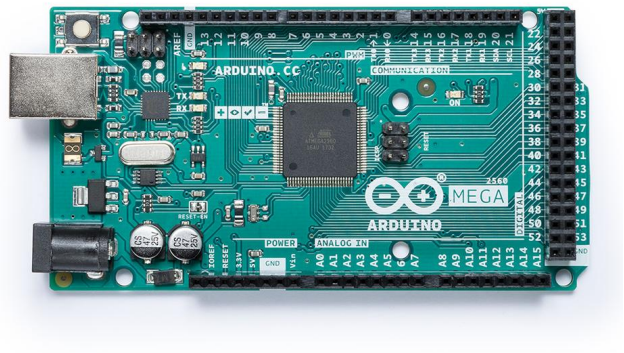
ESCALA: 1/1

Nº PLANO: 07/07

ANEXO 3. HOJA DE DATOS DE LOS COMPONENTES

A. Hoja de datos Arduino Mega 2560 Rev3 (Arduino D. , 2023)

Product Reference Manual
SKU: A000067



Description

Arduino® Mega 2560 is an exemplary development board dedicated for building extensive applications as compared to other maker boards by Arduino. The board accommodates the ATmega2560 microcontroller, which operates at a frequency of 16 MHz. The board contains 54 digital input/output pins, 16 analog inputs, 4 UARTs (hardware serial ports), a USB connection, a power jack, an ICSP header, and a reset button.

Target Areas

3D Printing, Robotics, Maker

Features

- **ATmega2560 Processor**
  - Up to 16 MIPS Throughput at 16MHz
  - 256k bytes (of which 8k is used for the bootloader)
  - 4k bytes EEPROM
  - 8k bytes Internal SRAM
  - 32 × 8 General Purpose Working Registers
  - Real Time Counter with Separate Oscillator
  - Four 8-bit PWM Channels
  - Four Programmable Serial USART
  - Controller/Peripheral SPI Serial Interface
- **ATmega16U2**
  - Up to 16 MIPS Throughput at 16 MHz
  - 16k bytes ISP Flash Memory
  - 512 bytes EEPROM
  - 512 bytes SRAM
  - USART with SPI master only mode and hardware flow control (RTS/CTS)
  - Master/Slave SPI Serial Interface
- **Sleep Modes**
  - Idle
  - ADC Noise Reduction
  - Power-save
  - Power-down
  - Standby
  - Extended Standby
- **Power**
  - USB Connection
  - External AC/DC Adapter
- **I/O**
  - 54 Digital
  - 16 Analog
  - 15 PWM Output

Contents

1 The Board

Arduino® Mega 2560 is a successor board of Arduino Mega, it is dedicated to applications and projects that require large number of input output pins and the use cases which need high processing power. The Arduino® Mega 2560 comes with a much larger set of IOs when we compare it with traditional Uno board considering the form factor of both the boards.

1.1 Application Examples

- **Robotics**: Featuring the high processing capacitity, the Arduino Mega 2560 can handle the extensive robotic applications. It is compatible with the motor controller shield that enables it to control multiple motors at an instance, thus making it perfect of robotic applications. The large number of I/O pins can accommodate many robotic sensors as well.
- **3D Printing**: Algorithms play a significant role in implementation of 3D printers. Arduino Mega 2560 has the power to process these complex algorithms required for 3D printing. Additionally, the slight changes to the code is easily possible with the Arduino IDE and thus 3D printing programs can be customized according to user requirements.
- **Wi-Fi**: Integrating wireless functionality enhances the utility of the applications. Arduino Mega 2560 is compatible with WiFi shields hence allowing the wireless features for the applications in 3D printing and Robotics.

1.2 Accessories

1.3 Related Products

- Arduino® Uno Rev 3
- Arduino® Nano
- Arduino® DUE without headers

2 Ratings

2.1 Recommended Operating Conditions

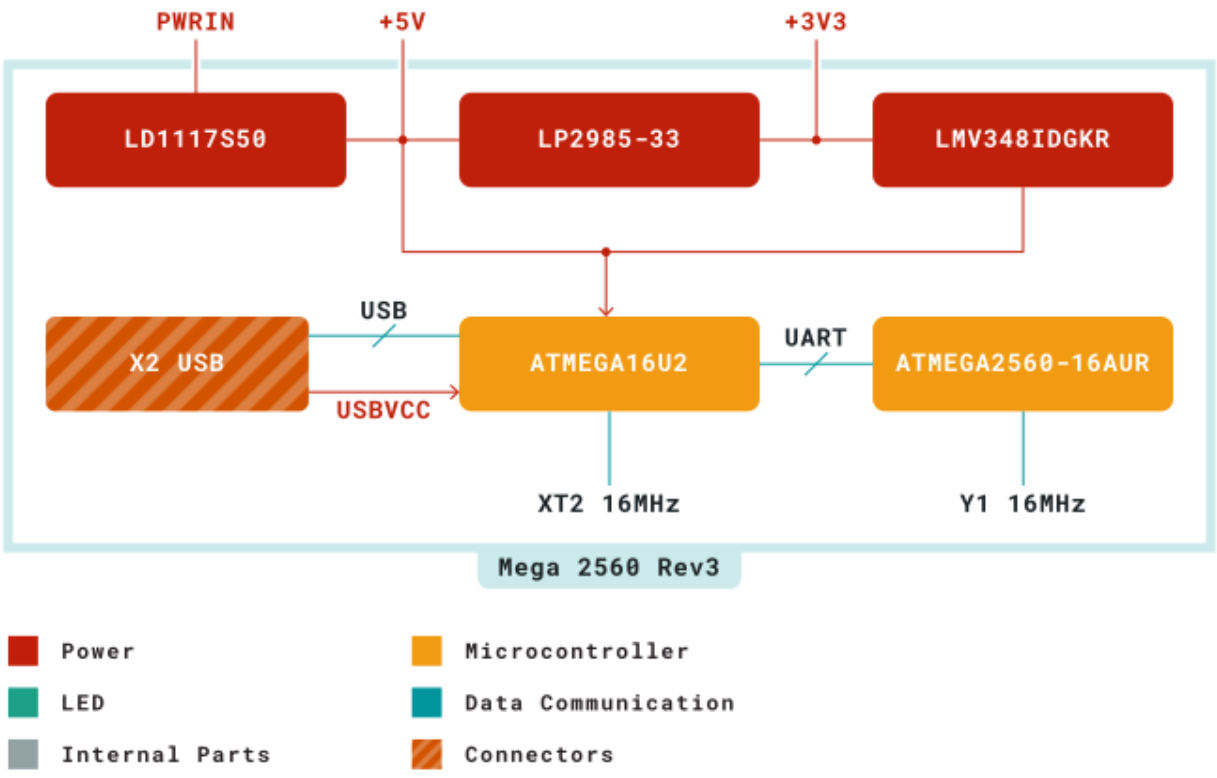
Symbol	Description	Min	Max
TOP	Operating temperature:	-40 °C	85 °C

2.2 Power Consumption

Symbol	Description	Min	Typ	Max	Unit
PWRIN	Input supply from power jack		TBC		mW
USB VCC	Input supply from USB		TBC		mW
VIN	Input from VIN pad		TBC		mW

3 Functional Overview

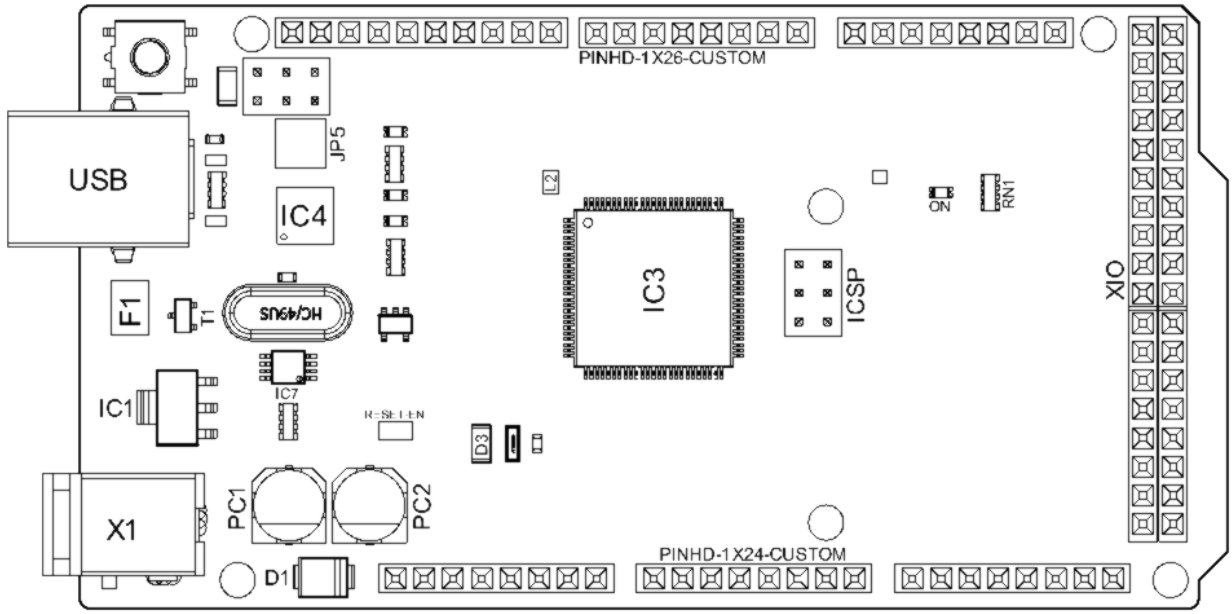
3.1 Block Diagram



Arduino MEGA Block Diagram

3.2 Board Topology

Front View



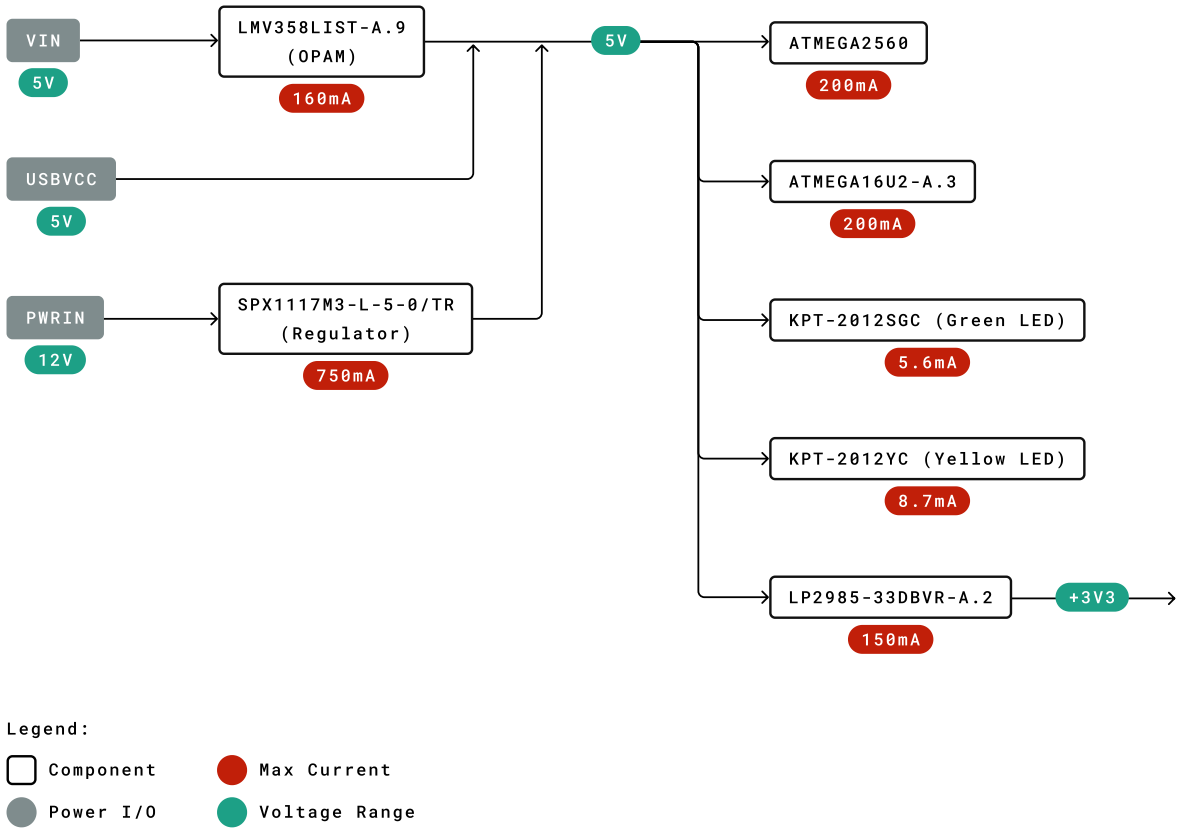
Arduino MEGA Top View

Ref.	Description	Ref.	Description
USB	USB B Connector	F1	Chip Capacitor
IC1	5V Linear Regulator	X1	Power Jack Connector
JP5	Plated Holes	IC4	ATmega16U2 chip
PC1	Electrolytic Alumninum Capacitor	PC2	Electrolytic Alumninum Capacitor
D1	General Purpose Rectifier	D3	General Purpose Diode
L2	Fixed Inductor	IC3	ATmega2560 chip
ICSP	Connector Header	ON	Green LED
RN1	Resistor Array	XIO	Connector

3.3 Processor

Primary processor of Arduino Mega 2560 Rev3 board is ATmega2560 chip which operates at a frequency of 16 MHz. It accommodates a large number of input and output lines which gives the provision of interfacing many external devices. At the same time the operations and processing is not slowed due to its significantly larger RAM than the other processors. The board also features a USB serial processor ATmega16U2 which acts an interface between the USB input signals and the main processor. This increases the flexibility of interfacing and connecting peripherals to the Arduino Mega 2560 Rev 3 board.

3.4 Power Tree



Power Tree

4 Board Operation

4.1 Getting Started - IDE

If you want to program your Arduino® MEGA 2560 while offline you need to install the Arduino® Desktop IDE **[1]** To connect the Arduino® MEGA 2560 to your computer, you'll need a Type-B USB cable. This also provides power to the board, as indicated by the LED.

4.2 Getting Started - Arduino Web Editor

All Arduino® boards, including this one, work out-of-the-box on the Arduino® Web Editor **[2]**, by just installing a simple plugin.

The Arduino® Web Editor is hosted online, therefore it will always be up-to-date with the latest features and support for all boards. Follow **[3]** to start coding on the browser and upload your sketches onto your board.

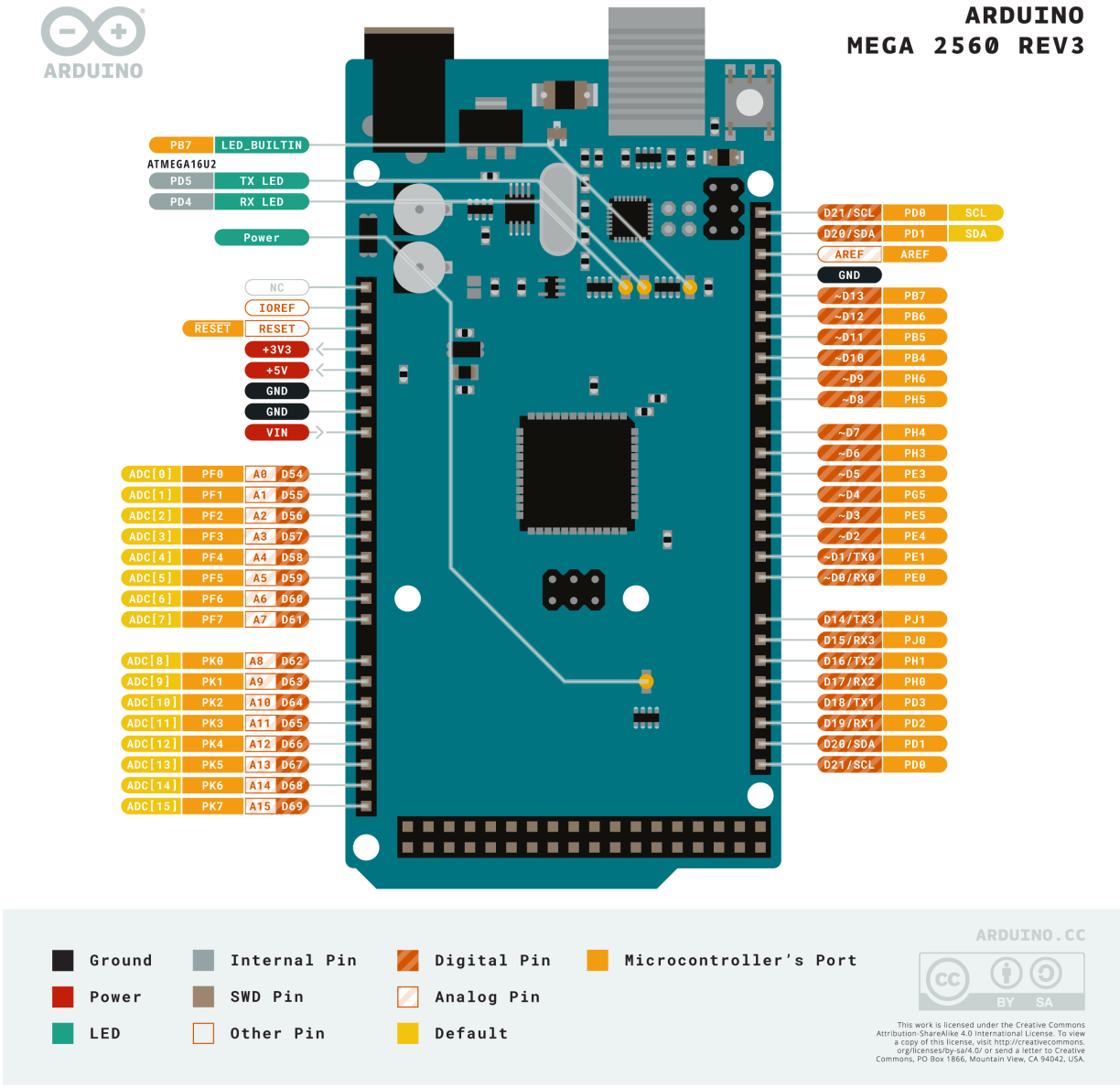
4.3 Sample Sketches

Sample sketches for the Arduino® MEGA 2560 can be found either in the "Examples" menu in the Arduino® IDE

4.4 Online Resources

Now that you have gone through the basics of what you can do with the board you can explore the endless possibilities it provides by checking exciting projects on ProjectHub **[5]**, the Arduino® Library Reference **[6]** and the online store **[7]** where you will be able to complement your board with sensors, actuators and more.

5 Connector Pinouts



Arduino Mega Pinout

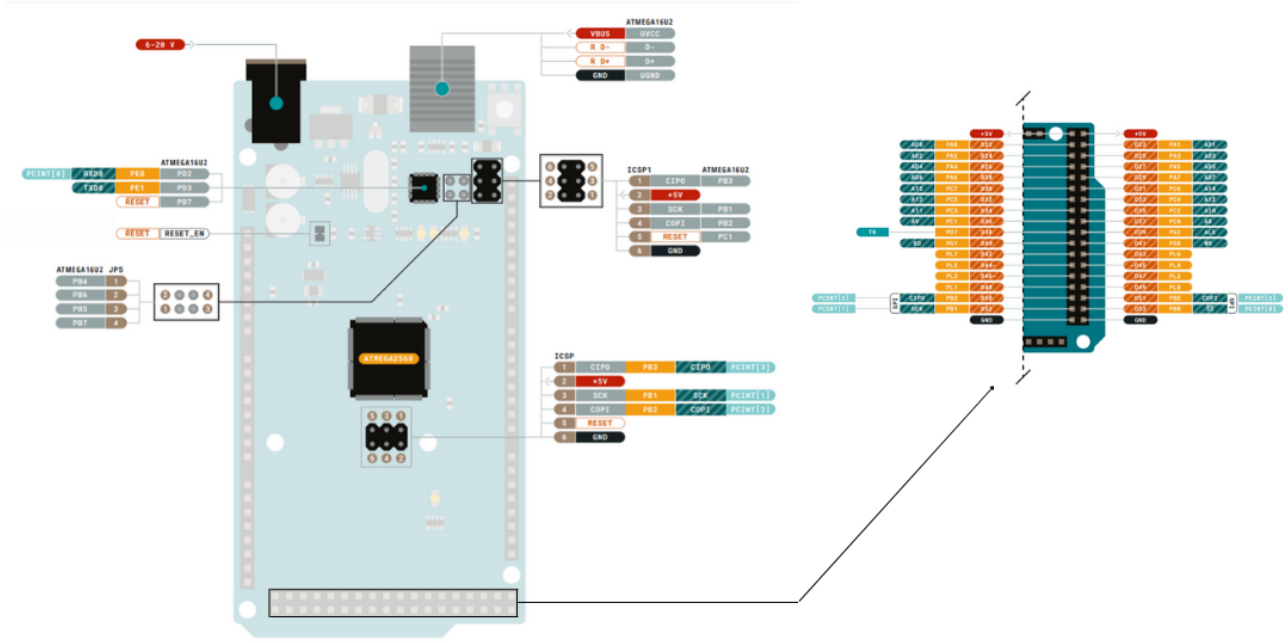
5.1 Analog

Pin	Function	Type	Description
1	NC	NC	Not Connected
2	IOREF	IOREF	Reference for digital logic V - connected to 5V
3	Reset	Reset	Reset
4	+3V3	Power	+3V3 Power Rail
5	+5V	Power	+5V Power Rail
6	GND	Power	Ground
7	GND	Power	Ground
8	VIN	Power	Voltage Input
9	A0	Analog	Analog input 0 /GPIO
10	A1	Analog	Analog input 1 /GPIO
11	A2	Analog	Analog input 2 /GPIO
12	A3	Analog	Analog input 3 /GPIO
13	A4	Analog	Analog input 4 /GPIO
14	A5	Analog	Analog input 5 /GPIO
15	A6	Analog	Analog input 6 /GPIO
16	A7	Analog	Analog input 7 /GPIO
17	A8	Analog	Analog input 8 /GPIO
18	A9	Analog	Analog input 9 /GPIO
19	A10	Analog	Analog input 10 /GPIO
20	A11	Analog	Analog input 11 /GPIO
21	A12	Analog	Analog input 12 /GPIO
22	A13	Analog	Analog input 13 /GPIO
23	A14	Analog	Analog input 14 /GPIO
24	A15	Analog	Analog input 15 /GPIO

5.2 Digital

Pin	Function	Type	Description
1	D21/SCL	Digital Input/I2C	Digital input 21/I2C Dataline
2	D20/SDA	Digital Input/I2C	Digital input 20/I2C Dataline
3	AREF	Digital	Analog Reference Voltage
4	GND	Power	Ground
5	D13	Digital/GPIO	Digital input 13/GPIO
6	D12	Digital/GPIO	Digital input 12/GPIO
7	D11	Digital/GPIO	Digital input 11/GPIO
8	D10	Digital/GPIO	Digital input 10/GPIO
9	D9	Digital/GPIO	Digital input 9/GPIO
10	D8	Digital/GPIO	Digital input 8/GPIO
11	D7	Digital/GPIO	Digital input 7/GPIO
12	D6	Digital/GPIO	Digital input 6/GPIO
13	D5	Digital/GPIO	Digital input 5/GPIO
14	D4	Digital/GPIO	Digital input 4/GPIO

Pin	Function	Type	Description
15	D3	Digital/GPIO	Digital input 3/GPIO
16	D2	Digital/GPIO	Digital input 2/GPIO
17	D1/TX0	Digital/GPIO	Digital input 1 /GPIO
18	D0/Tx1	Digital/GPIO	Digital input 0 /GPIO
19	D14	Digital/GPIO	Digital input 14 /GPIO
20	D15	Digital/GPIO	Digital input 15 /GPIO
21	D16	Digital/GPIO	Digital input 16 /GPIO
22	D17	Digital/GPIO	Digital input 17 /GPIO
23	D18	Digital/GPIO	Digital input 18 /GPIO
24	D19	Digital/GPIO	Digital input 19 /GPIO
25	D20	Digital/GPIO	Digital input 20 /GPIO
26	D21	Digital/GPIO	Digital input 21 /GPIO



Arduino Mega Pinout

5.3 ATMEGA16U2 JP5

Pin	Function	Type	Description
1	PB4	Internal	Serial Wire Debug
2	PB6	Internal	Serial Wire Debug
3	PB5	Internal	Serial Wire Debug
4	PB7	Internal	Serial Wire Debug

5.4 ATMEGA16U2 ICSP1

Pin	Function	Type	Description
1	CIPO	Internal	Controller In Peripheral Out
2	+5V	Internal	Power Supply of 5V
3	SCK	Internal	Serial Clock
4	COPI	Internal	Controller Out Peripheral In
5	RESET	Internal	Reset
6	GND	Internal	Ground

5.5 Digital Pins D22 - D53 LHS

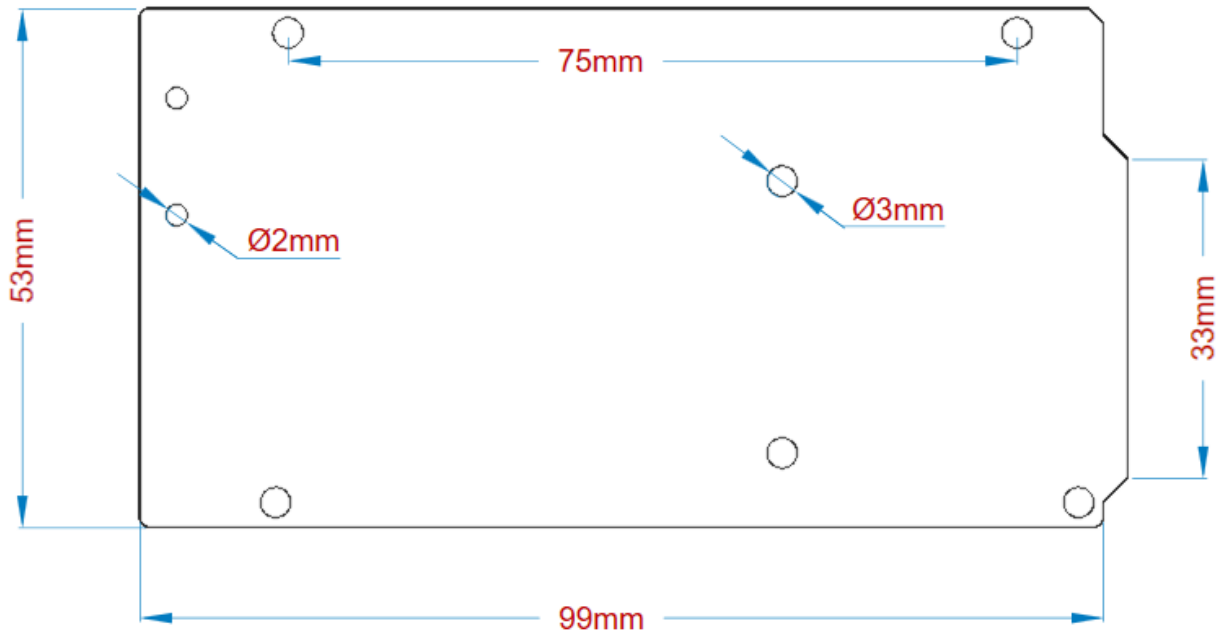
Pin	Function	Type	Description
1	+5V	Power	Power Supply of 5V
2	D22	Digital	Digital input 22/GPIO
3	D24	Digital	Digital input 24/GPIO
4	D26	Digital	Digital input 26/GPIO
5	D28	Digital	Digital input 28/GPIO
6	D30	Digital	Digital input 30/GPIO
7	D32	Digital	Digital input 32/GPIO
8	D34	Digital	Digital input 34/GPIO
9	D36	Digital	Digital input 36/GPIO
10	D38	Digital	Digital input 38/GPIO
11	D40	Digital	Digital input 40/GPIO
12	D42	Digital	Digital input 42/GPIO
13	D44	Digital	Digital input 44/GPIO
14	D46	Digital	Digital input 46/GPIO
15	D48	Digital	Digital input 48/GPIO
16	D50	Digital	Digital input 50/GPIO
17	D52	Digital	Digital input 52/GPIO
18	GND	Power	Ground

5.6 Digital Pins D22 - D53 RHS

Pin	Function	Type	Description
1	+5V	Power	Power Supply of 5V
2	D23	Digital	Digital input 23/GPIO
3	D25	Digital	Digital input 25/GPIO
4	D27	Digital	Digital input 27/GPIO
5	D29	Digital	Digital input 29/GPIO
6	D31	Digital	Digital input 31/GPIO
7	D33	Digital	Digital input 33/GPIO
8	D35	Digital	Digital input 35/GPIO
9	D37	Digital	Digital input 37/GPIO
10	D39	Digital	Digital input 39/GPIO
11	D41	Digital	Digital input 41/GPIO
12	D43	Digital	Digital input 43/GPIO
13	D45	Digital	Digital input 45/GPIO
14	D47	Digital	Digital input 47/GPIO
15	D49	Digital	Digital input 49/GPIO
16	D51	Digital	Digital input 51/GPIO
17	D53	Digital	Digital input 53/GPIO
18	GND	Power	Ground

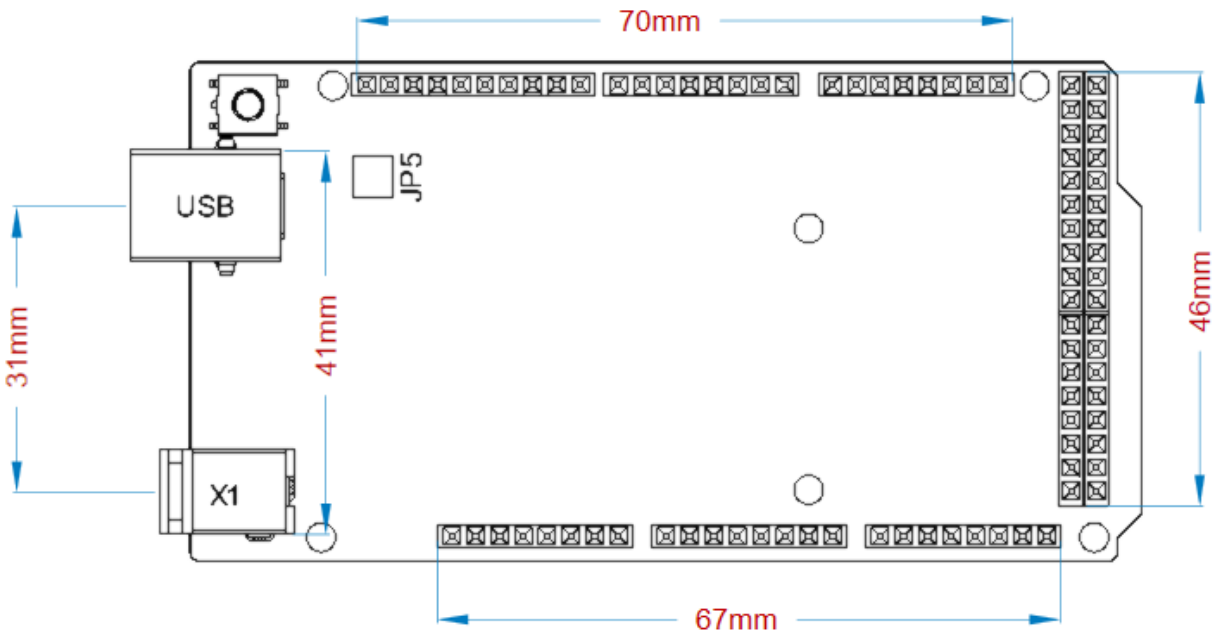
6 Mechanical Information

6.1 Board Outline



Arduino Mega Outline

6.2 Board Mount Holes



Arduino Mega Mount Holes

Certifications

7 Declaration of Conformity CE DoC (EU)

We declare under our sole responsibility that the products above are in conformity with the essential requirements of the following EU Directives and therefore qualify for free movement within markets comprising the European Union (EU) and European Economic Area (EEA).

8 Declaration of Conformity to EU RoHS & REACH 211 01/19/2021

Arduino boards are in compliance with RoHS 2 Directive 2011/65/EU of the European Parliament and RoHS 3 Directive 2015/863/EU of the Council of 4 June 2015 on the restriction of the use of certain hazardous substances in electrical and electronic equipment.

Substance	Maximum Limit (ppm)
Lead (Pb)	1000
Cadmium (Cd)	100
Mercury (Hg)	1000
Hexavalent Chromium (Cr6+)	1000
Poly Brominated Biphenyls (PBB)	1000
Poly Brominated Diphenyl ethers (PBDE)	1000
Bis(2-Ethylhexyl} phthalate (DEHP)	1000
Benzyl butyl phthalate (BBP)	1000
Dibutyl phthalate (DBP)	1000
Diisobutyl phthalate (DIBP)	1000

Exemptions : No exemptions are claimed.

Arduino Boards are fully compliant with the related requirements of European Union Regulation (EC) 1907 /2006 concerning the Registration, Evaluation, Authorization and Restriction of Chemicals (REACH). We declare none of the SVHCs (https://echa.europa.eu/web/guest/candidate-list-table), the Candidate List of Substances of Very High Concern for authorization currently released by ECHA, is present in all products (and also package) in quantities totaling in a concentration equal or above 0.1%. To the best of our knowledge, we also declare that our products do not contain any of the substances listed on the "Authorization List" (Annex XIV of the REACH regulations) and Substances of Very High Concern (SVHC) in any significant amounts as specified by the Annex XVII of Candidate list published by ECHA (European Chemical Agency) 1907 /2006/EC.

9 Conflict Minerals Declaration

As a global supplier of electronic and electrical components, Arduino is aware of our obligations with regards to laws and regulations regarding Conflict Minerals, specifically the Dodd-Frank Wall Street Reform and Consumer Protection Act, Section 1502. Arduino does not directly source or process conflict minerals such as Tin, Tantalum, Tungsten, or Gold. Conflict minerals are contained in our products in the form of solder, or as a component in metal alloys. As part of our reasonable due diligence Arduino has contacted component suppliers within our supply chain to verify their continued compliance with the regulations. Based on the information received thus far we declare that our products contain Conflict Minerals sourced from conflict-free areas.

10 FCC Caution

Any Changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate the equipment.

This device complies with part 15 of the FCC Rules. Operation is subject to the following two conditions:

(1) This device may not cause harmful interference

(2) this device must accept any interference received, including interference that may cause undesired operation.

FCC RF Radiation Exposure Statement:

1. This Transmitter must not be co-located or operating in conjunction with any other antenna or transmitter.
2. This equipment complies with RF radiation exposure limits set forth for an uncontrolled environment.
3. This equipment should be installed and operated with minimum distance 20cm between the radiator & your body.

English: User manuals for licence-exempt radio apparatus shall contain the following or equivalent notice in a conspicuous location in the user manual or alternatively on the device or both. This device complies with Industry Canada licence-exempt RSS standard(s). Operation is subject to the following two conditions:

(1) this device may not cause interference

(2) this device must accept any interference, including interference that may cause undesired operation of the device.

French: Le présent appareil est conforme aux CNR d'Industrie Canada applicables aux appareils radio exempts de licence. L'exploitation est autorisée aux deux conditions suivantes :

(1) l' appareil nedoit pas produire de brouillage

(2) l'utilisateur de l'appareil doit accepter tout brouillage radioélectrique subi, même si le brouillage est susceptible d'en compromettre le fonctionnement.

IC SAR Warning:

English This equipment should be installed and operated with minimum distance 20 cm between the radiator and your body.

French: Lors de l' installation et de l' exploitation de ce dispositif, la distance entre le radiateur et le corps est d 'au moins 20 cm.

Important: The operating temperature of the EUT can't exceed 85°C and shouldn't be lower than -40°C.

Hereby, Arduino S.r.l. declares that this product is in compliance with essential requirements and other relevant provisions of Directive 201453/EU. This product is allowed to be used in all EU member states.

11 Company Information

Company name	Arduino S.r.l.
Company Address	Arduino SRL, Via Andrea Appiani 25, 20900 Monza MB, Italy

12 Reference Documentation

Ref	Link
Arduino IDE (Desktop)	https://www.arduino.cc/en/Main/Software
Arduino IDE (Cloud)	https://create.arduino.cc/editor
Cloud IDE Getting Started	https://create.arduino.cc/projecthub/Arduino_Genuino/getting-started-with-arduino-web-editor-4b3e4a
Arduino Pro Website	https://www.arduino.cc/pro
Project Hub	https://create.arduino.cc/projecthub?by=part&part_id=11332&sort=trending
Library Reference	https://www.arduino.cc/reference/en/libraries/
Online Store	https://store.arduino.cc/

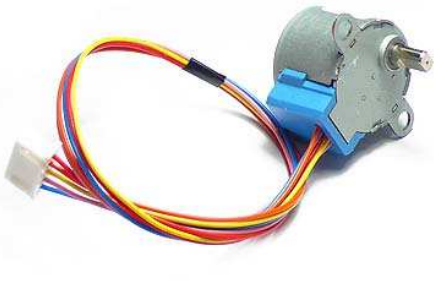
13 Revision History

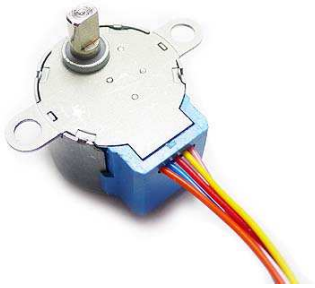
Date	Revision	Changes
29/09/2020	1	First Release

B. Hoja de datos del motor paso a paso 28BYJ-48 con driver ULN2003 (Datasheet, 2023)

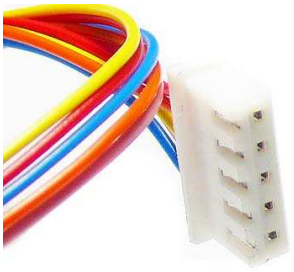
28BYJ-48 – 5V Stepper Motor

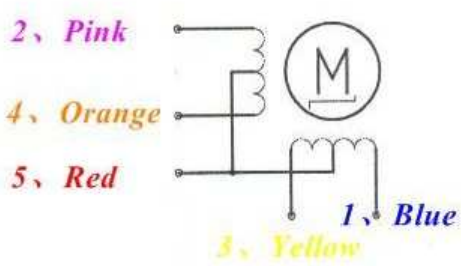
The 28BYJ-48 is a small stepper motor suitable for a large range of applications.

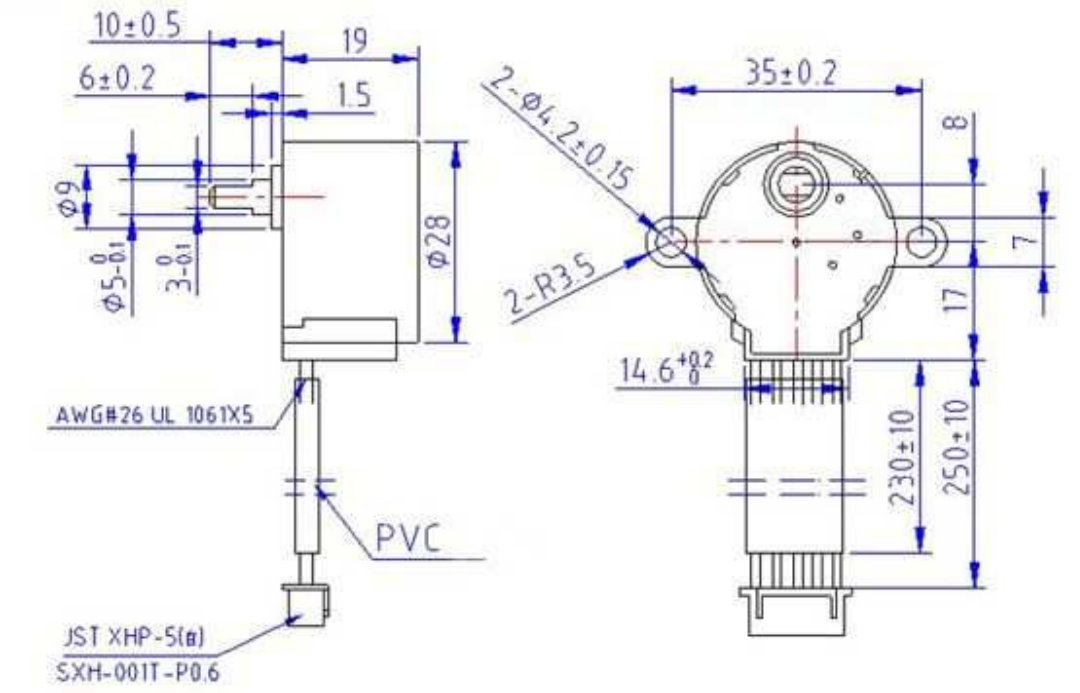




Rated voltage ：	5VDC
Number of Phase	4
Speed Variation Ratio	1/64
Stride Angle	5.625°/64
Frequency	100Hz
DC resistance	50Ω±7%(25℃)
Idle In-traction Frequency	> 600Hz
Idle Out-traction Frequency	> 1000Hz
In-traction Torque	>34.3mN.m(120Hz)
Self-positioning Torque	>34.3mN.m
Friction torque	600-1200 gf.cm
Pull in torque	300 gf.cm
Insulated resistance	>10MΩ(500V)
Insulated electricity power	600VAC/1mA/1s
Insulation grade	A
Rise in Temperature	<40K(120Hz)
Noise	<35dB(120Hz,No load,10cm)
Model	28BYJ-48 – 5V







C. Hoja de datos detecto de obstáculos con sensor infrarrojo FC51 (Az-Delivery, Módulo infrarrojo IR, 2023)

AZ-Delivery
Ihr Experte für Mikroelektronik!

IR-Abstand Sensor Module

Contents:

1. General Description

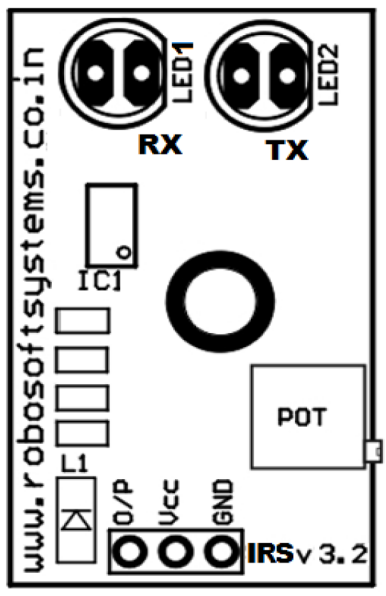
The IR Sensor-Single is a general purpose proximity sensor. Here we use it for collision detection. The module consist of a IR emitter and IR receiver pair. The high precision IR receiver always detects a IR signal

The module consists of 358 comparator IC. The output of sensor is high whenever it IR frequency and low otherwise. The on-board LED indicator helps user to check status of the sensor without using any additional hardware.

The power consumption of this module is low. It gives a digital output.
Application

2. Pin Configuration

The figure to the right is a top view of the IR Sensor module. The following table gives its pin description.

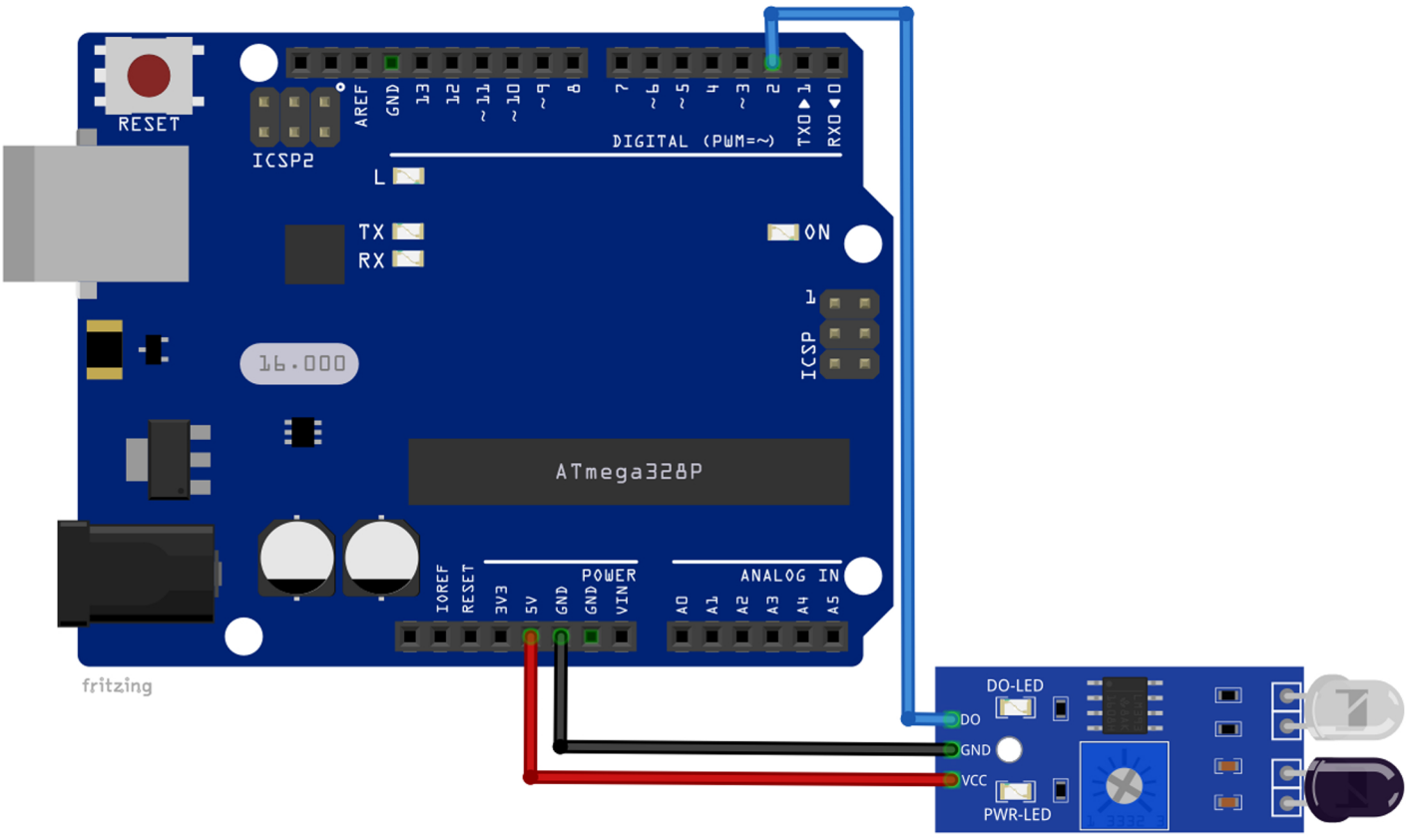


Pin No.	Connection	Description
1	Output	Digital Output (High or Low)
2	VCC	Connected to circuit supply
3	Ground	Connected to circuit ground

3. Application Ideas

- Obstacle detection
- Shaft encoder
- Fixed frequency detection

4. Schematic Diagrams



5. Overview of Schematic

The sensitivity of the IR Sensor is tuned using the potentiometer. The potentiometer is tuneable in both the directions. Initially tune the potentiometer in clockwise direction such that the Indicator LED starts glowing. Once that is achieved, turn the potentiometer just enough in anti-clockwise direction to turn off the Indicator LED. At this point the sensitivity of the receiver is maximum. Thus, its sensing distance is maximum at this point. If the sensing distance (i.e., Sensitivity) of the receiver is needed to be reduced, then one can tune the potentiometer in the anti-clockwise direction from this point.

Further, if the orientation of both Tx and Rx LED's is parallel to each other, such that both are facing outwards, then their sensitivity is maximum. If they are moved away from each other, such that they are inclined to each other at their soldered end, then their sensitivity reduces.

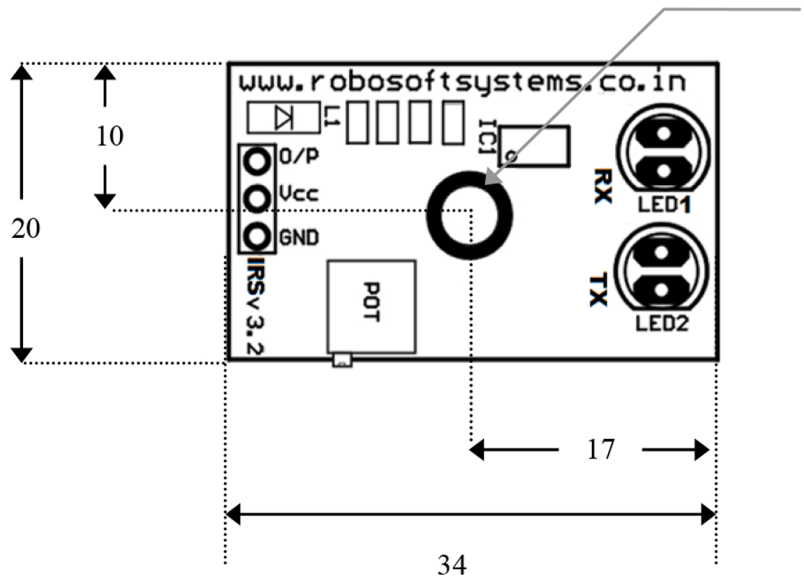
Tuned sensitivity of the sensors is limited to the surroundings. Once tuned for a particular surrounding, they will work perfectly until the IR illumination conditions of that region nearly constant. For example, if the potentiometer is tuned inside room/building for maximum sensitivity and then taken out in open sunlight, its will require retuning, since sun's rays also contain Infrared (IR) frequencies, thus acting as a IR source (transmitter). This will disturb the receiver's sensing capacity. Hence it needs to be retuned to work perfectly in the new surroundings.

The output of IR receiver goes low when it receives IR signal. Hence the output pin is normally low because, though the IR LED is continuously transmitting, due to no obstacle, nothing is reflected back to the IR receiver. The indication LED is off. When an obstacle is encountered, the output of IR receiver goes low, IR signal is reflected from the obstacle surface. This drives the output of the comparator low. This output is connected to the cathode of the LED, which then turns ON.

6. Maximum Ratings

Symbol	Quantity	Minimum	Typical	Maximum	Unit
o/p	Output Voltage	0	-	5	V
V_{CC}	Operating Voltage	3.3	5	5.5	V
GND	Ground Reference voltage	-	0	-	V

7. Pin Out Dimensions



Note : All dimension in mm
Error of ±5% is subjected because of component soldering

D. Hoja de datos OLED 1.3’ SSH1106 (Robotlandia, 2023)

VMA437

1.3 INCH OLED SCREEN FOR ARDUINO® (SH1106 DRIVER, SPI)

USER MANUAL

USER MANUAL

1. Introduction

To all residents of the European Union

Important environmental information about this product

This symbol on the device or the package indicates that disposal of the device after its lifecycle could harm the environment. Do not dispose of the unit (or batteries) as unsorted municipal waste; it should be taken to a specialized company for recycling. This device should be returned to your distributor or to a local recycling service. Respect the local environmental rules.

If in doubt, contact your local waste disposal authorities.

Thank you for choosing Velleman®! Please read the manual thoroughly before bringing this device into service. If the device was damaged in transit, do not install or use it and contact your dealer.

2. Safety Instructions

- This device can be used by children aged from 8 years and above, and persons with reduced physical, sensory or mental capabilities or lack of experience and knowledge if they have been given supervision or instruction concerning the use of the device in a safe way and understand the hazards involved. Children shall not play with the device. Cleaning and user maintenance shall not be made by children without supervision.

- Indoor use only.
  Keep away from rain, moisture, splashing and dripping liquids.

3. General Guidelines

- Refer to the Velleman® Service and Quality Warranty on the last pages of this manual.
- Familiarise yourself with the functions of the device before actually using it.
- All modifications of the device are forbidden for safety reasons. Damage caused by user modifications to the device is not covered by the warranty.
- Only use the device for its intended purpose. Using the device in an unauthorised way will void the warranty.
- Damage caused by disregard of certain guidelines in this manual is not covered by the warranty and the dealer will not accept responsibility for any ensuing defects or problems.
- Nor Velleman nv nor its dealers can be held responsible for any damage (extraordinary, incidental or indirect) – of any nature (financial, physical…) arising from the possession, use or failure of this product.
- Due to constant product improvements, the actual product appearance might differ from the shown images.
- Product images are for illustrative purposes only.
- Do not switch the device on immediately after it has been exposed to changes in temperature. Protect the device against damage by leaving it switched off until it has reached room temperature.
- Keep this manual for future reference.

4. What is Arduino®

Arduino® is an open-source prototyping platform based in easy-to-use hardware and software. Arduino® boards are able to read inputs – light-on sensor, a finger on a button or a Twitter message – and turn it into an output – activating of a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so, you use the Arduino programming language (based on Wiring) and the Arduino® software IDE (based on Processing).

Surf to www.arduino.cc and www.arduino.org for more information.

5. Overview

VMA437

OLED displays are great in many ways. They use very little power, are bright, easy to read with a large viewing angle and have high resolution considering their small size.

resolution: 128 x 64 dots
viewing angle: > 160°
working voltage: 3 to 5 V
recommended library: U8glib
interface: SPI
driver: SSH1106
working temperature: -30 °C to 70 °C
OLED colour: blue
dimensions: 35 x 33.5 mm

6. Pin Layout

VDD	2.8-5.5 V power supply
SCK	CLK clock
SDA	MOSI data
RES	reset
DC	data/command
CS	chip-select signal
GND	ground

7. Example

Connect the VMA437 to your Arduino® Uno-compatible board (VMA100) this way:

Connection.
Vcc======5V
Gnd======Gnd
D0 (SCL)======D4
RES======Reset
D1 (SCK)======D5
CS======D6
DC======D7

Go to the product page on www.velleman.eu and download the U8glib.zip file.
Start the Arduino® IDE and import this library: Sketch → Include Library → Add Zip library.
Once finished, go back to Sketch → Include Library → Manage library's, and scroll down until you find the U8glib library. Select this library and tap "Update". Now you have the latest version with examples.

Go to Files → Examples and scroll down to U8glib. Open the example Graphicstest.

In the sketch "Graphicstest", several types of displays can be selected. Just "un-comment" the one you need.
For the VMA437 you have to un-comment:
U8GLIB_SH1106_128X64 u8g(4, 5, 6, 7); // SW SPI Com: SCK = 4, MOSI = 5, CS = 6, A0 = 7 (new blue HalTec OLED).

Compile and upload the sketch to your VMA100 and enjoy!

The "Graphicstest" sketch with only the correct driver line for VMA437 looks like this:

```
/*

  GraphicsTest.pde

  >>> Before compiling: Please remove comment from the constructor of the
  >>> connected graphics display (see below).

  Universal 8bit Graphics Library, https://github.com/olikraus/u8glib/

  Copyright (c) 2012, olikraus@gmail.com
  All rights reserved.

  Redistribution and use in source and binary forms, with or without modification,
  are permitted provided that the following conditions are met:

  * Redistributions of source code must retain the above copyright notice, this list
    of conditions and the following disclaimer.

  * Redistributions in binary form must reproduce the above copyright notice, this
    list of conditions and the following disclaimer in the documentation and/or other
    materials provided with the distribution.

  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND
  CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
  INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
  MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
  DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR
  CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
  SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
  NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
  LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
  CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
  ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
  ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

*/

#include "U8glib.h"

// setup u8g object, please remove comment from one of the following constructor calls
// IMPORTANT NOTE: The following list is incomplete. The complete list of supported
// devices with all constructor calls is here: https://github.com/olikraus/u8glib/wiki/device

U8GLIB_SSD1306_128X64 u8g(U8G_I2C_OPT_NO_ACK);        // Display which does not send AC  VMA437 - Velleman , UN-comment this line as it is now
```

```
void u8g_prepare(void) {
  u8g.setFont(u8g_font_6x10);
  u8g.setFontRefHeightExtendedText();
  u8g.setDefaultForegroundColor();
  u8g.setFontPosTop();
}

void u8g_box_frame(uint8_t a) {
  u8g.drawStr( 0, 0, "drawBox");
  u8g.drawBox(5,10,20,10);
  u8g.drawBox(10+a,15,30,7);
  u8g.drawStr( 0, 30, "drawFrame");
  u8g.drawFrame(5,10+30,20,10);
  u8g.drawFrame(10+a,15+30,30,7);
}

void u8g_disc_circle(uint8_t a) {
  u8g.drawStr( 0, 0, "drawDisc");
  u8g.drawDisc(10,18,9);
  u8g.drawDisc(24+a,16,7);
  u8g.drawStr( 0, 30, "drawCircle");
  u8g.drawCircle(10,18+30,9);
  u8g.drawCircle(24+a,16+30,7);
}

void u8g_r_frame(uint8_t a) {
  u8g.drawStr( 0, 0, "drawRFrame/Box");
  u8g.drawRFrame(5, 10,40,30, a+1);
  u8g.drawRBox(50, 10,25,40, a+1);
}

void u8g_string(uint8_t a) {
  u8g.drawStr(30+a,31, " 0");
  u8g.drawStr90(30,31+a, " 90");
  u8g.drawStr180(30-a,31, " 180");
  u8g.drawStr270(30,31-a, " 270");
}

void u8g_line(uint8_t a) {
  u8g.drawStr( 0, 0, "drawLine");
  u8g.drawLine(7+a, 10, 40, 55);
  u8g.drawLine(7+a*2, 10, 60, 55);
  u8g.drawLine(7+a*3, 10, 80, 55);
  u8g.drawLine(7+a*4, 10, 100, 55);
}

void u8g_triangle(uint8_t a) {
  uint16_t offset = a;
  u8g.drawStr( 0, 0, "drawTriangle");
  u8g.drawTriangle(14,7, 45,30, 10,40);
  u8g.drawTriangle(14+offset,7-offset, 45+offset,30-offset, 57+offset,10-offset);
  u8g.drawTriangle(57+offset*2,10, 45+offset*2,30, 86+offset*2,53);
  u8g.drawTriangle(10+offset,40+offset, 45+offset,30+offset, 86+offset,53+offset);
}
```

```
void u8g_ascii_1() {
  char s[2] = " ";
  uint8_t x, y;
  u8g.drawStr( 0, 0, "ASCII page 1");
  for( y = 0; y < 6; y++ ) {
    for( x = 0; x < 16; x++ ) {
      s[0] = y*16 + x + 32;
      u8g.drawStr(x*7, y*10+10, s);
    }
  }
}

void u8g_ascii_2() {
  char s[2] = " ";
  uint8_t x, y;
  u8g.drawStr( 0, 0, "ASCII page 2");
  for( y = 0; y < 6; y++ ) {
    for( x = 0; x < 16; x++ ) {
      s[0] = y*16 + x + 160;
      u8g.drawStr(x*7, y*10+10, s);
    }
  }
}

void u8g_extra_page(uint8_t a)
{
  if ( u8g.getMode() == U8G_MODE_HICOLOR || u8g.getMode() == U8G_MODE_R3G3B2) {
    /* draw background (area is 128x128) */
    u8g_uint_t r, g, b;
    b = a << 5;
    for( g = 0; g < 64; g++ )
    {
      for( r = 0; r < 64; r++ )
      {
          u8g.setRGB(r<<2, g<<2, b );
          u8g.drawPixel(g, r);
      }
    }
    u8g.setRGB(255,255,255);
    u8g.drawStr( 66, 0, "Color Page");
  }
  else if ( u8g.getMode() == U8G_MODE_GRAY2BIT )
  {
    u8g.drawStr( 66, 0, "Gray Level");
    u8g.setColorIndex(1);
    u8g.drawBox(0, 4, 64, 32);
    u8g.drawBox(70, 20, 4, 12);
    u8g.setColorIndex(2);
    u8g.drawBox(0+1*a, 4+1*a, 64-2*a, 32-2*a);
    u8g.drawBox(74, 20, 4, 12);
    u8g.setColorIndex(3);
    u8g.drawBox(0+2*a, 4+2*a, 64-4*a, 32-4*a);
    u8g.drawBox(78, 20, 4, 12);
  }
```

```
  else
  {
    u8g.drawStr( 0, 12, "setScale2x2");
    u8g.setScale2x2();
    u8g.drawStr( 0, 6+a, "setScale2x2");
    u8g.undoScale();
  }
}

uint8_t draw_state = 0;

void draw(void) {
  u8g_prepare();
  switch(draw_state >> 3) {
    case 0: u8g_box_frame(draw_state&7); break;
    case 1: u8g_disc_circle(draw_state&7); break;
    case 2: u8g_r_frame(draw_state&7); break;
    case 3: u8g_string(draw_state&7); break;
    case 4: u8g_line(draw_state&7); break;
    case 5: u8g_triangle(draw_state&7); break;
    case 6: u8g_ascii_1(); break;
    case 7: u8g_ascii_2(); break;
    case 8: u8g_extra_page(draw_state&7); break;
  }
}

void setup(void) {

  // flip screen, if required
  //u8g.setRot180();

#if defined(ARDUINO)
  pinMode(13, OUTPUT);
  digitalWrite(13, HIGH);
#endif
}

void loop(void) {
  // picture loop
  u8g.firstPage();
  do {
    draw();
  } while( u8g.nextPage() );

  // increase the state
  draw_state++;
  if ( draw_state >= 9*8 )
    draw_state = 0;

  // rebuild the picture after some delay
  //delay(150);

}
```

8. More Information

Please refer to the VMA437 product page on www.velleman.eu for more information.

Use this device with original accessories only. Velleman nv cannot be held responsible in the event of damage or injury resulting from (incorrect) use of this device. For more info concerning this product and the latest version of this manual, please visit our website www.velleman.eu. The information in this manual is subject to change without prior notice.

Velleman® Service and Quality Warranty

Since its foundation in 1972, Velleman® acquired extensive experience in the electronics world and currently distributes its products in over 85 countries.

All our products fulfil strict quality requirements and legal stipulations in the EU. In order to ensure the quality, our products regularly go through an extra quality check, both by an internal quality department and by specialized external organisations. If, all precautionary measures notwithstanding, problems should occur, please make appeal to our warranty (see guarantee conditions).

General Warranty Conditions Concerning Consumer Products (for EU):

• All consumer products are subject to a 24-month warranty on production flaws and defective material as from the original date of purchase.

• Velleman® can decide to replace an article with an equivalent article, or to refund the retail value totally or partially when the complaint is valid and a free repair or replacement of the article is impossible, or if the expenses are out of proportion.

You will be delivered a replacing article or a refund at the value of 100% of the purchase price in case of a flaw occurred in the first year after the date of purchase and delivery, or a replacing article at 50% of the purchase price or a refund at the value of 50% of the retail value in case of a flaw occurred in the second year after the date of purchase and delivery.

• Not covered by warranty:

- all direct or indirect damage caused after delivery to the article (e.g. by oxidation, shocks, falls, dust, dirt, humidity...), and by the article, as well as its contents (e.g. data loss), compensation for loss of profits;

- consumable goods, parts or accessories that are subject to an aging process during normal use, such as batteries (rechargeable, non-rechargeable, built-in or replaceable), lamps, rubber parts, drive belts... (unlimited list);

- flaws resulting from fire, water damage, lightning, accident, natural disaster, etc.…;

- flaws caused deliberately, negligently or resulting from improper handling, negligent maintenance, abusive use or use contrary to the manufacturer's instructions;

- damage caused by a commercial, professional or collective use of the article (the warranty validity will be reduced to six (6) months when the article is used professionally);

- damage resulting from an inappropriate packing and shipping of the article;

- all damage caused by modification, repair or alteration performed by a third party without written permission by Velleman®.

• Articles to be repaired must be delivered to your Velleman® dealer, solidly packed (preferably in the original packaging), and be completed with the original receipt of purchase and a clear flaw description.

• Hint: In order to save on cost and time, please reread the manual and check if the flaw is caused by obvious causes prior to presenting the article for repair. Note that returning a non-defective article can also involve handling costs.

• Repairs occurring after warranty expiration are subject to shipping costs.

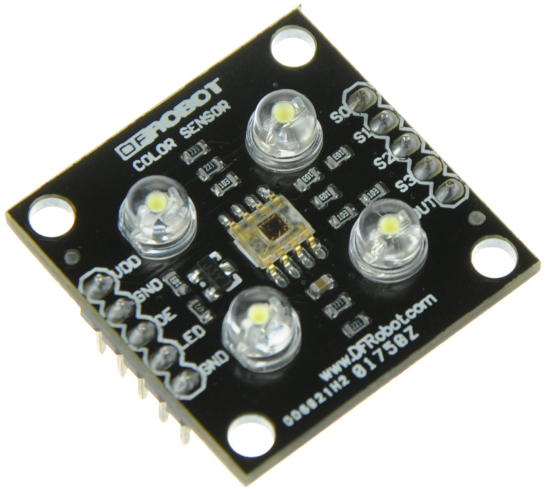
• The above conditions are without prejudice to all commercial warranties.

The above enumeration is subject to modification according to the article (see article's manual).

Made in PRC
Imported by Velleman nv
Legen Heirweg 33, 9890 Gavere, Belgium
www.velleman.eu

E. Hoja de datos sensor de colores TCS3200 (RS, 2023)

SKU:SEN0101 (https://www.dfrobot.com/product-540.html)



(https://www.dfrobot.com/product-540.html)

Introduction

TCS3200 GBB Color Sensor For Arduino (https://www.dfrobot.com/product-540.html) is a complete color detector, including a TAOS TCS3200 RGB sensor chip and 4 white LEDs. The TCS3200 can detect and measure a nearly limitless range of visible colors. Applications include test strip reading, sorting by color, ambient light sensing and calibration, and color matching, to name just a few.

The TCS3200 GBB Color Sensor For Arduino has an array of photo detectors, each with either a red, green, or blue filter, or no filter (clear). The filters of each color are distributed evenly throughout the array to eliminate location bias among the colors. Internal to the device is an oscillator which produces a square-wave output whose frequency is proportional to the intensity of the chosen color.

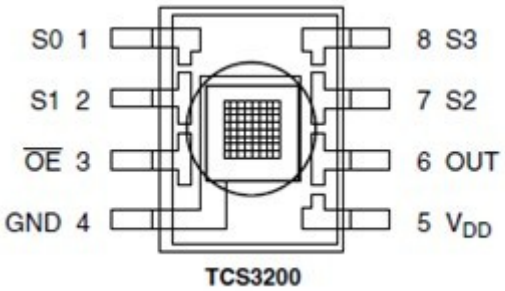
Specifications

-

- Single-Supply Operation (2.7V to 5.5V)
- High-Resolution Conversion of Light Intensity to Frequency

- Programmable Color and Full-Scale Output Frequency
- Power Down Feature
- Communicates Directly to Microcontroller
- S0~S1: Output frequency scaling selection inputs
- S2~S3: Photodiode type selection inputs
- OUT Pin: Output frequency
- OE Pin: Output frequency enable pin (active low), can be impending when using
- Support LED lamp light supplement control
- Size: 28.4x28.4mm

PinOut

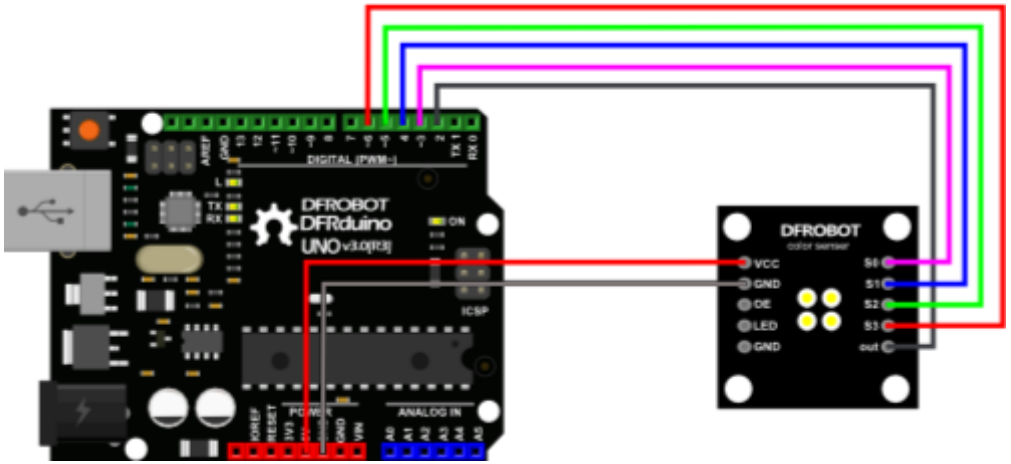


Pin Name	I/O	DESCRIPTION
GND(4)	GND	Power supply ground. All voltages are referenced to GND
OE(3)	I	Enable for fo (active low)

Pin Name	I/O	DESCRIPTION
OUT (3)	O	Output frequency (of)
S0, S1 (1, 2)	I	Output frequency scaling selection inputs
S2, S3 (7, 8)	I	Photodiode type selection inputs
VDD (5)	VDD	Supply voltage

Tutorial

Connection Diagram



Wiring instructions
VCC——5V
S0——D3
S2——D5

S2——D5
Wiring instructions
OUT——D2

S0, S1, S2, S3

To TCS3002D, when choose a color filter, it can allow only one particular color to get through and prevent other color. For example, when choose the red filter, Only red incident light can get through, blue and green will be prevented. So we can get the red light intensity. Similarly ,when choose other filters we can get blue or green light.

TCS3002D has four photodiode types. Red , blue, green and clear, reducing the amplitude of the incident light uniformity greatly, so that to increase the accuracy and simplify the optical. When the light project to the TCS3002D we can choose the different type of photodiode by different combinations of S2 and S3. Look at the form as follows.

S0	S1	OUTPUT FREQUENCY SCALING (fo)
L	L	Power down
L	H	2%
H	L	20%
H	H	100%

TCS3002D can output the frequency of different square wave (occupies emptiescompared 50%),different color and light intensity correspond with different frequency of square wave. There is a relationship between the output and light intensity. The range of the typical output frequency is 2HZ~500KHZ. We can get different scaling factor by different combinations of S0 and S1. Look at the form as follows.

S2	S3	PHOTODIODE TYPE
L	L	RED
L	H	BLUE

S2	S3	PHOTODIODE TYPE
H	L	Clear (no filter)
H	H	GREEN

Sample Code