Integración de una *suite* de sensores para potenciar la autonomía del manipulador robótico MaxArm dentro del ecosistema Robotat

Melanie Fernanda Morales Díaz

UVG UNIVERSIDAD DEL VALLE DE GUATEMALA

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Integración de una *suite* de sensores para potenciar la autonomía del manipulador robótico MaxArm dentro del ecosistema Robotat

Trabajo de graduación presentado por Melanie Fernanda Morales Díaz
para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

2025

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Integración de una *suite* de sensores para potenciar la autonomía del manipulador robótico MaxArm dentro del ecosistema Robotat

Trabajo de graduación presentado por Melanie Fernanda Morales Díaz
para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

2025

Vo.Bo.:

(f) 

M. Sc. Carlos Esquit

Tribunal Examinador:

(f) 

M.Sc. Carlos Esquit

(f) 

M. Sc. Miguel Enrique Zea Arenales

(f) 

Ing. Kurt Emmanuel Kellner

Fecha de aprobación: Guatemala, 13 de febrero de 2025.

Prefacio

Este trabajo está dedicado primeramente a Dios,
por darme la vida y las fuerzas para concluir con mi carrera.

A mis padres, Román y Blanqui,
por el sacrificio interminable que han hecho para que este momento llegara.
Por impulsarme a seguir y demostrarme que no hay sueños imposibles para mí.

A mi hermano Román,
por ser mi ejemplo y compañero de vida.

Mis sobrinos, Matías y Eithan,
quienes me impulsan a ser mejor cada día.

Mis abuelos Manuel, Lucero, Rosita y César,
porque, conmigo o desde el cielo,
han estado en cada paso de mi carrera.

A mis compañeros de Universidad, especialmente a mi PDF,
quien estuvo siempre para mí.

A mi asesor, Miguel Zea
quien me orientó y guió durante este tiempo,
gracias por su paciencia.

Y a mí,
por no rendirme, por ser fuerte y cumplir mis sueños.

Papis, hoy puedo decirles que orgullosamente
siempre he sido la hija de un mecánico de motos y una estilista,
pero ahora ustedes se convierten en los papás de una ingeniera mecatrónica.

Índice

Lista de figuras

Lista de cuadros

Resumen

El presente trabajo se centró en la integración de un manipulador robótico MaxArm dentro del ecosistema Robotat, utilizando una *suite* de sensores que incluyó sensores para medir distancia, sonido, luz, color y presión táctil. La integración requirió un enfoque multidisciplinario que abarcó tanto el diseño de hardware como el desarrollo de software, asegurando una comunicación eficiente y precisa entre el manipulador y el ecosistema.

Para lograr la comunicación inalámbrica, se implementó una conexión basada en el protocolo TCP/IP, utilizando un módulo ESP32 como servidor, y un software de control desarrollado en MATLAB como cliente. Este software incluyó una interfaz gráfica diseñada en MATLAB App Designer, que permitió al usuario interactuar con el sistema de manera intuitiva. Las funcionalidades de la interfaz incluyeron la selección de sensores, monitoreo en tiempo real, visualización de datos en gráficas dinámicas.

El proceso de calibración de los sensores se diseñó cuidadosamente para garantizar mediciones precisas y confiables. Se aplicaron técnicas como regresión polinómica, para el sensor de distancia, ajustes de umbrales, para el sensor de color, categorización de niveles, para el sensor de sonido, entre otras. Estas calibraciones redujeron significativamente los errores en las mediciones, destacando un error máximo de 3.15 % en el sensor de distancia y una precisión del 90 % en la identificación de colores básicos bajo condiciones de iluminación controlada.

Los resultados obtenidos validaron la efectividad del sistema integrado, destacando su capacidad para realizar tareas de monitoreo, clasificación y control en tiempo real. Las pruebas operativas mostraron que el manipulador robótico, en conjunto con los sensores calibrados y el software de control, cumplió con los objetivos planteados al inicio del proyecto. Además, se identificaron áreas de mejora, como la sensibilidad del sensor de color al ambiente lumínico o la perdida de datos con el sensor de distancia, que servirán como base para futuros desarrollos.

Este proyecto representa una contribución significativa en la integración de siste-

mas robóticos dentro de entornos avanzados como el Robotat, ofreciendo una plataforma modular y escalable para la investigación y aplicaciones en robótica. Los avances logrados sientan las bases para la implementación de manipuladores robóticos en tareas industriales, educativas y de investigación, y abren nuevas oportunidades para la mejora de tecnologías existentes en el campo de la automatización.

Abstract

The present work focused on integrating a MaxArm robotic manipulator within the Robotat ecosystem, using a suite of sensors that included devices for measuring distance, sound, light, color, and tactile pressure. The integration required a multidisciplinary approach encompassing both hardware design and software development, ensuring efficient and precise communication between the manipulator and the ecosystem.

To achieve wireless communication, a connection based on the TCP/IP protocol was implemented, using an ESP32 module as the server and a control software developed in MATLAB as the client. This software included a graphical interface designed in MATLAB App Designer, allowing users to interact with the system intuitively. The interface functionalities included sensor selection, real-time monitoring, and data visualization through dynamic graphs.

The sensor calibration process was carefully designed to ensure accurate and reliable measurements. Techniques such as polynomial regression for the distance sensor, threshold adjustments for the color sensor, and level categorization for the sound sensor, among others, were applied. These calibrations significantly reduced measurement errors, achieving a maximum error of 3.15

The results obtained validated the effectiveness of the integrated system, highlighting its ability to perform real-time monitoring, classification, and control tasks. Operational tests demonstrated that the robotic manipulator, together with the calibrated sensors and control software, met the objectives set at the beginning of the project. Additionally, areas for improvement were identified, such as the color sensor's sensitivity to ambient light or data loss with the distance sensor, which will serve as a foundation for future developments.

This project represents a significant contribution to the integration of robotic systems within advanced environments like Robotat, offering a modular and scalable platform for research and applications in robotics. The achievements lay the groundwork for the implementation of robotic manipulators in industrial, educational, and

research tasks, opening new opportunities to improve existing technologies in the field of automation.

CAPÍTULO 1

Introducción

El avance de la tecnología en robótica ha transformado diversos campos de la industria, la investigación científica y la educación, permitiendo la implementación de manipuladores robóticos capaces de realizar tareas complejas con alta precisión y adaptabilidad. Estos sistemas combinan hardware sofisticado con software avanzado, lo que los convierte en herramientas clave en la automatización de procesos.

Este trabajo se centra en el desarrollo e integración de un manipulador robótico dentro del ecosistema Robotat, una plataforma diseñada para la operación y control de sistemas robóticos. La integración de sensores, la calibración precisa de sus lecturas y el diseño de un sistema de comunicación inalámbrica fueron aspectos fundamentales para lograr un sistema eficiente y confiable. Los sensores empleados incluyeron módulos de distancia, color, luz, sonido y tacto, los cuales fueron ajustados y validados para garantizar lecturas consistentes en diversos entornos operativos.

Adicionalmente, se desarrolló una interfaz gráfica en MATLAB que permite al usuario interactuar con el manipulador de forma intuitiva. Esta interfaz facilita la selección de sensores, la obtención de lecturas en tiempo real y la visualización de datos mediante un sistema de comunicación basado en el protocolo TCP/IP. Este enfoque no solo asegura una interacción fluida entre el usuario y el manipulador, sino que también maximiza el rendimiento y la adaptabilidad del sistema.

El objetivo de este proyecto es documentar el proceso de desarrollo del manipulador robótico, desde la integración y calibración de los sensores hasta la implementación del sistema de comunicación y monitoreo. Este trabajo busca aportar una solución técnica integral que sirva como base para futuros desarrollos en robótica, destacando la importancia de la precisión en las lecturas de sensores y la interoperabilidad entre hardware y software.

La realización de este proyecto implicó enfrentar desafíos técnicos significativos, como la calibración de sensores para minimizar errores, el diseño de un sistema de

comunicación confiable y la validación del desempeño en un entorno dinámico. Los resultados obtenidos demuestran la eficacia del enfoque adoptado, ofreciendo una solución robusta y escalable para aplicaciones en robótica.

CAPÍTULO 2

Antecedentes

Para comprender de una manera más clara el funcionamiento del manipulador robótico, sus sensores y el ecosistema en el cual se va a desarrollar y dado que no se cuenta con trabajos de investigación similares, se tomaron cuatro proyectos como antecedente. Siendo el primero, el trabajo de graduación presentado por el Ingeniero Camilo Perafán, en el cual se presenta el desarrollo de una red de comunicación wifi en conjunto con el sistema de captura OptiTrack, para formar el ecosistema de experimentación ROBOTAT. El segundo, es el trabajo de graduación del Ingeniero José David Pellecer, en el cual presenta el desarrollo y la implementación de herramientas para controlar de forma inalámbrica el manipulador serial R17 dentro del ecosistema ROBOTAT. Cabe mencionar que no se cuentan con antecedentes que en conjunto presenten la implementación de un manipulador serial con sensores, se presentan estos dos trabajos como antecedentes al uso de sensores para dotar de autonomía a distintos brazos robóticos. Siento el primero, acerca de la implementación de sistemas de selección y posición de objetos, asistidos por imágenes para la detección de características como el color, la forma y tamaño de estos. El segundo y último, también con el objetivo de la detección de colores para la selección y posicionamiento de objetos, esta vez siendo asistido por dos sensores, uno de colores y el otro de distancias, para realizar tareas automáticas de selección y colocación.

Robotat: un ecosistema robótico de captura de movimiento y comunicación inalámbrica

Este proyecto[1] se enfoca en la realización de un ecosistema de tipo robótico para realizar captura de movimiento y comunicación inalámbrica. Haciendo uso de la red wifi junto con el sistema de captura OptiTrack, se creó un ambiente de experimentación en robótica llamado ROBOTAT, implementado en el salón de laboratorio 116 del Centro de Innovación y Tecnología (CIT). La implementación del hardware incluyó la instalación de una tarjeta Ethernet adicional, seis cámaras calibrada eficientemente, colocadas de manera estratégica para conseguir una cobertura óptima dentro del sis-

tema y garantizar la correcta captura del movimiento de los objetos en la plataforma y un router que proporcionaría la red wifi. Es importante mencionar algunos de sus resultados más relevantes, dentro de los cuales destaca el desarrollo e implementación de librerías tanto en lenguaje C como Python, para facilitar la conexión a la red wifi y la transmisión de datos eficiente con el sistema en ambas direcciones de manera inalámbrica, también se utilizaron protocolos de tipo MQTT para la obtención de datos del sistema OptiTrack.

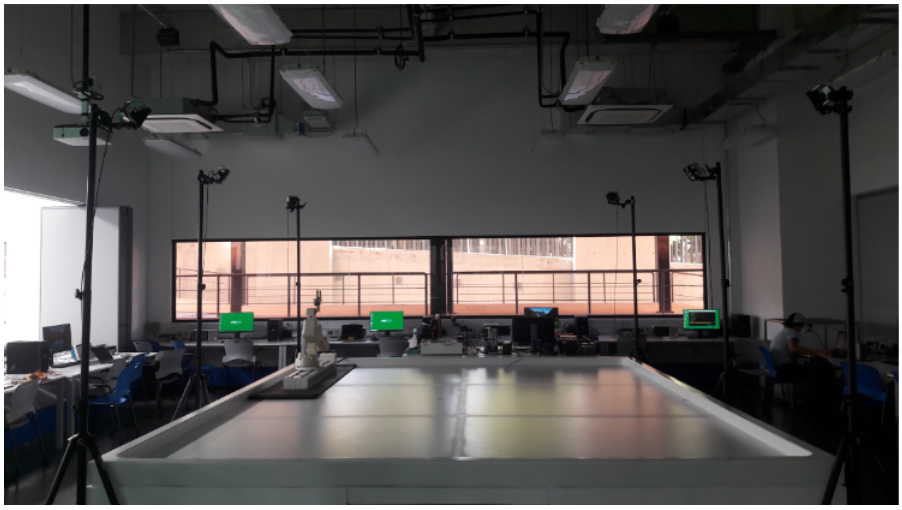


Figura 1: Ecosistema Robotat

Diseño e implementación de un paquete de herramientas de software para controlar de forma inalámbrica un manipulador serial R17 dentro de un ecosistema basado en captura de movimiento

Este proyecto[2] se centra en desarrollar un conjunto de herramientas de software para controlar un manipulador serial R17, esto mediante el sistema de captura de movimiento de OptiTrack. El proyecto se dividió en 3 fases principales: prueba de montaje y captura de datos con OptiTrack, modelado cinemático y desarrollo de software dentro del programa matemático MATLAB, e integración del manipulador en el ecosistema ROBOTAT. La base del trabajo se enfocó en el desarrollo y diseño de un sistema embebido con ESP32 para la conexión y transmisión de datos del manipulador, junto con la creación de librerías para adquirir y enviar datos de manera inalámbrica. Además de llevar a cabo análisis de cinemáticas directa e inversa para planificar así las trayectorias.

Figura 2: Manipulador robótico R17

Computer Vision-based Robotic Arm for Object Color, Shape, and Size Detection

Este proyecto[3] consistió en la implementación de un sistema robótico de agarre asistido por imágenes, capaz de seleccionar, disponer objetos para completar diversas tareas, logrando por medio de procesamiento de imágenes distinguir distintos objetos en función de sus colores, habiendo tomado como base solo 3 de ellos (rojo, amarillo y verde). Además contó con la capacidad de liberar objetos en ubicaciones predeterminadas, teniendo la posibilidad de utilizarse para un aspecto industrial. Su metodología inició con la selección de tecnologías en donde incluyen la cámara para la detección de objetos y el uso de Arduino Mega, además de servomotores. Luego, se desarrolló el sistema de visión que permitió la detección y selección precisa de objetos para su posterior manipulación. Aunado a esto, se implementaron distintos algoritmos para extraer la información de los objetos y poder "traducirlo" a un lenguaje que pueda entender el brazo. Finalmente, se llevaron a cabo distintas pruebas, para poder así evaluar la precisión y eficacia con la que trabaja el sistema. Se logró implementar un sistema capaz de detectar objetos según su forma y colores específicos, también poder colocarlos en lugares determinados, con una precisión alta, evidenciando así la capacidad del robot. Sin embargo, se tuvo varias limitaciones como la velocidad de procesamiento del sistema de captura y los costos significativos de la cámara y servomotores.

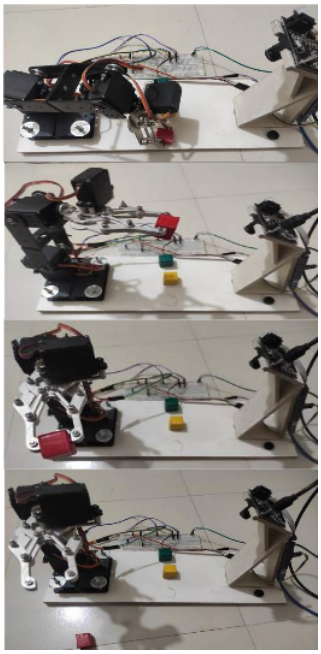


Figura 3: Sistema propuesto de visión por computadora

Automatic Control of Color Sorting and Pick/Place of a 6- DOF Robot Arm

Este proyecto[4], se centró en el diseño e implementación de un brazo robótico con 6 grados de libertad. Controlado de manera inalámbrica por medio del módulo wifi ESP 01 y una placa de desarrollo Arduino Uno, con una interfaz Blynk en donde el usuario enviaba instrucciones al manipulador, para realizar trabajos automáticos, como lo eran la clasificación por medio de colores, utilizando el sensor de color TCS3200 y tareas de recogida y colocación de objetos, por medio del sensor ultrasónico HC-SR04, para detectar la presencia de objetos. Logrando en conjunto crear y establecer modos y rutinas para el brazo robótico de manera automática, detectando el color de los objetos y siendo clasificados en las ubicaciones designadas por el usuario, además de poder establecer y desplegar la distancia a la que se encuentra un objeto del manipulador, teniendo una precisión del noventa porciento en la selección de color.

Picking process

Figura 4: Sistema de clasificación

CAPÍTULO 3

Justificación

Actualmente, el campo de la robótica ha presentado avances en campos médicos, industriales, educativos, entre otros, haciendo énfasis en la utilización de manipuladores robóticos tipo brazos, para la realización de distintas tareas. Estos manipuladores pueden ejecutar actividades repetitivas de manera precisa, ofreciendo así grandes beneficios en sectores claves como la industria manufacturera y la medicina. En estas ramas, la capacidad de los robots para reconocer, detectar y manipular objeto se ha vuelto crucial.

En Latinoamérica, específicamente en Guatemala, no se ha tenido la necesidad de adoptar la robótica como algo fundamental para el desarrollo del país, debido a la falta de recursos destinados al área de tecnología. Sin embargo, en la Universidad del Valle de Guatemala, se apuesta por la innovación y tecnología, por lo que fue adquirido un lote de manipuladores robóticos MaxArm, de Hiwonder. Si bien, ya se contaba con un brazo robótico, el R17, este era antiguo y se contaba solo con el manipulador robótico.

Estos nuevos manipuladores robóticos, a pesar de ser de bajo costo, cuentan con una suite de sensores variados, desde un sensor infrarrojo, hasta un sensor táctil. Sin embargo, estas herramientas son un tanto rústicas, por lo que se busca potenciarlos sobre los brazos robóticos, con la finalidad de brindarle autonomía al manipulador en cuestión.

El presente trabajo de investigación se basa en el desarrollo de un sistema robótico capaz de realizar tareas como detección y manipulación de objetos de forma autónoma e inalámbrica, basado en características físicas como el color, forma y tamaño de un objeto a través del uso de sensores. También se busca implementar una interfaz agradable, accesible e intuitiva para el usuario, con el fin de que la comunicación humano-robot sea eficaz. Esta permite al usuario controlar el brazo robótico, además de mostrar en tiempo real y de manera inalámbrica los datos obtenidos por los sensores dentro del ecosistema ROBOTAT. Es importante mencionar que en paralelo se está

realizando un trabajo de investigación que tiene como objetivo diseñar e implementar un software que permita controlar el movimiento del brazo robótico MaxArm.

CAPÍTULO 4

Objetivos

4.1. Objetivo general

Desarrollar e implementar herramientas de software para la operación y control remoto del brazo robótico MaxArm de Hiwonder dentro del ecosistema ROBOTAT, para aplicaciones de clasificación y manipulación de objetos.

4.2. Objetivos específicos

- Utilizar la red local wifi del ecosistema ROBOTAT para establecer una conexión inalámbrica estable y segura entre el brazo robótico MaxArm y su software de control, asegurando la transmisión confiable de datos y comandos en tiempo real.
- Estandarizar la calibración y el uso de una suite de sensores como por ejemplo, medidor de distancia, detección de color y luz ambiental, entre otros para su uso en combinación con el brazo robótico para la detección precisa de objetos y la realización eficiente de tareas de clasificación de color y colocación.
- Diseñar e implementar una interfaz de usuario intuitiva y accesible que permita una comunicación con el dispositivo de control y el manipulador serial, mostrando datos relevantes provenientes de los sensores y facilitando la interacción bidireccional entre el usuario y el sistema.

CAPÍTULO 5

Alcance

El presente trabajo se enfocará en la integración del manipulador robótico MaxArm dentro del ecosistema Robotat, logrando una sinergia entre hardware y software mediante la implementación de una *suite* de sensores calibrados y un sistema de control inalámbrico avanzado. El alcance del proyecto abarcará los siguientes aspectos principales:

1. Calibración de sensores: cada uno de los sensores integrados (distancia, color, luz, sonido y táctil) será sometido a procesos de calibración específicos para garantizar lecturas precisas y consistentes en condiciones controladas y reales. Este proceso incluirá la aplicación de técnicas como regresiones matemáticas, ajustes de umbrales y categorización de niveles, logrando minimizar errores en las mediciones y aumentar la confiabilidad del sistema.

2. Implementación de comunicación inalámbrica: se desarrollará una conexión TCP/IP eficiente utilizando un módulo ESP32 como servidor y un software cliente implementado en MATLAB. Esta comunicación permitirá la transmisión fluida de datos entre el manipulador robótico y el ecosistema Robotat, posibilitando el monitoreo remoto y en tiempo real.

3. Desarrollo de interfaz gráfica: para facilitar la interacción del usuario con el sistema, se diseñará una interfaz gráfica en MATLAB App Designer que permitirá seleccionar sensores, visualizar lecturas en tiempo real, realizar calibraciones automatizadas y generar gráficos dinámicos. La interfaz también simplificará la supervisión de las operaciones del manipulador, mejorando y facilitando la experiencia del usuario.

4. Validación y pruebas operativas: el sistema será sometido a pruebas exhaustivas en escenarios reales y controlados, mostrará su efectividad en tareas como clasificación de colores, detección de contacto y monitoreo de niveles de luz y

sonido. Los resultados validarán la precisión de las calibraciones realizadas y la estabilidad del sistema en su conjunto.

5. Escalabilidad y aplicaciones potenciales: el diseño del sistema ofrecerá flexibilidad para ser adaptado a múltiples aplicaciones, como clasificación de objetos, monitoreo de condiciones ambientales y control táctil. Esta modularidad lo hará ideal para investigaciones académicas, entornos educativos y aplicaciones industriales específicas.

No obstante, este proyecto se limitará a la integración y calibración de sensores comerciales preexistentes, así como al uso de un modelo de manipulador robótico ya disponible. Si bien se realizarán pruebas en condiciones controladas, se dejará abierta la posibilidad de extender los análisis y optimizaciones a entornos más complejos.

En resumen, este proyecto proporcionará un marco robusto y escalable para la integración de manipuladores robóticos dentro de ecosistemas avanzados como Robotat, sentando las bases para futuras innovaciones en robótica.

CAPÍTULO 6

Marco teórico

6.1. MaxArm

El Max Arm de Hiwonder es un manipulador robótico diseñado con fines educativos y de investigación en robótica y programación. Este dispositivo cuenta con una estructura robusta construida con materiales de alta calidad, que le proporcionan durabilidad y precisión en sus movimientos. Su compatibilidad con lenguajes de programación como Python y plataformas de desarrollo como Arduino, además de su placa de desarrollo ESP32 de código abierto, permite una fácil integración en diversos proyectos.

Figura 5: Manipulador robótico MaxArm de Hiwonder

6.1.1. Especificaciones físicas

- 158 mm de longitud
- 160 mm de ancho
- 260 mm de altura
- Peso: 1.3 kg

6.1.2. Otras especificaciones

- Material: tablero de metal y fibra de vidrio
- Grados de libertad (DOF) del brazo robótico: 4 DOF
- Fuente de poder: adaptador de CC 12V 5A
- Sistema de controlador: controlador de código abierto ESP32
- Software compatible: software de PC, aplicaciones para iOS y Android
- Métodos de comunicación: wifi y bluetooth
- Tipos de servo: servo bus HTS-35H y micro servo LFD-01M
- Métodos de control:ordenador, teléfono, mando inalámbrico, control por medio de *mouse*.

6.2. Sensores

Un sensor imita la capacidad que tiene el ser humano para percibir lo que se encuentra a su alrededor a través de sus sentidos. Es por esto que cada vez es más común encontrar este tipo de tecnología en los dispositivos. Los sensores pueden ser de distintos tipos según su naturaleza, dentro de los que más destacan se encuentran:

6.2.1. Clasificación de sensores

Según su funcionamiento

- Activos: estos requieren de una fuente externa de energía de donde reciben alimentación para su funcionamiento.
- Pasivos: no requieren una fuente de alimentación externa, solo hacen uso de las condiciones medioambientales para su funcionamiento.

Según sus señales

- Analógicos: estos brindan la información mediante una señal de tipo analógico, pueden tomar una infinidad de valores entre un valor máximo y uno mínimo.
- Digitales: estos brindan la información a través de una señal digital que puede ser 0 o 1, o bien tomar la forma de bits.

Según los elementos utilizados en su fabricación

- Resistivos: son los que en su fabricación hacen uso de elementos resistivos.
- Capacitivos: son los que en su fabricación hacen uso de condensadores.
- Inductivos: son los que en su fabricación hacen uso de bobinas.

6.2.2. Sensores del kit adquirido

El kit adquirido por la Universidad del Valle cuenta con una variedad de sensores y accesorios, los cuales se describen a continuación:

Sensor ultrasónico brillante

Este [5] es un sensor ultrasónico de alta precisión utilizado principalmente para medir la distancia a la que se encuentran los objetos frente a él. Integra en su interior

un circuito transmisor ultrasónico, un circuito receptor ultrasónico y un circuito de procesamiento digital. Este módulo utiliza comunicación I2C.

Especificaciones técnicas

- Voltaje de operación: 5 V DC
- Corriente de operación: 2 mA
- Frecuencia de operación: 40 Hz
- Distancia efectiva de medición: 2 cm - 400 cm
- Comunicación: I2 C

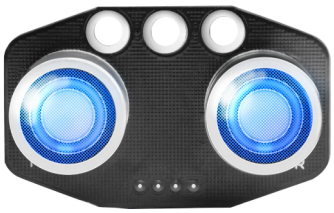


Figura 6: Sensor ultrasónico brillante de Hiwonder

Sensor de color

Este sensor [6] es utilizado para la detección del color de los objetos. Integra el circuito APDS-9960, el cual puede reconocer color en los objetos a través de la detección RGB.

Especificaciones técnicas

- Voltaje de operación: 5 V
- Corriente de operación: 100 mA máximo
- Distancia efectiva de medición: 2 mm - 100 mm
- Ángulo efectivo de medición: -20° a 20° hacia adelante

Figura 7: Sensor de color de Hiwonder

Sensor táctil

Este sensor [7] de tipo capacitivo puede detectar el toque del cuerpo humano y otros materiales conductores.

Especificaciones técnicas

- Voltaje de operación: 5 V DC
- Corriente de operación: 5 mA

Figura 8: Sensor táctil de Hiwonder

Sensor de sonido

Este sensor [8] está diseñado para detectar niveles de volumen. Integra un micrófono y un circuito amplificador basado en el LM358.

Especificaciones técnicas

- Voltaje de operación: 5 V DC
- Corriente de operación: 5 mA

Figura 9: Sensor de sonido de Hiwonder

Sensor de luz

Este sensor [9] se aplica principalmente a la detección de luz ambiental.

Especificaciones técnicas

- Voltaje de operación: 5 V DC
- Corriente de operación: 5 mA

Figura 10: Sensor de luz de Hiwonder

Cubos de madera de colores

Estos cubos se utilizan como elementos interactivos para los proyectos realizados con el manipulador robótico. Son utilizados dentro de las actividades realizadas con los sensores, ya que pueden detectarse su color y distancia.

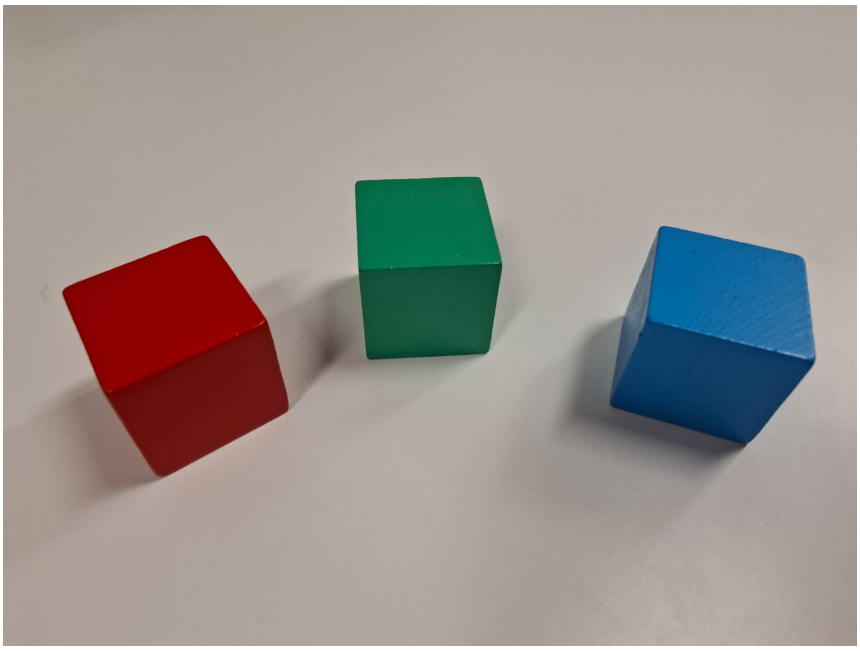


Figura 11: Cubos de madera de colores de Hiwonder

6.3. ESP32

Este[11] es un microcontrolador de bajo costo y alto rendimiento con conectividad integrada wifi y bluetooth, fabricado por Espressif Systems. Este dispositivo es ampliamente utilizado en sistemas embebidos y proyectos de robótica debido a sus características versátiles y su capacidad para manejar múltiples tareas. El MaxArm de Hiwonder utiliza específicamente el módulo Wroom32.

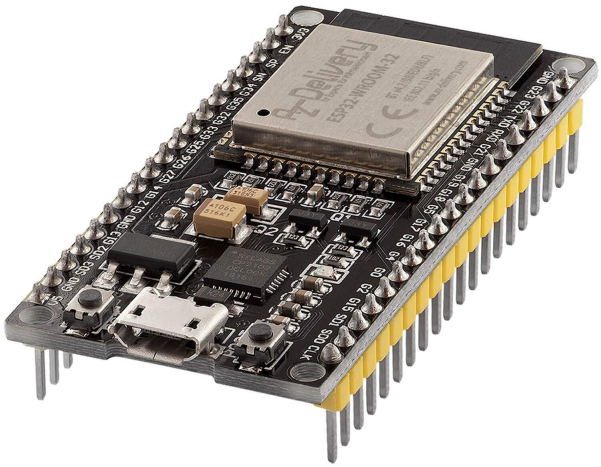


Figura 12: Microcontrolador ESP32

6.3.1. Especificaciones técnicas

- Microprocesador: CPU dual-core Tensilica LX6
- Frecuencia: hasta 240 MHz
- Memoria: 520 KB SRAM
- Almacenamiento: flash de 4 MB
- Conectividad: wifi 802.11 b/g/n, bluetooth v4.2 BR/EDR y BLE
- Periféricos: 34 pines entrada y salida de propósito general.
- Interfaces: SPI, I2 C, I2 S, UART, PWM, ADC, DAC.
- Alimentación: 2.2 V a 3.6 V

6.3.2. Características y aplicaciones

- Conectividad integrada: la capacidad de conectarse a redes wifi y dispositivos bluetooth hace que este módulo sea ideal para proyectos, permitiendo la creación de dispositivos inteligentes conectados a internet.
- Versatilidad de programación: puede ser programado utilizando varios entornos de desarrollo como Arduino IDE y MicroPython, lo que proporciona flexibilidad y facilidad de uso.

- Manejo de múltiples tareas: gracias a su procesador de doble núcleo y capacidad de multitarea, el ESP32 puede manejar tareas complejas y múltiples procesos simultáneamente.
- Aplicaciones en robótica: utilizado en proyectos de robótica para el control de motores, sensores y comunicación inalámbrica, permitiendo la construcción de robots autónomos y sistemas de control remoto.

6.4. Archivos JSON

Los archivos JSON (JavaScript Object Notation), son un formato ligero para el intercambio de datos. Este es básicamente un formato de texto que representa estructuras de datos basados en la sintaxis de objetos de JavaScript.

6.4.1. Estructura JSON

- Objetos: representados por llaves ‘‘, contienen pares de nombre/valor.
- Arreglos: representados por corchetes ‘[]‘, contienen una lista ordenada de valores.
- Valores: pueden ser una cadena de texto , número, objeto, arreglo, verdadero, falso, o nulo.

6.4.2. Funcionamiento

JSON es usado principalmente para transmitir datos entre un servidor y una aplicación web. Es un formato ideal para API (Application Programming Interfaces) por su simplicidad y eficiencia. El cual funciona de la siguiente manera:

- Solicitud: un cliente (navegador) hace una solicitud a un servidor.
- Respuesta: el servidor procesa la solicitud y responde con datos en formato JSON.
- Procesamiento: el cliente recibe los datos JSON y los procesa para actualizar la interfaz de usuario.

6.5. Regresiones

Las regresiones, son técnicas estadísticas fundamentales para modelar la relación entre una variable dependiente y una o más variables independientes. Estas técnicas

son ampliamente utilizadas en áreas como robótica, aprendizaje automático, economía y ciencia de datos, donde es esencial predecir o estimar resultados basados en datos históricos.

Este apartado, se centra en dos tipos principales de regresión: la regresión lineal y la regresión polinómica.

6.5.1. Regresión lineal

La regresión lineal es el método más sencillo y ampliamente utilizado para modelar una relación lineal entre una variable dependiente y y una variable independiente x. Este modelo se representa matemáticamente mediante una ecuación de la forma:

$$y = \beta_0 + \beta_1 x + \epsilon$$

Donde:

- y: es la variable dependiente o resultado que queremos predecir.
- x: es la variable independiente o predictor.
- β_0: es el término independiente (intersección con el eje y).
- β_1: es el coeficiente de la variable independiente, que indica la pendiente de la línea.
- ϵ: es el término de error, que representa la desviación entre los valores observados y los valores predichos.

La regresión lineal es útil para casos donde la relación entre las variables es aproximadamente lineal y fácil de interpretar. Por ejemplo, en robótica, puede utilizarse para predecir la distancia recorrida por un robot en función del tiempo de operación.

Ventajas de la regresión lineal

- Es fácil de implementar y entender.
- Tiene baja complejidad computacional.
- Permite interpretar la relación directa entre las variables.

Limitaciones de la regresión lineal

- No funciona bien con relaciones no lineales.
- Es sensible a valores atípicos.
- Requiere que los datos cumplan ciertos supuestos, como homocedasticidad y normalidad de los errores.

6.5.2. Regresión polinómica

La regresión polinómica es una extensión de la regresión lineal que permite modelar relaciones más complejas y no lineales entre las variables. En este caso, la ecuación toma la forma:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \cdots + \beta_n x^n + \epsilon$$

Donde:

- n: es el grado del polinomio.
- x^i: representa los términos elevados a diferentes potencias.

Cada potencia de x se trata como una nueva variable independiente, lo que permite capturar relaciones no lineales. Por ejemplo, en robótica, la regresión polinómica podría ser útil para modelar trayectorias curvas de un robot o el comportamiento de sensores que tienen respuestas no lineales.

Ventajas de la regresión polinómica

- Es flexible y capaz de modelar relaciones no lineales.
- Permite ajustar datos más complejos que la regresión lineal.

.

CAPÍTULO 7

Sensores, medición y calibración dentro del Robotat

La calibración de sensores fue un paso esencial en la integración del manipulador robótico dentro del ecosistema Robotat. La precisión y confiabilidad de los datos recopilados por los sensores dependían directamente de la calibración adecuada, lo que garantizaría mediciones precisas y consistentes. En este capítulo, se detalla el proceso de calibración llevado a cabo para los sensores integrados en el manipulador robótico, así como los métodos y herramientas utilizadas para lograr una calibración precisa.

Figura 13: Sensores kit *standard* MaxArm

7.1. Medición inicial con sensores dentro del Robotat

Antes de empezar el proceso de calibración, se realizaron mediciones iniciales con los sensores. Estas sirvieron como referencia para evaluar el rendimiento de los dispositivos y determinar el grado de corrección necesario. Los sensores involucrados incluían uno de distancia, uno de sonido, uno de luz, uno táctil y uno de color.

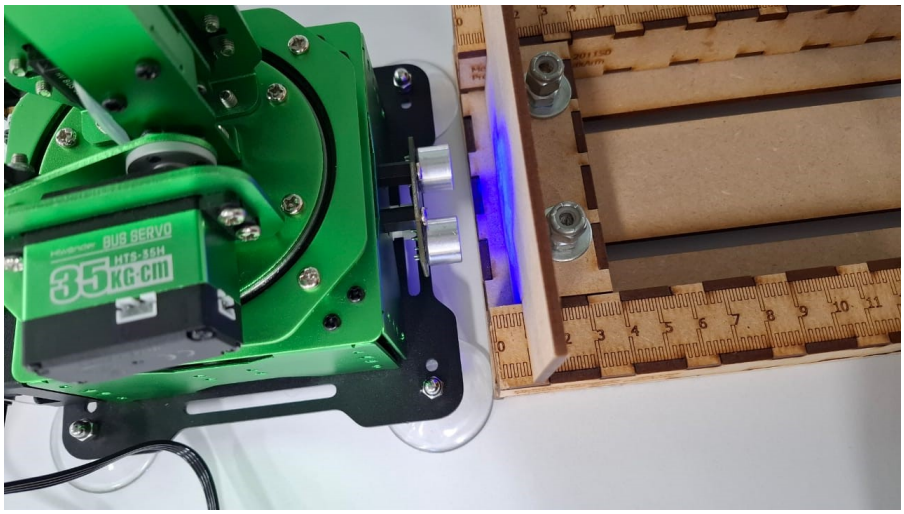


Figura 14: Ejemplo de medición realizada con regla ajustable para sensor de distancia

Durante las tomas de datos iniciales, pudo observarse que algunos sensores, en especial el sensor de distancia, presentaban desviaciones significativas en las mediciones, especialmente a distancias cortas. Esto resaltó la importancia de un proceso de calibración robusto para asegurar la precisión en las aplicaciones del manipulador.

7.2. Calibración de sensores

La calibración de los sensores se realizó utilizando una combinación de métodos, incluyendo la regresión polinómica, lineal, modificaciones de parámetros y umbrales, entre otros, para ajustar las lecturas de los distintos tipos de sensores. El proceso se llevó a cabo de la siguiente manera.

7.2.1. Proceso de calibración

Para cada uno de los sensores, se recopilaron datos crudos en distintas condiciones operativas, siendo las más importantes, dentro y fuera del ecosistema Robotat. A continuación, se detallan cada uno de los procesos que se llevaron a cabo para la realización de la calibración de los distintos sensores.

Sensor de distancia

Para este sensor, se implementó una regla ajustable que sirvió de apoyo y estandarización en la toma de datos. Esta se realizó en el software de diseño de Autodesk Inventor, la cual contaba con una pared ajustable guiada por dos carriles. Esta se detenía y aseguraba por medio de dos tornillos y sus respectivas tuercas. Así también, contaba con unidades de medida en centímetros en cada uno de sus laterales, lo que facilitó la visión y comparación de la toma de datos. Esta fue realizada con corte láser en MDF por su practicidad y tamaño.

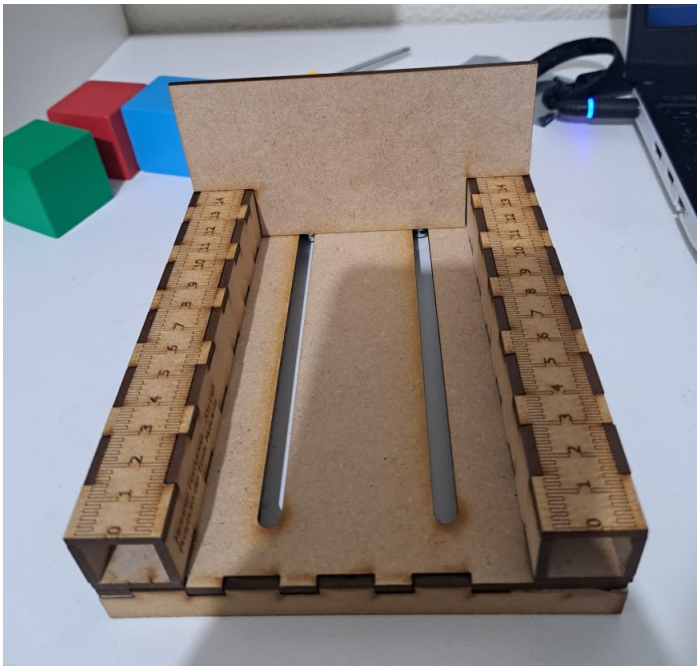


Figura 15: Regla ajustable

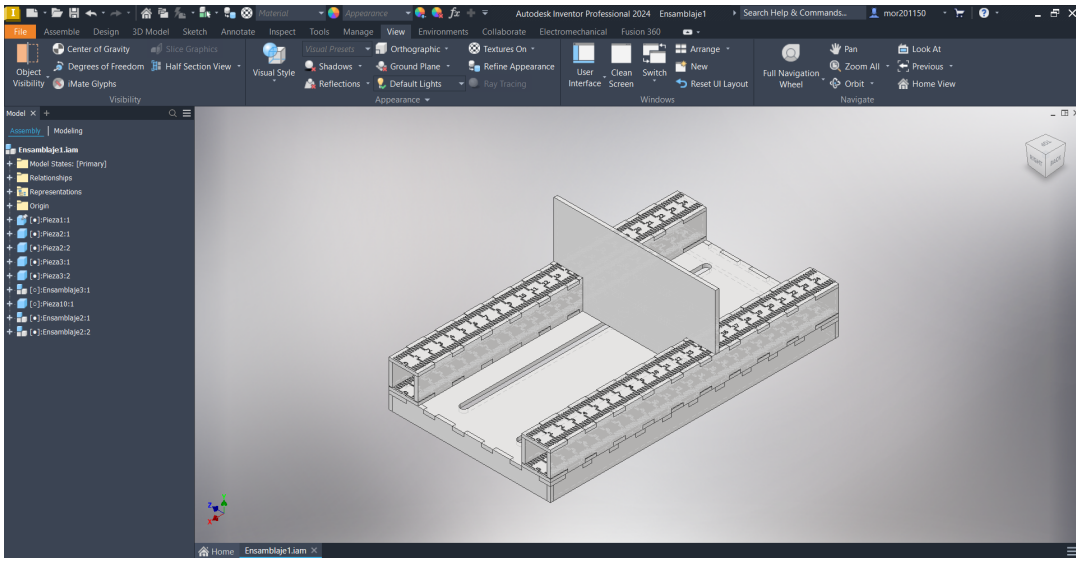


Figura 16: Regla ajustable diseñada en Inventor

Luego, se recopilaron datos crudos del sensor en diferentes condiciones operativas, siendo estas dentro y fuera del Robotat obteniendo los siguientes resultados:

Valores dentro del Robotat					
Valor esperado	Medición 1	Medición 2	Medición 3	Promedio	Error %
10	35	35	35	35	250.0
20	29	29	29	29	45.0
30	45	45	45	45	50.0
40	63	63	63	63	57.5
50	75	75	76	75.33	50.7
60	88	88	88	88	46.7
70	95	95	95	95	35.7
80	103	102	106	103.67	29.6
90	113	113	111	112.33	24.8
100	126	126	126	125.67	25.4
110	143	143	143	143	30.0
120	13140	13140	13140	13140	10850.0

Cuadro 1: Valores dentro del Robotat

Medición fuera del Robotat					
Valor esperado	Medición 1	Medición 2	Medición 3	Promedio	Error
10	24	24	24	24	140
20	23	23	23	23	15
30	33	32	33	32.67	9
40	51	51	51	51	28
50	60	60	60	60	20
60	71	71	71	71	18
70	75	77	75	75.67	8
80	89	86	85	86.67	8
90	98	97	98	97.67	9
100	107	105	110	107.33	8
110	118	116	117	117	6
120	125	125	125	125	4
130	13056	13056	13056	13056	9943

Cuadro 2: Medición fuera del Robotat

Se realizaron mediciones del sensor a diversas distancias conocidas, desde valores mínimos hasta máximos dentro de su rango operativo. Estas distancias oscilaron entre 10 mm y 120 mm, como se presenta en los Cuadros 1 y 2. En estos se puede observar que los datos sensados presentan una considerable variación con respecto a los valores esperados, especialmente en los extremos del rango analizado: antes de los 20 mm y después de los 110 mm. Este comportamiento resulta peculiar considerando que, según la hoja de datos proporcionada por el fabricante Hiwonder, el sensor Glowy tiene un rango de operación garantizado entre 20 mm y 4000 mm. Sin embargo, los resultados experimentales no reflejan esta especificación de manera consistente, lo que llevó a la decisión de reducir el rango de trabajo para garantizar lecturas más precisas siendo este de 20 mm a 120 mm.

Adicionalmente, se observó que los datos recogidos en las mediciones dentro del sistema Robotat mostraban anomalías que no estaban presentes en las mediciones realizadas fuera de este entorno. Es importante mencionar que, durante las pruebas dentro del Robotat, no solo estaban activas las cámaras de captura del sistema, sino también los pertenecientes al proyecto de graduación de otro compañero, que se encontraban operando en paralelo y, probablemente, interfiriendo con las mediciones del sensor Glowy. Esta interferencia potencial se identificó como una posible causa de las inconsistencias en las lecturas del sensor dentro y fuera del Robotat.

Para mitigar este problema, se diseñó en Autodesk Inventor una capucha especial para el sensor, con el objetivo de aislarlo de las ondas emitidas por los sistemas de captura de movimiento del Robotat y por los sensores externos. Este diseño pretendía minimizar la interferencia y permitir que las mediciones fueran más precisas. No obstante, a pesar de los esfuerzos en el diseño y la implementación de esta solución,

la capucha no logró cumplir su propósito, ya que las lecturas continuaron mostrando irregularidades atribuibles a las interferencias como es posible observarse en el Cuadro 3.

Figura 17: Capucha para sensor Glowy

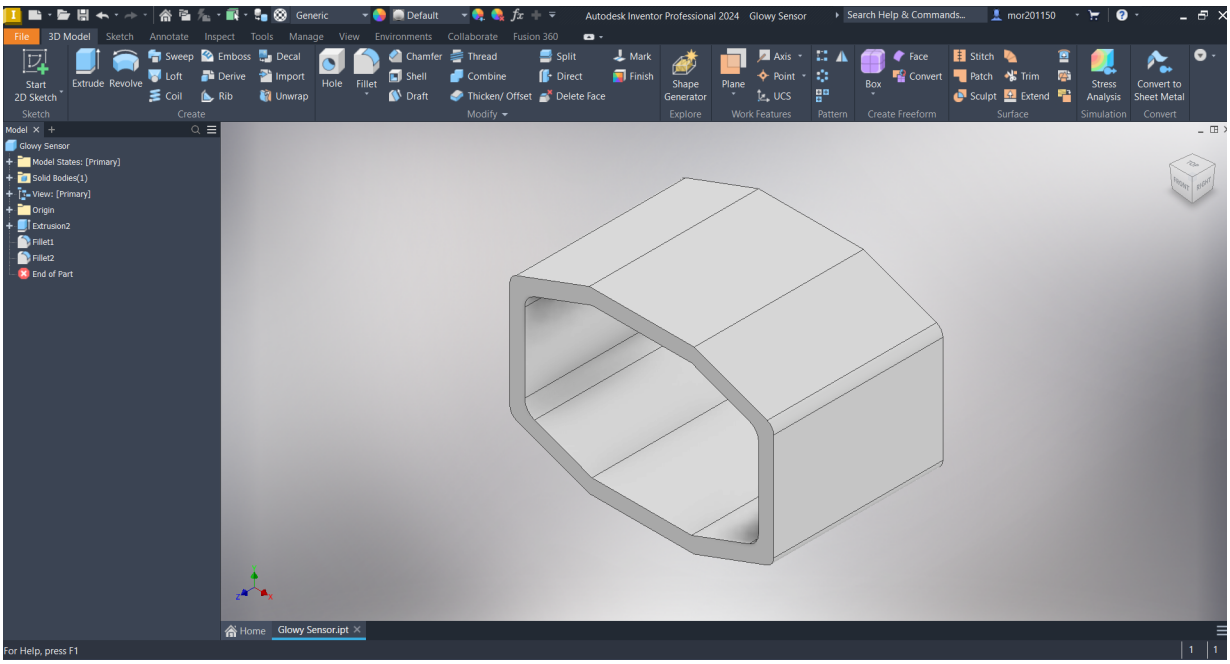


Figura 18: Capucha para sensor Glowy en Autodesk Inventor

Valores dentro del Robotat, con capucha en sensor					
Valor esperado	Medición 1	Medición 2	Medición 3	Promedio	Error %
10	36	36	36	36.0	260
20	37	37	37	37.0	85
30	48	48	48	48.0	60
40	59	59	59	59.0	48
50	71	71	71	71.0	42
60	83	83	83	83.0	38
70	90	90	90	90.0	29
80	102	102	102	102.0	28
90	114	114	114	114.0	27
100	125	126	125	125.3	25
110	139	138	139	138.7	26
120	152	152	152	152.0	27
130	13140	13140	13140	13140.0	10008

Cuadro 3: Valores dentro del Robotat, con capucha en sensor

Ante esta situación, se tomó la decisión de apagar los sistemas de captura de movimiento y los sensores ajenos al experimento durante las mediciones dentro del Robotat. Esta medida resultó ser más efectiva, ya que permitió realizar las pruebas sin interferencias externas, lo que finalmente posibilitó la obtención de datos más confiables para el análisis posterior. En donde se destacó la importancia de controlar el entorno operativo del sensor para garantizar la calidad precisión de los datos recogidos, especialmente en contextos donde múltiples sistemas tecnológicos están en funcionamiento simultáneamente.

Continuando a la etapa de calibración, se exploraron diversas técnicas matemáticas, como las regresiones de tipo lineal y polinómica, así como un ajuste único por tramos. No obstante, el ajuste por tramos por sí solo no cumplió con las expectativas, ya que aún se encontraba un error considerable, como es posible observarse en el Cuadro 4. Esto dejó en evidencia que, aunque el ajuste por tramos logró reducir parcialmente el error en ciertos rangos de operación del sensor, su implementación no fue suficiente para obtener resultados completamente satisfactorios.

Medición pruebas intervalos					
Valor esperado	Medición 1	Medición 2	Medición 3	Promedio	Error %
10	24	26	26	25.33	153.33
20	27	26	27	26.67	33.33
30	34	34	34	34.00	13.33
40	44	45	44	44.33	10.83
50	50	50	50	50.00	0.00
60	55	55	55	55.00	8.33
70	70	70	70	70.00	0.00
80	81	81	81	81.00	1.25
90	86	86	86	86.00	4.44
100	94	94	93	93.67	6.33
110	102	102	102	102.00	7.27
120	121	120	120	120.33	0.28

Cuadro 4: Medición pruebas intervalos

Dado que los ajustes iniciales por tramos no lograron reducir el error a un nivel aceptable, se tomó la decisión de aplicar técnicas de regresión más avanzadas, específicamente regresiones lineales y polinómicas, ajustadas a los distintos tramos del rango operativo del sensor. Esta metodología se diseñó para abordar las inconsistencias observadas en las mediciones, especialmente en los extremos del rango del sensor.

Utilizando los datos recopilados durante las pruebas a diferentes distancias, se procedió a implementar un modelo de calibración por tramos. Esto incluyó dos enfoques principales:

- Primer tramo: regresión polinómica de segundo grado
  Para distancias comprendidas entre los 20 mm y los 90 mm, donde se observaba un comportamiento no lineal en las mediciones, se aplicó una regresión polinómica de segundo grado. Este modelo permitió capturar la relación curva entre los valores crudos medidos por el sensor y las distancias reales. La ecuación resultante para este tramo se define como:

$$\text{Distancia calibrada primer tramo} = a \cdot (\text{Valor crudo})^2 + b \cdot \text{Valor crudo} + c \quad (1)$$

  Aquí, los coeficientes a, b y c fueron determinados a través del análisis de regresión polinómica, empleando herramientas de procesamiento de datos como Excel para generar gráficos y tablas que evidenciaran la relación entre las variables.

- Segundo tramo: regresión lineal
  Para el rango entre los 91 mm y los 129 mm, el comportamiento del sensor mostró

una relación más lineal entre los valores crudos y las distancias reales. En este caso, se optó por una regresión lineal, obteniendo una ecuación simplificada:

$$\text{Distancia calibrada segundo tramo} = m \cdot (\text{Valor crudo}) + b \tag{2}$$

Los coeficientes m y b fueron determinados mediante el método de mínimos cuadrados, también con soporte de herramientas gráficas y tabulares en Excel.

Una vez obtenidas estas ecuaciones de calibración, se integraron directamente en el código de Arduino del manipulador. El flujo de datos del sensor fue modificado para aplicar estas transformaciones en tiempo real. Cada lectura del sensor pasaba primero por la ecuación de calibración correspondiente a su tramo, ajustando así el valor crudo al valor calibrado antes de ser enviado al software de control del manipulador. Esto garantizó que los datos procesados fueran más precisos y confiables para su uso en las operaciones del sistema.

El siguiente fragmento de código ejemplifica cómo se implementaron estas ecuaciones en Arduino:

```
// Función para calibrar el valor del sensor
float calibrateDistance(int rawValue) {
    if (rawValue <= 91) {
        float a = 0.0029;
        float b = 0.7581;
        float c = -3.7035;
        return a * pow(rawValue, 2) + b * rawValue + c;
    } else {
        float m = 0.8195;
        float b = 18.72;
        return m * rawValue + b;
    }
}
```

Figura 19: Fragmento del código extraído de la calibración realizada en Arduino

La metodología de calibración por tramos, combinando regresiones polinómicas y lineales, resultó ser altamente efectiva para reducir el error en el rango operativo del sensor. Este enfoque permitió mejorar significativamente la precisión de las lecturas, especialmente en distancias cortas y medias, adaptando las estrategias de calibración al comportamiento específico del sensor en cada intervalo.

Posterior a la implementación de las ecuaciones de calibración, se realizaron pruebas para validar la precisión de las mediciones ajustadas. Los resultados mostraron una mejora notable en la exactitud, logrando que el sensor proporcionara lecturas confiables a lo largo de su rango operativo. El porcentaje de error más alto registrado fue de solo 3.15 %, lo que confirma la eficacia del modelo de calibración aplicado. Además, como medida complementaria, se integraron mensajes de advertencia para

los casos en los que el sensor opera fuera de sus límites óptimos. Cuando las distancias eran menores a 20 mm, se desplegaba la frase “El objeto se encuentra muy cerca”, mientras que para distancias superiores a 129 mm, se mostraba el mensaje “El objeto se encuentra muy lejos”.

Valor esperado	**Medición 1**	**Error**	**Error absoluto**
10	Límite inferior		
20	20.7203	0.0240	2.40 %
30	31.2605	0.0315	3.15 %
40	40.7526	0.0151	1.51 %
50	50.0219	0.0004	0.04 %
60	60.1073	0.0015	0.15 %
70	68.2763	-0.0215	2.15 %
80	79.0719	-0.0103	1.03 %
90	89.4669	-0.0053	0.53 %
100	98.2115	-0.0163	1.63 %
110	109.6845	-0.0026	0.26 %
120	119.5185	-0.0037	0.37 %
130	Límite superior		

Cuadro 5: Valores dentro del Robotat con cálculo de error, luego de calibración

Sensor de color

En cuanto al sensor de color, se recopilaron datos crudos bajo diversas condiciones de operación, evaluando su desempeño en habitaciones y lugares con diferentes niveles de iluminación. Este proceso permitió establecer una referencia precisa sobre su rango de trabajo. Dado que el sensor operaba de manera óptima a distancias menores de 10 mm, la recopilación de datos se realizó a una distancia cercana a su estructura para garantizar lecturas confiables. El proceso de calibración fue diseñado para asegurar que las lecturas del sensor identificaran con precisión los colores básicos: rojo, verde y azul.

El sensor fue conectado al sistema de control del manipulador y configurado para operar dentro de su rango máximo de sensibilidad. Además, se emplearon cubos de color estándar como patrones de referencia. Cada cubo fue colocado frente al sensor a una distancia fija de 1 cm, lo que permitió registrar mediciones consistentes y precisas para su posterior análisis y calibración.

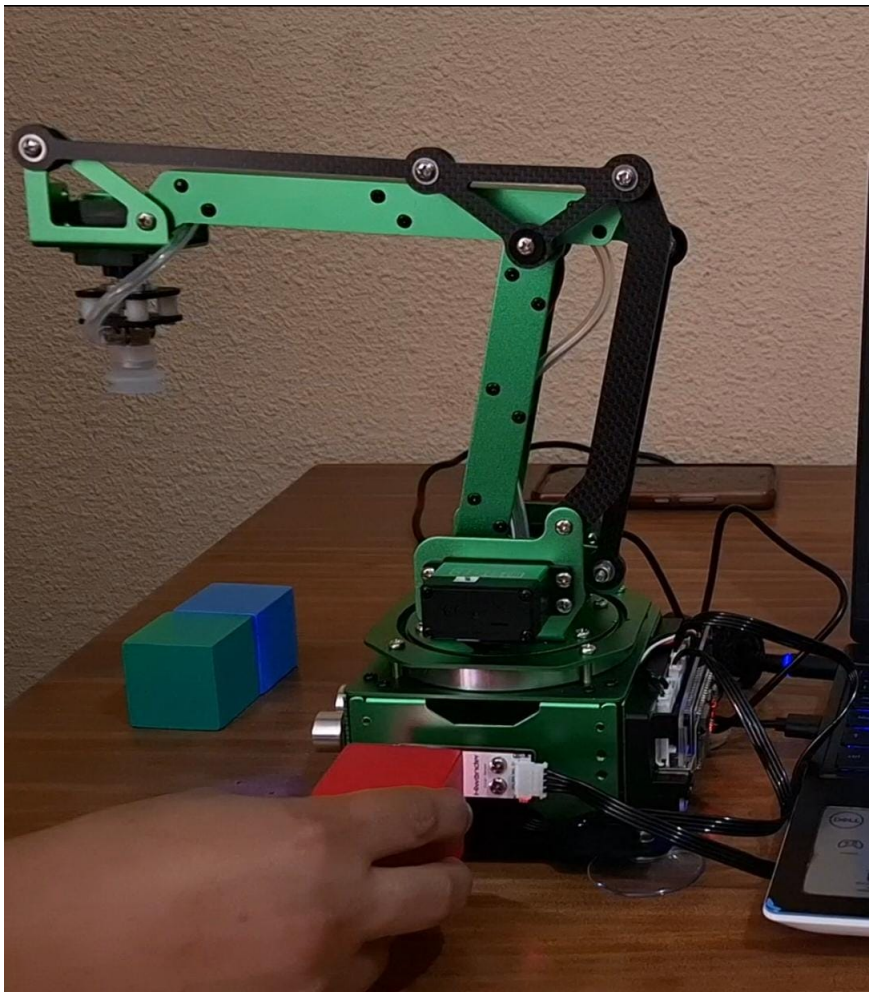


Figura 20: Prueba realizada con sensor de color

Estas son algunas de las mediciones iniciales realizadas en habitaciones con iluminaciones variables:

Mediciones dentro del Robotat					
Color esperado	Intento 1	Intento 2	Intento 3	Intento 4	Intento 5
Rojo	Rojo	Rojo	Verde	Rojo	Verde
Verde	Verde	Azul	Azul	Azul	Verde
Azul	Azul	Verde	Azul	Azul	Verde

Cuadro 6: Mediciones dentro del Robotat

Mediciones en mesa de laboratorio					
Color esperado	Intento 1	Intento 2	Intento 3	Intento 4	Intento 5
Rojo	Rojo	Rojo	Verde	Rojo	Rojo
Verde	Azul	Verde	Azul	Azul	Verde
Azul	Azul	Verde	Verde	Azul	Verde

Cuadro 7: Mediciones en mesa de laboratorio

Mediciones en cuarto de habitación					
Color esperado	Intento 1	Intento 2	Intento 3	Intento 4	Intento 5
Rojo	Rojo	Verde	Rojo	Rojo	Verde
Verde	Azul	Azul	Verde	Azul	Verde
Azul	Verde	Verde	Azul	Verde	Azul

Cuadro 8: Mediciones en cuarto de habitación

Al analizar las mediciones obtenidas en las diferentes condiciones de operación del sensor de color, se observó un comportamiento irregular en la detección de los colores verde y azul. En las pruebas realizadas dentro del Robotat (Cuadro 6), el sensor mostró una alta tasa de errores, especialmente en la identificación del verde, que fue confundido frecuentemente con azul, y viceversa. Por otro lado, las mediciones en la mesa de laboratorio (Cuadro 7), realizadas bajo condiciones más controladas de iluminación, presentaron una mejora relativa en la precisión, aunque persistieron inconsistencias significativas en la detección del azul, que fue erróneamente identificado como verde en varios intentos. Finalmente, en el cuarto de habitación (Cuadro 8), donde las condiciones de iluminación eran más variables, el sensor continuó presentando dificultades con la identificación precisa de los colores verde y azul, aunque el rojo fue identificado correctamente en la mayoría de los casos. Estos resultados evidenciaron que el sensor tenía limitaciones en su precisión, particularmente bajo condiciones de iluminación no homogéneas, lo que resaltó la necesidad de una calibración específica para abordar estas variabilidades ambientales.

Los datos recopilados del sensor de color fueron procesados para establecer umbrales precisos que definieran la identificación de los colores básicos: rojo, verde y azul. Este análisis se basó en las intensidades relativas de los canales RGB, asignando una

región de decisión a cada color según la dominancia del canal correspondiente. Por ejemplo, la región correspondiente al color rojo se definió cuando el canal R mostraba la mayor intensidad, el color verde por la dominancia del canal G, y el color azul por la predominancia del canal B. Estos umbrales iniciales fueron ajustados utilizando las recomendaciones del proveedor Hiwonder.

Para implementar esta calibración en el sistema de control del manipulador, se desarrolló un algoritmo de clasificación que compara en tiempo real las lecturas del sensor con los umbrales previamente establecidos. El sensor de color APDS-9960 mide las intensidades de luz en los canales rojo (R), verde (G) y azul (B), proporcionando valores crudos que representan la cantidad de luz captada en cada canal. En el código implementado en Arduino, estos valores son procesados mediante la función `map()`, que ajusta las lecturas a un rango estandarizado de 0 a 255. Este ajuste utiliza límites definidos por los umbrales mínimos (`r_f`, `g_f`, `b_f`) y máximos (`R_F`, `G_F`, `B_F`) para cada canal, lo que garantiza la consistencia y comparabilidad de las lecturas bajo distintas condiciones de operación.

El algoritmo luego, clasifica el color predominante con base en las intensidades relativas de los canales RGB. Por ejemplo, si el canal rojo (R) tiene la mayor intensidad en comparación con los canales verde (G) y azul (B), el color detectado será rojo (`RED`). De manera similar, el canal verde determina el color `GREEN` y el azul el color `BLUE`. Además, se establecieron umbrales mínimos (60) para cada canal, de manera que las lecturas deben superar este valor para ser consideradas válidas, evitando así clasificaciones erróneas debidas a ruido o lecturas débiles.

Para mejorar la robustez del sistema, el algoritmo toma múltiples lecturas consecutivas (cinco en este caso) y calcula un promedio. Esto reduce la influencia de errores aleatorios y garantiza una detección precisa antes de actualizar las salidas del sistema. Este diseño asegura que las lecturas del sensor sean interpretadas correctamente, incluso en condiciones de variabilidad en la iluminación, lo que mejora la precisión y confiabilidad del sistema de clasificación de colores.

```
// Función de detección de color
int ColorDetect() {
  // Retardo inicial de la detección de color
  while (!APDS.colorAvailable()) delay(5);
  // Definir variables
  int r, g, b, c;
  // Obtener los valores de los tres colores RGB
  APDS.readColor(r, g, b);
  // Escalar los valores
  r = map(r, r_f, R_F, 0, 255);
  g = map(g, g_f, G_F, 0, 255);
  b = map(b, b_f, B_F, 0, 255);

  // Determinar el color predominante según los valores RGB
  if (r > g) c = RED;
  else c = GREEN;
  if (c == GREEN && g < b) c = BLUE;
  if (c == RED && r < b) c = BLUE;

  // Si el valor del color es mayor a 50, devuelve el color detectado, de lo contrario devuelve 0
  if (c == BLUE && b > 60) return c;
  else if (c == GREEN && g > 60) return c;
  else if (c == RED && r > 60) return c;
  else return 0;
}
```

Figura 21: Fragmento del código utilizado para calibrar el sensor de color

En el caso específico del código mostrado, se observa cómo el sensor APDS-9960 mide las intensidades de los canales RGB y aplica un mapeo para escalarlas dentro del rango operativo del sensor. El algoritmo clasifica los colores dominantes basándose en condiciones lógicas definidas. Adicionalmente, el algoritmo implementa un promedio sobre múltiples lecturas consecutivas para reducir errores en la detección, aumentando así la precisión en el reconocimiento de colores.

Los resultados obtenidos durante la validación del sistema mostraron una precisión del 90 % para los colores rojo y verde en condiciones de iluminación controlada. Sin embargo, el color azul mostró ser más sensible a la luz intensa, reduciendo su precisión al 80 %. Este enfoque permitió optimizar el desempeño del sensor, adaptándolo tanto a las especificaciones del hardware como a las condiciones de operación reales, tal como se detalla en el Cuadro 9.

Color Real	**Correctas (n)**	**Incorrectas (n)**	**Precisión (%)**	**Notas**
Rojo	9	1	90 %	Lecturas consistentes
Verde	9	1	90 %	Resultados estables
Azul	8	2	80 %	Sensible a luz intensa

Cuadro 9: Resultados de la calibración del sensor de color con observaciones

Otros sensores

En cuanto a los sensores táctil, de luz y de sonido, se determinó que no serán utilizados directamente en las tareas de clasificación de objetos, ya que no aportan

características esenciales que contribuyan significativamente a dicha operación. No obstante, para garantizar su correcto funcionamiento y prevenir posibles interferencias en el sistema, se les realizaron ajustes específicos de calibración y configuración. Estos sensores se integraron al sistema no como elementos principales de clasificación, sino como dispositivos auxiliares o accionadores que facilitan el desarrollo de las tareas generales del manipulador. Su implementación permite, por ejemplo, la activación de procesos secundarios o la detección de condiciones del entorno que podrían influir en el éxito de las operaciones, asegurando así una mayor robustez en el funcionamiento global del sistema.

- **Sensor de sonido**

  El sensor de sonido fue configurado para registrar niveles de intensidad acústica en el entorno inmediato del manipulador robótico. La programación realizada en el microcontrolador utiliza lecturas analógicas crudas obtenidas del sensor, las cuales se procesan mediante la función analogRead. Estas lecturas, originalmente en un rango de 0 a 1023, se transforman en un porcentaje (0 % a 100 %) mediante una operación de mapeo con la función map. Este procedimiento asegura una representación uniforme y comprensible de los niveles de sonido, facilitando su interpretación tanto por el sistema como por los operadores humanos.

```
else if (selectedSensor == "Sonido") {
  int soundValue = analogRead(sound_sensor_pin); // Leer valor crudo
  int soundPercentage = map(soundValue, 0, 1023, 0, 100); // Convertir a porcentaje
  Serial.print("Nivel de sonido: ");
  Serial.print(soundPercentage);
  Serial.println("%");
```

Figura 22: Fragmento del código utilizado para ajustar el sensor de sonido

A diferencia de otros sensores utilizados en este proyecto, el proceso de ajuste del sensor de sonido no requirió la implementación de algoritmos complejos de calibración. Esto se debe a que las pruebas preliminares indicaron que el sensor ofrecía una precisión aceptable bajo condiciones estándar, lo que simplificó su integración al sistema. Sin embargo, se establecieron límites generales para clasificar las intensidades acústicas en porcentajes que son suficientes para cumplir con los objetivos establecidos en este proyecto.

El sensor de sonido fue utilizado principalmente para registrar niveles de intensidad acústica en el entorno inmediato del manipulador robótico. A diferencia de los sensores de distancia y de color, el proceso de ajuste para el sensor de sonido no requirió una calibración exhaustiva debido a que los datos proporcionados por el fabricante ya mostraban un buen nivel de precisión en condiciones estándar. Sin embargo, se realizaron ligeros ajustes en el código del microcontrolador para adaptar las lecturas del sensor y garantizar que los datos fueran interpretados correctamente.

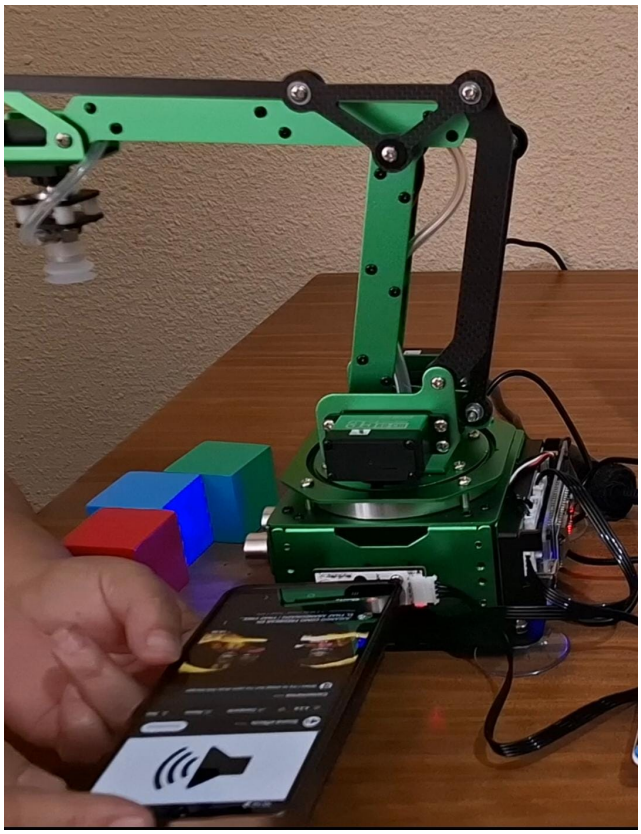


Figura 23: Prueba realizada con sonidos

El sensor de sonido se integró al manipulador robótico mediante una configuración que permitía su uso en tiempo real. A pesar de no requerir una calibración completa, los ajustes realizados en el código garantizaron que las lecturas del sensor fueran consistentes y apropiadas para las tareas del manipulador.

Además, el sensor de sonido fue implementado como un accionador inicial para las tareas de clasificación del manipulador robótico. Se configuró un umbral del 50 % de la intensidad máxima para desencadenar las operaciones del robot. Si el sensor detecta un nivel de sonido superior a este umbral, el sistema interpreta la señal como una instrucción para iniciar las tareas de clasificación. Esta funcionalidad aprovecha la sensibilidad del sensor al entorno acústico para habilitar un control reactivo y simplificado del sistema, asegurando que el robot responda únicamente a condiciones específicas de ruido ambiental.

Intensidad (%)	Rango de valor crudo (0–1023)
0 %	0
25 %	1 – 255
50 %	256 – 511
75 %	512 – 767
100 %	768 – 1023

Cuadro 10: Conversión de intensidad acústica a porcentajes en el sensor de sonido

- **Sensor de luz**

El sensor de luz, fue integrado en el manipulador robótico con el objetivo de monitorear los niveles de iluminación en su entorno de operación. Aunque tampoco se llevó a cabo un proceso de calibración completo, se realizaron ligeros ajustes basados en la configuración proporcionada por el fabricante, lo que permitió mejorar su desempeño.

Inicialmente, el sensor de luz ofrecía lecturas en valores crudos que representaban la intensidad lumínica del entorno. Estos valores eran proporcionales a la cantidad de luz incidente sobre el sensor, con un rango que iba desde 0 (oscuridad completa) hasta 1023 (máxima intensidad detectable). Para facilitar su interpretación y asegurar que las lecturas fueran adecuadas para las necesidades operativas, se implementaron ajustes de escala directamente en el código del manipulador.

```
else if (selectedSensor == "Touch") {
  int touchState = digitalRead(touch_sensor_pin); // Leer el estado táctil
  // Depuración en el monitor serial
  if (touchState == LOW) {
      Serial.println("Estado del sensor táctil: TOCADO (0)");
  } else {
      Serial.println("Estado del sensor táctil: NO TOCADO (1)");
  }
```

Figura 24: Fragmento del çódigo utilizado para ajustar el sensor táctil

A pesar de no realizar un proceso de calibración avanzado, las lecturas del sensor de luz mostraron ser suficientemente precisas para la aplicación, con un comportamiento consistente en diferentes niveles de iluminación ambiental. La integración y ajuste del sensor de luz aseguró un funcionamiento confiable, permitiendo que el manipulador evaluara adecuadamente las condiciones de iluminación en su entorno y adaptara sus operaciones según fuera necesario.

Porcentaje de oscuridad (%)	Rango de intensidad cruda
100	0 – 102
75	256 – 511
50	512 – 767
25	768 – 1019
0	1020 – 1023

Cuadro 11: Clasificación de intensidad lumínica según el sensor de luz

El sensor de luz, también fue configurado para actuar como un accionador dentro del sistema del manipulador robótico. Utilizando el porcentaje de oscuridad calculado a partir de las lecturas crudas del sensor, se estableció un umbral operativo del 50 %. Esto significa que, cuando el nivel de oscuridad detectado supere el 50 %, el sistema considera que las condiciones de luz son adecuadas para iniciar las tareas de clasificación del manipulador.

- **Sensor táctil**

El sensor táctil fue incluido en el manipulador robótico con el propósito de detectar interacciones físicas directas, como toques o presiones, esenciales para ciertas aplicaciones dentro del ecosistema Robotat. Este sensor opera de manera binaria, proporcionando dos estados: "0" para ausencia de contacto y "1" para detección de presión o contacto. Su integración al sistema no requirió transformaciones complejas de los datos, ya que su configuración en el código del manipulador permite que las lecturas sean consistentes y fáciles de interpretar.

Si bien el sensor táctil no aporta características relevantes para las tareas principales de clasificación de objetos realizadas por el manipulador, se decidió mantenerlo implementado también como un accionador. En este sentido, su funcionalidad permite que el sistema inicie las tareas operativas del manipulador al detectar contacto físico directo. Esta configuración asegura que el sensor pueda integrarse en posibles aplicaciones futuras o en escenarios donde las interacciones físicas sean necesarias.

Durante las pruebas operativas, el sensor táctil mostró un desempeño fiable, detectando contactos físicos con precisión y sin fluctuaciones significativas en sus valores. En caso de que se requiera una sensibilidad ajustada o condiciones de operación más específicas, el sensor podría ser recalibrado mediante modificaciones adicionales en el software en futuros desarrollos.

Estado del sensor	Interpretación
0	Sin contacto (NO TOCADO)
1	Contacto detectado (TOCADO)

Cuadro 12: Estados del sensor táctil implementados en Arduino

Creación de librería para calibración

Con el objetivo de estructurar y facilitar la reutilización de los procesos de calibración y funcionamiento de los sensores integrados en el manipulador robótico, se desarrolló una librería personalizada denominada `Calibracion`. Esta librería encapsuló las funciones necesarias para la gestión y calibración de los sensores de distancia, sonido, luz, color y táctil, permitiendo un acceso simplificado a sus características principales y asegurando la estandarización de su uso en futuros desarrollos.

La librería `Calibracion` fue diseñada para integrarse fácilmente en proyectos basados en Arduino. Aprovechó las librerías existentes para cada sensor, como `Ultrasound` para el sensor de distancia y `Arduino_APDS9960` para el sensor de color, combinándolas en un único paquete. Dentro de la librería, se incluyeron funciones predefinidas para:

- Calibrar y procesar las lecturas del sensor de distancia, aplicando las ecuaciones por tramos con regresiones polinómicas y lineales.
- Convertir las lecturas analógicas del sensor de sonido en porcentajes de volumen, clasificados en niveles bajo, moderado y alto.
- Normalizar las lecturas del sensor de luz para interpretar condiciones de iluminación en tiempo real.
- Detectar interacciones físicas con el sensor táctil, interpretadas de forma binaria (`0` para ausencia de contacto y `1` para detección de contacto).
- Clasificar colores básicos (rojo, verde y azul) a partir de las lecturas del sensor de color utilizando un algoritmo basado en umbrales previamente definidos.

La implementación de la librería también incluyó la gestión de comunicación wifi y la funcionalidad del servidor TCP, de lo que se hablará más adelante, facilitando la transferencia de datos sensoriales al sistema de control externo. Esto aseguró que las lecturas calibradas fueran accesibles para otros módulos del manipulador o interfaces de usuario en tiempo real.

En definitiva, la librería `Calibracion` no solo simplificó el manejo de los sensores, sino que también garantizó la replicabilidad y extensibilidad del sistema, permitiendo futuras modificaciones o mejoras sin afectar la estructura principal del código del manipulador robótico.

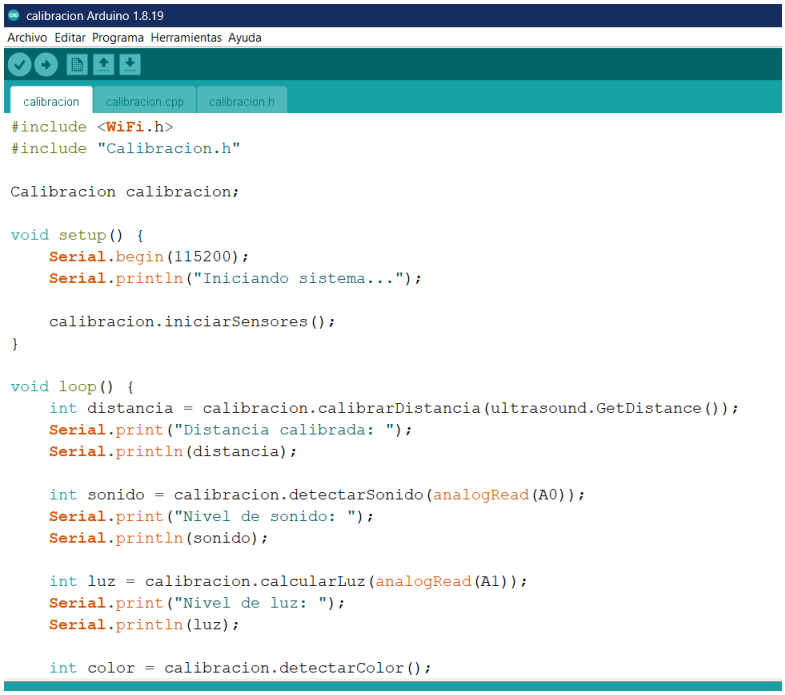


Figura 25: Librería implementada en Arduino

CAPÍTULO 8

Conexiones

A pesar de que el ecosistema Robotat ofrece una plataforma robusta para la integración y control de manipuladores robóticos, se identificó la necesidad de implementar una conexión inalámbrica eficiente para el manipulador específico utilizado en este proyecto. La capacidad de establecer una comunicación inalámbrica efectiva es crucial para permitir un control remoto y una interacción más fluida entre el manipulador y el software de control. Este capítulo describe el diseño e implementación de la conexión inalámbrica entre el manipulador y el software de control, así como los desafíos superados durante su desarrollo.

8.1. Conexión inalámbrica de manipulador a Robotat

En el proceso de integración del manipulador robótico dentro del ecosistema Robotat, la tarea más importante fue establecer una conexión inalámbrica eficiente y confiable entre el MaxArm y la red wifi del Robotat. Esta conexión, es esencial para permitir la comunicación en tiempo real y el control remoto de los sensores instalados en el manipulador, lo que facilita la operación y recolección de datos de los mismos.

8.1.1. Metodología

Para lograr esta conexión, se utilizó el módulo de comunicación wifi incorporado en la placa de desarrollo. La configuración de la conexión se llevó a cabo de la siguiente manera:

Configuración de la red wifi

La red wifi de Robotat fue configurada para proporcionar una conexión segura y estable al manipulador. Se establecieron los parámetros de la red, como el SSID y la contraseña, que se ingresaron en el código del manipulador.

```
// Configuración de WiFi
const char* ssid = "Robotat";
const char* password = "iemtbmcit116";
```

Figura 26: Fragmento del código extraído de la conexión wifi

Implementación del código de conexión

En el código del manipulador, se utilizó la librería "wifi.h" de Arduino para establecer la conexión a la red. En donde, el manipulador intenta conectarse a la red, utilizando el SSID y la contraseña proporcionados. Una vez conectado, obtiene una dirección IP asignada por el servidor de la red de Robotat, lo que le permite comunicarse con otros dispositivos en la misma red.

```
// Conectar a la red WiFi
Serial.print("Conectando a WiFi...");
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
}
Serial.println("\nConectado a WiFi.");
Serial.print("Dirección IP: ");
Serial.println(WiFi.localIP());
```

Figura 27: Fragmento del código extraído de la conexión wifi

Verificación de la conexión

Durante la etapa de desarrollo, se verificó la conexión del manipulador a la red de Robotat mediante la impresión de la dirección IP obtenida en el monitor serie. Esto permitió confirmar que el manipulador estaba correctamente conectado a la red y el envío de datos de manera inalámbrica era correcto.

```
Conectando a WiFi....
Conectado a WiFi.
Dirección IP: 192.168.7.239
```

Figura 28: Fragmento del mensaje desplegado en la terminal de Arduino

8.2. Conexión inalámbrica de manipulador a software de control

La conexión inalámbrica entre el manipulador y el software de control fue un aspecto crítico en el desarrollo de este proyecto, ya que permite la operación y el monitoreo remoto de los sensores integrados en el manipulador robótico. Para lograr esto, se estableció una comunicación TCP/IP entre el manipulador (equipado con un ESP32) y el software de control desarrollado en MATLAB.

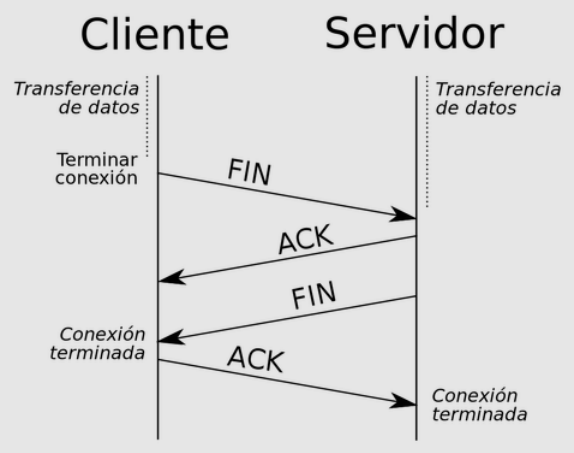


Figura 29: Protocolo TCP

Establecimiento de la comunicación

La configuración inicial implicó la conexión del manipulador a la red wifi de Robotat, permitiéndole obtener una dirección IP en la misma red que el software de control. El manipulador se configuró como servidor TCP, mientras que MATLAB actuó como cliente. Se estableció una comunicación de tipo cliente-servidor donde MATLAB podía enviar solicitudes al manipulador para recibir datos en tiempo real de los sensores. El ESP32 fue configurado como servidor TCP utilizando un puerto específico, lo que permitió la creación de un canal de comunicación confiable. Una vez configurado como servidor, el ESP32 quedó listo para aceptar conexiones entrantes desde el software de control en MATLAB, que actuaría como cliente en la comunicación. Se eligió el protocolo TCP/IP por su fiabilidad y control de errores, lo que garantiza la transmisión correcta de los datos entre el manipulador y el software.

Implementación en MATLAB

En el software de control desarrollado en MATLAB, se implementó un cliente TCP para conectarse al servidor TCP del ESP32. La conexión se establece utilizando la dirección IP y el puerto del ESP32 previamente configurado. A través de esta

conexión, MATLAB puede enviar y recibir datos al manipulador, permitiendo la monitorización y control de los sensores.

```
% Intenta crear una nueva conexión TCP
app.tcpClient = tcpclient('192.168.7.239', 8888);
disp('Conectado al ESP32.');
```

Figura 30: Fragmento de código configuración TCP Matlab

Intercambio de datos

La comunicación entre el ESP32 y MATLAB se basó en el intercambio de datos en formato JSON. Cuando el usuario selecciona un sensor específico en la interfaz de MATLAB y presiona el botón para obtener datos, se envía el nombre del sensor al ESP32. El ESP32 procesa la solicitud y devuelve los datos del sensor en formato JSON, proporcionando una forma estructurada y eficiente de intercambiar información.

```
% Obtener el sensor seleccionado
selectedSensor = app.SensorDropDown.Value;

% Enviar el comando al ESP32
command = [selectedSensor '\n'];
write(app.tcpClient, command);
disp(['Comando enviado al ESP32: ', command]);
```

Figura 31: Fragmento de código configuración para extraer datos de Matlab

```
// Función para crear un archivo JSON con los datos de los sensores
String createJson(float calibratedValue, int touchState, int soundValue, int lightValue, int colorDetected) {
    StaticJsonDocument<200> doc;
    doc["calibratedValue"] = calibratedValue;
    doc["touchState"] = touchState;
    doc["soundValue"] = soundValue;
    doc["lightValue"] = lightValue;
    doc["colorDetected"] = colorDetected;
    String jsonData;
    serializeJson(doc, jsonData);
    return jsonData;
}
```

Figura 32: Fragmento de código configuración para crear archivo JSON

Este enfoque facilitó la integración y permitió que MATLAB procesara los datos de manera directa. La interfaz de MATLAB se encargaría luego, de la decodificación del mensaje JSON recibido y su visualización en tiempo real, permitiendo al usuario monitorear las lecturas de los sensores seleccionados de manera eficiente. La implementación de esta conexión inalámbrica y la comunicación cliente-servidor en-

tre el manipulador y el software de control proporcionó una solución eficiente para la operación y el monitoreo de los sensores integrados en el manipulador robótico, permitiendo una interacción fluida y efectiva con el sistema.

CAPÍTULO 9

Software para comunicación del manipulador a Robotat

El software de control fue una parte fundamental en este proyecto, ya que facilitó la comunicación bidireccional entre el manipulador robótico y el ecosistema de Robotat, permitiendo la recolección y visualización de datos en tiempo real. Para este propósito, se desarrolló una interfaz gráfica en MATLAB que, mediante una conexión TCP/IP, se conectó al manipulador robótico equipado con un ESP32.

9.1. Diseño de interfaz

La interfaz gráfica fue diseñada con el propósito de ofrecer al usuario una experiencia intuitiva y sencilla para el control y monitoreo de los sensores del manipulador robótico. Se desarrolló en MATLAB App Designer, una herramienta que facilita la creación de aplicaciones interactivas mediante la generación automática del código asociado a los elementos gráficos.

La interfaz incluye los siguientes componentes:

- Botón de conexión: permite al usuario conectarse o desconectarse del manipulador robótico a través de la red wifi de Robotat.
- Menú desplegable (*dropdown*): facilita la selección del sensor a monitorear entre los disponibles en el manipulador, como el sensor de distancia, de sonido, de luz, táctil y de color.
- Indicador de conexión: permite al usuario observar el estado de la conexión entre los dispositivos, variando entre verde, para indicar estado conectado y rojo para

desconectado.

- Gráfica en tiempo real: muestra los datos obtenidos de los sensores seleccionados, a medida que se reciben los valores desde el manipulador.
- Botón para obtener datos: al ser presionado, obtiene los datos del sensor seleccionado para posteriormente ser mostrado en la gráfica, monitor e indicador.
- Monitor de datos: visualiza de forma numérica los valores de los sensores en tiempo real, lo que proporciona al usuario una visión precisa de las lecturas.
- Indicador de color de objeto: muestra en tiempo real, el color del objeto reconocido por el sensor, cambiando entre rojo, verde o azul.

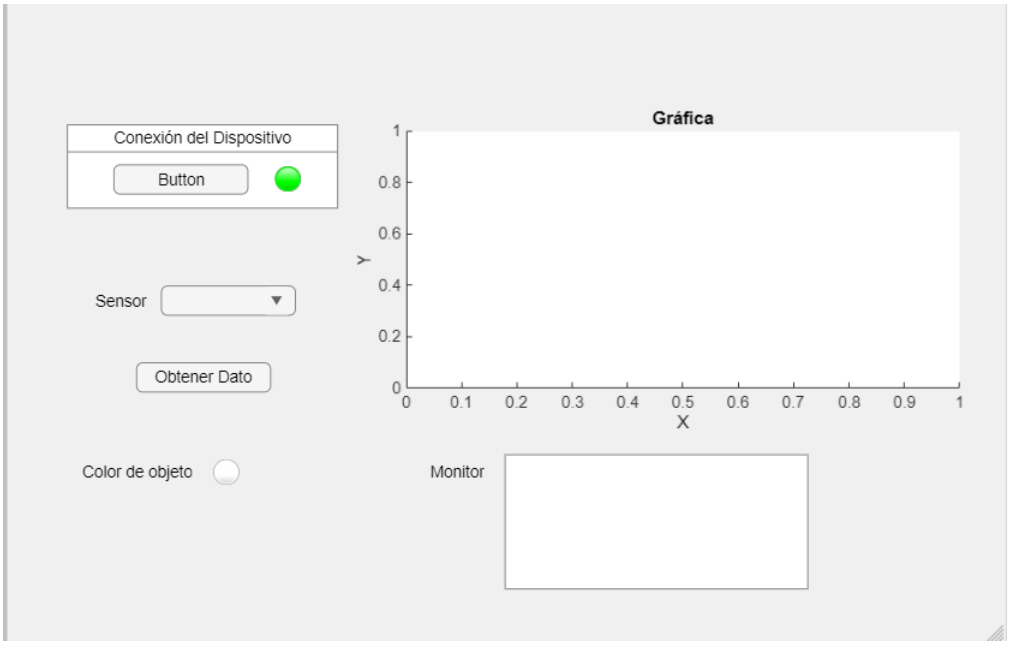


Figura 33: Primer diseño de interfaz

El diseño de la interfaz se centró en permitir una interacción fluida con el manipulador, manteniendo la simplicidad para que los usuarios puedan seleccionar sensores y observar los datos de manera rápida y eficaz.

9.2. Implementación de interfaz

La implementación de la interfaz fue realizada utilizando el lenguaje de programación MATLAB, aprovechando la librería tcpclient para establecer la comunicación con el ESP32.

9.2.1. Conexión con el manipulador

El primer paso en la implementación fue crear una función que permitiera la conexión entre MATLAB y el manipulador. Utilizando la función tcpclient, el software de control se conectó al servidor TCP configurado en el ESP32, esto se activó a través del botón de conexión dentro de la interfaz gráfica, que mostraba el estado de la conexión con el indicador ubicado a la par de él.

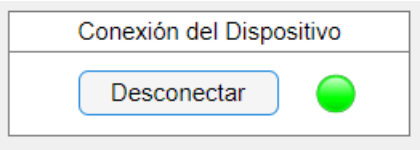


Figura 34: Sección de conexión del dispositivo

Una vez establecida la conexión, la interfaz permitió la interacción con el manipulador, habilitando la selección de sensores y el intercambio de datos.

Selección de sensores y monitoreo

Para facilitar el monitoreo de los diferentes sensores del manipulador, se implementó un menú desplegable que permite al usuario elegir el sensor que desea activar. Cada sensor, una vez seleccionado, envía una solicitud al ESP32, que procesa y devuelve los datos correspondientes en formato JSON.

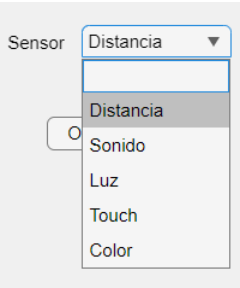


Figura 35: Desplegable integrado en la interfaz

Visualización en tiempo real

La visualización de los datos obtenidos de los sensores fue una de las características clave de la interfaz. Se implementó una gráfica en tiempo real que actualiza los valores a medida que se reciben los datos desde el manipulador, permitiendo al usuario observar cambios instantáneos en las lecturas de los sensores. De esta manera, la interfaz gráfica no solo proporcionaba un control eficiente sobre el manipulador, sino

que también garantizaba la visualización de los datos en tiempo real, lo que facilitó la toma de decisiones basadas en las lecturas de los sensores.

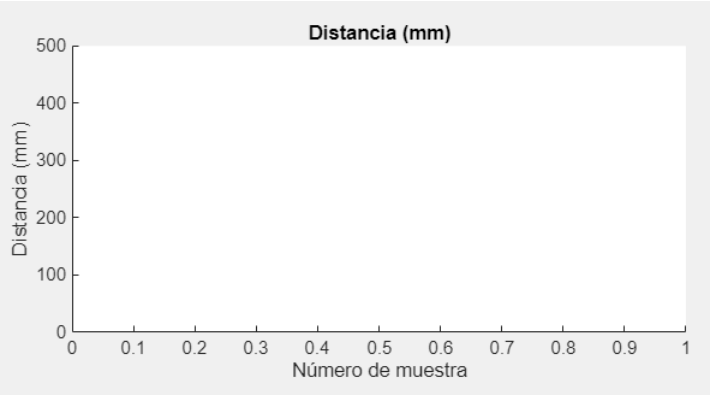


Figura 36: Gráfica personalizada para el sensor de distancia

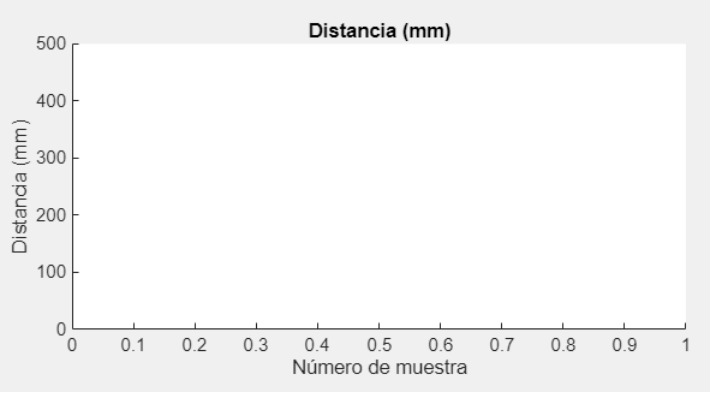


Figura 37: Gráfica personalizada para el sensor de sonido

Figura 38: Gráfica personalizada para el sensor luz

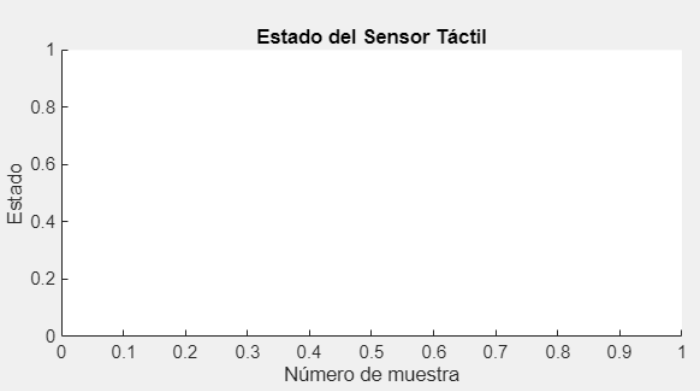


Figura 39: Gráfica personalizada para el sensor táctil

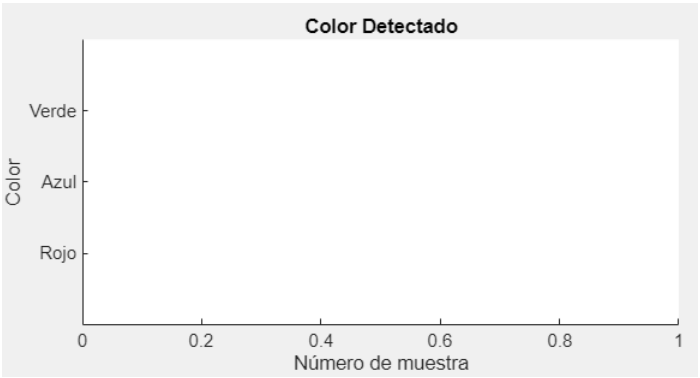


Figura 40: Gráfica personalizada para el sensor de color

Recepción y procesamiento de datos

La recepción y el procesamiento de datos constituyeron una de las fases críticas dentro del flujo de comunicación entre el manipulador y la interfaz gráfica. Este proceso se ejecutaba de manera dinámica y eficiente cada vez que el usuario solicitaba información de un sensor a través del sistema.

El flujo de datos comenzaba con una solicitud enviada por el usuario mediante la interfaz gráfica, donde se seleccionaba el sensor deseado y se activaba el botón correspondiente para la obtención de datos. Este evento generaba un comando específico que se transmitía al ESP32 a través de la conexión TCP. Una vez recibido el comando, el ESP32 identificaba el sensor seleccionado, realizaba la lectura de datos y enviaba la respuesta en formato JSON al sistema de MATLAB.

El formato JSON fue elegido debido a su estructura clara y su capacidad de transportar múltiples parámetros en un solo mensaje. Esta estructura aseguraba que los datos fueran procesados y visualizados de manera eficiente y evitaba posibles errores de interpretación durante la transmisión.

Al llegar a la interfaz, los datos eran decodificados y procesados en tiempo real. Dependiendo del sensor seleccionado, los valores se visualizaban en dos componentes principales de la interfaz:

- **Monitor de datos:** mostraba las lecturas numéricas en tiempo real, proporcionando al usuario un análisis instantáneo y directo de los valores capturados por el sensor (Figura 41 y 42).
- **Gráfica en tiempo real:** Actualizaba las lecturas visualmente, permitiendo identificar tendencias o variaciones en los datos mediante gráficos dinámicos personalizados para cada sensor. Esto facilitaba la interpretación visual y el análisis de los datos recogidos (Figura 43 y 44).

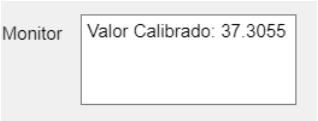


Figura 41: Ventana de monitor serie incorporado en interfaz

```
18:39:00.959 -> Sensor seleccionado: Distancia
18:39:00.959 -> Enviando datos de distancia: {"calibratedValue":37.30550003,
```

Figura 42: Fragmento de impresión monitor serie Arduino ante la solicitud de datos

El uso de un enfoque basado en eventos, como el botón para solicitar datos, aseguró que el sistema se mantuviera eficiente, evitando sobrecargar el procesamiento o la comunicación innecesariamente.

Adicionalmente, se implementó un mecanismo de mantenimiento de conexión (*keepAlive*) para asegurar la estabilidad de la comunicación entre el manipulador y la interfaz, especialmente en periodos prolongados de inactividad. Este temporizador enviaba señales periódicas al ESP32, evitando desconexiones inesperadas y asegurando que el sistema estuviera siempre listo para procesar nuevas solicitudes.

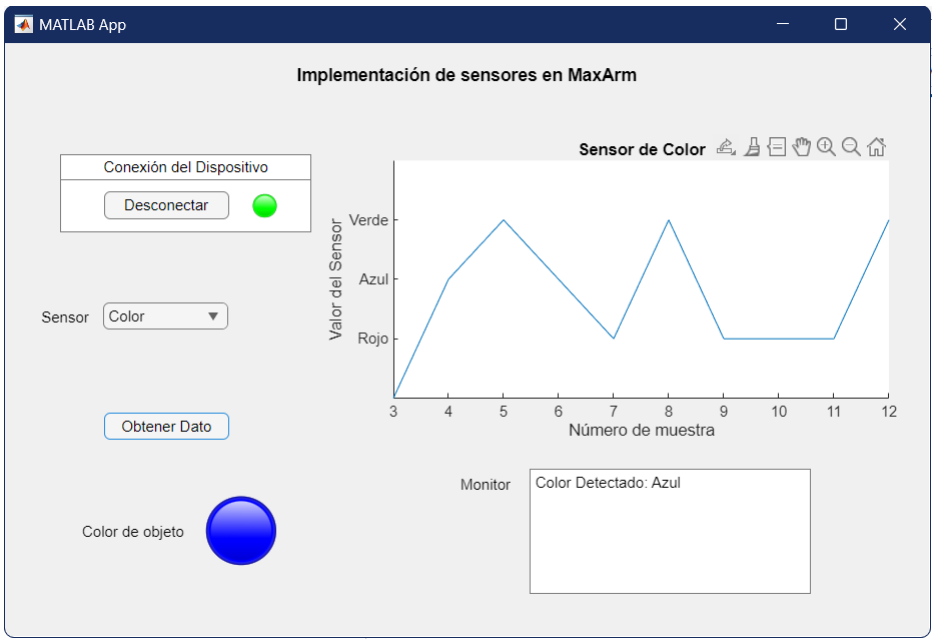


Figura 43: Interfaz en funcionamiento con toma de datos del sensor de color

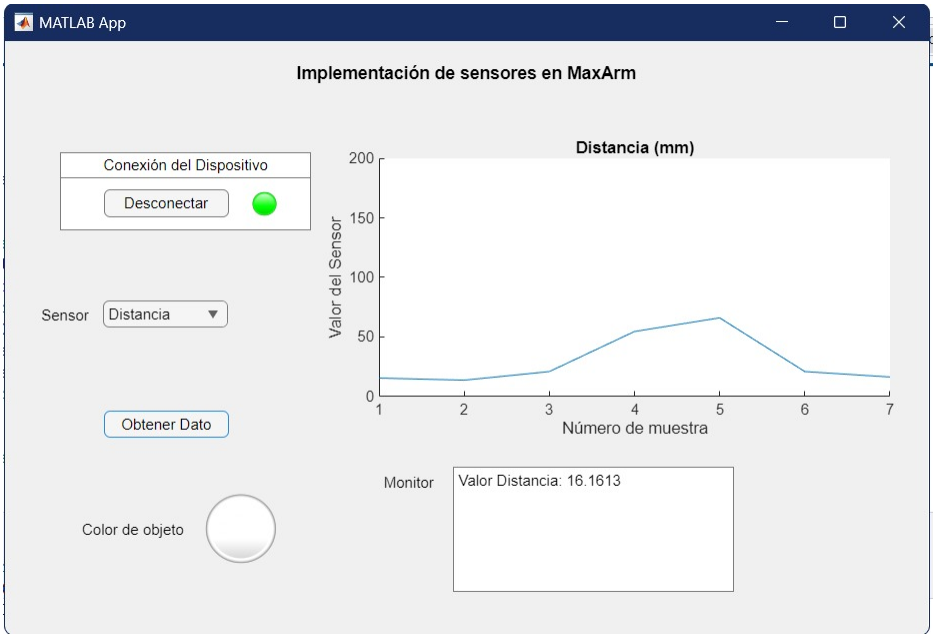


Figura 44: Interfaz en funcionamiento con toma de datos del sensor de distancia

El procesamiento también incluía ajustes específicos para cada sensor, como la conversión de unidades en sensores analógicos o la interpretación de estados binarios en sensores táctiles. Esto garantizó que los datos presentados al usuario fueran precisos, consistentes y adecuados para el análisis requerido en cada caso.

En conjunto, el flujo de recepción y procesamiento de datos implementado en este

proyecto aseguró una interacción eficiente entre el manipulador y el ecosistema Robotat, integrando la recopilación, transmisión y visualización de datos en una interfaz gráfica intuitiva y confiable.

9.3. Implementación integral de sensores y rutina de manipulación

En este apartado se unieron todas las implementaciones realizadas a lo largo de este trabajo de graduación con el propósito de cumplir las tareas de manipulación y clasificación de objetos en el manipulador robótico MaxArm dentro del ecosistema Robotat. Esto incluyó la integración de los sensores previamente calibrados y configurados, así como el aprovechamiento de la rutina de movimiento proporcionada por el fabricante, diseñada específicamente para operar el manipulador en tareas de clasificación basadas en la detección de colores, además de la integración de la conexión inalámbrica y la implementación de la interfaz gráfica para monitorear en tiempo real el estado de los sensores.

El código de control del manipulador, proporcionado por el fabricante, incluye una rutina predefinida para mover el manipulador de forma eficiente tras la detección de un objeto por los sensores de color y distancia. Este código, implementado en Arduino, se basó en una secuencia de acciones que incluyeron la detección de color, la selección de una posición predefinida en función del color detectado y la manipulación del objeto hacia la ubicación correspondiente.

9.3.1. Flujo de la rutina del fabricante

El funcionamiento de código del fabricante toma el siquiente flujo:

1. **Detección de color**: el sensor de color APDS-9960 detectó el color del objeto y clasifica el resultado en rojo, verde o azul.
2. **Determinación de posición para recogida**: luego de haber reconocido el color, el cubo se posicionó frente al sensor de distancia, en un rango establecido para poder ser recogido.
3. **Determinación de posición**: basado en el color detectado, se seleccionó una posición predeterminada para depositar el objeto. Por ejemplo:
   - Rojo: Posición 1.
   - Verde: Posición 2.
   - Azul: Posición 3.
4. **Movimiento del manipulador**: utilizando la rutina de control integrada, el manipulador mueve el objeto desde su posición inicial hacia la ubicación de

clasificación correspondiente. Esto se logra mediante la activación de motores específicos para ajustar la posición del brazo robótico.

5. **Liberación del objeto**: una vez alcanzada la posición de clasificación, el manipulador utiliza su boquilla de succión para liberar el objeto en la ubicación asignada.

El siguiente fragmento ilustra parte del código proporcionado por el fabricante para la rutina de movimiento:

```
if (detect_color) {
  setBuzzer(100); // Activar buzzer por 100 ms
  delay(1000);
  pos[0] = 0; pos[1] = -160; pos[2] = 100;
  set_position(pos, 1500); // Mover a la posición sobre el bloque de color
  delay(1500);
  pos[0] = 0; pos[1] = -160; pos[2] = 85;
  set_position(pos, 800); // Succionar el bloque de color
  Pump_on();  // Encender la bomba de succión
  delay(1000);
  pos[0] = 0; pos[1] = -160; pos[2] = 180;
  set_position(pos, 1000); // Elevar el brazo robótico
  delay(1000);
  pos[0] = x; pos[1] = y; pos[2] = 180;
  set_position(pos, 1500);  // Mover a la zona de colocación
  delay(1500);
  SetPWMServo(1, angle_pul, 800); // Configurar compensación de ángulo
  delay(200);
  pos[0] = x; pos[1] = y; pos[2] = z;
  set_position(pos, 1000);  // Mover a la zona de colocación
  delay(1000);
  Valve_on();   // Apagar la bomba y abrir válvula solenoide
  pos[0] = x; pos[1] = y; pos[2] = 200;
  set_position(pos, 1000); // Elevar el brazo robótico
  delay(1000);
  Valve_off();   // Cerrar la válvula solenoide
  go_home(1500); // Restaurar brazo robótico a posición inicial
  SetPWMServo(1, 1500, 800);
  delay(200);
  detect_color = 0;
  color_detect = true;
  ultrasound.Color(255, 255, 255, 255, 255, 255);
```

Figura 45: Fragmento del código del fabricante utilizado para cumplir con la rutina de movimiento

9.3.2. Adaptaciones y consideraciones

En este trabajo de graduación, se utilizó la rutina del fabricante como base para el control del manipulador. Sin embargo, no se realizaron modificaciones significativas a la cinemática del sistema, ya que este aspecto está siendo abordado en un trabajo de graduación paralelo enfocado específicamente en la cinemática del manipulador dentro del ecosistema Robotat. El enfoque principal de este trabajo se centró en la integración sensorial y en la calibración de los dispositivos para asegurar una clasificación precisa de los objetos. Por lo tanto, la rutina del fabricante proporcionó un marco sólido para implementar los movimientos necesarios sin requerir un rediseño completo.

CAPÍTULO 10

Conclusiones

1. **Integración funcional del manipulador MaxArm con el ecosistema Robotat.** Se logró la integración exitosa del manipulador MaxArm en el ecosistema Robotat mediante el uso de un microcontrolador ESP32 y una conexión TCP/IP. Este vínculo permitió que los sensores instalados en el manipulador fueran monitoreados y controlados en tiempo real a través de una interfaz gráfica diseñada en MATLAB, logrando así una comunicación bidireccional eficiente entre hardware y software.

2. **Calibración efectiva de sensores para condiciones específicas.** Los sensores instalados fueron calibrados para condiciones específicas del entorno Robotat, logrando una notable mejora en la precisión de sus lecturas. En particular, el sensor de distancia alcanzó un error máximo del 3.15 % tras aplicar técnicas de calibración por tramos con regresiones polinómicas y lineales, mientras que el sensor de color obtuvo una precisión del 90 % en condiciones de iluminación controlada. Esto demuestra que los métodos de calibración utilizados son efectivos para garantizar datos confiables.

3. **Identificación de limitaciones en la implementación de los sensores.** Durante las pruebas, se identificaron limitaciones importantes en los sensores. El sensor de color mostró sensibilidad a cambios en la iluminación ambiental, mientras que el sensor de distancia presentó interferencias cuando se encontraba dentro del ecosistema Robotat debido a señales externas.

4. **Desarrollo e implementación de software eficiente para monitoreo y control.** La interfaz gráfica desarrollada permitió seleccionar sensores, monitorear sus datos en tiempo real y visualizar lecturas mediante gráficos dinámicos. Esto no solo facilitó el uso del manipulador, sino que también aseguró que las lecturas fueran interpretadas correctamente, convirtiéndose en una herramienta indispensable para el control y análisis del sistema.

5. **Utilización de la rutina proporcionada por el fabricante para tareas**

de clasificación. Se empleó la rutina de movimiento provista por el fabricante del MaxArm para realizar las tareas de clasificación de objetos detectados por los sensores. Aunque esta rutina no fue modificada significativamente, su implementación permitió cumplir con las tareas de manipulación y clasificación. Esto demuestra que el prototipo puede adaptarse fácilmente a soluciones preconfiguradas.

6. **Contribuciones al desarrollo de manipuladores en entornos avanzados.** Este trabajo sienta las bases para futuras implementaciones de manipuladores en ecosistemas avanzados, destacando la importancia de la comunicación inalámbrica y la integración de sensores para la clasificación de objetos. Además, proporciona un marco modular que puede ser extendido para tareas más complejas, como la incorporación de algoritmos de inteligencia artificial o el uso de visión asistida por computadora.

CAPÍTULO 11

Recomendaciones

11.1. Recomendaciones

1. **Mejorar la robustez de los sensores en entornos dinámicos.** Se recomienda realizar ajustes adicionales en los sensores, especialmente en el de color y distancia, para minimizar las interferencias y aumentar su precisión en ambientes menos controlados. El desarrollo de sistemas de aislamiento más efectivos podría ser clave para resolver este desafío.

2. **Optimización del software de control.** Se sugiere explorar otras tecnologías de desarrollo de interfaces gráficas que permitan una mayor escalabilidad y funcionalidad, como el uso de plataformas web o aplicaciones móviles. Esto ampliaría la accesibilidad del sistema y mejoraría la experiencia del usuario.

3. **Pruebas en escenarios reales.** Para garantizar una mayor aplicabilidad del sistema, se recomienda realizar pruebas operativas en entornos industriales o educativos reales. Esto permitiría evaluar la robustez del sistema bajo condiciones más variadas y exigentes.

4. **Extensión de funcionalidades del MaxArm.** Aunque este proyecto se enfocó en tareas de clasificación y monitoreo, se recomienda explorar futuras implementaciones que incluyan la integración de algoritmos de inteligencia artificial para mejorar la toma de decisiones del manipulador.

5. **Colaboración interdisciplinaria.** Finalmente, se sugiere fomentar la colaboración con otros proyectos dentro del ecosistema Robotat para aprovechar sinergias, especialmente en áreas como la cinemática del manipulador, que se encuentra en desarrollo paralelo.

CAPÍTULO 12

Bibliografía

[1] C. Perafán Montoya, "Robotat: un ecosistema robótico de captura de movimiento y comunicación inalámbrica," es, Tesis doct., Universidad del Valle de Guatemala, 2022.

[2] J. D. Pellecer Orellana, "Diseño e implementación de un paquete de herramientas de software para controlar inalámbricamente un manipulador serial R17 dentro de un ecosistema basado en captura de movimiento," es, Tesis doct., Universidad del Valle de Guatemala, 2022.

[3] A. Noman, A. Eva, T. Yeahyea y R. Khan, "Computer Vision-based Robotic Arm for Object Color, Shape, and Size Detection," *Journal of Robotics and Control (JRC)*, vol. 3, págs. 180-186, feb. de 2022. DOI: 10.18196/jrc.v3i2.13906.

[4] L. Jie, T. P. Sen, N. M. A. Ghani y M. F. B. Abas, "Automatic Control of Color Sorting and Pick/Place of a 6-DOF Robot Arm," *Journal Européen des Systèmes Automatisés*, 2021. dirección: https://api.semanticscholar.org/CorpusID:237502640.

[5] Hiwonder, *Glowing ultrasonic sensor*. dirección: https://www.hiwonder.com/collections/module-sensor-sensor/products/glowing-ultrasonic-sensor.

[6] Hiwonder, *Color sensor*. dirección: https://www.hiwonder.com/products/color-sensor?_pos=1&_sid=61aa3351f&_ss=r.

[7] Hiwonder, *Touch sensor: Hiwonder robot sensor compatible with Arduino*. dirección: https://www.hiwonder.com/products/touch-sensor?_pos=1&_sid=f712999de&_ss=r.

[8] Hiwonder, *Sound sensor: Hiwonder robot sensor compatible with Arduino*. dirección: https://www.hiwonder.com/products/sound-sensor?_pos=1&_sid=beadfc161&_ss=r.

[9] Hiwonder, *Light sensor: Hiwonder robot sensor compatible with Arduino*. dirección: https://www.hiwonder.com/products/light-sensor?_pos=1&_sid=8ad24cb17&_ss=r.

[10] Hiwonder, *Infrared receiver: Hiwonder Robot Sensor compatible with Arduino*. dirección: https://www.hiwonder.com/products/infrared-receiver?_pos=1&_sid=05a89cf16&_ss=r.

[11] E. Systems, *ESP32-WROOM-32 Datasheet*, ver. 3.7, Available online, feb. de 2022. dirección: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf (visitado 23-11-2024).

[12] *JSON*, es. dirección: https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/JSON.

[13] I. E. Team. "What Is Regression Analysis? (Types, Importance and Uses)." Accessed on 2024-11-23. (oct. de 2023), dirección: https://www.indeed.com/career-advice/career-development/what-is-regression-analysis (visitado 23-11-2024).

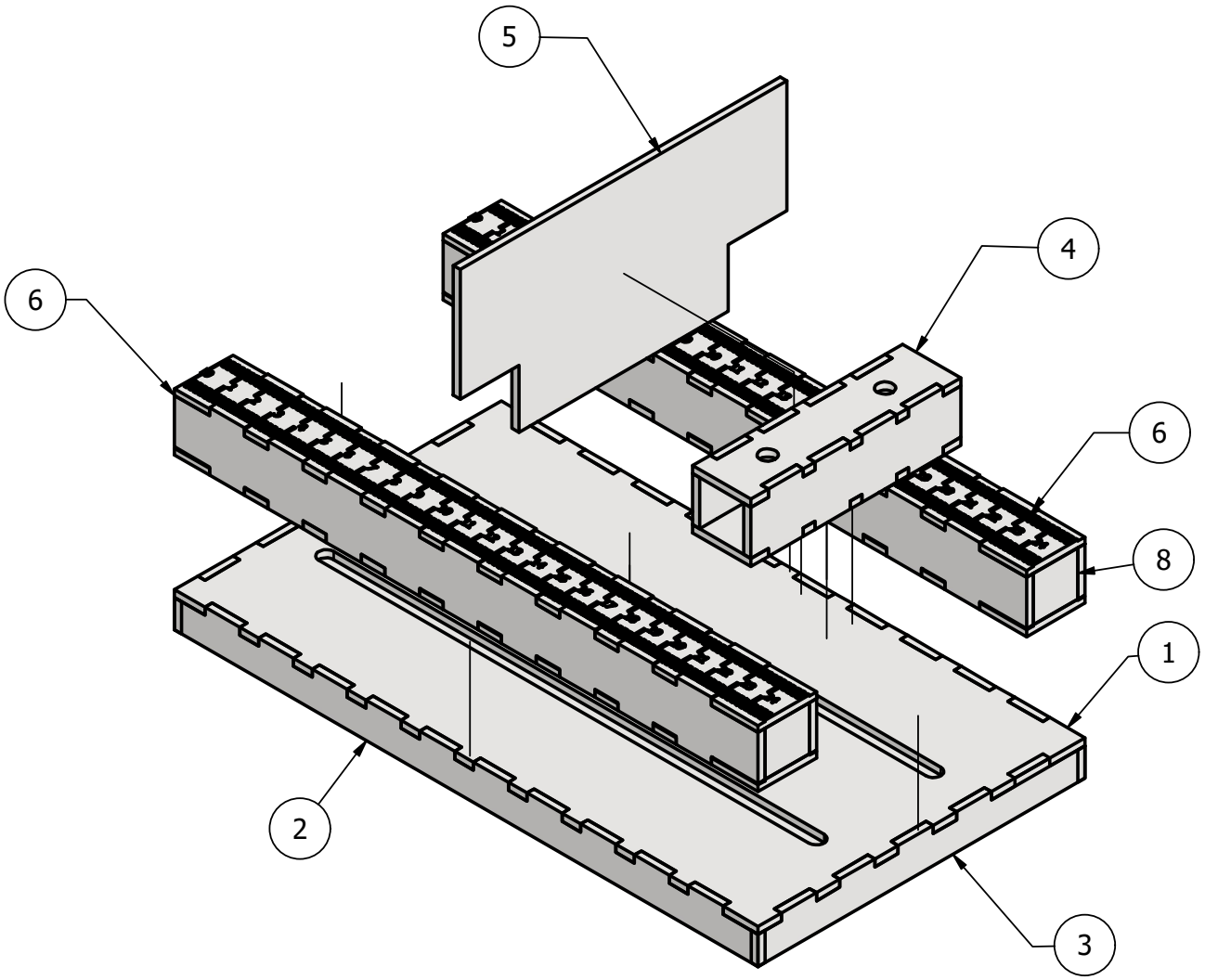
CAPÍTULO 13

Anexos

A continuación, se agregan los planos correspondientes a la regla ajustable y el capuchón para el sensor de distancia:

13.1. Planos de la regla ajustable

LISTA DE PIEZAS			
ELEMENTO	CTDAD	Nº DE PIEZA	DESCRIPCIÓN
1	1	Base_superior_Caja	Pieza de MDF de 3mm
2	2	Base_Lateral1_Caja	Pieza de MDF de 3mm
3	2	Bse_Lateral2_Caja	Pieza de MDF de 3mm
4	1	Soporte_tornillos	Pieza de MDF de 3mm
5	1	Cara_medicion	Pieza de MDF de 3mm
6	2	Regla_lateral	Pieza de MDF de 3mm
7	1	Base_inferior_caja	Pieza de MDF de 3mm
8	4	Tapones_regla	Pieza de MDF de 3mm



TODAS LAS MEDIDAS EN MILÍMETROS SALVO SE INDIQUE LO CONTRARIO. INTERPRETAR SEGÚN ASME Y14.5 - 2009.

TOLERANCIAS GENERALES
LINEAL .X ± N/A
.XX ± N/A
.XXX ± N/A
ANGULAR ± N/A
FRACCIONES ± N/A
RUGOSIDAD SUPERFICIAL N/A

MATERIAL: MDF 3mm
TRATAMIENTO: N/A
MASA: N/D

TERCER ÁNGULO DE PROYECCIÓN

DIBUJADO POR	Melanie Morales	FECHA 04/03/2024
DISEÑADO POR	Melanie Morales	FECHA 04/03/2024
REVISADO POR	Ing. M. Zea	FECHA 25/11/2024
APROBADO POR	Ing. M.Zea	FECHA 25/11/2024

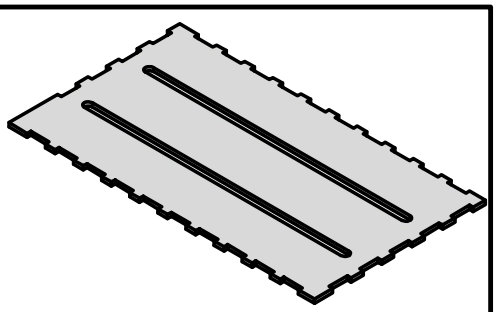
NOMBRE / NÚMERO DE PROYECTO: Regla para sensor distancia

UNIVERSIDAD DEL VALLE DE GUATEMALA
18 avenida, 11-95 zona 15, Vista Hermosa III
Guatemala, Guatemala 01015
PBX: (502) 2634-0336 / 40
info@uvg.edu.gt

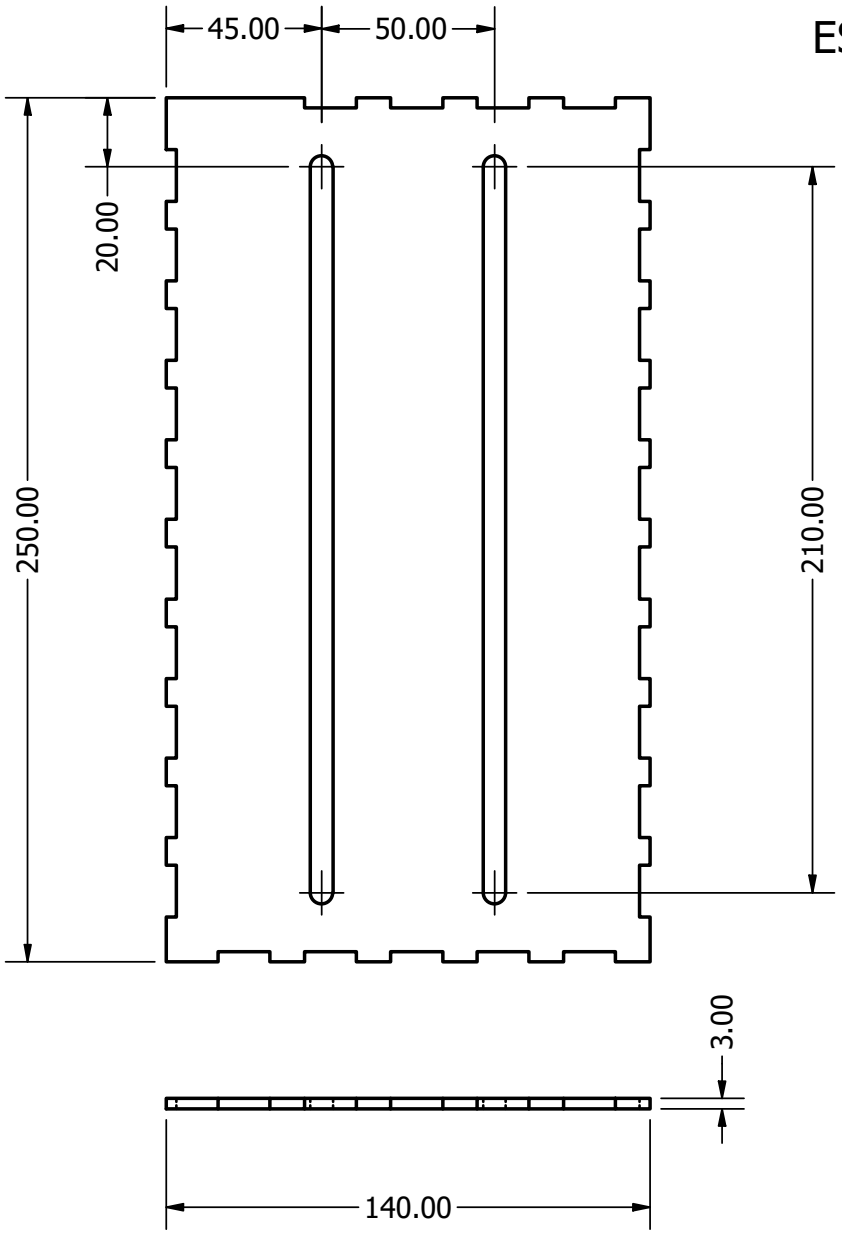
TÍTULO: Regla Ajustable

NÚMERO DE DIBUJO: 201150

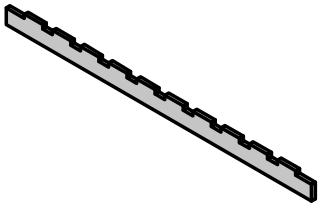
FORMATO A4 | ESCALA: 1 / 2 | UNIDADES: mm | PÁGINA 1 DE 9 | REV A



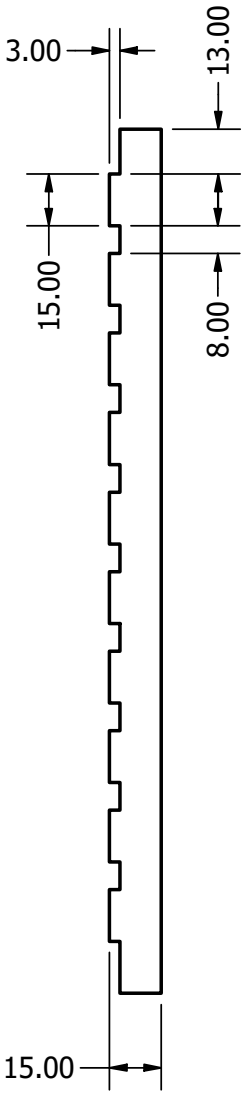
VISTA13
ESCALA 1 / 4

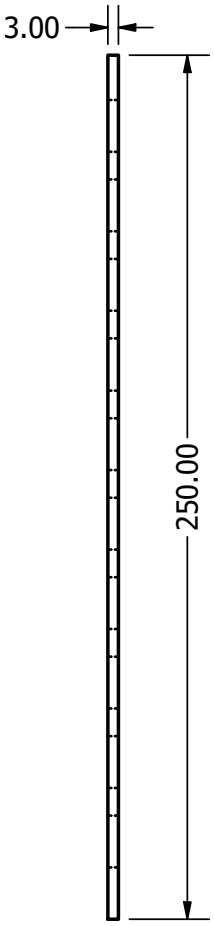


TÍTULO: Base_superior_Caja Pieza 1		
NÚMERO DE DIBUJO: 201150		
FORMATO A	ESCALA: 1:2	UNIDADES: mm
	PÁGINA 2 DE 9	REV A

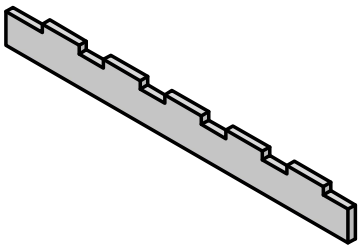


VISTA16
ESCALA 1 / 4

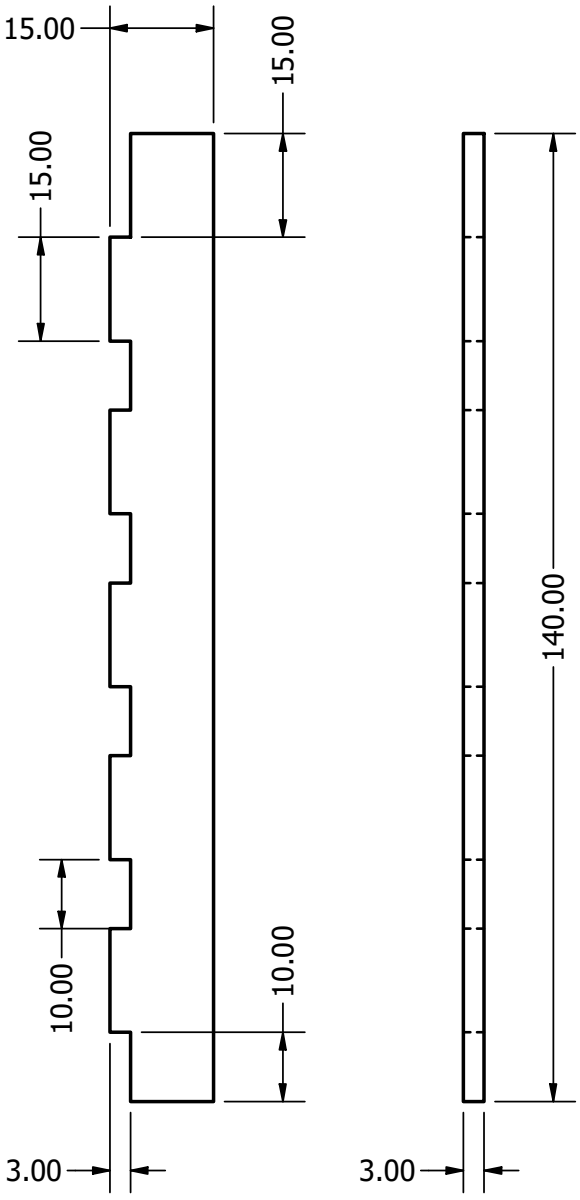




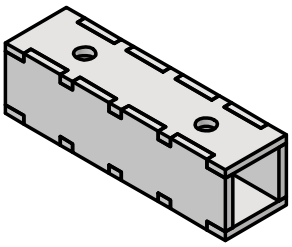
TÍTULO: Base_Lateral1_Caja Piezas 2		
NÚMERO DE DIBUJO: 201150		
FORMATO A	ESCALA: 1:2	UNIDADES: mm
	PÁGINA 3 DE 9	REV A



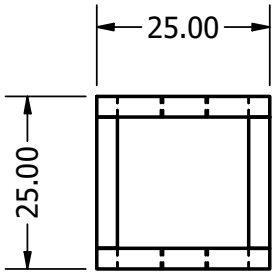
VISTA19
ESCALA 1 / 2

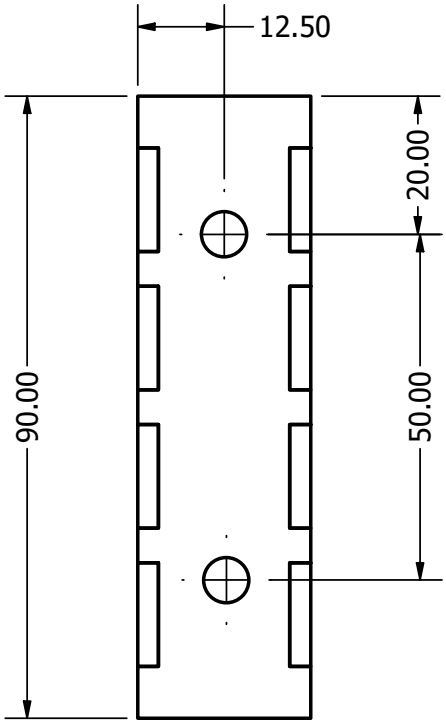


TÍTULO: Base_Lateral2_Caja Pieza 3		
NÚMERO DE DIBUJO: 201150		
FORMATO A	ESCALA: 1:1	UNIDADES: mm
	PÁGINA 4 DE 9	REV A

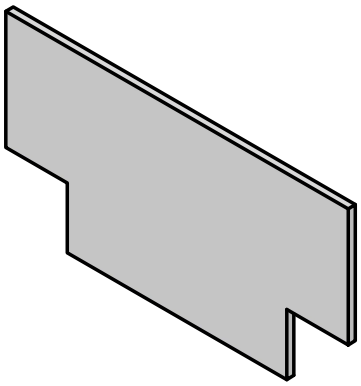


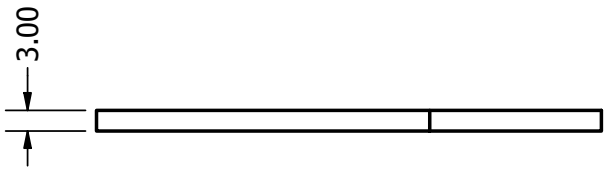
VISTA27
ESCALA 1 / 2



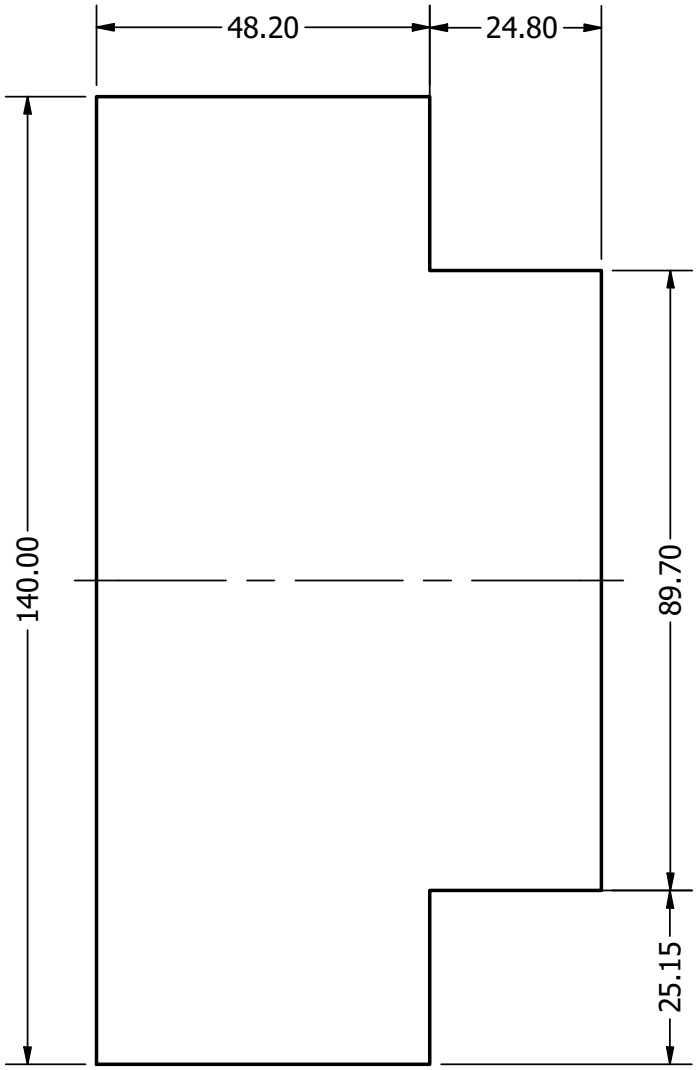


TÍTULO: Soporte_tornillos Pieza 4		
NÚMERO DE DIBUJO: 201150		
FORMATO A	ESCALA: 1:1	UNIDADES: mm
	PÁGINA 5 DE 9	REV A

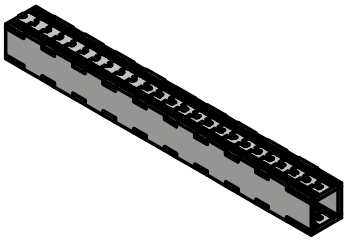




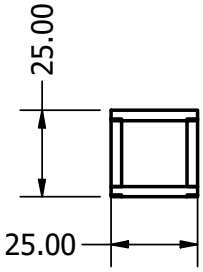
VISTA30
ESCALA 1 / 2

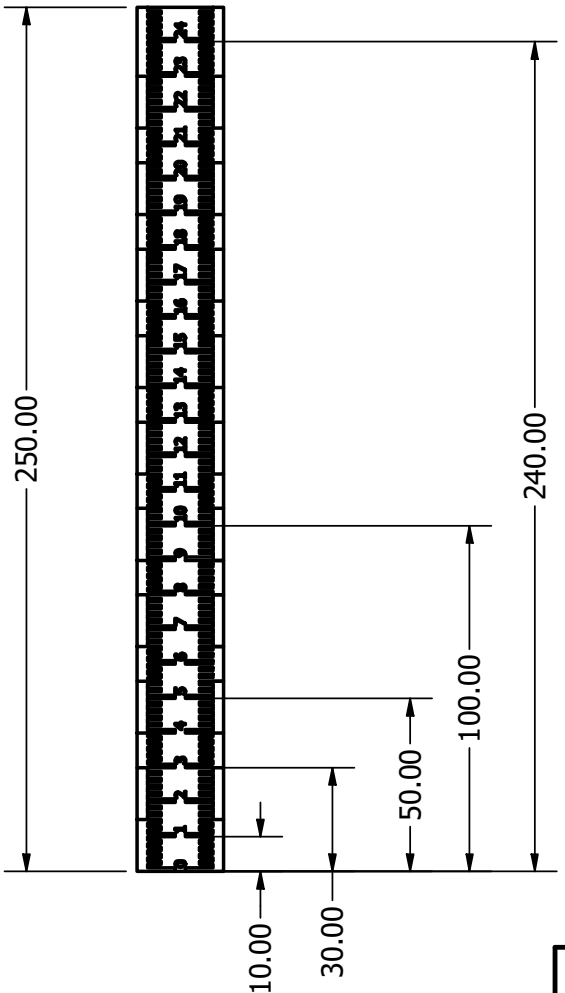


TÍTULO: Cara_medicion Pieza 5		
NÚMERO DE DIBUJO: 201150		
FORMATO A	ESCALA: 1:1	UNIDADES: mm
	PÁGINA 6 DE 9	REV A

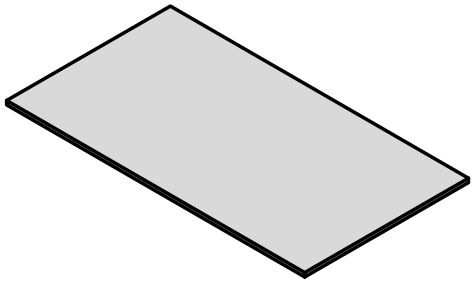


VISTA33
ESCALA 1 / 4

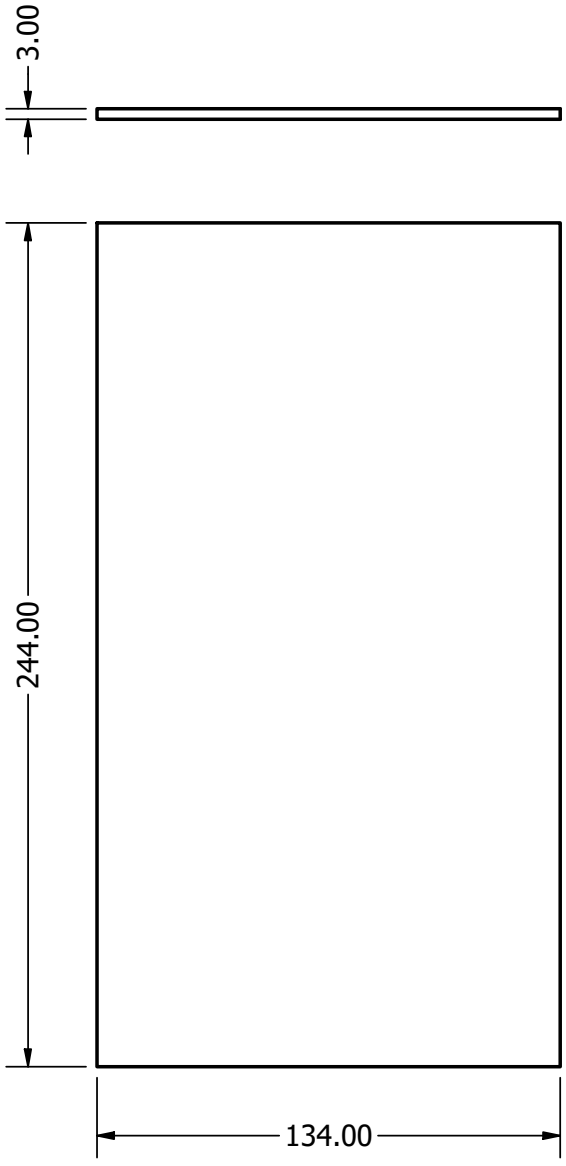




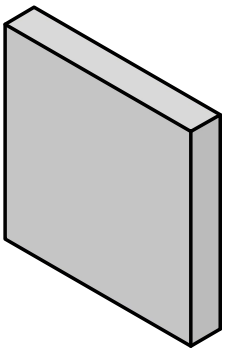
TÍTULO: Regla_lateral Pieza 6		
NÚMERO DE DIBUJO: 201150		
FORMATO A	ESCALA: 1:2	UNIDADES: mm
	PÁGINA 7 DE 9	REV A



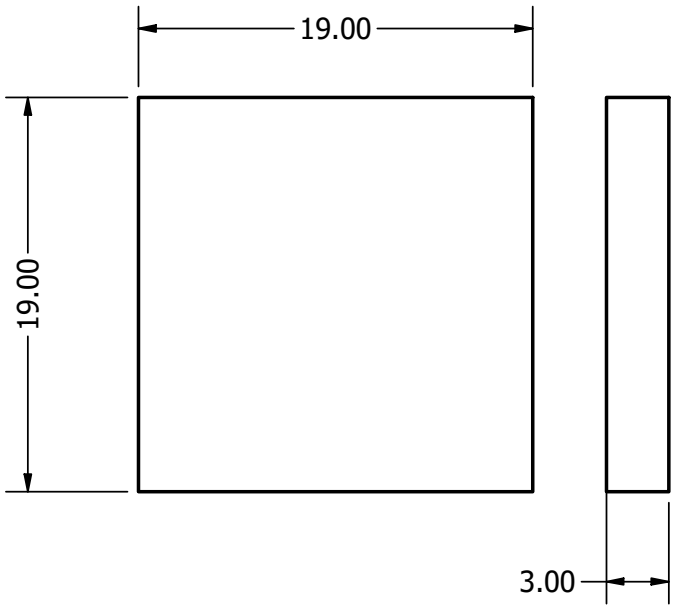
VISTA36
ESCALA 1 / 4



TÍTULO: Base_inferior Pieza 7		
NÚMERO DE DIBUJO: 201150		
FORMATO A	ESCALA: 1:2	UNIDADES: mm
	PÁGINA 8 DE 9	REV A



VISTA39
ESCALA 2 : 1



TÍTULO: Tapones_regla Pieza 8		
NÚMERO DE DIBUJO: 201150		
FORMATO A	ESCALA: 2:1	UNIDADES: mm
	PÁGINA 9 DE 9	REV A

Figura 46: Renderizado de la regla ajustable

Figura 47: Renderizado de la regla ajustable

Figura 48: Renderizado de la regla ajustable

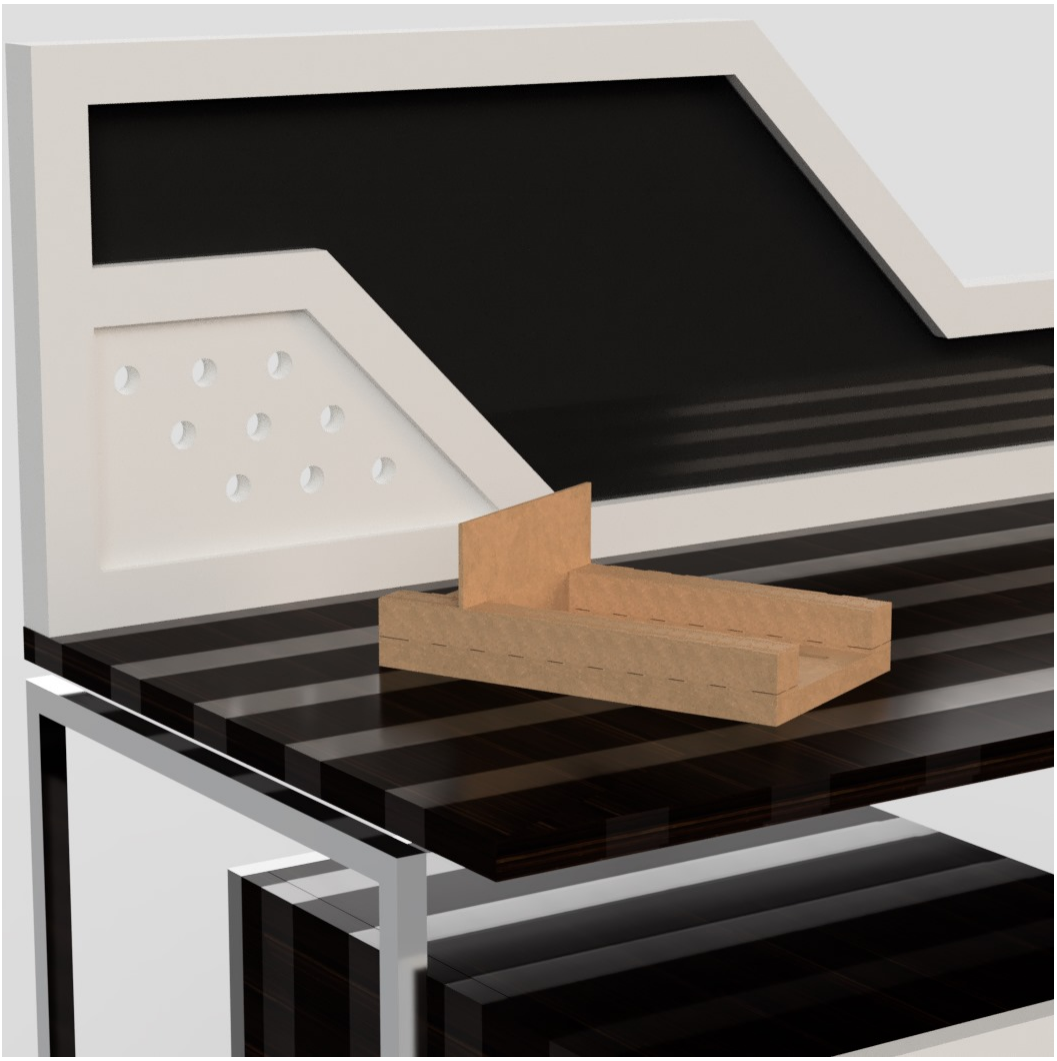
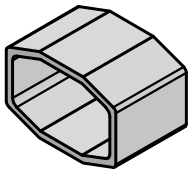
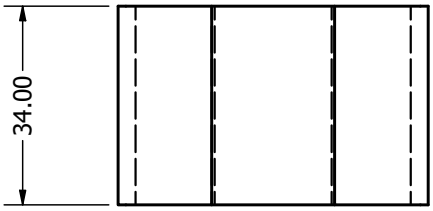


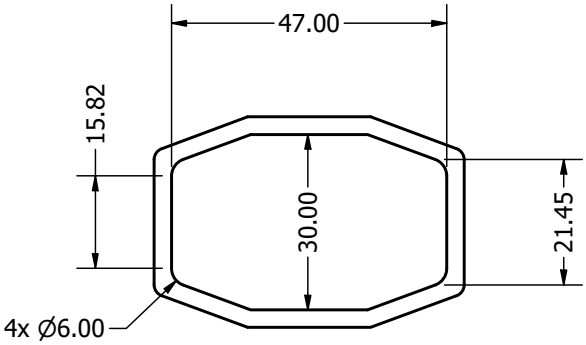
Figura 49: Renderizado de la regla ajustable

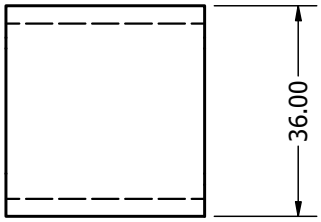
13.2. Plano del capuchón



VISTA43
ESCALA 1 / 2







TÍTULO: Glowy Sensor Fabricado en PLA		
NÚMERO DE DIBUJO: 201150		
FORMATO A	ESCALA: 1:1	UNIDADES: mm
	PÁGINA 10 DE 10	REV A

13.3. Repositorio y video

A continuación, se adjunta el vínculo para poder acceder a los archivos, así como también el enlace para acceder al video del prototipo.

1. https://github.com/melanie-morales2/Integraci-n-de-Sensores-en-el-Manipulador-Rob-tico-Maxarm-dentro-del-Ecosistema-ROBOTAT
2. https://youtu.be/3T1MtLltr7k