

**DISPOSITIVO DE CONTROL REMOTO PARA LA DETECCIÓN Y
CORRECCIÓN DE ERRORES EN EL PROCESO DE IMPRESIÓN 3D FDM EN
TIEMPO REAL**

**HENRY JOSÉ REQUENA MOLINA
KELVIN JOSEPH POZUELO MORALES**

**UNIVERSIDAD AUTÓNOMA DEL CARIBE
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA MECATRÓNICA
BARRANQUILLA - COLOMBIA
2023**

**DISPOSITIVO DE CONTROL REMOTO PARA LA DETECCIÓN Y
CORRECCIÓN DE ERRORES EN EL PROCESO DE IMPRESIÓN 3D FDM EN
TIEMPO REAL**

**HENRY JOSÉ REQUENA MOLINA
KELVIN JOSEPH POZUELO MORALES**

**Trabajo de grado presentado para optar al título de
Ingeniero Mecatrónico**

**ASESORES DISCIPLINARES:
ING. CARLOS GABRIEL DÍAZ SÁENZ, PhD (c).
ING. JEAN PIERRE COLL VELÁZQUEZ, MSc.**

**UNIVERSIDAD AUTÓNOMA DEL CARIBE
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA MECATRÓNICA
BARRANQUILLA - COLOMBIA
2023**

Nota de aceptación

Firma del jurado 1

Firma del jurado 2

DEDICATORIA

Dedico este trabajo a las personas con las que llegué a este país con sueños y metas: mis padres, Adriana Molina y Henry Requena, junto con mi hermano Adrian, doy las gracias por su amor y por el apoyo incondicional que me dieron durante todo este proceso. De forma muy grata, dedico este trabajo también a mi familia y amigos en Venezuela, en especial a mis abuelos Regulo y Dilia, así como en memoria de mis abuelos Aparicio y Herminia quienes en conjunto apoyaron en mi formación como persona.

Agradezco profundamente el enriquecimiento académico de mi profesor Alfredo Saldeño. Y a mi querida Melissa López, por tu apoyo incondicional, tu comprensión y tu cariño a lo largo de esta etapa de mi vida.

A todos ustedes, mi gratitud es infinita. Este logro es también el suyo, y les dedico este trabajo con todo mi cariño y reconocimiento.

Henry

Dedico este trabajo a mis padres, Josefa Morales y Julio Pozuelo, quienes siempre han sido mi fuente de inspiración, amor y apoyo inquebrantable a lo largo de mi vida. A mi abuela, Andrea Bornacelly, cuyo apoyo y cariño fueron base fundamental en mi camino. A mis queridas hermanas, por su apoyo incondicional y por compartir este viaje conmigo. A mi familia en general, por su aliento constante y por ser mi roca en los momentos más desafiantes. Este logro también es de ustedes, y espero que este proyecto refleje la gratitud y el amor que siento hacia todos ustedes.

Kelvin P.

TABLA DE CONTENIDO

LISTA DE FIGURAS	7
LISTA DE TABLAS.....	10
GLOSARIO.....	12
RESUMEN	14
ABSTRACT	15
INTRODUCCIÓN	16
1. PLANTEAMIENTO DEL PROBLEMA.....	18
1.1. FORMULACIÓN DEL PROBLEMA.....	18
1.2. JUSTIFICACIÓN Y ALCANCE.....	20
2. OBJETIVOS.....	23
2.1. OBJETIVO GENERAL	23
2.2. OBJETIVOS ESPECÍFICOS	23
3. MARCO DE REFERENCIA	24
3.1. ESTADO DEL ARTE	24
3.2. MARCO TEÓRICO.....	30
3.2.1 IMPRESIÓN 3D FDM (Fused Deposition Modeling).....	30
3.2.2 FORMATOS DE MODELOS PARA IMPRESIÓN 3D	31
3.2.3 LENGUAJE DE IMPRESORAS 3D	32
3.2.4 ERRORES EN LA IMPRESIÓN 3D FDM	34
4. PROCEDIMIENTO METODOLÓGICO.....	42
4.1. METODLOGÍA	42
4.1.1. Etapa 1. Entrenamiento de Red Neuronal	43
4.1.2. Etapa 2. Diseño del Sistema de Control	45
4.1.3. Etapa 3. Integración.....	47
4.1.4. Etapa 4. Validación del Dispositivo	48
4.2. TIPO DE ESTUDIO.....	49
4.2.1. CUANTITATIVA EXPERIMENTAL	49
4.2.2. POBLACIÓN Y MUESTRA	49
4.3. CRONOGRAMA – PLAN DE TRABAJO.....	50
5. PRESUPUESTO.....	51
5.1. PRESUPUESTO GENERAL.....	51

5.2.	PERSONAL CIENTÍFICO Y DE APOYO	51
5.3.	CONSULTORIA ESPECIALIZADA	52
5.4.	MATERIALES, INSUMOS Y EQUIPOS	52
6.	PRESENTACIÓN Y ANÁLISIS DE RESULTADOS	53
6.1.	DISEÑO DEL PROTOTIPO	53
6.1.1.	DISEÑO CAD	53
6.1.2.	ENSAMBLAJE	54
6.2.	DISEÑO DEL DISPOSITIVO FINAL.....	55
6.2.1.	Fabricación de Carcasa del Dispositivo.....	55
6.2.2.	Modelo Entrenado para Detección de Fallos.....	56
6.2.3.	Diseño de Sistema de Control	61
6.2.4.	Integración del Sistema de Control con Modelo de Detección	69
6.3.	MATERIALES	73
6.3.1.	Raspberry Pi 4 Model B.....	73
6.3.2.	Impresora 3D Ender 3	74
6.3.3.	Filamento de Impresión PLA.....	75
6.3.4.	Cámara Web HD 1080P	75
6.4.	RECOLECCIÓN DE DATOS.....	76
6.4.1.	Métricas del Modelo Entrenado	76
6.4.2.	Lectura de Temperaturas y Envío por MQTT	77
6.4.3.	Métricas de Broker MQTT	77
6.5.	ANÁLISIS DE RESULTADOS.....	79
6.5.1.	ANÁLISIS DE LAS PRUEBAS REALIZADAS POR EL DISPOSITIVO FINAL	79
	CONCLUSIONES Y RECOMENDACIONES	83
	BIBLIOGRAFÍA	84
	ANEXOS	90

LISTA DE FIGURAS

Figura 1. Mapa Global de Adopción de Impresión 3D por Región (2019)	20
Figura 2. Mapa Global de Adopción de Impresión 3D por Región (2022)	21
Figura 3. Sistema de Modelado por Deposito Fundido	30
Figura 4. Formato de Impresión STL	31
Figura 5. Ejemplo de Warping	34
Figura 6. Ejemplo de Stringing	34
Figura 7. Ejemplo de LayerShifting	35
Figura 8. Ejemplo de Spaghetti	35
Figura 9. Protocolo MQTT.....	36
Figura 10. Node Red	37
Figura 11. Diagrama de Bloques Metodología de Desarrollo Dispositivo.	42
Figura 12. Ejemplo de modificaciones para generación de Errores	43
Figura 13. Etiquetado con LabelImg	44
Figura 14. Diagrama de Conexión Serial	45
Figura 15. Esquema de Interfaz.....	46
Figura 16. CAD de Raspberry Pi	53
Figura 17. Tapa Inferior	53
Figura 18. Tapa Superior.....	54
Figura 19. Ensamblaje CAD de Dispositivo PrintGuard.....	54
Figura 20. Dimensiones Generales de Dispositivo PrintGuard.....	55
Figura 21. Piezas modelo para generación de fallos	56
Figura 22. Piezas con Errores para Dataset.....	57
Figura 23. Tamaño del Dataset	57
Figura 24. Configuración de LabelImg.....	58
Figura 25. Etiquetado de Imágenes.....	58
Figura 26. Sintaxis de archivo de etiquetado Yolo.....	59
Figura 27. Verificación de existencia de imágenes.....	59
Figura 28. Verificación de existencia de etiquetas.....	59
Figura 29. Segmentación de Dataset	60
Figura 30. Archivo de configuración para entrenamiento	60
Figura 31. Función de Escritura Serial.....	61

Figura 32.	Estructura de Software PrintGuard	62
Figura 33.	Flujo de Interfaz de Gráfica	62
Figura 34.	Usuarios Registrados en Firebase.....	63
Figura 35.	Método para Inicio de Sesión en Interfaz Gráfica	63
Figura 36.	Método para Registro de Usuarios en interfaz Gráfica	63
Figura 37.	Scaffolding PrintGuard (Quasar project)	64
Figura 38.	Interfaz de Control y Monitoreo PrintGuard	64
Figura 39.	Apartado de Options Interfaz PrintGuard.....	65
Figura 40.	Integración de Grafica Embebida en componente Web	65
Figura 41.	Nodos de Envío de Temperatura a Base de Datos InfluxDB	66
Figura 42.	Captura de Mensajes Provenientes de Dispositivo PrinGuard	66
Figura 43.	Control de Detecciones	67
Figura 44.	Método para Control de Temperatura.....	67
Figura 45.	Broker MQTT – EMQX	68
Figura 46.	Tópicos MQTT implementados.....	69
Figura 47.	Función para cargar modelo de detección.....	69
Figura 48.	Configuración e Iniciación del Modelo	70
Figura 49.	Implementación del Modelo para Obtener las Predicciones	70
Figura 50.	Interfaz Web de Configuración	71
Figura 51.	Lectura de Configuración de Puerto Serial y Cámara.....	71
Figura 52.	Código de subprocesos main.py.....	72
Figura 53.	Raspberry Pi 4 Model B	73
Figura 54.	Creality Ender 3	74
Figura 55.	Filamento PLA	75
Figura 56.	Cámara Web.....	75
Figura 57.	Matriz de Confusión.....	76
Figura 58.	Histórico de Lectura de Temperaturas.....	77
Figura 59.	Comportamiento de Suscripciones a Tópicos	77
Figura 60.	Comportamiento de Datos Enviados	78
Figura 61.	Tópicos habilitados y en funcionamiento	78
Figura 62.	IoU	80
Figura 63.	IoU de Warping.....	81

Figura 64. IoU de Stringing	81
Figura 65. IoU de spaghetti.....	82

LISTA DE TABLAS

Tabla 1.	Instrucciones en código G para impresión 3D	33
Tabla 2.	Cronograma – Plan de Trabajo.....	50
<i>Tabla 3.</i>	<i>Presupuesto general.....</i>	<i>51</i>
<i>Tabla 4.</i>	<i>Costo personal científico.....</i>	<i>51</i>
<i>Tabla 5.</i>	<i>Costo personal de apoyo.....</i>	<i>51</i>
<i>Tabla 6.</i>	<i>Costo consultoría especializada.....</i>	<i>52</i>
<i>Tabla 7.</i>	<i>Costo materiales e insumos.....</i>	<i>52</i>
<i>Tabla 8.</i>	<i>Costo trabajo de campo.....</i>	<i>52</i>
<i>Tabla 9.</i>	<i>Costo equipos usados</i>	<i>52</i>
Tabla 10.	Fabricación de Carcasa de Dispositivo Final	55
Tabla 11.	Especificaciones Raspberry Pi 4 Model B	73
Tabla 12.	Especificaciones Creality Ender 3	74
Tabla 13.	Especificaciones del PLA	75

LISTA DE ANEXOS

Anexo 1. Código Main.py	90
Anexo 2. Backend de Aplicativo Web de Configuración	90
Anexo 3. Frontend de Aplicativo Web de Configuración	91
Anexo 4. Librerías utilizadas en software de control	91
Anexo 5. Lectura de Puerto Serial y Cámaras Disponibles.....	92
Anexo 6. Configuración Serial.....	92
Anexo 7. Clase para uso del api de Telegram	93
Anexo 8. Función Completa de Envío de Comandos vía Serial.....	93
Anexo 9. Funciones para comando de temperaturas.....	94
Anexo 10. Funciones para control de ejes.....	94
Anexo 11. Funciones para detener y pausar impresión.....	94
Anexo 12. Funciones de Lectura de Temperatura.....	95
Anexo 13. Función para Cargar modelo de Detección	95
Anexo 14. Función para cambio de Sintaxis Booleana de Javascript a Python..	95
Anexo 15. Configuración de Protocolo MQTT en Python	96
Anexo 16. Inicialización de Variables	97
Anexo 17. Bucle Principal – Captura de foto en tiempo real.....	97
Anexo 18. Envío de Lectura de Temperaturas en Tiempo Real	97
Anexo 19. Código de detección de Errores	98
Anexo 20. Envío de Avisos de Interfaz Gráfica.....	98
Anexo 21. Envío de Avisos vía Telegram	99
Anexo 22. Validaciones Adicionales para Cámara no Conectada	99
Anexo 23. Dispositivo Conectado	99
Anexo 24. Verificación de Conexión MQTT del Dispositivo	100
Anexo 25. Interfaz de Pruebas 1 con Node-Red	100
Anexo 26. Interfaz de Pruebas 2 con Node-Red	101
Anexo 27. Pruebas de Interfaz de Configuración	102
Anexo 28. Validación del Sistema de Control con Node-Red.....	102

GLOSARIO

Impresión 3D: La impresión 3D, también llamada manufactura por adición (inglés), es un conjunto de procesos que producen objetos a través de la adición de material en capas que corresponden a las sucesivas secciones transversales de un modelo 3D. Los plásticos y las aleaciones de metal son los materiales más usados para impresión 3D, pero se puede utilizar casi cualquier cosa, desde hormigón hasta tejido vivo [1].

Defecto de Fabricación: El defecto de fabricación se da cuando el desperfecto obedece a fallas originadas en la fase de producción o elaboración, que evidencia una discrepancia entre lo que se trazó y lo que resultó al final del proceso productivo [2].

Polímero: Los polímeros se definen como macromoléculas que se obtienen por la unión de una o más moléculas pequeñas repetidas a lo largo de una cadena. La unidad que se repite en el polímero es el monómero y la reacción por la que se forman es la reacción de polimerización [3].

Visión Computacional: La visión de computación define un campo que les permite a las computadoras usar algoritmos complejos, que pueden basarse en el deep learning o tradicional, para comprender imágenes y videos digitales, y así extraer información útil [4].

Machine Learning: Machine learning, o aprendizaje automático, es el uso de algoritmos para organizar datos, reconocer patrones y hacer que computadores puedan aprender con esos modelos y generar insights inteligentes sin necesidad de pre-programación [5].

Internet de las cosas (IoT): es el proceso que permite conectar los elementos físicos cotidianos al Internet: desde los objetos domésticos comunes, como las bombillas de luz, hasta los recursos para la atención de la salud, como los dispositivos médicos; las prendas y los accesorios personales inteligentes; e incluso los sistemas de las ciudades inteligentes [6].

Fused Deposition Modeling (FDM): La impresión 3D FDM es una tecnología de fabricación en la que una impresora 3D coloca material plástico fundido en capas para crear un objeto. También se conoce como fabricación aditiva porque se agrega material para construir algo, en lugar de quitarlo de una pieza de trabajo [7].

Manufactura Aditiva (AM): La manufactura aditiva, también conocida como impresión 3D, es un proceso que se utiliza para crear un objeto físico (o 3D) mediante la superposición de capas de material a partir de un modelo digital. A diferencia de la fabricación sustractiva, que crea el producto final retirando material de un bloque, la fabricación aditiva combina varias piezas para conformar el producto final [8].

Garantía de Calidad (QA): El aseguramiento de la calidad es una herramienta de gestión que engloba todas las acciones sistemáticas necesarias para brindar la suficiente confianza en que un producto cumplirá con la calidad requerida. Aquí la confianza es doble, tanto internamente a la dirección de la empresa como externamente a los grupos de interés como clientes, acreedores, clientes, sociedad, agencias gubernamentales, terceros, etc [9].

Gestión de Calidad (QM): La gestión de calidad es una serie de procesos sistemáticos que le permiten a cualquier organización planear, ejecutar y controlar las distintas actividades que lleva a cabo. Esto garantiza estabilidad y consistencia en el desempeño para cumplir con las expectativas de los clientes [10].

Red Neuronal (NN): Una red neuronal es un método de la inteligencia artificial que enseña a las computadoras a procesar datos de una manera que está inspirada en la forma en que lo hace el cerebro humano. Se trata de un tipo de proceso de machine learning llamado aprendizaje profundo, que utiliza los nodos o las neuronas interconectados en una estructura de capas que se parece al cerebro humano. Crea un sistema adaptable que las computadoras utilizan para aprender de sus errores y mejorar continuamente. De esta forma, las redes neuronales artificiales intentan resolver problemas complicados, como la realización de resúmenes de documentos o el reconocimiento de rostros, con mayor precisión [11].

RESUMEN

Este proyecto se centra en el desarrollo de un dispositivo de control remoto destinado a la detección y corrección en tiempo real de fallos en el proceso de impresión 3D utilizando la tecnología Fused Deposition Modeling (FDM). Para lograr esto, se aprovechan recursos como la Raspberry Pi y una cámara web, que capturan imágenes del proceso de impresión. La pieza central de esta solución es una red neuronal entrenada específicamente para identificar errores comunes en la impresión 3D, tales como el warping, stringing y spaghetti. La información se transmite de manera eficiente a través del protocolo MQTT, con notificaciones instantáneas enviadas a través de la plataforma Telegram para permitir una acción inmediata por parte del operador. La metodología abarca desde el entrenamiento de la red neuronal hasta el diseño de estrategias efectivas de control de impresoras 3D, que se integran sin problemas con el dispositivo de control remoto. La evaluación de los resultados se realiza mediante una matriz de confusión y la métrica Intersection over Union (IoU), destacando la alta precisión en la detección de errores. Este dispositivo representa una solución prometedora para mejorar significativamente la precisión y la confiabilidad del proceso de impresión 3D, con un enfoque particular en su aplicación en entornos industriales y críticos.

Palabras claves: Impresión 3D, Control Remoto, Detección de Errores, Red Neuronal, Internet de las Cosas.

ABSTRACT

This project focuses on the development of a remote-control device aimed at real-time detection and correction of errors in the 3D printing process using Fused Deposition Modeling (FDM) technology. To achieve this, resources such as the Raspberry Pi and a webcam are utilized to capture images of the printing process. The centerpiece of this solution is a neural network specifically trained to identify common 3D printing errors, such as warping, stringing, and spaghetti-like structures. Information is efficiently transmitted through the MQTT protocol, with instant notifications sent via the Telegram platform to enable immediate action by the operator. The methodology encompasses everything from training the neural network to designing effective 3D printer control strategies, seamlessly integrated with the remote-control device. Evaluation of the results is carried out using a confusion matrix and the Intersection over Union (IoU) metric, highlighting the high precision in error detection. This device represents a promising solution to significantly enhance the accuracy and reliability of the 3D printing process, with a particular focus on its application in industrial and critical environments.

Keyword: 3D Printing, Remote Control, Error Detection, Neural Network, Internet of Things.

INTRODUCCIÓN

La impresión 3D es el proceso de creación de objetos mediante el depósito de capas de material unas sobre otras. Esta tecnología existe desde hace unas cuatro décadas, inventada a principios de los años 80. Aunque la impresión 3D empezó siendo una técnica lenta y costosa, los amplios avances tecnológicos han hecho que las tecnologías actuales sean más asequibles y rápidas que nunca. Esta ofrece un número considerable de ventajas, la más importante de las cuales es la capacidad de producir diseños muy complejos que serían imposibles de realizar de otro modo [12]. Además del uso profesional para la creación de prototipos y la fabricación de bajo volumen, se están generalizando entre los usuarios finales, comenzando con el llamado Movimiento Maker. El tipo más frecuente de impresoras 3D de grado de consumo es el modelado por deposición fundida (FDM, también FFF de fabricación de filamentos fundidos). Este trabajo se centra en la maquinaria FDM debido a su aparición generalizada y gran número de problemas abiertos como la precisión y la falla. Estas impresoras 3D pueden fallar al imprimir objetos a una velocidad estadística dependiendo del fabricante y modelo de la impresora. Los fallos pueden ocurrir debido a la desalineación de la cama de impresión, el cabezal de impresión, el deslizamiento de los motores, la deformación del material impreso, la falta de adhesión u otras razones [13].

La impresión en 3D debe ser supervisada durante todo su proceso para evitar la pérdida de material de impresión, el tiempo del operador y asegurar la calidad del producto. Aunque algunas impresoras 3D vienen con monitores simples incorporados, actualmente no existen sistemas avanzados de control remoto y supervisión en tiempo real de las impresoras 3D. La vigilancia de los fallos y la finalización de los trabajos durante los procesos de impresión en 3D debe hacerse in situ, ya sea por operadores o cámaras de vídeo. Una solución para supervisar todo el proceso de impresión en 3D en tiempo real es la videovigilancia remota. Esta solución funciona bien para pequeñas producciones, pero puede ser un desafío para las producciones más grandes y sensibles. La cantidad de datos de vídeo generados por más de seis impresoras seriales 3D puede ser abrumadora de

transmitir y analizar. Además, el envío de todos estos datos para su procesamiento externo en la nube requeriría un gran ancho de banda y comunicaciones de baja latencia [14].

Han surgido numerosos métodos para abordar esta problemática. En este documento, se introduce un dispositivo de control para impresoras 3D que incorpora un sistema de detección de errores en tiempo real a través de modelos entrenados de visión artificial. Este dispositivo permite el monitoreo y control remoto de la impresora, además de una detección temprana de fallos. Estas capacidades se logran mediante la combinación de un microprocesador Raspberry Pi 4 Model B, una cámara web y un servidor en la nube de Google Cloud.

1. PLANTEAMIENTO DEL PROBLEMA

1.1. FORMULACIÓN DEL PROBLEMA

La fabricación aditiva (AM) o impresión 3D es un término para el proceso de fabricación capa por capa de componentes a partir de archivos tridimensionales de diseño asistido por computadora (CAD) [15]. En la investigación y la industria, así como en gran medida por usuarios domésticos no industriales, a menudo se utiliza el método de extrusión de material definido de acuerdo con ISO / ASTM 52900, que también se conoce bajo la marca Fused Deposition Modeling (FDM) [16].

En el contexto de la rápida evolución de la tecnología de impresión 3D, donde la fabricación aditiva se ha consolidado como una metodología de producción versátil y disruptiva en diversas industrias, la falta de monitoreo exhaustivo y la escasa disponibilidad de productos especializados para el control y monitoreo remoto de impresiones 3D se han convertido en desafíos cruciales que obstaculizan la optimización de este proceso y limitan su aplicación en entornos de alta precisión y producción crítica [17].

La impresión 3D ha revolucionado la manera en que se conciben y manufacturan objetos tridimensionales, permitiendo la creación de componentes altamente personalizados y geoméricamente complejos con eficiencia y rapidez. Esta tecnología ha encontrado aplicaciones en campos tan diversos como la industria aeroespacial, la medicina, la automoción y la fabricación industrial, entre otros. Sin embargo, esta rápida adopción ha resaltado la necesidad imperante de abordar de manera integral y precisa los desafíos asociados con la supervisión y el control de las impresiones 3D.

Los sistemas de impresión actuales, a pesar de su sofisticación, muchas máquinas de impresión 3D no cuentan con un sistema designado para rastrear

y monitorear el progreso del proceso de impresión. Las impresoras 3D pueden continuar imprimiendo la pieza hasta que todas las capas hayan sido completadas, incluso si se agotó el filamento o si existen posibles defectos en la impresión [18].

Si bien existen herramientas y soluciones aisladas para monitorear aspectos específicos de la impresión, como la temperatura o la velocidad de impresión, la ausencia de sistemas integrales que proporcionen un panorama completo de múltiples variables críticas afecta negativamente la eficiencia y la fiabilidad del proceso. Los operadores se encuentran obligados a depender de una combinación de herramientas fragmentadas, lo que resulta en una complejidad adicional y costos elevados.

La falta de monitoreo adecuado y la carencia de soluciones integrales en el mercado limitan la confiabilidad de las impresiones 3D en ámbitos donde la precisión y la repetibilidad son fundamentales, como la manufactura de dispositivos médicos y piezas de ingeniería. La superación de esta problemática abriría la puerta a una mayor adopción de la impresión 3D en entornos industriales y críticos, impulsando la innovación y la eficiencia en la producción.

Es por ello, y en busca de una opción práctica y asequible, se propuso el desarrollo de un dispositivo de control a distancia integrado con un sistema que permita el monitoreo continuo del proceso de fabricación sin necesidad de interacción humana. Por lo que se genera la siguiente pregunta problema:
¿Cómo crear un sistema para controlar y monitorear impresoras 3d para la pronta detección y corrección de errores en el proceso de fabricación?

1.2. JUSTIFICACIÓN Y ALCANCE

Debido a la amplia adopción de la técnica, la preocupación por el impacto ambiental de FDM es una de las principales preocupaciones de los usuarios. El aumento de la demanda y el uso de tecnologías han dado lugar a un aumento significativo en las ventas mundiales de máquinas AM, especialmente máquinas FDM, debido a las ventajas para los usuarios [19].

De acuerdo con el reporte anual de la empresa francesa Sculpteo los datos estadísticos referentes a la utilización de la impresión 3D por continente en los años 2019 y 2022 ofrecen una visión esclarecedora de la evolución de esta tecnología en diferentes regiones del mundo.

En 2019, Europa se destacó como la región líder en adopción de la impresión 3D, representando un notable 58.7% del total. Le siguió Asia, que contribuyó con un significativo 20.2%, mientras que Norte América representó el 16.6%. Otras regiones, como Oceanía, tuvieron una presencia menor, con un 1.1%, mientras que Sur América y África representaron el 1.6% y el 0.8%, respectivamente.

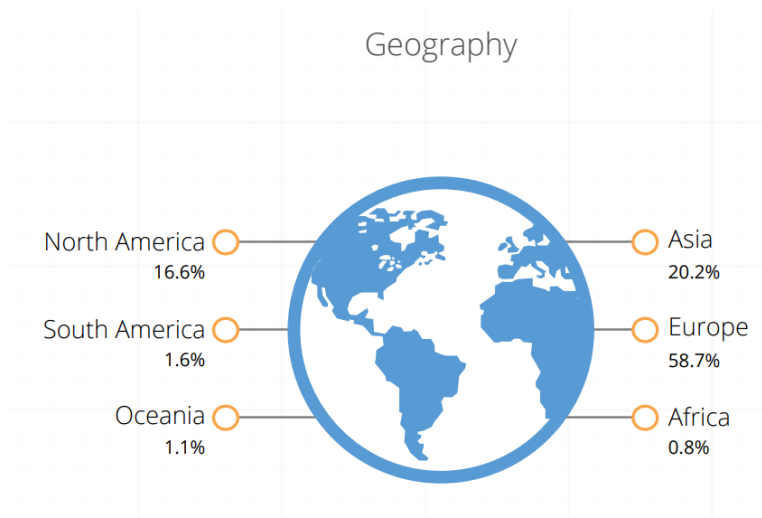


Figura 1. **Mapa Global de Adopción de Impresión 3D por Región (2019) [20]**

Para el año 2022, los patrones de utilización han experimentado cambios significativos. Europa continúa liderando en adopción, con un notable aumento al 63%. Norte América ha experimentado un crecimiento significativo, contribuyendo con un 23%, lo que indica un aumento en la adopción en esta región. Asia también ha incrementado su participación, alcanzando un 7%. Por otro lado, Sur América y África han aumentado su presencia, contribuyendo con un 4% y un 2%, respectivamente. En contraste, Oceanía se mantiene relativamente estable en un 1%.

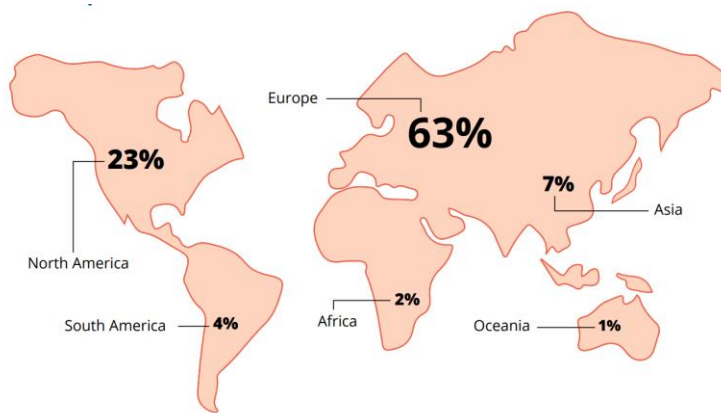


Figura 2. Mapa Global de Adopción de Impresión 3D por Región (2022) [21]

Estos datos reflejan una tendencia en la que Europa y Norte América continúan liderando en la adopción de la impresión 3D, aunque Asia está emergiendo como una región con un aumento constante en la adopción. La creciente inclusión de Sur América y África sugiere una expansión geográfica de esta tecnología lo que también se traduce en el aumento de las necesidades que se generan dentro de ella.

Por otro lado, con el proceso FDM se pueden utilizar varios termoplásticos como polilactida (PLA), acrilonitrilo butadieno estireno (ABS) y poliamida (PA) [22]. Además, los plásticos de alto rendimiento como la polieterimida (PEI) y la polietercetercetona (PEEK) se pueden procesar con sistemas FDM especiales [23]. Esto significa que el proceso también es de interés para

aplicaciones en los sectores aeroespacial y de tecnología médica. Sin embargo, para ser percibido como un proceso de producción serio en estos sectores industriales regulados, se deben cumplir requisitos especiales de garantía de calidad (QA) y gestión de calidad (QM).

En los procesos de AM en general y especialmente en FDM, hay una gran cantidad de parámetros de proceso y ambientales que influyen tanto en el proceso de impresión como en el resultado de impresión y tienen un impacto, por ejemplo, en las propiedades geométricas, mecánicas o superficiales, así como en la estabilidad del proceso [24]. Se deben desarrollar sistemas confiables de detección y análisis para caracterizar estos parámetros con el fin de garantizar un proceso de AM estable con especificaciones de proceso óptimas, resultados de proceso con requisitos de calidad definidos, así como un QM certificado.

El aprendizaje automático (ML) se utiliza cada vez más para monitorear el proceso de fabricación con el fin de analizar datos de imágenes y procesos y derivar predicciones sobre la calidad esperada de los componentes [25], [26]. Con la ayuda de cámaras y sensores, los datos de procesos especiales se pueden registrar y monitorear inicialmente en el sitio [27], [28]. Utilizando análisis de procesos basados en inteligencia artificial (IA), se pueden realizar procesos de aprendizaje basados en computadora. Estos se entrenan para evaluar los datos registrados o para predecir futuros cursos de datos y, por lo tanto, pueden proporcionar conclusiones tempranas sobre el proceso de impresión y la calidad de los componentes [24].

2. OBJETIVOS

2.1. OBJETIVO GENERAL

Desarrollar un dispositivo de control remoto para la detección y corrección de errores en el proceso de impresión 3D FDM en tiempo real.

2.2. OBJETIVOS ESPECÍFICOS

- Entrenar una red neuronal para la detección de fallos de impresión 3D.
- Diseñar una estrategia de control de impresoras 3d con implementación en un microprocesador de placa simple.
- Integrar el sistema de detección de fallos de impresión 3D con el dispositivo de control a distancia.
- Realizar pruebas de validación en ambiente controlado para garantizar el buen funcionamiento del dispositivo.

3. MARCO DE REFERENCIA

3.1. ESTADO DEL ARTE

En este capítulo se muestran una serie de investigaciones previas y trabajos relevantes que han abordado temas similares al presente proyecto. Lo cual permitirá comprender el contexto en el que se sitúa la investigación y proporcionará una base sólida para la formulación de la metodología a seguir.

- En Westphal, et al [16], se explora la relevancia de los parámetros ambientales y de proceso en la fabricación aditiva, específicamente en la Modelización por Deposición Fundida (FDM). Se emplean sensores para capturar datos ambientales durante la impresión, y se aplican algoritmos de aprendizaje automático para analizar y clasificar los resultados en distintas condiciones de impresión 3D. Se presenta una nueva metodología de preparación de datos que organiza las series temporales de los sensores, y se demuestra que los valores de presión del aire son esenciales para mejorar los análisis y reducir el sobreajuste. La arquitectura XceptionTime se destaca como la más efectiva en las pruebas, logrando altos niveles de precisión y clasificación. Comparado con escaneos 3D en garantía de calidad, este enfoque de aprendizaje automático permite distinguir con éxito entre diversas condiciones de impresión 3D, siendo un recurso valioso para la industria de fabricación aditiva y mejorando la confianza en estos procesos.
- En Xinbo Qi, et al [28], se analiza cómo se ha aplicado el algoritmo de red neuronal (NN) en varios aspectos de la cadena completa de la AM, incluido el diseño de modelos, el monitoreo in situ y la evaluación de la calidad. También se destacan los desafíos actuales en la aplicación de las NN a la AM y se proponen posibles soluciones para estos problemas. Por último, se plantean tendencias futuras para proporcionar una discusión general de esta área interdisciplinaria. En resumen, el artículo revisa cómo el aprendizaje

automático, particularmente a través de las redes neuronales, está siendo utilizado para mejorar diversos aspectos del proceso de fabricación aditiva y aborda las cuestiones clave y futuras en este campo.

- En Straub, et al [29], se propone un enfoque para abordar la detección de errores mediante el uso de un sistema de múltiples cámaras y software de procesamiento de imágenes. Este sistema se utiliza para evaluar el progreso de la impresión y detectar tanto defectos relacionados con la finalización incorrecta como problemas de calidad. Además de describir el enfoque, el artículo también explora las posibles aplicaciones y extensiones de este tipo de sistema en el contexto de la fabricación aditiva.
- En Bermudo, et al [30], tiene como objetivo ofrecer una solución para mejorar la accesibilidad y el control de múltiples impresoras 3D, con el propósito de ahorrar tiempo y esfuerzo en la obtención de los resultados deseados. Además, se busca implementar una herramienta de visión por computadora para supervisar y detectar posibles fallos en la impresión. En caso de que ocurra algún fallo, esta herramienta permitirá ahorrar material y prevenir daños en la máquina. La solución presentada en este proyecto está especialmente diseñada para gestionar y utilizar una sala de impresión 3D en un entorno educativo. Esto significa que el profesor o encargado no necesitará invertir tiempo adicional para operar las impresoras, ya que se busca maximizar la facilidad de uso con el fin de fomentar su utilización de manera efectiva.
- En Delli, et al [31], se propone un método que utiliza una cámara, procesamiento de imágenes y aprendizaje automático supervisado para evaluar automáticamente la calidad de las piezas impresas en 3D. Se capturan imágenes de las piezas semifinalizadas en varias etapas críticas del proceso de impresión de acuerdo con la geometría de la pieza. Se propone el uso de un método de aprendizaje automático, la máquina de

soporte vectorial (SVM), para clasificar las piezas en la categoría de 'buena' o 'defectuosa'. Se imprimieron piezas utilizando materiales ABS y PLA para demostrar el marco propuesto, y se proporciona un ejemplo numérico para ilustrar cómo funciona el método propuesto.

- En Fu, et al [32], este artículo aborda la importancia de la monitorización en tiempo real en el proceso de fabricación aditiva, específicamente en la impresión por deposición de filamento fundido. Se resumen diversos métodos y dispositivos utilizados en sistemas de monitorización para la impresión en 3D, destacando la monitorización del estado de salud de la impresora y la calidad del producto. El artículo concluye con una discusión sobre las direcciones actuales y futuras de investigación en este campo, subrayando la relevancia de la monitorización continua para garantizar la calidad y eficiencia en el proceso de impresión 3D.
- En Khan, et al [33], En este proyecto se desarrolló un modelo de Aprendizaje Profundo (Deep Learning) utilizando una Convolutional Neural Network (CNN) para la detección en tiempo real de defectos maliciosos en la impresión 3D, con el objetivo de prevenir pérdidas de producción y reducir la intervención humana en la verificación de calidad. La metodología propuesta se basa en la extracción de características de anomalías geométricas que surgen en patrones de relleno debido a problemas como extrusión inconsistente, rellenos débiles, falta de soportes o hundimientos, y se comparan con las características de una impresión 3D perfecta. Este enfoque se basa en conceptos de clasificación de imágenes y visión por computadora utilizando aprendizaje automático, aprovechando la disponibilidad de conjuntos de datos, sistemas de monitoreo y la capacidad para detectar relaciones causales de defectos. Para evaluar la calidad de las piezas, una cámara integrada en la impresora 3D captura imágenes en intervalos regulares y las procesa utilizando el modelo CNN. Los resultados del proyecto son un proceso de impresión 3D más optimizado y automatizado,

con el potencial de resolver el problema más extendido de la variabilidad de productos en la impresión 3D.

- En Paraskevoudis, et al [34], se presenta una nueva metodología para evaluar la calidad de objetos impresos en 3D durante la impresión. Se desarrollan Redes Neuronales para identificar defectos de impresión 3D, específicamente el "stringing", al analizar videos capturados del proceso. El "stringing" es un defecto que suele ocurrir durante la impresión en forma de hilos no deseados entre secciones del objeto. Se entrena un modelo de Inteligencia Artificial (IA) en imágenes que muestran claramente este problema y se implementa en tiempo real para detectar y predecir el "stringing" en un flujo de video. La metodología propuesta permite reconocer este defecto con rapidez y precisión, y además, ofrece la posibilidad de ajustar el proceso de impresión para corregir este problema, mejorando así la calidad del producto final.
- En Song, et al [35], se abordaron las limitaciones de la fabricación aditiva en términos de calidad debido a defectos como la deformación por warping, causada por el estrés térmico residual generado durante los procesos de fabricación. Se utilizó una variedad de datos térmicos en series temporales y el algoritmo de K-nearest neighbors (KNN) con dynamic time warping (DTW) para detectar y predecir la deformación por warping en piezas impresas utilizando impresoras de modelado por deposición fundida (FDM). Se entrenaron datos térmicos en series temporales multivariados extraídos de termopares mediante el uso de KNN basado en DTW para clasificar la deformación por warping. Los resultados mostraron que el enfoque propuesto puede predecir la deformación por warping con una precisión de más del 80% utilizando solo datos térmicos en series temporales correspondientes al 20% de todo el proceso de impresión. Además, la precisión de clasificación mostró el potencial prometedor del enfoque propuesto en la predicción de deformaciones por warping y en procesos de

fabricación reales, lo que podría reducir el tiempo y los costos adicionales resultantes de procesos defectuosos.

- En Roller, et al [36], se centra en las impresoras de modelado por deposición fundida (FDM) debido a su amplia adopción y la existencia de desafíos, como problemas de precisión y fallos en la impresión. Se subraya cómo estas impresoras, aunque tienen un papel importante en la fabricación, a menudo presentan tasas de fallos estadísticas que varían según el fabricante y el modelo, debido a una serie de factores como desalineaciones, deslizamiento de motores, deformaciones de material y falta de adherencia, entre otros. El objetivo principal del estudio es proporcionar una solución para detectar automáticamente estos fallos, especialmente porque las impresoras FDM a menudo se ubican en habitaciones separadas por problemas de ventilación, dificultando la supervisión en tiempo real.
- En Kim, Lee, et al [37], se abordó el problema comúnmente conocido como "error de forma de espagueti" en el proceso de extrusión de material (ME) en la impresión 3D. Este error se relaciona con el enredo del filamento y puede llevar a la detención del proceso. Para prevenirlo, se desarrolló un método de detección de fallas utilizando una cámara web y aprendizaje profundo. La cámara web captura imágenes que luego se analizan mediante aprendizaje automático basado en una red neuronal convolucional (CNN), demostrando un rendimiento destacado en clasificación de imágenes y reconocimiento de objetos. El modelo fue entrenado y evaluado, logrando una precisión del 97%. Posteriormente, se probó el modelo en un sistema de monitoreo de impresora 3D para reconocer el "error de forma de espagueti", logrando detectar el 96% de los procesos de deposición anormales. El método propuesto puede analizar el proceso de ME en tiempo real e informar al usuario o detener el proceso cuando se detecta una impresión anormal.

- En Verana, et al [38], se propone un diagnóstico de fallos para impresoras 3D basado en una red neuronal convolucional (CNN). Utilizando un conjunto de datos recopilados de impresoras 3D en funcionamiento, la CNN procesó, detectó y clasificó anomalías en la impresión 3D con una precisión notable. La CNN propuesta superó al soporte de máquina vectorial (SVM) y a la red neuronal artificial (ANN) en un 5.1% y un 25.7%, respectivamente. Esta metodología ofrece un enfoque efectivo para identificar y clasificar problemas en la impresión 3D, mejorando la eficacia del diagnóstico de fallos en este contexto.

3.2. MARCO TEÓRICO

En esta sección se presentarán las teorías, conceptos y enfoques relevantes que son fundamentales para comprender y abordar la problemática que se plantea. A través de la revisión y síntesis de la literatura académica, se busca proporcionar una base sólida para el entendimiento de la solución planteada.

3.2.1 IMPRESIÓN 3D FDM (Fused Deposition Modeling)

La impresión 3D de FDM es un método de fabricación aditiva patentado por el ingeniero mecánico estadounidense y fundador de Stratasys, Scott Crump, en 1989. Crump buscaba una forma más fácil de fabricar prototipos y experimentó con plásticos semisólidos, fundiéndolos manualmente por capas con una pistola de cola. Este método lo llamó «Modelado por deposición fundida», o FDM. Cuando Crump hubo desarrollado el software de fabricación asistida por ordenador (CAM) para automatizar el proceso, empezó a vender impresoras 3D por unos 12 000 \$. [39].

Una impresora 3D FFF (o FDM, por Fused Deposition Modeling) dibuja una capa de plástico fundido en su lecho de impresión o construye una placa. La fusión se produce dentro de un extrusor, que calienta el filamento de plástico a medida que los engranajes lo empujan a través de la boquilla.

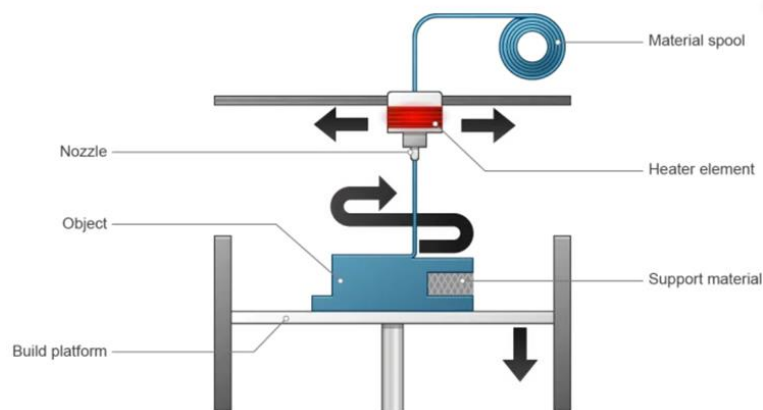


Figura 3. Sistema de Modelado por Deposito Fundido [12]

3.2.2 FORMATOS DE MODELOS PARA IMPRESIÓN 3D

- **Formato STL**

Este formato se aproxima a la superficie de un modelo sólido con triángulos. Para un modelo simple, como el cuadrado que se muestra en la figura 4(a), se puede aproximar a sus superficies con doce triángulos, como se muestra en la figura 4(b). Cuanto más compleja sea superficie, más triángulos se deberán realizar, como se muestra en la figura 4(c).

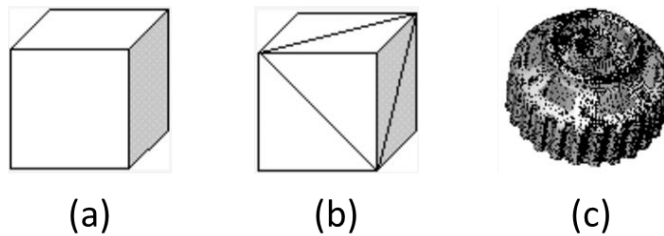


Figura 4. Formato de Impresión STL

Casi todos los sistemas de CAD actuales pueden generar un archivo STL. Para el usuario, el proceso a menudo es tan simple como seleccionar Archivo, Guardar como y STL. A continuación, se detallan los pasos para generar archivos STL de alta calidad mediante varios de los principales sistemas de CAD actuales. En todos los casos, se debe exportar el archivo STL como un archivo binario. Esto permite ahorrar tiempo y disminuir el tamaño del archivo.

Como regla general, al cambiar ciertas opciones, como la tolerancia de la cuerda o el control angular, se modificará la resolución del archivo STL. Cuanto mayor sea el archivo STL, más triángulos se generarán en la superficie del modelo. Para geometrías simples (con pocas curvas), es posible que el archivo sea de tan solo algunos cientos de kilobytes. Para modelos complejos, los archivos de tamaño entre 1 y 5 MB generarán piezas de buena calidad [40]

- **Formato OBJ**

En lo que se conoce, el formato de archivo OBJ almacena información sobre modelos 3D. Originalmente fue creado por “Wavefront Technologies” para su aplicación “Advanced Visualizer” para almacenar objetos geométricos compuestos de líneas, polígonos, curvas y superficies de forma libre.

Como tales, los archivos OBJ pueden codificar la geometría de la superficie de un modelo 3D, pero también pueden almacenar información de color y textura. Sin embargo, el formato no almacena ninguna información de escena (como la posición de la luz) ni animaciones.

Además, un archivo OBJ generalmente se genera mediante un software CAD (diseño asistido por computadora) como producto final del proceso de modelado 3D. La extensión de este correspondiente formato es simplemente “.obj”.

Por otro lado, el formato de archivo OBJ es de código abierto y neutral. Lo cual, a menudo se usa para compartir modelos 3D en aplicaciones gráficas, esto debido a que se disfruta de un buen soporte de importación y exportación de casi todos los programas CAD.

En los últimos años, también se ha vuelto popular como formato de archivo para la impresión 3D multicolor, ya que el formato de impresión 3D estándar “STL” no admite información de color y textura [41].

3.2.3 LENGUAJE DE IMPRESORAS 3D

El G-Code, también conocido como RS-274, lenguaje de programación G, o ISO-Code, es el lenguaje de programación más empleado en máquinas de control numérico (CNC). No es exclusivo de la impresión 3D. Por el contrario, es ampliamente empleado en todo tipo de máquinas como tornos, fresadoras, corte por láser, desde muy pequeñas hasta de tamaño industrial.

Un fichero en G-Code está formado por un conjunto de instrucciones sencillas que indican a una máquina las operaciones que debe realizar. Por ejemplo, desplazar una parte (cabezal, garras, topes), realizar un cambio de herramienta. Aunque el conjunto de instrucciones está “más o menos” estandarizado, existen distintas implementaciones como por ejemplo la ISO 6983, DIN66025, Siemens, FANUC, Haas.

La impresora 3D no deja de ser una máquina CNC. En este caso el firmware es el que se encarga de interpretar cada línea de G-Code y ejecutar las acciones oportunas en la impresora. Sin embargo, lógicamente, no todas las instrucciones del estándar se aplican en una impresora 3D. En la siguiente tabla se tiene un listado con algunas de las más importantes y que más frecuentemente se encuentran en el ámbito de impresoras FFF/FDM.

Código	Significado
M0	Parada
M1	Sleep
M2	Fin del programa
M70	Mostrar mensaje en pantalla
M104	Temperatura del extrusor
M106	Velocidad del ventilador
M107	Apagar ventilador
M140 y M190	Temperatura de la cama
M116	Esperar que temperaturas se estabilicen
M112	Parada de emergencia
G0	Movimiento rápido
G1	Movimiento controlado
G4	Pausa
G10	Retracción
G11	Desretracción
G20	Establecer unidades en pulgadas
G21	Establecer unidades en milímetros
G28	Mover al origen (Home)
G29	Autonivelado
G90	Posicionamiento absoluto
G91	Posicionamiento relativo
G92	Establecer posición

Tabla 1. Instrucciones en código G para impresión 3D [42]

3.2.4 ERRORES EN LA IMPRESIÓN 3D FDM

- **Warping**

El warping aparece al haber un cambio brusco de temperatura en el material. El material se enfría creando una contracción y a su vez una tensión interna que provoca que la pieza se levante. Por ejemplo, si se está imprimiendo con ABS entre 240 y 260 ° aproximadamente y la cama se encuentra entre 60 y 80 ° como la temperatura ambiente no controlada se presenta un cambio excesivo de temperatura creando una contracción lo cual provoca que el material se levante. Cuanto más grande sea la pieza más fácil se origina el warping [43].

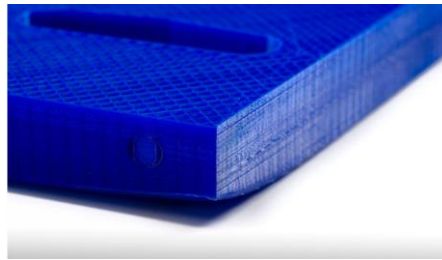


Figura 5. Ejemplo de Warping [44]

- **Stringing**

El stringing se produce básicamente cuando se utilizan filamentos, que luego quedan como finos hilos de filamento en el objeto impreso en 3D. Esto suele ocurrir debido a una configuración incorrecta, de modo que el filamento sigue goteando de la boquilla, aunque el extrusor esté a punto de moverse a otro lugar para continuar la impresión 3D FDM. [45]

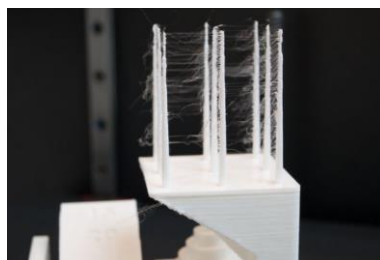


Figura 6. Ejemplo de Stringing [45]

- **Layer Shifting**

El desplazamiento de capa es un problema de impresión, que hace que las capas del objeto impreso se desplacen de sus posiciones previstas. Por lo general, se asocia con un movimiento anormal del eje X y / o el eje Y, lo que lleva a que el cabezal de la extrusora esté desalineado a mitad de la impresión.

Para solucionar el problema correctamente, es crucial reconocer en qué eje se desplazaron las capas [46].



Figura 7. Ejemplo de LayerShifting [47]

- **Spaghetti**

Este problema de la impresora 3D es exactamente lo que su nombre implica. La impresión resulta en un desorden de cuerdas de filamento que se asemejan a espaguetis. Este desorden de espaguetis se acumula dentro y alrededor de la impresión debido a una variedad de factores como el desprendimiento de la impresión de la cama, el colapso de los objetos impresos o incluso problemas como el código G erróneo [48].



Figura 8. Ejemplo de Spaghetti [49]

3.2.5 Tecnologías de Desarrollo IoT

- **Protocolo MQTT**

MQTT es un protocolo de mensajería basado en estándares, o un conjunto de reglas, que se utiliza para la comunicación de un equipo a otro. Los sensores inteligentes, los dispositivos portátiles y otros dispositivos de Internet de las cosas (IoT) generalmente tienen que transmitir y recibir datos a través de una red con recursos restringidos y un ancho de banda limitado. Estos dispositivos IoT utilizan MQTT para la transmisión de datos, ya que resulta fácil de implementar y puede comunicar datos IoT de manera eficiente. MQTT admite la mensajería entre dispositivos a la nube y la nube al dispositivo [50].

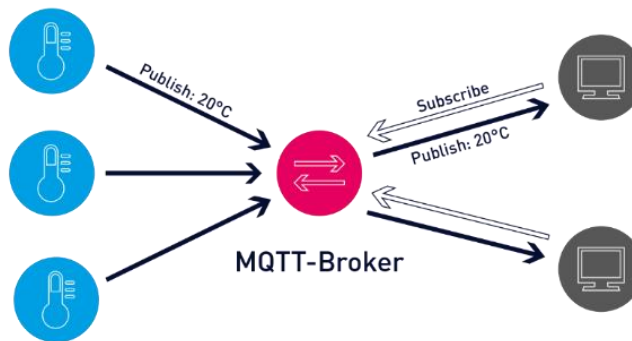


Figura 9. Protocolo MQTT

- **Node Red**

Node-RED es una herramienta de programación para conectar dispositivos de hardware, API y servicios en línea de formas nuevas e interesantes. Proporciona un editor basado en navegador que facilita la conexión de flujos utilizando la amplia gama de nodos de la paleta que se pueden implementar en su tiempo de ejecución con un solo clic.

Node-RED proporciona un editor de flujo basado en navegador que facilita la conexión de flujos utilizando la amplia gama de nodos de la paleta. Los flujos se pueden implementar en el tiempo de ejecución

con un solo clic. Las funciones de JavaScript se pueden crear dentro del editor utilizando un editor de texto enriquecido. [51].

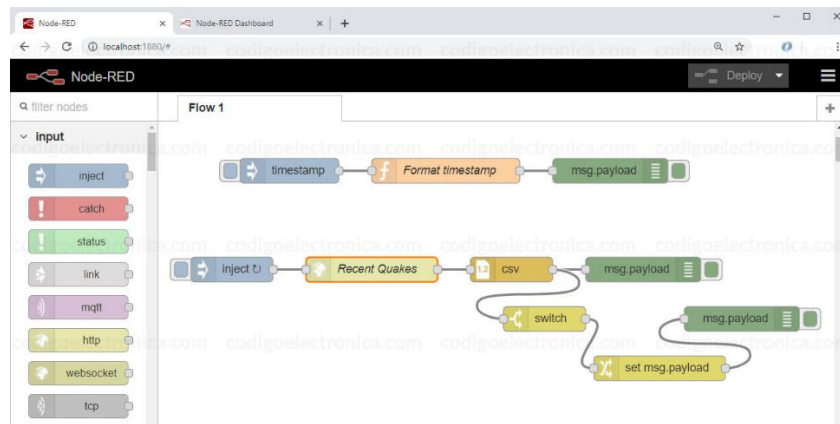


Figura 10. Node Red [52]

- **Python**

Python es un lenguaje de programación orientado a objetos de alto nivel y fácil de interpretar con sintaxis fácil de leer. Ideal para prototipos y tareas ad hoc, Python tiene un amplio uso en computación científica, desarrollo web y automatización. Como lenguaje de programación para principiantes y de uso general, Python es compatible con muchos de los principales científicos de computadoras y desarrolladores de aplicaciones en todo el mundo [53].

- **Google Cloud Platform**

Google Cloud Platform se trata de la suite de infraestructuras y servicios que Google utiliza a nivel interno y, ahora, disponible para cualquier empresa, de tal forma que sea aplicable a multitud de procesos empresariales

Cuando hablamos de Google Cloud Platform (GCP), estamos ante todas las herramientas de Google disponibles en la nube que hasta ahora se ofrecían por separado. Este conjunto de servicios ofrece prestaciones muy dispares; desde machine learning hasta Inteligencia

artificial pasando por el big data, todo englobado bajo el paraguas del cloud computing [54].

- **Telegram Python API**

Una API, o Interfaz de Programación de Aplicaciones (por sus siglas en inglés, Application Programming Interface), es un conjunto de reglas y protocolos que permiten a diferentes aplicaciones o sistemas interactuar y comunicarse entre sí. Las APIs definen las formas en que los componentes de software deben interactuar, lo que facilita el intercambio de datos y funcionalidades entre aplicaciones de manera estandarizada.

La API de bot le permite crear fácilmente programas que usan mensajes de Telegram para una interfaz. La API de Telegram y TDLib le permiten crear sus propios clientes de Telegram personalizados.

Esta API le permite conectar bots a nuestro sistema. Los bots de Telegram son cuentas especiales que no requieren un número de teléfono adicional para configurarse. Estas cuentas sirven como interfaz para el código que se ejecuta en algún lugar del servidor.

Para usar esto, no necesita saber nada sobre cómo funciona nuestro protocolo de cifrado MTProto: nuestro servidor intermediario manejará todo el cifrado y la comunicación con la API de Telegram por usted. Usted se comunica con este servidor a través de una interfaz HTTPS simple que ofrece una versión simplificada de la API de Telegram [55].

- **Comunicación Serial**

La comunicación en serie es un método comúnmente utilizado para intercambiar datos entre ordenadores y dispositivos periféricos. La transmisión serie entre el emisor y el receptor está sujeta a protocolos estrictos que proporcionan seguridad y fiabilidad y han llevado a su longevidad. Muchos dispositivos, desde ordenadores personales hasta dispositivos móviles, utilizan la comunicación en serie [56].

- **Pytorch**

Se trata de una biblioteca de código abierto diseñada con Python en mente y creada para proyectos de aprendizaje automático. Se especializa en diferenciación automática, cálculos de tensor y aceleración de GPU. Esto lo hace especialmente adecuado para aplicaciones de aprendizaje automático de vanguardia como el aprendizaje profundo. PyTorch es especialmente popular entre los investigadores debido a la personalización de Python. Crear capas de datos personalizadas y arquitecturas de red es especialmente fácil mediante Python [57].

- **Visión Artificial**

La visión artificial es un campo de la inteligencia artificial (IA) que permite a los ordenadores y sistemas extraer información significativa a partir de imágenes digitales, videos y otras entradas visuales, y tomar medidas o realizar recomendaciones en función de esa información. Si la IA permite a los ordenadores pensar, la visión artificial les permite ver, observar y comprender.

La visión artificial funciona de manera muy similar a la visión humana, excepto que los humanos tienen una ventaja inicial. La vista humana tiene la ventaja de toda una vida de contexto para entrenar cómo distinguir los objetos, a qué distancia están, si se están moviendo y si hay algo mal en una imagen.

La visión artificial entrena a las máquinas para realizar estas funciones, pero tiene que hacerlo en mucho menos tiempo con cámaras, datos y algoritmos en lugar de retinas, nervios ópticos y una corteza visual. Como un sistema entrenado para inspeccionar productos o ver un activo de producción puede analizar miles de productos o procesos por minuto, detectando defectos o problemas imperceptibles, puede superar rápidamente las capacidades humanas [58].

- **Yolo V5**

YOLO es un método para la identificación y el reconocimiento de objetos en tiempo real en fotografías. Es un acrónimo de You Only Look Once. Redmond et al. propuso el enfoque en un artículo que se publicó inicialmente en 2015 en la Conferencia IEEE/CVF sobre visión artificial y reconocimiento de patrones (CVPR).

El premio OpenCV People's Choice Award se le otorgó al periódico. A diferencia de los métodos de identificación de objetos anteriores, que reutilizaron los clasificadores para realizar la detección, YOLO propone el uso de un punto a punto. red neural que predice cuadros delimitadores y probabilidades de clase simultáneamente.

YOLO produce resultados de vanguardia al adoptar un enfoque fundamentalmente nuevo para el reconocimiento de objetos, superando fácilmente los métodos anteriores de detección de objetos en tiempo real [59].

- **OpenCV**

OpenCV (Open Source Computer Vision) es una librería software open-source de visión artificial y machine learning. Provee una infraestructura para aplicaciones de visión artificial.

Es una librería muy usada a nivel comercial, desde Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota, Applied Minds, VideoSurf, Zeitera. La librería tiene más de 2500 algoritmos, que incluye algoritmos de machine learning y de visión artificial para usar.

Estos algoritmos permiten identificar objetos, caras, clasificar acciones humanas en vídeo, hacer tracking de movimientos de objetos, extraer modelos 3D, encontrar imágenes similares, eliminar ojos rojos, seguir el movimiento de los ojos, reconocer escenarios [60].

- **IoT**

El término IoT, o Internet de las cosas, se refiere a la red colectiva de dispositivos conectados y a la tecnología que facilita la comunicación entre los dispositivos y la nube, así como entre los propios dispositivos. Gracias a la llegada de los chips de ordenador de bajo coste y a las telecomunicaciones de gran ancho de banda, ahora tenemos miles de millones de dispositivos conectados a Internet. Esto significa que los dispositivos de uso diario, como los cepillos de dientes, las aspiradoras, los coches y las máquinas, pueden utilizar sensores para recopilar datos y responder de forma inteligente a los usuarios.

El Internet de las cosas integra las “cosas” de uso diario con Internet. Los ingenieros en informática llevan agregando sensores y procesadores a los objetos cotidianos desde los años 90. Sin embargo, el progreso fue inicialmente lento porque los chips eran grandes y voluminosos. Los chips de ordenador de baja potencia llamados etiquetas RFID se utilizaron por primera vez para el seguimiento de equipos caros. A medida que se reducía el tamaño de los dispositivos informáticos, estos chips también se hacían más pequeños, más rápidos e inteligentes [61].

4. PROCEDIMIENTO METODOLÓGICO

4.1. METODLOGÍA

En esta sección se planteará la metodología a seguir durante el desarrollo de proyecto, la cual se comprende de 4 fases que contemplan el entrenamiento de la red neuronal de detección de fallos, el diseño del sistema de control, la integración del sistema de control con el modelo entrenado de detección y finalmente la fase de prueba y validación. En la ilustración 10 se puede observar el diagrama de bloques del desarrollo del dispositivo PrintGuard.

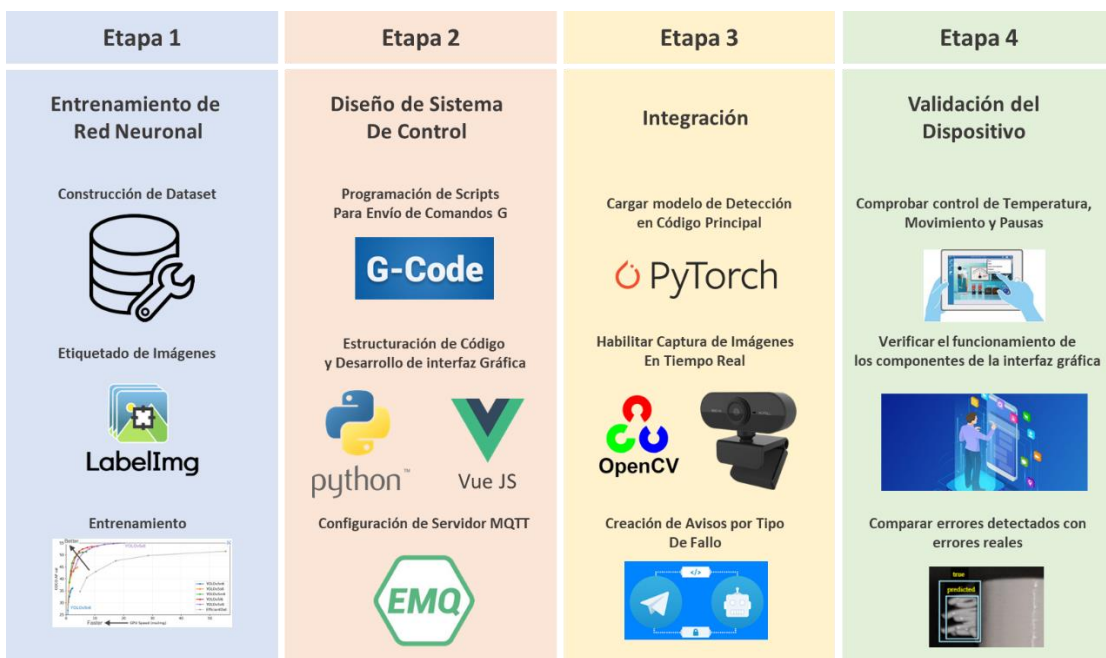


Figura 11. Diagrama de Bloques Metodología de Desarrollo Dispositivo PrintGuard (Autoría Propia).

4.1.1. Etapa 1. Entrenamiento de Red Neuronal

Para el entrenamiento de la red neuronal se utilizará el modelo preentrenado Yolov5, en su versión Yolov5n. Para enseñarle al modelo los fallos a detectar, se le deben ingresar imágenes etiquetadas que corresponden a los fallos correspondiente. En esta sección se explica el procedimiento a seguir para la obtención de los fallos, etiquetado de las imágenes y el entrenamiento final del modelo.

El entrenamiento de un modelo preentrenado Yolo requiere distintos parámetros de entrada como son:

- Nombres de las Etiquetas, en este caso corresponde al nombre del fallo que se está detectando.
- Dataset, son el conjunto de imágenes las cuales tienen los errores que el modelo debe detectar.
- Etiquetas, es un archivo de texto que contiene el nombre de la etiqueta y las coordenadas dentro de cada imagen que corresponden al recuadro del fallo a detectar.

- **Construcción de Dataset**

Para la construcción del dataset se fabricarán piezas configuradas con parámetros que propicien la generación de los fallos a detectar. Se utilizará el laminador Cura Slicer para la configuración de los parámetros y generación de GCODE, como ejemplo se muestra en la figura 12 la parametrización sin retracciones para generación de stringing..

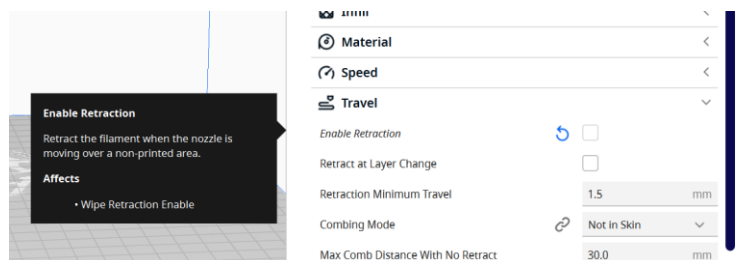


Figura 12. Ejemplo de modificaciones para generación de Errores (Autoría Propia)

- **Etiquetado de Imágenes**

Para el etiquetado de las imágenes se utilizará la herramienta de base en Python “LabelImg”. La cual toma como entrada las imágenes capturadas, luego cada una de estas imágenes son etiquetadas seleccionando la zona donde se encuentra el error y finalmente es generado en otra carpeta un archivo de texto con el mismo nombre de la imagen y dentro de el se encuentra el índice de la etiqueta y las respectivas coordenadas dentro de la imagen.



Figura 13. Etiquetado con LabelImg [62]

- **Entrenamiento**

El modelo preentrenado yolov5 es supervisado, por lo que necesita como entrada un porcentaje del dataset para realizar el entrenamiento y otro porcentaje para la validación del comportamiento del modelo. Para esto se escogerá un 80% de imágenes para entrenamiento y el otro 20% para validación.

Para el entrenamiento del modelo se requiere un computador con Python y todas las librerías que indica la documentación de Yolov5. Se inicia el entrenamiento corriendo el código Train.py que se encuentra dentro del repositorio de GitHub [63].

4.1.2. Etapa 2. Diseño del Sistema de Control

En esta etapa se busca hacer el diseño del sistema de control del dispositivo, con el cual se tenga la posibilidad de controlar a distancia la impresora, así como recibir información del comportamiento de la temperatura. Para esto se hará el desarrollo de una interfaz que muestre de forma más interactiva todo el sistema.

- **Programación de Scripts para Envío de Comandos en Código G**

La comunicación de las impresoras 3D con ordenadores se realiza a través de conexión serial, por lo que a través de Python se debe desarrollar un script que haga la conexión serial con la impresora y envíe los comandos según la necesidad.



Figura 14. Diagrama de Conexión Serial [64]

Una vez realizada la conexión serial, se envían los comandos desde Python. Los comandos para utilizar serían con la finalidad de: mover ejes, calentar resistencias, detener impresiones, etc. Estos comandos pueden ser observados en la tabla 1 del documento.

- **Estructuración de Código y Desarrollo de Interfaz Gráfica**

La estructuración del código hace referencia a la integración de todas las funciones de envío de comandos en código G, así como la definición del protocolo de comunicación con la interfaz gráfica y demás configuraciones que se deben tener listas al momento de hacer la integración con el modelo entrenado.

En esta etapa se desarrolla una interfaz gráfica basada en tecnologías web con Quasar-VueJS y Firebase.

Se realizaron evaluaciones exhaustivas de distintas alternativas, concluyendo que Quasar, una plataforma de desarrollo de aplicaciones web en tiempo real, se presenta como la elección más idónea. Esto se debe a su capacidad para la creación de aplicaciones web altamente interactivas y escalables. La interfaz constará de dos secciones principales: una de inicio de sesión y otra de monitoreo y control.

En la implementación de estas vistas, optaremos por emplear componentes que incorporan formularios con campos de texto, botones y otros elementos. Estos componentes serán estructurados mediante el uso de HTML y Javascript.

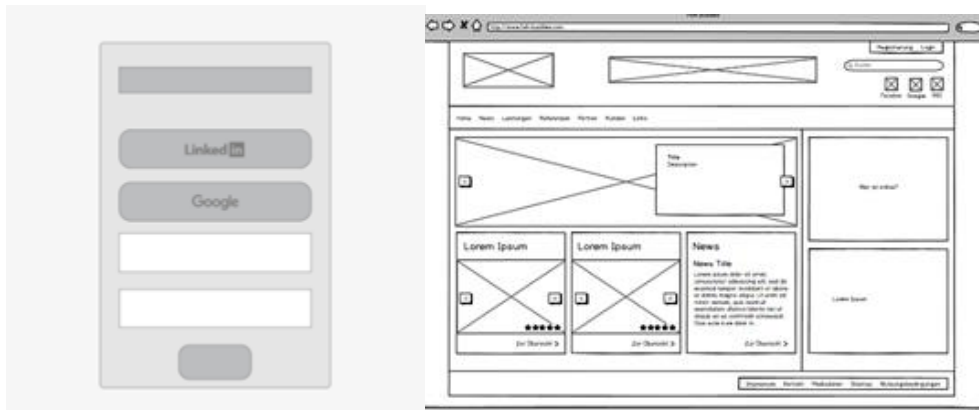


Figura 15. Esquema de Interfaz (Autoría Propia)

Para la parte lógica de la vista de inicio de sesión, se utiliza Firebase y un clúster en Firestore, el cual controla el registro y credenciales de acceso al sistema. Para la vista de monitoreo y control se utilizará Javascript con la librería paho-mqtt para el envío de datos al software de control integrado en dispositivo.

- **Configuración de Servidor MQTT**

Para el envío y recepción de datos de la nube se requiere el uso de un Broker MQTT el cual enlazaría el dispositivo físico y la interfaz web como clientes que interactúan entre ellos. Para esto se configurará un servidor Linux en Google Cloud Platform y se instalará el Broker MQTT.

4.1.3. Etapa 3. Integración

Para esta etapa se busca realizar la integración de todas las tecnologías que se han venido desarrollando por separado. Desde el modelo entrenado de detección, como el software de control de la máquina impresora y la interfaz web.

- **Cargar Modelo de Detección**

Dentro del software de control Python con el uso de la librería Pytorch y las funciones compartidas en la documentación de yolov5 se debe cargar la información del modelo. El cual más adelante será utilizado para la predicción de las imágenes que se le envíen.

- **Captura de Imágenes en Tiempo Real**

Hasta el momento el software de Python sería capaz de controlar la máquina. Para habilitar la detección de fallos se colocará una cámara web que se encargue de capturar las imágenes y enviar el frame capturado al modelo para realizar las detecciones.

- **Programación de Avisos por Tipo de Fallo**

Una vez listo el sistema de detección, estos fallos deben ser notificados por lo que se debe diseñar una estrategia que se encargue de enviar avisos al usuario sobre el error presentado y si lo requiere, una imagen del error.

- **Definición de Tópicos MQTT y Funciones de Envío y Recepción de Datos**

En esta última etapa de desarrollo se realiza el enlace mediante el protocolo MQTT entre la interfaz web y el software, la definición de tópicos corresponde a identificación de las variables que se transportarán de cliente a cliente.

4.1.4. Etapa 4. Validación del Dispositivo

En esta última etapa del proyecto se busca hacer la validación en un ambiente controlado de todo el sistema. Para observar el comportamiento y realizar los ajustes necesarios.

- **Comprobar control de movimiento, temperatura y pausas**

Para la validación del sistema de control, se realizarán pruebas enviando instrucciones desde la interfaz gráficas, ajustando los valores de temperatura y observando el comportamiento en físico, pausando impresiones en proceso y realizando el control del movimiento de los ejes en tiempo real.

- **Verificar el funcionamiento de los componentes de la interfaz gráfica**

Dentro de la interfaz gráfica se podrá visualizar la información de: temperaturas, posición del cabezal, botones, gráficos. Por lo que se debe verificar que cada uno de estos componentes se encuentre trabajando de forma óptima enviando la información correspondiente.

- **Comparar errores detectados con errores reales**

Se deben fabricar nuevamente piezas con y sin errores y comparar las detecciones del modelo con respecto a los errores (si los hay). De esta manera estimar el porcentaje de confiabilidad del modelo.

4.2. TIPO DE ESTUDIO

4.2.1. CUANTITATIVA EXPERIMENTAL

El proyecto se basará en un enfoque cuantitativo experimental, ya que se llevarán a cabo pruebas controladas bajo diversas condiciones ambientales y parámetros de impresión para evaluar la eficacia del dispositivo de detección de fallos en la impresión 3D con visión artificial. Durante el proceso, se realizarán ajustes en el software según sea necesario para obtener mediciones precisas. Este enfoque permitirá recopilar datos objetivos sobre el rendimiento del dispositivo, con el fin de tener un dispositivo con un alto porcentaje de confiabilidad.

4.2.2. POBLACIÓN Y MUESTRA

La presente investigación no requiere de población ya que en la propuesta planteada se desarrollará un prototipo para monitorear impresiones 3D.

La muestra de este proyecto será obtenida a partir de todas las imágenes generadas con errores por el equipo de trabajo y algunas imágenes encontradas en internet que hacen referencia a los mismos errores a detectar.

4.3. CRONOGRAMA – PLAN DE TRABAJO

En esta sección se muestra el orden de las actividades que se realizarán en el proyecto, así como las fechas tentativas para finalización de cada objetivo, esto con el fin de cumplir con las actividades planteadas en la metodología dentro del plazo disponible.



 Universidad Autónoma del Caribe Proyecto de Grado - Ingeniería Mecatrónica PLAN DE TRABAJO 					
Componentes	Descripción	Fecha Inicio	Fecha Final	Duración (Días)	Valores Presupuesto
OBJETIVO 1	Entrenar una red neuronal para la detección de fallos de impresión 3D.	18/03/2023	18/05/2023	61	COP 9.000.000
Entregable # 1	Dataset de Imágenes	18/03/2023	2/04/2023	16	COP 2.800.000
Actividad 1	Captura de video de impresión de piezas con fallos puntuales				
Actividad 2	Extracción de imágenes de cada video para etiquetado				
Entregable # 2:	Etiquetado de Imágenes	3/04/2023	10/05/2023	37	COP 200.000
Actividad 3	Generación de labels con Labelimg				
Entregable # 3:	Entrenamiento de Red Neuronal	11/05/2023	18/05/2023	8	COP 6.000.000
Actividad 4	Separación de dataset en: entrenamiento, pruebas, validaciones.				
Actividad 5	Entrenamiento de red neuronal				
OBJETIVO 2	Diseñar una estrategia de control de impresoras 3d con implementación en un microprocesador de placa simple	19/06/2023	19/07/2023	31	COP 3.300.000
Entregable # 4	Código de Sistema de Control	19/06/2023	19/07/2023	31	COP 2.800.000
Actividad 6	Programación de Scripts para Envío de Comandos en Código G				
Actividad 7	Estructuración de Código y Desarrollo de Interfaz Gráfica				
Actividad 8	Configuración de Servidor MQTT				
OBJETIVO 3	Integrar el sistema de detección de fallos de impresión 3D con el dispositivo de control a distancia.	20/07/2023	20/08/2023	32	COP 1.670.000,00
Entregable # 5	Sistema Final Integrado	20/07/2023	20/08/2023	32	
Actividad 9	Cargar Modelo de Detección				
Actividad 10	Captura de Imágenes en Tiempo Real				
Actividad 11	Programación de Avisos por Tipo de Fallo				
OBJETIVO 4	Realizar pruebas de validación en ambiente controlado para garantizar el buen funcionamiento del dispositivo.	21/08/2023	20/09/2023	31	
Entregable #6	Análisis de Validación en Ambiente Controlado	21/08/2023	20/09/2023	31	
Actividad 12	Comprobar control de movimiento, temperatura y pausas.				
Actividad 13	Verificar el funcionamiento de los componentes de la interfaz gráfica.				
Actividad 14	Comparar errores detectados con errores reales.				
Actividades Especiales	Actualización de la Líneas Base del Proyecto				
	Admon y Gerencia del Proyecto				
	Procesos de selección objetiva				
	Estudio de Resultados del proyecto				
TOTAL				155	\$ 13.970.000

Tabla 2. Cronograma – Plan de Trabajo

5. PRESUPUESTO

En este capítulo se muestra la inversión necesaria para el desarrollo del proyecto, incluyendo costos en materiales, herramientas, tecnologías, material científico y asesoría académica.

5.1. PRESUPUESTO GENERAL

PRESUPUESTO GENERAL DEL PROYECTO					
RUBROS	Fuentes de Financiamiento				Total
	Vicerrectoría de Investigación y Transferencia UAC	Facultad / Programa	Otras fuentes Externas	Contrapartida UAC	
1. Personal Científico	\$ 0	\$ 0	\$ 0	\$ 2.699.904	\$ 2.699.904
2. Personal de Apoyo	\$ 0	\$ 0	\$ 0	\$ 1.142.272	\$ 1.142.272
3. Consultoría Especializada y Servicios Técnicos	\$ 0	\$ 0	\$ 1.000.000	\$ 0	\$ 1.000.000
4. Materiales e Insumos	\$ 0	\$ 0	\$ 1.070.000	\$ 0	\$ 1.070.000
5. Salidas de Campo	\$ 0	\$ 0	\$ 0	\$ 0	\$ 0
6. Equipos	\$ 0	\$ 0	\$ 11.900.000	\$ 0	\$ 11.900.000
7. Bibliografía	\$ 0	\$ 0	\$ 0	\$ 0	\$ 0
8. Difusión de Resultados	\$ 0	\$ 0	\$ 0	\$ 0	\$ 0
9. Viajes	\$ 0	\$ 0	\$ 0	\$ 0	\$ 0
TOTAL PRESUPUESTO DEL PROYECTO	\$ 0	\$ 0	\$ 13.970.000	\$ 3.842.176	\$ 17.812.176

Tabla 3. Presupuesto general.

5.2. PERSONAL CIENTÍFICO Y DE APOYO

El presupuesto invertido en este rubro consiste en el costo del tiempo empleado por el personal de investigación vinculados a este proyecto, que incluye a los directores y a los auxiliares de investigación.

1. PERSONAL CIENTÍFICO										
Nombres y Apellidos	Función dentro del Proyecto	Tipo de Contrato	Valor Hora (\$)	Dedicación Horas/semana	No. de Semanas	Fuentes de Financiamiento				
						Vicerrectoría de Investigación y Transferencia	Facultad / Programa	Otras Fuentes Externas	Contrapartida UAC	SUB-TOTAL
Jean Pierre Coll	Co-Investigador	Asistente	\$ 37.706	1	32				\$ 1.206.592	\$ 1.206.592
Carlos Diaz Saenz	Invest. Principal	Titular	\$ 46.666	1	32				\$ 1.493.312	\$ 1.493.312
3.			FALSO						\$ 0	\$ 0
4.			FALSO						\$ 0	\$ 0
5.			FALSO						\$ 0	\$ 0
6.			FALSO						\$ 0	\$ 0
SUB-TOTAL						\$ 0	\$ 0	\$ 0	\$ 2.699.904	\$ 2.699.904

Tabla 4. Costo personal científico.

2. PERSONAL DE APOYO										
Nombres y Apellidos	Función dentro del Proyecto	Tipo de Vinculación	Valor Hora (\$)	Dedicación Horas/semana	No. de Semanas	Fuentes de Financiamiento				
						Vicerrectoría de Investigación y Transferencia	Facultad / Programa	Otras Fuentes Externas	Contrapartida UAC	SUB-TOTAL
Henry Requena Molina	Aux. Investigación	Semillero	\$ 2.231	8	32				\$ 571.136	\$ 571.136
Kelvin Pozuelo Morales	Aux. Investigación	Semillero	\$ 2.231	8	32				\$ 571.136	\$ 571.136
3.			FALSO						\$ 0	\$ 0
4.			FALSO						\$ 0	\$ 0
5.			FALSO						\$ 0	\$ 0
SUB-TOTAL						\$ 0	\$ 0	\$ 0	\$ 1.142.272	\$ 1.142.272

Tabla 5. Costo personal de apoyo.

5.3. CONSULTORIA ESPECIALIZADA

3. CONSULTORIA ESPECIALIZADA Y SERVICIOS TECNICOS EXTERNOS						
Descripción	Justificación	Fuentes de Financiamiento				
		Vicerrectoría de Investigación y Transferencia	Facultad / Programa	Otras Fuentes Externas	Contrapartida UAC	SUB-TOTAL
1. Comunicación y almacenamiento en la nube	Suscripción Mensual a google Cloud Platform			\$ 1.000.000		\$ 1.000.000
2.						\$ 0
3.						\$ 0
SUB-TOTAL		\$ 0	\$ 0	\$ 1.000.000	\$ 0	\$ 1.000.000

Tabla 6. Costo consultoría especializada.

5.4. MATERIALES, INSUMOS Y EQUIPOS

El presupuesto dedicado a esta sección incluye los materiales utilizados, los componentes del dispositivo final y todos los insumos para la fabricación y validación final.

4. MATERIALES E INSUMOS						
Descripción	Justificación	Fuentes de Financiamiento				
		Vicerrectoría de Investigación y Transferencia	Facultad / Programa	Otras Fuentes Externas	Contrapartida UAC	SUB-TOTAL
Controlador	Dispositivo de control para el procesado del sistema			\$ 800.000		\$ 800.000
Insumos electrónicos	Dispositivos y componentes para la construcción del sistema electrónico			\$ 150.000		\$ 150.000
Insumos Varios	Materiales de Impresión para Fabricación de Piezas			\$ 120.000		\$ 120.000
SUB-TOTAL		\$ 0	\$ 0	\$ 1.070.000	\$ 0	\$ 1.070.000

Tabla 7. Costo materiales e insumos.

5. SALIDAS DE CAMPO										
Descripción	Lugar	No. de Días	No. de Personas	Costo/día por persona	Fuentes de Financiamiento					
					Vicerrectoría de Investigación y Transferencia	Facultad / Programa	Otras Fuentes Externas	Contrapartida UAC	SUB-TOTAL	
										\$ 0
										\$ 0
										\$ 0
SUB-TOTAL					\$ 0	\$ 0	\$ 0	\$ 0	\$ 0	\$ 0

Tabla 8. Costo trabajo de campo.

6. EQUIPOS						
Descripción	Justificación	Cantidad	Fuentes de Financiamiento			
			Vicerrectoría de Investigación y Transferencia	Facultad / Programa	Otras Fuentes Externas	Contrapartida UAC
Computadoras		2			\$ 6.000.000	\$ 6.000.000
Tablet		1			\$ 1.300.000	\$ 1.300.000
Herramientas y dispositivo		-			\$ 4.600.000	\$ 4.600.000
SUB-TOTAL			\$ 0	\$ 0	\$ 11.900.000	\$ 11.900.000

Tabla 9. Costo equipos usados

6. PRESENTACIÓN Y ANÁLISIS DE RESULTADOS

En este último capítulo del proyecto se presentan todas las actividades realizadas para la finalización del proyecto. Se hace un análisis de los resultados y se muestra el diseño final del dispositivo.

6.1. DISEÑO DEL PROTOTIPO

6.1.1. DISEÑO CAD

El diseño del dispositivo PrintGuard fue realizado en el software SolidWorks. Primeramente, se obtuvo de la web un modelado CAD de una Raspberry Pi 4 Model B (figura 16) y a partir del modelo se diseñó la carcasa. La carcasa tiene 4 puntos de unión internos con tornillos además de una guía en el borde para un acople inicial (figura 17).

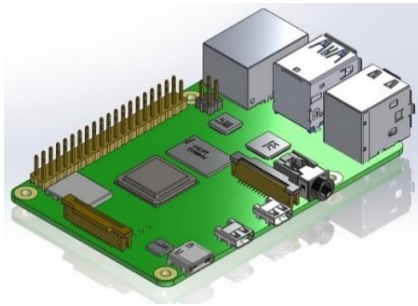


Figura 16. CAD de Raspberry Pi (Autoría Propia)

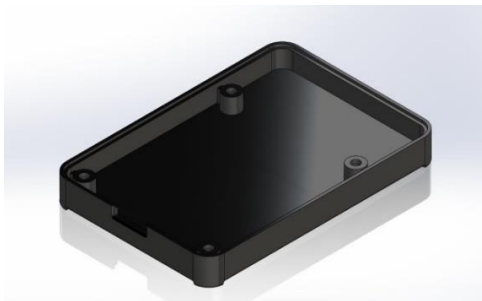


Figura 17. Tapa Inferior (Autoría Propia)

La tapa superior fue diseñada con la posibilidad de colocar un ventilador de 5V de 30mm para los casos donde la temperatura del ambiente puede ser muy alta lo cual reduce el rendimiento del dispositivo. Además, se agregó en relieve el nombre del dispositivo y el logo característico.

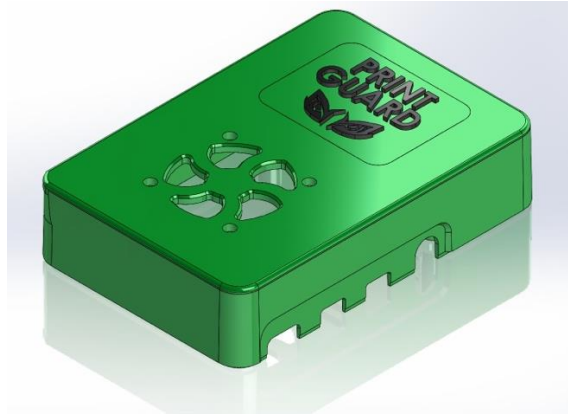


Figura 18. Tapa Superior (Autoría Propia)

6.1.2. ENSAMBLAJE

Después de diseñar cada componente del dispositivo se realizó el ensamblaje en el software Solidworks para proceder luego con la implementación del prototipo real mediante la fabricación de la carcasa con impresión 3D.

El dispositivo consta de una Raspberry Pi 4 Model B, un cable USB a micro USB (esto depende del caso del puerto usb disponible de la impresora). Un adaptador de corriente tipo C y otro puerto USB habilitado para la conexión de una cámara web.



Figura 19. Ensamblaje CAD de Dispositivo PrintGuard (Autoría Propia)

6.2. DISEÑO DEL DISPOSITIVO FINAL

En esta sección se muestra todo el proceso de diseño y fabricación del dispositivo final, incluyendo tecnologías físicas y softwares desarrollados.

6.2.1. Fabricación de Carcasa del Dispositivo

La carcasa del dispositivo final fue fabricada con impresión 3D FDM, utilizando como material PLA en presentaciones de color negro y verde fluorescente. La duración y consumo de material por cada pieza fue el siguiente:

Pieza	Tiempo (h)	Material (g)
Carcasa Inferior	4	20
Carcasa Superior	6	25
Total	10	45

Tabla 10. Fabricación de Carcasa de Dispositivo Final

Las dimensiones finales se pueden observar en la siguiente imagen con medidas en milímetros:



Figura 20. Dimensiones Generales de Dispositivo PrintGuard (Autoría Propia)

6.2.2. Modelo Entrenado para Detección de Fallos

En esta sección se muestra el procedimiento que se llevó a cabo para el entrenamiento del modelo de detección de fallos, desde la recolección de las imágenes hasta la validación de las métricas.

- **Construcción de Dataset**

Para el entrenamiento del modelo se utilizaron distintas piezas que se conocen comúnmente como las piezas de calibración para impresoras 3D. Estas piezas contienen las características necesarias para validar que una impresora no genere fallos puntuales. En el caso del presente proyecto se utilizarán con el fin de generar los fallos que se requieren detectar.

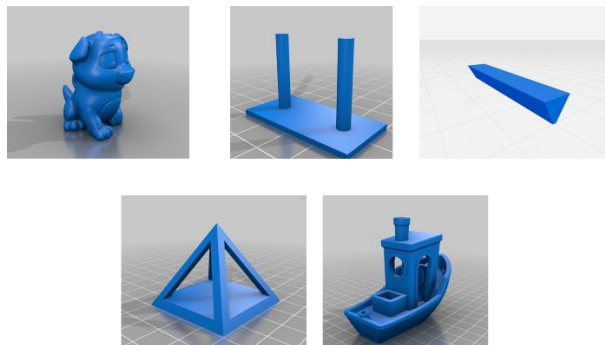


Figura 21. Piezas modelo para generación de fallos (Autoría Propia)

Como se indicó en la metodología los principales errores a detectar han sido warping, stringing y spaghetti. Aprovechando la amplia variedad de errores que se pueden generar por mala parametrización, se entrenó el modelo con errores adicionales como son layershifting, detachment y la detección de interacción humana en la placa de impresión. En la siguiente imagen se pueden observar algunas de las piezas generadas con error para el entrenamiento del modelo.

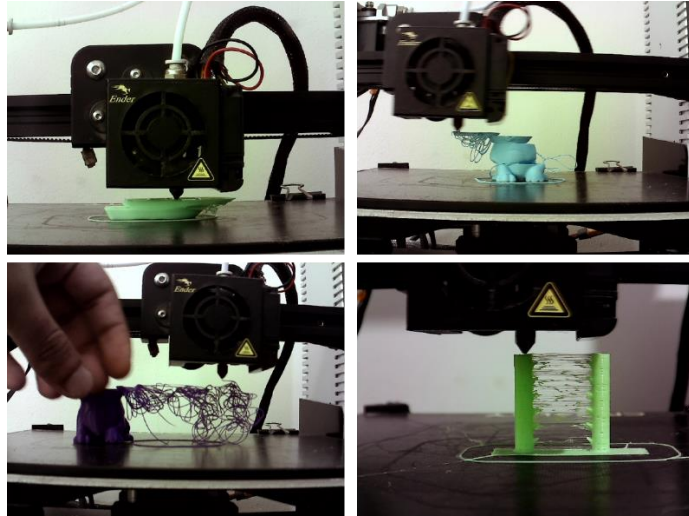


Figura 22. Piezas con Errores para Dataset (Autoría Propia)

Finalmente se obtuvo un dataset con aproximadamente 2500 imágenes de propia autoría, las cuales se ha distribuido de la siguiente manera:

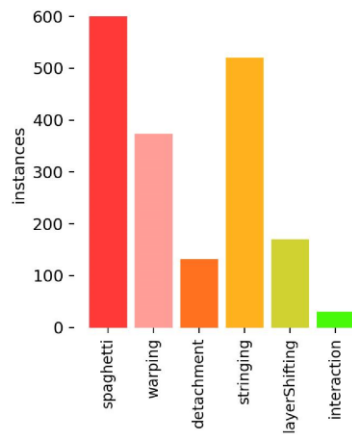


Figura 23. Tamaño del Dataset (Autoría Propia)

- **Etiquetado de Imágenes**

Para el etiquetado de las imágenes necesarias para el entrenamiento del modelo de detección llamado YoloV5 se utilizó Labellmg, creándose una carpeta destinada a guardar todas las imágenes recopiladas y otra carpeta llamada etiquetas en la cual se irán exportando los archivos de texto con las coordenadas del etiquetado. En la figura 24 se pueden observar algunas configuraciones que se deben hacer en Labellmg antes de realizar el etiquetado.

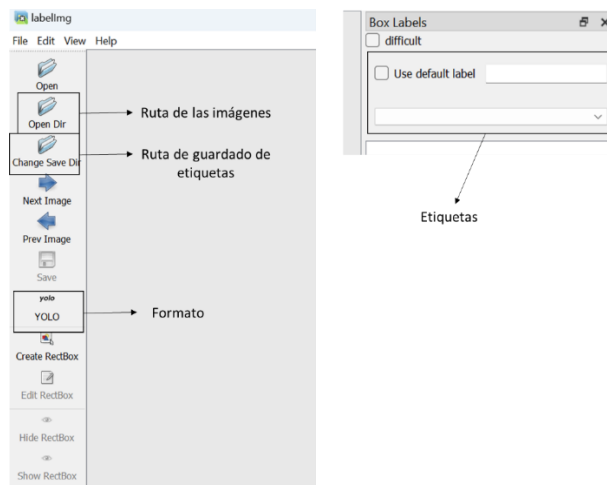


Figura 24. Configuración de Labellmg (Autoría Propia)

Una vez configurada la plataforma se procedió a etiquetar cada imagen correspondiente con el error hasta finalizar el total de imágenes.

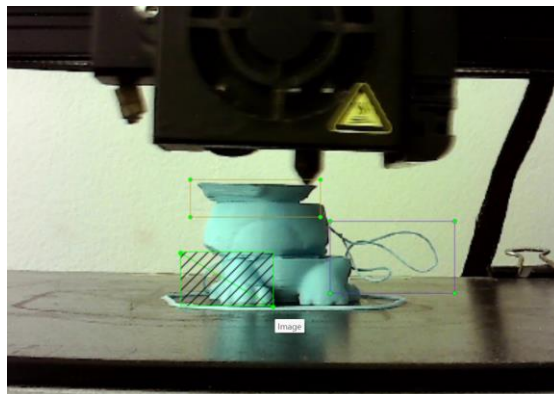


Figura 25. Etiquetado de Imágenes (Autoría Propia)

Finalmente se obtuvo una serie de archivos de texto los cuales contienen la información del etiquetado realizado. La sintaxis de un archivo de etiquetado en yolo se puede observar en la figura 26.

Índice de etiqueta	Coordenadas
0	0.500781 0.715625 0.042188 0.060417
0	0.175781 0.678125 0.035937 0.031250
2	0.189844 0.697917 0.117188 0.087500

Figura 26. Sintaxis de archivo de etiquetado Yolo (Autoría Propia)

Para la segmentación del dataset en validación y entrenamiento primero se desarrollaron unos códigos en Python que verifiquen que se encuentra la misma cantidad de imágenes como de etiquetas y si no elimine hasta igualar las relaciones imágenes-etiquetas (Figura 27 y 28).

```
In [1]: import os

def find_and_delete_missing_image_txt(txt_folder, image_folder):
    txt_files = os.listdir(txt_folder)
    image_files = os.listdir(image_folder)

    for txt_file in txt_files:
        # Obtener el nombre del archivo sin extensión
        name, extension = os.path.splitext(txt_file)

        # Verificar si existe una imagen con el mismo nombre
        image_file_path = os.path.join(image_folder, name + '.jpg') # Cambiar la extensión según el formato de tus imágenes
        if not os.path.exists(image_file_path):
            # Si no existe, eliminar el archivo .txt
            txt_file_path = os.path.join(txt_folder, txt_file)
            os.remove(txt_file_path)
            print(f"Archivo {txt_file} borrado porque la imagen no existe.")
        else:
            print(f"Archivo {txt_file} conservado.")

if __name__ == "__main__":
    # Especifica las rutas de las carpetas de texto e imágenes
    txt_folder_path = r"D:\VISION-ARTIFICIAL\rutaEtiquetas"
    image_folder_path = r"D:\VISION-ARTIFICIAL\rutaImágenes"

    find_and_delete_missing_image_txt(txt_folder_path, image_folder_path)
```

Figura 27. Verificación de existencia de imágenes (Autoría Propia)

```
In [2]: import os

def find_and_delete_missing_txt_images(image_folder, txt_folder):
    image_files = os.listdir(image_folder)
    txt_files = os.listdir(txt_folder)

    for image_file in image_files:
        # Obtener el nombre del archivo sin extensión
        name, extension = os.path.splitext(image_file)

        # Verificar si existe un archivo de texto con el mismo nombre
        txt_file_path = os.path.join(txt_folder, name + '.txt')
        if not os.path.exists(txt_file_path):
            # Si no existe, eliminar la imagen
            image_file_path = os.path.join(image_folder, image_file)
            os.remove(image_file_path)
            print(f"Imagen {image_file} borrada porque el archivo .txt no existe.")
        else:
            print(f"Imagen {image_file} conservada.")

if __name__ == "__main__":
    # Especifica las rutas de las carpetas de imágenes y texto
    txt_folder_path = r"D:\VISION-ARTIFICIAL\rutaEtiquetas"
    image_folder_path = r"D:\VISION-ARTIFICIAL\rutaImágenes"

    find_and_delete_missing_txt_images(image_folder_path, txt_folder_path)
```

Figura 28. Verificación de existencia de etiquetas (Autoría Propia)

Finalmente, con la seguridad de que existe la misma cantidad de etiquetas como imágenes se desarrolló un código para dividir las imágenes de la siguiente forma:

Dataset Final

Train:

- Images
- Labels

Valid:

- Images
- Labels

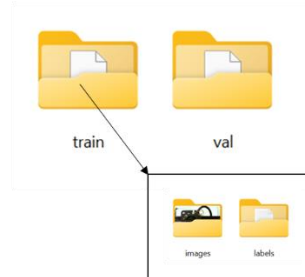


Figura 29. Segmentación de Dataset (Autoría Propia)

- **Entrenamiento del Modelo**

Para el entrenamiento del modelo se debe especificar en un archivo dataset.yaml el cual contiene las rutas de las carpetas valid y train así como el vector-cadena con los nombres de las etiquetas.

```
# train and val data
train: D:\VISION-ARTIFICIAL\TESIS-HENRY\failuresCompletos\train\images
val: D:\VISION-ARTIFICIAL\TESIS-HENRY\failuresCompletos\val\images

# number of classes
nc: 7

# class names
names: ['spaghetti', 'warping', 'detachment', 'stringing', 'layerShifting', 'interaction', 'nozzle']
```

Figura 30. Archivo de configuración para entrenamiento (Autoría Propia)

Una vez se tiene todo esto configurado, solo se tiene que correr desde la terminal el archivo train.py del repositorio de yolov5 especificando la ruta del archivo dataset.yaml, el archivo pre-entrenado YoloV5n, el número de imágenes a utilizar por ciclo (batch size), el número de iteraciones (epoch) y el tamaño de las imágenes de entrada. Las configuraciones del entrenamiento fueron:

- Batch Size: 2
- Epochs: 100
- Model: yolov5n.pt

6.2.3. Diseño de Sistema de Control

En esta sección se pretende mostrar los pasos realizados para el diseño del sistema de control de la impresora el cual comprende comunicación serial, envío de comandos, desarrollo de interfaz gráfica y configuración del bróker MQTT para intercambio de información entre la interfaz y el dispositivo.

- **Programación de Scripts para Envío de Comandos en Código G**

Para el sistema de control se utilizó la librería de Python PySerial, la cual especificándole el puerto y los baudios ya realiza automáticamente la conexión con la impresora. Se desarrolló una función encargada de hacer la escritura serial y otras funciones alternas que llaman a la función de escritura. De la siguiente forma:

```
"""FUNCIONES PARA ENVIAR GCODES A LA IMPRESORA"""  
# Función para enviar comandos GCODE a la impresora  
def send_gcode_command(command):  
    gcode_command = (command + "\n").encode('utf-8')  
    ser.write(gcode_command)  
    time.sleep(0.1)
```

Figura 31. Función de Escritura Serial (Autoría Propia)

En la figura 31 se observa la función de envío de comandos, la cual primeramente recibe como parámetro el comando en String, luego lo codifica a utf-8 y hace la escritura serial directa a la impresora.

Las funciones habilitadas para el dispositivo PrintGuard son:

- Cambio de Temperatura de Fusor
- Cambio de Temperatura de Cama
- Control de movimiento de ejes x, y, z.
- Lectura de Temperatura de Cama y Fusor
- Pausa y Detención de Impresiones

- **Estructuración de Código y Desarrollo de Interfaz Gráfica**

El código del dispositivo PrintGuard que finalmente se convierte en software del dispositivo, se estructuró de la siguiente forma:

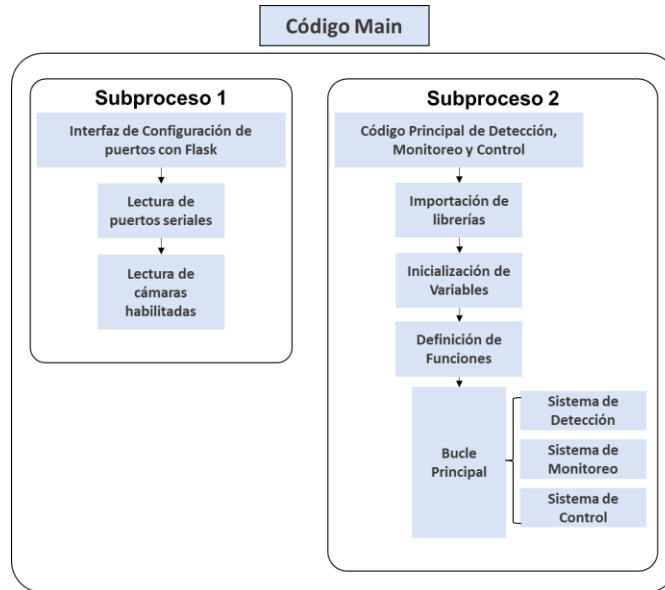


Figura 32. Estructura de Software PrintGuard (Autoría Propia)

De forma paralela al desarrollo del software del dispositivo se fue desarrollando la interfaz gráfica

Para la interfaz gráfica se utilizaron las tecnologías de Firebase y Quasar Framework, el cual utiliza como base Javascript para la codificación y diseño de páginas web. El flujo del aplicativo es el siguiente:

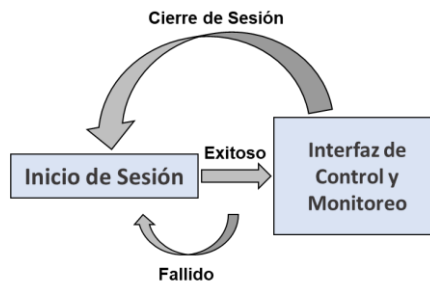
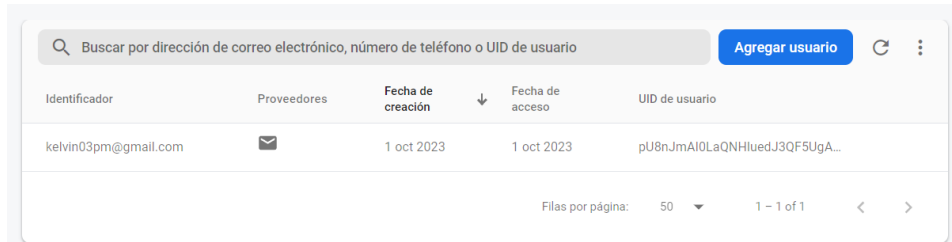


Figura 33. Flujo de Interfaz de Gráfica (Autoría Propia)

Para el diseño de la interfaz de inicio de sesión se utilizó Firebase, de manera que solo los usuarios registrados puedan ingresar a la interfaz de monitoreo.




Identificador	Proveedores	Fecha de creación	Fecha de acceso	UID de usuario
kelvin03pm@gmail.com		1 oct 2023	1 oct 2023	pU8nJmAI0LaQNHluedJ3QF5UgA...

Figura 34. Usuarios Registrados en Firebase (Autoría Propia)

Luego en el backend del aplicativo se programaron los métodos encargados de hacer el registro e inicio de sesión en la interfaz (Figura 35 y 36).

```
onSubmit() {
  if (this.username !== "" && this.password !== "") {
    signInWithEmailAndPassword(auth, this.username, this.password)
      .then((userCredential) => {
        localStorage.setItem("isLoggedIn", true);
        const userInfo = userCredential.user;
        console.log(userInfo);

        this.$q.notify({
          type: "positive",
          position: "bottom",
          message: "You've signed in and will be redirected!",
        });
      });
  }
}
```

Figura 35. Método para Inicio de Sesión en Interfaz Gráfica (Autoría Propia)

```
register() {
  if (this.username !== "" && this.password !== "") {
    createUserWithEmailAndPassword(auth, this.username, this.password)
      .then((userCredential) => {
        const userInfo = userCredential.user;
        console.log(userInfo);

        this.$q.notify({
          type: "positive",
          position: "bottom",
          message: `We have created your new account ${userInfo}`,
        });
      });
  }
}
```

Figura 36. Método para Registro de Usuarios en interfaz Gráfica (Autoría Propia)

Gracias a Quasar Framework se puede dar una vista más ordenada del proyecto, permitiendo representar cada parte de la interfaz por tipo de componentes (Scaffolding).

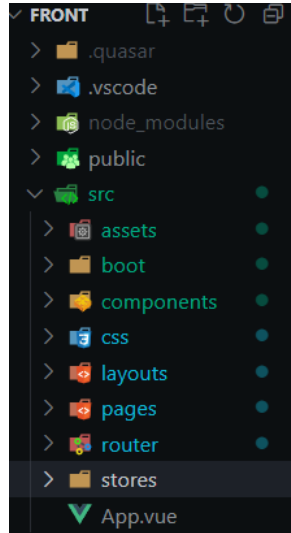


Figura 37. Scaffolding PrintGuard (Autoría Propia)

En la Figura 38 se puede observar la interfaz gráfica de control y monitoreo, la cual comprende de 6 secciones denominadas: Options, Temperature, Messages, AI Options, Axis, Temperature Control.



Figura 38. Interfaz de Control y Monitoreo PrintGuard (Autoría Propia)

- **Options**

Esta sección se encarga del envío de datos vía MQTT para detener, pausar impresiones, tomar una foto y la habilitación de avisos por telegram.

```
methods: {
  noti() {
    mqttService.publishToTopic("/3d/avisosTelegram", "true");
    console.log('You have activated telegram notifications');
  },
  pause() {
    mqttService.publishToTopic("/3d/pausePrint", "true");
    console.log('Pausing printing process');
  },
  stop() {
    mqttService.publishToTopic("/3d/stopPrint", "true");
    console.log('Stopping printing process');
  },
  photo() {
    mqttService.publishToTopic("/node/photo", "True");
    console.log('Taking snapshot of current printing');
  },
}
```

Figura 39. Apartado de Options Interfaz PrintGuard (Autoría Propia)

- **Temperature**

Con el uso de Influxb y Grafana, se desarrolló la sección de monitoreo de temperaturas. El software de control envía la información de la temperatura via MQTT a Node Red donde a través de un nodo lleva la información a una base de datos InfluxDB, esta información es extraída por Grafana y enviada al HTML de la interfaz de monitoreo,

```
<div
  class="no-marging q-pa-wm flex flex-center wrap"
  style="width: 400; height: 100px; border: 1px solid #ccc; border-radius: 5px;"
  >
  <div
    class="lg-text q-ma-wm flex text-white"
    style="width: 100%; height: 50%; border: 1px solid #ccc; border-radius: 5px; background-color: #333; color: #fff; padding: 10px;"
    >
    <span
      class="lg-sec no-marging q-pa-wm text-subtitle1"
      style="font-size: 24px; font-weight: bold; margin-bottom: 10px;"
      >TEMPERATURA (°C)
    </span>
    <div style="border: 1px solid #ccc; border-radius: 5px; width: 100%; height: 100%; position: relative; background-color: #444; color: #fff; padding: 10px;">
      <div style="position: absolute; top: 0; right: 0; font-size: 18px; font-weight: bold; color: #ffeb3b;"
        >
        <span style="font-size: 24px; font-weight: bold; margin-right: 10px;"
          >TEMPERATURA (°C)
        </span>
      </div>
    </div>
  </div>
  <div class="q-pa-wm column content-center" style="width: 100%; margin-top: 10px;"
    >
    <iframe
      src="http://34.132.7.144:3000/d-solo/a-cssdb/2/printguardlog?id=1&time=dark&panelId=2&refresh=5"
      width="100%"
      height="100%"
      frameborder="0"
    ></iframe>
  </div>
</div>
```

Figura 40. Integración de Grafica Embebida en componente Web (Autoría Propia)

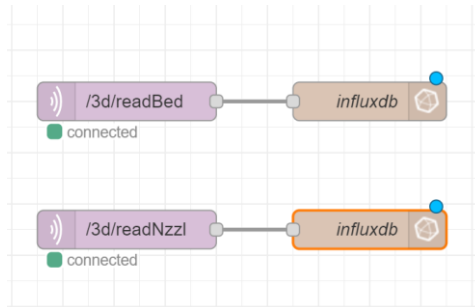


Figura 41. Nodos de Envío de Temperatura a Base de Datos InfluxDB (Autoría Propia)

- **Messages**

Esta sección muestra el historial de detecciones y otros mensajes correspondientes a la validación de los accionamientos por parte de las indicaciones enviadas desde la interfaz.

```

mounted() {
  // Obtener la fecha y hora actual
  this.fechaHoraActual = new Date();

  // Obtener los componentes de la fecha y hora
  this.hora = this.fechaHoraActual.getHours();
  this.minuto = this.fechaHoraActual.getMinutes();
  this.segundo = this.fechaHoraActual.getSeconds();

  // Formatear la fecha y hora como un timestamp
  this.timestamp = `${this.hora}:${this.minuto}:${this.segundo}`;

  mqttService.setMessageCallback((payload, destination) => {
    // Haz lo que necesites con los valores aquí
    if (destination === "/test/1") {
      console.log("funciona");
      console.log(payload);
      console.log(destination);

      this.messages.push(payload);
      console.log(this.messages);
      this.$nextTick(() => {
        //
      });
    }
  });
},

```

Figura 42. Captura de Mensajes Provenientes de Dispositivo PrinGuard (Autoría Propia)

- **AI Options**

Está compuesta por un conjunto de botones que de manera predeterminada están deshabilitados. Estos botones están relacionados con cada uno de los errores que pueden ser detectados por el sistema de visión artificial.

```
group(newGroup) {
  if (this.detection) {
    const setValues = new Set(newGroup);

    if (
      setValues.has("stringing") &&
      !this.alreadyPrinted.has("stringing")
    ) {
      console.log("activando stringing");
      this.alreadyPrinted.add("stringing");
      mqttService.publishToTopic("/3d/stringing", "true");
    }
    if (setValues.has("warping") && !this.alreadyPrinted.has("warping")) {
      console.log("activando warping");
      this.alreadyPrinted.add("warping");
      mqttService.publishToTopic("/3d/warping", "true");
    }
    if (
      setValues.has("detachment") &&
      !this.alreadyPrinted.has("detachment")
    ) {
      console.log("activando detachment");
      this.alreadyPrinted.add("detachment");
      mqttService.publishToTopic("/3d/detachment", "true");
    }
    if (
      setValues.has("interaction") &&
      !this.alreadyPrinted.has("interaction")
    ) {

```

Figura 43. Control de Detecciones (Autoría Propia)

- **Temperature Control**

La sección de controles de temperatura (°C) está compuesta por dos sliders, los cuales son componentes de entrada que permiten modificar la temperatura de la cama de impresión de 0°C a 110°C, y del extrusor de 0°C a 270°C.

```
extruderValue() {
  if (this.extruder !== this.extruderPreviousValue) {
    mqttService.publishToTopic("/3d/tempNzz1", `${this.extruder}`);
    console.log(`Extruder value: ${this.extruder}°C`);
  }
  this.extruderPreviousValue = this.extruder;
},
```

Figura 44. Método para Control de Temperatura (Autoría Propia)

- **Configuración de Servidor MQTT**

Una de las principales necesidades del proyecto fue el control a distancia de la impresora, sin necesidad de estar dentro de la red local del sitio donde se encuentran las máquinas. Es por ello que se implementó el uso del protocolo MQTT, instalando el Broker gratuito EMQX en un servidor de Google Cloud Platform.

Para la configuración del servidor se instaló una instancia de máquina virtual (IVM) con las siguientes características:

- Procesador Intel Broadwell e2-medium
- Arquitectura x86-64
- Disco Duro SCSI con 160gb de almacenamiento
- 4gb de memoria RAM.

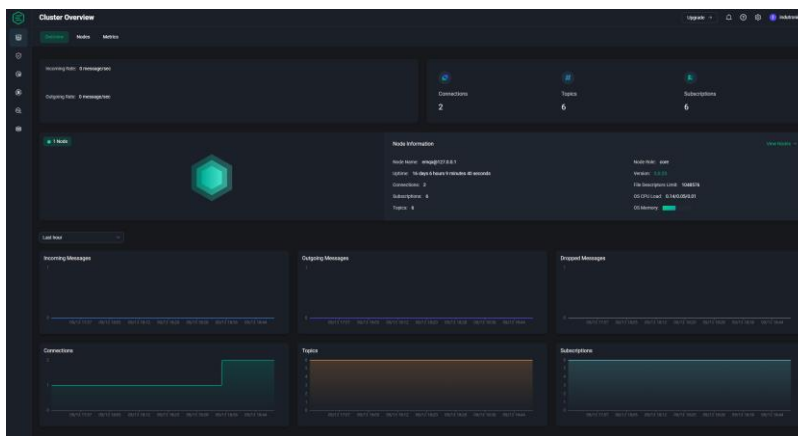


Figura 45. Broker MQTT – EMQX (Autoría Propia)

6.2.4. Integración del Sistema de Control con Modelo de Detección

En esta sección se explica detalladamente como interactúan todos los sistemas desarrollados en el proyecto, con el fin de dar paso a la consolidación del dispositivo final.

- **Comunicación entre Interfaz y Dispositivo**

Como se explicó en la metodología, las conexiones MQTT se realizan a través de tópicos los cuales representan información se va a enviar de un cliente a otro. Los tópicos utilizados fueron los siguientes:

Control de Movimiento de Ejes	Control de Temperaturas
<ul style="list-style-type: none">• Eje x: /3d/ejex• Eje y: /3d/ejey• Eje z: /3d/ejes• Autohome: /3d/autohome	<ul style="list-style-type: none">• Asignar temperatura a la boquilla: /3d/tempNzzi• Asignar temperatura a la cama: /3d/tempBed• Grafica de temperatura de la boquilla /3d/readNzzi• Grafica de temperatura de la cama /3d/readBed
Tópicos para selección de errores a detectar	Selector de Activación del Sistema de Detección
<ul style="list-style-type: none">• /3d/stringing• /3d/warping• /3d/interaction• /3d/spaghetti• /3d/detachment• /3d/layerShifting	<ul style="list-style-type: none">• /3d/selector

Figura 46. Tópicos MQTT implementados (Autoría Propia)

- **Cargue de Modelo Entrenado**

Una vez listo el sistema de control, se puede integrar el modelo de visión artificial utilizando la librería Pytorch y las funciones que aporta el repositorio de GitHub de YoloV5.

```
#Función para cargar el modelo de deteccion
def load_yolov5_model(weights_path, device='cpu'):
    # Load model
    model = custom(path=weights_path)
    return model
```

Figura 47. Función para cargar modelo de detección (Autoría Propia)

```

# Especifica la ruta a los pesos del modelo preentrenado
weights_path = r"RUTA DEL ARCHIVO BEST.PT"

# Carga el modelo YOLOv5
device = 'cuda' if torch.cuda.is_available() else 'cpu'
model = load_yolov5_model(weights_path, device)

```

Figura 48. Configuración e Iniciación del Modelo (Autoría Propia)

- **Captura de Imágenes en Tiempo Real**

Con el uso de la librería OpenCV se pueden obtener los frames con la cámara web y enviarlos al modelo para obtener las predicciones en tiempo real.

```

cap = cv2.VideoCapture(camera_index)
# Abre la cámara
ret, frame = cap.read()
frame = frame.copy()
if not ret:
    print("not ret")
    continue
# Realiza la detección en el frame actual
results = model(frame)

# Obtiene las detecciones
detections = results.pred[0]
for *xyxy, conf, cls in detections:
    label = f'{model.names[int(cls)]} {conf:.2f}'
    xyxy = [int(i) for i in xyxy]
    frame = cv2.rectangle(frame, (xyxy[0], xyxy[1]), (xyxy[2], xyxy[3]), (255, 0, 0), 2)
    frame = cv2.putText(frame, label, (xyxy[0], xyxy[1] - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 0, 0), 2)

```

Figura 49. Implementación del Modelo para Obtener las Predicciones (Autoría Propia)

- **Creación de Avisos por Tipo de Fallo**

Dentro de las variables que se obtienen durante la predicción se tiene una que se llama “label” esta variable es de tipo cadena e incluye el nombre del error y el porcentaje de confiabilidad, con el fin de enviar avisos se han implementado dos técnicas:

- Envío de aviso a través de un tópico hacia la interfaz gráfica. Para esto se configuró un nuevo tópico denominado “/3d/detección”.
- Implementación de la API de Telegram. A través de esta implementación permite generar avisos con un bot de Telegram y enviar una foto con la detección obtenida.

- **Configuraciones Finales del Software**

Esta última parte de la etapa de integración hace referencia a las configuraciones necesarias para que el dispositivo trabaje de la forma más interactiva posible con el usuario final, así como mejorar la robustez del sistema.

El código tiene la limitante de que si se inserta la cámara en otro puerto USB o esta cambia de índice en el arranque el código no podrá detectar el cambio, de igual forma se limita a la escritura manual del puerto serial para la configuración de la conexión por lo que se diseñó un web inicial de configuración con la librería de Python Flask. La cual hace la lectura de las cámaras y puertos disponibles y muestra en una interfaz las opciones a escoger.

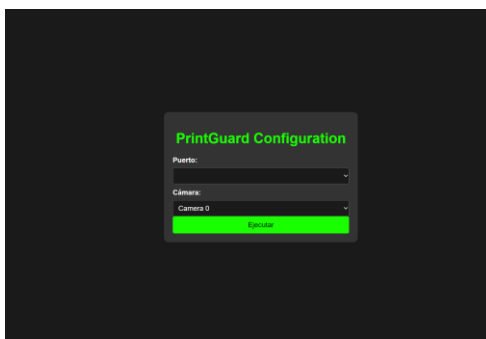


Figura 50. Interfaz Web de Configuración (Autoría Propia)

Para poder enviar la información de la configuración al código principal se utiliza el botón ejecutar, que guarda la configuración en un archivo de texto que luego es leído al inicio del código principal para la asignación a las variables correspondientes (Figura 51).

```
def leer_configuracion():
    config = {}
    try:
        with open("config.txt", "r") as f:
            lines = f.readlines()
            for line in lines:
                key, value = line.strip().split("-")
                config[key] = value
    except FileNotFoundError:
        print("Archivo de configuración no encontrado.")
    return config

config = leer_configuracion()
```

Figura 51. Lectura de Configuración de Puerto Serial y Cámara (Autoría Propia)

El software hasta el momento estuvo corriendo a través de la opción RUN desde Visual Studio Code, pero para al dispositivo final PrintGuard es necesario que este código corra como un servicio dentro de la Raspberry Pi con Ubuntu 22.04, de manera que al solo encender el dispositivo inicie el software. Para esto se desarrolló un código sencillo llamado main.py el cual utiliza la librería subprocess donde corre inicialmente el servidor de configuración Flask y luego el código Principal.

```
import subprocess

# Correr el primer script
flask_process = subprocess.Popen(["python", "app.py"])

# Esperar a que el servidor Flask termine
flask_process.wait()

# Correr el segundo script
subprocess.run(['python', 'codigoPrincipal.py'])
```

Figura 52. Código de subprocessos main.py (Autoría Propia)

Finalmente, para la configuración del software como servicio se realizaron los pasos indicados en la página “Digital Ocean” específicamente en el paso 5 de la documentación, donde se crea el archivo PrintGuard.service el cual se mantiene ejecutando siempre que el dispositivo esté encendido [65].

6.3. MATERIALES

En la presente sección se muestran los componentes utilizados para el desarrollo del proyecto, tanto dispositivos electrónicos como insumos.

6.3.1. Raspberry Pi 4 Model B



Figura 53. Raspberry Pi 4 Model B [66]

Tecnología	Descripción
Procesador	Broadcom BCM2711B0, quad-core Cortex-A72
Frecuencia de Reloj	1,5 GHz
GPU	VideoCore VI 500 MHz
Memoria	1/2/4 GB LPDDR4-3200
Conectividad Inalámbrica	Wi-Fi 2,4GHz / 5GHz IEEE 802.11.b/g/n/ac Bluetooth 5.0, BLE
Conectividad de Red	Gigabit Ethernet
Puertos	GPIO 40 pines 2 x Micro HDMI 2 x USB 2.0 2 x USB 3.0

Tabla 11. Especificaciones Raspberry Pi 4 Model B [67]

6.3.2. Impresora 3D Ender 3



Figura 54. Creality Ender 3 [68]

Descripción	Especificaciones
Dimensiones de la cama de impresión	200 x 200 x 250 mm
Peso	8kg
Resolución	Por encima de 0.1 mm
Velocidad de Impresión	Máximo 200 mm/s
Cama Caliente	Si
Tamaño de Boquilla	0.4 mm
Extrusor	Bowden (1)
Temperatura Máxima del Fusor	255 °C
Temperatura Máxima de la Cama	110 °C
Conexión	USB y SDCard

Tabla 12. Especificaciones Creality Ender 3 [69]

6.3.3. Filamento de Impresión PLA



Figura 55. Filamento PLA

Property	Value
Full Name	Polylactic acid (PLA)
Melting Point	150 to 160 °C (302 to 320 °F)
Glass Transition	60-65 °C
Injection Mold Temperature	178 to 240 °C (353 to 464 °F)
Density	1.210–1.430 g·cm ⁻³
Chemical Formula	(C ₃ H ₄ O ₂) _n
Crystallinity	37%
Tensile Modulus	2.7–16 GPa
Solubility	Chlorinated solvents, hot benzene, tetrahydrofuran, and dioxane (not water soluble).

Tabla 13. Especificaciones del PLA [70]

6.3.4. Cámara Web HD 1080P



Figura 56. Cámara Web

Cámara web Full HD 1080P. Vídeo realista de 1920 x 1080p, lente antirreflejos de 4 capas, proporcionando un video suave. La longitud focal fija hace que el objeto esté en el rango de longitud focal de 1.97 a 197 pulgadas, para proporcionar una imagen más clara.

6.4. RECOLECCIÓN DE DATOS

En esta sección se muestra la información recopilada en el uso del dispositivo final en cuanto a comportamiento del modelo entrenado, interacción de la interfaz con el dispositivo y reacciones de la impresora 3D. Esto con la finalidad de dar respuesta al último objetivo de la presente investigación.

6.4.1. Métricas del Modelo Entrenado

Luego del entrenamiento del modelo. Yolo con el uso de la librería matplotlib genera la matriz de confusión correspondiente a cada una de las etiquetas del modelo entrenado (figura 57).

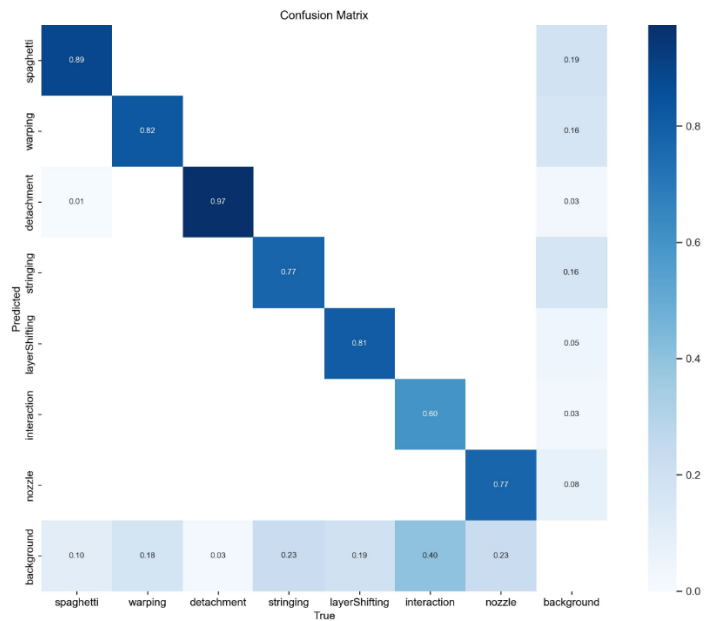


Figura 57. Matriz de Confusión (Autoría Propia)

6.4.2. Lectura de Temperaturas y Envío por MQTT

Obteniendo los valores de temperatura de boquilla y cama de impresión se pudo observar en la base de datos serializada influxdb un histórico de las pruebas realizadas (figura 58).



Figura 58. Historico de Lectura de Temperaturas (Autoría Propia)

6.4.3. Métricas de Broker MQTT

Dentro del Broker MQTT instalado en el servidor se pueden observar distintas métricas vitales para el análisis del comportamiento del sistema. Entre ellas podemos encontrar:

- **Suscripciones**
Se refiere al número de suscripciones entre todos los dispositivos a cada tópico, es decir, el total de tópicos a los cuales los dispositivos están pendientes si hay algún cambio para poder detectarlo.

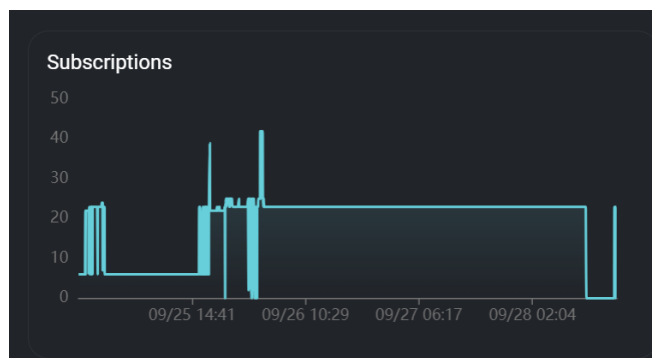


Figura 59. Comportamiento de Suscripciones a Tópicos (Autoría Propia)

6.5. ANÁLISIS DE RESULTADOS

En esta sección se muestran los resultados de las pruebas realizadas con el dispositivo, así como el análisis de los datos recolectados en la sección 4 del presente capítulo.

6.5.1. ANÁLISIS DE LAS PRUEBAS REALIZADAS POR EL DISPOSITIVO FINAL

- **Análisis de Matriz de Confusión de Modelo Entrenado**

A través de las pruebas de detección realizadas se pudo obtener un acercamiento al comportamiento del modelo, así como las características del ambiente para garantizar el mejor rendimiento de este.

En la figura 57 de la sección de recopilación de datos se puede observar la matriz de confusión del modelo. Dentro de la matriz se puede observar que el modelo tiene un buen rendimiento en cuanto a la seguridad en las detecciones de los errores mostrando que en el caso del Spaghetti tuvo un 89% de seguridad en la detección de este error y solo tuvo un 1% de seguridad en cosas que no eran este tipo de error, por ejemplo, Detachment. Este fue el único caso donde el modelo mostró una pequeña confusión con otro tipo de error. Y se puede explicar debido a que en el 100% de los casos cuando existe un detachment se presenta spaghetti por la impresión de la pieza en el aire. El error de spaghetti fue etiquetado en circunstancias donde el detachment siempre estuvo presente por lo que no significa una mala detección en cuanto a los avisos que pueda generar este tipo de detecciones a futuro.

- **Análisis de Precisión de Modelo Entrenado en Base a Métricas de Uniones sobre Intersecciones**

Para el presente análisis se generaron distintos errores con el dispositivo en funcionamiento, esto con la finalidad de comparar las imágenes detectadas con los errores reales. Cabe destacar que a pesar de que el modelo está preparado para la detección de distintos tipos de errores, se realizará el análisis solamente al enfoque del proyecto ya que fueron los errores con mayor número de imágenes en el entrenamiento.

Con el fin de evaluar los resultados de las detecciones de la red neuronal desarrollada, se utilizarán distintos métodos para el análisis de las métricas. En el análisis anterior se utilizó la matriz de confusión, adicional a esto se utilizará el análisis de Uniones bajo Intersecciones.

Como el modelo entrenado utiliza recuadros para mostrar las detecciones, se pueden comparar los recuadros de las detecciones con respecto a la imagen real y obtener así un porcentaje de precisión.

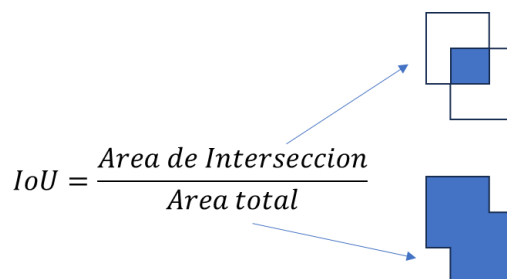
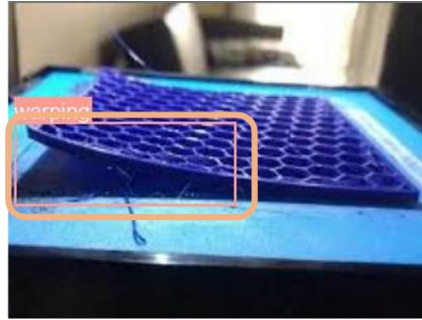


Figura 62. IoU (Autoría Propia)

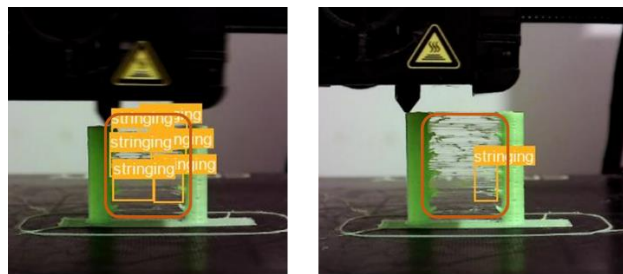
La fórmula de IoU hace referencia al cociente de las áreas en común entre el total del área de ambos recuadros. Si el IoU es menor a 0.5 se considera un falso positivo.



IoU > 0.5 (TP)

Figura 63. IoU de Warming (Autoría Propia)

Como se pudo observar en esta primera detección en el caso del warping se obtuvo un IoU mayor a 0.5 ya que el área de intersección entre la detección y la situación real eran muy parecidas.

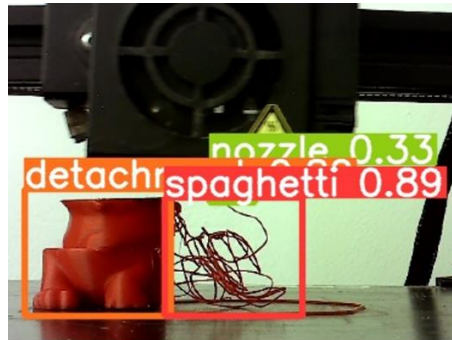


IoU > 0.5

IoU < 0.5

Figura 64. IoU de Stringing (Autoría Propia)

En el caso particular del stringing se puede observar que el modelo puede tener momentos en que no detecta toda la zona del error, y casos donde puede detectar completamente los errores presentes. En este caso y haciendo una media entre ambas predicciones aún así se mantiene un IoU mayor a 0.5.



IoU > 0.5

Figura 65. IoU de spaghetti (Autoría Propia)

Finalmente, para el caso del spaghetti siendo este error del que tuvo más información el modelo, tiene una excelente precisión a la hora de detectarlo.

- **Análisis de Comportamiento de Conexión MQTT**

En base a los gráficos obtenidos en la sección de recolección de datos sobre el comportamiento de las distintas métricas del bróker, se puede observar como la suscripción a los tópicos por parte de ambos clientes (Interfaz y Dispositivo) se mantienen constantes.

Por otro lado, en la gráfica de pérdida de mensajes solo se puede observar un segmento de pérdidas, esto durante la validación del dispositivo despertó la inquietud de los investigadores hasta dar con la problemática. Algunas impresoras 3D luego de enviar el comando de pause de impresión no permiten reanudar la impresión de forma remota o serial, se limita a reanudar la impresión pulsando el botón físico en la máquina, esto generaba que el sistema se quedara en espera de una respuesta por parte de la comunicación serial lo cual no permitía la recepción de datos en ese lapso.

CONCLUSIONES Y RECOMENDACIONES

Este proyecto ha culminado en un dispositivo de control remoto altamente efectivo para la detección y corrección de errores en impresoras 3D FDM en tiempo real. Mediante la implementación de una Raspberry Pi y el entrenamiento de una red neuronal con el modelo preentrenado YOLOv5 a partir de un dataset de imágenes de autoría propia, se logró detectar en mayor medida errores como el warping, spaghetti, stringing y como adicionales detachment y layershifting. Además, la detección de la interacción humana, a través de la detección de mano, añade un nivel adicional de supervisión y control durante el proceso de impresión.

El dispositivo ha demostrado su capacidad para cumplir con todas las funciones previamente establecidas. Con el diseño satisfactorio de la estrategia de control, permitió la detención y pausa de la impresora en tiempo real, así como control de ejes, monitoreo y cambio de temperatura. A partir de la integración de la estrategia de control con el sistema de visión artificial se facilitó la corrección de errores, previniendo pérdidas de material y tiempo.

Las pruebas de validación demostraron una alta capacidad del dispositivo desarrollado para emitir avisos, que permitieron al usuario detener de forma confiable las impresiones fallidas. Este logro es especialmente valioso en entornos de producción y fabricación, donde la eficiencia y la calidad son esenciales.

Se sugiere explorar en futuras investigaciones la posibilidad de abordar de manera más específica los errores de detachment y layershifting, que actualmente presentan desafíos en su corrección en tiempo real. Investigar estrategias adicionales, como el control del extrusor y la velocidad del ventilador, podría aumentar la versatilidad y capacidad de corrección del dispositivo.

En resumen, este proyecto ha alcanzado un hito importante en la mejora del proceso de impresión 3D FDM, al tiempo que abre la puerta a futuras investigaciones y mejoras que prometen una mayor eficiencia y precisión en la fabricación aditiva. La continua expansión y refinamiento del dispositivo de control remoto tienen el potencial de tener un impacto significativo en la industria de la impresión 3D.

BIBLIOGRAFÍA

- [1] Autodesk, «Autodesk,» [En línea]. Available: <https://latinoamerica.autodesk.com/solutions/3d-printing>. [Último acceso: 28 Agosto 2023].
- [2] F. P. Zuñiga, «Ambito Juridico,» 31 Marzo 2020. [En línea]. Available: <https://cutt.ly/RwktocM2>. [Último acceso: 28 Agosto 2023].
- [3] «Curiosoando.com,» 26 Noviembre 2019. [En línea]. Available: <https://curiosoando.com/que-es-un-polimero>. [Último acceso: 28 Agosto 2023].
- [4] NVIDIA, «NVIDIA,» [En línea]. Available: <https://www.nvidia.com/es-la/glossary/data-science/computer-vision/>. [Último acceso: 28 Agosto 2023].
- [5] «Salesforce,» 02 Julio 2018. [En línea]. Available: <https://www.salesforce.com/mx/blog/2018/7/Machine-Learning-y-Deep-Learning-aprende-las-diferencias.html>. [Último acceso: 28 Agosto 2023].
- [6] R. Hat, «Red Hat,» 20 Enero 2023. [En línea]. Available: <https://www.redhat.com/es/topics/internet-of-things/what-is-iot>. [Último acceso: 30 Agosto 2023].
- [7] «imprint3d,» [En línea]. Available: <https://imprint3d.net/impresoras-3d/que-es-la-impresion-3d-fdm/>. [Último acceso: 25 Septiembre 2023].
- [8] «AutoDesk,» [En línea]. Available: <https://www.autodesk.es/solutions/additive-manufacturing>. [Último acceso: 25 Septiembre 2023].
- [9] «Aprendo Mania,» [En línea]. Available: https://aprendomania.com/que-es-la-garantia-de-calidad-definicion-principios-y-factores/?expand_article=1. [Último acceso: 25 Septiembre 2023].
- [10] «Concepto,» [En línea]. Available: <https://concepto.de/gestion-de-calidad/>. [Último acceso: 25 Septiembre 2023].
- [11] «AWS Amazon,» [En línea]. Available: <https://aws.amazon.com/es/what-is/neural-network/>. [Último acceso: 25 Septiembre 2023].
- [12] D. Systemes, «Dassault Systemes,» [En línea]. Available: <https://www.3ds.com/es/make/guide/process/3d-printing>. [Último acceso: 30 Agosto 2023].

- [13] «Vision based error detection for 3D printing processes,» *Matec*, vol. 59, n° 06003, p. 7, 2016.
- [14] Lanner, «Lanner,» [En línea]. Available: <https://www.lanner-america.com/es/blog-es/solucion-de-monitorizacion-remota-para-impresoras-3d/>. [Último acceso: 30 Agosto 2023].
- [15] D. R. B. S. M. K. Ian Gibson, Ian Gibson, David Rosen, Brent Stucker, Mahyar Khorasani, Switzerland: Springer Nature, 2020.
- [16] E. Westphal, «Machine learning for the intelligent analysis of 3D printing conditions using environmental sensor data to support quality assurance,» *Science Direct*, vol. 50, n° <https://doi.org/10.1016/j.addma.2021.102535>, 2022.
- [17] J. McKay, «Advanced Motion Controls,» [En línea]. Available: <https://www.a-m-c.com/motion-control-in-3d-printing/>. [Último acceso: 30 Agosto 2023].
- [18] U. Delli, «Automated Process Monitoring in 3D Printing Using Supervised Machine Learning,» *Science Direct*, Texas, 2018.
- [19] M. Mastura, «Chapter 16 - Life cycle analysis of fused filament fabrication: A review,» *Science Direct*, pp. 415-434, 2021.
- [20] Sculpteo, «The State of 3d Printing,» Villejuif, Francia, 2019.
- [21] Sculpteo, «The State of 3D Printing,» Villejuif, Francia, 2022.
- [22] K. V. Wong, «A Review of Additive Manufacturing,» *Hindawi*, vol. 2012, n° <https://doi.org/10.5402/2012/208760>, 2012.
- [23] A. M. M. M. M. Picard, «Recent advances in additive manufacturing of engineering thermoplastics: challenges and opportunities,» *Publishing*, n° <https://doi.org/10.1039/D0RA04857G>, 2020.
- [24] I. K. T. K. D. T. P. Charalampous, «Learning-based error modeling in FDM 3D printing process,» *Emerald Insight*, vol. 1, n° ISSN: 1355-2546, pp. 507-517, 2021.
- [25] N. Y. Arup Dey, «A Systematic Survey of FDM Process Parameter Optimization and Their Influence on Part Characteristics,» *MDPI*, n° <https://doi.org/10.3390/jmmp3030064>, 2019.
- [26] H. S. E. Westphal, «A machine learning method for defect detection and visualization in selective laser sintering based on convolutional neural

- networks,» *Science Direct*, vol. 41, n° visualization in selective laser sintering based on convolutional neural networks,, 2021.
- [27] Y. W. Haixi Wu, «In situ monitoring of FDM machine condition via acoustic emission,» *Springer Link*, 2015.
- [28] G. C. Y. L. X. C. C. L. X. Qi, « Applying neural-network-based machine learning to additive manufacturing: current applications, challenges, and future perspective,» *Science Direct*, vol. 5, n° 4, 2019.
- [29] J. Straub, «Initial Work on the Characterization of Additive Manufacturing,» *MDPI*, vol. 3, n° 3020055, pp. 55-71, 2015.
- [30] D. M. Bermudo, «Gestion de múltiples impresoras 3D y detección de fallos a través de IA,» Universidad Autonoma de Barcelona, Barcelona, 2021.
- [31] U. Delli, «Automated Process Monitoring in 3D Printin Using Supervised Machine Learning,» *Science Direct*, pp. 865-870, 2018.
- [32] Y. Fu, «In situ monitoring for fused filament fabrication process: A review,» *Science Direct*, vol. 38, n° 101749, 2021.
- [33] M. F. Khan, «Real-time defect detection in 3D printing using machine learning,» *Science Direct*, vol. 42 part 2, pp. 521-528, 2021.
- [34] K. Paraskevoudis, «Real-Time 3D Printing Remote Defect Detection (Stringing) with Computer Vision and Artificial Intelligence,» *MDPI*, n° 81111464, 2020.
- [35] D. Song, «Forecasting Warping Deformation Using Multivariate Thermal Time Series and K-Nearest Neighbors in Fused Deposition Modeling,» *MDPI*, vol. 10, n° 24, 2020.
- [36] F. B. a. D. Roller, «Vision based error detection for 3D printing processes,» *Matec* , vol. 59, n° 06003, 2016.
- [37] «Image-based failure detection for material extrusion process using a convolutional neural network,» *Hyungjung Kim, Hyunsu Lee, Ji-Soo Kim & Sung-Hoon Ahn*, pp. 1291-1302, 2020.
- [38] M. Verana, «Deep Learning-Based 3D Printer Fault Detection,» *IEEE*, n° 21141163, 2020.
- [39] D. Systemes, «Dassault Systemes,» [En línea]. Available: <https://www.3ds.com/es/make/service/3d-printing-service/fdm-fused-deposition-modeling>. [Último acceso: 31 Agosto 2023].

- [40] 3. SYSTEMS, «3D SYSTEMS,» [En línea]. Available: <https://es.3dsystems.com/quickparts/learning-center/what-is-stl-file>. [Último acceso: 31 Agosto 2023].
- [41] «TutoPremium,» [En línea]. Available: <https://tutopremium.com/archivo-obj/>. [Último acceso: 31 Agosto 2023].
- [42] L. Llamas, «Luis Llamas,» 12 Diciembre 2019. [En línea]. Available: <https://www.luisllamas.es/que-es-el-g-code-y-su-importancia-en-la-impresion-3d/>. [Último acceso: 31 Agosto 2023].
- [43] «Servitec3D,» [En línea]. Available: <https://servitec3d.com/blog/que-es-el-warping-como-evitarlo/>. [Último acceso: 31 Agosto 2023].
- [44] A. M, «3dnatives,» 11 Enero 2023. [En línea]. Available: <https://www.3dnatives.com/es/warping-impresion-3d-que-hacer-100120232/>. [Último acceso: 31 Agosto 2023].
- [45] M. Jose, «3dnatives,» 23 Noviembre 2022. [En línea]. Available: <https://www.3dnatives.com/es/stringing-impresion-3d-231120222/>. [Último acceso: 31 Agosto 2023].
- [46] «Prusa,» 2020. [En línea]. Available: https://help.prusa3d.com/article/layer-shifting_2020. [Último acceso: 30 Agosto 2023].
- [47] «All3DP,» 2023 Junio 29. [En línea]. Available: <https://all3dp.com/2/layer-shifting-3d-printing-tips-tricks-to-solve-it/>. [Último acceso: 2023 Septiembre 12].
- [48] Prusa, «Prusa Research,» [En línea]. Available: https://help.prusa3d.com/article/spaghetti-monster_1999. [Último acceso: 2023 Agosto 30].
- [49] Martin, «The 3D Printer Bee,» [En línea]. Available: <https://the3dprinterbee.com/3d-printing-spaghetti/>. [Último acceso: 30 Agosto 2023].
- [50] Amazon, «Amazon Web Service,» [En línea]. Available: <https://aws.amazon.com/es/what-is/mqtt/>. [Último acceso: 2023 Agosto 31].
- [51] «OpenJS Foundation,» [En línea]. Available: <https://nodered.org/>. [Último acceso: 2023 Agosto 31].
- [52] Oscar, «Codigo Electronica,» 03 Agosto 2019. [En línea]. Available: <http://www.codigoelectronica.com/blog/que-es-node-red>. [Último acceso: 12 Septiembre 2023].

- [53] Oracle, «Developer Resource Center,» 31 Mayo 2022. [En línea]. Available: <https://developer.oracle.com/es/learn/technical-articles/what-is-python>. [Último acceso: 31 Agosto 2023].
- [54] J. Larkin, «Incentro,» 19 Agosto 2020. [En línea]. Available: <https://www.incentro.com/es-ES/blog/que-es-google-cloud-platform>. [Último acceso: 31 Agosto 2023].
- [55] Telegram, «Telegram.org,» [En línea]. Available: <https://core.telegram.org/>. [Último acceso: 31 Agosto 2023].
- [56] O. Weis, «Serial Port Monitor,» 04 Febrero 2020. [En línea]. Available: <https://www.serial-port-monitor.org/es/articles/serial-communication>. [Último acceso: 2023 Agosto 31].
- [57] D. R. Center, «Oracle,» 04 Mayo 2022. [En línea]. Available: <https://developer.oracle.com/es/learn/technical-articles/what-is-pytorch>. [Último acceso: 31 Agosto 2023].
- [58] «IBM,» [En línea]. Available: <https://www.ibm.com/es-es/topics/computer-vision>. [Último acceso: 31 Agosto 2023].
- [59] Arrendajo, «HashDork,» 06 Mayo 2022. [En línea]. Available: <https://hashdork.com/es/yolo/>. [Último acceso: 31 Agosto 2023].
- [60] L. Gracia, «Un Poco de Java,» 19 Octubre 2013. [En línea]. Available: <https://unpocodejava.com/2013/10/09/que-es-opencv/>. [Último acceso: 31 Agosto 2023].
- [61] Amazon, «Amazon Web Service,» [En línea]. Available: <https://aws.amazon.com/es/what-is/iot/>. [Último acceso: 31 Agosto 2023].
- [62] «GitHub,» [En línea]. Available: <https://github.com/HumanSignal/labelImg>. [Último acceso: 13 Septiembre 2023].
- [63] Ultralytics, «GitHub,» [En línea]. Available: <https://github.com/ultralytics/yolov5>. [Último acceso: 13 Septiembre 2023].
- [64] «Electronica y tecnologia,» [En línea]. Available: <https://jcelectronix.blogspot.com/p/comunicacion-serial-con-arduino.html>. [Último acceso: 13 Septiembre 2023].
- [65] D. Ocean, 5 Diciembre 2019. [En línea]. Available: <https://cutt.ly/zwxz39Om>. [Último acceso: 13 Septiembre 2023].

- [66] R. Solé, «Profesional Review,» 18 Julio 2021. [En línea]. Available: <https://www.profesionalreview.com/2021/07/18/que-es-raspberry-pi/>. [Último acceso: 31 Agosto 2023].
- [67] J. Pastor, «Xataka,» 12 Julio 2019. [En línea]. Available: <https://www.xataka.com/ordenadores/raspberry-pi-4-analisis-caracteristicas-precio-especificaciones>. [Último acceso: 31 Agosto 2023].
- [68] Creality, 2017. [En línea]. Available: <https://www.creality.com/products/ender-3-3d-printer>. [Último acceso: 31 Agosto 2023].
- [69] «3D Sourced,» 25 Junio 2023. [En línea]. Available: <https://www.3dsourced.com/3d-printer-reviews/creality-ender-3-review-price-specs/>. [Último acceso: 31 Agosto 2023].
- [70] J. Flynt, «3dInsider,» 9 Noviembre 2017. [En línea]. Available: <https://3dinsider.com/what-is-pla/>. [Último acceso: 31 Agosto 2023].

ANEXOS

Anexo 1. Código Main.py

```
import subprocess

# Correr el primer script
flask_process = subprocess.Popen(["python", "app.py"])

# Esperar a que el servidor Flask termine
flask_process.wait()

# Correr el segundo script
subprocess.run(['python', 'codigoPrincipal.py'])
```

Anexo 2. Backend de Aplicativo Web de Configuración

```
import os
from flask import Flask, render_template, request
import serial.tools.list_ports
import cv2
import time

if not os.path.exists("config.txt"):
    with open("config.txt", "w") as f:
        f.write("serial_port=\ncamera_index=\n")

app = Flask(__name__)

@app.route("/", methods=["GET", "POST"])
def index():
    closing = False
    if request.method == "POST":
        selected_port = request.form.get("selected_port")
        camera_index = request.form.get("camera_index")

        # Guardar los valores en un archivo de texto
        with open("config.txt", "w") as f:
            f.write(f"serial_port={selected_port}\n")
            f.write(f"camera_index={camera_index}\n")

        print("Valores seleccionados: Puerto {}, Cámara {}".format(selected_port, camera_index))

        closing = True
        time.sleep(3)

        # Detener la aplicación de Flask de manera controlada
        os.kill(os.getpid(), 9)

    # Enumerar los puertos seriales disponibles
    available_ports = list(serial.tools.list_ports.comports())

    # Enumerar las cámaras disponibles
    available_cameras = [f"Camera {i}" for i in range(0, 10) if cv2.VideoCapture(i).isOpened()]

    return render_template("index.html", available_ports=available_ports, available_cameras=available_cameras)

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=5000, debug=True)
```

Anexo 3. Frontend de Aplicativo Web de Configuración

```

<!DOCTYPE html>
<html>
<head>
<title>PrintGuard Configuration</title>
<style>
body {
font-family: Arial, sans-serif;
background-color: #e1e1e1;
margin: 0;
padding: 0;
display: flex;
flex-direction: column;
align-items: center;
justify-content: center;
min-height: 100vh;
}

.container {
background-color: #e33333;
border-radius: 10px;
box-shadow: 0px 2px 6px #000000;
padding: 20px;
width: 400px;
position: relative;
}

h1 {
text-align: center;
color: #e1e1e1;
}

label {
display: block;
margin-bottom: 8px;
font-weight: bold;
color: #e1e1e1;
}

select {
width: 100%;
padding: 8px;
border: 1px solid #006666;
border-radius: 4px;
font-size: 16px;
color: #e1e1e1;
background-color: #e1e1e1;
}

input[type="submit"] {
display: block;
width: 100%;
padding: 10px;
background-color: #e1e1e1;
color: #000000;
border: none;
border-radius: 4px;
font-size: 16px;
cursor: pointer;
}

/* Estilos para el mensaje de cierre */
#closing-message {
display: none;
margin-top: 10px;
text-align: center;
color: #0066ff;
font-weight: bold;
}

@keyframes spin {
0% { transform: rotate(0deg); }
100% { transform: rotate(360deg); }
}

.spinner {
border: 4px solid #000000;
border-top: 4px solid #e1e1e1;
border-radius: 50%;
width: 20px;
height: 20px;
animation: spin 1s linear infinite;
}
</style>
</head>
<body>
<div class="container">
<h1>PrintGuard Configuration</h1>
<form method="post" onsubmit="showClosingMessage();">
<label for="selected_port">Puerto:</label>
<select name="selected_port">
{&#x2013; for port in available_ports &#x2013;}
<option value="{ port }">{ port }</option>
{&#x2013; endfor &#x2013;}
</select>
<br>
<label for="camera_index" style="margin-top: 10px;">Cámara:</label>
<select name="camera_index">
{&#x2013; for camera in available_cameras &#x2013;}
<option value="{ loop.index0 }">{ camera }</option>
{&#x2013; endfor &#x2013;}
</select>
<br>
<input type="submit" value="Ejecutar">
</form>
<!-- Mensaje de cierre -->
<div id="closing-message">
<p>Configuración guardada. Iniciando PrintGuard...</p>
</div>
</div>
</body>
</html>
</script>
function showClosingMessage() {
var closingMessage = document.getElementById("closing-message");
var container = document.querySelector(".container");

// Mostrar mensaje de cierre
closingMessage.style.display = "block";

// Remover el formulario
var form = document.querySelector("form");
form.style.display = "none";

// Crear y agregar animación de cargando
var loadingDiv = document.createElement("div");
loadingDiv.id = "loading";
loadingDiv.style.textAlign = "center";
loadingDiv.style.marginTop = "20px";
loadingDiv.style.color = "#1aff00";
loadingDiv.style.fontSize = "16px";
loadingDiv.style.display = "flex";
loadingDiv.style.flexDirection = "column";
loadingDiv.style.alignItems = "center";

var spinnerDiv = document.createElement("div");
spinnerDiv.className = "spinner";
loadingDiv.appendChild(spinnerDiv);

var loadingText = document.createElement("p");
loadingText.textContent = "Cargando...";
loadingDiv.appendChild(loadingText);

container.appendChild(loadingDiv);
}
</script>

```

Anexo 4. Librerías utilizadas en software de control

```

#LIBRERIAS UTILIZADAS

import cv2
import torch
import paho.mqtt.client as mqtt
import serial
import time
import requests
import json
import threading

from flask import Flask, render_template, request
from hubconf import custom

```

Anexo 5. Lectura de Puerto Serial y Cámaras Disponibles

```
def leer_configuracion():
    config = {}
    try:
        with open("config.txt", "r") as f:
            lines = f.readlines()
            for line in lines:
                key, value = line.strip().split("=")
                config[key] = value
    except FileNotFoundError:
        print("Archivo de configuración no encontrado.")
    return config

config = leer_configuracion()

serial_port = config.get("serial_port", "")
camera_index = config.get("camera_index", "")
if camera_index != 'None':
    camera_index = int(camera_index)

# Busca el índice del primer espacio en blanco
indice_espacio = serial_port.find(' ')

# Si se encuentra un espacio en blanco, elimina todo lo que sigue a partir de ese punto
if indice_espacio != -1:
    serial_port = serial_port[:indice_espacio]

print("Puerto Serial:", serial_port)
print("Índice de la Cámara:", camera_index)
```

Anexo 6. Configuración Serial

```
#CONFIGURACION PARA COMUNICACION SERIAL
# Configuración de la comunicación serial
port = serial_port # Puerto serial de la impresora
baudrate = 115200 # Velocidad de comunicación (ajusta según tu configuración)
ser = serial.Serial(port, baudrate, timeout=1)
# Espera a que se establezca la comunicación serial
time.sleep(2)
```

Anexo 7. Clase para uso del api de Telegram

```
class TelegramBot():
    def __init__(self):
        self.token = None
        self.channel = None
        with open(properties, 'r') as fr:
            params = json.load(fr)
            self.token = params['token']
            self.channel = params['channel']
    def get_me(self):
        url = f"https://api.telegram.org/bot{self.token}/getMe"
        ans = requests.get(url)
        if ans.status_code == 200:
            salida = json.loads(ans.text)
            return salida
        return None
    def get_updates(self):
        url = f"https://api.telegram.org/bot{self.token}/getUpdates"
        print(url)
        ans = requests.get(url)
        if ans.status_code == 200:
            salida = json.loads(ans.text)
            return salida
        return None
    def send_message_to_channel(self, message):
        try:
            return self.send_message(self.channel, message)
        except Exception as exception:
            print(exception)
        return None
    def send_message(self, chat_id, message):
        url = f"https://api.telegram.org/bot{self.token}/sendMessage"
        data = {"chat_id":chat_id, "text":message}
        response = requests.post(url, data=data)
        if response.status_code == 200:
            salida = json.loads(response.text)
            return salida
        msg = f"Error code: {response.status_code}. Description: {response.text}"
        raise Exception(msg)
    def send_photo(self, chat_id, filename, caption):
        url = f"https://api.telegram.org/bot{self.token}/sendPhoto"
        data = {"chat_id":chat_id, "caption":caption}
        files = {"photo":(filename, open(filename, 'rb'))}
        response = requests.post(url, data=data, files = files)
        if response.status_code == 200:
            salida = json.loads(response.text)
            return salida
        msg = f"Error code: {response.status_code}. Description: {response.text}"
        raise Exception(msg)
    def send_photo_to_channel(self, filename, caption):
        try:
            return self.send_photo(self.channel, filename, caption)
        except Exception as exception:
            print(exception)
        return None
```

Anexo 8. Función Completa de Envío de Comandos vía Serial

```
# Función para enviar comandos GCODE a la impresora
def send_gcode_command(command):
    while True:
        try:
            gcode_command = (command + "\n").encode('utf-8')
            ser.write(gcode_command)
            time.sleep(0.1)
            break
        except serial.SerialException:
            print("Se perdió la conexión. Reconectando...")
            time.sleep(1) # Espera un segundo antes de intentar reconectar
            puerto_serial.close() # Cierra la conexión serial
            puerto_serial = serial.Serial(port, baudrate) # Vuelve a abrir la conexión serial
            time.sleep(1)
            continue
```

Anexo 9. Funciones para comando de temperaturas

```
#Cambiar temperatura de la boquilla
def asignTemperatura(temperaturaBoquilla):
    print(f"Asignando temperatura a la boquilla: {temperaturaBoquilla} °C")
    command = f"M104 S{temperaturaBoquilla}\n"
    # Envio del comando a la impresora
    send_gcode_command(command)
    # Lectura de la respuesta de la impresora
    response = ser.readline().decode('utf-8').strip()
    print(f"Respuesta de la impresora: {response}")

#Cambiar la temperatura de la cama
def asignBed(temperaturaCama):
    print(f"Asignando temperatura a la cama: {temperaturaCama} °C")
    command = f"M140 S{temperaturaCama}"
    # Envio del comando a la impresora
    send_gcode_command(command)
    # Lectura de la respuesta de la impresora
    response = ser.readline().decode('utf-8').strip()
    print(f"Respuesta de la impresora: {response}")
```

Anexo 10. Funciones para control de ejes

```
#Funcion para ejecutar el Autohome
def autohome():
    print("Haciendo Autohome...")
    send_gcode_command('G28') # Comando GCODE para Autohome

#Funcion para ejecutar movimiento en X
def move_x_axis(position, speed):
    send_gcode_command(f'G0 X{position} F{speed}') # Comando GCODE para mover el eje X con velocidad

# Función para mover el eje Y a una posición específica con velocidad
def move_y_axis(position, speed):
    send_gcode_command(f'G0 Y{position} F{speed}') # Comando GCODE para mover el eje Y con velocidad

# Mover el motor del eje Z hacia arriba
def move_z_up(distance):
    command = f"G91 ; Cambiar a coordenadas relativas\nG1 Z{distance} F300 ; Mover el eje Z hacia arriba"
    send_gcode_command(command)

# Mover el motor del eje Z hacia abajo
def move_z_down(distance):
    command = f"G91 ; Cambiar a coordenadas relativas\nG1 Z-{{distance}} F300 ; Mover el eje Z hacia abajo"
    send_gcode_command(command)
```

Anexo 11. Funciones para detener y pausar impresión

```
def pause_print():
    pause_command = "M0"
    send_gcode_command(pause_command)

def stop_print():
    stop_command = "M112"
    send_gcode_command(stop_command)
```

Anexo 12. Funciones de Lectura de Temperatura

En esta función se aplica una técnica de barrido de texto, donde se obtiene solo el valor de la temperatura la cual luego se convierte de String a flotante. Ya que la respuesta real al enviar un comando tiene la sintaxis “ok T:200.0 /200.0 B:60.0 /60.0 @:100”, por lo que hay que separar el mensaje para obtener el valor específico de la temperatura.

```
def read_bed_temperature():
    send_gcode_command('M105') # Comando GCODE para obtener la temperatura actual
    response = ser.readline().decode('utf-8').strip()
    temperature_index = response.find('B:')
    if temperature_index != -1:
        temperature_value = response[temperature_index+2:].split(' ')[0]
        return float(temperature_value)

# Función para leer y mostrar la temperatura actual de la impresora
def read_current_temperature():
    send_gcode_command('M105') # Comando GCODE para obtener la temperatura actual
    response = ser.readline().decode('utf-8').strip()
    if response.startswith('ok T'):
        temperatureNzsl = response.split('T')[1].split(' ')[0]
        temperatureNzsl = temperatureNzsl.replace(":", "").strip()
        return float(temperatureNzsl)
```

Anexo 13. Función para Cargar modelo de Detección

```
#Funcion para cargar el modelo de deteccion
def load_yolov5_model(weights_path, device='cpu'):
    # Load model
    model = custom(path=weights_path)
    return model
```

Anexo 14. Función para cambio de Sintaxis Booleana de Javascript a Python

```
#Funcion para obtener el valor booleano STR MQTT javascript y convertirlo en booleano de python
def str_to_bool(s):
    if s.lower() == 'true':
        return True
    elif s.lower() == 'false':
        return False
    else:
        raise ValueError("El string no representa un booleano válido")
```

Anexo 15. Configuración de Protocolo MQTT en Python

Esta función trabaja en segundo plano, a la espera de la escritura a cualquiera de los tópicos a los cuales el dispositivo está suscrito.

```
#Funcion para obtener los valores mqtt en tiempo real
def on_message(client, userdata, msg):
    if msg.topic == "/3d/tempNzzl":
        global valorTemp
        valor = msg.payload.decode("utf-8") # Convertir bytes a str
        valorTemp = int(valor)
        asignTemperature(valorTemp)

    if msg.topic == "/3d/tempBed":
        global valorBed
        bed = msg.payload.decode("utf-8") # Convertir bytes a str
        valorBed = int(bed)
        asignBed(valorBed)

    if msg.topic == "/3d/selector":
        global selectorDeteccion
        rSelector = msg.payload.decode("utf-8") # Convertir bytes a str
        selectorDeteccion = str_to_bool(rSelector)
        print("Selector configurado como: " + rSelector)
        print(selectorDeteccion)

    if msg.topic == "/3d/autohome":
        state = msg.payload.decode("utf-8") # Convertir bytes a str
        stateAH = str_to_bool(state)
        print
        if stateAH == True:
            autohome()

    if msg.topic == "/3d/ejex":
        global x
        x = msg.payload.decode("utf-8") # Convertir bytes a str
        x = int(x)
        print(f"Moviendo eje X a la posicion {x}mm")
        move_x_axis(x, 6000)

    if msg.topic == "/3d/ejey":
        global y
        y = msg.payload.decode("utf-8") # Convertir bytes a str
        y = int(y)
        print(f"Moviendo eje Y a la posicion {y}mm")
        move_y_axis(y, 6000)

    if msg.topic == "/3d/ejez":
        global z
        z = msg.payload.decode("utf-8") # Convertir bytes a str
        z = int(z)
        print(f"Moviendo eje Z: {z}mm")
        move_z_up(z)

    if msg.topic == "/3d/photo":
        global photo
        photo = msg.payload.decode("utf-8") # Convertir bytes a str
        photo = str_to_bool(photo)
        print(photo)

    if msg.topic == "/3d/avisosTelegram":
        global avisos
        avisos = msg.payload.decode("utf-8") # Convertir bytes a str
        avisos = str_to_bool(avisos)
        if avisos == True:
            print("Avisos Activos")
        else:
            print("Avisos Inactivos")
        print(avisos)

    if msg.topic == "/3d/stringing":
        global stringing
        stringing = msg.payload.decode("utf-8") # Convertir bytes a str
        stringing = str_to_bool(stringing)
        print(stringing)

    if msg.topic == "/3d/warping":
        global warping
        warping = msg.payload.decode("utf-8") # Convertir bytes a str
        warping = str_to_bool(warping)
        print(warping)

    if msg.topic == "/3d/spaghetti":
        global spaghetti
        spaghetti = msg.payload.decode("utf-8") # Convertir bytes a str
        spaghetti = str_to_bool(spaghetti)
        print(spaghetti)

    if msg.topic == "/3d/interaction":
        global interaction
        interaction = msg.payload.decode("utf-8") # Convertir bytes a str
        interaction = str_to_bool(interaction)
        print(interaction)

    if msg.topic == "/3d/detachment":
        global detachment
        detachment = msg.payload.decode("utf-8") # Convertir bytes a str
        detachment = str_to_bool(detachment)
        print(detachment)

    if msg.topic == "/3d/layerShifting":
        global layerShifting
        layerShifting = msg.payload.decode("utf-8") # Convertir bytes a str
        layerShifting = str_to_bool(layerShifting)
        print(layerShifting)

    if msg.topic == "/3d/pausePrint":
        print("Pausando impresión...")
        pause_print()

    if msg.topic == "/3d/stopPrint":
        print("Deteniendo impresión...")
        stop_print()

    client = mqtt.Client("sistemaVision")
    client.on_message = on_message
    client.connect("34.132.7.144", 1883)
    client.subscribe("/3d/tempNzzl")
    client.subscribe("/3d/tempBed")
    client.subscribe("/3d/selector")
    client.subscribe("/3d/autohome")
    client.subscribe("/3d/pausePrint")
    client.subscribe("/3d/ejex")
    client.subscribe("/3d/ejey")
    client.subscribe("/3d/ejez")
    client.subscribe("/3d/photo")
    client.subscribe("/3d/avisosTelegram")
    client.subscribe("/3d/stopPrint")

    #SUSCRIPCION A TOPICOS DE DETECCION
    client.subscribe("/3d/stringing")
    client.subscribe("/3d/warping")
    client.subscribe("/3d/interaction")
    client.subscribe("/3d/spaghetti")
    client.subscribe("/3d/detachment")
    client.subscribe("/3d/layerShifting")

    client.loop_start()
    label = ""
    topicDetect = "/3d/deteccion"
    topicBed = "/3d/readBed"
    topicNzzl = "/3d/readNzzl"
```


Anexo 16. Inicialización de Variables

```
# Especifica la ruta a los pesos del modelo preentrenado
weights_path = r"/home/printguard/Desktop/printGuard/runs/train/exp2/weights/best.pt"

# Carga el modelo YOLOv5
device = 'cuda' if torch.cuda.is_available() else 'cpu'
model = load_yolov5_model(weights_path, device)
selectorDeteccion = False
telegram_bot = TelegramBot()
photo = False
cap = ""
avisos = False
layerShifting = False
warping = False
stringing = False
spaghetti = False
interaction = False
detachment = False
message = 30
```

Anexo 17. Bucle Principal – Captura de foto en tiempo real

```
if __name__ == "__main__":
    while True:
        if photo == True and camera_index != 'None':
            client.publish(topicDetect, "Prueba")
            cap = cv2.VideoCapture(camera_index)
            print("Entro")
            ret, frame = cap.read()
            if not ret:
                print("not ret")
                continue
            cv2.imwrite("captura_camara.jpg", frame)
            time.sleep(1)
            print(telegram_bot.send_photo_to_channel("captura_camara.jpg", "Este es el estado actual de la impresora"))
            photo = False
            cap.release()
            cap = ""
        elif photo == True and camera_index == 'None':
            client.publish(topicDetect, "NoCameraNoPhoto")
            photo = False
```

Anexo 18. Envío de Lectura de Temperaturas en Tiempo Real

```
readBed = read_bed_temperature()
readNzzl = read_current_temperature()
if readBed != "":
    client.publish(topicBed, readBed)
    time.sleep(0.5)

if readNzzl != "":
    client.publish(topicNzzl, readNzzl)
    time.sleep(0.5)
```

Anexo 19. Código de detección de Errores

```
#SELECTOR PARA ACTIVAR O DESACTIVAR EL MODULO DE DETECCION DE FALLOS DESDE LA INTERFAZ GRAFICA
if selectorDeteccion == True and camera_index != 'None':
    cap = cv2.VideoCapture(camera_index)
    # Abre la cámara
    ret, frame = cap.read()
    frame = frame.copy()
    if not ret:
        print("not ret")
        continue
    # Realiza la detección en el frame actual
    results = model(frame)

    # Obtiene las detecciones
    detections = results.pred[0]
    for *xyxy, conf, cls in detections:
        label = f'{model.names[int(cls)]} {conf:.2f}'
        xyxy = [int(i) for i in xyxy]
        frame = cv2.rectangle(frame, (xyxy[0], xyxy[1]), (xyxy[2], xyxy[3]), (255, 0, 0), 2)
        frame = cv2.putText(frame, label, (xyxy[0], xyxy[1] - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 0, 0), 2)
```

Anexo 20. Envío de Avisos de Interfaz Gráfica

```
print(label)
if label.startswith("layerShifting"):
    message = 1
    client.publish(topicDetect, message)
    label = ""
    time.sleep(1)

if label.startswith("stringing"):
    message = 2
    client.publish(topicDetect, message)
    label = ""
    time.sleep(1)

if label.startswith("spaghetti"):
    message = 3
    client.publish(topicDetect, message)
    label = ""
    time.sleep(1)

if label.startswith("warping"):
    message = 4
    client.publish(topicDetect, message)
    label = ""
    time.sleep(1)

if label.startswith("detachment"):
    message = 5
    client.publish(topicDetect, message)
    label = ""
    time.sleep(1)

if label.startswith("interaction"):
    message = 6
    client.publish(topicDetect, message)
    label = ""
    time.sleep(1)
```

Anexo 21. Envío de Avisos vía Telegram

```
if aviso == True:
    if message == 1 and layerShifting == True:
        print("Entro")
        frame = frame.copy()
        cv2.imwrite("deteccion.jpg", frame)
        time.sleep(1)
        print(telegram_bot.send_photo_to_channel("deteccion.jpg", "Se han detectado capas desfasadas. Revise la calibración de los ejes.~"))

    if message == 2 and stringing == True:
        frame = frame.copy()
        cv2.imwrite("deteccion.jpg", frame)
        time.sleep(1)
        print(telegram_bot.send_photo_to_channel("deteccion.jpg", "Se han detectado hilos en la impresión. Revise la configuración de retracciones y temperatura de impresión.~"))

    if message == 3 and spaghetti == True:
        frame = frame.copy()
        cv2.imwrite("deteccion.jpg", frame)
        time.sleep(1)
        print(telegram_bot.send_photo_to_channel("deteccion.jpg", "~ALERTA! La impresión puede estar dañada, se ha detectado Spaghetti. Ingrese a la interfaz para detener la impresión.~"))

    if message == 4 and warping == True:
        frame = frame.copy()
        cv2.imwrite("deteccion.jpg", frame)
        time.sleep(1)
        print(telegram_bot.send_photo_to_channel("deteccion.jpg", "~ALERTA! La impresión puede estar dañada, se han detectado zonas con warping. Verificar temperatura de cama de impresión y ajustes de capa inicial!~"))

    if message == 5 and detachment == True:
        frame = frame.copy()
        cv2.imwrite("deteccion.jpg", frame)
        time.sleep(1)
        print(telegram_bot.send_photo_to_channel("deteccion.jpg", "~ALERTA! Tu pieza puede que esté despegada. Ingrese a la interfaz si necesita detener la impresión.~"))

    if message == 6 and interaction == True:
        frame = frame.copy()
        cv2.imwrite("deteccion.jpg", frame)
        time.sleep(1)
        print(telegram_bot.send_photo_to_channel("deteccion.jpg", "Estas manipulando la máquina? Interacción humana detectada.~"))
```

Anexo 22. Validaciones Adicionales para Cámara no Conectada

```
elif (selectorDeteccion == True and camera_index == 'None'):
    client.publish(topicDetect, "NoCameraNoDetection")
    selectorDeteccion = False
```

Anexo 23. Dispositivo Conectado



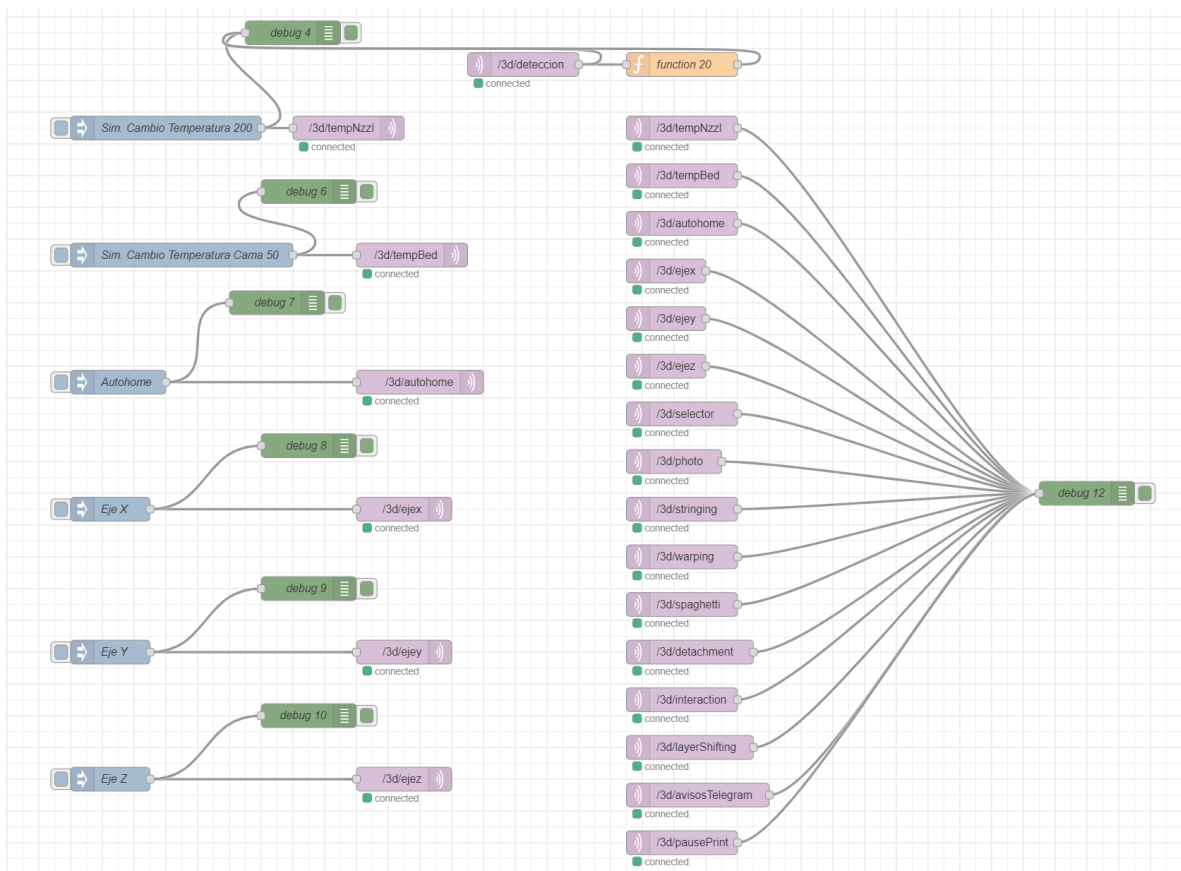
Anexo 24.Verificación de Conexión MQTT del Dispositivo

Clients

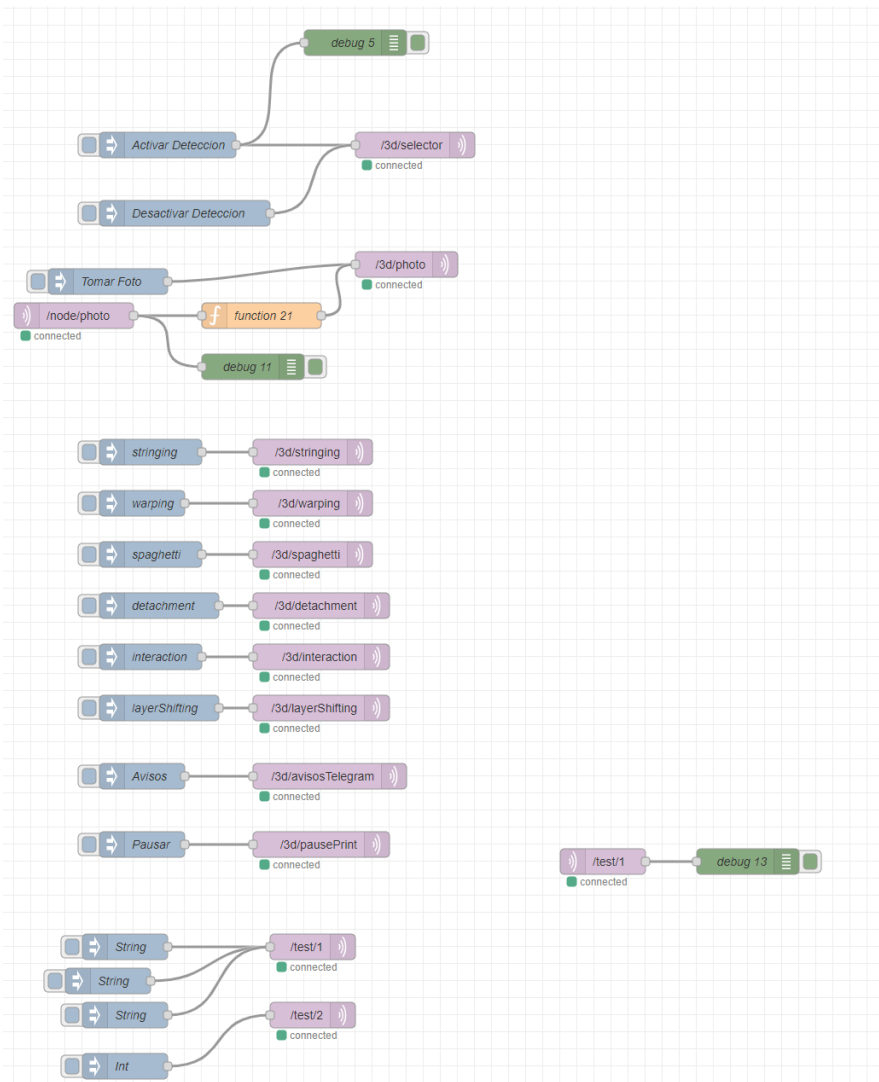
Client ID Username Node Search Reset Kick Out Refresh

Client ID	Username	Status	IP Address	Keepalive	Clean Start	Session Expiry Interval	Connected At
NodeRed		Connected	34.136.122.105:37392	60	true	0	2023-10-01 23:02:49
sistemaVision		Connected	190.84.117.96:13198	60	true	0	2023-10-02 10:23:41

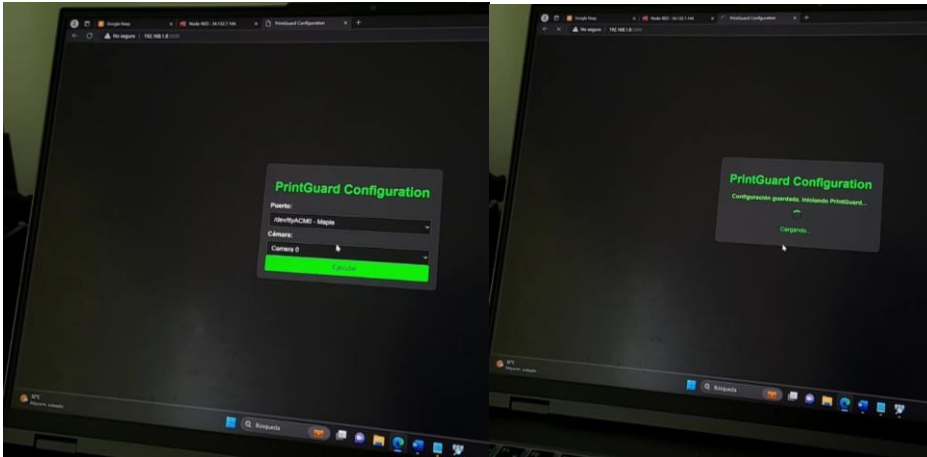
Anexo 25.Interfaz de Pruebas 1 con Node-Red



Anexo 26. Interfaz de Pruebas 2 con Node-Red



Anexo 27. Pruebas de Interfaz de Configuración



Anexo 28. Validación del Sistema de Control con Node-Red

