UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE GUAYAQUIL

CARRERA DE INGENIERÍA ELECTRÓNICA

"DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE CONTROL AUTOMÁTICO CON VISIÓN ARTIFICIAL Y REDES NEURONALES DESTINADO AL CONTROL DE CALIDAD DE ALIMENTOS"

Trabajo de titulación previo a la obtención del

Título de INGENIERO ELECTRÓNICO

AUTORES: WILLIAM ALEXI VIVAR ENCALADA

MIRIAM FERNANDA VIVAS MATICURENA

TUTOR: LENIN ESTUARDO CEVALLOS ROBALINO, PhD.

GUAYAQUIL – ECUADOR

2022

CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN

Nosotros, **William Alexi Vivar Encalada** con documento de identificación **N° 0941495830** y **Miriam Fernanda Vivas Maticurena** con documento de identificación **N° 0930071915**; manifestamos que:

Somos los autores y responsables del presente trabajo; y, autorizamos a que sin fines de lucro la Universidad Politécnica Salesiana pueda usar, difundir, reproducir o publicar de manera total o parcial el presente trabajo de titulación.

Guayaquil, 17 de mayo del año 2022.

Atentamente,

William Alexi Vivar Encalada
0941495830

Miriam Fernanda Vivas Maticurena
0930071915

CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE TITULACIÓN A LA UNIVERSIDAD POLITÉCNICA SALESIANA

Nosotros, **William Alexi Vivar Encalada** con documento de identificación **N° 0941495830** y **Miriam Fernanda Vivas Maticurena** con documento de identificación **N° 0930071915**, expresamos nuestra voluntad y por medio del presente documento cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del Proyecto técnico: "Diseño e implementación de un sistema de control automático con visión artificial y redes neuronales destinado al control de calidad de alimentos" el cual ha sido desarrollado para optar por el título de: INGENIERO ELECTRÓNICO, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En concordancia con lo manifestado, suscribimos este documento en el momento que hacemos la entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Guayaquil, 17 de mayo del año 2022.

Atentamente,

William Alexi Vivar Encalada
0941495830

Miriam Fernanda Vivas Maticurena
0930071915

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN

Yo, **Lenin Estuardo Cevallos Robalino, PhD**. Con documento de identificación N° 0603274226; Docente de la Universidad Politécnica Salesiana, declaro que bajo mi autoría fue desarrollado el trabajo de titulación DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE CONTROL AUTOMÁTICO CON VISIÓN ARTIFICIAL Y REDES NEURONALES DESTINADO AL CONTROL DE CALIDAD DE ALIMENTOS. Realizado por **William Alexi Vivar Encalada** con documento de identificación **N° 0941495830** y por **Miriam Fernanda Vivas Maticurena** con documento de identificación **N° 0930071915**, obteniendo como resultado final el trabajo de titulación bajo la opción Proyecto Técnico que cumple con todos los requisitos determinados por la Universidad Politécnica Salesiana.

Guayaquil, 17 de mayo del año 2022.

Atentamente,

Ing. Lenin Estuardo Cevallos Robalino, PhD.
0603274226

DEDICATORIA

Esta tesis está dedicada con todo mi corazón a mis padres que me han formado como la persona que soy en la actualidad, por su trabajo, amor y sacrificio en todos estos años, gracias a ustedes he logrado llegar hasta aquí y convertirme en lo que soy, son los mejores padres.

A mi compañera de tesis Miriam Vivas Maticurena que siempre estuvo ahí apoyándome desde el primer día de la carrera, es un gran privilegio haber recorrido estos caminos juntos y lograr esta meta.

A mis hermanos que me apoyaron y estuvieron siempre presente en el transcurso de este largo camino.

A los docentes de la Universidad Politécnica Salesiana por impartir su conocimiento en el transcurso de mi formación académica, por brindarme su consejo y ayuda.

A mis amigos y futuros colegas que me brindaron su total y completo apoyo de manera desinteresada durante toda la carrera.

William Alexi Vivar Encalada

Este proyecto técnico de titulación se lo dedico en primer lugar a Dios, por ayudarme a tener constancia, responsabilidad y sobre todo por darme la oportunidad de cuidar siempre la salud de mi familia y la mía.

A mi padre John Vivas López, eternamente agradecida porque siempre confió en mí, me ayudó a tener constancia, a enseñarme que siempre hay salida en todos los problemas, aprender a creer en mí. Agradecer siempre por nunca dejarme sola, por su constante amor que me llegó a realizar como persona.

A mi madre Miriam Maticurena Salazar, quien siempre me ha brindado su amor y su apoyo incondicionalmente, gracias siempre por estar a mi lado y darme aliento de seguir luchando, por no dejarme decaer cuando quería renunciar a muchas cosas que ahora se ve el fruto. Gracias mamá por ser el motor de mi vida, por ser mi ejemplo de superación y por enseñarme a nunca bajar la cabeza y afrontar todo.

A mis amigos/as agradecer por siempre estar presente, a mi compañero de tesis William Vivar Encalada por estar conmigo en este proyecto y podernos graduar juntos, como siempre se quiso. A mis amigos(as) en especial a Michelle por estar incondicionalmente conmigo, por su apoyo, por tomar su tiempo y ayudarme, A mis primas por nunca dejarme sola y darme consejos que son valiosos para mí. Gracias infinitas a los demás porque fueron y serán mi segunda familia.

Miriam Fernanda Vivas Maticurena

AGRADECIMIENTO

Agradezco en primer lugar a Dios, por guiarme en mí camino y por permitirme concluir mis estudios.

A mis padres que son mi ejemplo a seguir, que a través de su paciencia y buenos valores me ayudan a trazar mi camino.

A mi novia que estuvo ahí alentando en el transcurso de la elaboración de mi tesis, por sus palabras de ánimo y por comprenderme en los días difíciles.

A la comunidad que conforman la Universidad Politécnica Salesiana que con dedicación y entrega guían a los futuros profesionales de la patria.

William Alexi Vivar Encalada

Agradezco eternamente a Dios, por ser mi guía en los días que fueron más duros, por mantenerme con salud, felicidad y amor.

A mi familia por el apoyo incondicional que siempre me han brindado, que hicieron tanto sacrificio para que yo llegue a este logro tan importante para mí, ya que me abre muchas puertas tanto nacionales como internacionales.

A mis profesores de la Universidad Politécnica Salesiana, agradecer infinitamente por todos los conocimientos brindados, por enseñarnos a conocer un nuevo campo de trabajo, con experiencias y anécdotas inolvidables.

Miriam Fernanda Vivas Maticurena

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

Año	Alumno	Tutor de Proyecto	Proyecto de Titulación
2022	▪ MIRIAM FERNANDA VIVAS MATICURENA ▪ WILLIAM ALEXI VIVAR ENCALADA	LENIN E. CEVALLOS ROBALINO, PhD.	"DISEÑO E IMPLEMENTACION DE UN SISTEMA DE CONTROL AUTOMÁTICO CON VISIÓN ARTIFICIAL Y REDES NEURONALES DESTINADO AL CONTROL DE CALIDAD DE ALIMENTOS"

El presente trabajo de titulación tiene como objetivo diseñar e implementar un sistema de control automático con visión artificial y redes neuronales destinado al control de calidad de alimentos tales como la pera y la manzana.

La contaminación superficial en frutas afecta hoy en día a las industrias y a los consumidores, por ello, el control de calidad es un proceso imprescindible para asegurar la calidad óptima de los alimentos.

El enfoque del proyecto técnico se basa principalmente en clasificar y seleccionar frutas con un sistema de inteligencia artificial el cual cuenta de una red neuronal previamente entrenada y que es ejecutada en una Raspberry. La red neuronal a través de sus componentes electrónicos cuenta con visión en tiempo real de los objetos y los clasifica dependiendo de sus características, cumpliendo así su función y brindando productos de mejor calidad. El modelo de red que se ha propuesto podría usarse en diferentes ámbitos en el que se requiera inspección visual de objetos tales como: medio ambiente, agricultura, Biología, Industria, etc.

La parte mecánica del proyecto está controlada por una pareja de Arduinos que se encargan de mover las frutas por medio del motor acoplados a la banda transportadora, mediante la comunicación con la red neuronal se ejecuta la clasificación por medio de servomotores acoplados a pistones, el sistema también cuenta con sensores de proximidad que detectan las posiciones de los objetos en la banda transportadora. El prototipo cuenta con un display en el cual se supervisa el proceso en directo ayudando a minimizar errores en el caso de haberlos.

Se evalúa un lote de frutas en el cual se encuentran los dos tipos a clasificar, los resultados obtenidos son los más óptimos y se logra un gran porcentaje de acierto al momento de detectar cada fruta, al igual que aquellas que se encuentran en mal estado, identificando los posibles tipos de contaminantes que puedan afectar a los alimentos.

PALABRAS CLAVES: visión artificial**,** redes neuronales, Python**,** frutas.

ABSTRACT

Year	Students	Degree Project Tutor	Technical Degree Project
2022	▪ MIRIAM FERNANDA VIVAS MATICURENA ▪ WILLIAM ALEXI VIVAR ENCALADA	LENIN E. CEVALLOS ROBALINO, PhD.	"DESIGN AND IMPLEMENTATION OF AN AUTOMATIC CONTROL SYSTEM WITH ARTIFICIAL VISION AND NEURAL NETWORKS FOR FOOD QUALITY CONTROL"

The objective of this degree work is to design and implement an automatic control system with artificial vision and neural networks for quality control of foods such as pears and apples. Surface contamination in fruits today affects industries and consumers, therefore, quality control is an essential process to ensure optimal food quality.

The focus of the technical project is mainly based on classifying and selecting fruits with an artificial intelligence system which has a previously trained neural network and is executed on a Raspberry. Through its electronic components, the neural network has real-time vision of objects and classifies them depending on their characteristics, thus fulfilling its function and providing better quality products.

The network model that has been proposed could be used in different environments where visual inspection of objects is required such as: environment, agriculture, biology, industry, etc. The mechanical part of the project is controlled by a couple of Arduinos that are in charge of moving the fruits by means of the motor coupled to the conveyor belt, through communication with the neural network, the classification is executed by means of servomotors coupled to pistons, the system It has proximity sensors that detect the positions of objects also on the conveyor belt. The prototype has a display in which the process is supervised live, helping to minimize errors if there are any.

A batch of fruits is evaluated in which the two types to be classified are found, the results obtained are the most optimal and a high percentage of success is achieved when detecting each fruit, as well as those that are in poor condition, identifying the possible types of contaminants that can affect food.

KEYWORDS: artificial vision**,** neural networks, python**,** fruits.

INTRODUCCIÓN

Con el pasar del tiempo la tecnología se ha convertido en una gran aliada en diferentes ámbitos de la vida humana, a pesar del gran avance tecnológico aún existen trabajos que son realizados por el ser humano ya que se requiere de su capacidad para supervisar diversos procesos que las maquinas en su gran mayoría no pueden hacerlo.

En el presente proyecto técnico, se toma a consideración problemas que en el ámbito de la ingeniería electrónica se han podido generar diversas soluciones, uno de ellos es la inteligencia artificial, que mediante redes neuronales se pueden realizar desde procesos muy simples hasta los más complejos. Uno de ellos es la clasificación de objetos que se puede utilizar en diferentes ámbitos como lo es la industria alimentaria, que entre sus necesidades existe la de seleccionar y clasificar por diferentes parámetros los alimentos.

La contaminación en las frutas es un problema que afecta en gran cantidad a los consumidores, por lo tanto, es importante diseñar un sistema de control que tenga como objetivo proteger la calidad de los alimentos. Labor que es realizada por el personal de las empresas lo que conlleva a diferentes limitaciones tales como el traslado del personal al lugar de trabajo, los costos que generan, los tiempos definidos de la jornada laboral, todo esto se puede sustituir con un sistema de control por redes neuronales, mejorando así la producción de la empresa y los tiempos de producción.

El proyecto técnico se encuentra dividido en 4 capítulos, los cuales detallan el procedimiento para llevar a cabo la implementación del sistema de control con visión artificial y redes neuronales.

El primer capítulo describe el problema, explicando los antecedentes del mismo, su importancia y alcance, además de mencionar a los beneficiarios y el objetivo general del proyecto.

En el segundo capítulo se detalla el marco metodológico, explicando los temas principales, tales como sistema de control, redes neuronales, visión artificial, entre otros con sus respectivas definiciones.

El tercer capítulo detalla acerca del procedimiento para el diseño e implementación del prototipo, mencionando acerca de la estructura física, el sistema electrónico y la programación.

En el cuarto capítulo, se obtienen los resultados a partir de las pruebas realizadas programando la red neuronal junto con la visión artificial.

1. EL PROBLEMA

1.1 Descripción del problema

Actualmente, los alimentos contaminados por diferentes agentes externos como gérmenes o productos químicos, han originado la muerte de miles de personas y que enfermen millones. Estas dolencias saturan los servicios sanitarios y cuestan alrededor de 95.000 millones de dólares a las economías de ingresos bajos y medios (Noticias ONU, 2019).

Una de las prioridades para realizar este proyecto técnico es analizar y mejorar los estándares de calidad de la industria alimenticia, los cuales en diversas ocasiones se ven afectados por diferentes tipos de problemas como el proceso natural de descomposición de los alimentos, el difícil traslado de los sembríos hasta las plantas de procesamiento y su almacenamiento en gran escala. Además, otro de los factores a considerar son las plagas, las cuales afectan de una manera considerable a la calidad de los alimentos.

Estos problemas han abierto un nuevo campo para diseñar un prototipo de estación de control de calidad usando la visión artificial como su mayor herramienta, la cual permitirá realizar una inspección automática de los alimentos, reduciendo así el tiempo de verificación y garantizando un mayor nivel de calidad en sus productos, esto a su vez ayudaría a las empresas a poder comercializar sus alimentos no sólo dentro del país, abriendo brechas de exportación de alimentos tales como las frutas, ayudando a la economía del país.

1.2 Antecedentes

La contaminación de los alimentos es un grave problema de salud pública en todo el mundo que provoca enfermedades, afectando a los consumidores y teniendo como consecuencia la afectación en el área económica.

Hay varias razones que pueden provocar la contaminación de los alimentos. Sin embargo, en la mayoría de ocasiones se clasifica en tres categorías: biológica, física o química. Algunos microorganismos son capaces de formar estructuras altamente resistentes, los cuales pueden sobrevivir a los procesos normales de almacenamiento y preparación, esto debido a sus características físico químicas.

Los productos alimenticios pueden contaminarse en cualquier punto, así como durante la producción, procesamiento, envío y distribución, es por ello que las autoridades sanitarias, agricultores, consumidores, entre otros deben conocer acerca de la higiene de los alimentos e implementar un sistema de control que garantice la inocuidad alimentaria (Plata, 2003).

1.3 Importancia y Alcance

El sistema de control del proyecto técnico es de suma importancia para el sector alimenticio debido a que ayuda en la detección de las diferentes clases de contaminantes que existen en ciertos tipos de frutas.

1.4 Delimitación del problema

1.4.1 Delimitación Temporal

La puesta en práctica del proyecto técnico se realizó en el periodo 2021-2022.

1.4.2 Delimitación Espacial

El prototipo del proyecto va a ser aprovechado en la Universidad Politécnica Salesiana.

Figura 1. Delimitación espacial del prototipo del Proyecto Técnico (GoogleMaps, 2021)

1.4.3 Delimitación Académica

El proyecto dispone de un alcance académico que se sustenta mediante su aplicación en los diferentes procesos de controles de calidad.

Innovación: La aplicación del prototipo proporciona a la industria una alternativa para detectar objetos a través de una banda transportadora la cual gracias a su velocidad de respuesta permite que no sea necesaria la presencia de un trabajador que verifique el estado de los objetos, y su uso en el área de la robótica servirá para diversas funciones como motivo de módulo de práctica.

Impacto: En el ámbito industrial, será de gran relevancia debido a su velocidad de respuesta en la identificación de objetos en una línea de ensamble.

1.5 Beneficiarios

El grupo de beneficiarios específico se centra en los consumidores de las frutas seleccionadas para este proyecto, las cuales son manzanas y peras. Estas frutas han sido seleccionadas por su alto consumo a nivel mundial, con este proyecto se busca que los consumidores de estas frutas tengan alimentos sanos, mejorando así su calidad de vida, ya que una población mejor alimentada será menos propensa a contraer

enfermedades trasmitidas por los alimentos, aliviando la demanda de pacientes en los hospitales y centros de salud.

Una de las prioridades en este proyecto es alcanzar los más altos estándares de calidad, haciendo que la industria alimenticia del país pueda competir a nivel mundial con sus productos.

1.6 Propuesta de solución

La propuesta de solución del proyecto se basa principalmente en un conjunto de investigaciones previas para poder estudiar y comparar los diferentes factores que afectan a las frutas. Esto con la finalidad de proponer un sistema completo e innovador que mejore notablemente la calidad de las frutas.

Para un sistema de visión artificial una de las partes más importantes es la captura de imágenes, por lo cual se selecciona una cámara de alta resolución la cual se encarga mediante el software de analizar y clasificar los diferentes tipos de contaminantes de las frutas, también consta de sensores infrarrojos para identificar la ubicación de la fruta. En el transporte de las frutas consta con una banda trasportadora, para su diseño se toma en cuenta la masa y diámetro promedio de las frutas que son trasportadas.

El hardware controlador de este proyecto técnico está implementado por una Raspberry Pi 4 B+ principal, el cual es alimentado por una fuente de energía.

1.7 Objetivos

1.7.1 Objetivo General

Diseñar e implementar un sistema de control automático con visión artificial y redes neuronales destinado al control de calidad de alimentos.

1.7.2 Objetivos Específicos

- Determinar la factibilidad técnica, legal y financiera para la creación de un prototipo de estación de control de calidad. Esto con el fin de estudiar todos los posibles escenarios para un buen funcionamiento del proyecto.

- Desarrollar una propuesta de implementación de calidad donde la tecnología e infraestructura permita en el futuro tener un desarrollo a gran escala.

- Realizar una investigación de los diferentes factores que afectan a la calidad de los alimentos, para poder diferenciar sus niveles de contaminación.

- Crear una base de datos con las imágenes adquiridas por la cámara web de los posibles tipos de contaminantes que pueden afectar a los alimentos, depurar las imágenes tomando en cuenta los diferentes entornos en los que funcionara el sistema, para tener un óptimo desempeño.

- Creación de una red neuronal la cual servirá para comparar imágenes adquiridas y definir sus clases.

2. FUNDAMENTACIÓN TEÓRICA

2.1 Control de calidad

Según Nirian (2020), el control de calidad es el conjunto de medidas y procedimientos que se deben seguir para garantizar que la calidad de un producto se mantenga y mejore frente a un conjunto de puntos de referencia y que los errores encontrados se eliminen o reduzcan.

El enfoque del control de calidad es garantizar que el producto y la fabricación del producto no solo sean consistentes, sino que también estén en línea con los requisitos del cliente.

2.1.1 Control de calidad para las frutas

El control de calidad involucra una evaluación sensorial teniendo en cuenta características como la textura la cual ayuda a determinar el grado de madurez de una fruta, el color puede ser una de las características más importantes, ya que permite que el consumidor tenga la sensación de que un producto está en buenas condiciones. Además, también se realiza una evaluación técnica que analiza la fruta con instrumentos y equipos de laboratorio (Servicio de acreditación ecuatoriano, 2018).

2.2 Contaminación en frutas

La contaminación por microorganismos patógenos en frutas, las cuales se contaminan a través del suelo, agua, aire, animales, etc. pueden causar enfermedades durante el crecimiento y también el deterioro postcosecha.

Para lograr la calidad requerida por los consumidores es necesario que esté representada por el sector de la producción primaria, el de la transformación, el de la distribución y finalmente, del consumo (R. Díaz-Sobac, J. Vernon-Carter, 2009).

2.3 Causas del deterioro de las frutas

Durante el procesamiento y almacenamiento de las frutas pueden darse alteraciones que normalmente son provocadas por bacterias y/u hongos, las cuales causan un deterioro en su calidad, afectando el color, la textura, el sabor, el olor y el valor nutritivo. A continuación, en la tabla 2.1 se detallan las causas del deterioro.

Tabla 1
Causas del deterioro de las frutas

Durante la producción	Fisiológicas Biológicas Microorganismos Senescencia de frutas Bioquímicas Autooxidación
Durante la recolección y/o distribución	Infraestructura insuficiente o inadecuada para el transporte.

Durante el tratamiento de la fruta	Mecánicas Embalaje Almacenamiento

(Cinatur Group, 2016)

2.4 Frutas utilizadas para la aplicación del prototipo

2.4.1 Manzana

La manzana conocida como Malus domésticas, es un fruto comestible que aporta diversas vitaminas para quienes las consumen, siendo uno de los árboles frutales más extendidos a nivel mundial. A continuación, en la tabla 2.2 se detallan las principales características de la manzana (La Vanguardia, 2021).

Tabla 2

Características de la manzana

Definición del producto	Fruto del árbol Malus domestica, Malus sylvestris Miller o Pyrus malus L.
Fruto	Varían mucho en tamaño, forma, color y acidez, pero la mayoría son bastante redondas y tienen un tono rojo o amarillo, contienen pequeñas semillas que se encuentran en el centro del fruto.
Clima	Requieren de climas frescos y con abundante luz solar debido a que afecta su color.
Diámetro	Diámetro desde 55 a 90 mm.
Madurez	Aplicando una ligera presión y tomándolas por el centro se puede comprobar su madurez. La pulpa siempre debe ser firme y aromática.
Vida útil	1 o 2 semanas hasta 6 meses o más.

(SOSER S.A., 2010)

2.4.1.1 Royal Gala

Es una fruta crujiente, firme, de color rojo brillante o rojo/naranja con un fondo amarillo, con sabor dulce.

Figura 2. Manzana Royal Gala (Jardineria On, 2018)

2.4.2 Pera

La pera es la fruta del peral, árbol que pertenece a la familia de las rosáceas, es una fruta jugosa y una de las más importantes de las regiones templadas muy apreciada por sus propiedades nutritivas. Existen numerosas variedades cultivadas, que varían tanto en forma como en tamaño y colores (Guía Metabólica, 2019).

A continuación, en la tabla 2.3 se detallan las principales características de la pera.

Tabla 3

Características de la pera

Nombre científico	Pyrus communis
Fruto	Tiene una forma oval, cónica, redonda o globosa (dependiendo de la variedad).
Clima	El peral requiere de entre 600 y 1000 horas de frío y se adapta bien a lugares secos y cálidos. Florece a una temperatura de 7º C y es resistente a temperaturas comprendidas desde -18º C a -20º C y hasta -40º C en pleno reposo invernal.
Diámetro	Los frutos son pomos que no superan los 4 cm (especies silvestres) y hasta 18 cm (variedades cultivadas).
Madurez	Maduran de adentro hacia afuera, por lo que para comprobar si se encuentra madura hay que aplicar una presión suave cerca del extremo del tallo.
Vida útil	La vida útil de las peras a temperatura ambiente es de 4-6 días aproximadamente, en refrigeración duran 8 días aproximadamente.

(Zambrano, 2017)

2.4.2.1 Uvilla

Puede tener toques de color amarillo o verde, su pulpa es blanca, dulce, crujiente y firme al tacto.

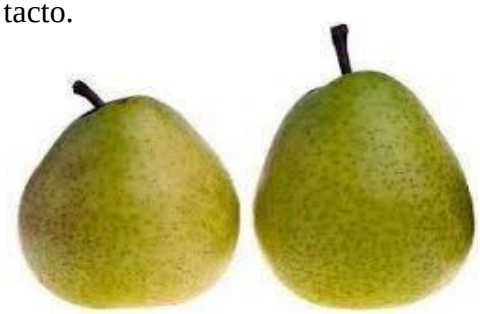

Figura 3. Pera Uvilla (Comax, 2018)

2.5 Sistema de control

Según Medina (2020), el sistema de control es un grupo de dispositivos que tienen como finalidad establecer, dirigir y administrar el funcionamiento de otros

sistemas, que tienen como objetivo disminuir fallos en procesos para lograr un resultado deseado.

2.5.1 Sistema de control lazo abierto

Consiste en procesos cronometrados, debido a que son expuestos a una perturbación externa, por ejemplo, semáforos, microondas, lavadoras, etc (Ingeniería Mecafenix, 2019).

2.5.2 Sistema de control lazo cerrado

Su función permite tener una retroalimentación la cual debe enviar una acción de control que compare la diferencia entre la entrada o salida del sistema (GSL Industrias, 2021).

2.6 Visión artificial

Es una tecnología industrial que consiste en sistemas automatizados e inteligentes permitiendo analizar el procesamiento de imágenes de una producción industrial (Infaimon, 2019).

2.6.1 Imágenes con visión artificial

Acorde a Centeno (2019) es una matriz de píxeles los cuales son la unidad más pequeña de información que compone una imagen, generalmente se organizan en una cuadrícula bidimensional.

La intensidad de cada píxel es variable; en los sistemas de color, cada píxel suele tener tres componentes, como rojo, verde y azul, mejor conocido por sus siglas en inglés como RGB. (Párr.7)

2.6.1.1 RGB

Siglas que están relacionadas con la representación de los colores red, green, y blue. Es un modelo cromático que se basa en la síntesis adictiva de la iluminación, la cual se utiliza en pantallas digitales como televisores, computadoras, etc (Castillo, 2019).

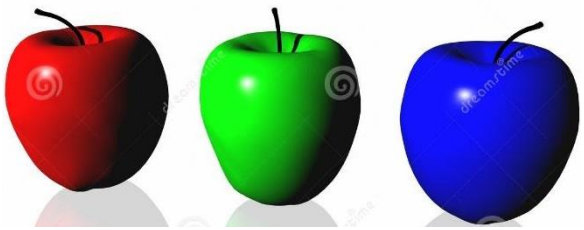

Figura 4. Imagen en contexto RGB (Dreamstime, 2016)

2.6.1.2 Escala de grises

Compuesta por 3 colores: blanco, negro y gris. Es un sistema que clasifica diferentes grados de luminosidad.

Figura 5. Imagen en contexto escalas de grises (Blog, 2014)

2.6.2 Cámaras inteligentes y sistemas de visión integradora

Las cámaras inteligentes se definen como una tecnología que se encuentran a un nivel superior en relación a sensores ópticos. Se destacan por su potencia de cálculo, la cual facilita que sus aplicaciones sean variadas en una cadena de producción. Su capacidad de procesamiento es innovadora, debido a que su almacenamiento y disponibilidad para la conexión con otros sistemas automatizados se dé por los mecanismos de entrada y salida.

En los sistemas de visión integradora se pueden conectar varios cabezales de visión remotos, reduciendo el coste en aplicaciones de visión donde se requieran varias tomas de la misma pieza.

La aplicación de los sistemas de visión artificial se puede encontrar en áreas como la automoción, que están relacionados con el control y detección de anomalías en el ensamblaje de vehículos. En el sector alimentario, es fundamental para el control de calidad de los alimentos, que permitan determinar el estado de los mismos; por último, en el área eléctrica se utiliza para la correcta soldadura de embalaje de piezas, para procesos pick-up y place.

2.7 Redes neuronales

Las redes neuronales son un prototipo basado en el funcionamiento del cerebro humano. Su finalidad es aprender a modificarse automáticamente de tal manera que le permita elaborar tareas complejas que no podrían realizarse con una clásica programación (Atria Innovation, 2019).

2.7.1 Clasificación por el número de capas.

2.7.1.1 Redes neuronales monocapas

Es la más simple, se conectan las neuronas que solo corresponden a la única capa que forma la red.

2.7.1.2 Redes neuronales multicapas

Es una red que está compuesta por diversas capas de neuronas, también llamadas capas ocultas que cumplen la función de intermediarias.

2.7.2 Clasificación por los tipos de conexiones

2.7.2.1 Redes neuronales no recurrentes

Esta red no es muy utilizada, no obstante usan algoritmos en común.

2.7.2.2 Redes neuronales recurrentes

Son redes que tienen memoria gracias a la temporalidad que se crea ya que por no tener capas definidas permiten conexiones libres entre neuronas.

2.7.2.3 Redes neuronales convolucionales

Es una red artificial que permite identificar patrones y aumentar la velocidad de entrenamiento de una manera más avanzada. Reduce conexiones y parámetros para una rápida ejecución

2.7.2.4 Redes neuronales de base radial

Red artificial que contiene tres capas de neuronas, se las utiliza en la clasificación, aproximación de funciones y predicción de series de tiempo. (Sossa, 2021)

2.7.3 Programación de una red neuronal

Para crear una red neuronal se necesitan el número de capas que tiene la red, el número de neuronas en cada capa y la función de activación que se usa para cada capa. Se utiliza un lenguaje de programación el cual ayuda a dar función a la red neuronal.

2.7.4 Phyton

Es un lenguaje de programación de código abierto en el cual se puede desarrollar una web, ciencia de datos y escribir pequeños programas que están diseñados para automatizar labores sencillas (scripting). (Alvarez, 2003)

2.7.5 Librerías de programación

2.7.5.1 OpenCV

OpenCv es una librería de código abierto para visión artificial, aprendizaje automático y procesamiento de imágenes. Admite una amplia variedad de lenguajes de programación como Python, C ++, Java, etc. (Rodríguez, 2021).

CV2 es una librería que forma parte de OpenCV, la cual procesa imágenes digitales dividiéndolas en matrices (Tipán, 2019).

2.7.5.2 SYS

Es un módulo que tiene varios métodos y variables que pueden alterar varios aspectos del entorno de tiempo de ejecución de Python. Permite operar sobre el intérprete dando acceso e información sobre las constantes, variables y funciones que tienen interacción con el mismo (No, 2020).

2.7.5.3 TIME

Es una librería que brinda funcionalidad de cronometraje para Arduino. Permite un boceto para obtener la hora y la fecha como: segundo, minuto, hora, día,

mes y año. Además, permite calcular fácilmente los tiempos transcurridos y sus valores se pueden compartir entre diferentes plataformas. (Leantec, 2015)

2.7.5.4 JSON

Acorde a Villalobos (2013) Json es una notación de objetos y matrices de Java Script, en el cual se puede intercambiar datos basado en texto.

2.7.5.5 Servo.h

Librería que se usa para el control de un servomotor Arduino, el cual controla y soporta 12 motores en las placas de Arduino.

2.7.5.6 NUMPY

Es una biblioteca de Python que permite a los usuarios crear matrices efectivas de varias dimensiones y funciones matemáticas (Aprende con Alf, 2020).

2.7.5.7 TensorFlow

Librería de código abierto para el cálculo numérico con un conjunto de herramientas que realiza una variedad de tareas, incluido el entrenamiento profundo de redes neuronales y la inferencia mediante flujo de datos y programación diferenciable.

Fue liberada baja la licencia Apache 2, la cual expresa que se puede hacer uso del Software manteniendo siempre los derechos de autor, pero no se puede hacer uso o redistribución del código fuente (Delgado, 2017).

2.7.5.8 TFLITE

TensorFlow Lite grupo de herramientas que facilita el funcionamiento de modelos de aprendizaje automáticos (Rodriguez, 2017).

2.8 LabelImg

Herramienta gratuita de código abierto para etiquetar imágenes gráficamente. Está escrito en Python y es una manera fácil de etiquetar imágenes para detectar objetos. (Nelson, 2020)

2.9 Google Colab

Es una herramienta que permite analizar datos, combinar texto, código y resultados de código en un solo documento (Santos, 2020).

2.10 Solidworks

Es un software de diseño CAD 3D que se utiliza para la planificación, el modelado, la evaluación de viabilidad, la creación de prototipos y la gestión de proyectos. Utiliza el principio de diseño paramétrico y genera tres tipos de archivos interconectados: la pieza, el ensamblaje y el dibujo (Talentum, 2020).

2.11 Sistema de transporte

Los sistemas de transporte son la pieza fundamental en las líneas de producción, ya que son la herramienta que moviliza el producto de un punto A hacia

un punto B, estos sistemas están elaborados dependiendo de los objetos a trasportar, se debe considerar los puntos de elevación, la consistencia física del producto, entre otros.

2.11.1 Banda transportadora

Las cintas de transporte o bandas de transporte consisten en una estructura metálica las cuales son parte de un sistema de transporte continuo que se mueve entre dos tambores, por lo general la banda transportadora es arrastrada por el roce de sus tambores, que a su vez es activada por un motor en uno de sus extremos y en el otro posee un tambor que suele girar libremente cumpliendo la función de retorno de la banda. (IRP, 2018).

Figura 6. Cinta transportadora de banda (IRP, 2018)

2.11.1.1 Rodillos

Son componentes de manipulación de materiales de forma cilíndrica. Son ampliamente utilizados en múltiples industrias, porque permiten agilizar las operaciones de transporte de objetos, dado que pueden mover diferentes tipos y volúmenes de productos, en distancias largas y cortas sin tener que contar con la intervención de un operario (Blog, 2018).

2.12 Sistema Electrónico

Son componentes electrónicos que, dependiendo de su conexión, permiten realizar distintas funciones para cualquier equipo electrónico a través de una fuente de corriente sin importar su complejidad (Barriga, 2020).

2.12.1 Raspberry

Es un ordenador con un formato compacto utilizado para desarrollar pequeños prototipos y para la formación sobre informática y electrónica.

Consta de un sistema operativo, Raspbian, basado en Linux y especial para aprovechar el hardware de este microordenador (Delgado A. , 2020).

Figura 7. Raspberry Pi 4B (Alo Tech Solutions, 2020)

2.12.1.1 Cámara Raspberry Pi V2

Es un módulo con una lente de foco fijo de alta calidad, capta imágenes estáticas de 3280 x 2464 pixeles y también capta vídeo de 1080p30 y 640x480p90.

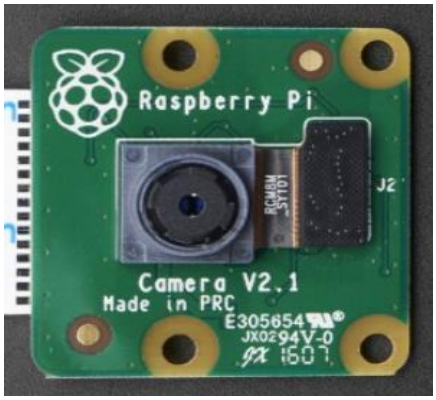

Figura 8. Raspberry PI Versión 2 de 8 Megapixels (SANDOROBOTICS, 2019)

2.12.1.2 RPi LCD

Pantalla de cristal líquido que se utiliza para visualizar imágenes y videos a color o escala de grises.

Figura 9. WS 3.5inch RPi LCD.

2.12.2 Arduino

Es una plataforma electrónica de código abierto basada en hardware y software libre, contiene un microcontrolador reprogramable las cuales realizan instrucciones que permiten la interacción con los circuitos de la placa (Fernandez, 2020).

2.12.2.1 Arduino nano

Arduino Nano es una placa basada en el microcontrolador ATmega328P. Contiene las mismas capacidades que un Arduino UNO, tanto en potencia del microcontrolador como en conectividad (Microcontroladores, 2020).

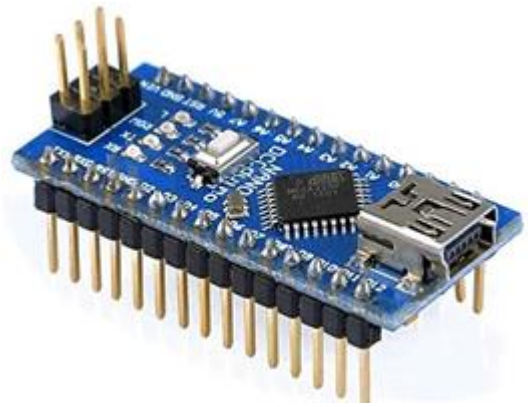

Figura 10. Arduino nano (Arduino, 2019)

2.12.2.2 Arduino IDE

Es un programa que se utiliza para compilar e interpretar códigos para el desarrollo de programas utilizados en una placa Arduino. Es compatible con Linux y Windows, por lo que su versatilidad es un punto más a favor de estos dispositivos para proyectos electrónicos (Netinbag).

2.12.3 Sensores

Un sensor es un dispositivo de medición que detecta y realiza el cambio de un fenómeno físico en un voltaje analógico medible o en una señal digital (MecatrónicaLATAM, 2021).

2.12.3.1 Sensores industriales

Los sensores industriales son capaces de variar magnitudes físicas o químicas y trasformar estos valores en variables eléctricas, estas variables son de diversos tipos tales como: temperatura, movimiento, humedad, presión, inclinación, intensidad lumínica, fuerza, etc. Estos sensores pueden estar conectados a un computador para facilitar la lectura de los datos (Brunete, Segundo, & Herrero, 2020).

2.12.3.2 Sensor Infrarrojo

El Módulo Sensor De Obstáculos Reflectivo Infrarrojo FC-51 es un dispositivo optoelectrónico, se utiliza en la industria para el conteo de la producción; en uso personal sirve para sistemas de seguridad por medio de infrarrojo IR que está compuesto por un transmisor que emite energía IR y un receptor que detecta la energía IR. El sensor puede funcionar con luz ambiente o en la obscuridad.

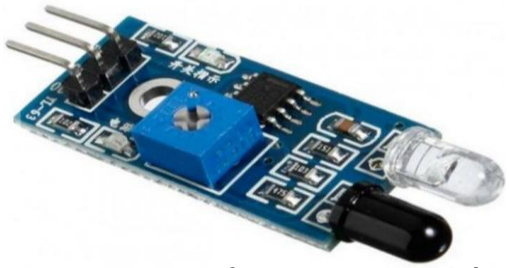

Figura 11. Sensor Infrarrojo (Web-Robótica, 2019)

2.12.3.3 Sensores capacitivos de proximidad

Estos sensores se basan en la interacción producida entre el objeto a detectar y el campo electrostático que genera el propio sensor. El funcionamiento es similar al caso inductivo, pero ahora el sensor es un condensador.

Figura 12. Sensor de proximidad capacitivo (Bookdown, 2020)

2.12.4 Motores

Los motores eléctricos son una máquina que involucran bobinas giratorias que son impulsadas por la fuerza magnética ejercida por un campo magnético sobre una corriente eléctrica, logrando transformar la energía eléctrica en energía mecánica.

Algunos de estos motores, convierten la energía de manera contraria a la explicada anteriormente, funcionando de este modo como generadores. Las locomotoras y automóviles híbridos pueden lograr realizar las dos tareas si son diseñados correctamente (Nuñez, 2020).

2.12.5 Servomotor

Motor eléctrico que permite el control preciso de posición angular, aceleración y velocidad. Contiene una combinación de piezas específicas, las cuales incluyen un motor de corriente continua o alterna. Un servomotor (figura 2.12) tiene integrado o adosado al menos un detector que permita conocer su posicionamiento y/o velocidad. A los detectores de posición se les llama encoders (González, 2016).

Figura 13. Servomotor MG996r (HobbyKing, 2020)

2.12.6 Estabilizadores de tensión

Es un regulador de voltaje que permite que el flujo de la corriente sea estable, con el fin de proteger artefactos eléctricos de problemas como sobrevoltaje, caída de tensión y variaciones de voltaje (Hiraoka).

2.12.7 Transistor

Es un elemento utilizado como amplificador de voltaje o corriente, o como interruptor. Está hecho de una pieza sólida de material semiconductor, con tres terminales: base, emisor y colector (Mecafenix, 2019).

2.12.8 Resistencia

Es un componente eléctrico pasivo que crea resistencia en el flujo de corriente eléctrica. Se utilizan para varios propósitos, como la limitación de corriente eléctrica, división de voltaje y configuración de constantes de tiempo (Medina J.).

2.12.9 Disipador de calor o HeatSink

Componente que se utiliza para evitar el sobrecalentamiento de los dispositivos electrónicos.

Figura 14. HeatSink (Indiamart, 2018)

2.12.10 Conversor de potencia

Es un sistema que altera las características de la tensión y corriente que recibe. Como objetivo primordial tiene la conversión de un tipo de energía a otro (Huerta, 2020).

2.12.10.1 Convertidor Buck

Son circuitos que convierte la tensión de entrada DC a una salida DC de menor nivel. Almacenan energía y la descargan a los niveles de voltaje que son buscados o deseados.

Figura 15. DollaTek XL4015 5A DC-DC (Electronico Caldas, 2019)

2.12.11 Pistones

Es una pieza que es parte de la maquinaria de un motor, la cual realiza movimientos de forma alternativa dentro de un cilindro, teniendo como función cambiar el volumen y la presión del fluido para conseguir movimiento (Menna).

2.12.12 Piñones

Son piezas que se utilizan para generar transmisión de fuerza y movimiento. Están fabricados con acero de alta calidad y se adaptan para obtener un alto rendimiento (SADITRANSMISIONES).

2.12.13 Fuente de alimentación

Son las encargadas de suministrar energía a los componentes electrónicos. Convierte la corriente alterna (CA) en corriente continua (CC) las cuales generan distintos voltajes utilizando inductores y condensadores (Alonso, 2021).

3. MARCO METODOLÓGICO

3.1 Metodología

El análisis cualitativo brinda el mejor soporte en cuanto al marco metodológico del proyecto técnico. Este tipo de investigación toma en consideración que la realidad tiende a estar en constante cambio, al interpretar la realidad se obtendrán resultados subjetivos, además considera los diferentes tipos de datos tales como imágenes, audios, documentos, observación, entre otros. (Bryman, 2004)

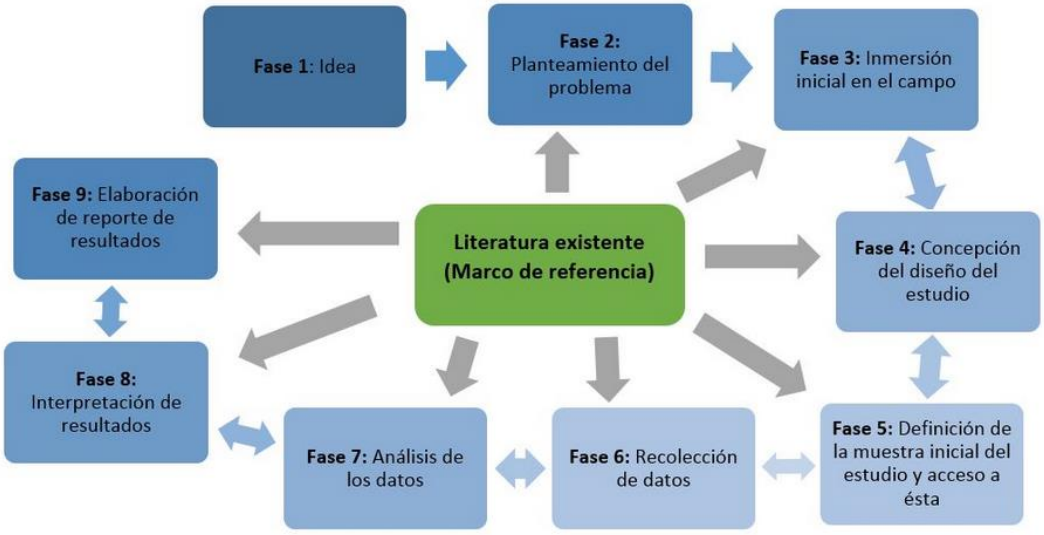

Figura 16. Proceso del análisis cualitativo (Hernández, Fernández y Baptista, 2014)

3.2 Análisis funcional

El análisis funcional es una técnica utilizada para identificar las diferentes funciones productivas que conlleva un proceso, los análisis se basan en lograr identificar las funciones principales las cuales se deben cumplir para llegar al objetivo final.

La finalidad de aplicar este método es diseñar un mapa funcional en el cual se pueda integrar cada parte del proyecto, esto permite establecer jerarquías en los procesos y que se logre una constancia en los resultados

El proceso está compuesto por 4 etapas las cuales se grafican en la figura 3.2, la etapa de transporte se encarga de iniciar el proceso llevando los objetos hasta su fase final, la segunda etapa se encarga de la visión artificial la cual obtiene las imágenes en tiempo real de los objetos, en tercer lugar, se encuentra la red neuronal la cual analiza los datos obtenidos y entrega una predicción calculada por la red. (ChileValora, 2012)

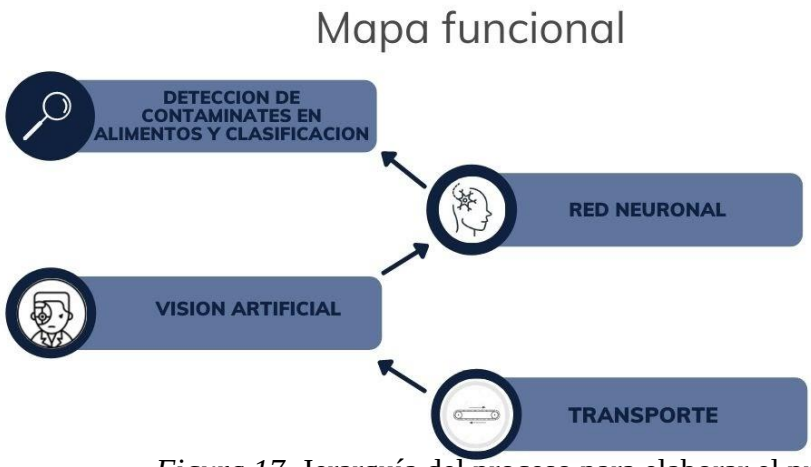

Figura 17. Jerarquía del proceso para elaborar el prototipo

3.3 Estructura física.

Para la fabricación de la estructura se toma como base una banda transportadora, en la cual se adiciona un sistema de pistones mecánicos los cuales se accionan mediante la programación de la red neuronal. Para crear las piezas, se utiliza el software Solidworks, el cual facilita su manipulación antes de la creación física y ayuda a dar una perspectiva de las piezas en 3D.

Este software cuenta con la opción de ensamble la cual permite formar la estructura final del prototipo y las perspectivas desde diferentes ángulos, una de las opciones es usar las diferentes capas para darle una apariencia más realista.

3.3.1 Banda transportadora

Una de las partes fundamentales es la banda transportadora, que permite la movilidad de los objetos y la superficie que mantiene contacto con los objetos, la cual mediante el accionamiento mecánico de los motores eléctricos transporta los objetos para cumplir con el proceso. Se inicia graficando el perfil de la cinta transportadora.

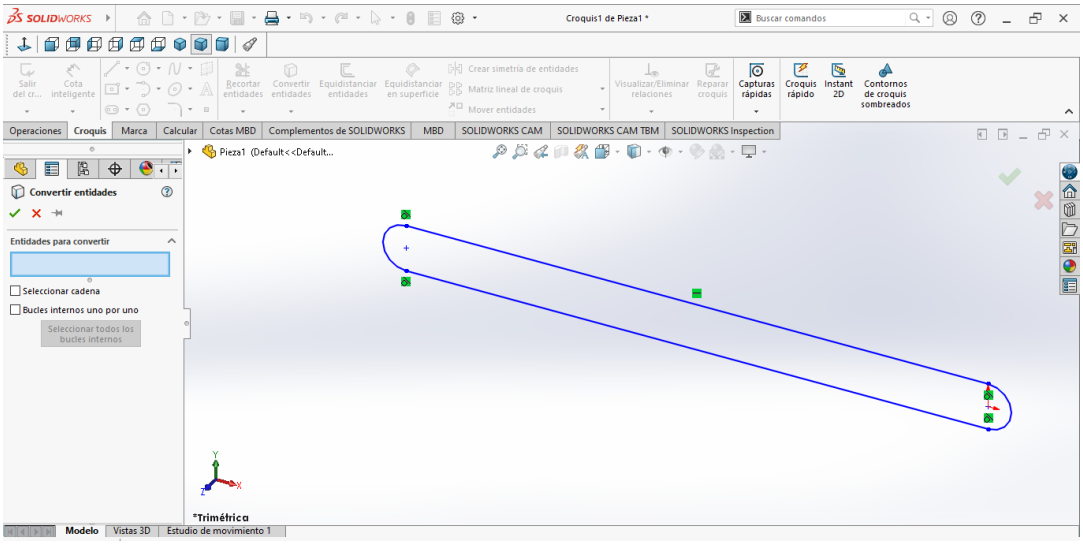

Figura 18. Perfil de la cinta transportadora.

Se aplica la herramienta de extrusión de materiales para darle forma a la gráfica que se elaboró anteriormente, se colocan las medidas específicas en centímetros.

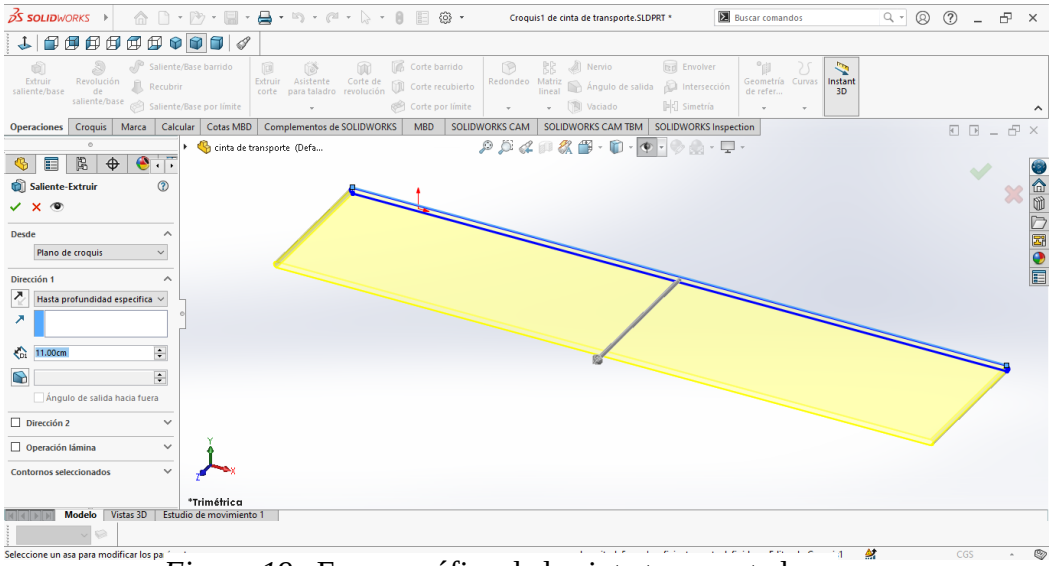

Figura 19. Forma gráfica de la cinta transportadora.

Se obtiene la banda de transporte de los objetos y se procede a guardar este archivo como una pieza, de esta manera se puede utilizar en el ensamble.

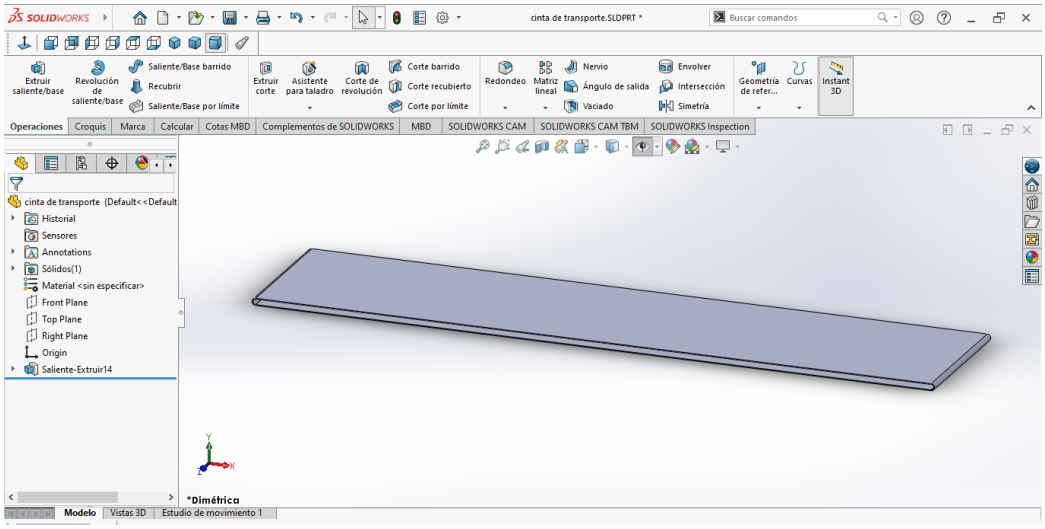

Figura 20. Banda de transporte de los objetos.

Se elabora el modelo de los soportes laterales de la banda transportadora, se escoge un plano de trabajo y se trazan las medidas correspondientes, se delimita toda la superficie para tener un mayor control de las medidas.

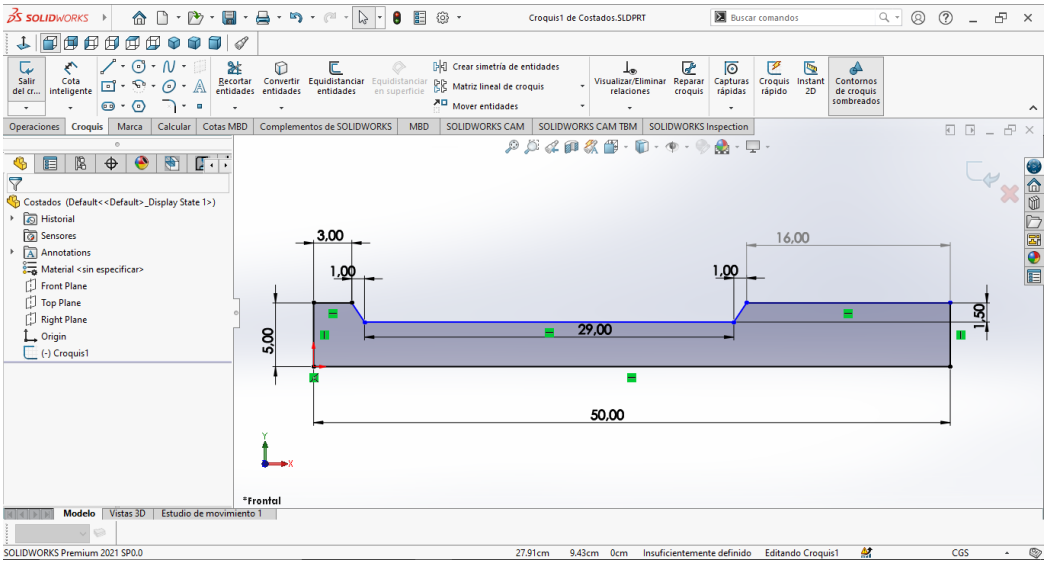

Figura 21. Modelo de soportes laterales de la banda transportadora.

Se grafican los orificios que sirven de soporte para los rodillos centrales que se elaboran y se toma en cuenta la distancia de los mismos, para tener una simetría en la pieza.

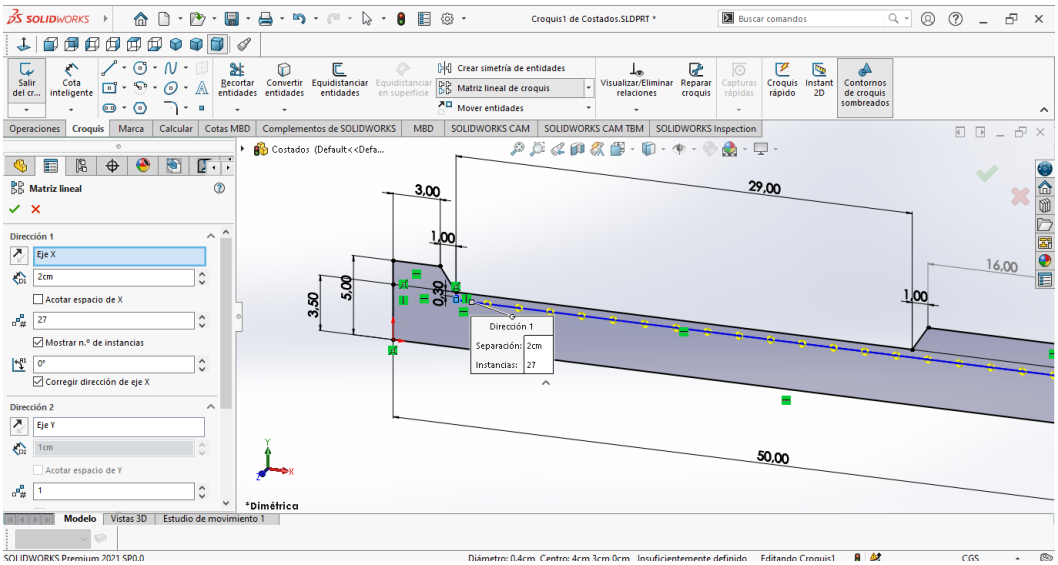

Figura 22. Gráfico de orificios para los soportes de los rodillos.

Se utiliza la herramienta matriz de línea para ordenar los orificios y tener una guía más específica de ellos.

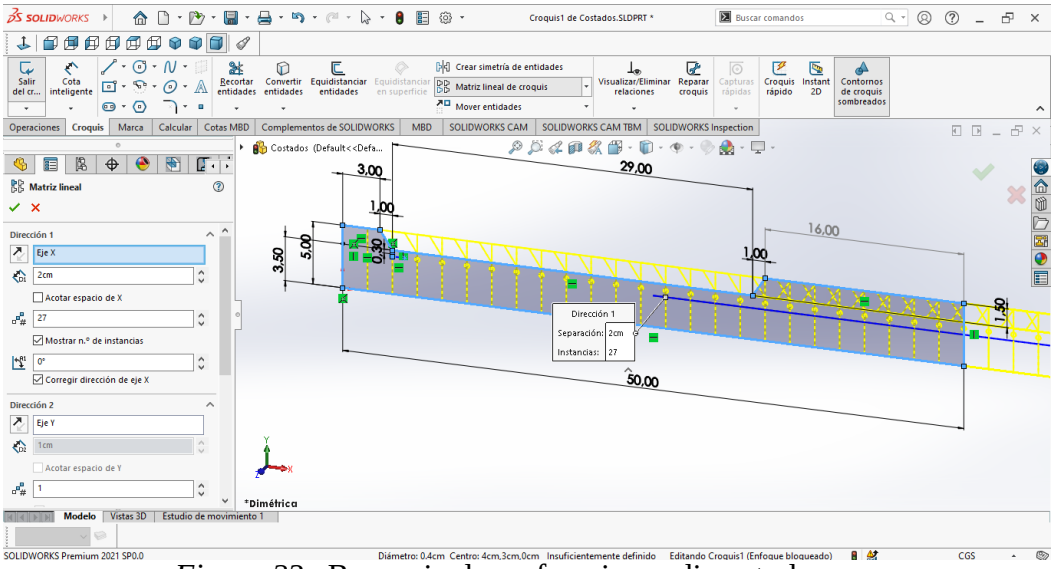

Figura 23. Bosquejo de perforaciones diametrales.

Se aplica la herramienta extruir para darle el espesor adecuado a la pieza y quitarle los espacios en los que se apoyan los soportes centrales, se obtiene una pieza modelada en 3D.

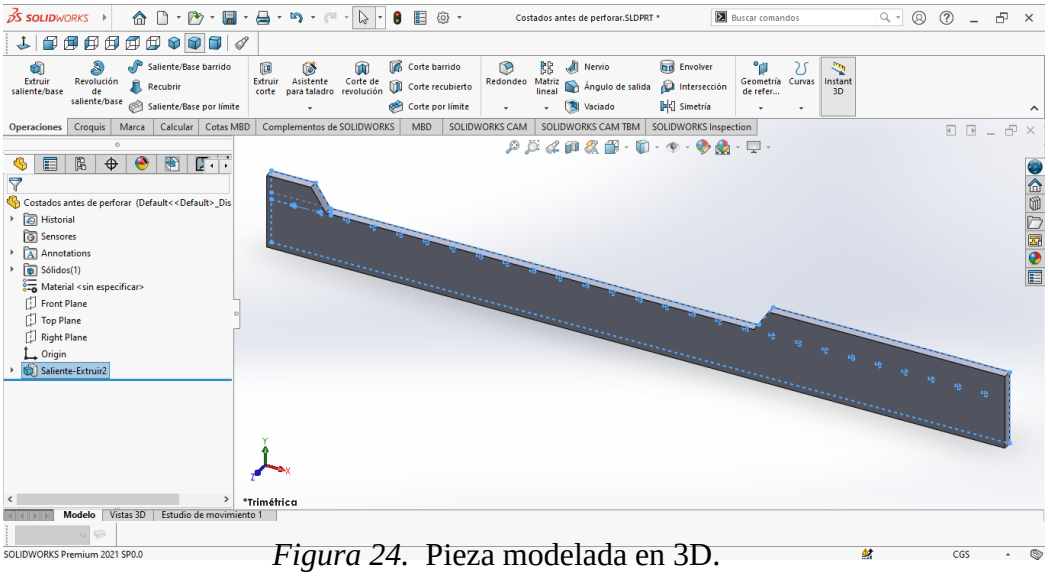

Figura 24. Pieza modelada en 3D.

3.3.2 Pistones

La clasificación de las frutas es una de las partes importantes del prototipo, por esto se diseña un elemento tipo pistón accionado por un servo motor, el cual se acciona por el Arduino al momento de la clasificación, separando las frutas de la banda transportadora.

Se grafica en el programa Solidworks las líneas que dan forma a la cabeza del pistón.

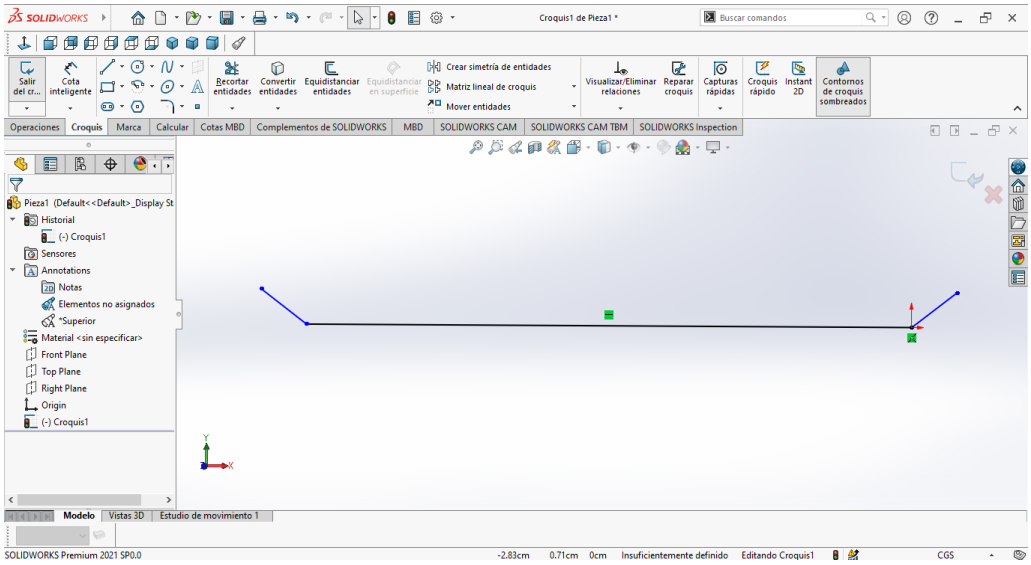

Figura 25. Trazado de la cabeza del pistón.

Se ejecuta la extrusión del plano, se agrega el grosor adecuado de los materiales y se procede a guardar la pieza.

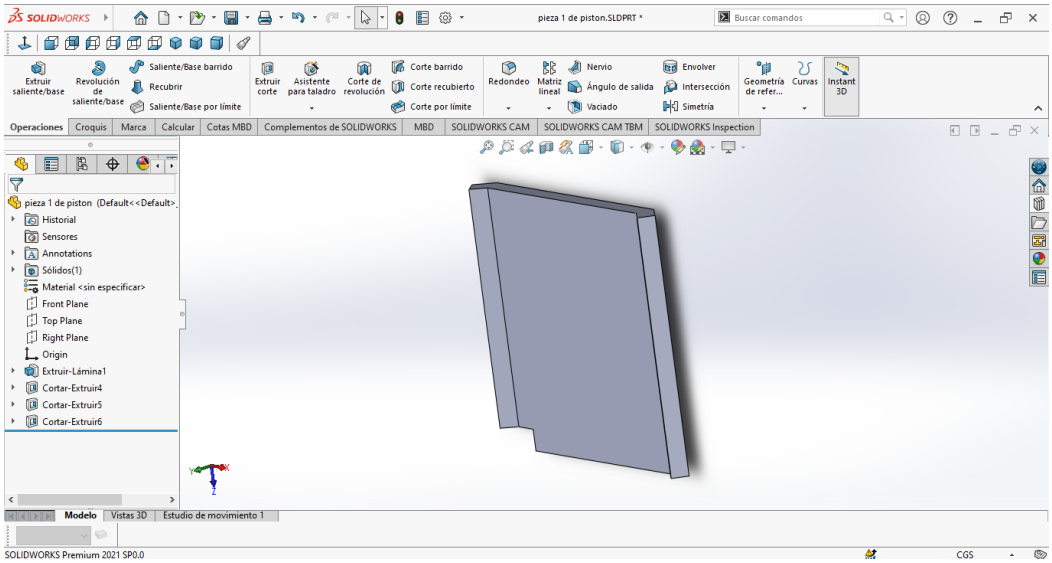

Figura 26. Extrusión de la cabeza del pistón.

Se elabora una caja la cual sirve para contener el mecanismo de los pistones, se ejecuta un cubo, con la herramienta de vaciado de objetos, se da forma a la caja.

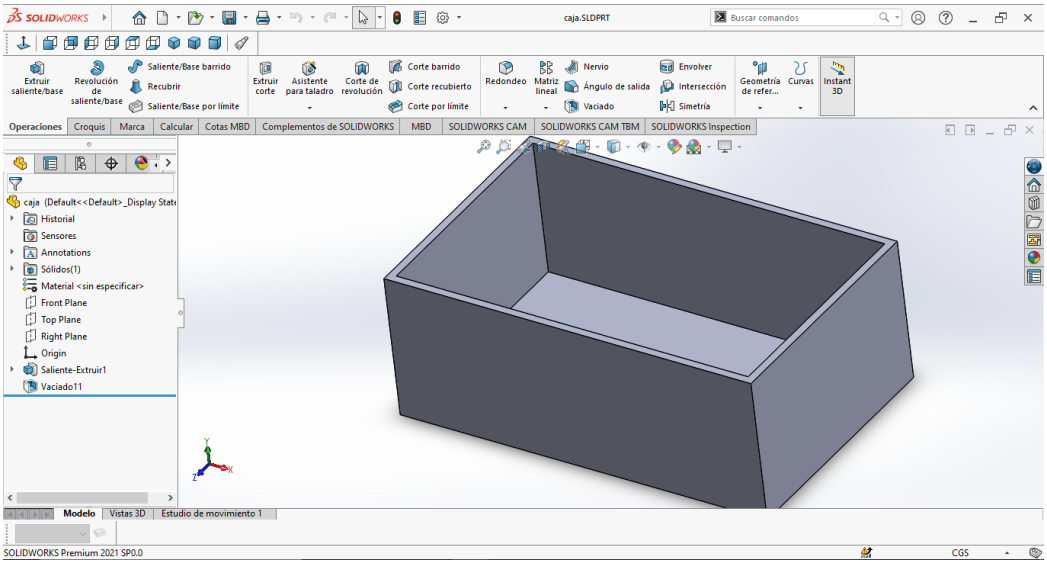

Figura 27. Contenedor de los pistones.

Se completa armando los componentes internos de los pistones los cuales movilizan las frutas de la banda transportadora según se requiera.

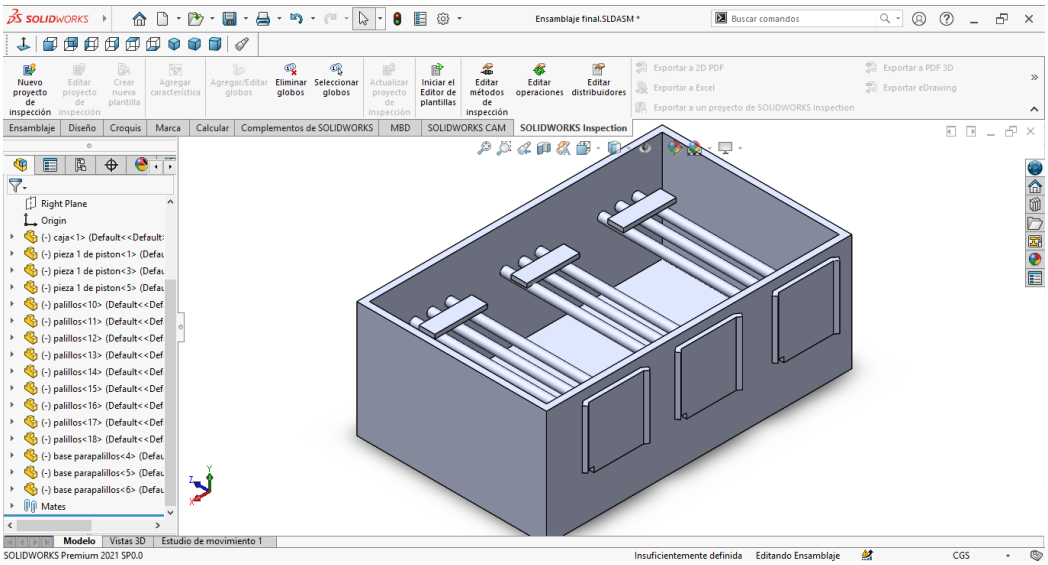

Figura 28. Armado de los pistones.

3.3.3 Piñones

Para la movilidad de la banda se consideran diferentes modelos de acoplamientos entre el motor eléctrico y el rodillo, se acopla con piñones para aumentar el torque de los motores ayudando a la banda a mover objetos más pesados, se grafica los objetos en el plano siguiendo medidas específicas de los piñones ya que estos vienen prefabricados, se los diseña en SolidWorks para completar el ensamble.

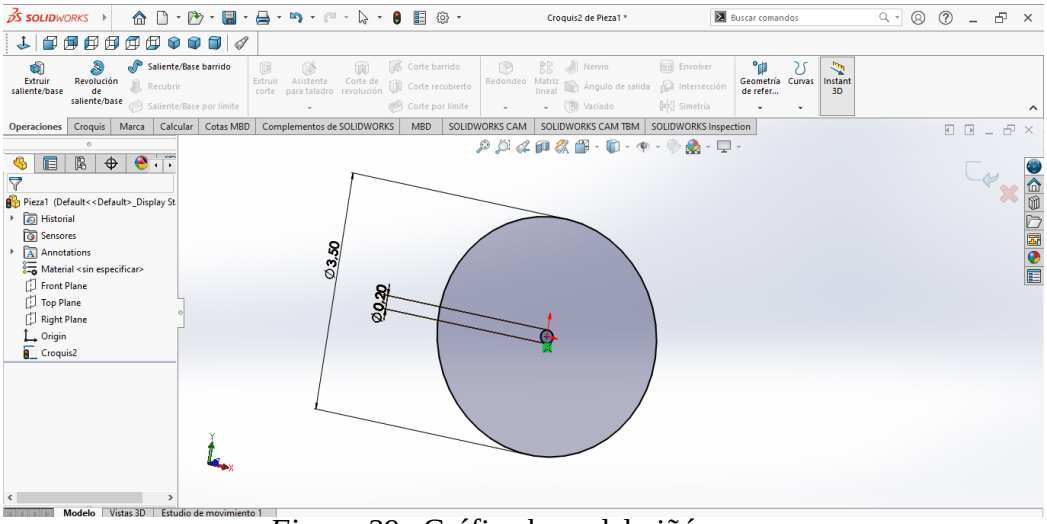

Figura 29. Gráfico base del piñón.

Se aplica la herramienta extruir para brindarle espesor a la pieza.

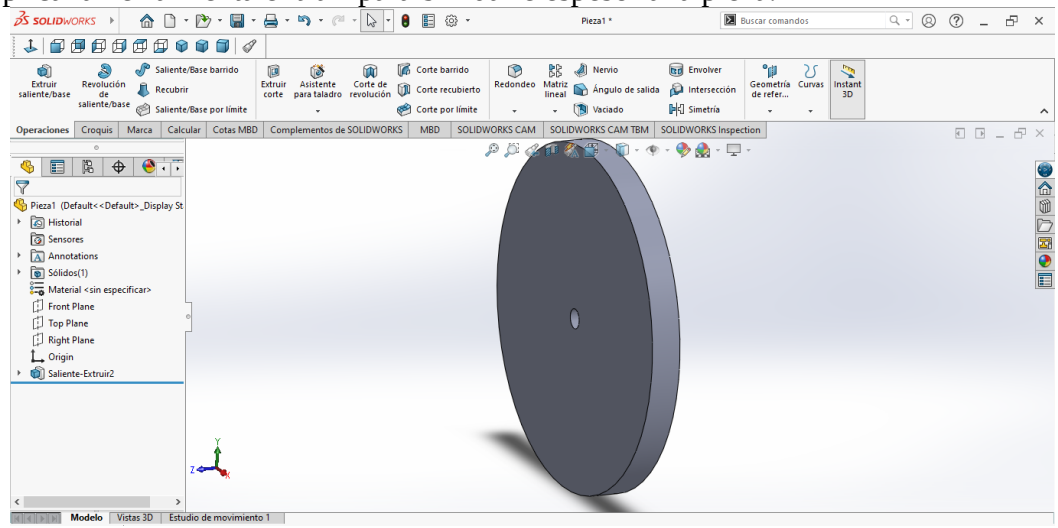

Figura 30. Base de piñón moldeada.

En la elaboración de los dientes de los piñones se grafican círculos tangentes a los círculos primarios del piñón.

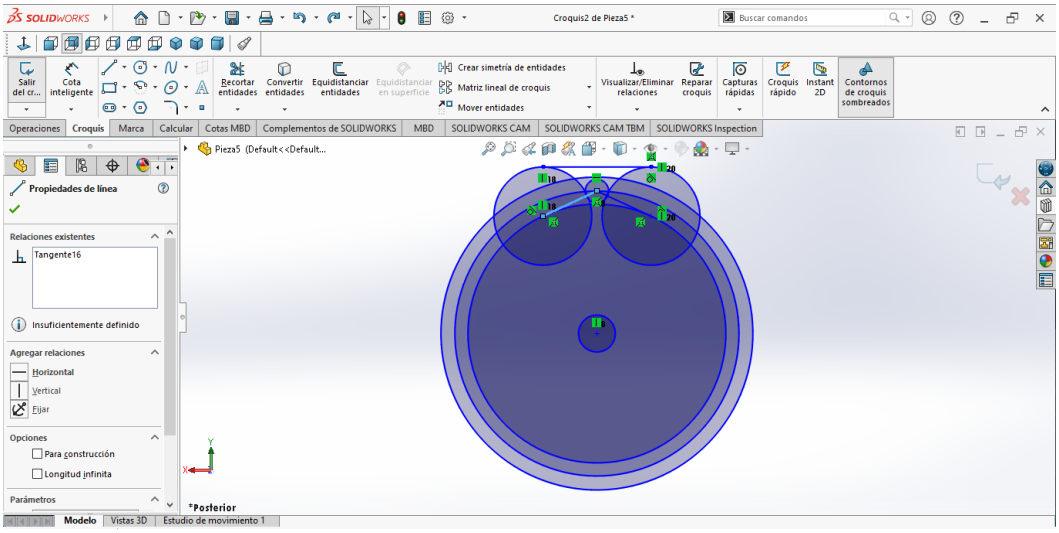

Figura 31. Gráfico base dentada.

Se recortan los círculos para dejar una perforación en la estructura creada, ya que da la forma dentada que se necesita.

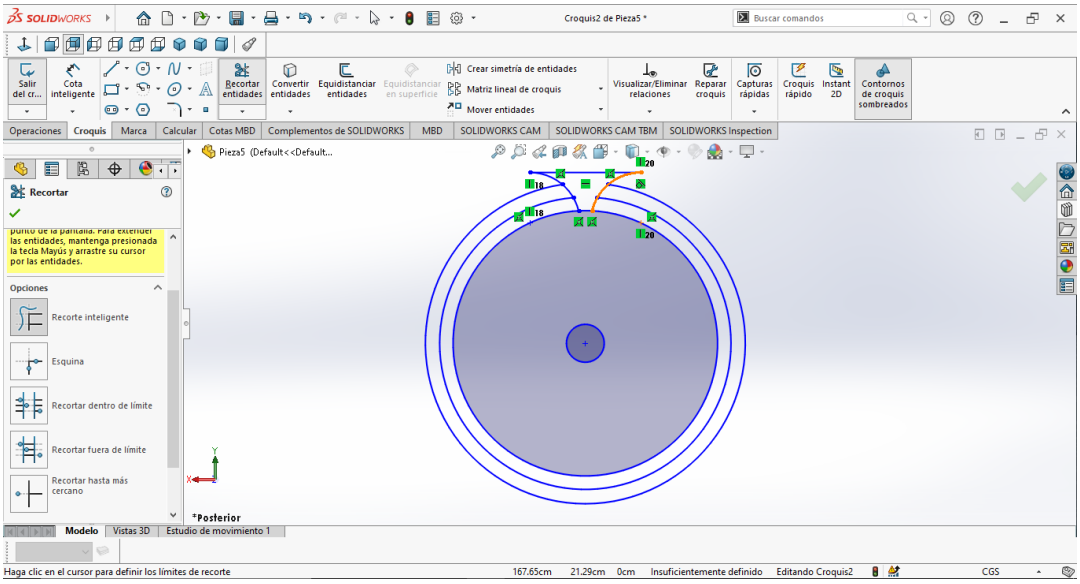

Figura 32. Recorte de base dentada.

Se aplica la herramienta extruir y se visualiza un corte en el disco, el cual es la base para los siguientes dientes.

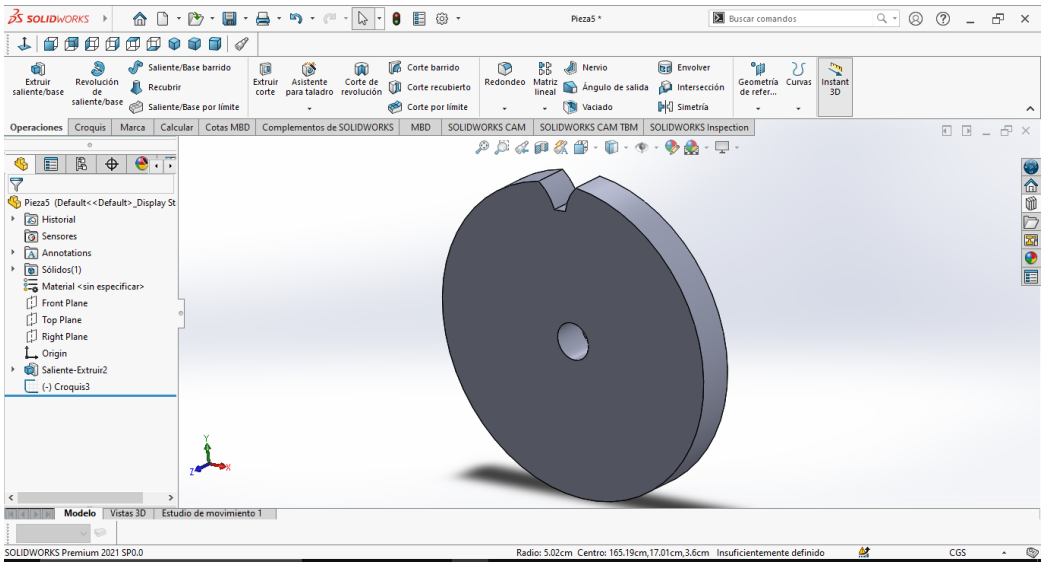

Figura 33. Modelo ranurado.

Se aplica la herramienta matriz de rotación, la cual brinda la posibilidad de replicar el corte antes aplicado al disco para crear los dientes del piñón.

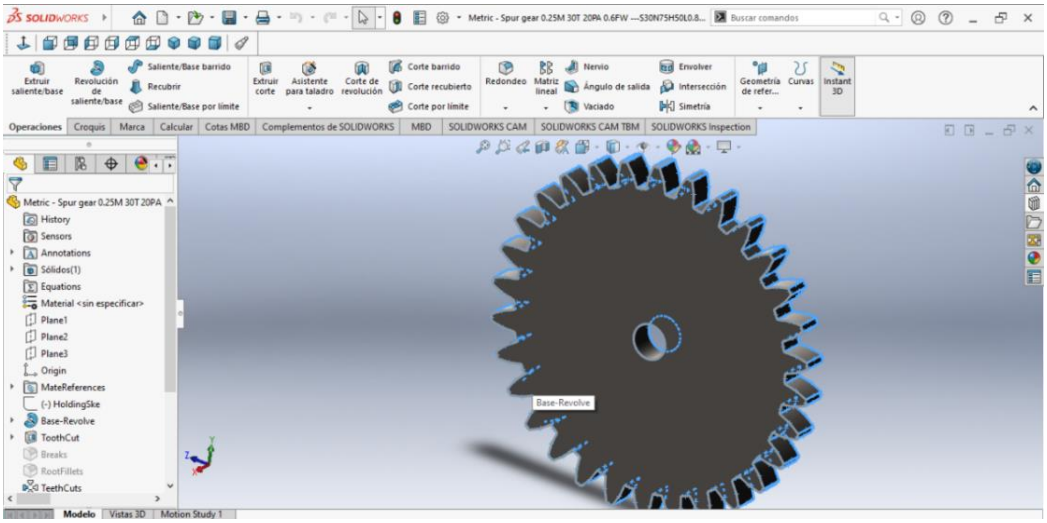

Figura 34. Piñón elaborado.

Para la contención de los piñones se debe crear una estructura en la cual se colocan los soportes para los piñones y brinde el movimiento adecuado, se grafica siguiendo las medidas deseadas.

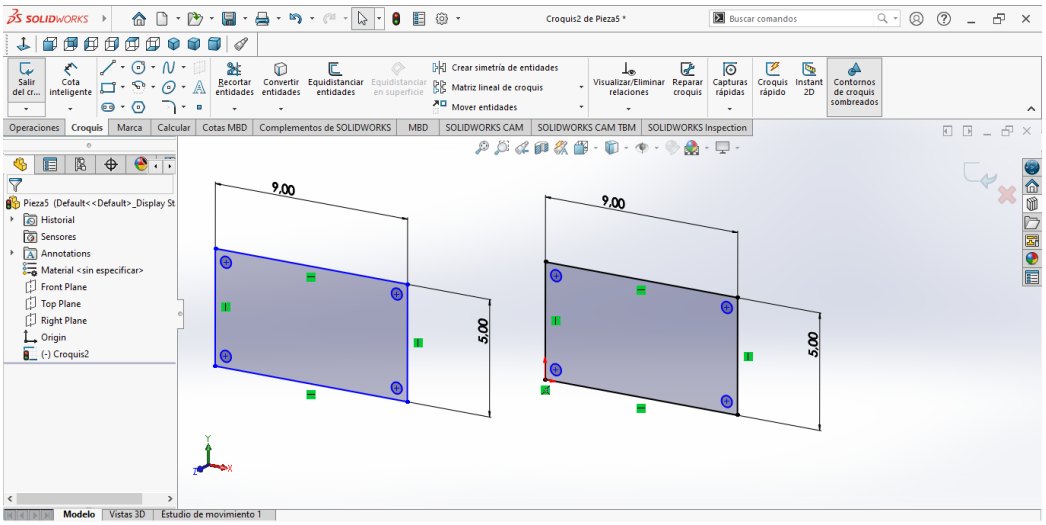

Figura 35. Soporte de piñones.

Se extruyen las bases con su respectivo grosor y se procede a guardarlas como piezas que son utilizadas en el ensamble del engranaje.

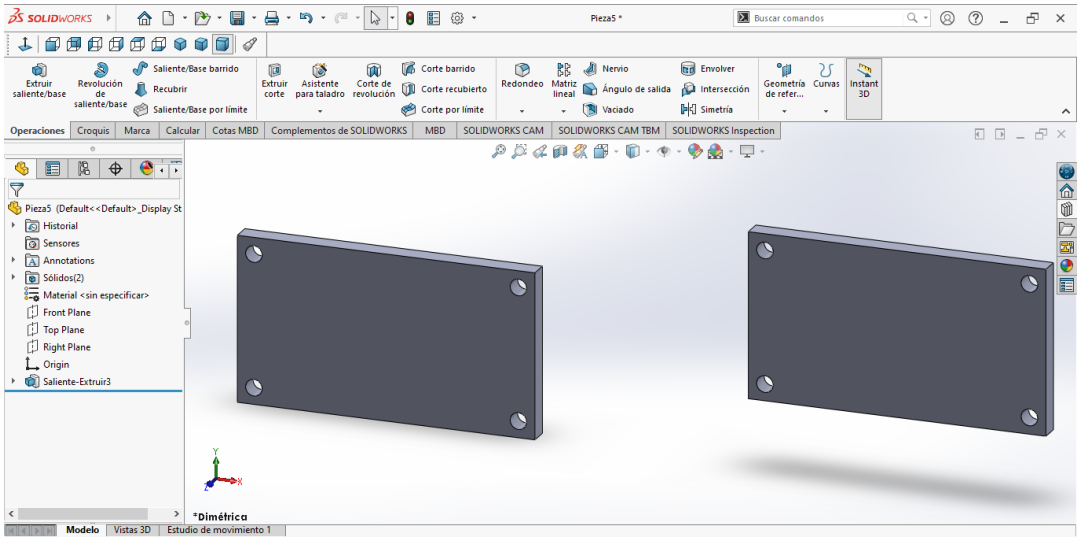

Figura 36. Soportes de piñones 3D.

Se utilizan las bases antes creadas para ordenar diversos piñones, los cuales transmiten la energía del motor eléctrico a la cinta transportadora. Se utilizan piñones con una variedad de dientes.

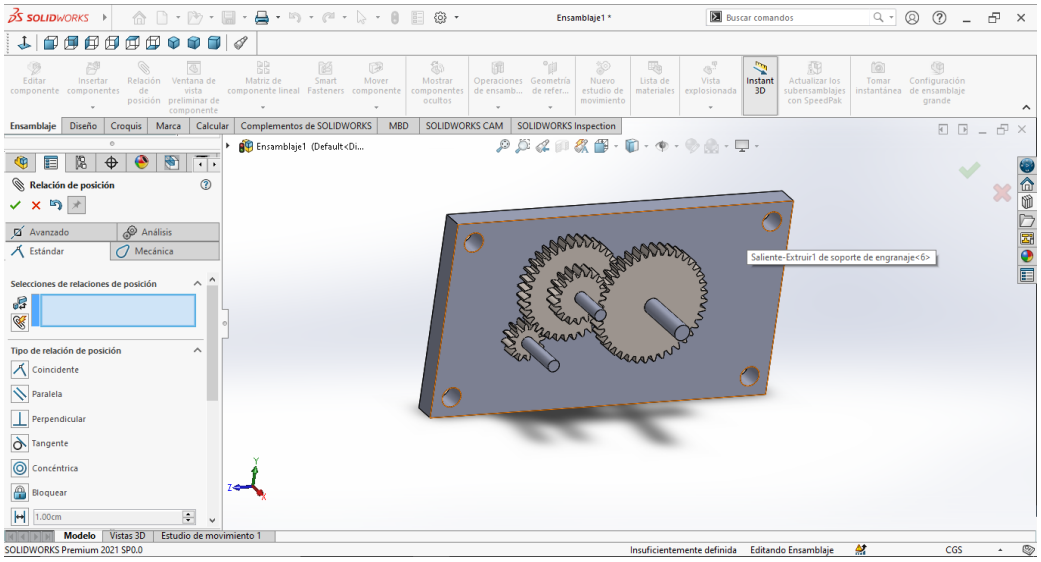

Figura 37. Juego de Piñones.

Se procede a unir la pieza final para contener los engranajes, se alinean los ejes de los piñones para poder conectar el motor eléctrico.

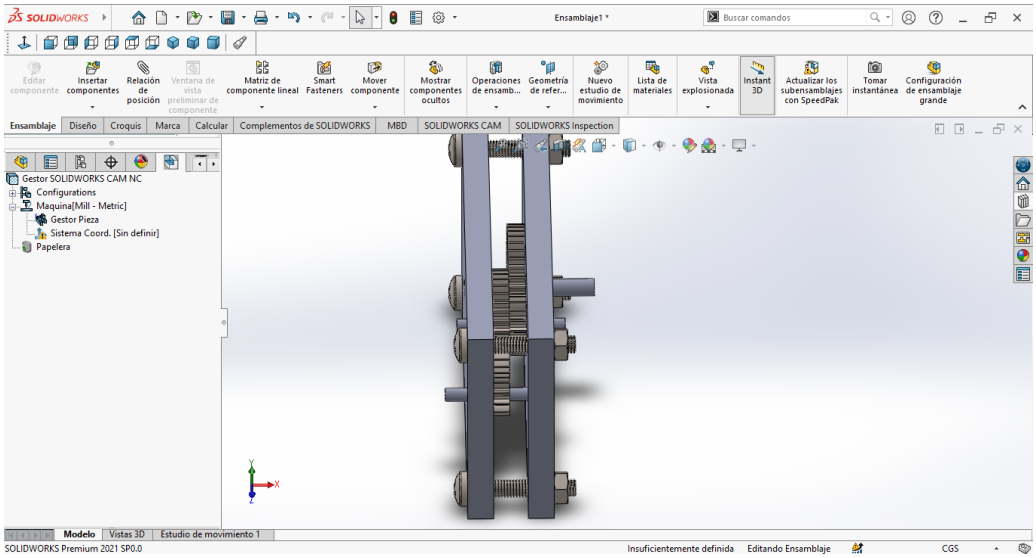

Figura 38. Contenedor de Piñones.

Se acoplan los tornillos y tuercas necesarias para ajustar las dos placas que contiene el juego de engranajes.

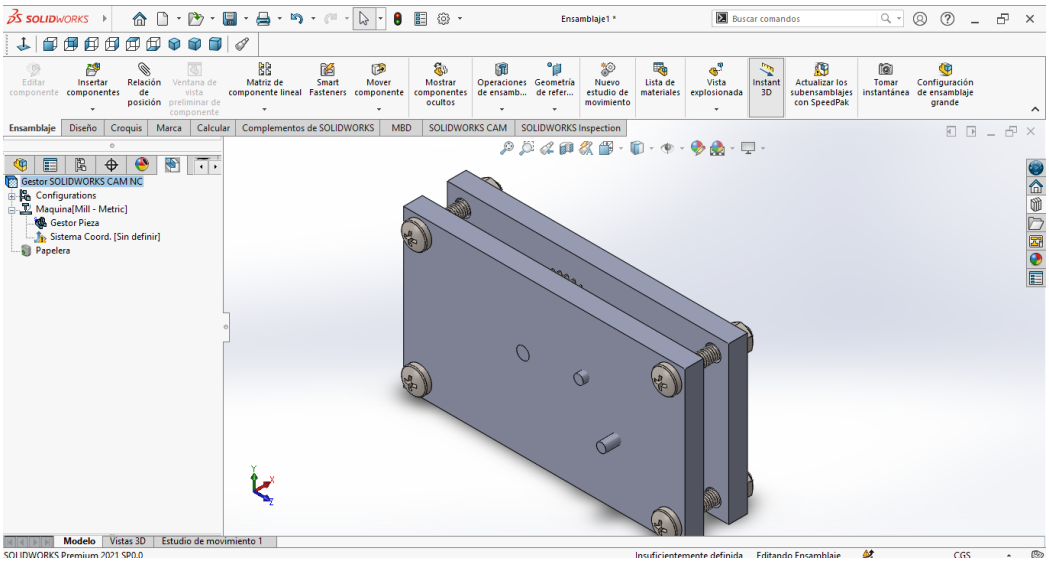

Figura 39. Empernado de contenedor.

Como revisión final se utiliza la visión de los planos entre la estructura, así se verifica que el ensamble se establezca apropiadamente.

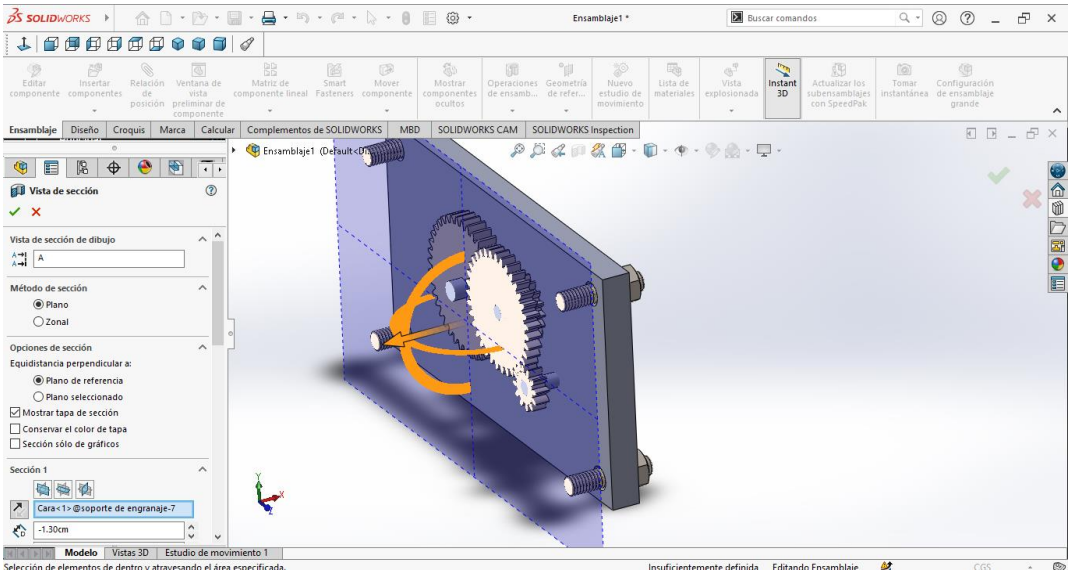

Figura 40. Visión Entre planos.

3.3.4 Rodillos

Se grafica en el plano la estructura que es el soporte para la banda transportadora en todo su recorrido y une los soportes laterales de la banda. Se grafica un círculo y se aplica la herramienta extruir, se forma un cilindro el cual ayuda a la rotación de los objetos con facilidad para la banda, se reparten 27 de estos cilindros a través de la estructura.

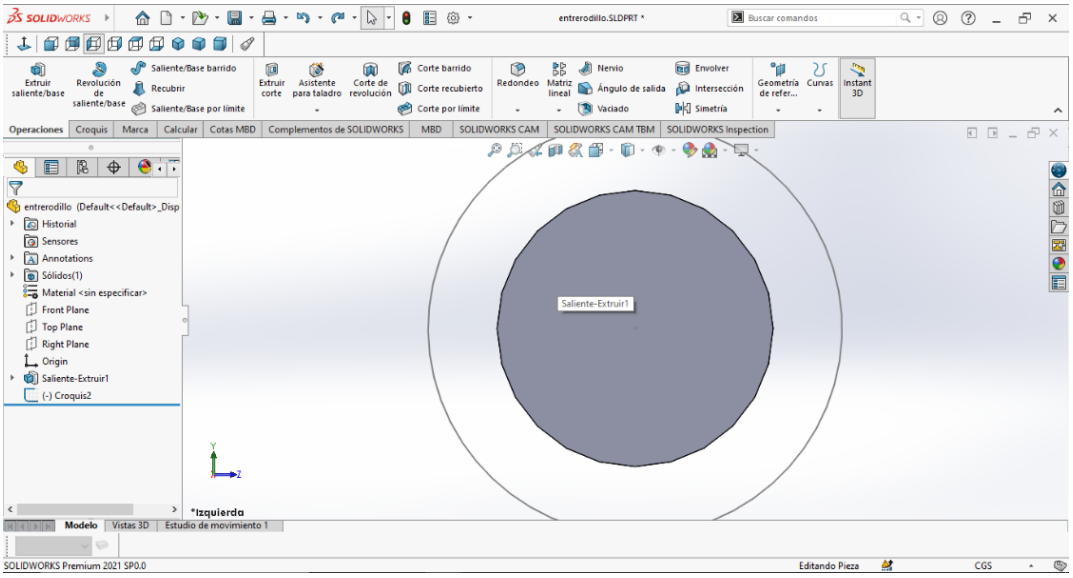

Figura 41. Gráfico base rodillos.

Se extruyen las gráficas de ambos lados

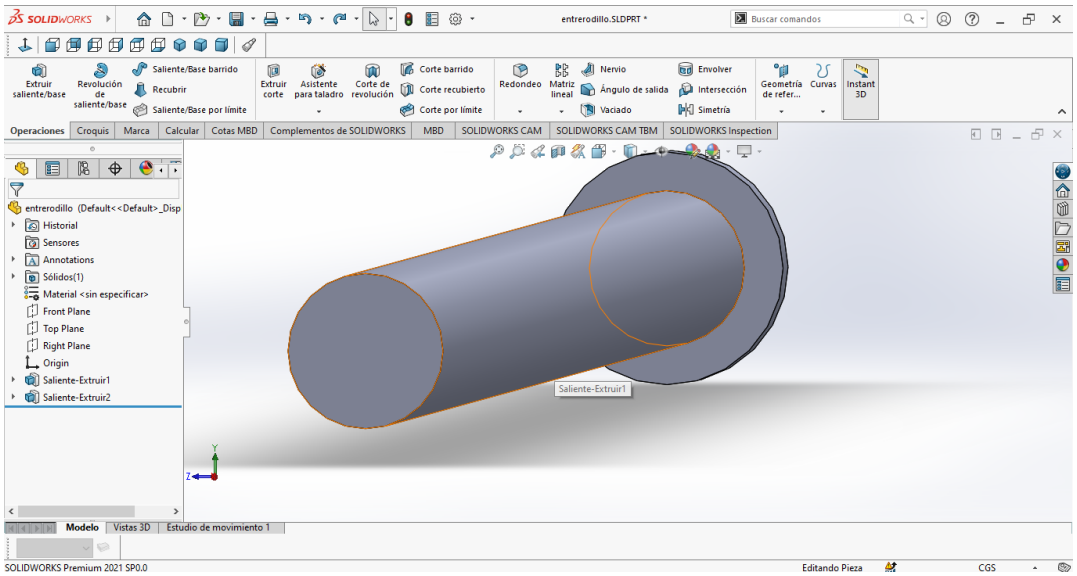

Figura 42. Extrusión de bases de rodillo.

Se define la estructura y se la guarda como una pieza para el ensamble final.

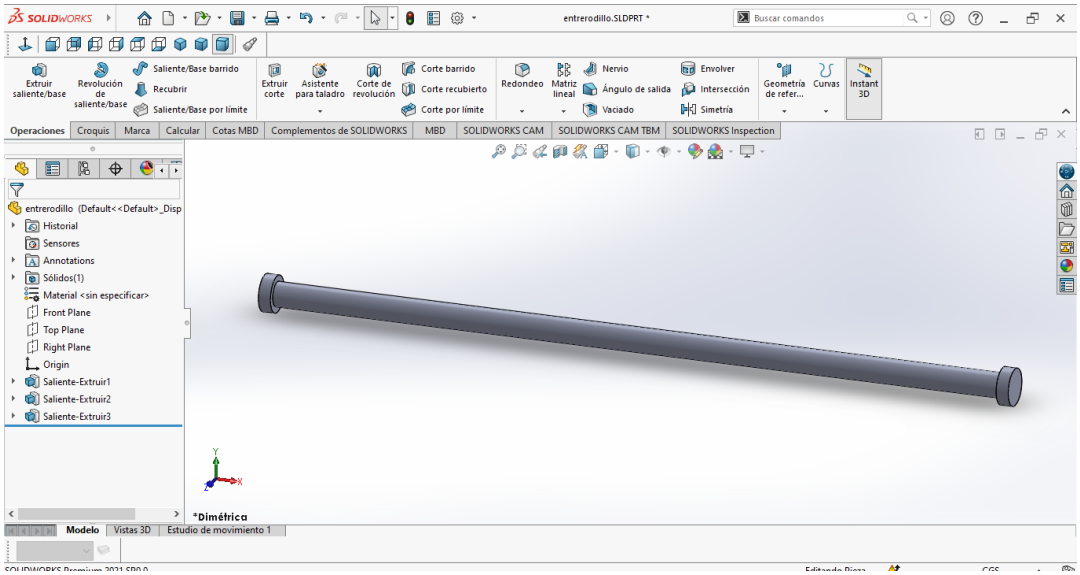

Figura 43. Rodillo 3D.

3.3.5 Ensamble

Inicialmente se abre un documento de ensamble, se extrae cada una de las piezas previamente elaboradas y se procede con el ensamble.

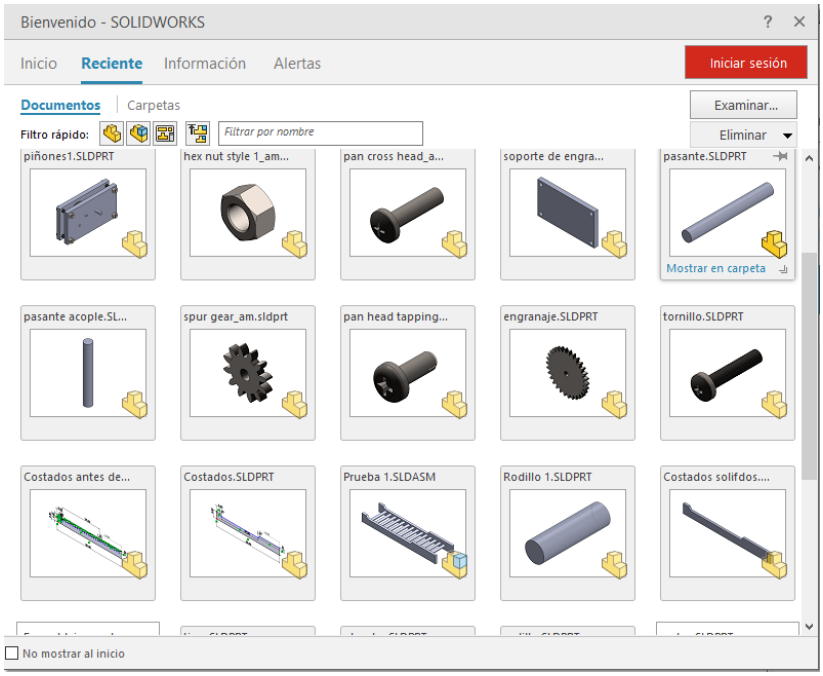

Figura 44. Galería de Piezas elaboradas.

Se toman las distancias adecuadas entre las piezas, con la herramienta relación de posición se colocan las piezas de manera ordenada en los lugares correspondientes, dando forma al soporte de la banda.

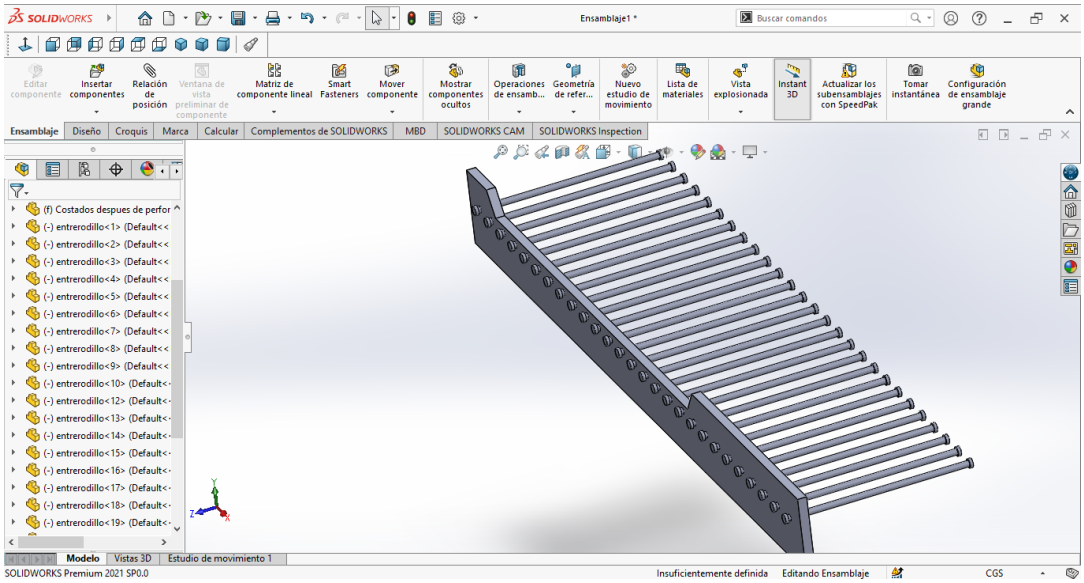

Figura 45. Soporte lateral con rodillos.

Se complementa con otro soporte lateral, se aplica la herramienta relación de posición para las guías de la estructura.

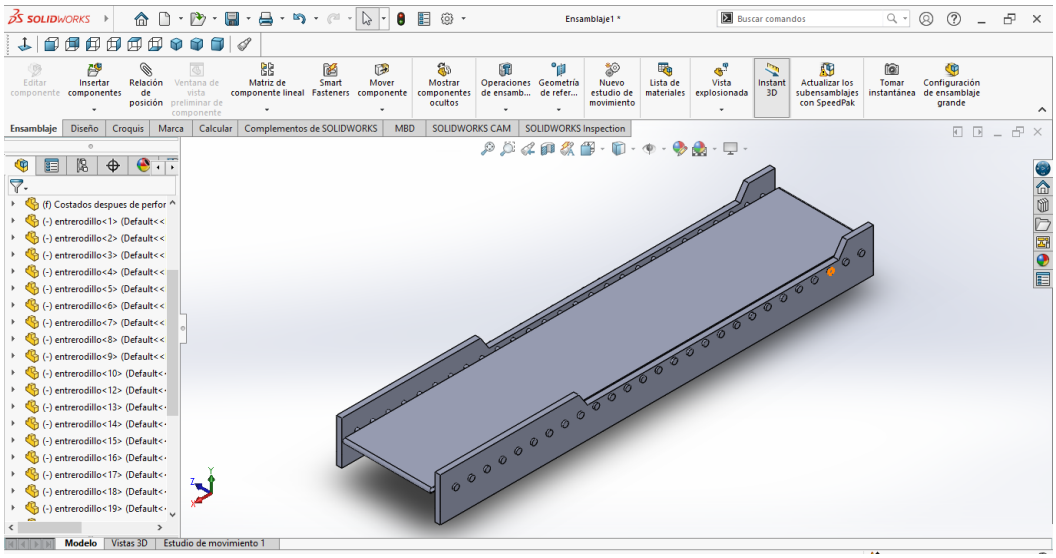

Figura 46. Banda transportadora ensamblada.

Se crea el contenedor de las frutas, se forma de un trapecio recto el cual se extruye para darle forma y con la herramienta de vaciado se crea la cavidad donde se contendrán las frutas.

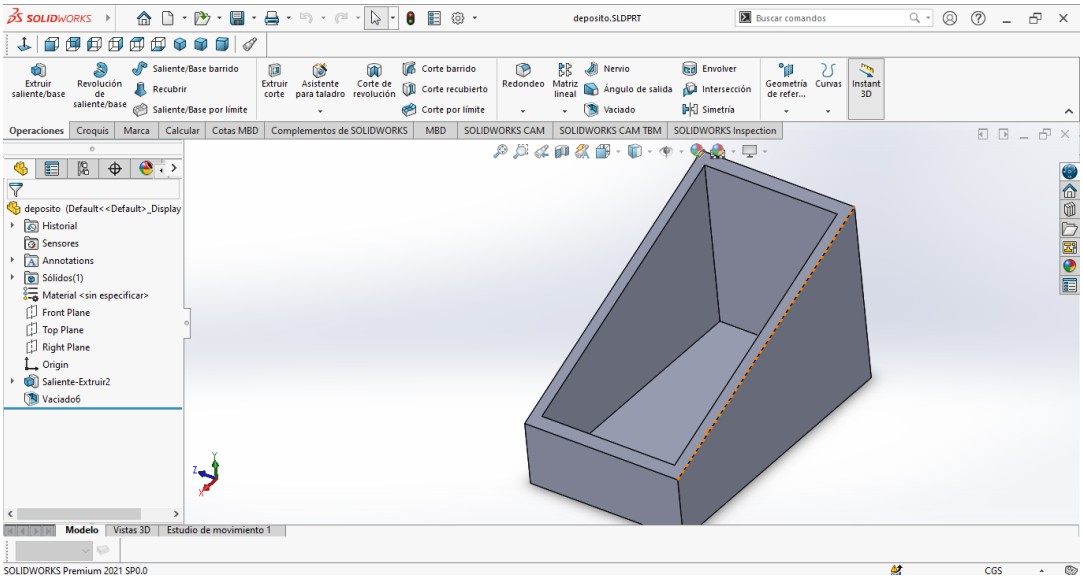

Figura 47. Contenedor de frutas.

Se adiciona la estructura que contiene los piñones la cual va acoplada a un extremo de la banda transportadora, se alinean las estructuras obteniendo una previsualización de los objetos.

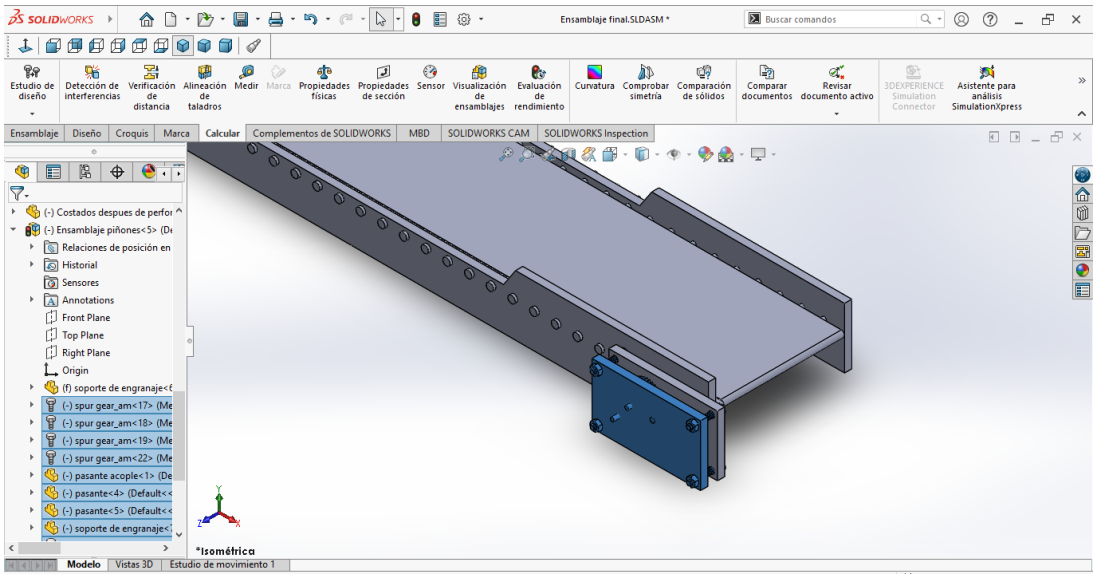

Figura 48. Acoplamiento de engranajes.

Completado el proceso se verifica planos para visualizar que las uniones se mantengan uniformes y no existan errores.

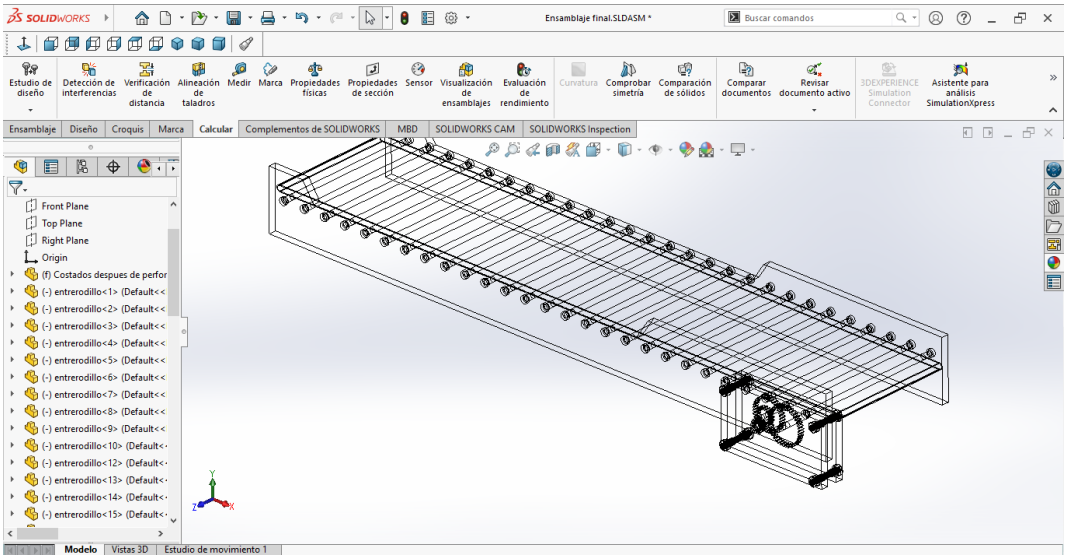

Figura 49. Visión del ensamble en capas.

Se acopla la estructura de los pistones y los contenedores se alinean según la conveniencia de la estructura.

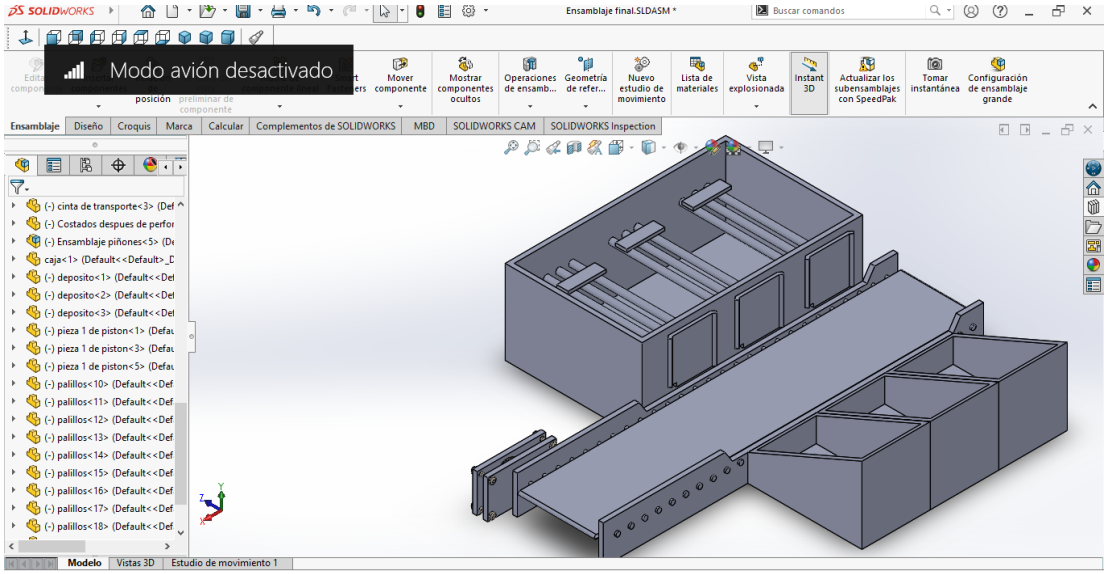

Figura 50. Alineación de los contenedores y pistones.

Se aplica una visualización de las diferentes capas de ensamble a la estructura.

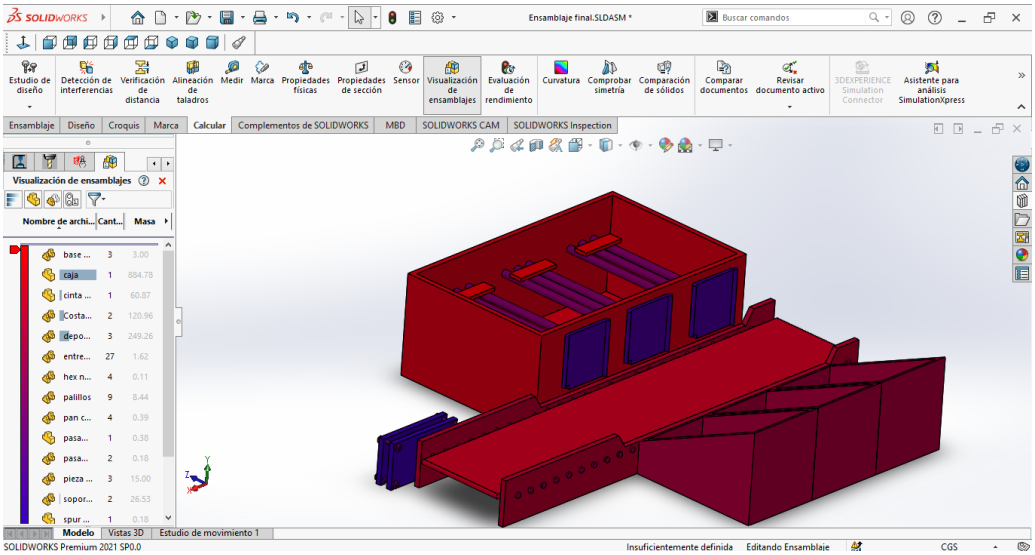

Figura 51. Visualización de ensamblaje.

3.4 Sistema Electrónico

3.4.1 Raspberry Pi 4 B+

Para el desarrollo de este prototipo, se comienza por la parte más importante que es el diseño del sistema de control, que como pieza principal se encuentra la Raspberry Pi 4 B+, la cual es la versión con mayor capacidad.

Su función es de control, es decir, le llega todo lo que sensa con respecto a los infrarrojos y la cámara, accede a ella y comienza a ejecutar los modelos capturando las imágenes que está obteniendo en ese momento. Además, sensa dependiendo de las diferentes condiciones que se apliquen en la programación ya sea por su forma, color o tamaño, ejerce el control en los motores, los cuales encienden la banda, permitiendo que se mueva y también los 3 motores que son los servomotores que sirven para la función de los piñones.

La Raspberry además se comunica de manera serial con el Arduino Nano, la cual permite la transmisión y recepción, ambos se envían mensajes entre ellos. El control hace referencia a que la comunicación con el Arduino permite que se ejecuten adecuadamente los motores.

Tabla 4

Especificaciones técnicas del Raspberry Pi 4 B+

Procesador	SoC Broadcom BCM2711, Cortex-A72 de cuatro núcleos (ARM v8) de 64 bits a 1,5 GHz
Memoria	8GB LPDDR4
Conectividad	• LAN inalámbrica de 2,4 GHz y 5,0 GHz IEEE 802.11b/g/n/ac, Bluetooth 5.0, BLE • Gigabit Ethernet • 2 x puertos USB 3.0 • 2 x puertos USB 2.0

GPIO	Encabezado GPIO estándar de 40 pines (totalmente compatible con versiones anteriores de placas anteriores)
Multimedia	• H.265 (descodificación 4Kp60); • H.264 (descodificación 1080p60, codificación 1080p30); • OpenGL ES, gráficos 3.0
Potencia de entrada	• 5V CC mediante conector USB-C (mínimo 3A1) • 5 V CC a través del encabezado GPIO (mínimo 3A1) • Alimentación a través de Ethernet (PoE) habilitada (requiere PoE HAT por separado)

(Mouser)

Figura. 52. Raspberry Pi 4 B+ y sus componentes. (Rodríguez B. , 2020)

3.4.2 Arduino Nano

En el prototipo se utilizaron dos Arduino Nano, el primero se encarga de la conexión del motor y los sensores, los cuales permiten saber cuándo un objeto está pasando por un lugar, también se les llama detectores de objetos, porque son guiados a detectar obstáculos.

Estos sensores, trabajan enviando información a la Raspberry y ésta a su vez dependiendo de las condiciones, permite enviar mensajes de control al Arduino hacia los motores. Por ejemplo, la Raspberry sería como el jefe y los Arduinos como los empleados, esto se debe a que el Arduino se encarga de informarle a la Raspberry que existe un objeto ahí y la Raspberry afirma la presencia del mismo, por consiguiente, le transmite al Arduino que realice la acción.

El segundo Arduino se encarga de ejecutar las condiciones y las acciones en el encendido de los motores y de los servomotores de los pistones.

Tabla 5

Especificaciones técnicas del Arduino Nano

Microcontrolador	ATmega328P: microcontrolador de la familia AVR de 8 bits
Tensión de funcionamiento	5V
Voltaje de entrada recomendado para pin Vin	7-12V
Pines de entrada analógica	6 (A0-A5)
Pines de E/S digitales	14 (de los cuales 6 proporcionan salida PWM)
Corriente CC en pines de E/S	40mA
Corriente CC en pin de 3,3 V	50mA
Memoria flash	32 KB (2 KB se utilizan para el cargador de arranque)
SRAM	2KB
EEPROM	1K
Frecuencia	16MHz
Comunicación	CII, SPI, USART

(Componentes101, 2021)

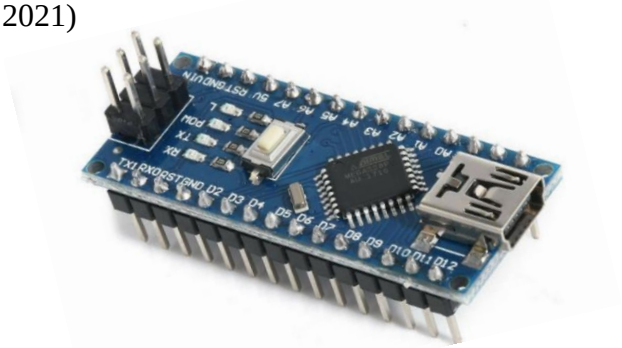

Figura 53. Microcontrolador Arduino Nano. (Komunicate, 2018)

3.4.3 WS 3.5inch RPi LCD

El LCD es un dispositivo de salida, cuya función permite observar lo que la Raspberry está viendo en ese momento, reflejando el análisis del objeto y ver como lo está detectando.

Tabla 6

Especificaciones técnicas de WS 3.5inch RPi LCD.

Tipo de LCD	TFT
Transmisión de señal	Señal SPI de alta velocidad de 125MHz, efecto de visualización claro y estable
Compatibilidad	Admite cualquier versión de Raspberry Pi
Controladores	Funciona con su propio Raspbian/Ubuntu/Kali/Retropie
Resolución de hardware	480x320

(Pishop)

Figura 54. RPi LCD.

3.4.4 Cámara Raspberry PI Versión 2 de 8 Megapíxeles

Es la encargada de que se permita reconocer los objetos. En la Raspberry, el código de Python se utilizó una librería llamada CV2, la cual da el acceso a usar la cámara en cualquier dispositivo. Cuando ya se encuentra activa, lo que hace es tomar cierta cantidad de fotos por segundo que se van presentando en la pantalla, normalmente las cámaras captan 30 fotogramas por segundo, la que se utilizó para el prototipo es de 50 fotogramas por segundo.

De las 50 fotos, se eligen cerca de 20 fotos de lo que se obtiene en ese momento y se procede a analizarlas, ya que existe una programación previa en donde se indica que clase de fruta se va a usar, en este caso las manzanas y las peras.

Tabla 7

Especificaciones técnicas Raspberry PI Versión 2 de 8 Megapíxeles.

Características	Lente de enfoque fijo a bordo
Resolución	Sensor de resolución nativa de 8 megapíxeles con capacidad para imágenes estáticas de 3280 x 2464 píxeles
Video	vídeo de 1080p30, 720p60 y 640x480p90
Tamaño	25 mm x 23 mm x 9 mm
Peso	3 g
Conexión	Se conecta a la placa Raspberry Pi a través de un cable plano corto (suministrado)
Compatibilidad	Camera v2 es compatible con la última versión de Raspbian, el sistema operativo preferido de Raspberry Pi

Fuente: (ThePiHut)

Figura 55. Cámara de resolución nativa de 8 megapíxeles. (Unit Electronics, 2020)

3.4.5 Tipos de motores

En el prototipo, se usaron dos tipos de motor, el primer tipo es un motor de 12 voltios con polaridad invertida, esto ayuda a que el motor cambie de dirección. Se realizó un sistema de engranajes para que reduzca la velocidad angular y aumentar el torque, ya que da fuerza en el movimiento de la banda transportadora.

El segundo tipo de motor es un servomotor, los cuales tienen un torque bien definido, que puede cargar o mover hasta cierta cantidad de kilogramos. El MG996s que se usó, según el datasheet puede mover objetos hasta 2.9 kilos.

El servomotor aparte de dar un movimiento o de definir un peso, puede además definir el ángulo de rotación, ya que la mayoría de servomotores no giran 360 grados, sino que giran en 180 grados. Por ello, se utilizó el motor de 12 voltios con polaridad invertida en la banda y los servomotores en los pistones, ya que los pistones son los que se encargan de clasificar las frutas que están en buen estado y las que no están en buen estado.

El motor de 12 voltios es analógico y el servomotor es digital, sus entradas y salidas son digitales, porque también puede enviar información.

Tabla 8

Especificaciones técnicas del servomotor MG996r.

Voltaje de operación	6.0 V - 7.2 DC
Velocidad de operación	0.2 s/60º (4.8 V), 0.16 s/60º (6 V)
Torque detenido	9.4kg/cm (4.8V), 11kg/cm (6V)
Banda muerta	20 µs
Dimensiones compactas	40.6*19.8*42.9 mm
Largo del cable	300mm
Tipo de piñoneria	Piñoneria metálica
Conexión	Conector universal tipo “S” compatible con la mayoría de receptores incluyendo Futaba, JR, GWS, Cirrus, Blue Bird, Blue Arrow, Corona, Berg, Spektrum y Hitec, entre otros.

(Naylamp Mechatronics, s.f.)

Figura 56. Servo tipo Piñoneria metálica (Az-Delivery, 2020)

3.4.6 Sensores

Los sensores que se utilizaron son sensores infrarrojos IR FC-51, los cuales ayudan al proceso lógico de este prototipo. Estos sensores tienen una distancia de 4cm para la medida de la detección de un objeto.

Para la elaboración del prototipo se utilizaron 4 sensores. El primer sensor se encarga del inicio del proceso. Al momento que el sensor detecte un objeto en el rango de sensado envia la información a su controlador para que continúe con la escala de procesos, caso contrario el sistema se mantendrá en reposo.

Los 3 sensores restantes corresponden uno para cada pistón. El segundo sensor conectado con el primer pistón es para la primera fruta, en este caso la manzana, el tercer sensor conectado con el segundo pistón es para la pera y el cuarto sensor conectado al tercer pistón es para las frutas que presentan un mal estado.

Una vez que llega la fruta al pistón correcto, se repite el proceso de comunicación entre el Arduino y la Raspberry, y ésta a la vez da la orden de apagar el motor de la banda.

Tabla 9

Especificaciones técnicas de Sensor infrarrojo IR FC-51.

Número de modelo	FC-51
Ángulo de cobertura	35 °
Voltaje de funcionamiento	3.0V – 6.0V
Rango de detección	2 cm – 30 cm (ajustable con el potenciómetro)
PCB tamaño	3,1 cm (largo) x 1,4 cm (W)
Dimensión total	4,5 cm (L) x 1,4 cm (W), 0.7cm (H)
El nivel de salida de discriminación	Las salidas de nivel lógico bajo cuando se detecta obstáculo
En activo nivel de salida	Salidas nivel lógico alto cuando no se detecta obstáculo
Consumo actual	a 3.3V: ~ 23 mA en 5.0V: ~ 43 mA

(Web-Robótica, 2019)

Figura 57. Sensor y sus componentes. (Esther Lopez, 2020)

3.4.7 Disipadores

Los disipadores son importantes para el proyecto ya que la Raspberry al realizar procesos de control, aumenta el calor dentro de sus componentes, causando un estrangulamiento térmico, por ello, se colocan los disipadores a la Raspberry para mantener una temperatura estable.

En los DC-DC, se ocupan los disipadores ya que se pierde la potencia en calor, por regular el voltaje.

Tabla 10

Especificaciones técnicas de Disipador

Material	Aluminum
Estilo	30pcs
Nombre de la marca	Pastall
Tableros compatibles	Raspberry Pi models A, B, B+, 2, 3, 4 boards

(Amazon)

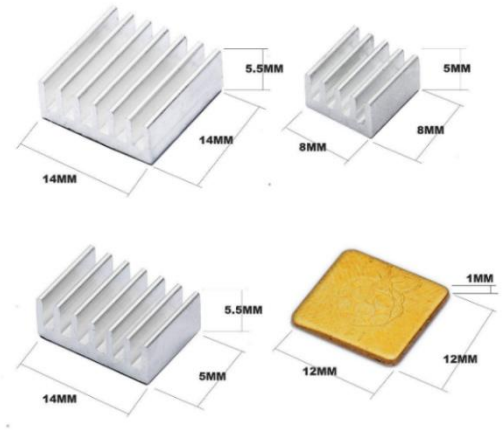

Figura 58. Disipador 30pcs. (Amazon, 2021)

3.4.8 Transistor TIP31C

Este transistor que trabaja con el Arduino para la alimentación del motor y de los sensores.

Tabla 11

Especificaciones técnicas del Transistor TIP31C.

Número de pines	3
Transistor tipo	3A
Corriente máxima de colector	100V
Voltaje máximo Colector-Emisor	1.2V @ 375mA, 3A

Máxima potencia	2W
Temperatura de operación máxima	150°C
Encapsulado	TO-220

(Octopart)

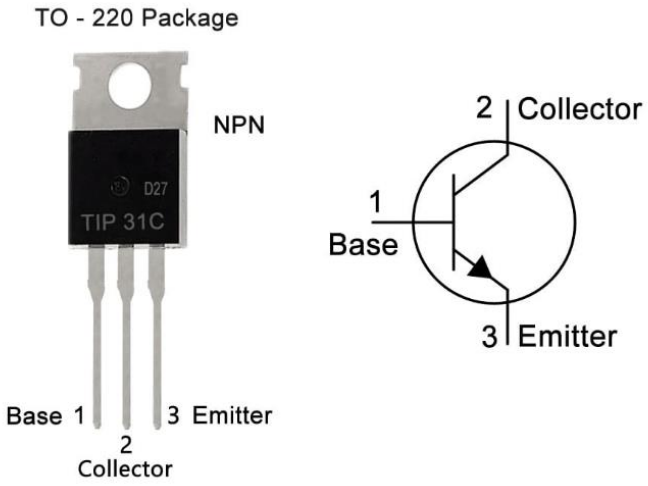

Figura 59. Transistor NPN TIP31C. (Components Info, 2020)

3.4.9 Resistencias

Esta resistencia ayuda a modificar el paso de la corriente del Arduino al transistor.

Tabla 12

Especificaciones técnicas de resistencia.

Resistencia	1K ohmios
Tolerancia	5%
Voltaje	350V
Polarización	Ninguna
Temperatura de funcionamiento	-55C – +155C

(Protosuppplies, 2021)

Figura 60. Resistor 1kΩ. (Electrocomponentes, 2019)

3.4.10 Fuente de poder

Es una fuente de poder de 24v, 14.64 amperios, la cual alimenta a todos los dispositivos electrónicos del proyecto.

Tabla 13

Especificaciones técnicas de la fuente de poder

Potencia	350W
Alimentación	DC24V
Entrada	AC85-265V
Salida	DC24V
Amperios (mA)	14600mA
Interior-exterior	Interior
Protección IP	IP20
Dimensiones del producto	115x215x30mm

Fuente: (LEDBOX, s.f.)

Figura 61. Fuente de poder 24v -14.64ª. (Grupo Velasco, 2018)

3.4.11 Cableado

Para el funcionamiento del prototipo se comienza energizando el sistema con una fuente DC 24v. La Raspberry y los dos Arduinos para comunicarse usan puertos USB, dando inicio al proceso. El primer Arduino se utiliza para activar el motor, el puerto de 5v tiene como función alimentar los sensores y los servomotores como se puede observar en la figura 3.12.

El puerto D2 es la única señal de salida, trabaja con un transistor el cual ayuda con el paso de la corriente para que el motor y la banda transportadora funcione. Los puertos D12, D11, D10, D9 son de entradas, por los cuales el Arduino envía datos de lectura a los sensores infrarrojos.

Los puertos D12, D11, D10 del segundo Arduino, hacen de entrada y salida y se encargan de comunicarse con los servomotores para que roten hasta cierto ángulo. En el cableado para las conexiones del prototipo se utilizan cables tipo UTP RJ45.

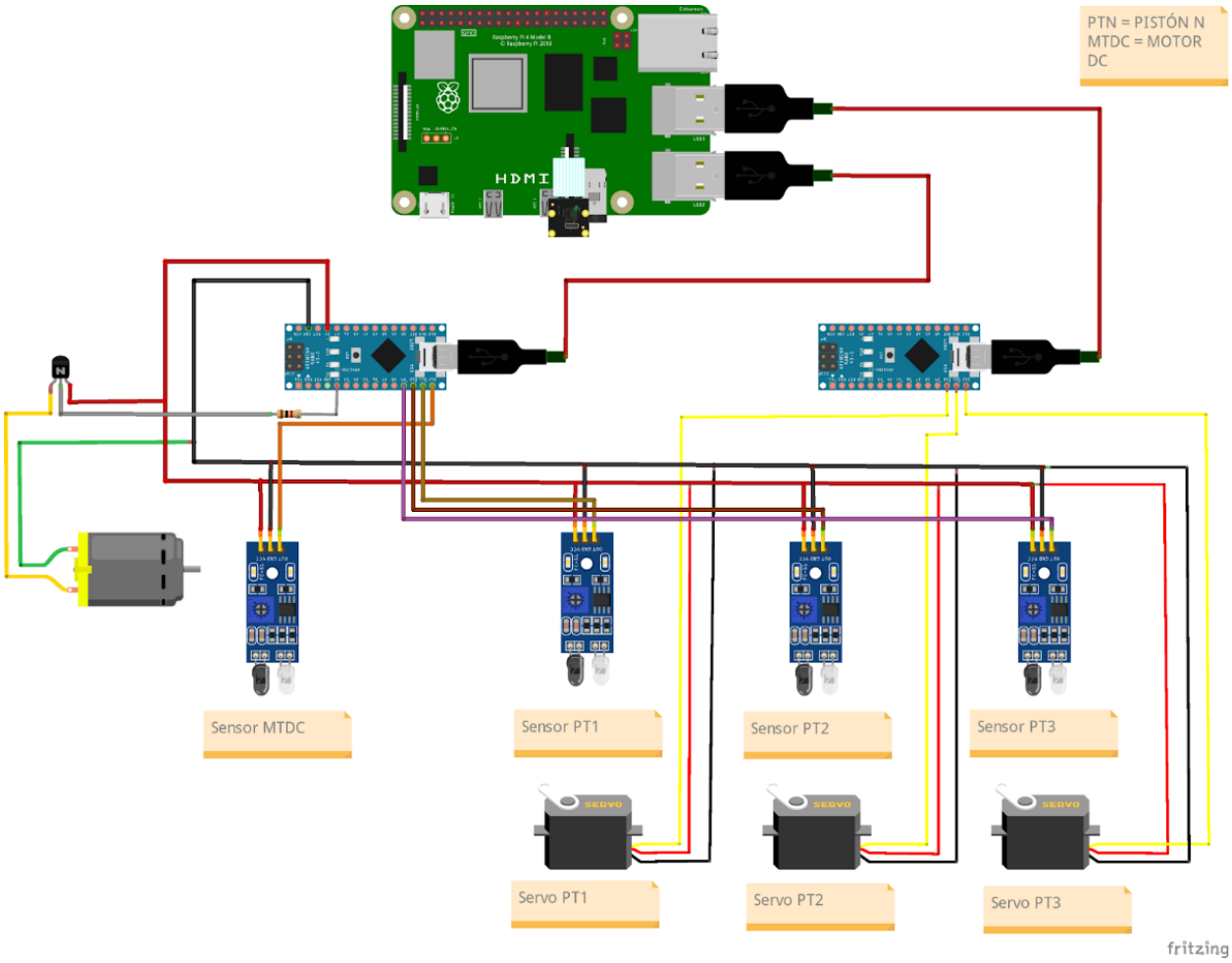

Figura 62. Diagrama del cableado del prototipo

3.5 Programación del sistema

El prototipo requiere un algoritmo que clasifique frutas en buen estado o mal estado, para este proceso se implementó el uso de visión artificial el cual ayuda a identificar y clasificar las frutas. Para el proceso se usó una Raspberry PI 4 B + la cual se comunica vía USB con dos Arduinos.

Una vez que el primer sensor envía la señal al primer Arduino, comienza la comunicación entre la Raspberry- Arduino, se activa el motor de 12 voltios y procede a accionar la banda, transportando la fruta hacia el rango de visión de la cámara, la cual analiza a qué sensor corresponde según la clasificación que le otorgue la red neuronal, si es una manzana se activa el sensor 2 y se detiene la banda, por lo tanto se activa el pistón 1, si es una pera se activa el sensor 3 deteniendo la banda y activando el pistón 2, si es detectada como una fruta en mal estado o que no esté en el rango de clasificación se activa el sensor 4 y se activa el pistón 3, el cual ayuda a la clasificación de aquellas frutas que se encuentran en mal estado, en el caso de que el sistema no las identifique como frutas hace que se active el pistón 3.

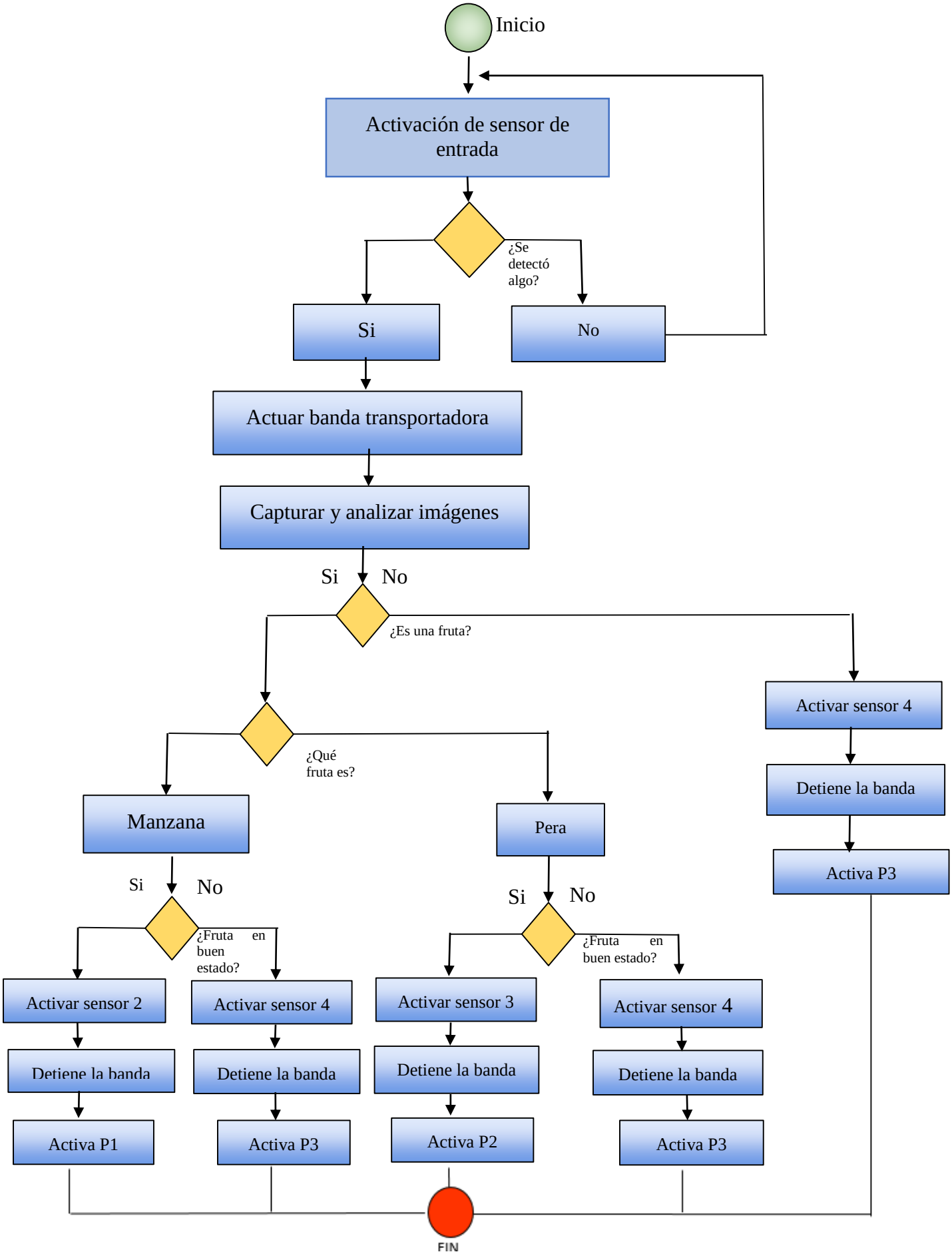

Figura 63. Diagrama de flujo del funcionamiento.

3.5.1 Programación de la tarjeta de control

Python

La programación en Python se inicia con un main script para ejecutar la rutina de detección de objetos con motores de control y servos.

Se utiliza "Sys" para proveer acceso a funciones y objetos por del intérprete.del sistema, en caso de no cumplir las condiciones de ejecución.

En "Time" se utiliza para calcular la cantidad de FPS que se está analizando.

"CV2" que es parte de OpenCV, se utilizó para capturar imágenes de la cámara, para así poder presentar texto y cuadros que permitan reconocer el objeto identificado.

El object_detector es un módulo que se utilizó para ejecutar la detección de objetos con un modelo TensorFlow Lite.

La función de "Utils “ en esta programación es la de graficar los cuadros dentro de la imagen obtenida de la cámara.

"onMotor" se encarga de mover los motores según las condiciones de la red neuronal y es enviado por el Arduino

"inoMotor " se lo utiliza para un objeto serial responsable de enviar el mensaje al sensor/motor de Arduino. "inoServMo" es objeto serial responsable de enviar mensajes a Arduino Servos.

```
"""Main script to run the object detection routine with control motors an servos."""
import argparse
import sys
import time

import cv2
from object_detector import ObjectDetector
from object_detector import ObjectDetectorOptions
import utils

#ino control
import serial,time,random,threading

servmotor = ["PT1","PT2","PT3"]
aleatory = "PT3"

def onMotors(answer, inoMotor,inoServMo):
  """It is responsible for moving the motors according to the conditions of the neural
network
  Args:
    answer: msg sent by arduino
    inoMotor: Serial object responsible for send message to arduino sensor/motor
    inoServMo: Serial object responsible for send message to arduino Servos
  """
```

Figura 64. Programación en Python.

Se programa el nombre del modelo de detección de objetos TFLite, la camera_id que es la identificación de la cámara que se pasa a OpenCV, el ancho del marco capturado desde la cámara, la altura del cuadro capturado desde la cámara y el num_threads que es el número de subprocesos de la CPU para ejecutar el modelo.

```
# Start capturing video input from the camera
cap = cv2.VideoCapture(camera_id)
cap.set(cv2.CAP_PROP_FRAME_WIDTH, width)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, height)

# Visualization parameters
row_size = 20  # pixels
left_margin = 24  # pixels
text_color = (0, 0, 255)  # red
font_size = 1
font_thickness = 1
fps_avg_frame_count = 10

# Initialize the object detection model
options = ObjectDetectorOptions(
    num_threads=num_threads,
    score_threshold=0.3,
    max_results=10,
    #label_allow_list=['apple']
    )
detector = ObjectDetector(model_path=model, options=options)

#Exec com with arduinos

thread = threading.Thread(target=execComunicationWithIno)
thread.start()

# Continuously capture images from the camera and run inference
while cap.isOpened():
  success, image = cap.read()
  if not success:
    sys.exit(
        'ERROR: Unable to read from webcam. Please verify your webcam
settings.'
```

Figura 65. Programación sobre la detección de objetos.

Se usan dos clases creadas en object_detector, las cuales tienen la lógica para usar el modelo ya entrenado para reconocer objetos.

Se utiliza "Json" para transformar los archivos de modelado, a un objeto Json para manipularlo fácilmente.

Se usó "Numpy" para los análisis matemáticos de matrices, necesario para usar las funciones de Tensorflow.

"TFLite" es un Tensorflow con las capacidades necesarias para importar, exportar y entrenar modelos de AI.

```
import json
import platform
from typing import List, NamedTuple

import cv2
import numpy as np
from tflite_support import metadata

# pylint: disable=g-import-not-at-top
try:
  # Import TFLite interpreter from tflite_runtime package if it's
available.
  from tflite_runtime.interpreter import Interpreter
  from tflite_runtime.interpreter import load_delegate
except ImportError:
  # If not, fallback to use the TFLite interpreter from the full TF
package.
  import tensorflow as tf

  Interpreter = tf.lite.Interpreter
  load_delegate = tf.lite.experimental.load_delegate

# pylint: enable=g-import-not-at-top


class ObjectDetectorOptions(NamedTuple):
  """A config to initialize an object detector."""

  enable_edgetpu: bool = False
  """Enable the model to run on EdgeTPU."""

  label_allow_list: List[str] = None
  """The optional allow list of labels."""
```

Figura 66. Programación de las clases para reconocer los objetos.

Arduino IDE

Para el primer Arduino se programa las variables y constantes del prototipo, como se observa en la figura 67.

```
const int sensorEntrada = 12;
const int sensorPiston1 = 11;
const int sensorPiston2 = 10;
const int sensorPiston3 = 9;
const int motor12V=2;

String msgPi;
```

Figura 67. Programación de variables en Arduino IDE.

En el setup se define en qué frecuencia se maneja el puerto serial y como se va a trabajar con los pines, es decir en donde las entradas y salidas de los sensores están ubicados, como se observa en la figura 68.

En la programación del puerto serial (readSerialPort();)se programa la comunicación de Raspberry- Arduino, se obtiene lo que la Raspberry envía definiéndolo en una variable llamada msgPi.

El serial.println ayuda en enviar los datos para que se cumpla la función de que la Raspberry analice los objetos y se cumpla el proceso.

La señal LOW indica que existe un objeto en la banda y la señal HIGH cuando no se detecta la presencia de un objeto.

```
  pinMode(sensorEntrada , INPUT);  //definir pin como entrada
  pinMode(sensorPiston1 , INPUT);  //definir pin como entrada
  pinMode(sensorPiston2 , INPUT);  //definir pin como entrada
  pinMode(sensorPiston3 , INPUT);  //definir pin como entrada
  pinMode(motor12V,OUTPUT);
}

void loop() {

    int valorEntrada,valorPiston1, valorPiston2, valorPiston3 = 0;

    valorEntrada = digitalRead(sensorEntrada);
    valorPiston1 = digitalRead(sensorPiston1);
    valorPiston2 = digitalRead(sensorPiston2);
    valorPiston3 = digitalRead(sensorPiston3);

    readSerialPort();
    if (valorEntrada == LOW) {
        Serial.println("IN");
    }
    else if(valorPiston1 == LOW){
      Serial.println("PT1");
    }
    else if(valorPiston2 == LOW){
      Serial.println("PT2");
    }
    else if(valorPiston3 == LOW){
      Serial.println("PT3");
    }
```

Figura 68. Programación de los procesos de los sensores en Arduino IDE

En el segundo Arduino se programan las variables y constantes de la conexión de los servomotores. La librería <Servo.h> es la encargada de controlar los servomotores, los cuales se encargan de ejecutar acciones. En esta parte de la programación se enlaza el pin digital con la clase servo, como se puede observar en la figura 69.

```
Servo servo1;
Servo servo2;
Servo servo3;


const int delayServ = 15;
const int servoPin1 = 12;
const int servoPin2 = 11;
const int servoPin3 = 10;

String msgPi;
```

Figura 69. Programación de variables de los servomotores en Arduino IDE.

Como se observa la programación en la figura 70, en el readSerialPort(); se realiza la lectura del mensaje enviado por la Raspberry y se activa la ejecución de los servos para mover el pistón ida y regreso.

Cuando el PT1 ON detecte la manzana, se activa servo1, PT2 ON detecta la pera y se activa servo2 y PT3 ON si detecta frutas en mal estado y objetos desconocidos, activa servo3.

```
void setup() {
  Serial.begin(9600);
  servo1.attach(servoPin1);
  servo2.attach(servoPin2);
  servo3.attach(servoPin3);
}

void loop() {

  readSerialPort();
  |
  if(msgPi == "PT1 ON"){
    moveDegree(servo1);
  }
  else if(msgPi == "PT2 ON"){
    moveDegree(servo2);
  }
  else if(msgPi == "PT3 ON"){
    moveDegree(servo3);
  }
  delay(100);
}
```

Figura 70. Programación de proceso de los servomotores

3.5.2 Red neuronal

En las redes neuronales se busca tener la mayor exactitud posible de la predicción de la tarea que se le asigna, para lograrlo se utiliza un modelo de red neuronal pre entrenado, que es una red que previamente se ejecutó entrenando con un gran conjunto de datos, generalmente con una asignación específica.

Se ejecuta un aprendizaje de transferencia basado en un conjunto de datos generales con los que se entrenan, se aprovechan estos mapas de características ya armados y probados en grandes conjuntos de datos sin tener que comenzar un modelo desde cero.

Se usan las representaciones aprendidas por una red anterior para extraer características significativas de nuevas muestras de las cuales se ejecuta en la nueva red . Se agrega un nuevo clasificador, el cual se entrena desde cero, encima del modelo previamente entrenado para que pueda reutilizar los mapas de características aprendidos previamente para el conjunto de datos. (Tensorflow)

3.5.3 Creación del dataset

Al crear un modelo de detección de objetos se necesita imágenes las cuales deben estar etiquetadas, las etiquetas deben ser correspondientes a las clases que se desee agregar para realizar este trabajo se cuenta con diversos programas para el etiquetado de imágenes, en este caso se utiliza Labellmg.

Inicialmente se deben crear dos carpetas las cuales contendrán las imágenes etiquetadas, la primera se la nombra como train esta carpeta se utiliza para entrenar el modelo, la segunda tendrá el nombre de validation la cual como su nombre lo indica cumplirá la parte de validar el modelo, es importante no repetir las imágenes en ninguna de las dos carpetas para tener un óptimo entrenamiento.

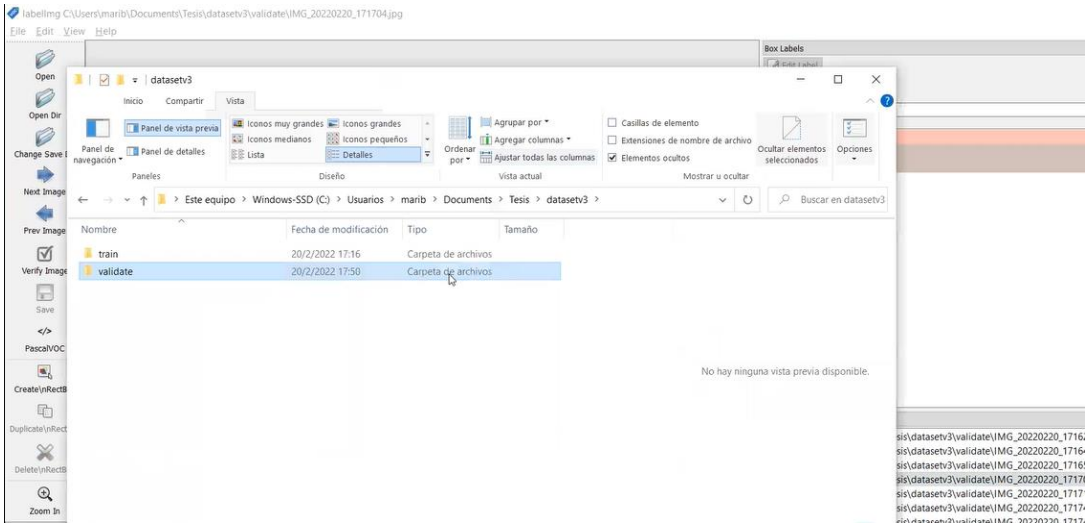

Figura 71. Formato de archivo.

Se coloca la imagen tomada con antelación de las frutas a etiquetar, en este caso son tres peras las cuales van a tener distintas etiquetas, una en buen estado y las otras en mal estado.

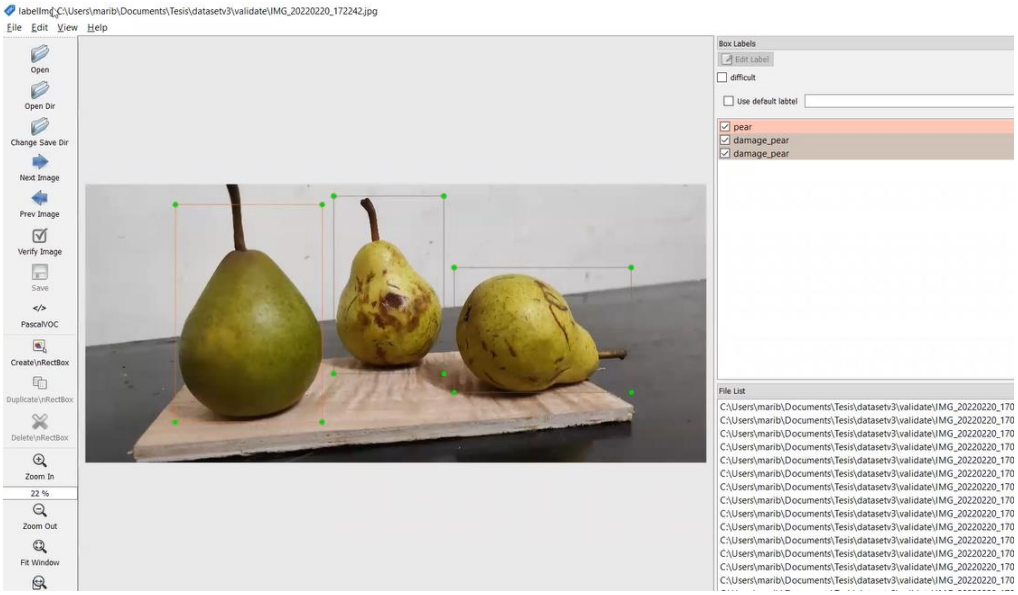

Figura 72. Etiquetado de Peras variado.

Se procede a etiquetar una por una las frutas de la imagen, denominando a las peras en mal estado como “damage_pear”.

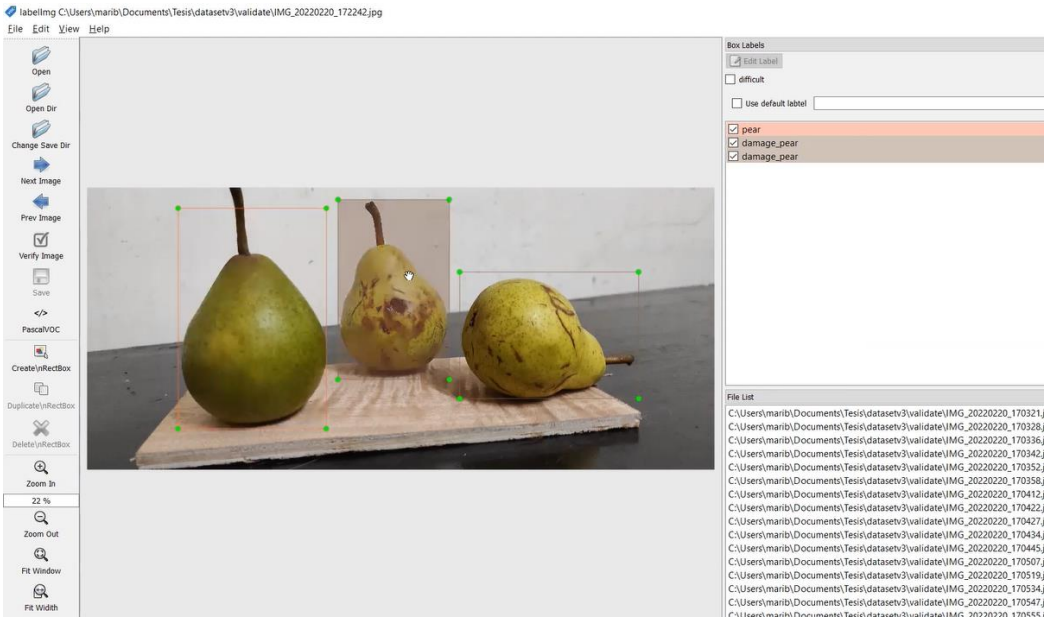

Figura 73. Etiquetado de Pera dañada.

Se etiqueta las frutas en distintas posiciones, lo cual brinda una mayor variedad de imágenes al modelo.

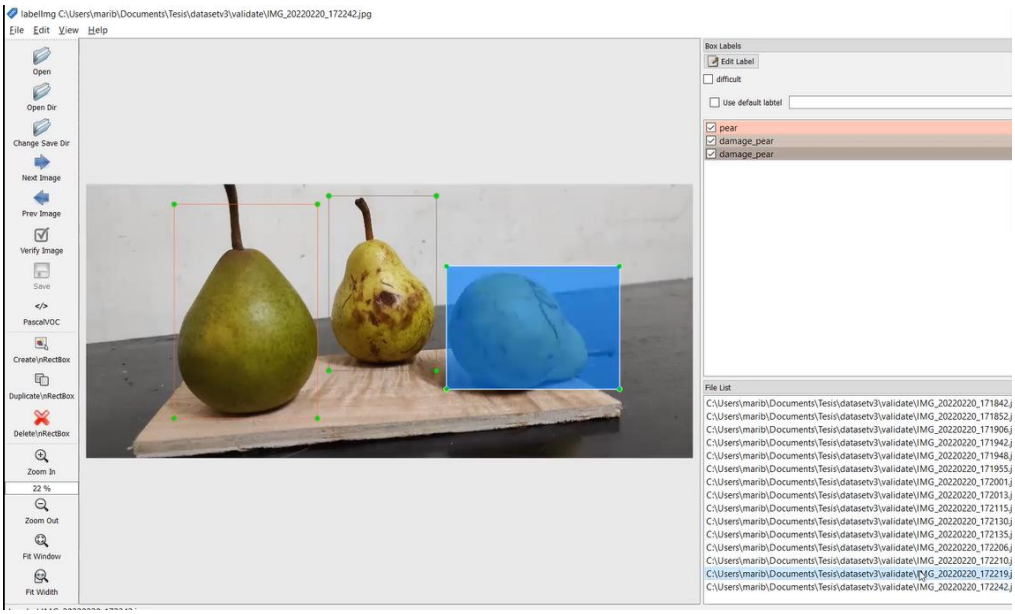

Figura 74. Etiquetado pera horizontal.

Como se observa en la figura 75, la etiqueta en una pera que se encuentra en buen estado se la denomina como “pear”.

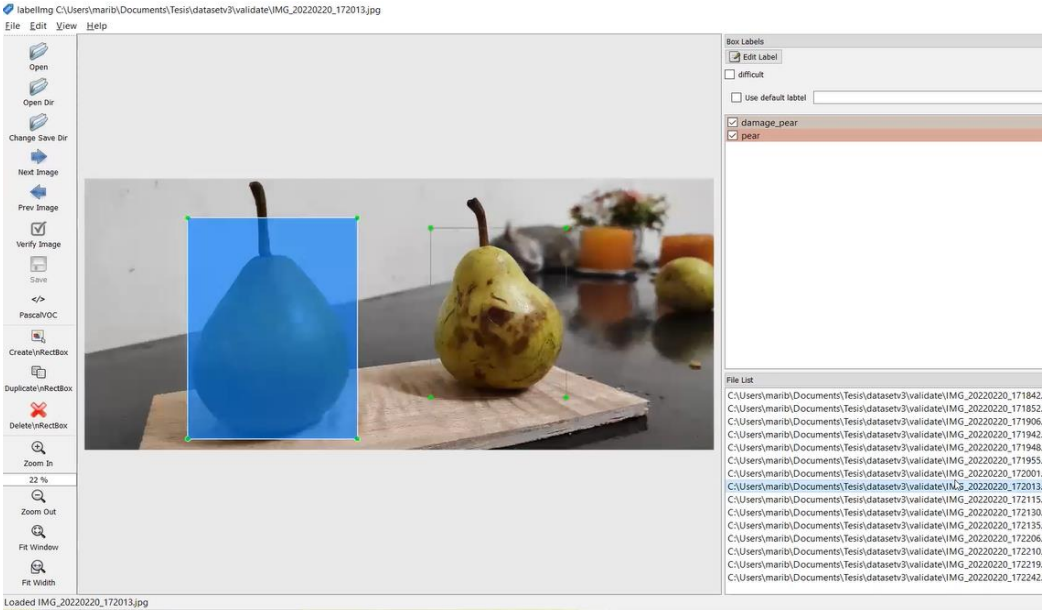

Figura 75. Etiquetado de pera en buen estado.

Se coloca una manzana en mal estado, etiquetando como “damage apple”.

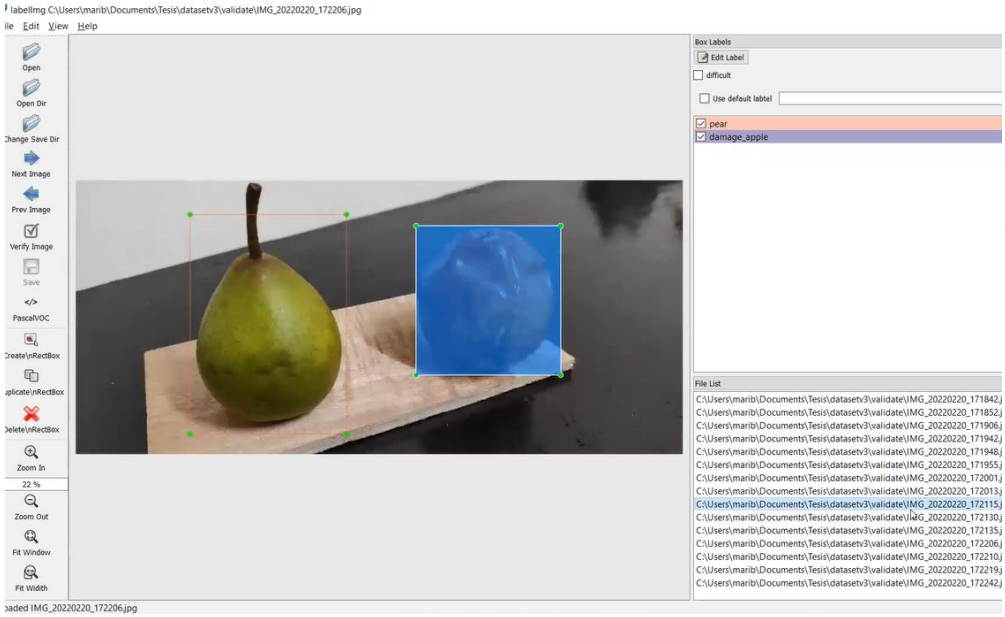

Figura 76. Etiquetado de manzana en mal estado.

Se coloca una manzana en buen estado y se la etiqueta como “apple”

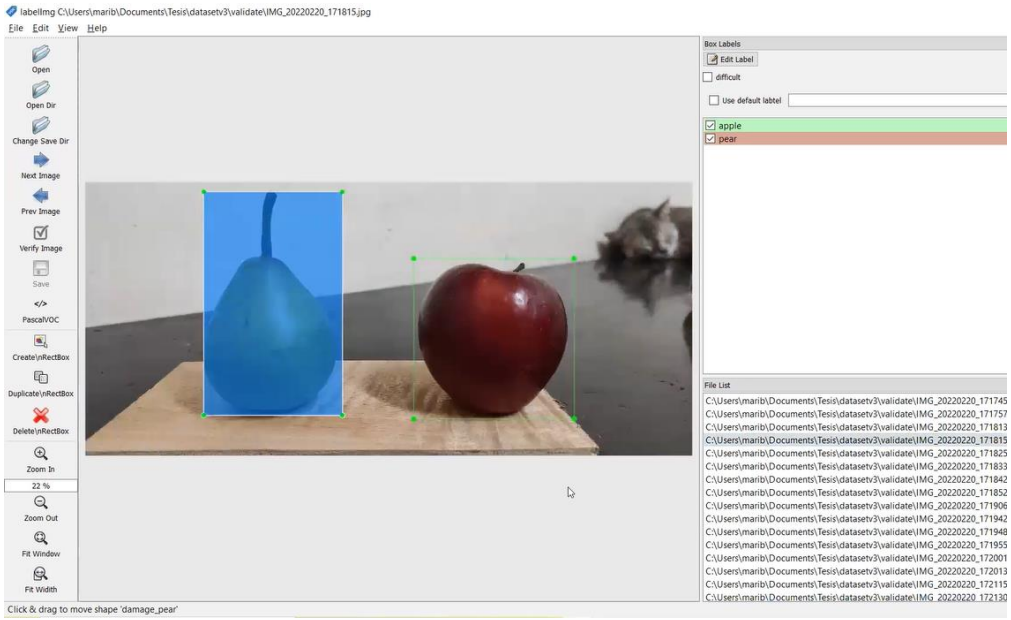

Figura 77. Etiquetado pera y manzana.

Se agregan más peras en diferentes posiciones para expandir la variedad de imágenes y se procede a etiquetar con las clases correspondientes, en este caso “pera”.

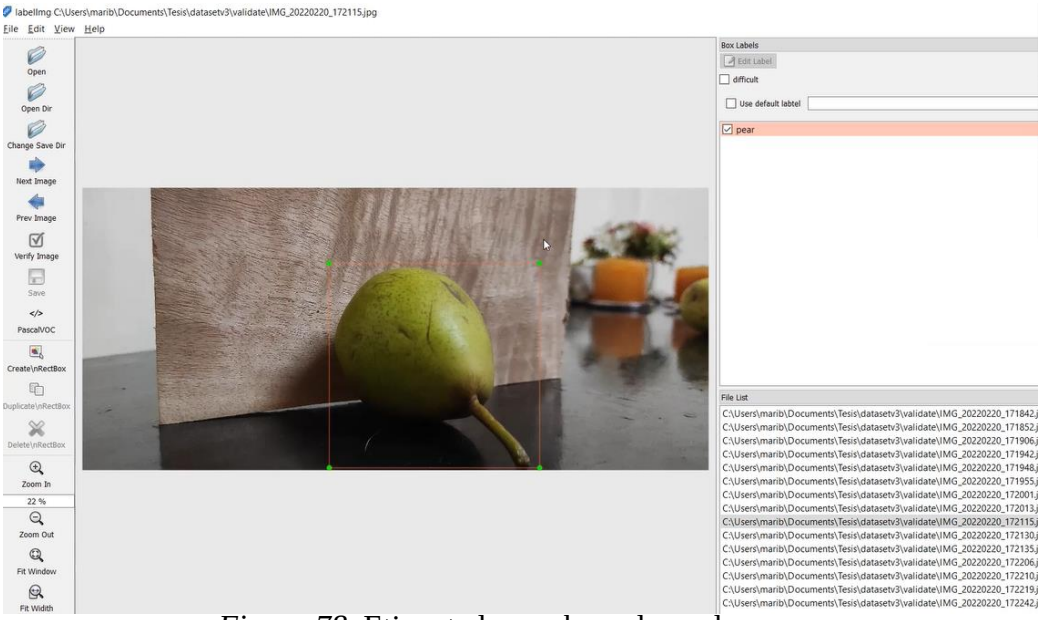

Figura 78. Etiquetado con base de madera.

Se crea un respaldo en la nube con las carpetas creadas, estas carpetas brindaran soporte en el entrenamiento.

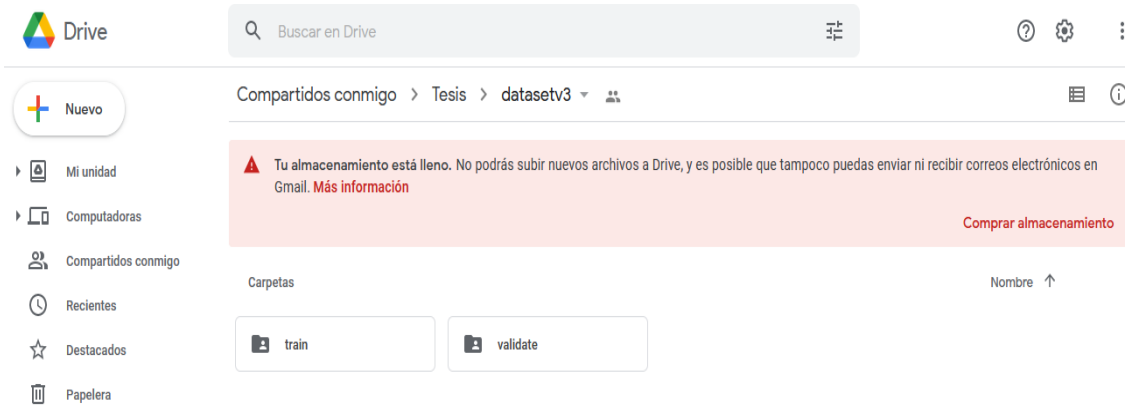

Figura 79. Respaldo de Dataset.

Al crear el Dataset se toman en cuenta las frutas que están expuestas a los diferentes contaminantes que puedan afectarles, ya que esto es parte fundamental del sistema de clasificación, tanto en el entrenamiento como en la validación de la red neuronal.

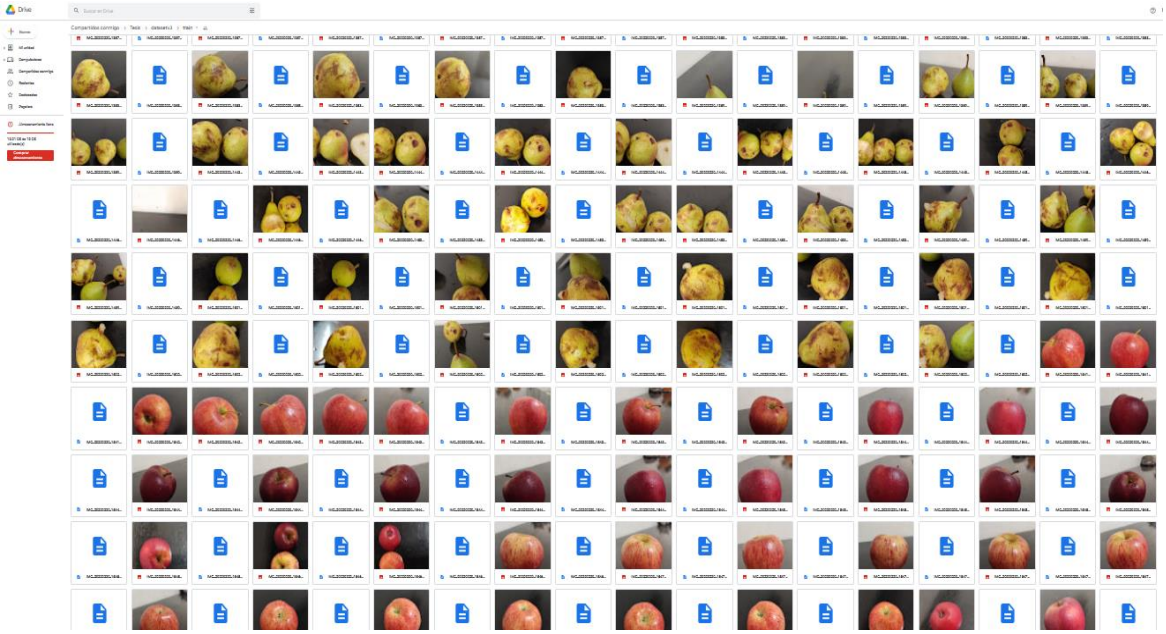

Figura 80. Carpeta Train.

Imágenes del etiquetado para la validación.

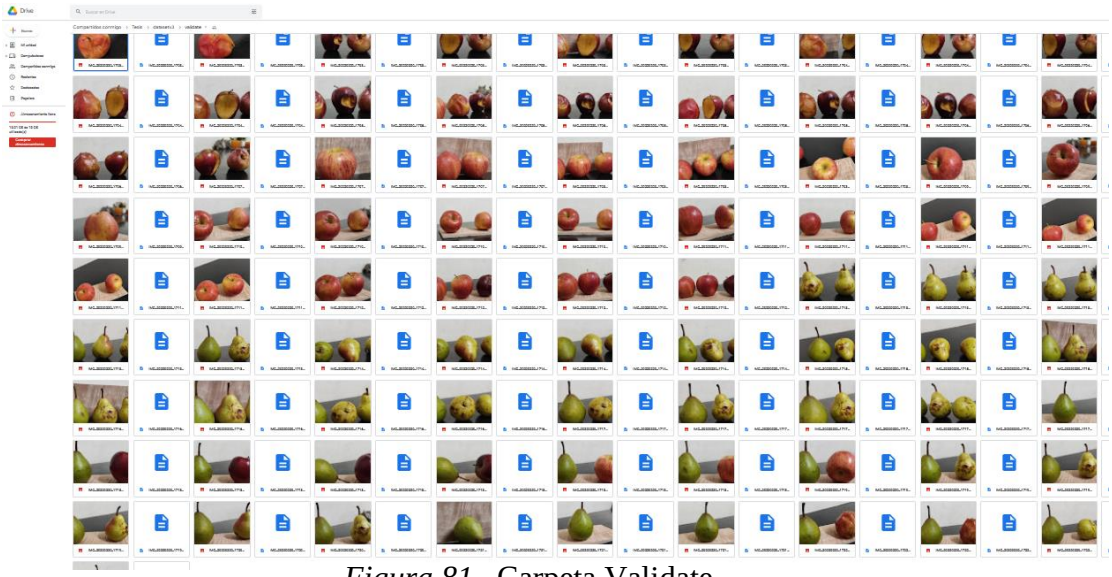

Figura 81. Carpeta Validate.

3.5.4 Entrenamiento de la red neuronal

Una de las partes más importantes de la red neuronal es el entrenamiento, ya que sin un buen entrenamiento no se tiene un buen margen de detección en los objetos, una computadora puede hacerlo, pero el tiempo que le tomaría a una computadora de características medias pueden ser horas o días dependiendo de qué tan grande sea la red.

Buscando el mayor rendimiento en el menor tiempo posible se utilizó Google Colab, el cual brinda una mayor eficiencia ya que se cuenta con la GPU de los servidores de Google y la facilidad de acceder al código desde cualquier sitio con conexión a internet, sin contar con que posee las interfaces ya instaladas para poder entrenar los modelos.

Se inicia desde el entorno de Google Drive creando un archivo nuevo, en el caso de no tenerlo instalado se procede a instalarlo.

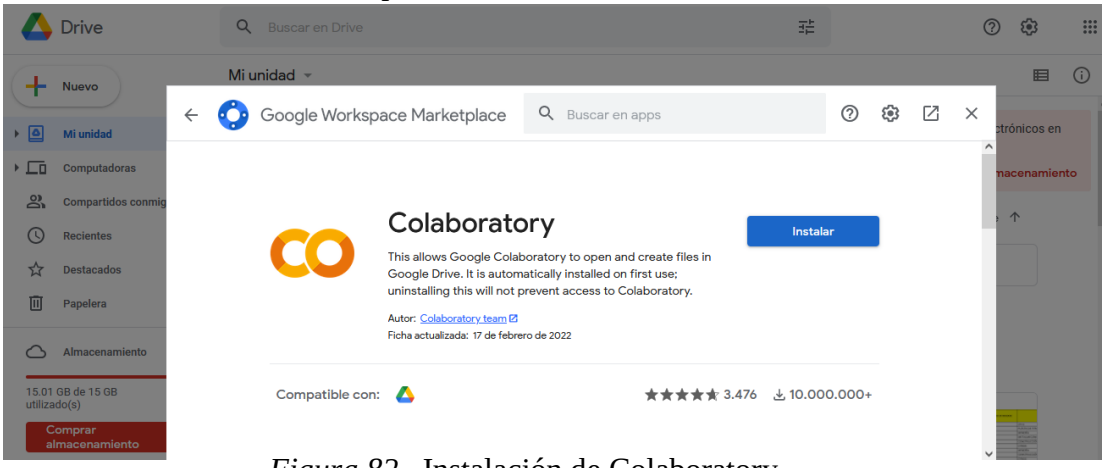

Figura 82. Instalación de Colaboratory.

Se inicia instalado TensorFlow Lite el cual brinda un mejor rendimiento para el procesador de la Raspberry.

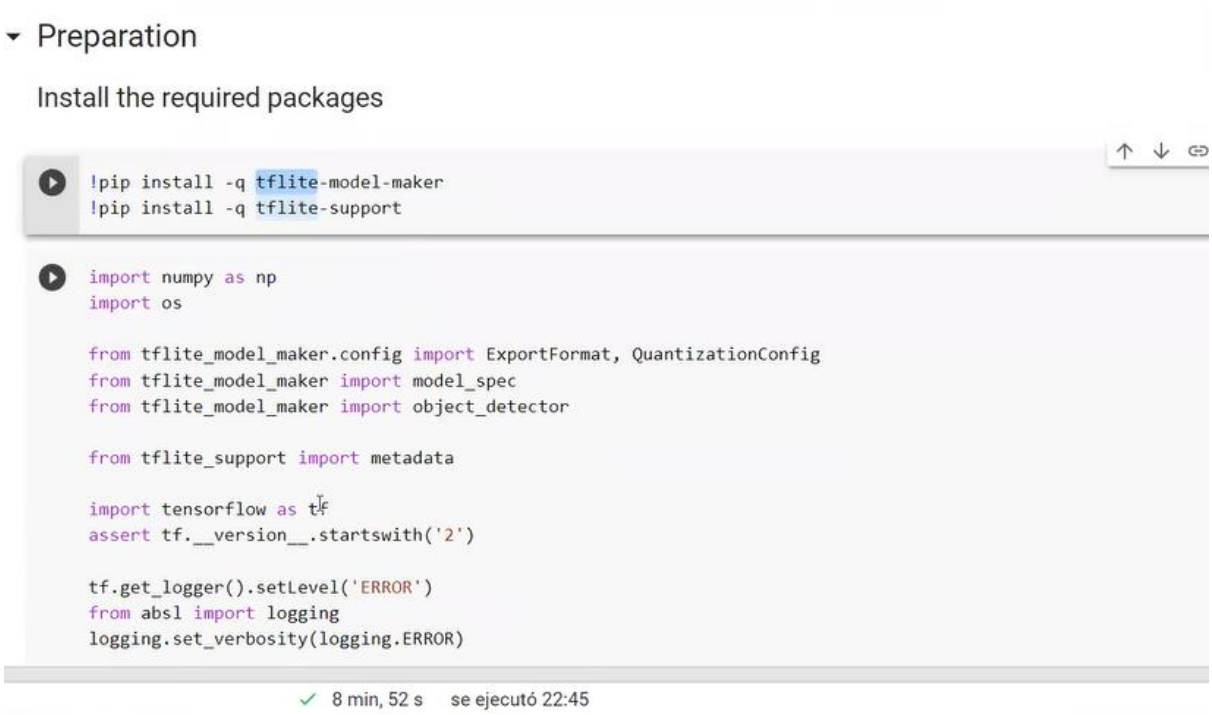

Figura 83. Instalación de TensorFlow Lite.

Se procede a cargar el dataset, escribiendo la dirección donde está ubicada, en este caso el Drive. Se colocan las clases que se van a entrenar y validar.

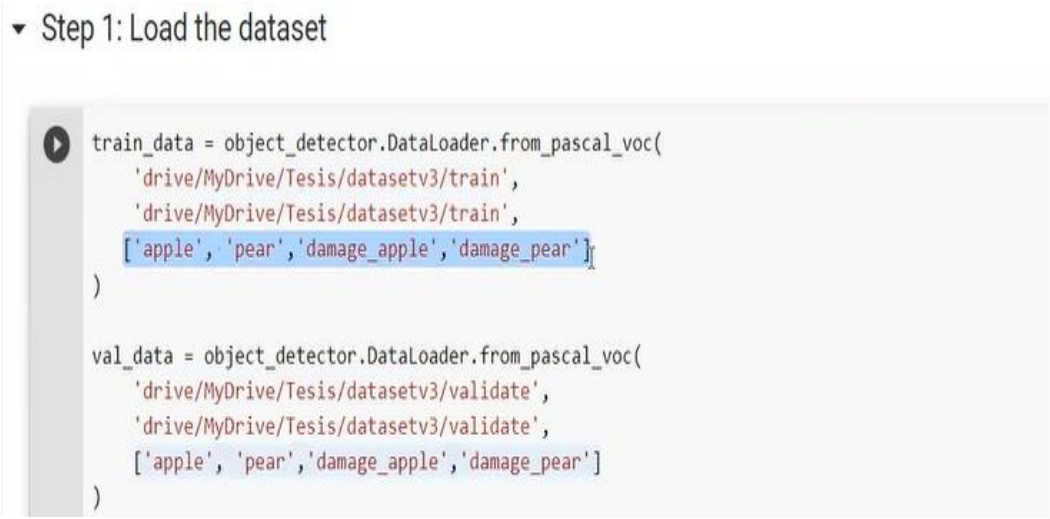

Figura 84. Direcciones del dataset.

Se selecciona qué modelo de arquitectura se va a usar para la red neuronal.

Figura 85. Selección de arquitectura.

Se seleccionan los parámetros que se consideren más adecuados para el entrenamiento de la red.

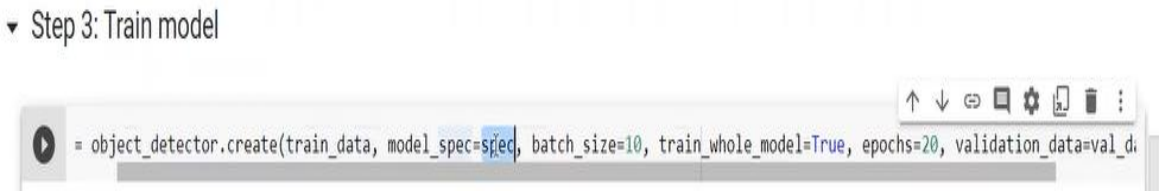

Figura 86. Parámetros de entrenamiento de la red.

4. PRUEBAS Y RESULTADOS

4.1 Elaboración del sistema mecánico

Una vez diseñada la estructura física del prototipo en el programa Solidworks, se procede a realizar la construcción de la banda transportadora, la cual consta de dos piezas laterales de Plywood debido a que este material brinda las propiedades de ser moldeable y ligero para las estructuras, las cuales están unidas por varas de madera horizontales. Se cuenta con una cinta transportadora fabricada de lona, ya que cuenta con la fricción necesaria y ayuda con la tensión suficiente para transportar objetos.

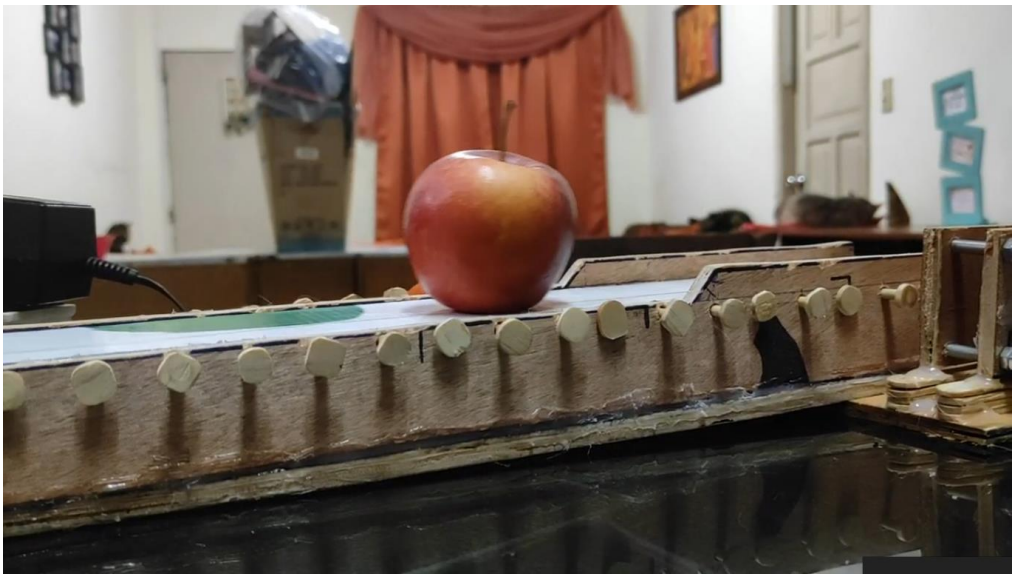

Figura 87. Construcción de banda transportadora.

Se elabora el sistema de fuerza que mantiene en orden a los engranajes, consta de dos placas de Plywood unidas por pernos pasantes que contienen el juego de engranajes que dan movimiento a la banda transportadora.

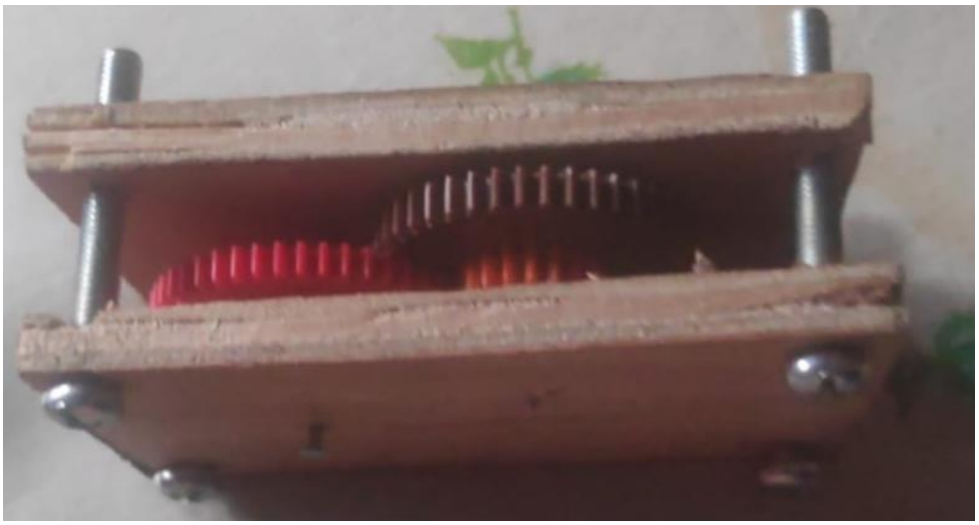

Figura 88. Contenedor de engranajes.

Se acopla los engranajes al motor y a su vez a la banda transportadora, ya que ayuda al movimiento de las frutas, se realizan las primeras pruebas y se verifica si la fuerza del engranaje es suficiente para el peso de cada fruta transportada.

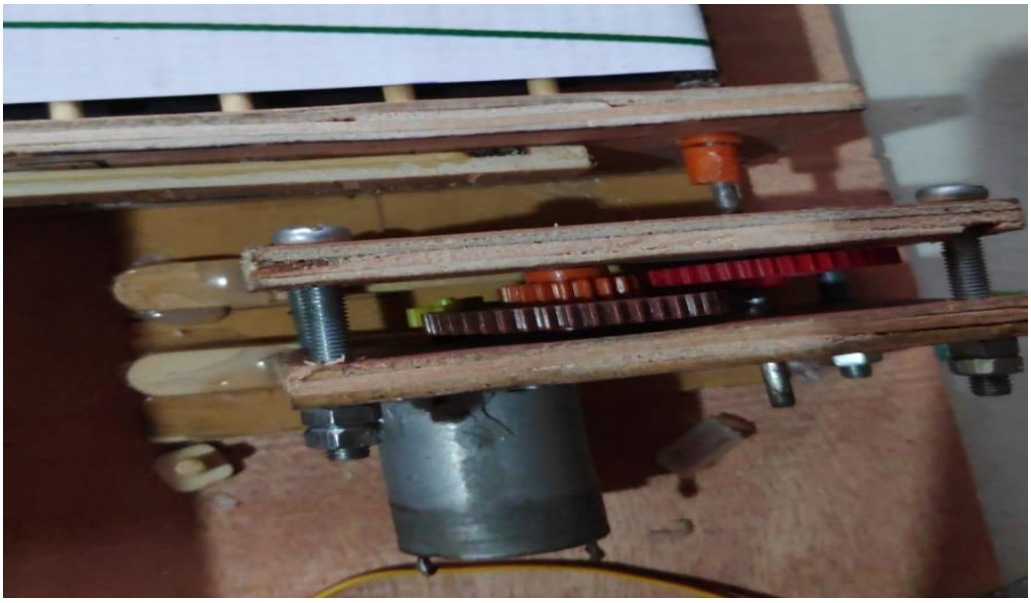

Figura 89. Unión de motor.

Se realizan las pruebas de funcionamiento de la banda transportadora, está elaborada de rodillos debido a que su bajo manejo de fricción permite una mejor transportación de la fruta para ser clasificada. Se verifico que el soporte de la banda es resistente, ya que fue diseñada para un peso de 200 a 400g.

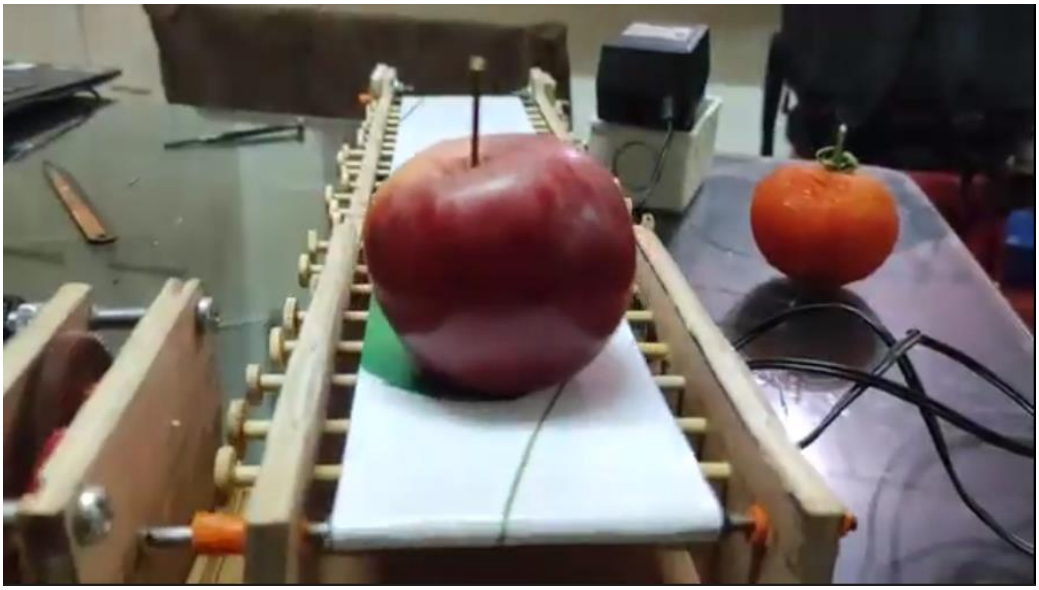

Figura 90. Prueba de banda transportadora.

Se construyó el contenedor de los pistones que trabajan con los sensores para realizar el sensado y clasificar las frutas, está ubicado a un costado de la banda transportadora. Está elaborados con material de Plywood de forma cuadrada debido a que esto permite tener más espacio para su accionamiento.

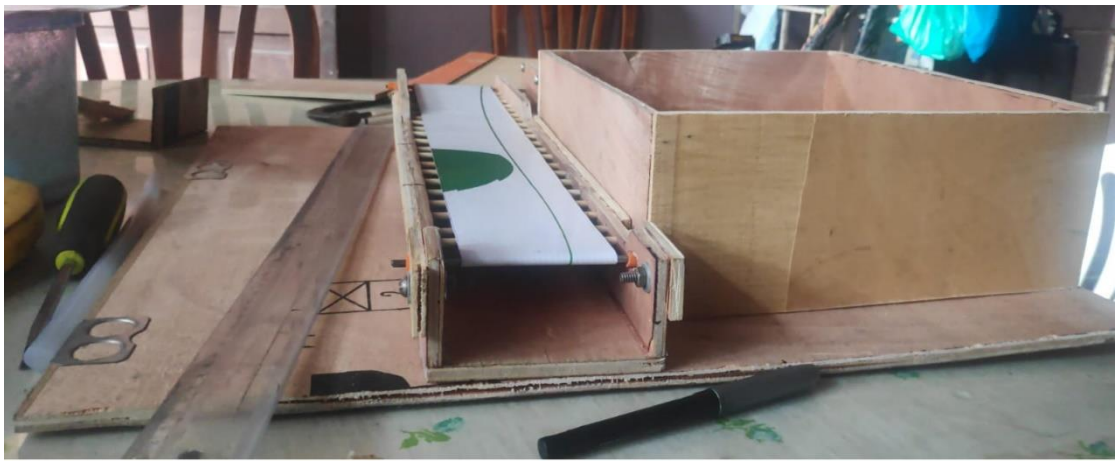

Figura 91. Construcción de contenedor de pistones.

La estructura de los pistones se realiza con varas de madera ya que estas son moldeables y ayudan a que el funcionamiento sea más rápido, ya que su peso es ligero. Se adhieren al contenedor y se realizan las conexiones entre los sensores y servomotores.

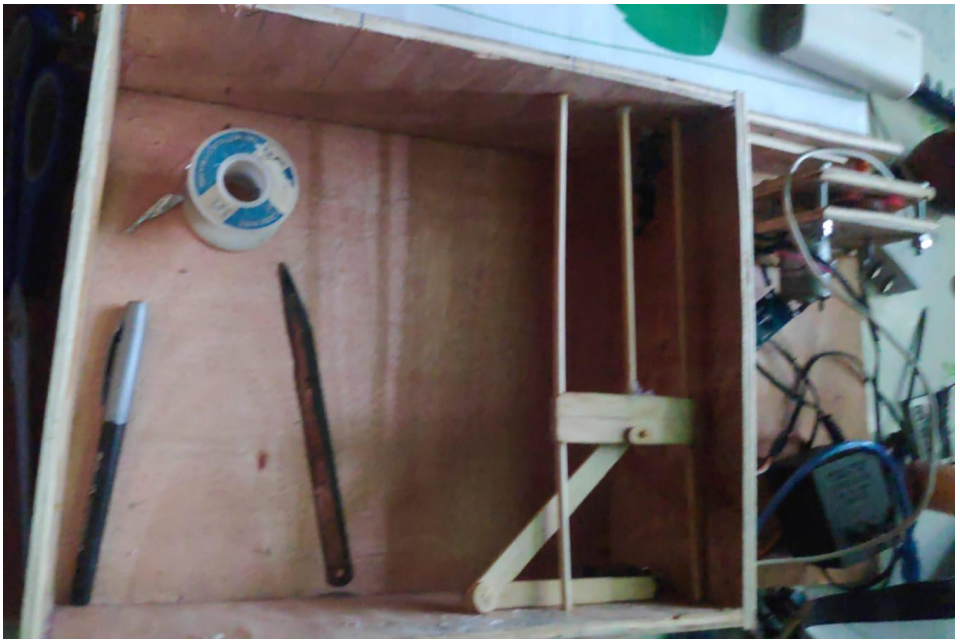

Figura 92. Construcción de pistones.

Se ejecutan las primeras pruebas y verificaciones del primer pistón para comprobar el proceso de funcionamiento el cual al detectar la fruta correspondiente realiza la acción de empujar a su cubículo. Esta misma prueba se lleva a cabo con la pera y aquellas frutas que se encuentran en mal estado.

Figura 93. Comprobación de pistones.

Se elaboraron tres cubículos de madera que tienen como función almacenar las frutas, están situados al frente del contenedor de pistones, ya que cuando ocurre el accionamiento de empujar las frutas caen en su respectivo cubículo logrando de esta manera una clasificación de frutas en buen estado y mal estado.

Figura 94. Clasificación de las frutas.

4.2 Elaboración del sistema del sistema electrónico

En la figura 95 se observa la conexión general y principal para el funcionamiento del prototipo, la cual se encuentra ubicada en la parte superior izquierda, al lado de la banda transportadora. Se encuentran las conexiones tales como la fuente de poder con el Arduino, un regulador de voltaje, empalme entre el motor y el primer Arduino, empalme entre el servomotor y el segundo Arduino, conexión de la cámara, LCD y banda transportadora.

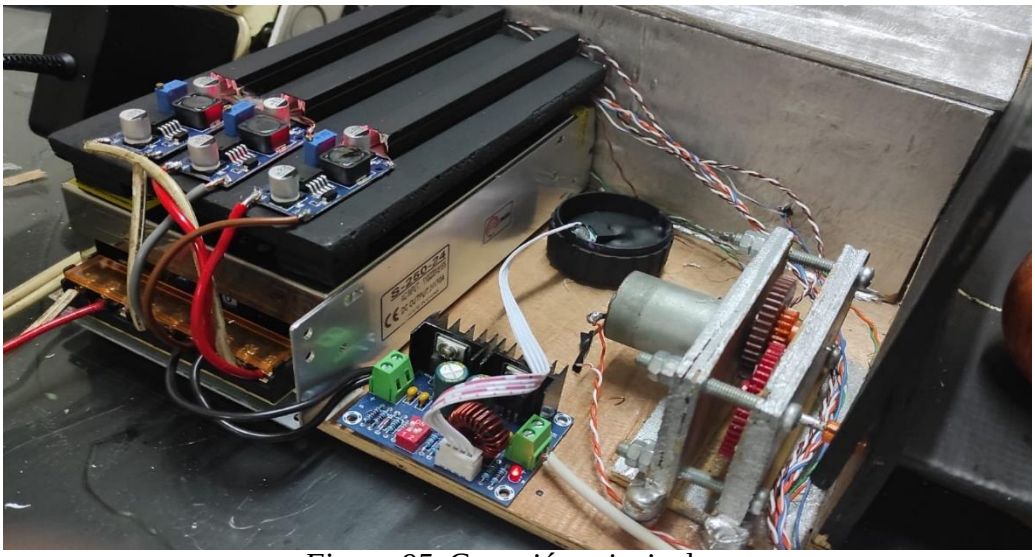

Figura 95. Conexión principal.

Se realizan pruebas de la conexión de la cámara con la Raspberry, la cual está ubicada estratégicamente dentro de un cubículo de madera para darle protección de cualquier agente externo. El cableado está cubierto por una canaleta que va adherida a la madera resguardando que las conexiones no se topen unas a otras, para que no exista ningún tipo de cortocircuito.

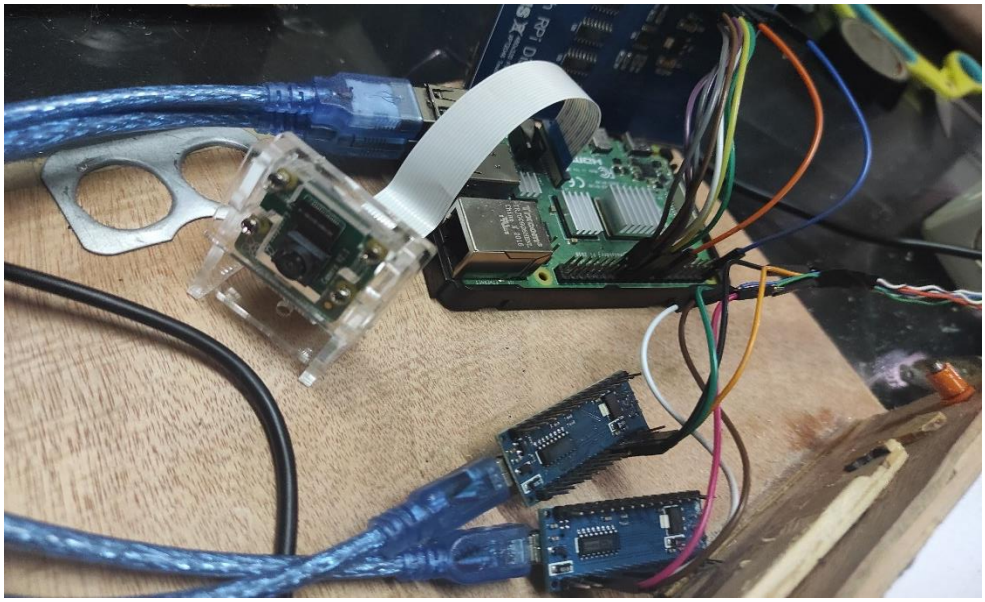

Figura 96. Cableado General.

Se realiza la conexión y prueba del funcionamiento de los pistones con sus respectivos sensores y servomotores, lo cual permitió verificar su eficacia al empujar y clasificar la fruta en su cubículo correspondiente.

Figura 97. Conexión de sensores.

Se posiciona cada sensor en la parte inferior del pistón debido a que si se lo ubica en otra zona podría no sensar bien, porque ciertas frutas como por ejemplo la pera debe ir acostada, ya que su base no es estable. El pistón tiene su parte frontal fabricada de madera, es de forma rectangular y esto permite que la fruta llegue a su cubículo correctamente.

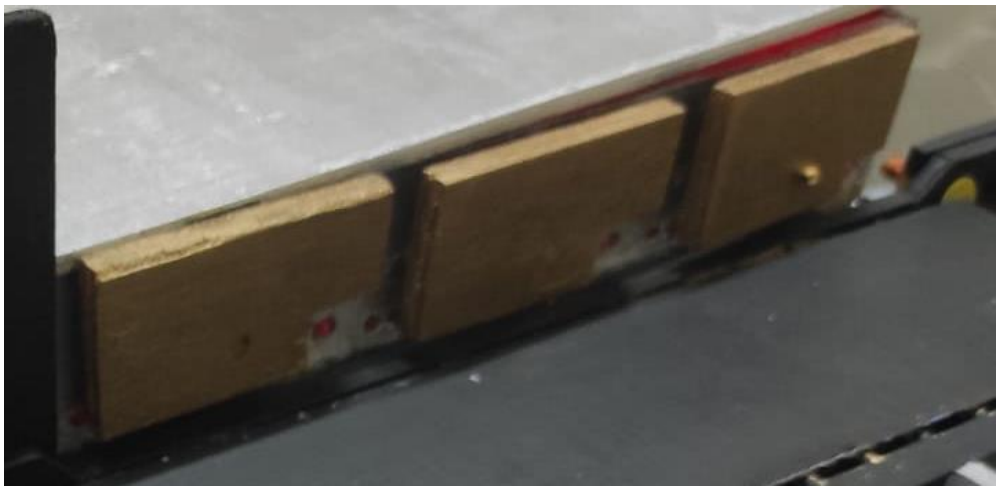

Figura 98. Posición de sensor.

Se fabricó un cubículo a la misma altura de la banda transportadora con la finalidad de instalar la cámara, el LCD y el primer sensor para obtener un mejor ángulo y para que el sensor pueda realizar su función correctamente, ya que es el que permite el accionamiento de la banda. El LCD va ubicado de forma horizontal permitiendo así tener una mejor visión con respecto a las frutas a analizar.

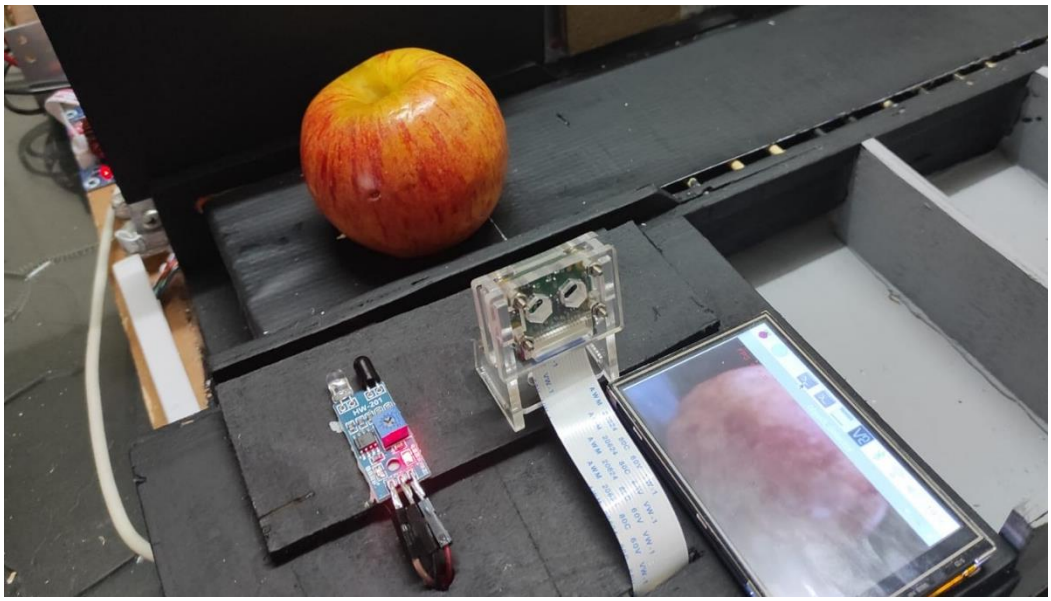

Figura 99. Posicionamiento de cámara.

4.3 Análisis de los resultados

Para elaborar un análisis de resultados, es importante realizar un estudio de los antecedentes del problema considerados para la elaboración del proyecto técnico, los cuales se basan en la contaminación de alimentos por diferentes agentes externos, provocando enfermedades, afectando a consumidores y a las industrias.

Actualmente, es común observar frutas en mal estado en mercados que es donde recurre más la población por sus bajos costos, y esto se debe a que no cuentan con un sistema de control que garantice la inocuidad alimentaria, es por eso que este prototipo busca analizar y mejorar los estándares de calidad de la industria alimenticia a través de la implementación de un sistema de control, usando la visión artificial como su mayor herramienta, ya que hoy en día es una tecnología que predomina mucho en las industrias porque permite obtener información sobre lo que sucede en el campo de visión.

Datos de entrenamiento de la red neuronal.

Este prototipo de clasificación y control de calidad se encuentra basado en redes neuronales, las cuales para ser entrenadas se utiliza TensorFlowLite, esta herramienta genera un formato ".tflite" ligero y portátil, estas características permiten que el archivo que se genera se ejecute de forma eficiente en dispositivos como Raspberry que disponen de memoria y procesamiento limitados.

TensorFlowLite contiene un sin número de scripts orientados a reentrenar modelos aplicando la técnica de Transfer Learning, durante la implementación del

modelo de red neuronal se realizó pruebas con distintos modelos y se analizó el más óptimo para el proyecto.

Una vez elegido el modelo más óptimo para la red neuronal se procede con la ejecución del análisis del dataset, con esto se logra configurar el cuello de botella o llamado bottleneck, estos términos se refieren a la penúltima capa de la red neuronal, la cual está encargada de la clasificación, en este punto la red neuronal se entrena para generar un conjunto de valores lo más acertados posibles para que la red neuronal sea capaz de distinguir entre las clases para las que fue configurada.

Es importante que al utilizar TFLite, se realice pruebas para verificar que la red es eficiente en el dispositivo que se esté probando, ya que cada dispositivo tiene diferencias ya sea por capacidad de procesamiento o por falta de memoria RAM.

A la hora de entrenar un modelo de red neuronal se debe usar un número de ejemplos más conocido con el nombre de batch o lote, que es el número de ejemplos que existen en un lote, estos valores tienen que ser de entre 1 y 1000, a la vez que se analiza se da consigo un step o paso, esto se le denomina a la cantidad de veces que se procesan las imágenes por dicho lote. En este caso al tener un dataset de 2000 imágenes y se configura un batch size de 20 no da como resultado una variable training steps de la siguiente manera 2000/20=1000 steps.

Con el etiquetado previo en Labellmg y la creación del DataSet, se entrena la red neuronal, se extrae la carpeta train la cual tienen imágenes de referencias del modelo de fruta a identificar, y se hace con “Train_data”, luego se especifica el modelo de arquitectura con “model_spec”, se utiliza “batch_size” para definir el número de ejemplos que recorren la red en cada entrenamiento, se define las épocas con las que se entrena la red neuronal

Figura 100. Entrenamiento del modelo.

Una vez realizado el entrenamiento, se valida el sistema con “Val_data” arrojando los siguientes resultados:

- El porcentaje de validación de las manzanas es de 0.7736483.
- El porcentaje de validación de las manzanas dañadas es de 0.8481276.
- El porcentaje de validación de las peras es de 0.6042374.
- El porcentaje de validación de las peras dañadas es de 0.55738646.

Se exporta el modelo con “Model_export” entregando un archivo “rpi.tflite” el cual se carga en la Raspberry.

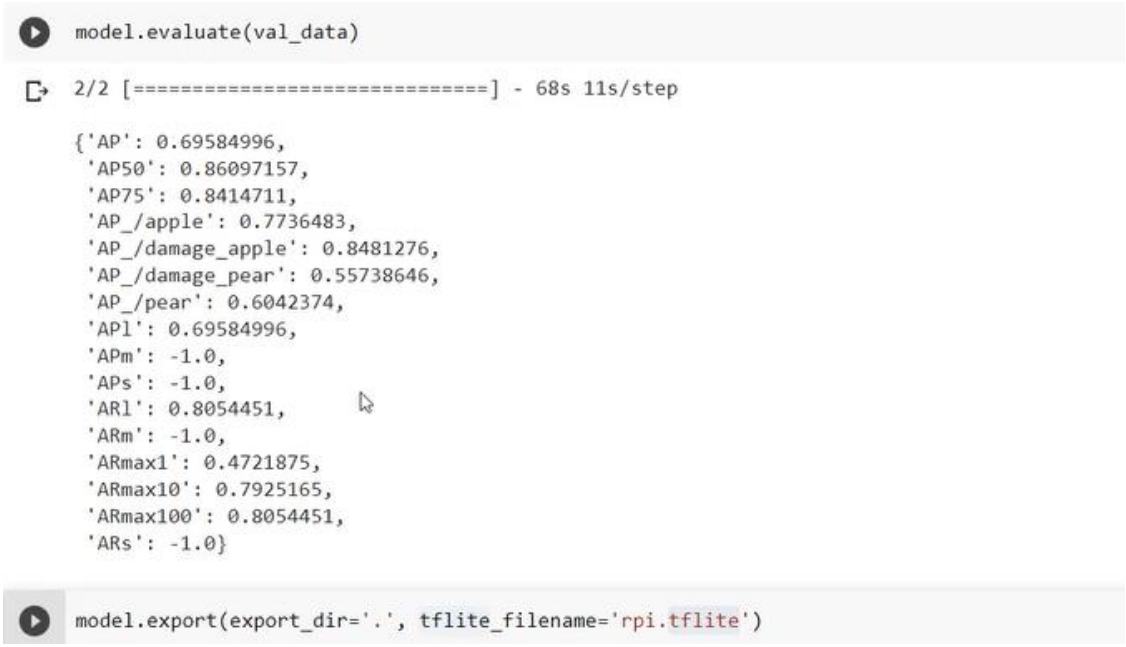

Figura 101. Datos de entrenamiento.

Se exporta el modelo con “Model_export” entregando un archivo “rpi.tflite” el cual se carga en la Raspberry.

Para obtener el porcentaje de aciertos se utilizó “det_lite 2” es una de las clases de redes neuronales en TensorFlowLite, que es menos rápida, pero más asertiva.

Se coloca una manzana en buen estado en la banda transportada para su respectiva identificación y se evalúa el porcentaje de acierto, el cual arrojo un resultado de 0.96 con un FPS de 4.1.

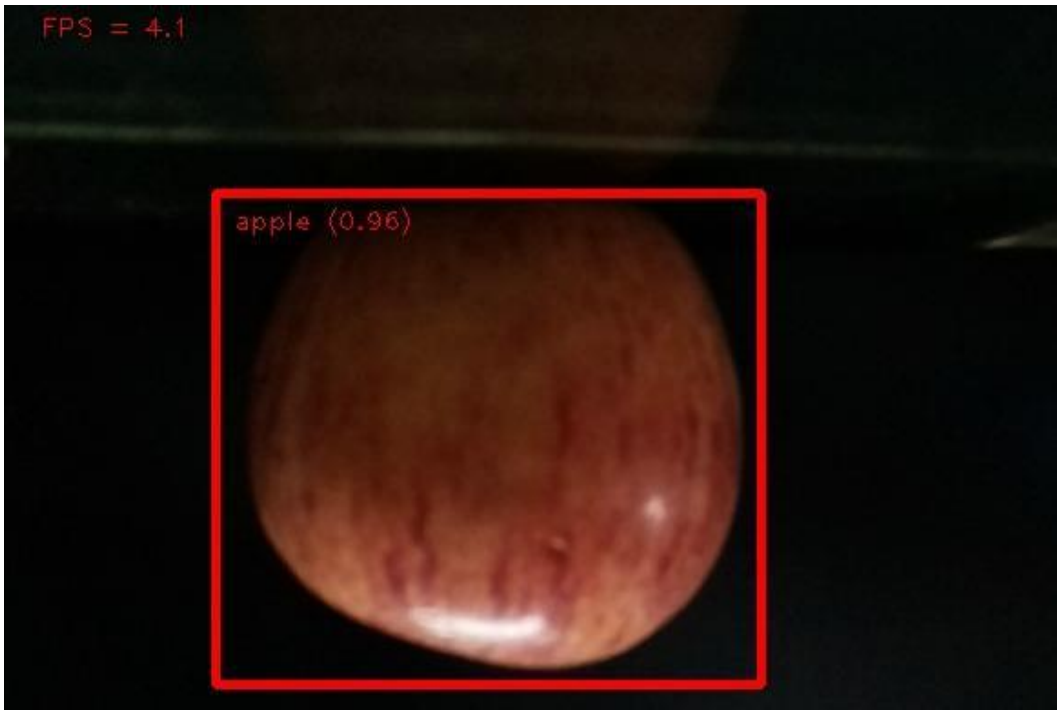

Figura 102. Visualización de manzana.

Se evalúan dos manzanas en buen estado con diferentes enfoques de luminosidad, ya que es importante porque ayuda a que se observe con mayor claridad el color de la fruta al momento que sea sensada, si no hay suficiente luminosidad podría no detectarla o confundirla como un objeto no identificado. Como resultado en la figura 103 se puede apreciar que tiene un mayor acierto la manzana con más luminosidad que es de 0.95 y un menor acierto la manzana que es de 0.65 con un FPS de 4.2 para ambas.

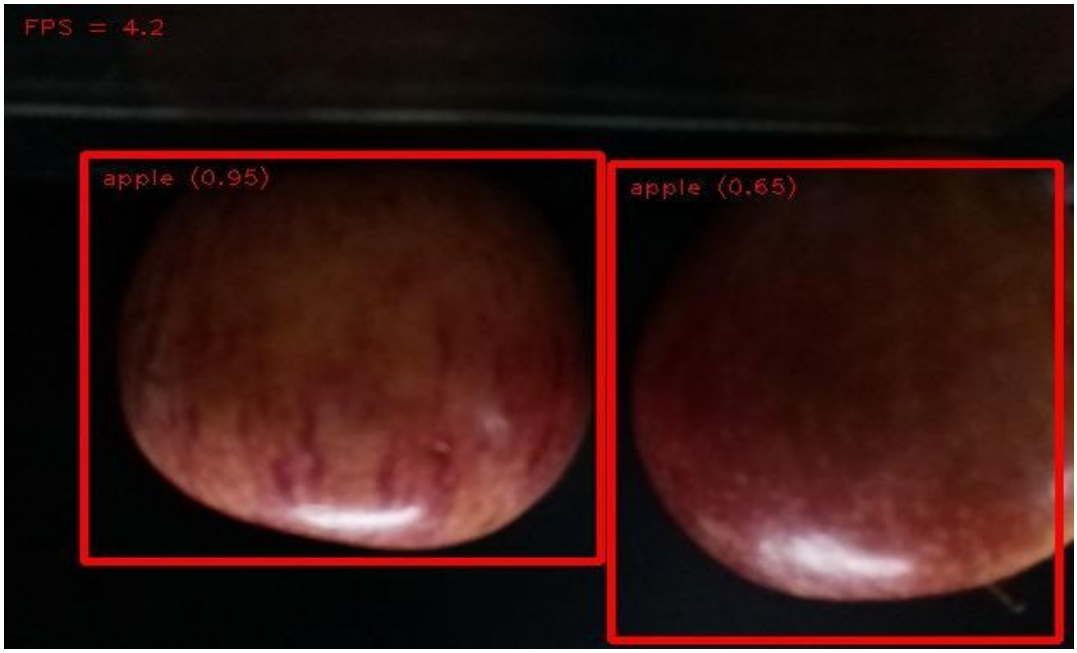

Figura 103. Visualización de dos manzanas.

Se observa como el sensor principal detecta la manzana y este a su vez lo refleja a la pantalla LCD, se visualiza el etiquetado correspondiente a la fruta con su respectivo porcentaje de acierto y empieza el funcionamiento de la banda transportadora el cual traslada la fruta hasta el pistón correspondiente para su clasificación, dejándola caer en el cubículo correspondiente.

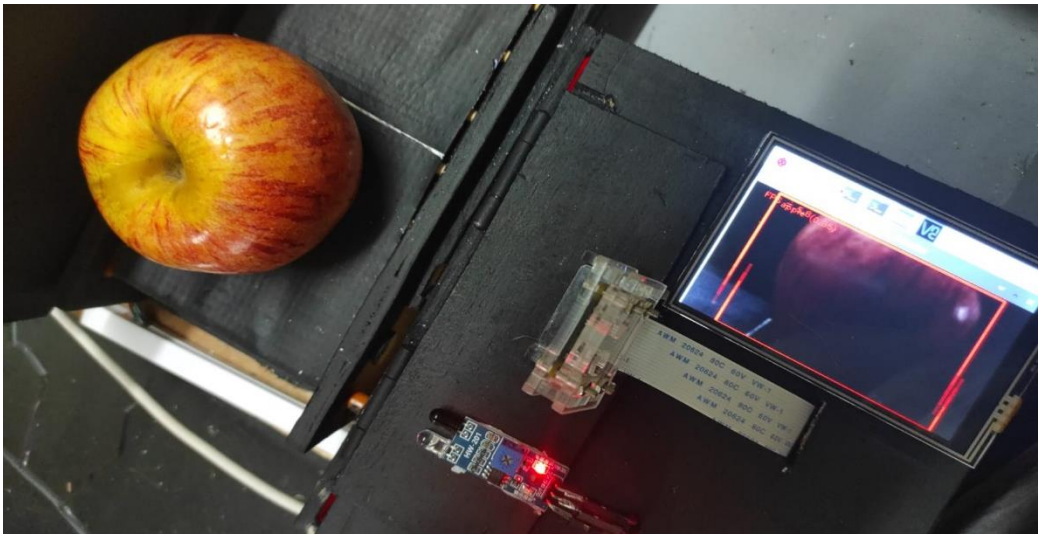

Figura 104. Etiquetado reflejado en el LCD..

Como se observa en la Figura 105 el resultado que se obtiene en la pera corresponde a un porcentaje del 0.88% de acierto con un FPS de 4.0s.

Cuando se hace el etiquetado, por ejemplo en la pera es importante que se visualice toda la fruta incluido el tallo, ya que esto puede ocasionar problemas a la hora del sensado, ya que al colocarla sin el tallo no sería reconocida.

Con la red entrenada se carga el archivo de entrenamiento en la Raspberry, se ejecuta el algoritmo de detección de objetos y se analiza los resultados.

Figura 105. Visualización de la pera.

Se evalúa colocando más de una pera en frente de la cámara, para verificar si detecta ambas frutas que se encuentran en buen estado. En este caso se puede observar que tienen dos porcentajes de aciertos con una mínima diferencia, la primera pera que está cerca de la cámara cuenta con un porcentaje de 0.77 y la segunda pera que cuenta con un porcentaje de 0.65. Este resultado indica que el prototipo es viable para la identificación y clasificación en el ámbito de la industria alimentaria con el fin de mejorar los estándares de calidad.

Figura 106. Visualización de dos peras.

Se activa el sensor principal que detecta la pera y este a su vez lo refleja a la pantalla LCD, se visualiza el etiquetado correspondiente a la fruta con su respectivo porcentaje de acierto empezando el funcionamiento de la banda transportadora el cual traslada la fruta hasta el pistón correspondiente para su clasificación.

Figura 107. Etiquetado de la pera en el LCD.

Actualmente uno de los principales problemas en la industria alimentaria es la contaminación de diferentes frutas por diversos agentes externos tales como bacterias, hongos, contaminación química, física, entre otros. Debido a esta problemática es importante establecer protocolos de sanidad, al igual que elaborar e implementar sistemas de control de calidad que permitan detectar cualquier tipo de daño en una fruta para que de esta manera los consumidores se encuentren satisfechos con alimentos saludables.

Es por esto que una de las prioridades para la elaboración del prototipo es la identificar aquellas frutas que se encuentren en mal estado, para posteriormente definir los posibles tipos y niveles de contaminación. Para ello, es importante haber realizado una investigación acerca de que los diferentes factores que afectan la calidad de alimentos y tomar en consideración el aspecto físico de las frutas que se encuentran en mal estado, ya sea por el color, si presenta algún golpe, entre otros. Esto permite que al momento de la clasificación se pueda realizar una correlación entre la fruta clasificada en mal estado y el tipo de contaminante.

Uno de los contaminantes en la pera que se usó para este prototipo fue un hongo denominado Moteado o Sarna, como se visualiza en la figura 108 cuenta con características físicas que son unas pequeñas manchas marrones que salen en la fruta cuando se encuentra madura, además estas manchas también se pueden generar por situaciones ambientales como el mantener almacenada la fruta en bajas temperaturas lo que provoca que la pera empiece un proceso de pudrición.

Como resultado de esta prueba se obtiene un porcentaje de acierto del 0.36 y 0.7 respectivamente con un FPS de 4.2.

Figura 108. Peras en mal estado.

Se observa una manzana afectada por un golpe y expuesta al ambiente por un día. Uno de sus contaminantes es un hongo denominado Penicillium expansum, mancha la fruta de color marron y ocasiona podredumbre a partir de un corte o golpe. Para verificar el funcionamiento de la red la visión artificial la reconoce clasificándola como una manzana en mal estado, dando como resultado un porcentaje de acierto del 0.61% con FPS de 4.1.

Figura 109. Manzana en mal estado.

Se activa la banda y se traslada la fruta hasta el tercer pistón correspondiente a las manzanas en mal estado el cual mediante el sensor de posición de las frutas se activa y se clasifica la fruta dejándola caer en el recipiente correspondiente.

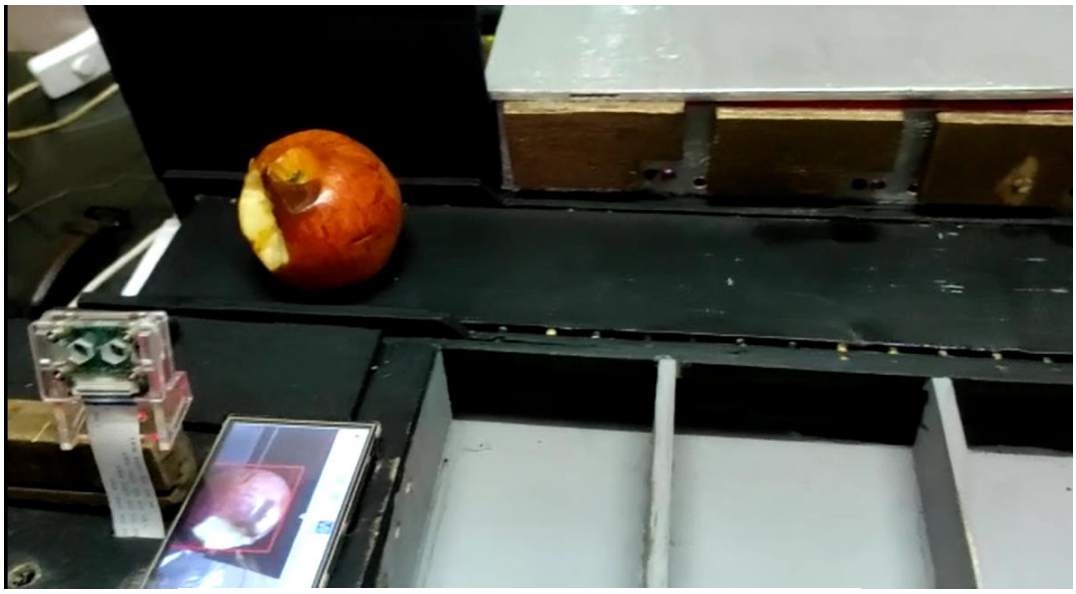

Figura 110. Visualización del etiquetado en el LCD.

Figura 111. Clasificación de fruta en mal estado a su cubículo correspondiente.

Este último proceso de identificar frutas en mal estado permitió corroborar la eficacia del funcionamiento del prototipo. El cual es útil para los pequeños negocios como mercados, tiendas, etc. debido a sus materiales y costos accesibles, permitiendo lograr un óptimo almacenamiento y de esta manera vender las frutas de buena calidad.

5. CONCLUSIONES

- Se concluye que realizar una estimación aproximada en el anteproyecto acerca de los precios de los componentes a utilizar para la elaboración del prototipo, permitió determinar la factibilidad financiera considerando costos y los beneficios.

- El análisis para la implementación del prototipo, a través de los datos obtenidos a partir de la investigación realizada, permitió elaborar un proceso jerárquico considerando los beneficiarios y recursos para su propuesta de solución, determinando de esta manera la factibilidad técnica.

- El prototipo cuenta con un sistema en el que la tecnología e infraestructura implementada permite mejorar la clasificación de las frutas, asegurando que las industrias y consumidores cuenten con productos de calidad y procesos más eficientes.

- Se concluye que las diversas investigaciones realizadas para la elaboración del proyecto técnico permitieron conocer los diferentes niveles de contaminantes que afectan la calidad de la manzana y la pera, los cuales posteriormente fueron detectados por la visión artificial y las redes neuronales.

6. RECOMENDACIONES

- Para la elaboración de la estructura física de un prototipo es recomendable utilizar el programa Solidworks para tener un diseño con dimensiones realistas.

- Se recomienda adquirir más conocimiento a estudiantes acerca de la visión artificial, ya que es una disciplina científica reconocida por sus métodos de procesar y analizar imágenes del mundo real.

- El uso de sensores infrarrojos es recomendable ya que estos sirven para la detección de objetos de diferentes formas, colores, etc.

- Realizar simulaciones, esto con el fin de verificar el funcionamiento de todos los componentes del prototipo.

- Utilizar una cámara con lente fijo, ya que tiene una distancia focal fija y no existen distorsiones en el momento de la captura de la imagen.

- Utilizar un LCD que ayude a visualizar correctamente el análisis que se está realizando en tiempo real.

- Se debe contar con iluminación adecuada para que la imagen capturada por la cámara sea la más óptima para el análisis de la red neuronal.

- Se recomienda usar el Tensorflow lite, ya que ayuda al procesador de la Raspberry, mejorando los tiempos de respuesta del equipo.

- No utilizar un modelo ya entrenado de Tensorflow directamente en la Raspberry Pi ya que se crean conflictos con algunas librerías, y el procesador tendrá inconvenientes procesando todos los datos.

- Al momento de etiquetar las imágenes se debe utilizar una muestra bastante grande de imágenes, ya que esto le otorga versatilidad a la red neuronal.

- Se recomienda capturar imágenes con fondos variados, esto ayuda a la red a identificar los objetos en diferentes superficies.

- Se recomienda que las imágenes que se utilicen para entrenar la red no se utilicen para la etapa de validación ya que esto da un conflicto en el entrenamiento, lleva a que la red tenga un entrenamiento óptimo falso por ya que solo esta memorizando la imagen.

- Se sugiere entrenar la red neuronal con una tarjeta GPU ya que esto reducirá los tiempos de entrenamiento de la red neuronal.

REFERENCIAS BIBLIOGRÁFICAS

agrolib.rs. (19 de Julio de 2019). Obtenido de https://es.agrolib.rs/kieffer-pera-asiatica-propiedades-cuidado-y-siembra-pera-112

Alonso, R. (26 de Agosto de 2021). Hardzone. Obtenido de https://hardzone.es/reportajes/que-es/fuente-alimentacion-caracteristicas/

Amazon. (s.f.). Obtenido de https://www.amazon.com/-/es/PT2019080501/dp/B07YR6M6F6?ref_=ast_sto_dp&th=1

Aprende con Alf. (4 de Octubre de 2020). Obtenido de https://aprendeconalf.es/docencia/python/manual/numpy/

Arduino. (s.f.). Obtenido de https://arduino.cl/arduino-nano/

Atria Innovation. (22 de Octubre de 2019). Obtenido de https://www.atriainnovation.com/que-son-las-redes-neuronales-y-sus-funciones/

Az-Delivery. (s.f.). Obtenido de https://www.az-delivery.de/es/products/mg90s-micro-servomotor

Barriga, A. J. (26 de Junio de 2020). Geek Electrónica. Obtenido de https://geekelectronica.com/que-son-los-circuitos-y-sistemas-electronicos/

Bionatural Fruits. (s.f.). Obtenido de https://bionaturalfruits.com/es/mercat/manzana-red-delicious/

Blog. (s.f.). Obtenido de http://maratraseltrimbre.blogspot.com/2014_02_01_archive.html

Blog, O. E. (6 de Agosto de 2018). Obtenido de https://omsespana.com/blog/descubre-para-que-sirven-los-rodillos-transportadores/

Brunete, A., Segundo, P. S., & Herrero, R. (28 de Julio de 2020). Bookdown. Obtenido de https://bookdown.org/alberto_brunete/intro_automatica/

Brunete, A., Segundo, P. S., & Herrero, R. (28 de Julio de 2020). Bookdown. Obtenido de https://bookdown.org/alberto_brunete/intro_automatica/

Casals, K. (16 de Julio de 2013). MakinGastronomy. Obtenido de https://kikocasals.com/2013/07/16/peras-bosc-y-azucar-crema/

Castillo, J. A. (20 de Enero de 2019). Obtenido de https://www.profesionalreview.com/2019/01/20/rgb-que-es/

Centro, S. (30 de Abril de 2011). El Comercio. Obtenido de https://www.elcomercio.com/actualidad/negocios/seis-variedades-de-manzanas-se.html

ChileValora. (Abril de 2012). Obtenido de https://www.oitcinterfor.org/sites/default/files/certificacion/ChileValora_GuiaApoyoAnalisisFuncional.pdf

Cinatur Group. (04 de Febrero de 2016). Obtenido de http://www.cinatur.com/es/causas-del-deterioro-de-frutas-hortalizas

Comax. (s.f.). Obtenido de https://www.ecomax.store/product-page/pera-uvilla-ambate%C3%B1a

Componentes101. (17 de Julio de 2021). Obtenido de https://components101.com/microcontrollers/arduino-nano

Components Info. (6 de Agosto de 2020). Obtenido de https://www.componentsinfo.com/tip31c-transistor-pinout-equivalent/

Delgado, A. (21 de Noviembre de 2020). Obtenido de https://www.geeknetic.es/Raspberry-Pi/que-es-y-para-que-sirve

Delgado, D. O. (15 de 09 de 2017). Open Webinars. Obtenido de https://openwebinars.net/blog/que-es-tensorflow/

Disgralec. (s.f.). Obtenido de http://www.disgralec.com/home/100-manzana-ambatena-5-u.html

Dreamstime. (s.f.). Obtenido de https://es.dreamstime.com/imagen-de-archivo-libre-de-regal%C3%ADas-tres-manzanas-image1811006

Electrocomponentes. (s.f.). Obtenido de https://www.electrocomponentes.es/fijas/39-resistencia-1k-ohm-025w.html

Esther Lopez. (08 de Octubre de 2020). Obtenido de https://unprogramador.com/sensor-infrarrojo-fc-51-con-arduino/

Exportación Fruta Fresca. (30 de mayo de 2017). Obtenido de http://higos27export.blogspot.com/2017/05/peras-y-sus-tipos.html

Fernandez, Y. (3 de Agosto de 2020). Obtenido de https://www.xataka.com/basics/que-arduino-como-funciona-que-puedes-hacer-uno#:~:text=Arduino%20es%20una%20plataforma%20de,para%20los%20creadores%20y%20desarrolladores.

González, A. G. (2 de Diciembre de 2016). Panamahitek. Obtenido de http://panamahitek.com/que-es-y-como-funciona-un-servomotor/

Grupo Velasco. (s.f.). Obtenido de http://www.velasco.com.ec/velasco/producto.php?id=4451

GSL Industrias. (9 de Junio de 2021). Obtenido de https://www.industriasgsl.com/blog/post/que-es-un-sistema-de-control#:~:text=Sistema%20de%20lazo%20cerrado%20(o,falla%20que%20recibe%20el%20controlador

Guía Metabólica. (12 de Febrero de 2019). Obtenido de https://metabolicas.sjdhospitalbarcelona.org/consejo/pera

Hiraoka. (s.f.). Obtenido de https://hiraoka.com.pe/blog/post/estabilizador-que-es-para-que-sirve-y-como-funciona#:~:text=Un%20estabilizador%2C%20tambi%C3%A9n%20conocido%20como,tensi%C3%B3n%20y%20variaciones%20de%20voltaje.

Infaimon. (28 de octubre de 2019). Obtenido de https://blog.infaimon.com/sistemas-de-vision-artificial-tipos-aplicaciones/

Ingeniería Mecafenix. (25 de Febrero de 2019). Obtenido de https://www.ingmecafenix.com/automatizacion/sistema-de-control/

IRP. (s.f.). Obtenido de https://irp-intralogistica.com/que-son-las-bandas-transportadoras/

IRP. (18 de Diciembre de 2018). Obtenido de https://irp-intralogistica.com/que-son-las-bandas-transportadoras/

Jardineria On. (s.f.). Obtenido de https://www.jardineriaon.com/manzana-royal-gala.html

Komunicate. (s.f.). Obtenido de https://komunicate.net/dispositivos/31-arduino-nano.html

La Vanguardia. (17 de Diciembre de 2021). Obtenido de https://www.lavanguardia.com/comer/frutas/20180611/3794/manzana-propiedades-beneficios-tipos.html

Leantec. (21 de Octubre de 2015). Obtenido de https://leantec.es/como-medir-el-tiempo-con-arduino-y-la-libreri/

LEDBOX. (s.f.). Obtenido de https://www.ledbox.es/generar-pdf-8889~fuente-de-alimentacion-24v-150w-6-5a-mean-well-lrs-150-24

Mecafenix, I. (8 de Agosto de 2019). Obtenido de https://www.ingmecafenix.com/electronica/el-transistor/

MecatrónicaLATAM. (4 de Mayo de 2021). Obtenido de https://www.mecatronicalatam.com/es/tutoriales/sensores/

Medina, G. (30 de Septiembre de 2020). Tecnoinformatic. Obtenido de https://tecnoinformatic.com/c-informatica-basica/sistemas-de-control/

Medina, J. (s.f.). Obtenido de https://aprende.com/blog/oficios/instalaciones-electricas/tipos-de-resistencias-electronicas/

Menna. (s.f.). ComoFunciona. Obtenido de https://como-funciona.co/un-piston/

Microcontroladores. (4 de Junio de 2020). Obtenido de https://microcontroladoress.com/arduino/arduino-nano/

Mouser. (s.f.). Obtenido de https://www.mouser.com/new/raspberry-pi/raspberry-pi-4-b/

Nelson, J. (16 de Marzo de 2020). Obtenido de https://blog.roboflow.com/labelimg/

Netinbag. (s.f.). Obtenido de https://www.netinbag.com/es/internet/what-is-an-arduinoreg-ide.html

Nirian, P. O. (4 de Mayo de 2020). Economipedia. Recuperado el 8 de Septiembre de 2021, de https://economipedia.com/definiciones/control-de-calidad.html

No, D. (3 de Abril de 2020). Esploradores. Obtenido de https://www.esploradores.com/sys/#:~:text=El%20m%C3%B3dulo%20sys%20de%20la,interact%C3%BAan%20estrechamente%20con%20el%20int%C3%A9rprete.

Noticias ONU. (12 de Febrero de 2019). Obtenido de https://news.un.org/es/story/2019/02/1451101

Nuñez, M. (22 de Diciembre de 2020). Obtenido de https://blog.generaclatam.com/motor-el%C3%A9ctrico

Octopart. (s.f.). Obtenido de https://octopart.com/tip31c-stmicroelectronics-15555?gclid=Cj0KCQiAjc2QBhDgARIsAMc3SqQfjDFRPd8CajXQwRwpOXrQzgdI7HeV_dzYF4A-C8JJdDTqzJlmJUgaApL6EALw_wcB

Osaka Electronics. (s.f.). Obtenido de https://osakaelectronicsltda.com/motores/servo-motores/servo-motor-mg90s.html

Pishop. (s.f.). Obtenido de https://www.pishop.co.za/store/ws-35inch-rpi-lcd-c-480x320-125mhz-high-speed-spi

Plata, G. V. (2003). La Contaminación de los Alimentos, un Problema por Resolver. Revista de la universidad industrial de Santander , 48.

Protosuppplies. (s.f.). Obtenido de https://protosupplies.com/product/resistor-1k-ohm-5-14w25-pack/

R. Díaz-Sobac, J. Vernon-Carter. (02 de Octubre de 2009). tandf online. Obtenido de https://www.tandfonline.com/doi/pdf/10.1080/11358129909487594

Rodriguez. (20 de Noviembre de 2017). 1millionbot. Obtenido de https://1millionbot.com/google-presenta-tensorflow-lite-para-moviles/

Rodríguez, B. (30 de Mayo de 2020). Obtenido de elchapuzasinformatico.com/2020/05/llega-una-nueva-raspberry-pi-4-con-8-gb-de-ram-y-sistema-operativo-de-64-bits/

Rodríguez, H. (27 de Abril de 2021). Crehana. Obtenido de https://www.crehana.com/ec/blog/desarrollo-web/que-es-opencv/

Rosalandia. (s.f.). Obtenido de https://rosalandia.com/varios/manzana-golden-delicious

SADITRANSMISIONES. (s.f.). Obtenido de https://saditransmisiones.com/pinones-industriales/

SANDOROBOTICS. (s.f.). Obtenido de https://sandorobotics.com/producto/af-3099/

Santos, T. (25 de Octubre de 2020). Alura. Obtenido de https://www.aluracursos.com/blog/google-colab-que-es-y-como-usarlo

Servicio de acreditación ecuatoriano. (19 de Septiembre de 2018). Obtenido de https://www.acreditacion.gob.ec/control-calidad-para-frutas-y-hortalizas/

SOSER S.A. (31 de Agosto de 2010). Obtenido de http://www.soser.cl/wp-content/uploads/2014/11/frutasyverduras.pdf

Sossa. (6 de Enero de 2021). Inteligencia-Artificial. Obtenido de https://inteligencia-artificial.dev/tipos-redes-neuronales/

Talentum. (20 de Enero de 2020). Obtenido de https://talentumdigital.cl/2020/01/20/por-que-elegir-solidworks/

ThePiHut. (s.f.). Obtenido de https://thepihut.com/products/raspberry-pi-camera-module

Tipán, D. (Febreo de 2019). Obtenido de https://bibdigital.epn.edu.ec/bitstream/15000/20061/1/CD-9489.pdf

Unit Electronics. (s.f.). Obtenido de https://uelectronics.com/producto/raspberry-pi-camara-modulo-v2-imx219-8mp/

Villalobos, J. (2 de Octubre de 2013). Codigo Programación. Obtenido de http://codigoprogramacion.com/cursos/javascript/introduccion-a-json-sintaxis-y-ejemplos.html#.YhOwfehByM9

Web-Robótica. (7 de Abril de 2019). Obtenido de https://www.web-robotica.com/arduino/conceptos-basicos-arduino/como-usar-el-modulo-

sensor-de-infrarrojos-ir-fc-51-para-evitar-obstaculos-con-robot-arduinogenuino

Xeridia. (06 de Mayo de 2019). Obtenido de https://www.xeridia.com/blog/la-vision-artificial-y-el-procesamiento-de-imagenes

Zambrano, D. P. (23 de Octubre de 2017). Animales y Biologia . Obtenido de https://naturaleza.animalesbiologia.com/plantas/tipos-de-frutas/pera-peral-pyrus-communis#:~:text=Caracter%C3%ADsticas%20de%20la%20pera.%20La%20pera%20es%20la,de%20una%20tonalidad%20de%20color%20verde%20pastel%2C%20

ANEXOS

Anexo 1. Costo de los materiales utilizados

Cantidad	Detalle	Valor Unitario	Valor Total
	Ensamble General		
4	Plancha de PLYWOOD	$2.50	$10
1	Caja de tornillos	$5	$5
4	Engranes	$5	$20
1	Broca	$0.50	$0.50
5	Tubos de Silicon grueso	$0.35	$1.75
30	Varas de madera	$0.05	$1.50
4	Tuercas 1/8	$0.02	$0.08
1	Lona	$2.50	$2.50
6	Tuercas 1/4	$0.10	$0.60
35	Cilindros de madera(paquete)	$4.00	$4.00
4	Varilla metálica	$1	$4.00
1	Cemento de contacto	$5	$5
2	Tarros de pintura sintética	$4	$8
1	Caja de clavos	$1.50	$1.50
1	Tarro de silicon liquida	$1	$1
1	Cinta aislante	$1.50	$1.50
	Subtotal		**$66.93**
	Sistema electrónico		
1	Cámara Raspberry Pi v2	$48.99	$48.99
1	Raspberry pi 4 B+	$189.99	$189.99
4	Sensores infrarrojo IR FC-51	$1.50	$6
1	Motor 12v dc	$7	$7
3	Servomotores mg996r	$3	$25.5
2	Arduino nano	$12	$24
1	Display Raspberry	$29.75	$29.75
1	Cable UTP 10m	$5	$5
1	Fuente genérica 24V-14.65A	$30	$30

2	DC-DC Stepdown	$15	$30
1	Soporte de PI Cam	$10	$10
1	Flex 16pin	$2	$2
1	1m Cable conector AC	$0.30	$0.30
1	Enchufe AC	$2	$2
1	Paquete de jumpers macho-hembra	$4	$4
1	Tarjeta SD 32GB	$10	$10
1	Pulsador ON/OFF	$2.5	$2.50
5	Disipadores	$1.50	$7.50
	Subtotal		**$434.53**
	Total		**$501.46**

Anexo 2. Pascal Voc de una imagen de dataset guardada en un archivo xml.

```
<annotation>
	<folder>train</folder>
	<filename>IMG_20220220_155443.jpg</filename>
	<path>C:\Users\marib\Documents\Tesis\datasetv3\train\IMG_20220220_1
55443.jpg</path>
	<source>
		<database>Unknown</database>
	</source>
	<size>
		<width>4640</width>
		<height>2088</height>
		<depth>3</depth>
	</size>
	<segmented>0</segmented>
	<object>
		<name>apple</name>
		<pose>Unspecified</pose>
		<truncated>0</truncated>
		<difficult>0</difficult>
		<bndbox>
			<xmin>1147</xmin>
			<ymin>230</ymin>
			<xmax>2952</xmax>
			<ymax>1903</ymax>
		</bndbox>
	</object>
</annotation>
```

Anexo 3. Códigos realizados en Python.

```
import argparse
import sys
import time

import cv2
from object_detector import ObjectDetector
from object_detector import ObjectDetectorOptions
import utils

#ino control
import serial,time,random,threading

servmotor = ["PT1","PT2","PT3"]
aleatory = "PT3"

def onMotors(answer, inoMotor,inoServMo):

  global aleatory
  if "IN" in str(answer):
    inoMotor.write("M12V ON".encode())
    print("MOTOR ON")
    aleatory = servmotor[random.randint(0,2)]
    print(aleatory)
  else:
    if aleatory:
      if aleatory in str(answer):
        sendStr = aleatory + " ON"
        inoMotor.write("M12V OFF".encode())
        inoServMo.write(sendStr.encode())
        print(sendStr)
        print("M12V OFF")

def execComunicationWithIno():
  print('Running tesis.py. Press CTRL-C to exit.')
  with serial.Serial("/dev/ttyUSB0", 9600, timeout=1) as arduinoMotor:
    with serial.Serial("/dev/ttyUSB1", 9600, timeout=1) as arduinoServMo:
      time.sleep(0.01) #wait for serial to open
      if arduinoMotor.isOpen() and arduinoServMo.isOpen():
        print("{} connected!".format(arduinoMotor.port))
        print("{} connected!".format(arduinoServMo.port))
        try:
          while True:

            while arduinoMotor.inWaiting()==0: pass
            if  arduinoMotor.inWaiting()>0:
              answer=arduinoMotor.readline() #IN
              onMotors(answer,arduinoMotor,arduinoServMo)
              arduinoMotor.flushInput() #remove data after reading
        except KeyboardInterrupt:
          print("KeyboardInterrupt has been caught.")
```

```
def run(model: str, camera_id: int, width: int, height: int, num_threads: int
        ) -> None:

  #Variable global to define which servo/motor on
  global aleatory
  # Variables to calculate FPS
  counter, fps = 0, 0
  start_time = time.time()

  # Start capturing video input from the camera
  cap = cv2.VideoCapture(camera_id)
  cap.set(cv2.CAP_PROP_FRAME_WIDTH, width)
  cap.set(cv2.CAP_PROP_FRAME_HEIGHT, height)

  # Visualization parameters
  row_size = 20  # pixels
  left_margin = 24  # pixels
  text_color = (0, 0, 255)  # red
  font_size = 1
  font_thickness = 1
  fps_avg_frame_count = 10

  # Initialize the object detection model
  options = ObjectDetectorOptions(
      num_threads=num_threads,
      score_threshold=0.3,
      max_results=10,
      #label_allow_list=['apple']
      )
  detector = ObjectDetector(model_path=model, options=options)

  #Exec com with arduinos

  thread = threading.Thread(target=execComunicationWithIno)
  thread.start()

  # Continuously capture images from the camera and run inference
  while cap.isOpened():
    success, image = cap.read()
    if not success:
      sys.exit(
          'ERROR: Unable to read from webcam. Please verify your webcam
settings.'
      )

    counter += 1
    image = cv2.flip(image, 1)

    # Run object detection estimation using the model.
    detections = detector.detect(image)

    #Define by default for piston 3
    if detections.count == 0:
```

```
      aleatory = "PT3"

    #Choose which piston should activate
    if "apple" in detections:
      aleatory = "PT1"
    elif "pear" in detections:
      aleatory = "PT2"
    else:
      aleatory = "PT3"


    # Draw keypoints and edges on input image
    image = utils.visualize(image, detections)

    # Calculate the FPS
    if counter % fps_avg_frame_count == 0:
      end_time = time.time()
      fps = fps_avg_frame_count / (end_time - start_time)
      start_time = time.time()

    # Show the FPS
    fps_text = 'FPS = {:.1f}'.format(fps)
    text_location = (left_margin, row_size)
    cv2.putText(image, fps_text, text_location, cv2.FONT_HERSHEY_PLAIN,
                font_size, text_color, font_thickness)

    # Stop the program if the ESC key is pressed.
    if cv2.waitKey(1) == 27:
      break
    cv2.imshow('object_detector', image)

  cap.release()
  cv2.destroyAllWindows()


def main():
  parser = argparse.ArgumentParser(
      formatter_class=argparse.ArgumentDefaultsHelpFormatter)
  parser.add_argument(
      '--model',
      help='Path of the object detection model.',
      required=False,
      default='efficientdet_lite0.tflite')
  parser.add_argument(
      '--cameraId', help='Id of camera.', required=False, type=int, default=0)
  parser.add_argument(
      '--frameWidth',
      help='Width of frame to capture from camera.',
      required=False,
      type=int,
      default=640)
  parser.add_argument(
      '--frameHeight',
```

```
      help='Height of frame to capture from camera.',
      required=False,
      type=int,
      default=480)
  parser.add_argument(
      '--numThreads',
      help='Number of CPU threads to run the model.',
      required=False,
      type=int,
      default=4)
  parser.add_argument(
      '--enableEdgeTPU',
      help='Whether to run the model on EdgeTPU.',
      action='store_true',
      required=False,
      default=False)
  args = parser.parse_args()

  run(args.model, int(args.cameraId), args.frameWidth, args.frameHeight,
      int(args.numThreads), bool(args.enableEdgeTPU))


if __name__ == '__main__':
  main()
```

```
import json
import platform
from typing import List, NamedTuple

import cv2
import numpy as np
from tflite_support import metadata

# pylint: disable=g-import-not-at-top
try:
  # Import TFLite interpreter from tflite_runtime package if it's available.
  from tflite_runtime.interpreter import Interpreter
  from tflite_runtime.interpreter import load_delegate
except ImportError:
  # If not, fallback to use the TFLite interpreter from the full TF package.
  import tensorflow as tf

  Interpreter = tf.lite.Interpreter
  load_delegate = tf.lite.experimental.load_delegate

# pylint: enable=g-import-not-at-top


class ObjectDetectorOptions(NamedTuple):
  """A config to initialize an object detector."""
```

```
  enable_edgetpu: bool = False
  """Enable the model to run on EdgeTPU."""

  label_allow_list: List[str] = None
  """The optional allow list of labels."""

  label_deny_list: List[str] = None
  """The optional deny list of labels."""

  max_results: int = -1
  """The maximum number of top-scored detection results to return."""

  num_threads: int = 1
  """The number of CPU threads to be used."""

  score_threshold: float = 0.0
  """The score threshold of detection results to return."""


class Rect(NamedTuple):
  """A rectangle in 2D space."""
  left: float
  top: float
  right: float
  bottom: float


class Category(NamedTuple):
  """A result of a classification task."""
  label: str
  score: float
  index: int

class Detection(NamedTuple):
  """A detected object as the result of an ObjectDetector."""
  bounding_box: Rect
  categories: List[Category]


def edgetpu_lib_name():
  """Returns the library name of EdgeTPU in the current platform."""
  return {
      'Darwin': 'libedgetpu.1.dylib',
      'Linux': 'libedgetpu.so.1',
      'Windows': 'edgetpu.dll',
  }.get(platform.system(), None)


class ObjectDetector:
  """A wrapper class for a TFLite object detection model."""

  _OUTPUT_LOCATION_NAME = 'location'
```

```
_OUTPUT_CATEGORY_NAME = 'category'
_OUTPUT_SCORE_NAME = 'score'
_OUTPUT_NUMBER_NAME = 'number of detections'

def __init__(
    self,
    model_path: str,
    options: ObjectDetectorOptions = ObjectDetectorOptions()
) -> None:

  # Load metadata from model.
  displayer = metadata.MetadataDisplayer.with_model_file(model_path)

  # Save model metadata for preprocessing later.
  model_metadata = json.loads(displayer.get_metadata_json())
  process_units = model_metadata['subgraph_metadata'][0][
      'input_tensor_metadata'][0]['process_units']
  mean = 127.5
  std = 127.5
  for option in process_units:
    if option['options_type'] == 'NormalizationOptions':
      mean = option['options']['mean'][0]
      std = option['options']['std'][0]
  self._mean = mean
  self._std = std

  # Load label list from metadata.
  file_name = displayer.get_packed_associated_file_list()[0]
  label_map_file = displayer.get_associated_file_buffer(file_name).decode()
  label_list = list(filter(len, label_map_file.splitlines()))
  self._label_list = label_list

  # Initialize TFLite model.
  if options.enable_edgetpu:
    if edgetpu_lib_name() is None:
      raise OSError("The current OS isn't supported by Coral EdgeTPU.")
    interpreter = Interpreter(
        model_path=model_path,
        experimental_delegates=[load_delegate(edgetpu_lib_name())],
        num_threads=options.num_threads)
  else:
    interpreter = Interpreter(
        model_path=model_path, num_threads=options.num_threads)

  interpreter.allocate_tensors()
  input_detail = interpreter.get_input_details()[0]

  # From TensorFlow 2.6, the order of the outputs become undefined.
  # Therefore we need to sort the tensor indices of TFLite outputs and to know
  # exactly the meaning of each output tensor. For example, if
  # output indices are [601, 599, 598, 600], tensor names and indices aligned
  # are:
  #   - location: 598
```

```
    #   - category: 599
    #   - score: 600
    #   - detection_count: 601
    # because of the op's ports of TFLITE_DETECTION_POST_PROCESS
    #
(https://github.com/tensorflow/tensorflow/blob/a4fe268ea084e7d323133ed7b986e
0ae259a2bc7/tensorflow/lite/kernels/detection_postprocess.cc#L47-L50).
    sorted_output_indices = sorted(
        [output['index'] for output in interpreter.get_output_details()])
    self._output_indices = {
        self._OUTPUT_LOCATION_NAME: sorted_output_indices[0],
        self._OUTPUT_CATEGORY_NAME: sorted_output_indices[1],
        self._OUTPUT_SCORE_NAME: sorted_output_indices[2],
        self._OUTPUT_NUMBER_NAME: sorted_output_indices[3],
    }

    self._input_size = input_detail['shape'][2], input_detail['shape'][1]
    self._is_quantized_input = input_detail['dtype'] == np.uint8
    self._interpreter = interpreter
    self._options = options

  def detect(self, input_image: np.ndarray) -> List[Detection]:
    """Run detection on an input image.

    Args:
        input_image: A [height, width, 3] RGB image. Note that height and width
          can be anything since the image will be immediately resized according
          to the needs of the model within this function.

    Returns:
        A Person instance.
    """
    image_height, image_width, _ = input_image.shape

    input_tensor = self._preprocess(input_image)

    self._set_input_tensor(input_tensor)
    self._interpreter.invoke()

    # Get all output details
    boxes = self._get_output_tensor(self._OUTPUT_LOCATION_NAME)
    classes = self._get_output_tensor(self._OUTPUT_CATEGORY_NAME)
    scores = self._get_output_tensor(self._OUTPUT_SCORE_NAME)
    count = int(self._get_output_tensor(self._OUTPUT_NUMBER_NAME))

    return self._postprocess(boxes, classes, scores, count, image_width,
                             image_height)

  def _preprocess(self, input_image: np.ndarray) -> np.ndarray:
    """Preprocess the input image as required by the TFLite model."""

    # Resize the input
    input_tensor = cv2.resize(input_image, self._input_size)
```

```
  # Normalize the input if it's a float model (aka. not quantized)
  if not self._is_quantized_input:
    input_tensor = (np.float32(input_tensor) - self._mean) / self._std

  # Add batch dimension
  input_tensor = np.expand_dims(input_tensor, axis=0)

  return input_tensor

def _set_input_tensor(self, image):
  """Sets the input tensor."""
  tensor_index = self._interpreter.get_input_details()[0]['index']
  input_tensor = self._interpreter.tensor(tensor_index)()[0]
  input_tensor[:, :] = image

def _get_output_tensor(self, name):
  """Returns the output tensor at the given index."""
  output_index = self._output_indices[name]
  tensor = np.squeeze(self._interpreter.get_tensor(output_index))
  return tensor

def _postprocess(self, boxes: np.ndarray, classes: np.ndarray,
                 scores: np.ndarray, count: int, image_width: int,
                 image_height: int) -> List[Detection]:
  """Post-process the output of TFLite model into a list of Detection objects.

  Args:
      boxes: Bounding boxes of detected objects from the TFLite model.
      classes: Class index of the detected objects from the TFLite model.
      scores: Confidence scores of the detected objects from the TFLite model.
      count: Number of detected objects from the TFLite model.
      image_width: Width of the input image.
      image_height: Height of the input image.

  Returns:
      A list of Detection objects detected by the TFLite model.
  """
  results = []

  # Parse the model output into a list of Detection entities.
  for i in range(count):
    if scores[i] >= self._options.score_threshold:
      y_min, x_min, y_max, x_max = boxes[i]
      bounding_box = Rect(
          top=int(y_min * image_height),
          left=int(x_min * image_width),
          bottom=int(y_max * image_height),
          right=int(x_max * image_width))
      class_id = int(classes[i])
      category = Category(
          score=scores[i],
          label=self._label_list[class_id],  # 0 is reserved for background
```

```
            index=class_id)
        result = Detection(bounding_box=bounding_box, categories=[category])
        results.append(result)

    # Sort detection results by score ascending
    sorted_results = sorted(
        results,
        key=lambda detection: detection.categories[0].score,
        reverse=True)

    # Filter out detections in deny list
    filtered_results = sorted_results
    if self._options.label_deny_list is not None:
      filtered_results = list(
          filter(
              lambda detection: detection.categories[0].label not in self.
              _options.label_deny_list, filtered_results))

    # Keep only detections in allow list
    if self._options.label_allow_list is not None:
      filtered_results = list(
          filter(
              lambda detection: detection.categories[0].label in self._options.
              label_allow_list, filtered_results))

    # Only return maximum of max_results detection.
    if self._options.max_results > 0:
      result_count = min(len(filtered_results), self._options.max_results)
      filtered_results = filtered_results[:result_count]

    return filtered_results
```

Anexo 4. Código para los sensores en Arduino.

```
const int sensorEntrada = 12;
const int sensorPiston1 = 11;
const int sensorPiston2 = 10;
const int sensorPiston3 = 9;
const int motor12V=2;

String msgPi;


void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);   //iniciar puerto serie
  pinMode(sensorEntrada , INPUT);  //definir pin como entrada
  pinMode(sensorPiston1 , INPUT);  //definir pin como entrada
  pinMode(sensorPiston2 , INPUT);  //definir pin como entrada
```

```
  pinMode(sensorPiston3 , INPUT);  //definir pin como entrada
  pinMode(motor12V,OUTPUT);
}

void loop() {
    //Valor que seran definidos por las entradas digitales
    int valorEntrada,valorPiston1, valorPiston2, valorPiston3 = 0;

    valorEntrada = digitalRead(sensorEntrada);
    valorPiston1 = digitalRead(sensorPiston1);
    valorPiston2 = digitalRead(sensorPiston2);
    valorPiston3 = digitalRead(sensorPiston3);

    readSerialPort();


    if (valorEntrada == LOW) {
        Serial.println("IN");
    }
    else if(valorPiston1 == LOW){
      Serial.println("PT1");
    }
    else if(valorPiston2 == LOW){
      Serial.println("PT2");
    }
    else if(valorPiston3 == LOW){
      Serial.println("PT3");
    }


    if(msgPi == "M12V ON"){
      digitalWrite(motor12V,HIGH);
    }else if(msgPi == "M12V OFF"){
      digitalWrite(motor12V,LOW);
    }
    delay(700);
}



void readSerialPort() {
  msgPi = "";
  if (Serial.available()) {
    delay(10);
    while (Serial.available() > 0) {
      msgPi += (char)Serial.read();
    }
    Serial.flush();
  }
}
```

Anexo 5. Código para los servomotores en Arduino.

```
Servo servo1;
Servo servo2;
Servo servo3;


const int delayServ = 15;
const int servoPin1 = 12;
const int servoPin2 = 11;
const int servoPin3 = 10;

String msgPi;



void setup() {
  Serial.begin(9600);
  servo1.attach(servoPin1);
  servo2.attach(servoPin2);
  servo3.attach(servoPin3);
}

void loop() {

  readSerialPort();



  if(msgPi == "PT1 ON"){
    moveDegree(servo1);
  }
  else if(msgPi == "PT2 ON"){
    moveDegree(servo2);
  }
  else if(msgPi == "PT3 ON"){
    moveDegree(servo3);
  }
  delay(100);
}


void moveDegree(Servo servo){

  for (int inicio = 160; inicio <= 180; inicio += 1) {
    Serial.println(inicio);
    servo.write(inicio);
    delay(15);
  }

  for (int inicio = 180; inicio <= 160; inicio -= 1) {
    Serial.println(inicio);
    servo.write(inicio);
```

```
    delay(15);
  }
}

void readSerialPort() {
  msgPi = "";
  if (Serial.available()) {
    //Serial.println(">:c");
    delay(10);
    while (Serial.available() > 0) {
      msgPi += (char)Serial.read();
      //Serial.println(">:c 1");
    }
    Serial.flush();
  }
}


String getValue(String data, char separator, int index){
  int found =0;
  int strIndex[] = {0,-1};
  int maxIndex = data.length() -1;
  for(int i=0; i<= maxIndex && found <= index; i++){
    if(data.charAt(i) ==separator  || i == maxIndex){
      found ++;
      strIndex[0] = strIndex[1] +1;
      strIndex[1] = (i == maxIndex) ? i+1: i;
    }
  }
  return found > index? data.substring(strIndex[0],strIndex[1]) : "";
}
```