UNIVERSIDAD RICARDO PALMA

FACULTAD DE INGENIERÍA

ESCUELA PROFESIONAL DE INGENIERÍA MECATRÓNICA

Diseño de robot móvil para la detección del uso de mascarillas y sensado de temperatura corporal utilizando visión artificial en lenguaje Python y Raspberry Pi.

TESIS

Para optar el título profesional de Ingeniero Mecatrónico

AUTORES

Alvarado Carlos, Manuel Antonio
ORCID: 0000-0003-2378-0188

Vilca Pari, Aarón Arnaldo
ORCID: 0000-0001-5041-9596

ASESOR

Sotelo Valer, Freedy
ORCID: 0000-0003-3079-2857

Lima, Perú

2022

Metadatos Complementarios

Datos del autor(es)

Alvarado Carlos, Manuel Antonio
DNI: 72030436
Vilca Pari, Aarón Arnaldo
DNI: 74607866

Datos de asesor

Sotelo Valer, Freedy
DNI: 25804755

Datos del jurado

JURADO 1
Mandujano Neyra, Demetrio Hugo
DNI: 07601347
ORCID: 0000-0002-3096-5626

JURADO 2
Rivas León, Javier Hipólito
DNI: 10250991
ORCID: 0000-0002-8365-4346

JURADO 3
Castro Salgero, Robert Gerardo
DNI: 06756101
ORCID: 0000-0001-9909-3435

Datos de la investigación

Campo del conocimiento OCDE: 02.11.02
Código del Programa: 712046

DEDICATORIA

Dedico esta investigación a mis padres y hermano, quienes me apoyan incondicionalmente durante mis años de estudio y me dieron la fuerza para seguir adelante.

Manuel Antonio Alvarado Carlos

Dedico esta investigación a Dios, a mis padres, abuelos, hermanos, compañeros y amigos quienes me brindaron consejos, apoyo y conocimientos a lo largo de mis años de estudio.

Aarón Arnaldo Vilca Pari

AGRADECIMIENTO

Nuestro sincero agradecimiento a los docentes de la escuela de ingeniería mecatrónica, por habernos brindado los conocimientos necesarios para desarrollar la presente investigación.

Manuel Alvarado
Aarón Arnaldo Vilca Pari

ÍNDICE

INDICE DE TABLAS

INDICE DE FIGURAS

RESUMEN

La presente investigación tuvo como objetivo diseñar un robot móvil para la detección del uso de mascarilla y sensado de temperatura corporal utilizando visión artificial en leguaje Python y Raspberry Pi. Para dicha finalidad, se plantearon diseños con respecto a la parte mecánica, eléctrica, electrónica y se desarrolló un programa para la detección de mascarillas y temperatura corporal de las personas que ingresen a una zona de gran tránsito. Esto responde a la problemática referida al contagio de enfermedades por vía respiratoria, como el COVID-19, que se da comúnmente en áreas cerradas debido a la falta de equipos automatizados que identifique el principal síntoma de la enfermedad y la negligencia de las personas en no portar sus respectivas mascarillas.

Para el sistema mecánico se estableció que la movilidad del robot se diera por medio de tres ruedas, dos de ellas conectadas a dos motores y una sola rueda loca para facilitar los giros en su desplazamiento sobre una superficie plana. El material escogido para la carcasa fue Acrilonitrilo butadieno estireno (ABS) para brindarle una mayor ligereza.

Para el sistema eléctrico, se contó con unas baterías de lipo capaz de energizar todo el sistema luego de un tiempo de carga determinada, así como unos reguladores de voltaje para los motores. Para el sistema electrónico, se eligió un sensor de temperatura, un sensor infrarrojo como seguidor de línea y leds para indicar un sendero determinado. En la programación, se empleó el lenguaje Python para trabajar los sistemas de reconocimiento mediante visión artificial por medio de una cámara web y la plataforma de Arduino para la movilidad del robot de manera automatizada.

Como resultado de la investigación, el diseño resultante obtuvo un peso de 59.74 kg y 8 horas de duración para la batería para un torque de 200 kg.cm y una velocidad de 60 RPM. Además, se consiguió automatizar los procesos de detección de mascarilla y temperatura para evitar el contacto o cercanía del personal ante un posible cliente contagiado en la mayor medida posible. Por lo tanto, conseguirá que los casos de personas con síntomas no ingresen al área común junto a otros clientes, logrando mantener un ambiente más seguro y salubre para ellos.

Palabras claves: Raspberry Pi, Python, Robot Móvil, Visión Artificial

ABSTRACT

The objective of this research was to design a mobile robot for the detection of the use of a mask and body temperature sensing using artificial vision in Python and Raspberry Pi language. For this purpose, designs were proposed regarding the mechanical, electrical, and electronic parts, and a program was developed for the detection of masks and body temperature of people entering a high-traffic area. This responds to the problem related to the spread of respiratory diseases, such as COVID-19, which commonly occurs in closed areas due to the lack of automated equipment that identifies the main symptom of the disease and the negligence of people in not wear their respective masks.

For the mechanical system, it was established that the mobility of the robot would be through three wheels, two of them connected to two motors and a single idler wheel to facilitate the turns in its movement on a flat surface. The material chosen for the casing was Acrylonitrile Butadiene Styrene (ABS) to provide greater lightness.

For the electrical system, there were lipo batteries capable of energizing the entire system after a certain charging time, as well as voltage regulators for the motors. For the electronic system, a temperature sensor, an infrared sensor as a line follower and LEDs were chosen to indicate a certain path. In the programming, the Python language was used to work the recognition systems through artificial vision through a webcam and the Arduino platform for the mobility of the robot in an automated way.

As a result of the investigation, the resulting design obtained a weight of 59.74 kg and 8 hours of battery life for a torque of 200 kg.cm and a speed of 60 RPM. In addition, it was possible to automate the mask and temperature detection processes to avoid contact or closeness of the staff to a possible infected client to the greatest extent possible. Therefore, it will ensure that cases of people with symptoms do not enter the common area with other clients, thus maintaining a safer and healthier environment for them.

Key words: Raspberry Pi, Python, Mobile Robot, Artificial Vision

INTRODUCCIÓN

La presente investigación se realizó en respuesta a la adopción de medidas preventivas ante el contagio y propagación a nivel mundial de la enfermedad denominada COVID-19. Debido a la necesidad de reactivar el sector de trabajo enfocado a los servicios de transporte nacional e internacional, el uso de mascarillas dentro de estos ambientes y una declaración jurada son solo algunas de las disposiciones dadas por el estado, mas no son determinantes para evitar que algún pasajero lleve consigo el virus, poniendo en riesgo la salud de las otras personas con las que compartirá espacio. Para que estos servicios sean más seguros y su reactivación se logre por completo, se debe contar con equipos de detección de síntomas que identifiquen a los posibles casos antes de que los clientes aborden. Esta investigación brinda como solución el diseño de un robot móvil para la detección del uso de mascarillas y sensado de temperatura corporal utilizando visión artificial en lenguaje Python y Raspberry Pi. La presente investigación está dividida en cuatro capítulos.

En el primer capítulo se expone el planteamiento del problema, se formula el problema general y los problemas específicos, así como la importancia de la investigación, justificación, objetivo general y específicos. En el segundo capítulo se muestran los antecedentes internacionales y nacionales relacionados con la investigación, así como las bases teóricas que explican los términos relacionados al trabajo de investigación, de manera clara y concisa. En el tercer capítulo se describe el desarrollo que se llevó a cabo para el sistema mecánico, eléctrico y electrónico, además del desarrollo del programa que automatizará al robot móvil. En el cuarto capítulo se detallan los resultados obtenidos en el trabajo de investigación. Finalmente, se redactan las conclusiones en función a los objetivos propuestos y las recomendaciones que se deben tomar en cuenta para futuras líneas de investigación que puedan generar. De igual manera, se colocaron anexos para ampliar la información presentada.

CAPÍTULO I: PLANTEAMIENTO Y DELIMITACIÓN DEL PROBLEMA

1.1 Formulación y Delimitación del Problema

La pandemia del COVID-19 y su alto impacto en la salud pública ha generado una serie de medidas sanitarias a nivel mundial para prevenir su avance. Una de las acciones de emergencia fue el cierre temporal de las fronteras debido a que personas de diferentes regiones viajaban sin saber que eran portadores del virus, ocasionando una propagación acelerada del mismo. Esto significó la clausura de servicios de aerolíneas y cancelación de viajes al exterior del país con la finalidad de mantener una cuarentena por tiempo indefinido, afectando severamente el sector turístico para evitar la propagación del virus.

En Perú, el turismo se encontraba en constante crecimiento hasta el 2019, según el Ministerio de Comercio Exterior y Turismo (MINCETUR). El aporte de este sector al país explicaba el 4% del PBI, asimismo estimaba en el primer trimestre de ese año superar las 4 millones 800 mil visitas y alcanzar el récord de más de US$ 5 mil millones respecto al ingreso de divisas por turismo receptivo. Dichas estimaciones lograron coincidir con los resultados proporcionados por el Banco Central de Reserva del Perú, como se muestran en la Tabla 1 y la Figura 1. Sin embargo, la pandemia afectó el crecimiento de este sector que pasó a recibir US$ 850 mil entre enero y octubre de 2020, lo cual significó una caída del 76,8% casi uniforme en todos los mercados de origen según MINCETUR.

Tabla N° 1

Indicadores del Sector Turismo entre los años 2005-2019

Año	Número de Turistas		Ingreso de Divisas	Egreso de Divisas	Divisas Per cápita		Balance (Saldo)		
	Entrada 1/	Salida 2/	(Mill. US$)	(Mill. US$)	Ingreso (US$)	Egreso (US$)	Turistas	Divisas (Mill. US$)	Per cápita (US$)
2005	1 570 566	1 841 235	1 438	969	916	526	- 270 669	469	389
2006	1 720 746	1 857 083	1 775	1 047	1 032	564	- 136 337	728	468
2007	1 916 400	1 914 886	2 007	1 243	1 047	649	1 514	764	398
2008	2 057 620	1 913 029	2 396	1 432	1 164	748	144 591	964	416
2009	2 139 961	1 890 508	2 440	1 403	1 140	742	249 453	1 036	398
2010	2 299 187	2 057 793	2 475	1 647	1 077	801	241 394	828	276
2011	2 597 803	2 131 899	2 814	1 768	1 083	829	465 904	1 046	254
2012	2 845 623	2 296 131	3 073	1 900	1 080	827	549 492	1 174	253
2013	3 163 639	2 363 879	3 916	2 105	1 238	890	799 760	1 811	347
2014	3 214 934	2 441 705	3 908	2 119	1 215	868	773 229	1 789	348
2015	3 455 709	2 594 881	4 140	2 539	1 198	979	860 828	1 601	220
2016	3 744 461	2 751 357	4 288	2 700	1 145	981	993 104	1 588	164
2017	4 032 339	2 875 312	4 439	2 893	1 101	1 006	1 157 027	1 546	95
2018	4 419 430	3 078 377	4 505	3 352	1 019	1 089	1 341 053	1 153	-70
2019	4 371 787	3 275 088	4 784	3 671	1 094	1 121	1 096 699	1 113	-26

Fuente: INEI con la Superintendencia Nacional de Migraciones del Perú

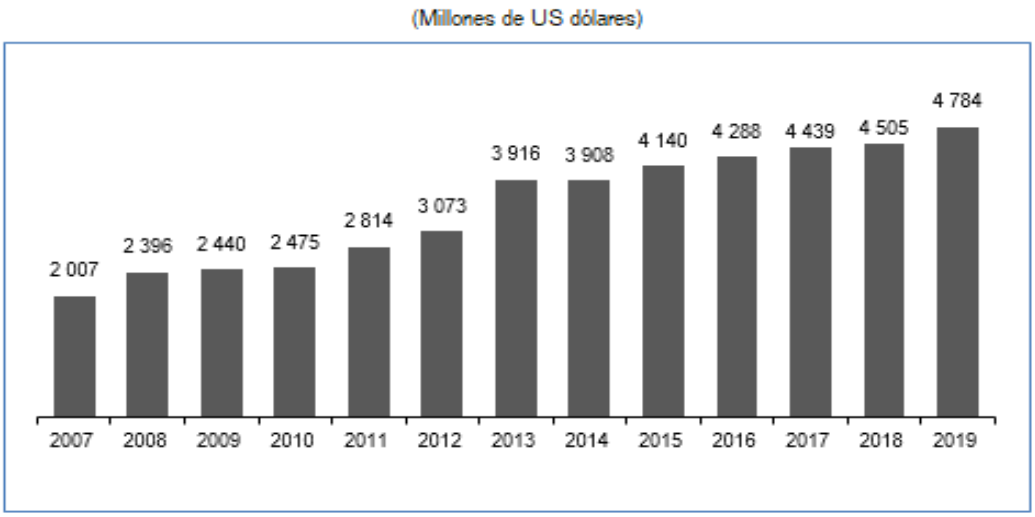


Figura 1: Ingreso de Divisas por Turismo entre los años 2007 y 2019
INEI en colaboración con el Banco Central de Reserva del Perú (BCR).

En base a estos datos, hubo una pronta necesidad de reanudar los servicios de viajes nacionales e internacionales luego de que el gobierno permitiera la reactivación de estas actividades. El aeropuerto peruano Jorge Chávez inició una serie de protocolos para salvaguardar la salud de sus clientes; entre estos, una declaración jurada de salud por parte de los pasajeros y el uso obligatorio de mascarillas durante todo el vuelo, tal como se puede apreciar en la Figura 2. Sin embargo, muchos de estos métodos terminaron por ser procesos de prevención repetitivos y la falta de equipos que determinen con mayor precisión los síntomas de la enfermedad se hicieron evidentes.

Figura 2:Medidas para el ingreso y tránsito de pasajeros en los aeropuertos.
Fuente: Aeropuerto Internacional Jorge Chávez, Perú.

Además, se incluyó el uso de termómetros infrarrojos por parte del personal de seguridad en diferentes puntos del embarque y desembarque. No obstante, esta acción expone un posible contagio a los trabajadores por recurrir a un contacto directo y continuo con los pasajeros. Esta última causal se ve reflejado en el comportamiento de algunos transeúntes pese a las normas impuestas para esta nueva normalidad, que aun así suelen bajarse la mascarilla o deciden no llevarla porque les genera malestar, desconociendo que, de ese modo, ponen en riesgo la salud de las personas que tienen a su alrededor. Aunque las recientes normas establecidas por el gobierno mediante el Decreto Supremo N° 041-2022-PCM, con fecha de 1 de mayo de 2022, ha establecido el uso opcional de mascarillas en espacios abiertos en el país, todavía se exige el uso en lugares cerrados y durante los viajes. Es por este motivo que se busca una forma de automatizar los procesos de medición de temperatura para realizar dicha labor de manera remota, ágil y con la menor intervención de personas en lo posible.

En consecuencia, debido a que aún hay un riesgo de exposición y contagio del COVID-19 en áreas tan transitadas como los aeropuertos, se justifica el uso de un robot móvil para la detección de personas sin mascarillas y sensado de temperatura corporal utilizando visión artificial.

1.1.1 Problema General

¿Cómo diseñar un robot móvil para la detección del uso de mascarilla y sensado de temperatura corporal utilizando visión artificial en lenguaje Python y Raspberry Pi?

1.1.2 Problemas Específicos

a) ¿Cómo diseñar el sistema mecánico del robot móvil?

b) ¿Cómo definir las características técnicas de la parte electrónica y del suministro eléctrico del robot móvil?

c) ¿Cómo desarrollar la programación para la etapa de visión artificial usando el lenguaje Python con el micro computador Raspberry Pi?

1.2 Importancia y justificación del estudio

1.2.1 Importancia de la Investigación

La importancia de esta tesis sobre el diseño de un robot móvil para la identificación del uso de mascarilla y sensado de temperatura corporal es plantear una solución tecnológica para evitar el contagio en lugares cerrados con afluencia alta de personas.

1.2.2 Justificaciones de la Tesis

A) Justificación Tecnológica

La justificación tecnológica del presente proyecto implica el automatizado del proceso de detección de mascarillas y temperatura de los pasajeros que abordarán un vuelo mediante visión artificial usando lenguaje Python, el diseño de una estructura móvil mediante SolidWorks, el microcomputador Raspberry Pi, baterías de Lipo 3,7 V - 60000 mAh y motorreductores de metal 25d 6-12v DC para la movilidad del robot.

B) Justificación Económica

La justificación económica del presente proyecto va referida a la reducción de costos que significará el diseño del robot móvil, teniendo en cuenta los materiales para su estructura mecánica, sus implementos eléctricos para su movilidad y los elementos electrónicos para la detección de mascarilla y temperatura. Al tratarse de un proyecto nacional, se podrá producir sin necesidad de contar con empresas extranjeras para su compra y mantenimiento, promoviendo la investigación mecatrónica dirigida a la salud.

C) Justificación Social

La justificación social de la presente tesis es diseñar un sistema mecatrónico que cumpla con los requerimientos técnicos mínimos para la prevención y disminución de la propagación del COVID-19 en espacios reducidos con gran aforo, tales como los aviones. También, se asegura el uso obligatorio de las

mascarillas que aún sigue vigente en territorio nacional y se avisa ante un posible caso al pasajero que se le reconoce el principal síntoma.

1.3 Objetivos

1.3.1 Objetivo General

Diseñar un robot móvil para la detección del uso de mascarilla y sensado de temperatura corporal utilizando visión artificial en lenguaje Python y Raspberry Pi.

1.3.2 Objetivos Específicos

a) Diseñar el sistema mecánico del robot móvil.
b) Determinar las características técnicas de la parte electrónica y del suministro eléctrico del robot móvil.
c) Desarrollar la programación para la etapa de visión artificial usando el lenguaje Python con el micro computador Raspberry Pi.

CAPÍTULO II: MARCO TEÓRICO

En el presente capítulo se describe el marco teórico de la investigación donde se menciona los antecedentes más relevantes para el desarrollo de la investigación. También, se presentan las bases teóricas que sustentan y complementan los tópicos que se utilizaron.

2.1 Trabajos relacionados con el tema

A continuación, se presentan los estudios más importantes y necesarios para esta investigación, basados en tesis referentes al desarrollo de sistemas de control y supervisión.

2.1.1 Antecedentes Internacionales

Tipanquiza (2021), en su trabajo de investigación para obtener el título de Ingeniero en Electrónica y Comunicaciones titulado "Dispositivo remoto para desinfección de entornos cerrados mediante luz ultravioleta" de la Universidad Técnica de Ambato, Ecuador. Se propone desarrollar el prototipo de un dispositivo remoto para la desinfección de entornos cerrados mediante luz ultravioleta, analizando los procesos manuales principales y automatizados existentes. El trabajo es de enfoque experimental porque empleó los conocimientos que adquirió a través del estudio necesario para implementar el sistema controlable de desinfección que usaría luz ultravioleta. Entre sus conclusiones más importantes manifiesta la importancia del desarrollo tecnológico en cada ámbito de la vida diaria. Es por ello que la tecnología adecuada posee gran o menor influencia en los detalles de la construcción de un prototipo, por lo que una exhaustiva comparación de dichos elementos logra definir aquellos que mejor se adaptan a los requisitos iniciales. En consecuencia, afirma que se tiene que realizar un previo estudio para aplicar la tecnología de acuerdo a la problemática existente, lo cual mejorará el planteamiento de soluciones mediante la tecnología.

Monge, Fernández y De Esteban (2019) en su estudio "Aeropuertos inteligente: Aceptación de la tecnología por parte de los pasajeros", se planteó como objetivo

analizar la percepción positiva hacia la tecnología que convierten a un aeropuerto en España como un lugar inteligente, según sus pasajeros. Resumiendo, la investigación demostró que el empleo de tecnología en autoservicio dentro de los procesos de creación de etiquetas para equipajes, su embarque y su facturación consigue una mayor satisfacción al pasajero. Además, comprobó que la edad de los pasajeros influye en su deseo de operar nuevas tecnologías en estos procesos, como el uso de videojuegos y el portar pasaportes biométricos. La investigación es de enfoque cuantitativo. La representatividad posee una muestra de 400 personas nacidas en España que viajaron por avión en los dos últimos años, mediante una técnica estadística descriptiva (Chi Cuadrado), y mediante el método muestreo aleatorio simple. Entre sus conclusiones más importantes indican que el turismo en España tiene una importancia creciente y es una parte fundamental de la actividad económica. El Aeropuerto se configura como una parte fundamental de la Ciudad Inteligente. Internet de las Cosas, la Comunicación de Campo Cercano, la tecnología de Código de Barras, la Identificación por Radiofrecuencia, las tecnologías de geolocalización, la Realidad inmersiva, los sistemas biométricos, la Inteligencia Artificial y la Robótica y el Blockchain o Tecnología de Bloques son tecnologías que han permitido la creación de distintos dispositivos que ayudan al pasajero en todas las etapas de su viaje, facilitan su paso por el aeropuerto y lo convierten en "Aeropuerto Inteligente". En consecuencia, afirma que siendo los aeropuertos una parte fundamental de la actividad económica, es importante la aplicación de tecnologías las cuales permitan a los pasajeros mejorar su experiencia en aeropuertos dándoles seguridad y confort, entre otros.

Tenorio (2017). En su tesis de pregrado para optar al título de ingeniero mecatrónico, titulado "Diseño e implementación de un sistema para el mapeo y navegación de un robot móvil" de la Universidad Autónoma de Occidente, Colombia. El trabajo de investigación es de enfoque literario y experimental porque se desarrolló en base a la recolección y comparación de características de los sistemas existente usando como fuente principal libros, proyectos de grado y artículos de revistas o formatos científicos, para desarrollar el proceso de diseño e implementación. En el presente estudio se muestra el desarrollo del diseño e implementación de un sistema de mapeo y navegación autónoma para

un robot móvil existente en UAO, con capacidad de ser implementado en diferentes plataformas con la misma configuración. En cuyas conclusiones más importantes manifiesta que el sistema se encuentra en capacidad de corregir localización del robot en el mapa cuando se presentan problemas debido a errores de la odometría. El software libre ROS es una herramienta que permite, entre otras cosas, la implementación de sistemas de navegación de plataformas móviles, proporcionando útiles herramientas de simulación y visualización que facilitan el desarrollo de estos sistemas.

2.1.2 Antecedentes Nacionales

Matencio (2021), En su trabajo de investigación para obtener el grado de bachiller en ingeniería Mecatrónica titulado “Diseño de un sistema robótico móvil para guiar e informar a pacientes en centros médicos” de la Pontificia Universidad Católica, Lima, Perú. Al diseñar una propuesta de sistema de asistencial móvil para informar y dirigir a los pacientes a sus clínicas hospitalarias, el trabajo de investigación fue de enfoque empírico y sistemático. Se utilizó el método VDI 2221 para desarrollar y diseñar sistemas y productos de ingeniería. Entre sus conclusiones más importantes, está que, dada la problemática, el uso de robots resulta muy útil para evitar el contagio de enfermedades infecciosas entre personas. Por otro lado, ayudan a mejorar la calidad del servicio en sistemas de salud con poca información y señales inadecuadas. Por ello, afirma que el uso de robots puede prevenir la propagación de enfermedades infecciosas a pequeña escala.

Echenique y Villalobos (2021), en su trabajo de investigación para la licenciatura en Administración y Hotelería titulado “Ventajas y Desventajas del Uso de Robots como Herramientas para Distancias Físicas en Hoteles” de la Universidad de Ciencias Aplicadas del Perú, Lima, Perú. Donde propone estudiar las ventajas y desventajas del uso de robots como herramienta de distanciamiento físico en los hoteles. Este trabajo se basa en el enfoque de la crítica literaria, ya que les permite explorar la literatura sobre el tema en estudio para justificar una hipótesis, estudiar la relación entre variables e incluso

encontrar los vacíos teóricos que existen, utilizando el tiempo como enfoque analítico integrado, permitiéndoles indagar en el tema emergente del uso de la robótica como herramienta de análisis físico y una propuesta de análisis de sus ventajas y desventajas. Como conclusión más importante, resalta que, según diversos estudios, una de las principales ventajas de utilizar robots de servicio es reducir el contacto físico entre las personas y al mismo tiempo reducir el miedo al contagio. Del mismo modo, el uso de robots en los hoteles aumenta la productividad, ya que pueden realizar operaciones complejas en menos tiempo; es decir, hacen el trabajo de manera más eficiente que los humanos. Por lo tanto, menciona que el uso de robots en la interacción humana reduce el contacto físico, evita infecciones y aumenta la productividad en operaciones repetitivas.

Inciso (2021). En su tesis de pregrado para obtener el título de Ingeniero Electrónico, titulado "Desarrollo de un Sistema de Control de Acceso con Imagen y Temperatura de Trabajadores de la Empresa SEELDOIN E.I.R.L." de la Universidad Tecnológica del Perú, Lima. La presente tesis de grado propone el desarrollo de un sistema tecnológico de reconocimiento facial de alta precisión y medición de temperatura para reducir el contagio del COVID-19 entre los empleados de una determinada empresa. Para el reconocimiento de rostro se utilizó la plataforma de código libre Python con las librerías Open CV y Face Recognition por medio del hardware Raspberry Pi. La muestra que empleó fue de 10 trabajadores voluntarios, cuyos rostros fueron capturados en cámara una vez al día por diez días. Para el sensado de temperatura se utilizó el sensor infrarrojo MLX90614. Entre sus conclusiones reconoció las fallas del reconocimiento facial a causa del uso de accesorios como lentillas o mascarillas quirúrgicas, dándole al sistema una precisión del 98% según los valores de la matriz de confusión. Además, si bien asegura que el sensor infrarrojo consiguió detectar la temperatura a distancia, no especificó la distancia máxima de detección o su precisión.

2.2 Estructura científica y teórica que sustenta el estudio.

Para la comprensión del desarrollo del presente proyecto, es necesario incluir la información teórica y técnica que abarca los temas del estudio para el diseño del robot móvil con visión artificial y sensado de temperatura.

2.2.1 Covid-19

Se define como una enfermedad contagiosa causada por el virus SARS-CoV-2. Sus síntomas en los pacientes pueden presentarse de menor intensidad a una moderada y actualmente se pueden curar sin necesidad de algún tratamiento especial.

Se propaga por medio de la boca o la nariz de una persona con el virus, la cual expulsa las partículas en saliva al momento de toser, hablar o respirar. El contagio ocurre al respirar cerca de alguien infectado o tocando una superficie donde las partículas aterrizaron y, acto seguido, tocarse los ojos, la nariz o la boca. Esta enfermedad es más fácil de contraer en espacios interiores o en aglomeraciones de personas. (OMS, 2021)

2.2.2 Síntomas del COVID-19

Existen diversos síntomas y estos cambian dependiendo del paciente. Como se explicó previamente, estos pueden ser leves o moderados, por lo que se clasificará a continuación cada uno de ellos. (OMS, 2021).

Los síntomas más comunes son:

- Desgaste físico.
- Tos.
- Fiebre.
- Ausencia del gusto u olfato.

Los síntomas menos recurrentes son:

- Dolor en la garganta y cabeza.
- Diarrea.
- Erupciones en la piel.

- Descoloramiento cutáneo en manos y pies.
- Irritación ocular.

Los síntomas más graves son:

- Disnea o problemas respiratorios.
- Mareos.
- Parálisis corporal.
- Fuertes dolores a la altura del pecho.

2.2.3 Fiebre

Es el aumento temporal de temperatura en el cuerpo. La fiebre no se considera una enfermedad, generalmente es una señal de que el cuerpo está tratando de combatir una enfermedad o infección. Para los adultos, la fiebre puede ser alarmante, pero no hay necesidad de preocuparse a menos que la temperatura suba a 39,4 grados centígrados (103 grados Fahrenheit) o más.

2.2.4 Robot

Es una un sistema mecatrónico controlada por computadora y programada para manipular objetos, moverse, realizar funciones determinadas e interactuar con su entorno. Su principal objetivo es el de reemplazar al ser humano en actividades repetitivas, complicadas o peligrosas de manera rápida, precisa y segura.

Según la Organización Internacional para la Estandarización (ISO), la definición de robot es “Manipulador multifuncional reprogramable con varios grados de libertad, controlado automáticamente, que puede estar fijo en un sitio o moverse; y que está diseñado para manipular materiales, piezas, herramientas o dispositivos especiales según trayectorias variables programadas para realizar diversas tareas o trabajos.”

El Instituto de Robótica de América define robot como lo siguiente: “Dispositivo multifuncional reprogramable diseñado para manipular y/o transportar material a través de movimientos programados para la realización de tareas variadas.” (Torres, 2012, p. 12-13)

A) Cinemática de un Robot

Es el estudio de los movimientos de un robot. Calcula posición, velocidad y aceleración en un punto de interés del robot a partir de las velocidades de control (servomotores o motores). Las ecuaciones cinemáticas de un robot son usadas en juegos de computadoras, simulación e implementadas en robots reales.

B) Modelo matemático del robot

Un modelo matemático es una expresión que permite representar el comportamiento de un proceso físico en función de las variables que intervienen en dicho proceso. En el ámbito de la robótica, los modelos matemáticos que existen para representar un mismo robot son muy variados.

Cada modelo matemático puede representar diversas propiedades cinemáticas y dinámicas. Por lo tanto, cada modelo robótico tendrá una utilidad diferente de acuerdo a las propiedades o comportamientos que el usuario final necesita observar. En el siguiente diagrama se muestra un modelo de un sistema lineal o de primer grado; es decir, una ecuación diferencial donde la derivada de mayor orden tiene exponente igual a 1.

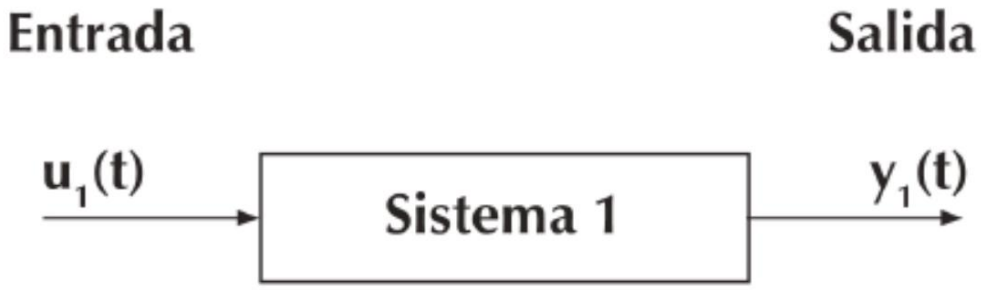


Figura 3: Diagrama – Modelo de un sistema lineal
Fuente: "Análisis de sistemas lineales", Rojas, A. (2014)

u(t): Variable de Entrada
y(t): Variable de Salida
t: Variable Tiempo

Para el modelamiento matemático de un robot, se tiene que tener presente siempre un sistema de referencia (Ow). Este sistema de referencia es fijo en

posición y orientación, para cada sistema independientemente se puede tomar por conveniencia cada eje de acuerdo a las necesidades. Para la siguiente figura se toma el eje Y positivo como la parte frontal. Por lo tanto, el lado derecho estaría el eje X, y el eje Z positivo apuntando hacia arriba.

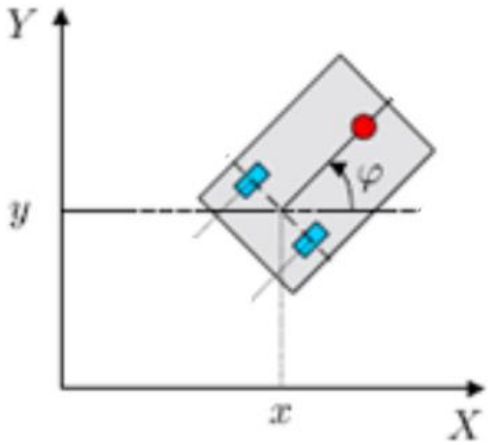


Figura 4: Plano cartesiano XY del posicionamiento de una figura

Fuente: Roboticos (2021)

De igual manera es posible fijar una referencia en el eje de robot (Ob); pero, con respecto al anterior mencionado (Ow), es móvil tanto en posición como orientación. Por lo tanto, es importante escoger los ejes X, Y y Z para estimar el correcto sistema de referencia. La siguiente figura muestra en el plano cartesiano el modelo cinemático de un robot diferencia.

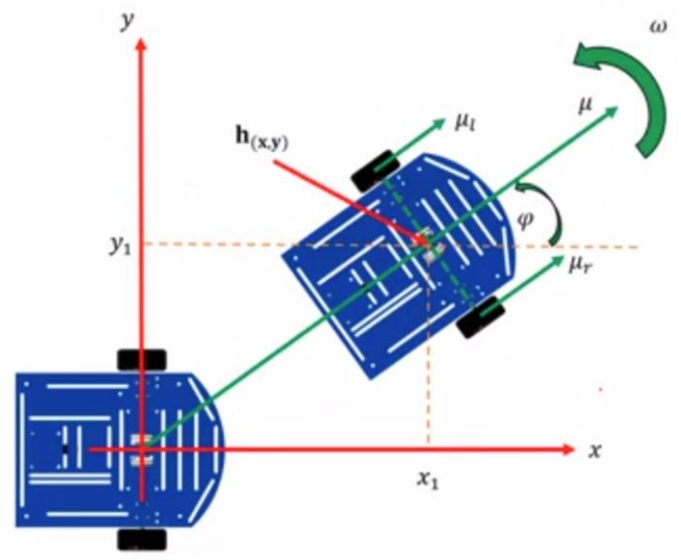


Figura 5: Plano Cartesiano XY del movimiento de una figura.

Fuente: Robotics (2021)

$$h_x = x_1$$
$$h_y = y_1$$
$$\dot{h}_x = \mu cos\varphi$$
$$\dot{h}_y = \mu sin\varphi$$
$$\dot{\varphi} = \omega$$

$$\begin{bmatrix} \dot{h}_x \\ \dot{h}_y \\ \dot{\varphi} \end{bmatrix} = \begin{bmatrix} cos\varphi & 0 \\ sin\varphi & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mu \\ \omega \end{bmatrix}$$

$$\dot{h} = j \,.\, \dot{q} \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (1)$$

C) Tipos de robots

Hay muchos tipos de robots con diferentes estructuras mecánicas y de ingeniería que definen sus aplicaciones y funciones. Sin embargo, clasificarse ampliamente como se muestra en la Tabla 2. (Reyes ,2011)

Tabla N° 2

Clasificación general de Robots

Móviles	Terrestres: ruedas, patas Submarinos, aéreo-espaciales
Humanoides	Diseño complejo
Industriales	Brazos mecánicos Robots manipuladores

Extraído del libro “Control de Robots Manipuladores” de Fernando

Fuente: Reyes Cortés (2011)

D) Robots Móviles

La creación de este tipo de robots ha ido en aumento en tiempos modernos, especialmente en la robótica dirigida a servicios y en el sector industrial gracias al auge de la automatización. Su composición dependerá de las funciones que el usuario requiera para la movilidad, entre las que se encuentran:

- Correr
- Caminar
- Saltar

- Rodar
- Deslizarse
- Volar
- Nadar

La manera en que se diseñan sus sistemas de locomoción está inspirada en seres o elementos de los ecosistemas animales, tales como alas, patas, aletas, etc.; con excepción de aquellos que poseen ruedas, ya que es un invento del ser humano que permite el desplazamiento entornos planos de manera eficiente. (Vásquez, 2015, p. 61)

E) Robots móviles con patas

Los robots con patas emulan la locomoción de los humanos y animales terrestres (como pueden ser los insectos). La pata, que puede incluir varios GDL, debe ser capaz de sostener parte del peso total del robot, y en muchos robots debe ser capaz de levantar y bajar el robot.

Las principales ventajas de estos robots son su capacidad de adaptación y la maniobrabilidad en terrenos difíciles.
Las principales desventajas son la energía necesaria para realizar un movimiento y la alta complejidad mecánica. (Vásquez, 2015, p. 61-62)

F) Robots móviles con ruedas

La rueda, gracias a su forma y composición relativamente simple, es actualmente el mecanismo de movimiento más usado y popular, pudiendo encontrarlo en vehículos comunes como los autos y también en robots móviles que requieran ir de un punto a otro para cumplir con sus funciones. (Vásquez, 2015, p. 66)

2.2.5 Clasificación de robots móviles según su configuración

A) Robot con accionamiento diferencial

Típicamente una plataforma móvil de tracción diferencial cuenta con dos pares de ruedas: dos ruedas de tracción que tienen acoplados dos motores DC y dos

ruedas de estabilización que mantienen el balance del vehículo.
Si la velocidad en ambos motores es la misma (en magnitud y sentido), el robot se mueve de forma lineal. Si la magnitud de la velocidad de ambos motores es la misma, pero con sentidos opuestos, el robot girara sobre sí mismo. En otro caso el robot se traslada y cambia su orientación con respecto al eje coordenado.

La traslación y rotación de este tipo de plataformas diferenciales se determinan por el movimiento independiente de cada una de las ruedas de tracción.
Se muestra el siguiente modelo en el plano cartesiano, el cual busca una relación entre las velocidades ambas ruedas con las velocidades lineal y angular del robot.

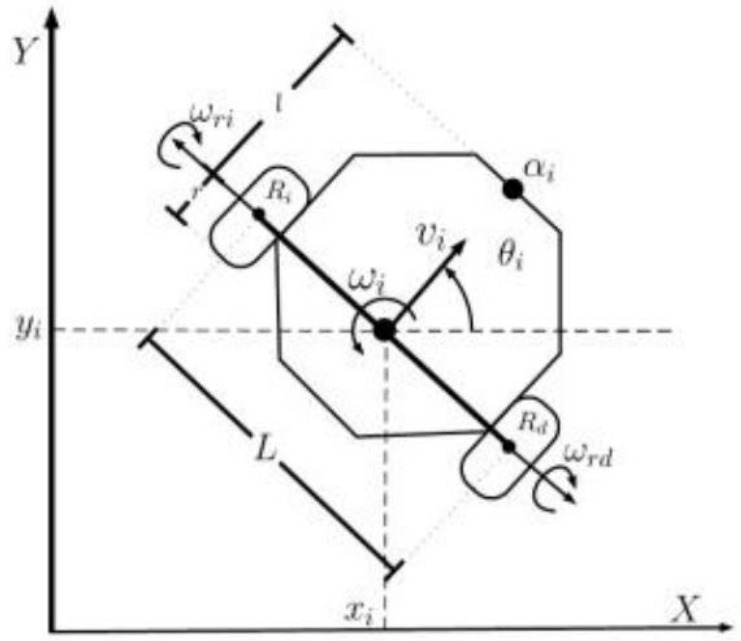


Figura 6:Robot de Accionamiento Diferencial
Fuente: López (2017)

Siendo:

• ωrd: la velocidad de rueda derecha
• ωri: la velocidad de rueda izquierda
• r: radio de las ruedas
• L: distancia entre ambas ruedas
• vi: velocidad lineal total del robot
• ωi: velocidad angular total del robot

$$\begin{bmatrix} wrd \\ wri \end{bmatrix} = \frac{1}{r}\begin{bmatrix} 1 & \frac{L}{2} \\ 1 & \frac{-L}{2} \end{bmatrix}\begin{bmatrix} vi \\ wi \end{bmatrix} \quad \dots (2)$$

B) Robot con accionamiento omnidireccional

La tecnología omnidireccional de las ruedas permite el desplazamiento en cualquier dirección. Estos sistemas de tracción están formados por ruedas fijas con rodillos montados en un ángulo de 45 grados en el perímetro de cada rueda, lo que permite que sus ruedas se mueven con total independencia unas de otras. De esta forma, el vehículo no solo puede desplazarse hacia delante y hacia atrás, sino que también puede moverse en diagonal y en círculos. La rueda completa se acciona mediante un motor eléctrico.

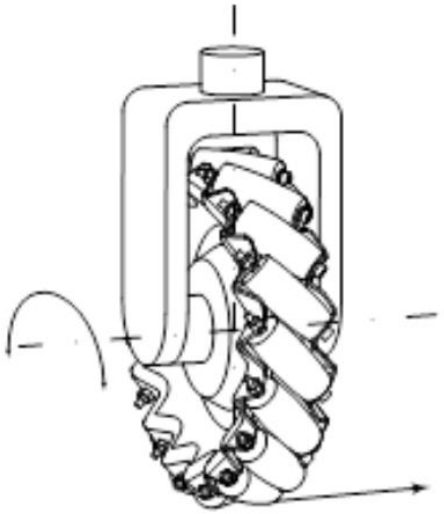


Figura 7:Rueda sueca 45°

Fuente: Rodríguez (2011)

Se muestra el siguiente modelo en el plano cartesiano, el cual busca una relación entre las velocidades las 4 ruedas con las velocidades lineal y angular del robot.

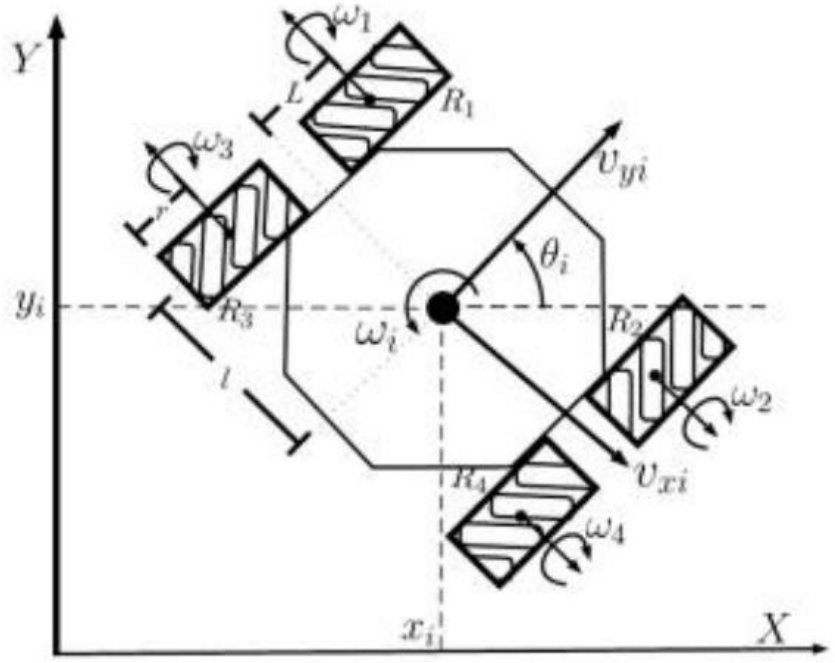


Figura 8:Robot omni-direccional 4 ruedas

Fuente: López G (2017)

Siendo:

• vxi: Velocidad lineal respecto al eje x del chasis

• vyi: Velocidad lineal respecto al eje y del chasis

• ωi: Velocidad angular total del motor

• ω1: Velocidad angular motor 1

• ω2: Velocidad angular motor 2

• ω3: Velocidad angular motor 3

• ω4: Velocidad angular motor 4

Su modelo cinemático será el siguiente:

$$\begin{bmatrix} \omega 1 \\ \omega 2 \\ \omega 3 \\ \omega 4 \end{bmatrix} = \frac{1}{r} \begin{bmatrix} 1 & 1 & -(L+l) \\ -1 & 1 & (L+l) \\ -1 & 1 & -(L+l) \\ 1 & 1 & (L+l) \end{bmatrix} \begin{bmatrix} vxi \\ vyi \\ \omega i \end{bmatrix} \quad \ldots (3)$$

El siguiente modelo muestra en el plano cartesiano, el cual busca una relación entre las velocidades las 3 ruedas con las velocidades lineal y angular del robot.

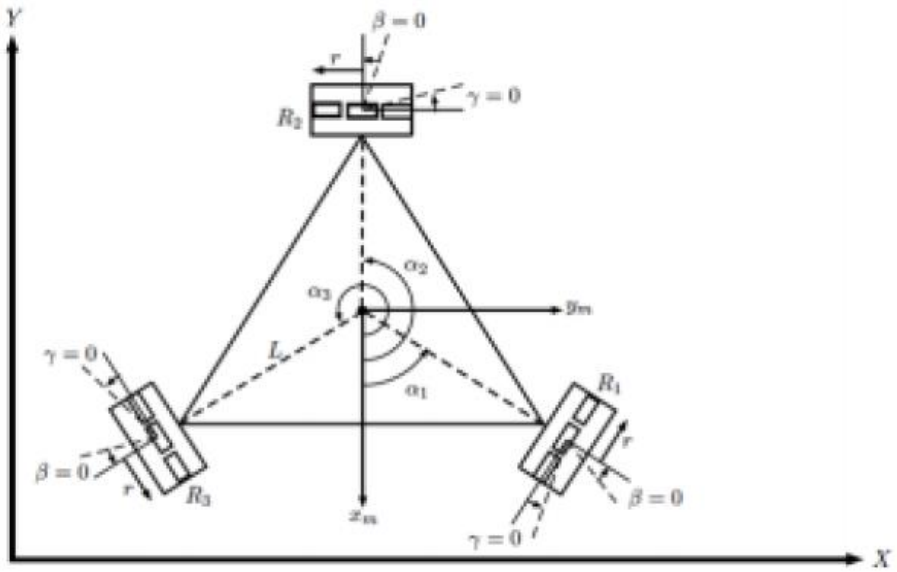


Figura 9:Robot omnidireccional 3 ruedas

Fuente: López (2017)

Siendo:

• vx: Velocidad lineal respecto al eje x del chasis

• vy: Velocidad lineal respecto al eje y del chasis

• ωi: Velocidad angular total del motor

• ω1: Velocidad angular motor 1

• ω2: Velocidad angular motor 2

• ω3: Velocidad angular motor 3

Su modelo cinemático será el siguiente:

$$\begin{bmatrix} \omega 1 \\ \omega 2 \\ \omega 3 \end{bmatrix} = \frac{1}{r} \begin{bmatrix} -\sqrt{\frac{3}{2}} & \frac{1}{2} & L \\ 0 & -1 & L \\ \sqrt{\frac{3}{2}} & \frac{1}{2} & L \end{bmatrix} \begin{bmatrix} vx \\ vy \\ \omega i \end{bmatrix} \quad \dots (3)$$

2.2.6 Estado de la Tecnología

Diferentes empresas a nivel mundial desarrollan tecnología enfocado a la asistencia social. Estos sistemas robóticos se encuentran en el mercado y sirven

de experiencias para verificar la funcionalidad de cada proyecto. A continuación, se expresa un listado de los diferentes robots desarrollados:

- KtBot de Tekniker: Este es un robot de servicio móvil, autónomo y multifuncional enfocado a ayudar en actividades de asistencia en entornos interiores.

- Spencer de KLM: Este robot fue diseñado para escoltar a pasajeros a su sala de embarque en aeropuertos.

- Pepper: Este robot humanoide, capaz de interactuar con las personas, es usado como guía en museos y hospitales.

- Robina de Toyota: Robot diseñado para el trabajo de guiar a visitantes a través de la sala de exposiciones de la empresa Toyota.

- Sanbot Elf: Robot diseñado para trabajo de guía y anfitrión, el cual puede interactuar con los usuarios a través de una pantalla táctil.

- Hospi: Robot creado por la marca Panasonic, fue creado para ayudar en tareas en hospitales, tales como entrega de medicamentos, muestras médicas, entre otros.

De los sistemas robóticos descrito líneas arriba, se presenta una tabla comparativa de los productos comerciales. En ella se busca identificar altura, velocidad, peso, autonomía y tecnología utilizada.

Tabla N° 3

Estado de tecnología – Robots de asistencia social

Nombre	Altura (cm)	Velocidad	Peso (Kg)	Autonomía	Tecnología utilizados
Robot guía de biblioteca	150	7Km/h	-	8 horas	- Sensor Ultrasónico MB1000 - Sensor Ultrasónico HC-SR04 - Sensor Inercial IMU 10 DOF - EasyVR 2.0 - Logitech HD Webcam C615
Spencer de KLM	193	4.68Km/h	250	8 horas	-Scanner láser 2D(x2) -Scanner láser 3D(x1) -Cámaras de profundidad ASUS Xtion (x4)
Pepper	120	3Km/h	28	12 horas	-Cámaras HD (x2) -Sensor giroscópico (x1) -Sensores láser (x6) -Sensores de proximidad (x6)
Robina	120	-	60	1 hora	-Sensores láser -Sensores de ultrasonido
Sanbot Elf	90	2.9km/h	19	10 horas	-Sensor giroscópico (x1) -Cámara RGB (x2) -Sensores infrarrojos (x15) -Sensor 3D (x1) -Sensores táctil (x13)
Hospi	139	3.6km/h	170	9 horas	-Sensores de ultrasonido (x27) -Pantalla táctil Panasonic

Fuente: “Estado de Tecnología”, Matencio (2021)

Con la información mostrada, se puede sintetizar que los sistemas robóticos tienen un desplazamiento cercano a la velocidad del caminar de una persona (3 km/h), siendo el robot guía de biblioteca el que tiene un diseño de velocidad mayor a lo común. Con el dato cuantificable que es la altura, se verifica de acuerdo al uso un tamaño acorde al de una persona pequeña el cual ronda los

150 cm aproximadamente, verificando el peso el cual va ligado al tamaño y material escogido para el diseño. En cuanto a la autonomía, se revisa el desempeño energético el cual supera la jornada laboral de 8 horas. Finalmente, los sensores más utilizados son los ultrasónicos, láser, IMU, y cámaras de video. (Matencio, 2021)

2.2.7 Sistema de Estructura Mecánica

Son sistemas que están constituidos por componentes o elementos que tienen como función específica transmitir movimientos para tareas como la locomoción o manipulación de objetos. Los robots tienen diferentes métodos de locomoción.

A) Sistema de locomoción

Estructura de mecanismos que permite el desplazamiento de vehículos a través de un determinado ambiente de trabajo y su autonomía incluso con cargas. En el caso de los robots terrestres, la locomoción se realiza mediante patas, ruedas y orugas. (ver Figura 10). (Pérez y Praso 2014).

Figura 10: Tipos de locomoción terrestre
Fuente: Kaddouh (2018).

B) Locomoción mediante ruedas

Es el desplazamiento mediante ruedas. Debido a su sencillez permite un mejor control comparado a la locomoción por patas. Pueden lograr grandes velocidades en terrenos planos, pero tienen pérdidas de potencia por rozamiento. Además, cuando se usan más de tres ruedas en un vehículo se debe implementar un sistema de suspensión para asegurar el contacto entre el suelo y las ruedas en terrenos irregulares. (Pérez y Praso 2014).

Según las características y el diseño de las ruedas su eficiencia varía en relación al desplazamiento de los vehículos. En la Figura 11 podemos ver algunas de las diferentes clases de ruedas que existen.

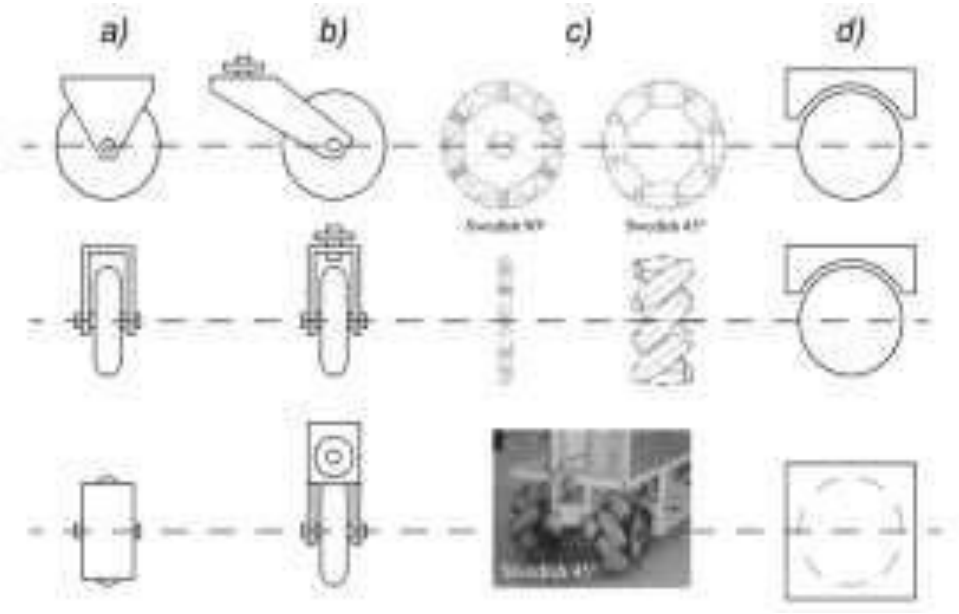


Figura 11:Tipos básicos de ruedas:(a) rueda estándar, (b) castor (c) sueca (d) rueda esférica

Fuente: Siegwart y Nourbakhsh (2004)

La rueda estándar realiza una rotación sobre su eje (lo cual permite avanzar o retroceder a un robot). En el caso de que sea una rueda direccionable (Figura 12 derecha) permite rotar sobre el eje de contacto (lo que hace girar a la izquierda o derecha al robot). Esta última, al tener dos movimientos posibles, decimos que tiene 2 GDL.

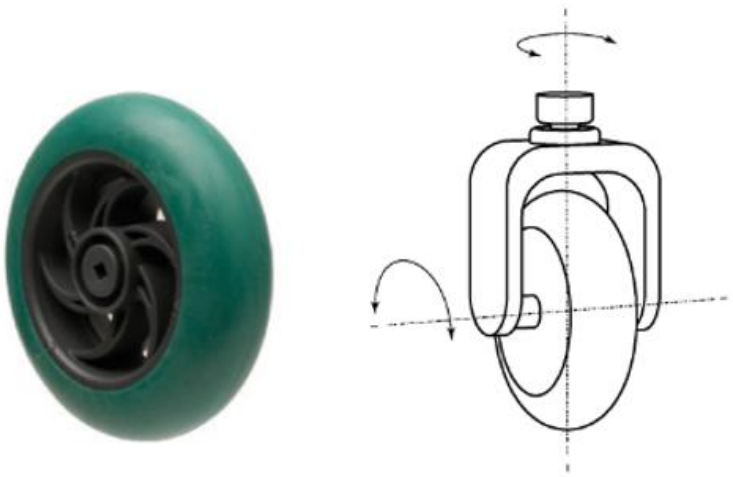


Figura 12: Rueda estándar.
Fuente: Vásquez (2015)

La rueda loca realiza una rotación sobre el eje de la rueda y sobre un eje de dirección desplazado del punto de contacto. Vemos que también tiene 2 GDL. Es la típica que llevan los carritos de la compra y su diseño permite que se direccione automáticamente en el sentido del movimiento del robot.

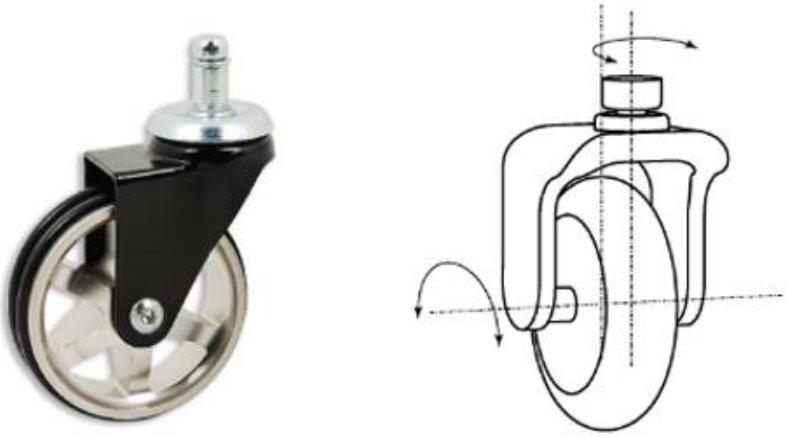


Figura 13:Rueda loca.
Figura: Vásquez (2015)

C) Chasis de Robot

Estructura principalmente diseñada para montar un sistema mecánico y/o electrónico. Tiene una serie de agujeros necesarios en el marco para facilitar la instalación de varios elementos derivados del sistema, tales como son cables, módulos, entre otros. Se puede utilizar para evitar obstáculos, seguimiento, pruebas de distancia, medición de datos,

velocidad y cualquier proyecto relacionado. (Ardobot 2022).

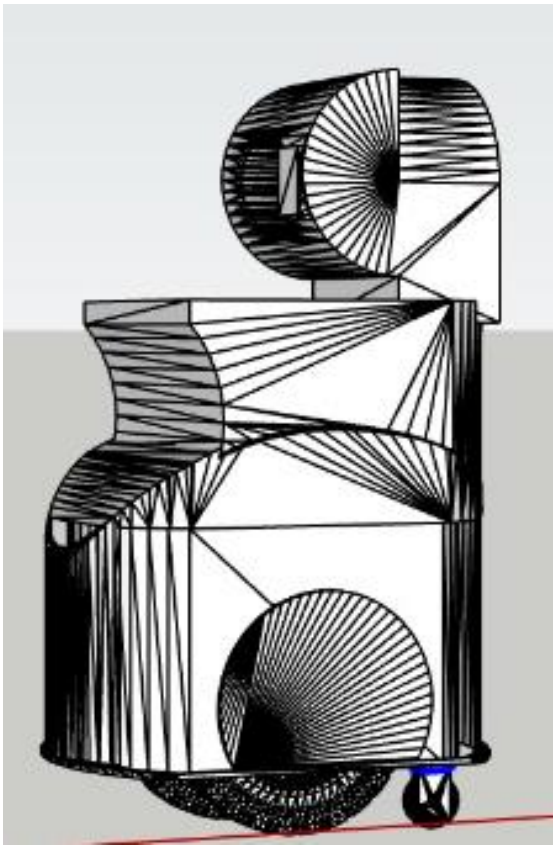


Figura 14:Chasis de robot modelado en SolidWorks.
Fuente: Elaboración propia.

2.2.8 Software de Diseño CAD

A) Simulación de modelos robóticos

La simulación en el campo de diseño de sistemas robóticos es importante. Esto nos permite modificar un modelo construido de manera rápida con el fin de analizar diferentes escenarios. A la vez, es posible corregir errores de programación que en físico podrían suponer un peligro tanto para las personas como para el sistema: Es más económico realizar pruebas y mejorar el sistema vía simulación, es sencillo comprender y visualizar los métodos de simulación que los analíticos, es posible analizar sistemas de mayor complejidad o con un detalle mayor, siendo la simulación el único medio para lograr la solución.

B) SolidWorks

SolidWorks es un software de CAD (diseño asistido por computadora) en 3D para el modelado de piezas, dibujos en 2D y ensamblajes en 3D. El software ofrece una amplia gama de soluciones para cubrir aspectos relacionados con las diferentes etapas del desarrollo del producto. Sus productos brindan la capacidad de crear, administrar, simular, diseñar, publicar, producir y simular datos de procesos de diseño. El presente software, luego de la creación de una pieza (ver Figura 15), nos permite elegir el material con el que deseamos trabajar y/o escoger otro que se adapte mejor al proyecto, dándonos características a conocer como son el peso, dureza, etc. El trabajo del software mientras se realiza la creación del producto es específica, sus soluciones aceleran el proceso y esto genera un ahorro tanto de tiempo como monetario, por lo que es un paso más hacia la innovación. (Solidbi 2018)

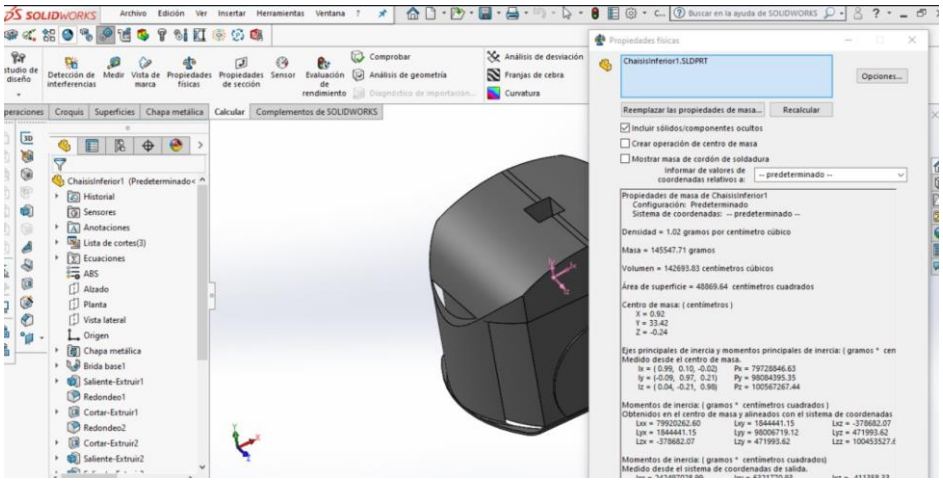


Figura 15:Pieza diseñada en SolidWorks

Fuente: Elaboración propia.

2.2.9 Sistema Eléctrico

Según la Universidad Católica de Santiago de Guayaquil (UCSG, 2018), un sistema eléctrico es el conjunto de conductores e instalaciones para almacenar, transportar y distribuir energía eléctrica dentro de una máquina para su correcto funcionamiento. Las variables a considerar para estos sistemas son: El voltaje, la

corriente, frecuencia, número de conductores, etc.

A) Fuente de Alimentación

Es aquel dispositivo que se encarga de transformar la corriente alterna (AC) de una red eléctrica comercial en corriente continua (DC) para alimentar los diferentes circuitos de un aparato electrónico al que esté conectado.

Para el robot móvil, se empleó dos baterías de Lipo de 3,7 V y 60000 mA: Una de ellas para energizar el Arduino, el Raspberry Pi y la cámara web, mientras que el segundo alimenta los motores de las ruedas. La ventaja de este tipo de baterías es que son recargables y de tamaño reducido, lo cual también significa un peso ligero. En la Figura 16 y la Figura 17 se puede observar la composición y dimensión de esta batería, viniendo en una presentación encartuchada de varias celdas.

Figura 16:Baterías de Lipo

Fuente: Empresa “Unit Electronics” (2022)

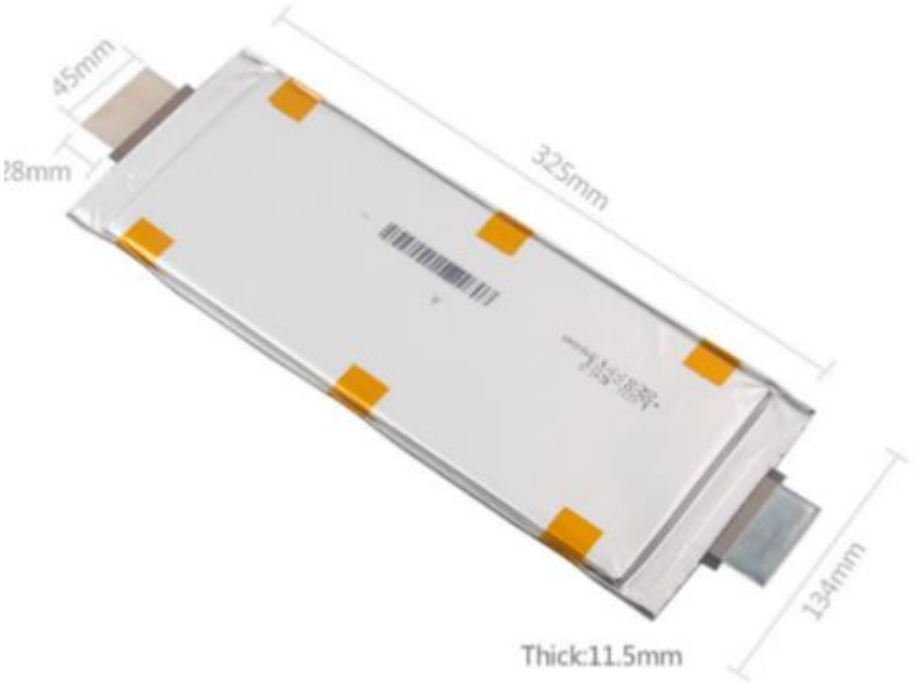


Figura 17:Dimensiones, en milímetros, de la batería Lipo.
Fuente: Empresa “Unit Electronics” (2022)

B) Motores

Dentro de la definición de los motores en robótica, se les denomina así a aquellos actuadores que transforman energía eléctrica en energía mecánica, capaz de movilizar un sistema para que realice un trabajo. Existen diversos tipos de motores dependiendo del uso que se le dé, pero los que destacan principalmente cuando se referimos a motores eléctricos para la movilidad de ruedas de un robot son dos: Motores AC y Motores DC.

A continuación, en el Tabla 4, se mostrará una comparativa entre ambos tipos de motores, las cuales muestran sus rasgos más destacables. Para usos prácticos de esta tesis, la elección definitiva fueron los motores DC.

Tabla N° 4

Diferencia entre los Motores Dc y los Motores AC.

Motores DC	Motores AC
Formado por un circuito magnético y circuito de armadura.	El rotor recibe una corriente inducida alterna. El estator es un campo magnético inducido.
Velocidad de giro aumenta con tensión aplicada.	Velocidad de giro regulado por variadores electrónicos de frecuencia.
Motor monofásico.	Motor monofásico o trifásico.
Par de Arranque: Muy fuerte	Par de Arranque: Escaso
Para trabajo pesado.	Para trabajo con precisión.

Fuente: Castillo (2014)

C) Motorreductor de Metal 25d 6-12v DC

Son motores de corriente continua de forma cilíndrica, con un diámetro de 25 milímetros y un eje de salida de la caja de cambios de 4 milímetros. Alcanza las 210 revoluciones por minuto (RPM), posee un torque de 12 kg-cm y puede operar cómodamente en rangos de 3V a 9V, aunque puede rotar con tensiones tan bajas como 1V. Voltajes por encima de los 9 puede afectar su tiempo de vida. Su peso es de 88 gramos y su corriente sin carga es de 300 mA, mientras que su corriente máxima es 5600 mA. Para el robot móvil de esta tesis, se emplearán dos de estos motores, como se ve en la Figura 18, que irán conectados a las ruedas para conseguir el desplazamiento lineal del robot móvil.

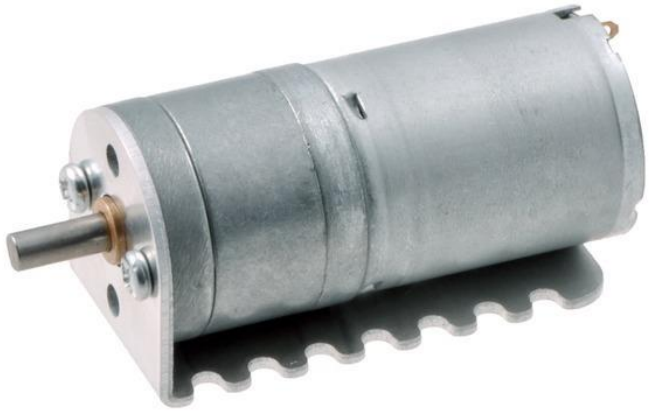


Figura 18: Motorreductor de Metal 25d 6-12v DC.

Fuente: Empresa “SandoRobotics” (2022)

D) Driver L298N

Es un módulo que permite controlar dos motores de corriente continua o un motor paso a paso bipolar de hasta 2 amperios. Cuenta con los componentes necesarios para trabajar sin necesidad de otros implementos, como los diodos de protección o un regulador LM7805 (que suministra 5V a la parte lógica). Posee jumpers de selección para las salidas del módulo (A y B). La salida A está conformada por OUT1 y OUT2 y la salida B por OUT3 y OUT4. Los pines de habilitación son ENA y ENB respectivamente, como se puede ver en la Figura 18.

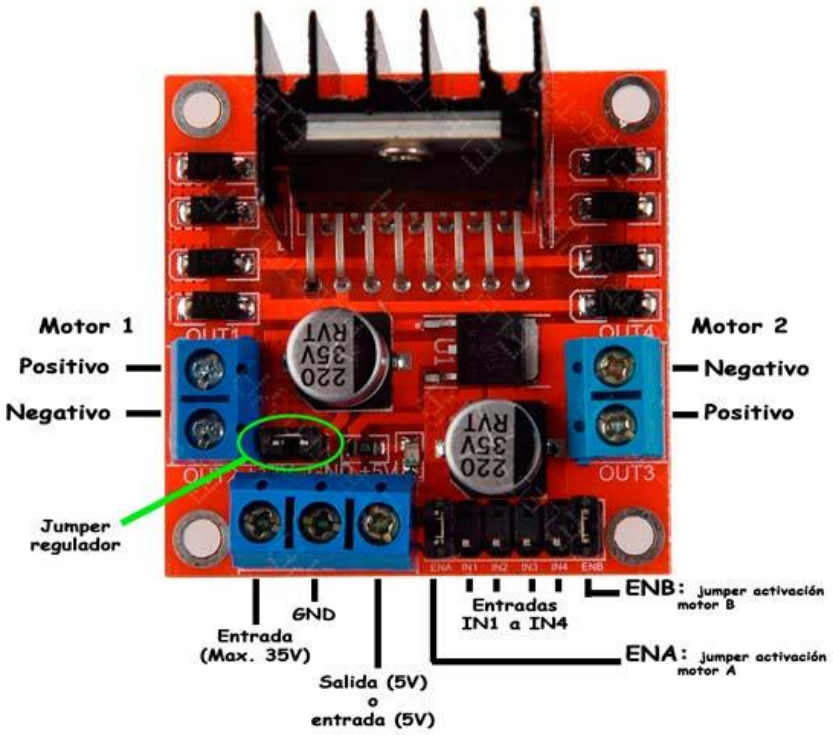


Figura 19: Componentes, entradas y salidas del Driver L298N

Fuente: Página web "El Octavo Bit" (2021)

2.2.10 Sistema Electrónico

Según la Universidad Católica de Santiago de Guayaquil (UCSG, 2018), los sistemas electrónicos son un conjunto de circuitos que interactúan entre sí mediante señales eléctricas para obtener un resultado determinado.

Todos ellos constan de tres bloques funcionales, tal como se puede apreciar en la Figura 20:

- Bloque de entrada: Aquí ingresa la señal por medio de un elemento accionador (como un interruptor) o por medio de sensores (como los finales de carrera).
- Bloque de proceso: Aquí se ubican los dispositivos que deciden la acción a realizarse. Se ocupa de transformar la señal de entrada en señal de salida, que acciona el módulo de salida.
- Bloque de salida: En este punto la acción por la que fue diseñada el sistema se realiza gracias a la señal procesada, ya sea por medio de motores, lámparas, timbres, etc.

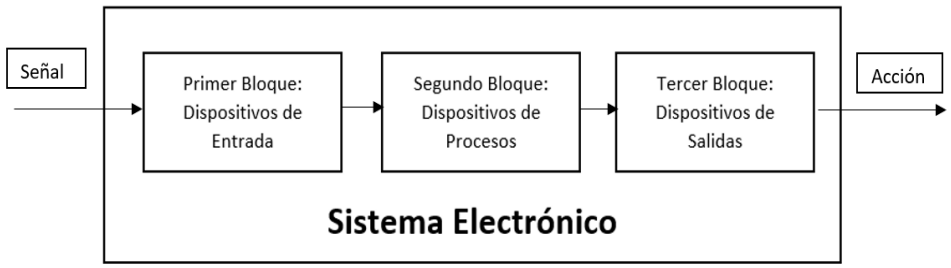


Figura 20: Gráfico del Diagrama de Bloques de un Sistema Electrónico
Fuente: Elaboración propia.

A) Sensores

Dispositivos que logran obtener datos de magnitudes físicas (luz, temperatura, sonido) u otras alteraciones en su entorno, consiguiendo procesar estos estímulos para convertirlos en señales eléctricas.
Se pueden clasificar dependiendo de la variable que midan, la naturaleza de la variable de salida o el tipo de variable que se tiene en la salida.

B) Sensores de Temperatura

Tal como su nombre lo indica, se emplean para medir la temperatura en el aire o agua. A continuación, en la Tabla 5 se podrá ver una lista de estos de sensores dividido en tres categorías por su uso industrial: Eléctricos,

Mecánicos y Radiación Térmica.

Tabla N° 5

Tipos de Sensores de Temperatura según su uso Industrial

Eléctricos	Mecánicos	Radicación Térmica
Termocuplas	Sistemas de Dilatación	Pirómetros de Radiación
Termorresistencias	Termómetros de vidrio con	Total (Banda Ancha)
Termistores	líquidos	Óptico
Diodos	Termómetros bimetálicos	Pasa banda
Sensores de Silicio con		Relación Termómetros
Sensores resistivos		Infrarrojos

Fuente: Universidad Nacional de Ingeniería (2019)

C) Sensor de Temperatura AMG8833

Fabricado y distribuido por la empresa Panasonic, es un sensor de cámara térmica infrarroja (IR) que posee 8 filas y 8 columnas. Está diseñada a 64 pixeles, por lo que la integración para añadir la visión térmica en proyectos que emplea Arduino u otras plataformas que procesen imágenes con la biblioteca SciPy Python.

Este sensor llega a comunicarse por medio del puerto I2C y lee cada uno de los pixeles que posee, logrando obtener 64 lecturas de la temperatura infrarroja individualmente. Es posible configurar la dirección I2C por medio del pin AD0 dependiendo si es que está conectado o no GND. En caso de que lo esté, será 0x68 la nueva dirección; en caso no lo esté, se mantendrá con 0x69. A continuación, se darán otras de sus especificaciones:

- El voltaje de operación es 3V a 5V DC
- Su consumo de corriente es 4.5 mA
- Su rango de trabajo en temperatura es 0°C hasta 80°C
- Su máxima distancia de detección son 7 metros con objetos grandes.
- Su tasa de lectura es 10 frames por segundo.
- Su precisión es ±2.5°C

- Sus dimensiones son 17*18 mm y pesa 3 gramos.

Figura 21:Pines del Sensor de Cámara Térmica IR AMG8833

Fuente: Empresa “Unit Electronics” (2022)

D) Seguidor de Línea TCRT5000

Como su nombre lo indica, es un seguidor de línea óptico infrarrojo que tiene un emisor IR y un fototransistor que capta la luz reflejada de un obstáculo. Se puede ajustar la distancia de detección y los pines de salida digital y analógico; además de ser bastante fácil de manipular con módulos o microcontroladores. Este sensor se aplicará para que el robot móvil pueda desplazarse en línea recta desde su posición original hasta un punto cercano al cliente para tomar su temperatura. En la Figura 22 podemos observar tanto el sensor conectado a su módulo que le permitirá la configuración de seguidor de línea, así como el sensor IR desacoplado en caso sólo se necesite de este.

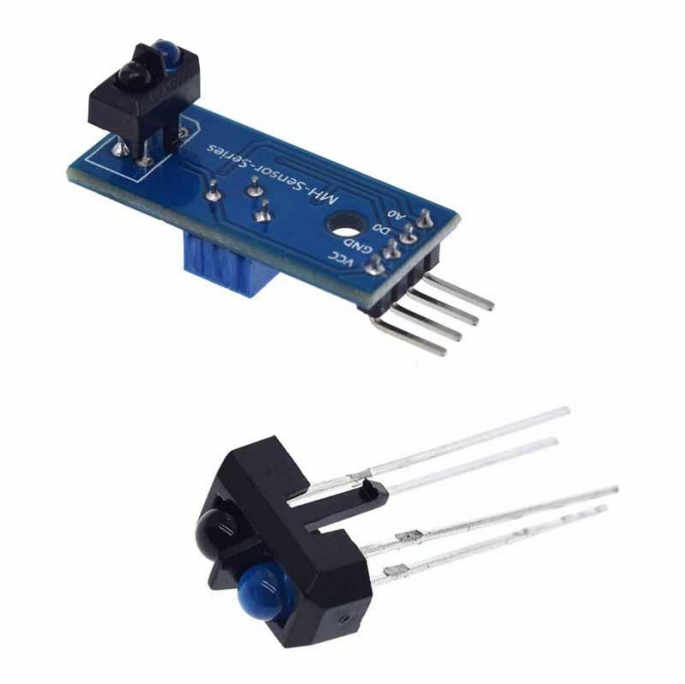


Figura 22: Seguidor de Línea TCRT5000

Fuente: Empresa “Unit Electronics” (2022)

Tabla N° 6

Características técnicas del módulo seguidor de línea TCRT5000

Voltaje de Operación	3.3V a 5V
Distancia Máxima de funcionamiento	2.5 mm
Peso	3 g
Consumo de corriente del módulo	15 mA
Corriente del colector del transitor	100 mA (máximo)
Temperatura de funcionamiento	-25°C a +85°C
Longitud de onda del módulo	950 nm
Dimensiones del módulo	31 mm × 13 mm × 2 mm

Fuente: Empresa “Unit Electronics” (2022)

E) Cámara Web HD C505

Se trata de una cámara web con video HD 720p, 30 cuadros por segundo (FPS) y 1.2 Mega píxeles. Compatible con Windows y macOS, será la encargada de captar las imágenes del usuario para determinar si posee un cubrebocas y, conectado al Arduino para trabajar con el sensor de temperatura infrarrojo por medio de un cable USB tipo A, si no excede la

medición de temperatura corporal establecida como fiebre. En la Figura 23 podemos ver el modelo C505 que se empleará para el robot móvil, así como en la Tabla 7 se detallaran sus características más destacables.

Figura 23:Cámara Web HD C505

Fuente: Empresa “Logitech” (2022)

Tabla N° 7

Características técnicas de la cámara web HD C505

Altura (mm)	31,91
Ancho (mm)	72,91
Profundidad (mm)	66,64
Longitud de Cable (m)	2
Peso (g)	75
Tipo de Enfoque	Fijo
Tipo de Lente	Plástico
Micrófono	Mono-Integrado
Campo visual diagonal (dFoV)	60°

Fuente: Empresa “Logitech” (2022)

F) Arduino

Es una plataforma electrónica de código abierto, fácil y flexible de aprender a usar para los desarrolladores que deseen crear diversos tipos de microordenadores. Posee 14 pines de entrada/salida digital, un cristal de 16Mhz, 6 entradas analógicas, conexión USB para energizarlo, un botón de reseteo, etc. Mediante su interfaz de entrada podemos conectar diferentes periféricos (cámara, teclado, sensores), cuya información será trasladada hacia su microcontrolador para el procesamiento de datos. Además, por medio de su interfaz de salida, lleva la información procesada a otros periféricos como pantallas, altavoces u otros controladores para su reproducción. En la Figura 24 podemos ver el modelo más comercial de esta plataforma, Arduino Uno.

Figura 24:Placa del modelo Arduino Uno
Fuente: Empresa Arduino (2022)

En la Tabla 8 se aprecia una comparativa entre los tipos de Arduino más destacables. En base a sus características, se optó por emplear el Arduino Uno para la constitución de esta tesis, ya que presenta un precio más cómodo en comparación al Arduino Mega, una memoria suficiente para las funciones que se requieren, una conectividad estándar de USB y compatibilidad con Wifi/Bluetooh/Shield (placas de circuitos modulares que se apila sobre el Arduino para darle funcionalidades extra) a diferencia de un Arduino Micro.

Tabla N° 8

Cuadro comparativo de las características entre los tipos de Arduino

	Arduino Uno	Arduino Mega	Arduino Micro
Precio (dólares)	20-23	36.60 - 39	20-24
Dimensiones (pulgadas)	2.7x2.1	4x2.1	0.7x1.9
Procesador	Atmega328p	ATmega2560	ATmega32U4
Frecuencia (Mhz)	16	16	16
Memoria Flash (kB)	32	256	32
EEPROM (kB)	1	4	1
Voltaje (V)	5	5	5
Pines de Entrada/Salida	14	54	20
Pines Analógicos	6	16	12
Conectividad USB	Estándar A/B	Estándar A/B	Micro-USB
Compatibilidad con "Shield"	Sí	Sí	No
Wifi/Bluetooth	No (un shield/módulo puede permitirlo)	No (un shield/módulo puede permitirlo)	No

Fuente: Gudino (2017)

G) Raspberry PI

Creada por la empresa Raspberry Pi Foundation, es una placa base simple (SBC) que bien podría ser la de una mini PC por su tamaño, es de bajo costo y ofrece variable funcionabilidad. Es aquí donde empieza el gran atractivo de esta placa base, ya que podremos elegir componentes extras según nuestras necesidades.

Figura 25:Raspberry PI 3A+

Fuente: Empresa Raspberry Pi (2022).

Existen dos tipos de modelo de Raspberry Pi, el modelo A y el modelo B, además de cuatro generaciones y revisiones. Como parte de la tesis, se escogió el modelo “Raspberry Pi 3A+” y en la Tabla 9 se enseñan sus características.

Tabla N° 9

Especificaciones técnicas del modelo Raspberry Pi 3A+

Procesador	Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz
Memoria RAM	512MB LPDDR2 SDRAM
Dimensiones	65mm x 56mm x 8.5mm
WiFi	2.4GHz y 5GHz 802.11.b/g/n/ac
Bluetooth	4.2/BLE
Conectores	40 pines GPIO HDMI Cámara CSI Pantalla DSI
Puertos	USB 2.0 MicroSD
Alimentación	5V/2.5A DC

Fuente: Datasheet “Raspberry Pi Modelo A+” (2018)

2.2.11 Automatización

A) Software de Automatización

Un software de automatización se encarga de crear instrucciones y procesos repetibles que reemplacen o reduzcan la interacción humana en los sistemas. Este proceso es fundamental para mejorar la tecnología de la información (TI) y la transformación digital en industrias.

Los entornos de desarrollo integrado (IDE) son aplicativos informáticos que ofrecen un servicio integral donde se pueden desarrollar estos softwares. Los IDE también reconocen un gran número de lenguajes de programación y ofrecen las siguientes ventajas:

- Bastante dinámico y excelente para las personas no tan versadas en el control de consolas.
- Permite formatear los códigos.
- Posee funciones para cambiar el nombre de las variables.

- Aviso de “Warnings” y en la pantalla en caso de existir un error al momento de copilar.
- Visualización de los proyectos creados de forma gráfica.
- Funciones de extracción de código para aplicar un nuevo método.

Entre los IDE destacados tenemos a Eclipse, un software libre de entorno Java usado a nivel profesional; NetBeans, un software libre de entorno Java que el diseño gráfico de las aplicaciones; BlueJ, software libre enfocado al aprendizaje de Java que no sea de uso profesional; JBuilder, software comercial que permite desarrollos gráficos y posee versiones de prueba; JCreator; sofware comercial que omite herramientas de desarrollo gráfico para hacerlo más rápido y eficiente; MS Visual Studio, software comercial para Windows y macOS; y Visual C++, un copilador para la programación C que es parte de Visual Studio.

B) MS Visual Studio

Es un IDE para macOS y Windows; a la vez es compatible con diferentes lenguajes de programación como C++, Visual Basic, Java, Python y PHP. Este IDE posibilita a los usuarios a crear sitios y aplicaciones web, así como servicios que sean compatibles con la plataforma .NET 2002. De esta forma se consigue producir aplicaciones que logren comunicarse a través de estaciones de trabajo, una página web, móviles, videoconsolas, etc.

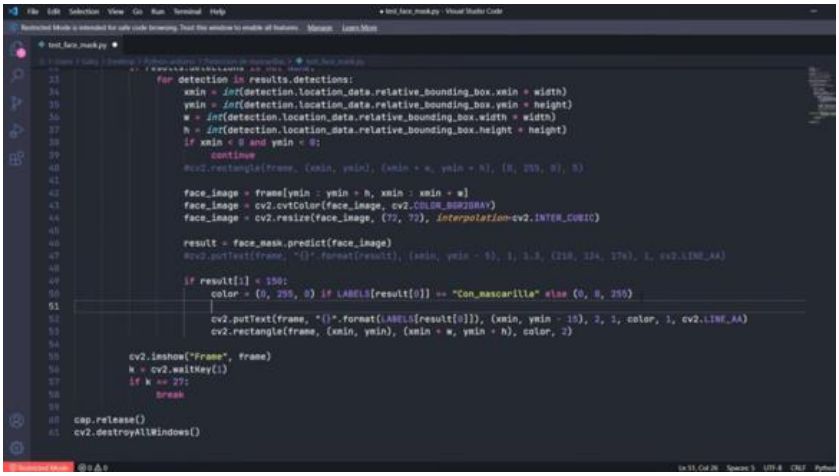


Figura 26:Lenguaje Python en el IDE MS Visual Studio.

Fuente: Elaboración propia.

2.2.12 Visión artificial

Dentro de este campo están incluidas las aplicaciones de uso industrial y otras áreas, siendo un conjunto de hardwares y softwares que ofrecen una orientación a diferentes elementos electrónicos a través de la captura y el procesamiento de imágenes para realizar una función. En la Tabla 10 se pueden observar sus aplicaciones en objetivos estratégicos. Estos sistemas emplean sensores digitales previamente resguardados en cámaras industriales para la obtención de imágenes, con el objetivo de que el software de la computadora pueda analizarlo y encontrar diversas características que lo lleven a tomar una decisión.

Tabla N° 10

Objetivos estratégicos de la visión artificial

Meta estratégica	Aplicaciones de visión artificial
Aumentar la calidad	Inspección, medición, calibración y verificación de montaje
Productividad aumentada	Las tareas repetitivas que antes se realizaban de forma manual ahora las realiza el sistema de visión artificial
Flexibilidad de producción	Medición y calibración, guiado robótico, verificación previa de operación
Menor tiempo de inactividad de la máquina y menor tiempo de configuración	Cambios programados por adelantado
Información más completa y control de proceso más estricto	Las tareas manuales ahora pueden proporcionar retroalimentación de datos
Menores costos de equipo de capital	Agregar visión a una máquina mejora su rendimiento, evita la obsolescencia
Disminuir costos de producción	Un sistema de visión vs. muchas personas, detección temprana de defectos en el proceso
Reducción de desechos	Inspección, medición y calibración
Control de inventario	Reconocimiento Óptico de Caracteres e identificación
Disminuir almacenamiento	Sistema de visión vs. operador

Fuente: “Introducción a la visión artificial”, Cognex (2018)

A) Deep Learning

Tecnología cuyo propósito es que un ordenador posea la facultad de aprender tal cuál un ser humano, consiguiendo que los sistemas de visión artificial sean más efectivos en diversos entornos. Durante el proceso de aprendizaje humano utilizamos imágenes. Nuestra mente aprenderá ciertas características de estas imágenes y sabrá tomar una imagen que no ha visto antes e identificar el objecto que se quiera reconocer. Esta es la forma humana de aprender, y es lo que hace que se nos dé tan bien hacer estas tareas de reconocimiento. Sin

embargo, para que lo haga un ordenador, hay que utilizar algoritmos complejos para describir lo que ha de reconocer. Con los métodos tradicionales de procesado de imagen es necesario especificar que nos encuentre una estructura de cierta forma, de tales posibles colores, con determinadas características, tamaño, etc. (Santos, 2020)

El Deep learning utiliza redes neuronales artificiales para acumular información. Su forma de trabajo es usándolas para extraer y transformar las variables. Cada una de sus redes emplea los datos de entrada para moldear una salida que retroalimenta al sistema varias veces hasta conseguir la mejor precisión en su respuesta. (Zúñiga, 2019)

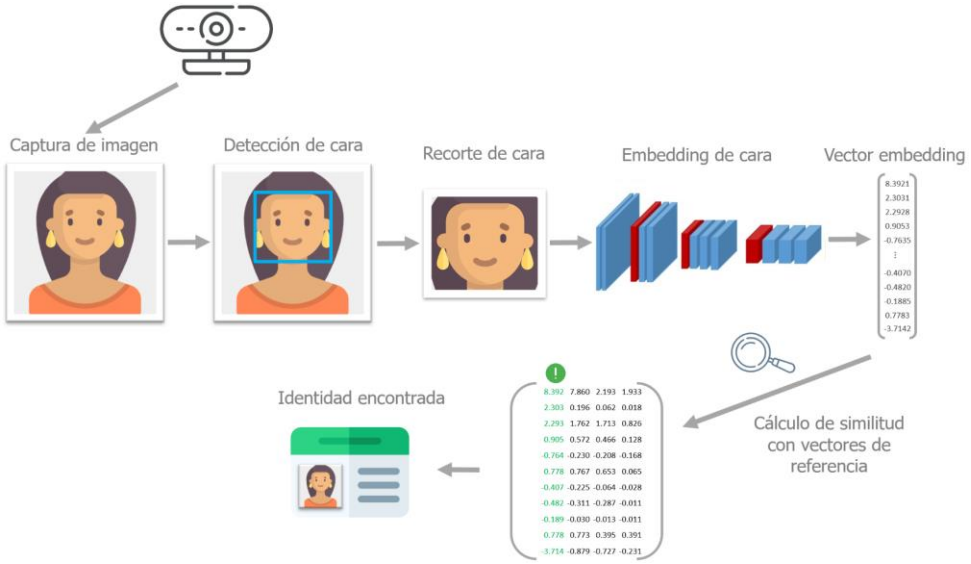


Figura 27:Diagrama de fases de un sistema de reconocimiento facial.
Fuente: Amat (2021)

2.3 Definición de Términos Básicos

En este apartado se define los términos básicos más importantes en el marco de la presente tesis:

a) Lenguaje Python: Lenguaje de programación de nivel alto diseñado para poseer un código fácilmente legible. Utilizado para crear aplicaciones de diversos rubros, como por ejemplo Instagram, Spotify, Netflix, Panda 3D, etc.

b) Open Source Computer Vision (Open CV): Es una librería que permite producir aplicaciones de visión por computadora en un marco de trabajo de alto nivel en tiempo

real, como serían las estructuras de datos, procesamiento y análisis de imágenes, análisis estructural, etc.

c) Prototipo: Consiste en la implementación incompleta de un sistema o del sistema completo, pero en una escala menor.

d) Redes Neuronales: Método empleado por una inteligencia artificial para enseñar a computadoras el procesamiento de datos de la misma forma que lo hace el cerebro humano.

e) Modelo Cinemático: La cinemática diferencia relaciona las velocidades del punto de interés con las velocidades que gobiernan el movimiento del robot.

f) Jacobiano: Relaciona las velocidades en el punto de interés con las velocidades de actuación y gracias a esta agrupación se puede realizar diferentes análisis como encontrar configuraciones singulares, analizar redundancia, determinar la cinemática diferencial inversa, es indispensable para el análisis y diseño de algoritmos de control.

CAPÍTULO III: METODOLOGÍA DEL ESTUDIO

3.1 Tipo y Metodología de Investigación

El presente proyecto de tesis es del tipo descriptivo, cuyo propósito será describir las fases que se deben seguir para el diseño de un robot; incluyendo la configuración, las pruebas de calibración del sistema y análisis de los datos adquiridos relacionados en el marco de la detección del uso de mascarillas y mensuración de temperatura de personas.

Fase 1.- Diseño de la estructura mecánica

- Analizar las características necesarias para el planteamiento de la estructura mecánica.
- Elegir el material adecuado para el diseño del robot.
- Calcular la velocidad máxima del robot para su aplicación en atención a personas.
- Seleccionar motores para el desplazamiento y los actuadores del robot.
- Diseñar la estructura mecánica con SolidWorks para visualizar el modelodesarrollado.

Fase 2.- Diseño del sistema eléctrico y electrónico

- Seleccionar componentes eléctricos requeridos para la operación del robot.
- Calcular la corriente requerida y selección del componente eléctrico para el almacenamiento de energía del robot.
- Seleccionar componentes electrónicos adecuados para el buen funcionamiento.

Fase 3.- Desarrollo del software del robot

- Desarrollar el software de reconocimiento facial.
- Seleccionar la cámara digital y base de datos de entrenamiento.
- Desarrollar el sistema de navegación del robot.
- Evaluar y ajustar las variables de la cámara.

3.2 Diseño del sistema mecatrónico

En el presente capítulo se describe el desarrollo del diseño del sistema de mecánico, eléctrico y electrónico, además se desarrolla la programación del robot móvil detector del uso de mascarilla y medición temperatura corporal.

A) Identificación de área de trabajo del sistema robótico

Para mencionar el requerimiento del sistema para su desarrollo en su ambiente de trabajo, se detallarán las características principales que el sistema debe tener para solucionar la problemática identificado anteriormente. Para ello, se enlistan los requerimientos generales y por subsistemas: mecánicos, electrónicos, eléctricos y control.

Especificaciones generales:

- Debe trabajar en un área amplia y suelo liso.
- Debe de ser capaz de permanecer en servicio una jornada laboral completa (8hrs).
- Debe de cumplir con sus funciones y alertar de ser necesario.

Especificaciones técnicas mínimas del Robot.

- Las dimensiones máximas del robot serán 77 x 60 x 140 cm (Largo x ancho x alto).
- La masa del robot no deberá exceder los 70 Kg.
- Su velocidad máxima será de 4 km/h
- El diseño del robot debe permite un fácil montaje y desmontaje de los componentes.
- El diseño del robot deberá tener bordes suaves para evitar dañar a las personas.
- El diseño mecánico del robot debe contemplar una estructura que reduzca las vibraciones.
- El robot debe ser diseñado con un material que cumpla con las normas de higiene y seguridad en salud para que su desinfección sea total y no afecte la salud de las personas con las que interactúe.
- Mínimo espesor de la plancha de ABS media pulgada.

B) Velocidad de trabajo

Se debe respetar el libre tránsito de las personas, para ello dependiendo de la dimensión del robot es recomendable que la velocidad máxima sea de 4km/h.

C) Autonomía

Es fundamental tener en cuenta cuánto es el tiempo de autonomía que deseamos cubrir con nuestro robot, quizás aplicar en horarios donde la afluencia de público es mayor o entre otros. Para nuestro diseño se garantizará una autonomía de 8 horas.

D) Carga nominal

El sistema robótico debe de ser capaz de cargar su propio peso además del diferente equipamiento eléctrico mínimo necesario.

A continuación, en la Tabla 11, se detallará los valores y unidades métricas de cada una de los valores a tener en cuenta para la realización del robot.

Tabla N° 11

Variables y valores del robot móvil

Item	Métrica	Unidad	Valor
1	Velocidad	m/s	0.6
2	Carga	Kg	60
3	Autonomía	Horas	5

Fuente: Elaboración propia

3.3 Matriz morfológica

Para el desarrollo del diseño en general y preliminar del sistema robótico se tuvo en cuenta diferentes componentes mínimos necesarios y se validó su uso mediante la descriptiva de sus ventajas y desventajas. La Tabla 12 es una matriz morfológica que se elaboró para visualizar de manera idónea la selección de cada componente, teniendo en cuenta 3 alineamientos las cuales fueron: Facilidad de configuración e instalación, desempeño esperado, condiciones de trabajo y costos.

Tabla N° 12

Matriz Morfológica de los elementos posibles a usarse para el robot móvil

ELEMENTO	ITEM 1	ITEM 2	ITEM 3
TIPOS DE RUEDAS	Rueda fija <u>Ventajas</u> - Fácil diseño - Item de menor costo - Adaptable a diferentes terrenos <u>Desventajas</u> - No tiene maniobrabilidad	Rueda Omnidireccional <u>Ventajas</u> -Mayor maniobrabilidad -Permite movimientos holonómicos <u>Desventajas</u> -Diseño Complejo - No pueden ser usadas en configuraciones móviles con ruedas impares. - Costo mayor	Rueda loca <u>Ventajas</u> -Maniobrabilidad -Permite movimientos seudo holonómicos -Pueden ser usadas en configuraciones móviles con ruedas impares. <u>Desventajas</u> -Diseño semi complejo
CONFIGURACIÓN MÓVIL	Dos ruedas <u>Ventajas</u> - Configuración de bajo costo - Puede ser lineal o holonómico <u>Desventajas</u> -Movimientos en misma dirección si se usa rueda fija.	Tres ruedas <u>Ventajas</u> - Configuración estable mecánicamente. - Movimiento en varias direcciones dependiendo de su configuración. <u>Desventajas</u> - Cinemática	Cuatro ruedas <u>Ventajas</u> -Movimiento holonómico con configuración y uso de ruedas omnidireccionales -configuración estable mecánicamente <u>Desventajas</u> -Configuración de

	-Movimiento Pseudo-Holonómico son ruedas ordinarias	compleja respecto al robot de dos ruedas fijas	costo mayor -Cinemática compleja respecto al robot de dos ruedas fijas.
Microcontrolador	Raspberry Pi <u>Ventajas</u> - Versatilidad en la integración de módulos adicionales. -Permite realizar tareas paralelas. Permite que si programación sea orientada a objetos. - Memoria Ram de hasta 6 Gb y velocidad de procesamiento de 1.5 GHz <u>Desventajas</u> Costo elevado	Arduino <u>Ventajas</u> Versatilidad en la integración de módulos adicionales Fácil programación e implementación. Menor costo <u>Desventajas</u> - Las tareas se realizan de manera secuencial - Posee menor memoria que su contendiente Raspberry -Posee una velocidad de procesamiento de 16 MHz, lo cual es menor que la de Raspberry Pi.	PIC <u>Ventajas</u> Bajo precio <u>Desventajas</u> - Es necesario contar con un programador para PIC - Las tareas se realizan de manera secuencial - Baja capacidad de memoria EPROM respecto a sus competidores, con 64 bytes disponibles

Fuente: Elaboración propia.

3.4 Diseño Mecánico

3.4.1 Dominio mecánico

Este proceso posee las funciones que se encargará de los movimientos principales y la protección del sistema, los cuales se detallan a continuación:

- Desplazar sistema: El robot debe contar con los tipos de ruedas necesarias para movilizarse.
- Transmitir potencia de motor a ruedas: La transmisión de potencia debe ser la adecuada para movilizar al motor
- Proteger y almacenar componentes: En esta función se define la forma del robot que protegerá a los componentes de golpes y los almacenará.
- Soportar componentes: Los componentes pequeños deben tener su propia carcasa que asegure que estos estén fijos.

3.4.2 Característica del material para la estructura mecánica

La estructura del robot tiene que contar un material cuyas características físicas sean las más adecuadas para su correcto funcionamiento. La Tabla 13 simplifica estos detalles: Por su utilidad debe de contar con materiales de mediana dureza, densidad baja y conductividad térmica baja para la correcta ventilación de los equipos electrónicos y eléctricos que lo componen.

Tabla N° 13

Características del material de la estructura mecánica

Requerimiento	Características del material	Propiedades
El robot debe de tener una carcasa que mantengan seguro los equipos eléctricos y electrónicos que contiene.	Dureza	Media

La carcasa debe de ser de un material liviano para que los elementos de tracción no sufran mucho esfuerzo y evitar recalentamientos.	Densidad	Baja
El lugar de trabajo será en establecimientos cerrados y ventilados de alto tránsito.	Conductividad Térmica	Media

Fuente: Elaboración propia

3.4.3 Selección de material

Para la presente selección de material se realizó la elección de 4 materiales más comunes para el armado estructuras mecánicas, y se realiza su comparación en el Tabla 14 mostrada a continuación:

Tabla N° 14

Comparación de algunos materiales más usados para la estructura

Características	ABS Polímero	Acero Inoxidable	Acero Galvanizado	Aluminio
Dureza (HB)	105-110	180	163	245
Densidad (Kg/m^3)	950	7980	7850	2698.4
Conductividad Térmica (W/ (m.K))	0.12	16.3	46	209.3

Fuente: Elaboración propia.

De acuerdo a la anterior tabla, el material que se asemeja más a nuestro requerimiento para ser tomado en cuenta para el diseño de nuestro robot es el material Acrilonitrilo butadieno estireno (ABS). Este presenta una dureza Brinell de 110 HB, siendo una dureza media y teniendo referencia que este material es usado en la industria automovilística en partes como parachoques, tablero de instrumentos, panel de la puerta, entre otros. En el aspecto de peso, tiene una densidad baja en comparación de los otros materiales, siendo el más bajo con 950 Kg/m^3 con una diferencia de más de 1500 Kg/m^3 con el material más cercano que es el aluminio. Adecuar el material ABS a un sistema robótico mejora el peso y consecuentemente ayuda a elegir un motor de bajo consumo debido al menor esfuerzo que realiza.

Es importante hacer referencia que el ABS es un termoplástico que surge por la polimerización del estireno y acrilonitrilo en la presencia del polibutadieno; es decir, es un copolímero, resultado de la combinación de los tres monómeros. Las porciones pueden variar del 15-35% de acrilonitrilo, 5-30% de butadieno y 40-60% de estireno, estas porciones en su resultado final variaran sus propiedades físicas, y pueden ser seleccionadas de acuerdo a cada requerimiento.

El ABS es un material amorfo, es decir que no tiene temperatura de fusión verdadera y esta se encuentra en un rango de 220 a 240°C, y cuenta con buenas propiedades mecánicas haciéndolo uno de los materiales más usados por la tecnología de manufactura aditiva FDM. Es considerado un plástico de ingeniería, porque su elaboración y procesamiento es algo más complejo que en los plásticos comunes. El ABS se originó por la necesidad de mejorar algunas propiedades del poliestireno de alto impacto. Para obtenerlo se mezclan emulsiones de dos polímeros, SAN y polibutadieno. La mezcla que se la realiza es coagulada, para obtener las siguientes propiedades mecánicas y físicas.

Tabla N° 15

Propiedades mecánicas y físicas del ABS

Propiedades mecánicas y físicas del ABS	
Módulo elástico	1.1 – 2.9 GPa
Coeficiente de Poisson	0,391 – 0,422
Resistencia mecánica a la compresión	31 – 86,2 MPa
Resistencia mecánica a la flexión	47,8 – 76 MPa
Resistencia mecánica a la tracción	27,6 – 55,2 MPa
Temperatura de transición vítrea	87,9 – 118 °C
Temperatura de fusión	200 – 245 °C

Fuente: Propiedades cuantitativas del ABS, Martínez (2012)

3.4.4 Selección del motor

Para seleccionar el motor que cumpla con el requerimiento técnico mínimo para el uso en la aplicación de tipo supervisión, fue necesario analizar las especificaciones de diseño de producto como lo son la velocidad de trabajo y la carga nominal para transportar. Se muestra el diagrama de cuerpo libre de una rueda en la Figura 27.

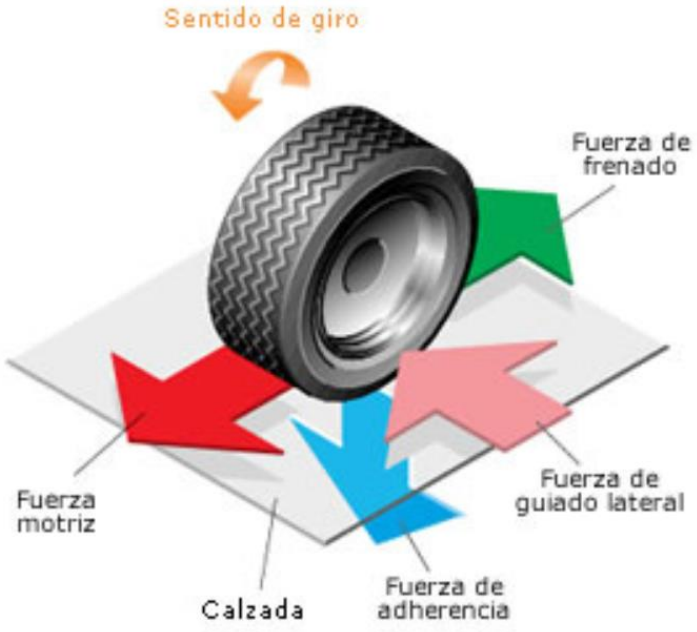


Figura 28: Variables consideradas para DCL.

Fuente: Enriquez (2020)

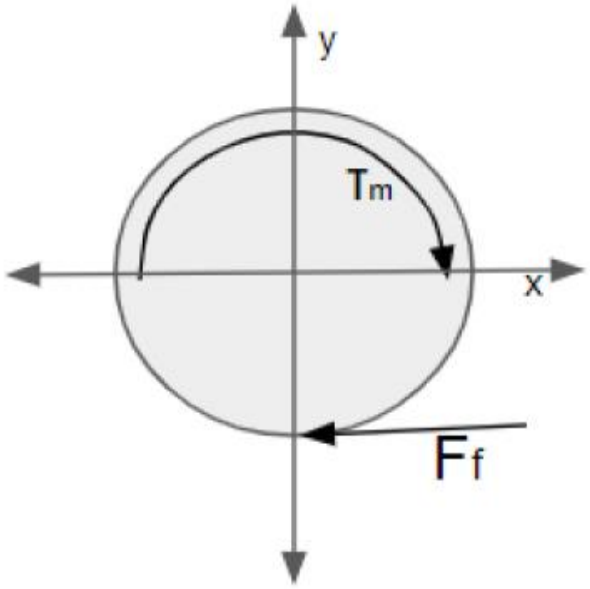


Figura 29:Diagrama cuerpo libre
Fuente: Cordoba (2017)

Donde:
Tm: Torque nominal del motor
Ff: Fuerza de fricción

Para la simplificación del análisis se toma plantea como punto inicial lo siguiente:

- Primero, la rueda como un objeto rígido (no se deforma).
- Segundo, la rueda no se desliza por tanto la velocidad del punto de contacto es siempre igual a cero.
- Tercero, para el cálculo se toma el caso en que el agente se mueve hacia el frente, implicando así que ambos motores van a la misma velocidad y poseen el mismo torque.
- Cuarto, se considera que el vehículo (robot móvil) se encuentra en terreno sin inclinaciones.

Se consideran las siguientes ecuaciones.

$$\sum T = Tn - r\ .\ Ff = 0 \ldots\ldots\ldots (4)$$

Siendo:

- Tn: torque nominal del motor
- Ff: fuerza de fricción
- r: radio de la rueda

Se asume la sumatoria de torques respecto a un eje que cruza de forma perpendicular al centro de la rueda, realizamos la sumatoria igual a cero para hallar un valor mínimo de torque que debe tener nuestro motor.

De lo descrito, se sabe que:

$$Tn = r\ .Ff \ldots\ldots (5)$$

La fuerza de fricción es proporcional a la fuerza normal que actúa en el vehículo, en este caso esta fuerza es igual al peso.

$$Tn = r\ .\mu_e\ .N = r\ .\mu_e\ .m\ .g \ldots\ldots (6)$$

Siendo:

- Tn: Torque nominal de la rueda
- r: radio de la rueda
- μ_e: Coeficiente de resistencia a la rodadura (estático).
- N: Fuerza normal
- m: masa del vehículo
- g: gravedad de la tierra.

El valor que requerimos hallar es el torque nominal que necesita nuestro sistema para mover el sistema robótico en reposo.

$$Tn = r\ .\mu_e\ \ .m\ .g \ldots\ldots\ldots (7)$$

Siendo:

r = 0.19 m.

m = 70 kg.

g = 9.81 m/s^2

μ_e = 0.3

Se establece el valor de coeficiente de fricción estática ($\mu_e = 0.3$), el cual fue basado en la consulta de tablas empíricamente establecidas con los valores de diferentes coeficientes dependiendo de las superficies de trabajo que se encuentran en contacto. En nuestro caso, se establece el contacto entre la goma de la rueda y el suelo del ambiente público (cemento pulido). En la siguiente tabla donde se muestra los coeficientes de resistencia estática y dinámica de diferentes materiales.

Tabla N° 16

Tabla de coeficiente de fricción de materiales ($\mu_e, estática, \mu_d, dinámica$)

Materiales en contacto	μ_e	μ_d
Acero // Hielo	0.03	0.02
Cobre// Hierro	1.1	0.3
Madera // Madera	0.7	0.4
Caucho // Cemento (seco)	1	0.8
Caucho // Cemento (Pulido)	0.3	0.25
Vidrio // Madera	0.2	0.25
Esquí (encerado) // Nieve (0°)	0.1	0.05
Acero // Acero	0.15	0.09

Fuente: Elaboración propia

Del cuadro anterior es posible distinguir que los valores más cercanos al cero (0) son los materiales que entre si tienen menor fuerza de fricción al estar en contacto, a la vez los materiales con variable igual o mayor a uno (1) tienen mayor fuerza de fricción al contacto entre ellos.

Continuando, se realizo la sumatoria de fuerza en condiciones de equilibrio. Para cambiar del estado de reposo al robot móvil es necesario hallar el valor de torque mínimo necesario para que este entre en movimiento. Este torque debe ser mayor a la fricción por el radio de la rueda, estableciendo la siguiente ecuación para la obtención del resultado.

$$r \,.\mu_e \,.\, m \,.\, g \, < Tn \ldots\ldots\ldots (8)$$

Así mismo, teniendo los datos ya especificados luego de realizado los cálculos, se obtiene el torque mínimo necesario. Para la selección de nuestro motor se tendrá que considerar aquel que cumpla con tener un torque mayor al calculado. La masa nominal que se toma para los cálculos (m=72 kg).

$$39.14\ Nm \ < Tn \ldots\ldots (9)$$

Debido a que con el resultado anterior se obtiene el torque nominal de todo el sistema, es necesario descomponer este con la ayuda un diagrama de cuerpo libre en los torques nominales de cada motor, obteniendo la siguiente ecuación:

$$Tn_1 = Tn \div 2 \ldots\ldots (10)$$
$$Tn_2 = Tn \div 2 \ldots\ldots (11)$$

De las ecuaciones se calcula los torques para ambos motores, dando como resultado lo siguiente:

$$Tn_1 > 19.57\ Nm \ldots\ldots (12)$$
$$Tn_1 > \ 199.56\ kg.cm$$
$$Tn_2 > 19.57\ Nm \ldots\ldots (13)$$
$$Tn_2 > 199.56\ kg.cm$$

Cálculo de la potencia requerida

Para llegar a la velocidad máxima deseada, se calcula el valor de la potencia requerida, tomando en cuenta la velocidad máxima de nuestro sistema robótico que es de 4 km/h, es decir 1.11 m/s

$$P_{vmax} = F \,.V_{max} \ldots\ldots (14)$$

Pvmax= Potencia requerida

F= Fuerza de empuje

Vmax=Velocidad máxima

$$F = \mu_e \ . m \ . g \ \ldots\ldots (15)$$

$$F = \mu_e \ . m \ . g$$

$$F = 130.47 \ N$$

La fuerza de empuje será dividida en los dos motores que llevará el sistema robótico, quedando 65.24 N de fuerza de empuje que tendrá que superar cada motor eléctrico.

$$P_{vmax} = (65.24N). (1.11 \ m/s)$$

$$P_{vmax} = 72 \ W = 0.072 \ kW$$

Cálculo de la velocidad angular

Para calcular la velocidad angular se debe de contar con la velocidad máxima que requiere el sistema robótico. Para este caso se tomó 4 km/h de acuerdo al estado tecnológico antes mencionado, para el cálculo se utiliza la siguiente expresión:

$$\omega = \frac{V_{max}}{r} \ldots\ldots\ldots\ldots (16)$$

$$Donde:$$

$$\omega = Velocidad \ angular$$

$$r = Radio \ del \ neumático$$

Sustituyendo los valores, se obtiene:

$$\omega = \frac{1.11 \ m/s}{0.19 \ m}$$

$$\omega = \frac{1.11 \ m/s}{0.19 \ m}$$

$$\omega = 5.84 \ rad/s$$

Para la elección de motores comerciales, se realiza la conversión de velocidad angular a rpm (revoluciones por minuto), con la siguiente ecuación.

$$Wrpm = \frac{\omega \ . (60)}{2\pi} \ldots\ldots\ldots\ldots (17)$$

$$Wrpm = \frac{5.84\ .\ 60}{2\pi}$$

$$Wrpm = 55.77\ rpm$$

Finalmente, realizado la conversión de Newton-metro a Kilogramo-centímetro, sabiendo que el torque nominal de cada motor para poder movilizarse un peso máximo de 70kg es de 72W y un torque de 199.56 kg.cm y teniendo en cuenta los valores obtenidos, procedemos a buscar motores comerciales que tengan un torque nominal mayor al calculado.

-Voltaje de operación:6V - 12V
-Máximo torque: 200kg.cm
-Velocidad máxima: 60 RPM

Se muestra información técnica de motores eléctricos con reductor de velocidad comerciales, los cuales son funcionales para el proyecto.

Tabla N° 17

Información técnica de motores 12v con reductor

Marca	Ningbo Current	Jiansu Devo	DongZhan
			
Procedencia	China	China	China
Modelo	GDM08	NMRV40	5D90+NMRV
Voltaje	12v dc	12v dc	12v dc
Velocidad	30-250 rpm	0.1-600 rpm	100 rpm (variable)
Torque output	14.95 Nm	8.96 Nm	11.95 Nm
Precio FOB	$86	$100	$78

Fuente: Página Web “Alibaba” (2022)

3.4.5 Diseño en Programa Cad – SolidWorks

Para la implementación y simulación de las diferentes piezas del robot se seleccionó el programa Solid Works, los planos constructivos se adjuntan en el anexo, a continuación, se mostrarán las figuras de cada pieza del robot móvil realizadas en el software Cad como parte de su estructura.

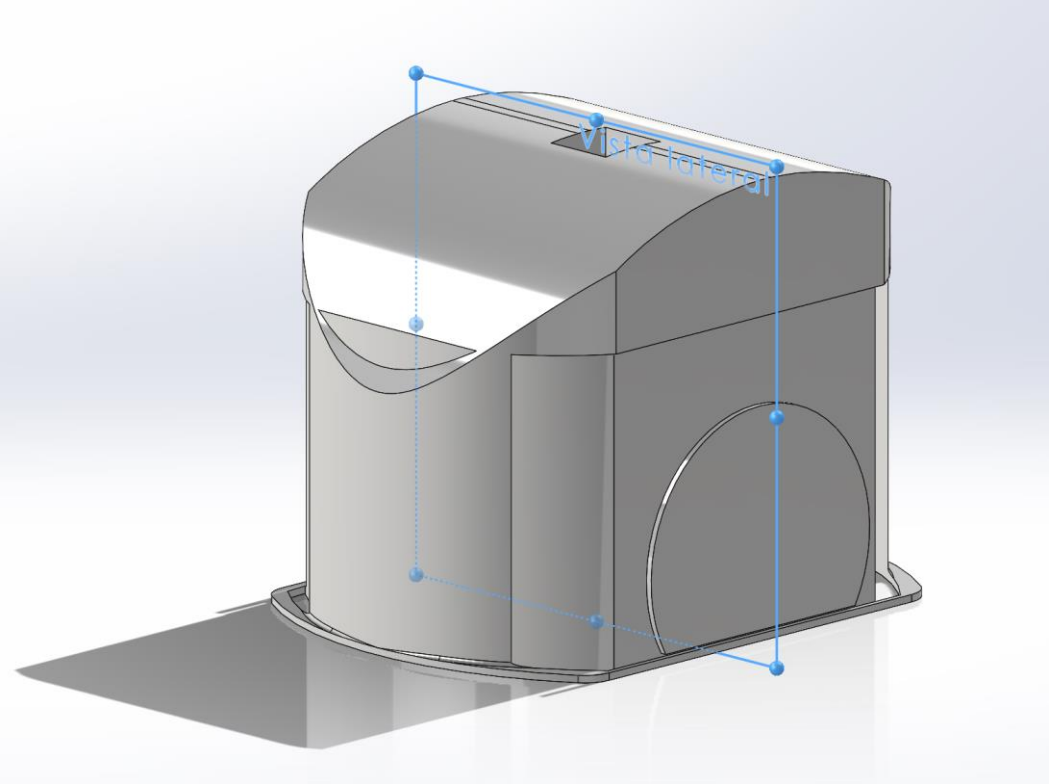


Figura 30:Chasis Inferior – Vista Solido

Fuente: Elaboración propia.

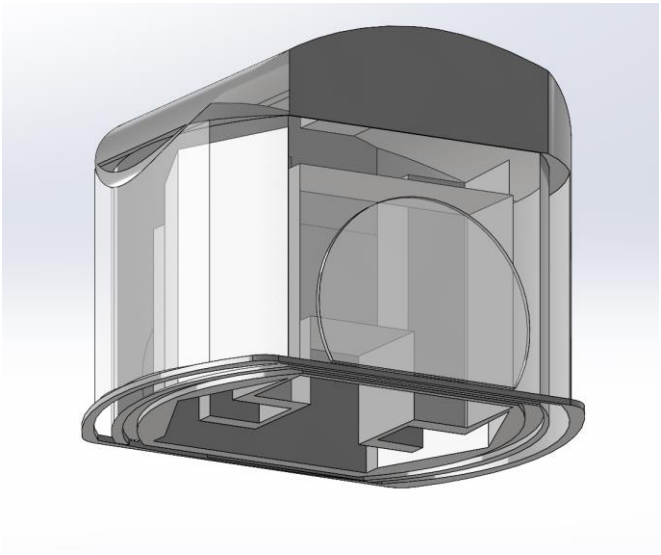


Figura 31:Chasis Inferior – Vista Alojamientos internos para equipos eléctricos, electrónicos y mecánicos

Fuente: Elaboración propia.

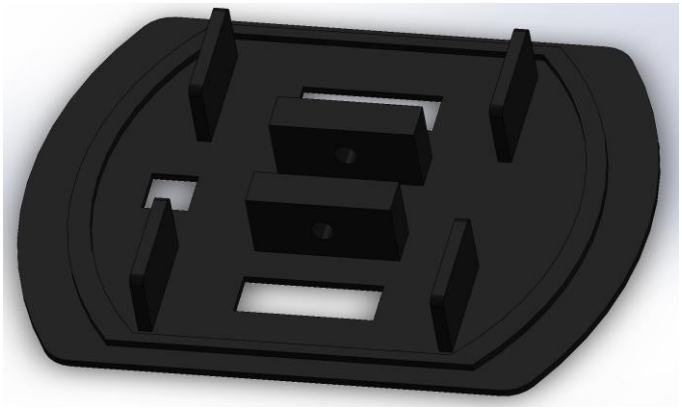


Figura 32:Chasis Base – Soporte de equipamiento

Fuente: Elaboración propia.

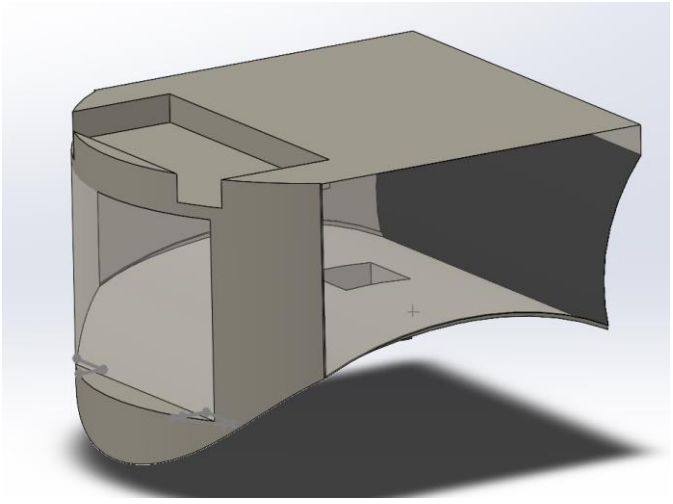


Figura 33:Chasis Intermedio – Suministro de mascarillas

Fuente: Elaboración propia.

Figura 34:Tubo Pasacables

Fuente: Elaboración propia.

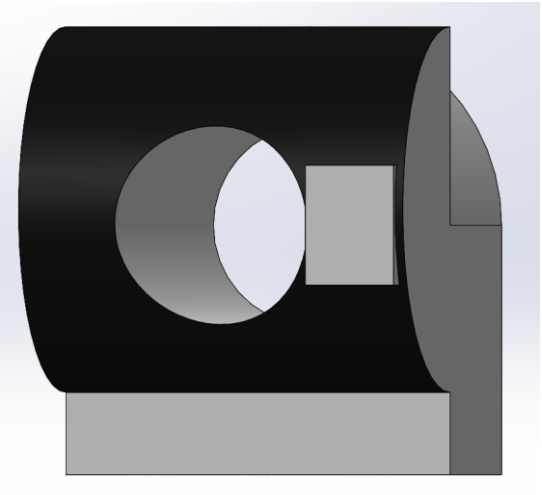


Figura 35:Alojamiento del sensor de temperatura corporal y cámara – Vista frontal

Fuente: Elaboración propia.

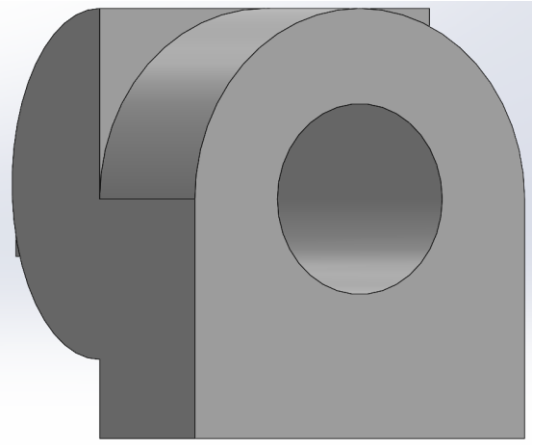


Figura 36:Alojamiento del sensor de temperatura corporal y cámara – Vista frontal Posterior

Fuente: Elaboración propia.

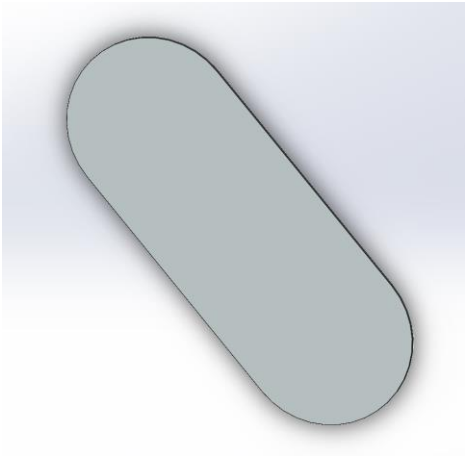


Figura 37:Visor Protector de Sensor

Elaboración propia.

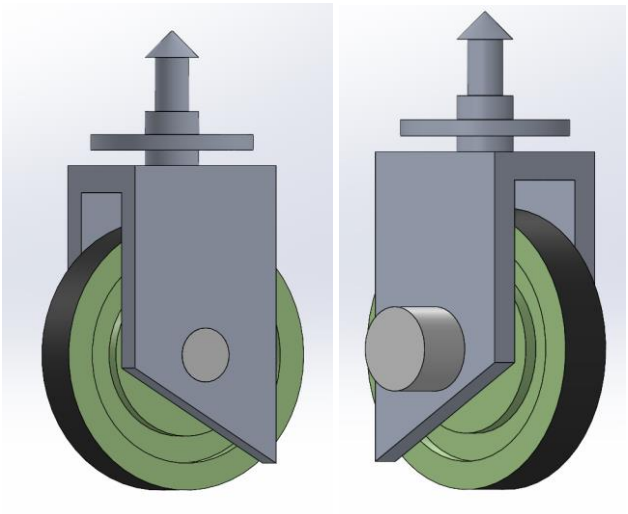


Figura 38:Rueda Loca – Vistas laterales

Fuente: Elaboración propia.

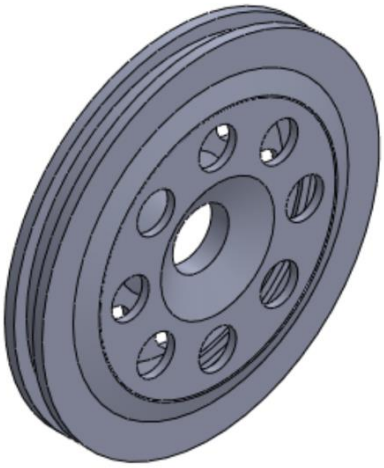


Figura 39: Aro de Rueda fija.

Fuente: Elaboración propia

Figura 40:Llanta de Rueda fija.
Fuente: Elaboración propia.

3.5 Sistema Eléctrico – Electrónico

A) Alimentación del Circuito

En el esquema mostrado en la Figura 41 presenta la forma en cómo se realizaron las conexiones en el sistema eléctrico-electrónico: El Raspberry Pi será alimentado por una Power Bank de valores especificados, permitiendo así también el funcionamiento de la cámara web y el sensor de temperatura, los cuales ambos requieren trabajar con visión artificial. A su vez, este alimentará al Arduino, el cual controlará el sensor de línea y el regulador de voltaje para los motores. Finalmente, los motores serán alimentados por las baterías de litio de valores especificados previamente.

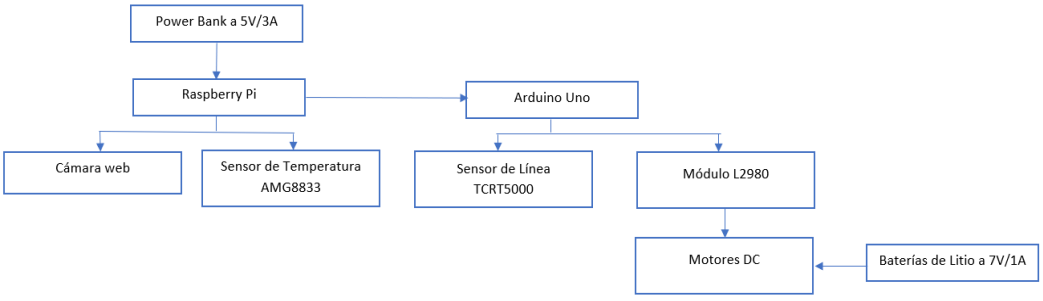


Figura 41: Esquema de conexiones del sistema eléctrico-electrónico
Fuente: Elaboración propia.

El esquema mostrado en la Figura 42 presenta la forma en cómo se realizaron las conexiones de las baterías con el Raspberry, así como también su conexión con

los motores para su accionamiento:

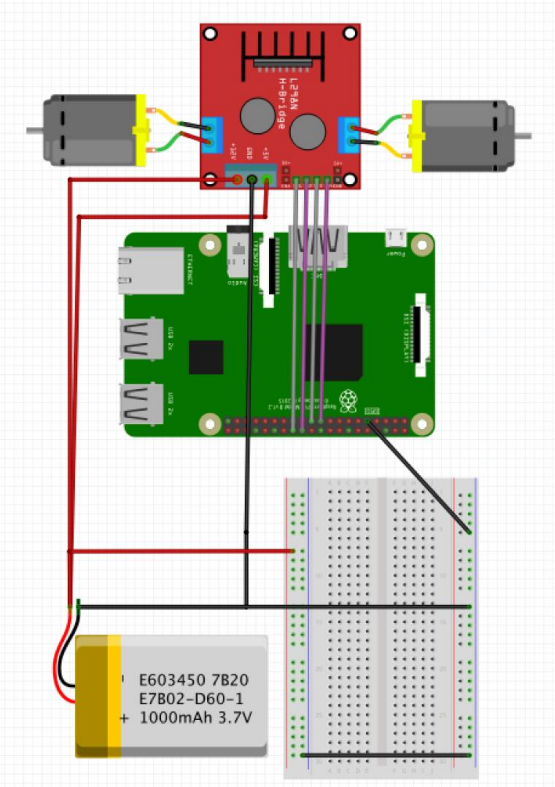


Figura 42:Ejemplo de conexión de los motores DC

Fuente: Benavente (2022)

Para determinar el consumo del Raspberry Pi al momento de encenderse, utilizamos la fórmula de potencia en base a los valores establecidos del Power Bank, de la siguiente manera:

- V = 5 V, Ir = 1400 mA
- $Potencia$: Pr $= V\ x\ Ir = 1400 * 10^{-3} * 5 = 7\ W \ldots$ (20)

A continuación, calculamos el tiempo de duración de la Power Bank con las siguientes variables y ecuaciones, considerando que la corriente puede encontrarse en el siguiente rango establecido:

- Corriente Ib = 20000 mA/H ~ 12800 mA/H
- Capacidad máxima:

$$T = \frac{Potencia\ de\ entrega\ de\ la\ batería}{Potencia\ de\ consumo\ del\ Raspberry\ Pi} = \frac{Ib*Vb}{Pr} \ldots (21)$$

$$T = \frac{20 * 5}{7} \sim \frac{12.8 * 5}{7} = 14h \sim 9h$$

Para la elección del microcontrolador, se considerador los siguientes ventajas y desventajas explicadas en la Tabla 18.

Tabla N° 18

Ventajas y desventajas de los microcontroladores

MICROCONTROLADOR	LENGUAJE	VENTAJAS	DESVENTAJAS
Arduino	C++	Bajo Costo Prototipo Rápido de proyectos Múltiples entradas y salidas Código abierto	Procesamiento limitado Poca memoria de almacenamiento
Raspberry pi	Python	Simplificad o y rápido Portable Cuenta con sistema operativo	Sensibles a la estática Menos puertos USB
PIC	Ensamblador C++	Tamaño pequeño Variedad de modelos Bajo costo	Susceptible al ruido electromagnético Software especializado

Fuente: Elaboración propia.

Teniendo en cuenta las características del Raspberry y Arduino, además de su facilidad de comprender su programación, se optó por estos dos microcontroladores para el sistema electrónico.

3.6 Programación

3.6.1 Diagrama de Flujo

Para iniciar la programación que dirigirá y controlará un sistema automatizado, como lo es este robot móvil, primero se debe definir las funciones que este realizará en base a las variables que se desean detectar y el resultado que se desea lograr. En la Figura 43 podemos observar el Diagrama de Flujo del Robot Móvil que desarrolla esta tesis, la cual se explicará de la siguiente manera:

1) El sistema al encenderse inicia en un estado de reposo, inmóvil en una posición por la que pueda recorrer una línea recta marcada previamente en el suelo del lugar. Al mismo tiempo, la cámara provista de visión artificial enfocará el área por donde los clientes o usuarios del negocio ingresarán por fila, de manera ordenada y uno a la vez.
2) Mientras el procesamiento de imagen pueda detectar que los clientes porten una mascarilla, la cuales deberán cubrir la mitad inferior de sus rostros durante todo el tiempo de ingreso, no se realizará ninguna acción.
3) En caso contrario de no portar una mascarilla durante el sensado, se dará un aviso sonoro de infracción, con una duración no mayor de cinco segundos para alertar al personal y a los clientes.
4) Cualquiera haya sido la acción, se procederá a realizar un sensado de la temperatura corporal de la persona que desea ingresar. Para ello, el robot móvil se acercará a esta persona gracias al accionar de sus motores y se desplazará guiado por su sensor seguidor de línea, la cual se mencionó que debe estar demarcada desde un inicio.
5) Los motores se apagarán en un leve frenado, ya que no empleará mucha velocidad de desplazamiento, cuando haya alcanzado el final de la línea demarcada. En ese punto, estará a una distancia cercana del cliente para poder detectar su temperatura mediante el sensor infrarrojo.
6) En caso la temperatura sea menor a 38°C, el robot móvil accionará sus motores para retroceder y regresar a su posición original, permitiendo que el cliente ingrese sin mayor contratiempo.
7) No obstante, si la temperatura es mayor a 38°C, este es un indicio de un

posible síntoma de COVID-19 u alguna otra enfermedad contagiosa de riesgo para los demás clientes del lugar, por lo que se procederá a dar un aviso luminoso de corto tiempo para avisar al personal de este caso y ellos puedan tomar las medidas pertinentes del caso.

8) No es necesario tele operar al robot, ya que sus funciones están desarrolladas de manera automática.

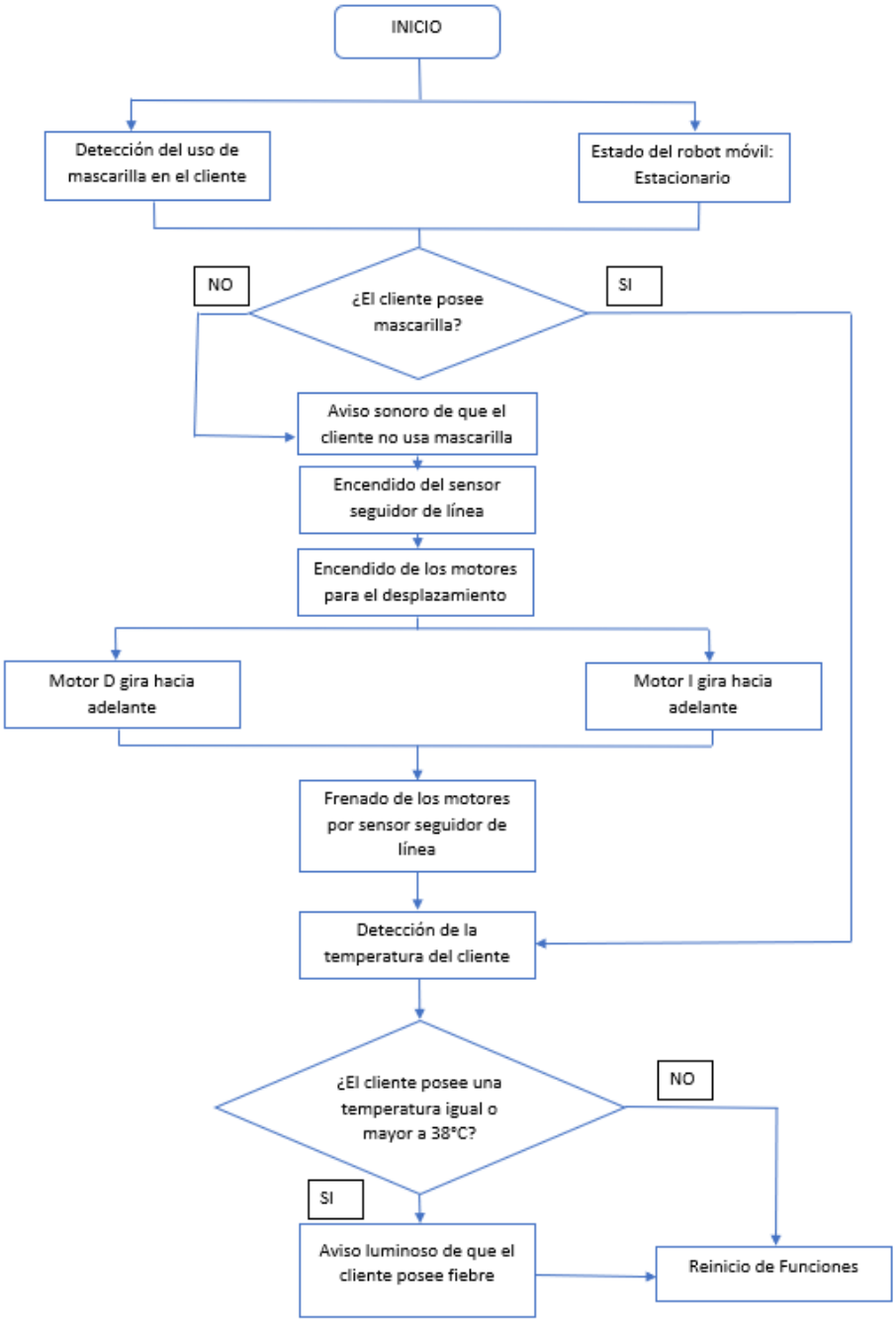


Figura 43:Diagrama de flujo del robot móvil un sistema de reconocimiento facial.

Fuente: Elaboración propia.

Tabla N° 19

Resumen de selección de componentes

Elemento	Planteamiento
Tipos de ruedas	2 ruedas fijas y 1 rueda loca
Configuración mecánica	3 ruedas –diferencial
Microcontrolador	Raspberry PI
Motor	Motorreductor de Metal 25d 6-12v DC
Sensor	AMG8833

Fuente: Elaboración propia.

CAPÍTULO IV: DISCUSIÓN DE RESULTADOS

4.1 Pruebas

4.1.1 Sistema mecánico

Se realizo las simulaciones de dimensiones como es que nuestro sistema robótico estaría representado junto a una persona de una altura de 1.90 m. La siguiente figura muestra el detalle descrito.

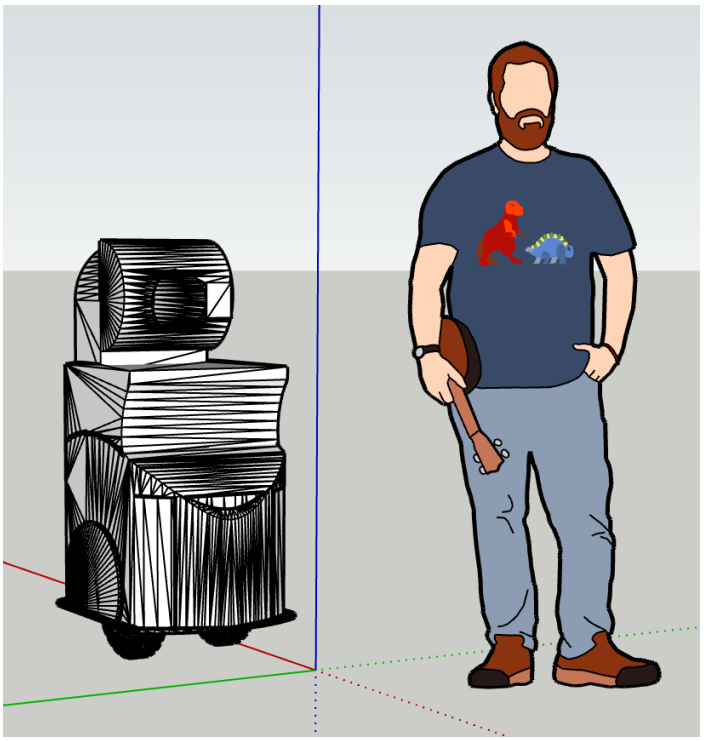


Figura 44: Simulación en programa Sketchup

Fuente: Elaboración propia.

En seguida, de acuerdo al modelo matemático cinemático de un robot de 3 ruedas diferencial, se trabajó mediante el programa Matlab para el procesamiento de nuestras ecuaciones y la simulación respectiva de los movimientos semi holonómicos que tendrá el sistema robótico. Para la ejecución de estas pruebas de movimientos, desde el programa SolidWorks se guardó cada una de nuestras piezas en formato ".stl", para que estas puedan ser procesadas como vértices y ceros realizando la creación de una función. Se muestran los nombres de los script creados para la ejecución de simulación del modelo cinemático de nuestro robot.

- Main.m
- MobileRobot.m
- MobilePlot.m
- stlRead.m
- Prueba_cinematica1
- Prueba_cinematica2

Los resultados de las pruebas fueron las siguientes:

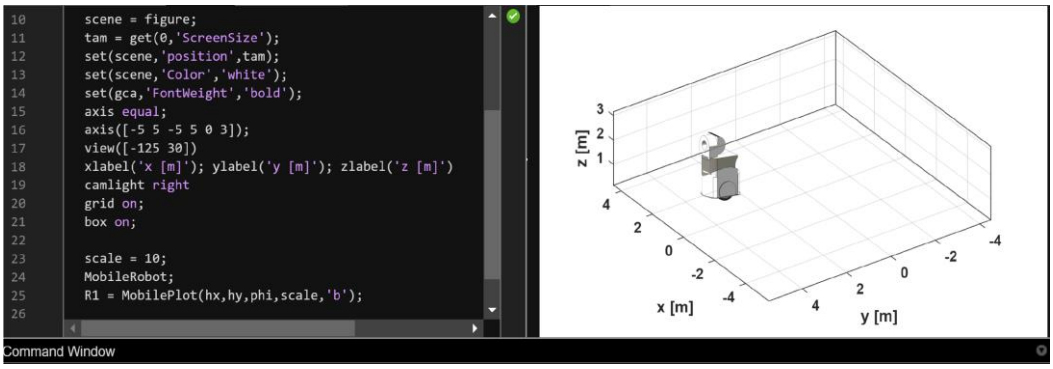


Figura 45:Script Main.m

Fuente: Elaboración propia.

Para las pruebas, se puede partir de diferentes valores ingresados en nuestras variables velocidad lineal y velocidad angular, como se verifica en la siguiente figura:

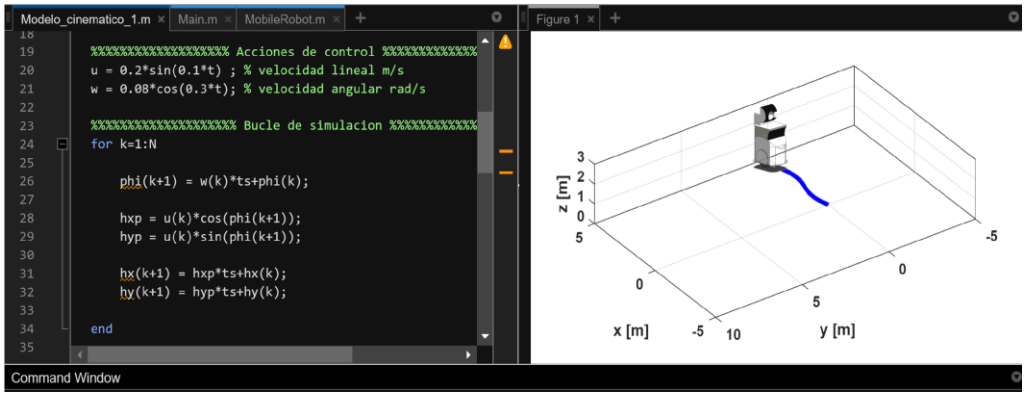


Figura 46:Script Prueba_cinematica1.m

Elaboración propia.

El siguiente comportamiento de robot se ejecutó con una velocidad inicial de 0.1

m/s y una velocidad angular de 0.08 rad/s, dando como resultado el siguiente comportamiento:

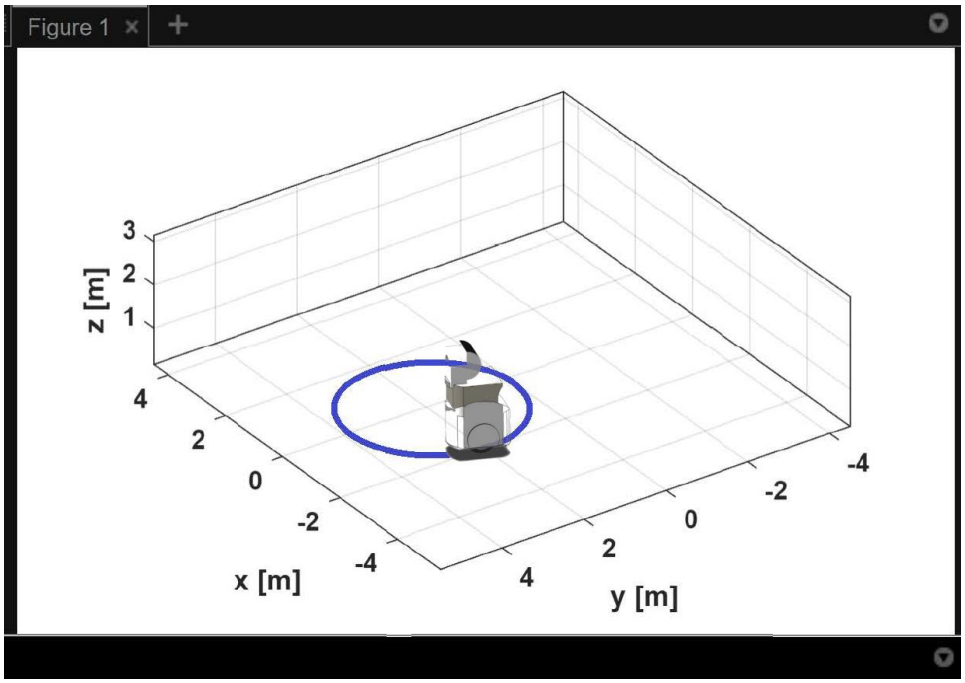


Figura 47:Script Prueba_cinematica2.m

Fuente: Elaboración propia.

Se ingresaron diferentes variables para verificar el comportamiento cinemático y se anexan todos los códigos procesados en el programa Matlab.

Al respecto de las pruebas mecánicas, como se indica anteriormente, el material por el cual se conforma el chasis del robot es de ABS, se muestra la selección de material en el programa CAD SolidWorks

Figura 48:Elección de material SolidWorks

Fuente: Elaboración propia.

Espesor de la plancha de ABS media pulgada (1.27cm)

4.1.2 Sistema eléctrico-electrónico

Las simulaciones para corroborar el correcto manejo y conexión del sistema electrónico se realizaron en la plataforma LabVIEW, donde se consiguió primero la esquematización del circuito que compondría al Arduino. En la Figura 49 podemos observar cuáles son sus entradas y salidas pertinentes para el sensor de línea TCR5000 y el regulador de voltaje L298D que conecta con los motores. Así mismo, también observamos las salidas de este regulador que conectan con los motores.

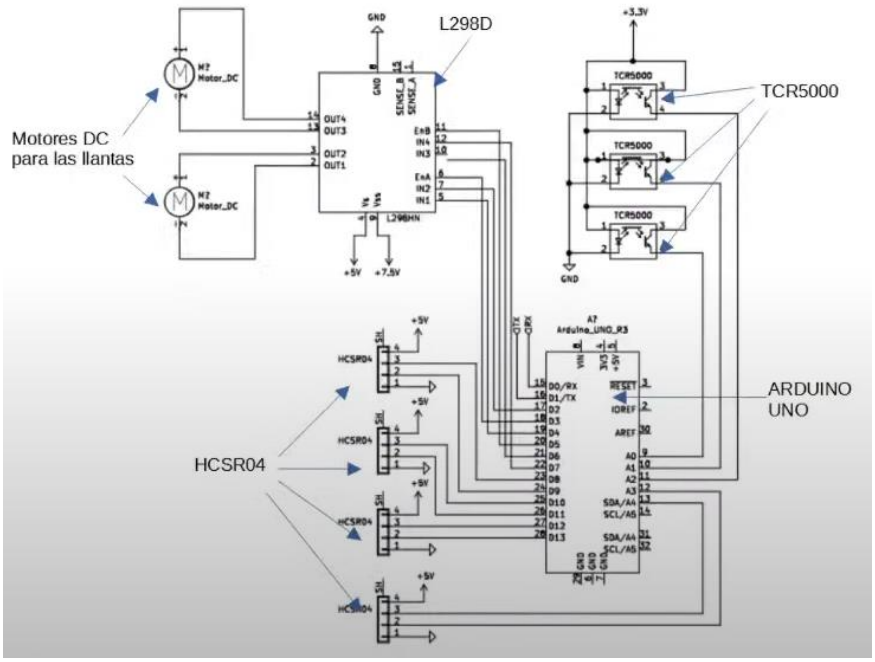


Figura 49:Diagrama de flujo del robot móvil un sistema de reconocimiento facial.
Fuente: Benavente (2022)

4.1.3 Programación

El primer elemento del sistema que se programó fue el Arduino con respecto al sensor seguidor de línea TCR5000. La programación se ve reflejada en la Figura 50.

```
// incluir bibliotecas
#include "sonar.h"
#include "seguidorLinea.h"
#include "serialComunication.h"

// variable para almacenar comando serial
char data;

void setup() {
  pinMode(s1, INPUT);
  pinMode(s2, INPUT);
  pinMode(s3, INPUT);
  pinMode(enab1, OUTPUT);
  pinMode(enab2, OUTPUT);
  pinMode(motor1, OUTPUT);
  pinMode(motor2, OUTPUT);
  pinMode(motor3, OUTPUT);
  pinMode(motor4, OUTPUT);
```

Figura 50: Programación del Seguidor de Línea TCR5000

Fuente: Elaboración propia.

Para la programación del Raspberry Pi, se empleó la librería Open CV con la intención de utilizar visión artificial para la detección de mascarillas y sensado de temperatura corporal. Para ello, primero se tuvo que realizar un entrenamiento al programa de reconocimiento, en el cual se incluyó dos carpetas donde se colocaran una gran muestra de imágenes de personas: Un grupo que portara mascarillas y otro grupo que no usara mascarillas.

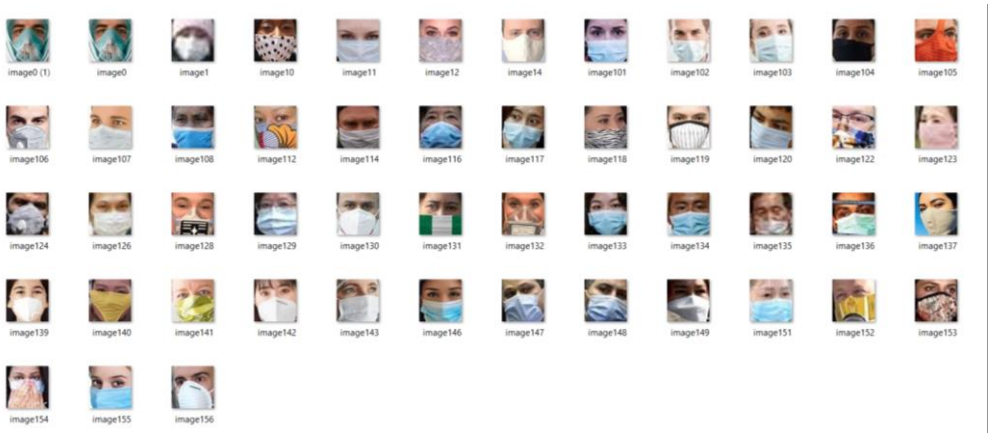


Figura 51: Carpeta de imágenes con personas usando mascarillas.

Fuente: Elaboración propia.

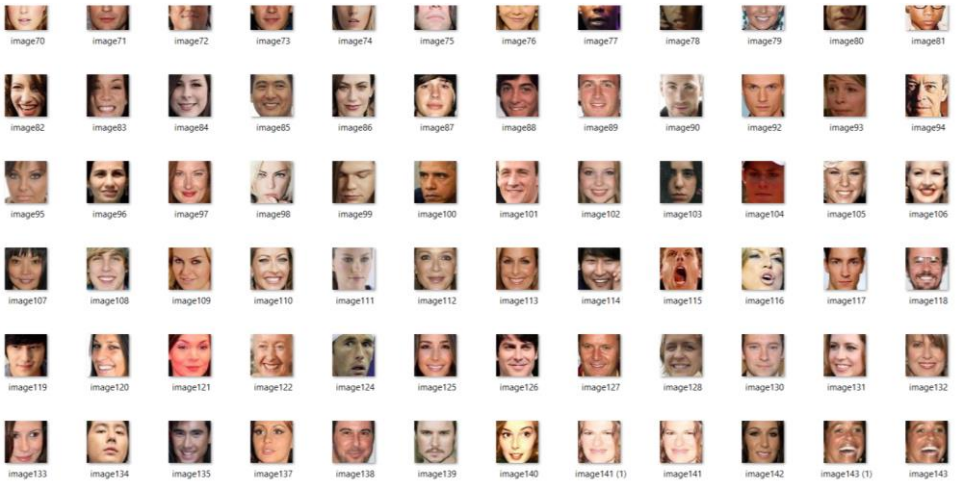


Figura 52: Carpeta de imágenes con personas sin mascarillas.

Fuente: Elaboración propia.

En base a estas carpetas, se organizó un entorno virtual en Python con el nombre de “Detección de Mascarillas y Sensado de temperatura”, con el cual se facilitaría la dirección de carpetas. Luego, se implementó la programación que se expone en la Figura 53 para llevar a cabo la captura de video a tiempo real, con la cual se podría identificar si las personas que pasaran por delante suyo portaran mascarilla o no.

```
File  Edit  Format  Run  Options  Window  Help
from tkinter import *
from PIL import Image, ImageTk
import sensorMeasure as sm
import RPi.GPIO as GPIO
import numpy as np
import pandas as pd
import pygame, time, cv2, imutils
import seccionPreguntas
import arduinoComunicacion


def iniciar():
    """ Función para iniciar captura de vídeo"""
    global cap
    cap = cv2.VideoCapture(0)
    visualizar()

def visualizar():
    """ Función para mostrar vídeo en la interfaz de usuario """
    global cap
    # revisar si la variable cap contiene información
    if cap is not None:
        # obtener la imagen de la camara
        ret, frame = cap.read()
        if ret == True:
            # redimencionar imagen a tamaño VGA
            frame = imutils.resize(frame, width=640)
            # convertir a espacio de color RGB
            frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            # convertir imagen tipo Array a tipo Image
            im = Image.fromarray(frame)
            img = ImageTk.PhotoImage(image=im)
```

Figura 53: Programación en Python empleando Open CV.

Fuente: Elaboración propia.

Para la programación de movimiento, al ser un robot móvil de trabajo automatizado, sólo necesitará de una función de avance y retroceso por la línea oscura demarcada que sería detectada por el sensor infrarrojo en la base del robot, por lo que se colocó una programación básica de motores que siguiera los lineamientos del algoritmo general especificado previamente.

Una vez que el robot se posicionará en el extremo final de la línea demarcada, se encontrará a una distancia no mayor de un metro de la persona para poder analizar su temperatura corporal. Como parte de las pruebas, fue necesario la escala de temperatura de una imagen térmica, realizando dos tipos de programación para los colores principales a diferenciar: El color azul y el color rojo.

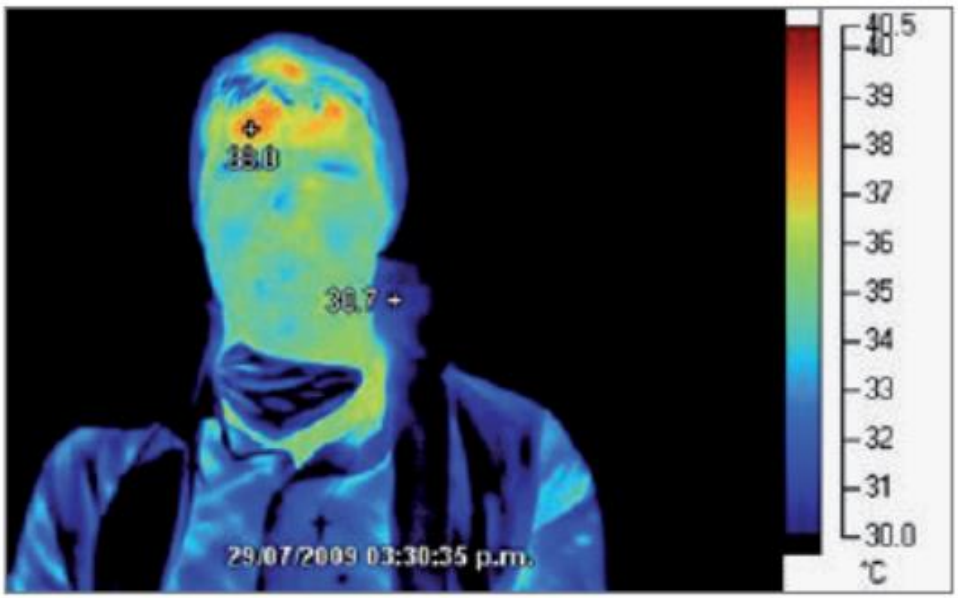


Figura 54: Escala de temperatura de una imagen térmica
Fuente: Elaboración propia.

```python
import cv2
import numpy as np

cap = cv2.VideoCapture(0)

redBajo1 = np.array([0, 100, 20], np.uint8)
redAlto1 = np.array([8, 255, 255], np.uint8)

redBajo2=np.array([175, 100, 20], np.uint8)
redAlto2=np.array([179, 255, 255], np.uint8)

while True: # se inicia la camara  permiendo leer la imagen a cada momento
  ret,frame = cap.read()
  if ret==True:
    frameHSV = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
    maskRed1 = cv2.inRange(frameHSV, redBajo1, redAlto1)
    maskRed2 = cv2.inRange(frameHSV, redBajo2, redAlto2)
    maskRed = cv2.add(maskRed1, maskRed2)
    maskRedvis = cv2.bitwise_and(frame, frame, mask= maskRed)
    cv2.imshow('frame', frame)
    cv2.imshow('maskRed', maskRed)
    cv2.imshow('maskRedvis', maskRedvis)
    if cv2.waitKey(1) & 0xFF == ord('s'):
      break
cap.release()
cv2.destroyAllWindows()
```

Figura 55: Programación de prueba para el color rojo,

Fuente: Elaboración propia.

```python
import cv2
import numpy as np

cap = cv2.VideoCapture(0)

#Limites  y rango del color azul Prueba 2
LimiteinferiorAzul = np.array([100, 100, 20], np.uint8)
LimiteSuperiorrAzul = np.array([125, 255, 255], np.uint8)


while True: # se inicia la camara  permiendo leer la imagen a cada momento
  ret,frame = cap.read()

  if ret==True:
    frameHSV = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
    maskazul = cv2.inRange(frameHSV, LimiteinferiorAzul, LimiteSuperiorrAzul)
    contornos,variable = cv2.findContours(maskazul, cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
    #mostras contornos en el video
    #primera prueba
    #cv2.drawContours(frame, contornos, -1, (0,255,0), 3)

    #seleccion de contornos
    for c in contornos:
        area=cv2.contourArea(c)
        if area > 3000:
          #Buscar el area central del objeto especificado para  mostrarla
          M = cv2.moments(c)
          if (M["m00"]==0): M["m00"]=1
          #valores de x,y
          x = int(M["m10"]/M["m00"])
          y = int(M['m01']/M['m00'])
          # se dibuja el circulo
          cv2.circle(frame, (x,y), 7, (0,255,0), -1)
          #fuente del texto
          font = cv2.FONT_HERSHEY_SIMPLEX
```

Figura 56: Programación de prueba para el color azul.

Fuente: Elaboración propia.

Una vez que la visión artificial podía detectar la diferencia de gamas de colores que la cámara web imprimía, se establecía el rango de temperatura permitido para que la persona fuese descartada de tener fiebre.

Cabe recalcar que la detección de mascarilla y el sensado de temperatura no estarán activadas en simultáneo, sino que el desarrollo de una de las funciones llevará automáticamente a la otra.

4.2 Resultados

4.2.1 Sistema mecánico

Iniciando el proceso de diseño se consideró los dominios para el diseño del robot, de los diferentes tipos de materiales los cuales se evaluaron de acuerdo a requerimientos técnicos necesarios para su construcción, se seleccionó el material (ABS) el cual es usado en diferentes industrias, siendo uno de los más importantes la industria automovilística (parachoques), es importante señalar que siendo un material maleable permite crear un sistema robótico con un peso considerablemente bajo y con una dureza buena para el uso que se requiere, siendo el principal valor de nuestro robot la supervisión, se muestra la siguiente figura la cual muestra las propiedades físicas de nuestro sistema robótico.

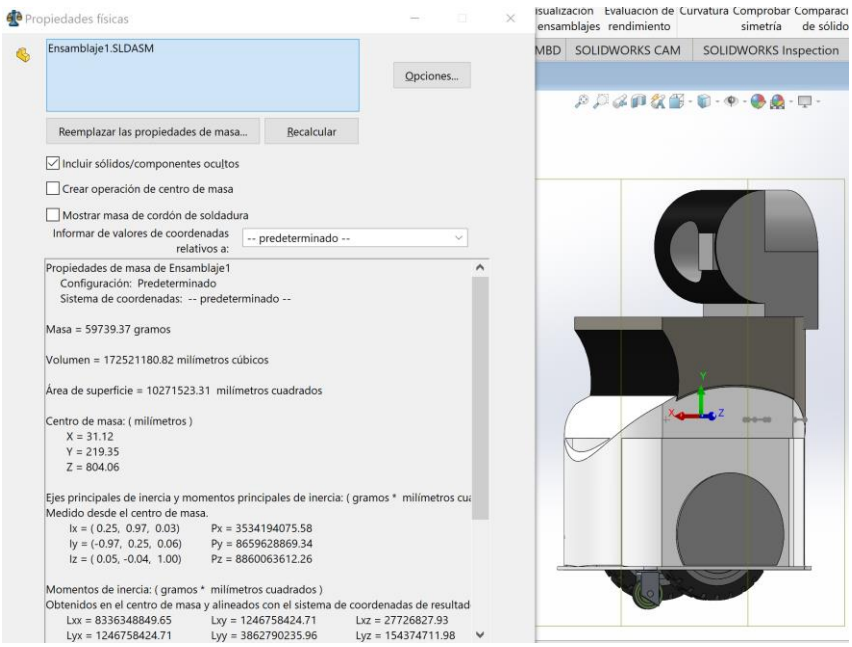


Figura 57:Propiedades físicas del sistema robótico

Fuente: Elaboración propia.

El peso del chasis está dentro de nuestros valores requeridos, como se verifica tiene un valor de 59.74 kilogramos.

Adicionalmente es importante señalar el logro de conseguir un peso considerable ya que este impacta directamente en la selección del motor y, a la vez, en uno de los pilares de los sistemas autónomos, como lo es el gasto energético (capacidad de consumo de baterías). Teniendo en cuenta ello, se logró obtener la autonomía de 8 horas continuas de uso en condiciones ideales.

Finalmente, se consiguió el diseño de un robot que cuenta con espacios para una correcta ventilación de los motores, cuidado de sistema eléctrico y espacios para el montaje los diferentes sensores.

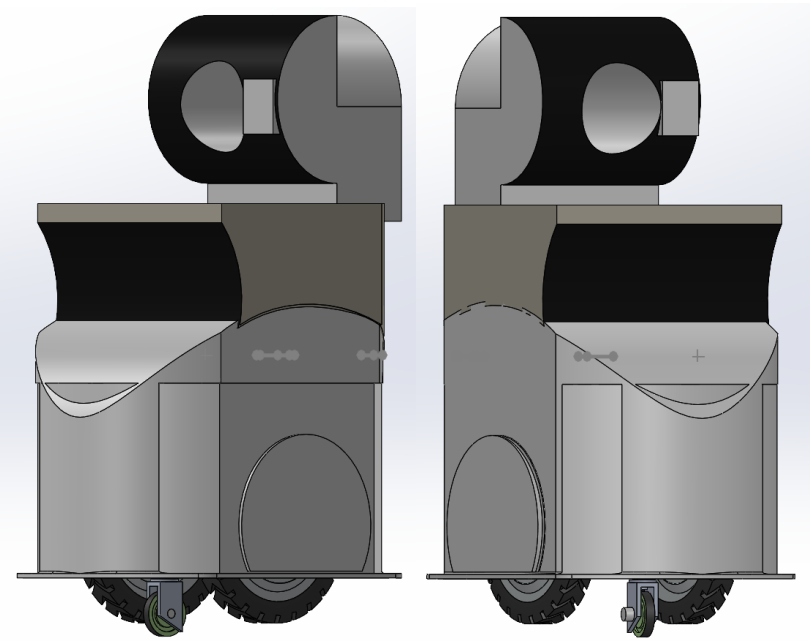


Figura 58:Modelo Robótico en SolidWorks – Vistas laterales

Fuente: Elaboración propia.

En el siguiente cuadro se muestra resumen general del sistema mecánico.

Tabla N° 20

Resumen de selección de componentes

Peso	59.74 kg
Dimensiones (general)	1.40 m x 0.77 m x 0.60 m
Material (Chasis)	Acrilonitrilo butadieno estireno (ABS)

Fuente: Elaboración propia.

4.2.2 Sistema eléctrico-electrónico

En las mediciones del sistema eléctrico con respecto al Arduino, se obtuvieron resultados aproximados a los valores obtenidos de manera teórica, lo cual asegura una energización estable.

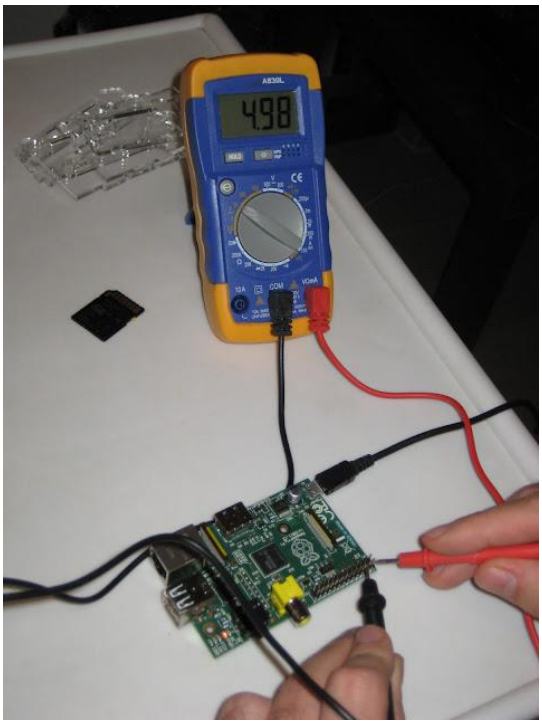


Figura 59:Medición del voltaje de entrada en el Raspberry Pi

Fuente: Elaboración propia.

Así mismo, se realizó una simulación en la plataforma de Protheus de la alimentación de los motores y su regulación de voltaje por medio de Arduino, como se puede observar en la Figura 60. Gracias a la fuente de códigos instalada en la placa, se pudo corroborar la salida de voltaje correcta para ambos motores.

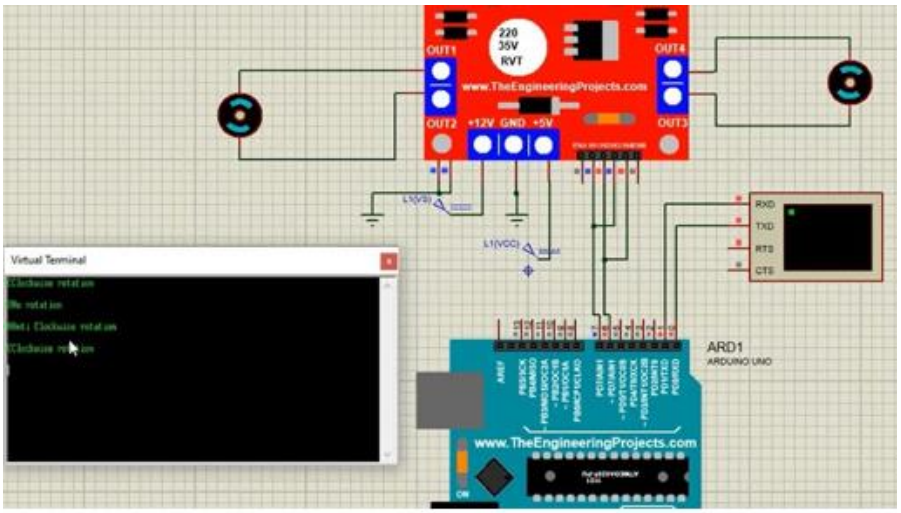


Figura 60:Medición del voltaje de salida para los motores del robot móvil

Fuente: Elaboración propia.

4.2.3 Programación

Para comprobar que la programación funcionaba, realizamos las conexiones pertinentes en la plataforma de Protheus con el Arduino Uno; luego, colocamos el código en la placa y corrimos el programa, como se puede observar en la Figura 61. Gracias a su entorno sencillo de usar, la simulación se dio con normalidad y mostró la activación de los elementos de manera correcta.

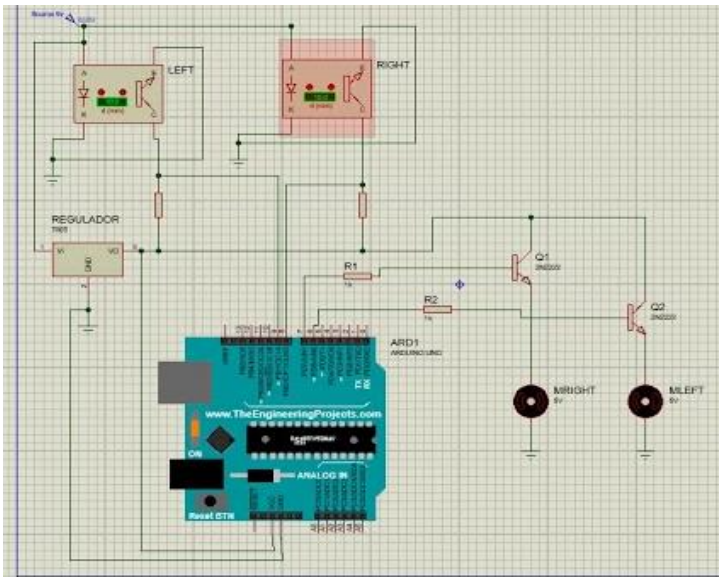


Figura 61: Simulación del encendido y apagado de los motores por medio del seguidor de línea controlado por Arduino Uno.

Fuente: Elaboración propia.

Con respecto al uso de la visión artificial, se pudo obtener una imagen a tiempo real por medio de la cámara web a la conexión en un ordenador, el cual puede comprobarse en la siguiente imagen.

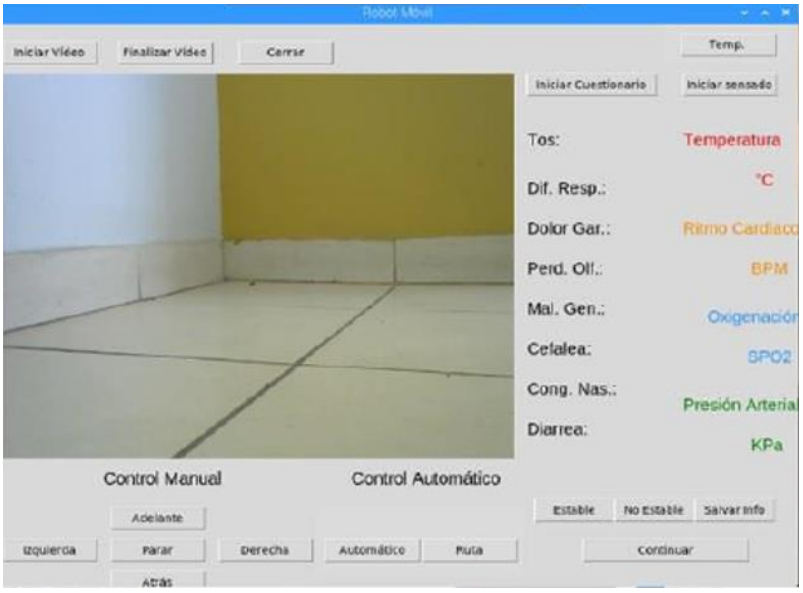


Figura 62: Vista de la cámara web controlada por el Raspberry Pi.

Fuente: Elaboración Propia.

Para la detección de mascarillas, se consiguió que la visión artificial de Raspberry identificara a aquellas personas que no la portaran dando un aviso luminoso color rojo en la pantalla, el cual luego se traduciría en un aviso sonoro para el robot.

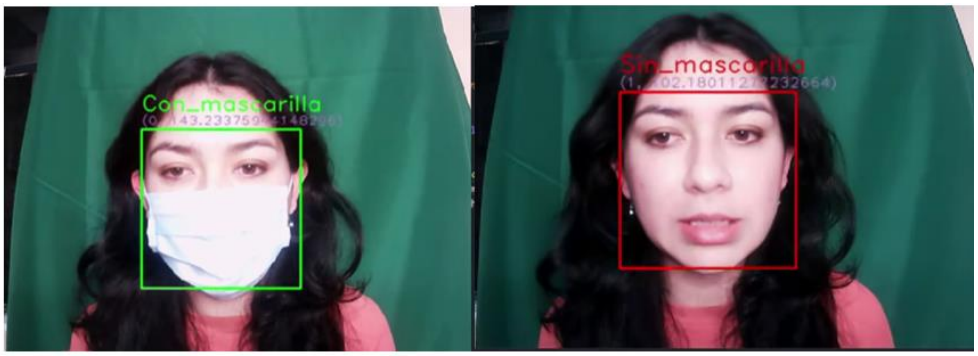


Figura 63: Detección de mascarilla facial por medio de visión artificial.
Fuente: Elaboración propia.

Finalmente, como se explicó en el algoritmo de funcionamiento del robot, en caso de que el usuario no portara mascarilla, el robot se acercaría a tomar su temperatura con la intención de comparar la temperatura medida con la temperatura establecida como límite dentro de la programación. De esa manera, se podría descartar si la persona tenía fiebre o no, lo cual daría aviso de un posible caso de COVID-19.

Figura 64: Detección de temperatura corporal por medio de visión artificial.
Fuente: Elaboración propia.

4.2.4. Presupuesto de los materiales empleados para el robot móvil

Tabla N° 21

Elementos del diseño final del robot móvil y sus costos

Nombre	Cantidad	Precio en soles	Descripción
Raspberry Pi 4	1	350	Tarjeta controladora
Arduino UNO	1	50	Tarjeta controladora
Carcasa	1	1278	Protector del robot
Protoboard	1	10	Conexión de sensores y cables
Micro SD de 64GB	1	30	Memoria para instalar SO de Raspberry Pi
Módulo AMG8833	1	500	Sensor de temperatura
Motorreductor Pololu 25D 6-12 VDC 2A 80RPM	2	180	Movimiento del robot
Rueda con Llanta	2	130	Movimiento del robot
Rueda Loca	1	10	Movimiento del robot
Cámara Web Logitech	1	170	Para reconocimiento facial
Powerbank	1	160	Alimentación de tarjetas de control y sensores
Batería LiPo 7VDC 1 Amp	1	150	Alimentación de motores
Clema de 3	3	4	Conexión de

terminales			alimentación
Interruptor	1	50	Encendido/apagado de robot
Conector XT60 macho	1	10	Conector de alimentación
Cable flexible rojo 30 cm	1	10	Conexión de alimentación
Cable flexible negro 30 cm	1	10	Conexión de alimentación
Cables dupont H-H	2	30	Conexión de sensores
Cables dupont M-H	1	15	Conexión de sensores
	TOTAL:	3147	

Fuente: Elaboración propia.

CONCLUSIONES

1) Se logró diseñar el robot móvil para la detección del uso de mascarillas y sensado de temperatura corporal utilizando visión artificial en lenguaje Python y Raspberry Pi, el cuál es una solución tecnológica que puede ser aplicada para la prevención ante el contagio del COVID-19.
2) Se diseñó el sistema mecánico del robot móvil en la plataforma de SolidWorks, especificando las diferentes piezas que conforman su carcaza, las ruedas y los diferentes cálculos que comprenden su tamaño y peso para una mejor movilidad, es importante destacar el uso de simulación el cual nos permitió visualizar y modificar sin realizar costos económicos significativos y verificar posibles errores.
3) Se definió las características técnicas de la parte electrónica y del suministro eléctrico del robot en base a los parámetros necesarios para el funcionamiento de los instrumentos de detección de mascarillas y sensado de temperatura, entre los que se encuentra el Arduino, Raspberry Pi, Cámara Web, baterías, motores y los sensores de temperatura e infrarrojo.
4) Se desarrolló la programación para la etapa de visión artificial usando el lenguaje Python con el microcomputador Raspberry Pi por medio de la biblioteca virtual Open CV, el cual permitió al robot identificar la imagen de la mascarilla en el rostro de las personas y la temperatura corporal por medio del sensor AMG8833.

RECOMENDACIONES

1) Se recomienda hacer una investigación más exhaustiva en el apartado de programación con la finalidad de conseguir un control remoto por interfaz para el robot móvil y poder controlar su trayectoria con mayor libertad.
2) Los parámetros de medición pueden ser modificados para adaptarse a otros síntomas de alguna enfermedad a futuro que pueda causar nuevamente un estado de emergencia y obligue a los sectores de trabajo a emplear nuevas medidas sanitarias con respecto a sus clientes.
3) Puede adaptarse el control del Raspberry Pi para todos los sistemas electrónicos y reemplazar al Arduino, con lo cual puede reducirse aún más los costos.

REFERENCIAS

ADSONG. (2020). *Digital relative humidity & temperatura sensor DHT11*. ADSONG.

Agurto, D. (2020). *Integración de un sistema de adquisición de datos mediante el uso de un Arduino Mega y Raspberry Pi3 como servidor web y base de datos.* Obtenido de Universidad Nacional de Piura: https://repositorio.unp.edu.pe/bitstream/handle/20.500.12676/2477/IEYT-AGU-VAL-2020.pdf?sequence=1&isAllowed=y

Barrientos, V., García, J., & Silva, R. (2007). Robots móviles: Evolución y Estado del Arte. *Polibits*, 12-17. Recuperado el 08 de 02 de 2021, de https://www.redalyc.org/pdf/4026/402640448003.pdf

BricoGeek. (2021). *Array de sensores infrarrojos QTR-8RC (Digital)*. Obtenido de Sensor Infrarrojo TCRT5000: • https://naylampmechatronics.com/sensores-proximidad/39-sensor-infrarrojo-tcrt5000.html

Camacho, E., & Escobedo, E. (2019). *Sistema Robótico móvil para el transporte de materiales ligeros.* Obtenido de Instituto Politécnico Nacional-Mexico: https://tesis.ipn.mx/jspui/bitstream/123456789/28235/1/tesis_final.pdf

Cognex (2018). *Introducción a la visión artificial*. Massachusetts, Estados Unidos: Editorial Iluminado.

Cooking-hacks. (s.f.). *e-Health Sensor Shield V2.0 for Arduino, Raspberry Pi and Intel Galileo [Biometric / Medical Applications]* . Obtenido de Cooking-hacks: https://www.cooking-hacks.com/ehealth-sensor-shield-biometric-medical-arduino-raspberry-pi.html

Cornejo, J., Vargas, M., & Cornejo-Aguilar, J. (Octubre de 2020). *Aplicaciones innovadoras de la robótica y biomédica en la salud pública durante la pandemia del COVID-19.* Obtenido de Revista Facultad de Medicina Humana URP. 20(4). Pp. 756-757: http://www.scielo.org.pe/pdf/rfmh/v20n4/2308-0531-rfmh-20-04-756.pdf

COVANTEC. (2019). *Programación en Phyton - Nivel básico*. Recuperado el octubre de 2021, de Ventajas y Desventajas: https://entrenamiento-python-basico.readthedocs.io/es/latest/leccion1/ventajas_desventajas.html

Echenique, K. & Vigo, E. (2021). *Las ventajas y desventajas del uso de robots como herramientas para el distanciamiento físico en hoteles*. (Trabajo de investigación) Universidad Peruana de Ciencia Aplicadas, Lima.

Flores, L., & Garay, M. (2017). Diseño e implementación de un robot móvil de desplazamiento autónomo basado en un controlador proporcional derivativo. *(Trabajo de Grado para optar el Título Profesional de Ingeniero Electrónico)*. Callao, Perú: Universidad Nacional Del Callao. Recuperado el 03 de 02 de 2021, de: https://alicia.concytec.gob.pe/vufind/Record/UNAC_f2596c58e132c21bd18f4e868f 740308

García, F. (2013). *Entorno de desarrollo Integrado (IDE*). Recuperado de: www.studocu.co/entorno-de-desarrollo-integrado-ide

Gonzales, S. (2021). *Arduino Ventajas y desventajas*. Recuperado el octubre de 2021, de https://es.scribd.com/document/448301825/ARDUINO-VENTAJAS-DESVENTAJAS-docx

Grayson, D. (2013). *PiBot-B: mobile robot with a Raspberry Pi*. Obtenido de Pololu.com: https://www.pololu.com/blog/233/pibot-b-mobile-robot-with-a-raspberry-pi

Inciso, J. (2021) *Desarrollo de un Sistema de Control de Acceso con Imagen y Temperatura de Trabajadores de la Empresa SEELDOIN E.I.R.L.* (Tesis de pregrado) Universidad Tecnológica del Perú, Lima.

Instituto Nacional de Estadística e Informática (2019) *Principales Indicadores del Sector Turismo.* Recuperado de: https://n9.cl/bb8t5

Kandola, A. (2020). *Medical News Today*. Recuperado el 05 de 02 de 2021, de Causas del Coronavirus; Su origen y cómo se propaga: https://www.medicalnewstoday.com/articles/es/causas-del-coronavirus-su-origen-y-como-se-propaga

Matencio, F. (2021). *Diseño de un sistema robótico móvil para guiar e informar a pacientes en centros médicos*. (Trabajo de investigación) Universidad Pontificia Universidad Católica del Perú, Lima.

Melexis. (2006). *MLX90614 Family*. Recuperado de: https://n9.cl/0xb61

Mochales Mielgo, M. (2020). *Python, el lenguaje de programación más popular de 2020*. Obtenido de Profile: https://profile.es/blog/python/

Monge, M., Fernández, M. y De esteban, J. (2019). *Aeropuertos inteligente: Aceptación de la tecnología por parte de los pasajeros*. (Trabajo de investigación) Universidad de Murcia, España.

Pez Nuss. (2020). *Conociendo Raspberry Pi*. Recuperado el octubre de 2021, de https://peznuss.medium.com/conociendo-raspberry-pi-3dd077b12c92

Raspberry Pi Foundation. (2022). *Raspberry Pi 3A+*. Recuperado de: https://n9.cl/lz4y1

Raspberry Pi Foundation. (2018). *Raspberry Pi Model A+ Datasheet*. Recuperado de: https://n9.cl/jiefj

Reshma, Shilpa, Shubhashree, Dhanraj & Deeksha. (2020). *Virtual Telepresence Robot Using Raspberry Pi. International Journal of Research in Engineering, Science and Management. Recuperado de:* http://journals.resaim.com/ijresm/article/view/99

Reyes, F. (2011). *Robótica: Control de robots manipuladores*. Madrid, España: Editorial

Robokits India. (s.f.). *INFRARED TEMPERATURE SENSOR GY-906 MLX90614*. Obtenido de Robokits India: https://robokits.co.in/sensors/temperature-humidity/infrared-temperature-sensor-gy-906-mlx90614

Santos, P. (2020). Deep learning en la visión artificial. Recuperado de: https://n9.cl/igoyw

Siegwart R y Nourbakhsh R. (2004). *Introduction to autonomous Mobile Robots 2nd*. Massachusetts, Estados Unidos: Editorial Library of congress cataloging.

Solidbi. (2018). *Solidworks qué es y para qué sirve.* Recuperado de: https://n9.cl/8vm6

Tenorio, D. (2017). *Diseño e implementación de un sistema para el mapeo y navegación de un robot móvil*. (Tesis de pregrado) Universidad Autónoma de Occidente, Colombia.

Tipanquiza, H. (2021) *Dispositivo remoto para desinfección de entornos cerrados mediante luz ultravioleta*. (Tesis de pregrado) Universidad Técnica de Ambato, Ecuador.

Torres, L. (2012). *Introducción a la robótica.* Ciudad de México: Editorial Éxodo. Alfaomega.

Unit Electronics. (2021). *Sensor de Temperatura Infrarrojo GY-906 MLX90614.* Recuperado de: https://www.studocu.com

Universidad Católica de Santiago de Guayaquil. (2018). *Componentes y Sistemas Electrónicos*. Recuperado de: https://www.studocu.com

Universidad Nacional de Ingeniería. (2019). *Sensores de Temperatura*. Recuperado de: https://www.studocu.com

Vázquez, A., Ramos de la Flor, F., Fernández, R. (2015). *Robótica educativa.* Madrid, España: Editorial RA-MA.

Zuñiga, I. (2019). *¿Qué es el Deep Learning? Guía práctica con ejemplos*. Recuperado de: https://solid-bi.es/deep-leerning-guia-ejemplos

ANEXOS

Anexo 1: Código completo de movilidad del robot móvil

```
from tkinter import *
from PIL import Image, ImageTk
import sensorMeasure as sm
import RPi.GPIO as GPIO
import numpy as np
import pandas as pd
import pygame, time, cv2, imutils
import seccionPreguntas
import arduinoComunicacion
def iniciar():
global cap
cap = cv2.VideoCapture(0)
visualizar()
def visualizar():
global cap
# revisar si la variable cap contiene información
if cap is not None:
# obtener la imagen de la cámara
ret, frame = cap.read()
if ret == True:
# redimensionar imagen a tamaño VGA
frame = imutils.resize(frame, width=640)
# convertir a espacio de color RGB
frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
# convertir imagen tipo Array a tipo Image
im = Image.fromarray(frame)
img = ImageTk.PhotoImage(image=im)
# mostrar vídeo
lblVideo.configure(image=img)
lblVideo.image = img
```

```
lblVideo.after(10, visualizar)
else:
lblVideo.image = ""
cap.release()
def finalizar():
""" función para terminar captura de vídeo """
global cap
cap.release()
def bpmMeasure():
189
""" funciones para obtener datos de sensores """
# obtener valor de
data = sm.getBpm()
while data == None:
print("repetir nuevamente")
data = sm.getBpm()
dataBpm = data[0]
dataSpo2 = data[1]
return dataBpm, dataSpo2
def tempMeasure():
dataTemp = sm.getTemp()
return dataTemp
def arteryPressMeasure():
dataPress = sm.getArteryPressure()
return dataPress
# función para cerrar ventana
def cerrar():
root.destroy()
class contador():
# constructor
def __init__(self):
self.cuenta = 1
def incrementar(self):
# incrementa el contador
```

```
self.cuenta += 1
def getConta(self):
# regresar valor actual del contador
return self.cuenta
conta = contador() # inicializar contador
cap = None
root = Tk()
root.geometry("1000x720") # tamaño de ventana
root.title('Robot Móvil') # título de ventana
# variables para la interfaz
temperature = StringVar()
bpm = StringVar()
spo2 = StringVar()
arteryPress = StringVar()
respTos = StringVar()
respRespiracion = StringVar()
respOlfato = StringVar()
respMalestar = StringVar()
respCefalea = StringVar()
respCongestión = StringVar()
respDiarrea = StringVar()
respGarganta = StringVar()
contadorPacientes = StringVar()
190
def salvarInfo(*args):
preguntas = ["Tos", "Dificultad para respirar", "Dolor de
garganta", "Perdida de olfato",
"Malestar general", "Cefalea", "Congestión nasal",
"Diarrea", "Temperatura (°C)",
"Ritmo cardíaco (Bpm)", "Oxigenación (Spo2)",
"Presión arterial (Kpa)", "Fecha y hora"]
fechaHora = time.strftime("%H:%M:%S") + ", " +
time.strftime("%d/%m/%y")
respuestas = [respTos, respRespiracion, respGarganta,
```

```
respOlfato, respMalestar, respCefalea,
respCongestión, respDiarrea, temperature, bpm,
spo2, arteryPress, fechaHora]
tabla = []
for i in range(len(respuestas)):
data = [preguntas[i], respuestas[i]]
tabla.append(data)
# salvar datos del paciente
hora = time.strftime("%H:%M:%S")
fecha = time.strftime("%d-%m-%y")
numeroPaciente = str(conta.getConta())
nombreArchivo = "paciente: " + numeroPaciente +" " + hora + "_"
+ fecha
df = pd.DataFrame(tabla, columns = ['Preguntas', 'Respuestas'])
df.to_csv("/home/pi/Desktop/Archivo de
pacientes/"+nombreArchivo+".csv")
def continuar(*args):
# limpiar valor de variables
time.sleep(1)
moverContinuar() # poner en posición
conta.incrementar() # aumentar el valor del contador en 1
time.sleep(0.8)
parar()
def estable(*args):
# obtener temperatura
pygame.mixer.init()
pygame.mixer.music.load("/home/pi/Documents/Test
Sensors/audio/estable.mp3")
pygame.mixer.music.set_volume(0.25)
pygame.mixer.music.play()
while pygame.mixer.music.get_busy() == True:
continue
def noEstable(*args):
# obtener temperatura
```

```
pygame.mixer.init()
pygame.mixer.music.load("/home/pi/Documents/main/audio/noEstable.m
p3")
pygame.mixer.music.set_volume(0.25)
191
pygame.mixer.music.play()
while pygame.mixer.music.get_busy() == True:
continue
# botones principales
btnIniciar = Button(root, text= "Iniciar Vídeo", activeforeground =
'blue', width=10, command=iniciar)
btnIniciar.place(x=5, y=20)
btnFinalizar = Button(root, text= "Finalizar Vídeo",
activeforeground = 'red', width=10, command=finalizar)
btnFinalizar.place(x=150,y=20)
btnCerrar = Button(root, text = "Cerrar", activeforeground = 'red',
width=10, command=cerrar)
btnCerrar.place(x=300,y=20)
btnEstable = Button(root, text = "Estable", fg = 'black',
activeforeground = 'blue', width = 8, command = estable)
btnEstable.place(x = 670, y =590)
btnNoEstable = Button(root, text = "No Estable", activeforeground =
'red', width = 8, command = noEstable)
btnNoEstable.place(x = 770, y = 590)
btnSalvarInfo = Button(root, text = "Salvar Info", activeforeground
= 'blue', width = 8, command = salvarInfo)
btnSalvarInfo.place(x = 870, y = 590)
btnContinuar = Button(root, text = "Continuar", activeforeground =
'blue', width = 20, command = continuar)
btnContinuar.place(x=730, y = 640)
# Ventana de vídeo
lblVideo = Label(root)
lblVideo.place(x=5,y=60)
""" Función para realizar preguntas """
```

```
def quest(*args):
pygame.mixer.init()
pygame.mixer.music.load("/home/pi/Documents/main/audio/intro.mp3")
pygame.mixer.music.set_volume(0.25)
pygame.mixer.music.play()
while pygame.mixer.music.get_busy() == True:
continue
time.sleep(2)
respuestas = seccionPreguntas.makeQuest()
print('resp', respuestas)
respTos.set(respuestas[0])
respRespiracion.set(respuestas[1])
respGarganta.set(respuestas[2])
respOlfato.set(respuestas[3])
respMalestar.set(respuestas[4])
respCefalea.set(respuestas[5])
respCongestión.set(respuestas[6])
192
respDiarrea.set(respuestas[7])
root.update()
""" Función para tomar signos vitales """
def task(*args):
# obtener temperatura
import time,sys
sys.path.append('../')
import amg8833_i2c
import numpy as np
def getTemp():
t0 = time.time()
sensor = []
while (time.time()-t0)<1: # esperar 1 segundo
try:
# AD0 = GND, addr = 0x68 | AD0 = 5V, addr = 0x69
sensor = amg8833_i2c.AMG8833(addr=0x69) # inicilizar
```

```
AMG8833
except:
sensor = amg8833_i2c.AMG8833(addr=0x68)
finally:
pass
time.sleep(0.2) # esperar establecer sensor
temp = 0 # variable para almacenar valor de temperatura
# Si no se encuentra sensor, regresar None
if sensor==[]:
print("No AMG8833 Found - Check Your Wiring")
temp = None
# Si se encuentra sensor, hacer toma de temperatura
else:
pix_to_read = 64 # read all 64 pixels
temperatura = []
for i in range(15):
status,pixels = sensor.read_temp(pix_to_read) # read
pixels with status
if status: # if error in pixel, re-enter loop and try
again
continue
T_thermistor = sensor.read_thermistor() # read
thermistor temp
temperatura.append(max(pixels)) #print(max(pixels))
time.sleep(1)
print(temperatura)
temp = np.mean(temperatura)
return temp
#if __name__ == "__main__":
#
# tempera = getTemp()
""" sección de cuestionario y respuestas """
193
btnQuiz = Button(root, text = "Iniciar Cuestionario",
```

```
activeforeground = 'blue', width = 15, command = quest)
btnQuiz.place(x = 660, y = 60)
lblQuiz1 = Label(root, text = "Tos: ", fg = 'black', font =
('Arial', 15))
lblQuiz1.place(x = 660, y = 130)
lblRespQuiz1 = Label(root, textvariable = respTos, fg = 'blue', font
= ('Arial', 15))
lblRespQuiz1.place(x = 780, y = 130)
lblQuiz2 = Label(root, text = "Dif. Resp.: ", fg = 'black', font =
('Arial', 15))
lblQuiz2.place(x = 660, y = 190)
lblRespQuiz2 = Label(root, textvariable = respRespiracion, fg =
'blue', font = ('Arial', 15))
lblRespQuiz2.place(x = 780, y = 190)
lblQuiz3 = Label(root, text = "Dolor Gar.: ", fg = 'black', font =
('Arial', 15))
lblQuiz3.place(x = 660, y = 240)
lblRespQuiz3 = Label(root, textvariable = respGarganta, fg = 'blue',
font = ('Arial', 15))
lblRespQuiz3.place(x = 780, y = 240)
lblQuiz4 = Label(root, text = "Perd. Olf.: ", fg = 'black', font =
('Arial', 15))
lblQuiz4.place(x = 660, y = 290)
lblRespQuiz4 = Label(root, textvariable = respOlfato, fg = 'blue',
font = ('Arial', 15))
lblRespQuiz4.place(x = 780, y = 290)
lblQuiz5 = Label(root, text = "Mal. Gen.: ", fg = 'black', font =
('Arial', 15))
lblQuiz5.place(x = 660, y = 340)
lblRespQuiz5 = Label(root, textvariable = respMalestar, fg = 'blue',
font = ('Arial', 15))
lblRespQuiz5.place(x = 780, y = 340)
lblQuiz6 = Label(root, text = "Cefalea: ", fg = 'black', font =
('Arial', 15))
```

```
lblQuiz6.place(x = 660, y = 390)
lblRespQuiz6 = Label(root, textvariable = respCefalea, fg = 'blue',
font = ('Arial', 15))
lblRespQuiz6.place(x = 780, y = 390)
lblQuiz7 = Label(root, text = "Cong. Nas.: ", fg = 'black', font =
('Arial', 15))
lblQuiz7.place(x = 660, y = 440)
lblRespQuiz7 = Label(root, textvariable = respCongestión, fg =
'blue', font = ('Arial', 15))
lblRespQuiz7.place(x = 780, y = 440)
lblQuiz8 = Label(root, text = "Diarrea: ", fg = 'black', font =
('Arial', 15))
lblQuiz8.place(x = 660, y = 490) # 820
194
lblRespQuiz8 = Label(root, textvariable = respDiarrea, fg = 'blue',
font = ('Arial', 15))
lblRespQuiz8.place(x = 780, y = 490)
""" botón para obtener sensado e indicadores """
# botón iniciar sensado
btnTask = Button(root, text="Iniciar sensado", activeforeground =
'blue', width = 10, command = task)
btnTask.place(x=855, y = 60)
# label de temperatura
lblTitleTemp = Label(root, text = 'Temperatura', fg = 'red', font =
('Arial', 15))
lblTitleTemp.place(x=855, y = 130) #150
lblTemp = Label(root, textvariable = temperature, fg = 'red', font
= ('Arial', 25))
lblTemp.place(x=855,y = 170)
lblTextTemp = Label(root, text = '°C', fg = 'red', font = ('Arial',
15))
lblTextTemp.place(x=945, y =180)
# label BPM
lblTitleBpm = Label(root, text = 'Ritmo Cardiaco', fg = 'dark
```

```
orange', font = ('Arial',15))
lblTitleBpm.place(x=855, y = 240)
lblBpm = Label(root, textvariable = bpm, fg = 'dark orange', font =
('Arial', 25))
lblBpm.place(x=855, y = 280)
lblTextBpm = Label(root, text = 'BPM', fg = 'dark orange', font =
('Arial', 15))
lblTextBpm.place(x = 940, y =290)
# label SPO2
lblTitleSpo2 = Label(root, text = 'Oxigenación', fg = 'dodger blue',
font = ('Arial', 15))
lblTitleSpo2.place(x = 885, y = 350)
lblSpo2 = Label(root, textvariable = spo2, fg = 'dodger blue', font
= ('Arial', 25))
lblSpo2.place(x = 855, y = 380)
lblTextSpo2 = Label(root, text = 'SPO2', fg = 'dodger blue', font =
('Arial', 15))
lblTextSpo2.place(x = 935, y = 400)
# label Presión arterial
lblTitlePress = Label(root, text = 'Presión Arterial', fg = 'green',
font = ('Arial', 15))
lblTitlePress.place(x = 855, y = 460)
lblPress = Label(root, textvariable = arteryPress, fg = 'green',
font = ('Arial', 25))
lblPress.place(x = 855, y = 490)
lblTextPress = Label(root, text = 'KPa', fg = 'green', font =
('Arial', 15))
lblTextPress.place(x = 940, y = 510)
""" Sección para control manual """
def moverRobot(giro):
arduinoComunicación.moveRobot(giro)
def moverDerecha(*args):
giro = "3"
195
```

```
moverRobot(giro)
def moverIzquierda(*args):
giro = "4"
moverRobot(giro)
def moverAdelante(*args):
giro = "2"
moverRobot(giro)
def moverAtrás(*args):
giro = "5"
moverRobot(giro)
def parar(*args):
giro = "6"
moverRobot(giro)
def moverAutomático(*args):
giro = "1"
moverRobot(giro)
def moverContinuar(*args):
giro = "7"
moverRobot(giro)
# indicador - label
lblManual = Label(root, text = "Control Manual", fg = 'Black', font
= ('Arial', 16))
lblManual.place(x = 130, y =550)
btnGiroIzquierda = Button(root, text = 'Izquierda', activebackground
= 'steelBlue4', activeforeground = 'white', width = 10, command =
moverIzquierda)
btnGiroIzquierda.place(x = 5, y = 640)
# botón adelante
btnGiroAdelante = Button(root, text = 'Adelante', activebackground
= 'steelBlue4', activeforeground = 'white', width = 10, command =
moverAdelante)
btnGiroAdelante.place(x = 140, y = 600)
# botón derecha
btnGiroDerecha = Button(root, text = 'Derecha', activebackground =
```

```
'steelBlue4', activeforeground = 'white', width = 10, command =
moverDerecha)
btnGiroDerecha.place(x = 275, y = 640)
# botón atrás
btnGiroAtrás = Button(root, text = 'Atrás', activebackground =
'steelBlue4', activeforeground = 'white', width = 10, command =
moverAtrás)
btnGiroAtrás.place(x = 140, y = 680)
# botón parar
btnParar = Button(root, text = 'Parar', activebackground = 'red2',
activeforeground = 'white', width = 10, command = parar)
btnParar.place(x = 140, y = 640)
""" Sección para el control automático """
lblAutomático = Label(root, text = "Control Automático", fg =
"Black", font = ('Arial', 16))
lblAutomático.place(x = 440, y = 550)
# botón de activación de modo automático
btnAutomático = Button(root, text = "Automático", activebackground
= 'steelBlue4', activeforeground = 'white', width = 10, command =
moverAutomático)
btnAutomático.place(x = 480, y = 640)
root.mainloop()
```

Anexo 2: Código para la detección del reconocimiento facial

```
# mover al centro
arduino.flushInput()
arduino.write(b'2')
serialData = arduino.readline().decode('utf-
8').rstrip()
print('serialData: ', serialData)
elif direccion == 'R':
# mover a la derecha
arduino.flushInput()
arduino.write(b'3')
serialData = arduino.readline().decode('utf-
8').rstrip()
print('serialData: ', serialData)
elif direccion == 'L':
# mover a la izquieda
arduino.flushInput()
arduino.write(b'4')
serialData = arduino.readline().decode('utf-
8').rstrip()
print('serialData: ', serialData)
# algoritmo evasor de obstáculos
if serialData == "Obstaculo":
# si detectó obstaculo, identificar si es una persona por
detección de rostro
if posx == 0:
# no se detectó rostro, evadir obstáculo
arduino.write(b'9') # mover atras
time.sleep(1)
arduino.write("A".encode()) # mover
izquierda 90°
time.sleep(1)
elif posx > 0:
```

```
# actualizar interfaz
# borrarDatos()
# si se detectó rostro, atender al
paciente
routinePatient() # aplicar rutina de
atención al paciente
# una vez terminado la rutina, evadir
persona
arduino.write(b'9') # mover atras
time.sleep(1)
arduino.write("A".encode()) # mover
izquierda 45°
time.sleep(20) # 20 segundos de esperas
continuar() # continuar busqueda de
pacientes
# si no se detectó un rosto, ni objeto de color
de interés,
# girar a la izquierda para buscar un rostro o un
objeto de interés
if serialData == 'No':
arduino.flushInput()
arduino.write(b'4') # giro a la izquierda
lblVideo.after(10, visualizar) # regresar a la
función para refrescar video
else:
# si no se activó el modo autónomo, regresar a la
función para refrescar video
lblVideo.after(10, visualizar)
# si se desactiva la cámara, no mostrar video y terminar
conexión con cámaras
else:
lblVideo.image = ""
cameraFace.release()
```

```
cameraTrack.release()
def routinePatient():
global buscar
global inconciente
inconciente = False
state = 1
while state < 2:
# toma de temperatura
temp()
elif inconciente == False:
```

Anexo 3: Código para la detección de temperatura del cliente

```
import RPi.GPIO as GPIO
import time
import max30100
import numpy as np
import board
import busio as io
import adafruit_mlx90614
from time import sleep
import spidev
import os
def mover_promedio(numbers):
window_size = 4
i = 0
while i < len(numbers) - window_size + 1:
this_window = numbers[i : i + window_size]
window_average = sum(this_window) / window_size
i += 1
try:
return int((window_average/100))
except:
pass
def display_filter(moving_average_bpm,moving_average_sp02):
""" Función para obtener temperatura del módulo AMG-8833 """
#obtener datos de temperatura del módulo AMG-8833
import time,sys
sys.path.append('../')
import amg8833_i2c
import numpy as np
def getTemp():
t0 = time.time()
sensor = []
while (time.time()-t0)<1: # esperar 1 segundo
```

```
try:
# AD0 = GND, addr = 0x68 | AD0 = 5V, addr = 0x69
sensor = amg8833_i2c.AMG8833(addr=0x69) # inicializar
AMG8833
except:
sensor = amg8833_i2c.AMG8833(addr=0x68)
finally:
pass
time.sleep(0.2) # esperar establecer sensor
temp = 0 # variable para almacenar valor de temperatura
# Si no se encuentra sensor, regresar None
if sensor==[]:
print("No AMG8833 Found - Check Your Wiring")
temp = None
# Si se encuentra sensor, hacer toma de temperatura
else:
pix_to_read = 64 # read all 64 pixels
temperatura = []
for i in range(15):
status,pixels = sensor.read_temp(pix_to_read) # read
pixels with status
if status: # if error in pixel, re-enter loop and try
again
continue
T_thermistor = sensor.read_thermistor() # read
thermistor temp
temperatura.append(max(pixels)) #print(max(pixels))
time.sleep(1)
print(temperatura)
temp = np.mean(temperatura)
return temp
#if __name__ == "__main__":
#
```

```
# tempera = getTemp()
# Leer datos del sensor
sensorValue = ReadChannel(mpxChannel) # leer canal 0 del
módulo MCP, para obtener lecturas del sensor de presión de aire
pressure = (sensorValue - sensorOffset) / 100.0 # convertir
valor analógico obtenido del sensor, a dato de presión en Kpa
dataPress.append(pressure) # almacenar
# Print out results
print("Value: {} ({} kPa)".format(sensorValue,pressure))
# Wait before repeating loop
time.sleep(delay)
# apagar/desactivar motor y electroválvula
GPIO.output(pinMotor, False)
GPIO.output(pinValve, False)
time.sleep(2)
# limpiar puertos de la Rpi
GPIO.cleanup()
return np.mean(dataPress)
```

Anexo 4: Programa para el sensor infrarrojo seguidor de línea

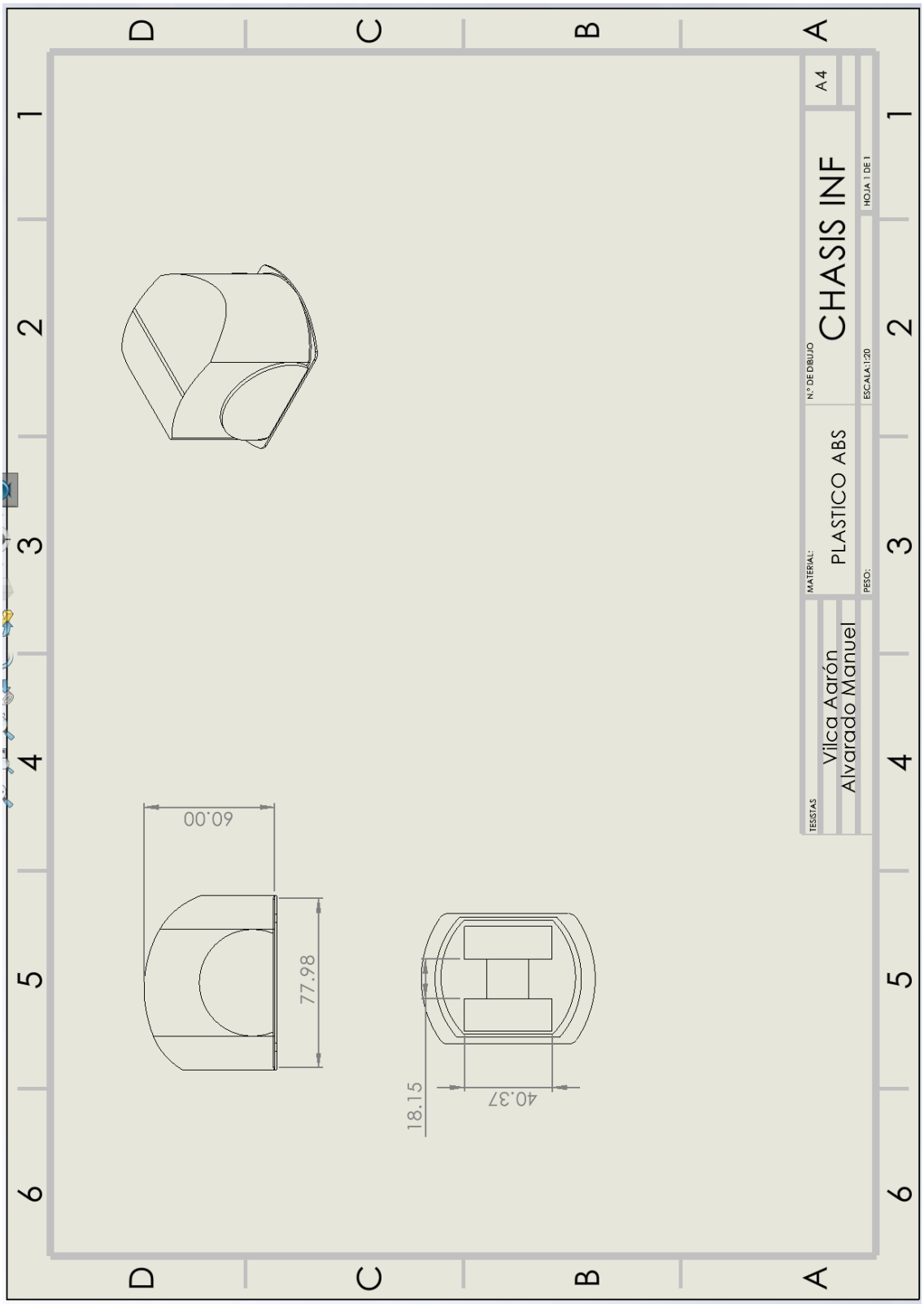
```
#include <Arduino.h> }
#include "Sensor_TCRT5000.h"
void Sensor::Inicializar()
{
  pinMode(Pin_sensor, INPUT);
}
int Sensor::Leer_sensor()
{
return digitalRead(Pin_sensor);
          }
// Definición de variables y constantes relacionadas con los sensores IR
int lecturaSensorIzq; // Almacena el valor de la lectura del sensor izquierdo
int lecturaSensorDer; // Almacena el valor de la lectura del sensor derecho
const int sensorIzqPin = A0; // El sensor izq irá conectado al pin analógico A0
const int sensorDerPin = A1; // El sensor derecho irá conectado al pin analógico A1
const int vel = 175;
void setup()
{
  // Se declaran todos los pines como salidas
  // Pines asociados a los motores
  pinMode (ENB, OUTPUT);
  pinMode( sensorIzqPin  , INPUT) ;
  pinMode( sensorDerPin  , INPUT) ;
  Serial.begin(9600); // Se inicia el puerto de comunicaciones en serie
}

void loop()
{
lecturaSensorIR(); // Se lee el valor de los sensores IR
// Se analiza el resultado de los sensores para hacer que el robot siga la línea negra
// Si el resultado de ambos sensores es 0 (zona blanca) el robot sigue se para
if(lecturaSensorIzq == 0 &amp;amp;&amp;amp; lecturaSensorDer == 0)
```

```
{
robotParar(); // El robot para
}
// Si ambos sensores retornan 0 (zona negra) el robot sigue recto
if (lecturaSensorIzq == 1 &amp;amp;&amp;amp; lecturaSensorDer == 1)
{
robotAvance(); // El robot avanza
Serial.println("robot avanza");
}
}
/*
Función lecturaSensorIR: leerá el valor del sensor de infrarrojos TCRT5000
y lo almacena en una variable. Dicho sensor retornará el valor 0 (LOW) si
el sensor está en zona blanca y el valor 1 (HIGH) si el sensor está en zona negra.
*/
void lecturaSensorIR()
{
lecturaSensorIzq = digitalRead(sensorIzqPin); // Almacena la lectura del sensor
izquierdo
lecturaSensorDer = digitalRead(sensorDerPin); // Almacena la lectura del sensor
derecho
Serial.println("El valor del sensor izquierdo es ");
Serial.println(lecturaSensorIzq);
Serial.println("El valor del sensor derecho es ");
Serial.println(lecturaSensorDer);
}
/*
Función robotAvance: esta función hará que ambos motores se activen a máxima
potencia por lo que el robot avanzará hacia delante
*/
void robotAvance()
{
// Motor izquierdo
// Al mantener un pin HIGH y el otro LOW el motor gira en un sentido
```

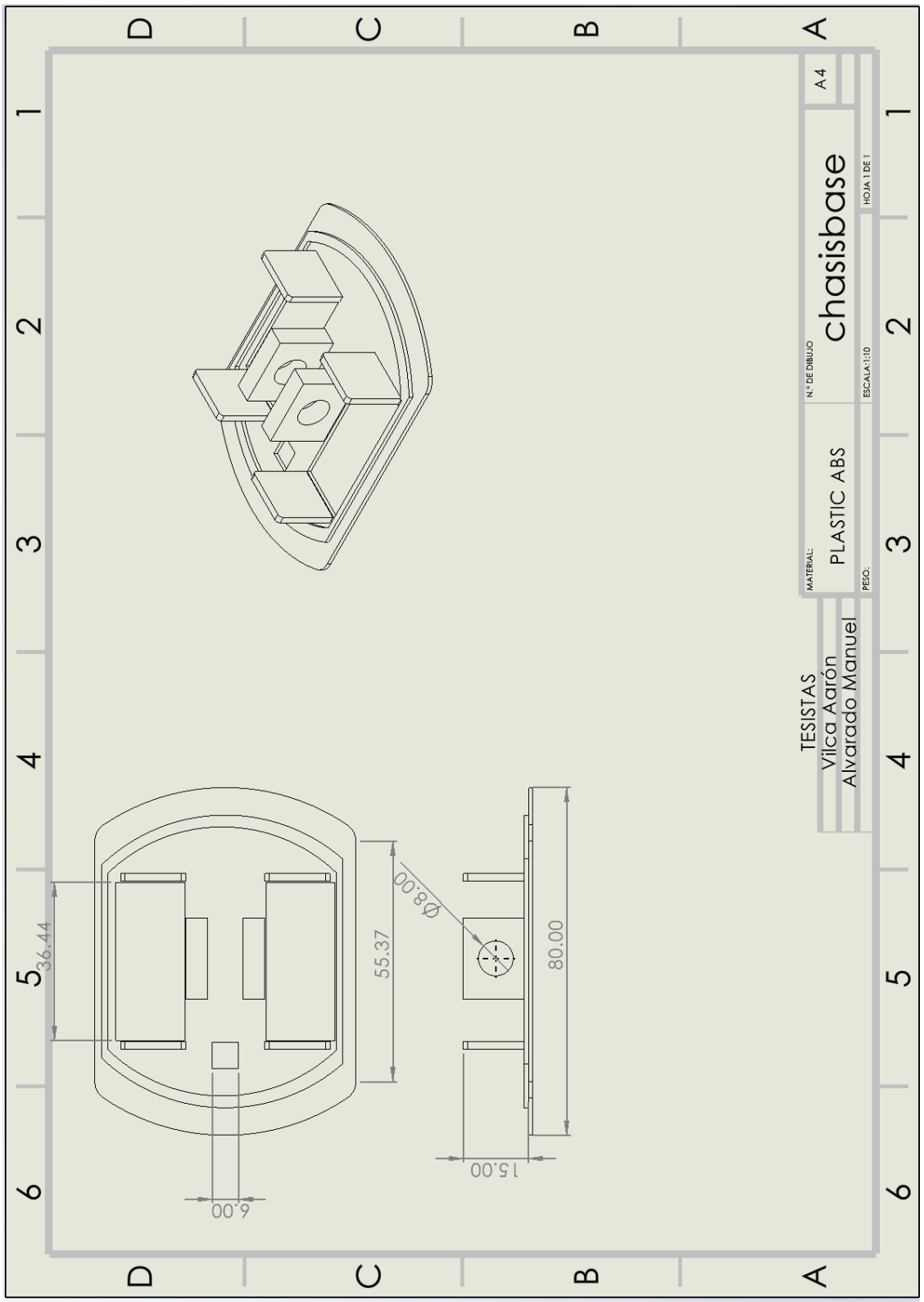
```
digitalWrite (IN1, HIGH);
digitalWrite (IN2, LOW);
analogWrite (ENA, vel); //Velocidad motor A
// Motor derecho
// Al mantener un pin HIGH y el otro LOW el motor gira en un sentido
digitalWrite (IN3, HIGH);
digitalWrite (IN4, LOW);
analogWrite (ENB, vel); //Velocidad motor B
}
/*
Función robotRetroceso: esta función hará que ambos motores se activen a máxima
potencia en sentido contrario al anterior por lo que el robot avanzará hacia atrás
*/
void robotRetroceso()
{
// Motor izquierdo
// Al mantener un pin LOW y el otro HIGH el motor gira en sentido contrario al anterior
digitalWrite (IN1, LOW);
digitalWrite (IN2, HIGH);
analogWrite (ENA, vel); //Velocidad motor A
 // Motor derecho
 // Al mantener un pin LOW y el otro HIGH el motor gira en sentido contrario al
anterior
digitalWrite (IN3, LOW);
 digitalWrite (IN4, HIGH);
analogWrite (ENB, vel); //Velocidad motor B
}
*/ Función robotParar: esta función parará ambos motores por lo que el robot se parará.
void robotParar()
{
// Motor izquierdo
// Se para el motor izquierdo
digitalWrite (IN1, LOW);
digitalWrite (IN2, LOW);
```

```
// Motor derecho
// Se para el motor derecho
digitalWrite (IN3, LOW);
digitalWrite (IN4, LOW);
```

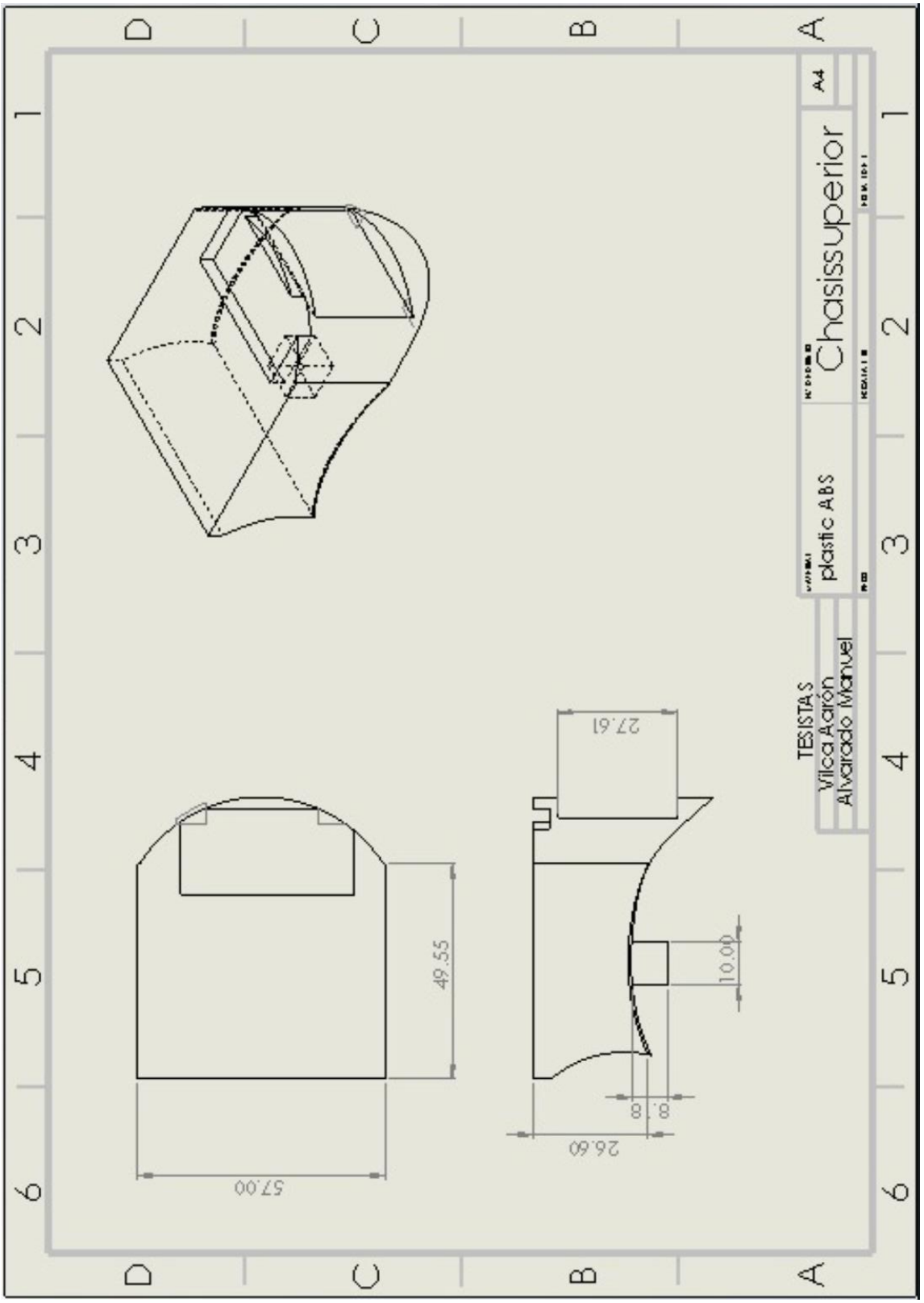
Anexo 5: Plano Chasis inferior



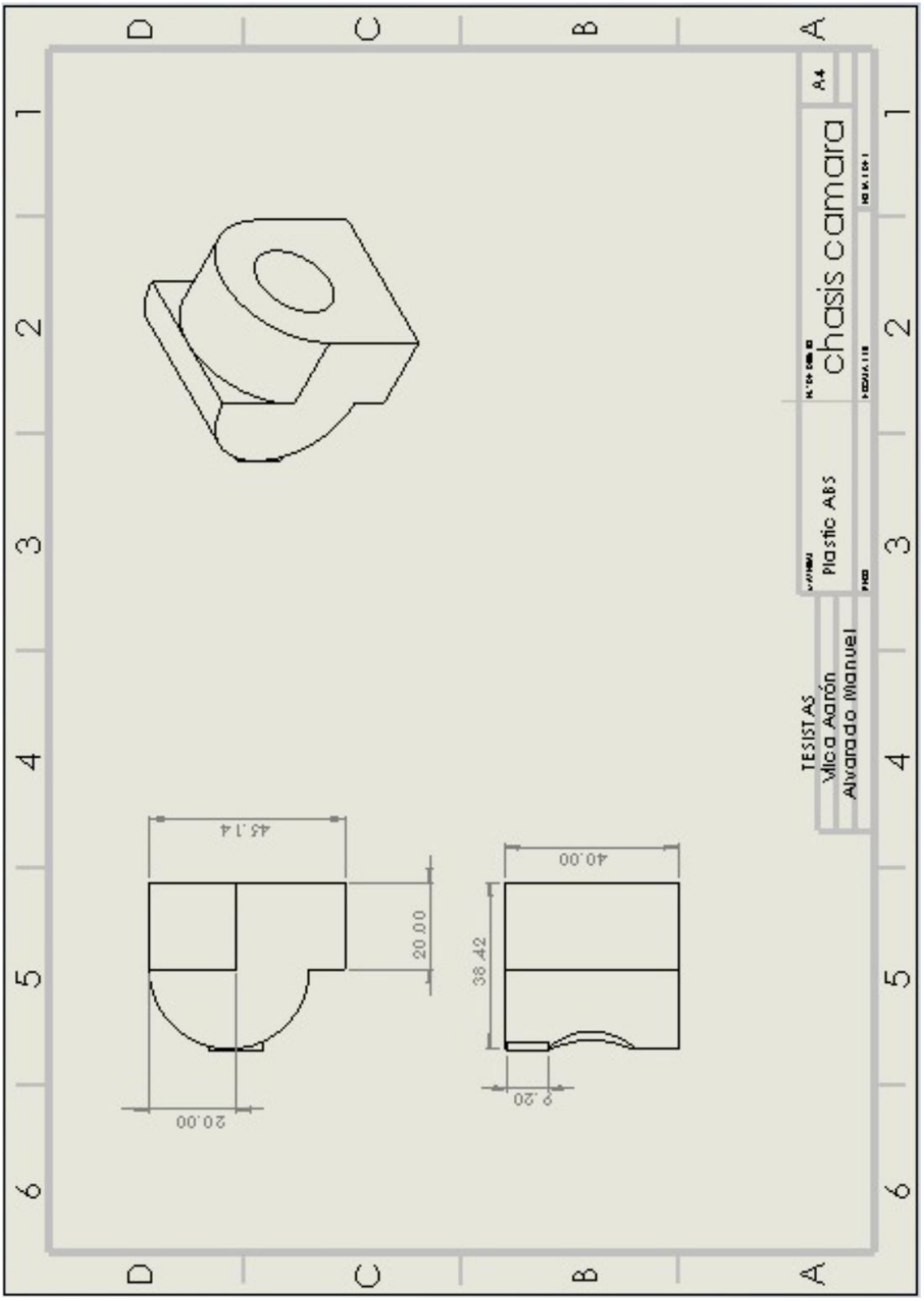
Anexo 6: Plano Chasis Base



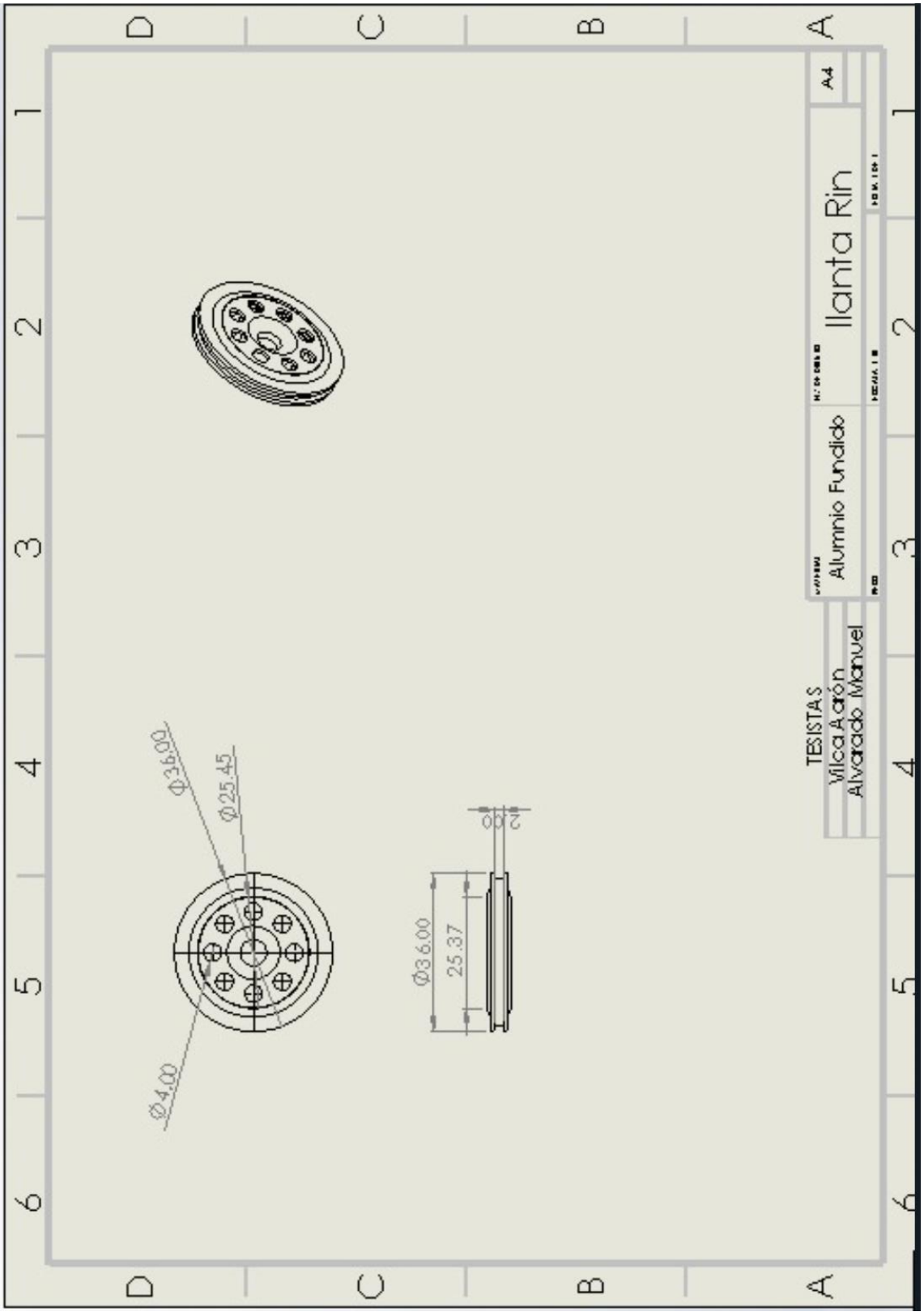
Anexo 7: Plano Chasis Superior



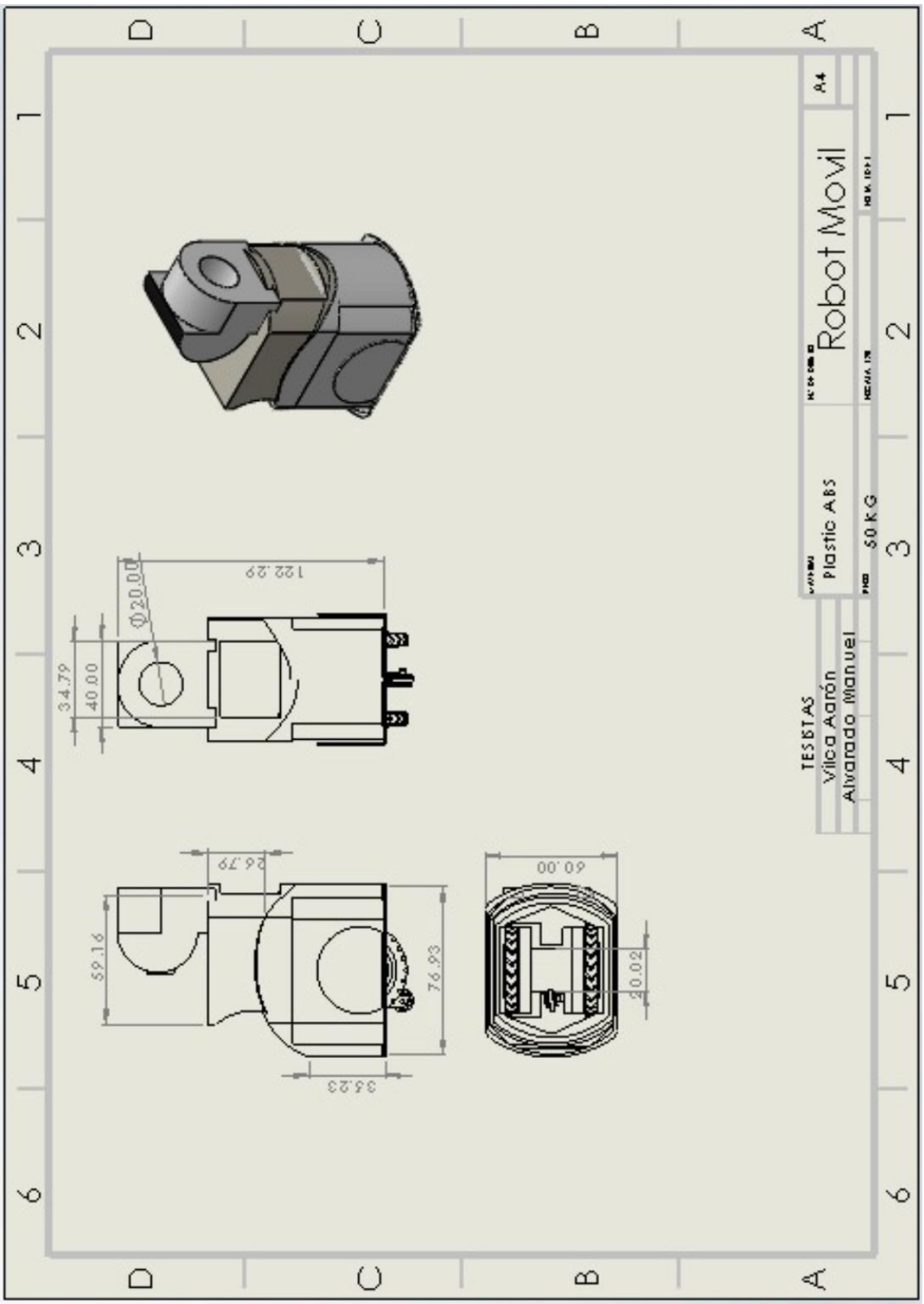
Anexo 8: Plano Chasis Cámara



Anexo 9: Plano Llanta Rin



Anexo 10: Plano Robot Móvil

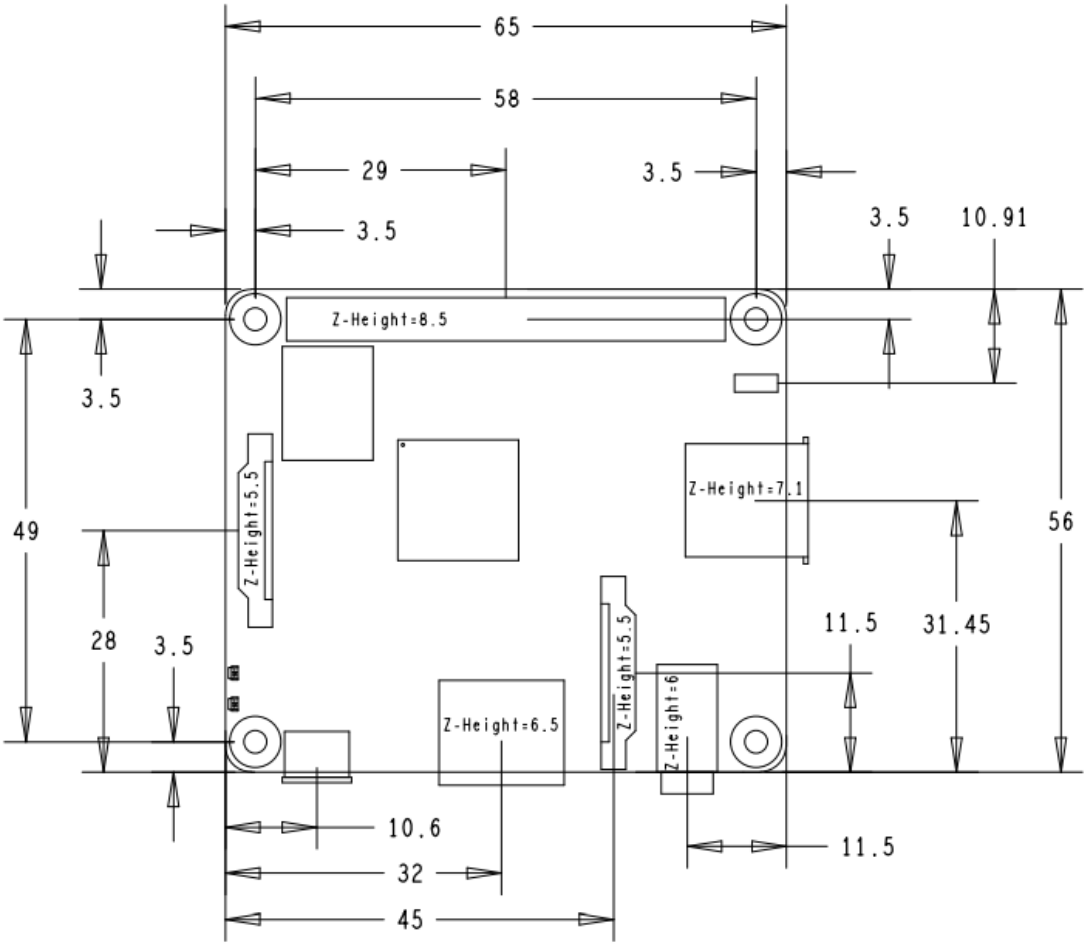


Anexo 11: Datasheet Raspberry Pi 3A

Specification

Processor:	Broadcom BCM2837B0, Cortex-A53 64-bit SoC @ 1.4 GHz
Memory:	512MB LPDDR2 SDRAM
Connectivity:	2.4 GHz and 5 GHz IEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2/BLE
Access:	Extended 40-pin GPIO header
Video & sound:	1 × full size HDMI MIPI DSI display port MIPI CSI camera port 4 pole stereo output and composite video port
Multimedia:	H.264, MPEG-4 decode (1080p30); H.264 encode (1080p30); OpenGL ES 1.1, 2.0 graphics
SD card support:	Micro SD format for loading operating system and data storage
Input power:	5 V/2.5 A DC via micro USB connector 5 V DC via GPIO header
Environment:	Operating temperature, 0–50°C
Compliance:	For a full list of local and regional product approvals, please visit: www.raspberrypi.org/products/raspberry-pi-3-model-a-plus
Production lifetime:	The Raspberry Pi 3 Model A+ will remain in production until at least January 2023

Anexo 12: Dimensiones en milímetros de Raspberry Pi 3A



Anexo 13: Ficha Técnica Board Mount Temperature Sensors Grid-EYE Hi-Perf Infrared Array sensor – AMG8833 Panasonic

Panasonic **Infrared Array Sensor Grid-EYE (AMG88)**

Infrared Array Sensor Grid-EYE

High Precision Infrared Array Sensor based on Advanced MEMS Technology

Features

- Temperature detection of two-dimensional area: 8 × 8 (64 pixels)
- Digital output (capability of temperature value output)
- Compact SMD package (adaptively to reflow mounting)
- RoHS compliant

Typical Applications

- High function home appliances (microwaves and air-conditioners)
- Energy saving at office (air-conditioning/lighting control)
- Digital signage
- Automatic doors/elevators

Ordering Information

AMG 8 8 _ _

- 1st 8: Vertical pixel 8 : 8 pixels
- 2nd 8: Horizontal pixel 8 : 8 pixels
- Applied voltage 3 : 3.3 V.DC 5 : 5 V.DC
- Applied voltage **3**: High gain **4**: Low gain

Types

Product name	Number of pixel	Operating voltage	Amplification factor	Part number
Infrared array sensor Grid-EYE	64 (Vertical 8 × Horizontal 8 Matrix)	3.3 V.DC	High gain	AMG8833
			Low gain	AMG8834
		5.0 V.DC	High gain	AMG8853
			Low gain	AMG8854

Tape and reel package : 1,000 pcs.

Rating

Item	Performance	
	High gain	Low gain
Applied voltage	3.3 V.DC±0.3 V.DC or 5.0 V.DC±0.5 V.DC	
Temperature range of measuring object	0 °C to 80 °C +32 °F to +176 °F	–20 °C to 100 °C –4 °F to +212 °F
Operating temperature range	0 °C to 80 °C +32 °F to +176 °F	–20 °C to 80 °C –4 °F to +176 °F
Storage temperature range	–20 °C to 80 °C –4 °F to +176 °F	–20 °C to 80 °C –4 °F to +176 °F

Absolute Maximum Ratings

Item	Absolute maximum ratings	Terminal
Applied voltage	−0.3 V.DC to 6.5 V.DC	VDD
Input voltage	−0.3 V.DC to VDD +0.3 V.DC	SCL, SDA, AD_SELECT
Output sink current	−10 mA to 10 mA	INT, SDA
Static electricity (Human body model)	1 kV	All terminals
Static electricity (Machine model)	200 V	All terminals

Characteristics

Item	Performance	
	High gain	Low gain
Temperature accuracy	Typical ±2.5 °C ±4.5 °F	Typical ±3.0 °C ±5.4 °F
Human detection distance *1	Max. 7 m 22.966 ft	
Viewing angle	Typical 60 °	
Optical axis gap	Within Typical ±5.6 °	
Current consumption	Typical 4.5 mA (normal mode) Typical 0.2 mA (sleep mode) Typical 0.8 mA (stand-by mode)	
Setup time	Typical 50 ms (Time to enable communication after setup) Typical 15 s (Time to stabilize output after setup)	

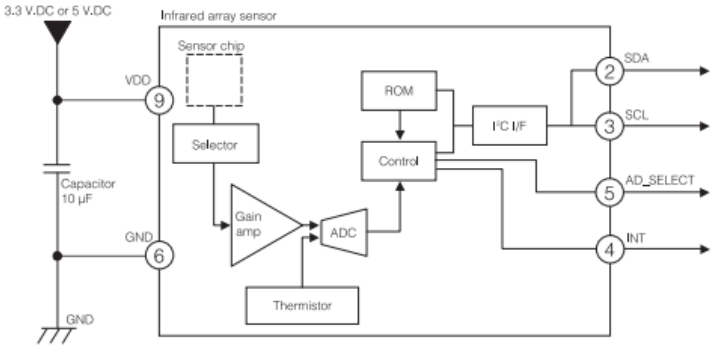
Note: ✻1 To have more than 4 °C 7.2 °F of temperature difference from background
Detection object size: 700 × 250 mm 27.559 × 9.843 inch (Assumable human body size)

Performance

Item	Performance
Number of pixel	64 (Vertical 8 × Horizontal 8 Matrix)
External interface	I^2C (fast mode)
Frame rate	Typical 10 frames/sec or 1 frame/sec
Operating mode *1	Normal Sleep Stand-by (10 sec or 60 sec intermittence)
Output mode	Temperature output
Calculate mode	No moving average or Twice moving average
Temperature output resolution	0.25 °C
Number of sensor address	2 (I^2C slave address)
Thermistor output temperature range	−20 °C to 80 °C −4 °F to +176 °F
Thermistor output resolution	0.0625 °C

Note: ✻1 Normal Mode : normal operation mode; Sleep Mode: detection is off (output and data reading not possible); Standby Mode: 1 frame measuring intermittently every 10 or 60 sec.

Internal Circuit



✻ INT terminal ④ normally has same voltage as VDD. When interrupting, same as GND (0V)

Anexo 14: Ficha Técnica Cámara Web HD C505

CARACTERÍSTICAS CLAVE Y ESPECIFICACIONES DEL PRODUCTO

1. **Video HD 720 en pantalla panorámica:** Un campo visual diagonal de 60° junto con la resolución HD 720p/30 fps, foco fijo y corrección de iluminación automática ayudan a ajustarla a la mayoría de condiciones de iluminación.
2. **Micrófono mono de largo alcance:** El único micrófono omnidireccional se concibió para ofrecer conversaciones claras y naturales hasta a 3 metros de distancia, incluso en ambientes de oficina ruidosos.
3. **El cable USB-A extralargo amplía las opciones de configuración:** Con su cable USB y su clip universal, colócala de forma segura sobre la pantalla de una laptop o sobre una pantalla externa, o instálala hasta a 2 m (7 ft) de distancia de la computadora.

Video	Compatible con resoluciones 720p (HD) a 30 fps para una mejor compatibilidad con la calidad ofrecida por tu aplicación o monitor.	
	Campo visual diagonal (dFOV) fijo de 60°	
	Corrección de iluminación automática RightLight™ 2 para una imagen clara en diversos entornos de iluminación desde luz escasa hasta luz solar directa.	
Audio	Reducción de ruido, un único micrófono omnidireccional con amplio radio de captación hasta a tres metros de distancia.	
Conectividad	Fácil conexión por USB-A; longitud de cable de 2 m (7 ft)	
Opciones de montaje	Clip universal para monitores, pantallas LCD o laptops	
Compatibilidad	Funciona con computadoras Windows, Mac o Chrome por USB-A y con aplicaciones de llamadas habituales como Microsoft® Teams, Skype for Business, Google Meet y Voice™, Zoom™, Cisco Jabber™ y otras, para garantizar la compatibilidad y la perfecta integración en el entorno de trabajo.	
General	Número de referencia	960-001372
	Dimensiones y peso	**Sin clip:** Altura x Anchura x Profundidad: 31,91 mm (1,26 in) x 72,91 mm (2,87 in) x 24,19 mm (0,95 in) Longitud del cable: 2 m (7 ft) **Con clip:** Altura x Anchura x Profundidad: 31,91 mm (1,26 in) x 72,91 mm (2,87 in) x 66,64 mm (2,62 in) Peso: 75 g (2,65 oz)
	Contenido de la caja	Cámara Web con cable USB-A fijo de 2 m (7 ft) Documentación del usuario
	Garantía	3 años

www.logitech.com/c505e

logitech

Ponte en contacto con tu distribuidor o con nosotros en www.logitech.com/vcsales

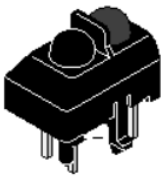
Logitech Inc. 7700 Gateway Blvd Newark, CA 94560, EE. UU. Publicación: Agosto de 2020

Anexo 15: Ficha Técnica TCRT5000

Reflective Optical Sensor with Transistor Output

Description

The TCRT5000(L) has a compact construction where the emitting-light source and the detector are arranged in the same direction to sense the presence of an object by using the reflective IR beam from the object. The operating wavelength is 950 mm. The detector consists of a phototransistor.

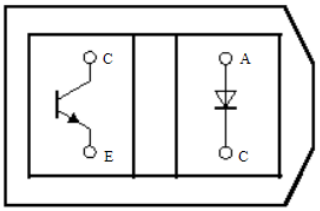


94 9442

Applications

- D Position sensor for shaft encoder
- D Detection of reflective material such as paper, IBM cards, magnetic tapes etc.
- D Limit switch for mechanical motions in VCR
- D General purpose – wherever the space is limited

15116



Top view

Features

- D Snap-in construction for PCB mounting
- D Package height: 7 mm
- D Plastic polycarbonate housing construction which prevents crosstalk
- D L = long leads
- D Current Transfer Ratio (CTR) of typical 10%

Order Instruction

Ordering Code	Sensing Distance	Remarks
TCRT5000	12 mm	Leads (3.5 mm)
TCRT5000(L)	12 mm	Long leads (15 mm)

Electrical Characteristics (T_{amb} = 25°C)

Input (Emitter)

Parameter	Test Conditions	Symbol	Min.	Typ.	Max.	Unit
Forward voltage	I_F = 60 mA	V_F		1.25	1.5	V
Junction capacitance	V_R = 0 V, f = 1 MHz	C_j		50		pF

Output (Detector)

Parameter	Test Conditions	Symbol	Min.	Typ.	Max.	Unit
Collector emitter voltage	I_C = 1 mA	V_{CEO}	70			V
Emitter collector voltage	I_E = 100 mA	V_{ECO}	7			V
Collector dark current	V_{CE} = 20 V, I_F = 0, E = 0	I_{CEO}		10	200	nA

Sensor

Parameter	Test Conditions	Symbol	Min.	Typ.	Max.	Unit
Collector current	V_{CE} = 5 V, I_F = 10 mA, D = 12 mm	I_{C} 1,2)	0.5	1	2.1	mA
Collector emitter saturation voltage	I_F = 10 mA, I_C = 0.1 mA, D = 12 mm	V_{CEsat} 1,2)			0.4	V
1) See test circuit						
2) Test surface: Mirror (Mfr. Spindler a. Hoyer, Part No 340005)						

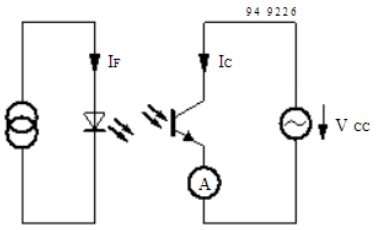


Figure 1. Test circuit

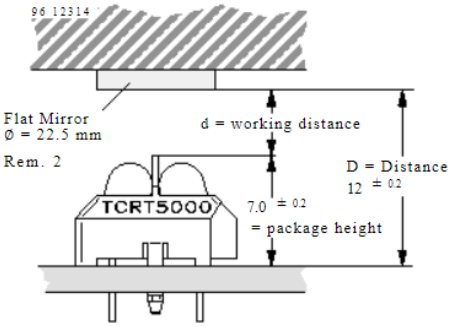


Figure 2. Test circuit

Typical Characteristics (Tamb = 25_C, unless otherwise specified)

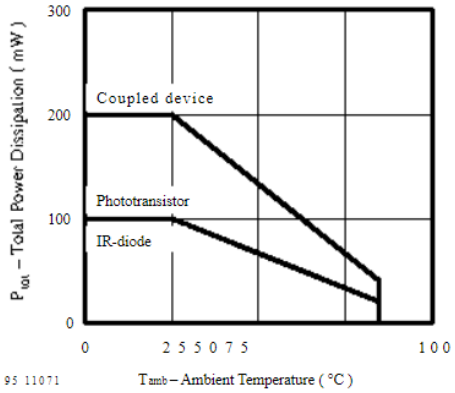


Figure 3. Total Power Dissipation vs. Ambient Temperature

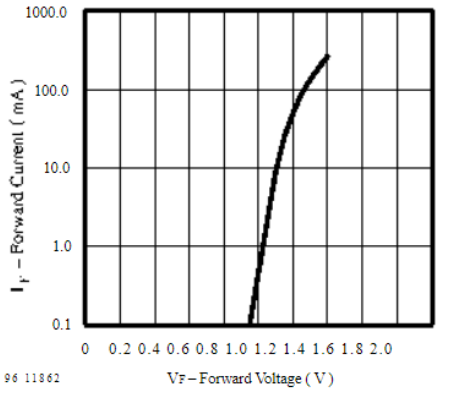


Figure 4. Forward Current vs. Forward Voltage

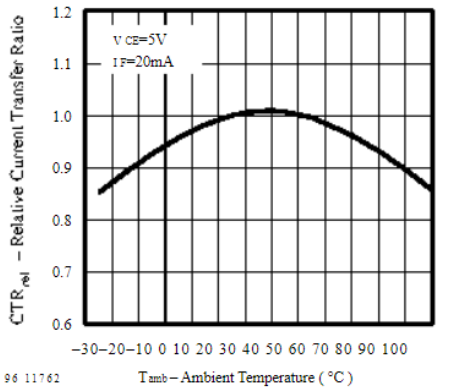


Figure 5. Rel. Current Transfer Ratio vs. Ambient Temp.

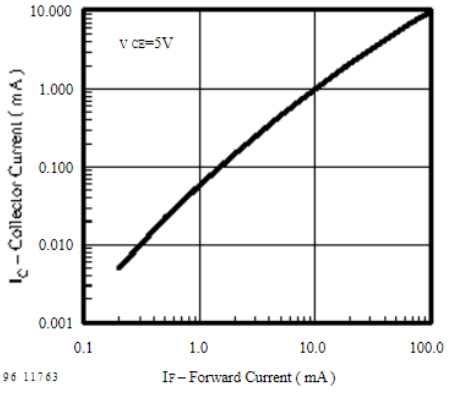


Figure 6. Collector Current vs. Forward Current

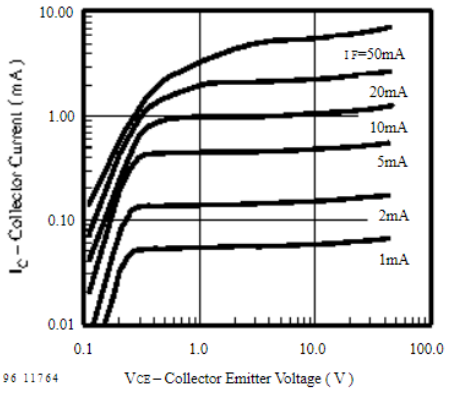


Figure 7. Collector Emitter Saturation Voltage vs. Collector Current

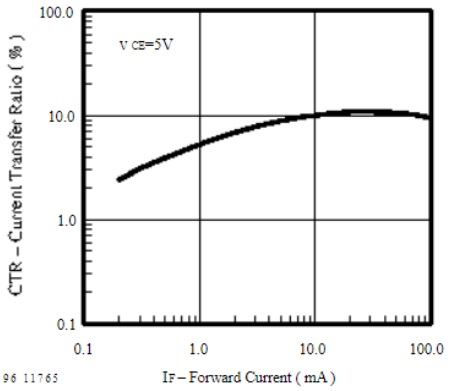
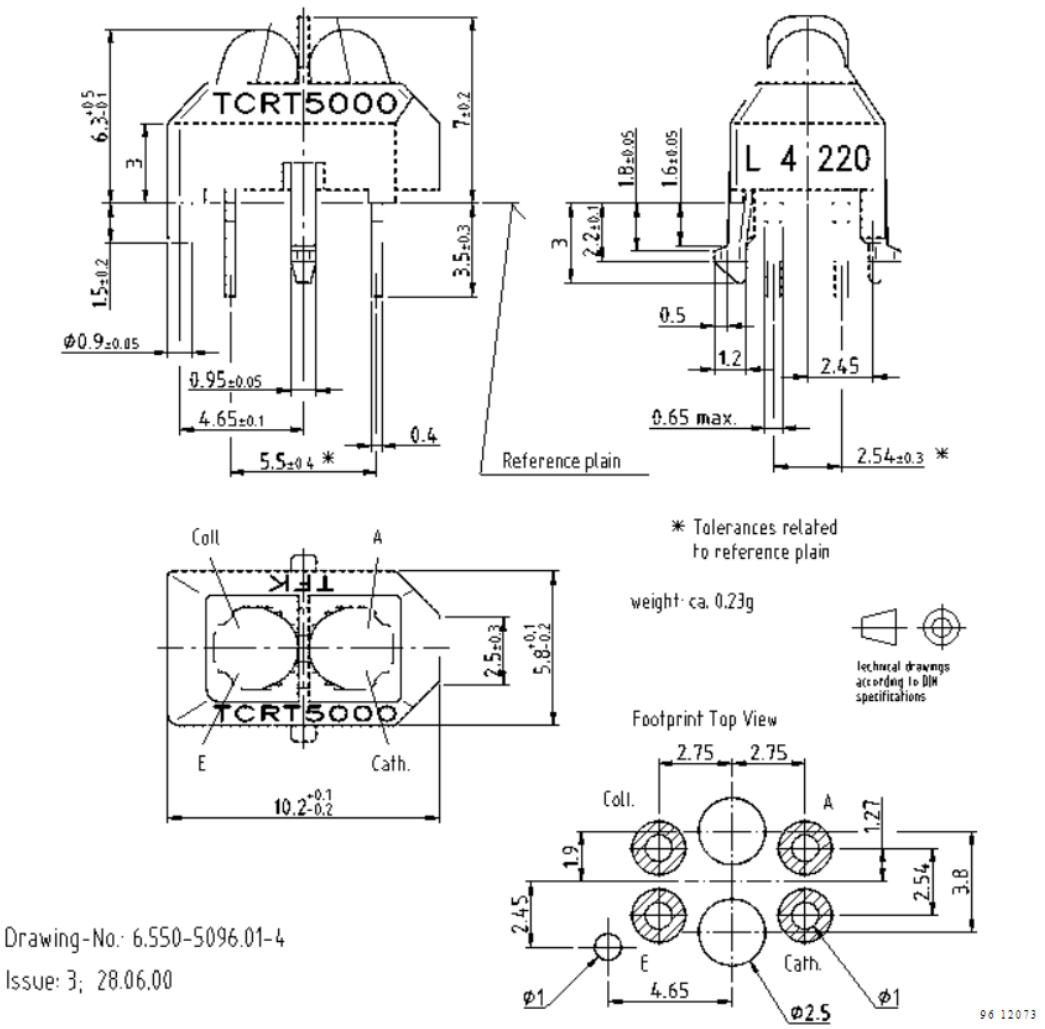


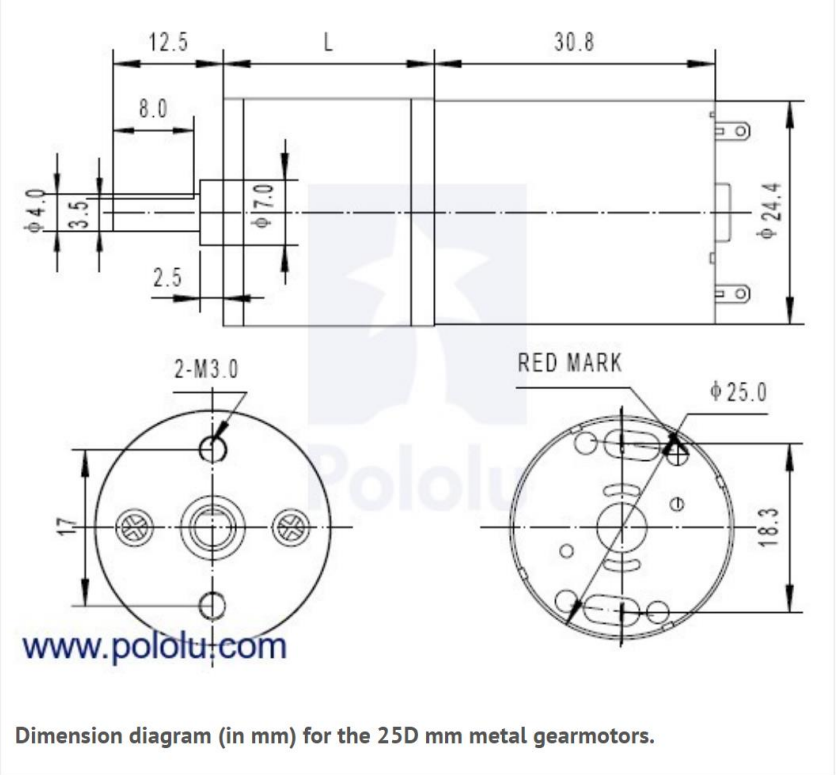
Figure 8. Current Transfer Ratio vs. Forward Current

Dimensions of TCRT5000 in mm



Anexo 16: Dimensiones Motorreductor 25d 6-12v DC

El siguiente diagrama muestra las dimensiones (en mm) de la línea de motorreductores 25D. El valor de L se muestra en la tabla siguiente.



Dimension diagram (in mm) for the 25D mm metal gearmotors.

Gear Ratio	L (mm)
4.4:1 9.7:1	17
20.4:1	19
34.1 47:1	21
75:1 99:1	23
172:1 227:1	25
378:1 499:1	27

Especificaciones a 12 V: 1030 RPM, Torque: 3.2 kg-cm, 5.6 A Corriente de Pico y 300 mA sin carga.

Anexo 17: Códigos Matlab para simulación modelo matemático cinemático

SCRIPT: MOBILE PLOT.M

```
function Mobile_Graph = MobilePlot(x,y,angz,scale,color)
global Uniciclo

Rz=[ cos(angz) -sin(angz) 0;...
   sin(angz) cos(angz) 0;...
   0 0 1];

robotPatch = Rz*Uniciclo.parte1Vertices;
robotPatch(1,:) = robotPatch(1,:)*scale+x;
robotPatch(2,:) = robotPatch(2,:)*scale+y;
robotPatch(3,:) = robotPatch(3,:)*scale;

Mobile_Graph(1) =
patch('Faces',Uniciclo.parte1Faces,'Vertices',robotPatch','FaceColor',color,'EdgeColor','
none');

robotPatch = Rz*Uniciclo.parte2Vertices;
robotPatch(1,:) = robotPatch(1,:)*scale+x;
robotPatch(2,:) = robotPatch(2,:)*scale+y;
robotPatch(3,:) = robotPatch(3,:)*scale;

Mobile_Graph(2) =
patch('Faces',Uniciclo.parte2Faces,'Vertices',robotPatch','FaceColor','k','EdgeColor','non
e');

robotPatch = Rz*Uniciclo.parte3Vertices;
robotPatch(1,:) = robotPatch(1,:)*scale+x;
robotPatch(2,:) = robotPatch(2,:)*scale+y;
robotPatch(3,:) = robotPatch(3,:)*scale;

Mobile_Graph(3) =
patch('Faces',Uniciclo.parte3Faces,'Vertices',robotPatch','FaceColor','y','EdgeColor','non
e');

robotPatch = Rz*Uniciclo.parte4Vertices;
robotPatch(1,:) = robotPatch(1,:)*scale+x;
robotPatch(2,:) = robotPatch(2,:)*scale+y;
robotPatch(3,:) = robotPatch(3,:)*scale;

Mobile_Graph(4) =
patch('Faces',Uniciclo.parte4Faces,'Vertices',robotPatch','FaceColor',
'y','EdgeColor','none');

robotPatch = Rz*Uniciclo.parte5Vertices;
robotPatch(1,:) = robotPatch(1,:)*scale+x;
```

```
robotPatch(2,:) = robotPatch(2,:)*scale+y;
robotPatch(3,:) = robotPatch(3,:)*scale;

Mobile_Graph(5) =
patch('Faces',Uniciclo.parte5Faces,'Vertices',robotPatch','FaceColor','w','EdgeColor','no
ne');

robotPatch = Rz*Uniciclo.parte6Vertices;
robotPatch(1,:) = robotPatch(1,:)*scale+x;
robotPatch(2,:) = robotPatch(2,:)*scale+y;
robotPatch(3,:) = robotPatch(3,:)*scale;

Mobile_Graph(6) =
patch('Faces',Uniciclo.parte6Faces,'Vertices',robotPatch','FaceColor',
'y','EdgeColor','none');

robotPatch = Rz*Uniciclo.parte7Vertices;
robotPatch(1,:) = robotPatch(1,:)*scale+x;
robotPatch(2,:) = robotPatch(2,:)*scale+y;
robotPatch(3,:) = robotPatch(3,:)*scale;

Mobile_Graph(7) =
patch('Faces',Uniciclo.parte7Faces,'Vertices',robotPatch','FaceColor','k','EdgeColor','non
e');

robotPatch = Rz*Uniciclo.parte8Vertices;
robotPatch(1,:) = robotPatch(1,:)*scale+x;
robotPatch(2,:) = robotPatch(2,:)*scale+y;
robotPatch(3,:) = robotPatch(3,:)*scale;

Mobile_Graph(8) =
patch('Faces',Uniciclo.parte8Faces,'Vertices',robotPatch','FaceColor','y','EdgeColor','non
e');

robotPatch = Rz*Uniciclo.parte9Vertices;
robotPatch(1,:) = robotPatch(1,:)*scale+x;
robotPatch(2,:) = robotPatch(2,:)*scale+y;
robotPatch(3,:) = robotPatch(3,:)*scale;

Mobile_Graph(9) =
patch('Faces',Uniciclo.parte9Faces,'Vertices',robotPatch','FaceColor',
'y','EdgeColor','none');

robotPatch = Rz*Uniciclo.parte10Vertices;
robotPatch(1,:) = robotPatch(1,:)*scale+x;
robotPatch(2,:) = robotPatch(2,:)*scale+y;
robotPatch(3,:) = robotPatch(3,:)*scale;

Mobile_Graph(10) =
patch('Faces',Uniciclo.parte10Faces,'Vertices',robotPatch','FaceColor','w','EdgeColor','n
```

```
one');
robotPatch = Rz*Uniciclo.parte11Vertices;
robotPatch(1,:) = robotPatch(1,:)*scale+x;
robotPatch(2,:) = robotPatch(2,:)*scale+y;
robotPatch(3,:) = robotPatch(3,:)*scale;

Mobile_Graph(11) =
patch('Faces',Uniciclo.parte11Faces,'Vertices',robotPatch,'FaceColor','w','EdgeColor','n
one');

end
```

SCRIPT: MAIN.M

```
clear
clc
close all

hx = 2;
hy = 3;
phi = 90*(pi/180);

%%%%%%%%%%%%%% Escena de simulación %%%%%%%%%%%%%%
scene = figure;
tam = get(0,'ScreenSize');
set(scene,'position',tam);
set(scene,'Color','white');
set(gca,'FontWeight','bold');
axis equal;
axis([-5 5 -5 5 0 3]);
view([-125 30])
xlabel('x [m]'); ylabel('y [m]'); zlabel('z [m]')
camlight right
grid on;
box on;

scale = 10;
MobileRobot;
R1 = MobilePlot(hx,hy,phi,scale,'b');
```

SCRIPT: MOBILE ROBOT.M

```
function MobileRobot

parte1 = stlRead('1_Estructura.stl');
parte2 = stlRead('2_ruedas.stl');
parte3 = stlRead('3_aros.stl');
parte4 = stlRead('4_motores.stl');
parte5 = stlRead('5_RuedaLoca.stl');

global Uniciclo
```

```
Uniciclo.parte1Vertices=parte1.vertices';
Uniciclo.parte1Faces = parte1.faces;

Uniciclo.parte2Vertices=parte2.vertices';
Uniciclo.parte2Faces = parte2.faces;

Uniciclo.parte3Vertices=parte3.vertices';
Uniciclo.parte3Faces = parte3.faces;

Uniciclo.parte4Vertices=parte4.vertices';
Uniciclo.parte4Faces = parte4.faces;

Uniciclo.parte5Vertices=parte5.vertices';
Uniciclo.parte5Faces = parte5.faces;

end
```

SCRIPT: MODELO CINEMATICO 1

```
clear
clc
close all

%%%%%%%%%%%% Variables de tiempo %%%%%%%%%%%%%%
tf = 60;
ts = 0.1;
t = 0:ts:tf;

N = length(t);

%%%%%%%%%%%%%%%%%% Condiciones iniciales del robot
%%%%%%%%%%
hx(1) = 0;
hy(1) = 0;
phi(1) = 0;

%%%%%%%%%%%%%%%%%%% Acciones de control
%%%%%%%%%%%%%%%
u = 0.2*sin(0.1*t) ; % velocidad lineal m/s
w = 0.08*cos(0.3*t); % velocidad angular rad/s

%%%%%%%%%%%%%%%%%%%% Bucle de simulacion
%%%%%%%%%%%%%
for k=1:N

    phi(k+1) = w(k)*ts+phi(k);

    hxp = u(k)*cos(phi(k+1));
    hyp = u(k)*sin(phi(k+1));
```

```
    hx(k+1) = hxp*ts+hx(k);
    hy(k+1) = hyp*ts+hy(k);

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%% Armar la escena de
simulacion %%%%%%%%%%%%%%
scene = figure;
tam = get(0,'ScreenSize');
set(scene,'position',tam);
set(scene,'Color','white');
set(gca,'FontWeight','bold');
axis equal;
axis([-5 5 -5 10 0 3]);
view([-125 30])
xlabel('x [m]'); ylabel('y [m]'); zlabel('z [m]')
camlight right
grid on;
box on;

scale = 10;
MobileRobot;
R1 = MobilePlot(hx(1),hy(1),phi(1),scale,'b');
hold on;
T1 = plot3(hx(1),hy(1),0,'b','LineWidth',3);

step = 15;

for k=1:step:N
    delete(R1);
    delete(T1);
    R1 = MobilePlot(hx(k),hy(k),phi(k),scale,'b');
    T1 = plot3(hx(1:k),hy(1:k),zeros(k),'b','LineWidth',3);

    pause(ts);
end
```

SCRIPT: MODELO CINEMATICO 2

```
clear
clc
close all

%%%%%%%%%%%% Variables de tiempo %%%%%%%%%%%%%%
tf = 60;
ts = 0.1;
t = 0:ts:tf;

N = length(t);
```

```
%%%%%%%%%%%%%%%%%%%%%% Parametros del robot
%%%%%%%%%%%%%%%%%%
a = 0.3; % [m]

%%%%%%%%%%%%%%%%%% Condiciones iniciales del robot
%%%%%%%%%%
x1(1) = 0;
y1(1) = 0;
phi(1) = 0;

%%%% Cinematica directa
hx(1) = x1(1)+a*cos(phi(1));
hy(1) = y1(1)+a*sin(phi(1));

%%%%%%%%%%%%%%%%%% Acciones de control
%%%%%%%%%%%%%%%
u = 0.2*sin(0.1*t) ; % velocidad lineal m/s
w = 0.08*cos(0.3*t); % velocidad angular rad/s

%%%%%%%%%%%%%%%%%% Bucle de simulacion
%%%%%%%%%%%%
for k=1:N

    phi(k+1) = w(k)*ts+phi(k);

    x1p = u(k)*cos(phi(k+1));
    y1p = u(k)*sin(phi(k+1));

    x1(k+1) = x1p*ts+x1(k);
    y1(k+1) = y1p*ts+y1(k);

    %%%% Cinematica directa
    hx(k+1) = x1(k+1)+a*cos(phi(k+1));
    hy(k+1) = y1(k+1)+a*sin(phi(k+1));

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Armar la escena de
simulacion %%%%%%%%%%%%%%%
scene = figure;
tam = get(0,'ScreenSize');
set(scene,'position',tam);
set(scene,'Color','white');
set(gca,'FontWeight','bold');
axis equal;
axis([-5 5 -5 10 0 3]);
view([-125 30])
xlabel('x [m]'); ylabel('y [m]'); zlabel('z [m]')
camlight right
grid on;
```

```
box on;

scale = 10;
MobileRobot;
R1 = MobilePlot(x1(1),y1(1),phi(1),scale,'b');
hold on;
T1 = plot3(hx(1),hy(1),0,'b','LineWidth',3);
step = 15;

for k=1:step:N
   delete(R1);
   delete(T1);
   R1 = MobilePlot(x1(k),y1(k),phi(k),scale,'b');
   T1 = plot3(hx(1:k),hy(1:k),zeros(k),'b','LineWidth',3);

   pause(ts);
end
```