INSTITUTO TECNOLÓGICO SUPERIOR DE TEZIUTLÁN

Tesis

"Diseño de sistema de control para robot médico de asistencia médica"

PRESENTA:

JOSÉ ALBERTO TEPAL GARCÍA

CON NÚMERO DE CONTROL
17TE0459

PARA OBTENER EL TÍTULO DE:
INGENIERO MECATRÓNICO

CLAVE DEL PROGRAMA ACADÉMICO
IMCT-2010-229

DIRECTOR (A) DE TESIS:
M.S.C. GABRIEL ÁNGEL RAMIREZ VICENTE

" La Juventud de hoy, Tecnología del Mañana"

TEZIUTLÁN, PUEBLA, MAYO 2022

PRELIMINARES

Agradecimientos

A MI FAMILIA

Por estar siempre conmigo, en cada situación,

sin importar lo complicada que haya sido, y porque

fueron la inspiración más importante para poder seguir

y cumplir mis objetivos. En especial doy gracias a mi madre,

que siempre me apoyó en cualquier aspecto, y me dio fortaleza.

A MIS AMIGOS Y COMPAÑEROS

Por disfrutar cada día con ellos cada momento en la carrera,

apoyarnos mutuamente, darme la oportunidad de conocerlos,

y por ser una parte momentánea importante de mi vida.

AL TECNOLÓGICO SUPERIOR DE TEZIUTLÁN

Por todas las oportunidades que me ha brindado, por ser

parte de una institución con calidad, ayudarme a crecer,

no solo como estudiante, también como persona. Me ayudó

a aprender de mis errores y volver a intentarlo, sin darme por vencido.

Resumen

En la vida cotidiana, se presentan situaciones en las cuales, algún elemento del ambiente de forma aleatoria o bien, causado por un descuido, pone en riesgo tanto a cosas a su alrededor como a las personas que se encuentren en un radio considerable de algún accidente o catástrofe, provocando pérdidas materiales, lesiones o en casos extremos, la muerte. Los accidentes suelen ser muy recurrentes sin importar el lugar y que tan accesible o no, lo sea. Por esa razón, la tecnología ofrece soluciones y apoyo en estas situaciones. Por lo consiguiente, en este trabajo se muestra el diseño de un robot capaz de asistir tanto a cuerpos de auxilio profesionales, como a personas capacitadas en el manejo de este, con el objetivo de facilitar y cumplir las tareas que se le asignen, así también, como no poner en peligro a demás personas al enfrentarse a cosas que en el momento se desconocen. Este robot se enfoca en distintas situaciones analizadas mediante casos de uso, con el fin de identificar las limitantes del proyecto. Todo el reporte se divide en 8 capítulos, que van desde el marco teórico, hasta las actividades realizadas, de esta manera, facilita entender los resultados que se obtuvieron y poder obtener conclusiones con base en lo que se experimentó en el desarrollo completo.

Introducción

Día con día, se suelen dar situaciones inesperadas que, en algunos casos, llegan a ser de gravedad por distintos factores, ya sea del ambiente, la inaccesibilidad del lugar o por agentes tóxicos en el aire que afecten la salud de las personas.

La tecnología ha jugado un papel muy importante en los temas de exploración, medicina, asistencia y búsqueda, ya que ayuda a los cuerpos de rescate y paramédicos a valorar la complejidad de los casos y de esa forma poder planear bien lo que se realizará para solucionar la situación.

Así que tomando en cuenta esos aspectos, se propone un prototipo de robot asistente que dé apoyo en las tareas de localización y asistencia médica básica a los servicios de rescate y primeros auxilios. Este robot cuenta con distintos dispositivos como cámara, GPS, y sensores que se encargan de tomar parámetros tanto del ambiente, como de alguna persona afectada por una situación en peligro, por ejemplo: En casos de incendio, fugas de gas o algún otro elemento toxico, accidentes automovilísticos, búsqueda y rescate; incluso se puede usar en casos de pandemia como la que se está viviendo en la actualidad, ya que puede evitar el contacto directo entre trabajadores de la salud y pacientes puestos en cuarentena, de esta forma se pueden estar verificando los parámetros básicos de salud sin tener riesgo de ser contagiado.

Índice general

Capítulo I

GENERALIDADES DEL PROYECTO

1.1 Descripción de la empresa

La empresa ALKIMIA MOBILE S.A. de C.V. La empresa ALKIMIA MOBILE S.A. de C.V. ubicada en calle Montecito No. 11 Col. Nápoles, Delegación Benito Juárez, CDMX es un grupo de consultores especializados en movilidad, plataformas IoT, sensores, big data, con proyectos y soluciones para el mercado.

El giro principal de la empresa es el de proveer soluciones y servicios relacionados con tecnologías disruptivas aplicadas y soluciones de energías renovables. La base tecnológica es soluciones web, móviles e internet de las cosas. En energía renovable, generadores solares, eólicos, híbridos, y bombeo solar.

El sector objetivo es el de la pequeña y mediana empresa, principalmente en las categorías de comercio al por menor, por mayor, retail y agricultura.

En la línea de negocios de servicios, se ofrece consultoría de negocio basados en tecnología, coaching empresarial y en aplicación de tecnología en procesos de venta, marketing y automatización.

Esta empresa ofrece soluciones probadas en el mercado enfocadas a marketing, ventas para el micro empresario, o para la pequeña /mediana empresa.

1.2 Problema de investigación a resolver

En el área de búsqueda y rescate, y de protección civil, se ve la necesidad de poder realizar tareas de manera más satisfactoria, que no afecten la integridad física de las personas encargadas en realizar actividades complejas, así también como ayudar a las personas afectadas por algún accidente. Si bien, los profesionales en esto suelen estar muy capacitados y preparados para cualquier situación, nunca se puede mantener una situación bajo control absoluto. Por lo tanto, se tiene el riesgo de sufrir un accidente, un suceso no planeado, el cual provoca un daño hacia una persona o un objeto, ya sea de manera accidental o intencional. Aquí entran tanto paramédicos, como equipos de rescate, bomberos, etc. Algunos de ellos llegan a sufrir algún daño físico, o incluso perder la vida por no conocer el ambiente o el terreno en el que se encuentran. Por lo tanto, el problema identificado, son los factores que constituyen los riesgos físicos ambientales, que pongan en riesgo la integridad de los distintos actores involucrados en las situaciones anteriormente mencionadas.

1.3 Preguntas de investigación

¿Qué características electrónicas debe tener un robot para ser un prototipo medianamente eficiente?

¿Qué componentes son idóneos para la realización de este proyecto?

¿Qué lenguaje de programación usar?

¿Cuántos casos de uso es posible cubrir?

¿Cómo obtener una interacción intuitiva con la interfaz de control?

1.4 Objetivos

Objetivo general

Desarrollar una interfaz de control remota para el manejo y monitoreo de las funcionalidades del robot.

Objetivos específicos

- Delimitar las funciones del prototipo.
- Investigar y proponer que componentes son los más viables.
- Realizar el diseño del circuito y las conexiones de la electrónica a usar, al igual que la alimentación necesaria.
- Proponer tanto el sistema operativo, software y el lenguaje de programación a utilizar.
- Establecer comunicación remota con el robot y verificar el funcionamiento de los sensores y actuadores.
- Diseñar una interfaz cómoda e intuitiva.

1.5 Justificación

Debido a los riesgos físicos ambientales que en muchas ocasiones se desconocen o no se pueden controlar en situaciones peligrosas como accidentes, la salud de las personas involucradas en el rescate puede verse afectada, o en casos más graves, se puede llegar a perder la vida.

El tener dispositivos enfocados a explorar, ofrecer asistencia médica básica, localización por GPS y obtener los parámetros del ambiente mediante sensores y una cámara con reconocimiento inteligente, puede ayudar a los equipos de paramédicos y de rescate a planear una forma fácil, rápida y segura de llegar a un objetivo, conociendo el estado de la persona afectada y su localización de forma más precisa, tomando en cuenta los distintos valores captados por los sensores del robot, con esto se busca salvaguardar la integridad física de los afectados, como de las personas preparadas y encargadas de ayudar en situaciones de rescate.

En los aspectos económicos y de seguridad resulta ser de gran ayuda, ya que el impacto sería notorio para organizaciones de protección civil al implementar sistemas robóticos como apoyo a los equipos de rescate a lugares complejos o peligrosos. Al no arriesgar vidas humanas solo se considera el mantenimiento del sistema mecánico o electrónico.

Con este proyecto se busca fomentar la investigación y desarrollo de mejores prototipos para actividades similares o diferentes, así también como la implementación de estos.

Capítulo II

MARCO TEÓRICO

2.1 Robots móviles

La robótica móvil actualmente, se encuentra considerada como un área de la tecnología encargada de manejar y resolver problemas de una alta complejidad, puesto que estos dispositivos teleoperados suelen ser empleados en situaciones que implican algún peligro para la vida humana. Un robot, al ser un sistema, está compuesto por aplicaciones de distintas áreas de la ingeniería, como lo son la programación, mecánica, control, instrumentación, electrónica y en algunos casos, de inteligencia artificial, por lo cual, mediante la unión de cada una, se pueden proponer y realizar soluciones innovadoras que resuelvan un problema específico, o bien, realizar múltiples tareas, limitando ciertos aspectos y situaciones.

De acuerdo al diseño de cada robot, pueden realizar ciertas tareas, como por ejemplo:

- Operación en ambientes hostiles.
- Seguridad en caso de accidentes.
- Reconocimiento en ambientes de difícil acceso.

Las propiedades características de los robots son la versatilidad y la autoadaptabilidad. La primera se entiende como la potencialidad estructural de ejecutar tareas diversas y/o ejecutar una misma tarea de forma diversa, lo cual implica una estructura mecánica de geometría variable. La autoadaptabilidad significa que un robot debe, por sí solo, alcanzar su objetivo a pesar de las perturbaciones imprevistas del entorno a lo largo de la ejecución de su tarea; esto supone una consciencia del entorno otorgada por la posesión de sentidos artificiales. (BERMÚDEZ, 2001)

Un robot operacional está constituido por cuatro sistemas relacionados entre sí: mecánico, sensórico, control y alimentación. El sistema mecánico permite el movimiento, utilizando diferentes elementos de acuerdo con el propósito del robot y ubicados estratégicamente. El sistema sensórico brinda información importante para la ubicación de elementos u obstáculos: los sensores propioceptivos permiten

determinar posición y velocidad; los sensores exteroceptivos determinan la interacción del entorno con él mismo. El sistema de control es el cerebro del robot, al interpretar y tratar la información; él debe contener un modelo del robot físico (actuadores y sensores), un modelo del entorno o estrategias para conocerlo, y los programas que permiten realizar las tareas (algoritmo de control). (BERMÚDEZ, 2001)

2.2 Signos vitales

Los signos vitales se pueden definir como muestreos o mediciones que se toman para verificar las funciones básicas del cuerpo, como lo son:

- La frecuencia respiratoria.
- Temperatura corporal.
- Pulso.
- Presión arterial.

Normalmente la temperatura del cuerpo suele variar dependiendo de diversos factores, ya sea por, alguna actividad física realizada con anterioridad, consumo de alimentos o varia también según el sexo de la persona.

La temperatura normal oscila entre 36.5°C a 37.2°C. Esta puede medirse directamente en la piel.

El pulso mide la cantidad de veces que el corazón late por minuto. Al tomar este signo vital, se puede indicar de igual manera tanto el ritmo cardiaco, como la fuerza del pulso. Las cifras normales se encuentran entre 60 y 100 latidos por minuto.

La frecuencia respiratoria, es básicamente la cantidad de veces que una persona realiza por minuto. Esta cantidad oscila entre 12 y 16 repeticiones por minuto.

La presión arterial no se considera un signo vital, sin embargo, se le suele monitorear junto con los signos vitales. Es la fuerza que ejerce la sangre, contra las paredes de las arterias. Los valores normales suelen ser de menos 120 mm Hgy, y menos de 80 mm Hg.

2.3 Placas de desarrollo para IA

Las placas de desarrollo cuentan con un microcontrolador que se puede programar varias veces, el cual ejecuta las instrucciones que el programador le da para cumplir un objetivo específico. Cuenta con entradas/salidas analógicas y/o digitales, esto para poder comunicarse con actuadores y/o sensores conectados a los pines de la placa.

Este tipo de placas, no se limita exclusivamente a una sola área de aplicación, ya que pueden ser usadas en robótica, sistemas de automatización industriales y de hogares, monitoreo, prototipos de sistemas complejos, entre muchas más.

Al no existir un único lenguaje de programación para usar en una placa de desarrollo, se puede elegir entre algunas opciones, como por ejemplo: C, C++, Python, JavaScript, etc.

Incluso algunos modelos, cuentan con la posibilidad de instalar un sistema operativo, por lo general basado en Linux, y gracias a eso, poder usar aplicaciones que ayuden al desarrollo de un proyecto.

Debido a que la mayoría son fáciles de usar, suelen ser muy populares, son de rápido aprendizaje y llegan a tener precios accesibles para la mayoría de público enfocado a desarrollar prototipos, son validadas por distintos desarrolladores para su amplio uso en el ámbito de la tecnología.

Algunas de las placas usadas para IA más accesibles, son:

- Raspberry Pi
- Jetson Nano

2.3.1 NVIDIA Jetson Nano

Jetson Nano, es una placa de desarrollo muy parecida a la placa Raspberry, con la diferencia de que tiene un poco más de potencia, además de ser hecha por NVIDIA especialmente para IA y robótica.

Las principales características son las siguientes:

Figura 2.3.1 Especificaciones Jetson Nano.

VIEW TECHNICAL SPECIFICATIONS >

GPU	128-core Maxwell
CPU	Quad-core ARM A57 @ 1.43 GHz
Memory	4 GB 64-bit LPDDR4 25.6 GB/s
Storage	microSD (not included)
Video Encode	4K @ 30 | 4x 1080p @ 30 | 9x 720p @ 30 (H.264/H.265)
Video Decode	4K @ 60 | 2x 4K @ 30 | 8x 1080p @ 30 | 18x 720p @ 30 (H.264/H.265)
Camera	2x MIPI CSI-2 DPHY lanes
Connectivity	Gigabit Ethernet, M.2 Key E
Display	HDMI and display port
USB	4x USB 3.0, USB 2.0 Micro-B
Others	GPIO, I^2C, I^2S, SPI, UART
Mechanical	69 mm x 45 mm, 260-pin edge connector

Fuente: Jetson Nano Developer Kit. NVIDIA Developer, https://developer.nvidia.com/embedded/jetson-nano-developer-kit, 2021

2.4 Sensores, actuadores y periféricos

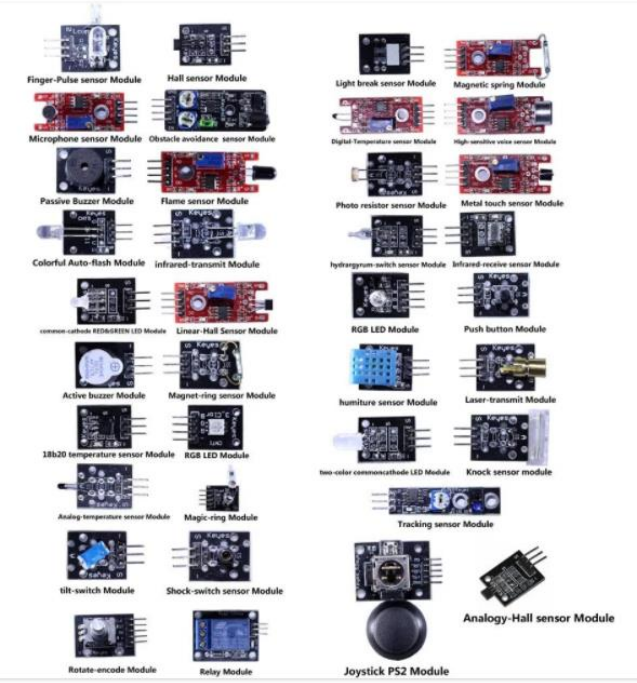
Los sensores, son dispositivos capaces de detectar el entorno que los rodea, de acuerdo a la función para la cual fueron hechos. Los datos que toman los sensores se definen también como variables de instrumentación, estás pueden ser tanto temperatura, distancia, humedad, intensidad lumínica, entre otros. Posteriormente

el sensor se encarga de transformar las variables de instrumentación en variables eléctricas.

Los sensores se pueden clasificar en:

- Digitales
- Analógicos
- Comunicación por Bus

Figura 2.4 Sensores.



Fuente: J, https://aprendiendoarduino.wordpress.com/2016/12/18/sensores-y-actuadores/, 2016

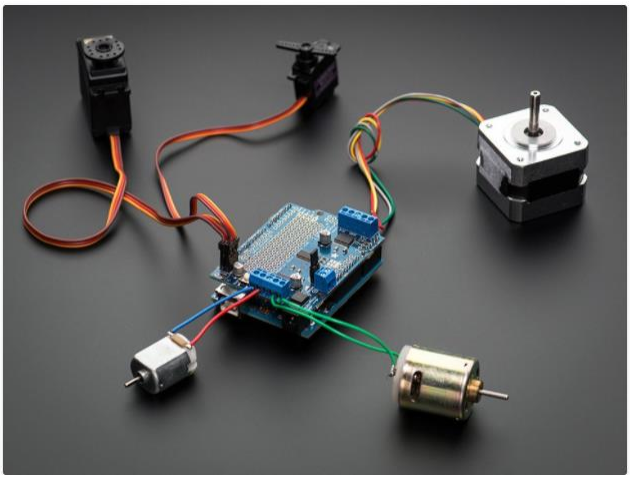
Un actuador, como su nombre lo dice, actúa bajo ciertas condiciones que se cumplan o se programen, es capaz de transformar energía eléctrica, así también como neumática o hidráulica, y activar un proceso. Los actuadores normalmente reciben órdenes en forma de señales mediante un controlador.

Existen distintos tipos de actuadores, como por ejemplo:

- Motores

- Electrónicos
- Hidráulicos
- Neumáticos
- Eléctricos
- Bombas

Figura 2.4.1 Actuadores.



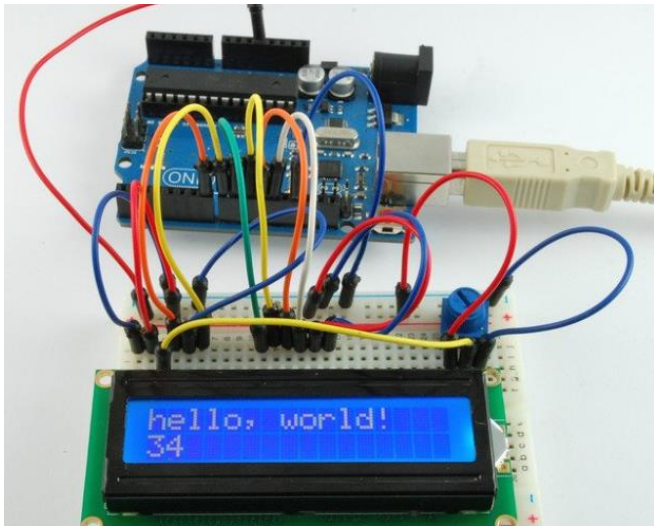
Fuente: J, https://aprendiendoarduino.wordpress.com/2016/12/18/sensores-y-actuadores/, 2016

Por último, los periféricos son dispositivos externos independientes conectados al controlador o unidad central de procesamiento mediante puertos de comunicación.

Algunos ejemplos de periféricos son:

- Cámaras
- Teclados
- Pantallas
- Memorias externas
- Micrófonos
- Mouse

Figura 2.4.2 Periféricos.



Fuente: J, https://aprendiendoarduino.wordpress.com/2016/12/18/sensores-y-actuadores/, 2016

2.5 BMS

Un sistema de gestión de batería o por sus siglas en inglés (BMS) tiene la capacidad de gestionar un control en tiempo real de la carga de cada batería. La elección de cada BMS depende de las baterías conectadas en serie, ya que se basa en distintas configuraciones, por ejemplo: 3s, 4s, 5s, etc.

Figura 2.5 BMS.

Fuente: Lithium Battery Management System (BMS), https://www.powertechsystems.eu/home/products/others/battery-management-system-bms/, 2020

2.6 Protocolo de Escritorio Remoto (RDP)

Es un protocolo propietario desarrollado por Microsoft que permite la comunicación en la ejecución de una aplicación entre una terminal (mostrando la información procesada que recibe del servidor) y un servidor (recibiendo la información dada por el usuario en el terminal mediante el ratón o el teclado)

La información gráfica que genera el servidor es convertida a un formato propio RDP y enviada a través de la red a la terminal, donde se interpreta la información contenida en el paquete del protocolo para reconstruir la imagen a mostrar en la pantalla del terminal; la introducción de órdenes en el terminal del usuario requiere saber que, las teclas que pulse el usuario en el teclado del terminal así como los movimiento y pulsaciones de ratón son redirigidos al servidor, permitiendo al protocolo un cifrado de seguridad donde se permite que toda la información que intercambien entre el cliente-servidor sea comprimida para un mejor rendimiento.

Emplea, por defecto, el puerto TCP 3389 para recibir peticiones, donde al iniciar la sesión desde un punto remoto, el ordenador servidor mostrará una pantalla de

bienvenida donde no se verá lo que el usuario está haciendo. Permite el uso de colores de 8, 16, 24 y 32 bits, con un cifrado de 128 bits utilizando el algoritmo criptográfico RC4; otorga seguridad a nivel de transporte y utilizar su impresora local al estar conectado al sistema remoto.

El redireccionamiento del audio permite al usuario ejecutar un programa de audio en una ventana remota y escuchar e sonido en el ordenador local; el redireccionamiento del sistema de fiches, por su parte, permite utilizar sus ficheros locales; mientras que, el redireccionamiento de puertos permite utilizar los puertos serie y paralelo.

Las aplicaciones remotas pueden funcionar en una máquina cliente servida por una conexión remota, a la par que permite utilizar un servidor IIS de manera que acepte conexiones en el puerto 443 para servidores de respaldo de Terminal Services mediante conexiones HTTPS, similar a como las llamadas remotas RPC sobre HTTP; actualizaciones recientes poseen un soporte para aplicaciones compatibles con clientes .Net Framework 3.0 y que sean capaces de tener efectos en la máquina local.

2.7 Lenguajes de programación

Es un lenguaje formal o artificial que le proporciona a un individuo la capacidad de programar una serie de instrucciones o secuencias de órdenes en forma de algoritmos con el fin de controlar el comportamiento lógico y físico de un sistema informático, a fin de que se puedan obtener diversas clases de datos o ejecutar alguna tarea dando paso a un programa informático. Los lenguajes de programación se conforman por un conjunto de símbolos, reglas gramaticales y semánticas, que en un conjunto permiten definir las estructuras válidas del lenguaje y su significado; este permite especificar de manera precisa qué datos debe trabajar un software específico, su almacenamiento y transmisión así como las acciones debe tomar el software bajo una variada gama de circunstancias.

Se clasifican en base a distintos criterios:

- Clasificación histórica: Con el surgimiento de nuevos estilos de programación expresivas, se distinguieron dichos estilos a partir de una serie de generaciones, cada una representando lenguajes con características comunes.
- Lenguajes de alto y de bajo nivel: Los lenguajes de programación se suele clasificar dentro de dos amplias categorías, referidas a su nivel de especificidad o generalidad.
- Clasificación por paradigmas: Distinguen modelos de cómputo y de estilos de estructurar y organizar las tareas que debe realizar un programa.
- Clasificación por propósito: Son aquellos lenguajes que se distinguen de un propósito general y un propósito específico.

2.8 Python

Dada la popularidad que ha adquirido en la última década, cabría esperar que se trate de un lenguaje nacido a comienzos de este siglo. Nada más lejos de la realidad: su origen se remonta a finales de los años 80 y principios de los 90. Su implementación comenzó en diciembre de 1989 cuando Guido van Rossum, trabajador del CWI (un centro de investigación holandés de carácter oficial) decidió empezar el proyecto como un pasatiempo dándole continuidad al lenguaje de programación ABC desarrollado por el equipo del que había formado parte en el CWI.

Su nombre se debe a la afición de Van Rossum al grupo Monty Python y su concepción se enfocaba en que fuera fácil de usar y aprender sin que esto penalizara sus capacidades. La causa de que no llegara a adquirir la suficiente importancia en su momento fue la falta de recursos en el hardware de la época.

El avance en las tecnologías de hardware ha sido una condición necesaria para el repunte de su popularidad. No obstante, la generalización del big data en los últimos años, seguida de la explosión de la inteligencia artificial, el machine learning, el deep learning y el surgimiento de la ciencia de datos o data science como una nueva área de trabajo con especialistas propios han revolucionado el panorama. Y es que muchas de las nuevas herramientas que han surgido, y que son explotadas por los

ingenieros de datos y los científicos de datos, han sido desarrolladas en Python o nos ofrecen Python como la forma predilecta de interactuar con ellas.

Algunos puntos destacables de Python son:

- Al ser open source no se debe pagar por una licencia.
- Tiene una gran comunidad.
- Es un lenguaje multiparadigma.
- Puede ser utilizado en distintos campos.
- Está disponible, y es apto para todas las plataformas.

2.9 Arduino

Esta placa es una de las más utilizadas para proyectos iniciales en cuanto a electrónica, puesto que cuenta con varias posibilidades para poder manejar lo que se necesita, además de tener una cierta facilidad al momento de programar e instalar las librerías necesarias para los sensores.

Si bien hay muchas otras placas, esta se destaca por lo antes mencionado, agregando la compatibilidad con distintos sistemas operativos para la programación en su IDE.

El funcionamiento de esta placa se divide en 3 etapas importantes:

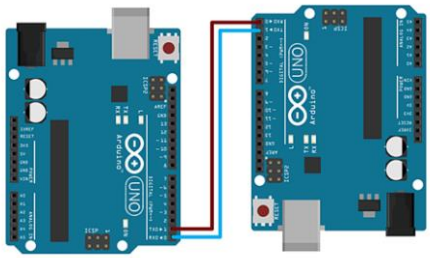
- La interfaz de entrada. Esta corresponde a los sensores y periféricos que se conectarán a la placa, mediante los pines y puertos disponibles.
- La segunda es el tratamiento de las entradas. Esta parte corresponde al microcontrolador y a la programación que le demos.
- Por último, la tercera es la interfaz de salida. Una vez cumplidas las etapas anteriores, se produce una salida.

2.10 Comunicación serial

La comunicación serial, suele ser utilizada muy frecuentemente en proyectos de Arduino. Esta se puede usar para tener una transferencia y recepción de datos mediante un ordenar u otro dispositivo que admita la comunicación, hacia la placa principal de Arduino.

Los pines que permiten esto son: Rx y Tx, siendo Rx el receptor y Tx el emisor. El puerto USB con el que cuenta la placa, permite poder mandar y recibir datos, por lo que tener una conexión a un computador no es una tarea compleja.

Figura 2.10 Conexión serial.



Fuente: Arduino - Comunicación Serial, https://www.ediciones-eni.com/open/mediabook.aspx?idR=256d9467932d6a4b20f41c2bbe0f9b06, 2021

2.11 Baterías 18650

Son pilas de litio recargables (también conocidas como baterías Li-ion), bastante parecidas a las AA, pero con un tamaño más grande, fabricadas para la acumulación de energía eléctrica que utilizan como electrolito una sal de litio.

Son baterías ligeras con una elevada capacidad energética y resistencia a la descarga pero con poco efecto de memoria y resisten un elevado número de ciclos; pueden o no estar protegidas con un circuito que se encarga de controlar cuando la batería no debe descargarse, cortando la tensión de salida y en qué momento se ha cargado lo suficiente, corta la tensión de entrada; tiene un voltaje de 3.7V y un tamaño promedio con cabezal y protecciones incluidas es de 66.5 mm; posee una capacidad de hasta 6000 mAh siendo los parámetros normales de 1600-3600 mAH y con 25-

30A. Poseen una elevada densidad de energía y almacenan mucha más carga por unidad de volumen, con poco peso y poco espesor, siendo excelentes dispositivos portátiles.

Capítulo III

DESARROLLO Y METODOLOGÍA

3.1 Procedimiento y descripción de actividades realizadas

En este capítulo, se muestran los apartados del desarrollo y las actividades realizadas para poder entender más a fondo la parte de resultados y la forma en cómo se llega a estos.

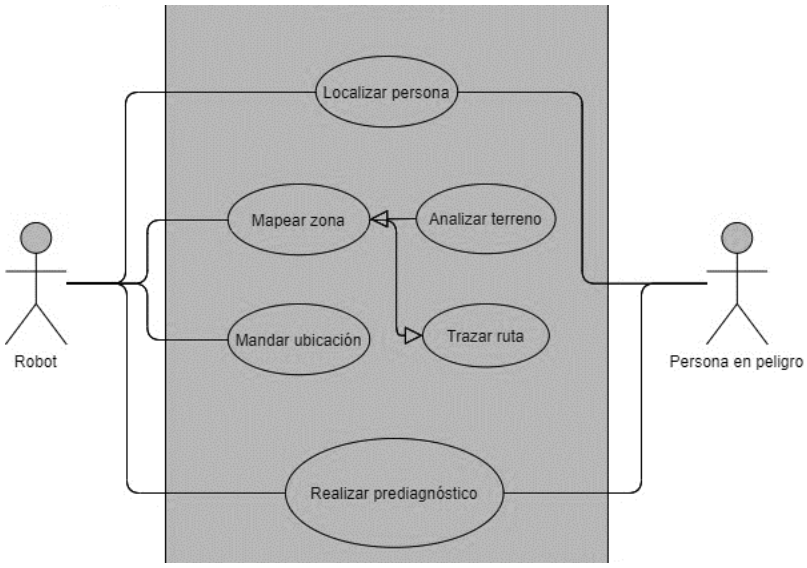
3.1.1 Elaboración y análisis de casos de uso

Para desarrollar un proyecto en el cuál se manejan distintas situaciones, se debe tener muy en claro cuáles son los aspectos que se relacionan en cada escenario posible, por lo que la realización de diagramas de casos de uso, resulta ser una herramienta bastante útil al momento de analizar de forma individual y en conjunto cada situación, al definir una secuencia de acciones da como resultado un valor observable y se obtiene una estructura funcional.

A continuación se muestran los diagramas de casos de uso:

Caso 1 – Rescate/Búsqueda

Figura 3.1.1 Caso de uso 1.

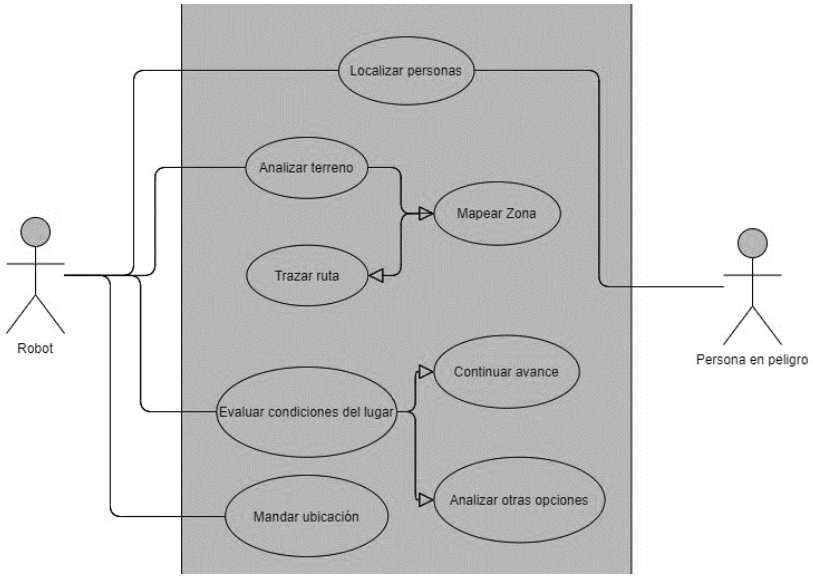


Fuente: Autoría propia basada en VisualParadigm, 2021

Palabras claves de Caso 1: Localización, mapeo, ubicación, ruta, prediagnóstico.

Caso 2 – Deslaves

Figura 3.1.2 Caso de uso 2.

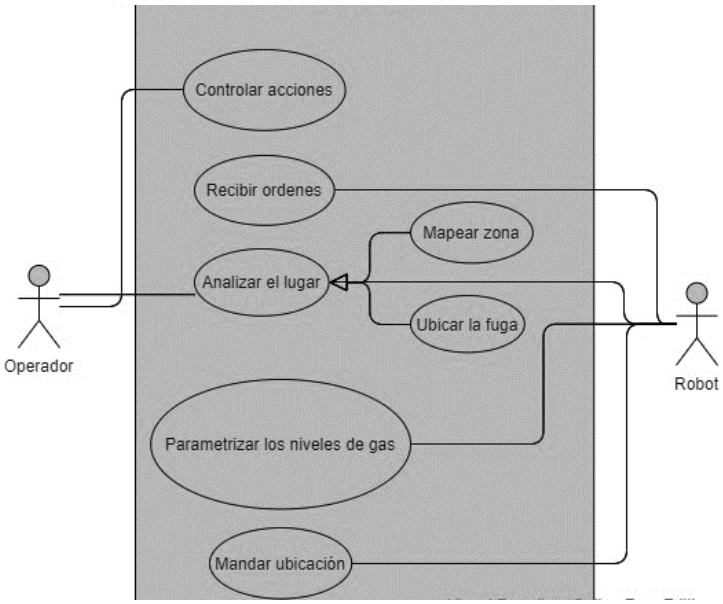


Fuente: Autoría propia basada en VisualParadigm, 2021

Palabras claves de Caso 2: Localización, mapeo, ruta, evaluación, continuación, planeación, ubicación.

Caso 3 – Fuga de gas

Figura 3.1.3 Caso de uso 3.

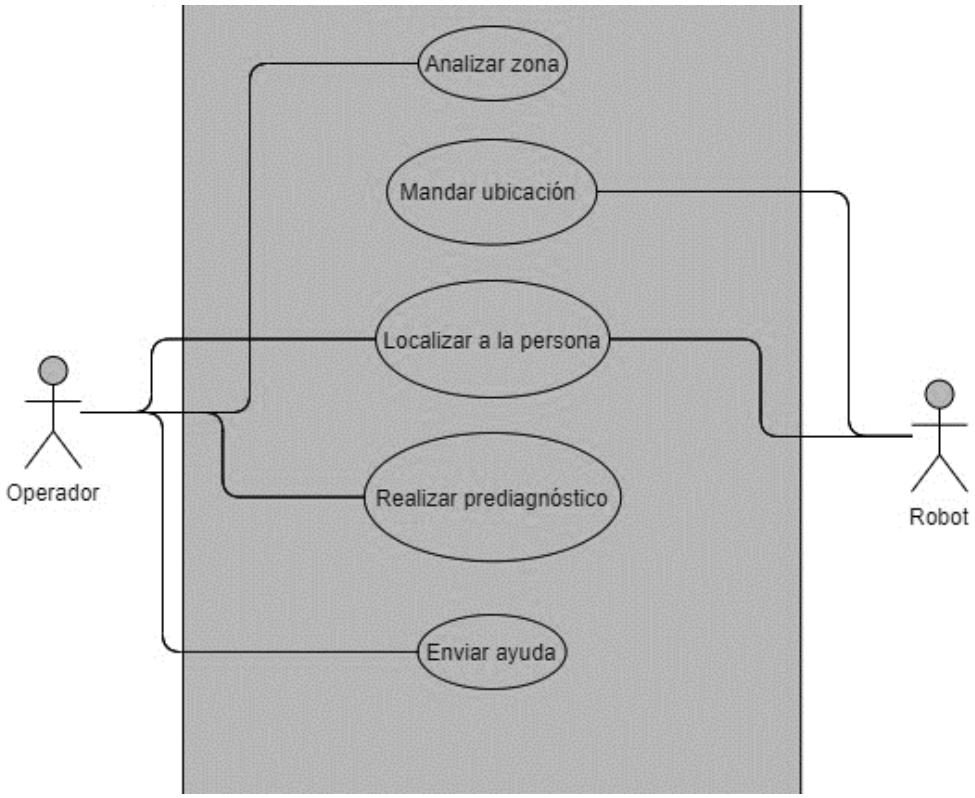


Fuente: Autoría propia basada en VisualParadigm, 2021

Palabras claves de Caso 3: Control, ordenes, mapeo, ubicación, medición.

Caso 4 – Accidente automovilístico

Figura 3.1.4 Caso de uso 4.

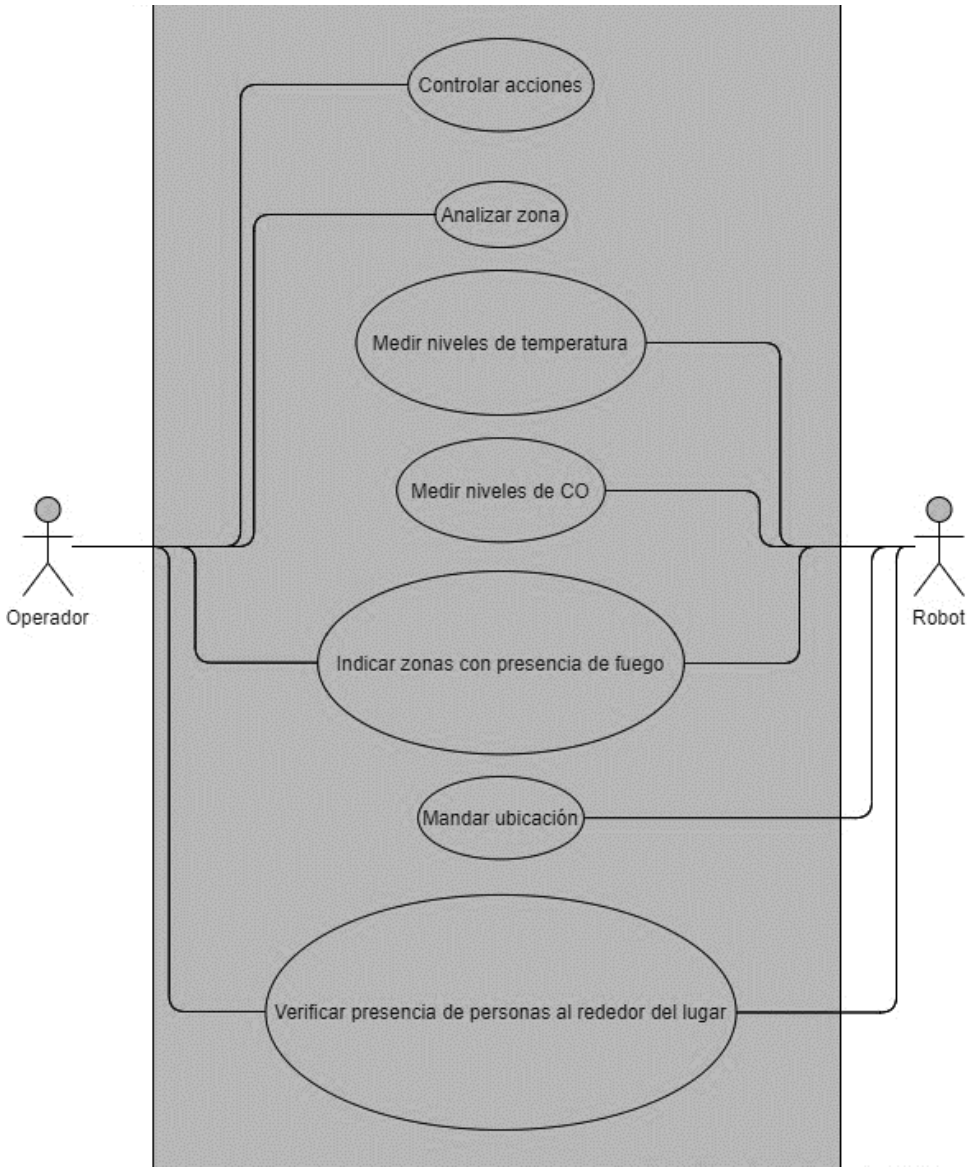


Fuente: Autoría propia basada en VisualParadigm, 2021

Palabras claves de Caso 4: Ubicación, localización, prediagnóstico, intervención humana.

Caso 5 – Incendios

Figura 3.1.5 Caso de uso 5.

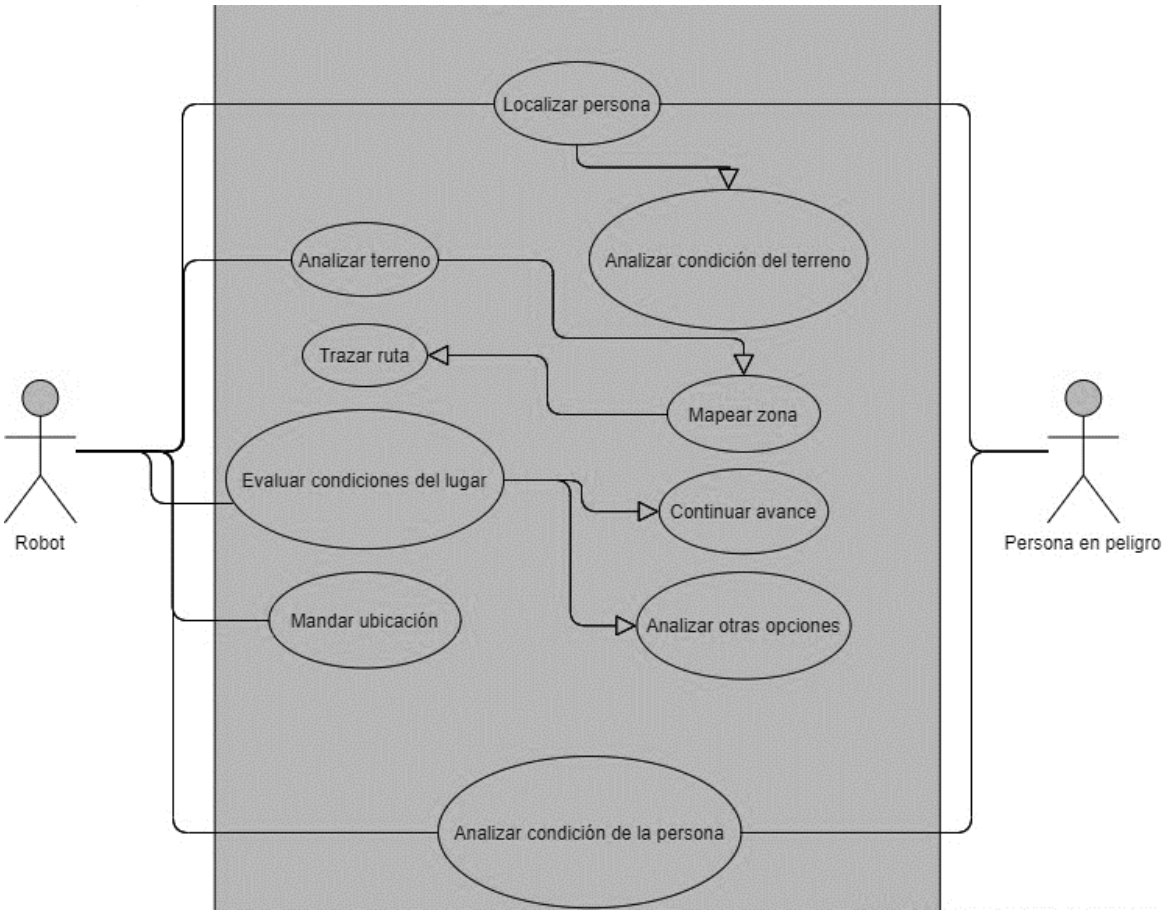


Fuente: Autoría propia basada en VisualParadigm, 2021

Palabras claves de Caso 5: Control, mapeo, medición, detección, ubicación.

Caso 6 – Sismos/Terremotos

Figura 3.1.6 Caso de uso 6.

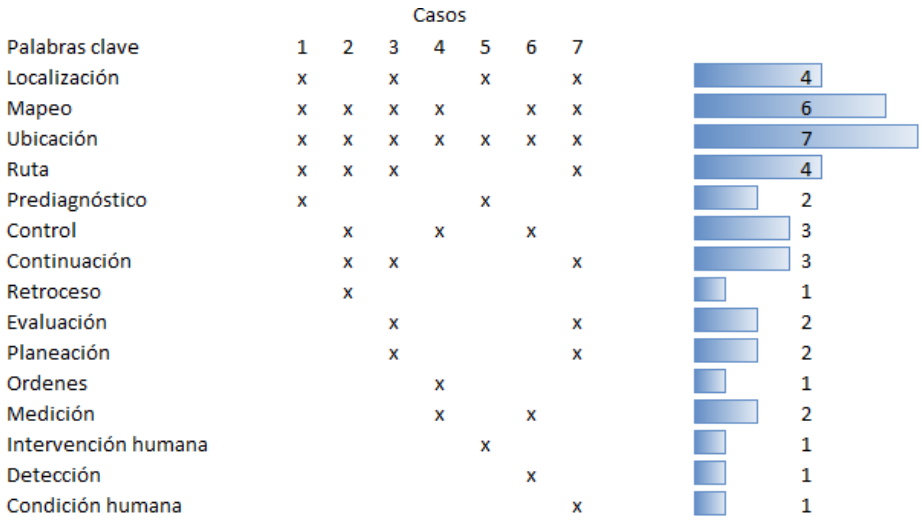


Fuente: Autoría propia basada en VisualParadigm, 2021

Palabras claves de Caso 6: Localización, mapeo, ruta, evaluación, continuación, planeación, ubicación, condición humana.

Para identificar palabras claves que tengan relación entre cada caso, se colocan debajo de cada diagrama, por lo que, esto facilita el reconocimiento de lo que se va a enfrentar en la mayoría de situaciones el robot y poder elegir los materiales ideales para su funcionamiento.

Figura 3.1.7 Palabras Clave.



Fuente: Autoría propia basada en Microsoft Excel, 2021

3.1.2 Elección de elementos electrónicos a usar

Tomando en cuenta las palabras claves que se identificaron, se pasa a la elección de materiales que ayuden al proceso de solución de una o varias situaciones. Investigando características de los productos, teniendo un presupuesto limitado, así también como el tiempo, se eligen los siguientes componentes:

- Motorreductor

  Modelo: JGA25-370

  Velocidad RPM: 130

  Color: Tono plateado

  Tamaño: Aprox. 25mm x 70mm

  Diámetro del Eje: 4mm

  Tipo de Eje: "D"

Voltaje Nominal: 12V DC

Tipo: Motor de engranaje

Conmutación: Sin escobillas

Peso: 85g

Figura 3.1.8 Motorreductor.

Fuente: Catálogo virtual de tienda en línea, 2021

El motorreductor al tener una velocidad intermedia y un buen torque de aproximadamente 1 kg, resulta ser óptimo para el movimiento de las cuatro ruedas.

- Modulo Gps Neo6m

  Voltaje de Alimentación: 3.3~5v

  Consumo de corriente: 45 mA

  Comunicación: UART

  Velocidad de comunación: 9600 bps

  Memoria: EEPROM

  Bateria de respaldo

  LED indicador de bloqueo GPS

  Velocidad de transmisión predeterminada: 9600

  Tamaño del módulo: 25 x 35 mm

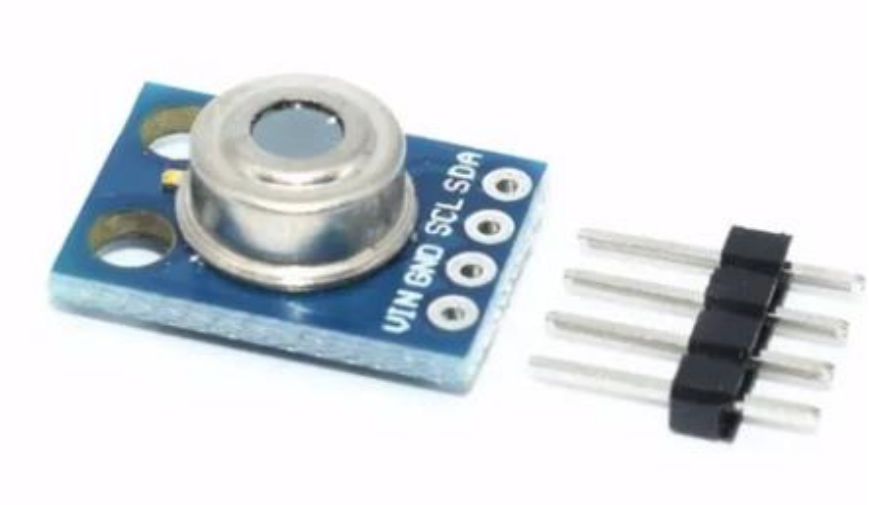
Figura 3.1.9 GPS.

Fuente: Catálogo virtual de tienda en línea, 2021

Un módulo GPS funciona perfectamente para tener conocimiento de la ubicación del dispositivo en tiempo real con un error de precisión de 2.5m.

- Sensor De Temperatura Infrarrojo Gy-906 Mlx90614
  Sensor: MLX90614ESF-BAA
  Voltaje de Operación: 3.3V – 5V DC
  Rango de temperatura ambiente de trabajo: -40°C hasta +170°C
  Rango de temperatura de objeto: -70°C hasta +380°C
  Precisión: ±0.5°C
  Angulo de visión: 90° (FOV)
  Distancia para temperatura de objeto: Min. 2cm y Max. 3cm
  ADC incorporado de 17 bits
  Salida de PWM: 10 bits
  Protocolo de comunicación SMBUS (I2C)
  No necesita componentes adicionales

Figura 3.1.10 Sensor de Temperatura.



Fuente: Catálogo virtual de tienda en línea, 2021

Los sensores de temperatura pueden ser usados en máquinas o personas.

- Sensor De Pulso Cardiaco
  Voltaje de alimentación: 3 V a 5 V
  Consumo de energía: 4 mA
  Medición: Pulso cardíaco
  Número de cables: 3
  Largo de cables: 18 cm
  Diámetro sensor: 1.5 cm

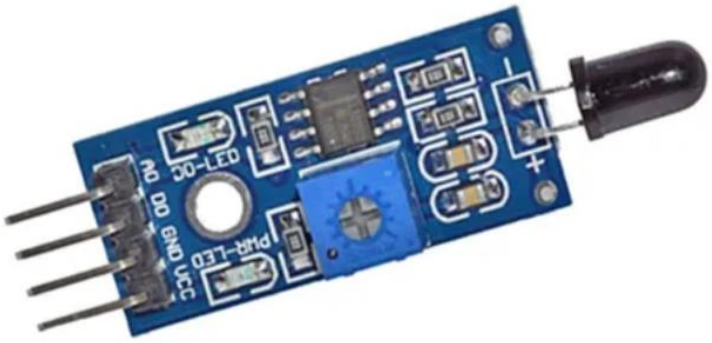
Figura 3.1.11 Sensor de Pulso Cardiaco.

Fuente: Catálogo virtual de tienda en línea, 2021

Los sensores que se encargan de medir signos vitales, son indispensables para obtener el estado de salud básico de una persona.

- Sensor Infrarrojo Detector De Fuego Oky3053
  Compatible con Arduino, Raspberry, PIC, etc.
  1 Salida digital
  1 Salida analógica
  Indicador LED salida digital
  Potenciómetro para regular sensibilidad
  Dimensiones: 32*14mm

Figura 3.1.12 Sensor de Fuego.



Fuente: Catálogo virtual de tienda en línea, 2021

En el ambiente se suelen generar situaciones que no se pueden controlar, como los incendios o pequeñas llamas, por lo tanto, un sensor de fuego y temperatura

- Modulo Sensor De Temperatura Y Humedad Dht22
  Señal de salida: Señal digital vía bus único
  Voltaje de alimentación mínimo: 3.3 V
  Voltaje de alimentación máximo: 6 V
  Corriente durante medición mínima: 1 -mA
  Corriente durante medición máxima: 1.5 mA
  Corriente en modo de espera mínima: 40 µA
  Corriente en modo de espera máxima: 50 µA
  Precisión de HR: +/- 2%
  Rango de operación: Humedad 0-100% RH; temperatura -40 ~ 80 Celsius
  Humedad de precisión: + -2% RH (Max + -5% RH); temperatura <+ -0.5Celsius
  Resolución o sensibilidad: Humedad 0.1% HR; temperatura 0.1 Celsius
  Repetibilidad: Humedad + -1% RH; temperatura + -0.2 Celsius
  Histéresis de humedad: + -0.3% RH
  Estabilidad a largo plazo: + -0.5% RH / año
  Periodo de detección media: 2s

Temperatura de operación mínima: -40

Temperatura de operación máxima: 80° C

Color: Blanco

Número de pines: 4 pines

Figura 3.1.13 Sensor de Temperatura y Humedad.

Fuente: Catálogo virtual de tienda en línea, 2021

- Servomotor Mg995

  Engranes de metal

  Peso: 55 g

  Voltaje de operación: 4.8 a 6V

  Grados / Angulo de Rotación Máximo: 180°

  Velocidad de operación a 4.8V es de 0.17 Seg/60°

  Soporta hasta 15Kg-cm con 6 V (Para uso continuo se recomienda 12kg Máximo)

  Dimensión: 40x20x36.5mm

Figura 3.1.14 Servomotor.

Fuente: Catálogo virtual de tienda en línea, 2021

Para el movimiento del robot en la parte superior, en la cual se localiza la cámara, y entre otros sensores, los servomotores son la opción más viable para un rango de visión óptimo.

- TF Mini Lidar
  Marca: MakerFocus
  Rango: 0.3-12m
  Voltaje de entrada: 4.5-6V

Figura 3.1.15 Mini Lidar.

Fuente: Catálogo virtual de tienda en línea, 2021

Los sensores Lidar sirven para medir distancias, teniendo límites en rangos mínimos y máximos.

- Módulo de cámara para NVIDIA Jetson Nano
  Megapíxeles: 8MP
  Modelo: IMX219
  Campo de visión: 170°
  Distorsión inferior al 22%
  Ideal para inteligencia artificial

Figura 3.1.16 Cámara.

Fuente: Catálogo virtual de tienda en línea, 2021

Para visualizar el entorno de manera remota, se necesita una cámara compatible con la placa principal Jetson Nano.

- NVIDIA Jetson Nano
  RAM: 2GB
  Arquitectura: ARM
  Sistema Operativo: Linux

Figura 3.1.17 Jetson Nano.

Fuente: Catálogo virtual de tienda en línea, 2021

La placa de desarrollo de la empresa NVIDIA es la pieza fundamental del prototipo, desde poder controlar los movimientos hasta la lectura de algunos sensores y el procesamiento de imágenes.

- Módulo Puente H

  Chip L298n Dual Puente H.

  Voltaje Alimentación de motor: + 5 V ~ + 35 V

  Corriente Maxima: 2A / puente

  Parte lógica - Rango de alimentación de terminal VSS: 4.5 - 5.5 V

  Parte lógica - Corriente de funcionamiento: 0 ~ 36ma

  Voltaje de entrada para señal de control: 4.5-5.5 V baja - 0 V alta

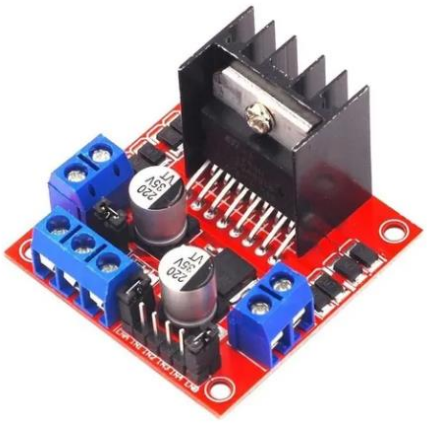
  Consumo: 20 W

  Temperatura de trabajo: -25 ~ + 130

  Tamaño: 55mm * 60mm * 30mm

  Peso: 33g

Figura 3.1.18 Puente H.



Fuente: Catálogo virtual de tienda en línea, 2021

Para realizar giros en motores, en sentido horario o anti horario, se necesitan módulos de Puentes H. Al manejar 4 motores individuales, se necesitan 2 módulos, debido a que cada uno cuenta con dos salidas.

- Display Indicador De Carga Radox 535-160
  Compatible con baterías selladas de plomo, baterías de litio y baterías de níquel
  Alimentación: 8v - 100v
  Medidas 6.1 X 3.4 X 1.4 cm
  Peso: 20g

Figura 3.1.19 Indicador de Carga.

Fuente: Catálogo virtual de tienda en línea, 2021

Un display facilita la lectura de carga total almacenada en las baterías.

- Módulo de carga BMS

  Voltaje de alimentación de BMS: 13 ~ 16 V

  Sobre el rango de voltaje de descarga: 2.3 ~ 3 V ± 0.05 V

  Corriente máxima de funcionamiento: 0 ~ 25 A

  Corriente transitoria máxima: 34 ~ 40 A

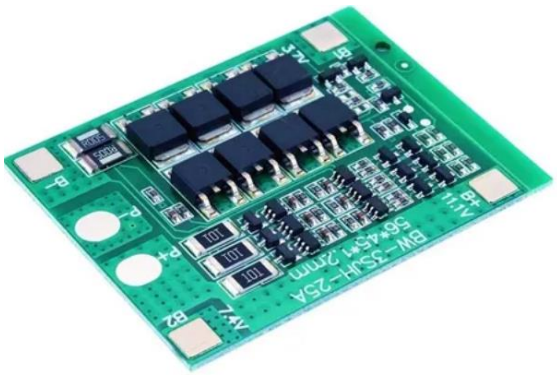
  Corriente de reposo: < 30 uA

  Temperatura de trabajo: - 40 ~ 50 °C

  Vida efectiva: > 30,000 h

  ADECUADO PARA: • 10.8 V (voltaje nominal de batería de polímero). • 11.1 V(18650 o 3.7 V voltaje nominal de batería de litio) • 12.6 V(tensión de carga completa de batería de litio)

Figura 3.1.20 BMS (2).



Fuente: Catálogo virtual de tienda en línea, 2021

Las baterías necesitan estar protegidas por módulos que sean compatibles con la configuración asignada, que eviten el daño de carga y descarga.

- Espaciador porta pila 18650

  Material: ABS + PC

  Modelo: 18650

  Color principal: negro

  Tamaño: 22 x 22 mm / 0.8 x 0.8 pulgadas (L x W)

Diámetro interno: 18.3 mm / 0.7 pulgadas

Figura 3.1.21 Espaciador.

Fuente: Catálogo virtual de tienda en línea, 2021

Para formar un paquete de baterías es necesario utilizar espaciadores.

- Baterías 18650

  Voltaje: 3.7v

  Capacidad: 3700mAh

Figura 3.1.22 Baterías 18650.

Fuente: Catálogo virtual de tienda en línea, 2021

La alimentación de los componentes del robot se obtiene mediante baterías.

- Xt60 Conectores

Figura 3.1.23 XT60 Conectores.

Fuente: Catálogo virtual de tienda en línea, 2021

Los conectores de tipo XT60 se utilizan normalmente en paquetes de baterías.

- Power Bank

  Capacidad: 7500mAh

  Voltaje: 5v

Figura 3.1.24 Power Bank.

Fuente: Catálogo virtual de tienda en línea, 2021

Para dividir la alimentación de los motores y los sensores, de la placa principal, se requiere un segundo banco de baterías para extender el tiempo de uso del dispositivo.

3.1.3 Programa de actividades

Mediante este cronograma se organizan todas las partes las cuales se esperan cubrir en tiempo y forma.

Figura 3.1.25 Programa de actividades.

N	Actividad	Asignaciones	Duración	Fecha inicio	Fecha finalizacion	Estado	%	Observaciones
1	Control a distancia del robot							
		Retos a cubrir	1 dia	27-sep	27-sep	Completo	100%	
		Investigar tipos de control remoto	2 dias	27-sep	28-sep	Completo	100%	
		Analizar control a utilizar	1 dias	29-sep	29-sep	Completo	100%	
2	Vision remota							
		Retos a cubrir	1 dia	29-sep	29-sep	Completo	100%	
		Investigar sobre vision computacional	2 dias	30-sep	01-oct	Completo	100%	
		Comprobar viabilidad de los sensores propuestos	1 dia	01-oct	01-oct	Completo	100%	
3	Envio de datos							
		Retos a cubrir	1 dia	04-oct	04-oct	Sin completar	0%	
		Investigar sobre tipos de comunicación	3 dias	04-oct	06-oct	Sin completar	0%	
		Investigar protocolos de comunicación	2 dias	07-oct	08-oct	Sin completar	0%	
		Analizar el tipo de comunicación y protocolos a utilizar	1 dias	08-oct	08-oct	Sin completar	0%	
4	Diagramas y programación							
		Diagrama de flujo	2 dias	11-oct	12-oct	En proceso	100%	
		Diagrama de control	2 dias	13-oct	14-oct	Completo	100%	
		Programación del robot	3 semanas	14-oct	01-nov	En proceso	75%	
		· Analisis del software usado por los kits	1 dia	19-oct	19-oct	En proceso	100%	Se determino el no utilizar el software de los kits ya que no son compatibles con el modelo de placa adquirido
		· Busqueda de software a utilizar	1 dia	20-oct	20-oct	En proceso	100%	
		· Descarga del software	1 dia	20-oct	20-oct	En proceso	100%	El software se descarga al momento de instalarse ya que esto se realiza en la terminal de la placa al introducir los codigos, esto tambien se hace para asegurar que se estan utilizando las ultimas versiones de cada software
		· Instalación del sistema operativo a la Jetson Nano	1 dia	21-oct	21-oct	Completo	100%	Al momento de realizar la instalación se comprobó que no es compatible con el sistema operativo de las placas de 4GB, ya que, al ser este grabado en la memoria, no permite su instalación. Por el contrario, el sistema especial para las placas de 2GB no presentó problemas al
		· Instalación de librerias necesarias para la programación	1 dia	21-oct	21-oct	En proceso	50%	Se descargaron librerias con ejemplos para el reconocimeinto de imagen
		· Conexión remota de la Jetson Nano	2 dias	22-oct	25-oct	En proceso	100%	La conexión remota se divide en dos partes, se logró accesar a la placa jetson por medio de la terminal. La segunda parte consta de realizarlo por medio de una interfaz gráfica y es la que hace falta.
		· Pruebas de conexión remota	1 dia	25-oct	25-oct	Sin completar	100%	
		Programación de la camara					25%	
		· Programación de los sensores y movimiento del robot	9 dias	26-oct	05-nov	Sin completar	40%	Una vez que se termine de instalar el software necesario y se tengan conectados los sensores, comenzará la programacion de estos
5	Desarrollo físico del prototipo							
		Búsqueda de proveedores	1 dia	14-oct	14-oct	Completo	100%	
		Adquisición de materiales	*desconocido*	15-oct	*desconocido*	Completo	100%	Los componentes llegaron al domicilio de pepe
		Impresión de piezas 3D	6 dias	19-oct	26-oct	En proceso	66%	Fueron modificadas algunas piezas antes de su impresión, aún faltan algunas de ser impresas
		Ensamble del robot	3 dias	27-oct	29-oct	Sin completar	0%	El ensamble se realizará en conjunto con la programación para determinar que sean colocados los elementos de manera correcta y que estos no interfieran con la funcionalidad del
6	Pruebas							
		Pruebas del robot	4 dias	01-nov	04-nov	Sin completar	0%	
		Corrección de errores	3 dias	05-nov	09-nov	Sin completar	0%	
		Ajustes finales	3 dias	10-nov	12-nov	Sin completar	0%	
7	Entrega del prototipo							
		Demostración final del robot	1 dia	15-nov	15-nov	Sin completar	0%	
					Progreso total del proyecto:		62%	

Fuente: Autoría propia basada en Excel, 2021

Capítulo IV

RESULTADOS

4.1 Resultados

En este capítulo se muestra el desarrollo que conforma el control del robot, tanto del sistema operativo, como la parte electrónica y de programación.

4.1.1 Cableado de motores

Cuando se termina de elegir los componentes necesarios, se pasa a las conexiones de los motores.

Figura 4.1.1 Motores.

Fuente: Autoría propia, 2021

Una vez ya impresas las piezas mecánicas, se procede a colocar los motores en su respectivo acople, para poder soldarlas.

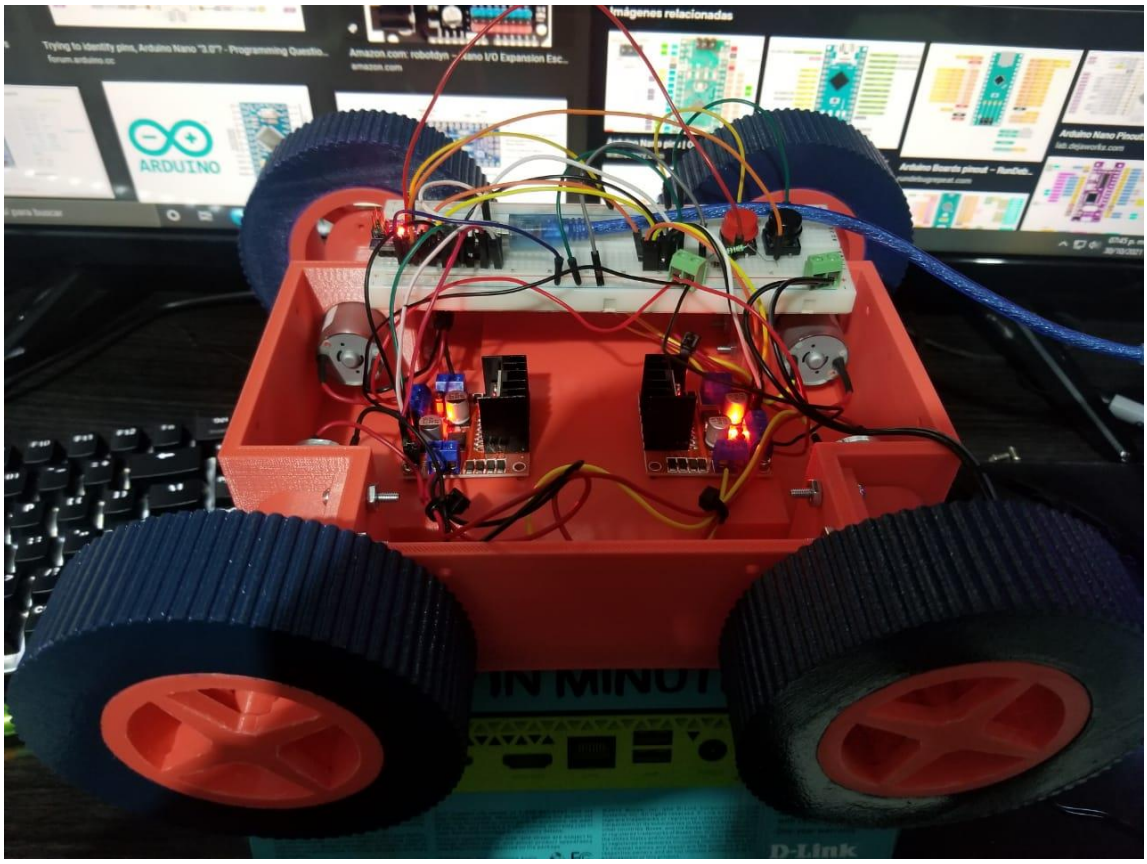
Figura 4.1.2 Cableado de motores.

Fuente: Autoría propia, 2021

Se mide la distancia que tendrán los motores del driver L298, se corta cada cable y se procede a soldar cada uno en las terminales de los motores, para después recubrir la soldadura con tubo termoretractil, en este caso se usa rojo-negro y rojo-amarillo, la combinación R-N se usa para los motores traseros, y la combinación R-A para los motores delanteros.

4.1.2 Pruebas básicas de movimiento mediante Arduino

Figura 4.1.3 Pruebas de movimiento.



Fuente: Autoría propia, 2021

Cuando se termina de soldar cada motor, se colocan en la base, y para verificar que los 4 funcionan de manera correcta se hace un circuito simple, con dos botones que controlan el giro de los 4 motores dependiendo cual se presione, para esto se sube a la placa de Arduino el correspondiente código.

```
int IN1=8;
int IN2=7;
int IN3=13;
int IN4=12;
int IN1D=4;
int IN2D=5;
int IN3D=9;
int IN4D=10;

const int b1=3;
const int b2=2;

int b1state=0;
int b2state=0;

void setup()
{
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);

  pinMode(IN1D, OUTPUT);
  pinMode(IN2D, OUTPUT);
  pinMode(IN3D, OUTPUT);
  pinMode(IN4D, OUTPUT);

  pinMode(b1, INPUT);
  pinMode(b2, INPUT);
}
```

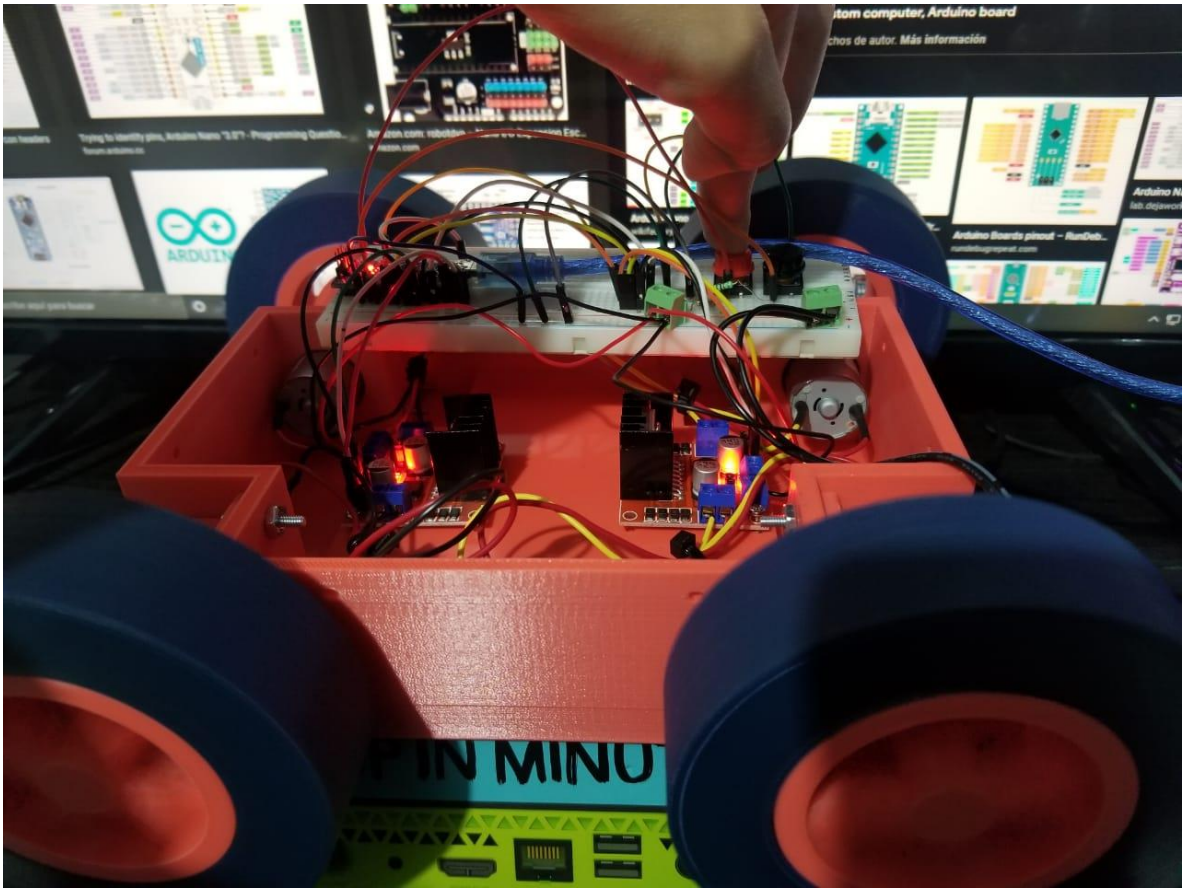
Primero se realiza la declaración de variables, se les asigna un pin a cada una, al igual que a los pines de entrada de ambos botones. Se especifica que pines son salidas y que otros son entradas. Posteriormente, se inicia el programa principal con distintas sentencias if-else.

```
void loop()
{
  if (digitalRead(b1)>b1state)
  {
    digitalWrite(IN1, HIGH);
    digitalWrite(IN2, LOW);
    digitalWrite(IN3, HIGH);
    digitalWrite(IN4, LOW);
    digitalWrite(IN1D, HIGH);
    digitalWrite(IN2D, LOW);
    digitalWrite(IN3D, HIGH);
    digitalWrite(IN4D, LOW);
  }
  else
  {
    delay(100);
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, LOW);
```

```
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, LOW);
    digitalWrite(IN1D, LOW);
    digitalWrite(IN2D, LOW);
    digitalWrite(IN3D, LOW);
    digitalWrite(IN4D, LOW);
  }
  if (digitalRead(b2)>b2state)
  {
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, HIGH);
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, HIGH);
    digitalWrite(IN1D, LOW);
    digitalWrite(IN2D, HIGH);
    digitalWrite(IN3D, LOW);
    digitalWrite(IN4D, HIGH);
  }
  else
  {
    delay(100);
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, LOW);
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, LOW);
    digitalWrite(IN1D, LOW);
    digitalWrite(IN2D, LOW);
    digitalWrite(IN3D, LOW);
    digitalWrite(IN4D, LOW);
  }
}
```

Se hace una comparación entre la variable del botón 1 y el estado, o valor ya definido de b1state. Por lo tanto, si b1 es mayor que b1state el giro hacia un sentido de los motores se activa, mientras el botón se mantenga oprimido, la sentencia se cumplirá. Cuando se deja de presionar el botón, los motores se detienen y todos los pines pasan a estado bajo, de igual manera se repite con el botón 2, solo que ahora los motores girarán en sentido contrario.

Figura 4.1.4 Pruebas de movimiento parte 2.



Fuente: Autoría propia, 2021

Al verificar que cada motor funciona correctamente, se pasa a los demás puntos.

4.1.3 Diseño de banco de batería

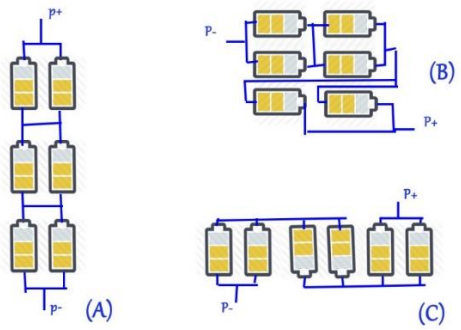
Actualmente, las baterías de iones de litio son muy utilizadas en aplicaciones de robótica móvil. Como se sabe, hay baterías de Polímero de litio, las cuales también suelen ser usadas, y ser una buena opción fácil y rápida, si bien, cuentan con una capacidad bastante buena, suelen tener un tamaño grande, por lo que, al no tener el espacio suficiente en la base, es mucho más favorable usar un banco especial para esta situación.

Los paquetes de baterías tienen una ventaja ante las LiPo, ya que de acuerdo a la necesidad que se tenga, se puede personalizar y armar el banco de una manera que

sea pueda solventar la energía que consumirá un proyecto, dando un margen de tiempo considerable de uso.

Debido a que el consumo de corriente total por parte de los sensores y actuadores da un total de aproximadamente 2900mAh a 3000mAh, tomando en cuenta la corriente necesaria al igual que el voltaje de 12v, se colocan las baterías de acuerdo a la configuración elegida.

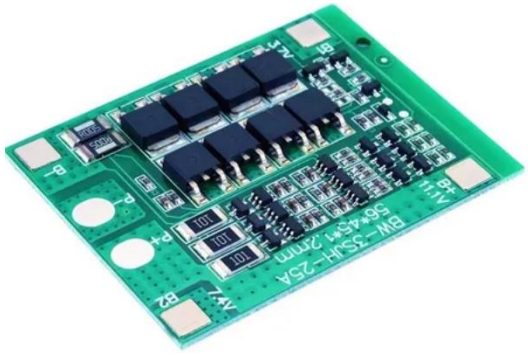
Figura 4.1.5 Configuración de baterías.



Fuente: Real examples of 3s2p 18650 battery. Lithium Ion Battery Manufacturer and Supplier in China-DNK Power, https://www.dnkpower.com/18650-battery-pack-calculator/real-examples-of-3s2p-18650-battery/, 2018

En la imagen 4.1.5 se muestra la configuración 3s2p, es decir 3 baterías en serie y 2 en paralelo.

Figura 4.1.6 Módulo BMS para 3s.

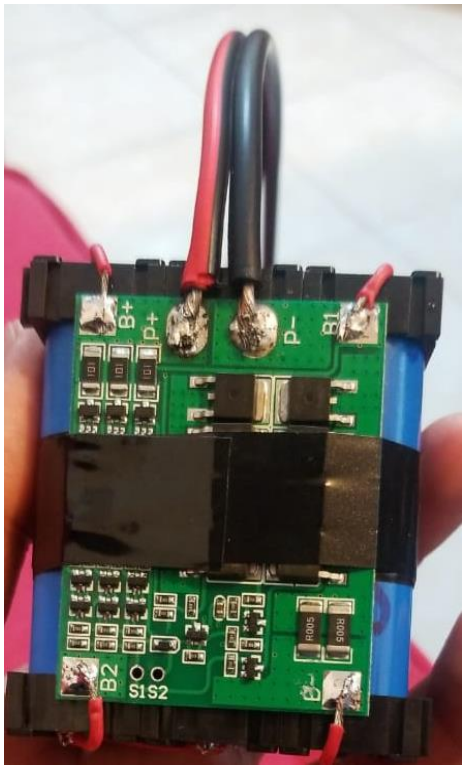


Fuente: Catálogo virtual de tienda en línea, 2021

Posteriormente cada salida de los cables se une con estaño a la placa BMS, este es un modelo específico para baterías 3s, el cual tiene una vida efectiva de más de 30,000h, una característica notable de algunos de estos módulos es la facilidad de cargar, puesto que por el mismo cable que alimenta se puede cargar perfectamente. No necesita de un cargador o balanceador especial; el mismo módulo balancea las baterías y evita sobredescargas y sobrecargas.

Para que el banco tenga más soporte, sin necesidad de pegar las baterías y evitar un posible fallo en estas a causa de un corto inesperado, se usan separadores que ayuda a que las baterías se mantengan en su lugar con un especio pequeño que las divide.

Figura 4.1.7 Paquete de baterías.



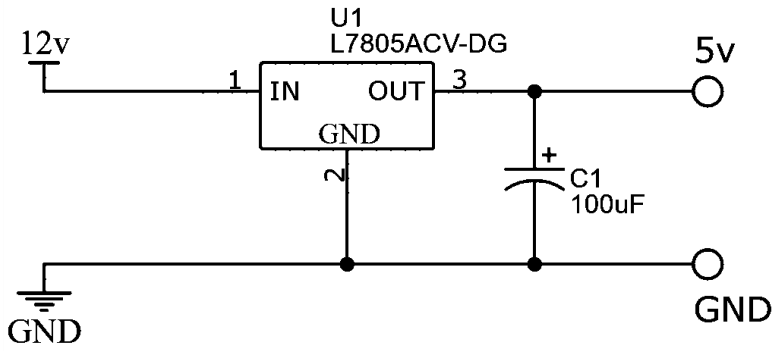
Fuente: Autoría propia, 2021

De esta manera queda armado el paquete de baterías a 11.1v con aproximadamente 6A.

4.1.4 Diseño de circuito de alimentación

Debido a que no todos los componentes soportan 11.1v, se usa la manera de reducir el voltaje mediante un regulador de tensión L7805CV. La familia de estos dispositivos electrónicos se caracteriza por que son reguladores lineales muy estables.

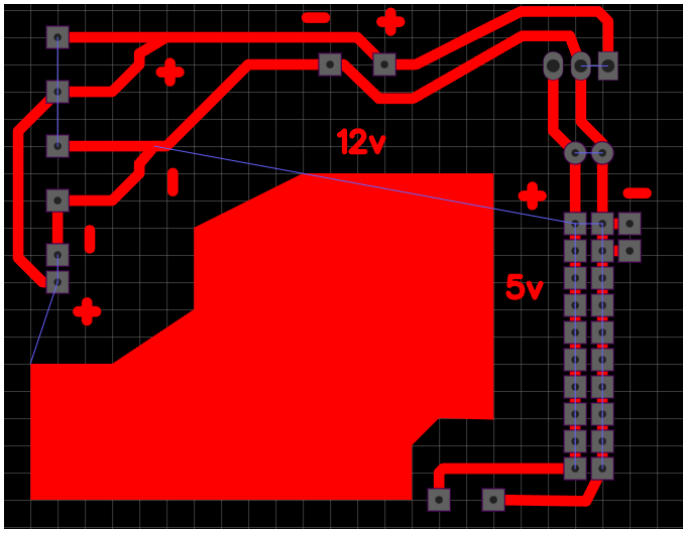
Figura 4.1.8 Circuito regulador.



Fuente: Autoría propia basada en Simulador EasyEDA, 2021

En la imagen se muestra un circuito simple que otorga una salida de 5v para manejar los sensores que requieren ese voltaje.

Figura 4.1.9 Diseño de placa.



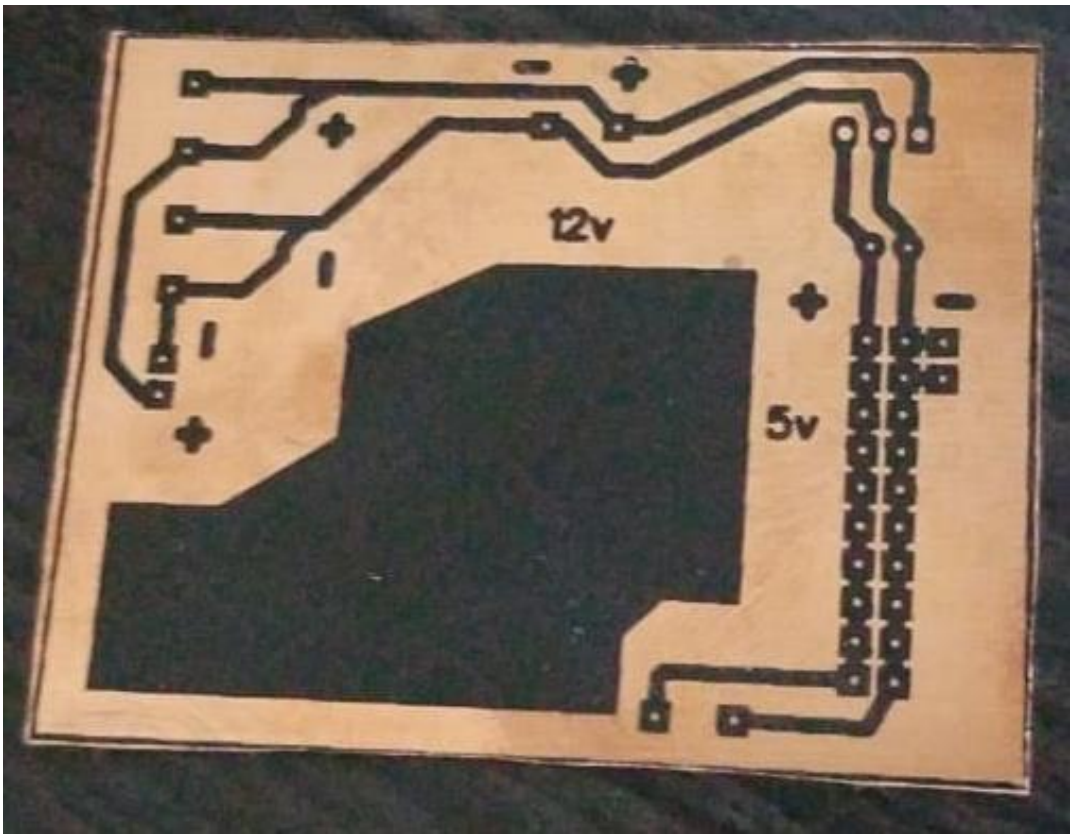
Fuente: Autoría propia basada en Simulador EasyEDA, 2021

En el editor PCB de EasyEDA, se colocan los espacios para los pines que une los cables de alimentación de cada componente, teniendo en cuenta la parte de 12-11.1v y la parte regulada a 5v.

Terminando el diseño y verificando las conexiones, se genera una imagen png.

El proceso que se usa para pasar el circuito a una placa de cobre, es la serigrafía, de esa forma se obtiene una mejor presentación en el circuito, evitando problemas que normalmente ocurren al utilizar el método de planchado.

Figura 4.1.10 Circuito en cobre.

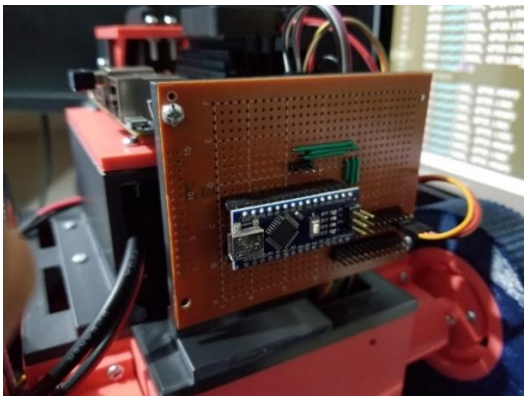


Fuente: Autoría propia, 2021

4.1.5 Diseño de circuito para Arduino

Para realizar el circuito de Arduino, se hace uso del modelo nano y con una placa pre perforada solo se sueldan pines que vayan a las entradas y salidas de Arduino.

Figura 4.1.11 Circuito Arduino.

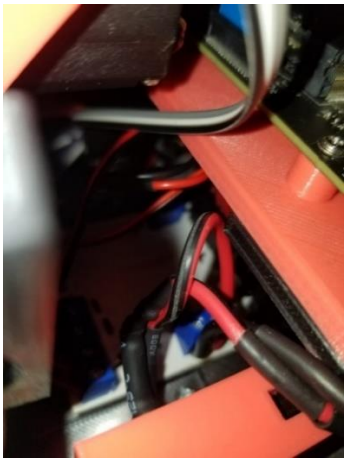


Fuente: Autoría propia, 2021

Todas las salidas necesarias en la placa de alimentación, se conectan mediante cables puenteados directamente a la placa en la que se encuentra el Arduino, como se muestra en la imagen. En cuanto al microcontrolador, este se encuentra fijado por unos pines hembra, soldados a la placa.

4.1.6 Conexiones físicas

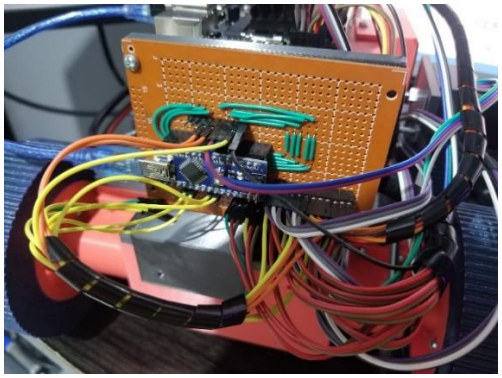
Figura 4.1.12 Placa de alimentación.



Fuente: Autoría propia, 2021

La placa de alimentación se encuentra entre la base del cabezal y la carcasa de la batería, en ella están soldadas los pines para los cables de alimentación, para el ventilador y 4 clemas tanto para la batería, como para los puentes H y el botón principal de encendido, ya que cuentan con cables de mayor calibre.

Figura 4.1.13 Conexiones a Arduino.

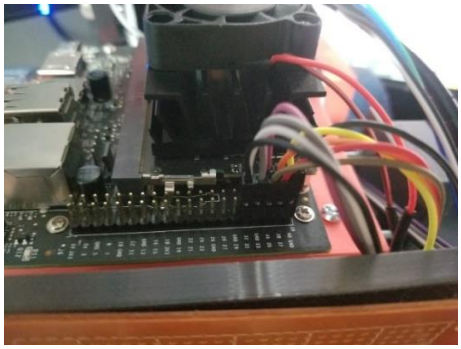


Fuente: Autoría propia, 2021

Los cables de alimentación están cubiertos para evitar que se jalen o se unan con los demás cables de los sensores y los servomotores. Los componentes se conectan directo a los pines en la parte inferior derecha, distribuidos de la siguiente manera:

- GND
- 5v
- Señal

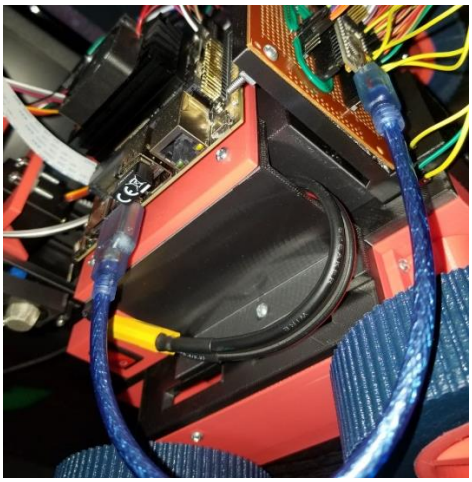
Figura 4.1.14 Conexiones a Jetson Nano.



Fuente: Autoría propia, 2021

Los pines de los puentes H se conectan directo a la placa Jetson Nano.

Figura 4.1.15 Conexión por USB.



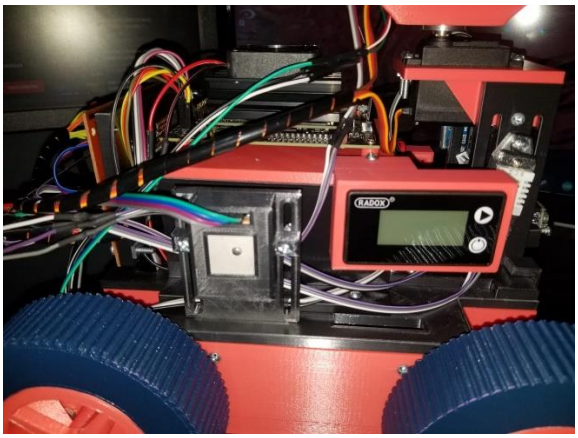
Fuente: Autoría propia, 2021

Arduino se conecta por su puerto principal, mediante el cable de datos que tiene por defecto, hacia la placa principal.

4.1.7 Implementación de la electrónica en el diseño mecánico

A continuación se muestra la implementación completa de los sensores, actuadores, y placas de desarrollo al diseño mecánico.

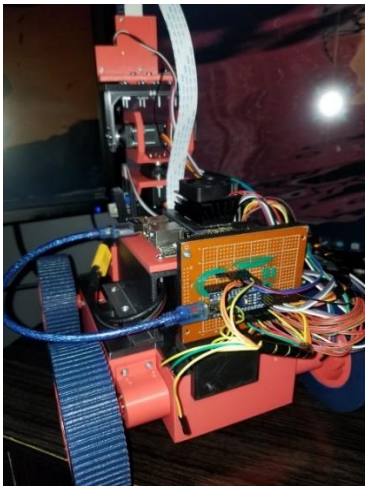
Figura 4.1.16 Robot vista derecha.



Fuente: Autoría propia, 2021

En esta parte se encuentra el GPS y la pantalla que indica el porcentaje se carga, y en el soporte del cabezal el sensor de humedad y temperatura.

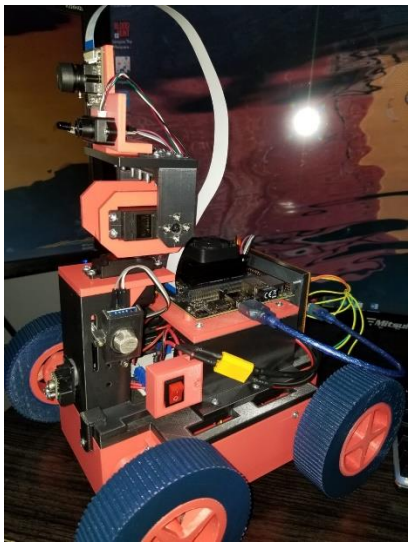
Figura 4.1.17 Robot vista trasera.



Fuente: Autoría propia, 2021

Por la parte trasera del robot se encuentra la placa de Arduino con sus respectivas conexiones.

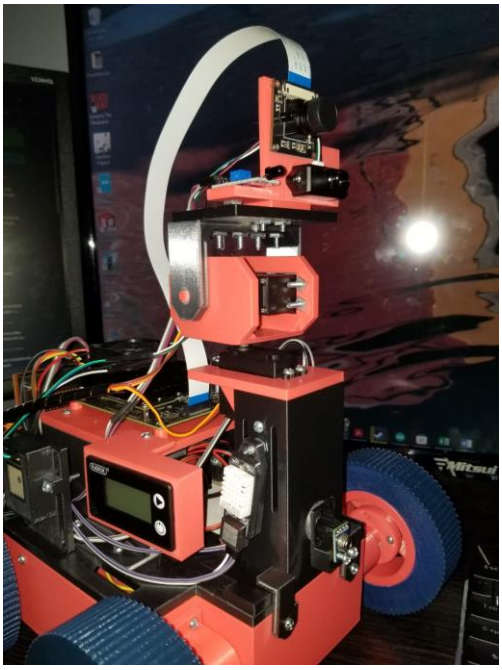
Figura 4.1.18 Robot vista izquierda.



Fuente: Autoría propia, 2021

Aquí se puede notar el sensor de gas, el botón de encendido y apagado, de igual manera se ve la placa Jetson sobre su base y un ventilador para disipar el calor.

Figura 4.1.19 Robot vista semifrontal.



Fuente: Autoría propia, 2021

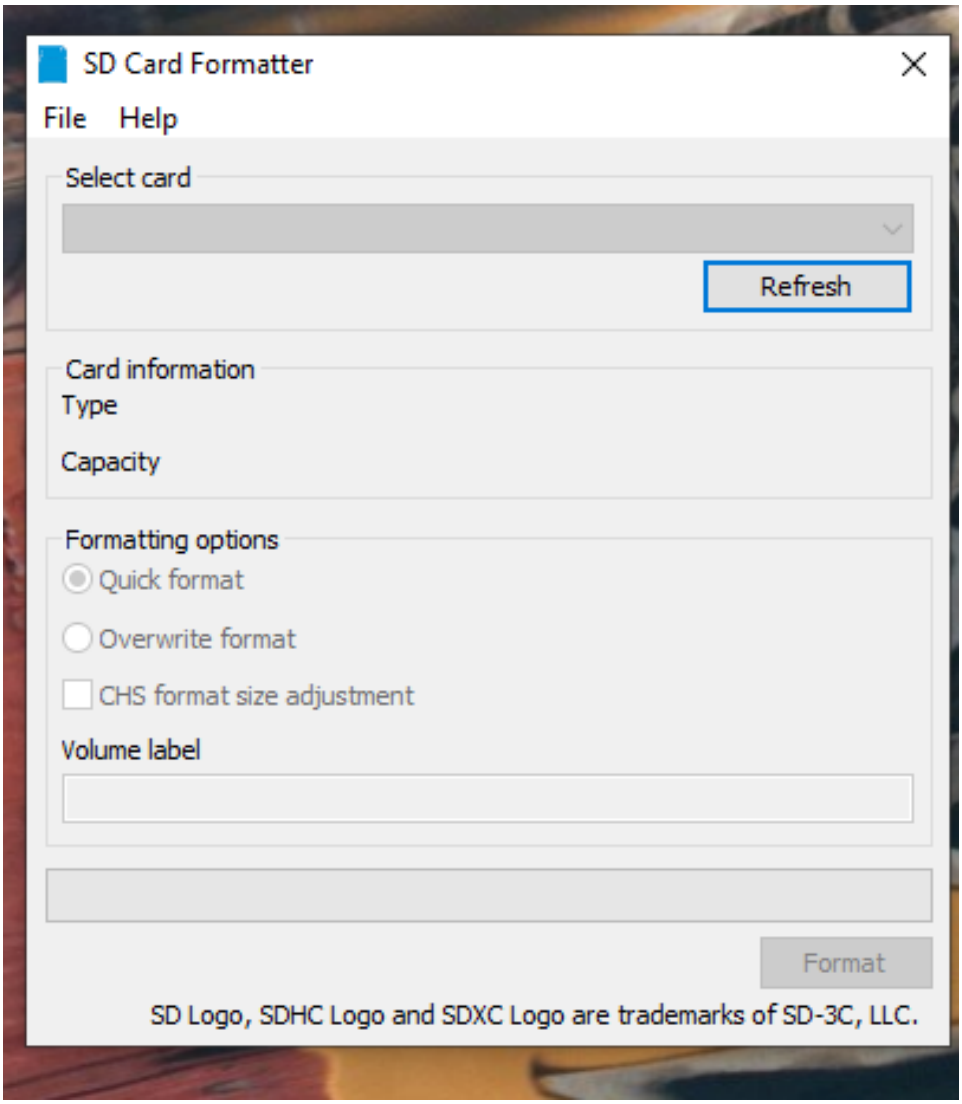
En la figura 4.1.19 se muestran los sensores de temperatura en la parte inferior, la cámara, el sensor de fuego, el sensor Lidar y los servomotores.

4.1.8 Descarga e instalación de Jetpack 4.6

Para que la placa Jetson Nano funcione, se debe instalar el sistema operativo de NVIDIA que se basa en Linux, más específicamente en Ubuntu. NVIDIA ocupa un sistema operativo llamado Jetpack, a día de hoy se encuentra en la versión 4.6.

Para un correcto rendimiento del sistema, se usa una tarjeta SD de 64GB.

Figura 4.1.20 SD Card Formater.

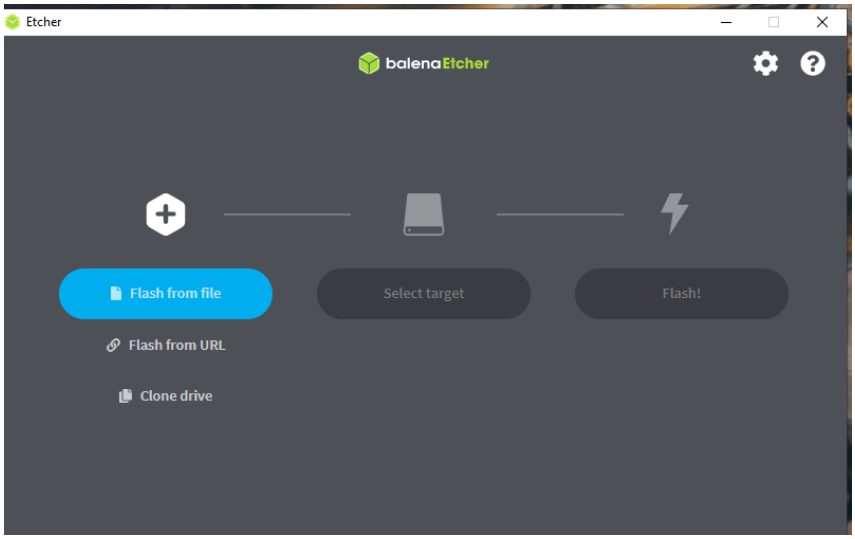


Fuente: Autoría propia basada en SD Card Formatter, 2021

Primero se debe formatear la tarjeta, la página recomienda usar dos programas, una para formatear (SD Card Formatter) y otro para flashear el sistema operativo en la tarjeta SD.

Solo se selecciona la unidad extraíble, y se selecciona Quick format, no se recomienda colocar un nombre al volumen.

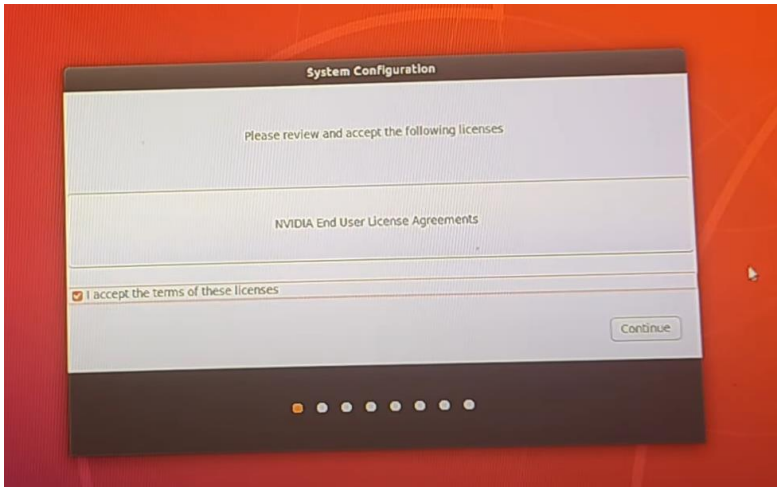
Figura 4.1.21 Balena.



Fuente: Autoría propia basada en Balena, 2021

Balena es el segundo programa recomendado en la página oficial. Cuando el sistema operativo se descarga, se selecciona el archivo en formato .zip para flashear la memoria, después se selecciona el disco con el cual se va trabajar y escribir el sistema operativo, por último se da clic en Flash y se espera hasta que el proceso se termine.

Figura 4.1.22 Pantalla inicial Jetpack.



Fuente: Autoría propia basada en Sistema Operativo Jetpack, 2021

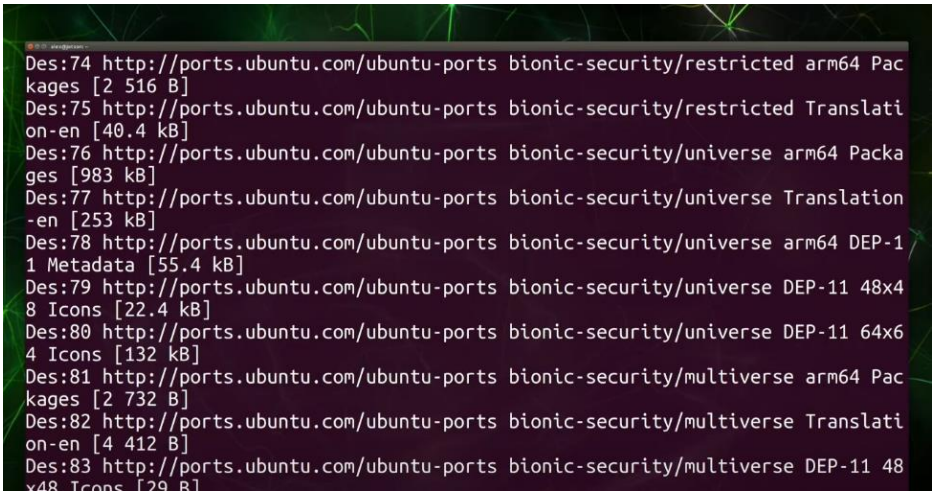
Para terminar con la instalación se debe llenar cada paso solicitado por sistema. Una vez que se termina el llenado de datos, la instalación está completa.

4.1.9 Configuración del sistema en Jetson Nano

Cuando se instala un sistema operativo, siempre es recomendable actualizar paqueterías del sistema, los comandos en Linux son los siguientes:

- sudo apt-get update
- sudo apt-get upgrade

Figura 4.1.23 Consola.



Fuente: Autoría propia basada en Sistema Operativo Jetpack, 2021

Al finalizar las actualizaciones, se verifica la versión de Python:

- python3

La versión debe ser la 3.6.9, después se ejecuta import cv2, normalmente no debe aparecer ninguna advertencia en pantalla, de manera que de esa forma se confirma la correcta instalación.

4.1.10 Control Remoto (RDP)

Para tener acceso remoto de la placa, directamente a una computadora, se necesita instalar los paquetes del protocolo de comunicación xrdp, con los siguientes comandos:

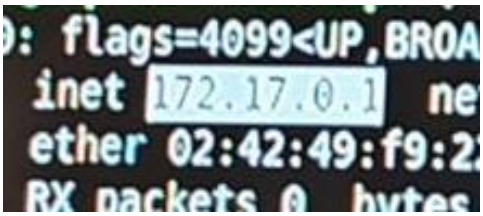
- sudo apt install xrdp
- sudo systemctl enable –now xrdp

Por último se necesita conocer la dirección ip del dispositivo. Se puede obtener con el siguiente comando:

- ip address

La dirección se encuentra en la parte "inet":

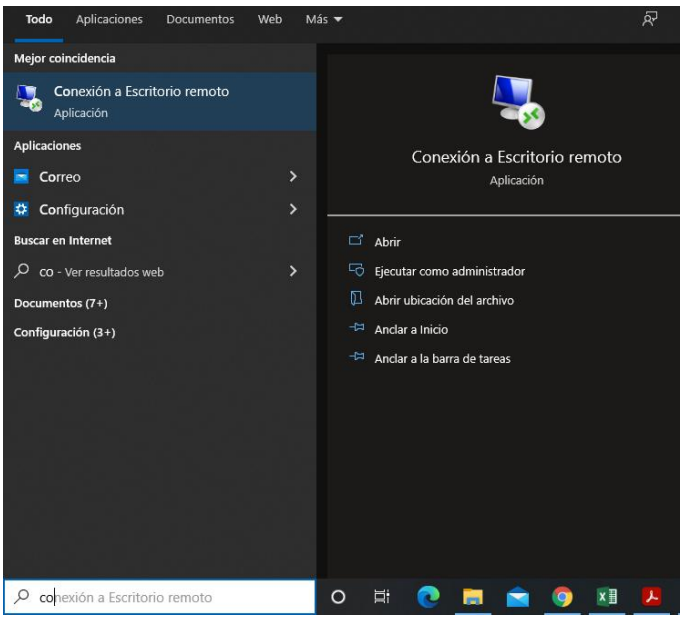
Figura 4.1.24 Dirección IP.



Fuente: Autoría propia basada en Sistema Operativo Jetpack, 2021

En la barra de búsqueda de Windows, se coloca lo siguiente:

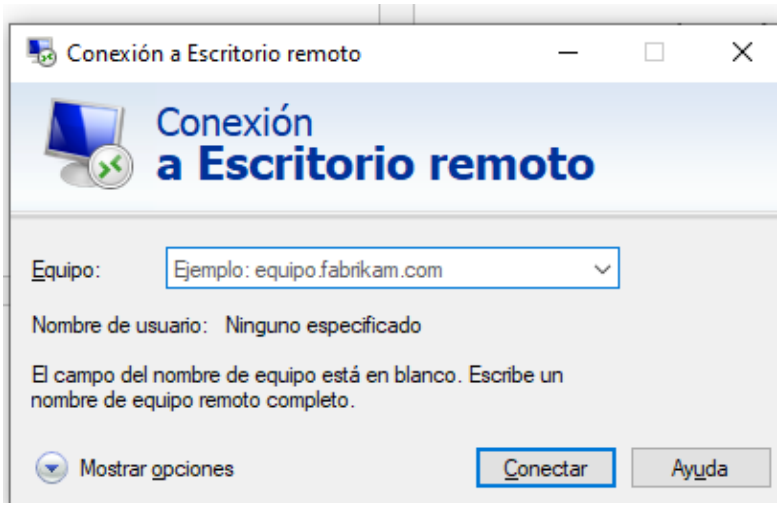
Figura 4.1.25 Aplicación de escritorio remoto.



Fuente: Autoría propia, 2021

Ahora, para poder tener el acceso de forma correcta y completa, se coloca la dirección ip en la parte que indica "Equipo".

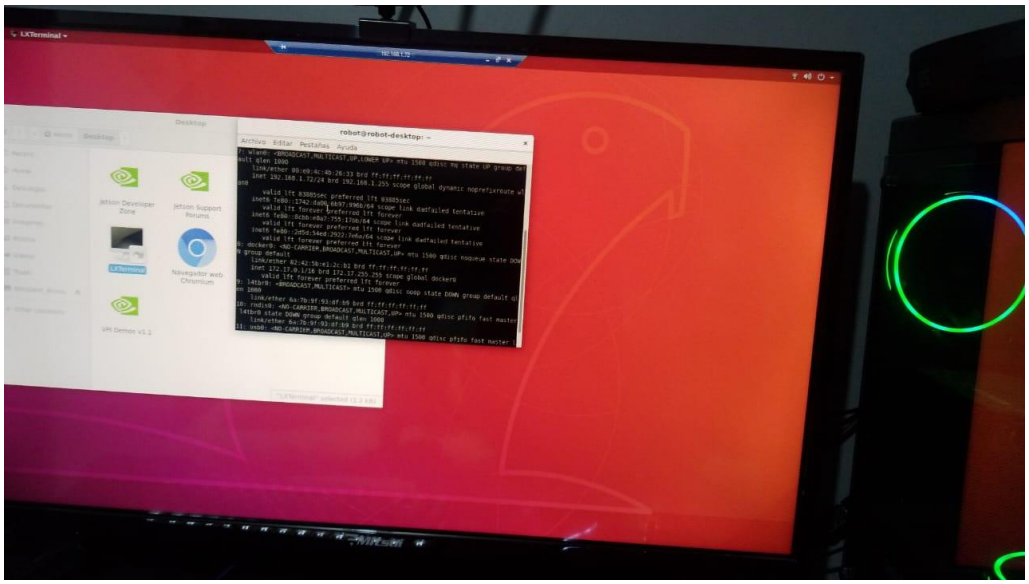
Figura 4.1.26 Menú de aplicación.



Fuente: Autoría propia, 2021

Al entrar, pedirá usuario y contraseña (Usuario y contraseña del sistema instalado en la placa Jetson Nano). Y ya se tendría el acceso remoto del robot.

Figura 4.1.27 Escritorio remoto.



Fuente: Autoría propia, 2021

4.1.11 Pruebas con Cámara

Como primer paso para realizar pruebas con la cámara, se debe clonar el repositorio de Jetson Inference, con el comando:

- git clone –recursive https://github.com/dusty-nv/jetson-inference

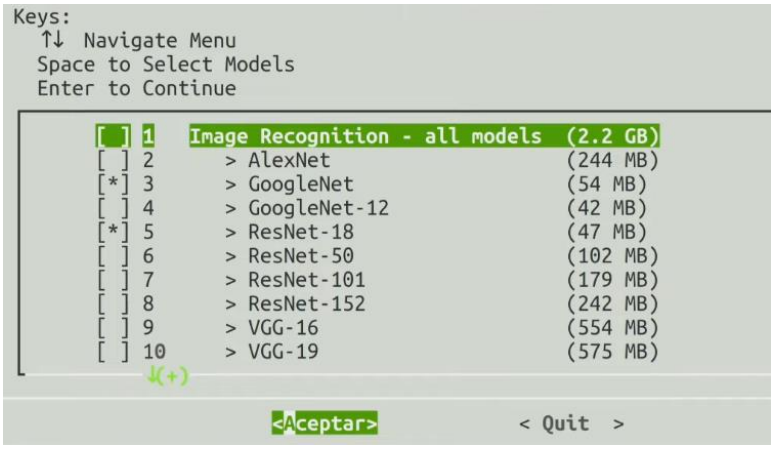
Al terminarse la descarga del repositorio, se procede a verificar si se descargó en la carpeta de destino con el comando "ls -l".

Como segundo paso, se debe entrar a la carpeta "jetson-inference" utilizando el comando "cd jetson-inference" y dentro de esa carpeta se ejecuta lo siguiente:

- docker/run.sh

Al esperar unos segundos, arroja una pantalla, la cual ayudará para definir que modelos descargar para visión computacional.

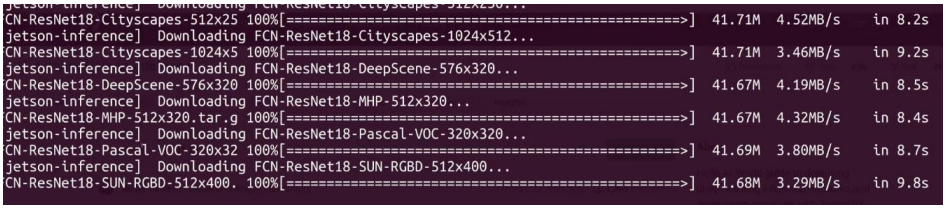
Figura 4.1.28 Instalador de modelos.



Fuente: Autoría propia basada en Repositorio Jetson Inference, 2021

Dar clic en aceptar, para generar la descarga de los modelos.

Figura 4.1.29 Proceso de descarga de modelos.



Fuente: Autoría propia basada en Repositorio Jetson Inference, 2021

Finalizando el proceso, se debe dirigir a la carpeta de /python/examples haciendo uso del comando cd. Cuando se tenga acceso a la carpeta, se ejecuta lo siguiente:

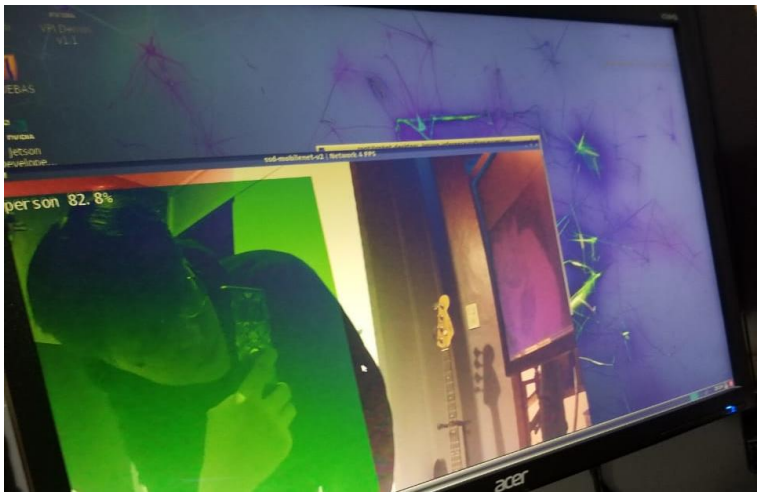
- python3 detectnet.py

O bien

- ./detectnet.py csi://0

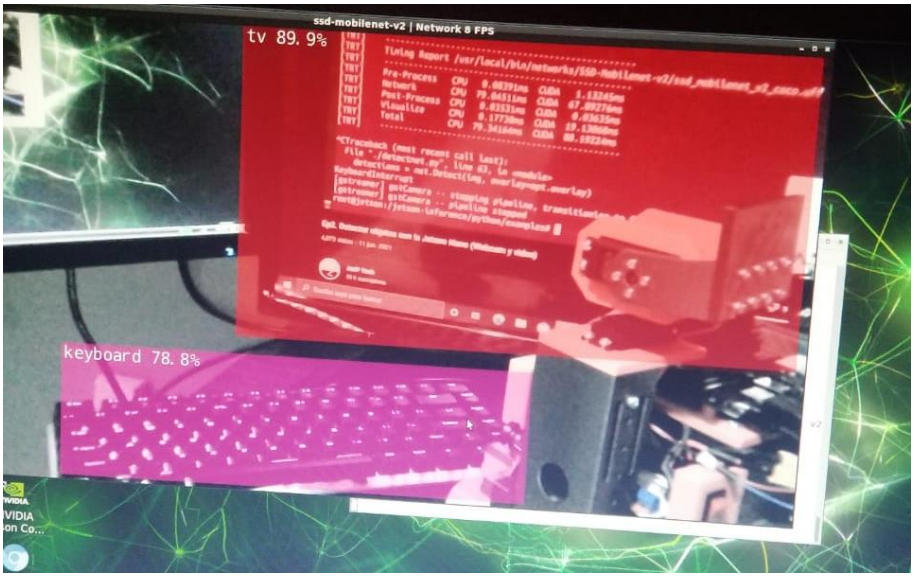
Si se cuenta con una cámara con conexión de cable tipo ffc se utiliza la configuración "csi://0" en caso contrario, si se utiliza una webcam se usa la configuración "/dev/video0". Esto nos generará una ventana con la visión de la cámara y la detección de objetos en tiempo real.

Figura 4.1.30 Prueba 1 de reconocimiento con cámara.



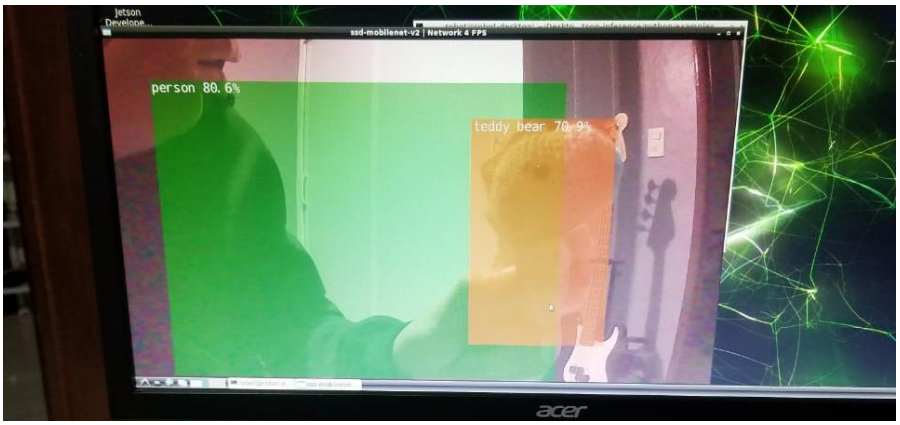
Fuente: Autoría propia basada en Repositorio Jetson Inference, 2021

Figura 4.1.31 Prueba 2 de reconocimiento con cámara.



Fuente: Autoría propia basada en Repositorio Jetson Inference, 2021

Figura 4.1.32 Prueba 3 de reconocimiento con cámara.



Fuente: Autoría propia basada en Repositorio Jetson Inference, 2021

Si bien, se logra generar la salida de imagen en el monitor, resultó un problema, puesto que cada vez que se inicia la cámara, la placa Jetson se apaga. Se verificó en distintos foros oficiales de NVIDIA para solucionar el problema, y a pesar de la ardua búsqueda y experimentación no se encontró nada que pudiera solucionar el error.

4.1.12 Instalación del IDE de Arduino en Jetpack

Para obtener el software de Arduino e instalarlo en la placa Jetson Nano, se direcciona a la página oficial de Arduino en el apartado "Software", se elige la opción "Linux ARM 64 bits" y se descargará.

Figura 4.1.33 Instalación de IDE de Arduino en Jetpack.

Fuente: Autoría propia, 2021

Se da clic dos veces en el archivo descargado y se ejecuta la terminal en la carpeta "arduino-1.8.16-linux64" y se ejecuta el siguiente comando:

- sudo chmod –R 777 arduino-1.8.16/

Esto con la finalidad de otorgarle permisos a la carpeta contenedora del paquete de instalación.

Una vez realizado, se entra a la carpeta arduino-1.8.16 y se ejecuta:

- sudo ./install.sh

La instalación suele durar pocos segundos.

4.1.13 Código de Arduino

Debido a algunos problemas en la programación con la comunicación, se maneja de forma separada y no todo en conjunto la parte del movimiento y la lectura de sensores, quedando como códigos de prueba.

Para el movimiento, a la placa de Arduino se le sube el programa de "StandarFirmata" que viene por defecto en los ejemplos del software.

La comunicación de los sensores se conforma de un código creado desde cero.

```
#include<DHT.h>
#include<Wire.h>
#include<Adafruit_MLX90614.h>

Adafruit_MLX90614 mlx = Adafruit_MLX90614();

DHT dht(2,DHT22);
float Temp, Humd,TC,MQ;

void setup() {
  Serial.begin(9600);
  dht.begin();
  mlx.begin();
}

void loop() {
  Temp=dht.readTemperature();
  Humd=dht.readHumidity();
  MQ=analogRead(A0);
  TC=mlx.readObjectTempC();

  Serial.print(TC);
  Serial.print(" - ");
  Serial.print(Temp);
  Serial.print(" - ");
  Serial.print(Humd);
  Serial.print(" - ");
  Serial.println(MQ);

  delay(200);
}
```

La mayoría de sensores cuenta con librerías para su uso fácil, de esta forma solo se inicializan, se lee la señal que llega a cada pin y se mandan a imprimir por el puerto serial.

4.1.14 Programación en Python

Por parte del entorno de Python, se manejan de igual forma dos códigos tanto para el control del movimiento como para la lectura de sensores.

```
#!/usr/bin/env python

import RPi.GPIO as GPIO
import time
```

```
from tkinter import *
import tkinter
from time import sleep
from pyfirmata import Arduino, util, SERVO
import random


IN1 = 35
IN2 = 37
IN3 = 38
IN4 = 40
IN1D = 36
IN2D = 33
IN3D = 32
IN4D = 31

# set pin numbers to the board's
GPIO.setmode(GPIO.BOARD)

# initialize In1, In2, In3 and In4
GPIO.setup(IN1, GPIO.OUT, initial=GPIO.LOW)
GPIO.setup(IN2, GPIO.OUT, initial=GPIO.LOW)
GPIO.setup(IN3, GPIO.OUT, initial=GPIO.LOW)
GPIO.setup(IN4, GPIO.OUT, initial=GPIO.LOW)
GPIO.setup(IN1D, GPIO.OUT, initial=GPIO.LOW)
GPIO.setup(IN2D, GPIO.OUT, initial=GPIO.LOW)
GPIO.setup(IN3D, GPIO.OUT, initial=GPIO.LOW)
GPIO.setup(IN4D, GPIO.OUT, initial=GPIO.LOW)

board = Arduino('/dev/ttyUSB0')
sleep(1)
board.digital[5].mode = SERVO
board.digital[3].mode = SERVO

def servo(posiciones):
        board.digital[5].write(posiciones)

def servo2(posiciones2):
        board.digital[3].write(posiciones2)

root = Tk()
root.title("CONTROL ROBOT")
root.minsize(1280,768)
```

El código se compone de importación de las librerías necesarias, la declaración de pines a los puentes H, e inicializando las salidas.

Se coloca que hay una comunicación con Arduino mediante un puerto en la placa, y se asignan los pines conectados a los servomotores, se definen dos funciones para cada servo y se genera una ventana con Tkinter.

```
angulo = Scale(root,
                                command = servo,
                                from_=180,
                                to = 0,
                                orient = HORIZONTAL,
                                length = 300,
                                troughcolor = 'black',
                                width = 50,
                                cursor = 'dot',
                                label = 'Servo (LEFT - RIGHT)')
angulo.place(x=10, y=450)

angulo = Scale(root,
                                command = servo2,
                                from_=120,
                                to = 0,
                                orient = HORIZONTAL,
                                length = 300,
                                troughcolor = 'black',
                                width = 50,
                                cursor = 'dot',
                                label = 'Servo (DOWN - UP)')
angulo.place(x=10, y=550)
```

En esta parte, dentro de la ventana de la interfaz se generan dos barras moviles, las cuales permiten el movimiento continuo de cada servo individualmente.

```
def adelante():
        GPIO.output(IN1, GPIO.LOW)
        GPIO.output(IN2, GPIO.LOW)
        GPIO.output(IN3, GPIO.LOW)
        GPIO.output(IN4, GPIO.LOW)
        GPIO.output(IN1D, GPIO.LOW)
        GPIO.output(IN2D, GPIO.LOW)
        GPIO.output(IN3D, GPIO.LOW)
        GPIO.output(IN4D, GPIO.LOW)
        time.sleep(0.1)
        # Forward
        GPIO.output(IN1, GPIO.HIGH)
        GPIO.output(IN2, GPIO.LOW)
        GPIO.output(IN3, GPIO.HIGH)
        GPIO.output(IN4, GPIO.LOW)
        GPIO.output(IN1D, GPIO.LOW)
        GPIO.output(IN2D, GPIO.HIGH)
```

```
        GPIO.output(IN3D, GPIO.LOW)
        GPIO.output(IN4D, GPIO.HIGH)

def atras():
        GPIO.output(IN1, GPIO.LOW)
        GPIO.output(IN2, GPIO.LOW)
        GPIO.output(IN3, GPIO.LOW)
        GPIO.output(IN4, GPIO.LOW)
        GPIO.output(IN1D, GPIO.LOW)
        GPIO.output(IN2D, GPIO.LOW)
        GPIO.output(IN3D, GPIO.LOW)
        GPIO.output(IN4D, GPIO.LOW)
        time.sleep(0.1)
        # Backward
        GPIO.output(IN1, GPIO.LOW)
        GPIO.output(IN2, GPIO.HIGH)
        GPIO.output(IN3, GPIO.LOW)
        GPIO.output(IN4, GPIO.HIGH)
        GPIO.output(IN1D, GPIO.HIGH)
        GPIO.output(IN2D, GPIO.LOW)
        GPIO.output(IN3D, GPIO.HIGH)
        GPIO.output(IN4D, GPIO.LOW)

def izquierda():
        # Left
        GPIO.output(IN1, GPIO.LOW)
        GPIO.output(IN2, GPIO.HIGH)
        GPIO.output(IN3, GPIO.HIGH)
        GPIO.output(IN4, GPIO.LOW)
        GPIO.output(IN1D, GPIO.LOW)
        GPIO.output(IN2D, GPIO.HIGH)
        GPIO.output(IN3D, GPIO.HIGH)
        GPIO.output(IN4D, GPIO.LOW)
        time.sleep(0.1)

def derecha():
        # Right
        GPIO.output(IN1, GPIO.HIGH)
        GPIO.output(IN2, GPIO.LOW)
        GPIO.output(IN3, GPIO.LOW)
        GPIO.output(IN4, GPIO.HIGH)
        GPIO.output(IN1D, GPIO.HIGH)
        GPIO.output(IN2D, GPIO.LOW)
        GPIO.output(IN3D, GPIO.LOW)
        GPIO.output(IN4D, GPIO.HIGH)
        time.sleep(0.1)

def stop():
        # Stop
        GPIO.output(IN1, GPIO.LOW)
        GPIO.output(IN2, GPIO.LOW)
        GPIO.output(IN3, GPIO.LOW)
        GPIO.output(IN4, GPIO.LOW)
```

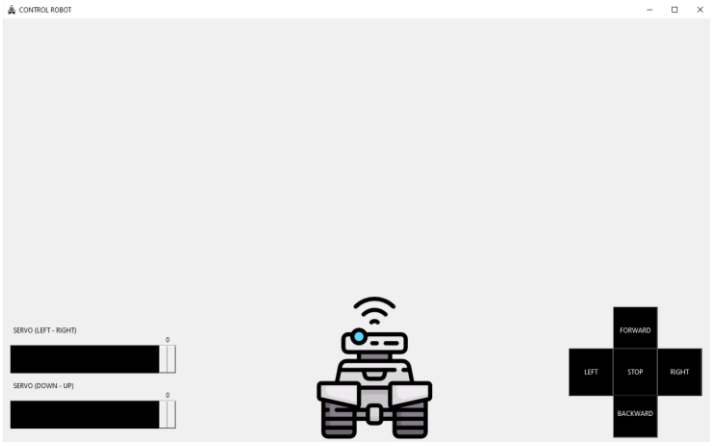
```
        GPIO.output(IN1D, GPIO.LOW)
        GPIO.output(IN2D, GPIO.LOW)
        GPIO.output(IN3D, GPIO.LOW)
        GPIO.output(IN4D, GPIO.LOW)
        time.sleep(0.1)
```

El código anterior muestra las funciones que posteriormente cada botón de la interfaz ejecuta para el avance, retroceso y giro del robot.

```
boton1 = Button(root, text="FORWARD", command=adelante, width = 10, height
= 5)
boton2 = Button(root, text="BACK", command=atras, width = 10, height = 5)
boton3 = Button(root, text="LEFT", command=izquierda, width = 10, height =
5)
boton4 = Button(root, text="RIGHT", command=derecha, width = 10, height =
5)
boton5 = Button(root, text="STOP", command=stop, width = 10, height = 5)

boton1.place(x="1080", y="410")
boton2.place(x="1080", y="560")
boton3.place(x="1000", y="485")
boton4.place(x="1160", y="485")
boton5.place(x="1080", y="485")

boton1.config(bg="black")
boton2.config(bg="black")
boton3.config(bg="black")
boton4.config(bg="black")
boton5.config(bg="black")

boton1.config(fg="white")
boton2.config(fg="white")
boton3.config(fg="white")
boton4.config(fg="white")
boton5.config(fg="white")

root.mainloop()
```

Se generan los botones para indicar en pantalla al usuario el movimiento que se va a realizar con cada botón, y se configura las propiedades a cada uno, como los colores, el tamaño, la posición, el texto y lo que se va a ejecutar.

Figura 4.1.34 Interfaz de control.



Fuente: Autoría propia, 2021

En la imagen anterior, se puede mostrar la interfaz generada en Python.

```
import serial, time

arduino = serial.Serial("COM7", 9600)
time.sleep(2)
a = []

while True:
    dato = arduino.readline()
    for Word in dato.split():
        try:
            a.append(float(Word))
        except ValueError:
            pass
    TC = a[0]
    Tem = a[1]
    Hm=a[2]
    Gas=a[3]

    print("Obj C°: ", TC, "///Ambiente °C: ", Tem, "///Humedad %: ", Hm,
"///Gas: ", Gas)
    a = []
```

El código de los sensores genera una impresión de los valores de forma continua, por lo tanto, si uno de ellos capta un valor diferente, se verá reflejado en tiempo real al momento de imprimir lo que detecta un sensor.

Capítulo V

CONCLUSIONES

5.1 Conclusiones del proyecto

El presente proyecto logró permitir realizar los siguientes puntos:

- Prototipo físico con conexiones de sensores, actuadores y placas de desarrollo comunicadas por puerto serial.
- Conexión remota del robot desde un ordenar con sistema operativo Windows.
- Control de movimiento por interfaz.
- Envío y lectura de datos emitidos por los sensores.
- Detección de objetos en tiempo real.

5.2 Conclusiones relativas a los objetivos específicos

En cuanto a los objetivos específicos, si bien, en algunos puntos no se pudieron lograr los puntos establecidos al 100%, la base de lo que es el robot queda un poco más clara y de cierta forma establecida para futuras mejoras en cuanto al rendimiento, uso e inclusión de más funcionalidades.

Distintos factores, en especial el tiempo, se prioriza como uno de los problemas más significativos que se tuvieron para lograr los objetivos. Finalmente, en cuanto a la composición electrónica se evaluaron los sensores con buen rendimiento, fáciles de conseguir, y que estuvieran dentro del presupuesto establecido.

5.3 Conclusiones relativas al objetivo general

Como se mencionó anteriormente, algunos puntos no se cumplieron de manera satisfactoria, puesto que debido a algunos errores que fueron surgieron impidieron el avance completo del objetivo. Por otro lado, se pretende seguir trabajando en este prototipo para poder dar una solución a los problemas, si bien, el robot no fue diseñado para trabajar en un entorno real, puede tener gran potencial para mejorar versiones futuras, tanto en el aspecto de control que concierne este documento, como también la parte mecánica.

5.4 Aportaciones originales

En este proyecto, incorpora la idea de conjuntar un sistema relativamente nuevo de reconocimiento de objetos con la ayuda de la placa de desarrollo de NVIDIA (Jetson Nano) con monitorización de distintos sensores, teniendo un control remoto desde cualquier computadora mediante protocolo de escrito remoto (RDP).

5.5 Trabajos a futuro

Tomando en cuenta los problemas que surgieron a lo largo del proyecto, a corto o largo plazo se pueden implementar mejoras, e incluso otras alternativas para la conexión remota, ya que una opción viable al menos para el movimiento y el envío de datos que recolectan los sensores del robot, es agregar un módulo RF de larga distancia, con pantalla OLED y un buzzer integrados a un control físico. Para la parte de transmisión de video, los VTX suelen ser una buena opción, ya que son muy usados en drones, se arreglará el problema con la cámara, la cual provoca que la placa se reinicie el sistema.

Mejorar la organización del cableado, y agregar una pantalla al robot, para poder implementar otras funciones y facilitar el mantenimiento y/o programación de este.

5.6 Recomendaciones

Se recomienda manejar por separado el control del movimiento y la transmisión de video para un mejor rendimiento y visualización tanto de imagen, como de la lectura de sensores.

Usar una frecuencia de 5.8GHz para un VTX, es muy común y no presenta muchos problemas.

Capítulo VI

COMPETENCIAS DESARROLLADAS

6.1 Competencias desarrolladas o aplicadas

Las competencias desarrolladas en este proyecto fueron:

- Análisis e investigación de información
- Trabajo en equipo
- Toma de decisiones
- Autodidactismo
- Planificación

Competencias específicas:

- Diseño de circuitos
- Desarrollo de código
- Manejo básico de sistemas operativos basados en Linux

Capítulo VII

FUENTES DE INFORMACIÓN

7.1 Fuentes de información

- Carroll, E. (21 de Mayo de 2019). RDP Stands for "Really DO Patch!"- Understanding the Wormable RDP Vulnerability CVE-2019-0708. Obtenido de McAfee: https://www.mcafee.com/blogs/other-blogs/mcafee-labs/rdp-stands-for-really-do-patch-understanding-the-wormable-rdp-vulnerability-cve-2019-0708/
- Joyanes Aguilar, L., García, J. L., & Sánchez, C. (2008). Fundamentos de programación: Algoritmos, estructura de datos y objetos. España: McGraw-Hill.
- Jugunaru Mathieu, M. (2014). Introducción a la programación. Grupo Editorial Patria.
- Madridiario. (22 de Julio de 2019). Baterías 18650: Características, usos y ventajas. Obtenido de https://www.madridiario.es/noticia/470346/recomendamos/baterias-18650:-caracteristicas-usos-y-ventajas.html
- Bermudez, G. B. (2001, 31 diciembre). ROBOTS MÓVILES. ROBOTS MÓVILES. Recuperado 4 de diciembre de 2021, de https://core.ac.uk/download/pdf/229164924.pdf
- Vital Signs (Body Temperature, Pulse Rate, Respiration Rate, Blood Pressure) - Health Encyclopedia - University of Rochester Medical Center. (s. f.). Urmc.Rochester.Edu. Recuperado 4 de diciembre de 2021, de https://www.urmc.rochester.edu/encyclopedia/content.aspx?ContentTypeID=85&ContentID=P03963
- Todo sobre las placas de desarrollo en 2020. (2020, 11 abril). Placas de Desarrollo. Recuperado 4 de diciembre de 2021, de https://www.placasdedesarrollo.com/
- Jetson Nano Developer Kit. (2021, 14 abril). NVIDIA Developer. Recuperado 4 de diciembre de 2021, de https://developer.nvidia.com/embedded/jetson-nano-developer-kit

- J. (2016, 18 diciembre). Sensores y Actuadores. Aprendiendo Arduino. Recuperado 4 de diciembre de 2021, de https://aprendiendoarduino.wordpress.com/2016/12/18/sensores-y-actuadores/
- Lithium Battery Management System (BMS) –. (2020, 22 enero). PowerTech Systems. Recuperado 4 de diciembre de 2021, de https://www.powertechsystems.eu/home/products/others/battery-management-system-bms/
- I. (2019, 11 octubre). DHT22: el sensor de temperatura y humedad de precisión. Hardware libre. Recuperado 4 de diciembre de 2021, de https://www.hwlibre.com/dht22/
- P. (2019, 7 marzo). TFmini con Arduino (sensor de distancia). HETPRO. Recuperado 4 de diciembre de 2021, de https://hetpro-store.com/TUTORIALES/tfmini-con-arduino/
- TFmini-i LiDAR Laser Range Sensor (12m) Wiki - DFRobot. (s. f.). DFROBOT. Recuperado 4 de diciembre de 2021, de https://wiki.dfrobot.com/TFmini_i_LiDAR_Laser_Range_Sensor_12m_SKU_SEN0505?gclid=Cj0KCQiAkZKNBhDiARIsAPsk0WjuRB3hmEW_FgEVOQGX_E7HzVtLjzWuuoIsdXf5D2aNq6s2fdxmM6gaAq0TEALw_wcB
- L. (2021, 8 junio). Detector de llama con Arduino y sensor infrarrojo. Luis Llamas. Recuperado 4 de diciembre de 2021, de https://www.luisllamas.es/detector-llama-arduino/
- S. (2018, 7 marzo). sensor cardiaco arduino –. Soloelectronicos.com. Recuperado 4 de diciembre de 2021, de https://soloelectronicos.com/tag/sensor-cardiaco-arduino/
- Hernández, L. D. V. (2020, 16 junio). Termómetro infrarrojo con Arduino MLX90614. Programar fácil con Arduino. Recuperado 4 de diciembre de 2021, de https://programarfacil.com/blog/arduino-blog/termometro-infrarrojo-con-arduino-mlx90614/

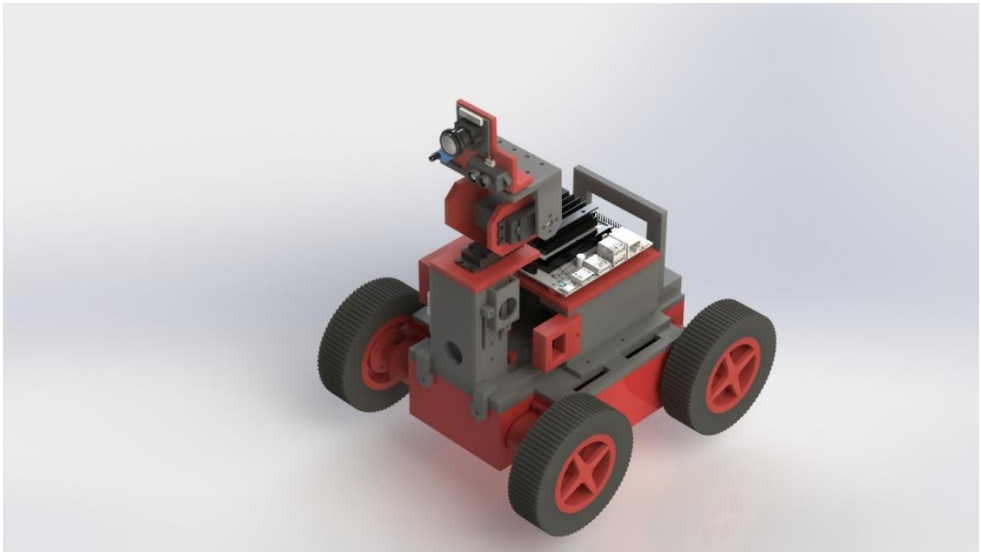
- Yañez, C. Y. (2021, 26 julio). ¿Qué es y como funciona Arduino? | CEAC. CEAC. Recuperado 4 de diciembre de 2021, de https://www.ceac.es/blog/que-es-y-como-funciona-arduino
- Visus, A. (2020, 15 octubre). ¿Para qué sirve Python? Razones para utilizar este lenguaje de programación. esic. Recuperado 4 de diciembre de 2021, de https://www.esic.edu/rethink/tecnologia/para-que-sirve-python
- Arduino - Aprender a desarrollar para crear objetos inteligentes - Comunicación Serial | Editiones ENI. (s. f.). Editions ENI. Recuperado 4 de diciembre de 2021, de https://www.ediciones-eni.com/open/mediabook.aspx?idR=256d9467932d6a4b20f41c2bbe0f9b06
- D. (2018, 24 mayo). Real examples of 3s2p 18650 battery. Lithium Ion Battery Manufacturer and Supplier in China-DNK Power. Recuperado 4 de diciembre de 2021, de https://www.dnkpower.com/18650-battery-pack-calculator/real-examples-of-3s2p-18650-battery/

Capítulo VIII

ANEXOS

8.1 Anexos

Figura 8.1 Ensamble con componentes electrónicos.



Fuente: Autoría propia basada en CAD SolidWorks, 2021

Figura 8.2 Ensamble con componentes electrónicos parte 2.

Fuente: Autoría propia basada en CAD SolidWorks, 2021

Figura 8.3 Código de cámara en Python.

```python
import jetson.inference
import jetson.utils

net = jetson.inference.detectNet("ssd-mobilenet-v2", threshold=0.5)
camera = jetson.utils.videoSource("csi://0")      # '/dev/video0' for V4L2
display = jetson.utils.videoOutput() # 'my_video.mp4' for file

while True:
        img = camera.Capture()
        detections = net.Detect(img)
        display.Render(img)
        display.SetStatus("Object Detection | Network {:.0f} FPS".format(net.GetNetworkFPS()))
```

Fuente: Autoría propia basada en editor de texto Python, 2021

Figura 8.4 Código de pruebas básicas de movimiento.

MOTORES_ROBOT

```
int IN1=8;
int IN2=7;
int IN3=13;
int IN4=12;
int IN1D=4;
int IN2D=5;
int IN3D=9;
int IN4D=10;

const int b1=3;
const int b2=2;

int b1state=0;
int b2state=0;

void setup()
{
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);

  pinMode(IN1D, OUTPUT);
  pinMode(IN2D, OUTPUT);
  pinMode(IN3D, OUTPUT);
  pinMode(IN4D, OUTPUT);

  pinMode(b1, INPUT);
  pinMode(b2, INPUT);
}

void loop()
{
  if (digitalRead(b1)>b1state)
  {
    digitalWrite(IN1, HIGH);
    digitalWrite(IN2, LOW);
```

Fuente: Autoría propia basada en IDE Arduino, 2021

Figura 8.5 Código de pruebas básicas de movimiento parte 2.

MOTORES_ROBOT

```
void loop()
{
  if (digitalRead(b1)>b1state)
  {
    digitalWrite(IN1, HIGH);
    digitalWrite(IN2, LOW);
    digitalWrite(IN3, HIGH);
    digitalWrite(IN4, LOW);
    digitalWrite(IN1D, HIGH);
    digitalWrite(IN2D, LOW);
    digitalWrite(IN3D, HIGH);
    digitalWrite(IN4D, LOW);
  }
  else
  {
    delay(100);
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, LOW);
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, LOW);
    digitalWrite(IN1D, LOW);
    digitalWrite(IN2D, LOW);
    digitalWrite(IN3D, LOW);
    digitalWrite(IN4D, LOW);
  }
  if (digitalRead(b2)>b2state)
  {
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, HIGH);
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, HIGH);
    digitalWrite(IN1D, LOW);
    digitalWrite(IN2D, HIGH);
    digitalWrite(IN3D, LOW);
    digitalWrite(IN4D, HIGH);
  }
  else
```

Fuente: Autoría propia basada en IDE Arduino, 2021

Figura 8.6 Código de pruebas básicas de movimiento parte 3.

```
    }
    else
    {
      delay(100);
      digitalWrite(IN1, LOW);
      digitalWrite(IN2, LOW);
      digitalWrite(IN3, LOW);
      digitalWrite(IN4, LOW);
      digitalWrite(IN1D, LOW);
      digitalWrite(IN2D, LOW);
      digitalWrite(IN3D, LOW);
      digitalWrite(IN4D, LOW);
    }
  }
```

Fuente: Autoría propia basada en IDE Arduino, 2021

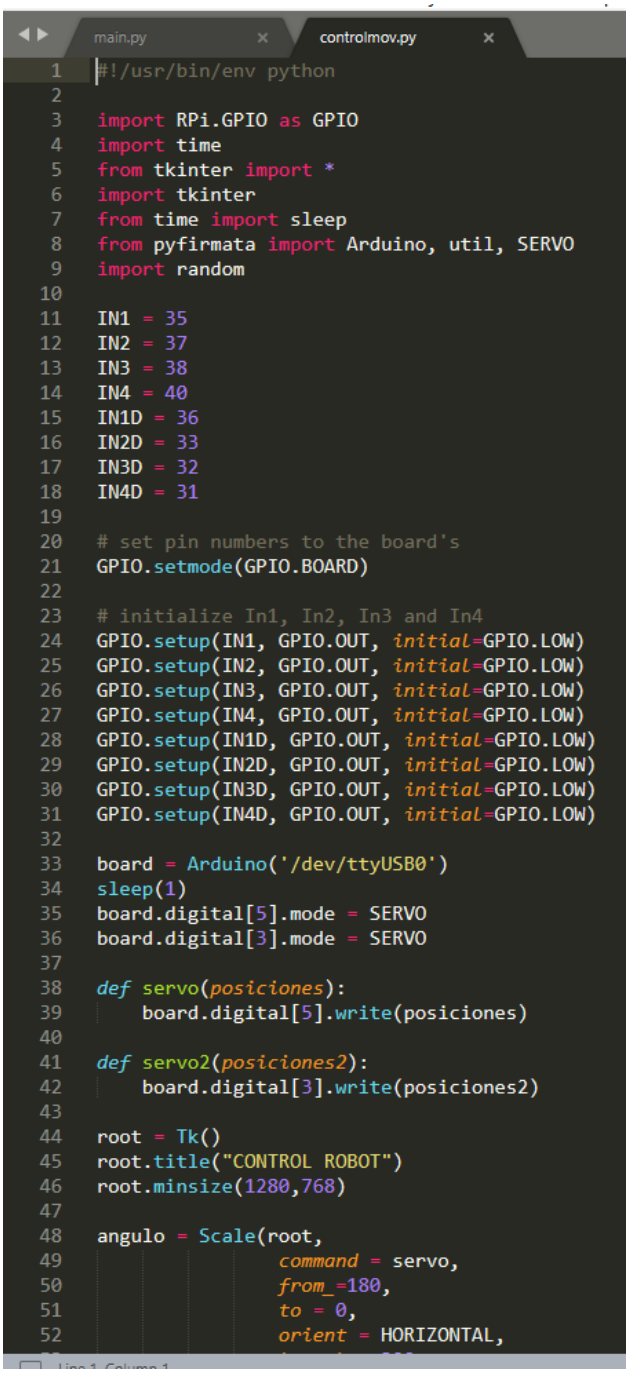
Figura 8.7 Firmata.

Fuente: Autoría propia basada en IDE Arduino, 2021

Figura 8.8 Código sensores.

Fuente: Autoría propia basada en IDE Arduino, 2021

Figura 8.9 Código control de movimiento.



```python
#!/usr/bin/env python

import RPi.GPIO as GPIO
import time
from tkinter import *
import tkinter
from time import sleep
from pyfirmata import Arduino, util, SERVO
import random

IN1 = 35
IN2 = 37
IN3 = 38
IN4 = 40
IN1D = 36
IN2D = 33
IN3D = 32
IN4D = 31

# set pin numbers to the board's
GPIO.setmode(GPIO.BOARD)

# initialize In1, In2, In3 and In4
GPIO.setup(IN1, GPIO.OUT, initial=GPIO.LOW)
GPIO.setup(IN2, GPIO.OUT, initial=GPIO.LOW)
GPIO.setup(IN3, GPIO.OUT, initial=GPIO.LOW)
GPIO.setup(IN4, GPIO.OUT, initial=GPIO.LOW)
GPIO.setup(IN1D, GPIO.OUT, initial=GPIO.LOW)
GPIO.setup(IN2D, GPIO.OUT, initial=GPIO.LOW)
GPIO.setup(IN3D, GPIO.OUT, initial=GPIO.LOW)
GPIO.setup(IN4D, GPIO.OUT, initial=GPIO.LOW)

board = Arduino('/dev/ttyUSB0')
sleep(1)
board.digital[5].mode = SERVO
board.digital[3].mode = SERVO

def servo(posiciones):
    board.digital[5].write(posiciones)

def servo2(posiciones2):
    board.digital[3].write(posiciones2)

root = Tk()
root.title("CONTROL ROBOT")
root.minsize(1280,768)

angulo = Scale(root,
               command = servo,
               from_=180,
               to = 0,
               orient = HORIZONTAL,
```

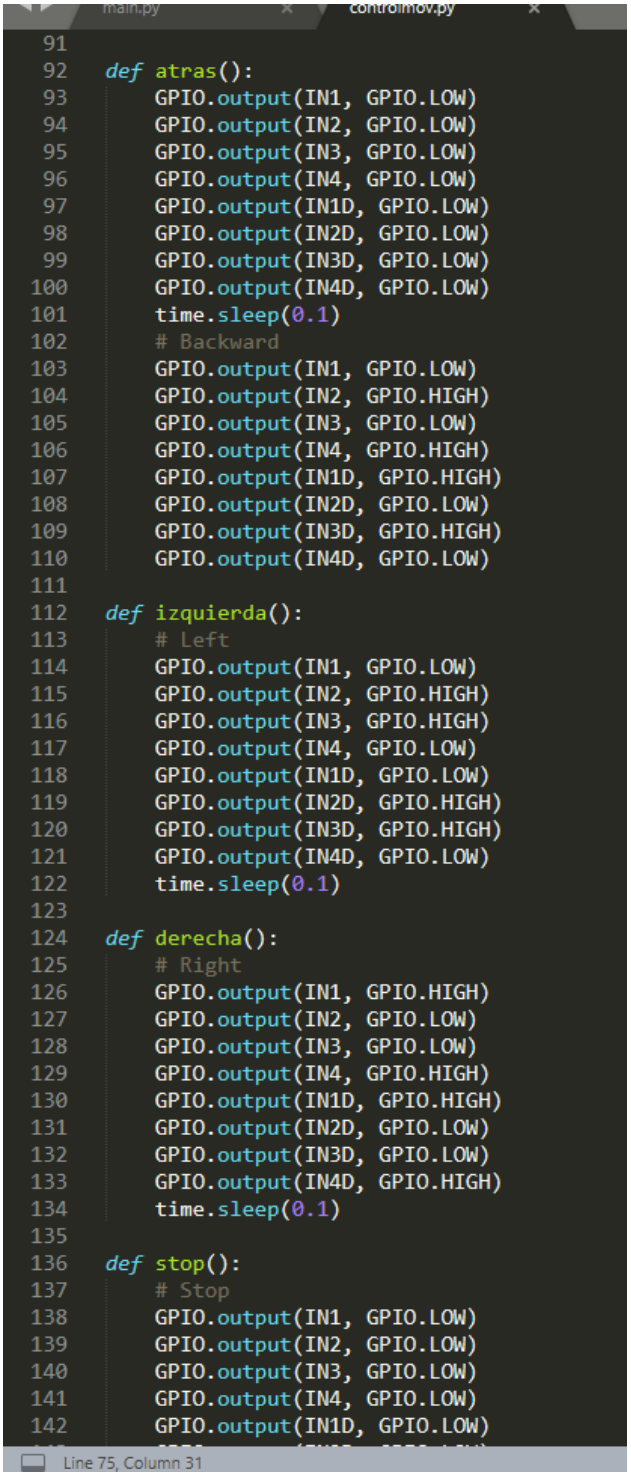
Fuente: Autoría propia basada en editor de texto Sublime Text 3, 2021

Figura 8.10 Código control de movimiento parte 2.

```
46  root.minsize(1280,768)
47
48  angulo = Scale(root,
49                 command = servo,
50                 from_=180,
51                 to = 0,
52                 orient = HORIZONTAL,
53                 length = 300,
54                 troughcolor = 'black',
55                 width = 50,
56                 cursor = 'dot',
57                 label = 'Servo (LEFT - RIGHT)')
58  angulo.place(x=10, y=450)
59
60  angulo = Scale(root,
61                 command = servo2,
62                 from_=120,
63                 to = 0,
64                 orient = HORIZONTAL,
65                 length = 300,
66                 troughcolor = 'black',
67                 width = 50,
68                 cursor = 'dot',
69                 label = 'Servo (DOWN - UP)')
70  angulo.place(x=10, y=550)
71
72  def adelante():
73      GPIO.output(IN1, GPIO.LOW)
74      GPIO.output(IN2, GPIO.LOW)
75      GPIO.output(IN3, GPIO.LOW)
76      GPIO.output(IN4, GPIO.LOW)
77      GPIO.output(IN1D, GPIO.LOW)
78      GPIO.output(IN2D, GPIO.LOW)
79      GPIO.output(IN3D, GPIO.LOW)
80      GPIO.output(IN4D, GPIO.LOW)
81      time.sleep(0.1)
82      # Forward
83      GPIO.output(IN1, GPIO.HIGH)
84      GPIO.output(IN2, GPIO.LOW)
85      GPIO.output(IN3, GPIO.HIGH)
86      GPIO.output(IN4, GPIO.LOW)
87      GPIO.output(IN1D, GPIO.LOW)
88      GPIO.output(IN2D, GPIO.HIGH)
89      GPIO.output(IN3D, GPIO.LOW)
90      GPIO.output(IN4D, GPIO.HIGH)
91
92  def atras():
93      GPIO.output(IN1, GPIO.LOW)
94      GPIO.output(IN2, GPIO.LOW)
95      GPIO.output(IN3, GPIO.LOW)
96      GPIO.output(IN4, GPIO.LOW)
97      GPIO.output(IN1D, GPIO.LOW)
```

Fuente: Autoría propia basada en editor de texto Sublime Text 3, 2021

Figura 8.11 Código control de movimiento parte 3.




```
91
92  def atras():
93      GPIO.output(IN1, GPIO.LOW)
94      GPIO.output(IN2, GPIO.LOW)
95      GPIO.output(IN3, GPIO.LOW)
96      GPIO.output(IN4, GPIO.LOW)
97      GPIO.output(IN1D, GPIO.LOW)
98      GPIO.output(IN2D, GPIO.LOW)
99      GPIO.output(IN3D, GPIO.LOW)
100     GPIO.output(IN4D, GPIO.LOW)
101     time.sleep(0.1)
102     # Backward
103     GPIO.output(IN1, GPIO.LOW)
104     GPIO.output(IN2, GPIO.HIGH)
105     GPIO.output(IN3, GPIO.LOW)
106     GPIO.output(IN4, GPIO.HIGH)
107     GPIO.output(IN1D, GPIO.HIGH)
108     GPIO.output(IN2D, GPIO.LOW)
109     GPIO.output(IN3D, GPIO.HIGH)
110     GPIO.output(IN4D, GPIO.LOW)
111
112 def izquierda():
113     # Left
114     GPIO.output(IN1, GPIO.LOW)
115     GPIO.output(IN2, GPIO.HIGH)
116     GPIO.output(IN3, GPIO.HIGH)
117     GPIO.output(IN4, GPIO.LOW)
118     GPIO.output(IN1D, GPIO.LOW)
119     GPIO.output(IN2D, GPIO.HIGH)
120     GPIO.output(IN3D, GPIO.HIGH)
121     GPIO.output(IN4D, GPIO.LOW)
122     time.sleep(0.1)
123
124 def derecha():
125     # Right
126     GPIO.output(IN1, GPIO.HIGH)
127     GPIO.output(IN2, GPIO.LOW)
128     GPIO.output(IN3, GPIO.LOW)
129     GPIO.output(IN4, GPIO.HIGH)
130     GPIO.output(IN1D, GPIO.HIGH)
131     GPIO.output(IN2D, GPIO.LOW)
132     GPIO.output(IN3D, GPIO.LOW)
133     GPIO.output(IN4D, GPIO.HIGH)
134     time.sleep(0.1)
135
136 def stop():
137     # Stop
138     GPIO.output(IN1, GPIO.LOW)
139     GPIO.output(IN2, GPIO.LOW)
140     GPIO.output(IN3, GPIO.LOW)
141     GPIO.output(IN4, GPIO.LOW)
142     GPIO.output(IN1D, GPIO.LOW)
```

Fuente: Autoría propia basada en editor de texto Sublime Text 3, 2021

Figura 8.12 Código control de movimiento parte 4.

main.py | controlmov.py

```
130         GPIO.output(IN1D, GPIO.HIGH)
131         GPIO.output(IN2D, GPIO.LOW)
132         GPIO.output(IN3D, GPIO.LOW)
133         GPIO.output(IN4D, GPIO.HIGH)
134         time.sleep(0.1)
135
136     def stop():
137         # Stop
138         GPIO.output(IN1, GPIO.LOW)
139         GPIO.output(IN2, GPIO.LOW)
140         GPIO.output(IN3, GPIO.LOW)
141         GPIO.output(IN4, GPIO.LOW)
142         GPIO.output(IN1D, GPIO.LOW)
143         GPIO.output(IN2D, GPIO.LOW)
144         GPIO.output(IN3D, GPIO.LOW)
145         GPIO.output(IN4D, GPIO.LOW)
146         time.sleep(0.1)
147
148
149     boton1 = Button(root, text="FORWARD", command=adelante, width = 10, height = 5)
150     boton2 = Button(root, text="BACK", command=atras, width = 10, height = 5)
151     boton3 = Button(root, text="LEFT", command=izquierda, width = 10, height = 5)
152     boton4 = Button(root, text="RIGHT", command=derecha, width = 10, height = 5)
153     boton5 = Button(root, text="STOP", command=stop, width = 10, height = 5)
154
155     boton1.place(x="1080", y="410")
156     boton2.place(x="1080", y="560")
157     boton3.place(x="1000", y="485")
158     boton4.place(x="1160", y="485")
159     boton5.place(x="1080", y="485")
160
161     boton1.config(bg="black")
162     boton2.config(bg="black")
163     boton3.config(bg="black")
164     boton4.config(bg="black")
165     boton5.config(bg="black")
166
167     boton1.config(fg="white")
168     boton2.config(fg="white")
169     boton3.config(fg="white")
170     boton4.config(fg="white")
171     boton5.config(fg="white")
172
173     root.mainloop()
174
```

Fuente: Autoría propia basada en editor de texto Sublime Text 3, 2021

Figura 8.13 Código de sensores en Python.

```python
import serial, time

arduino = serial.Serial("COM7", 9600)
time.sleep(2)
a = []

while True:
    dato = arduino.readline()
    for Word in dato.split():
        try:
            a.append(float(Word))
        except ValueError:
            pass
    TC = a[0]
    Tem = a[1]
    Hm=a[2]
    Gas=a[3]

    print("Obj C°: ", TC, "///Ambiente °C: ", Tem, "///Humedad %: ", Hm, "///Gas: ", Gas)
    a = []
```

```
Obj C°:  26.31 ///Ambiente °C:  16.3 ///Humedad %:  75.2 ///Gas:  412.0
Obj C°:  26.55 ///Ambiente °C:  16.3 ///Humedad %:  75.2 ///Gas:  410.0
Obj C°:  26.49 ///Ambiente °C:  16.3 ///Humedad %:  75.2 ///Gas:  407.0
Obj C°:  26.29 ///Ambiente °C:  16.3 ///Humedad %:  75.2 ///Gas:  403.0
Obj C°:  26.53 ///Ambiente °C:  16.3 ///Humedad %:  75.2 ///Gas:  399.0
Obj C°:  26.47 ///Ambiente °C:  16.3 ///Humedad %:  75.2 ///Gas:  397.0
Obj C°:  26.67 ///Ambiente °C:  16.3 ///Humedad %:  75.2 ///Gas:  395.0
Obj C°:  26.31 ///Ambiente °C:  16.3 ///Humedad %:  75.2 ///Gas:  392.0
Obj C°:  26.27 ///Ambiente °C:  16.3 ///Humedad %:  75.2 ///Gas:  388.0
Obj C°:  26.37 ///Ambiente °C:  16.3 ///Humedad %:  75.2 ///Gas:  386.0
Obj C°:  26.21 ///Ambiente °C:  16.3 ///Humedad %:  75.2 ///Gas:  381.0
Obj C°:  26.43 ///Ambiente °C:  16.3 ///Humedad %:  75.2 ///Gas:  380.0
Obj C°:  26.49 ///Ambiente °C:  16.3 ///Humedad %:  75.2 ///Gas:  377.0
Obj C°:  26.31 ///Ambiente °C:  16.3 ///Humedad %:  75.2 ///Gas:  374.0
Obj C°:  26.37 ///Ambiente °C:  16.3 ///Humedad %:  75.2 ///Gas:  372.0
Obj C°:  26.35 ///Ambiente °C:  16.3 ///Humedad %:  75.2 ///Gas:  371.0
Obj C°:  26.47 ///Ambiente °C:  16.3 ///Humedad %:  75.2 ///Gas:  369.0
Obj C°:  26.55 ///Ambiente °C:  16.3 ///Humedad %:  75.2 ///Gas:  367.0
Obj C°:  26.21 ///Ambiente °C:  16.3 ///Humedad %:  75.2 ///Gas:  364.0
Obj C°:  26.23 ///Ambiente °C:  16.3 ///Humedad %:  75.2 ///Gas:  361.0
```

Fuente: Autoría propia basada en editor de texto Sublime Text 3, 2021

Figura 8.14 Carta de autorización de publicación.

Tecnológico Nacional de México
Instituto Tecnológico Superior de Teziutlán

CARTA DE AUTORIZACIÓN DEL(LA) AUTOR(A) PARA LA CONSULTA Y PUBLICACIÓN ELECTRÓNICA DEL TRABAJO DE INVESTIGACIÓN

El que suscribe:

JOSÉ ALBERTO	TEPAL	GARCÍA

Con Número de Control	**17TE0459**
Perteneciente al Programa Educativo	**INGENIERÍA MECATRÓNICA**

Por este conducto me permito informar que he dado mi autorización para la consulta y publicación electrónica del trabajo de investigación en los repositorios académicos.

Registrado con el producto:	**TESIS**

Cuyo Tema es:

DISEÑO DE SISTEMA DE CONTROL PARA ROBOT MÉDICO DE ASISTENCIA MÉDICA

Correspondiente al periodo:

AGOSTO 2021-MAYO 2022

Y cuyo(a) director(a) de tesis es:

M.S.C. GABRIEL ÁNGEL RAMIREZ VICENTE

ATENTAMENTE

JOSÉ ALBERTO TEPAL GARCÍA

Nombre y firma

Fecha de emisión: **04/05/2022**
c.c.p. Subdirección Académica

Índice de figuras