

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería



**Diseño e implementación de un robot autónomo diferencial
inalámbrico adaptado al ecosistema robótico de la
Universidad del Valle de Guatemala**

Trabajo de graduación presentado por José Javier Estrada Quezada
para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

2022

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería




**Diseño e implementación de un robot autónomo diferencial
inalámbrico adaptado al ecosistema robótico de la
Universidad del Valle de Guatemala**

Trabajo de graduación presentado por José Javier Estrada Quezada
para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,


2022


Vo.Bo.:

(f) 
MSc. Miguel Enrique Zea Arenales

Tribunal Examinador:

(f) 
MSc. Miguel Enrique Zea Arenales

(f) 
Ing. Diego Alberto Morales Ibañez

(f) 
Dr. Luis Alberto Rivera Estrada

Fecha de aprobación: Guatemala, 06 de enero de 2022.

Agradezco desde el centro de mi corazón a toda mi familia quienes siempre han sido mi apoyo durante este camino. Principalmente mis padres José Mario y Virginia, quienes son un gran ejemplo de vida y siempre me han orientado para dar lo mejor de mí en todo lo que enfrente, apoyándome incondicionalmente en todas mis metas y sueños. De no ser por su esfuerzo y apoyo no hubiese podido llegar a donde estoy.

De igual manera agradezco a mis amigos que me han acompañado durante todo este viaje brindándome su ayuda incondicional y muchos buenos momentos. Simultáneamente quiero agradecer a mi asesor Miguel Zea por todo su apoyo para terminar este trabajo con la excelencia distintiva de la Universidad.

Prefacio	v
Lista de figuras	xii
Lista de cuadros	xiii
Resumen	xv
Abstract	xvii
1. Introducción	1
2. Antecedentes	3
2.1. AMiR	3
2.2. Cellulo	5
2.3. Alice	5
2.4. Jasmine	5
2.5. E-Puck	7
2.6. Kobot	8
2.7. Kilobot	10
2.8. S-bot	10
2.9. R-One	11
2.10. Plataforma robótica fase 1	12
2.11. Plataforma robótica fase 2	14
2.12. Red de comunicación WiFi para una mesa de pruebas de robótica de enjambre	15
3. Justificación	17
4. Objetivos	19
4.1. Objetivo general	19
4.2. Objetivos específicos	19
5. Alcance	21

6. Marco teórico	23
6.1. Comunicación inalámbrica WiFi	23
6.2. Microprocesadores y microcontroladores	23
6.2.1. Microprocesador	23
6.2.2. Microcontrolador	24
6.2.3. Teensy 3.2	24
6.2.4. PIC32MX250F128B	25
6.2.5. ESP8266	26
6.2.6. ESP32	26
6.3. Controladores de motores	27
6.3.1. Controlador MC33926	27
6.4. Encoders	27
6.5. Protocolo de transporte de información MQTT	28
6.6. Topología de red	29
6.7. Sistemas embebidos	30
6.8. Robot móvil diferencial	30
6.9. Control de robot diferencial	31
6.10. Robótica de enjambre	32
6.11. PlatformIO	33
6.12. Inventor	33
6.13. Diseño de PCB	33
6.13.1. Altium Designer	33
6.13.2. Gerbers	33
6.14. OptiTrack	34
6.15. TPU	34
7. Diseño electrónico de la plataforma	35
7.1. Selección de componentes	36
7.1.1. Unidad de control	36
7.1.2. Driver de motores	37
7.1.3. Encoders	37
7.1.4. Alimentación	38
7.2. Potencia de la plataforma	38
7.2.1. Configuración de baterías	39
7.2.2. Regulador de voltaje	39
7.3. Interacción física con la plataforma	40
7.3.1. Encendido y apagado	40
7.3.2. Indicadores LED	40
7.3.3. Módulo de carga	41
8. Diseño mecánico de la plataforma	43
8.1. Placas electrónicas	44
8.2. Diseño de rueda	44
8.2.1. Diseño y fabricación de aro	44
8.2.2. Diseño y fabricación de llanta	45
8.3. Cuerpo de plataforma	45

9. Programación de la plataforma	47
9.1. Librería de encoders	48
9.2. Librería de motores	48
9.3. Librería de I2C	49
9.4. Librería de MQTT	50
10. Pruebas	51
10.1. Pruebas de Payload	51
10.2. Pruebas motores y encoders	51
10.3. Pruebas del sistema de potencia	51
10.4. Pruebas de comunicación MQTT	52
11. Conclusiones	53
12. Recomendaciones	55
13. Bibliografía	57
14. Anexos	59
14.1. Planos de construcción	59
14.2. Placas electrónicas	68
14.3. Repositorio de Github	68

1.	Placa central del robot AMiR [2].	4
2.	Prototipo del robot AMiR.	4
3.	Prototipo del robot Cellulo [3].	5
4.	Prototipo del robot Alice [4].	6
5.	Prototipo del robot Jasmine [5].	6
6.	Diagrama de la estructura del E-puck [6].	7
7.	Prototipo del robot E-Puck [6].	8
8.	Diagrama de la estructura del Kobot [7].	9
9.	Prototipo del robot Kobot [7].	9
10.	Prototipo del robot Kilobot [8].	10
11.	Prototipos del robot S-bot [9].	11
12.	Diagrama de la estructura del S-bot [9].	11
13.	Diagrama de la estructura del robot R-one [10].	12
14.	Prototipo del robot R-one [10].	12
15.	Prototipo del robot realizado en la fase 1 [11].	13
16.	Estructura del robot realizado en la fase 2 [12].	14
17.	Prototipo del robot realizado en la fase 2 [12].	15
18.	Detalle del uso de la red y protocolo MQTT para el robotat [13].	16
19.	Diagrama general de un microprocesador [11].	23
20.	Diagrama general de un microcontrolador [11].	24
21.	Teensy 3.2 [11].	25
22.	PIC32MX250F128B [12].	25
23.	Módulo ESP8266 [13].	26
24.	Microcontrolador ESP32 [15].	27
25.	Diagrama general del protocolo MQTT [13].	28
26.	Topologías de red [18].	29
27.	Sistemas embebidos [19].	30
28.	Diagrama general de un robot diferencial [20].	31
29.	Diagrama dinámico de un robot diferencial [20].	32
30.	Ejemplo de robótica de enjambre [13].	33
31.	Microcontrolador ESP32.	37

32. Driver dual MC33926.	37
33. Driver dual MC33926.	38
34. Baterías Efest 18650.	38
35. Regulador de voltaje a 3.3V.	40
36. Módulo de semáforo indicador LED.	41
37. Módulo de carga y protección BMS 2S.	41
38. Curvas de corriente y voltaje en carga de baterías seleccionadas y fuente a 8.4V	42
39. Ensamblaje final de la plataforma robótica.	43
40. Vista de esquemáticos utilizados para las placas diseñadas.	44
41. Diseño final del aro.	45
42. Diseño final de la llanta.	45
43. Partes diseñadas y fabricadas en corte láser.	46
44. Cilindro diseñado.	46
45. Diseño final de la plataforma.	50
46. Vista de placas diseñadas.	68

Lista de cuadros

1. Orden de señales en pasos	28
2. BOM plataforma robótica	35
3. Estudio de características de microcontroladores	36
4. Power Budget plataforma robótica.	39

El desarrollo de esta plataforma robótica busca mejorar las fases pasadas para poder tener una plataforma más robusta y con posibilidades de integrarse a un ecosistema robótico localizado en la Universidad del Valle de Guatemala. Este desarrollo buscó reducir el tamaño en el plano xy de las placas electrónicas lo cual se logró reduciendo en $714mm^2$. Al igual se incorporó un puerto disponible para conectar una carga útil por medio del protocolo de comunicación I2C para tener la opción de ampliar las capacidades de la plataforma.

Se realizó el diseño electrónico en el cual se seleccionaron los componentes ideales para la plataforma enfocándose en bajo costo y fácil adquisición local en electrónicas del país. El diseño mecánico realizado busca unificar las piezas de forma robusta, utilizando técnicas de fabricación disponibles en la Universidad para que su fabricación sea de bajo costo y rápida de generar. La programación de la plataforma se realizó en Visual Studio Code utilizando herramientas de PlatformIO, estas herramientas permiten el uso del SDK de EspressIf para programar el ESP32 en lenguaje C. Se desarrollaron diversas librerías para expandir las capacidades de la plataforma, permitiendo replicar estas librerías dentro de cualquier otro proyecto fácilmente.

The development of this robotic platform seeks to improve the previous phases in order to have a more robust platform with the possibility of integrating into a robotic ecosystem located at the Universidad del Valle de Guatemala. This development sought to reduce the size in the xy plane of the electronic boards which was achieved by reducing it by $714mm^2$. Likewise, a port was added to connect a payload via the I2C communication protocol to have the option of extending the capacities of the platform.

The electronic design was carried out in which the ideal components for the platform were selected, focusing on low cost and easy local procurement of electronics in the country. The mechanical design carried out seeks to unify the parts in a robust way, using manufacturing techniques available at the University so that their manufacture is low cost and quick to generate. The programming of the platform was done in Visual Studio Code using PlatformIO tools, these tools allow the use of the EspressIf SDK to program the ESP32 in C language. Several libraries were developed to expand the capabilities of the platform, allowing to replicate these libraries within any other project easily.

Se diseñó e implementó una plataforma autónoma enfocada para aplicaciones de robótica de enjambre. La plataforma se incorporará al ecosistema robótico de la Universidad del Valle de Guatemala. Se utilizará comunicación Wi-Fi para interactuar en el ecosistema y el sistema de captura de movimiento, interactuar con la plataforma y la transmisión de datos. En esta iteración se utilizará el microcontrolador *ESP32* para poder interactuar por medio de Wi-Fi al servidor del ecosistema robótico utilizando el protocolo MQTT.

El diseño mecánico se integrará al diseño electrónico buscando optimizar el espacio para reducir las dimensiones comparadas con las iteraciones anteriores desarrolladas en los años 2017 y 2018. Se mejorará el sistema de potencia de la plataforma incorporando un sistema para facilitar la carga y reducir las altas temperaturas que se dieron en la iteración pasada por la alta disipación de energía. Se omitirá la integración con un sensor específico y se reemplazará por el uso del sistema de captura de movimiento OptiTrack. Se adecuó un puerto de comunicación I2C para las futuras posibles aplicaciones de cargas útiles. Estas mejoras se desarrollarán en paralelo con la integración adecuada en el ecosistema robótico de la Universidad del Valle de Guatemala.

Las plataformas robóticas redondas han sido las predilectas para aplicaciones de robots móviles terrestres y en aplicaciones de enjambre. Por lo que ha investigado sobre las iteraciones realizadas por diversas instituciones incluyendo las dos fases previas realizadas por la Universidad del Valle de Guatemala. Esto permite tener un precedente para contar con un punto de inicio, a continuación se muestran los antecedentes más importantes.

2.1. AMiR

El robot *Autonomous miniature mobile robot (AMiR)* es una plataforma robótica de bajo costo diseñada para robótica de enjambre. Este fue desarrollado por investigadores del departamento de ingeniería de sistemas de comunicación y computación de la Universidad de Putra Malasia (UPM). La plataforma busca ser una solución de bajo costo para aplicaciones inteligentes de enjambre y educativas. El costo es un factor importante ya que para soluciones de enjambre se utiliza una buena cantidad de robots. Las dimensiones de este robot son $6cm \times 7.3cm \times 4.7cm$, incluyendo módulos de percepción, locomoción y comunicación. Además incluye un periférico para añadir módulos externos apilables. El módulo de comunicación es de bajo rango y el resto de módulos son de bajo consumo, apto para la batería de $3.7V$, $400mAh$ incorporada. El periférico libre permite incorporar módulos extras como GPS, sensor de color, entre otros periféricos para tener flexibilidad en sus aplicaciones, esto se logró al implementar un protocolo de comunicación y pines de alimentación para los posibles módulos [\[1\]](#).

Para la detección de obstáculos y permitir que AMiR sea autónomo se incorporan emisores IR y foto transistores IR. El infrarrojo es el encargado de percibir el entorno, estos se utilizaron en una topología de 60° entre transmisor y receptor. Utiliza el foto transistor TEFT-4300 y el diodo emisor TSKS-5400 por su amplio ángulo de utilidad y rápido tiempo de respuesta. Los sensores infrarrojos también se utilizaron como comunicación entre robots.

Ya que el receptor de otro robot puede interpretar la señal que el emisor del primer robot envía. Esto depende de la recepción de datos en el IR [2].

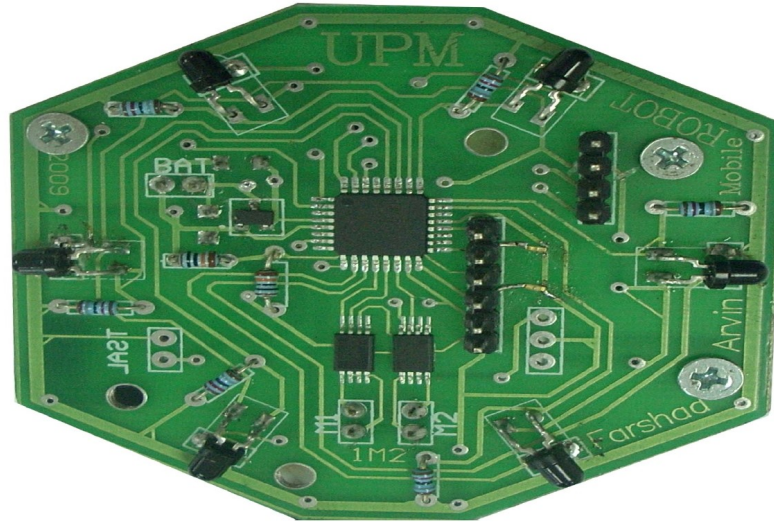


Figura 1: Placa central del robot AMiR [2].

Se utiliza un microcontrolador ATMEGA168 a una velocidad de 8MHz. El microcontrolador tiene 16kB de SRAM disponibles y requiere $250\mu A$ para funcionar. El movimiento del robot es diferencial lo que permite utilizar dos llantas únicamente. Cada una controlada de forma individual con velocidades diferentes para cada uno de sus motores [2].

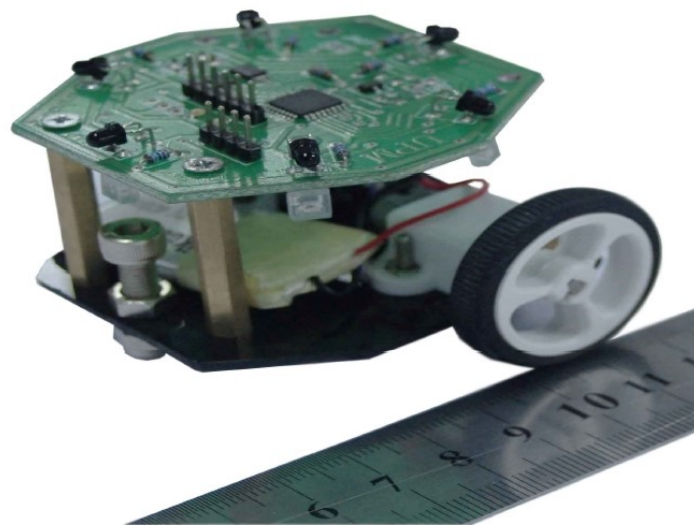


Figura 2: Prototipo del robot AMiR.

2.2. Cellulo

Esta plataforma robótica surge de la necesidad de investigar la intersección de robots versátiles, robot implementables en clases y las limitaciones de un salón de clases. El movimiento de esta plataforma se basa en movimiento holonómico, retroalimentación háptica y localización exacta. Este robot tiene unas dimensiones de $75mm \times 75mm$, utiliza un procesador *PIC32MZ1024ECG064* y está hecho de PLA.

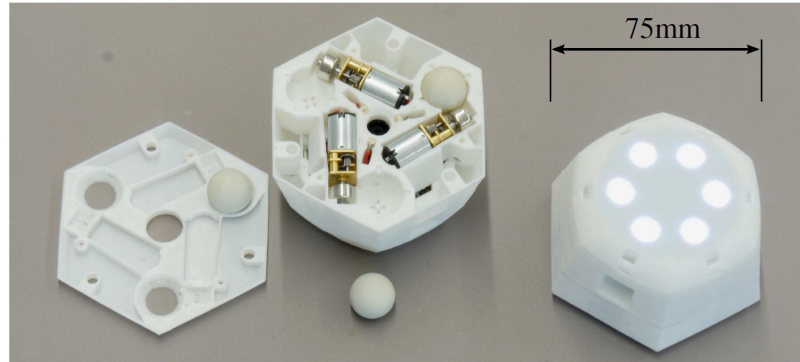


Figura 3: Prototipo del robot Cellulo [3].

Esta plataforma opera en hojas de papel y su localización se guía de la lectura de la impresión en estas. Cuenta con comunicación inalámbrica *Bluetooth*, cuenta con 6 botones interactivos retro iluminados por LEDs RGB [3].

2.3. Alice

Alice es un micro-robot móvil MMR por sus siglas en inglés. Cuenta con una autonomía de 10 horas, es modular y cuenta con un software flexible. Puede caer de hasta 1 metro sin sufrir daño crítico. Sus dimensiones son $21mm \times 21mm \times 12mm$ con un peso de tan solo $5g$ y llega a una velocidad máxima de $40 \frac{mm}{s}$. Es controlado por 2 motores *Swatch* y el microcontrolador *PIC16F84 @ 4MHZ* y cuenta con sensores IR para detección de obstáculos y comunicación cercana. Alice se ha utilizado para realizar mapas, fútbol robótico, investigaciones biológicas y aplicaciones lúdicas [4].

2.4. Jasmine

Jasmine es un micro robot de hardware abierto al público de costo accesible. Sus dimensiones son $30mm \times 30mm \times 20mm$ y cuenta con dos microcontroladores Atmel AVR Mega comunicados por I2C, los cuales son *Atmel Mega88* utilizado para control de motores, odometría y gestión de energía interna y el *Atmel Mega168* utilizado en la comunicación, control remoto y sensores. Se utilizan sensores IR para el sensado de proximidad y comunicación colocados a 60° entre sí con un rango máximo de $200mm$ y un mínimo de $100mm$.

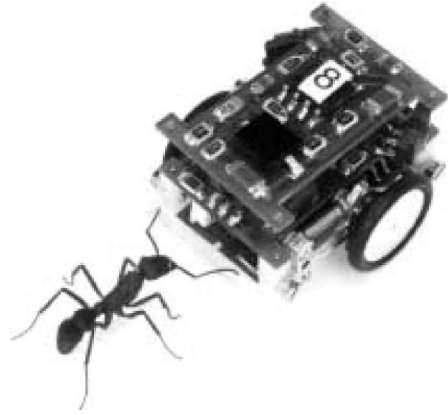


Figura 4: Prototipo del robot Alice [4].

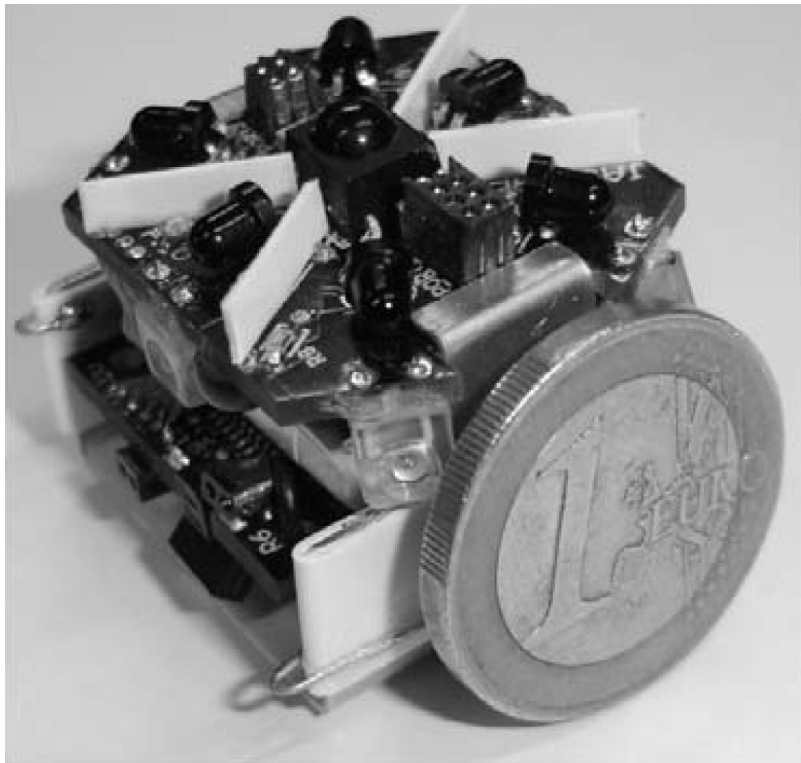


Figura 5: Prototipo del robot Jasmine [5].

Jasmine utiliza dos motores DC con cajas reductoras junto a dos llantas diferenciales para su movimiento. Junto a los motores se utilizan encoders odométricos que normaliza el movimiento del robot y estiman la distancia desplazada por el mismo. Su programación esta basada en el lenguaje C utilizando el compilador *gcc*. Básicamente se repiten cuatro pasos constantemente, estos son: lectura de sensores, comunicación cercana por IR, toma de decisiones y ejecución delas mismas [5].

2.5. E-Puck

E-puck es un robot móvil desarrollado por el *École Polytechnique Fédérale de Lausanne* (EPFL) diseñado para estudios de robótica de enjambre. Su diseño mecánico y programación se encuentran como licencia de código abierto. Su diseño mecánico tiene un diámetro de $75mm$ y su altura depende de los módulos que se utilizan. El módulo principal contiene la batería, motores y llantas y el PCB principal con el procesador principal.

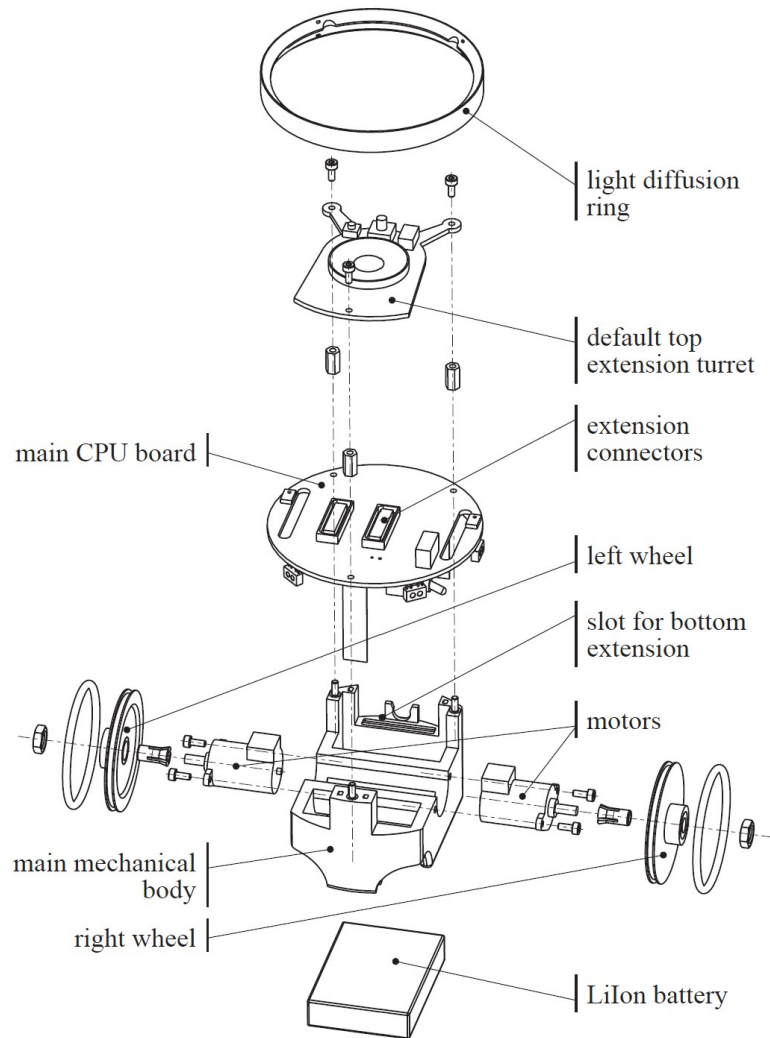


Figura 6: Diagrama de la estructura del E-puck [6].

El microcontrolador utilizado es un *PIC30F6014A @64MHz* el cual se encarga del movimiento, sensado e interacción con el usuario. El E-Puck utiliza 8 sensores de proximidad IR, un acelerómetro de tres ejes, 3 micrófonos y una cámara al frente del robot. Utilizando los sensores integrados se controlan 2 motores paso a paso para controlar el movimiento del robot, una bocina y 8 LEDs que funcionan como indicadores.



Figura 7: Prototipo del robot E-Puck [6].

Se utilizan 2 de los LED's incorporados para indicar el estado de la batería. Cuenta con un puerto disponible para programar la memoria del robot y depurar código. Se puede controlar con un control remoto de IR como los utilizados por televisores. Utiliza principalmente comunicación *Bluetooth* para comunicarse con ordenadores y hasta 7 E-pucks. También se tienen comportamientos pre-programados los cuales se pueden seleccionar con un interruptor de 16 posiciones [6].

2.6. Kobot

Kobot es una plataforma robótica circular ideada para aplicaciones de robótica de enjambre. Sus dimensiones son $\varnothing = 120mm$ y su peso es de $350g$ se enfocó su diseño en ser liviano, pequeño, con gran eficiencia energética y barato. Utiliza motores DC de buena eficiencia, bajo perfil y alto toque, controlados por drivers Si9988.

Se utiliza un microcontrolador *PIC16F877A @20MHz* junto a algoritmos de control para determinar el comportamiento del Kobot. Utiliza sensores IR para determinar la cercanía de distintos robots u obstáculos. Su comunicación inalámbrica utiliza el protocolo *IEEE802.15.4/ZigBee* el cual permite una buena comunicación de punto a punto con una gran eficiencia energética.

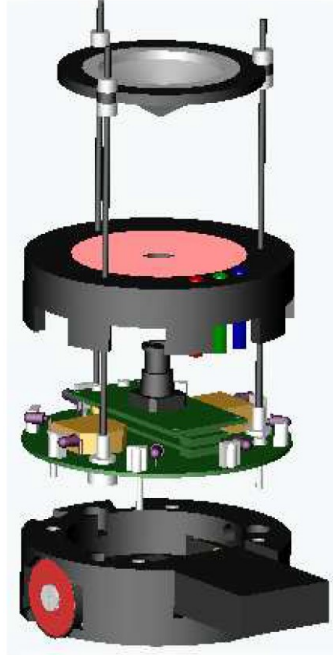


Figura 8: Diagrama de la estructura del Kobot [7].



Figura 9: Prototipo del robot Kobot [7].

Incorpora un módulo de cámara para incluir programas de visión de computadora. Se utiliza el procesador *PXA255 @200MHz* el cual es basado en Linux, una plataforma ideal para algoritmos de visión de computadoras. Cuenta con un tipo de programación paralela la cual consiste en que un programa permite la descarga de nueva información e interacción con el usuario mientras el segundo programa reescribe la memoria del controlador con la información añadida [7].

2.7. Kilobot

Kilobot es una plataforma para robótica de enjambre de bajo costo, su armado es sencillo y accesible para tener cientos o miles de estos para aplicaciones de enjambre. Sus medidas son 3.3cm de radio, su velocidad es de hasta $1\frac{\text{cm}}{\text{s}}$ y cuenta con una batería de 3.4V y 160mAh . La carga de estos robots es muy sencilla ya que sus patas son el pin de entrada para la carga de batería y cuenta con una antena para cerrar el circuito.

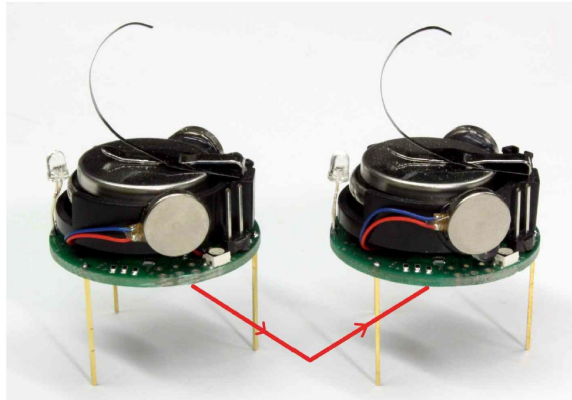


Figura 10: Prototipo del robot Kilobot [8].

La comunicación entre robots se da por sensores infrarrojos que utilizan las sombras en la superficie del trabajo para enviar datos y detectar obstáculos. El movimiento de este robot no es por medio de llantas sino utiliza motores de vibración para mover las tres patas que este tiene. El procesador que utiliza el kilobot es un *Atmega328 @8MHz* en el cual se implementa un controlador *SDASH*. Este controlador es un algoritmo desarrollado para armar y mantener la forma colectiva del kilobot. Este controlador permite que el robot se desplace, gire, se comunique y calcule la distancia entre robots cercanos [8].

2.8. S-bot

S-bot es un robot circular con diámetro de 116mm y una altura de 100mm . Su movimiento no es diferencial sino utiliza dos motores DC para desplazar un tipo de banda como el movimiento de los tanques. Cuenta con 15 sensores infrarrojos para la detección de obstáculos, sensor de torque, sensor de fuerza, acelerómetro de 3 ejes y una cámara omnidireccional. Incorpora dos grippers para aplicaciones físicas con objetos u otros robots. Utiliza comunicación WiFi y su procesador central es un Intel XScale basado en Linux y utiliza tres microcontroladores PIC auxiliares para el control de los módulos comunicados por I2C al procesador central [9].



Figura 11: Prototipos del robot S-bot [9].

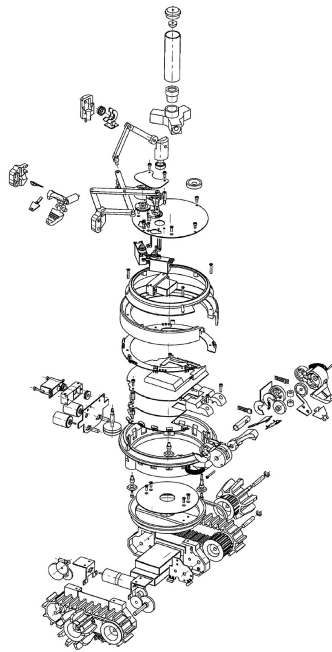


Figura 12: Diagrama de la estructura del S-bot [9].

2.9. R-One

R-one es un robot diseñado par cumplir con tres principales aplicaciones estas son educación, alcance e investigación. Este robot logra cumplir con estos pilares bajo tres requisitos críticos que son el bajo costo, diseño avanzado y facilidad de uso. En el año 2012 se lanzo como un proyecto de *hardware* abierto.



Figura 13: Diagrama de la estructura del robot R-one [10].

Su movimiento es diferencial utilizando encoder y sensores de luz, a esto se le añaden sensores de proximidad y un giroscopio y acelerómetro de tres ejes. Se comunica por IR entre robots permitiendo medir la distancia y conocer la orientación de sus vecinos. El microcontrolador utilizado es un *ARM 32bits @50MHz* [10].



Figura 14: Prototipo del robot R-one [10].

2.10. Plataforma robótica fase 1

Esta plataforma se desarrolló en la Universidad del Valle de Guatemala en el año 2017 como parte de un proyecto de graduación. En este diseño se utilizó una falda de 6 sensores ultrasónicos *HC-SR04*, la cual se utiliza para detectar objetos. Los sensores se encuentran en una PCB separada para permitir el uso de módulos. En esta PCB se controlan los sensores

de forma que lo sensado se envíe a una entrada digital de la unidad de control.

El microcontrolador utilizado fue un *Teensy 3.2* basado en ARM Cortex programado en el lenguaje e IDE de *Arduino*. Añadido se tiene un giroscopio y acelerómetro de tres ejes. El movimiento de la plataforma es diferencial lo cual utiliza dos llantas con un motor cada una de ellas. Los motores utilizan un controlador *H DRV8833* el cual emplea señales PWM para el manejo de las llantas. Se utiliza un encoder en cada motor para datos de velocidad, distancia y sentido de giro.

Se utiliza un módulo *ESP-8266* como centro de comunicación inalámbrica. Se utilizó el protocolo TCP para asegurar un envío veloz y confiable de datos. Se recomienda en este trabajo el rediseño de la falda de sensores, mejorar el desempeño del microcontrolador y tener en cuenta la calibración de los encoders [11].

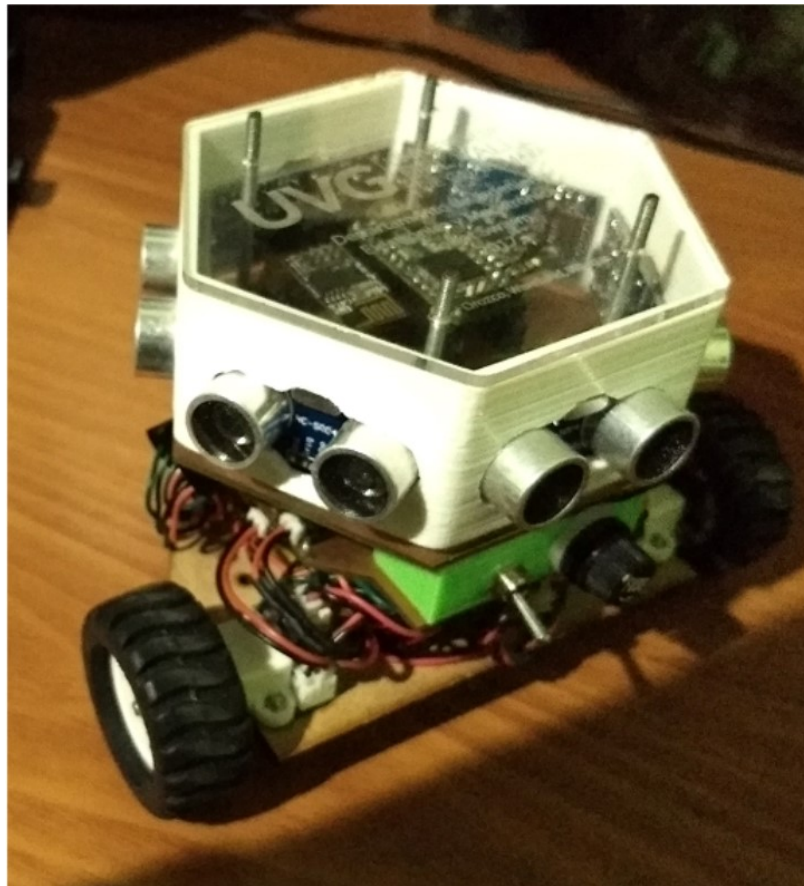


Figura 15: Prototipo del robot realizado en la fase 1 [11].

2.11. Plataforma robótica fase 2

En esta segunda iteración se utiliza un microcontrolador distinto para optimizar el desempeño. También se rediseñó el sensor de proximidad utilizado y se rediseñó la estructura mecánica de la plataforma. En este caso se utilizó un microcontrolador *PIC32MX250F128B @50MHz*. El movimiento sigue siendo diferencial utilizando dos motores, sus controladores y sus encoders. La comunicación permaneció idéntica utilizando un módulo *ESP8266* el cual utiliza comandos AT y el protocolo TCP.

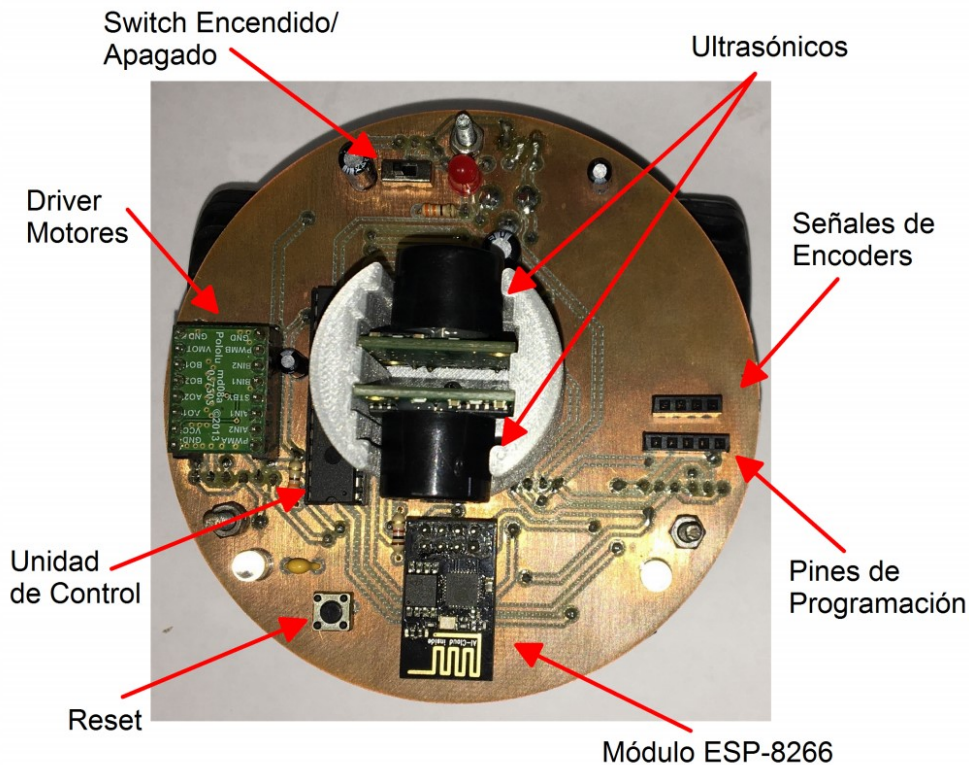


Figura 16: Estructura del robot realizado en la fase 2 [12].

En el rediseño mecánico se redujo el área en el plano xy en 48.27 cm lo que representa una reducción del 36.44% . El nuevo sensor utilizado los sensores ultrasónicos *LV-MaxSonar-EZ1* en una modalidad de sonar utilizando un servomotor para poder sondear el área y detectar obstáculos cercanos.

Se establecieron librerías en Matlab para controlar la plataforma desde un ordenador. Algunas funciones fueron establecidas desde la iteración pasada. En esta iteración se mejoró la librería incluyendo funciones para obtener la dirección IP. El principal factor a mejorar es el sistema de potencia ya que se llegan a temperaturas altas en el módulo Wi-Fi y el regulador de voltaje llegando a 41.8°C . Esto no representa un problema a los componentes, pero no es una condición ideal.

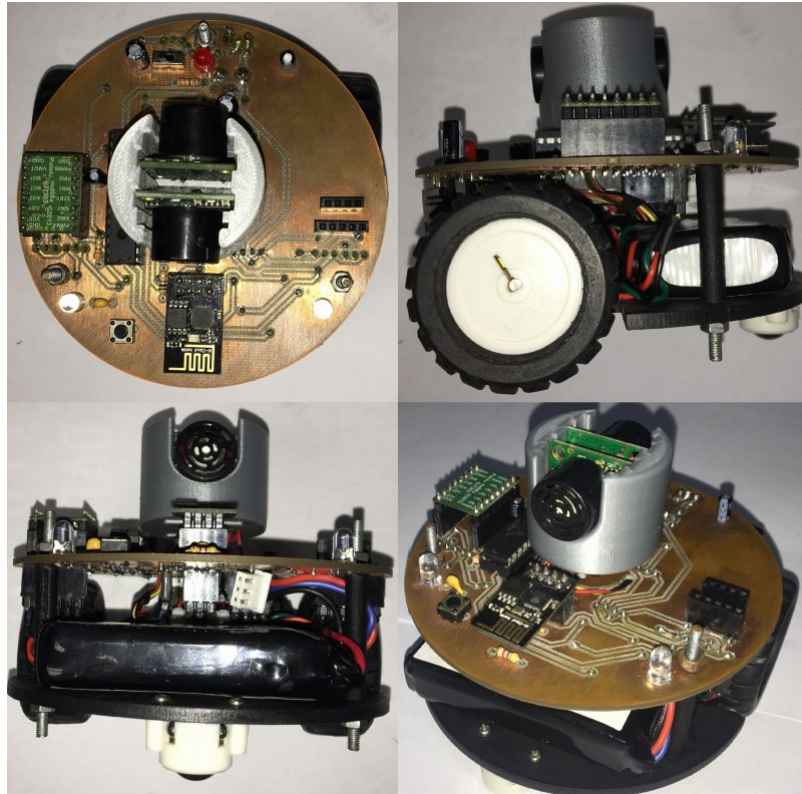


Figura 17: Prototipo del robot realizado en la fase 2 [12].

En las recomendaciones tenemos que podríamos utilizar componentes de superficie para reducir el tamaño del PCB. Se puede mejorar los sensores utilizados para minimizar el tiempo de barrido en la detección de la plataforma. Reducir el tamaño de la batería y las llantas, junto con buscar reducir los cotos de la plataforma [12].

2.12. Red de comunicación WiFi para una mesa de pruebas de robótica de enjambre

En este trabajo de graduación de la Universidad del Valle de Guatemala se estableció una red de comunicación Wi-Fi basada en el protocolo MQTT. Esta red se basa en el módulo *ESP8266* el cual se programó en el lenguaje *Micropython*. La topología de esta red permite el envío de datos desde los robots a una interfaz de usuario. En esta interfaz es posible cargar un nuevo programa a los robots. Esta es la base de la red a utilizar en el ecosistema robótico a implementar en la Universidad del Valle de Guatemala [13].

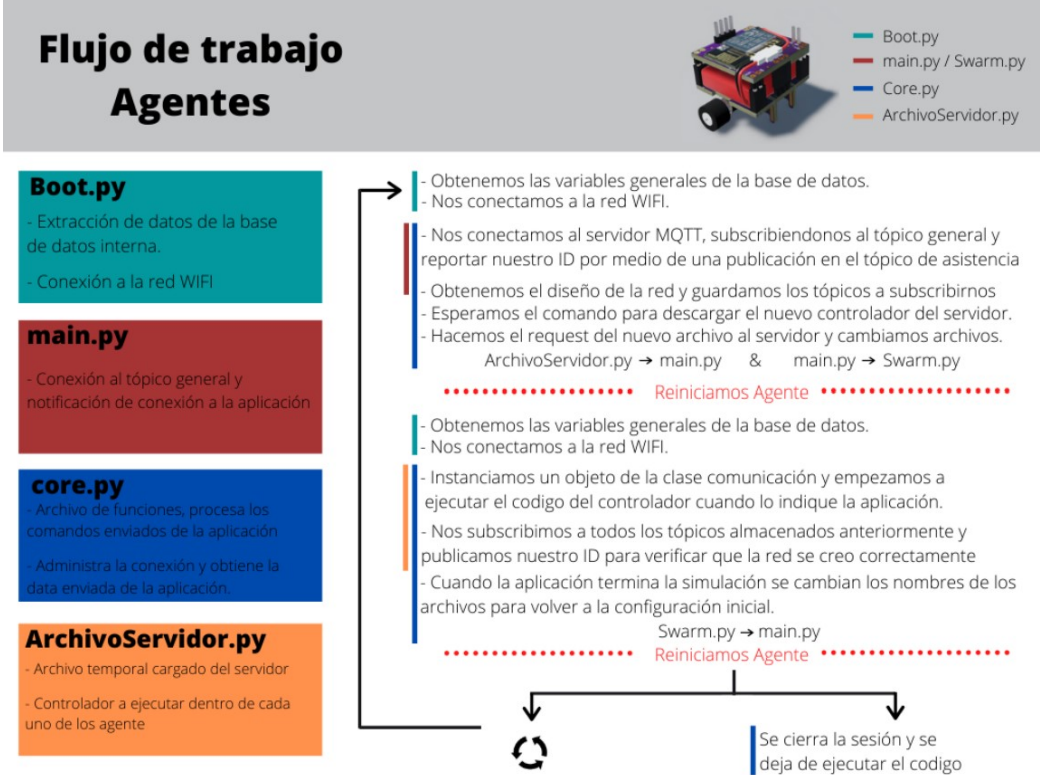


Figura 18: Detalle del uso de la red y protocolo MQTT para el robotat [13].

Este proyecto surge con la necesidad de expandir el trabajo de fases anteriores en el desarrollo de un robot autónomo diferencial. Se busca expandir la capacidad de interactuar en un ecosistema complejo no limitado a una mesa de trabajo. Se busca participar en un ecosistema de agentes robóticos interconectados por WiFi mediante el protocolo MQTT, el ecosistema se denomina Robotat.

Con los antecedentes mencionados anteriormente se tiene un buen punto de inicio para este proyecto, el cual surge de la necesidad de mejorar el diseño de la plataforma robótica e integrarla a la red principal del ecosistema robótico de la Universidad del Valle de Guatemala. Tomando en cuenta los avances realizados en las dos iteraciones pasadas de la plataforma robótica se mejoraran los sistemas necesarios y se añadirá la opción de incluir visión por computadora y la integración en el ecosistema robótico.

Se requiere también mejorar las plataformas existentes para poder incluirlas en las nuevas herramientas de la Universidad del Valle de Guatemala. Como en este caso el sistema de captura de movimiento *Optitrack*. Ya que al poder incorporar estas plataformas, la Universidad del Valle de Guatemala contará con un ecosistema único en la región que favorecerá la investigación, estudio y desarrollo de nuevas tecnologías que incorporen sistemas robóticos. Se obtendrá un robot que permite expandir sus capacidades mediante módulos externos y que se integra de buena manera al ecosistema robótico.

4.1. Objetivo general

Diseñar e implementar un robot diferencial de bajo costo con comunicación Wi-Fi adaptable al sistema de captura de movimiento del ecosistema robótico de la Universidad del Valle de Guatemala.

4.2. Objetivos específicos

- Eficientizar la distribución de potencia y las dimensiones del diseño mecánico de la fase previa de la plataforma robótica.
- Expandir las capacidades de la plataforma integrándola a un entorno de captura de movimiento y comunicación inalámbrica basada en el microcontrolador ESP32, comunicación Wi-Fi y el protocolo MQTT.
- Implementar una interfaz de distribución de potencia y comunicación digital que permita la incorporación de diferentes tipos de carga útil (payload) para sensado y actuación.

Este trabajo surge como continuación de la plataforma desarrollada en las dos etapas anteriores. La primera siendo el trabajo del año 2017 documentada en *Reingeniería de Megaproyectos Fase 1* y la segunda fase en el 2018 documentada en *Diseño e implementación de una nueva plataforma móvil para aplicaciones en robótica de enjambre..* Las recomendaciones más importantes de la primera fase fueron diseñar una nueva falda se sensores para mejorar las lecturas, mejorar el desempeño del microcontrolador utilizado y calibración de encoders. En cuanto a la segunda fase se recomienda reducir las dimensiones de la plataforma con diversas estrategias y mejorar los sensores utilizados. Junto a estas recomendaciones se tienen los nuevos requerimientos del ecosistema, se requiere que se pueda comunicar de forma nativa el microcontrolador por MQTT con el entorno, que sea un modelo que permita implementar diferentes módulos con sensores o actuadores sobre la plataforma y se requiere mejorar el sistema de potencia de la plataforma.

La plataforma diseñada tendrá la posibilidad de integrarse en aplicaciones de robótica de enjambre dentro del ecosistema robótico de la Universidad. Se mejorará el diseño electrónico, de potencia y mecánico de las fases anteriores para tener un modelo robusto, compacto y versátil. Se sustituyó el microcontrolador utilizando un ESP32 para poder implementar comunicación inalámbrica MQTT con el ecosistema y tener un procesador potente con gran capacidad computacional. Se omite la presencia de un tipo de sensor específico para dejar un espacio de carga útil disponible por medio de comunicación I2C permitiendo a la plataforma ampliar por medio de futuros módulos externos sus capacidades, estos módulos no forman parte de este trabajo de graduación. La comunicación implementada permite recibir información del sistema de captura de movimiento *OptiTrack* para capturar la orientación y posición actual del robot.

Las pruebas sufrieron complicaciones por la situación actual de la pandemia a nivel mundial. Las diferentes restricciones por bioseguridad afecta el transito de componentes y restringe el uso de algunas instalaciones de la Universidad.

6.1. Comunicación inalámbrica WiFi

WiFi es un acrónimo el cual se refiere a *Wireless Fidelity* el cual es un tipo de interconexión de red inalámbrica de alta velocidad y corta distancia. Este es un medio en el cual los dispositivos habilitados se conectan entre sí o a internet por medio de un punto de acceso de red inalámbrica [14].

6.2. Microprocesadores y microcontroladores

6.2.1. Microprocesador

Un microprocesador es un circuito integrado el cual cuenta con una unidad lógica y aritmética, registros de propósito general, *stack pointer*, *program counter*, reloj interno y circuitos de interrupciones.

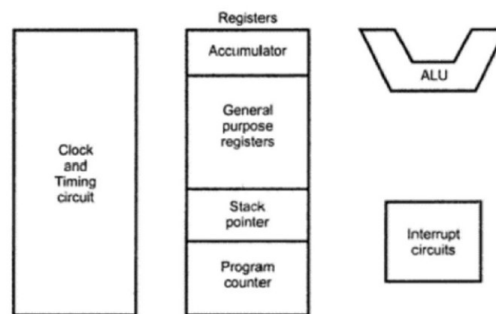


Figura 19: Diagrama general de un microprocesador [11].

La unidad lógica y aritmética, el reloj interno y los circuitos de interrupciones son componentes electrónicos los cuales están interconectados. Mientras que el *stack pointer*, registros de propósito general y *program counter* son espacios de la memoria del procesador. Esta memoria esta implementada en el silicio del procesador y es rápida pero con alto costo y corta extensión comparado con una memoria RAM [11].

6.2.2. Microcontrolador

Un microcontrolador incorpora las características de un microprocesador y añade memorias RAM y ROM, entradas, salidas y contadores dentro de un mismo circuito integrado. Su escalabilidad es menos a la de un microprocesador, pero tiene menor latencia entre los dispositivos periféricos y el procesador, lo cual implica una mayor velocidad [11].

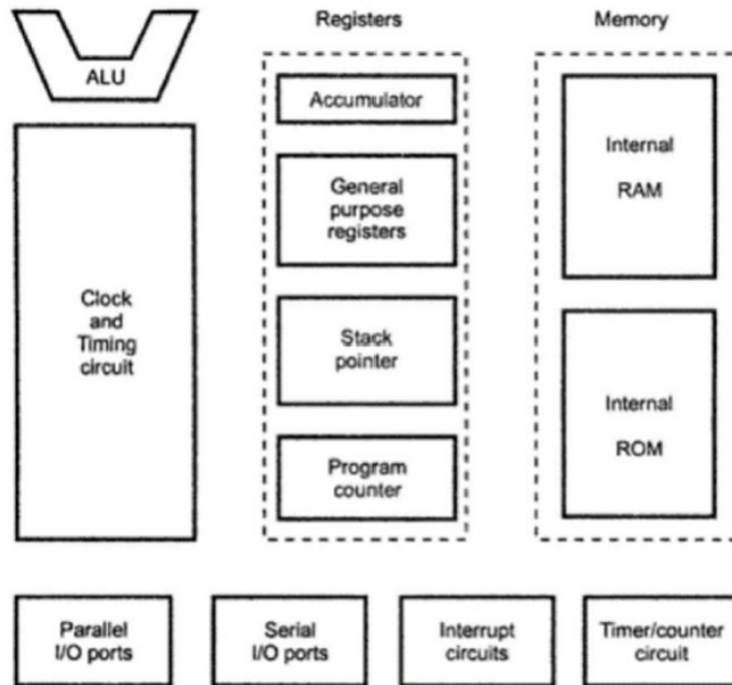


Figura 20: Diagrama general de un microcontrolador [11].

6.2.3. Teensy 3.2

Esta placa de desarrollo es parte de la tercera generación de Teensy. Cuenta con un microcontrolador *MK20DX256VLH7* basado en el procesador ARM cortex M4. Tiene 256kB de memoria flash e igual de caché y 64kB de memoria RAM. Tiene 24 pines digitales de entrada y salida opera a 3.3V pero soporta 5V. Tiene 21 entradas analógicas y una salida analógica de 12 bits de resolución. Cuenta con 12 temporizadores, 12 salidas PWM, interfaces SPI, I2C y una de audio digital [11].

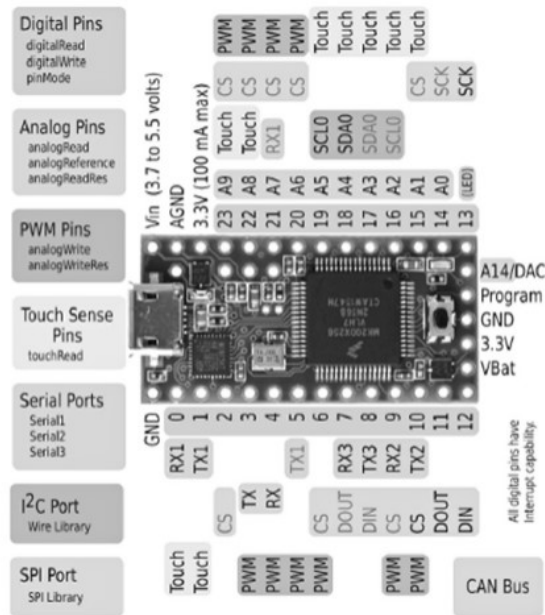


Figura 21: Teensy 3.2 [11].

6.2.4. PIC32MX250F128B

Es un microcontrolador con tecnología de 32 bits, que opera a 50MHz alimentado por 2.3 o 3.6V y consume 0.5mA por MHz. Cuenta con un PLL programable y varias fuentes de osciladores de reloj. Tiene una memoria flash de 256kB y SRAM de 64kB. Cuenta con dos módulos UART, 5 temporizadores, 5 módulos *Capture and Compare*, 5 interrupciones externas y 3 comparadores analógicos [12].



Figura 22: PIC32MX250F128B [12].

6.2.5. ESP8266

Este módulo WiFi es un SOC autónomo que maneja protocolos TCP/IP. El módulo integra un microcontrolador Tensilica L106 de 32 bits de bajo consumo. Cuenta con una interfaz UART de comunicación. Su voltaje de operación es entre 3 y 3.3V. Su velocidad de reloj es de 160MHz y soporta IPv4 y protocolos TCP/UDP/HTTP/FTP. El estándar que utiliza es el IEEE802.11b/g/n. Inicialmente está programado para interpretar comandos AT, pero puede programarse en lenguaje C y Micropython [13].

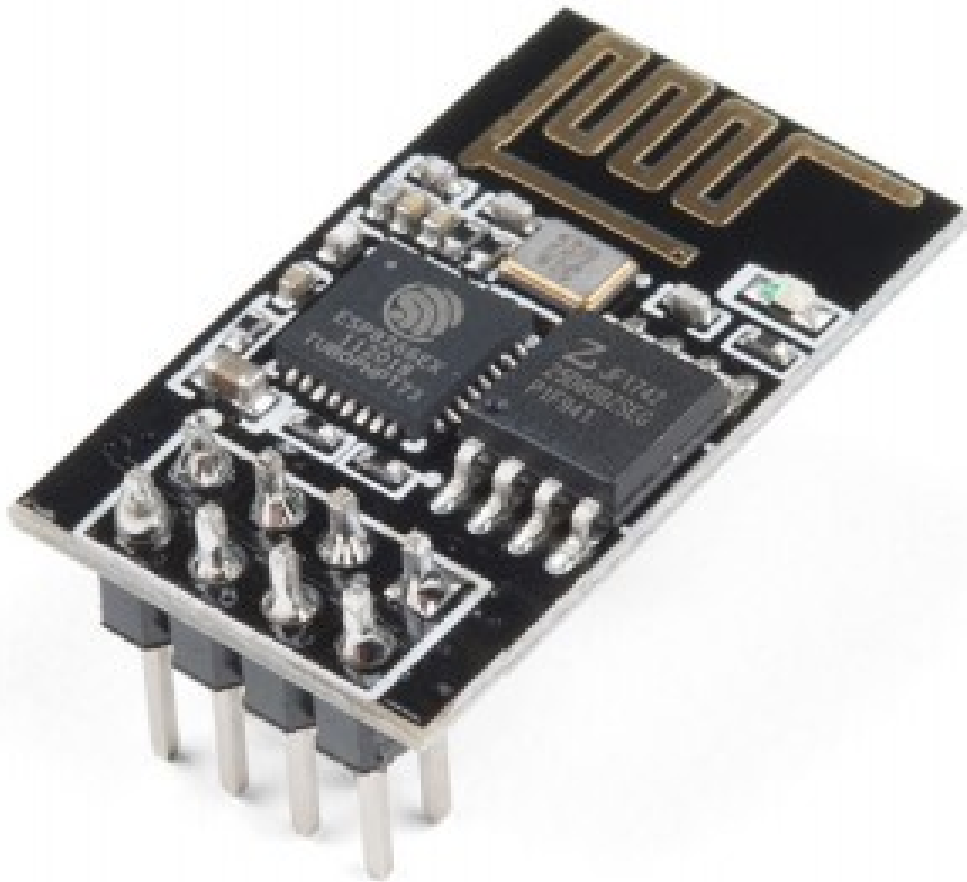


Figura 23: Módulo ESP8266 [13].

6.2.6. ESP32

Este módulo integra un microprocesador Xtensas dual-core de 32 bits LX6 que puede correr hasta 240MHz. Cuenta con 448kB de ROM para reinicio y funciones del núcleo, 520kB de SRAM para datos e instrucciones. Cuenta con una antena para utilizar comunicación WiFi utiliza el protocolo 802.11b/g/n hasta una velocidad de 150Mbps, también cuenta con comunicación Bluetooth. Cuenta con interfaces para una tarjeta SD comunicación UART,

SPI, SDIO, I2C, LEDPWM, Motor PWM, I2S, IR, contador de pulsos, convertidores analógicos a digitales y convertidores digitales a analógico. Opera con un voltaje entre 3.0 a 3.6V y soporta temperaturas entre -40 a 85°C [15].

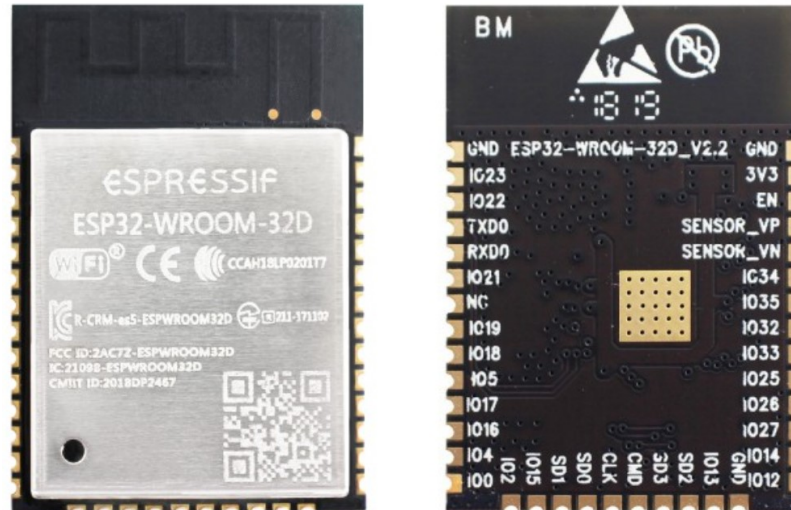


Figura 24: Microcontrolador ESP32 [15].

6.3. Controladores de motores

Los controladores de motores DC se encargan de asignar el sentido de giro y la velocidad del motor. Estos también brindan protección de sobre corriente, sobre temperatura y voltaje de reversa tanto a los motores como al controlador del sistema. Por lo general estos controladores utilizan señales PWM y puentes H para generar el control del motor [12].

6.3.1. Controlador MC33926

El controlador MC33926 de la marca Pololu es un puente H diseñado para control de motores. Puede llegar a una corriente máxima de 5.0A, permite modular señales PWM hasta 20kHz y se alimenta con un voltaje entre 5.0 V y 28V en operación continua. Cuenta con protección contra bajo voltaje, voltaje excesivo, protección de sobre corriente y sobre temperatura [16].

6.4. Encoders

El encoder es un dispositivo que interpreta el movimiento radial como una señal eléctrica. Esta señal puede ser procesada para interpretar la posición, velocidad, sentido de giro y cantidad de revoluciones dadas. Existen diferentes tipos de tecnología que se puede utilizar para un encoder. Principalmente, se tienen encoders mecánicos, ópticos, resistivos y magnéticos. El encoder magnético es uno de los más exactos y utiliza la polarización Norte - Sur

de un imán. Al rotar el motor el polo del imán cambia de posición y podemos detectar los flancos de cambio para crear la señal eléctrica interpretada.

En los encoders rotatorios se tienen dos señales de salida. Las cuales se comportan de la siguiente forma durante un paso. Este comportamiento indica la dirección de giro dependiendo de cual pin envíe la señal primero. El comportamiento se expresa en el cuadro [17](#).

Posición	Bit 1	Bit 2
Paso 1	0	0
1/4	1	0
1/2	1	1
3/4	0	1
Paso 2	0	0

Cuadro 1: Orden de señales en pasos

6.5. Protocolo de transporte de información MQTT

El protocolo de transporte de telemetría de cola de mensajes por sus siglas en ingles MQTT es un protocolo de transporte de mensajes cliente/servidor el cual se basa en las publicaciones y suscripciones de los "tópicos." temas designados. Cada vez que un mensaje es publicado en el "tópico"seleccionado, todos los clientes que estén suscritos a este tópico reciben este mensaje. Este protocolo es ideal para aplicaciones de IOT ya que se envían pequeños paquetes de datos lo cual reduce la cantidad de ancho de banda requerido. Este protocolo se distingue por tener un uso de mensajes generales para las suscripciones y publicación de datos independientes de la aplicación. El transporte en este protocolo es transparente y con flujo optimizado permitiendo que se reduzca el tráfico en la red. Este protocolo cuenta con un mecanismo de notificación de desconexión [13](#).

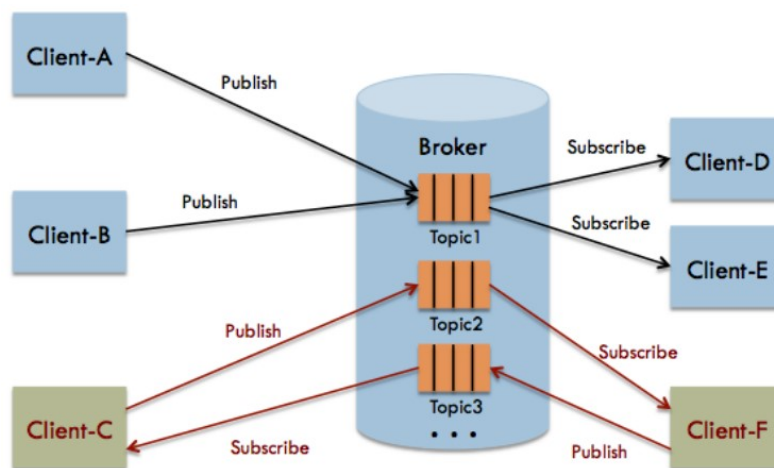


Figura 25: Diagrama general del protocolo MQTT [13](#).

6.6. Topología de red

La topología de una red específica a la disposición física de la conexión de dispositivos en la red. Las topologías más comunes son las siguientes:

Red lineal

Red establecida en la cual el enlace se establece mediante una única vía. Cada nodo tiene conexión con su antecesor y sucesor a excepción de los nodos de inicio y final [18].

Red de estrella

En este caso los nodos se enlazan de manera densa y uniforme al rededor del nodo central del sistema. Existen jerarquías en la cual un nodo solamente se relaciona con un punto superior en jerarquía. Pueden tener pérdidas de vías de enlace lo cual es un gran problema. Esta estructura permite un apropiado manejo y es muy económico [18].

Red poligonal

En esta estructura los nodos se unen entre sí formando un anillo. En esta estructura no hay jerarquía, y suele ser una solución confiable pero con una desventaja económica [18].

Red en malla

Esta estructura une a cada uno de los nodos del sistema. Es cara de implementar pero cuenta con mejores características de seguridad y estabilidad [18].

Red mixta

Este tipo de configuración une las anteriores estructuras lo cual permite tener un gran desempeño y equilibrarlo con el factor económico [18].

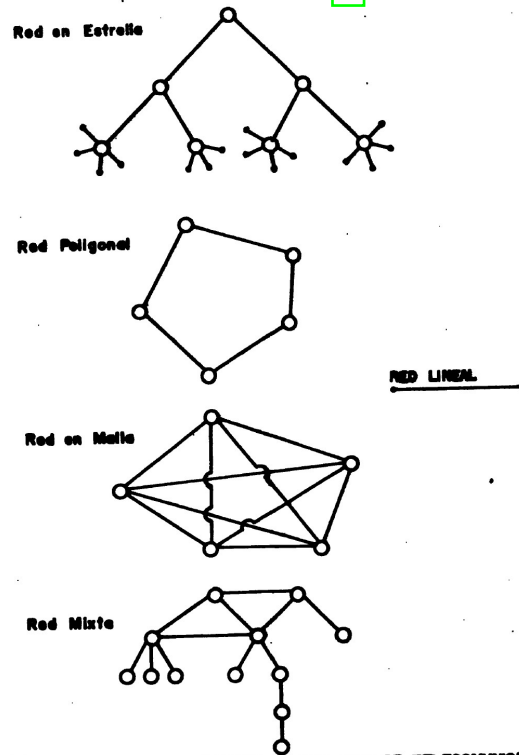


Figura 26: Topologías de red [18].

6.7. Sistemas embebidos

Son sistemas no programables de propósito general, sino un sistema computacional dedicado. En su mayoría estos sistemas están compuestos por software y hardware diseñados específicamente para la tarea que tiene que cumplir. La mayoría de microcontroladores se utilizan para aplicaciones de sistemas embebidos. Estos sistemas tienen un software que utiliza pequeñas cantidades de memoria, capacidades limitadas de procesamiento y limitan el consumo [19].



Figura 27: Sistemas embebidos [19].

6.8. Robot móvil diferencial

Un robot móvil diferencial cuenta con dos ruedas de tracción acopladas a un motor DC cada una. La traslación y rotación de este tipo de plataformas se determina con el movimiento independiente de cada una de las ruedas. Este modelo cinemático se denomina como el modelo unicycle.

El análisis de la cinemática se basa en varias hipótesis: el robot se mueve en una superficie plana, los ejes de referencia son perpendiculares al suelo, el robot no es flexible, el contacto entre la rueda y el suelo se da en un punto a la vez y no existe deslizamiento. Basando en el modelo del unicycle, es decir, un carro con velocidad lineal v y velocidad angular w las cuales describen la posición y orientación del robot en coordenadas x , y y θ . El modelo puede ser un unicycle real y un unicycle diferencial, denominado *differential drive* en inglés. Las ecuaciones que describen la posición en función de las velocidades según el modelo del unicycle real son las siguientes:

$$\dot{x} = r\dot{\phi} \cos(\theta) \quad \dot{y} = r\dot{\phi} \sin(\theta) \quad \dot{\theta} = w$$

Basado en el modelo del robot diferencial podemos describir su cinemática con las ecuaciones:

$$\dot{x}_0 = v \cos(\theta), \quad \dot{y}_0 = v \sin(\theta), \quad \dot{\theta} = \omega$$

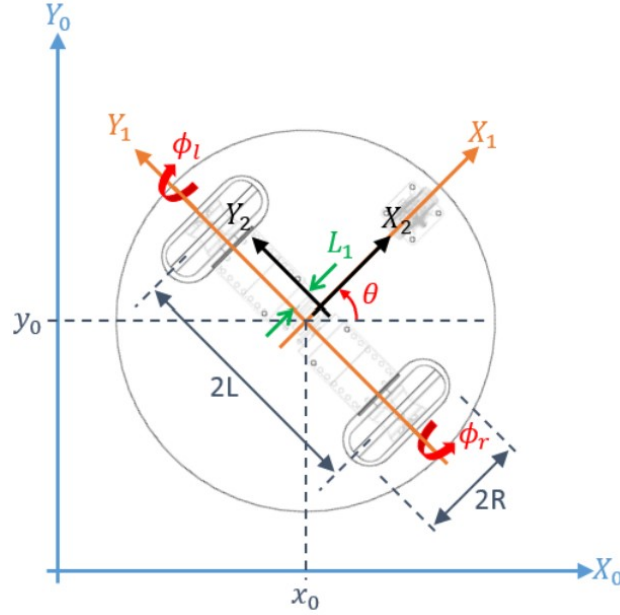


Figura 28: Diagrama general de un robot diferencial [20].

Donde se tiene $p = [x_0, y_0]^T$ como la posición del robot y se tiene θ como la orientación del mismo. Y se obtienen las entradas v y ω , que representan la velocidad lineal y angular respectivamente. Con esto se puede definir las siguientes ecuaciones para interpretar las velocidades de las ruedas con las velocidades del robot, donde tenemos la velocidad de la rueda derecha V_r y la velocidad de la rueda izquierda V_l [20].

$$v = \frac{V_r + V_l}{2} \quad \omega = \frac{V_r - V_l}{2}$$

6.9. Control de robot diferencial

Utilizando estrategias de control clásico se pueden definir un controlador ya sea para parquear en un punto meta o seguir una trayectoria específica. Planteando el sistema se pueden plantear dos tipos de error que son el error de orientación y el error de posición. Con estos errores definidos se puede aplicar un control PID convencional tanto a la velocidad lineal como a la velocidad angular asegurando llegar a la ubicación y orientación definida. El controlador PID normal queda de la siguiente forma:

$$v = PID_{e_p} = k_{pp}e_p + k_{ip} \int_0^t e_p(\tau) d\tau + k_{dp}\dot{e}_p$$

$$w = PID_{e_o} = k_{po}e_o + k_{io} \int_0^t e_o(\tau) d\tau + k_{do}\dot{e}_o$$

Estos controladores son útiles, pero el robot se mueve lento al estar cerca de la meta. Por lo que es recomendable alterar el PID de control de velocidad lineal. Este se altera utilizando



Figura 30: Ejemplo de robótica de enjambre [13].

6.11. PlatformIO

PlatformIO es una plataforma profesional que busca integrar herramientas para desarrollo de sistemas embebidos. Cuenta con extensiones para 46 diferentes plataformas, 25 frameworks, soporte para 1007 tarjetas de desarrollo y 11145 librerías. Cuenta con un Debugger integrado para poder interpretar y encontrar errores y con tests unitarios que permiten verificar el desempeño del proyecto [21].

6.12. Inventor

Inventor es un software CAD desarrollado por Autodesk que proporciona herramientas de calidad profesional enfocadas en el diseño mecánico 3D, documentación y simulación de productos [22].

6.13. Diseño de PCB

6.13.1. Altium Designer

Altium Designer es un software que brinda todas las herramientas necesarias para el proceso de diseño electrónico, desde el diseño del esquemático del circuito, simulación del mismo y fabricación de PCB [23].

6.13.2. Gerbers

Este es un tipo de archivo ASCII vector que contiene la información física del diseño de cada PCB. Contiene la información de las vías, tracks, pads, entre otras capas importantes, todas las características agrupadas en forma de vector [24].

6.14. OptiTrack

OptiTrack es un sistema de captura de movimiento que utiliza cámaras de gran velocidad para poder tener un sistema con gran precisión 3D. Utiliza identificadores denominados *markers* pasivos para reconocer el punto de interés y el arreglo de las cámaras en distintos ángulos permiten tener una recreación 3D del sistema [25].

6.15. TPU

El poliuretano de termoplásticos o TPU es un elastómero plástico fácil de moldear al derretirlo a temperaturas altas. Este cuenta con una gran durabilidad y flexibilidad. Es altamente usado en el mundo de impresión 3D debido a su alta elongación antes de ruptura, su resistencia a la abrasión, clima y luz UV [26].

Diseño electrónico de la plataforma

En cuanto al diseño electrónico de la plataforma no se tiene una gran diferencia a las otras dos fases. Se busca realizar un proyecto de tamaño reducido pero con componentes locales de bajo costo. A continuación se tiene la lista de materiales utilizados en el proyecto con su cantidad, designación y costo tanto unitario como total. Se busco lo más bajo de costo utilizando productos disponibles en cantidad en las electrónicas del país.

Elemento	Cantidad	Designación	Costo Q	Subtotal Q
ESP32	1	NA	125.00	125.00
Batería Effest 3500 mAh	2	18650	130.00	260.00
Driver motores	1	MC 33926	80.00	80.00
Encoder magnético	2	NA	50.00	100.00
Protección baterías	1	BMS 2S 8A	30.00	30.00
Regulador de voltaje	1	3.3v 3A	27.00	27.00
Motor con caja reductora	2	298:1	60.00	120.00
Placa de cobre	2	10 X 15 cm	19.00	38.00
LED	4	NA	1.00	4.00
Resistencias	4	330	1.00	4.00
Módulo acelerómetro	1	MPU9250	200.00	200.00
Filamento TPU	3 m	TPU	4.00	12.00
Filamento PETG	25 m	PETG	2.00	50.00
Total Q.				1050.00

Cuadro 2: BOM plataforma robótica

Se utilizan módulos para facilitar las mejoras futuras sin editar la plataforma completa. Se busca integrar la comunicación inalámbrica con el control y procesamiento de señales en un mismo microcontrolador. Se omite el uso de sensores para utilizar el sistema de captura de movimiento Optitrack, pero se deja habilitado un puerto para colocar cargas útiles en un futuro.

7.1. Selección de componentes

7.1.1. Unidad de control

Teniendo en cuenta las dos fases anteriores se pueden comparar los dos microcontroladores utilizados contra la propuesta a utilizar en esta fase. La propuesta que se tiene es el ESP32 por su versatilidad de integrar de forma nativa comunicación inalámbrica por medio de Bluetooth o WiFi junto a una capacidad de computo alta. Se realizó un *trade study* para poder seleccionar la mejor opción en base a criterios importantes. Entre estos criterios se tienen la comunicación inalámbrica, velocidad de procesamiento, módulos de comunicación, capacidad de memoria, entradas/salidas y voltaje disponible.

Criterio	Peso	Rango		ESP32	ESP8266	PIC32MX250F128B	Teensy 3.2
Comunicación inalámbrica	5	Si	No	1	1	0	0
Velocidad	1	0.12	0.16	4	3	1	2
Módulos de comunicación	4	7	0	6	1	4	6
Capacidad de memoria	3	500kB	0kB	4	1	2	2
Entradas/salidas	2	32	0	4	1	2	3
Voltaje		3.3 - 5 V	Ninguno	2	1	1	2
Puntuación				21	8	10	15

Cuadro 3: Estudio de características de microcontroladores

En cuanto el primer criterio se evaluó con un punto si cuenta con comunicación inalámbrica de lo contrario no se le otorgan puntos. En los criterios relacionados a la velocidad de procesamiento, entradas/salidas, módulos de comunicación y de capacidad de memoria se asignaron puntos en orden colocando más puntos al que tiene mayor capacidad y menos puntos al de menor capacidad. Para el criterio de voltaje disponible se asignaron 2 puntos si se tienen dos tipos de voltajes disponibles y se asigna 1 punto si sólo maneja un tipo de voltaje.

Con los resultados del *trade study* realizado se tiene que se utilizará el módulo ESP32. Inclusive comparando una opción combinando el ESP8266 y el PIC32MX250F128B aún se tienen más puntos a favor del ESP32. Este microcontrolador se puede programar en lenguaje C usando el SDK de Espressif o con Arduino basado en C++. En este caso se programó en Visual Studio Code utilizando el SDK de Espressif y PlataformIO. El empaquetado utilizado es el WROM DEV KIT1 adquirido en una electrónica local.

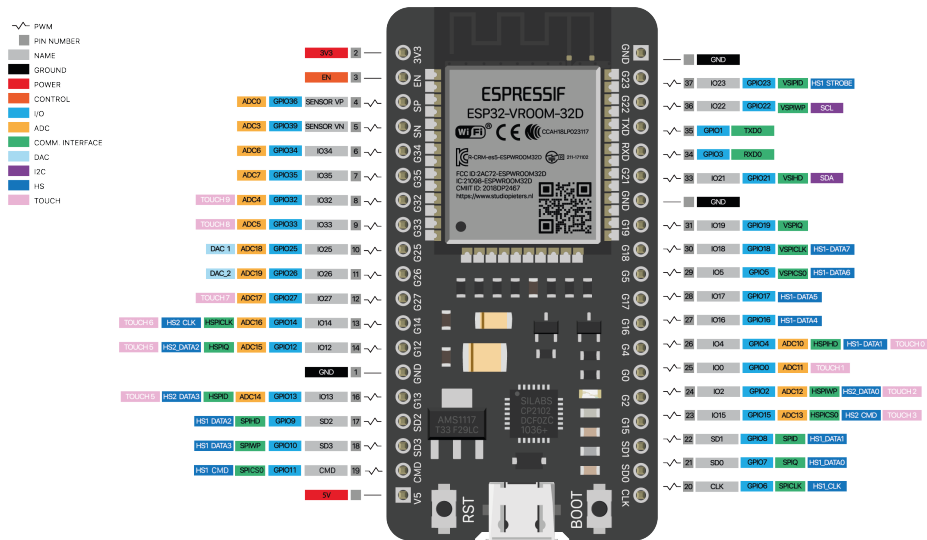


Figura 31: Microcontrolador ESP32.

7.1.2. Driver de motores

Para el driver de los motores se seleccionó el MC33926 de Pololu. Este es un driver dual por lo que es perfecto para las aplicaciones en robots diferenciales. Se seleccionó este driver de motor para poder alimentar ambos motores con una fuente entre 5V y 28V. Permite tener un control individual de los motores con una señal PWM de hasta 20kHz.

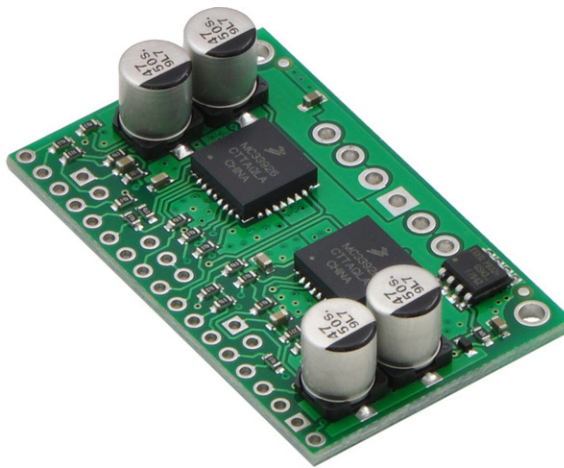


Figura 32: Driver dual MC33926.

7.1.3. Encoders

Para leer la velocidad y posición de los motores a tiempo real se utiliza un encoder. En este caso se utilizó un encoder magnético para poder leer las revoluciones de cada motor. La lectura de cada encoder requiere dos pines del microcontrolador, y su programación la

se explica a fondo en el capítulo de programación. Estos encoders se alimentan con 3.3V y emiten pulsos cuadrados para verificar el movimiento del motor.

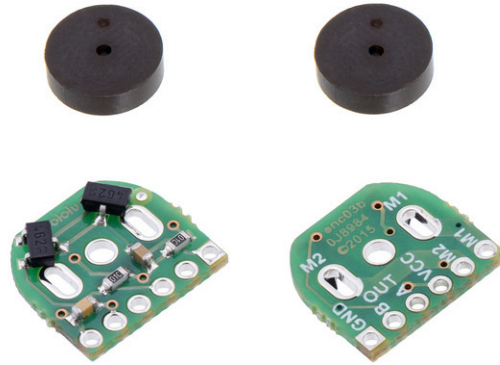


Figura 33: Driver dual MC33926.

7.1.4. Alimentación

La plataforma requiere un nivel entre 8.4V y 6V de alimentación para los motores y una alimentación de 3.3V para alimentar el microcontrolador, MPU, indicadores y encoders. Esto creó una necesidad de tener dos fuentes de alimentación o una con un regulador de voltaje. En el capítulo de potencia se especifica cómo se solucionó esta problemática.



Figura 34: Baterías Efest 18650.

7.2. Potencia de la plataforma

Esta fue una de las problemáticas más grandes en la fase anterior. Esto porque los convertidores lineales se calentaban en sobre-medida, el cual afectaba el desempeño de algunos componentes.

7.2.1. Configuración de baterías

Debido a los niveles de voltaje se seleccionó utilizar dos baterías 18650 en serie. Estas baterías tienen un voltaje nominal de 3.7V con mínimo de 3.0V y un máximo de 4.2V, por lo que al tenerlas en serie se cumple con el requisito de voltaje para los motores. Las baterías utilizadas tienen una capacidad de 3000 mAh y realizando el *power budget* de los componentes utilizados se puede ver la duración de estas.

Subsistema	Componentes	Potencia (mW)	Cantidad	Subtotal (mW)
Indicadores	Indicador batería	42	1	42
	Switch	2	1	2
	LEDs	200	4	800
Potencia	Regulador de voltaje	50	1	50
Control	ESP32	240	1	240
Puntuación	Encoder	20	2	40
	MPU8250	40	1	40
	Carga útil	500	1	500
Total a 3.3 V				1714
Actuadores	Motores	1100	2	2200
Total a 7.4V				2200

Cuadro 4: Power Budget plataforma robótica.

Revisando el *power budget* se sabe que el consumo en amperios de todo el sistema fue de 818 mA. Las baterías cuentan con una capacidad de 3000 mAh por lo que dividiendo la capacidad de estas entre la corriente consumida se tiene un tiempo máximo de duración de la carga. Al realizar este cálculo se sabe que la plataforma tendrá una autonomía de 3 horas y 40 minutos, lo cual es satisfactorio ya que otorga una autonomía ideal para posibles experimentos en laboratorios didácticos u otras aplicaciones de robótica enjambre. Esta prueba se realizó dejando las baterías cargadas al 100% y dejando la plataforma corriendo hasta agotar la batería y efectivamente se tuvo un tiempo promedio cercano a las 3 horas.

7.2.2. Regulador de voltaje

La problemática del nivel de voltaje para alimentar al ESP32 es que se debe garantizar un nivel de 3.3V constante con gran exactitud. Este regulador en la fase anterior dio el problema de sobrecalentamiento, por lo que para esta fase se utilizó un convertidor Buck en lugar de un convertidor lineal. El convertidor Buck utilizado es un circuito diseñado para dar una salida constante y exacta de 3.3V.

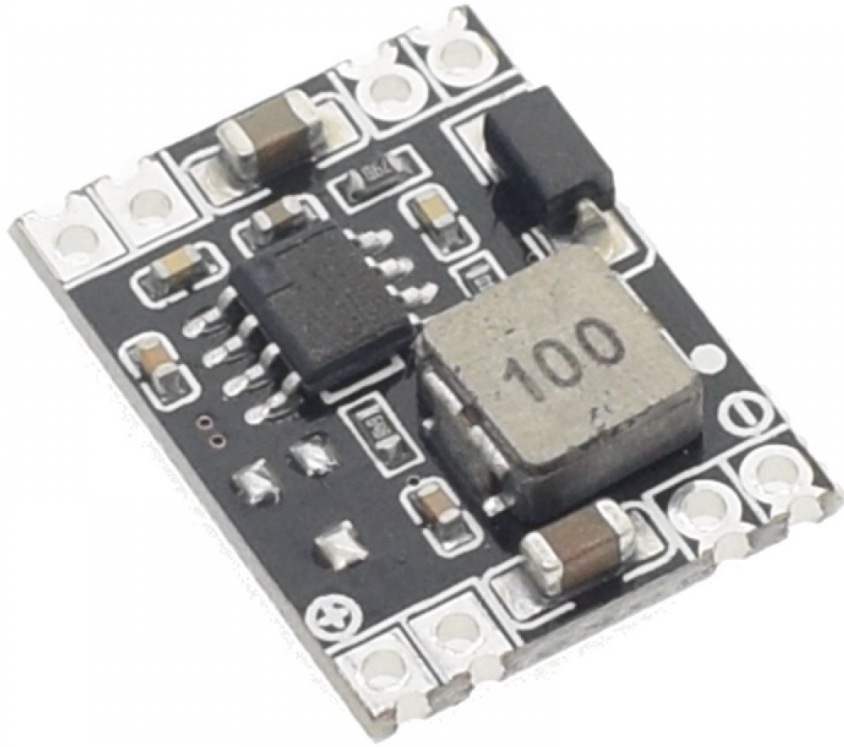


Figura 35: Regulador de voltaje a 3.3V.

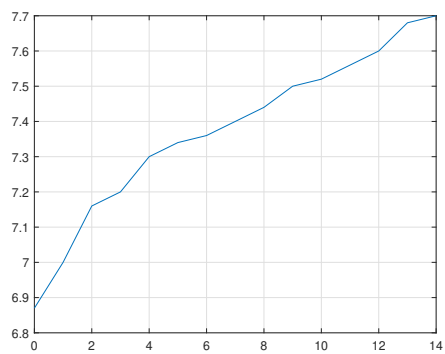
7.3. Interacción física con la plataforma

7.3.1. Encendido y apagado

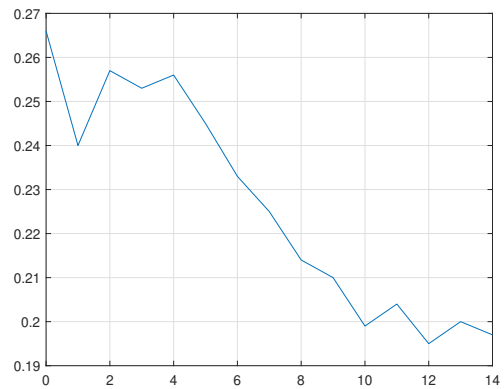
Para facilitar el apagado y encendido de la plataforma se implemento un Switch que corta la alimentación de todo el sistema evitando la descarga no deseada de las baterías. Este switch se colocó en la tapa del diseño para poder tener un fácil acceso.

7.3.2. Indicadores LED

Se cuenta con un indicador LED el cual indica el porcentaje de batería restante en intervalos de 100 %, 75 %, 50 %, 25 % y 0 %. También se dejaron 4 pines y un pin de referencia listos para implementar LEDs extras, buzzers o cualquier otro actuador lógico. Se colocaron LEDs de tres colores teniendo un LED rojo, LED amarillo y un LED verde, estos indicarán cuando este tenga un error, cuando se este realizando un proceso o cuando el robot llegue a



(a) Voltaje en carga.



(b) Corriente en carga.

Figura 38: Curvas de corriente y voltaje en carga de baterías seleccionadas y fuente a 8.4V

Diseño mecánico de la plataforma

El elemento más importante para el diseño mecánico de la plataforma es la placa del circuito eléctrico. Las dimensiones de la placa más grande restringen el diámetro de la plataforma final. Las estrategias de manufactura que se utilizaron son impresión 3D y corte láser. La forma determinada para un robot diferencial es un círculo ya que facilita el análisis para la implementación de controladores.

Se diseñó todo el chasis de la plataforma para proteger los circuitos internos y tener un acabado estético de calidad. Se implementó una forma modular para permitir la flexibilidad del uso de cargas útiles de diferentes tamaños. El diseño se realizó en Autodesk Inventor 2021 buscando tener diseños sencillos para facilitar la fabricación de replicas a gran escala. A continuación se puede ver el ensamblaje final realizado en inventor. Las dimensiones de cada parte diseñada se encuentran en sus planos los cuales están en Anexos.

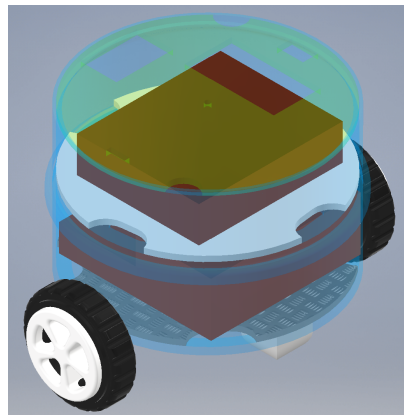
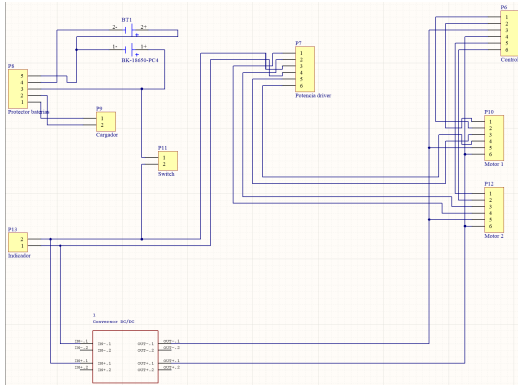


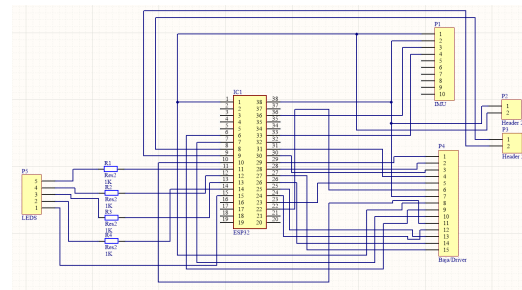
Figura 39: Ensamblaje final de la plataforma robótica.

8.1. Placas electrónicas

Utilizando *Altium Designer* se diseñaron dos placas, una de potencia y otra de control. En la placa de potencia se incluye el switch principal, las baterías, el regulador, el driver de motores y el indicador de batería. En la placa de control se incluye el microcontrolador ESP32, la MPU9250 y LEDs indicadores de la plataforma. Se utilizó una estrategia de fabricación de una sola cara, lo cual facilitó el proceso de soldadura y evita la colocación de vías.



(a) Esquemático de potencia.



(b) Esquemático de control.

Figura 40: Vista de esquemáticos utilizados para las placas diseñadas.

Las placas diseñadas se encuentran en anexos. La placa más grande fué un rectángulo de $84 \times 85 \text{ mm}$ por lo que se tiene un área superficial de 7140 mm^2 . Comparando estas placas con la fase anterior, se tiene que la placa es circular de un radio de 50 mm por lo que en cuanto a área se tiene una reducción de 714 mm^2 . Esto afirma que se logró el objetivo de reducir el tamaño de la plataforma.

8.2. Diseño de rueda

8.2.1. Diseño y fabricación de aro

El diseño del aro toma en cuenta el eje de salida del motor por lo que se tomó como referencia. Este se extendió para permitir que la rueda no estuviese extremadamente pegada a la plataforma. El aro se asemeja a un aro de automóvil para poder remover material teniendo como resultado un aro más liviano y más rápido de fabricar. Se dejó un área de sujeción a presión para la llanta facilitando el posible cambio ante llantas dañadas. Se fabricó el aro mediante impresión 3D utilizando filamento PLA. Este material permite una buena resistencia mecánica y su bajo precio permite tener una gran calidad a un bajo costo de fabricación. El tiempo de fabricación de los dos aros fue de dos horas y treinta minutos, con un acabado medio y un relleno del 20%. El aro tiene un diámetro total de 40 mm y un eje que se extiende por 15 mm .

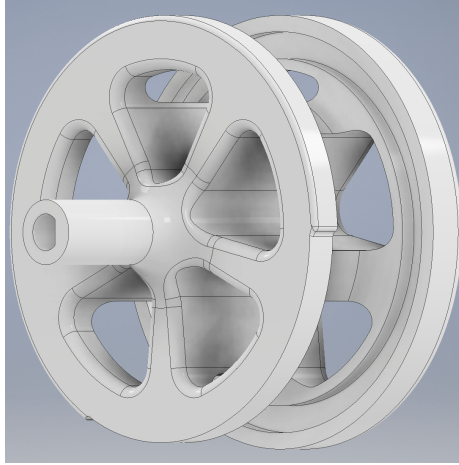


Figura 41: Diseño final del aro.

8.2.2. Diseño y fabricación de llanta

La llanta se diseñó tomando en cuenta el aro. Su diseño incorporó surcos paralelos al eje central del aro para aumentar su tracción asegurando un buen funcionamiento en la mesa de trabajo tratando de disminuir el espacio muerto del motor. Su fabricación se realizó en 3D utilizando el filamento TPU y un relleno en el patrón de cruz 3D asegurando una mayor flexibilidad. Su diseño y el material utilizado brinda rigidez la cual permite mantener la integridad y tamaño de la llanta sin importar su relleno. Su fabricación tomó un tiempo de una hora y diez minutos por cada llanta. Sus dimensiones finales fueron $53.664mm$ de diámetro y $15.5mm$ de ancho.

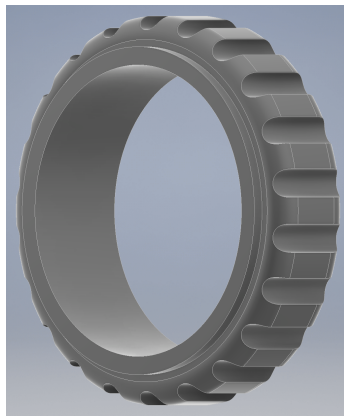
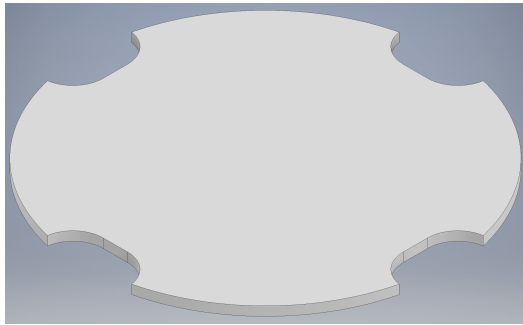


Figura 42: Diseño final de la llanta.

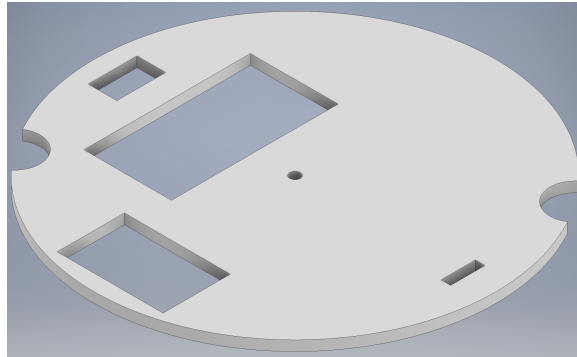
8.3. Cuerpo de plataforma

Una vez se tiene la rueda y las placas electrónicas se empezó a diseñar el cuerpo final del robot. En este apartado se utilizaron las dos estrategias de fabricación disponibles. Se

utiliza el corte láser para fabricar las plataformas donde se acomodan las placas diseñadas y la tapa donde se acomodan indicadores, actuadores manuales, puertos disponibles y el microcontrolador. Estas partes utilizan una plancha de acrílico de 1/8 de pulgada. El corte láser permite realizar estas piezas de manera rápida en cuestión de minutos. El diseño de estas piezas se basa en un círculo de 120mm de diámetro. Se colocaron ranuras donde puedan pasar los cables comunicando las dos placas y se optimiza el flujo del aire para evitar el sobrecalentamiento de componentes electrónicos.



(a) Plataforma diseñada.



(b) Tapa diseñada.

Figura 43: Partes diseñadas y fabricadas en corte láser.

El cuerpo modular se basa en cilindros donde puedan ingresar sin problema las plataformas anteriormente mencionadas. Se fabricaron mediante impresión 3D por su geometría y permite reducir el ancho de las paredes para disminuir el peso de la plataforma. Estos cilindros se acoplan mediante ranuras del tamaño exacto y por efecto de gravedad permanecen en su sitio. Esta forma de acople permite un fácil acceso a las dos placas electrónicas.

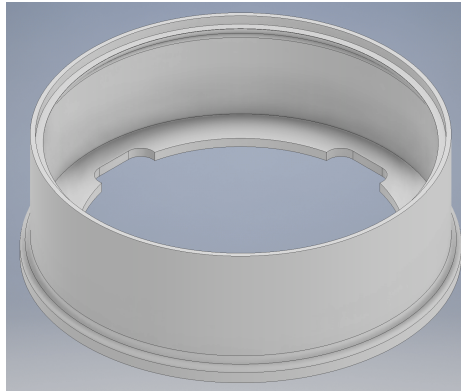
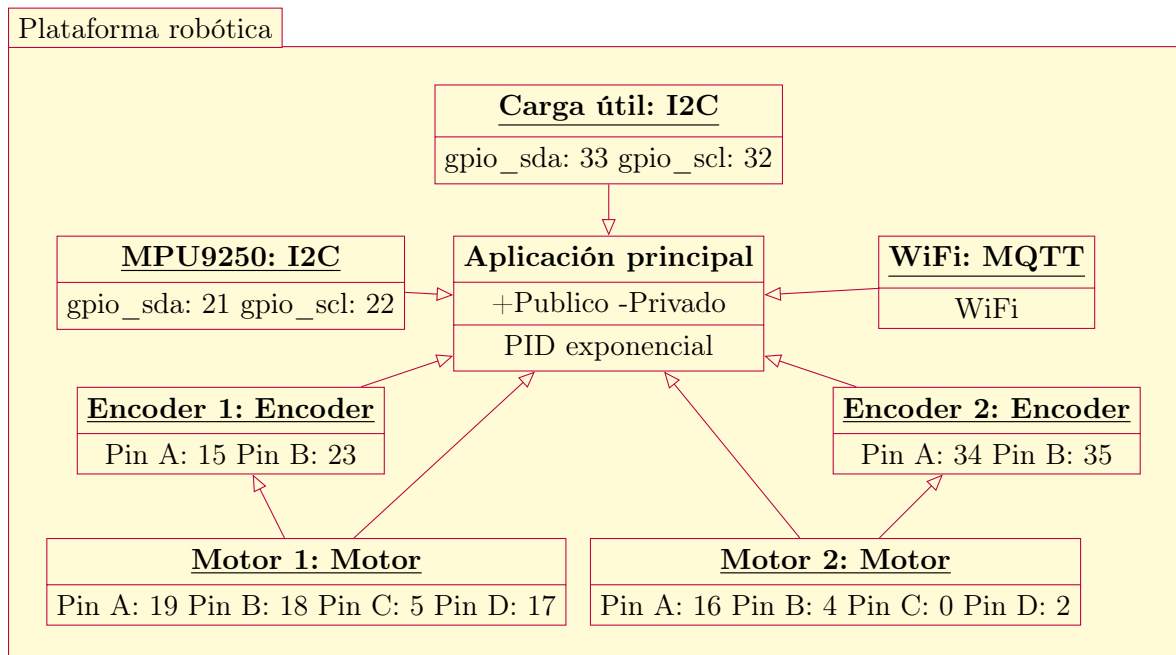


Figura 44: Cilindro diseñado.

Programación de la plataforma

Las librerías fueron programadas en el lenguaje C utilizando el SDK de Espressif. Se utilizó como base la programación orientada a objetos. Se realizaron distintas librerías para poder interactuar con los diversos módulos, estas librerías se suben al repositorio del ecosistema robótico para poder ser reutilizado, este repositorio se encuentra representado en Anexos. Se utiliza la función *ESP_ERROR_CHECK*, la cual representa un *Assert* el cual regresa un valor de OK si no se tiene ningún error y de lo contrario puede regresar alguna excepción o errores.



9.1. Librería de encoders

Esta librería crea un objeto por encoder. Este objeto almacena las características como la dirección, pines de lectura y estrategias de pasos. Se utilizan dos tipos de pasos para estos encoders. El primero es a paso completo para poder contar 12 vueltas por vuelta del motor también se tiene una tabla para medios pasos para mayor exactitud. Entre las funciones creadas se tiene la inicialización de los encoders, lectura de los mismos para obtener dirección y posición actual, reiniciar el encoder, cambiar la orientación de giro y crear nuevas tareas.

Encoder
Estado: [Dirección, Posición] Dirección: [Sin dirección, derecha, izquierda] Posición: Posición actual del encoder Tabla de pasos: Ver cuadro 1 Pin A: GPIO de señal A Pin B: GPIO de señal B Fila: Fila actual de la tabla Cola de tareas: Queue
<code>encoder_init(encoder_info, pin_a, pin_b)</code> <code>encoder_activar_medio_paso(encoder_info, enable)</code> <code>encoder_cambio_direccion(encoder_info)</code> <code>encoder_deiniciar(encoder_info)</code> <code>encoder_crear_cola()</code> <code>encoder_elegir_cola(encoder_info, queue)</code> <code>encoder_obtener_estado(encoder_info, Estado)</code> <code>encoder_reset(encoder_info)</code>

9.2. Librería de motores

Esta librería se complementa en gran manera con la librería de encoders. Se tomó en cuenta la documentación del driver seleccionado para poder activar los motores. El microcontrolador ESP32 cuenta con dos módulos llamados MCPWM el cual genera señales PWM para motores. Se sabe que el driver necesita la señal del PWM y la señal negada de la misma a su vez tiene dos pines de entrada para determinar el funcionamiento del motor. En la librería se utilizan estos dos pines para determinar la dirección de giro del motor. En la librería se crea un objeto que representa el motor en el cual se tienen los cuatro pines utilizados por motor, el módulo a utilizar, el temporizador a utilizar y el ciclo de trabajo para la señal de PWM.

Motor
Estado: [Dirección, Ciclo de trabajo] Dirección: [Sin dirección, derecha, izquierda] Pin A: GPIO de señal A PWM Pin B: GPIO de señal B $P\bar{W}M$ Pin C: GPIO de señal C IN1 Pin D: GPIO de señal D IN2 Mod: Modulo MCPWM a utilizar
motor_init(motor_info, mod, pin_a, pin_b, pin_c, pin_d) motor_adelante(motor_info, mod, duty_cycle) motor_atras(motor_info, mod, duty_cycle) motor_paro(motor_info, mod)

9.3. Librería de I2C

Para poder utilizar el protocolo de comunicación I2C se pueden configurar hasta 2 canales distintos. Cada canal se puede configurar en su modo dependiendo si es esclavo o maestro, los pines y la velocidad de reloj. Esta librería no solo se utilizó para habilitar el puerto libre de la carga útil sino también para comunicarse con la MPU9250.

Motor
i2c_num: Numero de red i2c periph_adress: Dirección i2c de dispositivo externo reg_adress: Dirección de almacenar datos gpio_sda: GPIO de señal SDA gpio_scl: GPIO de señal SCL data: Datos enviados/recividos data_len: Tamaño de datos enviados/recividos
i2c_master_init(i2c_num, gpio_sda, gpio_scl) i2c_write_bytes(i2c_num, periph_adress, reg_adress, data, data_len) i2c_write_byte(i2c_num, periph_adress, reg_adress, data) i2c_write_bit(i2c_num, periph_adress, reg_adress, bit, length, value) i2c_write_bits(i2c_num, periph_adress, reg_adress, bit, value) i2c_read_bytes(i2c_num, periph_adress, reg_adress, data, data_len) i2c_read_byte(i2c_num, periph_adress, reg_adress, data) i2c_read_bit(i2c_num, periph_adress, reg_adress, bit, result)

Para utilizar la MPU9250, se utiliza el protocolo I2C para extraer los valores del magnetómetro, giroscopio y acelerómetro. Este módulo debe calibrarse cada que se tenga en una posición final por lo que se calibró al colocarse en el ensamblaje final.

9.4. Librería de MQTT

Se utiliza la librería desarrollada en la tesis de Camilo Perafán de comunicación MQTT pensada para el ecosistema robotico de la Universidad [\[11\]](#). Esta librería contiene funciones que automatizan la conexión del agente al servidor MQTT. Otra de las funciones utilizadas es la de recepción de datos. En esta función se obtiene un conjunto de datos que corresponden a lo leído por el Optitrack sobre el agente específico. Esos datos incluyen la posición y ubicación en cuaterniones. Esta función se utiliza en cada iteración del ciclo principal para obtener los datos requeridos de posición, orientación y un punto deseado.

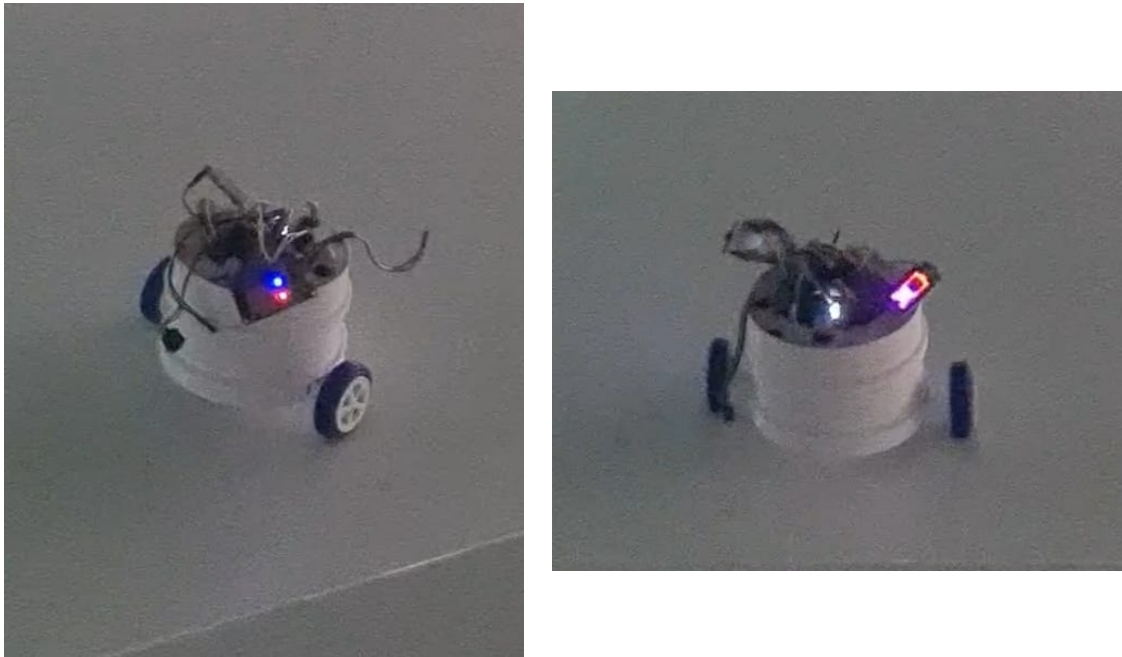


Figura 45: Diseño final de la plataforma.

10.1. Pruebas de Payload

Se realizaron pruebas con el puerto diseñado para una carga útil. Para probar el funcionamiento de este puerto se alimentó un Arduino Leonardo programado como un esclavo de comunicación I2C. Se utilizó el módulo 1 del ESP32 para tener un segundo bus de comunicación totalmente independiente del utilizado por la MPU.

10.2. Pruebas motores y encoders

Se utilizaron distintos ciclos de trabajo en ambos motores para cambiar la velocidad independiente de cada uno de estos. A su vez se utilizó el puerto serial en la computadora para ver las lecturas de los encoders. La lectura de los ticks del encoder permiten encontrar la velocidad del motor y posteriormente se podría realizar una fusión de sensores con los datos del Optitrack mediante un filtro de Kalman para poder tener un mejor control de la velocidad independiente de cada motor.

10.3. Pruebas del sistema de potencia

Colocando las baterías descargadas se tomaron puntos del voltaje y corriente a lo largo de una hora. Esto para poder tener un estimado de la carga de las baterías. Se alimentaron directamente de una fuente de alimentación a 8 V DC. Las curvas de carga se pueden ver en las **figura ??**.

Se realizaron mediciones de temperatura de componentes y compartimientos del robot para asegurarnos que el sistema de potencia diseñado no aumente la temperatura del sistema. Se utilizó un multímetro, ya que no se cuenta con una cámara infrarroja, y los resultados dejan ver que el convertidor utilizado es el componente que más se calienta. Este tiende a estar dos grados centígrados más que la temperatura del ambiente.

10.4. Pruebas de comunicación MQTT

Se colocó el robot libre en la plataforma con los marcadores necesarios para que el sistema de captura de movimiento lo detectara. Al moverse por la plataforma se detectaba la posición y orientación del robot para poder enviar estos datos por la comunicación MQTT. El cliente, en este caso el robot diseñado, recibe el conjunto de datos y luego los separa para poder distinguir entre las coordenadas de x , y y z junto a su orientación en cuaterniones.

Se realizó una prueba en la cual se coloca la plataforma robótica en un punto y las ruedas deben parar el funcionamiento al estar en un punto específico. Esto se realizó sujetando al robot ya que la velocidad de este es mayor a la velocidad de la comunicación con el servidor.

- Se optimizó satisfactoriamente el sistema de potencia de la plataforma dotando a la misma con una autonomía cercana a la 3 horas.
- Se eliminó la problemática del sobrecalentamiento de la plataforma mediante el regulador buck.
- Se cuenta con una reducción de $714mm^2$ en el plano xy en cuanto al tamaño de las placas electrónicas utilizadas.
- Se diseñó un cuerpo externo al robot el cual protege el interior del mismo y mejora el apartado estético de la plataforma.
- Se utilizó adecuadamente el microcontrolador ESP32 como unidad de control y comunicación otorgando una plataforma unificada y robusta.

CAPÍTULO 12

Recomendaciones

- Se recomienda utilizar diversos controladores de movimiento para robots diferenciales para poder ampliar las capacidades de la plataforma.
- Se recomienda pasar de un diseño basado en módulos a una placa propia de la Universidad con componentes de superficie para poder ahorrar espacio y reducir el tamaño de la plataforma.
- Se recomienda aumentar la holgura que se presenta entre piezas de la plataforma para que estas se unan sin dificultad.
- Se recomienda mejorar la velocidad de comunicación de datos para poder aumentar la velocidad.

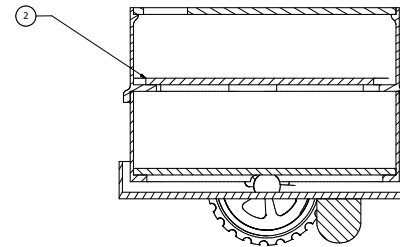
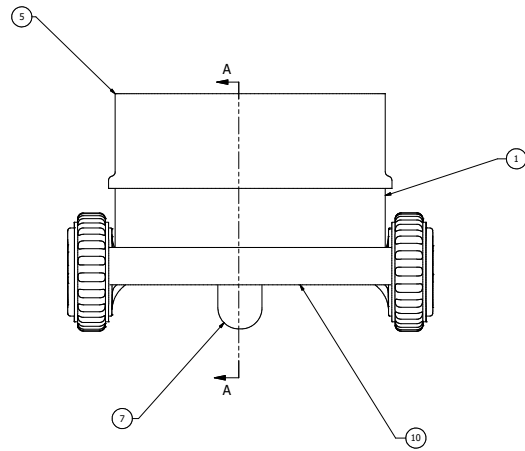
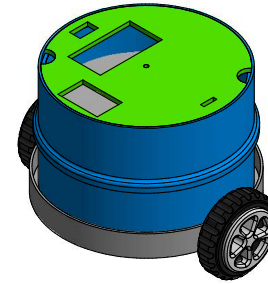
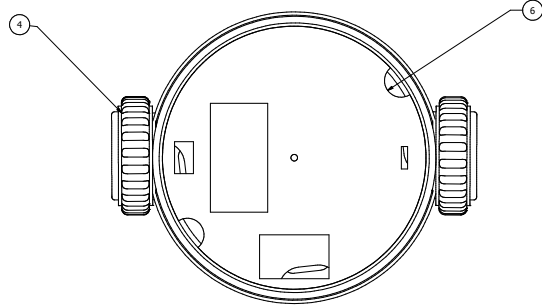
-
- [1] F. Arvin y M. Bekravi, “Endoreless position and error correction techniques for miniature mobile robots,” *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 21, n.º 1631-1645, págs. 1-15, 2013.
 - [2] F. Arvin, K. Samsudin y A. Ramli, “Development of a Miniature Robot for Swarm Robotic Application,” *International Journal od Computer and Electrical Engineering*, vol. 1, n.º 4, págs. 1-7, 2009.
 - [3] A. Ozgur, S. Lemaignan, W. Johal, M. Beltran, M. Briod, L. Pereyre, F. Mondada y P. Dillenbourg, “Cellulo: Versatile Handheld Robots for Education,” *Centre for Robotics and Neural Systems, Plymouth University, U.K*, págs. 1-9, 2017.
 - [4] G. Capraro, O. Arras y R. Siegwart, “The Autonomous Miniature Robot Alice: from Prototypes to Applications,” *Swiss Federal Institute of Technology Lausanne*, págs. 1-6, 2000.
 - [5] S. Kernbach, R. Thenius, O. Kernbach y T. Schmickl, “Re-embodiment of Honeybee Aggregation Behavior in an Artificial Micro-Robotic System,” *International Society for Adaptive Behavior*, vol. 17(3), n.º 237-259, págs. 1-24, 2013.
 - [6] F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klapotocz, S. Magnenat, J.-C. Zufferey, D. Floreano y A. Martinoli, “The e-puck, a Robot Designed for Education in Engineering,” *Ecole Polytechnique Federale de Lausanne, Switzerland*, págs. 1-7, 2007.
 - [7] A. Turgut, F. Gökce, H. Celikkanat, L. Bayindir y E. Sahin, “Kobot: A mobile robot designed specifically for swarm robotics research,” *Middle East Technical University*, págs. 1-16, 2007.
 - [8] M. Rubenstein, C. Ahler y R. Nagpal, “Kilobot: A Low Cost Scalable Robot System for Collective Behaviors,” *IEEE International Conference on Robotics and Automation*, págs. 1-6, 2012.
 - [9] F. Mondada, G. Pettirano, I. Kwee, D. Floreano, J.-L. Deneubourg, S. Nolfi, L. Gambardella y M. Dorigo, “Swarm-Bot: A New Distributed Robotic Concept,” *Autonomous Robots*, vol. 17, n.º 193-221, págs. 1-29, 2004.

- [10] J. McLurkin, J. Rykowski, M. Quillan y A. Lynch, "Using Multi-Robot Systems for Engineering Education: Teaching and Outreach With Large Numbers of an Advanced, Low-Cost Robot," *IEEE transactions on education*, vol. 56, n.º 1, págs. 1-10, 2013.
- [11] E. Hernández, J. Del Cid, J. Mérida, O. Wantland, V. Villegas, A. Orozco y R. Ajtún, "Reingeniería de Megaproyectos Fase 1," 2017.
- [12] C. Lima, "Diseño e implementación de una nueva plataforma móvil para aplicaciones en robótica de enjambre," 2018.
- [13] J. Castañeda, "Diseño e implementación de una red de comunicación WiFi e interfaz gráfica para una mesa de pruebas de robótica de enjambre," 2020.
- [14] J. Blázquez, *Introducción a los sistemas de comunicación inalámbricos*. Universitat Oberta de Catalunya, 2016.
- [15] A. Maier, A. Sharp e Y. Vagapov, "Comparative Analysis and Practical Implementation of the ESP32 Microcontroller Module for the Internet of Things," *Glyndwr University, Plas Coch, Mold Road*, págs. 1-6, 2017.
- [16] Pololu, "Dual MC33926 Motor Driver Carrier," 2021. dirección: <https://www.pololu.com/product/1213>.
- [17] S. Basler, *Encoder und Motor-Feedback-Systeme Winkellage und Drehzahlerfassung in der industriellen Automation*. Springer Vieweg, 2016.
- [18] G. Quintero y J. Velez, *Diseño Topológico para Red de Datos Emcali*. Cooperación Universidad Autónoma de Occidente, 1989.
- [19] D. Pérez, *Sistemas Embebidos y Sistemas Operativos Embebidos*. Centro de Investigación en Comunicación y Redes, 2009.
- [20] D. Hernández, J. Eguibar, C. Cortés y J. Reyes, "Diseño, construcción y modelo dinámico de un robot móvil de tracción diferencial aplicado al seguimiento de trayectorias," *Memorias del XXII congreso internacional anual de la SOMIM*, págs. 1-7, 2017.
- [21] PlatformIO, "What is PlatformIO?," 2014. dirección: <https://docs.platformio.org/en/latest/what-is-platformio.html>.
- [22] Autodesk, "Inventor," 2021. dirección: <https://bit.ly/3Ck5k8a>.
- [23] A. Limited, "Altium," 2021. dirección: <https://www.altium.com/altium-designer/>.
- [24] V. |. E. Team, 2019. dirección: <https://www.vse.com/blog/2019/10/29/gerber-files-explained-understanding-their-role-in-pcb-manufacturing/>.
- [25] I. D. O. NaturalPoint, "Optitrack," 2021. dirección: <https://optitrack.com/cameras/primex-41/>.
- [26] S. Omnexus, "Complete Guide on Thermoplastic Polyurethanes (TPU)," 2021. dirección: <https://omnexus.specialchem.com/selection-guide/thermoplastic-polyurethanes-tpu>.

14.1. Planos de construcción

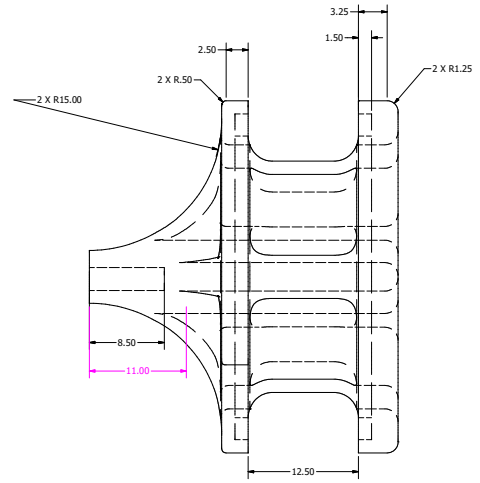
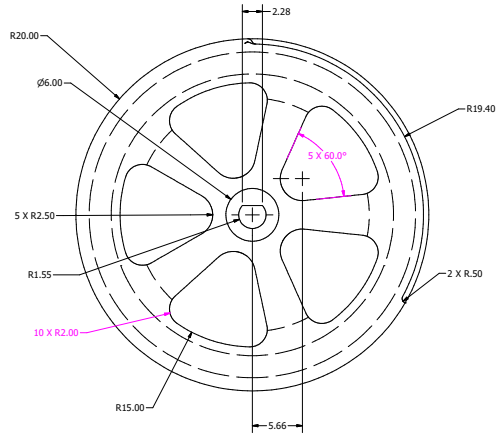
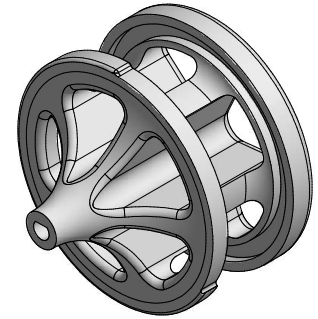
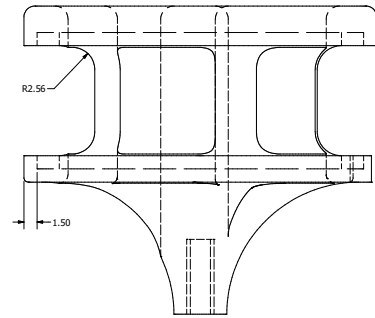
A continuación se pueden observar los planos realizados en Inventor de las piezas finales de la plataforma robótica diseñada.

LISTADO DE PARTES			
ITEM	CANTIDAD	NOMBRE DE PARTE	DESCRIPCIÓN
1	1	CILINDRO 1	CORTE LASER
2	2	CIRCUITO	CORTE LASER
3	2	Motor	MOTOR POLOLU
4	2	Rueda	IMPRESIÓN 3D INTEGRANDO RUEDA Y ARG. DISEÑADOS
5	1	CILINDRO 2	IMPRESIÓN 3D
6	1	TAPA	CORTE LASER
7	1	RUEDA LOCA	COMPRADA
10	1	FONDO	IMPRESIÓN 3D

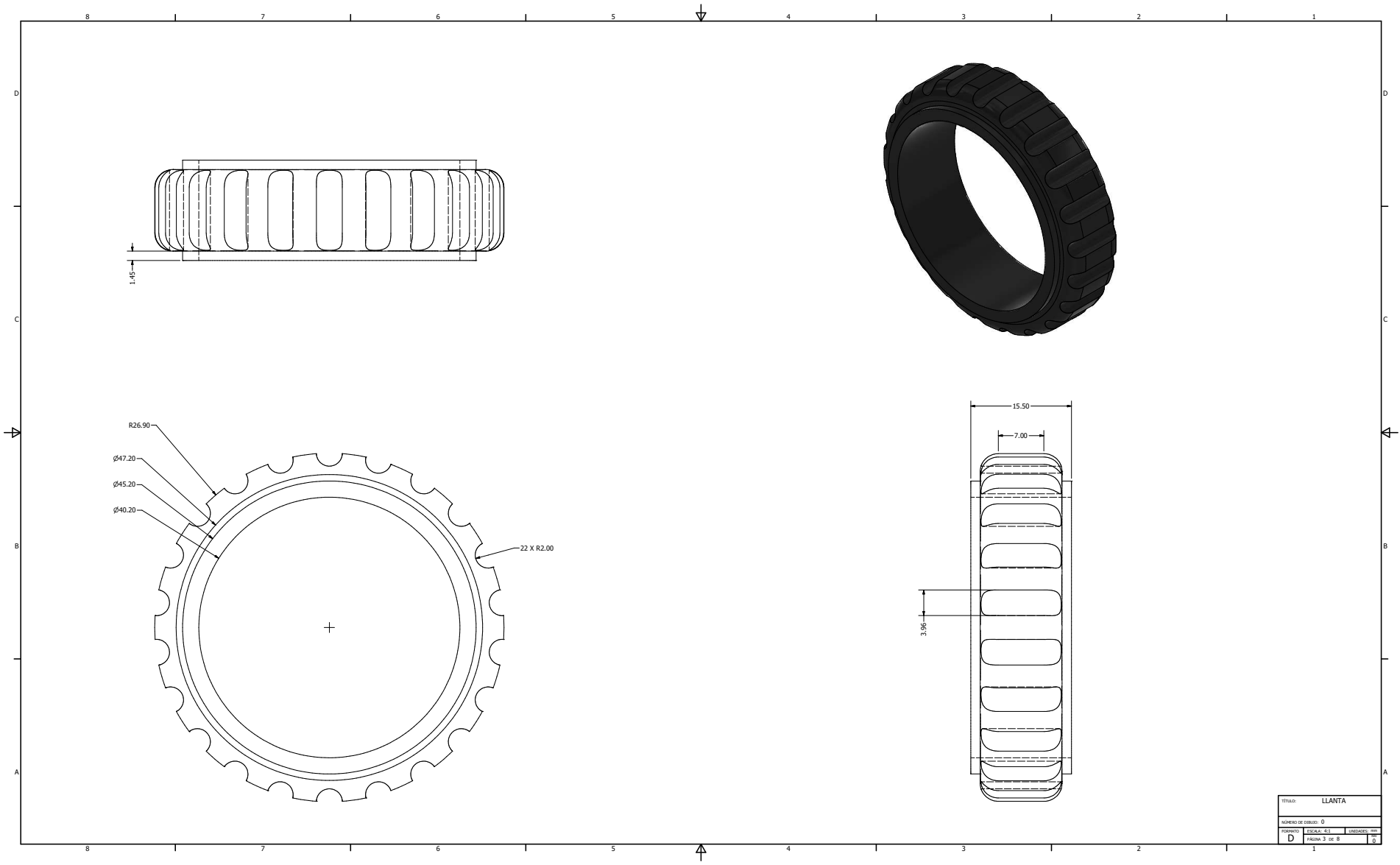


SECTION A-A
SCALE 1 : 1

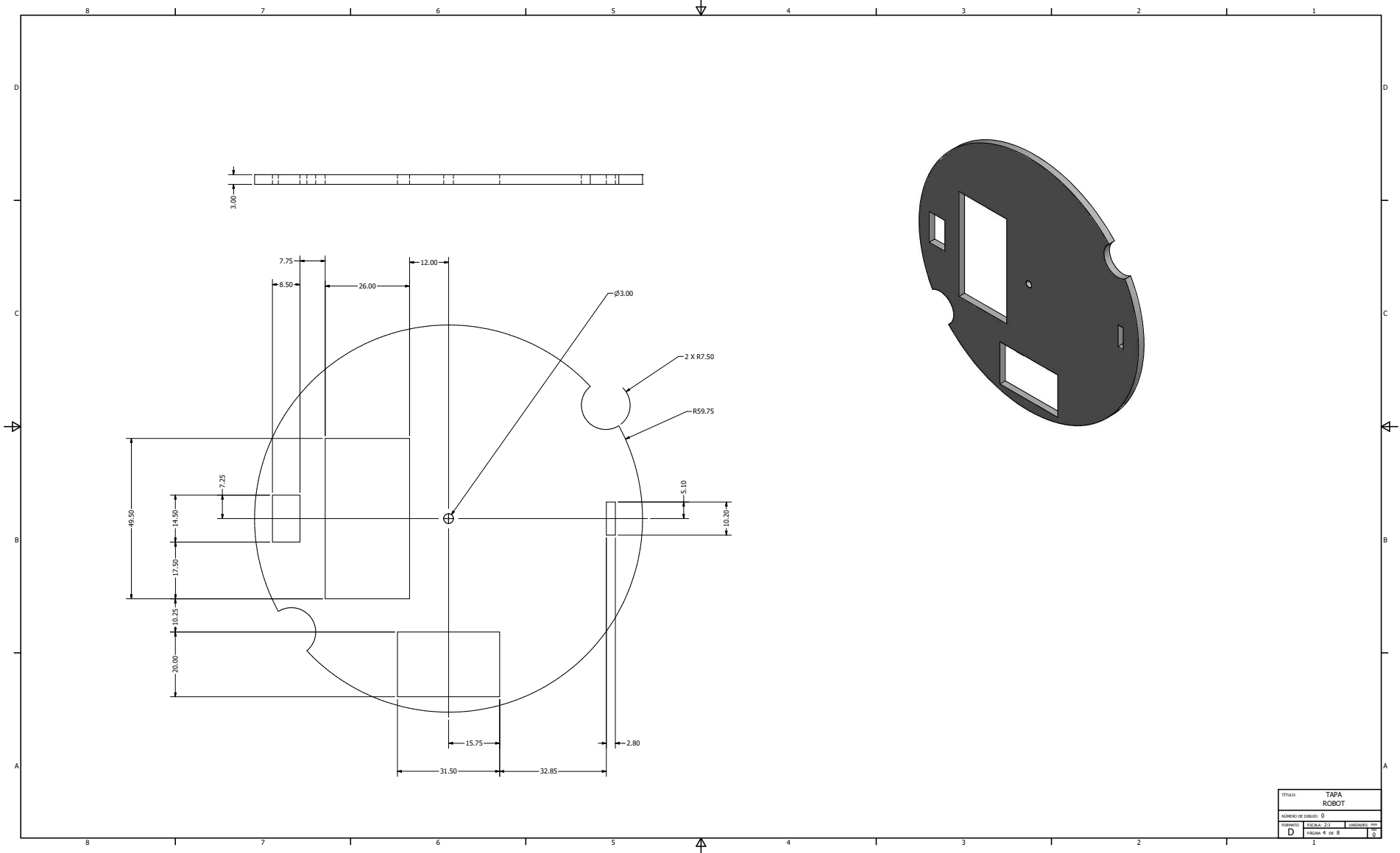
INSTITUCIÓN UNIVERSIDAD DEL VALLE DE GUATEMALA ESCUELA DE INGENIERÍA EN MECÁNICA	TÍTULO DEL PROYECTO ENSAMBLE FINAL ROBOTAT	UNIVERSIDAD DEL VALLE DE GUATEMALA ESCUELA DE INGENIERÍA EN MECÁNICA CARRERAS DE LA AV. 100, ZONA 10, GUATEMALA, GUATEMALA TEL: (502) 24400000 WWW.UDELV.GU
	NOMBRE DEL ALUMNO JOSE ESTRADA	
FECHA DE ENTREGA DEL PROYECTO 20/09/2021	FECHA DE ENTREGA DEL TUTOR 18/09/2021	NÚMERO DE OBRERO 0
TÍTULO DEL PROYECTO ENSAMBLE FINAL ROBOTAT	ESCALA 1 : 1	UNIDADES DEL DISEÑO 0
FECHA DE ENTREGA DEL PROYECTO 20/09/2021	FECHA DE ENTREGA DEL TUTOR 18/09/2021	PÁGINA 1 DE 8



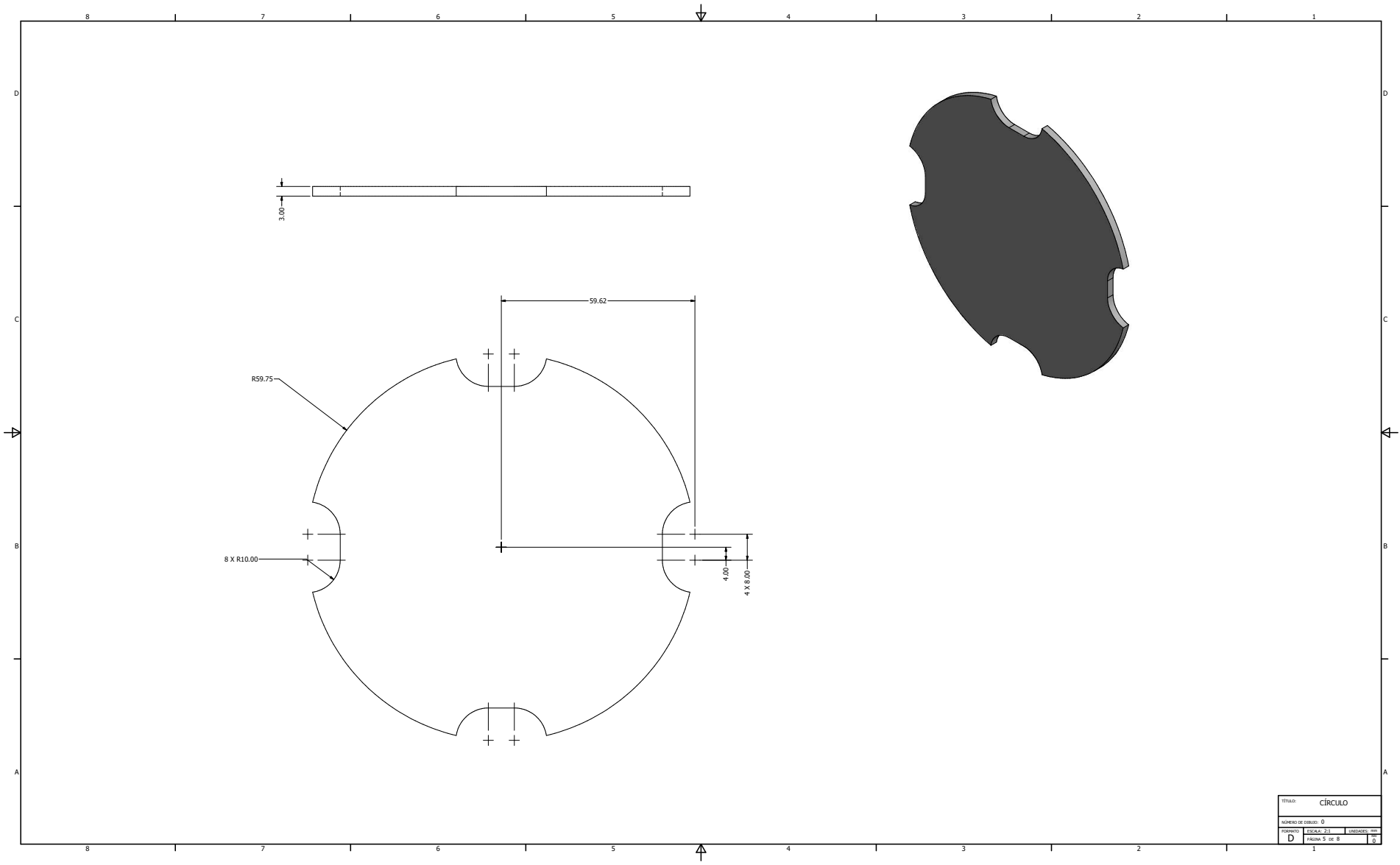
TÍTULO:	ARO		
NÚMERO DE DISEÑO:	0		
FORMATO:	DSCA - A1	UNIDADES:	mm
	FOLIO 2 de 8		0



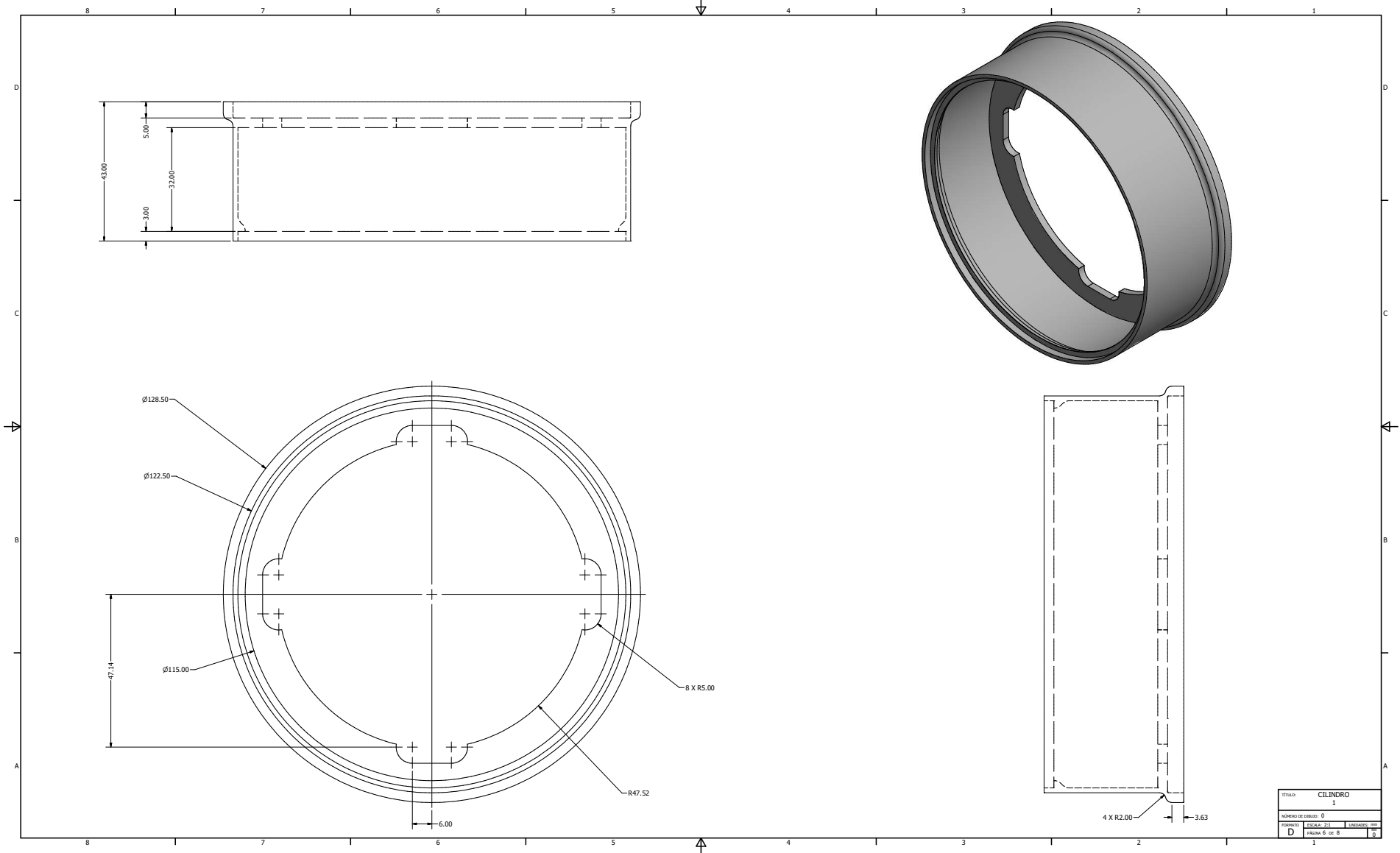
TÍTULO:	LLANTA		
NÚMERO DE DISEÑO:	0		
PROYECTO:	D	ESCALA:	1:1
		UNIDADES:	mm
		PÁGINA:	3 de 8
			0



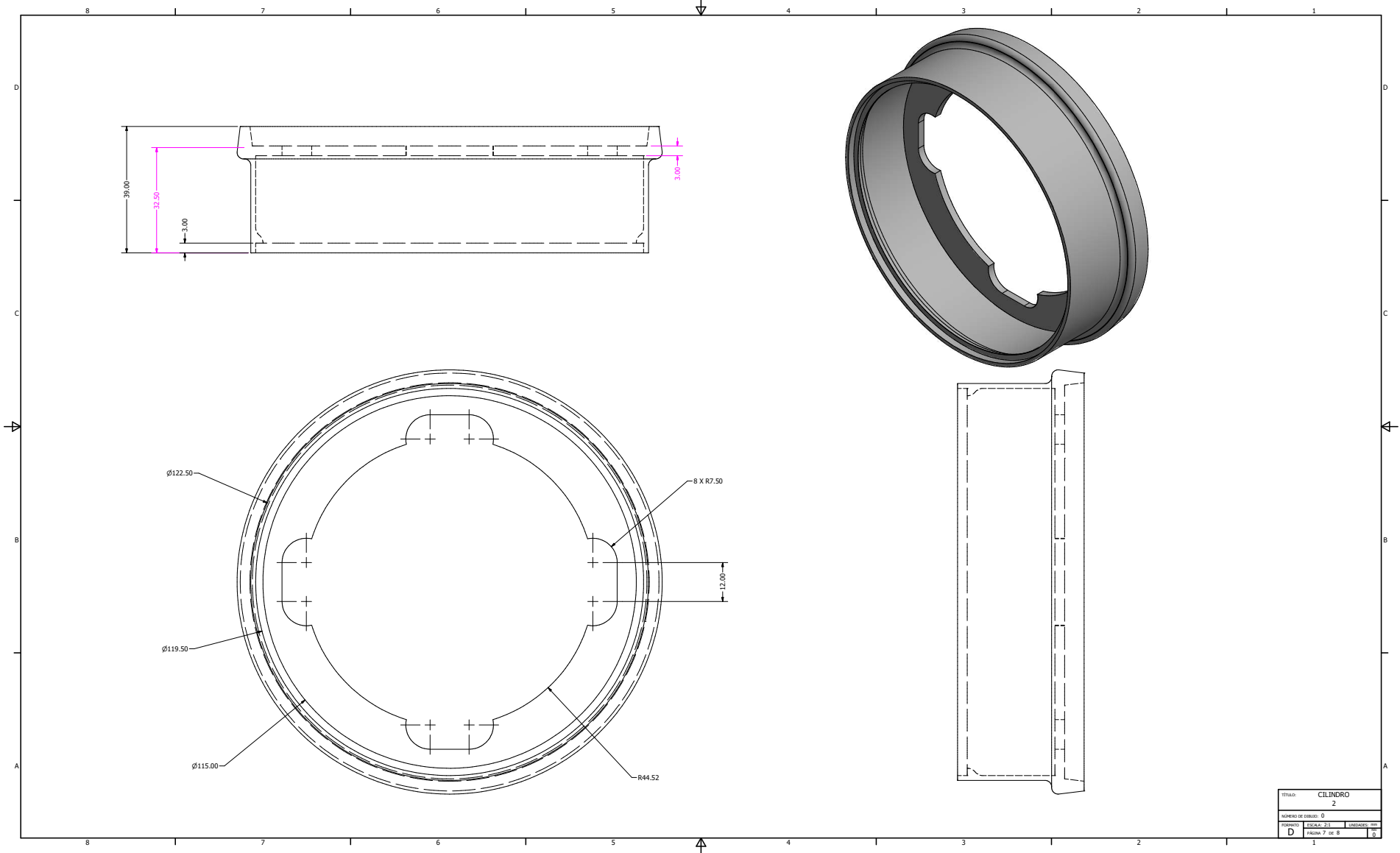
TÍTULO:		TAPA ROBOT	
NÚMERO DE CROQUIS:		0	
FORMATO:	D	ESCALA:	2:1
PÁGINA 4 de 8		UNIDADES:	mm
			0



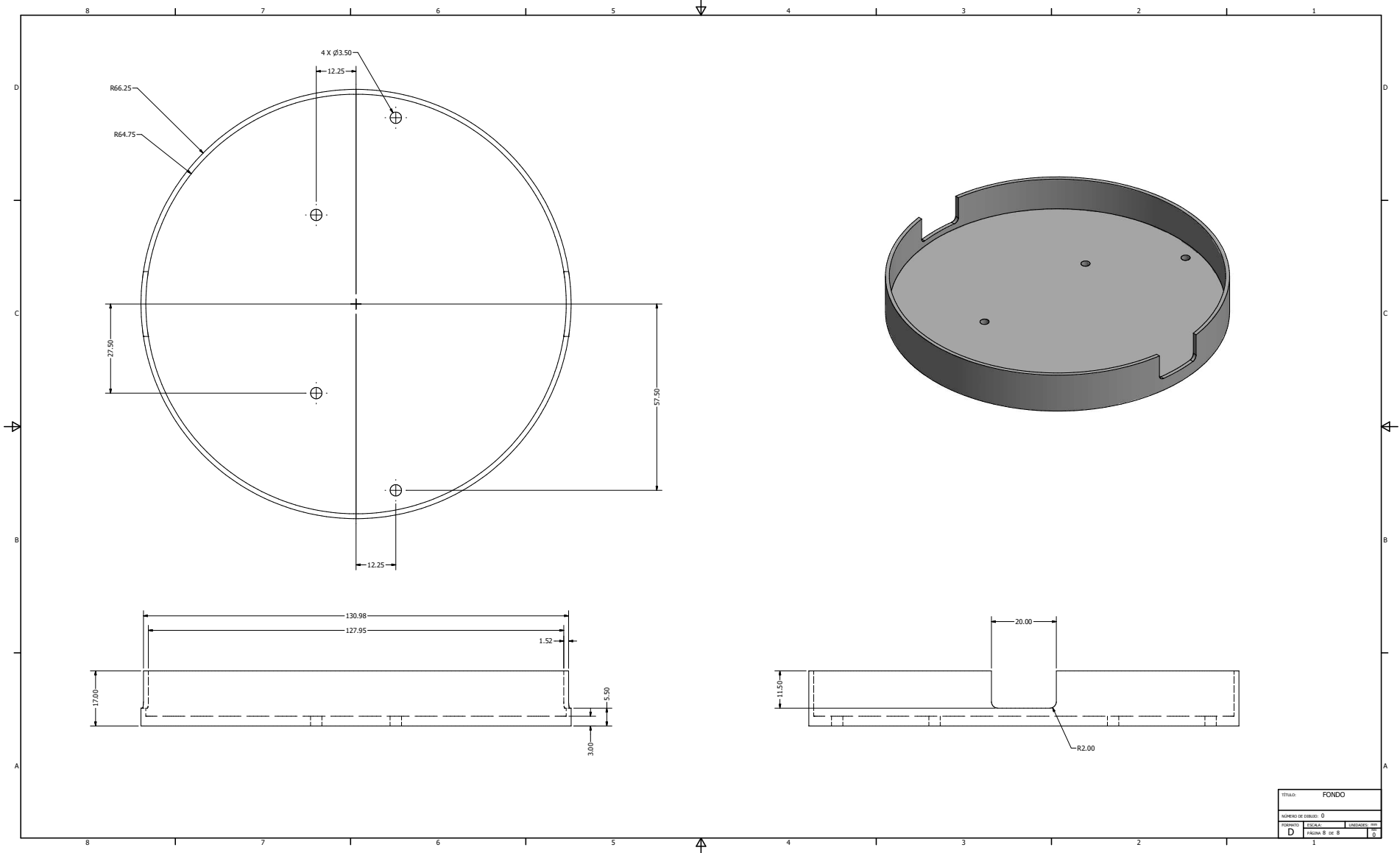
TÍTULO:	CÍRCULO		
NÚMERO DE DIBUJO:	0		
PROYECTO:	DISEÑO 21	UNIDADES:	mm
D	PÁGINA 5 de 8		0



TÍTULO:	CILINDRO		
	1		
NÚMERO DE CILINDRO:	0		
FORMATO:	DSCA - 21	UNIDADES:	mm
D	PÁGINA 6	de 8	0



TÍTULO:	CILINDRO		
NÚMERO DE ORDEN:	2		
FORMA:	D	ESCALA:	2:1
UNIDADES:	mm	PÁGINA:	7 de 8
			0



TÍTULO:	FONDO		
NÚMERO DE DISEÑO:	0		
FORMATO:	D	ESCALA:	UNIDADES: mm
	PÁGINA 8	de 8	0

14.2. Placas electrónicas

Se muestran las dos placas electrónicas diseñadas en Altium Designer con una vista de las capas más importantes es decir, top layer, bottom layer, and keep out layer. Esto nos permite generar los gerbers necesarios para la fabricación de las mismas.

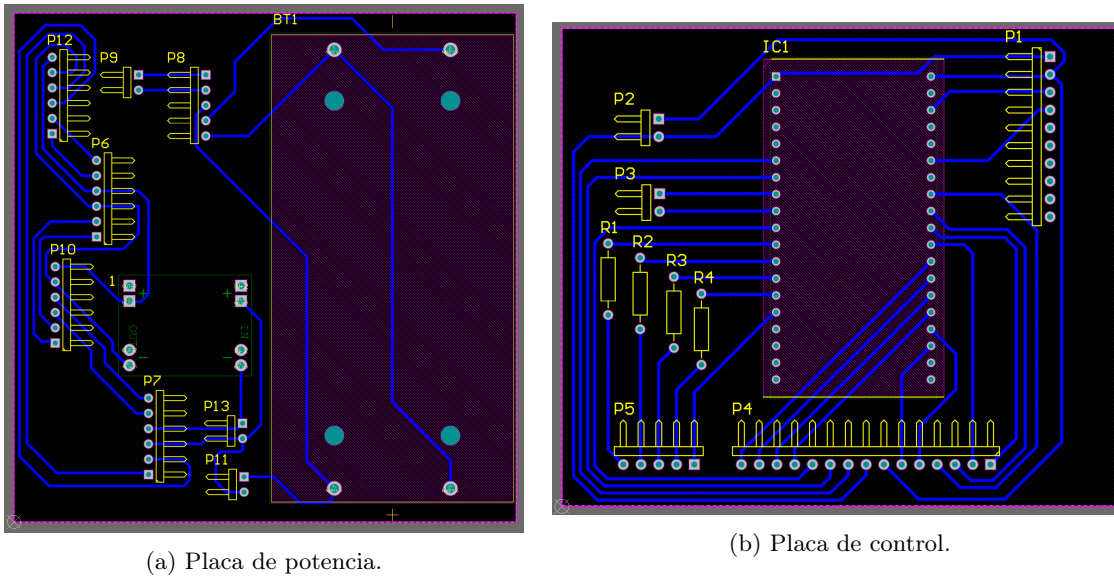


Figura 46: Vista de placas diseñadas.

14.3. Repositorio de Github

El repositorio de programación y librerías del ecosistema se encuentra privado el link de acceso es: <https://github.com/mezea-uvg/robotat>. La rama en la que se trabajó es la *17078-Dev* en esta rama se encuentran las librerías y el programa principal utilizado.

En el repositorio privado https://github.com/JoseJavierEQ/Robotat_tesis_2021 se encuentran todos los archivos finales del proyecto. Tanto los archivos gerbers para los PCBs, archivos de Inventor y los archivos necesarios para la programación del robot.

