PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

Diseño de una plataforma educativa modular para fomentar el interés académico hacia la robótica

Tesis para obtener el título profesional de Ingeniero Mecatrónico

AUTOR:
Sebastian Ruiz Figueroa

Asesor:
Mg. José Balbuena Galván

Lima, noviembre, 2024

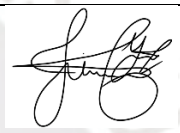
Informe de Similitud

Yo, José Guillermo Balbuena Galván, docente de la Facultad de Ciencias en Ingeniería – Sección Ingeniería Mecatrónica de la Pontificia Universidad Católica del Perú, asesor(a) de la tesis/el trabajo de investigación titulado *Diseño de una plataforma educativa modular para fomentar el interés académico hacia la robótica*, del autor *Sebastián Ruiz Figueroa*,

dejo constancia de lo siguiente:

- El mencionado documento tiene un índice de puntuación de similitud de 12%. Así lo consigna el reporte de similitud emitido por el software *Turnitin* el 11/11/2024.
- He revisado con detalle dicho reporte y la Tesis o Trabajo de Suficiencia Profesional, y no se advierte indicios de plagio.
- Las citas a otros autores y sus respectivas referencias cumplen con las pautas académicas.

Lugar y fecha: San Migue – Lima, Perú, 25 de noviembre del 2024.

<table>
<tr><td colspan="2">Apellidos y nombres del asesor:
Balbuena Galván, José Guillermo</td></tr>
<tr><td>DNI:74146973</td><td rowspan="2">Firma  </td></tr>
<tr><td>ORCID: https://orcid.org/0000-0002-5896-1942</td></tr>
</table>

RESUMEN

En las carreras universitarias, es común encontrar a alumnos que no se sientan a gusto con la carrera que estudian. Muchos de ellos sienten que no les motiva aprender conceptos teóricos y sienten haber perdido su tiempo. Mediante una encuesta rápida realizada a 35 alumnos de la carrera de mecatrónica en la PUCP, se corroboró que en efecto, aproximadamente el 50% de los alumnos no se siente motivado durante las clases. Así mismo, más del 50% de los alumnos dijo que de no perder el tiempo ya invertido, cambiarían de carrera. Si hubo algo en lo que la basta mayoría estuvo de acuerdo, fue en que más experiencias prácticas permitirían generar un mayor interés en carreras relacionadas con la robótica.

El problema con las experiencias prácticas relacionadas con la robótica es la falta de conocimientos de estudiantes que recién empiezan la carrera. Muchos no tienen conocimiento en circuitos eléctricos, programación o mecánica alguna. Existen los llamados robots modulares, los cuales permiten participar de estas experiencias prácticas por medio de la simplificación del armado de un robot sin conocimiento previo en robótica. Sin embargo, muchos de estos robots o bien son muy costosos, como es el caso del Turtle bot, o no están especializados en el aprendizaje robótico, como es el caso de los Cubelets.

De esta manera, nació la idea de REM bot, una plataforma robótica móvil de bajo costo, diseñada para brindar experiencias prácticas a alumnos de ingeniería interesados en aprender más sobre robótica. Este robot cuenta con 4 tipos de módulo, comenzando por el módulo principal al cual se le pueden acoplar diferentes tipos de configuración móvil (omnidireccional, diferencial, etc.). Otros tipos de módulo incluyen al módulo de control (encargado de conectar todos los diferentes módulos y detectar automáticamente el cambio), módulo de sensores (módulo que permite navegación autónoma por medio de un arreglo de sensores), y finalmente el módulo de actuadores (con un mecanismo de empuje de biela manivela).

Al ser software de código abierto, el docente a cargo de la experiencia práctica puede decidir qué tanta libertad le da a sus estudiantes para configurar al robot. Por ejemplo, manteniéndolo simple con la configuración más básica, conectando el módulo principal y el de control, o algo más complejo involucrando un algoritmo de control y la conexión de los 4 tipos de módulo. De esta manera se pueden brindar experiencias prácticas a alumnos con diferentes niveles de conocimiento y así fomentar el interés en la robótica.

Este trabajo incluye toda la etapa del diseño del robot, planos mecánicos, diseño de una interfaz gráfica para interactuar con el mismo, y algunas pruebas hechas en un prototipo parcialmente funcional del REM bot.

AGRADECIMIENTOS

Quiero agradecer en primer lugar a mi asesor de tesis, José Balbuena Galván, por haber ido más allá de lo que debía al ayudarme a sacar esta tesis adelante. Agradecerle su paciencia con cada consulta que le hice, su sabiduría con cada consejo y aporte que hizo a mi tesis, por su disposición en permitirme usar el laboratorio y prestarme tantos componentes electrónicos, y más allá de todo por su amistad. Lo recordare con cariño el resto de mi vida.

Igualmente agradecer a mis padres Guido Ruiz y Teresa Figueroa por su apoyo constante, particularmente en cada tropezón que he tenido a lo largo de estos 5 años de carrera. Gracias por el amor con el que me han formado en el profesional que soy hoy en día y por creer en mí desde el comienzo.

De la misma manera agradecer a mi enamorada Dania Laos Venturo, por todo el amor y el apoyo a lo largo de estos 5 duros años. No hubiera podido con toda la carga de no ser por su cariño y constante ánimo.

Finalmente agradecer a mis amigos Franco Rivadeneira y Jimm Cisneros. A Franco por el apoyo técnico con lo que pude compensar mi falta de experiencia armando robots de manera física, así como cada impresión en PLA que me ayudo a hacer y cada componente que me prestó. A Jimm por su ayuda a entender la electrónica de mi proyecto, y por apoyarme en tantas ocasiones o bien prestándome componentes del laboratorio o ayudándome a mejorar el diseño de mi proyecto con sus consejos.

DEDICATORIA

A Dios, quien es quien me ha permitido llegar a donde me encuentro. Por guiar cada paso y decisión que he tomado y por juntarme con tantas personas maravillosas a lo largo de estos 5 años de carrera.

A mis padres, por su constante amor y por creer en mí, sobre todo en tiempos difíciles done la situación jugó en mi contra.

ÍNDICE

1 Introducción 1
1.1 Problemática 1
1.2 Propuesta de Solución 2
1.3 Objetivos 3
1.3.1 Objetivo general de la tesis 3
1.3.2 Objetivos específicos de la tesis 3
1.4 Metodología 4
1.5 Alcance 5
2 Marco Teórico 6
2.1 Definiciones Básicas 6
2.1.1 Robot modular 6
2.1.2 Categorización IEEE de robots 6
2.1.3 Aprendizaje en base a proyectos 7
2.2 Antecedentes 7
2.2.1 Sistemas integrados 7
2.2.2 Sistemas integrados comerciales 9
2.2.3 Patentes 11
2.2.4 Modular Connectors vs Hardwirring 13
3 Diseño Conceptual 14
3.1 **Lista de requerimientos** 14
3.1.1 Requerimientos de Sistema 14
3.1.2 Requerimientos de Módulo 14
3.2 Estructura de funciones 15
3.2.1 Caja negra 15
3.2.2 Lista de funciones del Módulo principal 16
3.2.3 Lista de funciones del Módulo de Sensores 16
3.2.4 Lista de funciones del Módulo Móvil 17
3.2.5 Lista de funciones del Módulo de Control Autónomo 17
3.2.6 Lista de funciones del Módulo de Actuadores 17
3.3 Matriz morfológica 20
3.3.1 Módulo de Control Principal 20
3.3.2 Módulo de Sensores para Obstáculos 20
3.3.3 Módulo Móvil 20

3.3.4 Módulo de Control Autónomo 21
3.3.5 Módulo de Actuadores 21
3.3.6 Dominio de Energía 21
3.3.7 Dominio Mecánico 21
3.4 Conceptos de solución 22
3.4.1 Concepto de solución 1 23
3.4.2 Concepto de solución 2 26
3.4.3 Concepto de solución 3 29
3.5 Evaluación Técnica Económica 32
3.5.1 Evaluación Técnica 32
3.5.2 Evaluación Económica 32
3.5.3 Gráfico Ponderado 32
4 REM Bot 34
4.1 Diseño Integral 34
4.1.1 Descripción general del CAD 34
4.1.2 Descripción del funcionamiento del Sistema 36
4.1.3 Diagrama de operaciones (interacción entre componentes del sistema) 36
4.2 Diseño Mecánico 38
4.2.1 Cálculo de Torque 38
4.2.2 Módulo Principal 39
4.2.3 Módulo de Control Autónomo 49
4.2.4 Módulo de Sensores para Obstáculos 54
4.2.5 Módulo de Actuadores (Mecanismo Biela Manivela) 56
4.3 Diseño Electrónico 61
4.3.1 Diagrama de Bloques 61
4.3.2 Diagrama de Conexiones 62
4.3.3 Diseño de Circuitos Eléctricos 66
4.3.4 Diseño esquemático 67
4.3.5 Diseño de Tarjetas Impresas 72
4.4 Diseño de Software 77
4.4.1 Arquitectura de Control 77
4.4.2 Estrategias de Control 78
4.4.3 Arquitectura de Software 81
4.4.4 Diagrama de Flujo de Software 87
5 Pruebas Preliminares 90

5.1 Pruebas de Motor 90
5.2 Pruebas de Conexión I2C 92
5.3 Pruebas de Detección de Módulos 95
6 Costos Parciales 97
Conclusiones 98
Recomendaciones 100
Referencias 101
Anexos 103
Anexo 1 103
Anexo 2 121
Anexo 3 124
Anexo 4 130
Anexo 5 136
Anexo 6 137
Anexo 7 137
Anexo 8 137
Anexo 9 137
Anexo 10 137
Anexo 11 137
Anexo 12 137
Anexo 13 137
Anexo 14 137
Anexo 15 137
Anexo 16 138
Anexo 17 141
Anexo 18 143
Anexo 19 144
Anexo 20 145
Anexo 21 146
Anexo 22 147
Anexo 23 148
Anexo 24 149
Anexo 25 151
Anexo 26 152
Anexo 27 153

Anexo 28 154
Anexo 29 155
Anexo 30 157
Anexo 31 160
Anexo 32 161
Anexo 33 162
Anexo 34 163
Anexo 35 164
Anexo 36 165
Anexo 37 166
Anexo 38 167
Anexo 39 168
Anexo 40 169
Anexo 41 170
Anexo 42 171
Anexo 43 172
Anexo 45 174
Anexo 46 175
Anexo 47 176
Anexo 49 178
Anexo 50 179
Anexo 51 180
Anexo 52 181
Anexo 53 182
Anexo 54 182
Anexo 55 184
Anexo 56 185
Anexo 57 186
Anexo 58 187
Anexo 59 188
Anexo 60 189
Anexo 61 190
Anexo 62 191
Anexo 63 192
Anexo 64 193

Anexo 65 194
Anexo 66 195
Anexo 67 196
Anexo 68 197
Anexo 69 198
Anexo 70 199
Anexo 71 200
Anexo 72 201
Anexo 73 202
Anexo 74 203
Anexo 75 204

1 Introducción

En esta sección se presentará la problemática, objetivos y propuesta de solución del proyecto. El concepto del proyecto nace a partir de la necesidad de los alumnos universitarios del Perú, quienes carecen de acceso a recursos que les permitan experimentar a una temprana edad lo que es la robótica.

1.1 Problemática

Se realizó una encuesta a 35 alumnos de Ingeniería Mecatrónica en la PUCP. La naturaleza de la mecatrónica involucra electrónica, informática y mecánica. Es por ello por lo que se consideró que los alumnos de Ingeniería Mecatrónica serían quienes estarían más interesados en la robótica. De los alumnos encuestados, un 54.4% manifestó que, de no perder el tiempo ya invertido, cambiaría de carrera. Además, el 43% de los encuestados indicó no sentirse motivado durante las clases, como se muestra en el **Anexo 29**. En la ingeniería mecatrónica, no es hasta las etapas avanzadas de la carrera cuando los estudiantes empiezan a experimentar con la robótica, los circuitos eléctricos y la programación. En la PUCP, durante los primeros cuatro ciclos, solo unos pocos cursos, como Diseño Digital o Fundamentos de Programación proporcionan un acercamiento al mundo de la robótica (PUCP, 2023).

La encuesta reflejó que el enfoque teórico genera alumnos desmotivados. De los encuestados, un 97.1% afirmó que, de haber tenido la oportunidad de realizar proyectos mecatrónicos desde una etapa temprana, se sentirían más motivados durante las clases al experimentar en primera mano las aplicaciones que los conocimientos teóricos aprendidos. Es por ello que es necesario implementar proyectos que permitan a los alumnos aplicar en la vida real lo aprendido en el ámbito teórico.

Un sistema robótico abarca conceptos mecánicos, de electrónica, programación y control, entre otros. Armar un primer robot puede resultar abrumador para alguien sin experiencia ni conocimientos previos. En este contexto, la robótica modular se presenta como una herramienta que simplifica la implementación de un robot (Lenskiy et al., 2014).

Robot modular (RM) se define como un tipo de robot que cuenta con hardware diseñado para permitir la conexión y desconexión física de módulos. Un módulo es una de las partes removibles que añaden complejidad a las acciones hechas por el sistema en conjunto. (Yim et al, 2002).

Un robot modular móvil simple puede ayudar a introducir conceptos de robótica por medio de la simplificación del armado de un robot sin conocimiento previo en robótica (Gilpin & Rus, 2010). De esta manera, motiva a los estudiantes demostrando las aplicaciones que puede tener la robótica. No solo desde la universidad, sino incluso desde una etapa más temprana durante el colegio (Guo et al., 2020).

A pesar de los beneficios que un robot modular optimo puede brindar, la complejidad de la mayoría de los modelos existentes provoca que el costo de ellos se eleve (Pei y Nie, 2018). Es por ello que, si bien la robótica modular sería la solución al problema, este tipo de robots no se encuentran muy al alcance de las escuelas y universidades peruanas debido a la falta de presupuesto o la dificultad de obtención.

Se cuenta con kits de robótica accesibles para el mercado peruano. Estos permiten a los usuarios el poder armar robots en su mayoría móviles. Sin embargo estos kits de armado no permiten al usuario poder experimentar con una variedad de componentes distintos. Por ejemplo experimentar con más de un tipo de controlador, o en un mismo sistema utilizar sensores y actuadores al mismo tiempo (Naylamp, 2024). Kits de robótica mas completos suelen ser vendidos fuera del país y a mayores precios.

Partiendo de la anterior idea, se toma como ejemplo el robot modular de Xiamoi. Este solo se encuentra a la venta en el continente asiático. Es un robot modular terrestre el cual se adapta para poder jugar con niños, siguiéndolos a todas partes o interactuando verbalmente con ellos. Con un precio de venta de 3599 CNY (Yuan Chino), equivalente a 523 USD. Un robot como este es poco accesible para universidades en las que se pretendiera brindar uno de estos robots a cada estudiante, incluso si se entregara dicho robot en grupo (Pedros, 2022).

Otro ejemplo es robot ‘Turtle Bot”. Este robot móvil terrestre cuenta con una estructura modular (Robotis, 2017). Está más dirigido a ser utilizado en laboratorios con el propósito de realizar tareas varias como cruzar una pista de obstáculos. Sin embargo, el costo de este robot es muy elevado. Teniendo un costo de 1,195 USD en su versión económica y 1,850 en su versión completa (Halfacree Gareth, 2022).

En base a la información expuesta, se que puede ver que estos robots existentes no se ajustan a las necesidades de un alumno peruano. Porque o bien el costo es muy elevado como el caso del Turtle Bot, o bien su propósito es entretener más allá de aprender o investigar en el campo de la robótica. Tal es el caso del Xiaomi Modular Bot.

1.2 Propuesta de Solución

La propuesta de solución es realizar el diseño de un robot modular el cual de bajo costo y disponible en el mercado local. Esto genera una gran ventaja frente a los robots actuales los cuales son muy costosos y difíciles de conseguir. El diseño le permitirá al usuario unir diferentes módulos que darán diferentes funciones al robot. Por ejemplo, módulos móviles, módulos con diferentes tipos de controladores o módulos con diferentes tipos de sensores. Las funciones que realizará el robot estarán enfocadas en un robot móvil terrestre. Su principal tarea será cursar a través de un camino específico. Los diferentes módulos cambiarán la manera en la que el robot atraviesa el camino, por ejemplo, de manera autónoma o tele operado. Utilizando diferentes sensores que le permita realizar la tarea por medio de varios métodos.

La razón por la que se escogió un robot móvil es debido a la versatilidad y el nivel de dificultad en cuanto a conceptos de diseño comparado con un robot acuático o aéreo. Los módulos se pueden añadir de manera amplia, abriendo paso para módulos con cámaras para procesamiento de imágenes, incluso módulos con brazos robots y demás. Al ser modulares, las conexiones entre módulos serán simples. No será lo mismo que implementar un robot desde cero. Así la interacción de los usuarios con los diferentes componentes que forman un robot se vuelve más simple. En cuanto a nivel de software, este será software libre de código abierto. Lo cual le permitirá ser versátil y tener diferentes niveles de complejidad.

1.3 Objetivos

En la sección a continuación se enlistarán los objetivos generales y específicos de la tesis.

1.3.1 Objetivo general de la tesis

Diseñar un sistema mecatrónico modular orientado a estudiantes de los primeros ciclos de las carreras de ciencias e ingeniería que sea de bajo costo para luego implementar un prototipo parcialmente funcional. Este sistema será capaz de recorrer un terreno plano con diferentes obstáculos sencillos. Debido a sus conexiones modulares y a su armado simple, permitirá a los estudiantes implementar un robot funcional de manera sencilla.

1.3.2 Objetivos específicos de la tesis

- Diseñar los módulos principales del sistema mecatrónico móvil, incluyendo el módulo móvil con 4 motores DC y 2 Arduinos, así como el módulo de control con una Jetson nano, de manera que puedan integrarse para formar un robot móvil básico.
- Diseñar módulos adicionales, como el módulo de sensores con un arreglo de sensores de ultrasonido, y el módulo de actuadores con un sistema biela-manivela, que permitan al robot realizar tareas más complejas.
- Establecer un protocolo de comunicación eficiente entre los diferentes módulos, de manera que se logre un buen funcionamiento independientemente de las combinaciones utilizadas.
- Implementar una interfaz gráfica de usuario que permita controlar de forma remota al robot, y que sea capaz de detectar la conexión de los distintos módulos.
- Realizar pruebas de concepto para validar la modularidad de la plataforma y el funcionamiento de la interfaz gráfica.
- Realizar un análisis de costos de implementación del sistema y compararlo con soluciones similares del mercado para demostrar que es de bajo costo.

1.4 Metodología

El trabajo seguirá la metodología VDI 2206 y 2221. Esta metodología consta en ciertos pasos que llevan al desarrollo de conceptos de solución.

Lo primero es definir una problemática clara. Evidenciar un problema que podría resolverse, en este caso la falta de herramientas para construir robots en etapas tempranas de la carrera de ingeniería mecatrónica. Para ello se realizó una encuesta a los estudiantes de ingeniería mecatrónica de la PUCP. Es entonces que entra en juego la propuesta de solución. Esta es una descripción breve del sistema en cuestión, lo cual resume de qué manera podrá este sistema resolver la problemática.

A continuación, se plantean 2 tipos de objetivos, objetivos generales y objetivos específicos. Primero el Objetivo general del curso el cual describe lo que se quiere lograr como producto final del ensayo redactado en el curso. Este es acompañado por objetivos específicos del curso, los cuales indican pasos a seguir para llegar al objetivo planteado en el objetivo general.

Lo siguiente es el un pequeño marco teórico con conceptos clave para entender el sistema tales como; Robot Modular, PBL entre otros. El marco teórico viene de la mano del estado del arte. El estado del arte es una recolección de los sistemas integrados existentes. En este caso particular dividido en 3 secciones, sistemas integrales de la industria, sistemas integrales educacionales y finalmente patentes.

Ya teniendo una referencia sólida, se realiza una lista de requerimientos a los que se compromete el sistema a cumplir en su diseño final. Estos se dividen entre deseosos y exigencias. Como el diseño va dirigido a alumnos universitarios estos deseos y exigencias se realizarán consultando con profesores y alumnos.

Acto seguido se definen las funciones que regirán el sistema mecatrónico. Estas no especifican que componentes se utilizarán para llevarse a cabo. Son ejemplificadas en el diagrama de caja negra el cual presenta las entradas y salidas del sistema, así como el diagrama de funciones el cual presenta las interconexiones entre los diferentes dominios del sistema (Mecánico, energía, control, actuadores, entre otros). Para este proyecto en particular envuelve dominios como el mecánico, de energía, de control, entre otros.

Una vez establecidas las funciones se procede a presentar la matriz morfológica, la cual ayuda a seleccionar componentes entre las opciones encontradas para poder dar con diferentes conceptos de solución de los cuales se seleccionará el concepto óptimo. Teniendo en cuenta el aspecto económico y técnico.

A continuación, se procede a realizar la selección de componentes de cada uno de los módulos, justificando apropiadamente cada elección con cálculos y demás justificaciones. Para luego proceder a la descripción del diseño final.

La descripción del diseño final se divide en 4 secciones, siendo la primera la sección general. Acto seguido se explica el subsistema mecánico, haciendo referencia a los planos mecánicos. Luego se explican los diferentes circuitos eléctricos, diagramas de bloques y diagramas eléctricos. Por último, se explica el dominio del software, donde se explica la interacción humano-máquina, algoritmos de control e interfaz de usuario realizadas.

Finalmente se muestran las pruebas hechas al prototipo parcial en cuestión, tanto las pruebas hechas a componentes independientes como a el sistema integrado, y se finaliza el documento por medio de conclusiones y una reflexión final.

1.5 Alcance

En cuanto al público destinado, este tipo de robot estará enfocado en alumnos de ingeniería mecatrónica quienes estén cursando Estudios Generales Ciencias. Estos alumnos carecen de experiencia en cuanto a conceptos de robótica a diferencia de otras universidades las cuales son quienes se verán más beneficiados por un tipo de robot el cual es fácil de armar. Considerando que en otros países tienen la oportunidad de un acercamiento a la robótica incluso desde el colegio (Patiño et al., 2019).

El sistema integrado tendrá 4 diferentes tipos de módulos; Módulo Principal, Módulo de Control Autónomo, Módulo de Sensores para Obstáculos y Módulo de Actuadores de Empuje. El módulo principal incluirá el sistema de alimentación, la configuración omnidireccional y 1 o más controladores para controlar los motores que configuran el sistema de movimiento. El módulo de control autónomo contiene el controlador principal del robot añadiendo la posibilidad de realizar navegación autónoma. Finalmente el modulo de sensores permitirá al robot poder detectar y evadir obstáculos mientras el módulo de actuadores le permitirá utilizar actuadores.

El funcionamiento más básico será por medio del módulo principal unido al módulo de control. Este tendrá 3 grados de libertad, los cuales incluyen la capacidad de moverse en el plano y la posibilidad de rotar.

Las conexiones entre módulos serán simples y removibles para que de esta manera pueda armarse el robot con facilidad. Es por ello que se emplearán conectores rápidos exteriores a los módulos que estén bien etiquetados para toda conexión que deba ser hecha por el alumno.

Por último, el sistema será de código abierto, permitiendo así que se puedan importar o implementar librerías externas que faciliten enormemente la programación. Se realizará un prototipo parcial del sistema, mostrando el movimiento omnidireccional de manera sencilla por medio de un código base de poca complejidad.

2 Marco Teórico

En la sección a continuación se presentará el marco teórico que permitirá comprender conceptos importantes para la investigación. Así mismo, se mostrarán diversos proyectos existentes en la actualidad los cuales realizan tareas similares al sistema que se busca diseñar. Estas tareas incluyen acciones y elementos generales del robot, así como tareas más específicas.

2.1 Definiciones Básicas

2.1.1 Robot modular

Diseño robótico el cual permite la utilización de módulos los cuales pueden funcionar de manera independiente y al ensamblarlos es posible formar sistemas de mayor complejidad. Estos módulos pueden ser elementos mecánicos, electrónicos, entre otros (Gilpin y Rus, 2010).

2.1.2 Categorización IEEE de robots

Los robots en el ámbito educacional pueden ser categorizados en 4 diferentes tipos de robots. Esto depende de 2 criterios, el primero tomando en cuenta si son robots Físicos, o robots Virtuales. Los robots físicos son conocidos por generar un mayor nivel de interacción con los estudiantes. De igual manera, los robots físicos están más orientados a operaciones prácticas por los estudiantes. Por otro lado, los robots virtuales están más orientados al desarrollo de software, y consiguen un nivel de interacción menor con los estudiantes. Como se busca tener la mayor interacción posible, se decidió hacer un robot físico.

El segundo criterio para categorizar los robots educacionales es de acuerdo a su función. Están los robots de servicio instructivo, y los robots aplicativos. Los robots aplicativos están más orientados a desarrollar la creatividad de los estudiantes resolviendo problemas reales mediante la robótica. Por otro lado, los robots instructivos son más un apoyo para enseñar conceptos no necesariamente orientados a la robótica. Es por ello que se optó por un robot físico de disciplina aplicada. También conocido como Robot para uso multi funcional. La Figura 2. 1 a continuación muestra la categorización de robots proporcionada por la IEEE (Pei y Nie, 2018). En el caso del sistema en cuestión se considera un robot multi funcional.

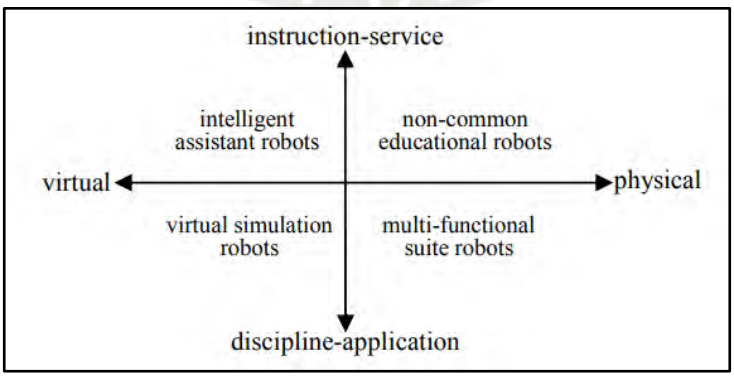


Figura 2. 1 Categorización IEEE de robots educacionales (Fuente: IEEE)

2.1.3 Aprendizaje en base a proyectos

El ‘Aprendizaje en base a proyectos’ (PBL, Project Based Learning) es una metodología de aprendizaje la cual busca utilizar la complejidad de proyectos para brindar una educación más integral. Esta metodología permite al alumno aprender mediante el desarrollo de proyectos los cuales le permiten a los alumnos ver la aplicación en el mundo real de la teoría aprendida en clase (Sanger & Ziyatdinova, 2014) (Abdulkadir et al., 2019).

Un ejemplo de esto se ve en el curso ‘ROB 102: Intro to AI & Programming’, curso de pregrado proporcionado por la universidad de Michigan. En este curso, los alumnos tienen la oportunidad de desarrollar sus habilidades de programación mediante el PBL. Los alumnos hacen uso de robots modulares desarrollados por la misma universidad, a los cuales se les añaden diferentes sensores y componentes para que puedan atravesar una pista de obstáculos (Universiry Of Michigan, 2021).

Este tipo de aprendizaje ha mostrado un mayor interés por parte de los alumnos según comentan los instructores del curso. Estos buscan recrear la emocionante experiencia que tuvieron como estudiantes la primera vez que introdujeron algunas líneas de código para hacer que un robot de la vida real se moviera (umrobotics, 2022).

2.2 Antecedentes

2.2.1 Sistemas integrados

En la actualidad, existen sistemas integrados los cuales buscan cumplir una función similar a la del robot a diseñar. A continuación, se presentan diferentes sistemas integrados los cuales son robots modulares orientados a la educación. Estos permiten a los alumnos el poder diseñar sus propios robots a diferentes niveles de complejidad. Sin embargo, estos robots tienen aspectos en los que se les puede mejorar. Se comparará los atributos de los siguientes sistemas integrados.

a) University of Michigan Modular Robot: sistema diseñadopara ser utilizado como material didáctico orientado a los cursos ROB 102. Introduction to AI and Programming y ROB 103 (Robotics Mechanisms). Es un robot orientado a la introducción a conceptos de navegación autónoma. Cuenta con un sensor LIDAR de 360 °, el cual permite construir en realidad virtual el ambiente que lo rodea y reconocer su posición en medio de este. De esta manera haciendo posible la navegación autónoma (Universiry Of Michigan, 2021).

b) E-Puck 2: es un robot educacional desarrollado por la Escuela Politécnica Federal de Lausana (Suiza) en el 2004, que ayuda a los estudiantes a aprender sobre sistemas embebidos y robótica. Sin embargo, en el 2018 se lanzó una nueva versión llamada E-Puck 2 y contiene 15 sensores, incluyendo 4 micrófonos, una cámara de color, 8 sensores de proximidad infrarrojos, parlantes, 8 leds rojos, 4 leds RGB, entre otros.

Tiene una dimensión de 70 mm de diámetro, 45 mm de altura y un peso de 130g. Además, su fuente de energía es una batería recargable de litio de 1800 mAh. Tiene de microcontrolador un STM32F407 de 32 bits. Para el movimiento emplea dos ruedas con un diámetro de 41 mm las cuales son accionadas por motores paso a paso (GCtronic, s. f.).

c) EMERGE Bot (Easy Modular Embodied Robot Generator): es un robot modular desarrollado por la IT University of Copenhagen. Es de bajo costo y Open-Source (código abierto). Consta en módulos en forma de cubo, cuyo hardware puede ser reconfigurado para realizar una gran variedad de proyectos distintos (IT University of Copenhagen, s. f.).

Un módulo cuenta con un motor central, una PCB, imanes y contactos eléctricos que permiten el enlace con más módulos, así como un sensor infrarrojo. Le permite a los alumnos realizar proyectos para experimentar con las diferentes morfologías que les permite desarrollar este robot modular. En la Figura 2. 2 a continuación se puede apreciar un proyecto en el que el robot navega a través de un camino evitando colisionar con las paredes (IT University of Copenhagen, s. f.).

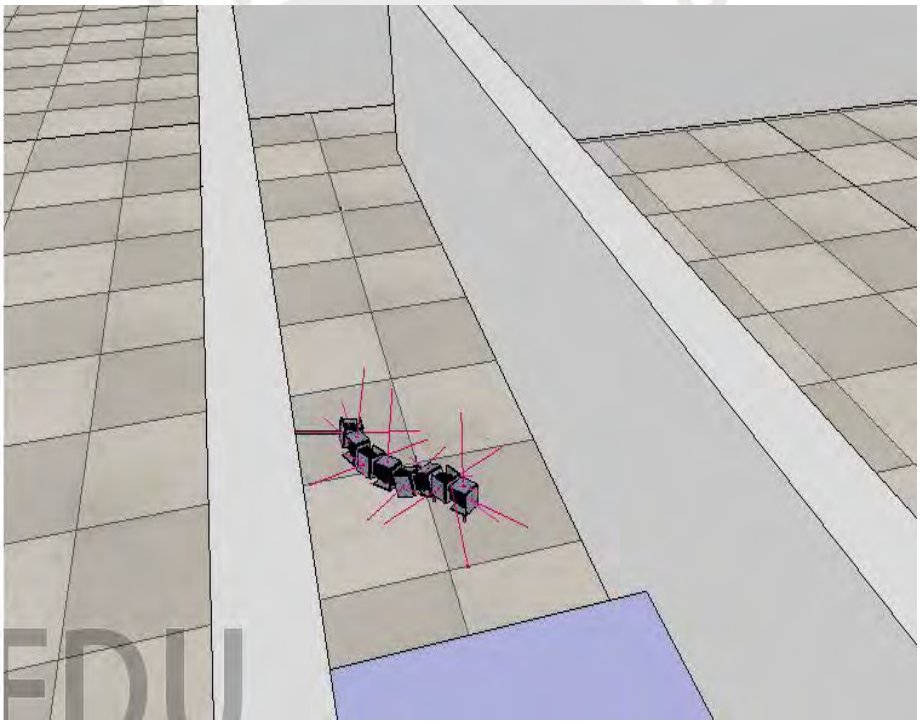
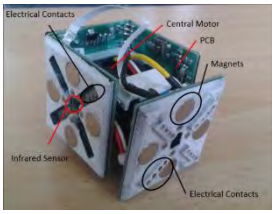


Figura 2. 2 EMERGE bot : Effect of Sensors Project

(Fuente: IT University of Copenhagen)

A continuación, se muestra la Tabla 2. 1 de los distintos sistemas integrados que existen y son utilizados en diferentes instituciones educativas los cuales cumplen con una función general similar.

Tabla 2. 1 Comparación de Sistemas Integrados (Fuente: Elaboración propia)

Parámetros	University of Michigan Modular robot	E-Puck 2	EMERGE Bot
Imagen de referencia			
Dimensiones (mm)	-	70 x 70 x 45	50 x 50 x 50 por módulo
Sensor de Distancia	- LIDAR	- Infra Rojo	- Infra Rojo
Orientación de Aprendizaje	Orientado a la programación	Orientado a la Robótica	Orientado a Cursar Obstáculos
Comunicación	-	Bluetooth	-
Sistema de Tracción	Omnidireccional	-	-
Cuenta con Cámara	No	Si	No
Alimentación	-	Batería Lipo (carga USB)	-

2.2.2 Sistemas integrados comerciales

Así como existen robots modulares orientados a la educación diseñados con un propósito más específico y más orientado al nivel universitario, en la industria se producen robots modulares que de igual manera buscan impactar en la educación a un nivel más escolar. Este tipo de sistemas mecatrónicos suelen no ser muy especializados ni muy accesibles al público en general. A continuación, se presentarán dichos robots encontrados en el mercado destacando sus atributos y características principales.

a) Xiaomi Modular Bot

El robot modular de Xiaomi está orientado a un mercado más infantil. Se ha lanzado en el continente asiático como un juguete de tecnología avanzada el cual permite a niños pequeños crear robots uniendo los módulos los cuales funcionan de manera independiente. Las conexiones de estas piezas se dan por imanes, lo cual resulta muy sencillo de ensamblar incluso para un niño pequeño (Pedros, 2022).

b) Cubelets Discovery Set

Otro robot similar disponible en el mercado es el set de módulos 'Cubelets Discovery Set'. Este robot consiste en módulos en forma de cubo, los cuales tienen diferentes funciones independientes. De igual manera, estos pueden ser conectados y formas diferentes configuraciones. Al igual que el robot de Xiaomi, este es un producto comercial más orientado hacia los niños y las conexiones entre módulos son de manera magnética. Hay 3 tipos de cubos, de sensores, de actuadores y de controladores (Mod Robotics, s. f.).

c) Clic Bot

El Clic Bot es un robot modular que funciona de manera muy similar al robot modular de Xiaomi. Con la diferencia de que este fue diseñado en estados unidos. Este cuenta con funciones como manipulación de objetos, movilidad con diferentes configuraciones, y una inteligencia artificial integrada la cual le permite interactuar con los niños. Al igual que los otros dos robots, la conexión entre módulos se realiza de manera magnética (Key I Robot, s. f.).

d) Turtle Bot

Entre los robots presentados en el documento, este es el modelo más orientado a la investigación. El turtle bot es un robot móvil el cual cuenta con un sensor LIDAR que le permite reconstruir el ambiente en el que se encuentra, de manera similar al robot de la universidad de Michigan.

Este incluye una cámara de alta resolución que permite implementar aplicaciones de procesamiento de imágenes. Así mismo, tiene un compartimiento superior utilizado para cargar objetos de hasta 15 kg de masa (Clear Path, s. f.). Todo esto puede apreciarse en la Tabla 2. 2 a continuación.

Tabla 2. 2 Comparación de Sistemas Integrados Comerciales (Fuente: Elaboración propia)

Parámetros	Xiaomi Modular Bot	Cubelets Discovery Set	Clic Bot	Turtlebot
Imagen de referencia				
Conexión entre módulos	Imanes	Imanes	Imanes	-
Continente de Venta	Asía	América	América	Global
Dimensiones (mm)	300 (altura máxima)	46 x 46 x 46 (por cubo)	-	341 x 339 x 351
Peso (Kg)	0.262	-	-	
IA Incorporada	Si	No	Si	
Tipo de Producto	Juguete	Educativo	Juguete	Educativo/ Investigación
Alimentación	-	Batería en uno de los cubos (recargable)	-	Batería (no especificada) con estación de carga

2.2.3 Patentes

Así como existen robots ya implementados, también existen múltiples patentes de diseños similares al robot que se busca desarrollar. Algunos de estos diseños se encuentran en proceso para ser implementados mientras que otros se encuentran simplemente en etapa de diseño. De igual manera, sirven para tener una referencia de los diferentes tipos de diseños similares que existen.

Sistema de robot modular (Patente: US20160005331A1)

La siguiente patente corresponde a un robot modular orientado a la educación desarrollado por la empresa 'Barobo Inc'. Este robot en particular cuenta con múltiples grados de libertad. Tiene un sistema de acople simple el cual permite la integración de una gran cantidad de módulos que le permite llevar a cabo distintas funciones.

Este robot utiliza un encoder para poder sensar los niveles de libertad que tiene. Una particularidad de esta patente es que los módulos pueden unirse de manera fácil y compacta debido al reducido tamaño de estos. De igual manera, el nivel de libertad del sistema depende de la configuración (Ryland, 2019). Un boceto se puede ver en la Figura 2. 3 a continuación.

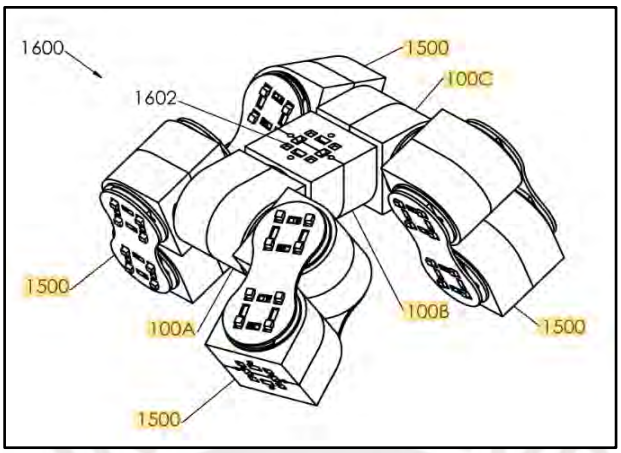


Figura 2. 3 Planos del Sistema de robot modular patente US20160005331A1

(Fuente: Barobo Inc)

Modular robotic teaching tool (Patente: US6877574B2)

El siguiente robot modular de igual manera tiene un enfoque educacional. Esta patente fue desarrollada por el 'MIT' (Massachusetts Institute of Technology). Sin embargo a diferencia del anterior, este cuenta con ruedas. De manera similar cuenta con módulos (de mayor tamaño en comparación con la anterior patente) los cuales incluyen módulo de control remoto, 2 motores desmontables para diferentes configuraciones de movimiento entre otros.

De igual manera el robot incluye un manual de instrucciones el cual indica las diferentes configuraciones que este posee, así como instrucciones de como ensamblar el robot. Debido a la complejidad del ensamblado y del diseño, este robot está más orientado a la educación universitaria (Thompson et al., s. f.). El boceto se puede apreciar en la Figura 2. 4 a continuación.

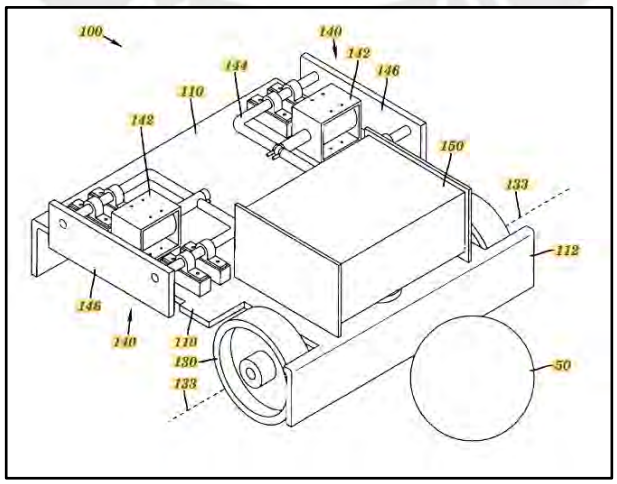


Figura 2. 4 Planos del Sistema de robot modular patente US6877574B2

(Fuente: Massachusetts Institute of Technology)

Autonomous Modular Robot (Patent: US10802499B2)

El robot Autónomo Modular desarrollado por Viabot Inc. está más orientado al desarrollo de tareas múltiples de manera automatizada. Al ser un robot modular, este es capaz de adaptarse a una amplia variedad de tareas las cuales incluyen aspirar pisos, cortar césped entre otras tareas.

De igual manera este robot incluye un mecanismo acoplable que le permite levantar objetos no muy pesados. Incluye un sensor de dimensión el cual le permite al robot aproximar el tamaño del objeto que debe levantar. Si bien es educativo estudiar robots que son utilizados en la industria, no es lo mismo tener la oportunidad de ensamblar un robot diseñado por uno mismo. Los robots modulares tienen un grán potencial en el ámbito educativo. La siguiente Figura 2. 5 muestra el diseño de dicho robot (Ratanaphanyarat & Ding, 2020).

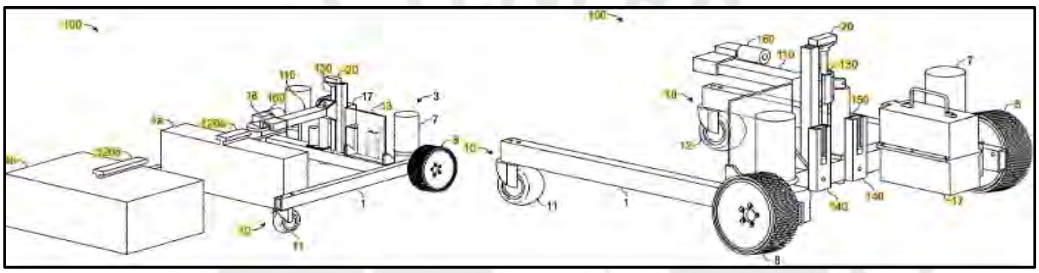


Figura 2. 5 Planos del Sistema de robot modular patente US10802499B2

(Fuente: Vía Bot .Inc)

2.2.4 Modular Connectors vs Hardwirring

Los conectores modulares son conectores especializados en robótica modular. Estos incluyen múltiples conexiones unidas como un solo conector. Este concepto se aprecia de mejor manera con la Figura 2. 6 a continuación.

Figura 2. 6 Modular Connectors (Fuente: Positronic)

La mayor ventaja de los conectores modulares es su manera simple de ser instalados. De igual manera este tipo de conectores son flexibles, y puede alterarse su diseño para cumplir con las necesidades del usuario. De igual manera si bien son más costosos que el cableado convencional, estos conectores permiten un costo total menor a largo plazo. Debido al costo reducido en mantenimiento, menor costo de instalación y su mayor versatilidad (Positronic, 2021).

3 Diseño Conceptual

3.1 Lista de requerimientos

A continuación, se presentarán los requerimientos más importantes para el diseño del sistema. Estos requerimientos sirven como guía para determinar el alcance del desarrollo del proyecto. Están divididos entre requerimientos para todo el sistema en conjunto, así como para cada módulo individual. En el **Anexo 16** se puede ver a mayor detalle una tabla con todos los requerimientos.

3.1.1 Requerimientos de Sistema

Los siguientes requerimientos involucran a todo el sistema como tal, y no se refieren a un módulo en particular.

- Sistema mecatrónico capaz de atravesar un camino plano que puede o no tener obstáculos. Esto se realizará con diferentes métodos según la configuración de los módulos conectados.
- El sistema contará con 1 módulo principal conectado a 3 tipos de módulos los cuales añadirán operaciones adicionales. Estos serán el módulo de sensores para obstáculos, módulo de actuadores y el módulo de control autónomo.
- Uniones entre módulos bien sea por forma, o utilizando pocas uniones atornilladas.
- El robot será capaz de moverse en el plano XY, 2 grados de libertad. Tendrá adicionalmente la capacidad de rotar sobre su eje debido a la configuración omnidireccional, añadiendo 1 grado de libertad adicional.

3.1.2 Requerimientos de Módulo

Los siguientes requerimientos son específicos para cada módulo. Se hace referencia a cual módulo pertenecen.

- Todos los módulos serán alimentados por un sistema de baterías localizado en el módulo principal.
- Los módulos deberán tener un etiquetado visible para las conexiones, o bien emplear conexiones de entrada única.

- El módulo de control deberá tener una manera de determinar el estado de los demás módulos para así poder funcionar con las diferentes configuraciones pensadas para el sistema.
- El módulo de sensores para obstáculos deberá tener un sensor de posición, de manera que se puedan detectar los obstáculos presentes. Se emitirá al módulo de control la distancia al obstáculo.
- El módulo de actuadores incluirá un empujador lineal el cual le permitirá impactar en obstáculos pequeños, marcar estos obstáculos o presionar botones. Así mismo enviará al módulo principal señales del estado del módulo.

3.2 Estructura de funciones

A continuación, se mostrarán diagramas se presentarán las diferentes entradas y salidas del sistema mediante un diagrama, así como las principales funciones y subfunciones divididas según el módulo al que corresponde.

3.2.1 Caja negra

La caja negra o Black Box es un recurso el cual muestra de manera ilustrativa las señales de entrada y salida que tendrá el sistema. En la Figura 3. 1 se describe de manera ordenada las entradas y salidas del robot móvil, estas pueden ser de energía, señales o materia.

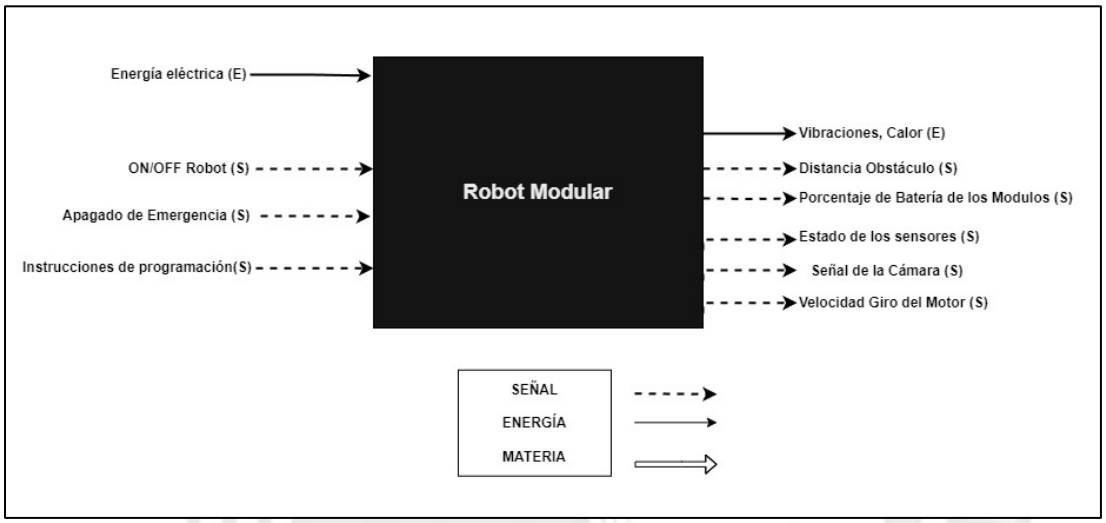


Figura 3. 1 Diagrama de Caja Negra (Fuente: Elaboración propia)

Señales de entrada

- Energía eléctrica: energía la cual alimentará al sistema. Esta provendrá de una batería ubicada en cada módulo de manera independiente.
- On/Off Robot: señal que le indicará al sistema cuando encenderse y cuando apagarse.
- Apagado de Emergencia: señal de emergencia la cual permitirá al usuario apagar el sistema de manera inmediata en caso de emergencia.
- Instrucciones de programación: son las instrucciones de programación las cuales permitirán el funcionamiento del robot.

Señales de salida

- Vibraciones y calor: energía emitida por la perdida de energía al producirse movimiento en el sistema.
- Porcentaje de batería de los módulos: porcentaje de batería restante en los módulos para alertar al usuario cuando esta deba ser recargada.
- Estados de los sensores: estados detectados por los diferentes sensores acoplados al módulo de sensores.
- Distancia obstáculos: distancia a obstáculos detectada por el módulo de sensores.
- Velocidad de Giro Motor: velocidad de giro del motor detectada por módulo móvil.
- Señal de la cámara: señal de video proporcionada por una cámara.

3.2.2 Lista de funciones del Módulo principal

Este módulo será el encargado de conectar el resto de los módulos e integrarlos con diferentes configuraciones para el funcionamiento del sistema y deberá siempre conectado al módulo móvil y al módulo de control autónomo para poder funcionar en la configuración más básica del robot. Los demás módulos serán opcionales, cada uno agregando una función adicional. Este módulo se aprecia con más detalle en el **Anexo 17**.

- Energizar sistema: función encargada de alimentar el módulo.
- Detener sistema: función que detiene el funcionamiento del módulo, bien sea por la señal on/off o por la parada de emergencia.
- Acondicionar Voltaje: función que acondiciona el voltaje de los diferentes componentes que conforman al módulo.
- Cargar Programas: función que recibe las instrucciones de programación para el funcionamiento del sistema.
- Sensar porcentaje de batería módulo principal: función que sensa el porcentaje de batería restante del módulo principal.
- Determinar estado módulo control autónomo: determina si el módulo de control autónomo está conectado.
- Determinar estado módulo sensores: determina si el módulo de sensores está conectado.
- Determinar estado módulo móvil: determina si el módulo móvil está conectado.
- Determinar estado módulo actuadores: determina si el módulo de actuadores está conectado.
- Determinar porcentaje de batería: determinar porcentaje de batería de los módulos del robot.
- Determinar Velocidad de giro: determinar velocidad de giro del motor.
- Determinar Distancia Obstáculo: determinar distancia a obstáculos.
- Determinar Ruta: determinar ruta a seguir en base a imagen de cámara.
- Determinar Ruta: determinar ruta a seguir en base a obstáculos detectados.
- Conexión Cámara: función que capta señal de video por medio de una cámara.

3.2.3 Lista de funciones del Módulo de Sensores

Módulo por medio de cual se le podrán añadir sensores que puedan detectar obstáculos. Adicionalmente a esta función, este módulo cuenta con una alimentación propia, así como detectores de la batería restante y si está conectado al módulo principal. Esto se aprecia en el **Anexo 18.**

- Sensar porcentaje de batería módulo sensores: sensa el porcentaje de batería del módulo de sensores.
- Sensar distancia a obstáculo: sensa la distancia del robot a obstáculos que podrían haber.
- Sensar estado modulo sensores: sensa si el modulo se encuentra conectado al módulo principal. Emite una señal al modulo principal si detecta que está conectado.

3.2.4 Lista de funciones del Módulo Móvil

El módulo móvil es el encargado de accionar el desplazamiento del móvil, así como accionar el cambio de dirección; siendo de los más importantes, puesto que este módulo es vital para que el robot modular pueda avanzar. Cabe resaltar que este módulo es parte del módulo principal, por lo que se consideran un solo módulo. Sin embargo, se toman como entidades diferentes para el diagrama de funciones por cuestiones de orden. Esto se aprecia en el

- Sensar velocidad: función que sensa la velocidad de giro del motor.
- Sensar estado del módulo móvil:
- Accionar mecanismo de desplazamiento: función que proporciona la fuerza para poder accionar el mecanismo de desplazamiento.
- Accionar direccionamiento: función que proporciona la fuerza para poder accionar el mecanismo de direccionamiento.
- Desplazar Móvil: función que desplaza el móvil.
- Direccionar Móvil: función que direcciona el móvil.
- Albergar Componentes: esta función es la que permite que todos los demás componentes se integren formando un móvil.

3.2.5 Lista de funciones del Módulo de Control Autónomo

Si bien el módulo principal cuenta con un controlador propio, el modulo de Control autónomo cuenta con el controlador maestro ya que se utiliza una conexión I2C y se pueden tener múltiples controladores. Adicionalmente cuenta con un componente que le permita localizar su posición y así realizar navegación autónoma. Las funciones específicas de ambos controladores se detallan en el **Anexo 20**.

- Sensar porcentaje de batería módulo de control: sensa el porcentaje de batería restante en el módulo de sensores.
- Sensar Posición del móvil: sensa la posición en la que se encuentra el móvil para realizar navegación autónoma.
- Sensar estado Modulo de Control: sensa si el modulo de control autónomo está conectado al módulo principal o no.
- Determinar posición móvil: determina la posición en la que se encuentra el móvil para realizar navegación autónoma.
- Determinar calcular ruta robot: calcula la ruta del robot para que pueda manejar de manera autónoma.

3.2.6 Lista de funciones del Módulo de Actuadores

Módulo el cual permitirá al sistema poder utilizar actuadores, siendo el actuador principal un dispositivo de empuje que le permitirá al robot empujar obstáculos. Las funciones específicas de detallan en el **Anexo 21**

- Sensar porcentaje de batería módulo de actuadores: sensa el porcentaje de batería restante en el módulo de actuadores.
- Sensar estado de módulo de actuadores: sensa si el módulo está conectado al módulo principal.
- Accionar movimiento de manipulador: acciona el movimiento del actuador, en este caso un empujador.

a) Diagrama completo

A continuación, la Figura 3. 2 presenta un resumen de todos los módulos del robot. Ese diagrama muestra la interacción de rodas las funciones anteriormente mencionadas entre sí, mostrando todas las entradas y salidas del sistema.

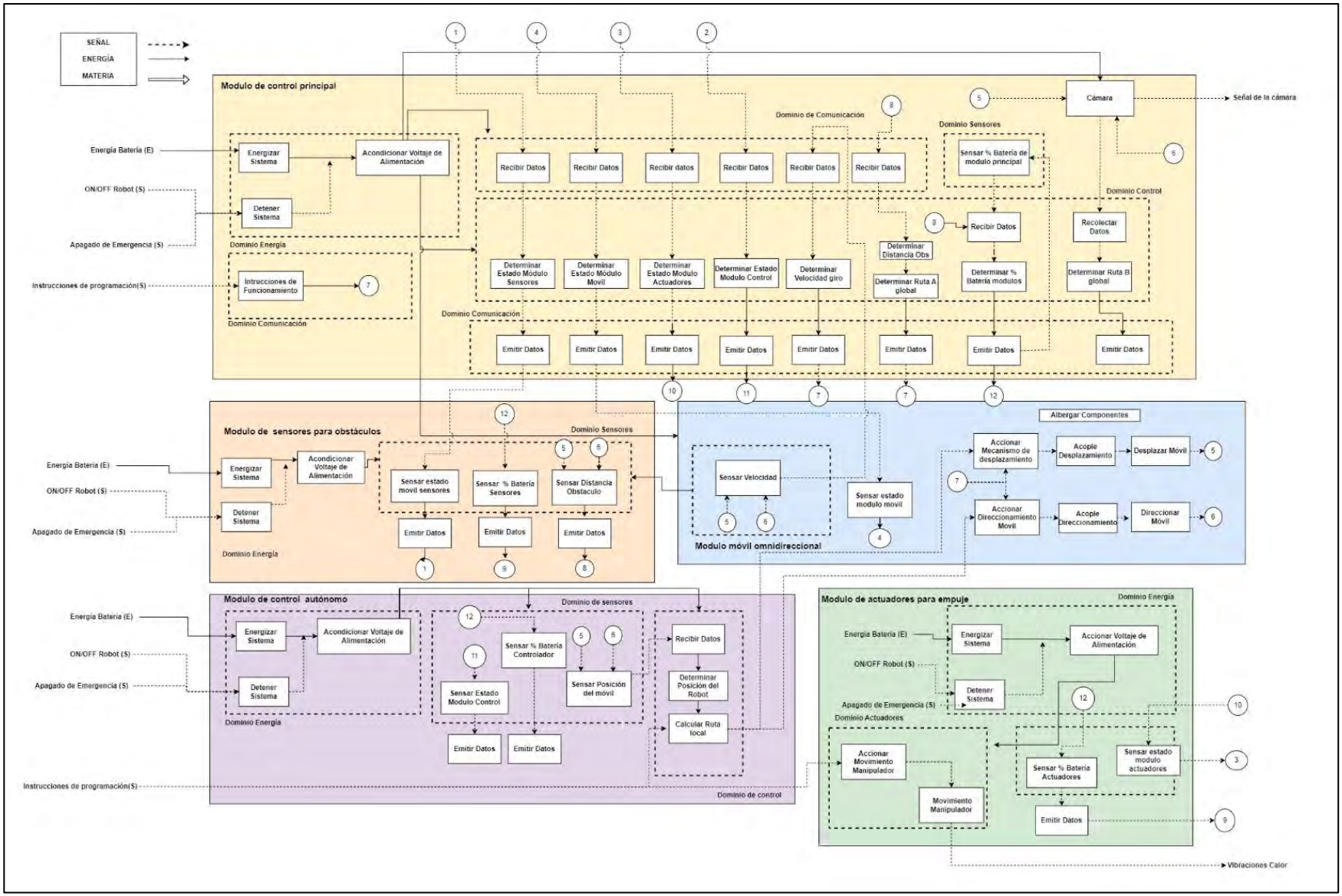


Figura 3. 2 Diagrama de Funciones (Fuente: Elaboración propia)

3.3 Matriz morfológica

A continuación, se presenta la matriz morfológica dividida en los módulos correspondientes. Esta permite tener diferentes opciones para cumplir con cada una de las funciones que han sido determinadas en el diagrama de funciones. A partir de ella, se eligieron 3 conceptos de solución. Posteriormente se realizó una evaluación técnico económica para dar con el concepto de solución óptimo.

3.3.1 Módulo de Control Principal

En primera instancia se mostrará el módulo de control principal. El módulo principal cuenta con 5 dominios. El dominio de comunicación necesita recibir señales de prendido/apagado, así como instrucciones de programación y energía para alimentar al módulo. Fueron seleccionados 3 tipos de receptores de señales, un receptor Bluetooth, uno Wifi y recepción por medio de cable USB. En el dominio de sensores se debía detectar el porcentaje de batería del módulo principal. Para ello se proponen 3 métodos. Con revisor resistivo, medidor de voltaje y medidor de corriente y voltaje. De igual manera se proponen 3 tipos de cámaras para poder recibir imágenes, así como 3 diferentes tipos de controladores y 3 algoritmos empleados para efectuar cálculo de rutas. Estos componentes pueden apreciarse en el **Anexo 22**.

3.3.2 Módulo de Sensores para Obstáculos

Este módulo cuenta con 2 tipos de dominio. El dominio de energía al que se le hará referencia más adelante, y el dominio de sensores como tal. El dominio de sensores ofrece 3 diferentes tipos de sensores, capaces de determinar la distancia a un objeto. Estos incluyen sensores infrarrojos, de ultrasonido y LIDAR. Por otro lado, se ofrecen las mismas opciones para Sensar el porcentaje de batería del módulo. Adicionalmente se ofrecen 3 alternativas para determinar el estado del módulo y enviar dicha señal al módulo principal. Mediante detectores con imanes, detectores de voltaje o sensores a presión. Estos componentes se aprecian en el **Anexo 23**.

3.3.3 Módulo Móvil

El módulo móvil permite que el sistema pueda movilizarse. Este es el único módulo que no cuenta con alimentación propia, pues es alimentado por el módulo de control principal. Consta de 2 dominios, el dominio de sensores y el dominio de actuadores.

Para el dominio de sensores se cuenta con 2 tipos de sensores. El enconder, como única opción para determinar la velocidad rotacional del sistema, y los sensores para determinar el estado el módulo.

El dominio de actuadores incluye diferentes motores para accionar el mecanismo de desplazamiento, así como 2 configuraciones omnidireccionales que permiten el direccionamiento. Finalmente, se desplazará el móvil con 3 tipos de ruedas. Todos estos componentes pueden ser apreciados en el **Anexo 24**.

3.3.4 Módulo de Control Autónomo

Este módulo tiene 3 diferentes dominios. El dominio de energía (el cual se explicará más adelante), el dominio de sensores y el dominio de control. El dominio de control tiene 2 tareas, determinar la posición del móvil la cual se llevará a cabo por uno de 3 microcontroladores propuestos; y finalmente la tarea de calcular la ruta. Se han investigado 3 diferentes algoritmos que permitirán el cálculo de la ruta.

Adicionalmente en el dominio de sensores se encuentran los mismos 3 componentes para Sensar el porcentaje de batería del módulo, así como determinar el estado del módulo. Finalmente se cuenta con un sensor el cual determinará la posición del robot, permitiendo así el cálculo de una ruta autónoma a seguir. Estos componentes se encuentran detallados en el **Anexo 25**.

3.3.5 Módulo de Actuadores

El módulo de actuadores cuenta con 3 dominios, el dominio de energía con las mismas opciones que en los otros módulos el dominio de sensores que deberá Sensar los porcentajes de batería y el estado del módulo y finalmente el dominio de actuadores como tal.

Este último dominio ofrece 3 tipos de actuadores lineales los cuales pueden empujar obstáculos livianos en su camino. Entre ellos se encuentran un solenoide lineal, un actuador lineal hidráulico y un actuador neumático de empuje. El detalle se muestra en el **Anexo 26**.

3.3.6 Dominio de Energía

El dominio de energía consta de 3 principales funciones. La primera es energizar el sistema, función para la cual se proponen 3 tipos de baterías. La segunda función es el acondicionar el voltaje para cada componente, para lo cual se proponen 3 diferentes tipos de reguladores. Finalmente se tiene la función de detener el sistema, para ello se propone un pulsador de emergencia, una palanca y un switch. Estos componentes se pueden apreciar de mejor manera en el **Anexo 27**.

3.3.7 Dominio Mecánico

En este caso el dominio mecánico se refiere al uso de diferentes materiales para imprimir las carcasas que tendrán en su interior los diferentes módulos. Dichas carcasas serán fabricadas por medio de impresión 3D con las opciones reflejadas en el **Anexo 28**.

3.4 Conceptos de solución

En base a la matriz morfológica se desarrollaron 3 conceptos de solución. Estos contemplan una selección variada de diferentes opciones propuestas anteriormente en la matriz morfológica. Se resumen los 3 conceptos en la Tabla 3. 1 a continuación.

Tabla 3. 1 Matriz Morfológica: Energía (Fuente: Elaboración propia)

Modulo	Solución 1	Solución 2	Solución 3
Modulo Principal	Modulo WiFi	Modulo Bluetooth	Modulo Wifi
	Medidor corriente y voltaje	Divisor Resistivo	Divisor Resis
	Cámara Web	Cámara Area	Cámara Web
	Jetson	RaspBerry	Arduino
Módulo de Sensores para Obstáculos	Det Volt	Presion	Imanes
	Medidor Voltaje	Divisor Resis	Divisor Resis
	Arreglo Ultrasonido	Lidar	Lidar
Modulo Móvil	Presion	Presion	Imanes
	Encoder	Sensor Ultra	Encoder
	Motor a paso	Motor Reductor	Servo Motor
	4	3	4
	Omni	Amarillas	Mecanumm
Módulo de Control Autónomo	Det Volt	Presión	Imanes
	Medidor corriente y voltaje	Divisor Resis	Divisor Resis
	IMU	POZYX	IMU
	Arduino UNO	RaspBerry	Arduino UNO
Módulo de Actuadores de empuje	Det Volt	Presión	Imanes
	Medidor corriente y voltaje	Divisor Resis	Divisor Resis
	Solenoide	Mecanismo Piñón Cremallera	Mecanismo Biela Manivela
Dominio de Energía	Regulador	Zenner	Regulador Lineal
	Ion Litio	Lipo	Alcalina
	Switch	Palanca	Botón
Conexión Módulos	Conectores Modulares	Conectores Rápidos	Sin Conectores
Material para Carcasa	PLA	ABS	PETG

3.4.1 Concepto de solución 1

Del primer concepto de solución destaca la selección de la Jetson nano para el procesamiento de imagen de la cámara, así como los conectores modulares para mayor modularidad y la carcasa impresa a base de PLA. Así mismo se emplean conectores modulares en todos los módulos. De igual manera el arreglo de energía consta de una batería Lipo, un regulador de voltaje y un medidor de corriente y voltaje. Esto puede apreciarse en la Figura 3. 3 a continuación.

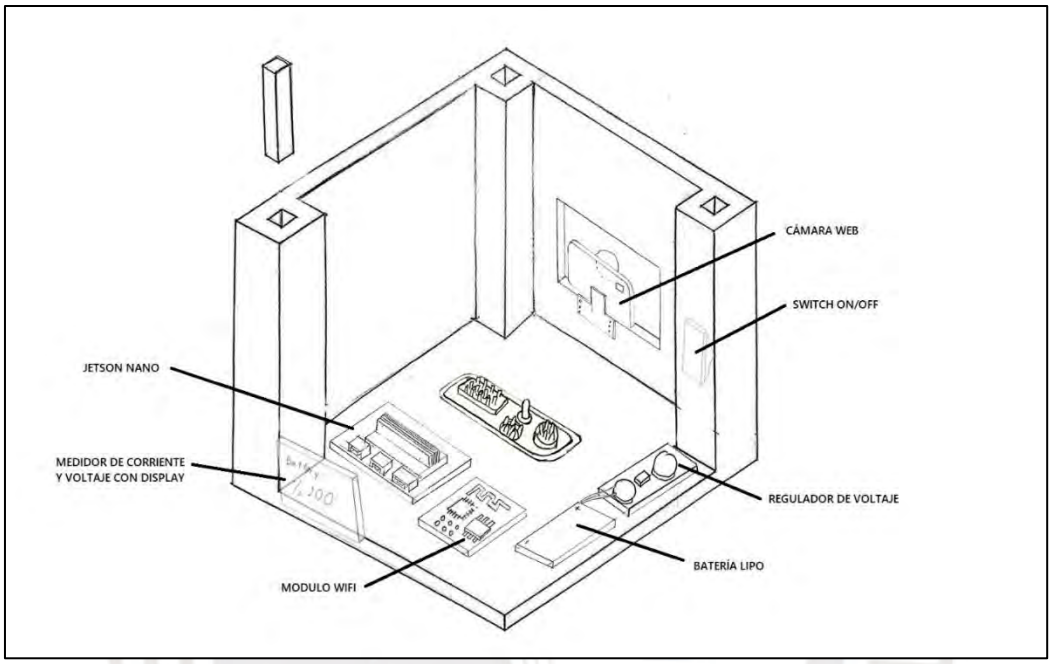


Figura 3. 3 Modulo Principal 1 (Fuente: Elaboración propia)

El módulo móvil no cuenta con sistema de alimentación, tiene sensores de presión para detectar si está conectado al módulo principal y una configuración omnidireccional de 4 ruedas mostrada en la Figura 3. 4.

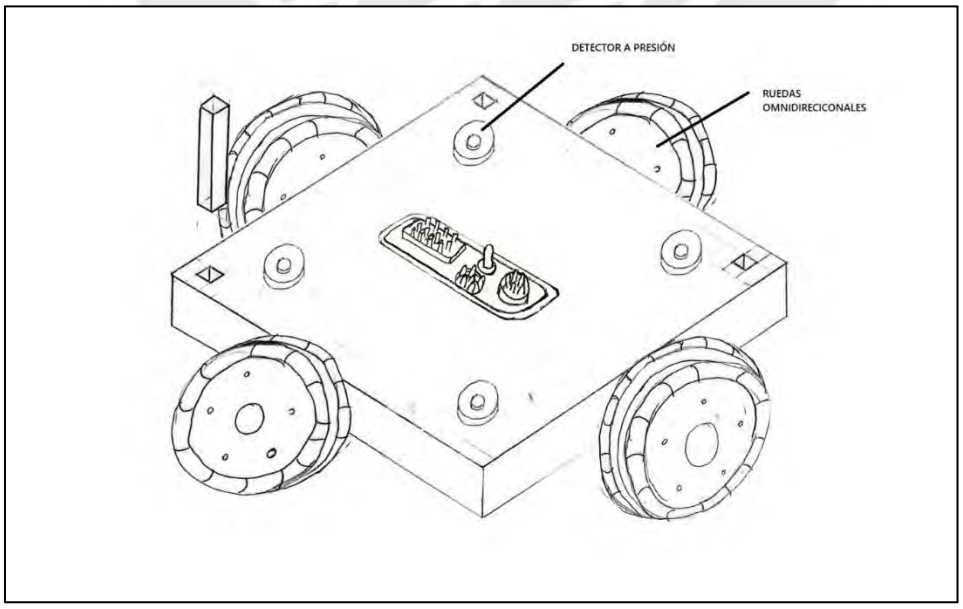


Figura 3. 4 Modulo Móvil 1 (Fuente: Elaboración propia)

En la Figura 3. 5 a continuación se puede ver el módulo móvil por abajo, destaca el uso de motores a paso para generar movimiento.

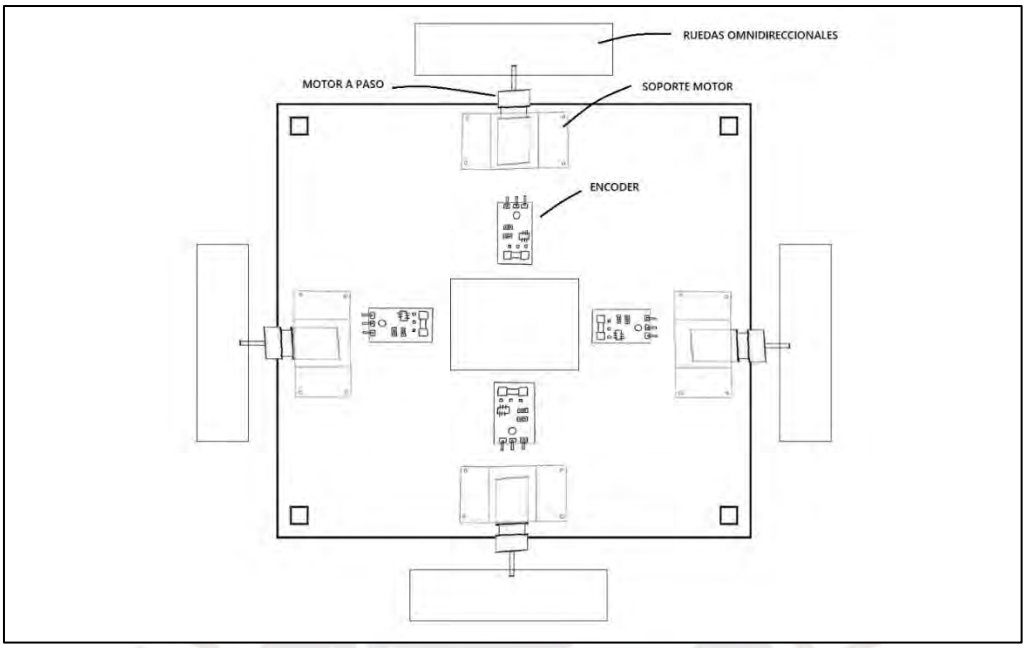


Figura 3. 5 Modulo Móvil Vista Abajo 1 (Fuente: Elaboración propia)

De manera similar el módulo de control autónomo cuenta con el mismo sistema de energía que el módulo principal. Esta vez cuenta con in IMU para determinar la posición y un Arduino para procesar datos visto en la Figura 3. 6.

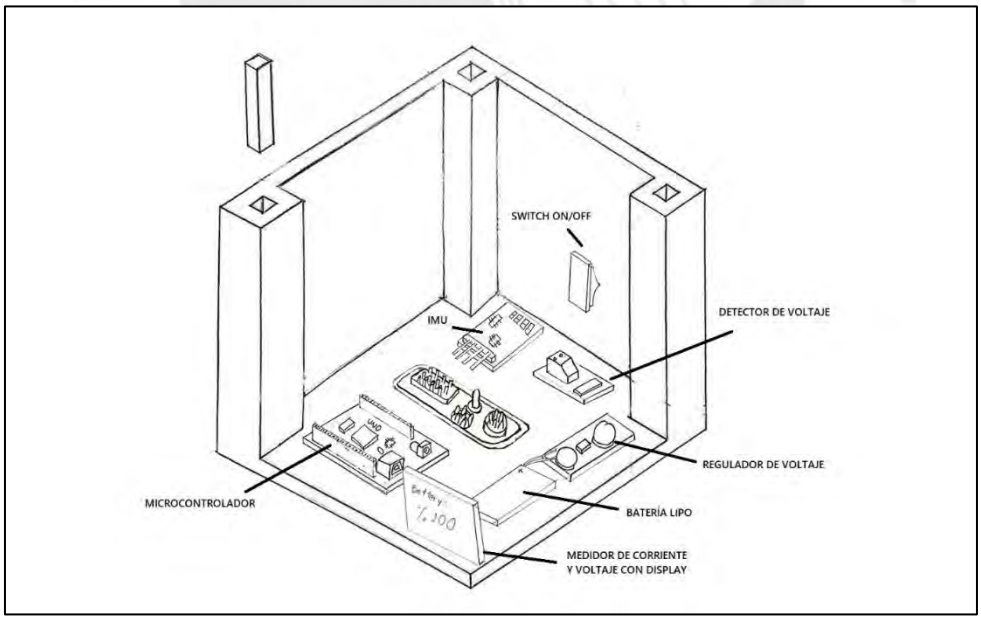


Figura 3. 6 Módulo de Control Autónomo 1 (Fuente: Elaboración propia)

El módulo de sensores cuenta con el mismo arreglo de energía, y para detectar obstáculos un arreglo de sensores ultrasónicos mostrados en la Figura 3. 7. Se detecta la conexión del modulo con detectores de voltaje.

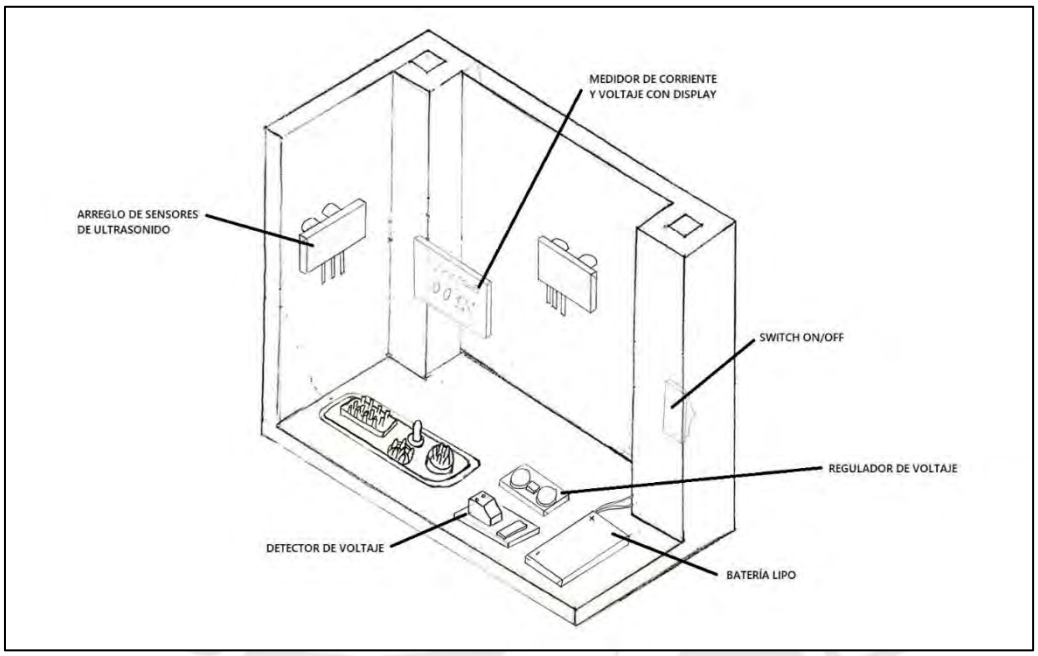


Figura 3. 7 Módulo de Sensores para Obstáculos 1 (Fuente: Elaboración propia)

Finalmente, la Figura 3. 8 muestra el módulo de actuadores para empujo, compuesto por un actuador neumático y el mismo sistema de alimentación. Se utiliza el mismo método con detectores de voltaje para determinar la conexión del módulo.

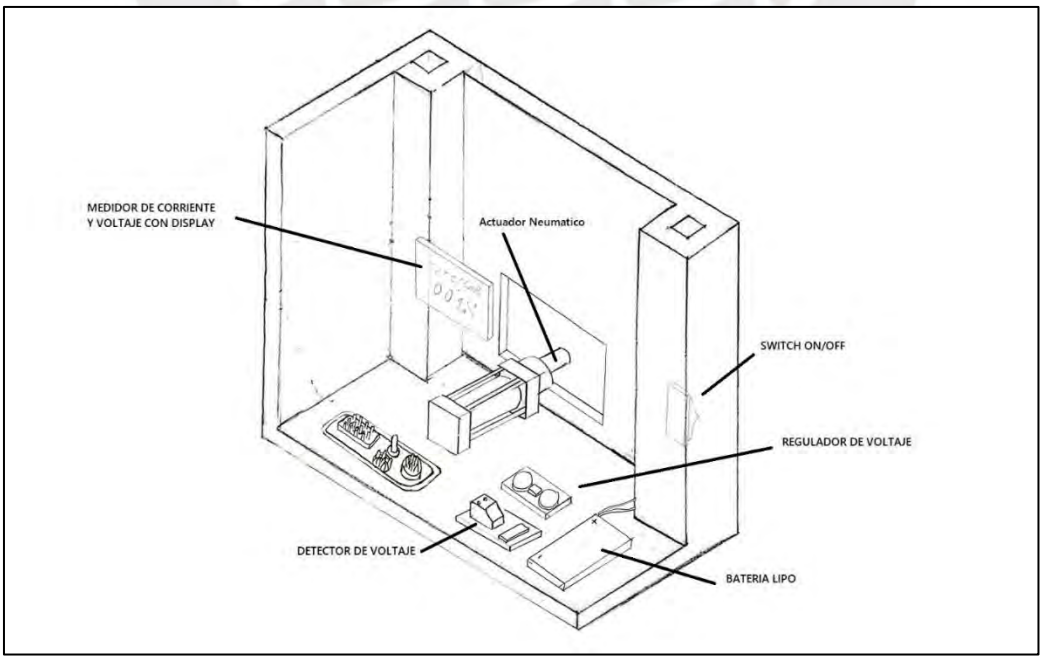


Figura 3. 8 Módulo de Actuadores de Empuje (Fuente: Elaboración propia)

3.4.2 Concepto de solución 2

Del segundo concepto de solución destaca la selección de un sistema embebido para el procesamiento de imagen de la cámara, así como los conectores modulares para mayor modularidad y la carcasa impresa a base de PLA. Así mismo el arreglo de energía consta de una batería Ion Litio, un circuito de diodo Zener para regular voltaje y un divisor resistivo para medir batería restante. Esto puede apreciarse en la Figura 3. 9 a continuación.

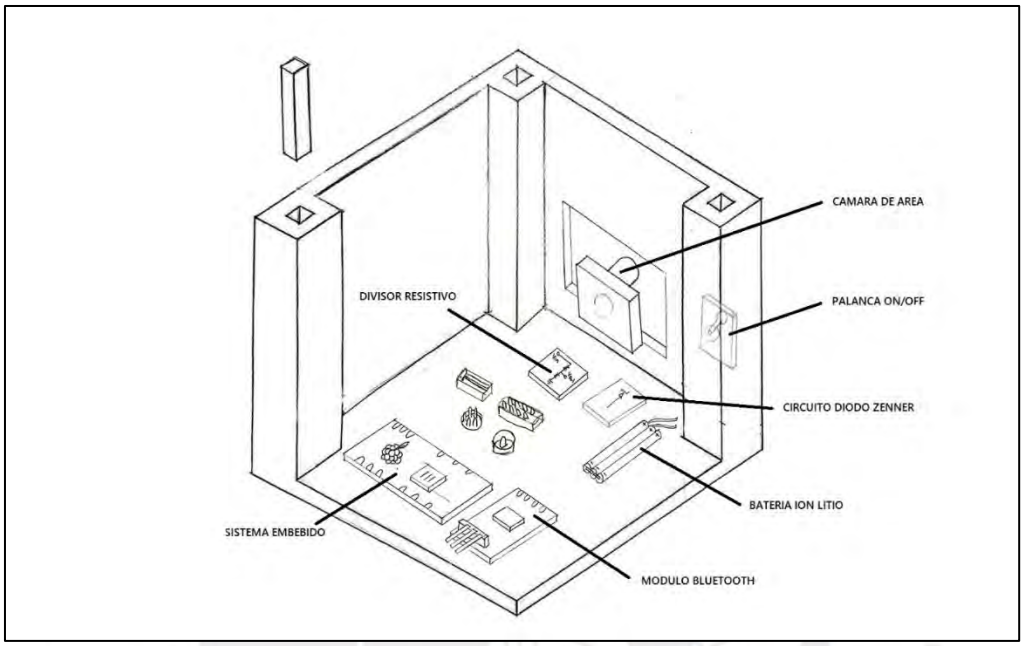


Figura 3. 9 Modulo Principal 2 (Fuente: Elaboración propia)

El módulo móvil no cuenta con sistema de alimentación, tiene sensores de presión para detectar si está conectado al módulo principal y una configuración omnidireccional esta vez de 3 ruedas mostrada en la figura Figura 3. 10.

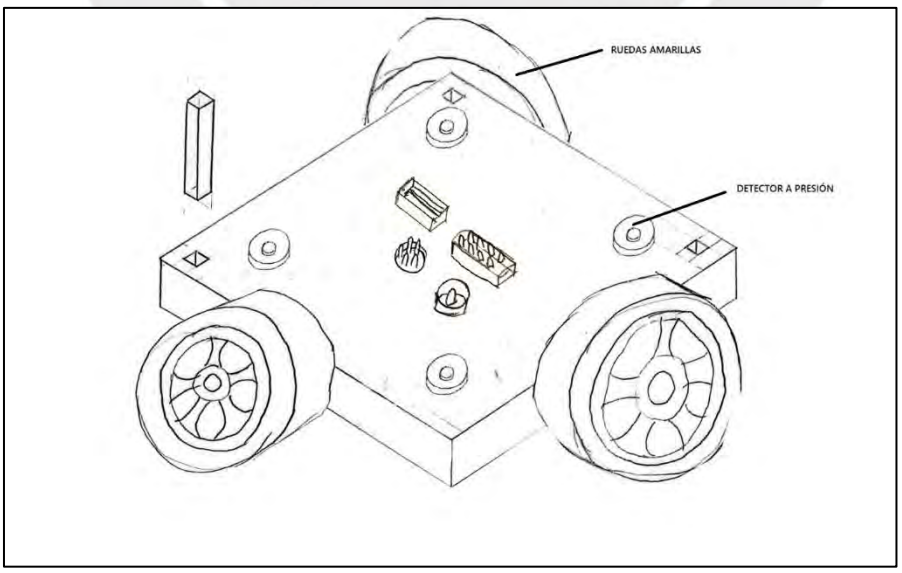


Figura 3. 10 Modulo Móvil 2 (Fuente: Elaboración propia)

En la figura Figura 3. 11 a continuación se puede ver el módulo móvil por abajo, utilizando motores reductores para realizar el movimiento.

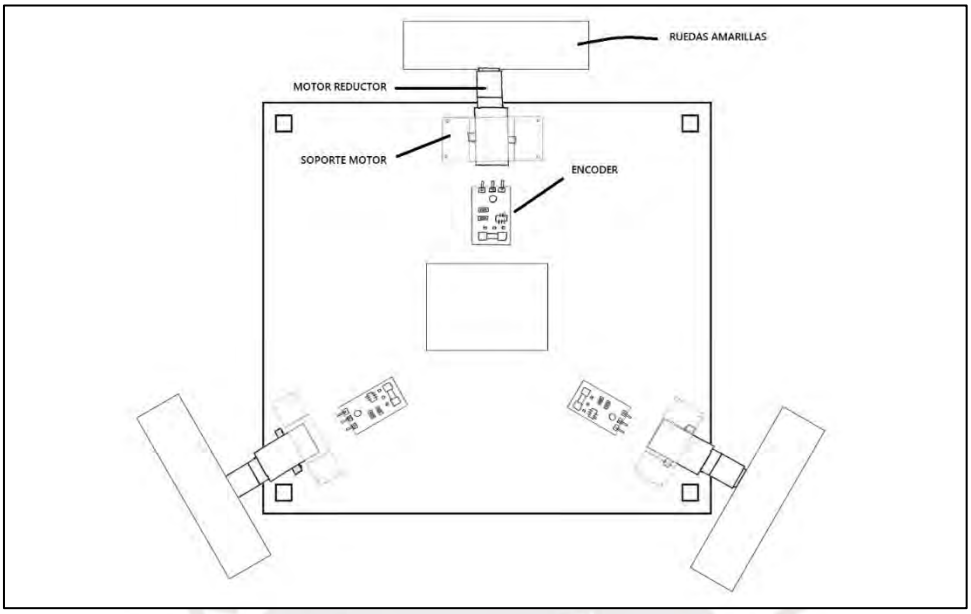


Figura 3. 11 Modulo Móvil Vista Abajo 2 (Fuente: Elaboración propia)

De manera similar el módulo de control autónomo cuenta con el mismo sistema de energía que el módulo principal. Esta vez cuenta con in POZYX para determinar la posición y un Sistema embebido para procesar datos visto en la figura Figura 3. 12.

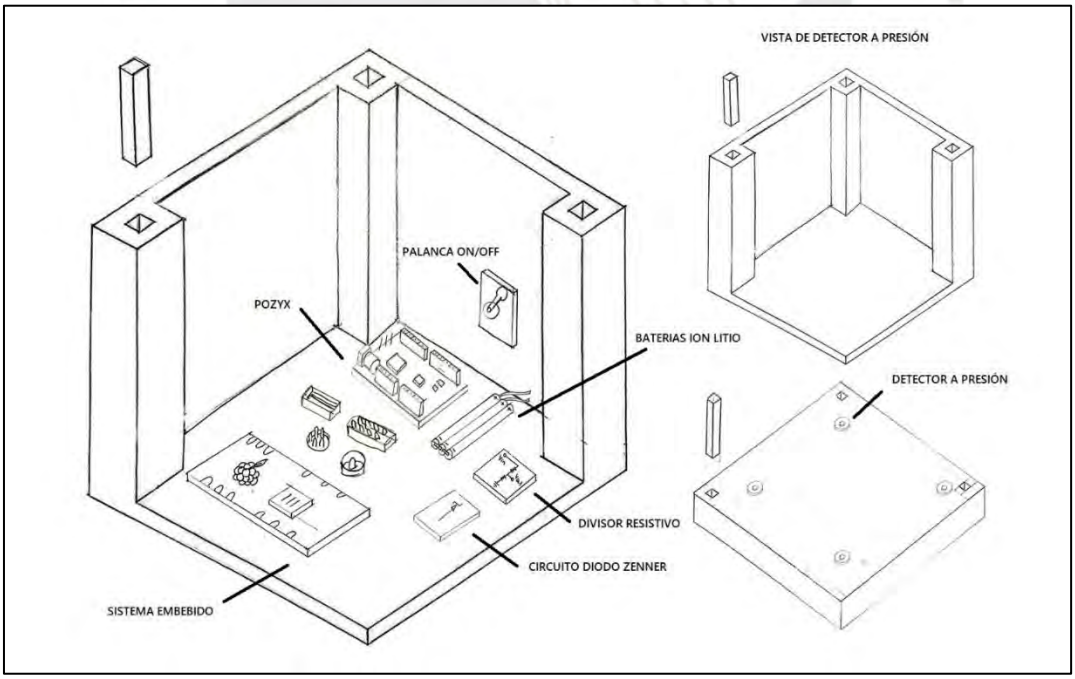


Figura 3. 12 Módulo de Control Autónomo 2 (Fuente: Elaboración propia)

El módulo de sensores cuenta con el mismo arreglo de energía, y para detectar obstáculos un arreglo de sensores utiliza un sensor LIDAR mostrado en la Figura 3. 13. Se detecta el estado del módulo con sensores de presión.

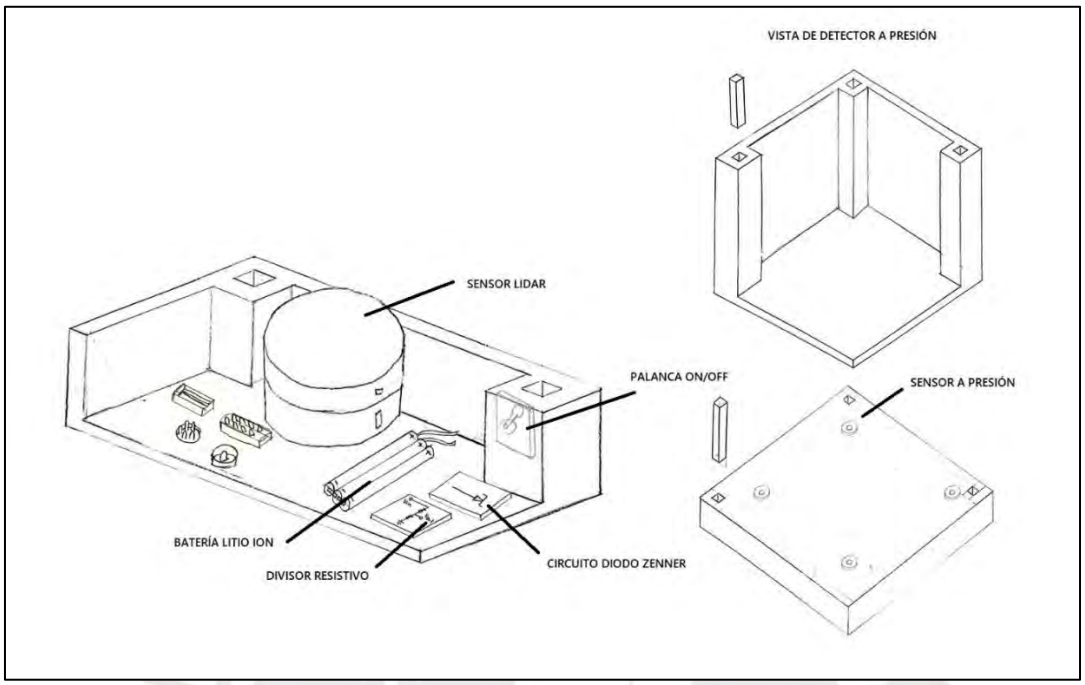


Figura 3. 13 Módulo de Sensores para obstáculos 2 (Fuente: Elaboración propia)

Finalmente, la Figura 3. 14 muestra el módulo de actuadores para empujo, compuesto por un mecanismo biela manivela y el mismo sistema de alimentación visto anteriormente.

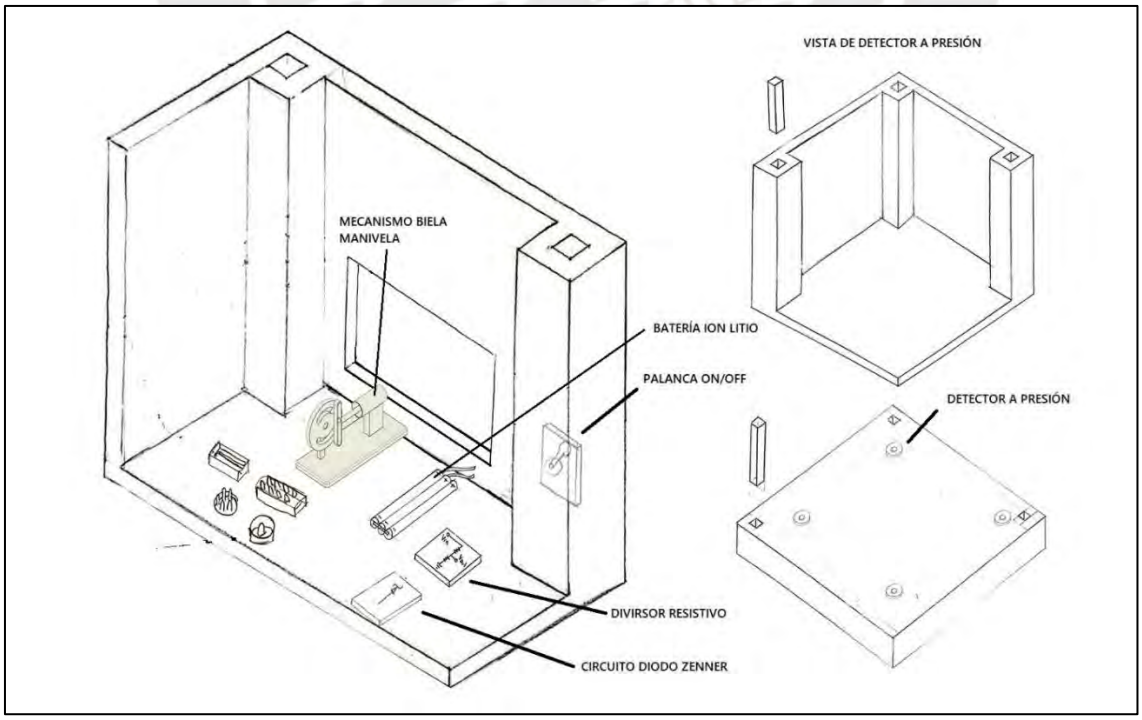


Figura 3. 14 Módulo de Actuadores para empuje 2 (Fuente: Elaboración propia)

3.4.3 Concepto de solución 3

Del ultimo concepto de solución destaca la selección de un microcontrolador para el procesamiento de imagen de la cámara, así como no usar ningún tipo de conectores ara los cables. Así mismo el arreglo de energía consta de baterías alcalinas, un regulador lineal para regular voltaje y un divisor resistivo para medir batería restante. Esto puede apreciarse en la Figura 3. 15 a continuación.

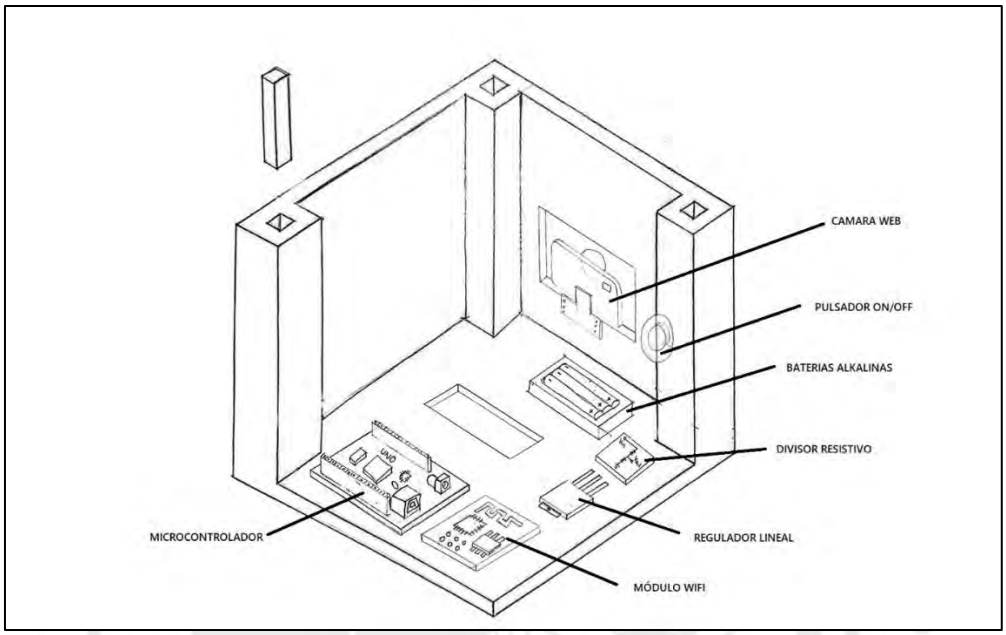


Figura 3. 15 Modulo Principal 3 (Fuente: Elaboración propia)

El módulo móvil no cuenta con sistema de alimentación, tiene sensores de presión para detectar si está conectado al módulo principal y una configuración omnidireccional esta vez de 3 ruedas Mecanum mostradas en la Figura 3. 16.

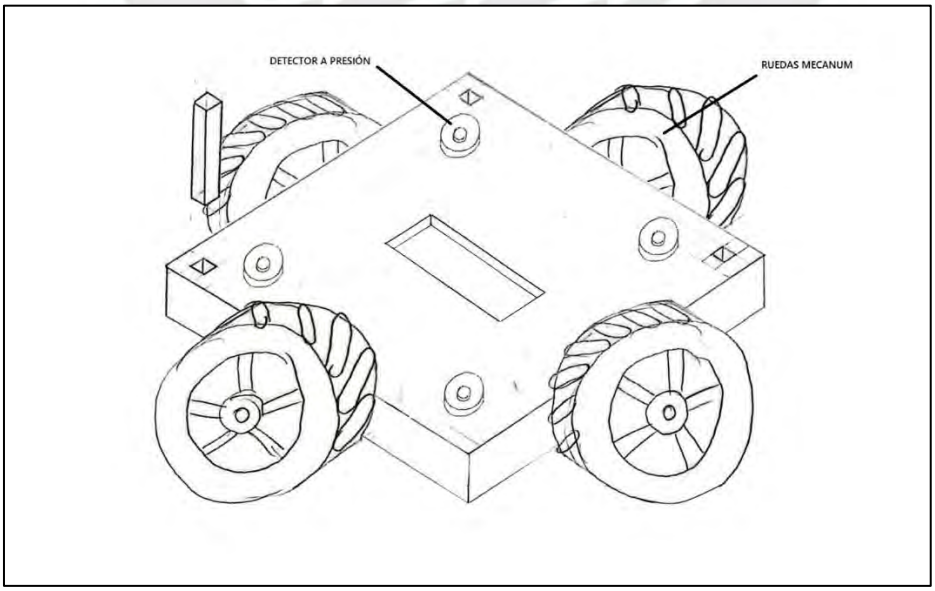


Figura 3. 16 Modulo Móvil 3 (Fuente: Elaboración propia)

En la Figura 3. 17 a continuación se puede ver el módulo móvil por abajo, utilizando servo motores para realizar el movimiento.

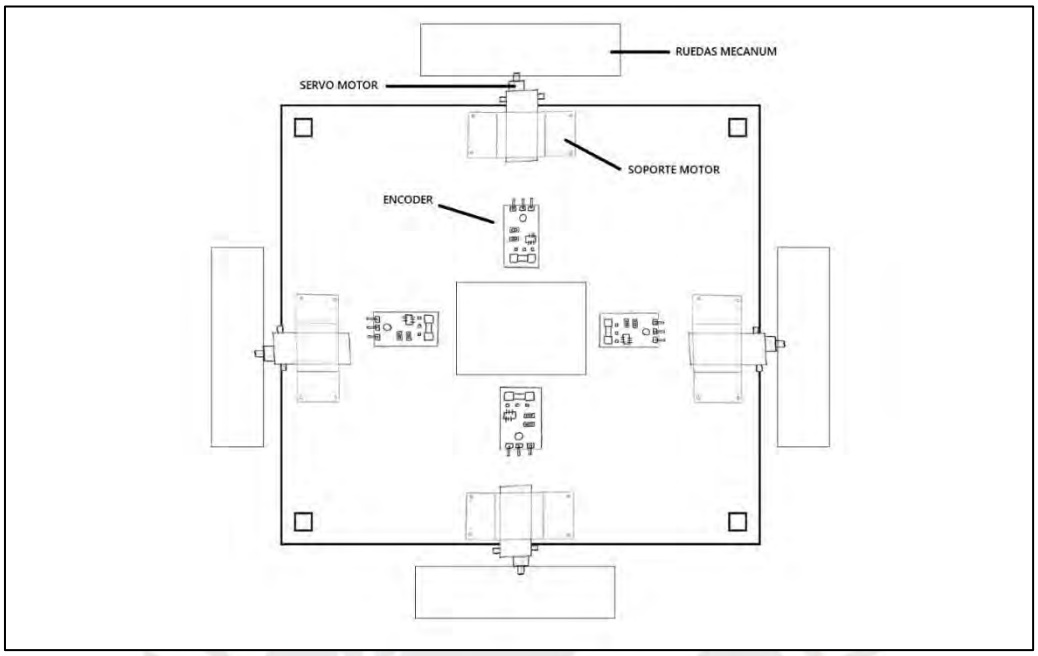


Figura 3. 17 Modulo Móvil Vista Abajo 3 (Fuente: Elaboración propia)

El módulo de control autónomo cuenta con el mismo sistema de energía que el módulo principal. Nuevamente cuenta con un IMU para determinar la posición y otro microcontrolador para procesar datos visto en la Figura 3. 18.

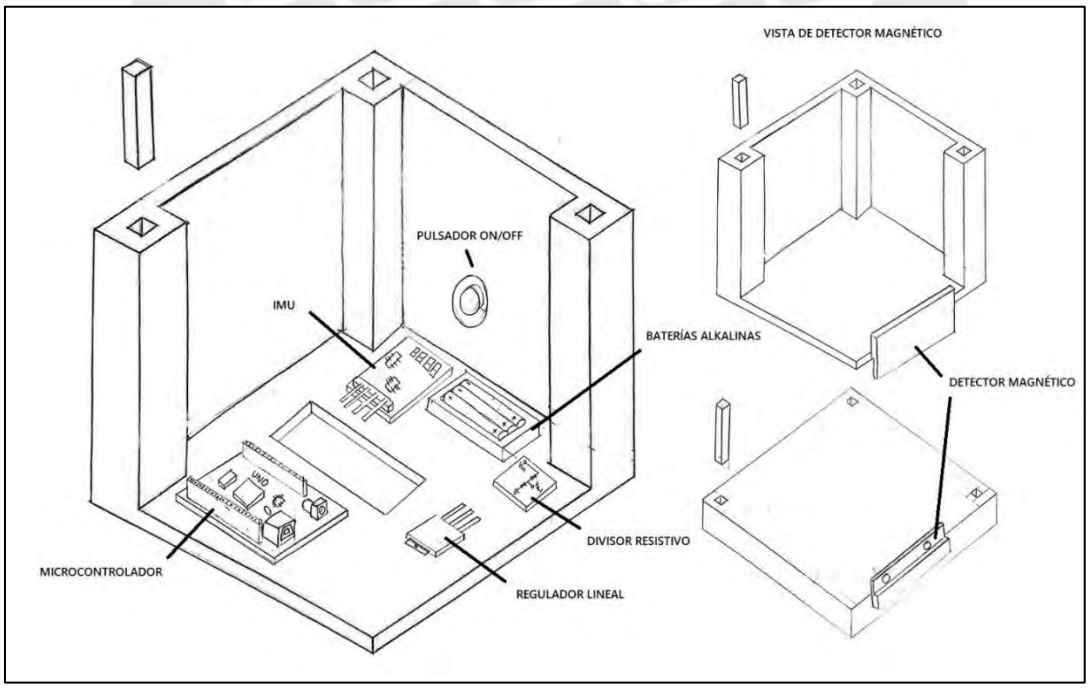


Figura 3. 18 Módulo de Control Autónomo 3 (Fuente: Elaboración propia)

El módulo de sensores cuenta con el mismo arreglo de energía, y para detectar obstáculos un sensor LIDAR mostrado en la Figura 3. 19. Adicionalmente detecta el estado del modulo con detectores magnéticos.

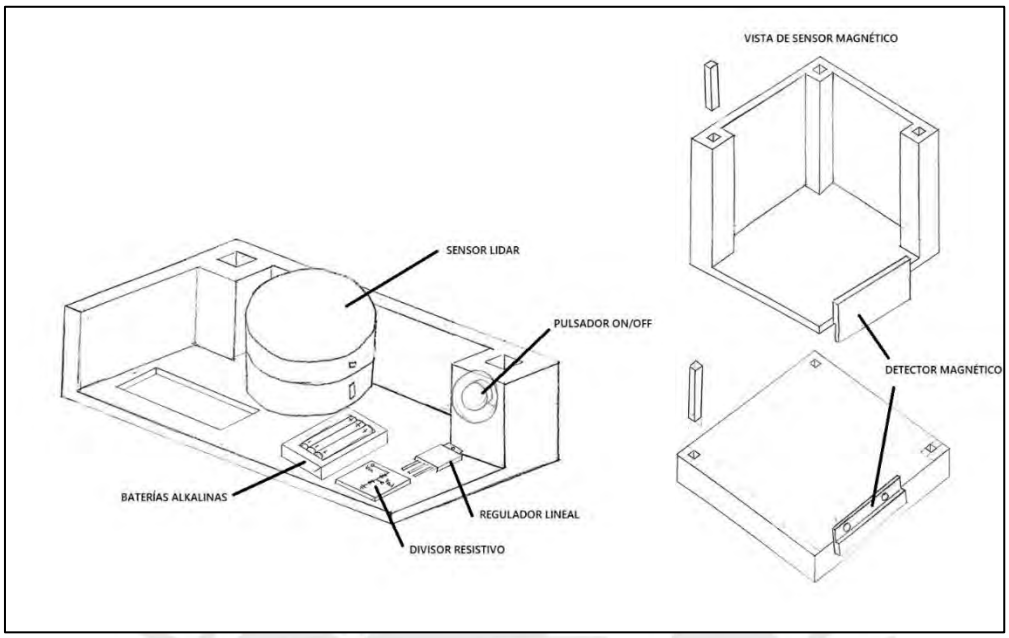


Figura 3. 19 Módulo de Sensores para obstáculos 3 (Fuente: Elaboración propia)

El módulo de actuadores para empuje cuenta con el mismo arreglo de energía, y esta vez para empujar obstáculos usa un mecanismo de Piñón cremallera. Emplea nuevamente detectores magnéticos. Esto se aprecia en la Figura 3. 20 a continuación.

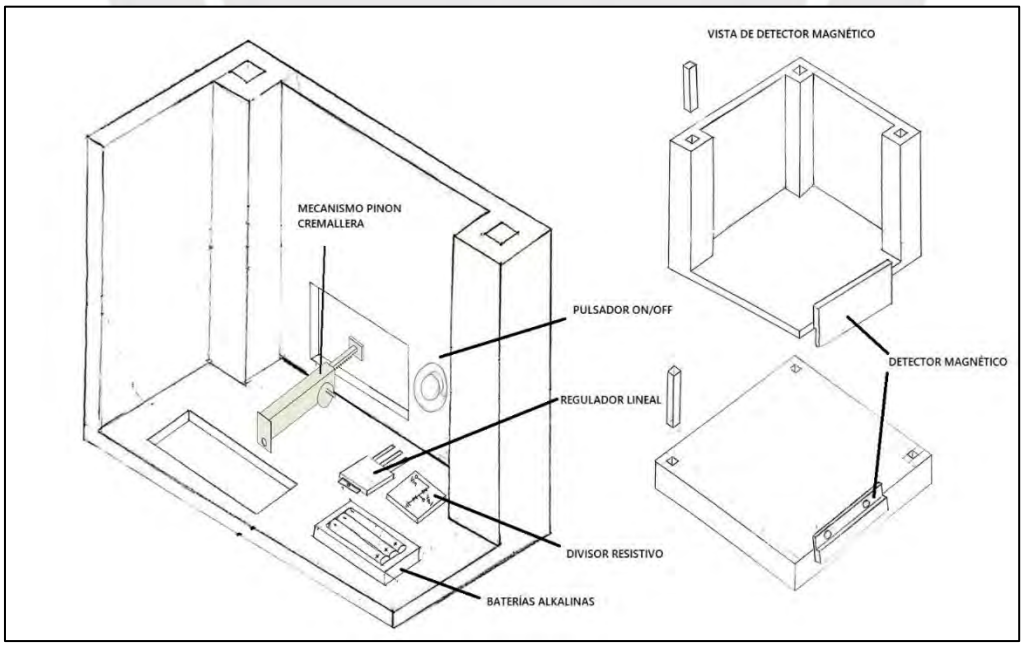


Figura 3. 20 Módulo de Actuadores para empuje 3 (Fuente: Elaboración propia)

3.5 Evaluación Técnica Económica

En la sección a continuación se presentarán los factores técnicos y económicos que se utilizaron para evaluar el desempeño de los diferentes conceptos de solución desarrollados a partir de la matriz morfológica. Se mostrará un resumen, sin embargo, se puede apreciar la versión completa en el **Anexo 30**.

3.5.1 Evaluación Técnica

En primer lugar, se tomaron criterios técnicos los cuales cambian según la selección de componentes. Estos se presentan a continuación:

- **Autonomía energética**
- **Nivel de modularidad**
- **Conectividad de señales**
- **Estabilidad mecánica**

3.5.2 Evaluación Económica

De manera similar, se evaluaron criterios económicos según su peso correspondiente. Los criterios económicos son los siguientes:

- **Costo de materiales:**
- **Disponibilidad en el mercado local:**
- **Procesos de Fabricación:**
- **Costo de mantenimiento:**

3.5.3 Gráfico Ponderado

A continuación, en la Tabla 3. 2 y la Tabla 3. 3 se resumirá lo mencionado en la evaluación técnica y la evaluación económica, a partir de lo cual se procederá a seleccionar la alternativa óptima.

Evaluación Técnica

Tabla 3. 2 Evaluación Técnica (Fuente: Elaboración propia)

Criterio técnico		Sol 1		Sol 2		Sol 3		Sol Ideal	
	g	p	gp	p	gp	p	gp	p	gp
Autonomía Energética	3	3	9	2	6	2	6	4	12
Nivel de Modularidad	2	3	6	2	4	1	2	4	8
Conectividad de Señales	3	3	9	2	6	3	9	4	12
Estabilidad Mecánica	2	3	6	2	4	3	6	4	8
Sumatoria			30		20		23		40
xi			0.75		0.5		0.575		1

Evaluación Económica

Tabla 3. 3 Evaluación Económica (Fuente: Elaboración propia)

Criterio económico		Sol 1		Sol 2		Sol 3		Sol Ideal	
	g	p	gp	p	gp	p	gp	p	gp
Costo de Mantenimiento	3	3	9	2	6	2	6	4	12
Costo de materiales	4	2	8	3	12	3	12	4	16
Disponibilidad en el mercado local	3	3	9	1	3	2	6	4	12
Procesos de Fabricación	2	3	6	3	6	2	4	4	8
Sumatoria			32		27		28		48
xi			0.667		0.563		0.583		1

Comparación gráfica:

La **¡Error! No se encuentra el origen de la referencia.** a continuación proyecta en un eje x e l puntaje ponderado de cada criterio técnico, y en un eje y el promedio ponderado de cada criterio económico. Finalmente se toma el concepto de solución más cercano a la solución ideal.

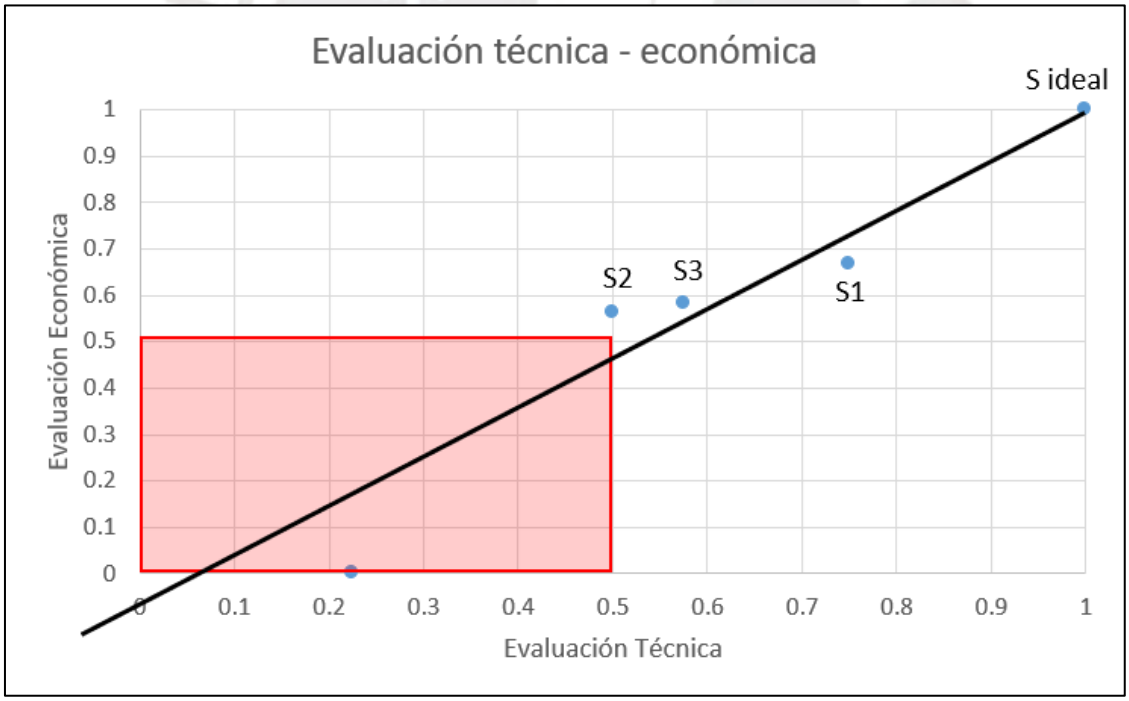


Figura 3. 21 Gráfico de Pesos Ponderados (Fuente: Elaboración propia)

4 REM Bot

4.1 Diseño Integral

En esta sección se va a explicar el proceso de diseño preliminar el denominado REM bot (Robot Educativo Modular). Este diseño preliminar abarca la división del sistema en dominio mecánico, electrónico y de software; así como la selección de componentes en base a justificaciones pertinentes.

4.1.1 Descripción general del CAD

Una vez definidos todos los conceptos de solución y realizada la evaluación técnica económica en la Figura 3. 21, se tuvo que el concepto de solución 3 visto en la sección 3.4.1 fue el óptimo. Sin embargo, la forma de las carcasas fue rediseñada para contar con una mejor distribución de componentes. Así mismo, se unifico el módulo móvil y el módulo principal, llamándolos módulo principal simplemente. Por último, se decidió que el módulo principal albergará un sistema de alimentación para todos los módulos, y se cambió la detección de módulos utilizando la comunicación I2C y Arduinos nano en cada módulo.

De esta manera se llega al diseño integral se puede ver de manera ilustrativa en la Figura 4. 1 a continuación. Se pueden apreciar los 4 módulos. El módulo principal es de color rojo, el módulo de Control de color celeste, el módulo de sensores de color amarillo y finalmente el módulo de actuadores de color naranja. El plano de ensamble de los módulos se puede apreciar en el

Anexo 75.

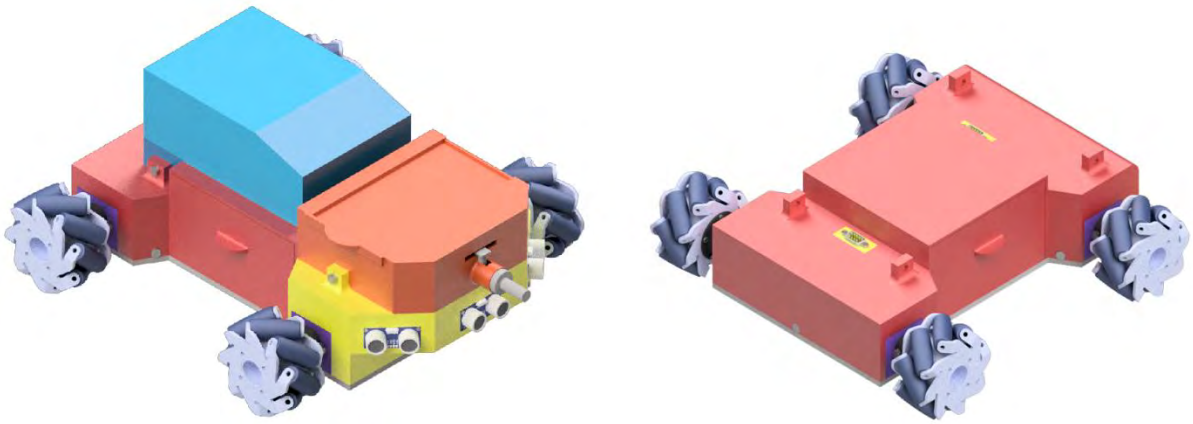


Figura 4. 1 Módulo Principal (Fuente: Elaboración propia)

4.1.2 Descripción del funcionamiento del Sistema

Para poder comunicarse, se emplea un protocolo de comunicación llamada I2C. Este consta en un dispositivo maestro, el cual puede enviar señales y controlar por un mismo puerto a múltiples dispositivos esclavos. En este caso el dispositivo maestro se encuentra en el módulo de control, siendo este la Jetson nano. Por medio de las conexiones DB9, es posible simplifica la conexión de los pines en un solo conector que el usuario puede conectar sin dificultad.

Este es un cambio importante a considerar, pues en el concepto de solución 1 de la sección 3.4.1 contemplaba detectores de voltaje para determinar si un módulo estaba conectado o no. El utilizar el protocolo de comunicación I2C permite realizar dicha detección disminuyendo el consumo de energía y el costo de componentes adicionales innecesarios.

Otro cambio a considerar es el uso de 2 drivers de motores, cada uno para controlar 2 motores. Estos son albergados en el módulo principal. De igual manera, el arreglo omnidireccional ahora emplea ruedas mecanum. Estas facilitan el avanzar en cualquier dirección en el plano sin tener que rotar.

La función más simple del robot es ser tele operado. Para conseguir esto no es necesario que estén conectados los módulos de sensores ni de actuadores, pero si el módulo de control pues este maneja a los drivers que manejan los motores mediante el controlador principal. El módulo de control mediante el IMU añade también la posibilidad de navegar de manera autónoma. Permitiendo al usuario dar al robot coordenadas a donde debería desplazarse.

Al agregar el módulo de sensores, el robot no solo podrá desplazarse de un punto a otro, sino que también podrá detectar y evadir cualquier obstáculo en su camino. Adicionalmente, el módulo de actuadores le proporciona al robot un generador de movimiento lineal. El propósito de este actuador depende de la creatividad del usuario. Puede ser usado como empujador de algún botón, o se le puede agregar un sello en la punta para experimentar estampando el sello sobre superficies.

4.1.3 Diagrama de operaciones (interacción entre componentes del sistema)

A continuación, la Figura 4. 2 presenta el diagrama de funcionamiento. Este diagrama permite tener una idea más clara de la manera en la que se integra el funcionamiento del sistema. Cabe recalcar que existen 4 maneras de manejar el robot. Siendo la primera tele operada, seguida por seguimiento de linea con la cámara (ruta global), luego de manera semi automática con el módulo de sensores (ruta global) y finalmente de manera totalmente autónoma con el módulo de control autónomo (ruta local). El usuario determina que configuración desea utilizar.

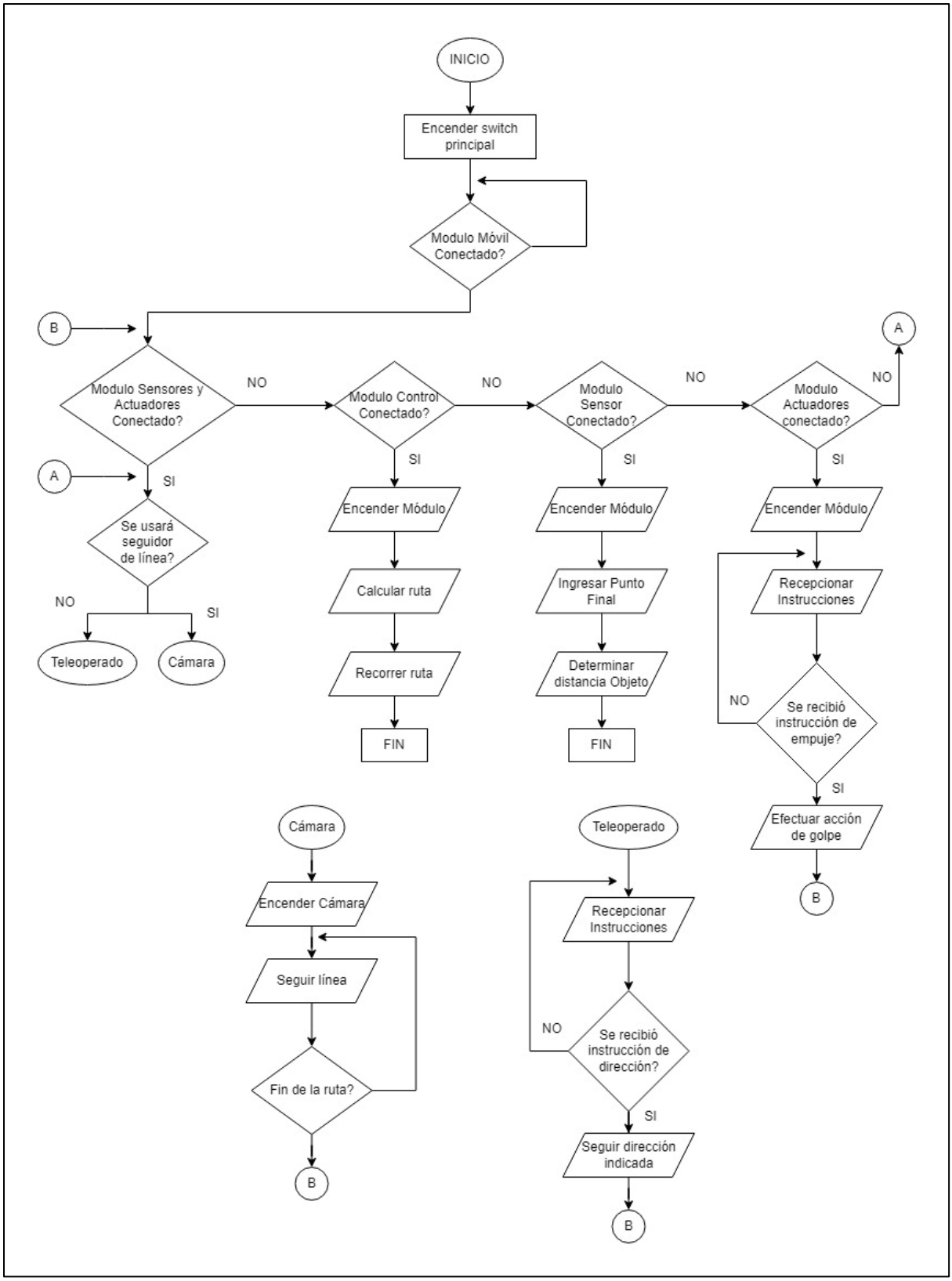


Figura 4. 2 Diagrama de funcionamiento (Fuente: Elaboración propia)

4.2 Diseño Mecánico

La sección a continuación comprende el diseño mecánico del sistema.

4.2.1 Cálculo de Torque

A continuación de calcula el torque requerido. Lo primero a realizar es estimar el peso total del sistema. Se utilizará para el cálculo los componentes de mayor peso, considerando un factor de seguridad de 1.5. Primero se procede a estimar el peso de cada componente en la Tabla 4. 1 a continuación.

Tabla 4. 1 Calculo de Peso Total

Componente	Cantidad	Peso(gr) / Unidad	Peso(gr)
Rueda	4	108	432
Motor	4	165	660
Carcasa Modulo Principal	1	748	748
Carcasa Módulo Control	1	200	200
Driver de Motores	2	25	50
Arduino Uno	2	40	80
Arduino Nano	3	6	18
Demas Componentes	1	200	200
Peso Total			2,388

De esta manera se puede estimar un peso total de 2,388 gramos. Así mismo, considerando que el robot será utilizado en un ambiente controlado dentro de un laboratorio, se considerará una velocidad máxima de 2 m/s. Ya que el robot será utilizado en ambientes de laboratorio cerrados los cuales no siempre son muy espaciosos.

Una vez conocido el peso, asumimos que el sistema subirá por una pendiente máxima de 20 grados. Considerando un coeficiente de rozamiento de 0.2, ruedas con radio de 4 cm y un factor de seguridad de 1.5, se procede a calcular el Torque total necesario en N.m. Se utilizará la siguiente fórmula (Fuerza total*Radio*Factor de Seguridad). Esto se puede ver en la Tabla 4. 2 adjunta a continuación.

Tabla 4. 2 Calculo de Torque Mínimo Necesario

Datos	Cantidad
Peso Robot (N)	23.43
Fuerza Perpendicular (N)	8
Fuerza de Fricción (N)	4.4
Fuerza Total (N)	12.4
Radio Ruedas (m)	0.004
Factor de Seguridad	1.5
Torque (N.m)	0.074
Torque (Kg.cm)	0.76
Torque/motor (Kg.cm)	0.19

De esta manera obtenemos que el torque total mínimo requerido será de 0.074 N.m o 0.76 Kg.cm. De esta manera se podrá elegir un motor apropiado.

4.2.2 Módulo Principal

Selección de Motores

Se procedió a seleccionar la opción del motor 37D con enconder, debido a que no solo cumple con las especificaciones de Torque mencionadas anteriormente (torque mínimo de 0.19 Kg.cm) sino que también incluye un enconder. Este enconder permitirá realizar los algoritmos de control necesarios para coordinar los 4 motores necesarios para la configuración omnidireccional especificada en los requerimientos del sistema. Esto se aprecia en la Tabla 4. 3 a continuación.

Tabla 4. 3 Selección de Motores

Modelo	MOTOR DC 37D 12V/300RPM CON ENCODER	MOTOR DC 37D 12V/300RPM	MICROMOTOR DC N20 12V/300RPM CON ENCODER
Imagen de referencia			
Diámetro x Longitud (mm)	25 x 70	37 x 60	12 x 40
Voltaje Operacional (V)	12	12	12
RPM	300	300	300
Torque (N.mm)	27.5	27.5	-
Torque (Kg.cm)	0.28	0.28	-
Encoder	Si	No	Si
Precio (soles)	70	65	50
Referencia	Naylamp	Naylamp	Mouser

Selección de Ruedas

Se eligieron las ruedas mecanum debido a su capacidad para proporcionar una configuración omnidireccional, lo que permite al robot moverse en cualquier dirección sin necesidad de rotar sobre su eje.

Esta versatilidad es fundamental para maximizar la maniobrabilidad en espacios reducidos. Además, las ruedas mecanum permiten un diseño más compacto, lo que se alinea con las especificaciones de tamaño máximo de 40 cm de ancho y alto. Gracias a su disposición, estas ruedas son ideales para crear un sistema que cumpla con los requisitos de movilidad y dimensiones buscadas. Las especificaciones se aprecian en la Tabla 4. 4 a continuación.

Tabla 4. 4 Selección de Ruedas del Módulo Móvil

Modelo	Mecanum	Omnidireccional	Amarilla
Imagen de referencia			
Diámetro (mm)	80	58	65
Peso (gr)	3.3-5	-	-
¿Mecanismo Adicional Necesario?	No	No	Si
Precio (soles)	70	115	20
Referencia	Naylamp	Mercado Libre	Naylamp

Modelado del subsistema de movimiento

Para poder conseguir una configuración omnidireccional, se optó por utilizar 4 motores. Anterior mente se seleccionaron cuidadosamente los motores para así poder cumplir con el Torque necesario considerando tanto el peso del robot como la inclinación máxima a la que estará sometida el mismo. La disposición de los motores puede verse en la Figura 4. 3 a continuación. Para mayor detalle en cuanto a las piezas utilizadas y sus dimensiones consultar del **Anexo 43** al **Anexo 61**. Siendo el **Anexo 61** el plano de ensamble del módulo principal.

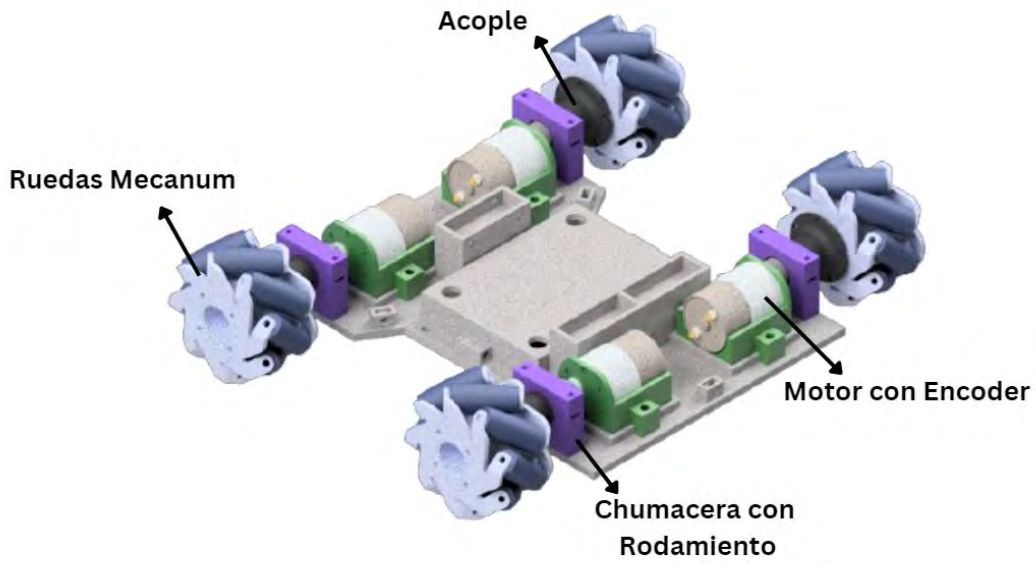


Figura 4. 3 Modelo 3D del subsistema de Movimiento

Para sujetar las ruedas al motor se diseñó un acople de 2 partes, una va atornillada a los agujeros existentes en las ruedas mecanum, y la otra va directo al motor con un mecanismo de prisionero. Este se puede ver a mayor detalle en la siguiente Figura 4. 4. Finalmente, se une ambas partes por medio de un agujero que conecta ambos acoples, por el cual se utiliza un tornillo largo que atraviesa ambas partes, para luego ser ajustado por una tuerca.

Figura 4. 4 Acople Motor Rueda

Adicionalmente, se agregaron rodamientos para que el largo del acople del motor no haga que el mismo pandee. Estos rodamientos son apreciables en la Figura 4. 3 anteriormente vista, destacados por su color morado.

Selección de Controlador de Motores

Para el controlador de los motores, se decidió ir por algo simple de programar. Por esta razón se descartó el STP32. Por otro lado, el Arduino MEGA por si solo no cuenta con suficientes puertos seriales para realizar la conexión de 2 drivers a la vez. Debido a estas razones, se optó por usar 2 Arduino como controlador principal.

Al necesitar 3 conexiones por motor, más las conexiones de los Encoders los pines del Arduino 1 no serían suficientes. Así mismo, en el mercado existen Drivers que se adaptan al Arduino Uno, mas no al Arduino Mega. Esto se puede apreciar en la tabla Tabla 3. 4 a continuación.

Tabla 3. 4 Selección de Controlador para Motores

Modelo	**Arduino MEGA**	**Arduino UNO**	**Micro Controlador STM32**
Imagen de referencia			
Dimensiones (mm x mm x mm)	108 x 53 x 13	73 x 53 x13	51 x 21 x 20
Voltaje Operacional (V)	3.3-5	5	3.3-5
Numero de pines I/O	54	14	32
Precio (soles)	135	65	55.45
Referencia	Naylamp	Naylamp	Mouser

Selección de Driver de Motores

En la Tabla 3. 5 a continuación se muestran 3 diferentes drivers de motores. A pesar de ser el más costoso, se decidió optar por el driver de motores VNH, debido a la corriente máxima que soporta. Esta es el único de los 3 drivers que podría soportar las corrientes máximas que brindaría el motor seleccionado anteriormente. Los demás drivers soportarían las corrientes de funcionamiento continuo, sin embargo corren riesgo de quemarse de presentarse picos mayores al límite del driver, lo cual es muy probable.

Tabla 3. 5 Selección de Drivers de Motores

Modelo	DRIVER PUENTE H L298N 2A	DRIVER PUENTE H VNH5019	DRIVER PUENTE H L9110S 0.8A
Imagen de referencia			
Dimensiones (mm x mm x mm)	43 x 43 x 27	38 x 28 x 25	-
Voltaje Operacional (V)	5 - 35	24	2.5 - 12
Motores Simultáneos	2	2	1
Corriente Máxima (A)	2	15	1.5
Precio (soles)	15	48	12
Referencia	Naylamp	Naylamp	Naylamp

Selección de Conector

El conector es lo que brindará modularidad, puesto que por medio de el, se simplificará la conexión entre diferentes componentes. El conector bullet implicaría múltiples conectores diferentes. Por otro lado, el conector molex ofrece más puertos en una sola conexión. Sin embargo, el conector DB9 permitiría realizar múltiples conexiones de una sola vez, ofreciendo más pines por una sola conexión. De esta manera se cumpliría el requerimiento que hace referencia a la facilidad que debe haber en las conexiones del sistema. Esto se evidencia en la Tabla 3. 6 a continuación.

Tabla 3. 6 Selección de Conector

Modelo	DB9	Conector Mollex	Connector Bullet
Imagen de referencia			
Diámetro (mm x mm x mm)	30 x 10 x 9	-	-
Pines	8	3	1
Precio (soles)	20	20	30
Referencia	Mercado Libre	Mercado Libre	Amazon

Selección de Mando

Para poder controlar el sistema de la manera más simple se utilizará un control de juegos. En el caso del mando Logitech, este es únicamente cableado. Por otro lado, el control de Ribonix a pesar de ser más barato no es muy accesible en el mercado local. Es por ello que se eligió el mando de PS4, debido a la accesibilidad y a que tiene tanto conexión cableada como bluetooth. Las especificaciones se ven evidenciadas en la Tabla 3. 7 a continuación.

Tabla 3. 7 Selección de Mando

Modelo	**Mando PS4**	**Riboxin B07PQ62D7V**	**Logitech F310 b**
Imagen de referencia			
Conexión	Cableado USB / Bluetooth	Bluetooth	Cableado USB
Precio (soles)	133	72	104
Referencia	Mercado Libre	Amazon	Mercado Libre

Selección de Material de Carcasa

El material seleccionado para la impresión de carcasas será PLA. Sin embargo, entre todas las marcas se seleccionará la marca Esun. Debido a su buen precio en relación a la calidad, así como su accesibilidad en el mercado local. Las diferentes marcas se ven ejemplificadas en la Tabla 3. 8 a continuación.

Tabla 3. 8 Selección de Material de Impresión

Modelo	ESUN	ANYCUBIC	SUNLU
Imagen de referencia			
Diametro (mm)	1.75	1.75	1.75
Peso (gr)	1000	1000	1000
Minima – Máxima Temp Extrussión °C	205 - 225	190 - 230	190 - 220
Margen de Tolerancia del Diametro	±0.01	±0.02	±0.02
Precio (Soles)	70	85	65
Referencia	Tienda Crear	Amazon	Mercado Libre

Selección de Regulador de Voltaje

El regulador de voltaje seleccionado es el Step Down de 5 Amperios. No solo es una opción económica, sino que cumple con las corrientes máximas que utilizarán los diferentes componentes electrónicos que necesitan conversión de voltaje al mismo tiempo. Este mismo regulador se utilizará en todos los módulos. Los detalles de las diferentes opciones se ven en la Tabla 3. 9 a continuación.

Tabla 3. 9 Selección de Regulador de Voltaje

Modelo	CONVERTIDOR VOLTAJE DC-DC STEPDOWN 5A XL4005	CONVERTIDOR VOLTAJE DC-DC STEP-UP-DOWN 1.5A XL6009	CONVERTIDOR VOLTAJE DC-DC STEP-DOWN 2A MP2315 CON SELECCIONADOR
Imagen de referencia			
Dimensiones (mm x mm x mm)	43 x 21 x 13	54 x 23 x 12	22 x 17 x 4
Corriente Funcionamiento (A)	5	1.5	2
Rango Vin	5 - 32	3.5 - 32	4.5 - 15
Rango Vout	0.8 - 30	1.25 - 35	0.8 - 12
Precio (soles)	15	18	6
Referencia	Naylamp	Naylamp	Naylamp

Cabe mencionar que las conexiones electrónicas correspondientes al módulo principal, se pueden apreciar en la Figura 4. 15 Diagrama de Conexiones Principal.

Modelado del subsistema Principal

El módulo principal está compuesto por 3 tapas removibles, ajustables por medio de tornillos. De color amarillo se puede denotar los conectores DB9, los cuales permiten interconectar todas las señales por medio de una sola conexión. Para mayor detalle en cuanto a las piezas utilizadas y sus dimensiones consultar del **Anexo 43** al **Anexo 61**. Siendo el **Anexo 61** el plano de ensamble del módulo principal. En la Figura 4. 5 y Figura 4. 6 a continuación se puede apreciar el módulo principal.

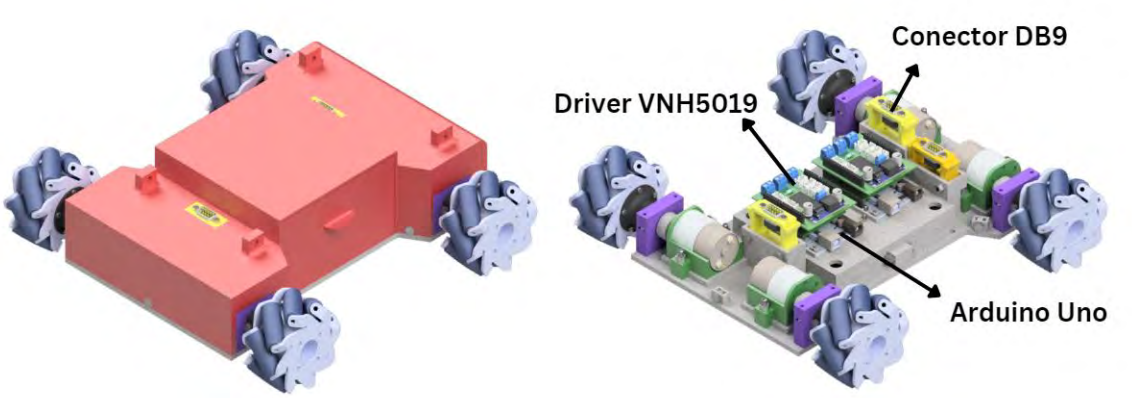


Figura 4. 5 Modelado de Módulo Principal

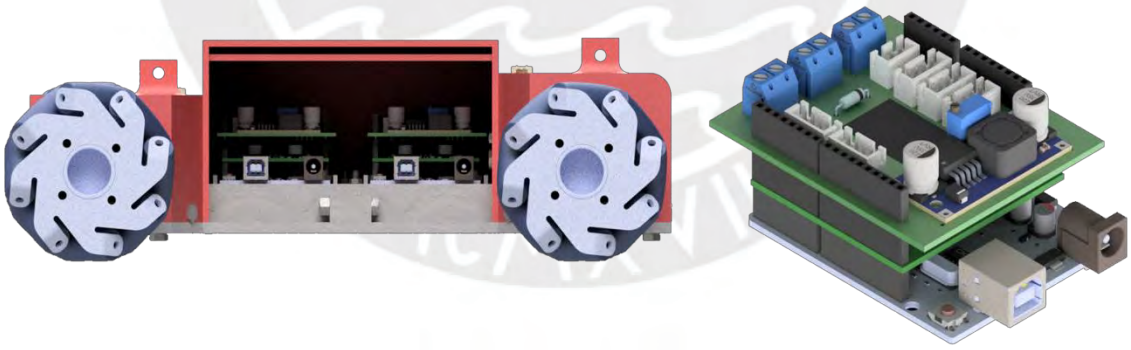


Figura 4. 6 Modelo de Módulo Principal Vista Lateral

En el centro resaltan los 2 Arduino, quienes por encima llevan un driver para controlar a sus respectivos 2 motores, así como un elemento azul el cual representa la PCB del módulo. Los componentes electrónicos interiores se pueden apreciar de mejor manera en la Figura 4. 5.

4.2.3 Módulo de Control Autónomo

Selección de detector de conexión

Para la detección del módulo, se utilizará un Arduino nano, para así poder implementar el protocolo de comunicación I2C. Este protocolo constantemente esperará una respuesta por parte de cada módulo, solo al recibir una respuesta procederá a utilizarlo. Será más barato que utilizar el sensor magnético, de igual manera será más efectivo y sencillo de programar que con el sensor de voltaje. Esto se ve en la Tabla 3. 10 a continuación.

Tabla 3. 10 Selección de detector de conexión

Modelo	Arduino Nano	Sensor Contacto Magnético	Sensor de Voltaje
Imagen de referencia			
Dimensiones (mm x mm x mm)	18.5 x 43.2 x 20	107 x 50 x 16	-
Voltaje de Funcionamiento (V)	7 - 12	-	0 - 25
Forma de Detección	Conexión I2C	Señal Enviada por medio de detección Magéntica	Detección de Voltake (0.02445V – 25V)
Precio (soles)	35	69	2.5
Referencia	Naylamp	Falabela	Unit Electronics

Selección de Controlador

Para el proyecto en cuestión, se requiere un controlador que no solo tenga una cantidad adecuada de pines digitales disponibles en un solo controlador, sino que también sea capaz de procesar imágenes, lo cual es esencial para implementar un sistema de seguimiento de línea mediante una cámara. En este sentido, la Jetson Nano se presenta como la solución ideal, ya que ofrece un mayor número de pines digitales en comparación con otras opciones como la Raspberry Pi o el Arduino. Además, la Jetson Nano tiene una capacidad de procesamiento de imágenes superior, lo que permite integrar fácilmente una cámara para llevar a cabo tareas de visión por computadora, como el seguimiento de línea. Esta combinación de características hace que la Jetson Nano sea la opción más adecuada para este tipo de aplicación.
Esto se ve en la Tabla 3. 11 a continuación.

Tabla 3. 11 Selección de Controlador

Modelo	Raspberry PI	Arduino	Jetson Nano
Imagen de referencia			
Dimensiones (mm x mm x mm)	86 x 57 x 30	73 x 53 x13	100 x 80 x 29
Voltaje Operacional (V)	3.3-5	5	5
Numero de Pines Digitales	17	14	28
Numero de Pines Analógicos	-	6	-
Frecuencia de Procesamiento (G Hz)	1.5	0.002	5
Precio (soles)	345	65	750
Referencia	Naylamp	Naylamp	Naylamp

Selección de IMU

Para el proyecto, se requiere un IMU (Unidad de Medición Inercial) que proporcione datos precisos de orientación y movimiento a un costo accesible. Si bien varios modelos cumplen con características y especificaciones similares, el MPU6050 es una opción que destaca por su fiabilidad. Sin embargo, el modelo MPU6500 ofrece un rendimiento prácticamente equivalente a un precio considerablemente más bajo. Dado que uno de los requisitos principales del proyecto es mantener un costo reducido, se opta por el MPU6500 como la solución más adecuada. Las características comparativas de ambos modelos se pueden ver detalladamente en la Tabla 3. 12 a continuación.

Tabla 3. 12 Selección de IMU

Modelo	**MPU 6050**	**MPU GY-91 MPU9250**	**MPU9250**
Imagen de referencia			
Dimensiones (mm x mm x mm)	20 x 16 x 3	25 x 14 x 3	20 x 16 x 3
Voltaje Operacional (V)	3.3-5	3.3-5	3.3-5
Rango Acelerómetro	2g/4g/8g/16g	2g/4g/8g/16g	2g/4g/8g/16g
Rango Giroscopio (Grad/seg)	250/500/1000/2000	250/500/1000/2000	250/500/1000/2000
Precisión			
Precio (soles)	18	75	60
Referencia	Naylamp	Naylamp	Naylamp

Selección de Módulo Bluetooth

El módulo Bluetooth permitirá controlar el sistema de manera remota y sin cables. Se elegirá el Módulo Bluetooth HC06, debido a su bajo precio. No es necesario elegir el HM-10 debido a que la distancia a la que estará el control será corta, además que el HC06 tiene un costo menor al HC05. Esto se evidencia en la
Tabla 3. 13 a continuación.

Tabla 3. 13 Selección de Módulo Blue tooth

Modelo	Modulo Bluetooth 4.0 HM-10	Modulo Bluetooth HC06	Modulo Bluetooth HC05
Imagen de referencia			
Voltaje Operacional (V)	3.3 - 5	3.3 - 5	3.6 - 6
Alcance (m)	< 100	10	10
Precio (soles)	58	25	28
Referencia	Naylamp	Naylamp	Naylamp

Selección de Regulador

El regulador seleccionado será el mismo que se seleccionó en el módulo principal visto en la Tabla 3. 9.

Cabe mencionar que las conexiones electrónicas correspondientes al módulo principal, se pueden apreciar en la Figura 4. 16 Diagrama de Conexiones Módulo de Control.

Modelado Módulo de Control

El módulo de control puede acoplarse por medio del conector DB9 visto anteriormente. Por medio de la vista trasera y superior en la Figura 4. 8, es posible apreciar de mejor manera los componentes internos. A continuación, la Figura 4. 7 muestra una vista frontal del módulo de control.

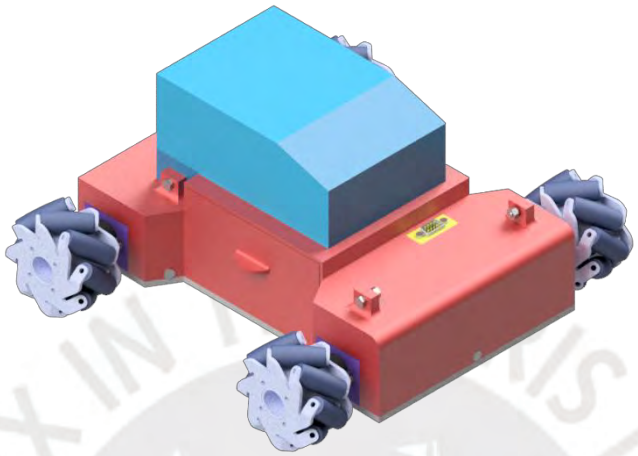


Figura 4. 7 Módulo de Control (vista delantera)

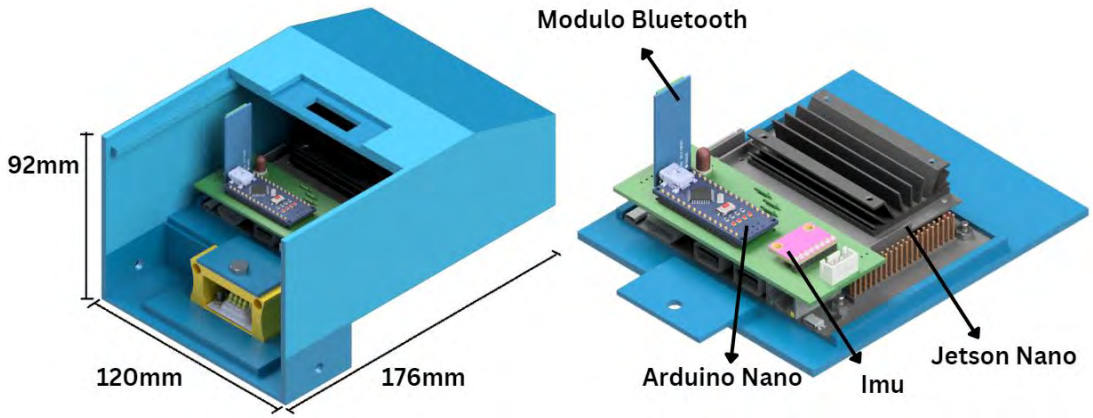


Figura 4. 8 Módulo de Control (vista trasera y superior)

Los componentes incluyen un IMU, para poder realzar navegación autónoma, una Jetson Nano para el procesamiento de datos de todo el sistema (en a comunicación I2C este sería el dispositivo maestro), y finalmente un módulo bluetooth, el cual permitirá manejar el carrito de manera tele operada. Para mayor detalle en cuanto a las piezas utilizadas y sus dimensiones consultar del **Anexo 62** al **Anexo 65Anexo 61**. Siendo el **Anexo 65** el plano de ensamble del módulo de control.

4.2.4 Módulo de Sensores para Obstáculos

Selección de Sensor Ultrasonido

Debido a que la distancia máxima necesaria será reducida (al ser un robot pensado para un ambiente controlado de laboratorio), se elegirá el sensor HC-SR04. Este será más barato, y así será más adecuado para realizar un sistema de bajo costo. Los detalles de los diferentes sensores están detallados en la Tabla 3. 14 a continuación.

Tabla 3. 14 Selección de Sensor de Ultrasonido

Modelo	Sensor Ultrasonido HC – SR04	Sensor Contaco Magnético	Sensor de Voltaje
Imagen de referencia			
Dimensiones (mm x mm x mm)	40 x 20 x 15	40 x 20 x 15	40 x 25 x 18
Voltaje de Funcionamiento (V)	5	5	5
Rango de Detección (cm)	2 – 450	4 – 300	25 – 450
Precio (soles)	8	18	40
Referencia	Naylamp	Naylamp	Naylamp

Selección de detector de conexión
Para el detector de conexión, se utilizará la el Arduino nano como se mencionó en el módulo de control autónomo. Esto se detalla en la Tabla 3. 10.

Selección de regulador
El regulador seleccionado será el mismo que se seleccionó en el módulo principal. Esto se detalla en la Tabla 3. 9.

Cabe mencionar que las conexiones electrónicas correspondientes al módulo principal, se pueden apreciar en la Figura 4. 17 Diagrama de Conexiones Módulo de Sensores.

Modelado Módulo de Sensores
El módulo de sensores es un caso especial, ya que para incorporar el módulo se debe intercambiar la tapa removible de la parte frontal del robot. Este tiene un reemplazo mostrado en la **¡Error! No se encuentra el origen de la referencia.**. Se puede ver el uso de un arreglo d e 3 sensores de ultrasonido para poder realizar navegación autónoma con detección de obstáculos. Adicionalmente se puede apreciar 2 conectores DB9, no correspondiente al módulo de sensores y otro al módulo de actuadores. Finalmente, cabe resaltar el Arduino nano para la comunicación I2C, y el regulador de voltaje para regular la alimentación de voltaje del módulo. Este módulo tiene una tapa removible que facilita el acceso a los componentes apreciado en la parte derecha de la **¡Error! No se encuentra el origen de la referencia.**.

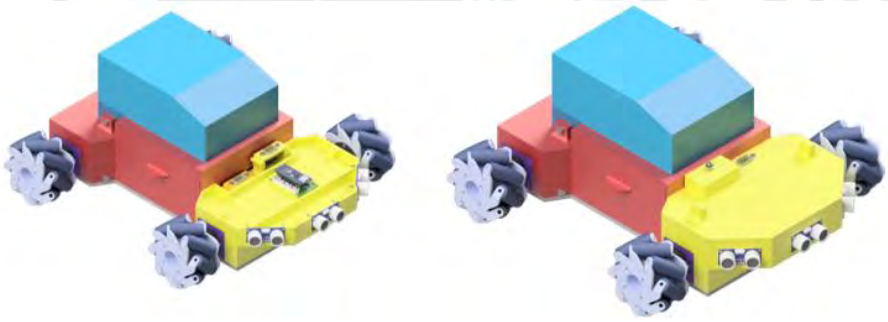


Figura 4. 9 Módulo de Sensores (sin tapa vs con tapa)

Para mayor detalle en cuanto a las piezas utilizadas y sus dimensiones consultar del **Anexo 66** al **Anexo 68***Anexo 61*. Siendo el **Anexo 68** el plano de ensamble del módulo de control.

4.2.5 Módulo de Actuadores (Mecanismo Biela Manivela)

Selección de Motores

Para el proyecto, se necesita un motor que sea adecuado para realizar acciones simples como presionar botones o marcar obstáculos con un sello mediante un mecanismo biela-manivela. Dado que estas tareas no requieren un alto nivel de torque, se descarta la opción de un motor con mayor potencia, ya que esto incrementaría innecesariamente el costo. Además, una caja reductora no es necesaria para este tipo de aplicación, ya que no se requiere una reducción significativa de la velocidad. Por lo tanto, se opta por el motor DC 6V/200 RPM sin caja reductora, que es suficiente para cumplir con los requisitos del proyecto y, al mismo tiempo, se ajusta a la necesidad de mantener los costos bajos. Esta elección se detalla en la Tabla 3. 15 a continuación.

Tabla 3. 15 Selección de Motores Para Mecanismo Biela Manivela

Modelo	MOTOR DC TT 6V/200RPM	MOTOR DC TT EN T CON CAJA REDUCTORA 6V/200RPM	MOTOR PAP STEPPER NEMA 17 - 1.7A 0.49N.M - SL42STH48-1684A
Imagen de referencia			
Peso (gr)	30	35	362
Voltaje Funcionamiento (V)	3 - 6	3 - 6	3 - 6
Torque Continuo (kg.cm)	0.8	0.8	-
Torque Detenido (kg.cm)	1.1	1.1	5
Precio (soles)	15	18	6
Referencia	Naylamp	Naylamp	Naylamp

Selección de driver de Motores

En este caso, se presentan los mismos tres drivers de motor mencionados anteriormente. El motor seleccionado tiene una corriente pico de 150 mA, lo que significa que cualquiera de los tres drivers podría ser capaz de controlarlo sin problemas. Sin embargo, se ha optado por el driver L298N, ya que es más comúnmente disponible en el mercado local en comparación con el L9110S. Aunque el L298N es ligeramente más caro, su mayor disponibilidad facilita su adquisición y asegura una mayor compatibilidad en el contexto del proyecto. Por esta razón, se selecciona el L298N como el driver de motor a utilizar como se ve en la Tabla 3. 16 a continuación

Tabla 3. 16 Selección de Drivers de Motores

Modelo	DRIVER PUENTE H L298N 2A	DRIVER PUENTE H VNH5019	DRIVER PUENTE H L9110S 0.8A
Imagen de referencia			
Dimensiones (mm x mm x mm)	43 x 43 x 27	38 x 28 x 25	35 x 20 x 15
Voltaje Operacional (V)	5 - 35	24	2.5 - 12
Motores Simultáneos	2	2	1
Corriente Máxima (A)	2	15	1.5
Precio (soles)	15	48	12
Referencia	Naylamp	Naylamp	Naylamp

Selección de Sensor Infrarrojo

Este sensor es necesario para poder determinar cuando el mecanismo biela manivela ha podido realizar una vuelta. En este caso, los 3 tipos de sensor infrarrojo permitirían percibir cuando se a completado una vuelta. Sin embargo, debido al precio se optará por la tercera opción. Ya que esta opción cumple con los requisitos, y es la más barata. En la Tabla 3. 17 a continuación se muestra la selección del sensor infrarrojo.

Tabla 3. 17 Selección de Sensor Infrarojo

Modelo	Sensor Infrarojo Sharp GP2Y0A21	Sensor Infrarojo Sharp GP2Y0A02	Sensor de Obsltaculos Infrarojo FC-51
Imagen de referencia			
Dimensiones (mm x mm x mm)	29.5 x 13 x 13.5	29.5 x 13 x 21.6	40 x 15 x -
Voltaje de Funcionamiento (V)	4.5 – 5.5	4.5 – 5.5	4.5 – 5.5
Rango de Detección (cm)	10 - 80	20 - 150	0.2 - 3
Precio (soles)	35	55	4
Referencia	Naylamp	Naylamp	Naylamp

Selección de detector de conexión

Para el detector de conexión, se utilizará la el arduino nano como se menciono en el módulo de control autónomo. Esto se detalla en la Tabla 3. 10.

Cabe mencionar que las conexiones electrónicas correspondientes al módulo principal, se pueden apreciar en la Figura 4. 18 Diagrama de Conexiones de Módulo Actuadores.

Modelado Mecanismo Biela Manivela

Para que los alumnos puedan experimentar con un actuador lineal se ha desarrollado un mecanismo biela manivela. Este puede apreciarse en la Figura 4. 10. Está conformado por un motor reductor pequeño, una rueda de 5cm de diámetro, y 2 extremidades. La que está unida a la rueda mide 5.2 cm de largo, mientras que la ultima extremidad mide 5.4 cm de punta a punta.

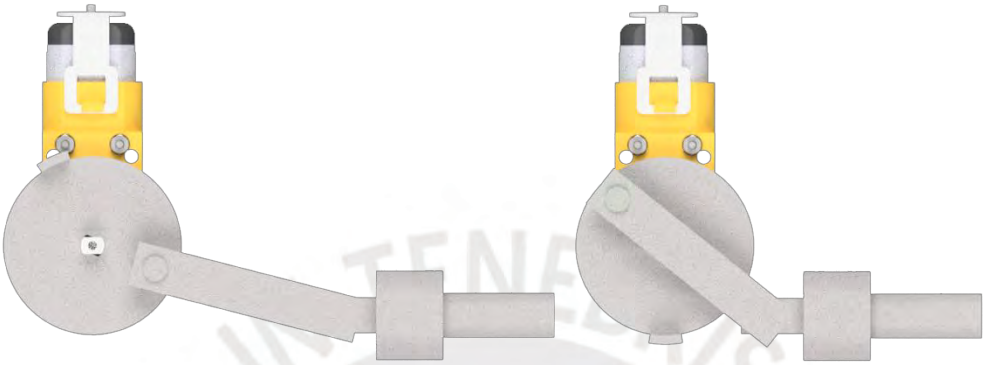


Figura 4. 10 Mecanismo Biela Manivela

Cuando el mecanismo está completamente extraído, sobresale del robot por 5.4 cm aproximadamente. Al estar retraído, tiene un alcance de 0.2 cm. Esto se aprecia de mejor manera en la Figura 4. 11 a continuación.

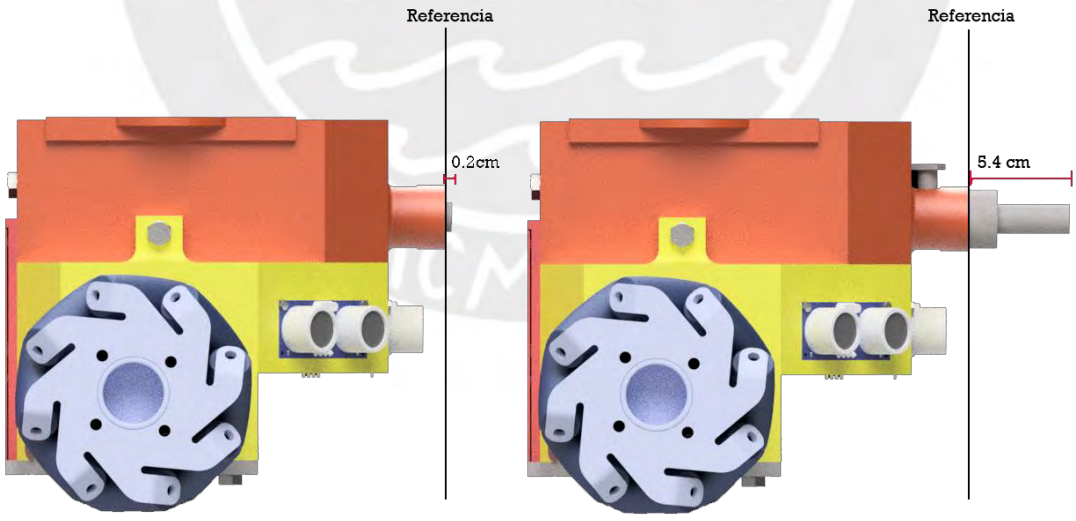


Figura 4. 11 Alcance de Mecanismo Biela Manivela

Modelado Modulo Actuadores

Finalmente, mostramos el módulo de Actuadores completo. Este incluye el mecanismo Biela Manivela, así como un Arduino nano que permite controlar el módulo de actuadores desde el módulo de control. De la mano del Arduino, se tiene los correspondientes conectores para las conexiones necesarias, así como un driver de motores que permitirá controlar al motor que genera el movimiento del mecanismo. Finalmente, se tiene un sensor infrarrojo que podrá determinar cuando el mecanismo ha completado una vuelta por medio de una sobresaliente en la rueda que es detectada por el sensor. Esto se aprecia en la Figura 4. 12. De igual manera cabe resaltar que el módulo de actuadores puede ser utilizado sin el módulo de sensores como se muestra en la Figura 4. 13. Para mayor detalle en cuanto a las piezas utilizadas y sus dimensiones consultar del **Anexo 69** al **Anexo 74***Anexo 61*. Siendo el **Anexo 74** el plano de ensamble del módulo de control.

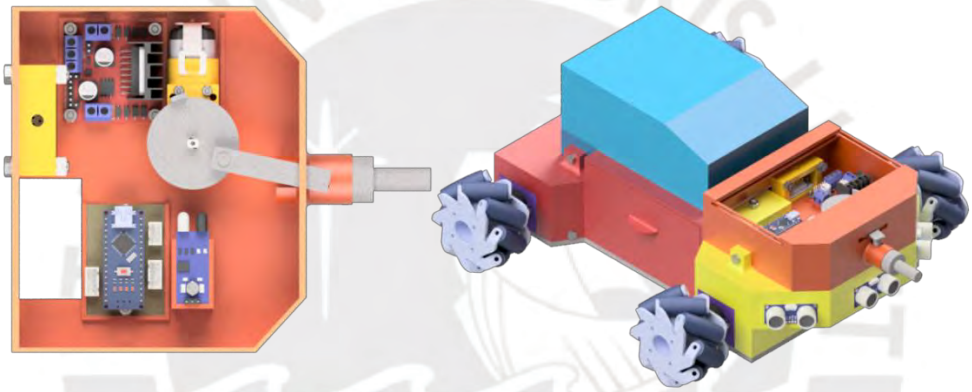


Figura 4. 12 Módulo de Actuadores

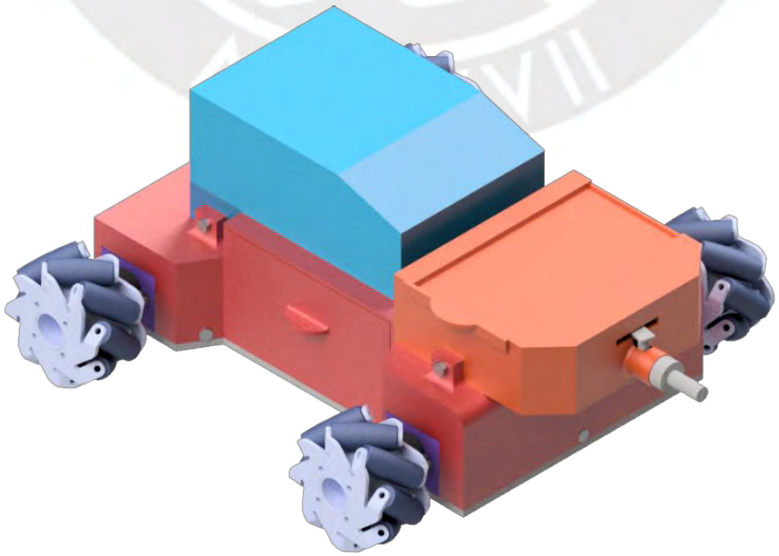


Figura 4. 13 Módulo de Actuadores (sin módulo de sensores)

4.3 Diseño Electrónico

La sección a continuación comprende el diseño electrónico del sistema.

4.3.1 Diagrama de Bloques

En la Figura 4. 14 a continuación se presenta el diagrama de bloques del sistema. Este permite comprender las conexiones correspondientes de manera más sencilla. Por un lado tenemos el módulo Móvil, el cual incorpora un voltaje de entrada que es distribuido hacia los 4 motores y 4 encoders del el sistema. De igual manera, por medio del regulador se alimenta a 2 Arduino uno, los cuales permitirán tener la cantidad suficiente de pines digitales para realizar el control omnidireccional. Cada Arduino realiza el control de 1 driver, el cual tiene capacidad de controlar a 2 motores.

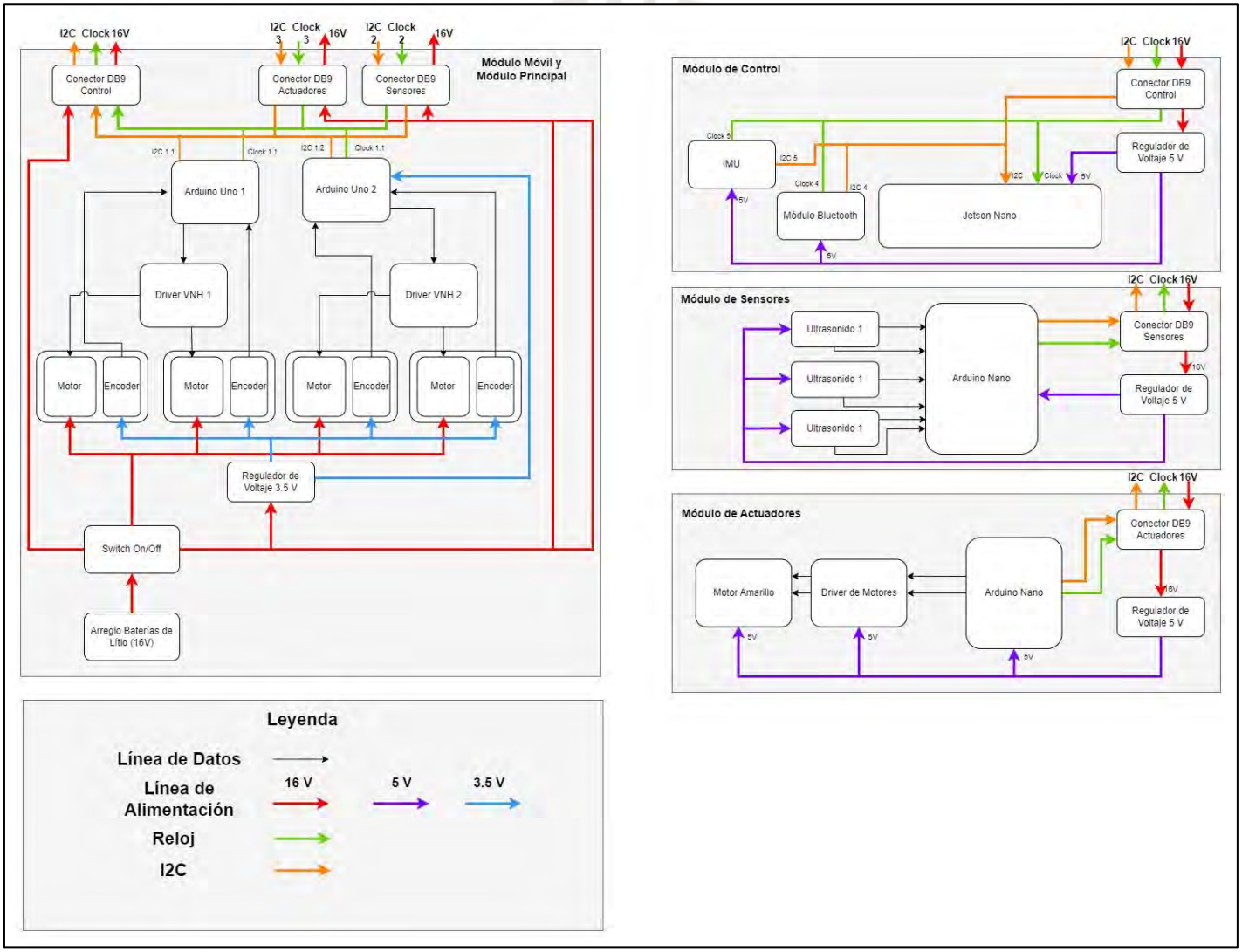


Figura 4. 14 Diagrama de Bloques general (fuente: Elaboración Propia)

4.3.2 Diagrama de Conexiones

Conexiones de Módulo Principal

En la Figura 4. 15 se puede apreciar de manera detallada las conexiones hechas en el módulo principal. Todas las señales I2C irán dirigidas hacia el dispositivo maestro en el módulo de Control.

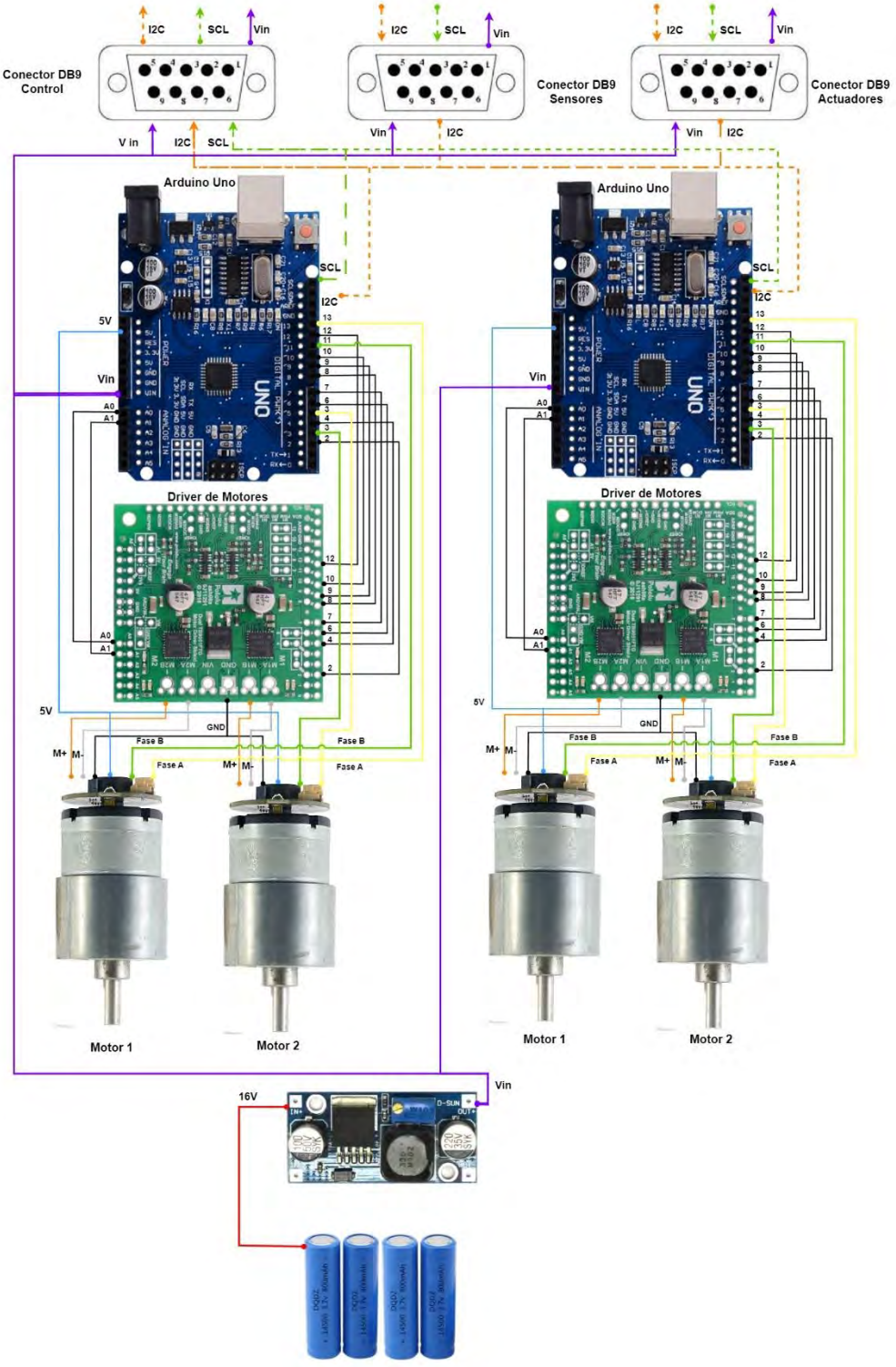


Figura 4. 15 Diagrama de Conexiones Principal

Conexiones de Módulo de Control

De manera similar, en la Figura 4. 16 se puede apreciar de manera detallada las conexiones hechas en el módulo de Control. Todas las señales I2C son dirigidas de los 3 módulos anteriores hacia este módulo. Las conexiones con la siguientes.

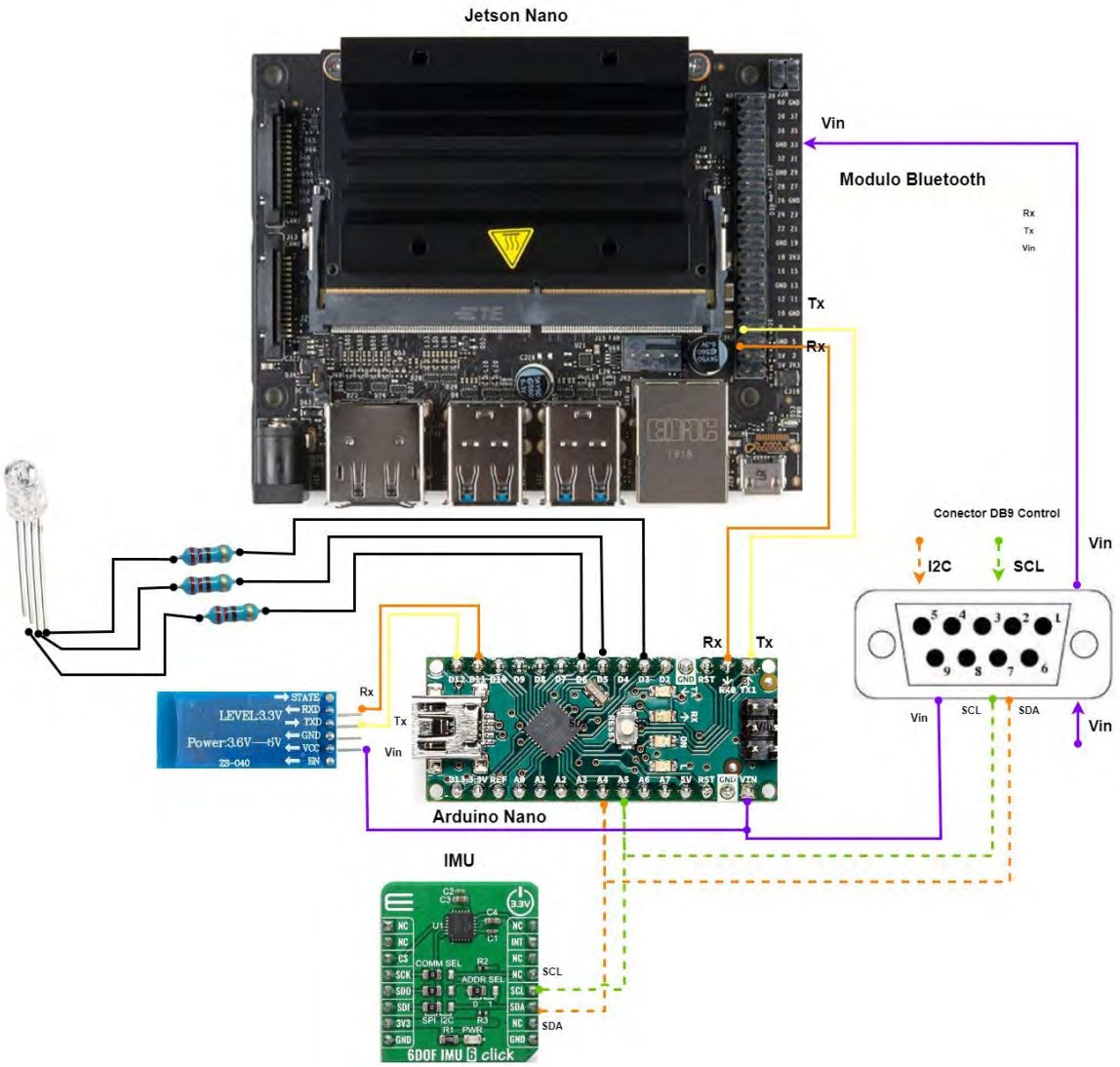


Figura 4. 16 Diagrama de Conexiones Módulo de Control

Conexiones de Módulo de Sensores Para Obstáculos

Al igual que en los demás módulos, tiene como entrada a través del conector DB9 3 conexiones importantes. Siendo estas las 2 entradas para la comunicación I2C, y finalmente el voltaje regulado por el Relé en el módulo principal. A continuación, la Figura 4. 17 presenta el diagrama de conexiones para el módulo de sensores.

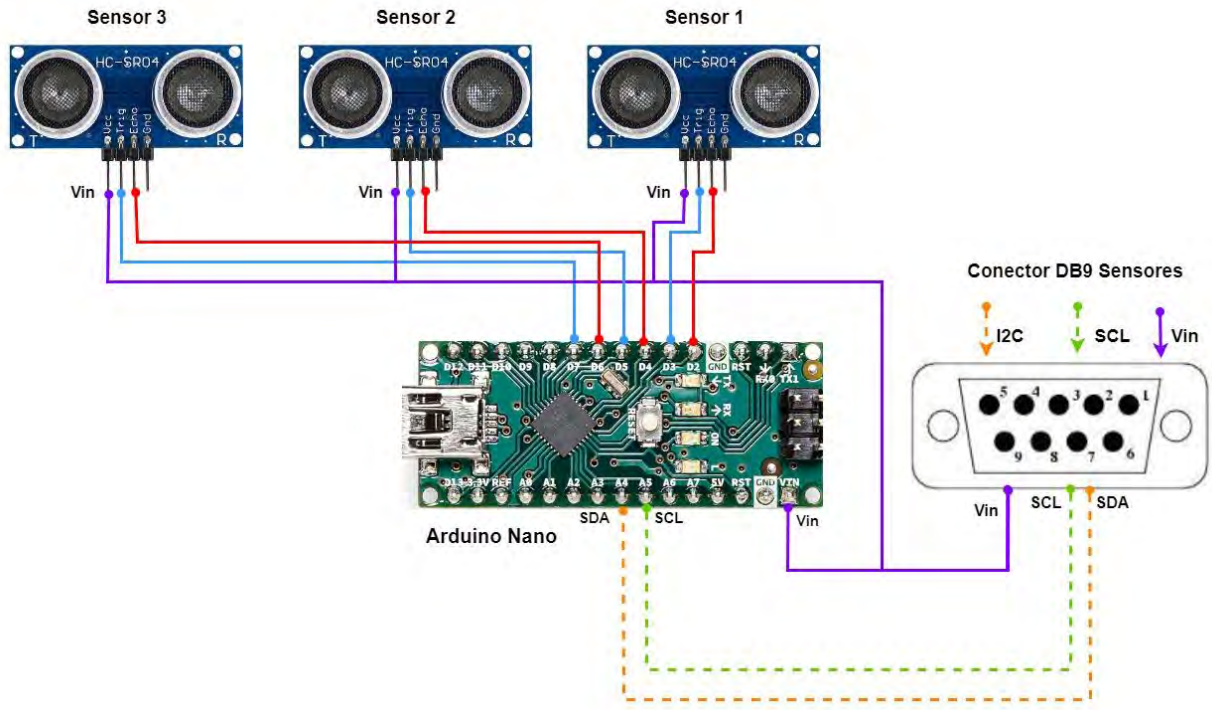


Figura 4. 17 Diagrama de Conexiones Módulo de Sensores

Conexiones de Módulo de Actuadores de movimiento Lineal

Finalizando esta sección, se presenta el diagrama de conexiones del módulo de actuadores. Este es muy simple a comparación de las demás conexiones necesarias para los otros módulos. El diagrama es presentado en la Figura 4. 18 a continuación.

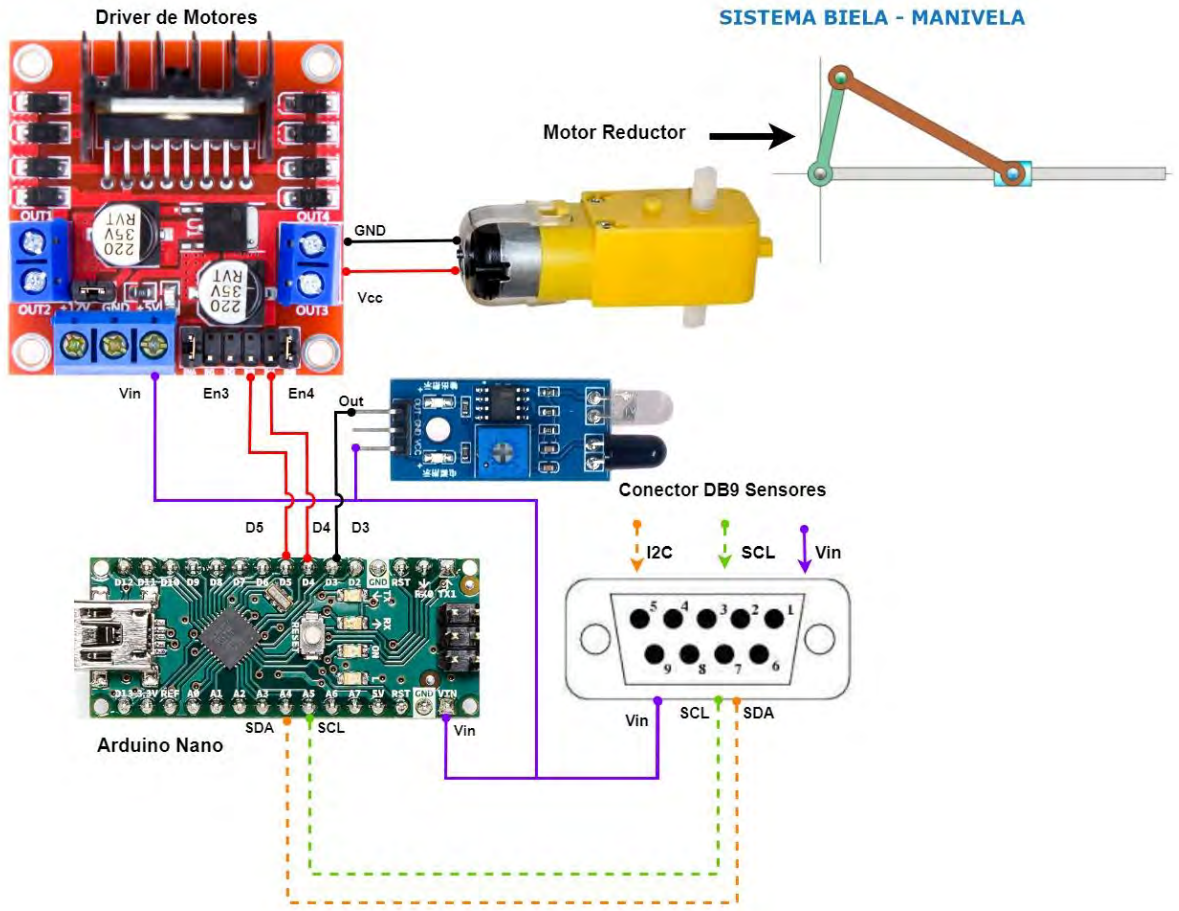


Figura 4. 18 Diagrama de Conexiones de Módulo Actuadores

4.3.3 Diseño de Circuitos Eléctricos

Selección de Batería

Para conocer la batería necesaria para el sistema, lo primero a considerar es el consumo energético por hora. En la tabla a continuación se puede visualizar de manera resumida el consumo energético aproximado de cada componente electrónico. Esto se puede apreciar en la Tabla 4. 5 a continuación.

Tabla 4. 5 Calculo de Capacidad Energética

Item	Componente	Módulo	Cantidad	Voltaje	Corriente (mA)	Potencia Total (W)	Tiempo (h)	Capacidad necesaria (Ah)
1	Arduino Uno	Omnidireccional	2	7.2	65	1.57	0.5	0.065
2	VNH5019 Driver	Omnidireccional	2	16	8	0.256	0.5	0.008
3	Relé de 12 V	Omnidireccional	1	16	37.5	0.6	0.5	0.01875
4	Diodo 1N4007	Omnidireccional	1	16	0	0	0.5	0
5	Motor DC 37D	Omnidireccional	4	16	800	64	0.5	1.6
6	Encoder	Omnidireccional	4	5	30	1	0.5	0.06
7	Arduino Nano	Control	1	7.2	30	0.235	0.5	0.015
8	Jetson Nano	Control	1	16	800	16	0.5	0.4
9	Led RGB	Control	1	3.3	60	0.198	0.5	0.03
10	IMU	Control	1	3.3	10	0.033	0.5	0.005
11	Modulo Bluetooth	Control	1	3.3	40	0.132	0.5	0.02
12	Ultrasonido	Sensores	3	5	15	0.225	0.5	0.0225
13	Arduino Nano	Sensores	1	7.2	30	0.235	0.5	0.015
14	Motor Amarillo	Actuadores	1	5	140	2.5	0.5	0.07
15	Driver de Motores	Actuadores	1	5	36	0.18	0.5	0.018
16	Arduino Nano	Actuadores	1	7.2	30	0.235	0.5	0.015
								2.36225

A partir de ello, sabemos que las baterías a utilizar deberán tener una capacidad energética de al menos 2.36 Ah para poder permitir que el sistema funcione durante media hora considerando todos los componentes funcionando en paralelo.

4.3.4 Diseño esquemático

Para poder realizar una tarjeta que se acomode a las necesidades del sistema, lo primero por hacer es el diseño esquemático. Este diseño será dividido por módulos. Así mismo, dentro de cada módulo se contará con una división del sistema por etapas.

Esquemático Módulo Principal

En primer se muestra la etapa de potencia del sistema. Se hace uso de 3 borneras dobles. Una corresponderá al voltaje conectando las baterías directamente. El segundo corresponde al voltaje entregado por el switch, y finalmente el tercero corresponde al voltaje proporcionado por el Switch. Todos los voltajes son de 16V, aunque el Vin y Vswitch alternan entre 0 y 16 V según corresponda. Esto se aprecia en la Figura 4. 19 a continuación.

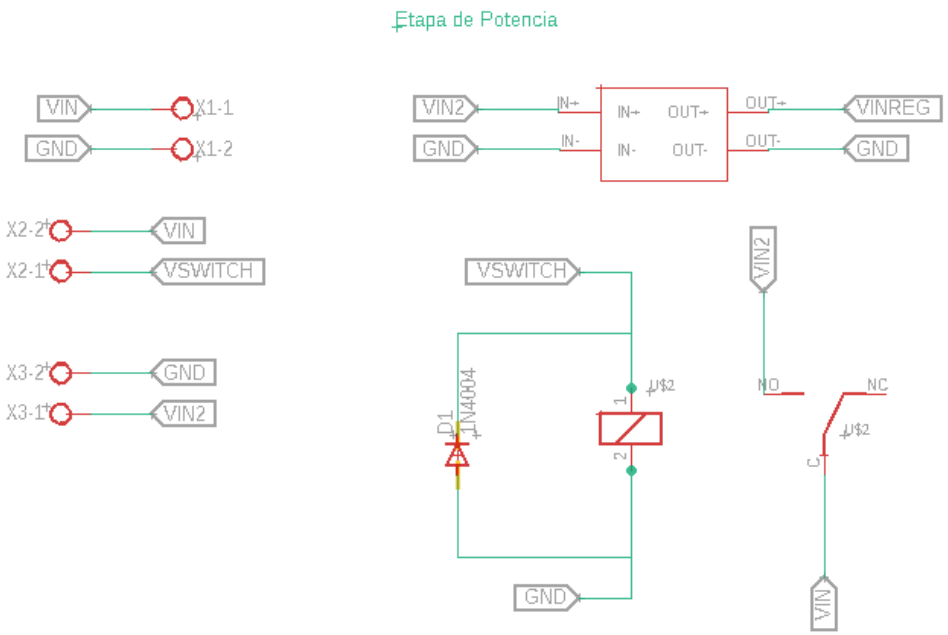


Figura 4. 19 Etapa de Potencia (módulo principal)

A la derecha de las borneras, se encuentra el regulador de voltaje, así como el circuito de protección de componentes eléctricos. El voltaje Vin2 se mantendrá en 0 mientras el switch esté apagado. Al encenderse, este será igual al voltaje Vin. Mediante un relé y un diodo zener, se protegen los componentes, en caso las baterías fueran conectadas en el sentido equivocado.

En la Figura 4. 20 se puede apreciar la etapa de los encoders, y las borneras de la salida I2C. Cada encoder contiene 4 conexiones, 1 a tierra, 2 conexiones para las fases de velocidad y una de alimentación de 5 V. Por otro lado, las borneras de I2C tienen 3 conexiones. Una de datos, otra de señal de reloj y una de tierra.

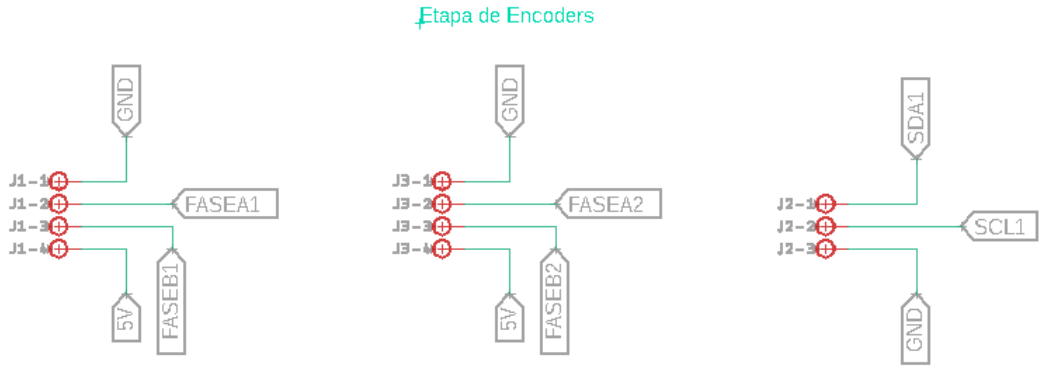


Figura 4. 20 Diagrama Esquemático de Encoders

La Figura 4. 21 muestra las conexiones del Arduino. Debido a que el driver fue diseñado para el Arduino este puede acoplarse como un shield. Por eso no se toma en consideración sus conexiones en el esquemático. Por otro lado, se realizan las conexiones de datos y reloj para I2C, y las conexiones de las fases de los enconders. Así mismo, se alimenta al Arduino con un voltaje regulado proveniente del regulador de voltaje. Este circuito controla a 2 motores, por lo que se deberá repetir en su totalidad para los dos motores restantes.

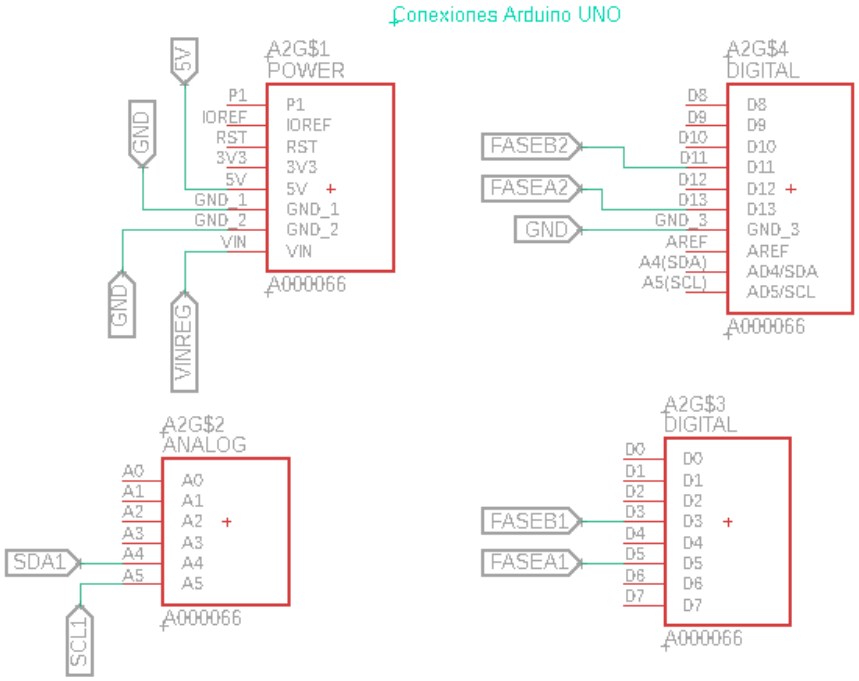


Figura 4. 21 Esquemático de Arduino

Finalmente, tenemos 3 conectores DB9. Estos conectores deberán conectar 4 tipos de señales. Las primeras 2 siendo 5V y GND, seguido de las dos señales para la comunicación I2C. La primera bornera es para el módulo de sensores, la segunda para el módulo de actuadores, y la última para el módulo de control. Esto se aprecia en la Figura 4. 22 a continuación.

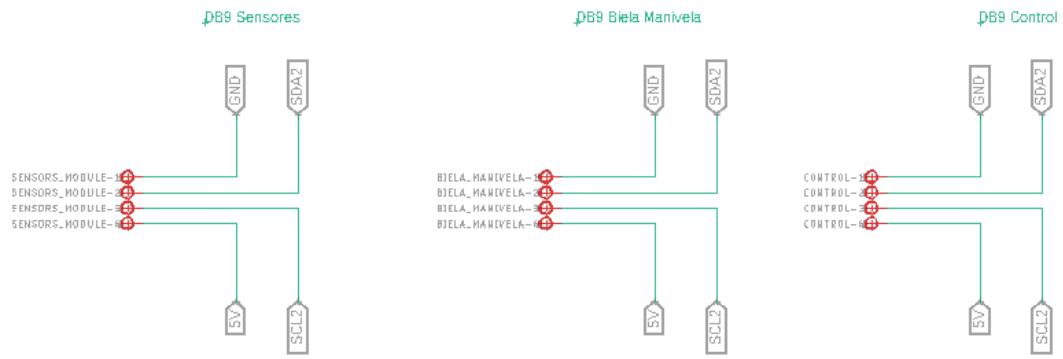


Figura 4. 22 Diagrama Esquemático de Conectores DB9

Esquemático Módulo de Control

En esta primera etapa de control, se muestran 2 controladores. Por un lado la Jetson Nano, encargada del control de todo el sistema integrad, y por otro lado el Arduino nano encargado de conectar a la Jetson nano las señales de SCL y SDA para la comunicación I2C. Así mismo, el Arduino permite controlar 3 leds RGB. Estos indicarán al usuario que módulos están conectados. Los LEDs pueden apreciarse en la parte inferior de la imagen. La Figura 4. 23 a continuación es la primera etapa del diseño esquemático del módulo de control.

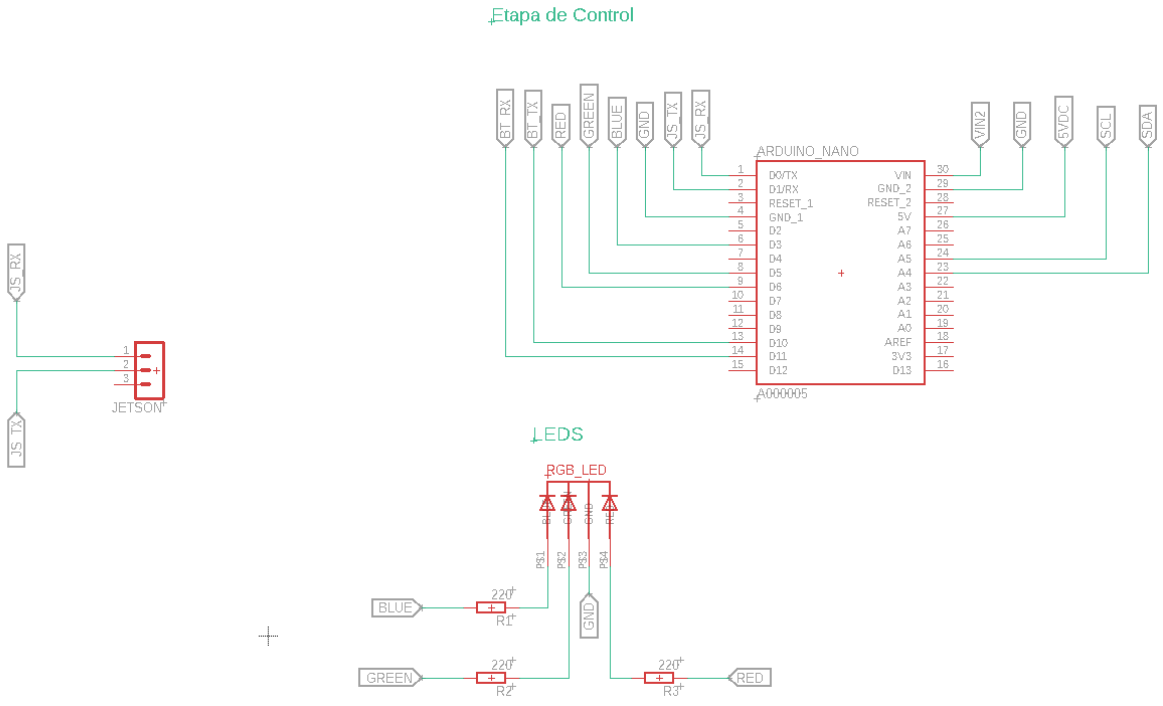


Figura 4. 23 Esquemático Etapa de Control (módulo de Control)

Finalmente, se presentan los componentes adicionales que son necesarios para el funcionamiento del sistema. Por un lado, se tiene al IMU, este permitirá al robot ir de un punto

A a un punto B de manera autónoma. Por otro lado, el módulo Bluetooth permitirá al sistema el poder recibir instrucciones de algún usuario por medio de un mando con conexión bluetooth. Por último, se aprecia en la parte izquierda las borneras que tienen la señal de SDA y SCL para la comunicación I2C, proveniente de los otros 3 módulos. Todo esto se aprecia en la Figura 4. 24 a continuación.

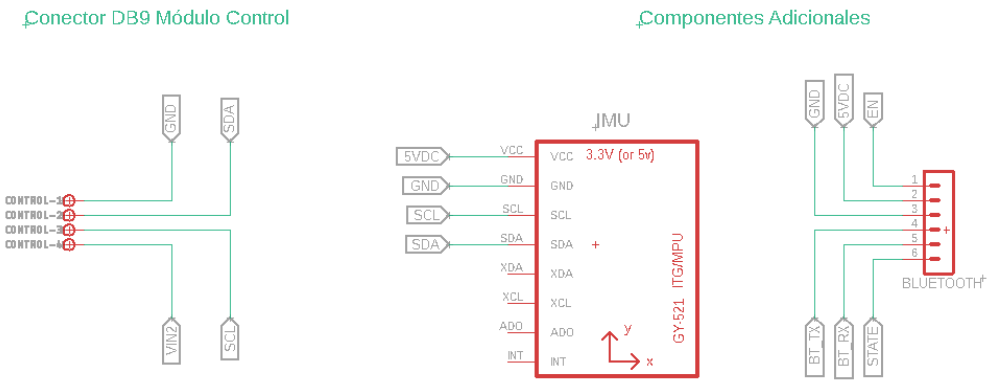


Figura 4. 24 Componentes Adicionales y Borneras de Entrada (módulo de control)

Esquemático Módulo Sensores Para Obstáculos

De manera similar, la Figura 4. 25 presenta el diseño esquemático del circuito para el módulo de sensores para obstáculos. Este diagrama está dividido en 3 secciones, comenzando por la sección del conector DB9. Este tiene 4 entradas, Vin y GND para el voltaje necesario para el sistema, y finalmente SDA y SCL para la comunicación I2C. De igual manera, está la sección de sensores. Cada sensor tiene 2 pines para alimentación y 2 pines para señales. Estos pines de señales deberán ser conectados al Arduino Nano a través de pines digitales. Finalmente vemos la sección del Arduino Nano con las conexiones del SCL y SDA, así como las conexiones para los 3 sensores.

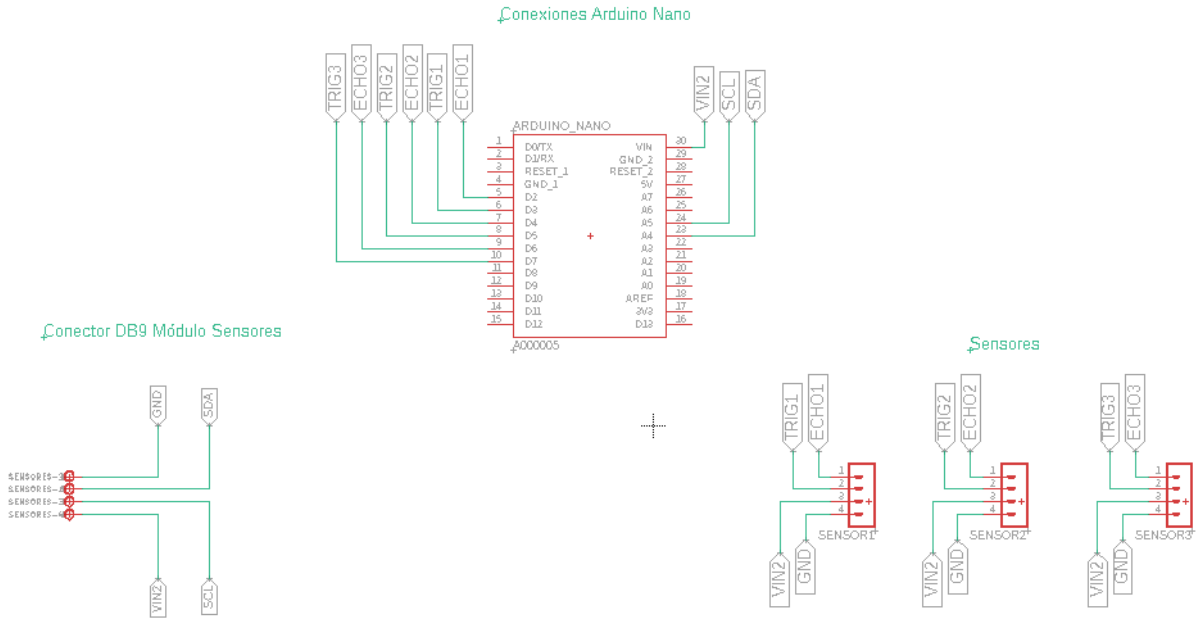


Figura 4. 25 Diagrama Esquemático del Módulo de Sensores

Esquemático Módulo Actuadores de movimiento Lineal

En el caso del módulo de actuadores, se necesitaba conectar al Arduino nano 2 dispositivos. Por un lado un driver de motores que permitiría controlar al motor que permite el funcionamiento del mecanismo biela manivela, y por otro lado un sensor infra rojo el cual detectaría cuando se complete una vuelta del eje de motor. El diseño esquemático puede apreciarse en la Figura 4. 26 a continuación.

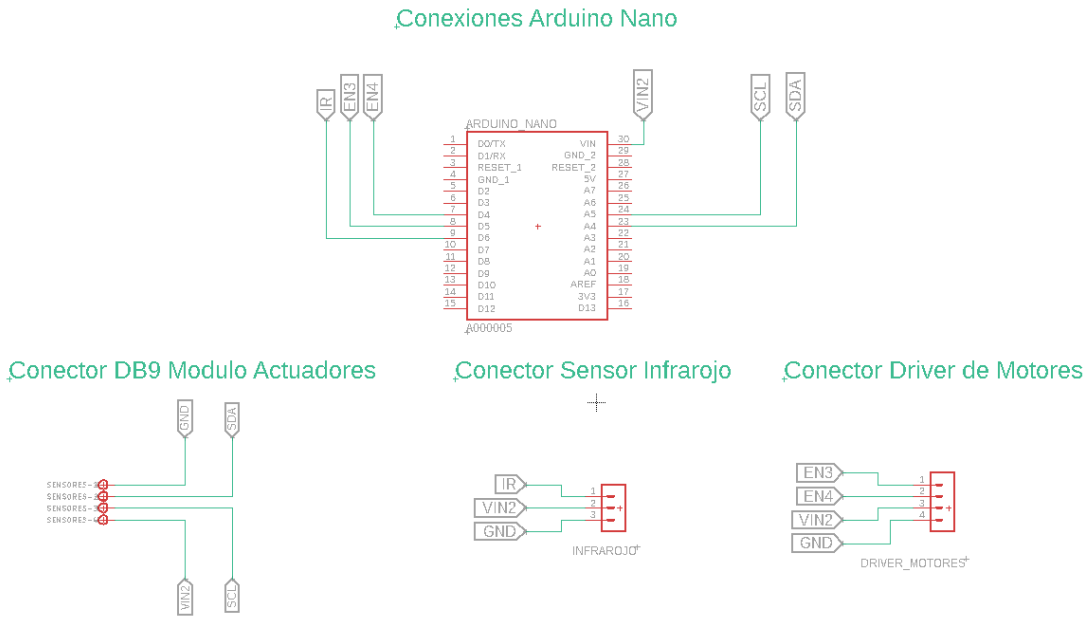


Figura 4. 26 Diagrama Esquemático del Módulo de Actuadores

4.3.5 Diseño de Tarjetas Impresas

Tarjeta de Módulo Principal

A continuación, se presenta la tarjeta del módulo principal. Esta tarjeta mide 51.4 x 48.3 mm, tamaño adecuado para encajar 2 de estas tarjetas en el espacio designado. Al ser pocas conexiones, se optó por realizar una tarjeta de una sola capa. A continuación, se presenta las conexiones de la tarjeta de los primeros 2 motores, Figura 4. 27, y los otros 2 motores en la Figura 4. 28.

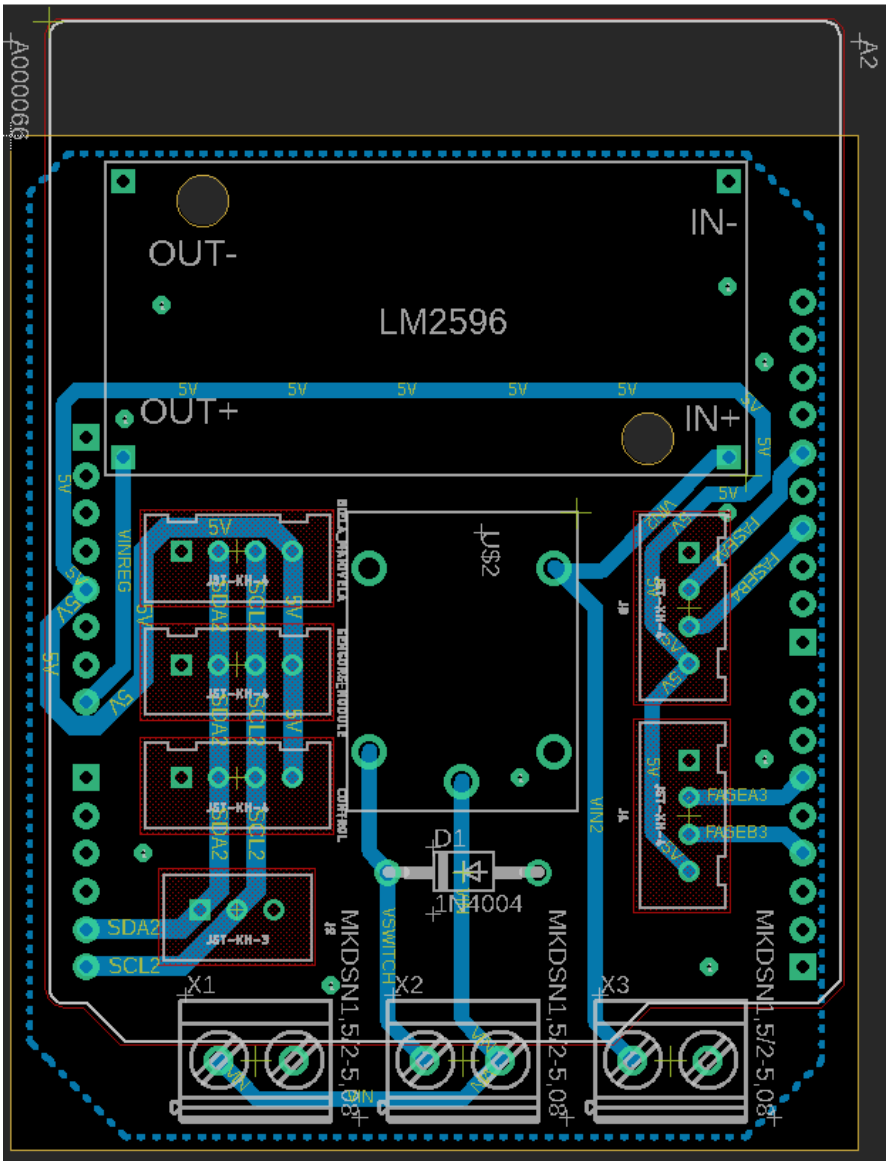


Figura 4. 27 Conexiones de Tarjeta M1 y M2 de módulo principal

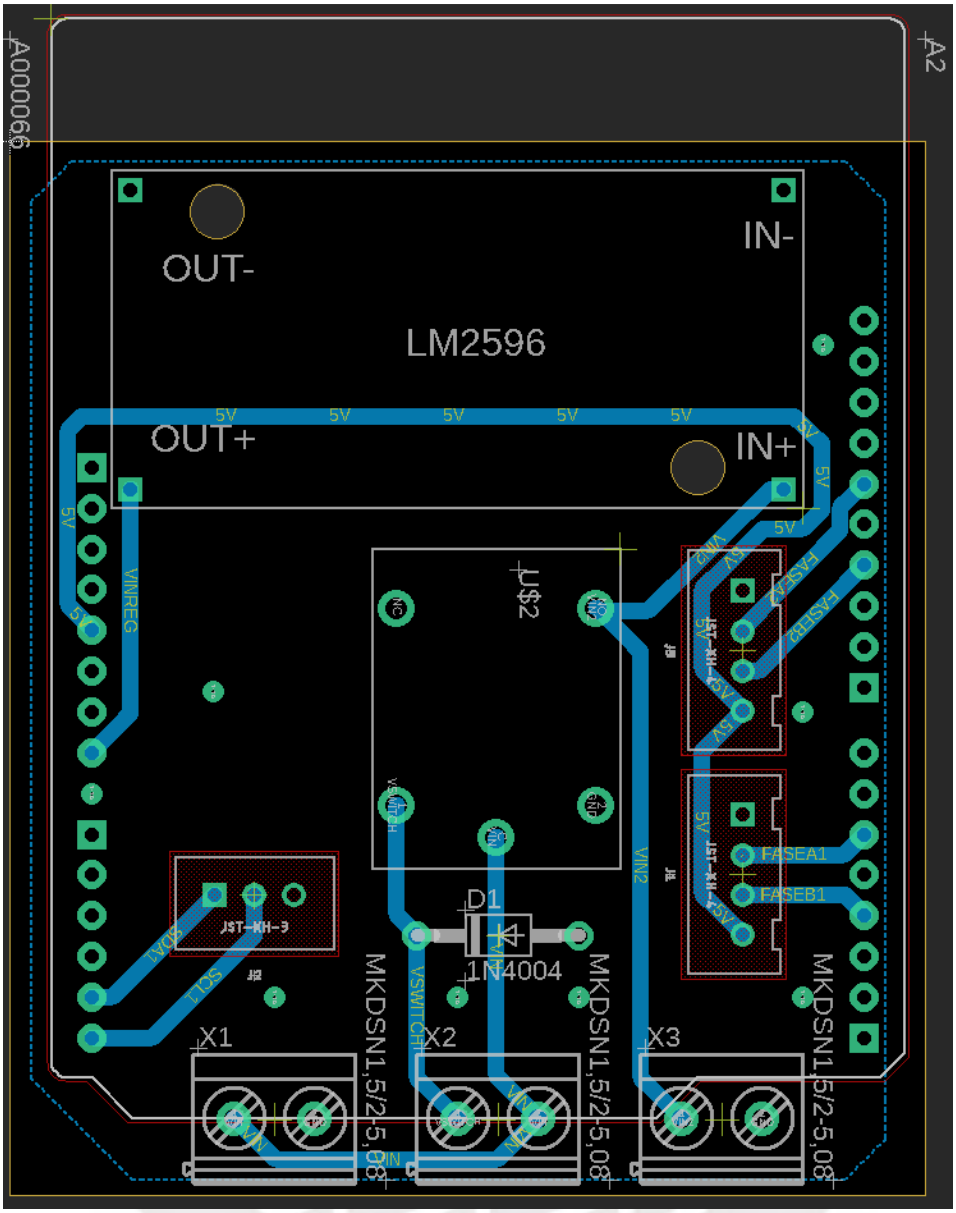


Figura 4. 28 Conexiones de Tarjeta M3 y M4 de módulo principal

Tarjeta de Módulo de Control

De manera similar, las conexiones de la tarjeta del módulo de control permitieron que se pueda utilizar una sola capa. En la tarjeta destaca una forma en ´L´, permitiendo así aprovechar el espacio al máximo. La tarjeta mide 36.8 x 95.2 mm. Todo esto se aprecia en la Figura 4. 29 a continuación.

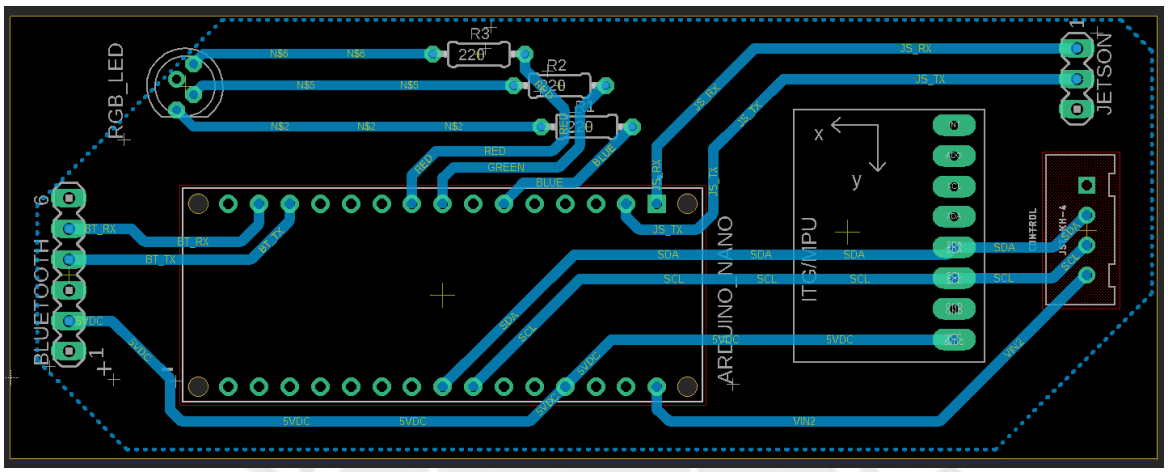


Figura 4. 29 Conexiones de Módulo de Control

Tarjeta de Sensores Para Obstáculos

Esta es una tarjeta bastante compacta, debido a la reducida cantidad de componentes requeridos. Tiene un ancho y alto de 32.7 x 48.2 mm respectivamente. Así como en el resto de las tarjetas, no fue necesaria la implementación de una tarjeta multicapa. A continuación, la Figura 4. 30 presenta las conexiones de la tarjeta del módulo de sensores para obstáculos

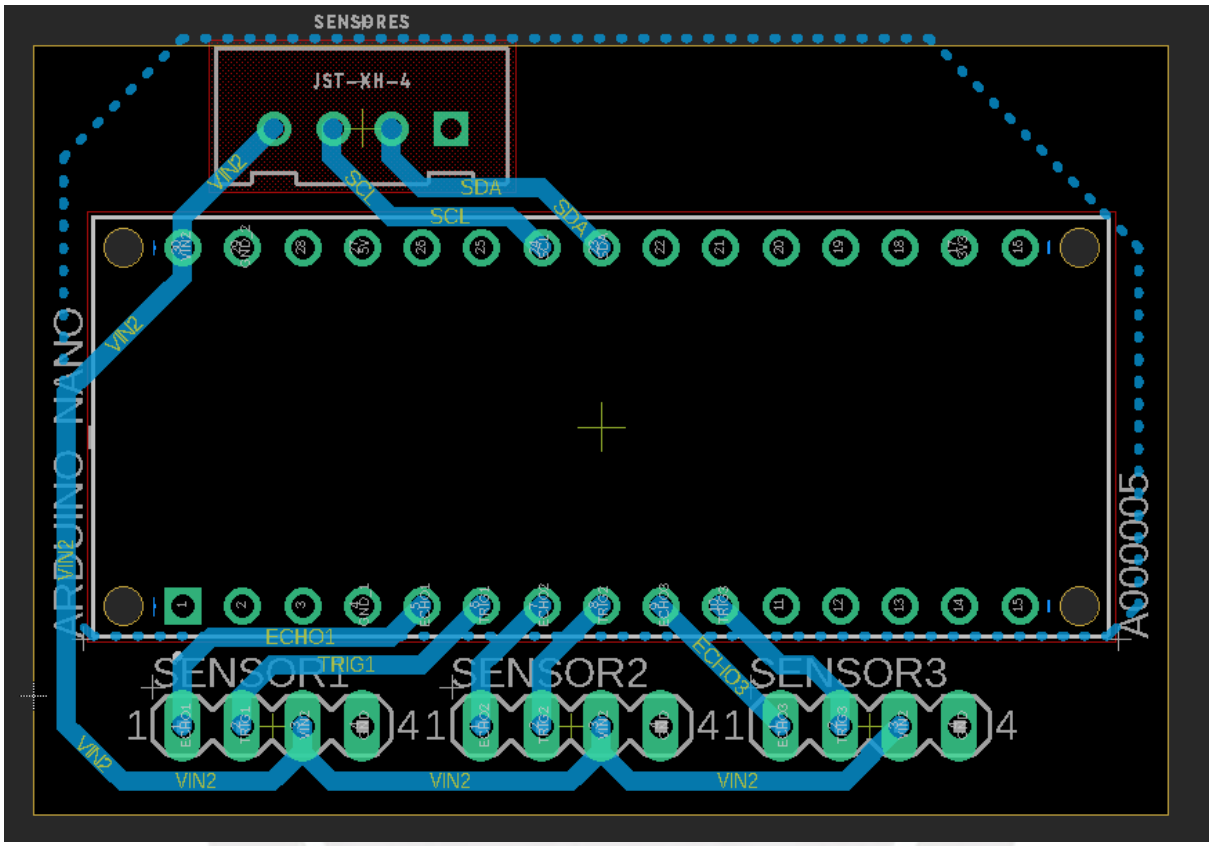


Figura 4. 30 Conexiones de Módulo de Sensores

Tarjeta de Modulo Actuadores

Finalmente, al igual que la tarjeta del módulo de sensores, es una tarjeta bastante compacta, debido a la reducida cantidad de componentes requeridos. Tiene un ancho y alto de 34 x 48.3 mm respectivamente. Así como en el resto de las tarjetas, no fue necesaria la implementación de una tarjeta multicapa. La Figura 4. 31 presenta las conexiones de la tarjeta del módulo de actuadores.

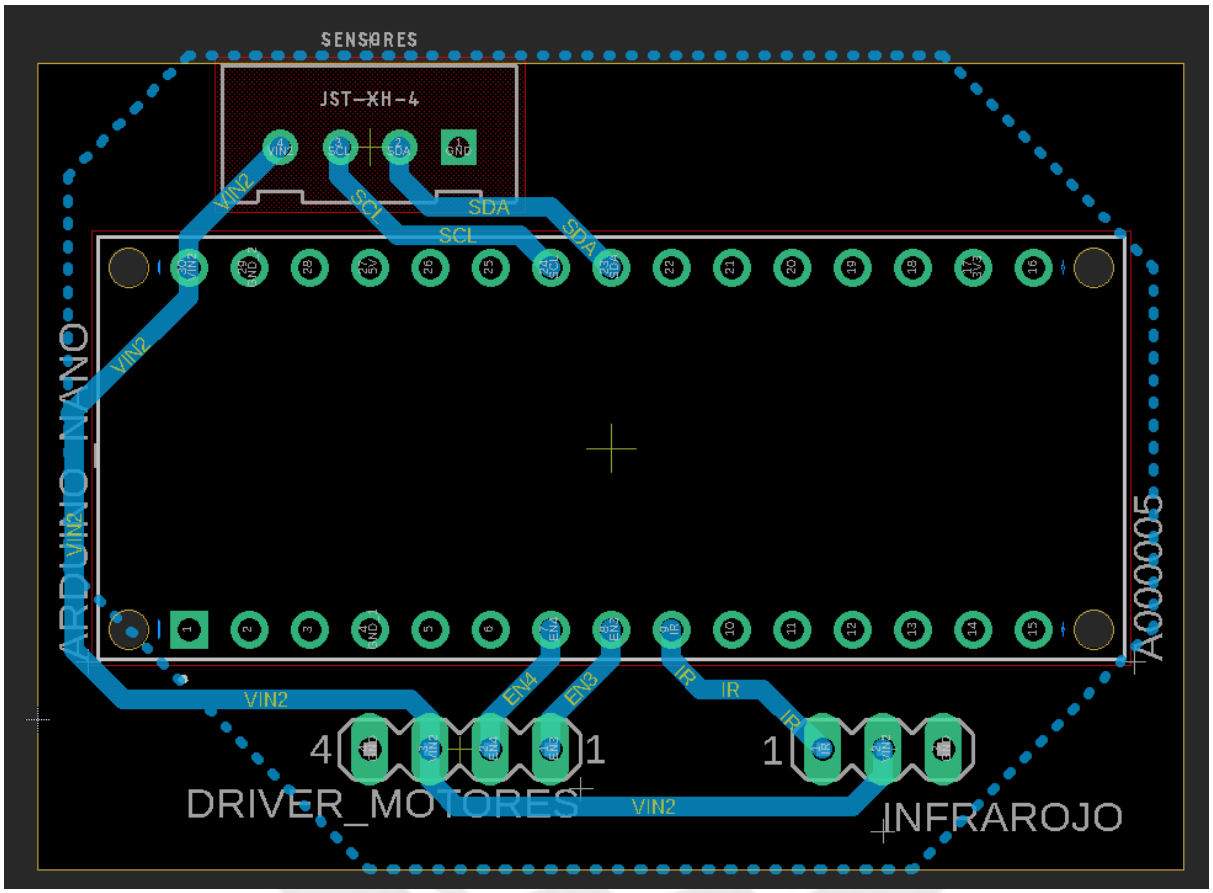


Figura 4. 31 Conexiones de Módulo de Sensores

4.4 Diseño de Software

La sección a continuación comprende el diseño de software del sistema.

4.4.1 Arquitectura de Control

Como se mencionó anteriormente, para poder controlar el sistema se hace uso de una interfaz gráfica. Esta interfaz permitirá controlar el sistema de 2 maneras. La primera es por medio de un control de PS4 el cual cuenta con una conexión bluetooth, y la segunda por medio de la misma interfaz la cual cuenta con botones que direccionan al sistema en el eje X Y del plano. El control cuenta con 2 joysticks para poder direccionar al sistema, así como 2 botones para incrementar y disminuir la velocidad, 1 para parar y 2 para rotar en ambos sentidos horarios. Esto se aprecia en la Figura 4. 32 a continuación.

Figura 4. 32 Control PS4 Etiquetado

Toda conexión entre la interfaz gráfica y el sistema se realizará de manera inalámbrica. Como se mencionó anteriormente, el módulo de control cuenta con un módulo bluetooth que permite mandar señales de manera inalámbrica. Por medio de este módulo es posible enviar las instrucciones para controlar al sistema. Así como las señales de los estados de los módulos para determinar la conexión de los mismos a través de la interfaz gráfica, y por último de querer hacerlo conectar una cámara a la Jetson nano y mandar señales de imagen. Esto se puede apreciar en la Figura 4. 33 a continuación.

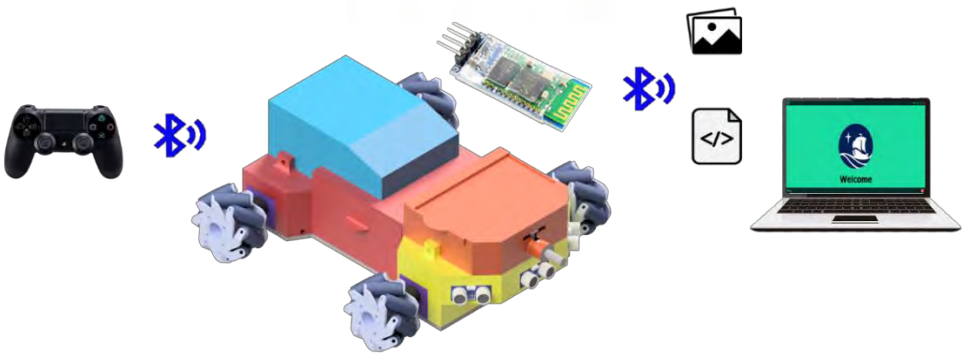


Figura 4. 33 Diagrama de Arquitectura de Control

4.4.2 Estrategias de Control

Estrategia de Control Propuesta

Para poder desarrollar la cinemática detrás de la configuración omnidireccional, lo primero a realizar es definir un sistema de referencia para la velocidad lineal del robot, así como la velocidad angular general. Esto puede apreciarse en la Figura 4. 34 a continuación.

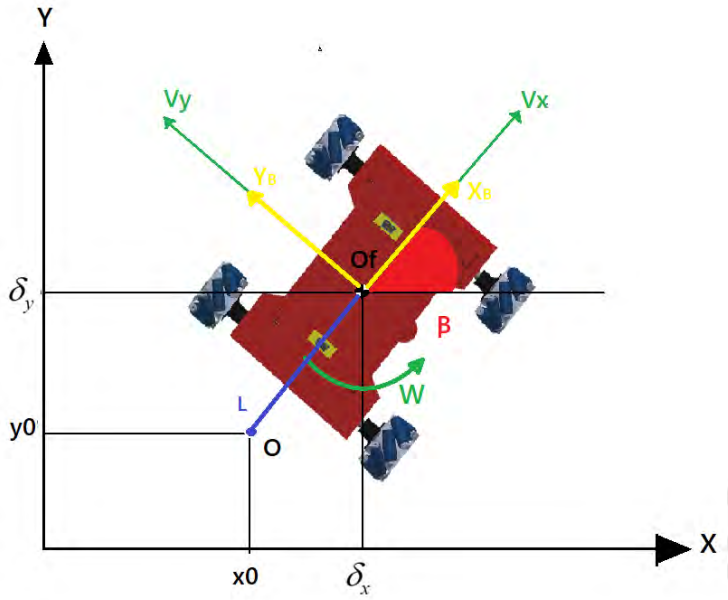


Figura 4. 34 Sistema de Referencia (Elaboración Propia)

A partir de la figura, definimos al punto (x, y) como el centroide del robot, β como el ángulo entre el robot y el eje X, V_x como la velocidad lineal en X_B, V_y la velocidad lineal en Y_B, L como la distancia desplazada por el centroide del sistema (del punto O al O_f), y finalmente w como la velocidad angular respecto a los ejes $\{X_B, Y_B, Z_B\}$. El nuevo vector de velocidades para el punto O_f es definido como $\dot{\delta} = [\dot{\delta_x} \quad \dot{\delta_y} \quad \dot{\delta_w}]$. De esta manera, obtenemos las siguientes relaciones (S. Álvarez, 2019).

$$\delta_x = x_0 + L\cos\beta \qquad (1)$$

$$\delta_y = y_0 + L\sin\beta \qquad (2)$$

Por medio de derivadas parciales, se obtiene el siguiente sistema de ecuaciones (S. Álvarez, 2019).

$$\begin{bmatrix} \dot{\delta_x} \\ \dot{\delta_y} \\ \dot{\delta_w} \end{bmatrix} = \begin{bmatrix} \cos\beta & -\sin\beta & -L\sin\beta \\ \sin\beta & \cos\beta & L\cos\beta \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} V_x \\ V_y \\ w \end{bmatrix} \qquad (3)$$

Simplificando el modelo matemático se obtiene la siguiente ecuación.

$$\dot{\delta}(t) = M(\beta)V(t) \qquad (4)$$

Para realizar la navegación autónoma, necesaria para cumplir con los requisitos, se plantea un control PID de manera que se puedan compensar los errores durante desplazamiento. Se define la trayectoria deseada como $\delta_d(t)$. En base a la ecuación 4, se propone el siguiente controlador PID (S. Álvarez, 2019).

$$v_c(t) = M^{-1}{}_{(\beta)}(\dot{\delta_d} + k\tilde{\delta}) \qquad (5)$$

Donde $\tilde{\delta}$ representa el error, definido por $\tilde{\delta} = (\delta_d - \delta)$, equivalencia en la cual 'δ' representa la posición actual del robot. De esta forma, para obtener el sistema de ecuaciones se utiliza el comando 'linsolve' de Matlab. (S. Álvarez, 2019)

$$v_c(t) = linsolve(M_{(\beta)}, [\dot{\delta_d} + PID(\tilde{\delta})])$$

El algoritmo PID está basado en la siguiente estructura.

$$u(t) = Kp * e(t) + Ki * \int_0^t e(t)\,dt + Kd * \frac{de(t)}{dt}$$

En el dominio de Laplace:

$$G(s) = K[1 + \frac{1}{s * Ti} + \frac{s * Td}{1 + \frac{Td}{N} * s}]$$

En este caso particular, tomando como base el modelo simulado en la publicación "Navegación Autónoma de un Robot Móvil omnidireccional en trayectorias predeterminadas con evitación de obstáculos" Figura 6.15, se utilizará solamente un control proporcional, con un valor de Kp igual a 0.5. Este diagrama corresponde al control PID aplicado a **cada rueda** de manera interna. El esquema de control sería el visto en la a Figura 4. 35 a continuación (S. Álvarez, 2019).

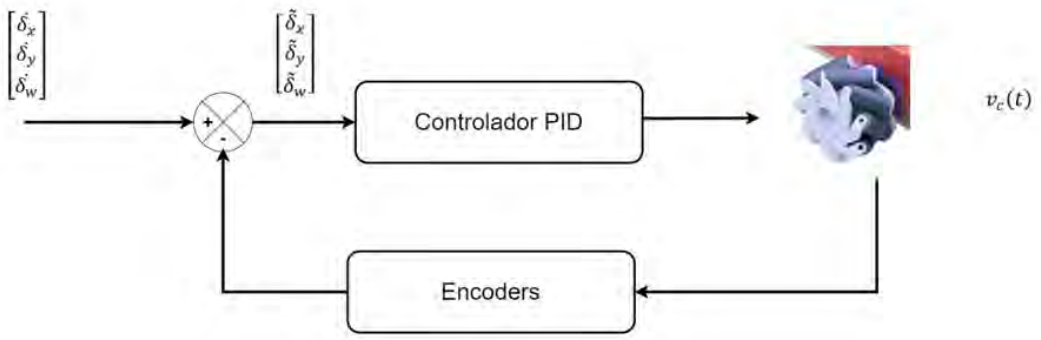


Figura 4. 35 Diagrama de Control (Ing. Santiago Álvarez)

Control de posicionamiento en base a Velocidad Angular de las Ruedas

Para poder expresar las velocidades angulares en función a las velocidades lineales V_x y V_y, se definen referencias para las velocidades lineales de cara rueda. Esto se puede comprender de manera más clara en la Figura 4. 36, donde se ve el sistema de referencia de velocidades, donde $L_x + L_y = L$. Estos cálculos están basados en los cálculos implementados en el paper 'Control de Trayectoria de un Robot Móvil Omnidireccional en un Entorno Controlado' (B. Castillo, 2023).

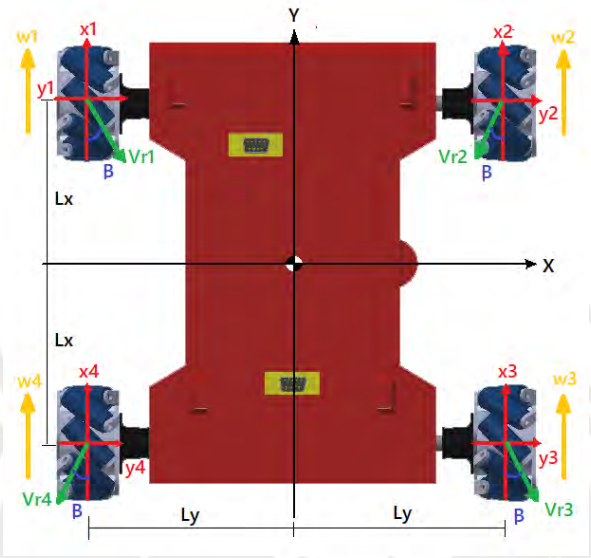


Figura 4. 36 Sistema de Referencia de Velocidades

A partir de ello, se define el siguiente sistema de ecuaciones matriciales, en el que la velocidad angular de cada motor, queda definida en función a L , las velocidades lineales V_x y V_y y finalmente la velocidad angular del sistema general W. Donde R corresponde al radio de la rueda ($R = 0.04m$). En el caso del sistema, L tiene un valor aproximado de 0.23m (B. Castillo, 2023).

$$\begin{bmatrix} w1 \\ w2 \\ w3 \\ w4 \end{bmatrix} = \frac{1}{R}\begin{bmatrix} 1 & -1 & -L \\ 1 & 1 & -L \\ 1 & -1 & L \\ 1 & 1 & L \end{bmatrix}\begin{bmatrix} V_x \\ V_y \\ w \end{bmatrix} \quad (6)$$

El sistema de ecuaciones matricial puede expresarse por medio de las siguientes ecuaciones.

$$w1 = \frac{1}{R}\left(V_x - V_y - (L * w)\right) \quad (7)$$

$$w2 = \frac{1}{R}\left(V_x + V_y - (L * w)\right) \quad (8)$$

$$w3 = \frac{1}{R}\left(V_x - V_y + (L * w)\right) \quad (9)$$

$$w4 = \frac{1}{R}\left(V_x + V_y + (L * w)\right) \quad (10)$$

Por medio de cinemática inversa, se puede dar con las ecuaciones de las velocidades V_x, V_y y w. Siendo las siguientes (B. Castillo, 2023).

$$\begin{bmatrix} V_x \\ V_y \\ w \end{bmatrix} = \frac{R}{4} \begin{bmatrix} -1 & 1 & -1 & 1 \\ 1 & 1 & 1 & 1 \\ \frac{1}{L} & \frac{-1}{L} & \frac{-1}{L} & \frac{1}{L} \end{bmatrix} \begin{bmatrix} w1 \\ w2 \\ w3 \\ w4 \end{bmatrix} \qquad (11)$$

Este sistema de ecuaciones matricial equivale a las siguientes ecuaciones.

$$V_x = \frac{R}{4}(-w1 + w2 - w3 + w4) \qquad (1)$$

$$V_y = \frac{R}{4}(w1 + w2 + w3 + w4) \qquad (2)$$

$$w = \frac{R}{4}\left(\frac{w1}{L} - \frac{w2}{L} - \frac{w3}{L} + \frac{w4}{L}\right) \qquad (3)$$

A partir de estos modelos cinemáticos, es posible determinar las velocidades angulares necesarias para poder controlar el robot.

4.4.3 Arquitectura de Software

Interfaz Gráfica

Para el desarrollo de la interfaz gráfica que ayudará a controlar el robot, se utilizó la IDE Visual Studio Code. Se programó todo en Python. Para poder realizar el desarrollo del front end se utilizaron las librerías de PyQt5. El código de la programación se encuentra en el **Anexo 1**. Lo primero que verá el usuario al iniciar la aplicación es el menú de inicio, este incluye el logo de la PUCP. Esto se muestra en la Figura 4. 37 a continuación.

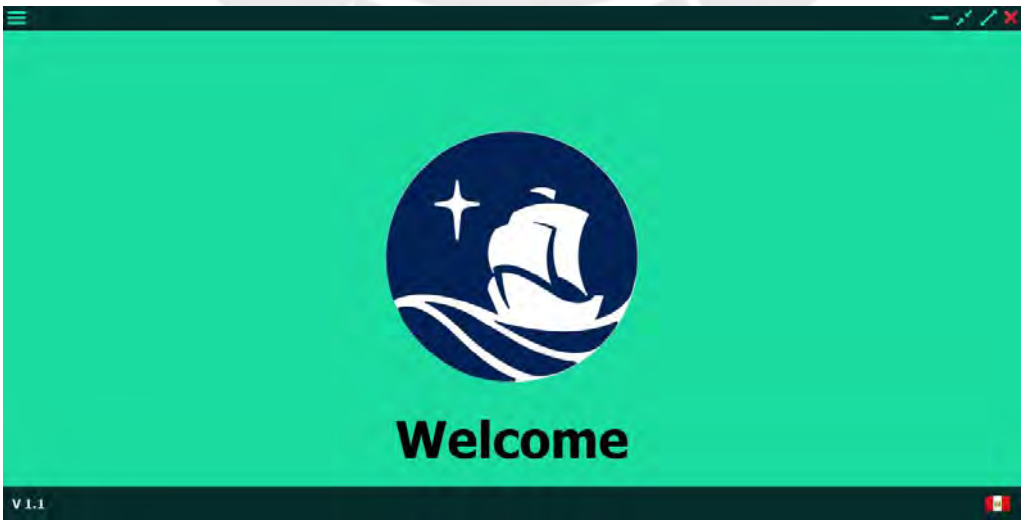


Figura 4. 37 Menú de Inicio de Interfaz de usuario

En la parte superior izquierda de la pantalla se aprecia un botón con 3 rayas. Al hacer click en ese botón, se despliega un menú deslizante. Este incluye 6 diferentes opciones a las cuales puede acceder el usuario. Estas funciones incluyen el menú de inicio, estado de los módulos, batería restante, acceso a cámara, controles del robot, y conexión bluetooth del control de PS4 para controlar alternativamente el robot. Todo esto se aprecia en la Figura 4. 38 a continuación.

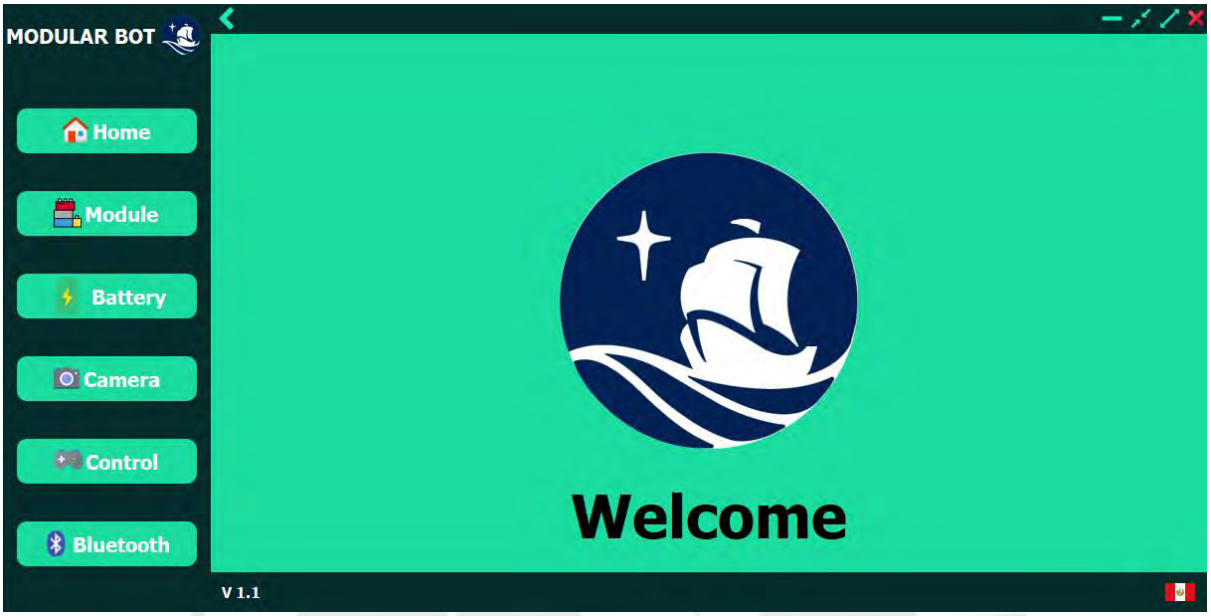


Figura 4. 38 Menú deslizante de Interfaz gráfica

La opción 'Module' permite al usuario visualizar los módulos del robot que se encuentran conectados. Estos se muestran de tono más oscuro al estas deshabilitados, y en color claro al estar deshabilitados. Esto se puede apreciar en la Figura 4. 39 a continuación.

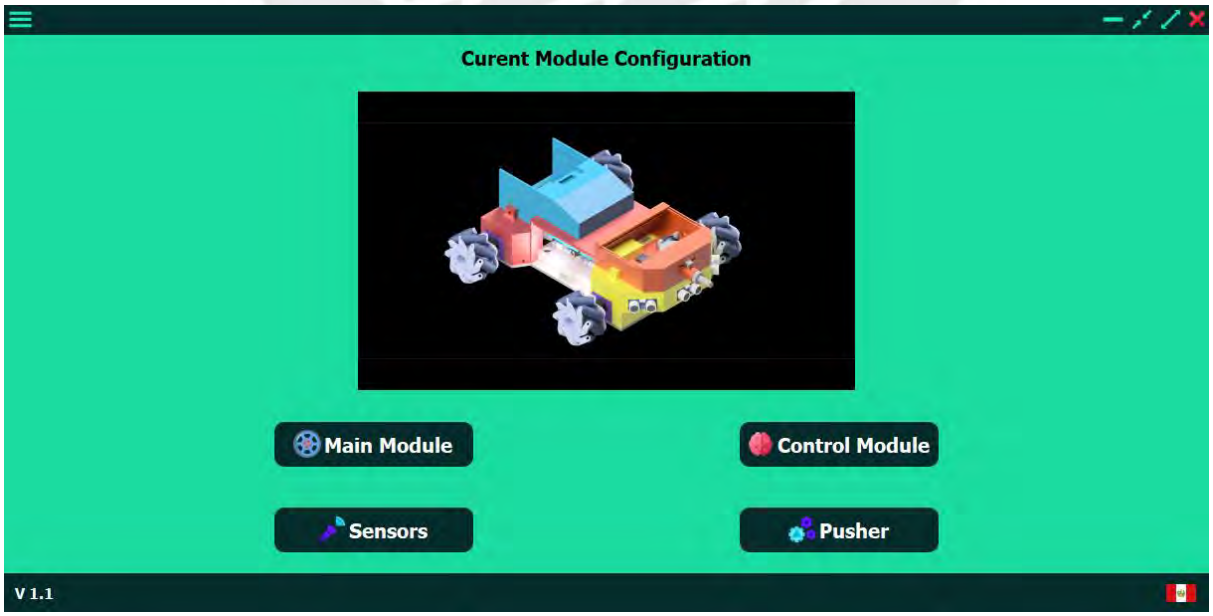


Figura 4. 39 Configuración de Módulos

La opción ‘Battery’ muestra el porcentaje de batería restante que le queda al sistema. Este se ve presentado por una barra de carga, así como un número entero del 0 al 100 por ciento. Esto se aprecia en la Figura 4. 40 a continuación.

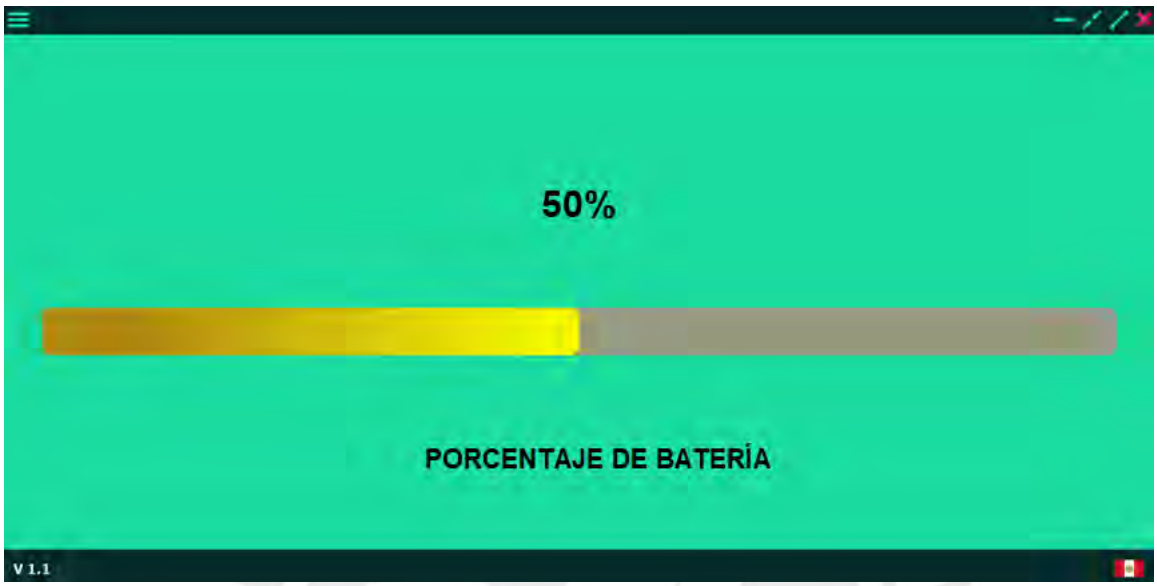


Figura 4. 40 Porcentaje de Batería Restante

La opción ‘Camera’ permite al usuario visualizar la imagen de cámara a la que estaría conectado el robot. Esta imagen dependerá de la cámara que haya conectado el usuario. Esto se ve en la Figura 4. 41 a continuación.

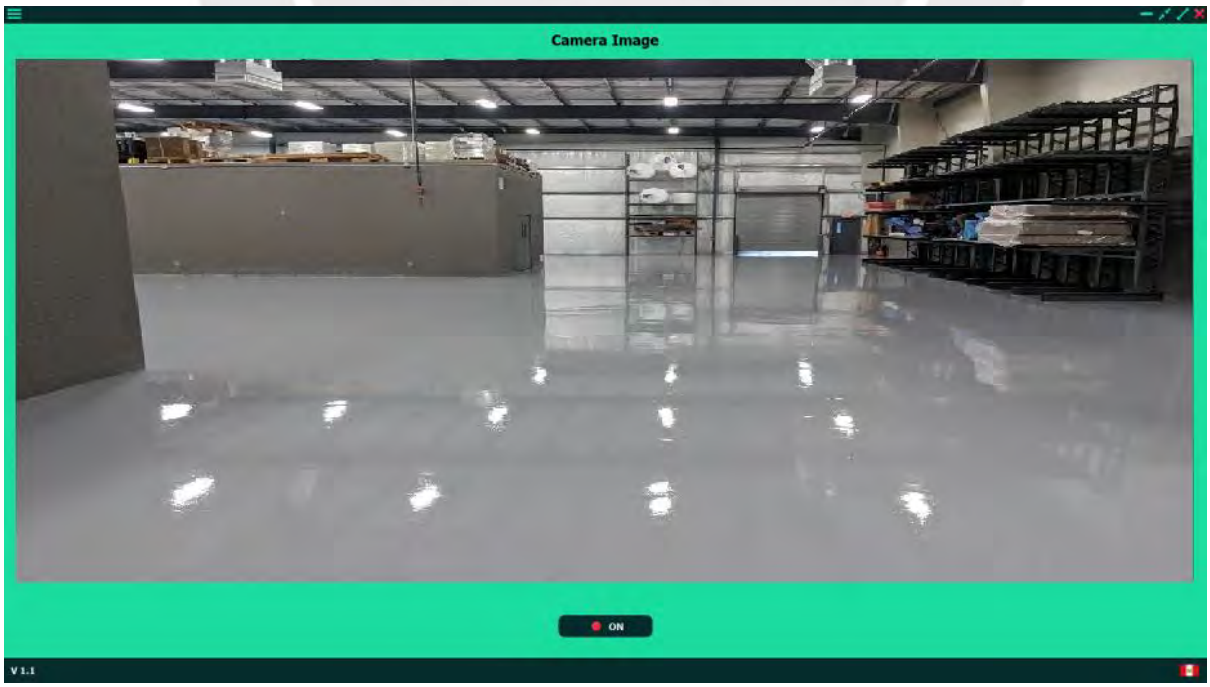


Figura 4. 41 Imagen de Cámara

De manera similar, al dar click a la opción ‘Control’, también se despliega la imagen de la cámara. Sin embargo, al lado derecho de esta se puede ver un panel de control. Este panel de control incluye direcciones en norte, sur, este, oeste, noreste, sudeste, sudoeste y noroeste. Así mismo incluye botones de rotación en sentido del reloj, y en sentido contra reloj. Esto se aprecia el la Figura 4. 42 a continuación.

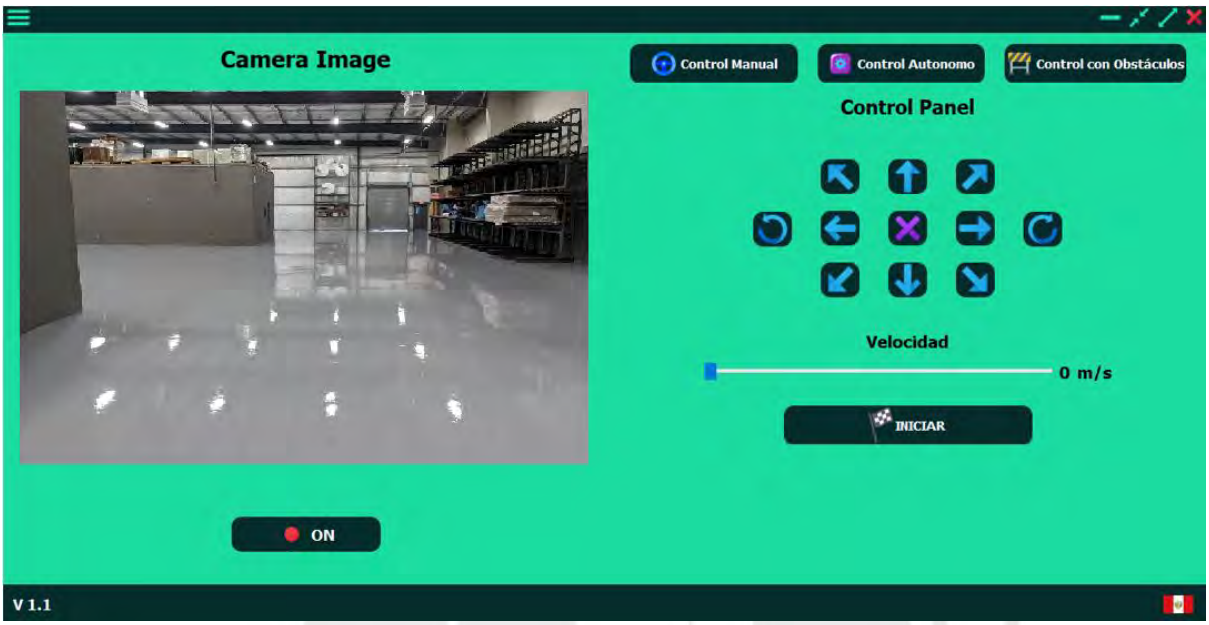


Figura 4. 42 Panel de Control

Adicionalmente a los botones para controlar el sistema de manera manual, se puede ver en la parte superior las opciones de control del robot. Dependiendo de la configuración del robot, este puede tener un modo de control autónomo que le permite avanzar una distancia específica, y un módulo de control de obstáculos que le permite detectar si hay un obstáculo en su camino. En la Figura 4. 43, se puede aprecias un ejemplo de la ejecución de la modalidad de control con obstáculos.

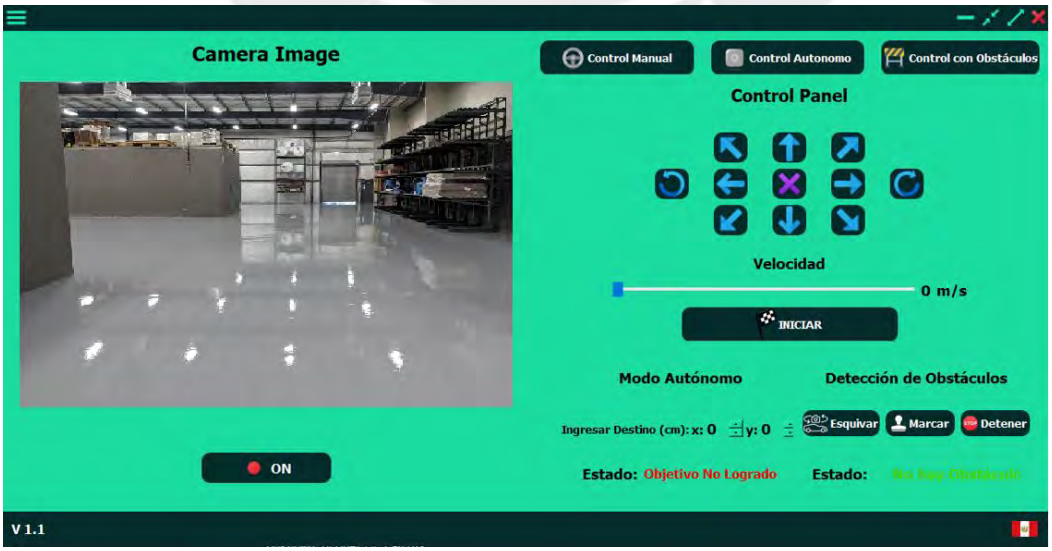


Figura 4. 43 Funciones de Panel de Control

Finalmente, la opción de menú ‘Bluetooth’ permite revisar si se encuentra activo el control de PS4 o no. Ya que es posible controlar al robot tanto por medio del panel de control en la interfaz gráfica, así como utilizando un control bluetooth como el de PS4 como se muestra en la Figura 4. 44 a continuación.

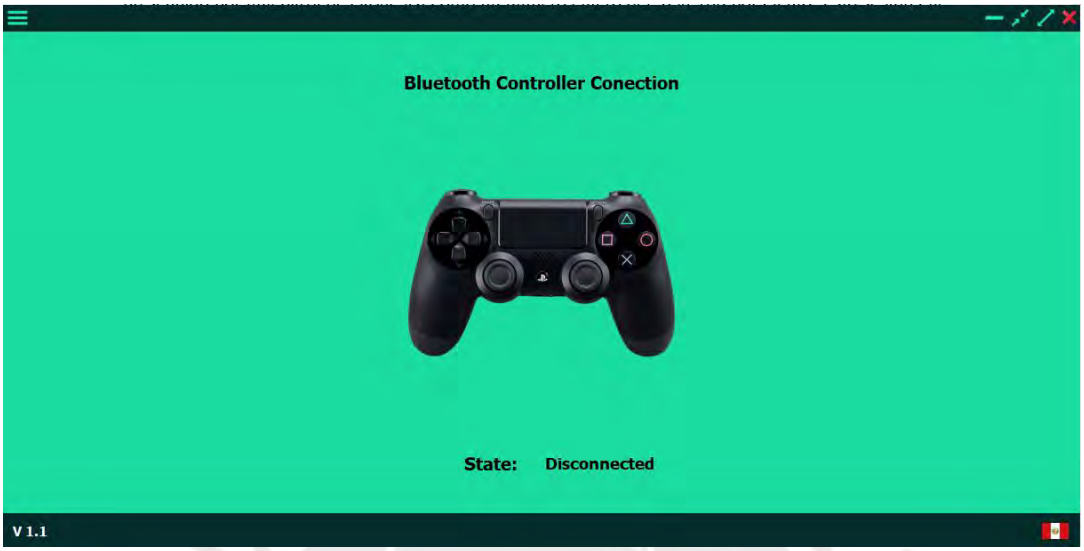


Figura 4. 44 Estado de Conexión Bluetooth

Finalmente, en la imagen a continuación se puede apreciar el flujo que siguen las diferentes secciones de la interfaz gráfica de manera visual. De manera que sea más fácil de comprender.

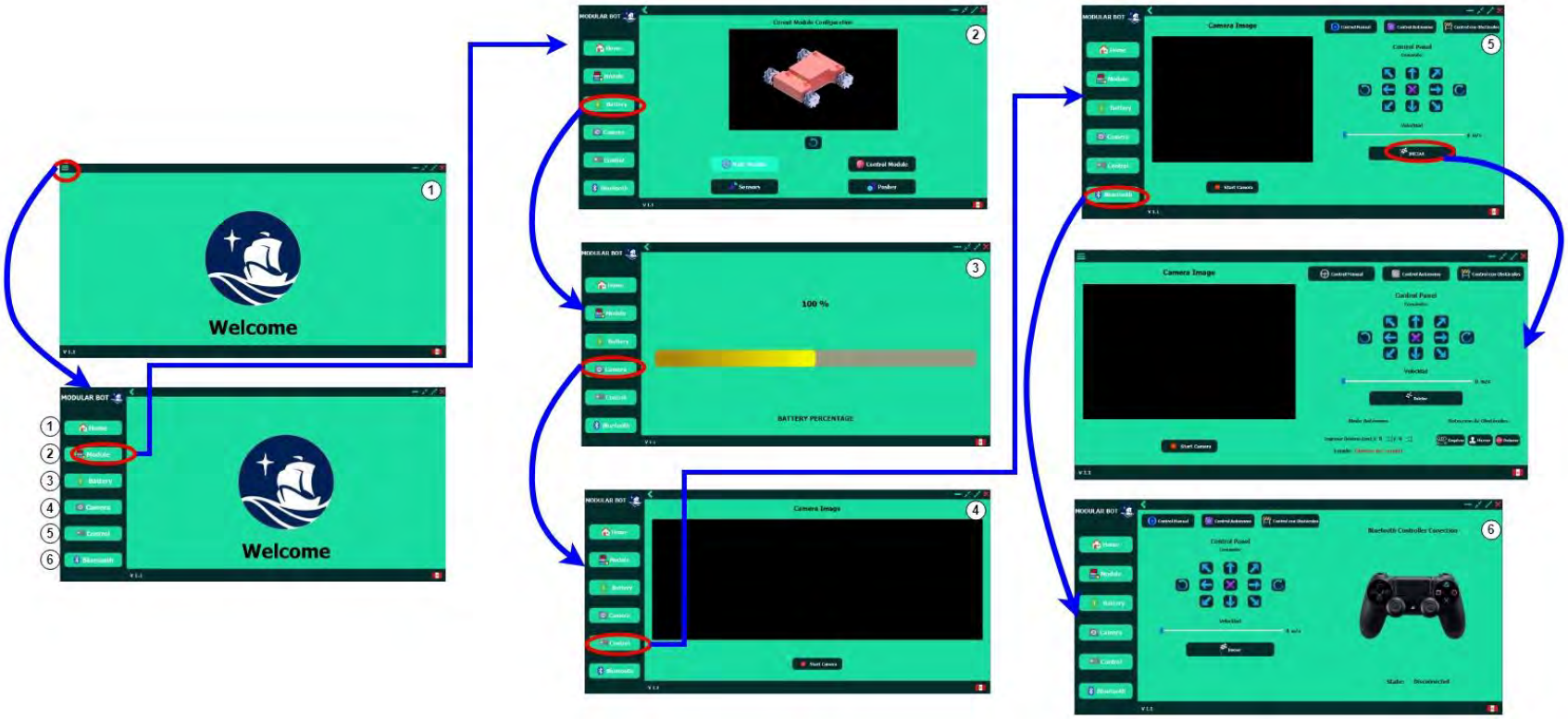


Figura 4. 45 Flujo de Secciones de Interfaz Gráfica

4.4.4 Diagrama de Flujo de Software

La Figura 4. 46, Figura 4. 47 y Figura 4. 48 muestra el diagrama de funcionamiento de software. Este permite entender el funcionamiento de la interfaz gráfica de manera más sencilla, así como las validaciones necesarias para que el programa funcione de manera apropiada. El diagrama se presenta a continuación.

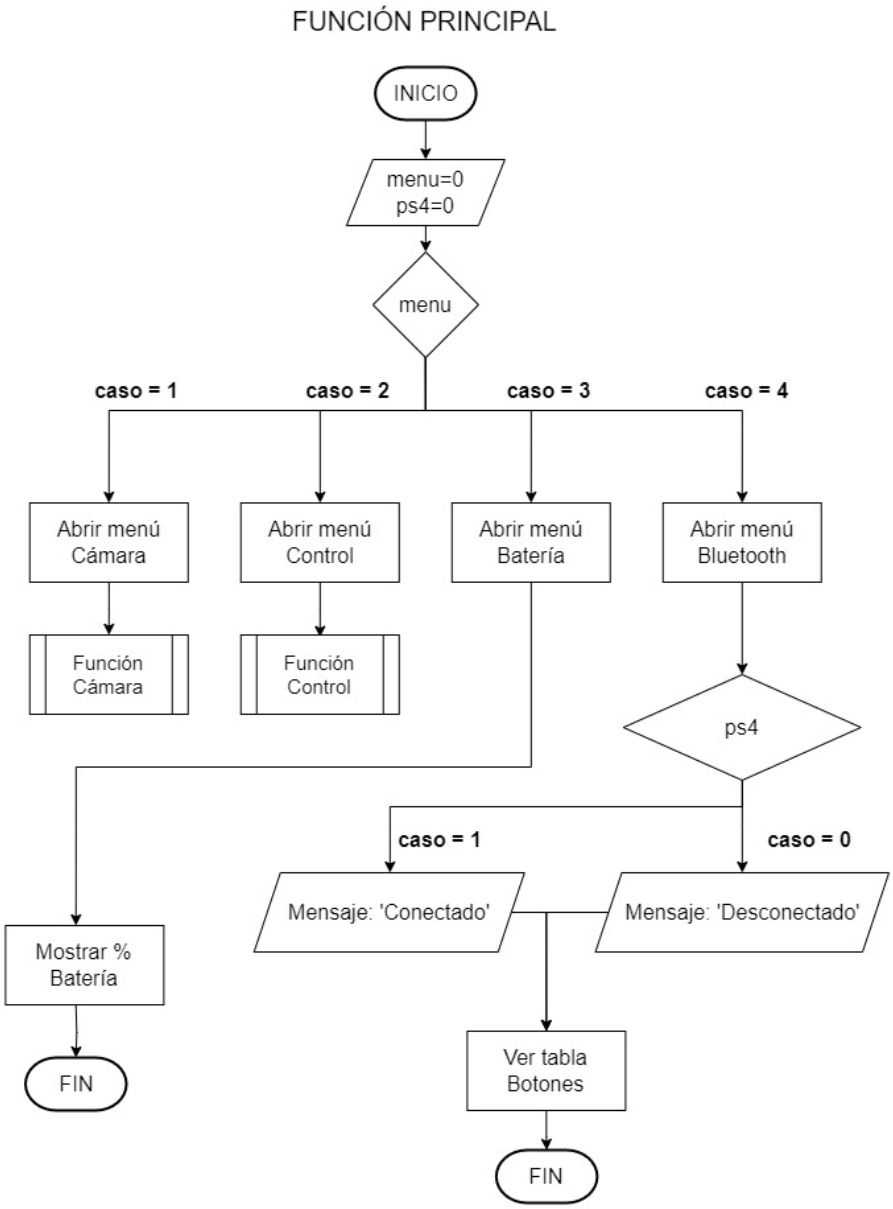


Figura 4. 46 Diagrama de Flujo de Software Función Principal

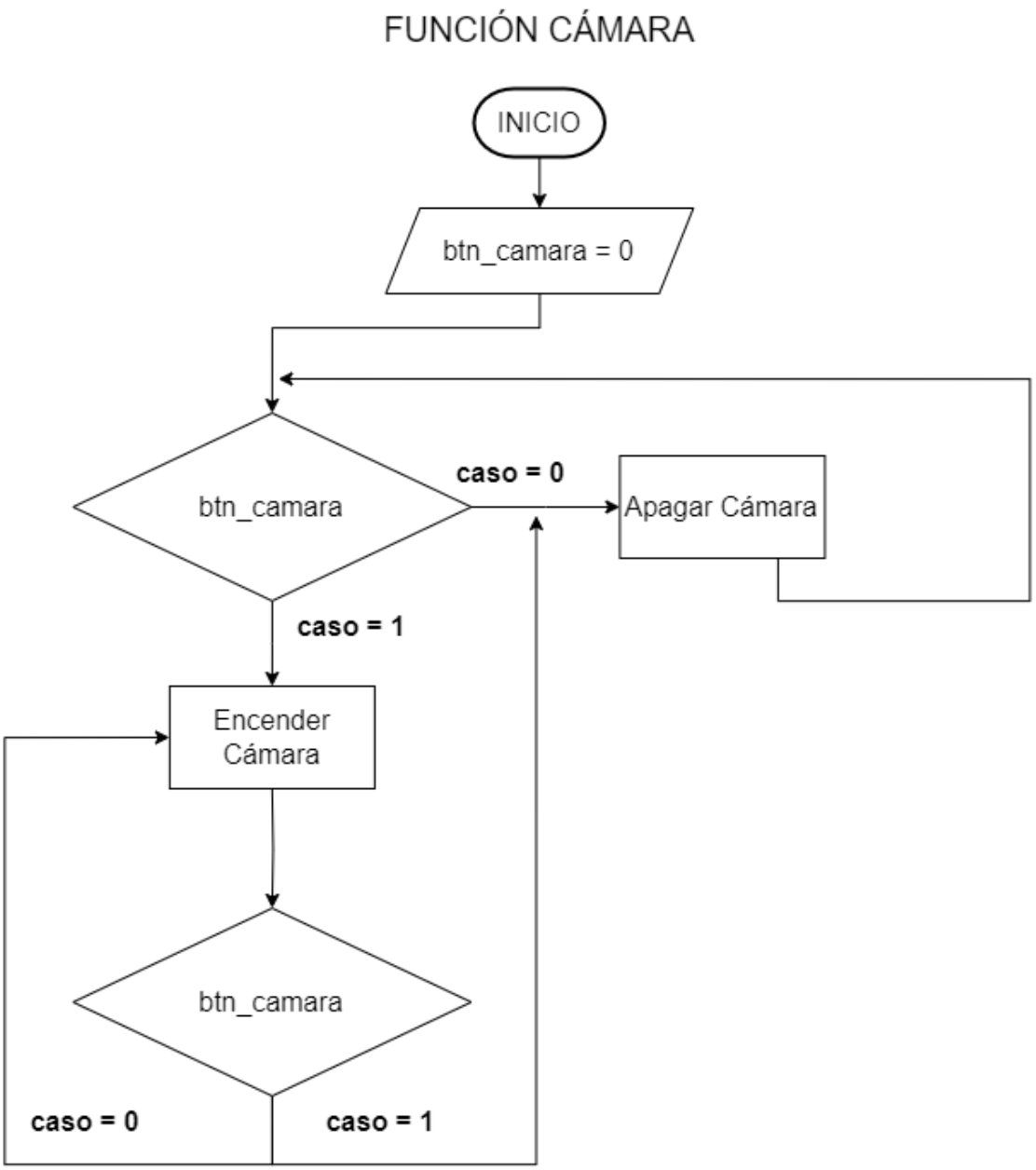


Figura 4. 47 Diagrama de Flujo de Software Función Cámara

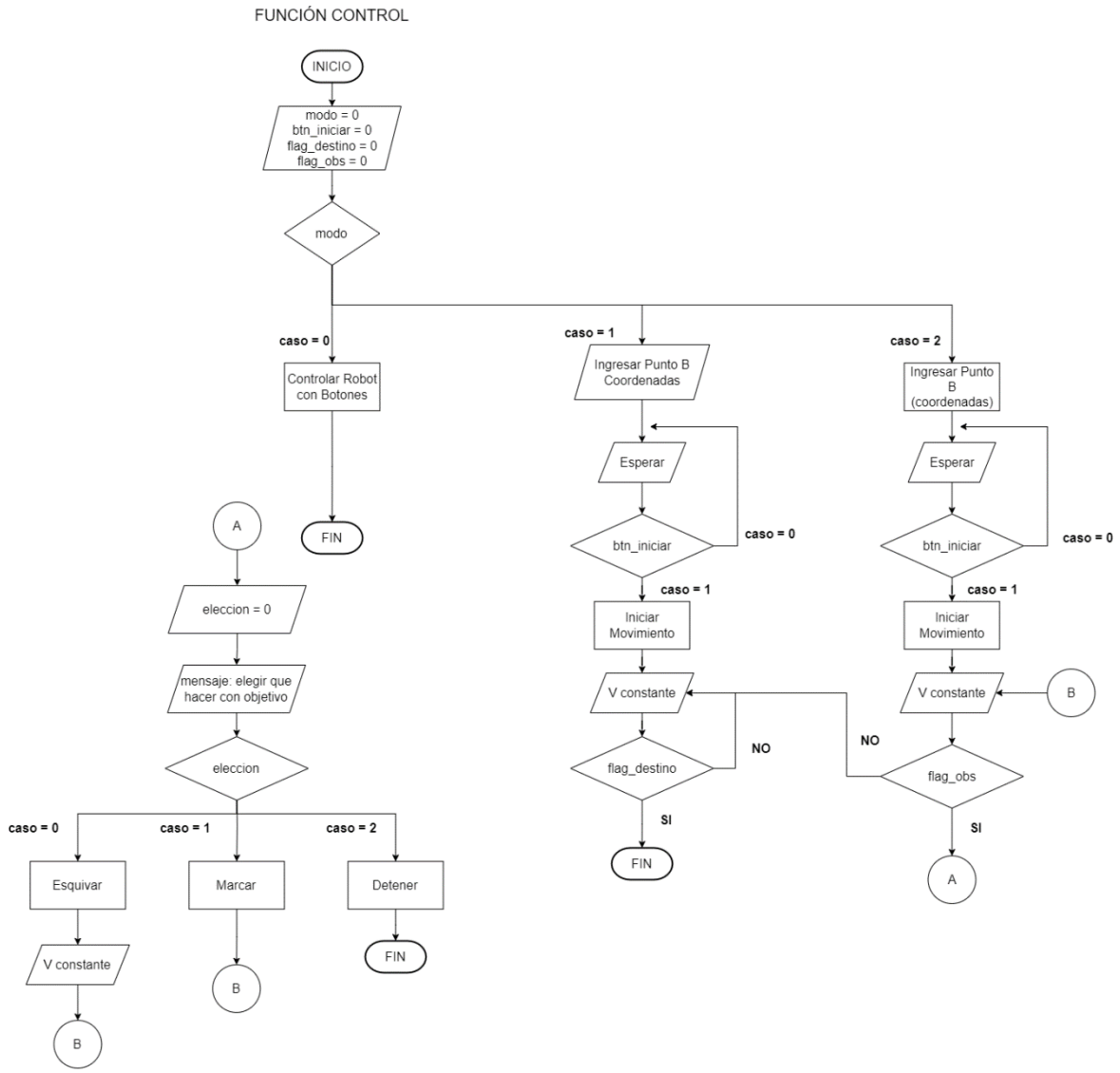


Figura 4. 48 Diagrama de Flujo de Software Función Control

5 Pruebas Preliminares

Para fines de realizar un prototipo de las funciones parciales del robot, principalmente las funciones del módulo principal y algunas funciones del módulo de Control, se realizaron pruebas de los diferentes componentes que conforman estos módulos. A continuación, se presentarán algunas pruebas preliminares que servirán como base para realizar un prototipo parcialmente funcional.

5.1 Pruebas de Motor

Pruebas de alimentación de Voltaje

Lo primero en realizarse fue probar los motores de manera independiente, conectando estos directamente a una fuente regulable, alimentándolos con 12 voltios. El objetivo de esta prueba fue asegurarse de que los 4 motores funcionaran al ser alimentados por los 12 Voltios. Esto se aprecia en la
Figura 5. 1. El video puede verse en el **Anexo 6**.

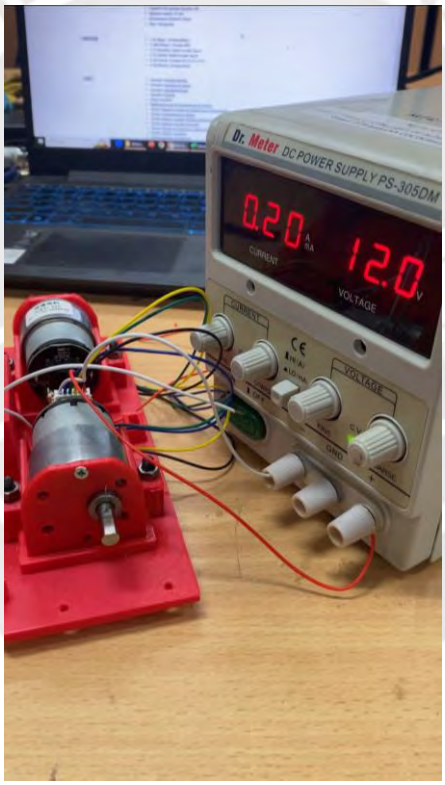


Figura 5. 1 Prueba de Motores con Fuente Regulable Directa

Esta primera prueba verificó el funcionamiento directo de los 4 motores al ser alimentados con un voltaje de 12 V.

Pruebas de funcionamiento de Drivers de Motor

La siguiente prueba tuvo como objetivo el probar el correcto funcionamiento de los motores, siendo estos controlados a los drivers de motores.

La Figura 5. 2 a continuación, muestra el funcionamiento de los 4 motores al mismo tiempo por medio de los drivers de motores y los Arduino uno. En este caso se cargó el mismo programa en cada Arduino de manera independiente.

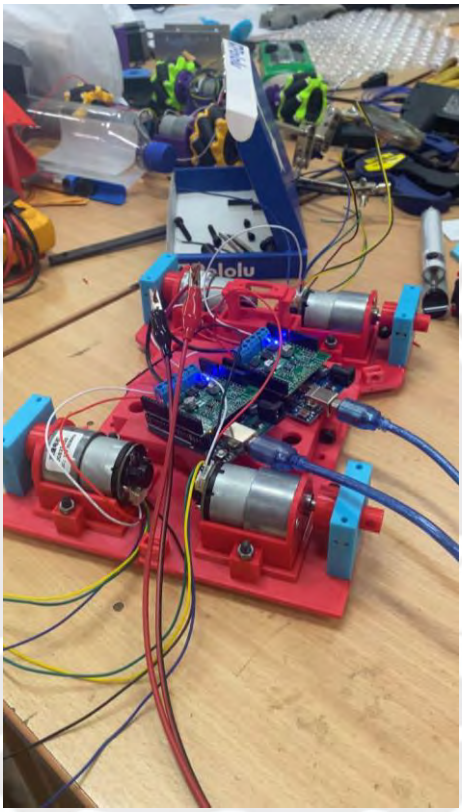


Figura 5. 2 Prueba de Motores con Drivers

El programa de prueba de los motores consistió en un programa de Arduino que controlaba las velocidades de cada motor. En primer lugar, la prueba del **Anexo 7** muestra a los 4 motores moviéndose a diferentes velocidades y direcciones de manera independiente. El segundo video del **Anexo 8** muestra a los 2 motores del lado izquierdo a una velocidad constante, mientras que los motores del lado derecho giran en el sentido contrario. Por último la tercera función mostrada en el **Anexo 9** muestra a los 3 motores moviéndose a una velocidad constante en una misma dirección.

Al poder probar diferentes combinaciones en las que los 4 motores funcionan con diferentes sentidos y velocidades se puede concluir que los drivers de motores funcionan de manera satisfactoria. Dando por concluido el objetivo de este set de pruebas.

5.2 Pruebas de Conexión I2C

Encendido y Apagado de los Leds

La Figura 5. 3 a continuación muestra las pruebas que se realizaron para la conexión I2C. En este caso por cuestiones de disponibilidad se utilizó una Raspberry pi 3 en lugar de la Jetson nano. El objetivo principal de estas fue comprobar el correcto funcionamiento de la conexión I2C.

Para la primera prueba, se verificó la conexión I2C controlando los 2 leds incorporados en el Arduino por el pin 13. Esto se verificó conectando 2 Arduinos, tomando a la Raspberry Pi como maestro. El código para configurar la raspberry pi como maestro puede verse en el **Anexo 2**.

Al presionar el botón '1' se prendía el led del primer Arduino, al presionar '0' se apagaba. En el caso del segundo Arduino se presiona '3' para encender y '2' para apagar. De igual manera, el video visto en el **Anexo 10** muestra la interacción correcta de los LEDs.

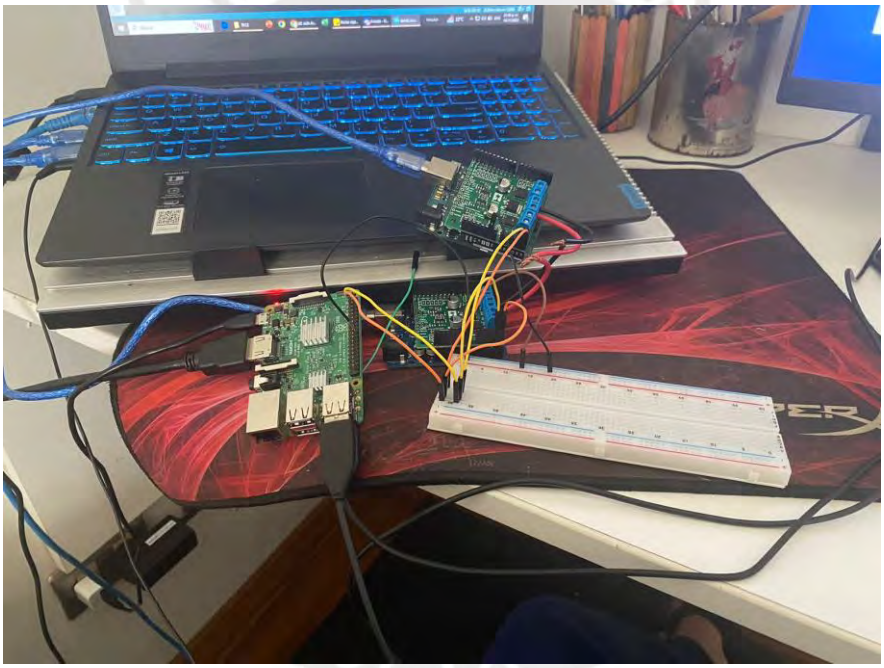


Figura 5. 3 Prueba de Conexión I2C

El video del **Anexo 10** muestra de manera exitosa como se pudo controlar ambos leds de los 2 Arduinos independientes por medio de un solo dispositivo maestro. Dando por exitosa la prueba.

Prueba de Motores con I2C

De manera similar a los Leds en la anterior prueba, se procedió a probar la conexión I2C utilizando los 4 motores. El objetivo de esta segunda prueba fue verificar que los 4 motores pudiesen ser controlados con la Raspberry.

En el **Anexo 11** se puede ver un video que muestra como a través de la raspberry pi fue posible controlar los 4 motores al mismo tiempo. En el comienzo del video se aprecia como primero fueron controlados los 4 motores alimentados por la batería, mientras la raspberry era alimentada por la computadora. A continuación, se puede ver en la Figura 5. 4 una captura de la prueba en cuestión. El código para dicha prueba puede verse en el **Anexo 3**.

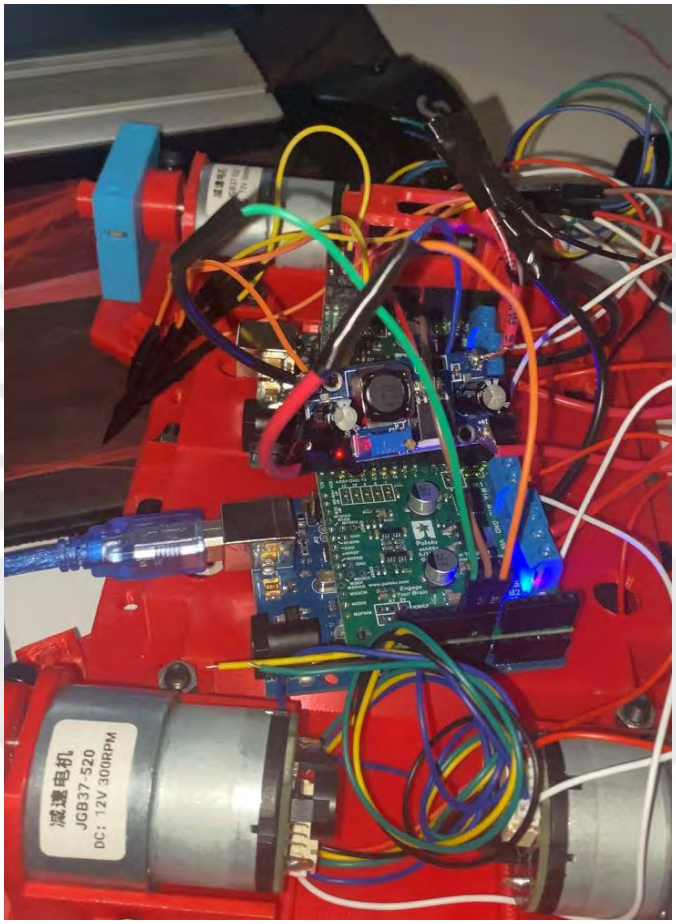


Figura 5. 4 Prueba de Motores con I2C

Al ser capaz de controlar los 4 motores por medio de la Raspberry se da por concluida esta segunda prueba de manera exitosa.

Prueba de Movimiento con Motores

Finalmente, se configuraron los motores para poder realizar pruebas de movimiento. Se tenía 2 objetivos principales con esta prueba. En primer lugar, verificar el funcionamiento del movimiento simple del REM bot por medio de una conexión I2C. Por otro lado, el segundo objetivo de esta prueba fue el realizar esta acción de manera completamente inalámbrica. Para ello, se utilizó un cargador portátil para alimentar a la raspberry, así como la aplicación 'Real VNC Viewer' para poder controlar la Raspberry de manera remota.

De esta manera, se pudo ver los primeros pasos de la configuración omnidireccional como se aprecia en el video del **Anexo 12** y en la Figura 5. 5 . El código del **Anexo 3** utilizado en esta prueba incluye movimientos en direcciones de norte, sur, este, oeste, así como sus variantes de manera diagonal. Finalmente, el código incluye movimiento rotatorio en sentido del reloj, y contra sentido del reloj. El video muestra una pequeña prueba de movimiento lineal. Debido a un fusible quemado, a pesar de completar la programación no se pudo completar las pruebas en el sistema de movimiento.

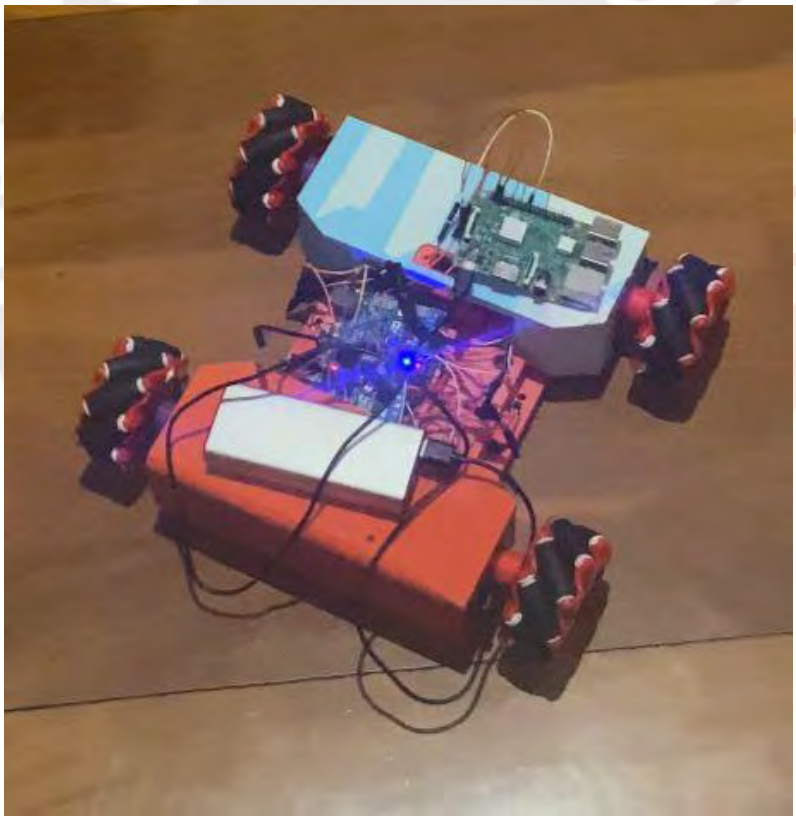


Figura 5. 5 Prueba de Sistema de Movimiento Omnidireccional

Respecto al primer objetivo, si bien la prueba buscaba un movimiento un poco más complejo, se consiguió realizar un movimiento simple que demostró la correcta ejecución de las conexiones y el código empleado. Así mismo, se cumplió por completo el segundo objetivo que buscaba realizar esta prueba de manera remota.

5.3 Pruebas de Detección de Módulos

Prueba de detección inalámbrica por medio de Raspberry Pi

Habiendo realizado las pruebas correspondientes con los motores, se procedió a realizar la prueba de detección de módulos de manera inalámbrica. El primer objetivo de esta primer prueba fue de corroborar por medio de jumpers si el código empleado en Python del **Anexo 4** en la Raspberry permitía detectar correctamente los módulos conectados.

Para ello, se utilizó una Raspberry pi, conectada por medio de USB a 4 Arduinos. Para fines prácticos, se utilizó un Arduino como maestro, y otros 3 Arduinos como esclavos. Cada uno de estos esclavos representa al módulo de control, el módulo de sensores y el módulo de actuadores respectivamente. Mediante el software VViewer y el uso de un monitor, se pudo realizar la prueba de manera remota mediante una conexión bluetooth entre una laptop y la Raspberry pi. Esto se puede ver en la
Figura 5. 1, y se aprecia de mejor manera en el video contenido en el **Anexo 13**.

En el video se puede observar que la terminal del código de Python se conecta al puerto serial del Arduino maestro, mostrando así los módulos que se encuentran conectados al ingresar el comando ‘8’. Al desconectar uno de los Arduinos y volver a ejecutar el comando ‘8’, se muestra eficientemente el estado desconectado del mismo, ya que el mensaje visto en pantalla coincide con los módulos conectados en la vida real.

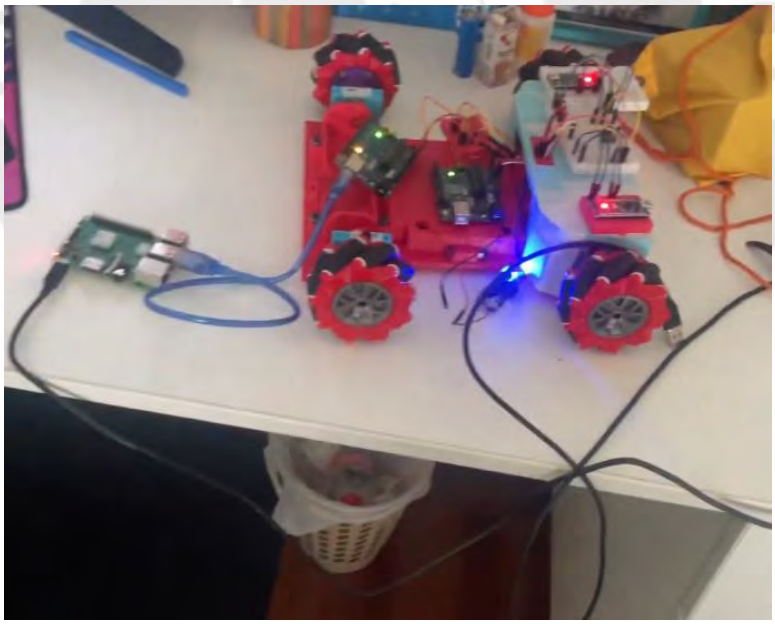


Figura 5. 6 Prueba Inalámbrica de Detección de Módulos

El segundo objetivo fue verificar la consulta de comandos. Para poder consultar la acción de un comando específico, de debía ingresar a la terminal la palabra ‘help’ seguida de el nombre de algún comando. Igualmente, al ingresar solo ‘help’ la terminal mostrará todos los comandos disponibles. Esto se puede apreciar en la segunda mitad del video, dado por completado el segundo objetivo. La lista completa de los comandos se puede apreciar en el **Anexo 5**.

Prueba de detección por medio de la interfaz gráfica

Utilizando la misma configuración vista en la prueba anterior (conexión I2C con 4 Arduinos), se procedió a mostrar de manera visual la detección de módulos por medio de la interfaz gráfica. Se utilizó la sección de módulo, vista anteriormente en la sección de **Arquitectura de Software**, para poder mandar al Arduino maestro el comando correspondiente. Al presionar el botón que lleva al usuario a la sección modular, se envía el comando y se realiza la consulta. El objetivo de esta prueba es la correcta detección de los módulos de una manera visual y más amigable con el usuario. El código utilizado es el código de Python que dio vida a la interfaz gráfica, presentado en el **Anexo 1**.

Los símbolos mostrados en la pantalla correspondieron a sus respectivos módulos cambiando de color mostrando la conexión o desconexión dejando como evidencia el cumplimiento del objetivo propuesto. Así mismo, un botón de refresco de imagen actualiza la imagen de la parte superior al módulo que actualmente se encuentra conectado. Todo esto puede apreciarse en la Figura 5. 7 y en el video del **Anexo 14**.

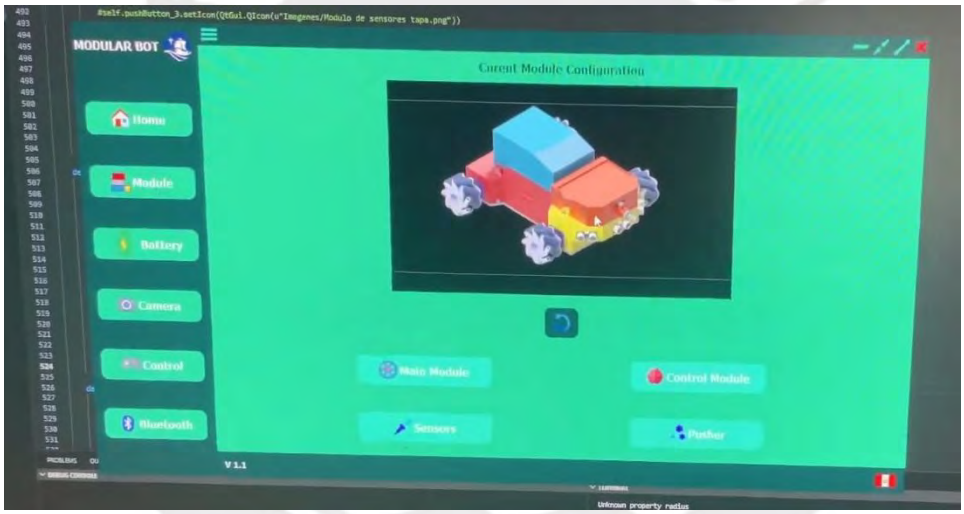


Figura 5. 7 Detección de módulos por medio de Interfaz gráfica

De igual manera, para que el usuario sea consciente de los comandos que internamente envían los botones para controlar al robot, se agregó una sección en la parte de control donde se indica que comando se le está mandando al controlador maestro. Esta acción puede apreciarse en el video del **Anexo 15** donde se envían los comandos correspondientes vistos en la sección anterior y en el **Anexo 5**.

6 Costos Parciales

A partir de la selección de componentes, se puede calcular un costo aproximado del sistema integrado del robot. Estos costos serán separados por módulo, y se tomará en cuenta el costo de cada pieza particularmente. El resumen de costos puede apreciarse en la Tabla 6. 1 a continuación.

Tabla 6. 1 Resumen de Costos Parciales del Sistema en Soles

Item	Componente	Módulo	Cantidad	Precio/Unidad	Precio Total
1	Arduino Uno	Omnidireccional	2	35	70
2	VNH5019 Driver	Omnidireccional	2	55	110
3	Relé de 12 V	Omnidireccional	1	10	10
4	Diodo 1N4007	Omnidireccional	1	0.5	0.5
5	Motor DC 37D	Omnidireccional	4	65	260
6	Arduino Nano	Control	1	30	30
7	Jetson Nano	Control	1	500	500
8	Led RGB	Control	1	0.5	0.5
9	IMU	Control	1	18	18
10	Modulo Bluetooth	Control	1	25	25
11	Ultrasonido	Sensores	3	8	24
12	Arduino Nano	Sensores	1	30	30
13	Motor Amarillo	Actuadores	1	18	18
14	Driver de Motores	Actuadores	1	15	15
15	Arduino Nano	Actuadores	1	30	30
16	Sensor Infrarrojo	Actuadores	1	4	4
17	PLA para Impresión Carcasas	General	1	97.5	97.5
18	Reguladores de Voltaje	General	4	5	20
19	PCBs	General	1	250	250
				Precio Total	1512.5

A partir del precio total, se puede determinar que la selección de componentes permitirá que el sistema pueda venderse a un menor costo que los robots modulares ofrecidos por la competencia. Tal como el robot modular ‘Turtle Bot’, el cual tiene un precio de venta de 4500 soles.

Conclusiones

En cuanto al estado del arte es posible resaltar que modelos comerciales tales como el Xiaomi Modular bot o el Clic bot son más complejos y por lo tanto más costosos. Por otro lado, robots dedicados a la educación son mucho más simples en diseño, pero mucho más prácticos a la hora de aprender. De esa manera se buscó que el sistema diseñado tenga características más similares al tipo de robots dedicados a la educación. Características tales como un diseño simple y económico.

Por otro lado, se logró cumplir con todos los requerimientos plasmados en la lista de requerimientos. En primer lugar se logró diseñar los 4 módulos diferentes que pueden comunicarse entre sí y unirse muy fácilmente por medio de conexiones modulares. Al haber una sola conexión directa por módulo 'la cual está muy bien indicada' no fue necesario etiquetar cada conexión. En cuanto a las uniones por forma o con pocas uniones con tornillos se logró implementar uniones para los módulos que no requieren más de un par de tornillos fáciles de insertar y de remover. Se logró utilizar una configuración omnidireccional como se pensó desde un inicio, y se respetó que el tamaño total del robot no sobrepasara los 40cm por lado, ancho y profundo. Así mismo, se acondicionó el sistema de alimentación desde el módulo principal.

En cuanto al diseño preliminar, se decidió tomar lo mejor de cada concepto de diseño para así poder realizar un diseño más efectivo. Al momento de seleccionar cada modelo de los componentes electrónicos, se selecciono cada uno tomando en cuenta no solo el menor costo, sino las ventajas que este brindaría al sistema. Como el caso de los drivers VNH para los 4 motores del módulo principal, los cuales a pesar de tener un costo elevado se adaptan de manera muy eficiente al Arduino uno, sin mencionar que soportan los picos de corriente de los motores seleccionados. De esta manera se consiguió que el diseño tenga un costo de fabricación de 1550 soles, significativamente menor a competencias como el Turtle bot con costo superior a 1000 USD. El realizar cálculos de torque requerido, permitieron que se pueda seleccionar un motor adecuado. Este considera una pendiente de 20 grados, y un factor de seguridad de 1.5 determinando un torque necesario de 0.19 Kg.cm, por lo que se pudo elegir un motor con un torque de 0.28 Kg.cm.

Para realizar el diseño electrónico, el desarrollo de los 4 diagramas de conexiones permitió tener una idea más clara de las conexiones físicas entre cada componente. El conocer los pines que serían necesarios, permitió la realización de las tarjetas electrónicas que permitieron lograr un orden y mejor eficiencia en cada uno de los módulos. El diseño electrónico supo aprovechar el espacio de manera apropiada, así como minimizar gastos al mantener las tarjetas en una sola capa y de área reducida. Finalmente, el calculo realizado para la capacidad energética permitió seleccionar baterías apropiadas para el funcionamiento del robot. Dando como resultado una autonomía necesaria de 2.36 Ah, y seleccionando unas baterías de 2.8 Ah.

El diseño de software, involucra una interfaz gráfica que permite a los alumnos el ver de manera más clara que módulos se encuentran conectados. Esto les permitirá saber si las conexiones que han hecho se han realizado de manera apropiada, así como conocer la energía restante de las baterías y controlar el robot. Finalmente, podrán hacer uso de la navegación autónoma según los módulos que estén conectados. El diseño de la interfaz es bastante simple y gráfico, permitiendo que a la interacción de los alumnos con el sistema sea menos compleja y más amena.

Finalmente, las pruebas del prototipo dejan como evidencia un funcionamiento parcial de lo que sería el robot final. Esto permite conocer parcialmente de lo que es capaz el robot, así como afianzar la confiabilidad de los diseños electrónicos y los diseños de software implementados al sistema.

Recomendaciones

Si bien los requerimientos del sistema fueron cumplidos de manera satisfactoria, hubo cosas que aun podrían mejorar para un trabajo futuro. En primera instancia, al momento de realizar los diseño mecánicos era necesario imprimir las piezas en 3D para poder validar por completo el adecuado diseño de cada pieza. Esto costo mucho tiempo y recursos para rediseñar las diferentes piezas de manera que encajen y puedan ensamblarse de manera sencilla.

De manera similar, al momento de realizar el prototipo final hubo inconvenientes con los drivers de motores; debido a que un cable suelto provocó un corto circuito que quemó uno de los mismos. Esto significó que no se pudiera completar el prototipo. La razón del corto fue debido al desorden de los cables al no contar con la tarjeta electrónica impresa en físico. Para un futuro trabajo de investigación, si no es posible imprimir la tarjeta se tomará en cuenta el aislar correctamente los componentes para evitar inconvenientes de último minuto.

En cuanto a los tornillos utilizados, el diseño se ajustó para que la gran mayoría fuesen de 5mm de diámetro. Sin embargo, esto agrega peso innecesario al sistema. En otra ocasión se podría tomar en cuenta el uso de tornillos más pequeños en uniones que lo permitan. De esta manera, poder disminuir el peso del robot y la carga que tendrán que aguantar los motores.

En cuanto a temas energéticos, si bien 30 minutos son suficientes para una experiencia completa para un alumno, este tiempo de duración de las baterías podría mejorar. Si bien esto implicaría un incremento de costo al tener que utilizar baterías más costosas, es una inversión que valdría la pena con tal de poder utilizar el robot por más tiempo sin tener que volver a cargarlo.

Finalmente, una última recomendación sería intentar reducir el tamaño del sistema. Si bien el sistema ha cumplido con el límite de dimensiones de 40cm, este podría ser más pequeño y así más económico y fácil de usar. Ya que no todos los ambientes de laboratorio tiene espacios tan amplios. Para lograr esto, las pruebas de manera física son indispensables, y se tomará en consideración para un futuro trabajo.

Referencias

Abdulkadir, S. A., Miskon, S., & Abdullah, N. S. (2019). The role of ICT in project-based learning: A literature review. *International Conference on Research and Innovation in Information Systems, ICRIIS*, *December-2019*. https://doi.org/10.1109/ICRIIS48246.2019.9073379

Clear Path. (s. f.). *TurtleBot 4 - Clearpath Robotics*. Recuperado 3 de abril de 2023, de https://clearpathrobotics.com/turtlebot-4/

Patiño, R., Barrios D., Bernedo, L., Alsina, P., Garcia, L. (2019). EDUROSC-Kids: An Educational Robotics Standard Curriculum for Kids. https://ieeexplore.ieee.org/document/9018550

Dhaouadi, R., & Sleiman, M. A. (2011). Development of a modular mobile robot platform: Applications in motion-control education. *IEEE Industrial Electronics Magazine*, *5*(4), 35-45. https://doi.org/10.1109/MIE.2011.943024

GCtronic. (s. f.). *e-puck2 - GCtronic wiki*. Recuperado 3 de abril de 2023, de https://www.gctronic.com/doc/index.php?title=e-puck2

Gilpin, K., & Rus, D. (2010). Modular robot systems. *IEEE Robotics and Automation Magazine*, *17*(3), 38-55. https://doi.org/10.1109/MRA.2010.937859

Guo, T., Li, M., Han, Y., Lu, G., Qie, T., & Zhang, Q. (2020). Design of Educational Mobile Robot. *Proceedings - 2020 Chinese Automation Congress, CAC 2020*, 3234-3238. https://doi.org/10.1109/CAC51589.2020.9326549

Halfacree Gareth. (2022). *Clearpath Robotics Launches the TurtleBot 4, Offering an Affordable Autonomous ROS 2 Robot Platform - Hackster.io*. Clearpath Robotics Launches the TurtleBot 4, Offering an Affordable Autonomous ROS 2 Robot Platform. https://www.hackster.io/news/clearpath-robotics-launches-the-turtlebot-4-offering-an-affordable-autonomous-ros-2-robot-platform-6bc3d6a10cbd

IT University of Copenhagen. (s. f.). *EMERGE*. Recuperado 3 de abril de 2023, de https://sites.google.com/view/emergemodular

Key I Robot. (s. f.). *Meet ClicBot: 1000 robots in one*. Recuperado 3 de abril de 2023, de https://keyirobot.com/pages/products-page

Lenskiy, A., Junho, H., Dongyun, K., & Junsu, P. (2014). Educational platform for learning programming via controlling mobile robots. *Proceedings of 2014 International Conference on Data and Software Engineering, ICODSE 2014*. https://doi.org/10.1109/ICODSE.2014.7062695

Mod Robotics. (s. f.). *Cubelets robot blocks - Modular Robotics | Cubelets robot blocks*. Recuperado 4 de abril de 2023, de https://modrobotics.com/

Orientación Universitaria Perú. (2020). *¿Cuántas carreras de ingeniería existen en Perú?* ¿Cuántas carreras de ingeniería existen en Perú? https://orientacion.universia.edu.pe/infodetail/orientacion/orientacion_vocacional/cuantas-carreras-de-ingenieria-existen-en-peru-6445.html

Pedros, C. (2022, septiembre 27). *Lo último de Xiaomi es un futurista robot modular que cobra vida mediante realidad virtual*. Lo último de Xiaomi es un futurista robot modular que cobra vida mediante realidad virtual. https://www.mundoxiaomi.com/noticias/ultimo-xiaomi-futurista-robot-modular-que-cobra-vida-mediante-realidad-virtual

Robotis, (2017). *Turtle Bot 3. https://en.robotis.com/model/page.php?co_id=prd_turtlebot3#*

Yim, M., Zhang, Y., Duff, D. (2002, febrero 01). Modular Robots. *When a task or terrain varies, reconfigurable robots can change their shape to get the job done. https://spectrum.ieee.org/modular-robots*

Naylamp. (2024). ArduBoard Starter Kit . *ArduBoard Starter Kit* . *https://naylampmechatronics.com/ardusystem/92-arduboard-starter-kit.html*

Naylamp. (2024). Chasis Plataforma Robot Móvil. https://naylampmechatronics.com/robotica/105-chasis-plataforma-robot-movil-2wd.html

Pei, Z., & Nie, Y. (2018). Educational robots: Classification, characteristics, application areas and problems. *Proceedings - 2018 7th International Conference of Educational Innovation through Technology, EITT 2018*, 57-62. https://doi.org/10.1109/EITT.2018.00020

Positronic. (2021). *Modular Connectors vs. Hardwiring - Positronic*. https://www.connectpositronic.com/en/2021/07/23/modular-connectors-vs-hardwiring/

PUCP. (2023). *Plan de estudios - Estudios Generales CienciasEstudios Generales Ciencias*. https://facultad.pucp.edu.pe/generales-ciencias/informacion-para-el-estudiante/plan-de-estudios/?especialidad=ingenieria-mecatronica

Ratanaphanyarat, G., & Ding, D. (2020). *US10802499B2 - Autonomous modular robot - Google Patents*. https://patents.google.com/patent/US10802499B2/en?q=(modular+robot)&oq=modular+robot

Ryland, G. G. (2019). *US20160005331A1 - Modular robot system - Google Patents*. https://patents.google.com/patent/US20160005331A1/en

Sanger, P. A., & Ziyatdinova, J. (2014). Project based learning: Real world experiential projects creating the 21st century engineer. *Proceedings of 2014 International Conference on Interactive Collaborative Learning, ICL 2014*, 541-544. https://doi.org/10.1109/ICL.2014.7017830

Thompson, M. K., Thompson, J. M., Speth, R. L., & Sallum, H. M. (s. f.). *US6877574B2 - Modular robotic teaching tool - Google Patents*. 2005. Recuperado 3 de abril de 2023, de https://patents.google.com/patent/US6877574B2/en

umrobotics. (2022). *(307) Robotics 102: Intro to AI & Programming - YouTube* [Video recording]. https://www.youtube.com/watch?app=desktop&v=jZ0U339ewKo&feature=youtu.be&ab_channel=umrobotics

Universiry Of Michigan. (2021). *ROB 102: Intro to AI & Programming | Michigan Robotics*. ROB 102: Intro to AI & Programming. https://robotics.umich.edu/academics/courses/course-offerings/rob-102-intro-to-ai-programming/

Alvarez, S. (2019). *Navegación Autónoma de un Robot Móvil Omnidireccional en Trayectorias Predeterminadad con evitación de Obstáculos*. https://repositorio.uta.edu.ec/items/204a9fd5-57d4-4685-b367-cc082d4a2059

Castillo, B. (2023). *Control de posicionamiento en base a Velocidad Angular de las Ruedas. https://dspace.uazuay.edu.ec/bitstream/datos/13424/1/18949.pdf*

Anexos

Anexo 1

Código de interfaz gráfica desarrollada en Python

```
#Autor: Sebastian Ruiz

import sys
import serial
from PyQt5.uic import loadUi
from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtCore import Qt
from PyQt5.QtCore import QPropertyAnimation
import time
from PyQt5.QtCore import QThread, pyqtSignal
import cv2

width = 960 #Ancho del video mostrado en la interfaz
height = 480 #Alto del video mostrado en la interfaz

slave1 = 0
slave2 = 0
slave3 = 0

mode = 0

flag_ruta = 0


class ArduinoThread(QThread):
    data_received = pyqtSignal(str)

    def __init__(self, port, baudrate):
        super().__init__()
        self.ser = None
        self.is_running = False
        try:
            self.ser = serial.Serial(port, baudrate, timeout=0.1)  # Use a short timeout
            self.is_running = True
        except serial.SerialException as e:
            print(f"Error opening serial port: {e}")

    def run(self):
        while self.is_running:
            try:
                data = self.ser.readline().decode().strip()
                if data:
                    self.data_received.emit(data)  # Emit the received data
            except Exception as e:
```

```
            print(f"Error: {e}")

    def send_data(self, data):
        if self.ser:
            self.ser.write(data.encode())

    def stop(self):
        self.is_running = False
        if self.ser:
            self.ser.close()


class VideoThread(QThread):
    change_pixmap_signal = pyqtSignal(bytes)

    data_received = pyqtSignal(str)


    #Se inicia la clase Video Threat
    def __init__(self,directory):
        super().__init__()
        self.is_running = True
        self.directory = directory


    def run(self):
        global width
        global height
        global posicion_label
        global distancia_abs

        # Se define la camara que se desea utilizar
        cap = cv2.VideoCapture(0)

        # Se determina el tamanio de captura de imagen definido por las variables locales
        cap.set(cv2.CAP_PROP_FRAME_WIDTH, width)
        cap.set(cv2.CAP_PROP_FRAME_HEIGHT, height)

        counter = 1

        #Se inicia el primer threat
        while self.is_running:

            # Capture a frame
            ret, frame = cap.read()

            if ret:
                # En caso se halla cambiado el tama;o de senal del video por agrandar o encoger la pantalla se
cambia el tamano de imagen.
                if width != 960 or height != 540:
```

```
                frame = cv2.resize(frame, (width, height))

            # Convertimos imagen a jpg
            is_success, buffer = cv2.imencode(".jpg", frame)
            if is_success:
                # Emit the bytes signal
                self.change_pixmap_signal.emit(buffer.tobytes())


        cap.release()

    def stop(self):
        self.is_running = False



class MainPage(QtWidgets.QMainWindow):
    def __init__(self):
        super(MainPage, self).__init__()
        loadUi("Interfaz.ui", self)
        self.setWindowFlag(QtCore.Qt.FramelessWindowHint)
        self.setAttribute(QtCore.Qt.WA_TranslucentBackground)

        self.camera_thread = VideoThread("your_directory_here")
        self.camera_thread.change_pixmap_signal.connect(self.update_image)
        self.btn_grabar.clicked.connect(self.toggle_camera)
        self.camera_running = False

        # Try to initialize the Arduino thread
        self.arduino_connected = False
        self.arduino_thread = ArduinoThread('COM5', 9600)
        try:

            if self.arduino_thread.is_running:
                self.arduino_connected = True
                self.arduino_thread.data_received.connect(self.update_arduino_response)
        except serial.SerialException as e:
            print(f"Error initializing ArduinoThread: {e}")

        # Connect the btn_modulo button to send the command to Arduino
        self.btn_modulo.clicked.connect(self.send_command_to_arduino)




############################   EVENTOS DE CLICK   ############################
```

```
# CLICK PARA ARRASTRAR VENTANA

    def moveWindow(e):
        # Primero verificamos si la ventana esta maximizada
        if self.isFullScreen() == False: #Not maximized
            #Solo movemos si la ventana no esta agrandada
            #Arrastramos la ventana cuando se hace click izquierdo con el mouse
            if e.buttons() == Qt.LeftButton:
                #Movemos ventana
                self.move(self.pos() + e.globalPos() - self.clickPosition)
                self.clickPosition = e.globalPos()
                e.accept()

#########################################################################

    self.header.mouseMoveEvent = moveWindow
    self.footer.mouseMoveEvent = moveWindow

# CLICK PARA AGRANDAR / CERRAR / MINIMIZAR VENTANA


    #Minimize Window
    self.minimize_window_button.clicked.connect(lambda: self.showMinimized())
    self.minimize_window_button.enterEvent = self.minimize_window_button_enter_event #Cambio
Color del boton
    self.minimize_window_button.leaveEvent = self.minimize_window_button_leave_event #Cambio
Color del boton

    #Close window
    self.close_window_button.clicked.connect(lambda: self.close())
    self.close_window_button.enterEvent = self.close_window_button_enter_event #Cambio Color del
boton
    self.close_window_button.leaveEvent = self.close_window_button_leave_event #Cambio Color del
boton

    #Expand / Restore Window
    self.max_button.clicked.connect(lambda: self.showFullScreen())
    self.max_button.enterEvent = self.max_button_enter_event #Cambio Color del boton
    self.max_button.leaveEvent = self.max_button_leave_event #Cambio Color del boton

    #Achicar la interfaz
    self.min_button.clicked.connect(lambda: self.showNormal())
    self.min_button.enterEvent = self.min_button_enter_event #Cambio Color del boton
    self.min_button.leaveEvent = self.min_button_leave_event #Cambio Color del boton


# CLICK PARA BOTONES DE MENÚ

    #Boton Home
    self.btn_home.clicked.connect(self.distribuir_home)
```

```
        self.btn_home.enterEvent = self.btn_home_enter_event #Cambio Color del boton
        self.btn_home.leaveEvent = self.btn_home_leave_event #Cambio Color del boton

        #Boton Module
        self.btn_modulo.clicked.connect(self.distribuir_modulo)
        self.btn_modulo.enterEvent = self.btn_modulo_enter_event #Cambio Color del boton
        self.btn_modulo.leaveEvent = self.btn_modulo_leave_event #Cambio Color del boton

        #Boton Battey
        self.btn_bateria.clicked.connect(self.distribuir_bateria)
        self.btn_bateria.enterEvent = self.btn_bateria_enter_event #Cambio Color del boton
        self.btn_bateria.leaveEvent = self.btn_bateria_leave_event #Cambio Color del boton

        #Boton Camera
        self.btn_camara.clicked.connect(self.distribuir_camara)
        self.btn_camara.enterEvent = self.btn_camara_enter_event #Cambio Color del boton
        self.btn_camara.leaveEvent = self.btn_camara_leave_event #Cambio Color del boton

        #Boton Controls
        self.btn_controles.clicked.connect(self.distribuir_controles)
        self.btn_controles.enterEvent = self.btn_controles_enter_event #Cambio Color del boton
        self.btn_controles.leaveEvent = self.btn_controles_leave_event #Cambio Color del boton

        #Boton Bluetooth
        self.btn_bluetooth.clicked.connect(self.distribuir_bluetooth)
        self.btn_bluetooth.enterEvent = self.btn_bluetooth_enter_event #Cambio Color del boton
        self.btn_bluetooth.leaveEvent = self.btn_bluetooth_leave_event #Cambio Color del boton

    # CLICK PARA CONTROLAR ROBOT
        self.btn_north.enterEvent = self.btn_north_enter_event #Cambio Color del boton
        self.btn_north.leaveEvent = self.btn_north_leave_event #Cambio Color del boton
        self.btn_north.clicked.connect(self.north_command)

        self.btn_south.enterEvent = self.btn_south_enter_event #Cambio Color del boton
        self.btn_south.leaveEvent = self.btn_south_leave_event #Cambio Color del boton
        self.btn_south.clicked.connect(self.south_command)


        self.btn_east.enterEvent = self.btn_east_enter_event #Cambio Color del boton
        self.btn_east.leaveEvent = self.btn_east_leave_event #Cambio Color del boton
        self.btn_east.clicked.connect(self.east_command)

        self.btn_west.enterEvent = self.btn_west_enter_event #Cambio Color del boton
        self.btn_west.leaveEvent = self.btn_west_leave_event #Cambio Color del boton
        self.btn_west.clicked.connect(self.west_command)

        self.btn_NE.enterEvent = self.btn_NE_enter_event #Cambio Color del boton
        self.btn_NE.leaveEvent = self.btn_NE_leave_event #Cambio Color del boton
        self.btn_NE.clicked.connect(self.NE_command)
```

```
        self.btn_SE.enterEvent = self.btn_SE_enter_event #Cambio Color del boton
        self.btn_SE.leaveEvent = self.btn_SE_leave_event #Cambio Color del boton
        self.btn_SE.clicked.connect(self.SE_command)

        self.btn_SW.enterEvent = self.btn_SW_enter_event #Cambio Color del boton
        self.btn_SW.leaveEvent = self.btn_SW_leave_event #Cambio Color del boton
        self.btn_SW.clicked.connect(self.SW_command)


        self.btn_NW.enterEvent = self.btn_NW_enter_event #Cambio Color del boton
        self.btn_NW.leaveEvent = self.btn_NW_leave_event #Cambio Color del boton
        self.btn_NW.clicked.connect(self.NW_command)


        self.btn_counter_clock.enterEvent = self.btn_counter_clock_enter_event #Cambio Color del boton
        self.btn_counter_clock.leaveEvent = self.btn_counter_clock_leave_event #Cambio Color del boton
        self.btn_counter_clock.clicked.connect(self.counter_clock_command)

        self.btn_clock.enterEvent = self.btn_clock_enter_event #Cambio Color del boton
        self.btn_clock.leaveEvent = self.btn_clock_leave_event #Cambio Color del boton
        self.btn_clock.clicked.connect(self.clock_command)


        self.btn_center.enterEvent = self.btn_center_enter_event #Cambio Color del boton
        self.btn_center.leaveEvent = self.btn_center_leave_event #Cambio Color del boton
        self.btn_center.clicked.connect(self.center_command)


# CLICK PARA CONTROLAR MODO

        self.btn_ctrl_manual.clicked.connect(self.distribuir_ctrl_manual)
        self.btn_ctrl_manual.enterEvent = self.btn_ctrl_manual_enter_event #Cambio Color del boton
        self.btn_ctrl_manual.leaveEvent = self.btn_ctrl_manual_leave_event #Cambio Color del boton

        self.btn_ctrl_autonomo.clicked.connect(self.distribuir_ctrl_autonomo)
        self.btn_ctrl_autonomo.enterEvent = self.btn_ctrl_autonomo_enter_event #Cambio Color del boton
        self.btn_ctrl_autonomo.leaveEvent = self.btn_ctrl_autonomo_leave_event #Cambio Color del boton

        self.btn_ctrl_obs.clicked.connect(self.distribuir_ctrl_obs)
        self.btn_ctrl_obs.enterEvent = self.btn_ctrl_obs_enter_event #Cambio Color del boton
        self.btn_ctrl_obs.leaveEvent = self.btn_ctrl_obs_leave_event #Cambio Color del boton

        self.btn_start_route.clicked.connect(self.start_route)
        self.btn_start_route.enterEvent = self.btn_start_route_enter_event #Cambio Color del boton
        self.btn_start_route.leaveEvent = self.btn_start_route_leave_event #Cambio Color del boton

        self.btn_esquivar.enterEvent = self.btn_esquivar_enter_event #Cambio Color del boton
        self.btn_esquivar.leaveEvent = self.btn_esquivar_leave_event #Cambio Color del boton

        self.btn_marcar.enterEvent = self.btn_marcar_enter_event #Cambio Color del boton
```

```
        self.btn_marcar.leaveEvent = self.btn_marcar_leave_event #Cambio Color del boton

        self.btn_stop_movement.enterEvent = self.btn_stop_movement_enter_event #Cambio Color del boton
        self.btn_stop_movement.leaveEvent = self.btn_stop_movement_leave_event #Cambio Color del
boton


# CLICK PARA CONTROLAR CAMARA

        self.btn_grabar.enterEvent = self.btn_grabar_enter_event #Cambio Color del boton
        self.btn_grabar.leaveEvent = self.btn_grabar_leave_event #Cambio Color del boton


# CLICK PARA VER CONEXION MODULO

        self.btn_im_ref.enterEvent = self.btn_im_ref_enter_event #Cambio Color del boton
        self.btn_im_ref.leaveEvent = self.btn_im_ref_leave_event #Cambio Color del boton
        self.btn_im_ref.clicked.connect(self.recargar_imagen)



# CLICK PARA DESPLEGAR MENU

        self.menu_button.clicked.connect(lambda: self.slideLeftMenu())


##############   FUNCIONES PARA EVENTOS DE CLICK   ####################



# Conectar posición de mouse
    def mousePressEvent(self, event):
            # Get the current position of the mouse
            self.clickPosition = event.globalPos()

######################################################## MENU DESLIZANTE

    def slideLeftMenu(self):
        #Obtenemos la medida del ancho del menu
        width = self.slide_menu_container.width()

        # Si esta minimized
        if width == 0:
            # Expand menu
            newWidth = 300

            self.menu_button.setIcon(QtGui.QIcon(u"Iconos/left.png"))


        # Si el menu esta abierto
        else:
```

```
            # Restore menu
            newWidth = 0

            self.menu_button.setIcon(QtGui.QIcon(u"Iconos/menu.png"))


        # Transision de animacion
        self.animation = QPropertyAnimation(self.slide_menu_container, b"maximumWidth")#Animate
minimumWidht
        self.animation.setDuration(250)
        self.animation.setStartValue(width)#Start value is the current menu width
        self.animation.setEndValue(newWidth)#end value is the new menu width
        self.animation.setEasingCurve(QtCore.QEasingCurve.InOutQuart)
        self.animation.start()

########################## DISTRIUIR ESPACIOS

    def distribuir_home (self):

        self.lower_main_body.setMaximumSize(10000,0)
        self.upper_main_body.setMaximumSize(10000,10000)

        self.battery.setMaximumSize(0,10000)
        self.home.setMaximumSize(10000,10000)
        self.module.setMaximumSize(0,10000)


    def distribuir_modulo (self):

        self.lower_main_body.setMaximumSize(10000,0)
        self.upper_main_body.setMaximumSize(10000,10000)

        self.battery.setMaximumSize(0,10000)
        self.home.setMaximumSize(0,10000)
        self.module.setMaximumSize(10000,10000)

    def distribuir_bateria (self):

        self.lower_main_body.setMaximumSize(10000,0)
        self.upper_main_body.setMaximumSize(10000,10000)

        self.battery.setMaximumSize(10000,10000)
        self.home.setMaximumSize(0,10000)
        self.module.setMaximumSize(0,10000)

    def distribuir_camara (self):

        self.upper_main_body.setMaximumSize(10000,0)
        self.lower_main_body.setMaximumSize(10000,10000)
```

```
    self.bluetooth.setMaximumSize(0,10000)
    self.camera.setMaximumSize(10000,10000)
    self.controls.setMaximumSize(0,10000)


def distribuir_controles (self):

    self.upper_main_body.setMaximumSize(10000,0)
    self.lower_main_body.setMaximumSize(10000,10000)

    self.bluetooth.setMaximumSize(0,10000)
    self.camera.setMaximumSize(10000,10000)
    self.controls.setMaximumSize(600,10000)

    self.frame_opciones_manejo.setMaximumSize(0,0)

    self.btn_ctrl_autonomo.setEnabled(True)
    self.btn_ctrl_obs.setEnabled(True)
    self.btn_ctrl_manual.setEnabled(True)
    self.btn_start_route.setEnabled(False)

def distribuir_bluetooth (self):

    self.upper_main_body.setMaximumSize(10000,0)
    self.lower_main_body.setMaximumSize(10000,10000)

    self.bluetooth.setMaximumSize(10000,10000)
    self.camera.setMaximumSize(0,10000)

def distribuir_ctrl_manual (self):

    self.frame_opciones_manejo.setMaximumSize(0,0)

    self.btn_ctrl_autonomo.setEnabled(False)
    self.btn_ctrl_obs.setEnabled(False)
    self.btn_ctrl_manual.setEnabled(True)
    self.btn_start_route.setEnabled(True)


def distribuir_ctrl_autonomo (self):

    self.frame_opciones_manejo.setMaximumSize(10000,10000)
    self.frame_obstaculo.setMaximumSize(0,0)
    self.frame_auto.setMaximumSize(100000,10000)

    self.btn_ctrl_autonomo.setEnabled(True)
    self.btn_ctrl_obs.setEnabled(False)
    self.btn_ctrl_manual.setEnabled(False)
    self.btn_start_route.setEnabled(True)
```

```
    def distribuir_ctrl_obs (self):

        self.frame_opciones_manejo.setMaximumSize(10000,10000)
        self.frame_obstaculo.setMaximumSize(100000,10000)
        self.frame_auto.setMaximumSize(100000,10000)

        self.btn_ctrl_autonomo.setEnabled(False)
        self.btn_ctrl_obs.setEnabled(True)
        self.btn_ctrl_manual.setEnabled(False)
        self.btn_start_route.setEnabled(True)


################################ CONTROL

    def start_route (self):
        global flag_ruta

        if flag_ruta == 0:

            self.btn_start_route.setIcon(QtGui.QIcon(u"Iconos/icons8-stop-48.png"))
            self.btn_start_route.setText("Detener")
            flag_ruta = 1

        elif flag_ruta == 1:
            self.btn_start_route.setIcon(QtGui.QIcon(u"Iconos/bandera2.png"))
            self.btn_start_route.setText("Iniciar")
            self.btn_ctrl_autonomo.setEnabled(True)
            self.btn_ctrl_obs.setEnabled(True)
            self.btn_ctrl_manual.setEnabled(True)
            self.btn_start_route.setEnabled(False)
            self.frame_opciones_manejo.setMaximumSize(0,0)
            flag_ruta = 0


#################### CAMARA

    def send_command_to_arduino(self):
        if self.arduino_thread.is_running:
            self.arduino_thread.start()
            command = '8'
            self.arduino_thread.send_data(command)

    def update_arduino_response(self, data):

        global slave1, slave2, slave3, mode

        # Print the response received from the Arduino
        print(f"Received from Arduino: {data}")

        if data == 'Slave 1 connected':
```

```
            self.btn_home_3.setStyleSheet("QPushButton {border: none; background-color: rgb(27, 250, 200);
color: rgb(255, 255, 255) ; border-radius: 10px;}")
            slave1 = 1
        elif data == 'Slave 2 connected':
            self.btn_home_4.setStyleSheet("QPushButton {border: none; background-color: rgb(27, 250, 200);
color: rgb(255, 255, 255) ; border-radius: 10px}")
            slave2 = 1
        elif data == 'Slave 3 connected':
            self.btn_home_5.setStyleSheet("QPushButton {border: none; background-color: rgb(27, 250,
200);  color: rgb(255, 255, 255) ; border-radius: 10px}")
            slave3 = 1

        elif data == 'Slave 1 disconnected':
            self.btn_home_3.setStyleSheet("QPushButton {border: none; border-radius: 10px; background-
color: rgb(6, 44, 44);  color: rgb(255, 255, 255); }")
            slave1 = 0
        elif data == 'Slave 2 disconnected':
            self.btn_home_4.setStyleSheet("QPushButton {border: none; border-radius: 10px; background-
color: rgb(6, 44, 44);  color: rgb(255, 255, 255); }")
            slave2 = 0
        elif data == 'Slave 3 disconnected':
            self.btn_home_5.setStyleSheet("QPushButton {border: none; border-radius: 10px; background-
color: rgb(6, 44, 44); color: rgb(255, 255, 255);  }")
            slave3 = 0


        if (slave1 == 0 and slave2 == 0 and slave3 == 0): # Basic Mode
            mode = 0

        elif (slave1 == 1 and slave2 == 0 and slave3 == 0): # Control Mode
            mode = 1

        elif (slave1 == 1 and slave2 == 1 and slave3 == 0): # Control and Sensors Mode
            mode = 2

        elif (slave1 == 1 and slave2 == 0 and slave3 == 1): # Control and Pusher Mode
            mode = 3

        elif (slave1 == 1 and slave2 == 1 and slave3 == 1): # Control , Sensors and Pusher Mode
            mode = 4

        elif (slave1 == 0 and slave2 == 1 and slave3 == 1): # Sensors and Pusher Mode, this would never
happen.
            mode = 5

    def recargar_imagen(self):

        if mode == 0:
            self.pushButton_3.setIcon(QtGui.QIcon(u"Imagenes/Modulo Movil Con Tapa.png"))
```

```
        elif mode == 1:
            self.pushButton_3.setIcon(QtGui.QIcon(u"Imagenes/Modulo de Control Vista 1.png"))

        elif mode == 2:
            self.pushButton_3.setIcon(QtGui.QIcon(u"Imagenes/Modulo de sensores tapa.png"))

        elif mode == 3:
            self.pushButton_3.setIcon(QtGui.QIcon(u"Imagenes/Módulo de Actuadores SIn Sensores.png"))

        elif mode == 4:
            self.pushButton_3.setIcon(QtGui.QIcon(u"Imagenes/Robot Final.png"))

        elif mode == 5:
            self.pushButton_3.setIcon(QtGui.QIcon(u"Imagenes/Modulo Movil Con Tapa.png"))

    def toggle_camera(self):
        if not self.camera_running:
            # Start the camera feed
            self.camera_thread.is_running = True
            self.camera_thread.start()
            self.camera_running = True
            self.btn_grabar.setText("Stop Camera")
            self.btn_grabar.setIcon(QtGui.QIcon(u"Iconos/icons8-stop blue-48.png"))
        else:
            # Stop the camera feed
            self.camera_thread.stop()
            self.camera_thread.wait()
            self.camera_running = False
            self.btn_grabar.setText("Start Camera")
            self.btn_grabar.setIcon(QtGui.QIcon(u"Iconos/icons8-record-48.png"))

    def update_image(self, image_bytes):
        pixmap = QtGui.QPixmap()
        pixmap.loadFromData(image_bytes)
        self.camera_image.setPixmap(pixmap)


##################################### CAMBIAR DE COLOR A LOS BOTONES

########### Botones de Max Min
    def minimize_window_button_enter_event (self, event):
        # Change the button color when hovered
        self.minimize_window_button.setStyleSheet("QPushButton {border: none; background-color: rgb(27,
250, 200);border-radius: 10px; color: rgb(255, 255, 255) }")
    def minimize_window_button_leave_event (self, event):
        # Change the button color when hovered
        self.minimize_window_button.setStyleSheet("QPushButton {border: none; background-color:
transparent;border-radius: 10px; color: rgb(255, 255, 255) }")

    def close_window_button_enter_event (self, event):
```

```
        # Change the button color when hovered
        self.close_window_button.setStyleSheet("QPushButton {border: none; background-color: rgb(27,
250, 200);border-radius: 10px; color: rgb(255, 255, 255) }")
    def close_window_button_leave_event (self, event):
        # Change the button color when hovered
        self.close_window_button.setStyleSheet("QPushButton {border: none; background-color:
transparent;border-radius: 10px; color: rgb(255, 255, 255) }")

    def min_button_enter_event (self, event):
        # Change the button color when hovered
        self.min_button.setStyleSheet("QPushButton {border: none; background-color: rgb(27, 250,
200);border-radius: 10px; color: rgb(255, 255, 255) }")
    def min_button_leave_event (self, event):
        # Change the button color when hovered
        self.min_button.setStyleSheet("QPushButton {border: none; background-color: transparent;border-
radius: 10px; color: rgb(255, 255, 255) }")

    def max_button_enter_event (self, event):
        # Change the button color when hovered
        self.max_button.setStyleSheet("QPushButton {border: none; background-color: rgb(27, 250,
200);border-radius: 10px; color: rgb(255, 255, 255) }")
    def max_button_leave_event (self, event):
        # Change the button color when hovered
        self.max_button.setStyleSheet("QPushButton {border: none; background-color: transparent;border-
radius: 10px; color: rgb(255, 255, 255) }")

########### Botones de Menu

    def btn_home_enter_event (self, event):
        # Change the button color when hovered
        self.btn_home.setStyleSheet("QPushButton {border: none; background-color: rgb(27, 250,
200);border-radius: 10px; color: rgb(255, 255, 255) }")
    def btn_home_leave_event (self, event):
        # Change the button color when hovered
        self.btn_home.setStyleSheet("QPushButton {border: none; background-color: rgb(27, 219,
159);border-radius: 10px; color: rgb(255, 255, 255) }")

    def btn_modulo_enter_event (self, event):
        # Change the button color when hovered
        self.btn_modulo.setStyleSheet("QPushButton {border: none; background-color: rgb(27, 250,
200);border-radius: 10px; color: rgb(255, 255, 255) }")
    def btn_modulo_leave_event (self, event):
        # Change the button color when hovered
        self.btn_modulo.setStyleSheet("QPushButton {border: none; background-color: rgb(27, 219,
159);border-radius: 10px; color: rgb(255, 255, 255) }")

    def btn_bateria_enter_event (self, event):
        # Change the button color when hovered
        self.btn_bateria.setStyleSheet("QPushButton {border: none; background-color: rgb(27, 250,
200);border-radius: 10px; color: rgb(255, 255, 255) }")
```

```
    def btn_bateria_leave_event (self, event):
        # Change the button color when hovered
        self.btn_bateria.setStyleSheet("QPushButton {border: none; background-color: rgb(27, 219,
159);border-radius: 10px; color: rgb(255, 255, 255) }")

    def btn_camara_enter_event (self, event):
        # Change the button color when hovered
        self.btn_camara.setStyleSheet("QPushButton {border: none; background-color: rgb(27, 250,
200);border-radius: 10px; color: rgb(255, 255, 255) }")
    def btn_camara_leave_event (self, event):
        # Change the button color when hovered
        self.btn_camara.setStyleSheet("QPushButton {border: none; background-color: rgb(27, 219,
159);border-radius: 10px; color: rgb(255, 255, 255) }")

    def btn_controles_enter_event (self, event):
        # Change the button color when hovered
        self.btn_controles.setStyleSheet("QPushButton {border: none; background-color: rgb(27, 250,
200);border-radius: 10px; color: rgb(255, 255, 255) }")
    def btn_controles_leave_event (self, event):
        # Change the button color when hovered
        self.btn_controles.setStyleSheet("QPushButton {border: none; background-color: rgb(27, 219,
159);border-radius: 10px; color: rgb(255, 255, 255) }")

    def btn_bluetooth_enter_event (self, event):
        # Change the button color when hovered
        self.btn_bluetooth.setStyleSheet("QPushButton {border: none; background-color: rgb(27, 250,
200);border-radius: 10px; color: rgb(255, 255, 255) }")
    def btn_bluetooth_leave_event (self, event):
        # Change the button color when hovered
        self.btn_bluetooth.setStyleSheet("QPushButton {border: none; background-color: rgb(27, 219,
159);border-radius: 10px; color: rgb(255, 255, 255) }")

########### Botones de Controles

    def btn_north_enter_event (self, event):
        # Change the button color when hovered
        self.btn_north.setStyleSheet("QPushButton {border: none; background-color: rgb(27, 250,
200);border-radius: 10px; color: rgb(255, 255, 255) }")
    def btn_north_leave_event (self, event):
        # Change the button color when hovered
        self.btn_north.setStyleSheet("QPushButton {border: none; background-color: rgb(6, 44, 44);border-
radius: 10px; color: rgb(255, 255, 255) }")
    def north_command(self):
        # Get the current position of the mouse
        self.btn_comando.setText("Comando: W")
    def btn_east_enter_event (self, event):
        # Change the button color when hovered
        self.btn_east.setStyleSheet("QPushButton {border: none; background-color: rgb(27, 250, 200);border-
radius: 10px; color: rgb(255, 255, 255) }")
    def btn_east_leave_event (self, event):
```

```
        # Change the button color when hovered
        self.btn_east.setStyleSheet("QPushButton {border: none; background-color: rgb(6, 44, 44);border-radius: 10px; color: rgb(255, 255, 255) }")
    def east_command(self):
        # Get the current position of the mouse
        self.btn_comando.setText("Comando: D")
    def btn_south_enter_event (self, event):
        # Change the button color when hovered
        self.btn_south.setStyleSheet("QPushButton {border: none; background-color: rgb(27, 250, 200);border-radius: 10px; color: rgb(255, 255, 255) }")
    def btn_south_leave_event (self, event):
        # Change the button color when hovered
        self.btn_south.setStyleSheet("QPushButton {border: none; background-color: rgb(6, 44, 44);border-radius: 10px; color: rgb(255, 255, 255) }")
    def south_command(self):
        # Get the current position of the mouse
        self.btn_comando.setText("Comando: S")
    def btn_west_enter_event (self, event):
        # Change the button color when hovered
        self.btn_west.setStyleSheet("QPushButton {border: none; background-color: rgb(27, 250, 200);border-radius: 10px; color: rgb(255, 255, 255) }")
    def btn_west_leave_event (self, event):
        # Change the button color when hovered
        self.btn_west.setStyleSheet("QPushButton {border: none; background-color: rgb(6, 44, 44);border-radius: 10px; color: rgb(255, 255, 255) }")
    def west_command(self):
        # Get the current position of the mouse
        self.btn_comando.setText("Comando: A")


    def btn_NW_enter_event (self, event):
        # Change the button color when hovered
        self.btn_NW.setStyleSheet("QPushButton {border: none; background-color: rgb(27, 250, 200);border-radius: 10px; color: rgb(255, 255, 255) }")
    def btn_NW_leave_event (self, event):
        # Change the button color when hovered
        self.btn_NW.setStyleSheet("QPushButton {border: none; background-color: rgb(6, 44, 44);border-radius: 10px; color: rgb(255, 255, 255) }")
    def NW_command(self):
        # Get the current position of the mouse
        self.btn_comando.setText("Comando: WA")
    def btn_SW_enter_event (self, event):
        # Change the button color when hovered
        self.btn_SW.setStyleSheet("QPushButton {border: none; background-color: rgb(27, 250, 200);border-radius: 10px; color: rgb(255, 255, 255) }")
    def btn_SW_leave_event (self, event):
        # Change the button color when hovered
        self.btn_SW.setStyleSheet("QPushButton {border: none; background-color: rgb(6, 44, 44);border-radius: 10px; color: rgb(255, 255, 255) }")
    def SW_command(self):
```

```
        # Get the current position of the mouse
        self.btn_comando.setText("Comando: SA")
    def btn_NE_enter_event (self, event):
        # Change the button color when hovered
        self.btn_NE.setStyleSheet("QPushButton {border: none; background-color: rgb(27, 250, 200);border-
radius: 10px; color: rgb(255, 255, 255) }")
    def btn_NE_leave_event (self, event):
        # Change the button color when hovered
        self.btn_NE.setStyleSheet("QPushButton {border: none; background-color: rgb(6, 44, 44);border-
radius: 10px; color: rgb(255, 255, 255) }")
    def NE_command(self):
        # Get the current position of the mouse
        self.btn_comando.setText("Comando: WD")
    def btn_SE_enter_event (self, event):
        # Change the button color when hovered
        self.btn_SE.setStyleSheet("QPushButton {border: none; background-color: rgb(27, 250, 200);border-
radius: 10px; color: rgb(255, 255, 255) }")
    def btn_SE_leave_event (self, event):
        # Change the button color when hovered
        self.btn_SE.setStyleSheet("QPushButton {border: none; background-color: rgb(6, 44, 44);border-
radius: 10px; color: rgb(255, 255, 255) }")
    def SE_command(self):
        # Get the current position of the mouse
        self.btn_comando.setText("Comando: SD")


    def btn_counter_clock_enter_event (self, event):
        # Change the button color when hovered
        self.btn_counter_clock.setStyleSheet("QPushButton {border: none; background-color: rgb(27, 250,
200);border-radius: 10px; color: rgb(255, 255, 255) }")
    def btn_counter_clock_leave_event (self, event):
        # Change the button color when hovered
        self.btn_counter_clock.setStyleSheet("QPushButton {border: none; background-color: rgb(6, 44,
44);border-radius: 10px; color: rgb(255, 255, 255) }")
    def counter_clock_command(self):
        # Get the current position of the mouse
        self.btn_comando.setText("Comando: Q")
    def btn_clock_enter_event (self, event):
        # Change the button color when hovered
        self.btn_clock.setStyleSheet("QPushButton {border: none; background-color: rgb(27, 250,
200);border-radius: 10px; color: rgb(255, 255, 255) }")
    def btn_clock_leave_event (self, event):
        # Change the button color when hovered
        self.btn_clock.setStyleSheet("QPushButton {border: none; background-color: rgb(6, 44, 44);border-
radius: 10px; color: rgb(255, 255, 255) }")
    def clock_command(self):
        # Get the current position of the mouse
        self.btn_comando.setText("Comando: R")
    def btn_center_enter_event (self, event):
        # Change the button color when hovered
```

```
        self.btn_center.setStyleSheet("QPushButton {border: none; background-color: rgb(27, 250,
200);border-radius: 10px; color: rgb(255, 255, 255) }")
    def btn_center_leave_event (self, event):
        # Change the button color when hovered
        self.btn_center.setStyleSheet("QPushButton {border: none; background-color: rgb(6, 44, 44);border-
radius: 10px; color: rgb(255, 255, 255) }")
    def center_command(self):
        # Get the current position of the mouse
        self.btn_comando.setText("Comando: X")


########### Botones de Modalidades

    def btn_ctrl_manual_enter_event (self, event):
        # Change the button color when hovered
        self.btn_ctrl_manual.setStyleSheet("QPushButton {border: none; background-color: rgb(27, 250,
200);border-radius: 10px; color: rgb(255, 255, 255) }")
    def btn_ctrl_manual_leave_event (self, event):
        # Change the button color when hovered
        self.btn_ctrl_manual.setStyleSheet("QPushButton {border: none; background-color: rgb(6, 44,
44);border-radius: 10px; color: rgb(255, 255, 255) }")

    def btn_ctrl_autonomo_enter_event (self, event):
        # Change the button color when hovered
        self.btn_ctrl_autonomo.setStyleSheet("QPushButton {border: none; background-color: rgb(27, 250,
200);border-radius: 10px; color: rgb(255, 255, 255) }")
    def btn_ctrl_autonomo_leave_event (self, event):
        # Change the button color when hovered
        self.btn_ctrl_autonomo.setStyleSheet("QPushButton {border: none; background-color: rgb(6, 44,
44);border-radius: 10px; color: rgb(255, 255, 255) }")

    def btn_ctrl_obs_enter_event (self, event):
        # Change the button color when hovered
        self.btn_ctrl_obs.setStyleSheet("QPushButton {border: none; background-color: rgb(27, 250,
200);border-radius: 10px; color: rgb(255, 255, 255) }")
    def btn_ctrl_obs_leave_event (self, event):
        # Change the button color when hovered
        self.btn_ctrl_obs.setStyleSheet("QPushButton {border: none; background-color: rgb(6, 44,
44);border-radius: 10px; color: rgb(255, 255, 255) }")

    def btn_start_route_enter_event (self, event):
        # Change the button color when hovered
        self.btn_start_route.setStyleSheet("QPushButton {border: none; background-color: rgb(27, 250,
200);border-radius: 10px; color: rgb(255, 255, 255) }")
    def btn_start_route_leave_event (self, event):
        # Change the button color when hovered
        self.btn_start_route.setStyleSheet("QPushButton {border: none; background-color: rgb(6, 44,
44);border-radius: 10px; color: rgb(255, 255, 255) }")

    def btn_esquivar_enter_event (self, event):
```

```
        # Change the button color when hovered
        self.btn_esquivar.setStyleSheet("QPushButton {border: none; background-color: rgb(27, 250,
200);border-radius: 10px; color: rgb(255, 255, 255) }")
    def btn_esquivar_leave_event (self, event):
        # Change the button color when hovered
        self.btn_esquivar.setStyleSheet("QPushButton {border: none; background-color: rgb(6, 44,
44);border-radius: 10px; color: rgb(255, 255, 255) }")

    def btn_marcar_enter_event (self, event):
        # Change the button color when hovered
        self.btn_marcar.setStyleSheet("QPushButton {border: none; background-color: rgb(27, 250,
200);border-radius: 10px; color: rgb(255, 255, 255) }")
    def btn_marcar_leave_event (self, event):
        # Change the button color when hovered
        self.btn_marcar.setStyleSheet("QPushButton {border: none; background-color: rgb(6, 44, 44);border-
radius: 10px; color: rgb(255, 255, 255) }")

    def btn_stop_movement_enter_event (self, event):
        # Change the button color when hovered
        self.btn_stop_movement.setStyleSheet("QPushButton {border: none; background-color: rgb(27, 250,
200);border-radius: 10px; color: rgb(255, 255, 255) }")
    def btn_stop_movement_leave_event (self, event):
        # Change the button color when hovered
        self.btn_stop_movement.setStyleSheet("QPushButton {border: none; background-color: rgb(6, 44,
44);border-radius: 10px; color: rgb(255, 255, 255) }")

########### Botones de Camara

    def btn_grabar_enter_event (self, event):
        # Change the button color when hovered
        self.btn_grabar.setStyleSheet("QPushButton {border: none; background-color: rgb(27, 250,
200);border-radius: 10px; color: rgb(255, 255, 255) }")
    def btn_grabar_leave_event (self, event):
        # Change the button color when hovered
        self.btn_grabar.setStyleSheet("QPushButton {border: none; background-color: rgb(6, 44, 44);border-
radius: 10px; color: rgb(255, 255, 255) }")


########### Botones refresco imagen

    def btn_im_ref_enter_event (self, event):
        # Change the button color when hovered
        self.btn_im_ref.setStyleSheet("QPushButton {border: none; background-color: rgb(27, 250,
200);border-radius: 10px; color: rgb(255, 255, 255) }")
    def btn_im_ref_leave_event (self, event):
        # Change the button color when hovered
        self.btn_im_ref.setStyleSheet("QPushButton {border: none; background-color: rgb(6, 44, 44);border-
radius: 10px; color: rgb(255, 255, 255) }")
```

```
if __name__ == "__main__":
    app = QtWidgets.QApplication(sys.argv)
    ex = QtWidgets.QFileDialog()
    mi_app = MainPage()
    mi_app.show()
    sys.exit(app.exec_())
```

Anexo 2

Código para configuración de comunicación I2C entre Raspberry pi 3 y 2 Arduinos

En Raspberry Pi 3:

```
from smbus import SMBus

addr1= 0x8 #bus address

addr2= 0x10 #bus address

bus = SMBus(1)

numb = 1

print ("Enter command:")
while numb==1:

        ledstate = input(">>>>   ")

        if ledstate == "1": #Encender LED Arduino 1
                bus.write_byte(addr1, 0x1) #switch on
                bus.write_byte(addr2, 0x1) #switch on

        elif ledstate == "0": # Apagar LED Arduino 1
                bus.write_byte(addr1, 0x0) #switch off
                bus.write_byte(addr2, 0x0) #switch off

        elif ledstate == "3": # Encender LED Arduino 1
                bus.write_byte(addr1, 0x1) #switch off
                bus.write_byte(addr2, 0x1) #switch off
```

```
        elif ledstate == "2": # Apagar LED Arduino 1
                bus.write_byte(addr1, 0x0) #switch off
                bus.write_byte(addr2, 0x0) #switch off

        else:
                numb=0
```

En Arduino Uno 1:

```
# include <Wire.h>
const int ledPin = 13;

void setup() {

  Wire.begin(0x8);

  Wire.onReceive(receiveEvent);

  pinMode(ledPin, OUTPUT);
  digitalWrite(ledPin,LOW);

}
void receiveEvent(int howMany) {


  while (Wire.available()){
    char c = Wire.read();
    digitalWrite(ledPin, c);
  }
}
void loop(){
  delay(100);
}
```

En Arduino Uno 2:

```
# include <Wire.h>

const int ledPin = 13;

void setup() {

  Wire.begin(0x10);

  Wire.onReceive(receiveEvent);

  pinMode(ledPin, OUTPUT);
```

```
  digitalWrite(ledPin,LOW);

}

void receiveEvent(int howMany) {


  while (Wire.available()){
    char c = Wire.read();
    digitalWrite(ledPin, c);
  }
}
void loop(){
  delay(100);
}
```

Anexo 3

Código para configuración de motores para movilidad omnidireccional simple

En Raspberry Pi 3:

```
from smbus import SMBus

addr1= 0x8 #bus address para Arduino 1

addr2= 0x10 #bus address para Arduino 2

bus = SMBus(1)

numb = 1

print ("Enter command:")
while numb==1:

        ledstate = input(">>>>   ")

        if ledstate == "w": #AVANZAR
                bus.write_byte(addr1, 0x1) #switch on
                bus.write_byte(addr2, 0x1) #switch on

        elif ledstate == "s": #RETROCEDER
                bus.write_byte(addr1, 0x0) #switch off
                bus.write_byte(addr2, 0x0) #switch off

        elif ledstate == "a": #IZQUIERDA
                bus.write_byte(addr1, 0x2) #switch off
                bus.write_byte(addr2, 0x2) #switch off

        elif ledstate == "d": #DERECHA
                bus.write_byte(addr1, 0x3) #switch off
                bus.write_byte(addr2, 0x3) #switch off


        elif ledstate == "x": #DETENER
                bus.write_byte(addr1, 0x4) #switch off
                bus.write_byte(addr2, 0x4) #switch off

        elif ledstate == "e": #NE
                bus.write_byte(addr1, 0x5) #switch off
```

```
        bus.write_byte(addr2, 0x5) #switch off
elif ledstate == "q": #NW
        bus.write_byte(addr1, 0x6) #switch off
        bus.write_byte(addr2, 0x6) #switch off

elif ledstate == "c": #SE
        bus.write_byte(addr1, 0x7) #switch off
        bus.write_byte(addr2, 0x7) #switch off

elif ledstate == "z": #SW
        bus.write_byte(addr1, 0x8) #switch off
        bus.write_byte(addr2, 0x8) #switch off

elif ledstate == "3": #Giro Clock
        bus.write_byte(addr1, 0x9) #switch off
        bus.write_byte(addr2, 0x9) #switch off

elif ledstate == "1": #Giro anti clock
        bus.write_byte(addr1, 0x10) #switch off
        bus.write_byte(addr2, 0x10) #switch off

else:
        numb=0
```

En Arduino Uno 1:

```
#include "DualTB9051FTGMotorShield.h"
#include <Wire.h>

DualTB9051FTGMotorShield md;

//RUEDAS izquierdas

void setup()
{
 Serial.begin(115200);
 Serial.println("Dual TB9051FTG Motor Shield");
 md.init();

 Wire.begin(0x8); // 10 posi, 8 nega
 Wire.onReceive(receiveEvent);
}

void loop()
{
 // This is the empty loop function.
 // Your main code logic, if any, that needs to run continuously should go here.
}

void receiveEvent(int howMany)
{
 md.enableDrivers();
 delay(1); // wait for drivers to be enabled so fault pins are no longer low
 // md.disableDrivers(); // Is this needed? It's commented out in your original code.
 // delay(500);
  md.enableDrivers();
  delay(1);
 while (Wire.available())
 {

  char c = Wire.read();

  if (c == 0x0) //RETROCEDER
  {
   md.setM1Speed(200); // atras
   md.setM2Speed(200); // adelante
  }
  else if (c == 0x1)//AVANZAR
  {
   md.setM1Speed(-200); // atras
   md.setM2Speed(-200); // adelante
  }
  else if (c == 0x2) //IZQUIERDA
  {
   md.setM1Speed(-200); // atras quiero que vaya adelante
```

```
    md.setM2Speed(200); // adelante quiero que vaya atras
   }
   else if (c == 0x3)//DERECHA
   {
    md.setM1Speed(200); // atras quiero que vaya atras
    md.setM2Speed(-200); // adelante quiero que vaya adelante
   }

   else if (c == 0x4) //DETENER
   {
    md.setM1Speed(0); //
    md.setM2Speed(0); //
   }
   else if (c == 0x5) //Diagonal NE
   {
    md.setM1Speed(0); // atras detenida
    md.setM2Speed(-200); // adelante quiero que vaya adelante
   }

   else if (c == 0x6) //Diagonal NW
   {
    md.setM1Speed(-200); // atras quiero que vaya adelante
    md.setM2Speed(0); // adelante detenida
   }

   else if (c == 0x7) //Diagonal SE
   {
    md.setM1Speed(200); // atras quiero ir atras
    md.setM2Speed(0); // adelante detenida
   }

   else if (c == 0x8) //Diagonal SW
   {
    md.setM1Speed(0); // atras detenida
    md.setM2Speed(0); // adelante quiero ir atras
   }

   else if (c == 0x9) //Giro clock
   {
    md.setM1Speed(-200); // atras quiero ir adelante
    md.setM2Speed(-200); // adelante quiero ir adelante
   }

   else if (c == 0x10) //Giro anti clock
   {
    md.setM1Speed(200); // atras quiero ir atras
    md.setM2Speed(290); // adelante quiero ir atras
   }
  }
}
```

En Arduino Uno 2:

```
#include "DualTB9051FTGMotorShield.h"
#include <Wire.h>

DualTB9051FTGMotorShield md;

//RUEDAS Derechas

void setup()
{
  Serial.begin(115200);
  Serial.println("Dual TB9051FTG Motor Shield");
  md.init();

  Wire.begin(0x10); // 10 posi, 8 nega
  Wire.onReceive(receiveEvent);
}

void loop()
{
  // This is the empty loop function.
  // Your main code logic, if any, that needs to run continuously should go here.
}

void receiveEvent(int howMany)
{
  md.enableDrivers();
  delay(1); // wait for drivers to be enabled so fault pins are no longer low

  // md.disableDrivers(); // Is this needed? It's commented out in your original code.
  // delay(500);
  md.enableDrivers();
  delay(1);
  while (Wire.available())
  {


    char c = Wire.read();

    if (c == 0x0) //RETROCEDER
    {
      md.setM1Speed(-200); // atras
      md.setM2Speed(-200); // adelante
    }
    else if (c == 0x1)//AVANZAR
    {
      md.setM1Speed(200); // atras
      md.setM2Speed(200); // adelante
    }
    else if (c == 0x2) //IZQUIERDA
```

```
    {
      md.setM1Speed(-200); // atras quiero que vaya atras
      md.setM2Speed(200); // adelante quiero que vaya adelante
    }
    else if (c == 0x3)//DERECHA
    {
      md.setM1Speed(200); // atras quiero que vaya adelante
      md.setM2Speed(-200); // adelante querio que vaya atras
    }
    else if (c == 0x4) //DETENER
    {
      md.setM1Speed(0); //
      md.setM2Speed(0); //
    }

    else if (c == 0x5) //Diagonal NE
    {
      md.setM1Speed(200); // atras quiero que vaya adelante
      md.setM2Speed(0); // adelante detenida
    }

    else if (c == 0x6) //Diagonal NW
    {
      md.setM1Speed(0); // atras detenida
      md.setM2Speed(200); // adelante quiero que vaya adelante
    }
    else if (c == 0x7) //Diagonal SE
    {
      md.setM1Speed(0); // atras detenida
      md.setM2Speed(-200); // adelante quier ir atras
    }
    else if (c == 0x8) //Diagonal SW
    {
      md.setM1Speed(-200); // atras quiero ir atras
      md.setM2Speed(0); // adelante detenida
    }

    else if (c == 0x9) //Giro clock
    {
      md.setM1Speed(-200); // atras quiero ir atras
      md.setM2Speed(-200); // adelante quiero ir atras
    }
    else if (c == 0x10) //Giro anti clock
    {
      md.setM1Speed(200); // atras quiero ir adelante
      md.setM2Speed(200); // adelante quiero ir adelante
    }

  }
}
```

Anexo 4

Código para detección de módulos por medio de conexión I2C

En Arduino Maestro:

```
#include <Wire.h>

void setup() {
  Wire.begin();        // join i2c bus
  Serial.begin(9600);  // start serial for output
}

void loop() {
  char command;

  // Prompt user for command
  //Serial.println("Enter command: 2 to turn off, 3 to turn on (slave 1), 4 to turn off, 5 to turn on (slave 2), 6 to turn off, 7 to turn on (slave 3), 8 to get connected Arduinos:");
  while (!Serial.available()); // Wait for input
  command = Serial.read();     // Read command from serial

  // Send command to all slave Arduinos
  Wire.beginTransmission(8);    // Address of slave 1 Arduino (Nano)
  Wire.write(command);          // Send command
  Wire.endTransmission();       // Stop transmitting

  Wire.beginTransmission(9);    // Address of slave 2 Arduino (Nano)
  Wire.write(command);          // Send command
  Wire.endTransmission();       // Stop transmitting

  Wire.beginTransmission(10);   // Address of slave 3 Arduino (Uno)
  Wire.write(command);          // Send command
  Wire.endTransmission();       // Stop transmitting

  // Handle special command to get connected Arduinos
  if (command == '8') {
    delay(500); // Wait for all slaves to respond

    // Request responses from all slave Arduinos
    Serial.println("Connected Arduinos:");

    // Request response from slave 1
    Wire.requestFrom(8, 1);  // Request 1 byte from slave 1
    if (Wire.available()) {
      char response = Wire.read(); // Read response from slave 1
      Serial.print("Slave 1 connected\n");
    }
    else {
      char response = Wire.read(); // Read response from slave 1
      Serial.print("Slave 1 disconnected\n");
```

```
  }

  // Request response from slave 2
  Wire.requestFrom(9, 1);  // Request 1 byte from slave 2
  if (Wire.available()) {
    char response = Wire.read(); // Read response from slave 1
    Serial.print("Slave 2 connected\n");
  }
  else {
    char response = Wire.read(); // Read response from slave 1
    Serial.print("Slave 2 disconnected\n");
  }

  // Request response from slave 3
  Wire.requestFrom(10, 1);  // Request 1 byte from slave 3
  if (Wire.available()) {
    char response = Wire.read(); // Read response from slave 1
    Serial.print("Slave 3 connected\n");
  }
  else {
    char response = Wire.read(); // Read response from slave 1
    Serial.print("Slave 3 disconnected\n");
  }
 }

 delay(100);  // Delay for stability
}
```

En Arduino Esclavo 1:

```
#include <Wire.h>
#define LED_PIN 13

void setup() {
 Wire.begin(8);              // join i2c bus with address #8
 Wire.onReceive(receiveEvent); // register event
 pinMode(LED_PIN, OUTPUT);    // initialize the LED pin as an output
}

void loop() {
 delay(100);
}


void receiveEvent(int bytes) {
 char command = Wire.read(); // Read command from I2C bus
```

```
  // Execute command
  if (command == '1') {
    digitalWrite(LED_PIN, HIGH); // Turn on LED
  } else if (command == '0') {
    digitalWrite(LED_PIN, LOW);  // Turn off LED
  } else if (command == '8') {
    // Send response indicating connection status
    Wire.write('1');
  }
}
```

En Arduino Esclavo 2:

```
#include <Wire.h>
#define LED_PIN 13

void setup() {
  Wire.begin(9);                // join i2c bus with address #8
  Wire.onReceive(receiveEvent); // register event
  pinMode(LED_PIN, OUTPUT);     // initialize the LED pin as an output
}

void loop() {
  delay(100);
}

void receiveEvent(int bytes) {
  char command = Wire.read(); // Read command from I2C bus

  // Execute command
  if (command == '3') {
    digitalWrite(LED_PIN, HIGH); // Turn on LED
  } else if (command == '2') {
    digitalWrite(LED_PIN, LOW);  // Turn off LED
  } else if (command == '8') {
    // Send response indicating connection status
    Wire.write('1');
  }
}
```

En Arduino Esclavo 3:

```
#include <Wire.h>
#define LED_PIN 13

void setup() {
  Wire.begin(10);                // join i2c bus with address #8
  Wire.onReceive(receiveEvent); // register event
  pinMode(LED_PIN, OUTPUT);     // initialize the LED pin as an output
}

void loop() {
  delay(100);
}

void receiveEvent(int bytes) {
  char command = Wire.read(); // Read command from I2C bus

  // Execute command
  if (command == '5') {
    digitalWrite(LED_PIN, HIGH); // Turn on LED
  } else if (command == '4') {
    digitalWrite(LED_PIN, LOW);  // Turn off LED
  } else if (command == '8') {
    // Send response indicating connection status
    Wire.write('1');
  }
}
```

En Raspberry:

```
import serial
import time

SERIAL_PORT = '/dev/ttyACM0'
BAUD_RATE = 9600

ser = serial.Serial(SERIAL_PORT, BAUD_RATE)
time.sleep(2)

def send_command(command):
    ser.write(command.encode())

def read_commands(filename):
    commands = {}
    with open(filename, 'r') as f:
        for line in f:
            key, value = line.strip().split(':')
            commands[key] = value
    return commands

def read_commands(filename):
    commands = {}
    with open(filename, 'r') as f:
        for line in f:
            key, value = line.strip().split(':')
            commands[key] = value
    return commands

def print_help(commands, command=None):
    if command:
        if command in commands:
            print(f"{command}: {commands[command]}")
        else:
            print(f"No help available for '{command}'.")
    else:
        for key, value in commands.items():
            print(f"{key}: {value}")

commands = read_commands('commands.txt')

try:
            while True:
```

```
            user_input = input("Enter command: ")

            if user_input.startswith('help'):
                command = user_input.split()[1] if len(user_input.split()) > 1 else None
                print_help(commands, command)

            elif user_input in ['0','1','2','3','4','5','8']:
                send_command(user_input)

                if user_input == '8':
                    response = ser.readline().decode().strip()
                    print(response)
                    response = ser.readline().decode().strip()
                    print(response)
                    response = ser.readline().decode().strip()
                    print(response)
                    response = ser.readline().decode().strip()
                    print(response)

            else:
                print("Invalid input")

except:
    print("Exiting program")
    ser.close()
```

Anexo 5

Lista de comandos utilizados para simulación de control del Robot

help: Este comando mostrara las acciones de cada comando.
W: Robot hacia adelante.
A: Robot izquierda.
S: Robot atrás.
D: Robot derecha.
1: Luz arduino uno 1 on.
0: Luz arduino uno 1 off.
3: Luz arduino nano 1 on.
2: Luz arduino nano 1 off.
5: Luz arduino nano 2 on.
4: Luz arduino nano 2 off.
8: Estado módulos.
Q: giro contra reloj.
R: giro reloj
WA: dirección Noroeste
WD: dirección Noreste
SD: direccion Sureste
SA: direccion Suroeste
X: detener

Anexo 6

Video de prueba alimentación de motores
https://youtube.com/shorts/OpveDbkEhlI?feature=share

Anexo 7

Video de funcionamiento de motores 1
https://youtube.com/shorts/c3rs1m8kdjI

Anexo 8

Video de funcionamiento de motores 2
https://youtu.be/8c_W9Lcx3oY

Anexo 9

Video de funcionamiento de motores 3
https://youtu.be/YEsJAs6TnEU

Anexo 10

Video de primera conexión I2C
https://youtu.be/yIe2l5VMeF0

Anexo 11

Video de prueba de Motores con I2C
https://youtu.be/l2slOlWaFm0

Anexo 12

Video de prueba movilidad omnidireccional simple
https://youtu.be/MuG2DF5r-V8

Anexo 13

Video de prueba de Detección de módulos por medio de Interfaz Gráfica
https://youtu.be/416GX4xLJxY

Anexo 14

Video de prueba de detección inalámbrica de módulos y búsqueda de comandos
https://youtu.be/90nfJL79tYs

Anexo 15

Video de prueba de visualización de comandos mandados al controlador
https://youtu.be/JYZcOPtgDrA

Anexo 16

Tabla de Lista de requerimientos (Fuente: Elaboración propia)

Proyecto			Robot Modular Educacional	Fecha: 04/05/2023
Cliente:			PUCP	Elaborado
N	Módulo o Sistema	Deseo o Exigencia	Descripción	Dominio
1	Sistema	E	Sistema mecatrónico capaz de atravesar un camino plano que puede o no tener obstáculos. Esto se realizará con diferentes métodos según la configuración de los módulos conectados.	Función Principal
2	Sistema	E	El sistema contará con 1 módulo principal conectado a 3 tipos de módulos los cuales añadirán operaciones adicionales. Estos serán el módulo de sensores para obstáculos, módulo de actuadores y el módulo de control autónomo.	Operaciones Adicionales
3	Sistema	E	Uniones entre módulos por medio de forma o con pocas uniones atornilladas de manera que sean fáciles de remover.	Ensamble
4	Sistema	E	Para realizar el prototipo de utilizarán componentes del mercado local o componentes baratos de importar.	Fabricación
5	Sistema	E	La movilidad del sistema se realizará por medio de una configuración omnidireccional.	Diseño
6	Sistema	E	La velocidad de desplazamiento máxima que podrá tener el sistema será de 1m/s. La velocidad rotacional máxima será de 10 rad/s.	Cinemática
7	Sistema	E	El robot podrá soportar una carga máxima de 0.5kg. Esto incluye la carga debido a los objetos con los que interactúe el actuador.	Mecánica
8	Sistema	E	El robot será capaz de moverse en el plano XY, 2 grados de libertad. Tendrá adicionalmente la capacidad de rotar debido a la configuración omnidireccional, añadiendo 1 grado de libertad adicional.	Cinemática

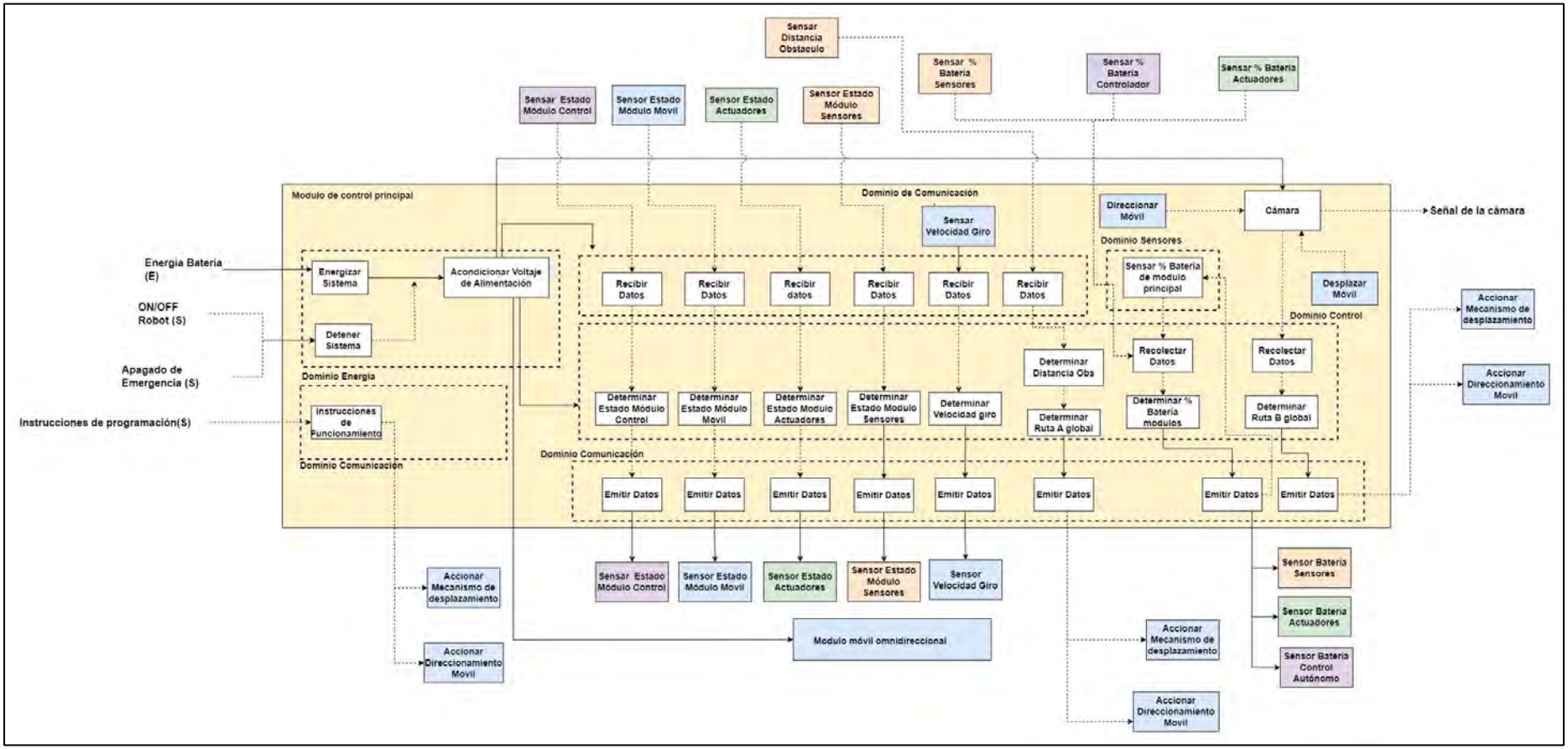
9	Sistema	D	Se implementarán 4 meses de diseño conceptual y 4 meses de diseño técnico.	Tiempo
10	Sistema	D	Las dimensiones totales del robot con todos los módulos acoplados será de máximo 40 cm de alto (incluyendo las ruedas).	Geometría
11	Módulo	E	Dimensiones máximas totales de 20 cm. (Largo, ancho, alto) por modulo, excluyendo al módulo principal.	Geometría
12	Módulo	E	Todos los módulos serán alimentados por un sistema de baterías localizado en el módulo principal.	Electrónica
13	Módulo	D	El material de las carcasas que contendrán los módulos del robot será PLA.	Material
14	Módulo	E	Los módulos deberán tener un etiquetado muy claro para las conexiones, o bien emplear conexiones de entrada única.	Integración
15	Módulo	E	El módulo principal de control se comunicará con los demás módulos.	Comunicaciones
16	Módulo	E	El módulo de control deberá tener una manera de determinar el estado de los demás módulos para así poder funcionar con las diferentes configuraciones pensadas para el sistema.	Especificaciones
17	Módulo	E	En base a la ruta determinada por el usuario, el módulo de control deberá enviar señales al mecanismo de avance para seguir la ruta.	Señales
18	Módulo	E	El módulo de control autónomo deberá tener un controlador encargado de dirigir la movilidad del sistema calculando la ruta autónoma así como la manual. De igual manera, este será el módulo encargado de detectar las conexiones de los demás módulos.	Control
19	Módulo	E	Adicionalmente, al tener que calcular la ruta autónoma este módulo determinará la ruta que deberá seguir. Para ello utilizará un componente que le permita conocer su ruta actual, junto con modelos cinemáticos según el tipo de ruta deseado.	Software

20	Módulo	E	El módulo de sensores para obstáculos deberá tener un sensor de posición, de manera que se puedan detectar los obstáculos presentes. Se emitirá al módulo de control la distancia al obstáculo.	Sensores
21	Módulo	E	El módulo principal deberá incluir motores para girar las ruedas y un encoder para poder realizar el control del movimiento. Enviará señales al módulo de control de la velocidad medida por el sensor y el estado del módulo.	Mecánica
22	Módulo	D	El módulo de actuadores incluirá un empujador lineal el cual le permitirá impactar en obstáculos pequeños, marcar estos obstáculos o presionar botones. Así mismo enviará al módulo principal señales del estado del módulo.	Actuadores

Anexo 17

Diagrama de Módulo de control principal

Este módulo será el encargado de conectar el resto de los módulos e integrarlos con diferentes configuraciones para el funcionamiento del sistema. Este módulo de control principal deberá estar siempre conectado al módulo móvil para poder funcionar en la configuración más básica del robot. Los demás módulos serán opcionales, cada uno agregando una función adicional. Este módulo se aprecia en la figura a continuación.



Dominio de energía

- Energizar sistema: función encargada de alimentar el módulo.
- Detener sistema: función que detiene el funcionamiento del módulo, bien sea por la señal on/off o por la parada de emergencia.
- Acondicionar Voltaje: función que acondiciona el voltaje de los diferentes componentes que conforman al módulo.

Dominio de comunicación

- Cargar Programas: función que recibe las instrucciones de programación para el funcionamiento del sistema.

Dominio de sensores

- Sensar porcentaje de batería módulo principal: función que sensa el porcentaje de batería restante del módulo principal.

Dominio de control

- Determinar estado módulo control autónomo: determina si el módulo de control autónomo está conectado.
- Determinar estado módulo sensores: determina si el módulo de sensores está conectado.
- Determinar estado módulo móvil: determina si el módulo móvil está conectado.
- Determinar estado módulo actuadores: determina si el módulo de actuadores está conectado.
- Determinar porcentaje de batería: determinar porcentaje de batería de los módulos del robot.
- Determinar Velocidad de giro: determinar velocidad de giro del motor.
- Determinar Distancia Obstáculo: determinar distancia a obstáculos.
- Determinar Ruta: determinar ruta a seguir en base a imagen de cámara.
- Determinar Ruta: determinar ruta a seguir en base a obstáculos detectados.

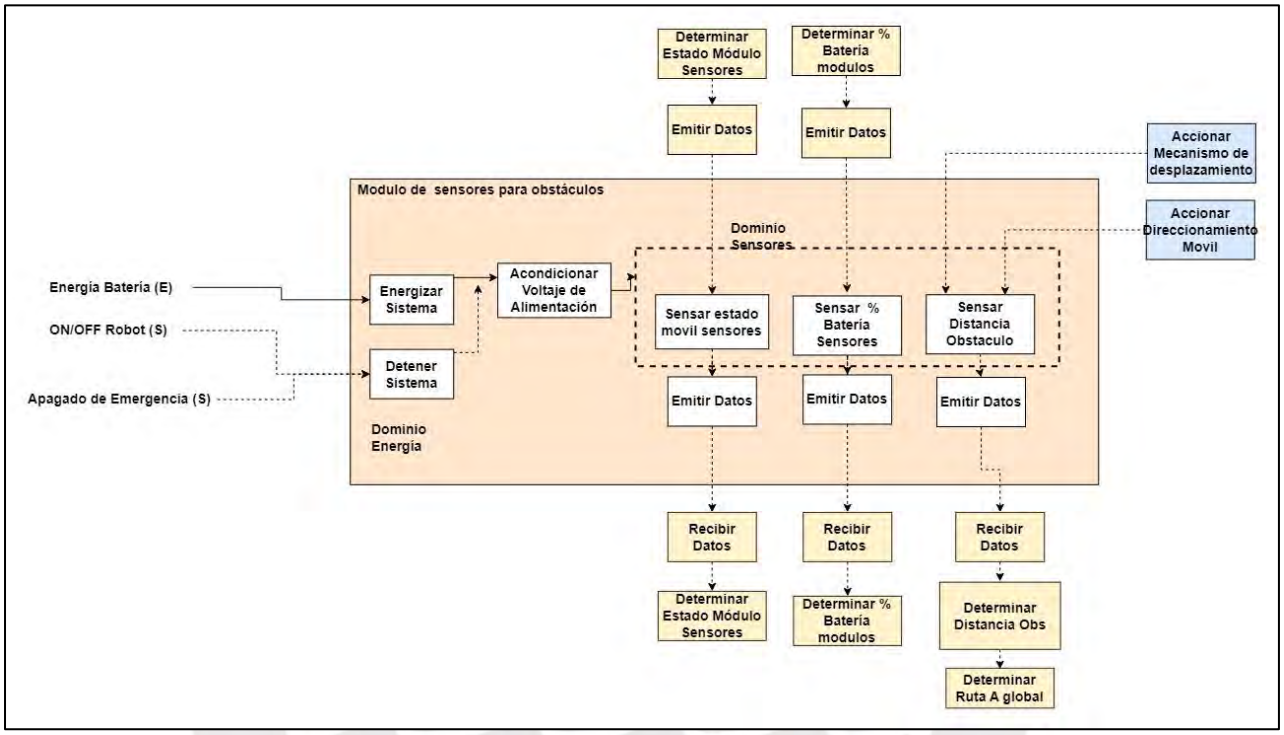
Dominio de Actuadores

- Cámara: función que capta señal de video por medio de una cámara.

Anexo 18

Diagrama de Módulo de sensores para obstáculos

Módulo por el medio de cual se le podrán añadir sensores que puedan detectar obstáculos. Adicionalmente a esta función, este módulo cuenta con una alimentación propia, así como detectores de la batería restante y si está conectado al módulo principal. Esto se aprecia en la figura a continuación.



Dominio de energía

- Energizar sistema: función encargada de alimentar el módulo.
- Detener sistema: función que detiene el funcionamiento del módulo en base a la señal on/off.
- Acondicionar Voltaje: función que acondiciona el voltaje de los diferentes componentes que conforman al módulo.

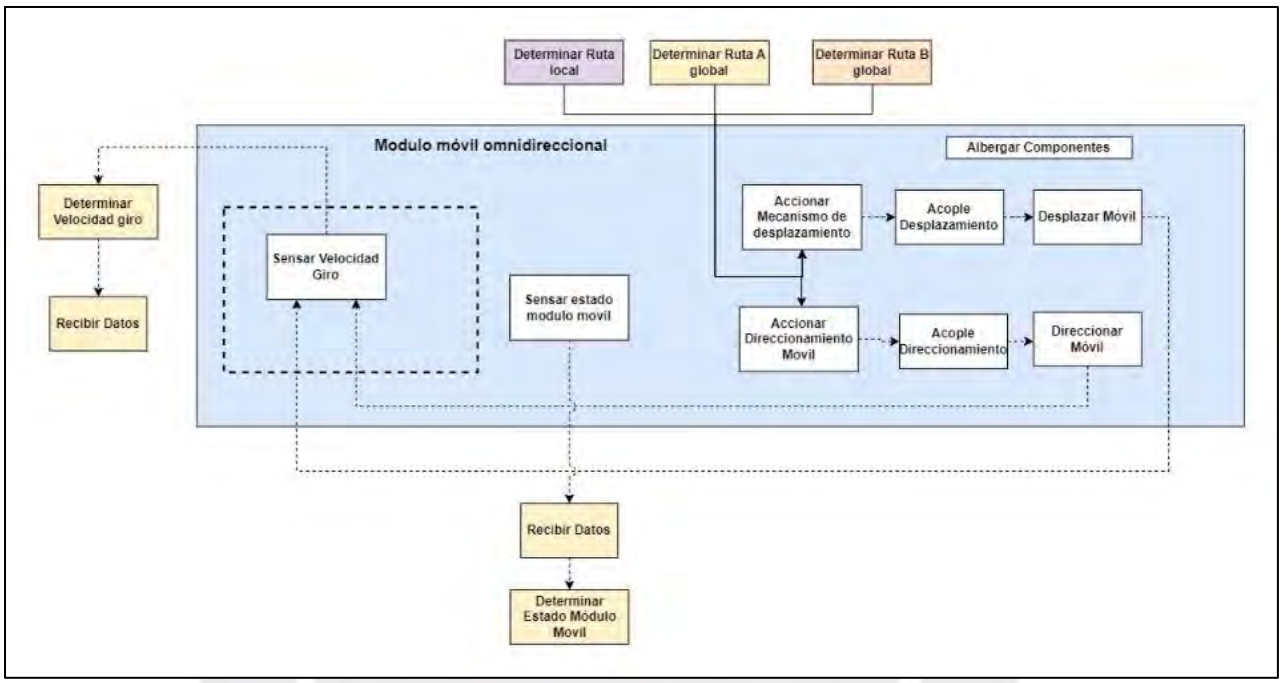
Dominio de sensores

- Sensar porcentaje de batería módulo sensores: sensa el porcentaje de batería del módulo de sensores.
- Sensar distancia a obstáculo: sensa la distancia del robot a obstáculos que podría haber.
- Sensar estado modulo sensores: sensa si el módulo se encuentra conectado al módulo principal. Emite una señal al módulo principal si detecta que está conectado.

Anexo 19

Diagrama de Módulo Móvil

El módulo móvil es el encargado de accionar el desplazamiento del móvil, así como accionar el cambio de dirección. Este módulo es de los más importantes, puesto que este módulo es vital para que el robot modular pueda avanzar. Cabe resaltar que este módulo es parte del módulo principal, por lo que se consideran un solo módulo. Sin embargo, se toman como entidades diferentes para el diagrama de funciones por cuestiones de orden. Esto se aprecia en la figura a continuación.



Dominio de Sensores:

- Sensar velocidad: función que sensa la velocidad de giro del motor.
- Sensar estado del módulo móvil:

Dominio de Actuadores

- Accionar mecanismo de desplazamiento: función que proporciona la fuerza para poder accionar el mecanismo de desplazamiento.
- Accionar direccionamiento: función que proporciona la fuerza para poder accionar el mecanismo de direccionamiento.
- Desplazar Móvil: función que desplaza el móvil.
- Direccionar Móvil: función que direcciona el móvil.

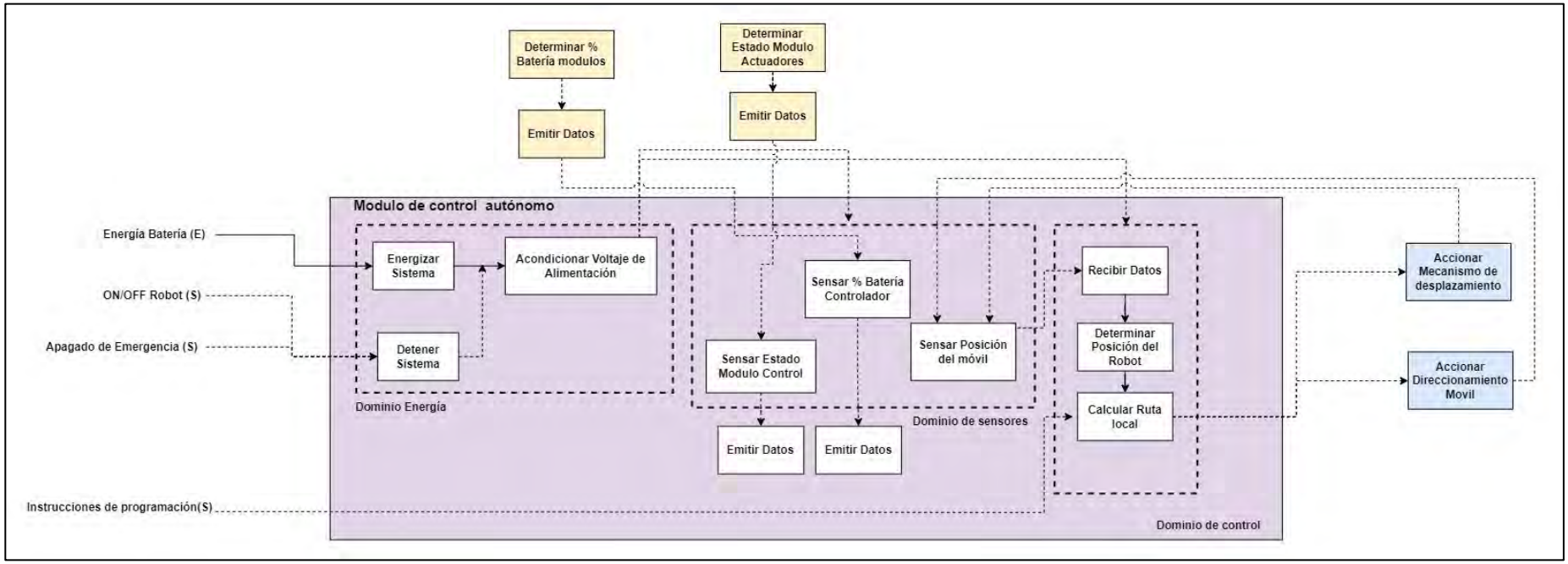
Dominio Mecánico

- Albergar Componentes: esta función es la que permite que todos los demás componentes se integren formando un móvil.

Anexo 20

Diagrama de Módulo de control autónomo

El módulo de control 1, así como el módulo de control 2 tendrán las mismas funciones, entradas y salidas. Con la diferencia que cada módulo de control contará con un controlador diferente. Permitiendo al usuario experimentar con más de un tipo de controlador. Las funciones específicas de ambos controladores se detallan en la figura a continuación.



Dominio de energía

- Energizar sistema: función encargada de alimentar el módulo.
- Detener sistema: función que detiene el funcionamiento del módulo en base a la señal on/off.
- Acondicionar Voltaje: función que acondiciona el voltaje de los diferentes componentes que conforman al módulo.

Dominio de sensores

- Sensar porcentaje de batería módulo de control: sensa el porcentaje de batería restante en el módulo de sensores.
- Sensar Posición del móvil: sensa la posición en la que se encuentra el móvil para realizar navegación autónoma.
- Sensar estado Modulo de Control: sensa si el modulo de control autónomo está conectado al módulo principal o no.

Dominio de control

- Determinar posición móvil: determina la posición en la que se encuentra el móvil para realizar navegación autónoma.
- Determinar calcular ruta robot: calcula la ruta del robot para que pueda manejar de manera autónoma.

Anexo 21

Diagrama de Módulo de actuadores

Módulo el cual permitirá al sistema poder utilizar actuadores. El actuador principal será un dispositivo de empuje que le permitirá al robot empujar obstáculos. Las funciones específicas de detallan en la figura a continuación.

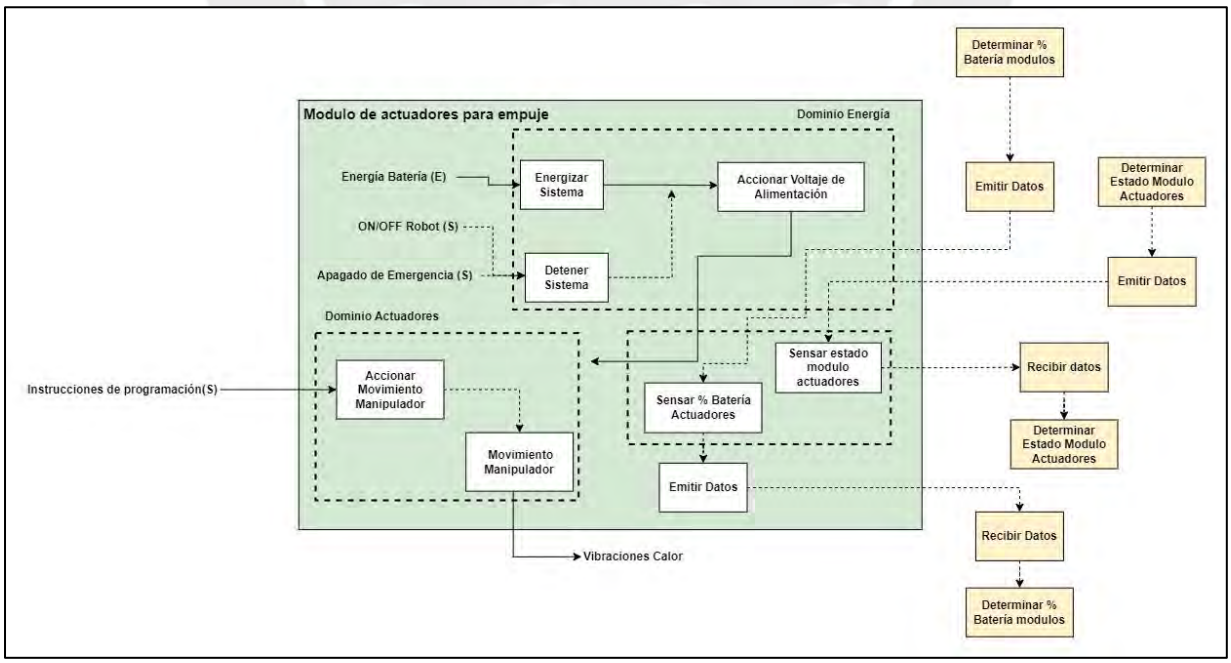


Figura 3. 22 Diagrama de Actuadores (Fuente: Elaboración propia)

Dominio de energía

- Energizar sistema: función encargada de alimentar el módulo.
- Detener sistema: función que detiene el funcionamiento del módulo en base a la señal on/off.
- Acondicionar Voltaje: función que acondiciona el voltaje de los diferentes componentes que conforman al módulo.

Dominio de sensores

- Sensar porcentaje de batería módulo de actuadores: sensa el porcentaje de batería restante en el módulo de actuadores.
- Sensar estado de módulo de actuadores: sensa si el módulo está conectado al módulo principal.

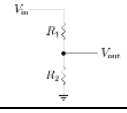
Dominio de actuadores

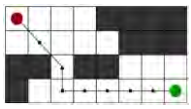
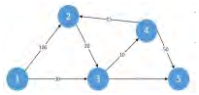
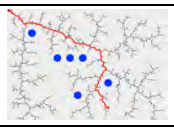
- Accionar movimiento de manipulador: acciona el movimiento del actuador, en este caso un empujador.

Anexo 22

Matriz Morfológica de Módulo de Control Principal

En primera instancia se mostrará el módulo de control principal. El módulo principal cuenta con 5 dominios. El dominio de comunicación necesita recibir señales de prendido/apagado, así como instrucciones de programación y energía para alimentar al módulo. Fueron seleccionados 3 tipos de receptores de señales, un receptor Bluetooth, uno Wifi y recepción por medio de cable USB. En el dominio de sensores se debía detectar el porcentaje de batería del módulo principal. Para ello se proponen 3 métodos. Con revisor resistivo, medidor de voltaje y medidor de corriente y voltaje. De igual manera se proponen 3 tipos de cámaras para poder recibir imágenes, así como 3 diferentes tipos de controladores y 3 algoritmos empleados para efectuar cálculo de rutas. Estos componentes pueden apreciarse en la tabla a continuación.

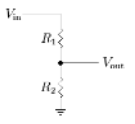
Módulo	Dominio	Funciones parciales	Alternativa 1	Alternativa 2	Alternativa 3
Módulo Principal	Dominio de Comunicación	Recibir Instrucciones de Funcionamiento	Módulo Bluetooth	Módulo Wifi	Cable USB
					
	Dominio de Sensores	Sensar % Batería de modulo principal	Medidor de voltaje con display	Divisor Resistivo	Medidor de corriente y voltaje
					
	Dominio de Actuadores	Capturar Imagen	Cámara Lineal	Cámara de Área	Cámara web
					
		Calcular Rutas	Algoritmo A Star	Algoritmo Dijksrta	Algoritmo RRT

	Dominio de Control (Software)				
	Dominio de Control (Hardware)	**Determinar Velocidad de Giro**	Microcontrolador + sistema embebido	Sistema Embebido	Single Board Computing Device
		Determinar Ruta A			
		Determinar Ruta B			
		Determinar % Batería Módulos			
	Dominio de Comunicación	**Recibir Datos**	Conectores Modulares	Conectores Rápidos	Cables sueltos
		Emitir Datos			
	Dominio de Energía	**Ver Tabla Adjunta**	**Ver Tabla Adjunta**	**Ver Tabla Adjunta**	**Ver Tabla Adjunta**

Anexo 23

Matriz Morfológica de Módulo de Sensores para Obstáculos

Este módulo cuenta con 2 tipos de dominio. El dominio de energía al que se le hará referencia más adelante, y el dominio de sensores como tal. El dominio de sensores ofrece 3 diferentes tipos de sensores, capaces de determinar la distancia a un objeto. Estos incluyen sensores infrarrojos, de ultrasonido y LIDAR. Por otro lado, se ofrecen las mismas opciones para Sensar el porcentaje de batería del módulo. Adicionalmente se ofrecen 3 alternativas para determinar el estado del módulo y enviar dicha señal al módulo principal. Mediante detectores con imanes, detectores de voltaje o sensores a presión. Estos componentes se aprecian en la tabla a continuación.

Módulo	**Dominio**	**Funciones parciales**	**Alternativa 1**	**Alternativa 2**	**Alternativa 3**
Módulo de sensores para obstáculos	**Dominio de Sensores**	**Determinar estado Modulo**	Sensor por Imanes	Detector de voltaje	Sensor a presión
					
		Sensar % Batería de los módulos	Medidor de voltaje con display	Divisor Resistivo	Medidor de corriente y voltaje
					
		Sensar distancia obstáculo	Sensor Infrarrojo	Arreglo de Ultrasonido	Sensor LIDAR

					
	Dominio de Comunicación	**Recibir Datos**	Conectores Modulares	Conectores Rápidos	Cables sueltos
		Emitir Datos			

Anexo 24

Matriz Morfológica de Módulo Móvil

El módulo móvil permite que el sistema pueda movilizarse. Este es el único módulo que no cuenta con alimentación propia, pues es alimentado por el módulo de control principal. Consta de 2 dominios, el dominio de sensores y el dominio de actuadores.

Para el dominio de sensores se cuenta con 2 tipos de sensores. El enconder, como única opción para determinar la velocidad rotacional del sistema, y los sensores para determinar el estado el módulo.

El dominio de actuadores incluye diferentes motores para accionar el mecanismo de desplazamiento, así como 2 configuraciones omnidireccionales que permiten el direccionamiento. Finalmente, se desplazará el móvil con 3 tipos de ruedas. Todos estos componentes pueden ser apreciados en la tabla a continuación.

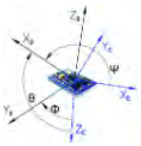
Dominio	Funciones parciales	Alternativa 1	Alternativa 2	Alternativa 3
Dominio de Sensores	**Determinar estado Modulo**	Sensor por Imanes	Detector de voltaje	Sensor a presión
				
	Sensar Velocidad	Encoder		
				
Dominio de Actuadores	**Accionar Desplazamiento**	Servo Motor	Motor a Paso	Motor Reductor
	Accionar Direccionamiento			
	Acople Desplazamiento	Acople Directo		
	Acople Direccionamiento			
	Desplazar Móvil	Ruedas Omnidireccionales	Ruedas Amarillas	Ruedas Mecanum
	Girar Móvil			
Dominio Mecánico	**Albergar Componentes**	Configuración Omnidireccional 3 ruedas + Carcasa Impresa		Configuración Omnidireccional 4 ruedas + Carcasa Impresa
				
Dominio de Comunicación	**Comunicar Señales**	Conectores Modulares	Conectores Rápidos	Cables sueltos
				

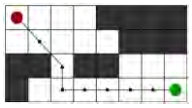
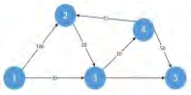
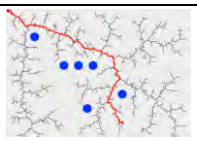
Anexo 25

Matriz Morfológica de Módulo de Control Autónomo

Este módulo tiene 3 diferentes dominios. El dominio de energía (el cual se explicará más adelante), el dominio de sensores y el dominio de control. El dominio de control tiene 2 tareas, determinar la posición del móvil la cual se llevará a cabo por uno de 3 microcontroladores propuestos; y finalmente la tarea de calcular la ruta. Se han investigado 3 diferentes algoritmos que permitirán el cálculo de la ruta.

Adicionalmente en el dominio de sensores se encuentran los mismos 3 componentes para Sensar el porcentaje de batería del módulo, así como determinar el estado del módulo. Finalmente se cuenta con un sensor el cual determinará la posición del robot, permitiendo así el cálculo de una ruta autónoma a seguir. Estos componentes se encuentran detallados en la tabla a continuación.

Módulo	Dominio	Funciones parciales	Alternativa 1	Alternativa 2
Dominio de Sensores	**Determinar estado Modulo**	Sensor por Imanes	Detector de voltaje	Sensor a presión
				
	Sensar % Batería de los módulos	Medidor de voltaje con display	Divisor Resistivo	Medidor de corriente y voltaje
				
	Sensar Posición Móvil	POZYX	GPS	IMU
				
Dominio de Control (Hardware)	**Determinar Posición Móvil**	Microcontrolador	Sistema Embebido	Microcontrolador + Sistema Embebido
				
Dominio de Control	**Calcular Ruta Robot**	Algoritmo A Star	Algoritmo Dijksrta	Algoritmo RRT

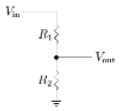
(Software)				
Dominio de Comunicación	**Comunicar Señales**	Conectores Modulares	Conectores Rápidos	Cables sueltos
				
Dominio de Energía	**Ver Tabla Adjunta**	**Ver Tabla Adjunta**	**Ver Tabla Adjunta**	**Ver Tabla Adjunta**

Anexo 26

Matriz Morfológica de Módulo de Actuadores

El módulo de actuadores cuenta con 3 dominios, el dominio de energía con las mismas opciones que en los otros módulos el dominio de sensores que deberá Sensar los porcentajes de batería y el estado del módulo y finalmente el dominio de actuadores como tal.

Este último dominio ofrece 3 tipos de actuadores lineales los cuales pueden empujar obstáculos livianos en su camino. Entre ellos se encuentran un solenoide lineal, un actuador lineal hidráulico y un actuador neumático de empuje. El detalle se muestra en la tabla a continuación.

Módulo	Dominio	Funciones parciales	Alternativa 1	Alternativa 2	Alternativa 3
Módulo de Actuadores	**Dominio de Actuadores**	**Determinar estado Modulo**	Sensor por Imanes	Detector de voltaje	Sensor a presión
					
		Sensar % Batería de los módulos	Medidor de voltaje con display	Divisor Resistivo	Medidor de corriente y voltaje
					
		Accionar Movimiento Manipulador	Servo Motor	Solenoide Lineal	

					
		Movimiento Manipulador	Mecanismo Biela Manivela	Mecanismo Piñón Cremallera	Solenoide Lineal
					
	Dominio de Comunicación	**Comunicar Señales**	Conectores Modulares	Conectores Rápidos	Cables sueltos
					
	Dominio de Energía	**Ver Tabla Adjunta**	**Ver Tabla Adjunta**	**Ver Tabla Adjunta**	**Ver Tabla Adjunta**

Anexo 27

Matriz Morfológica de Dominio de Energía

El dominio de energía consta de 3 principales funciones. La primera es energizar el sistema, función para la cual se proponen 3 tipos de baterías. La segunda función es el acondicionar el voltaje para cada componente, para lo cual se proponen 3 diferentes tipos de reguladores. Finalmente se tiene la función de detener el sistema, para ello se propone un pulsador de emergencia, una palanca y un switch. Estos componentes se pueden apreciar de mejor manera en la tabla a continuación.

Dominio	Módulos Pertenecientes	Función	Opción 1	Opción 2	Opción 3
Dominio Energía	**Control Principal, Control autónomo, Módulo Sensores, Módulo actuadores**	**Acondicionar voltaje de alimentación**	Regulador lineal	Regulador conmutado	Diodo Zenner
					

	Control Principal, Control autónomo, Módulo Sensores, Módulo actuadores	**Energizar sistema**	Batería Alcalina	Batería de LiPo	Batería Ion-Litio
					
	Control Principal, Control autónomo, Módulo Sensores, Módulo actuadores	**Detener sistema**	Pulsador de emergencia	Palanca	Switch de emergencia
					

Anexo 28

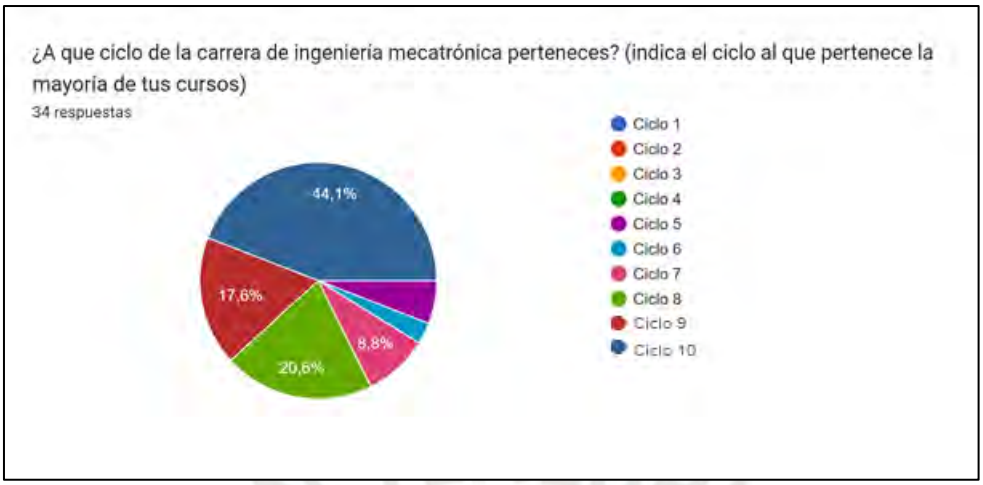
Matriz Morfológica de Dominio Mecánico

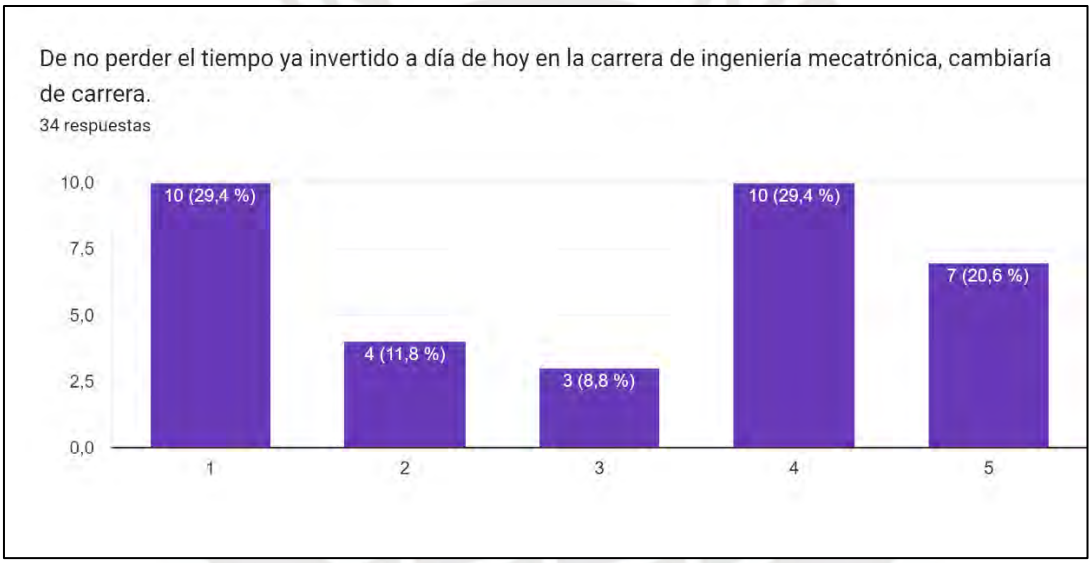
En este caso el dominio mecánico se refiere al uso de diferentes materiales para imprimir las carcasas que tendrán en su interior los diferentes módulos. Dichas carcasas serán fabricadas por medio de impresión 3D con las opciones reflejadas en la tabla a continuación.

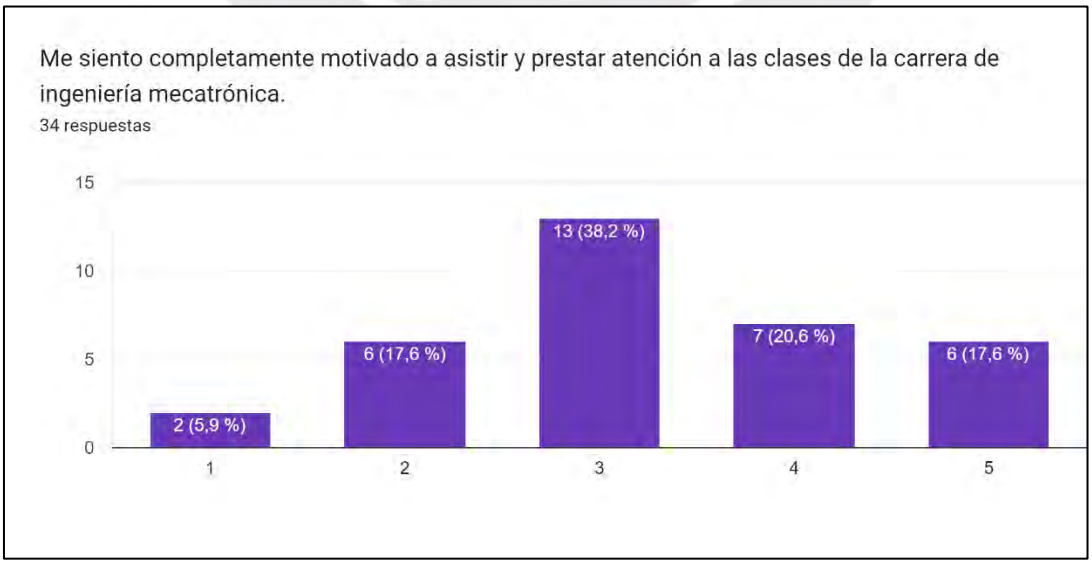
Dominio	Función	Opción 1	Opción 2	Opción 3
Dominio Mecanico	**Material de Impresión**	Caja protectora de plástico ABS	Caja protectora de PLA	Caja protectora de PETG
				

Anexo 29

Encuesta a Alumnos de Ingeniería de la PUCP

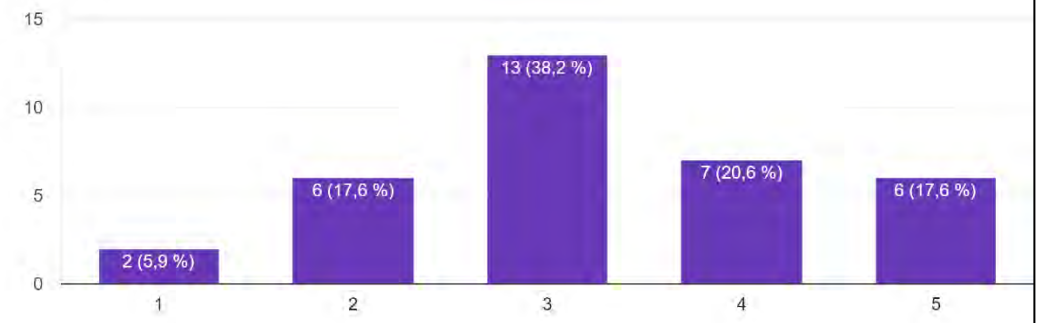






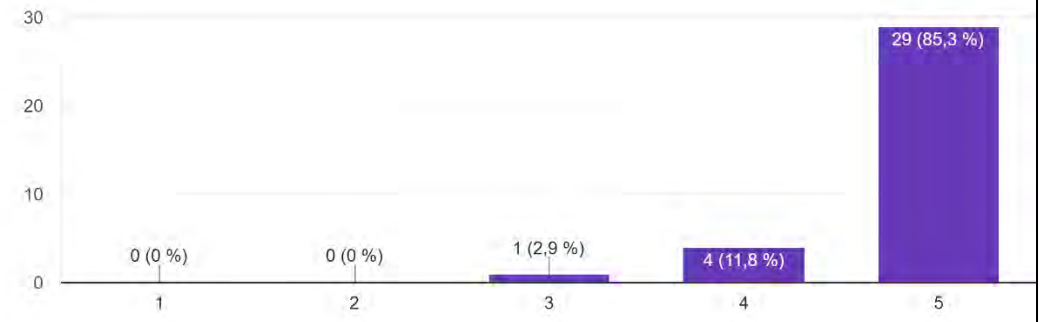
Me siento completamente motivado a asistir y prestar atención a las clases de la carrera de ingeniería mecatrónica.

34 respuestas



De tener más experiencias prácticas, y no solo experiencias teóricas, los alumnos se sentirían más motivados con la carrera.

34 respuestas



Anexo 30

Evaluación Técnico Económica

En la sección a continuación se presentarán los factores técnicos y económicos que se utilizaron para evaluar el desempeño de los diferentes conceptos de solución desarrollados a partir de la matriz morfológica.

Evaluación Técnica

En primer lugar, se tomaron criterios técnicos los cuales cambian según la selección de componentes. Estos se presentan a continuación:

- **Autonomía energética:**
  Este criterio toma en cuenta la autonomía del sistema energético que se propone para cada uno de los conceptos de solución. Se le otorga un peso ponderado de 3/11, debido a que es importante que el sistema tenga una buena autonomía energética.

- **Nivel de modularidad:**
  El nivel de modularidad es un criterio que considera los conectores utilizados para la comunicación y unión entre componentes. El tipo de uniones determinará la facilidad de conexión física entre módulos, es por ello que se considera un criterio importante y se le otorga un peso ponderado de 3/11.

- **Conectividad de señales:**
  Este criterio toma en cuenta que tan efectiva es el recibimiento de señales por parte del sistema. Entre las opciones tenemos conexión por bluetooth, wifi o USB. Es importante considerar que la conexión debe ser buena, y al mismo tiempo no debe entorpecer el movimiento del sistema. Se le otorga un peso ponderado de 3/11.

- **Estabilidad mecánica:**
  Finalmente se considera la estabilidad mecánica del robot considerando una configuración omnidireccional bien de 3 ruedas o 4 ruedas. Si bien este criterio es importante, al ser diseñado para cruzar terrenos planos no es un criterio tan relevante como los demás. Por lo que se le otorga un peso ponderado de 2/11.

Evaluación Económica

De manera similar, se evaluaron criterios económicos según su peso correspondiente. Los criterios económicos son los siguientes:

- **Costo de materiales:**
  El costo de materiales depende de lo costosos que puedan llegar a ser los componentes. Particularmente componentes como la Jetson nano o la cámara son los que pueden llegar a ser más costosos. Al ser el objetivo del sistema el ser de bajo costo es un criterio muy relevante con un peso ponderado de 4/12.

- **Disponibilidad en el mercado local:**
  Este factor influye en el costo de los componentes debido a que tener que importar implica un costo de transporte significativo. Es por ello que se le otorga un peso ponderado de 3/12.

- **Procesos de Fabricación:**
  Dependiendo del proceso de fabricación de las carcasas de los módulos, cambiarán los costos de estos. Se ha decidido que las carcasas serán impresas en 3D, sin embargo, los distintos materiales de impresión cambiarían el costo de fabricación. Al no haber diferencias muy significativas se le otorga un peso ponderado de 2/12.

- **Costo de mantenimiento:**
  El costo de mantenimiento toma en cuenta el mantenimiento que se le debe dar a los componentes y demás partes del sistema para asegurar el continuo funcionamiento. Tal como el mantenimiento al mecanismo de empuje, a las ruedas, los motores y demás. Al ser importante que sea lo mas duradero posible se le otorga un peso ponderado de 3/12

Gráfico Ponderado

A continuación, en las siguientes tablas se resumirá lo mencionado en la evaluación técnica y la evaluación económica, a partir de lo cual se procederá a seleccionar la alternativa óptima.

<u>Evaluación Técnica</u>

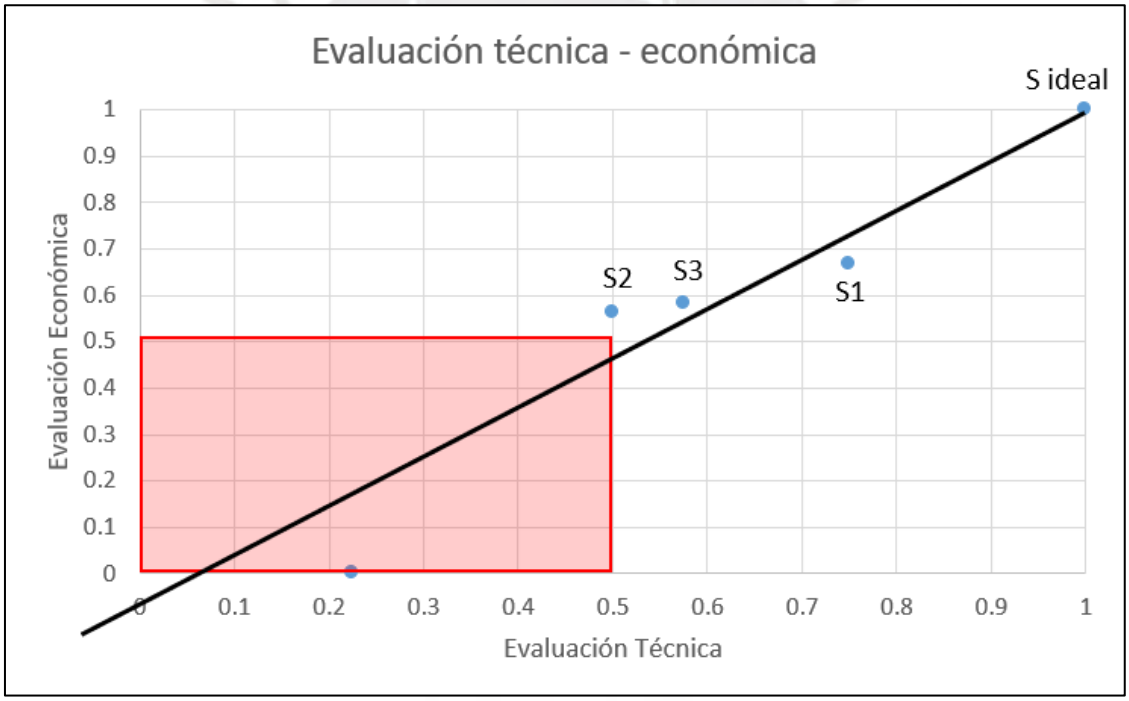
Criterio técnico		Sol 1		Sol 2		Sol 3		Sol Ideal	
	g	p	gp	p	gp	p	gp	p	gp
Autonomía Energética	3	3	9	2	6	2	6	4	12
Nivel de Modularidad	2	3	6	2	4	1	2	4	8
Conectividad de Señales	3	3	9	2	6	3	9	4	12
Estabilidad Mecánica	2	3	6	2	4	3	6	4	8
Sumatoria			30		20		23		40
xi			0.75		0.5		0.575		1

Evaluación Económica

Criterio económico		Sol 1		Sol 2		Sol 3		Sol Ideal	
	g	p	gp	p	gp	p	gp	p	gp
Costo de Mantenimiento	3	3	9	2	6	2	6	4	12
Costo de materiales	4	2	8	3	12	3	12	4	16
Disponibilidad en el mercado local	3	3	9	1	3	2	6	4	12
Procesos de Fabricación	2	3	6	3	6	2	4	4	8
Sumatoria			32		27		28		48
xi			0.667		0.563		0.583		1

Comparación gráfica:

La figura a continuación proyecta en un eje x el puntaje ponderado de cada criterio técnico, y en un eje y el promedio ponderado de cada criterio económico. Finalmente se toma el concepto de solución más cercano a la solución ideal.



Anexo 31

Data sheet Motor 37D 300RPM

37D Metal Gearmotors

Pololu

Pololu 37D Metal Gearmotors are powerful brushed DC motors paired with 37mm-diameter gearboxes. There are nine different gearbox options available, ranging from 6.3:1 to 150:1, and two different motor options: 12 V and 24 V. The 24 V versions offer approximately the same speed and torque at 24 V as their 12 V counterparts do at 12 V, with approximately half the current draw. This datasheet includes two sets of performance graphs for each version, one at its nominal voltage and one at half of its nominal voltage. Each version is available with an integrated 64 CPR quadrature encoder on the motor shaft.

Note: The original versions of these gearmotors had gearboxes with all spur gears. In August 2019, these were replaced by functionally identical "Helical Pinion" versions that feature helical gears for the first stage of the gearbox, which reduces noise and vibration and improves efficiency. The picture on the right shows the helical pinion gear and first mating gear.

Performance summary and table of contents

Rated Voltage	Pololu Item #	Gear Ratio	No Load		At Maximum Efficiency				Max Power	Stall Extrapolation[2]		Graph Pages
			Speed	Current	Speed	Torque	Current	Output		Torque	Current	
		:1	RPM	A	RPM	kg·mm	A	W	W	kg·mm	A	
12 V	4750[1]	1	10,000	0.2						5	5.5	
	4747, 4757	6.25	1600		1300	4.9	1.2	6.4	12	30		5, 6
	4748, 4758	10	1000		850	6.6	0.91	5.7	12	49		7, 8
	4741,4751	18.75	530		470	10	0.76	5.0	12	85		9, 10
	4742, 4752	30	330		280	18	0.78	5.1	12	140		11, 12
	4743, 4753	50	200		180	22	0.66	4.0	10	210		13, 14
	4744, 4754	70	150		130	32	0.68	4.2	10[3]	270		15, 16
	4745, 4755	102.08	100		87	42	0.72	3.8	8[3]	340		17, 18
	4746, 4756	131.25	76		66	60	0.74	4.1	6[3]	450		19, 20
	2828, 2829	150	67		58	65	0.72	3.8	6[3]	490		21, 22
24 V	4690[1]	1	10,000	0.1						5.5	3.0	
	4688, 4698	6.25	1600		1300	5.5	0.58	7.4	14	35		23, 24
	4689, 4699	10	1000		850	7.5	0.49	6.6	14	55		25, 26
	4681,4691	18.75	530		450	13	0.49	6.1	13	95		27, 28
	4682, 4692	30	330		280	19	0.46	5.5	13	150		29, 30
	4683, 4693	50	200		170	27	0.41	4.9	12	230		31, 32
	4684, 4694	70	140		120	39	0.42	5.0	10[3]	310		33, 34
	4685, 4695	102.08	100		86	51	0.42	4.5	8[3]	390		35, 36
	4686, 4696	131.25	79		68	63	0.40	4.4	6[3]	470		37, 38
	4687, 4697	150	68		59	73	0.41	4.4	6[3]	560		39, 40

Notes:

(1) Max efficiency data and performance graphs currently unavailable for the motors without gearboxes (items #4750 and #4690).

(2) Listed stall torques and currents are theoretical extrapolations; units will typically stall well before these points as the motors heat up. Stalling or overloading gearmotors can greatly decrease their lifetimes and even result in immediate damage. The recommended upper limit for continuously applied loads is 100 kg·mm, and the recommended upper limit for instantaneous torque is 250 kg·mm. Stalls can also result in rapid (potentially on the order of seconds) thermal damage to the motor windings and brushes; a general recommendation for brushed DC motor operation is 25% or less of the stall current.

(3) Output power for these units is constrained by gearbox load limits; spec provided is output power at max recommended load of 100 kg·mm.

[Enlace]

Anexo 32

Data sheet Arduino Uno

Arduino® UNO R3

Product Reference Manual
SKU: A000066

Description

The Arduino UNO R3 is the perfect board to get familiar with electronics and coding. This versatile development board is equipped with the well-known ATmega328P and the ATMega 16U2 Processor.
This board will give you a great first experience within the world of Arduino.

Target areas:

Maker, introduction, industries

Enlace

Anexo 33

Data sheet Driver VNH

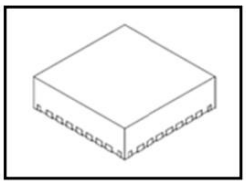
TOSHIBA

TB9051FTG

Toshiba Bi-CMOS Linear Integrated Circuit Silicon Monolithic

TB9051FTG

PWM type single channel H-Bridge DC brushed motor driver for automotive use



P-QFN28-0606-0.65-001

1. Outline

This product is a motor driver IC which incorporates the output driver for the direct drive of a DC brushed motor intended for the automotive use.

The motor drive output can be highly efficient operation by the PWM control which realizes low-on resistance.

Forward / Reverse / brake mode can be selected according to PWM1 signal and PWM2 signal, and the motor operation mode and stop mode can be selected by ENABLE pin.

Moreover, the output current capacity is 5A (max), it is suitable for various automotive applications such as a throttle and valve control, various engine bulbs, storing of door mirrors, and a seat positioning.

2. Application

Automotive applications such as a throttle and valve control, various engine bulbs, and storing of door mirrors

3. Feature

- Motor driver block: Single channel H-Bridge driver (Ron(Pch+Nch))<0.45 Ω (Max @Tj = 150°C, VBAT = 8 V)
- Abnormality detection function: Over-current detection, over-temperature detection, VBAT undervoltage detection, VCC undervoltage detection, and VCC high voltage detection
- Built-in initial diagnosis function: Power supply abnormality detection circuit (VBAT undervoltage, VCC undervoltage and VCC high voltage.)
- Output type: PWM control output
- Motor operation: Forward /Reverse/ Brake
- Current limitation control: Current limiter with chopper type
- Output high-side current monitoring function (OCM pin)
- DIAG output
- Built-in the through current prevention circuit
- Operating voltage range: VBAT = 4.5 to 28 V (Maximum ratings of power supply voltage 40V (max): 0.5 sec.)
- Operating temperature range: Ta = -40°C to 125°C
- Compact type flat package: P-QFN28-0606-0.65-001
- AEC-Q100 Qualified

・If the label of shipping box is indicated to be "[[G]]/RoHS COMPATIBLE2", "[[G]]/RoHS [[Chemical symbol(s) of controlled substance(s)", and "RoHS COMPATIBLE" or "RoHS COMPATIBLE, [[Chemical symbol(s) of controlled substance(s)]]>MCV", this product is compliant with the EU RoHS Directive (2011 / 65 / EU) in the meaning of the statement.

Enlace

Anexo 34

Data sheet Regulador de voltaje

XLSEMI

Datasheet

5A 300KHz 32V Buck DC to DC Converter **XL4005**

Features

- Wide 5V to 32V Input Voltage Range
- Output Adjustable from 0.8V to 30V
- Maximum Duty Cycle 100%
- Minimum Drop Out 0.6V
- Fixed 300KHz Switching Frequency
- 5A Constant Output Current Capability
- Internal Optimize Power MOSFET
- High efficiency
- Excellent line and load regulation
- TTL shutdown capability
- EN pin with hysteresis function
- Built in thermal shutdown function
- Built in current limit function
- Built in output short protection function
- Available in TO-263 package

Applications

- LCD Monitor and LCD TV
- Digital Photo Frame
- Set-up Box
- ADSL Modem
- Telecom / Networking Equipment

General Description

The XL4005 is a 300KHz fixed frequency PWM buck (step-down) DC/DC converter, capable of driving a 5A load with high efficiency, low ripple and excellent line and load regulation. Requiring a minimum number of external components, the regulator is simple to use and include internal frequency compensation and a fixed-frequency oscillator.

The PWM control circuit is able to adjust the duty ratio linearly from 0 to 100%. An enable function, an over current protection function is built inside. When short protection function happens, the operation frequency will be reduced from 300KHz to 60KHz. An internal compensation block is built in to minimize external component count.

TO263-5L

Figure1. Package Type of XL4005

[Enlace]

Anexo 35

Data sheet Arduino Nano

Arduino® Nano

Product Reference Manual
SKU: A000005

Description

Arduino® Nano is an intelligent development board designed for building faster prototypes with the smallest dimension. Arduino Nano being the oldest member of the Nano family, provides enough interfaces for your breadboard-friendly applications. At the heart of the board is **ATmega328 microcontroller** clocked at a frequency of 16 MHz featuring more or less the same functionalities as the Arduino Duemilanove. The board offers 20 digital input/output pins, 8 analog pins, and a mini-USB port.

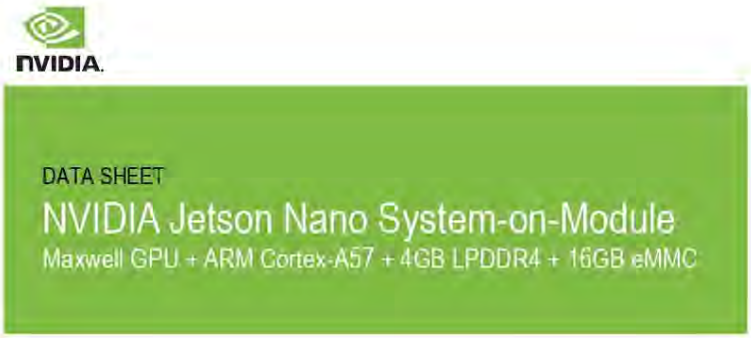
Target Areas

Maker, Security, Environmental, Robotics and Control Systems

[Enlace]

Anexo 36

Data sheet Jetson Nano



Maxwell GPU ◊

128-core GPU | End-to-end lossless compression | Tile Caching | OpenGL® 4.6 | OpenGL ES 3.2 | Vulkan™ 1.1 | CUDA® | OpenGL ES Shader Performance (up to): 512 GFLOPS (FP16)
Maximum Operating Frequency: 921MHz

CPU

ARM® Cortex® -A57 MPCore (Quad-Core) Processor with NEON Technology | L1 Cache: 48KB L1 instruction cache (I-cache) per core; 32KB L1 data cache (D-cache) per core | L2 Unified Cache: 2MB | Maximum Operating Frequency: 1.43GHz

Audio

Industry standard High Definition Audio (HDA) controller provides a multichannel audio path to the HDMI interface.

Memory

Dual Channel | System MMU | Memory Type: 4ch x 16-bit LPDDR4 | Maximum Memory Bus Frequency: 1600MHz | Peak Bandwidth: 25.6 GB/s | Memory Capacity: 4GB

Storage

eMMC 5.1 Flash Storage | Bus Width: 8-bit | Maximum Bus Frequency: 200MHz (HS400) | Storage Capacity: 16GB

Boot Sources

eMMC and USB (recovery mode)

Networking

10/100/1000 BASE-T Ethernet | Media Access Controller (MAC)

Imaging

Dedicated RAW to YUV processing engines process up to 1400Mpix/s (up to 24MP sensor) | MIPI CSI 2.0 up to 1.5Gbps (per lane) | Support for x4 and x2 configurations (up to four active streams).

Operating Requirements

Temperature Range (T_j): -25 – 97C* | Module Power: 5 – 10W | Power Input: 5.0V

Display Controller

Two independent display controllers support DSI, HDMI, DP, eDP:

- MIPI-DSI (1.5Gbps/lane): Single x2 lane | Maximum Resolution: 1920x960 at 60Hz (up to 24bpp)
- HDMI 2.0a/b (up to 6Gbps) | DP 1.2a (HBR2 5.4 Gbps) | eDP 1.4 (HBR2 5.4Gbps) | Maximum Resolution (DP/eDP/HDMI): 3840 x 2160 at 60Hz (up to 24bpp)

Clocks

System clock: 38.4MHz | Sleep clock: 32.768kHz | Dynamic clock scaling and clock source selection

Multi-Stream HD Video and JPEG

Video Decode

- H.265 (Main, Main 10): 2160p 60fps | 1080p 240fps
- H.264 (BP/MP/HP/Stereo SEI half-res): 2160p 60fps | 1080p 240fps
- H.264 (MVC Stereo per view): 2160p 30fps | 1080p 120fps
- VP9 (Profile 0, 8-bit): 2160p 60fps | 1080p 240fps
- VP8: 2160p 60fps | 1080p 240fps
- VC-1 (Simple, Main, Advanced): 1080p 120fps | 1080i 240fps
- MPEG-2 (Main): 2160p 60fps | 1080p 240fps | 1080i 240fps

Video Encode

- H.265:2160p 30fps | 1080p 120fps
- H.264 (BP/MP/HP): 2160p 30fps | 1080p 120fps
- H.264 (MVC Stereo per view): 1440p 30fps | 1080p 60fps
- VP8: 2160p 30fps | 1080p 120fps

JPEG (Decode and Encode): 600 MP/s

Peripheral Interfaces

xHCI host controller with integrated PHY: 1 x USB 3.0, 3 x USB 2.0 | USB 3.0 device controller with integrated PHY | EHCI controller with embedded hub for USB 2.0 | 4-lane PCIe: one x1/2/4 controller | single SD/MMC controller (supporting SDIO 4.0, SD HOST 4.0) | 3 x UART | 2 x SPI | 4 x I2C | 2 x I2S: support I2S, RJM, LJM, PCM, TDM (multi-slot mode) | GPIOs

Mechanical

Module Size: 69.6 mm x 45 mm | PCB: 8L HDI | Connector: 260 pin SO-DIMM

Note: Refer to the software release feature list for current software support; all features may not be available for a particular OS.

◊ Product is based on a published Khronos Specification and is expected to pass the Khronos Conformance Process. Current conformance status can be found at www.khronos.org/conformance.

* See the *Jetson Nano Thermal Design Guide* for details. Listed temperature range is based on module T_j characterization.

Enlace

Anexo 37

Data sheet IMU

MPU-6000 and MPU-6050
Product Specification
Revision 3.3

[Enlace]

Anexo 38

Data sheet Modulo Bluetooth

1. Product's picture

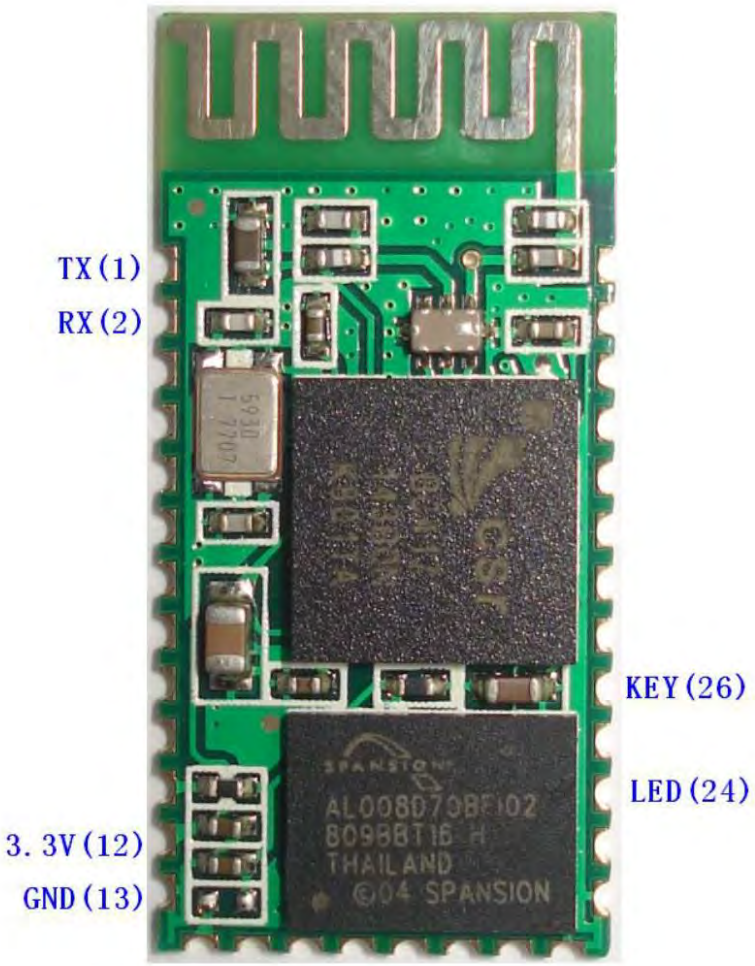


Figure 1 A Bluetooth module

www.wavesen.com Phone: 020-84083341 Fax: 020-84332079 QQ:1043073574
Address: Room 527, No.13, Jiangong Road, Tianhe software park, Tianhe district, Guangzhou Post: 510660
Technology consultant: support@wavesen.com Business consultant:sales@wavesen.com
Complaint and suggestion: sunbirdit@hotmail.com

Enlace

Anexo 39

Data sheet Sensor Ultrasonido

HC-SR04 Ultrasonic Sensor

Elijah J. Morgan

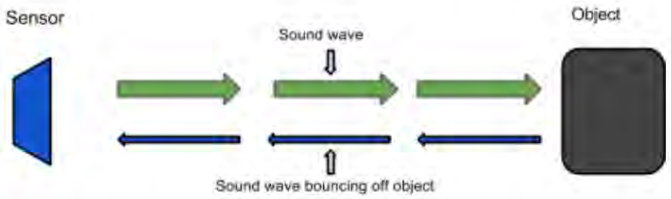
Nov. 16 2014

The purpose of this file is to explain how the HC-SR04 works. It will give a brief explanation of how ultrasonic sensors work in general. It will also explain how to wire the sensor up to a microcontroller and how to take/interpret readings. It will also discuss some sources of errors and bad readings.

1. **How Ultrasonic Sensors Work**
2. **HC-SR04 Specifications**
3. **Timing chart, Pin explanations and Taking Distance Measurements**
4. **Wiring HC-SR04 with a microcontroller**
5. **Errors and Bad Readings**

1. How Ultrasonic Sensors Work

Ultrasonic sensors use sound to determine the distance between the sensor and the closest object in its path. How do ultrasonic sensors do this? Ultrasonic sensors are essentially sound sensors, but they operate at a frequency above human hearing.



The sensor sends out a sound wave at a specific frequency. It then listens for that specific sound wave to bounce off of an object and come back (Figure 1). The sensor keeps track of the time between sending the sound wave and the sound wave returning. If you know how fast something is going and how long it is traveling you can find the distance traveled with equation 1.

Equation 1. $d = v \times t$

The speed of sound can be calculated based on the a variety of atmospheric conditions, including temperature, humidity and pressure. Actually calculating the distance will be shown later on in this document.

It should be noted that ultrasonic sensors have a cone of detection, the angle of this cone varies with distance, Figure 2 show this relation. The ability of a sensor to

Enlace

Anexo 40

Data sheet Motor Amarillo

verical
An Arrow Company

3777
ADAFRUIT

Buy Now

Looking for a discount?
Check out our current promotions!

Give us a call
1-855-837-4225
International: 1-415-281-3866

Email Us
Sales and New Orders: sales@verical.com
Order Support: support@verical.com
Suppliers: Visit our seller page

Company Address
Arrow Electronics, Inc
9201 East Dry Creek Road
Centennial, CO 80112

[Enlace]

Anexo 41

Data Sheet Driver de Motores L298N

Hoja de datos del controlador de motores **L298N** (Rojo)

Controlador de Motores L298N (Rojo)

Descripción:

La base de este módulo es el circuito integrado L298N, el cual es un doble puente H.
Este es capaz de manejar niveles altos voltaje y de corriente, además de estar diseñado para soportar cargas inductivas tales como relés, solenoides, motores de corriente continua y motores paso a paso. Este tipo de cargas las soporta gracias a unos diodos, los cuales absorben las corrientes inversas que producen estas cargas.

Dispone de dos puentes para habilitar o deshabilitar las salidas independientemente de las señales de entrada. También incorpora un interruptor para la conexión y desconexión de toda la placa.

Otra de las cosas muy útiles de la que dispone, es un regulador 7805, el cual, estabiliza la tensión de entrada de la placa a 5V y la entrega por una salida.

Descripción de las partes del Driver:

-1: Conector para la salida 3 y 4.

-2: Conector para la salida 1 y 2.

-3: Driver L298N.

-4: Salida de 5V.

-5: GND

-6: VCC

www.leantec.es store@leantec.es

[Enlace]

Anexo 42

Data Sheet Sensor Infrarrojo

http://www.agelectronica.com

OKY3127: SENSOR DE PROXIMIDAD INFRARROJO FC-51

OKYSTAR

Descripción

Este módulo fotoelectrico basa su principio de funcionamiento en un transmisor y receptor IR para identificar obstáculos delante del sensor en un rango de 2 a 30cm de distancia. El módulo cuenta con un potenciómetro que permite al usuario ajustar el rango de detección

El sensor tiene una respuesta muy buena y estable incluso con luz ambiente o en completa oscuridad, se puede interconectar con Arduino, Raspberri Pi o cualquier microcontrolador que tenga un nivel de tensión de IO de 3.3V a 5V.

Características

- Circuito de detección basado en el LM393 y tecnología IR
- Comparador estable y preciso
- Sensibilidad ajustable
- LED indicador de alimentación
- LED indicador de obtaculo
- Orificio de montaje, diametro 3mm
- Excelente desempeño en luz ambiente
- Dimensiones aproximadas 31*14mm

Especificaciones

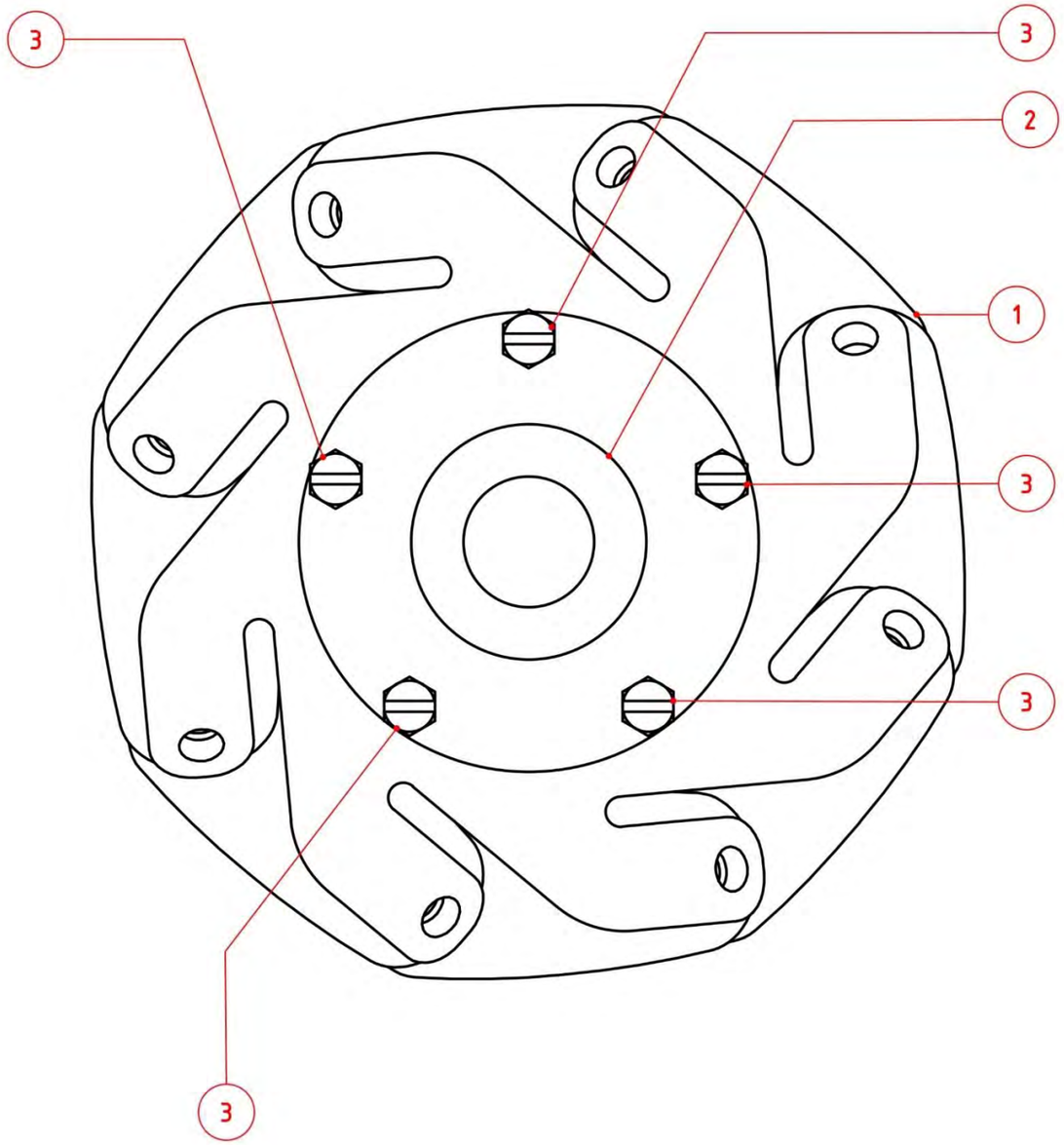
Voltaje de trabajo	3V~6V
Angulo de cobertura	35°
Rango de detección	2~30cm
Consumo de corriente	23mA (3.3V)
	43mA 85V)

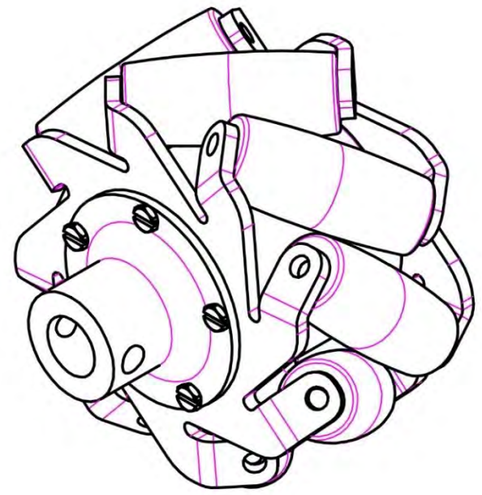
AG Electrónica ¿Qué vamos a innovar hoy?

http://www.agelectronica.com

[Enlace]

Anexo 43



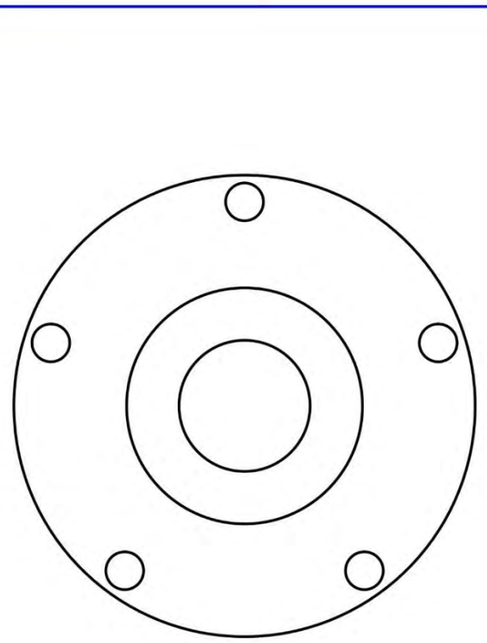


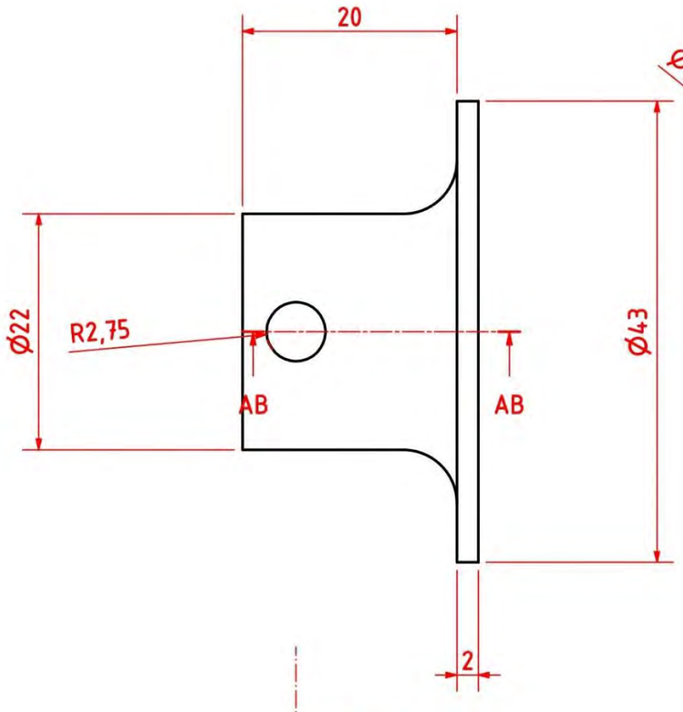
3	5	ANSI B18.6.4 - 4-24 - 0.375, SRHHTSTBI		ANSI B18.6.4	Steel, Mild
2	1	Acople de pla sin prisonero			Genérico
1	1	Rueda Mecanum			
IT	CANT	NOMBRE	DESCR.	NORMA	MATERIAL

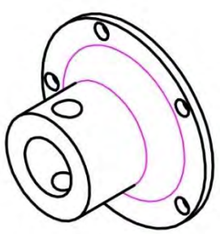
PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA – ING. MECATRÓNICA

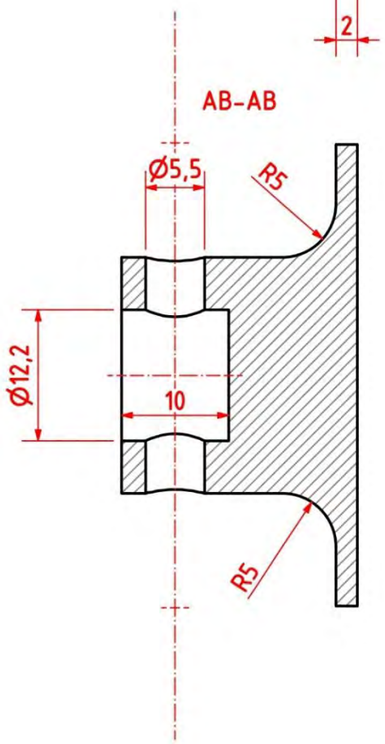
MÉTODO DE PROYECCIÓN	1MTR02 - 0000 - HORARIO10M4 RUEDA MECANUM CON ACOPLE	ESCALA 2:1
20191965	RUIZ FIGUEROA, SEBASTIAN	FECHA: 31.10.2023
REVISADO POR:		LÁMINA: A3

Anexo 44





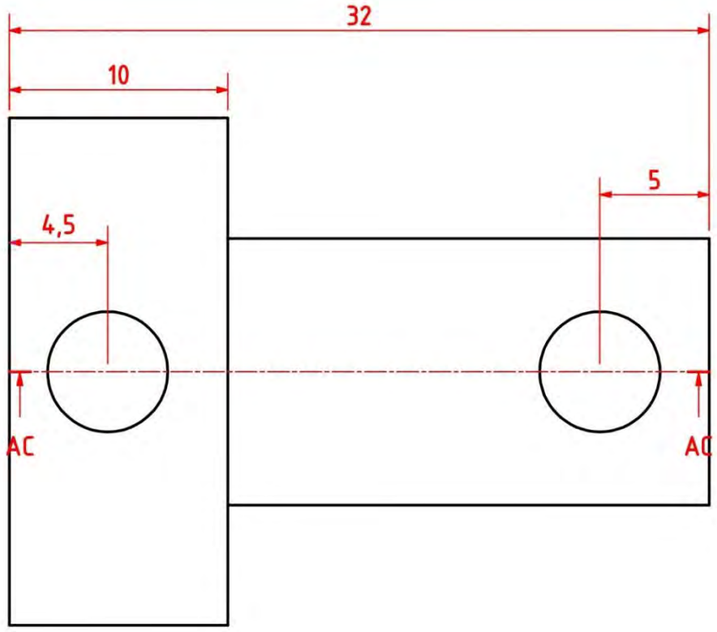


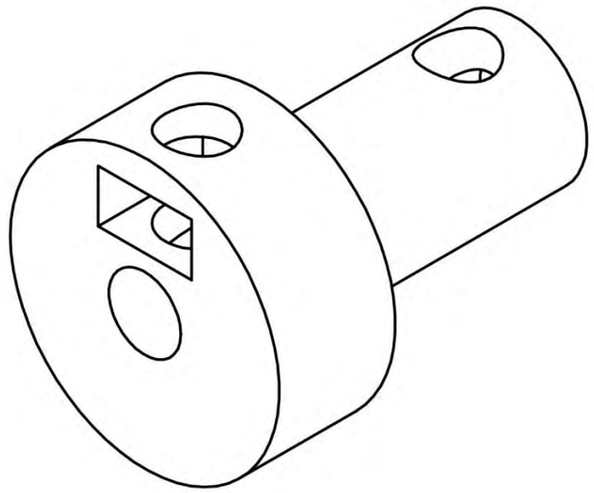


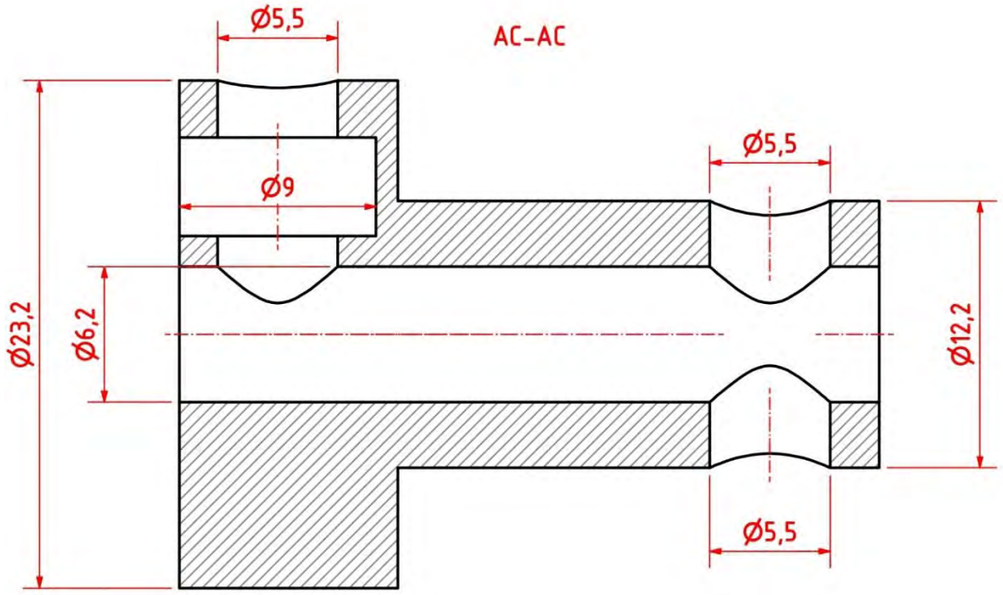
ACABADO SUPERFICIAL	TOLERANCIA GENERAL	MATERIAL
	-	PLA

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ FACULTAD DE CIENCIAS E INGENIERÍA – ING. MECATRÓNICA		
MÉTODO DE PROYECCIÓN	1MTR02 - 0000 - HORARIO10M4 RUEDA MECANUM CON ACOPLE	ESCALA 2:1
20191965	RUIZ FIGUEROA, SEBASTIAN	FECHA: 31.10.2023
REVISADO POR:		LÁMINA: A3

Anexo 45



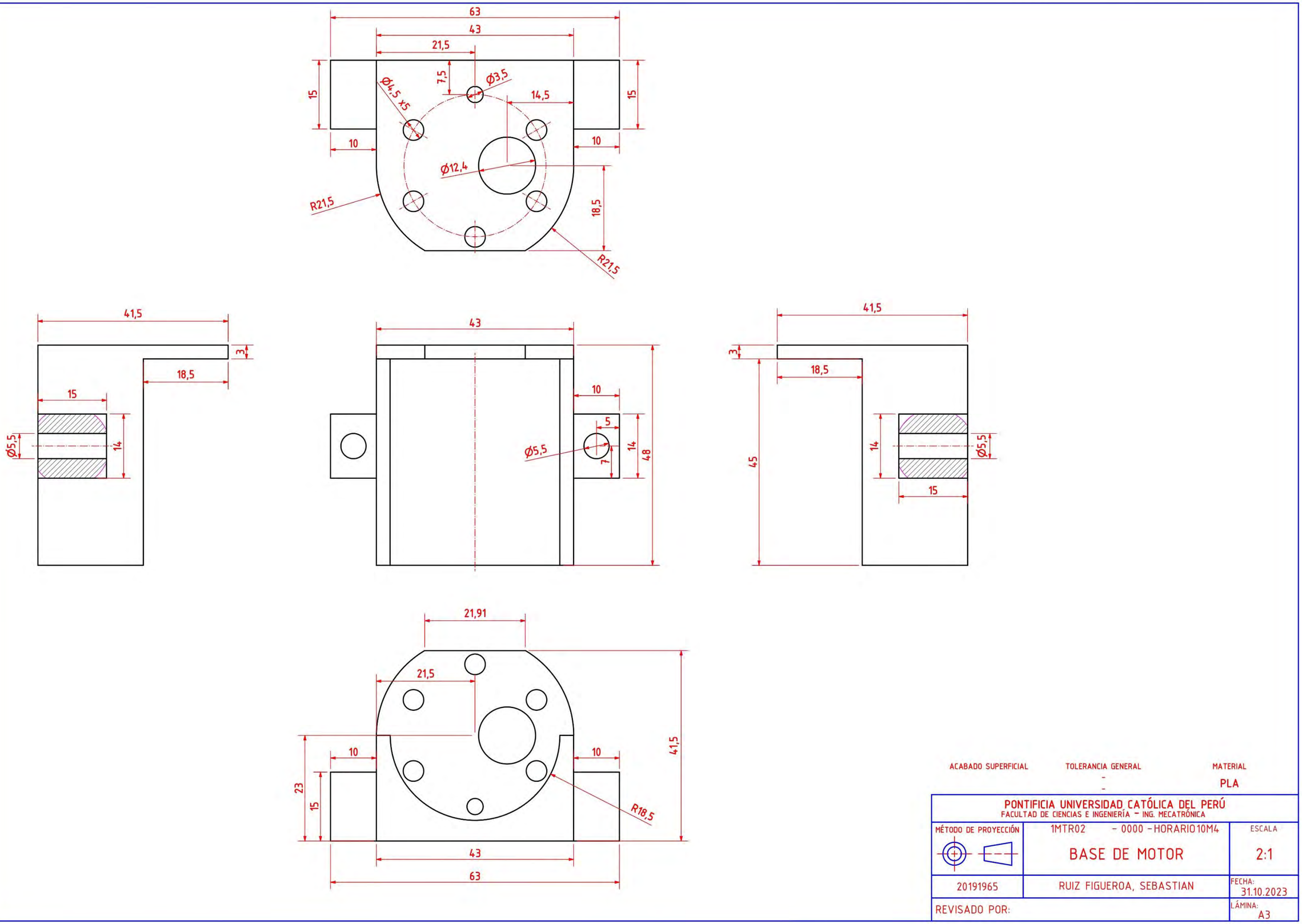




ACABADO SUPERFICIAL	TOLERANCIA GENERAL	MATERIAL
	-	PLA

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ FACULTAD DE CIENCIAS E INGENIERÍA – ING. MECATRÓNICA		
MÉTODO DE PROYECCIÓN	1MTR02 - 0000 - HORARIO10M4 ACOPLE MOTOR RUEDA	ESCALA 4:1
20191965	RUIZ FIGUEROA, SEBASTIAN	FECHA: 31.10.2023
REVISADO POR:		LÁMINA: A3

Anexo 46

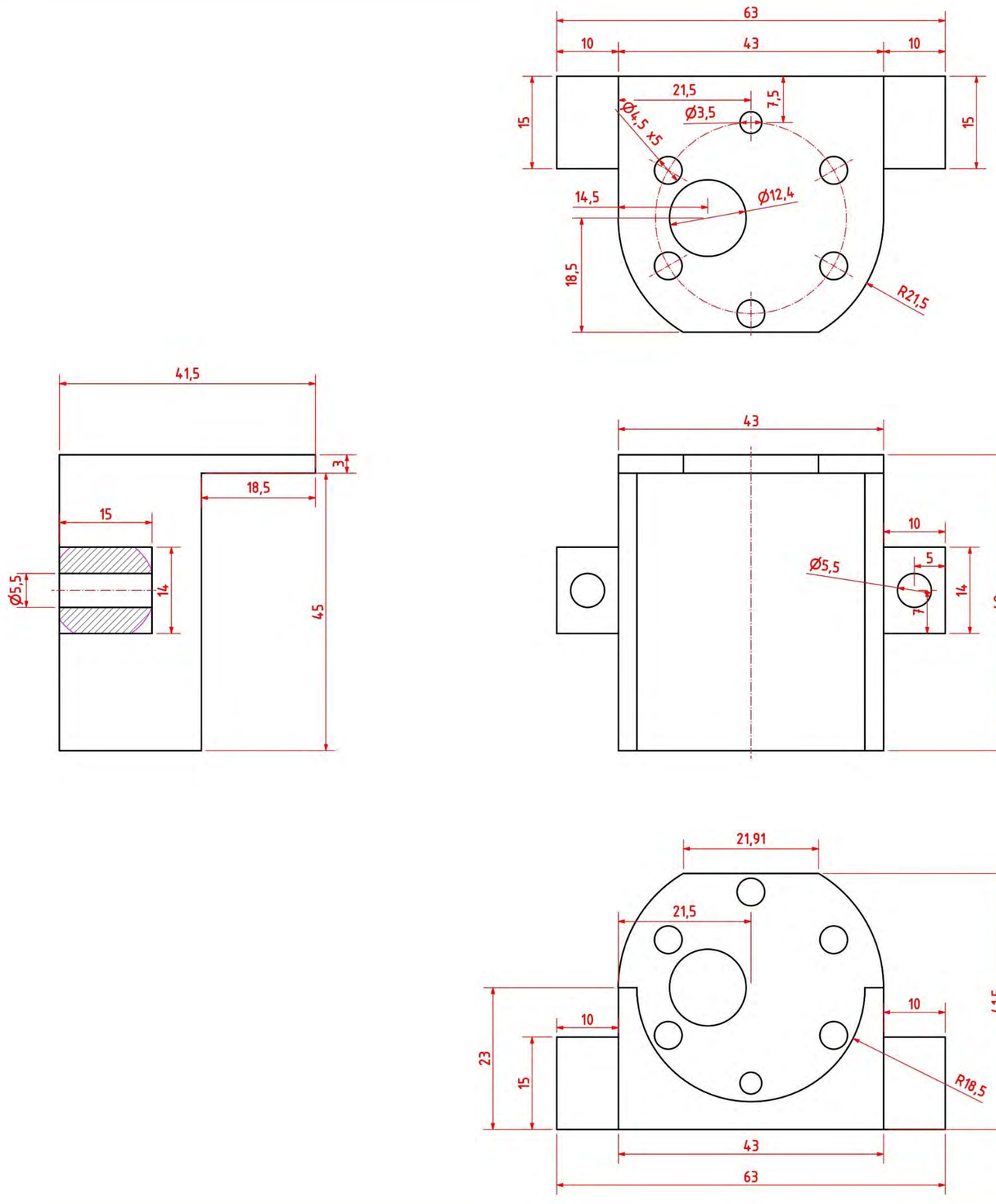


ACABADO SUPERFICIAL	TOLERANCIA GENERAL	MATERIAL
	-	PLA

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA – ING. MECATRÓNICA

MÉTODO DE PROYECCIÓN	1MTR02 – 0000 – HORARIO10M4	ESCALA
	BASE DE MOTOR	2:1
20191965	RUIZ FIGUEROA, SEBASTIAN	FECHA: 31.10.2023
REVISADO POR:		LÁMINA: A3

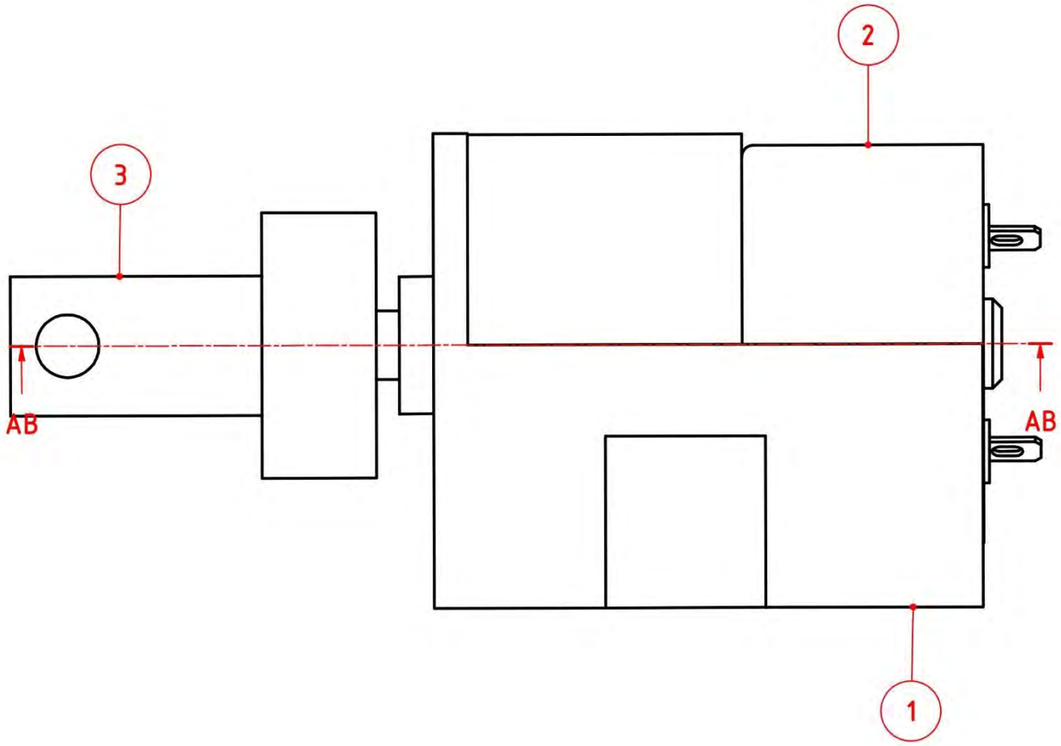
Anexo 47

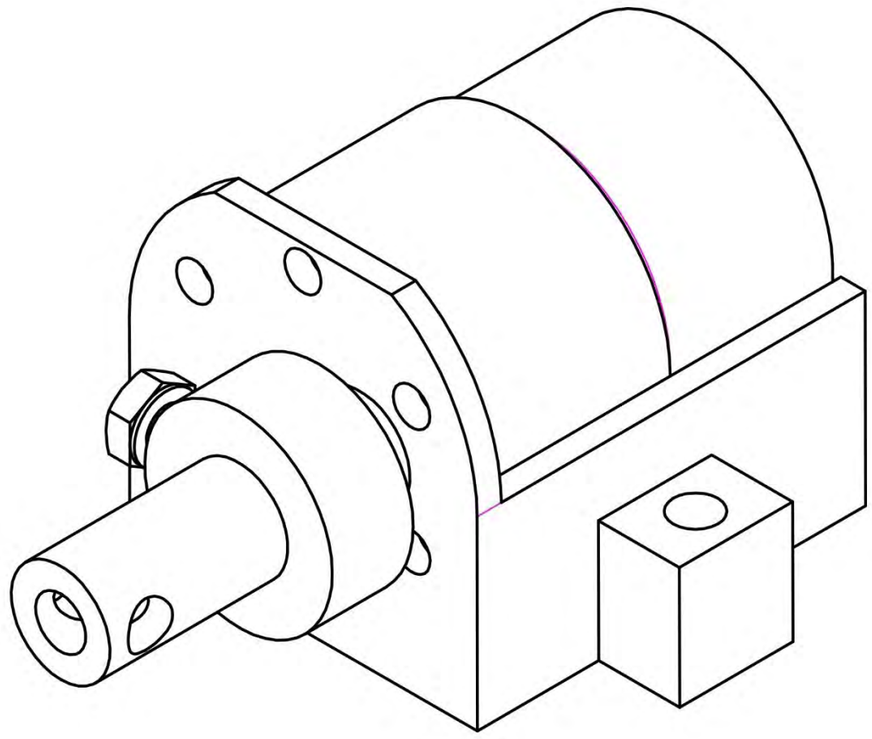


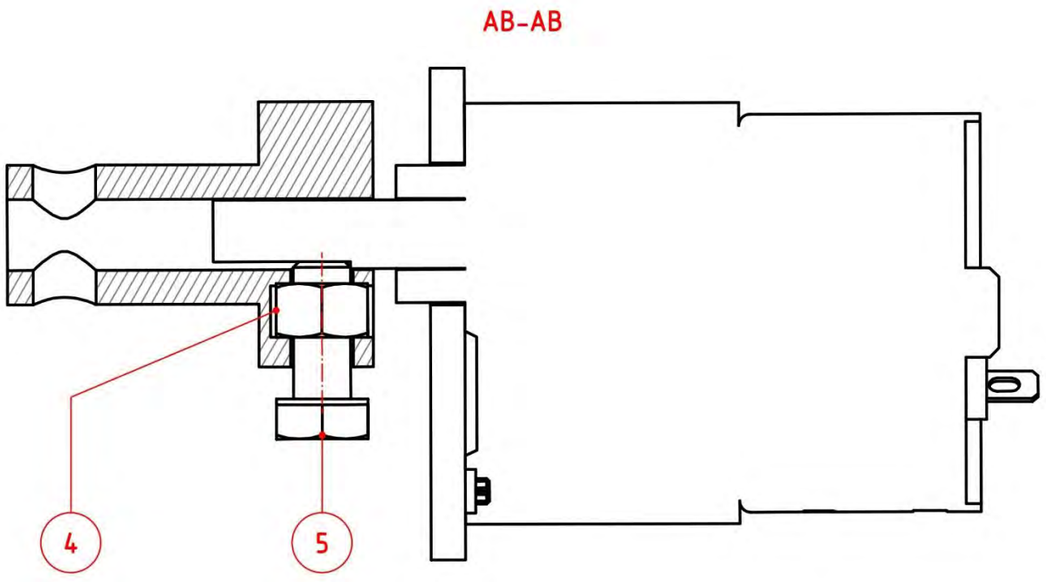
ACABADO SUPERFICIAL	TOLERANCIA GENERAL	MATERIAL
	-	PLA

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ FACULTAD DE CIENCIAS E INGENIERÍA – ING. MECATRÓNICA		
MÉTODO DE PROYECCIÓN	1MTR02 - 0000 - HORARIO10M4 BASE DE MOTOR INVERSO	ESCALA 2:1
20191965	RUIZ FIGUEROA, SEBASTIAN	FECHA: 31.10.2023
REVISADO POR:		LÁMINA: A3

Anexo 48





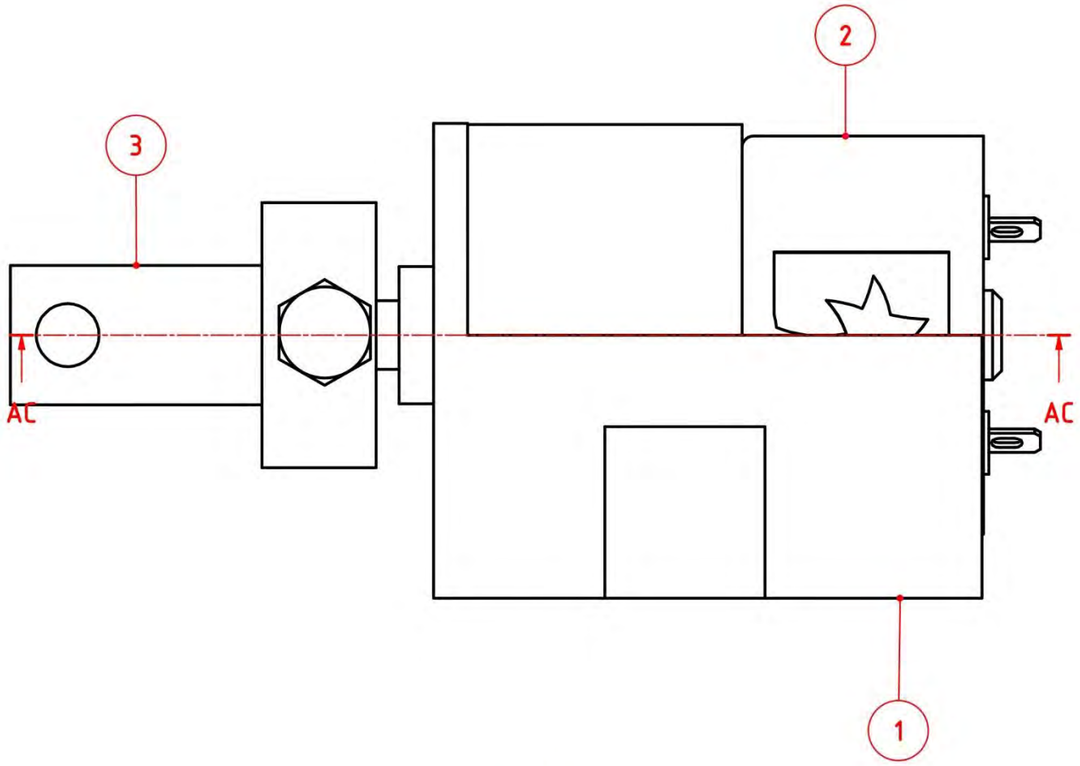


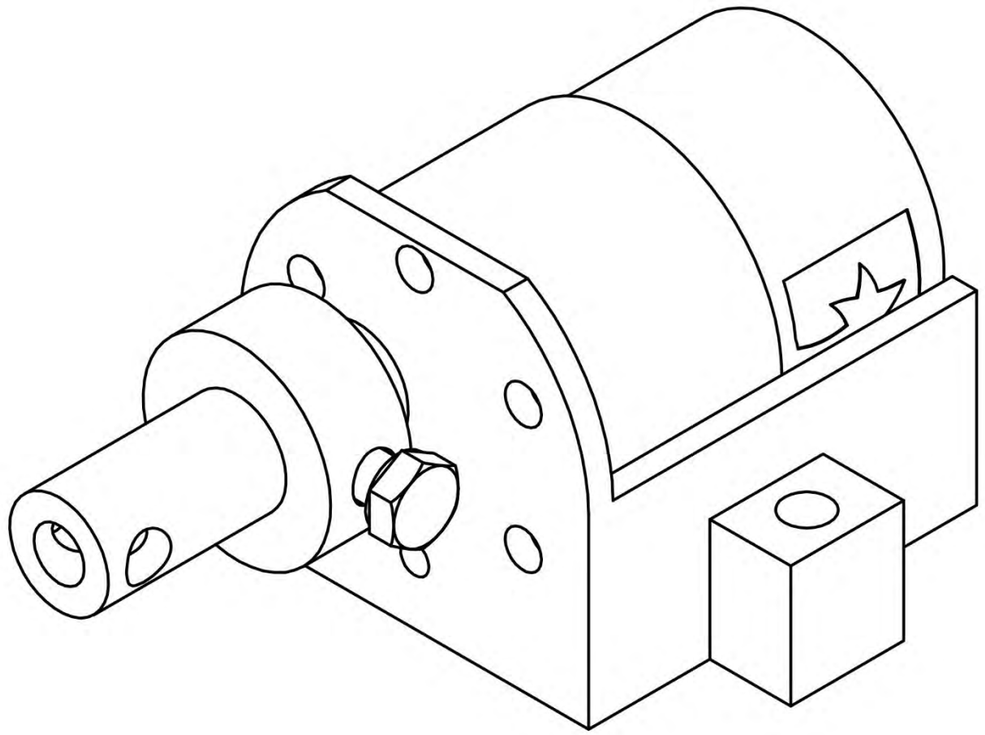
IT	CANT	NOMBRE	DESCR.	NORMA	MATERIAL
5	1	AS 1110 - M5 x 12		AS 1110	Steel, Mild
4	1	AS 1112 - M5 Type 5		AS 1112	Steel, Mild
3	1	Eje Motor Rueda			PLA
2	1	37D Gear Motor 100 RPM			
1	1	Base De Motor			PLA

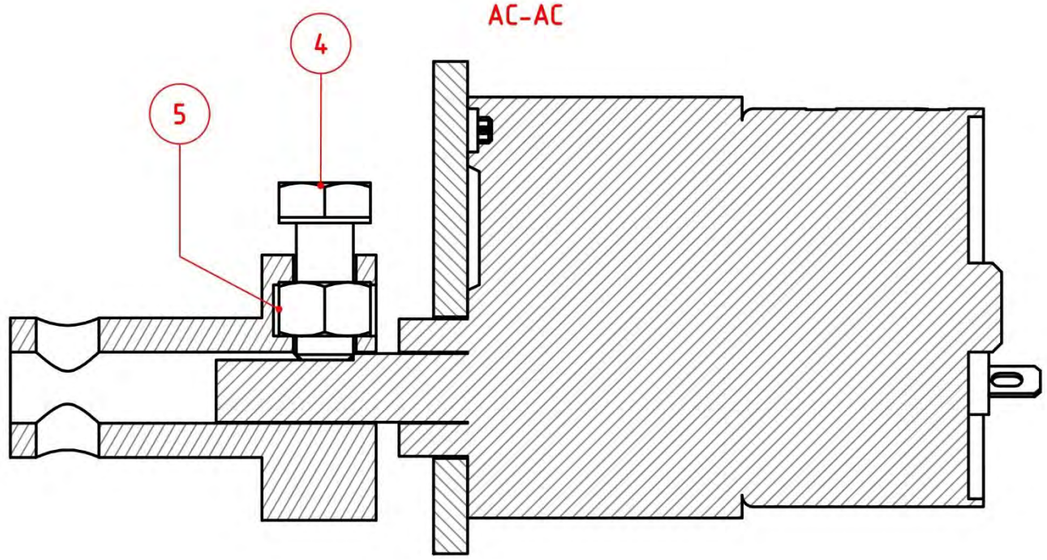
PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA – ING. MECATRÓNICA

MÉTODO DE PROYECCIÓN	1MTR02 - 0000 - HORARIO10M4 PLANO DE ENSAMBLE ACOPLE MOTOR	ESCALA 2:1
20191965	RUIZ FIGUEROA, SEBASTIAN	FECHA: 31.10.2023
REVISADO POR:		LÁMINA: A3

Anexo 49





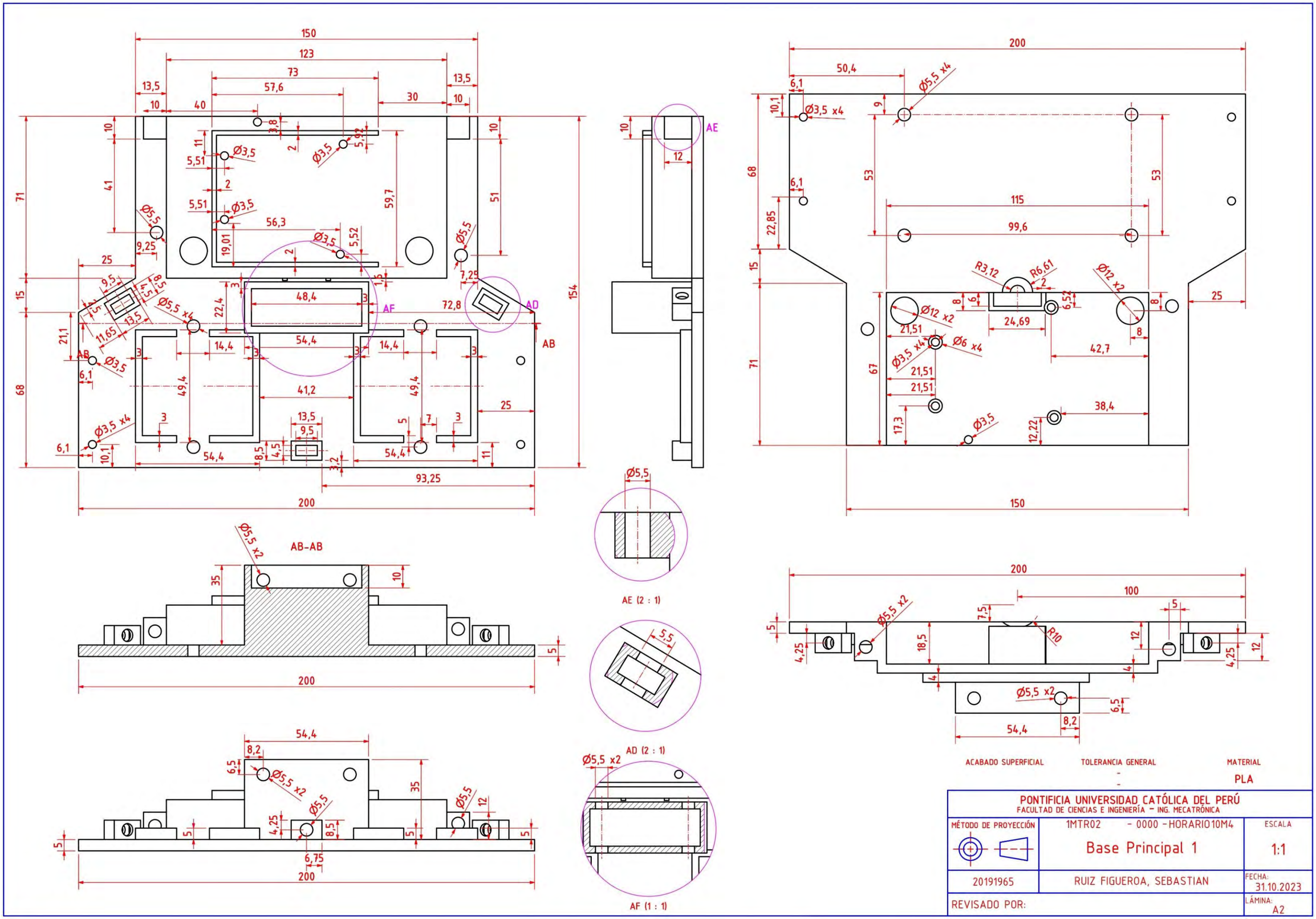


IT	CANT	NOMBRE	DESCR.	NORMA	MATERIAL
5	1	AS 1112 - M5 Type 5		AS 1112	Steel, Mild
4	1	AS 1110 - M5 x 12		AS 1110	Steel, Mild
3	1	Acople Motor Rueda Espejo			PLA
2	1	37d-gearmotor-100rpm			Generic
1	1	Base			PLA

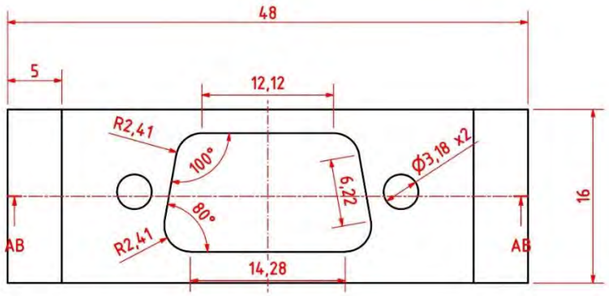
PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA – ING. MECATRÓNICA

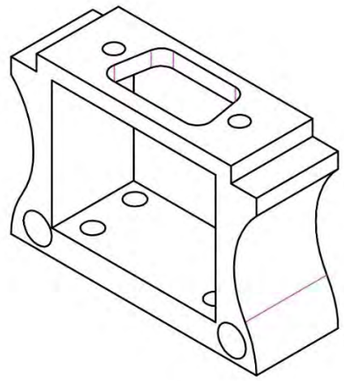
MÉTODO DE PROYECCIÓN	1MTR02 - 0000 - HORARIO10M4 PLANO DE ENSAMBLE ACOPLE MOTOR ESPEJO	ESCALA 2:1
20191965	RUIZ FIGUEROA, SEBASTIAN	FECHA: 31.10.2023
REVISADO POR:		LÁMINA: A3

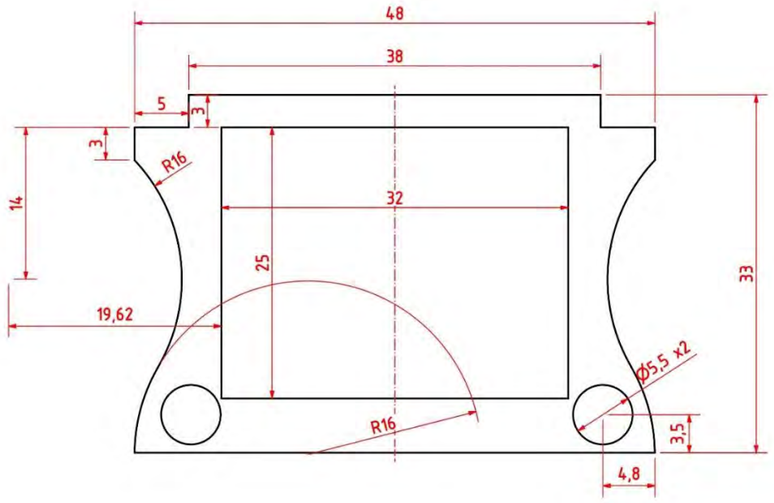
Anexo 50

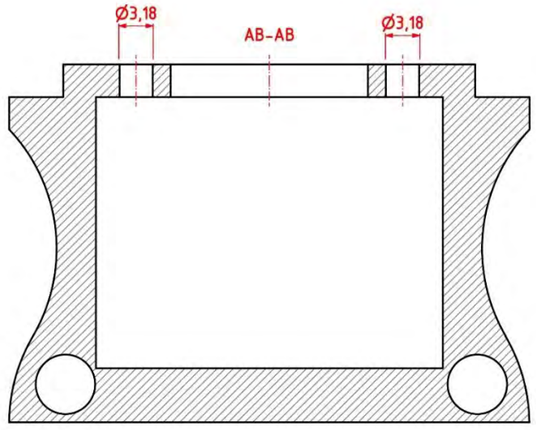

AB-AB
AE (2 : 1)
AD (2 : 1)
AF (1 : 1)
ACABADO SUPERFICIAL
TOLERANCIA GENERAL
MATERIAL
PLA
PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA – ING. MECATRÓNICA
MÉTODO DE PROYECCIÓN
1MTR02 - 0000 - HORARIO10M4
Base Principal 1
ESCALA
1:1
20191965
RUIZ FIGUEROA, SEBASTIAN
FECHA:
31.10.2023
REVISADO POR:
LÁMINA:
A2

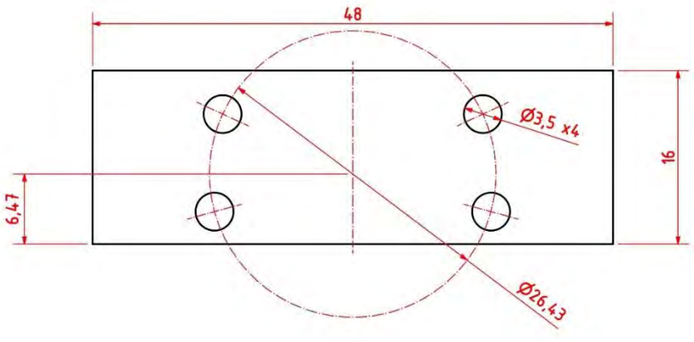
Anexo 51







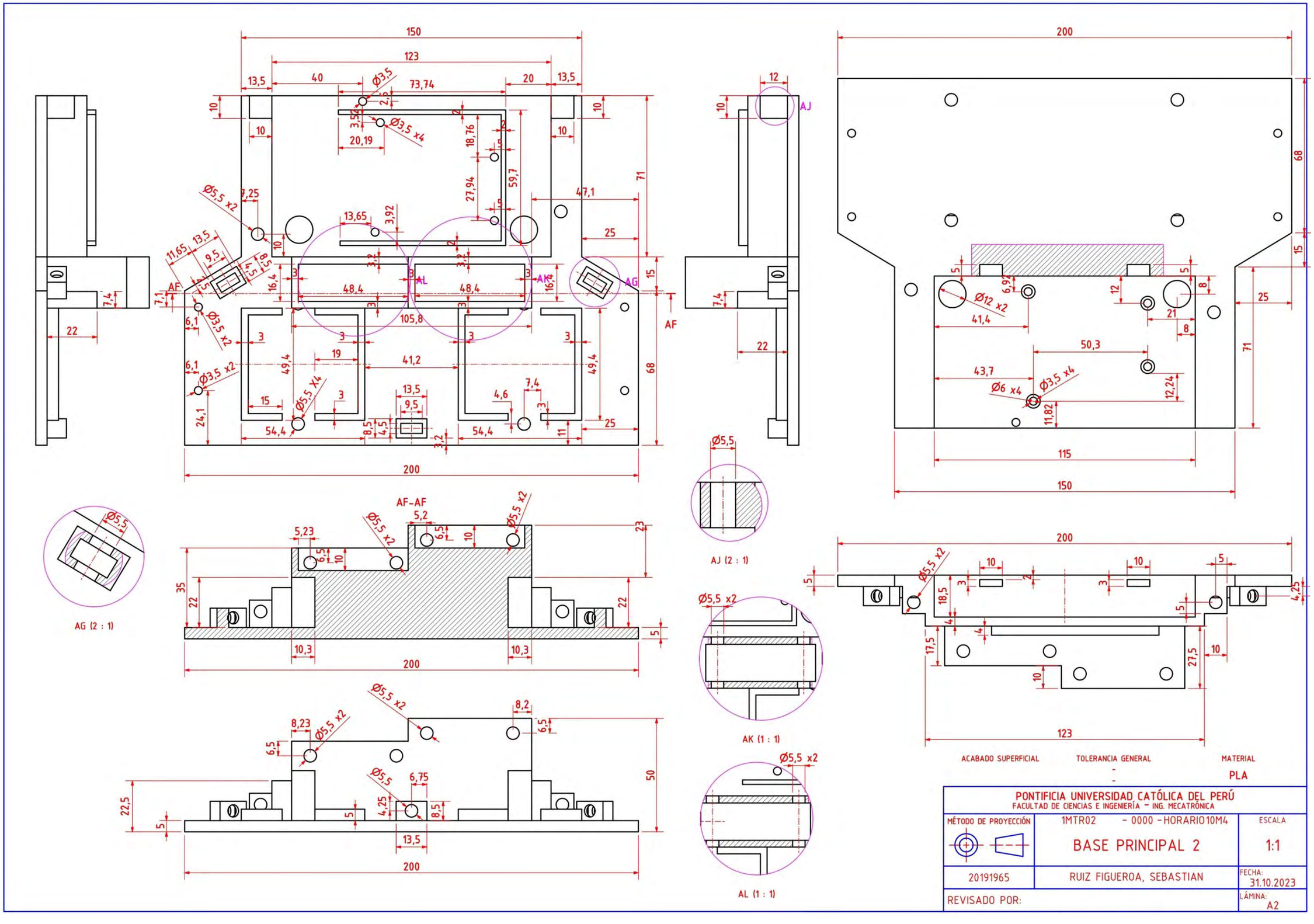




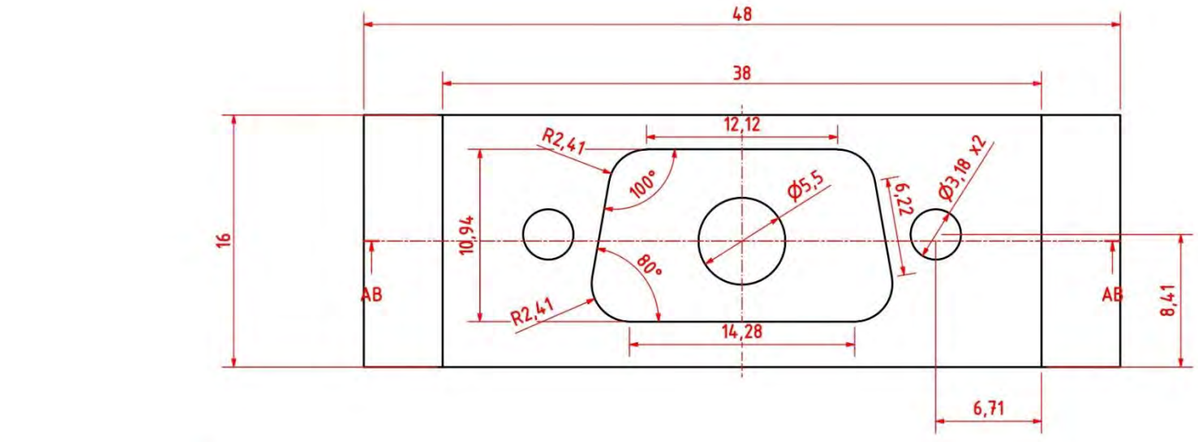
ACABADO SUPERFICIAL	TOLERANCIA GENERAL	MATERIAL
	-	PLA

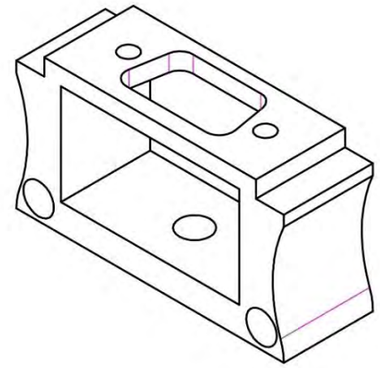
PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ FACULTAD DE CIENCIAS E INGENIERÍA – ING. MECATRÓNICA		
MÉTODO DE PROYECCIÓN	1MTR02 - 0000 - HORARIO10M4 SOPORTE DB9 ALTO	ESCALA 3:1
20191965	RUIZ FIGUEROA, SEBASTIAN	FECHA: 31.10.2023
REVISADO POR:		LÁMINA: A3

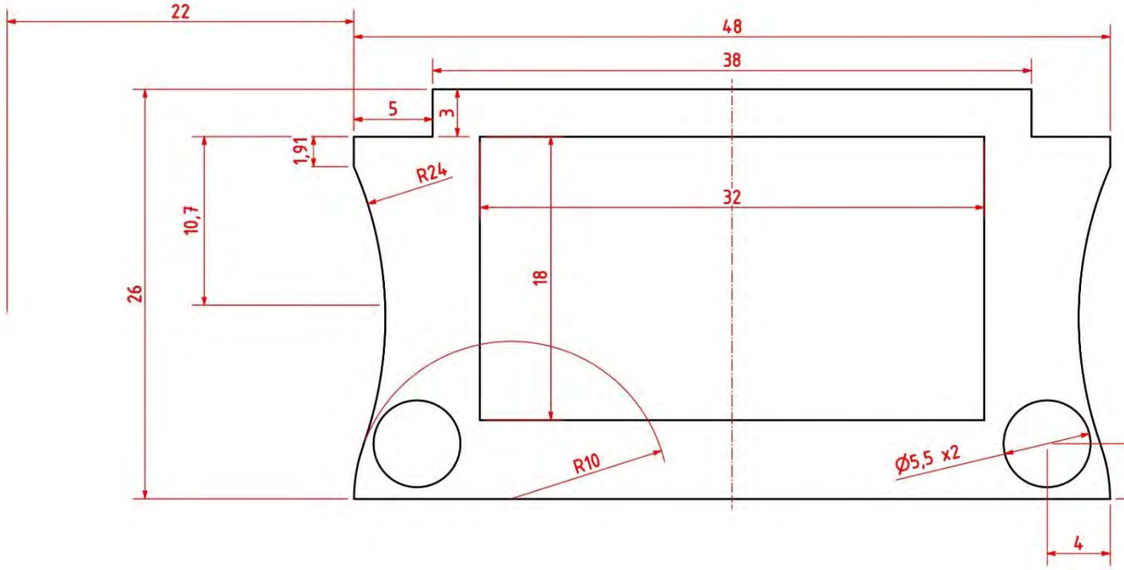
Anexo 52

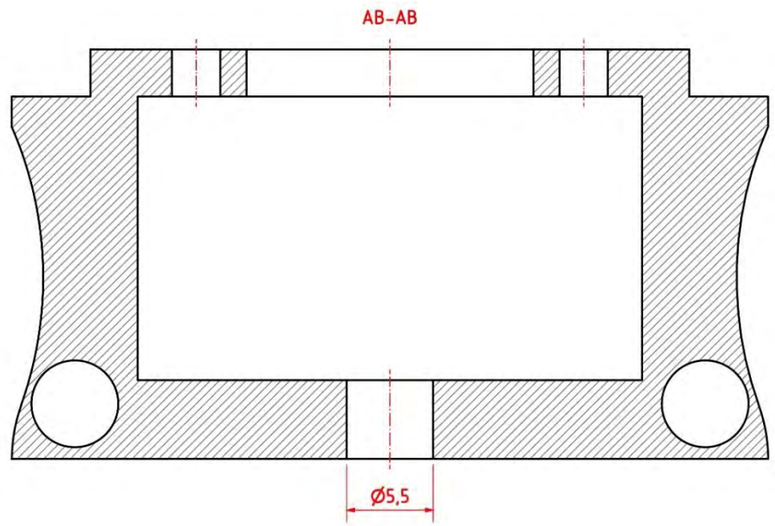

AF-AF
AG (2 : 1)
AJ (2 : 1)
AK (1 : 1)
AL (1 : 1)
ACABADO SUPERFICIAL
TOLERANCIA GENERAL
MATERIAL
PLA
PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA – ING. MECATRÓNICA
MÉTODO DE PROYECCIÓN
1MTR02 - 0000 - HORARIO10M4
BASE PRINCIPAL 2
ESCALA
1:1
20191965
RUIZ FIGUEROA, SEBASTIAN
FECHA:
31.10.2023
REVISADO POR:
LÁMINA:
A2

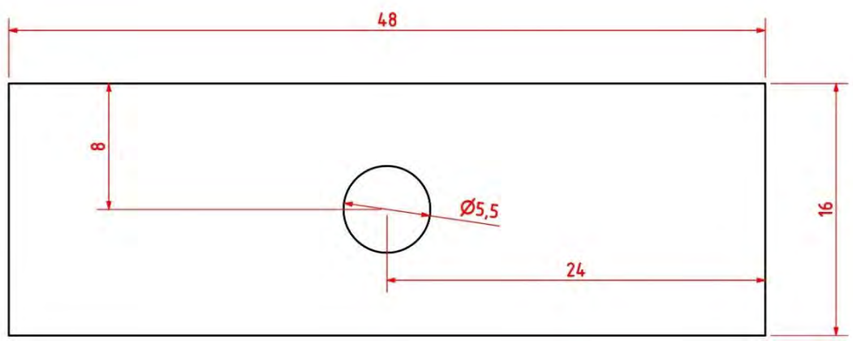
Anexo 53









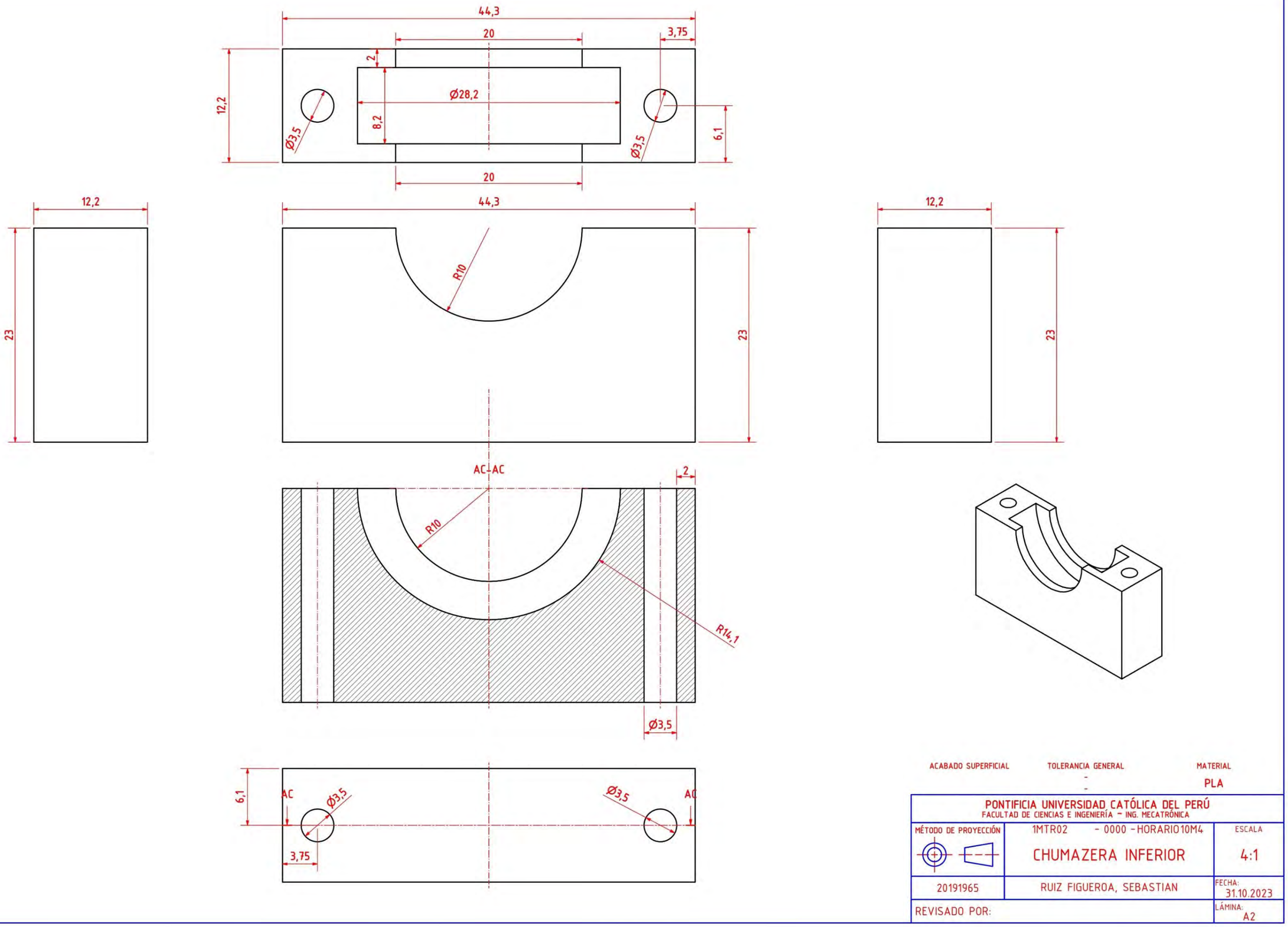


ACABADO SUPERFICIAL	TOLERANCIA GENERAL	MATERIAL
	-	PLA

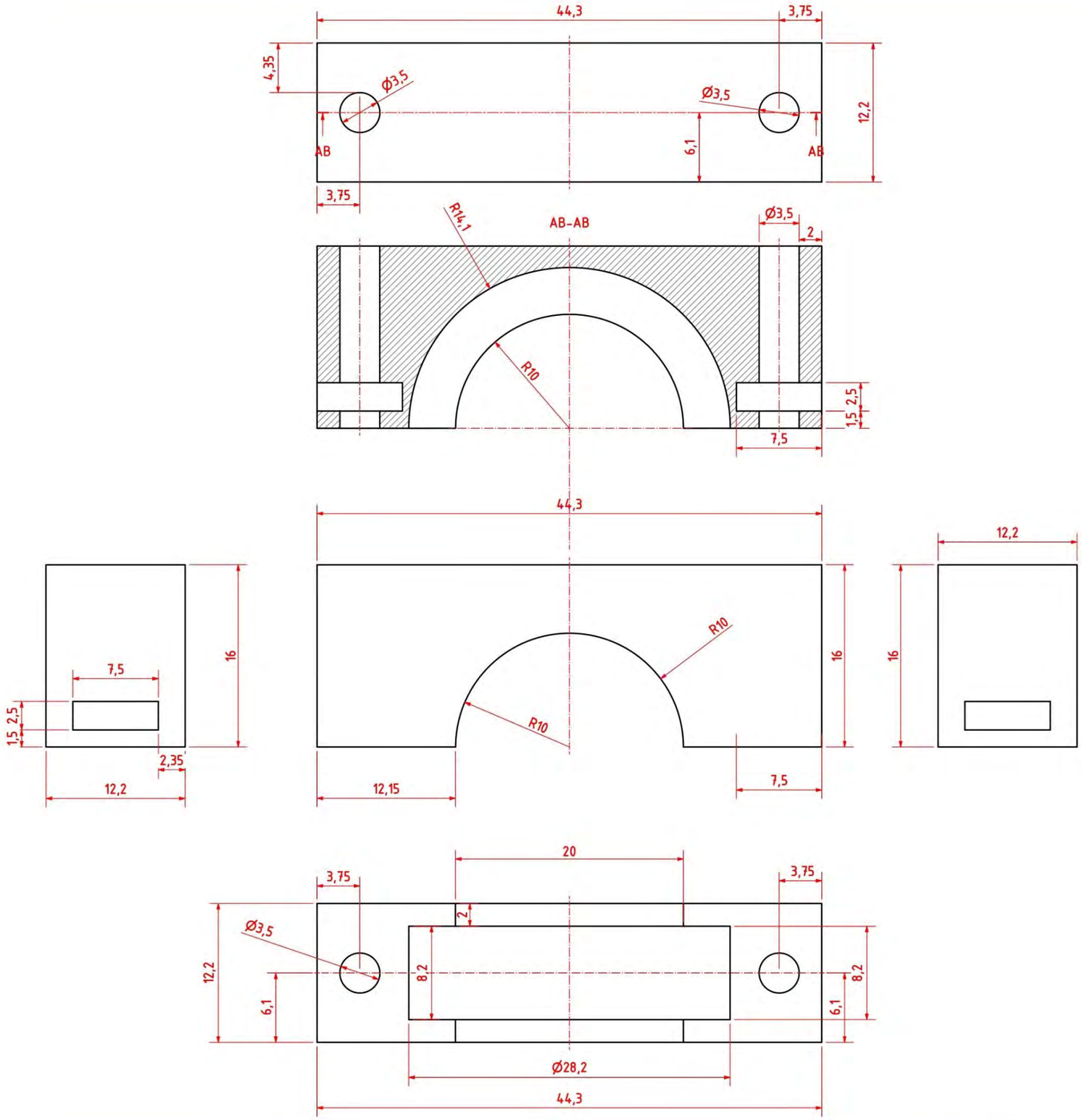
PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA – ING. MECATRÓNICA

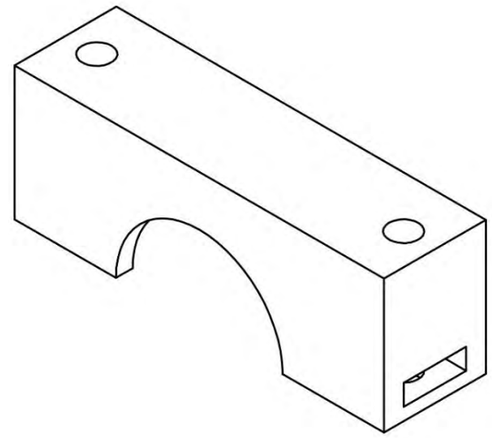
MÉTODO DE PROYECCIÓN	1MTR02 - 0000 - HORARIO10M4	ESCALA
	SOPORTE DB9 BAJO	4:1
20191965	RUIZ FIGUEROA, SEBASTIAN	FECHA: 31.10.2023
REVISADO POR:		LÁMINA: A3

Anexo 54


44,3
20
3,75
2
12,2
Ø28,2
8,2
Ø3,5
Ø3,5
6,1
20
12,2
44,3
12,2
R10
23
23
23
AC-AC
2
R10
R14,1
Ø3,5
6,1
AC
Ø3,5
Ø3,5
AC
3,75
ACABADO SUPERFICIAL
TOLERANCIA GENERAL
-
-
MATERIAL
PLA
PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA - ING. MECATRÓNICA
MÉTODO DE PROYECCIÓN
1MTR02 - 0000 - HORARIO10M4
CHUMAZERA INFERIOR
ESCALA
4:1
20191965
RUIZ FIGUEROA, SEBASTIAN
FECHA:
31.10.2023
REVISADO POR:
LÁMINA:
A2

Anexo 55



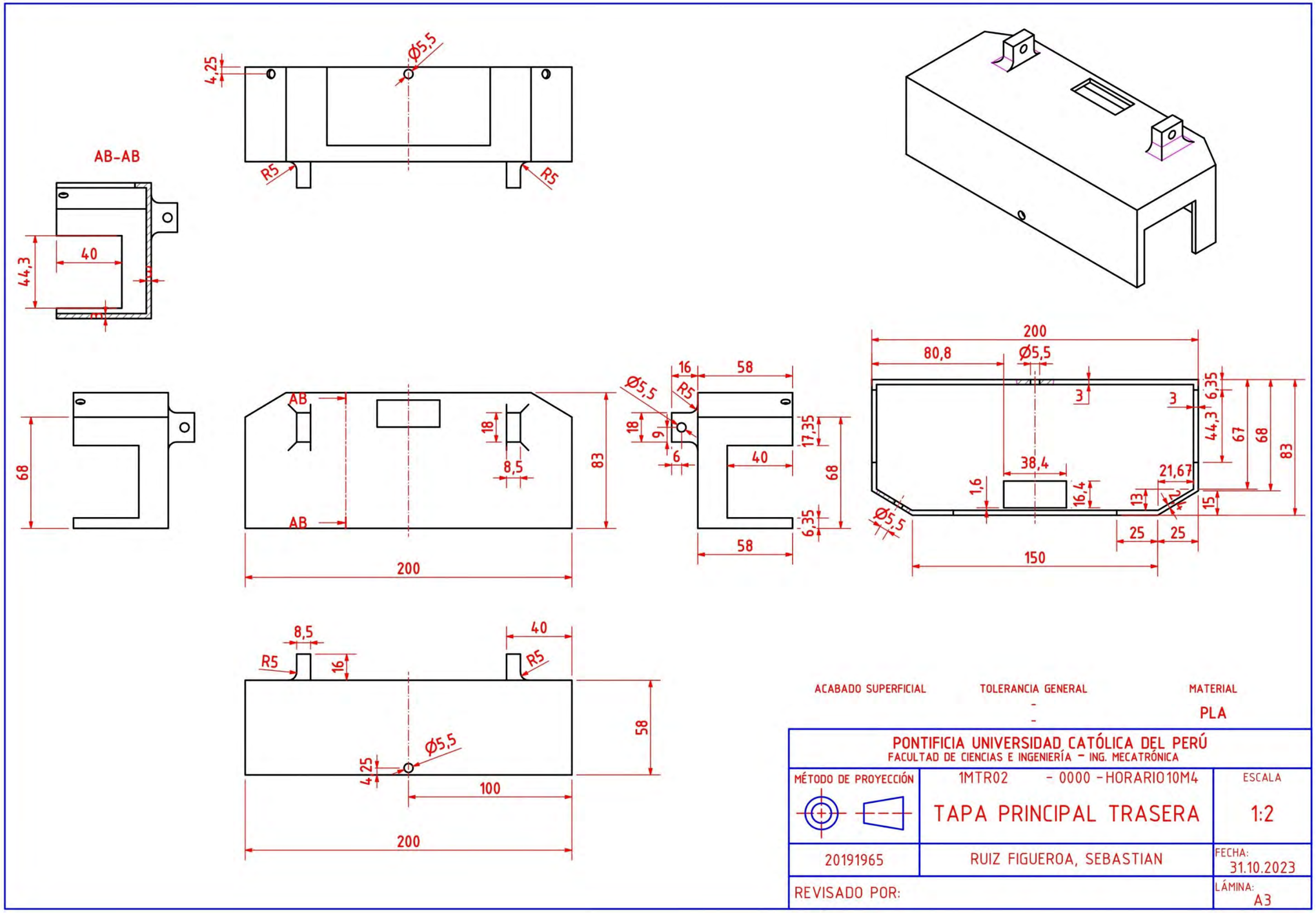


ACABADO SUPERFICIAL	TOLERANCIA GENERAL	MATERIAL
	-	PLA

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA – ING. MECATRÓNICA

MÉTODO DE PROYECCIÓN	1MTR02 – 0000 – HORARIO10M4	ESCALA
	CHUMAZERA SUPERIOR	4:1
20191965	RUIZ FIGUEROA, SEBASTIAN	FECHA: 31.10.2023
REVISADO POR:		LÁMINA: A2

Anexo 56

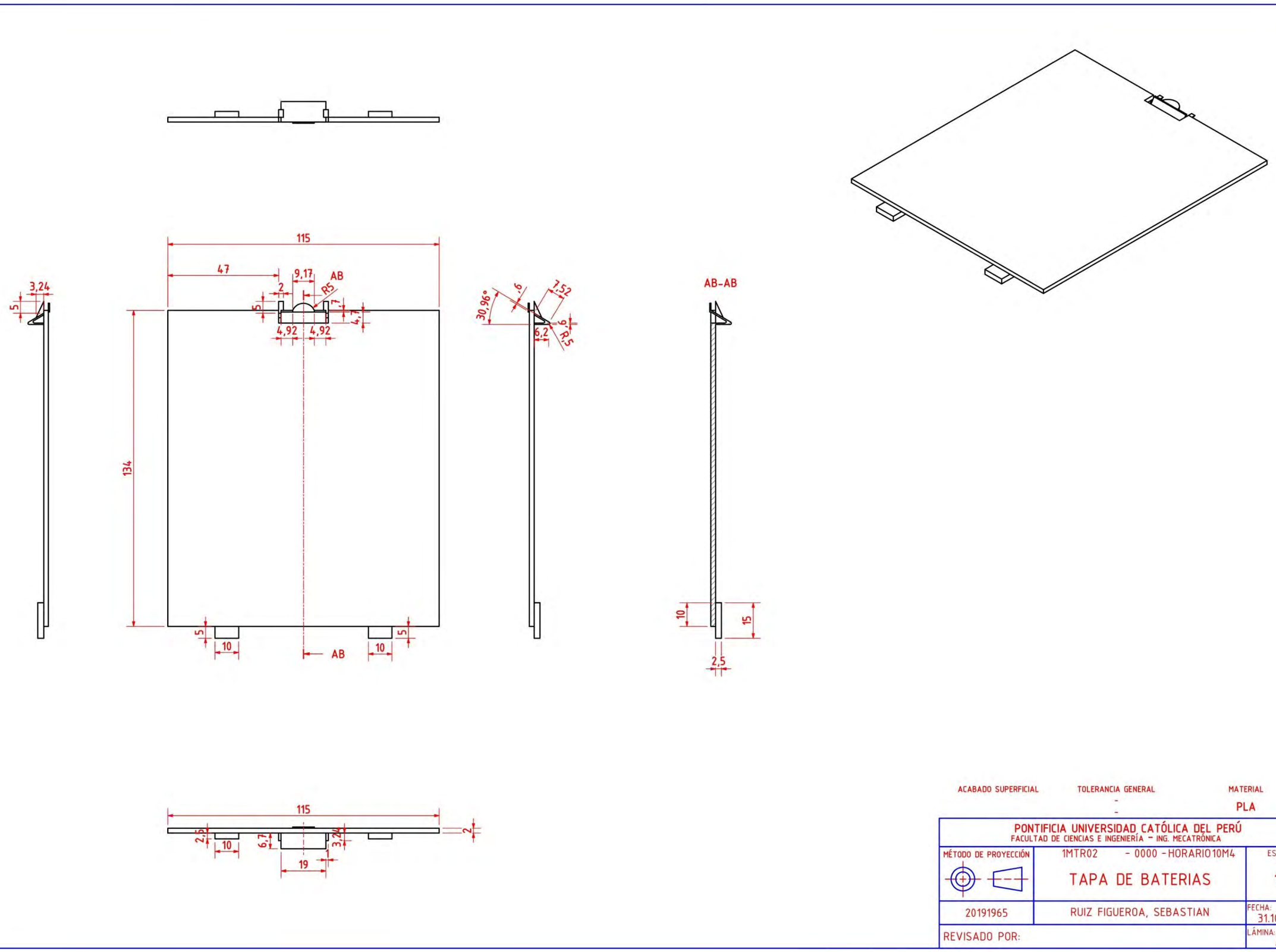


ACABADO SUPERFICIAL	TOLERANCIA GENERAL	MATERIAL
	-	PLA

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA – ING. MECATRÓNICA

MÉTODO DE PROYECCIÓN	1MTR02 - 0000 - HORARIO10M4	ESCALA
	TAPA PRINCIPAL TRASERA	1:2
20191965	RUIZ FIGUEROA, SEBASTIAN	FECHA: 31.10.2023
REVISADO POR:		LÁMINA: A3

Anexo 57

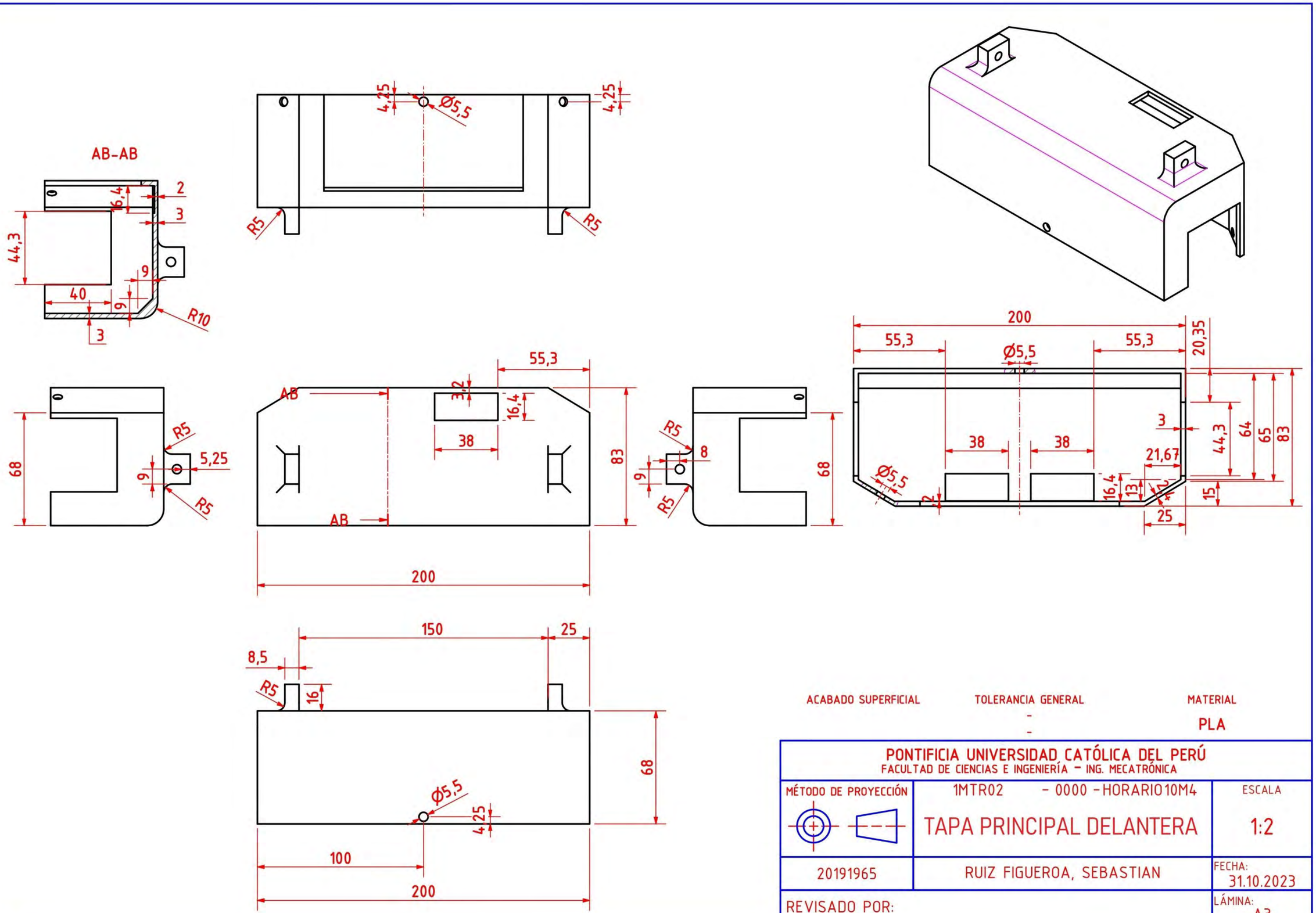


ACABADO SUPERFICIAL	TOLERANCIA GENERAL	MATERIAL
	-	PLA

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA – ING. MECATRÓNICA

MÉTODO DE PROYECCIÓN	1MTR02 - 0000 - HORARIO10M4	ESCALA
	TAPA DE BATERIAS	1:1
20191965	RUIZ FIGUEROA, SEBASTIAN	FECHA: 31.10.2023
REVISADO POR:		LÁMINA: A3

Anexo 58

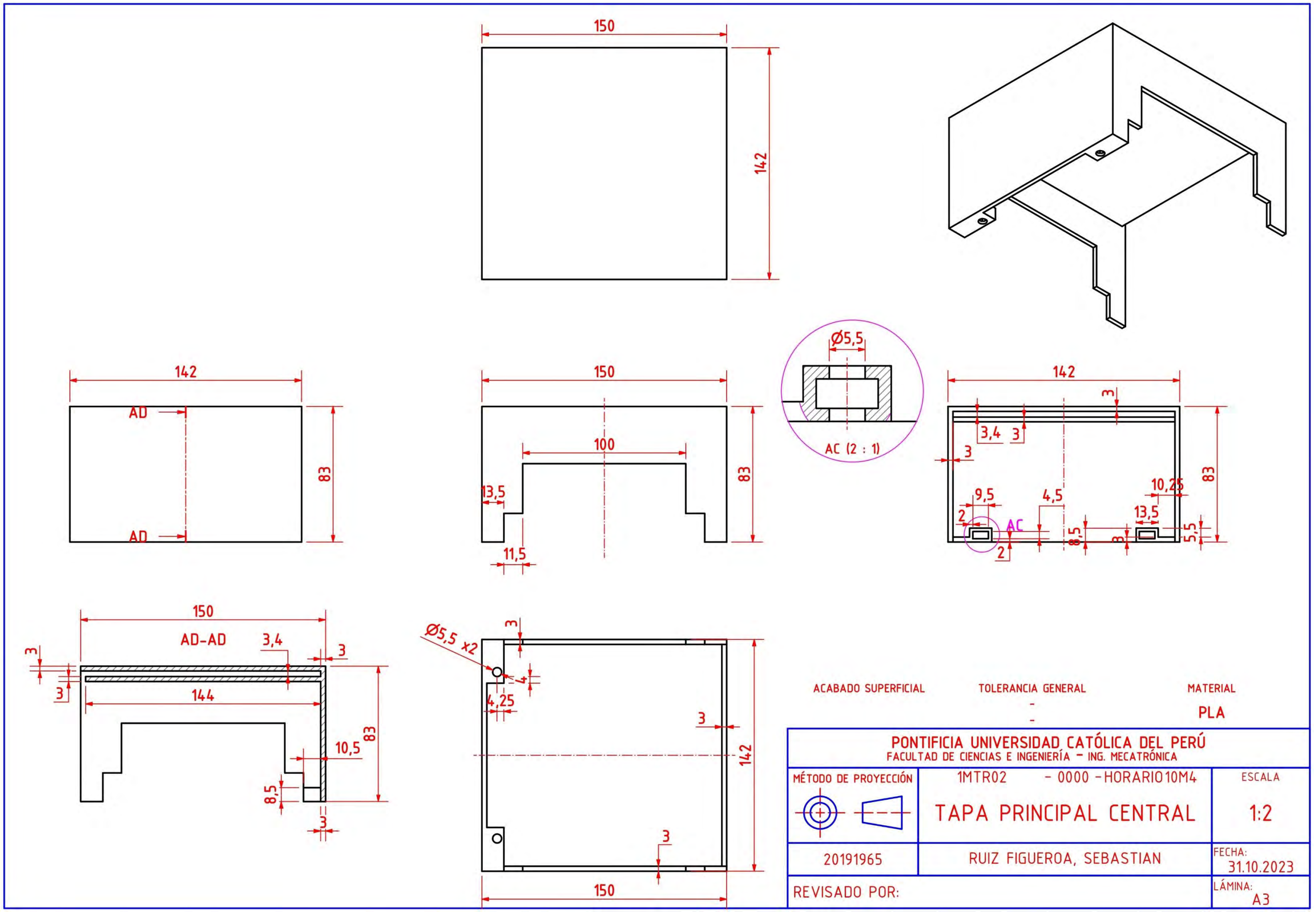


ACABADO SUPERFICIAL	TOLERANCIA GENERAL	MATERIAL
	-	PLA

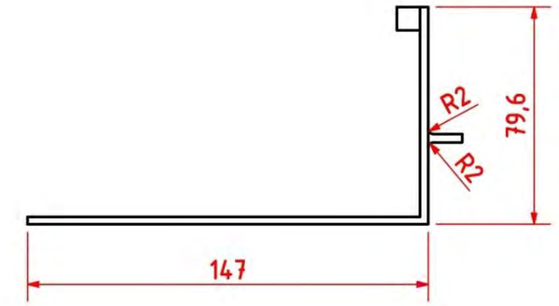
PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA – ING. MECATRÓNICA

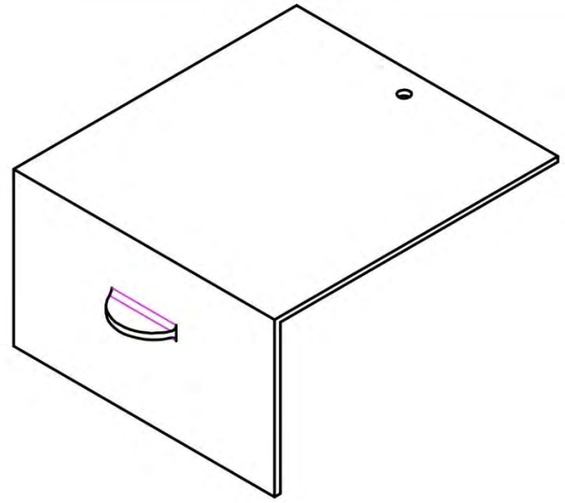
MÉTODO DE PROYECCIÓN	1MTR02 – 0000 – HORARIO10M4	ESCALA
	TAPA PRINCIPAL DELANTERA	1:2
20191965	RUIZ FIGUEROA, SEBASTIAN	FECHA: 31.10.2023
REVISADO POR:		LÁMINA: A3

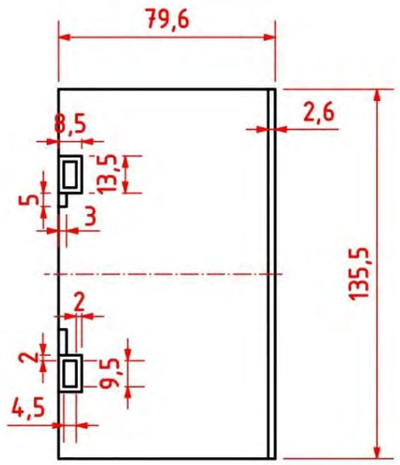
Anexo 59

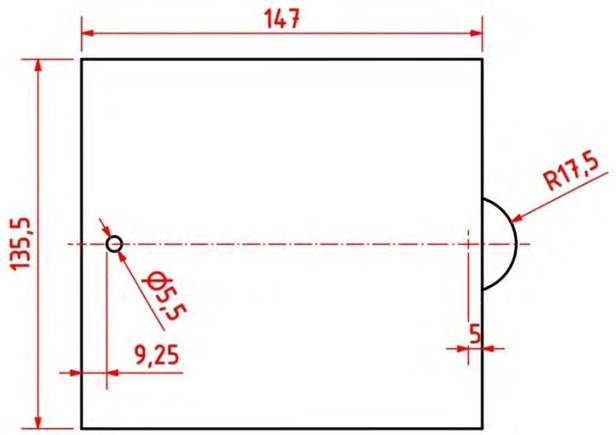

150
142
142
AD
AD
83
150
100
13,5
11,5
83
Ø5,5
AC (2 : 1)
142
3
3,4
3
3
9,5
4,5
10,25
13,5
2
AC
2
8,5
3
5,5
83
150
AD-AD
3,4
3
3
3
144
10,5
8,5
83
3
Ø5,5 x2
3
4
4,25
3
142
3
150
ACABADO SUPERFICIAL
TOLERANCIA GENERAL
-
-
MATERIAL
PLA
PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA – ING. MECATRÓNICA
MÉTODO DE PROYECCIÓN
1MTR02 - 0000 - HORARIO10M4
TAPA PRINCIPAL CENTRAL
ESCALA
1:2
20191965
RUIZ FIGUEROA, SEBASTIAN
FECHA:
31.10.2023
REVISADO POR:
LÁMINA:
A3

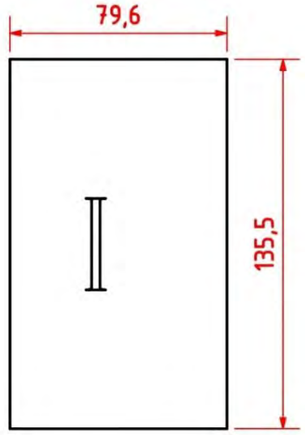
Anexo 60

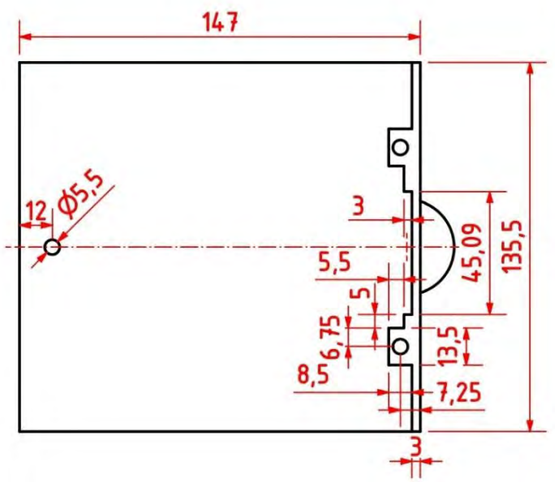


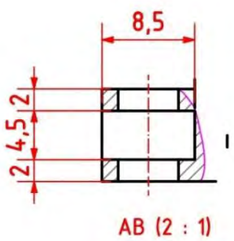


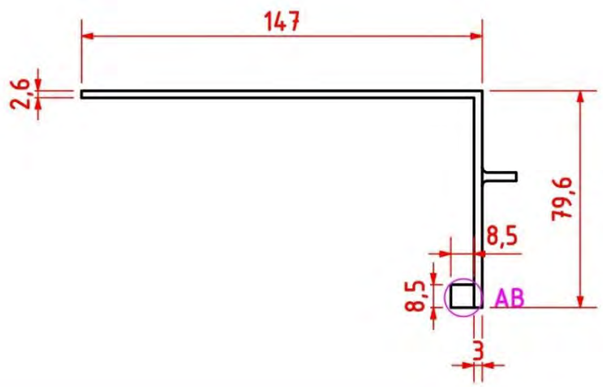










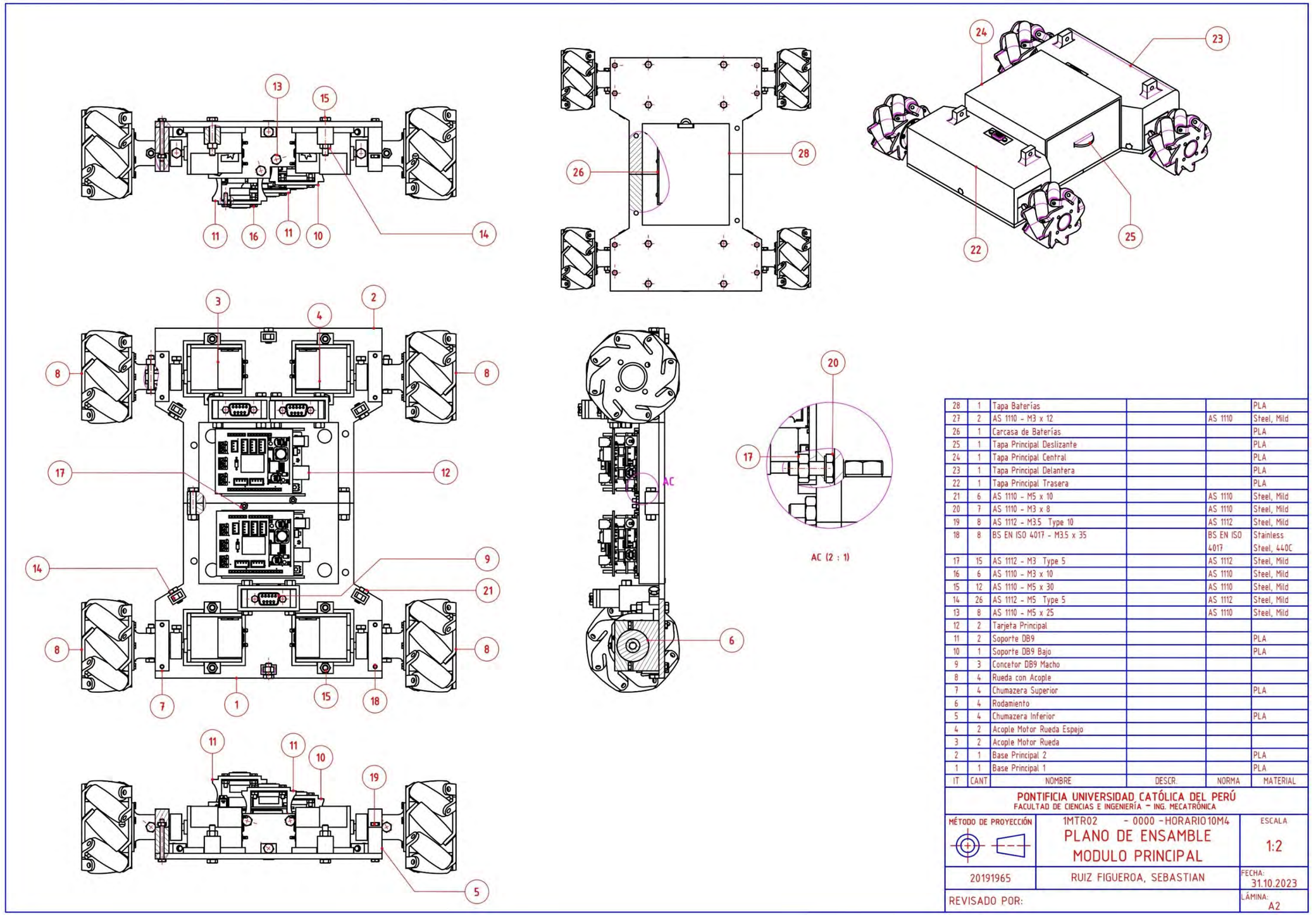


ACABADO SUPERFICIAL	TOLERANCIA GENERAL	MATERIAL
	-	PLA

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA – ING. MECATRÓNICA

MÉTODO DE PROYECCIÓN	1MTR02 - 0000 - HORARIO10M4	ESCALA
	TAPA PRINCIPAL DESLIZANTE	1:2
20191965	RUIZ FIGUEROA, SEBASTIAN	FECHA: 31.10.2023
REVISADO POR:		LÁMINA: A3

Anexo 61

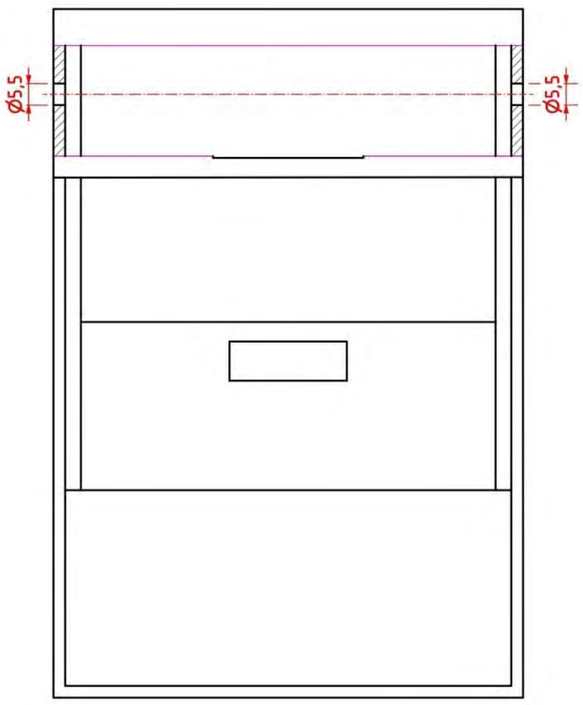


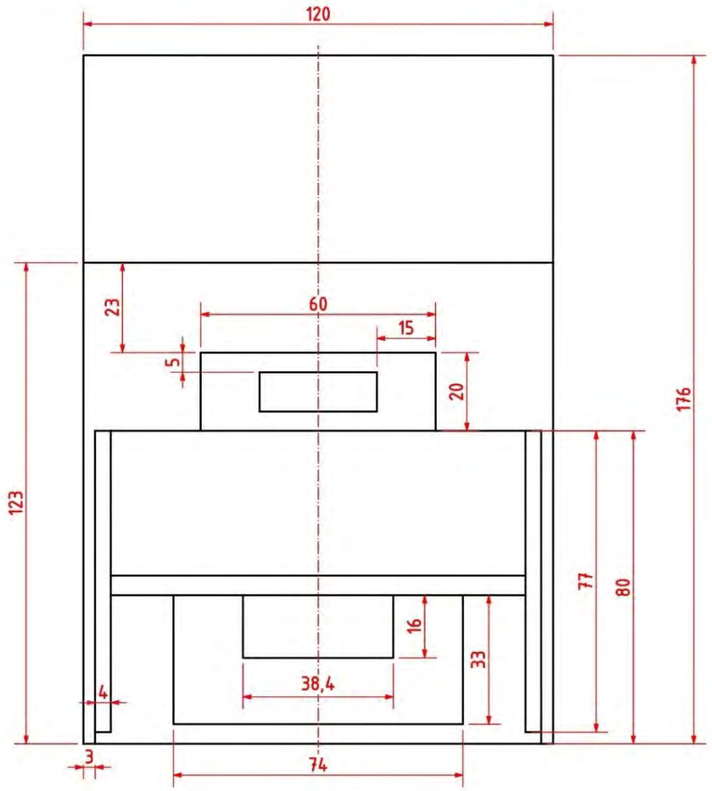
IT	CANT	NOMBRE	DESCR.	NORMA	MATERIAL
28	1	Tapa Baterías			PLA
27	2	AS 1110 - M3 x 12		AS 1110	Steel, Mild
26	1	Carcasa de Baterías			PLA
25	1	Tapa Principal Deslizante			PLA
24	1	Tapa Principal Central			PLA
23	1	Tapa Principal Delantera			PLA
22	1	Tapa Principal Trasera			PLA
21	6	AS 1110 - M5 x 10		AS 1110	Steel, Mild
20	7	AS 1110 - M3 x 8		AS 1110	Steel, Mild
19	8	AS 1112 - M3.5 Type 10		AS 1112	Steel, Mild
18	8	BS EN ISO 4017 - M3.5 x 35		BS EN ISO 4017	Stainless Steel, 440C
17	15	AS 1112 - M3 Type 5		AS 1112	Steel, Mild
16	6	AS 1110 - M3 x 10		AS 1110	Steel, Mild
15	12	AS 1110 - M5 x 30		AS 1110	Steel, Mild
14	26	AS 1112 - M5 Type 5		AS 1112	Steel, Mild
13	8	AS 1110 - M5 x 25		AS 1110	Steel, Mild
12	2	Tarjeta Principal			
11	2	Soporte DB9			PLA
10	1	Soporte DB9 Bajo			PLA
9	3	Concetor DB9 Macho			
8	4	Rueda con Acople			
7	4	Chumazera Superior			PLA
6	4	Rodamiento			
5	4	Chumazera Inferior			PLA
4	2	Acople Motor Rueda Espejo			
3	2	Acople Motor Rueda			
2	1	Base Principal 2			PLA
1	1	Base Principal 1			PLA

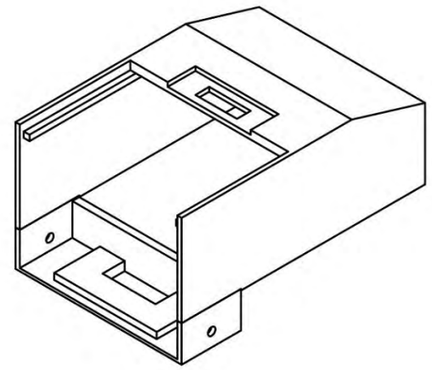
PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA – ING. MECATRÓNICA

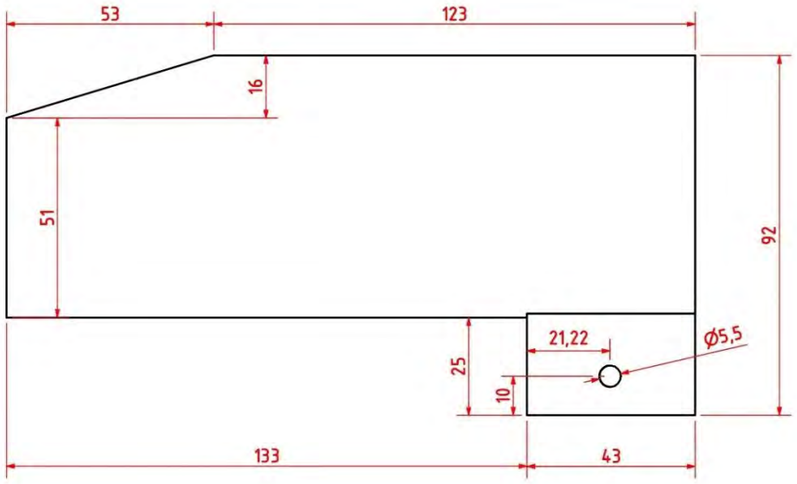
MÉTODO DE PROYECCIÓN	1MTR02 - 0000 - HORARIO10M4 PLANO DE ENSAMBLE MODULO PRINCIPAL	ESCALA 1:2
20191965	RUIZ FIGUEROA, SEBASTIAN	FECHA: 31.10.2023
REVISADO POR:		LÁMINA: A2

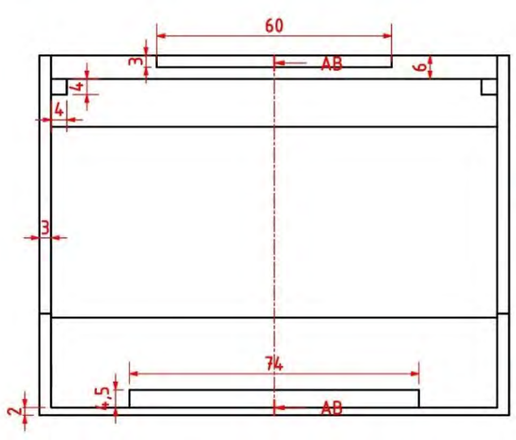
Anexo 62

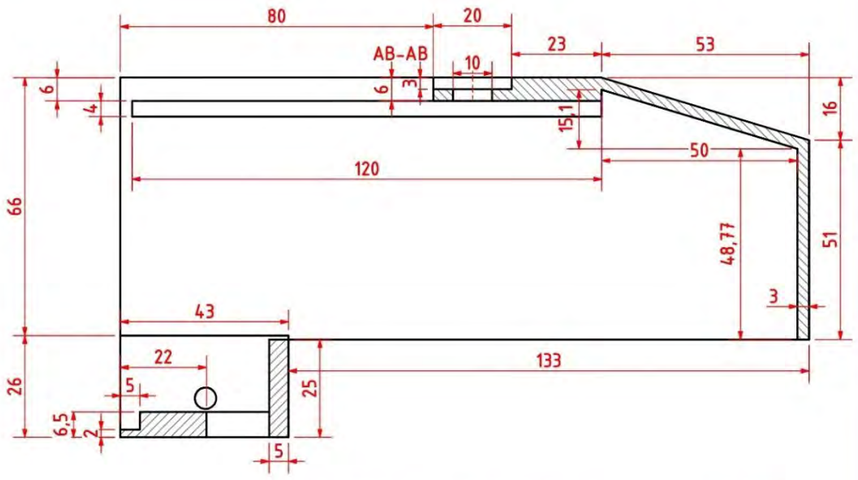








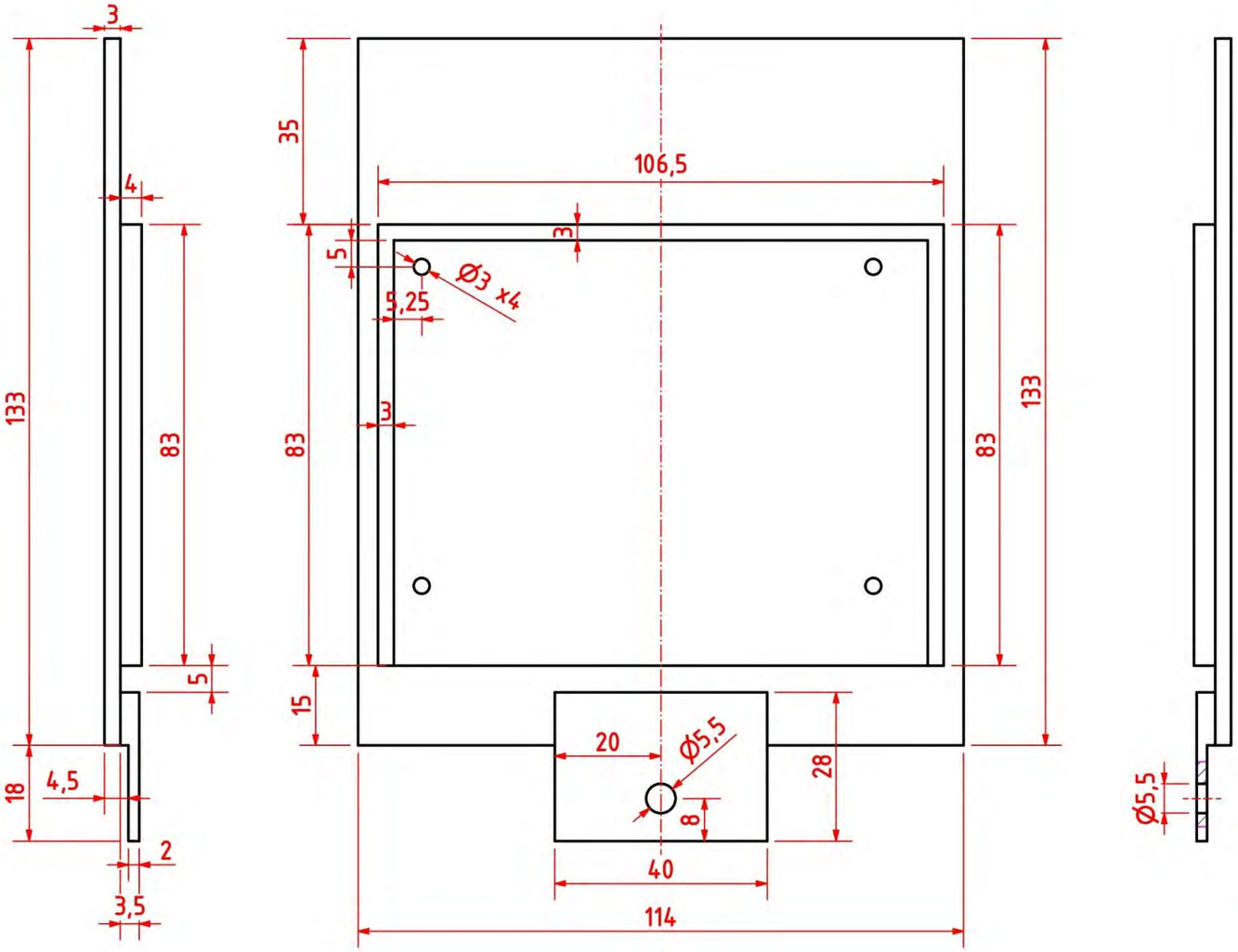


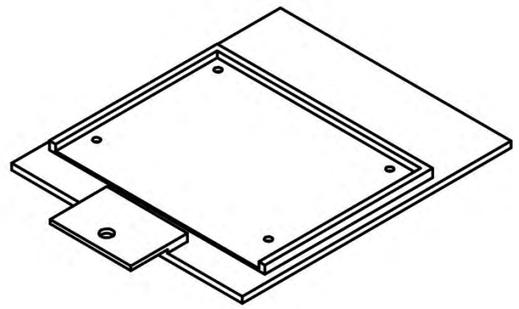


ACABADO SUPERFICIAL	TOLERANCIA GENERAL	MATERIAL
	-	PLA

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ FACULTAD DE CIENCIAS E INGENIERÍA – ING. MECATRÓNICA		
MÉTODO DE PROYECCIÓN	1MTR02 - 0000 - HORARIO10M4 CARCASA MODULO CONTROL	ESCALA 1:1
20191965	RUIZ FIGUEROA, SEBASTIAN	FECHA: 31.10.2023
REVISADO POR:		LÁMINA: A3

Anexo 63



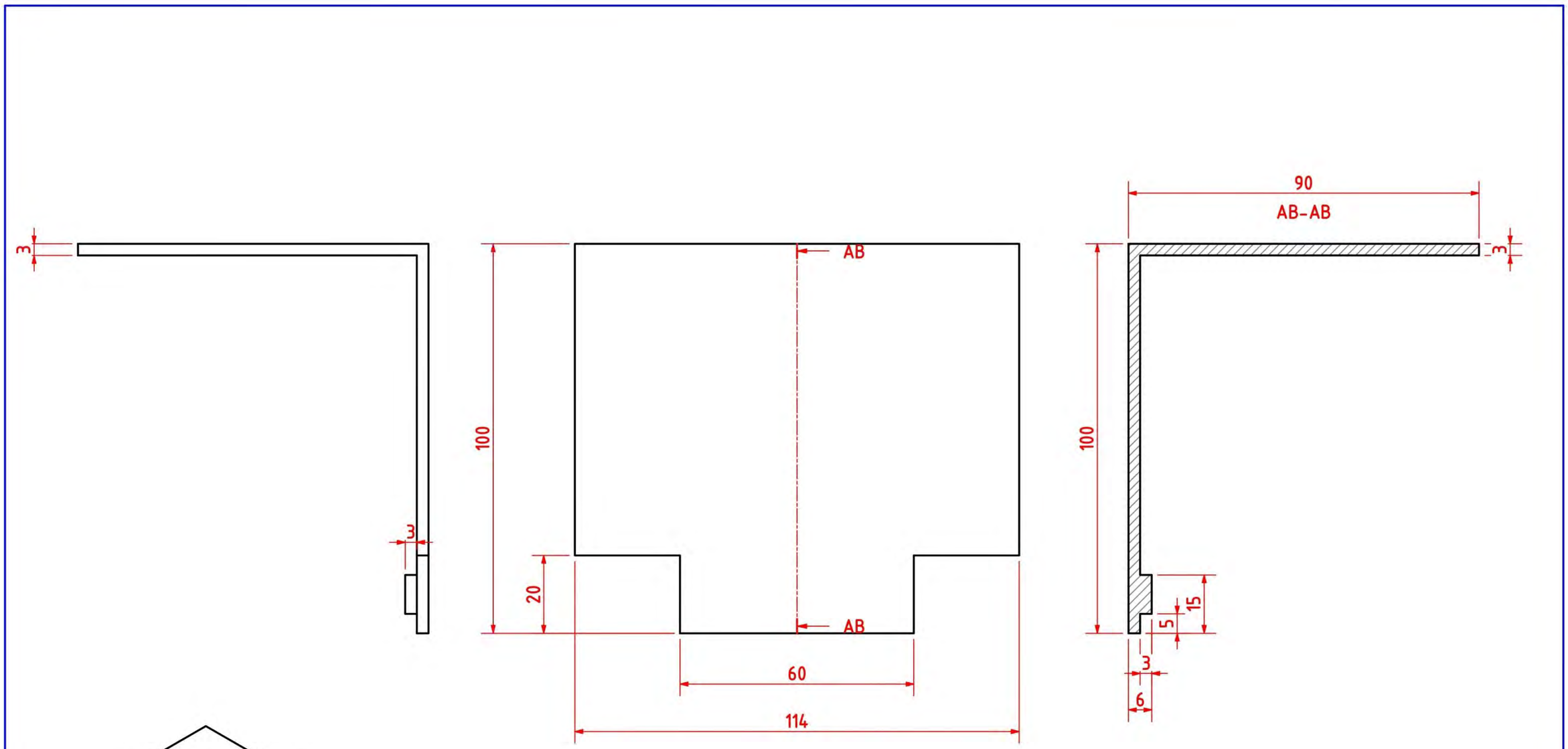


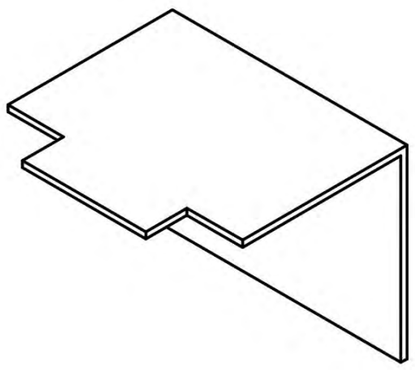
ACABADO SUPERFICIAL	TOLERANCIA GENERAL	MATERIAL
	-	PLA

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA – ING. MECATRÓNICA

MÉTODO DE PROYECCIÓN	1MTR02 - 0000 - HORARIO10M4	ESCALA
	BASE JETSON NANO	1:1
20191965	RUIZ FIGUEROA, SEBASTIAN	FECHA: 31.10.2023
REVISADO POR:		LÁMINA: A3

Anexo 64



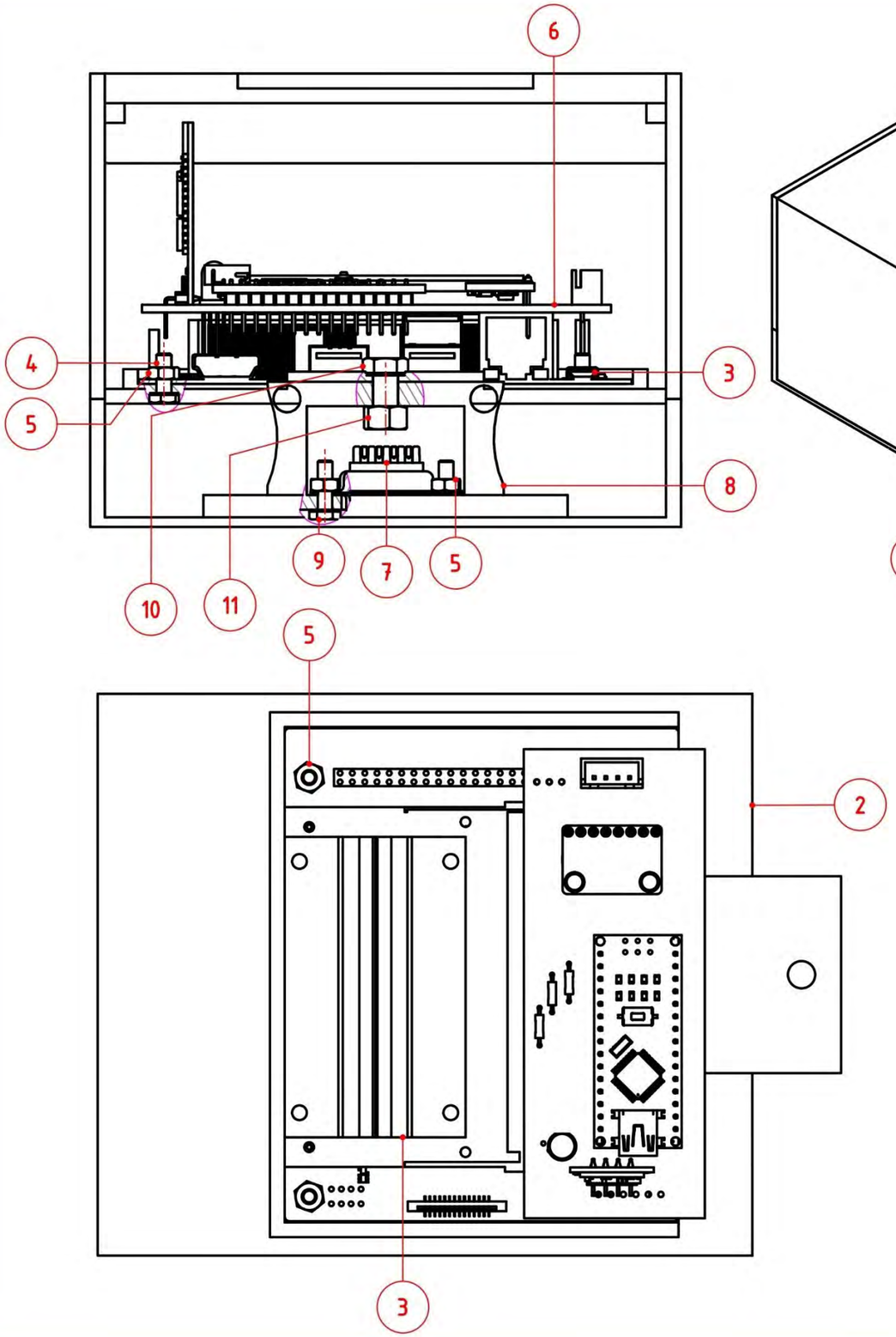


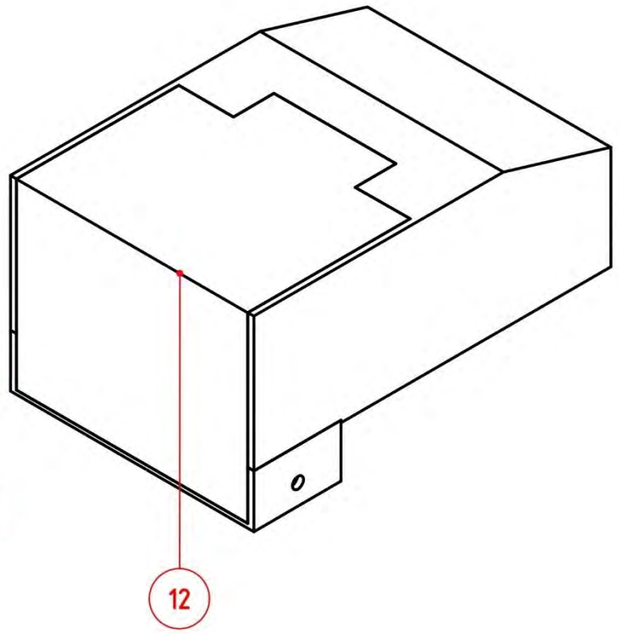
ACABADO SUPERFICIAL	TOLERANCIA GENERAL	MATERIAL
	-	PLA

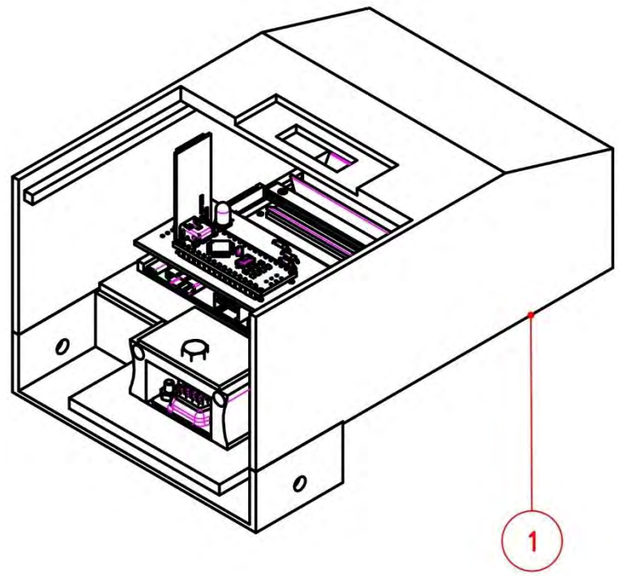
PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA – ING. MECATRÓNICA

MÉTODO DE PROYECCIÓN	1MTR02 - 0000 - HORARIO10M4	ESCALA
	TAPA MODULO CONTROL	1:2
20191965	RUIZ FIGUEROA, SEBASTIAN	FECHA: 31.10.2023
REVISADO POR:		LÁMINA: A3

Anexo 65





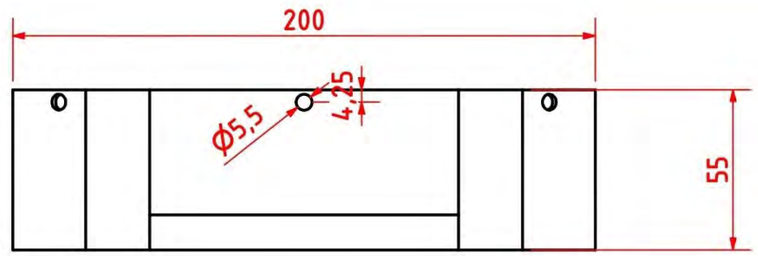


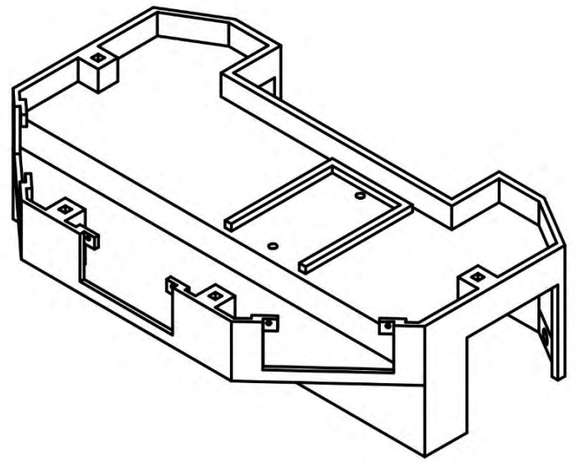
IT	CANT	NOMBRE	DESCR.	NORMA	MATERIAL
12	1	Tapa de Módulo de Control			Steel, Mild, Welded
11	1	AS 1112 - M5 Type 5		AS 1112	Steel, Mild
10	1	AS 1110 - M5 x 10		AS 1110	Steel, Mild
9	2	AS 1110 - M3 x 10		AS 1110	Steel, Mild
7	1	Conector DB9 Hembra			
8	1	Soporte DB9			PLA
5	6	AS 1112 - M3 Type 5		AS 1112	Steel, Mild
4	4	AS 1110 - M3 x 8		AS 1110	Steel, Mild
6	1	Tarjeta de Control			
3	1	Jetson Nano			Generic
2	1	Base Jetson			PLA
1	1	Carcasa Control			PLA

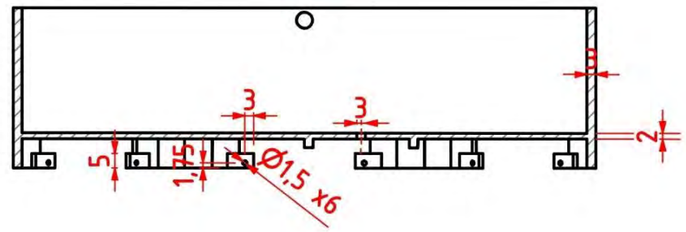
PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA – ING. MECATRÓNICA

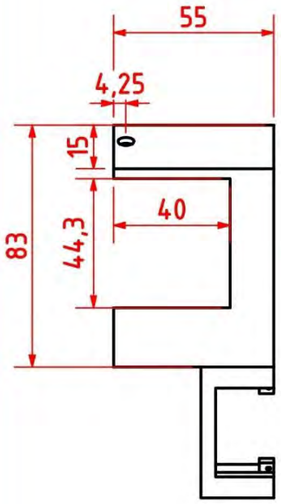
MÉTODO DE PROYECCIÓN	1MTR02 - 0000 - HORARIO10M4 PLANO DE ENSAMBLE MODULO DE CONTROL	ESCALA 1:1
20191965	RUIZ FIGUEROA, SEBASTIAN	FECHA: 31.10.2023
REVISADO POR:		LÁMINA: A3

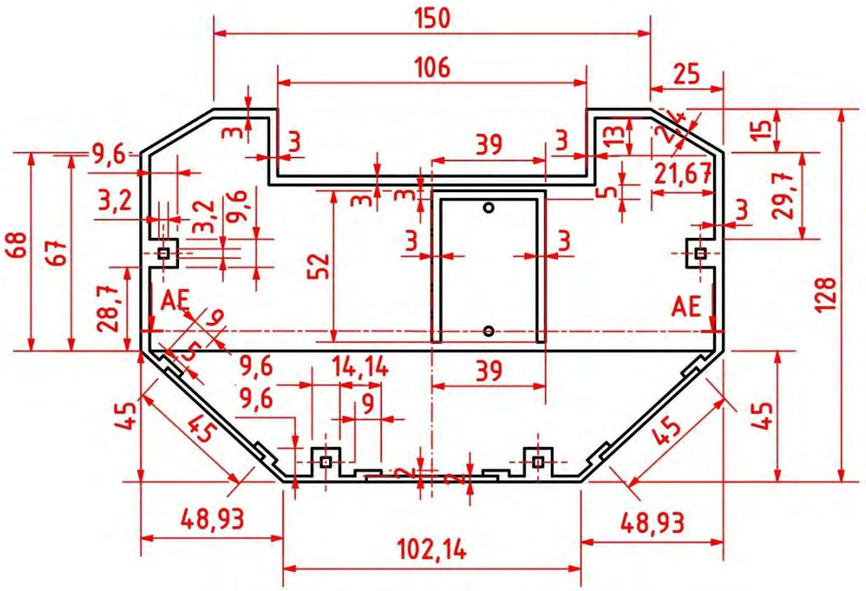
Anexo 66

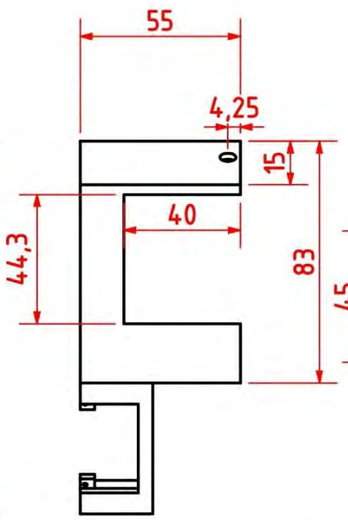


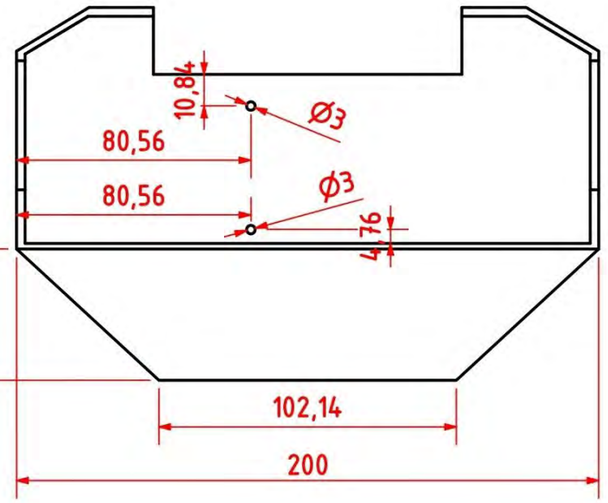


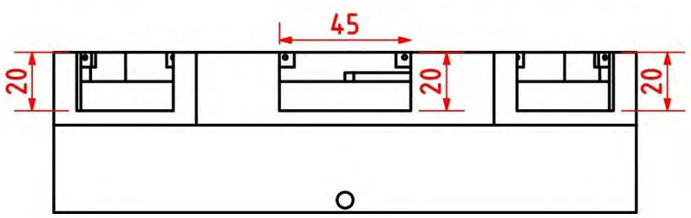










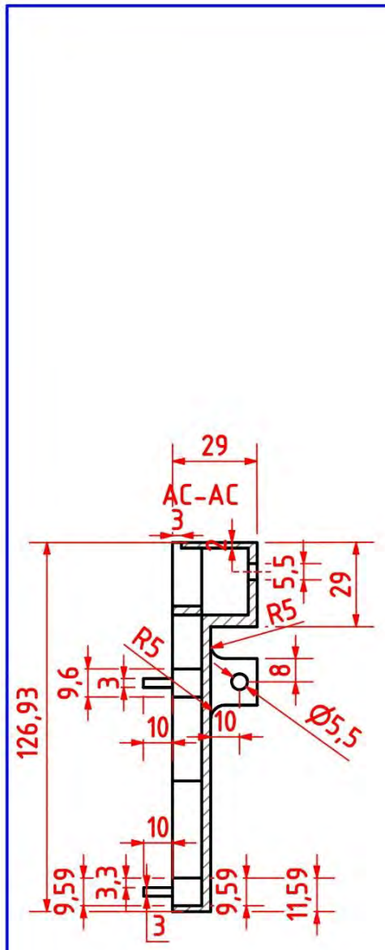


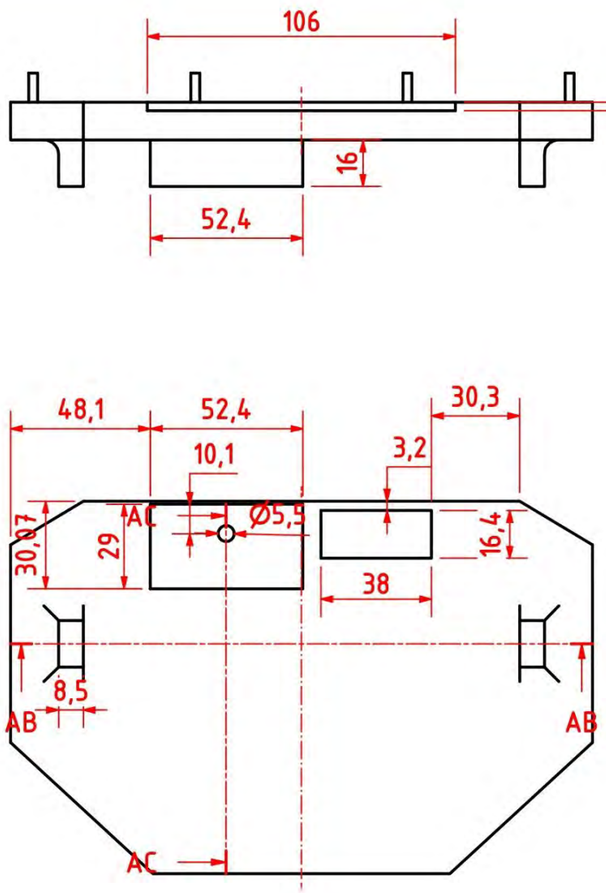
ACABADO SUPERFICIAL	TOLERANCIA GENERAL	MATERIAL
	-	PLA

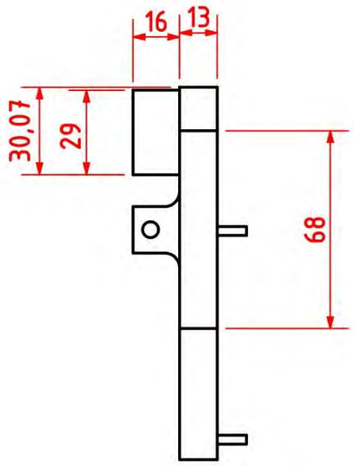
PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA – ING. MECATRÓNICA

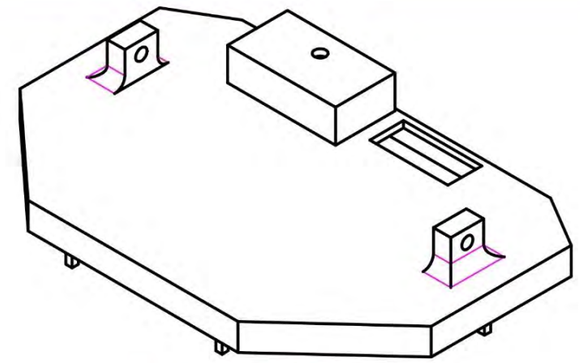
MÉTODO DE PROYECCIÓN	1MTR02 - 0000 - HORARIO10M4	ESCALA
	CARCASA MODULO SENSORES	1:2
20191965	RUIZ FIGUEROA, SEBASTIAN	FECHA: 31.10.2023
REVISADO POR:		LÁMINA: A3

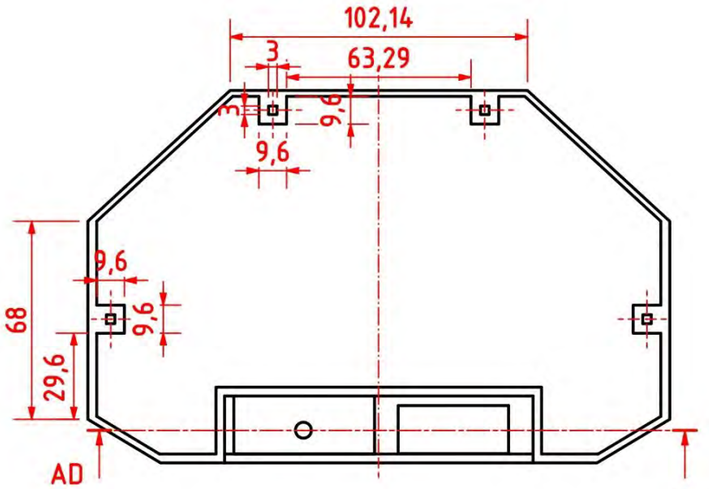
Anexo 67

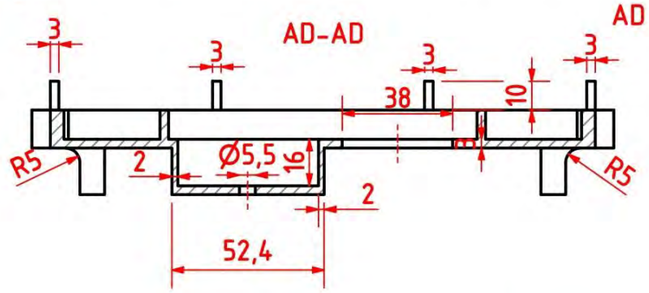


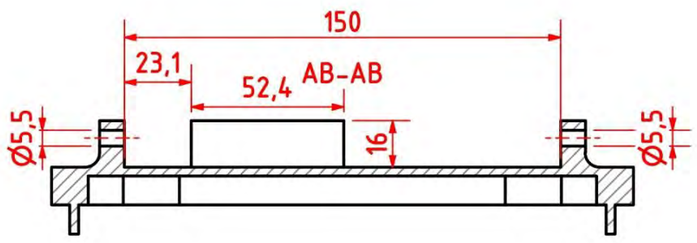








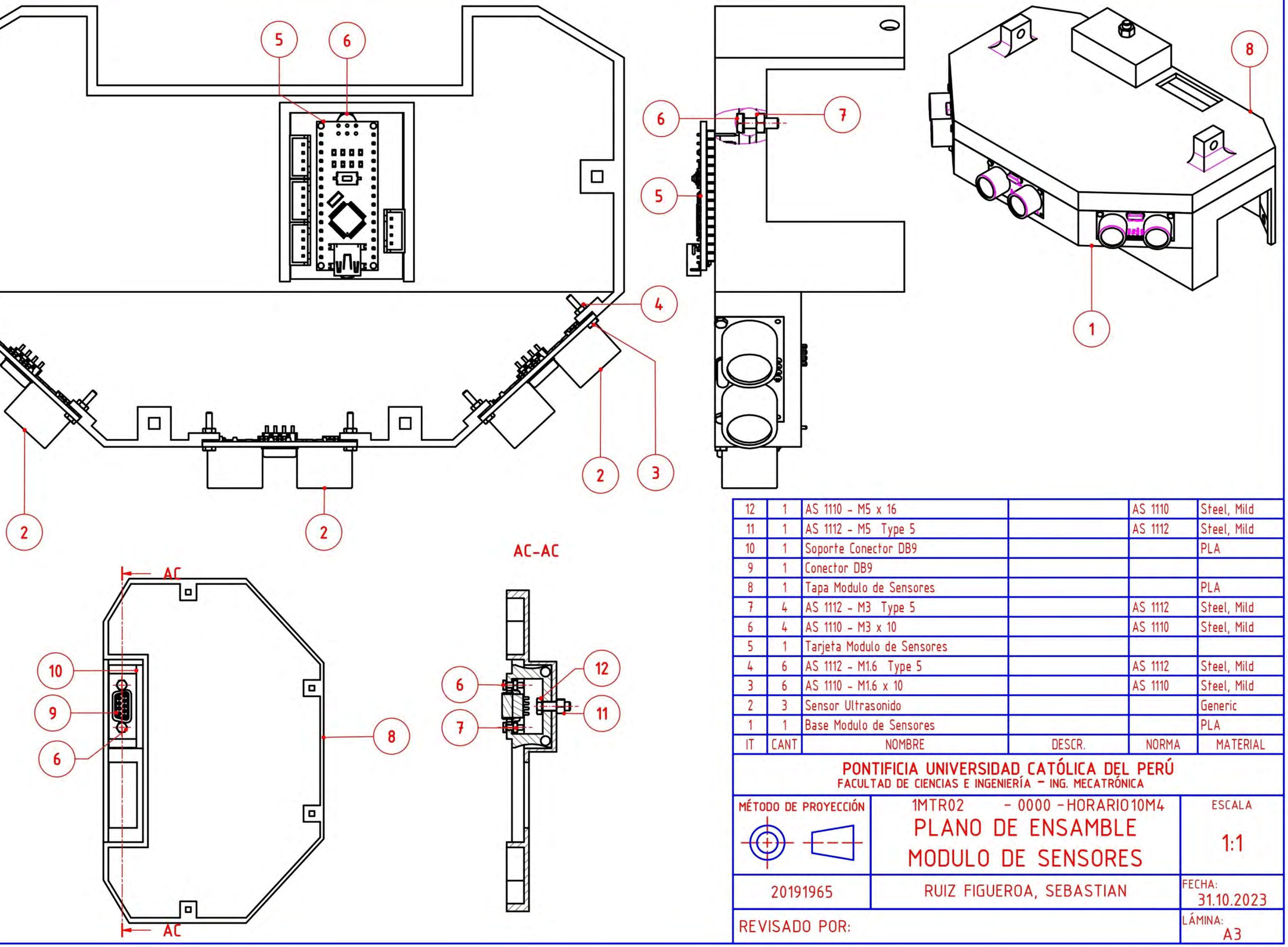




ACABADO SUPERFICIAL	TOLERANCIA GENERAL	MATERIAL
	-	PLA

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ FACULTAD DE CIENCIAS E INGENIERÍA – ING. MECATRÓNICA		
MÉTODO DE PROYECCIÓN	1MTR02 - 0000 - HORARIO10M4 TAPA MODULO SENSORES	ESCALA 1:2
20191965	RUIZ FIGUEROA, SEBASTIAN	FECHA: 31.10.2023
REVISADO POR:		LÁMINA: A3

Anexo 68

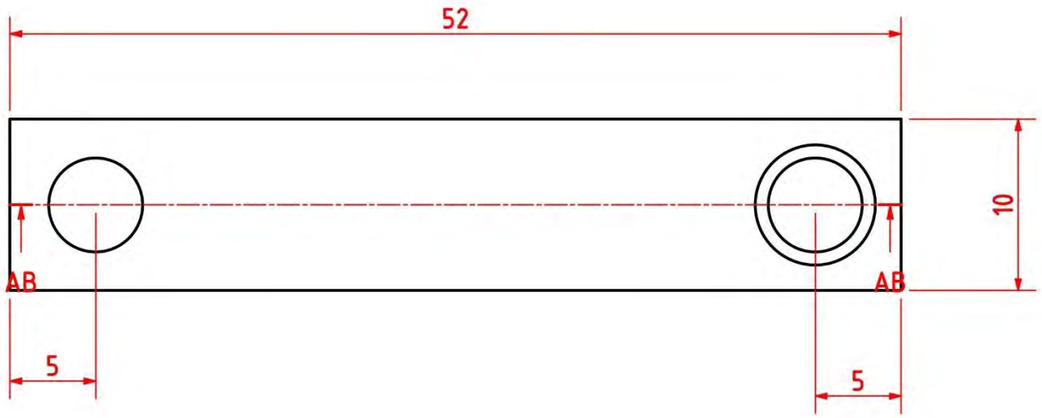


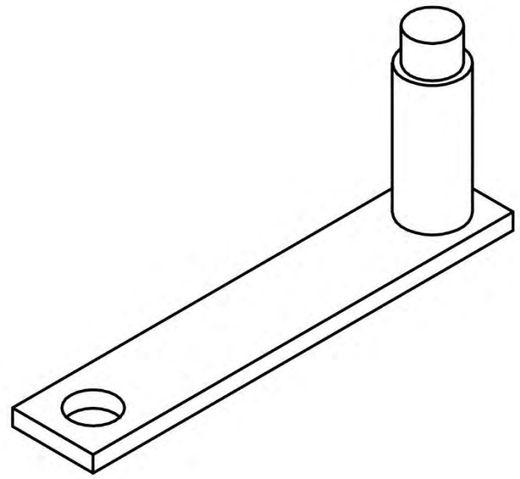
IT	CANT	NOMBRE	DESCR.	NORMA	MATERIAL
12	1	AS 1110 - M5 x 16		AS 1110	Steel, Mild
11	1	AS 1112 - M5 Type 5		AS 1112	Steel, Mild
10	1	Soporte Conector DB9			PLA
9	1	Conector DB9			
8	1	Tapa Modulo de Sensores			PLA
7	4	AS 1112 - M3 Type 5		AS 1112	Steel, Mild
6	4	AS 1110 - M3 x 10		AS 1110	Steel, Mild
5	1	Tarjeta Modulo de Sensores			
4	6	AS 1112 - M1.6 Type 5		AS 1112	Steel, Mild
3	6	AS 1110 - M1.6 x 10		AS 1110	Steel, Mild
2	3	Sensor Ultrasonido			Generic
1	1	Base Modulo de Sensores			PLA

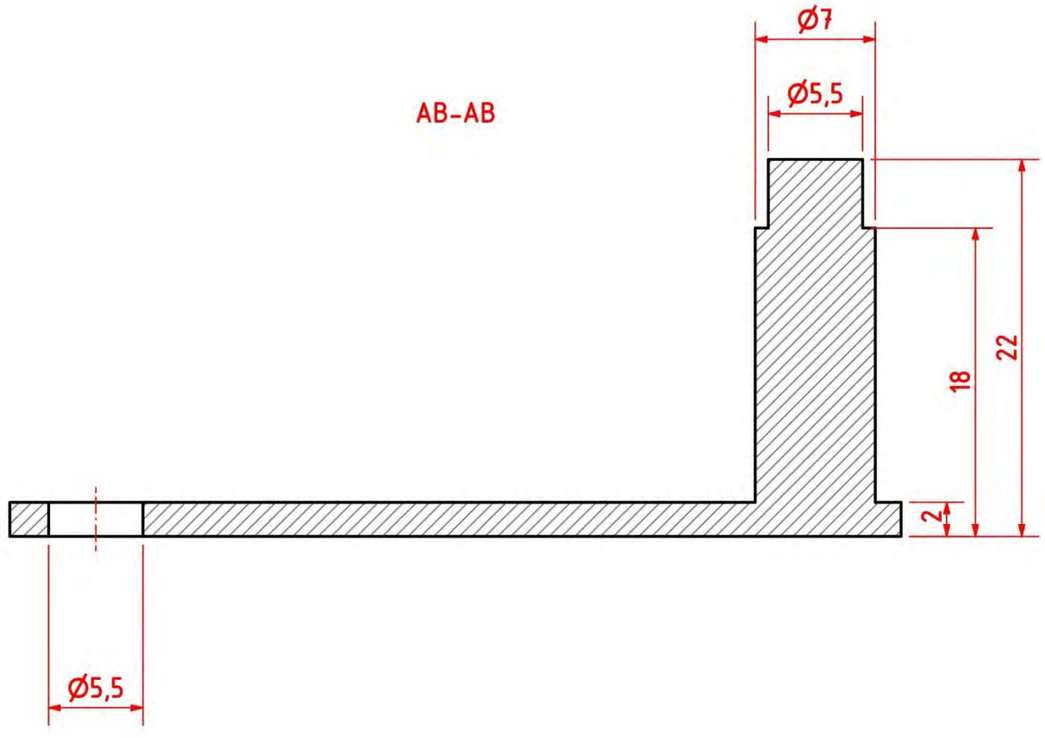
PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA – ING. MECATRÓNICA

MÉTODO DE PROYECCIÓN	1MTR02 - 0000 - HORARIO10M4 PLANO DE ENSAMBLE MODULO DE SENSORES	ESCALA 1:1
20191965	RUIZ FIGUEROA, SEBASTIAN	FECHA: 31.10.2023
REVISADO POR:		LÁMINA: A3

Anexo 69



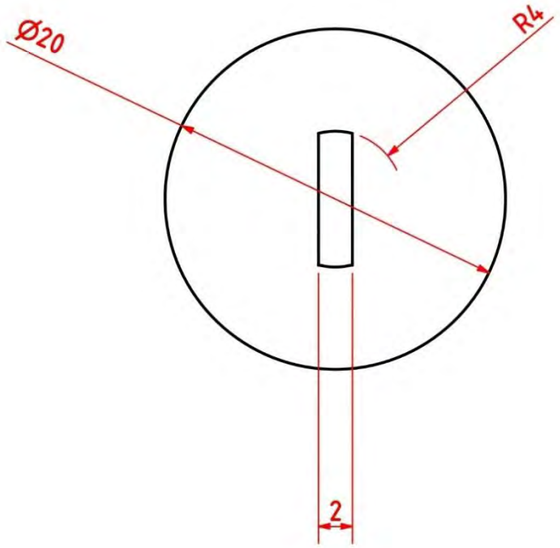


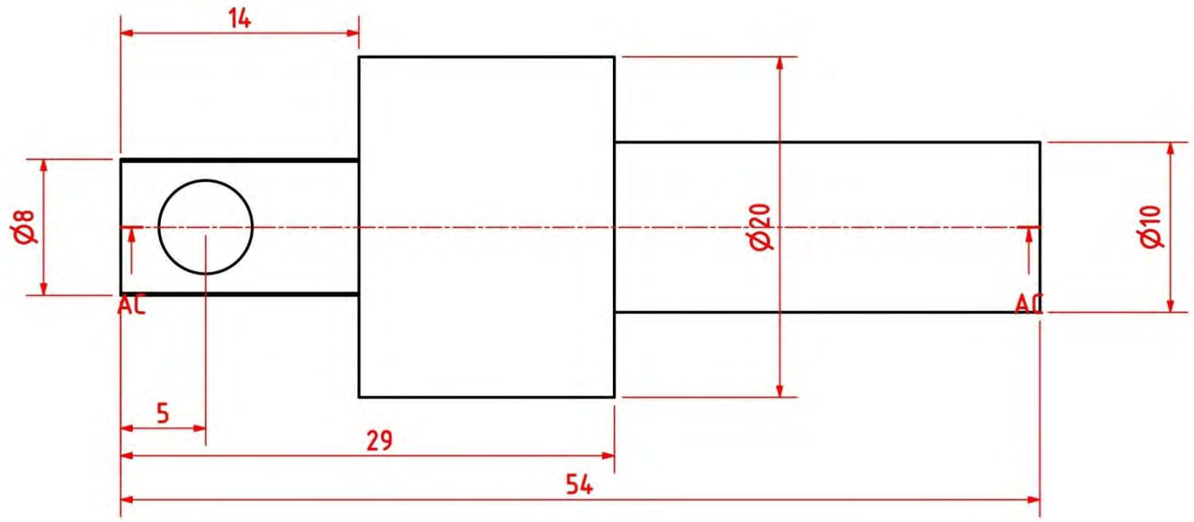


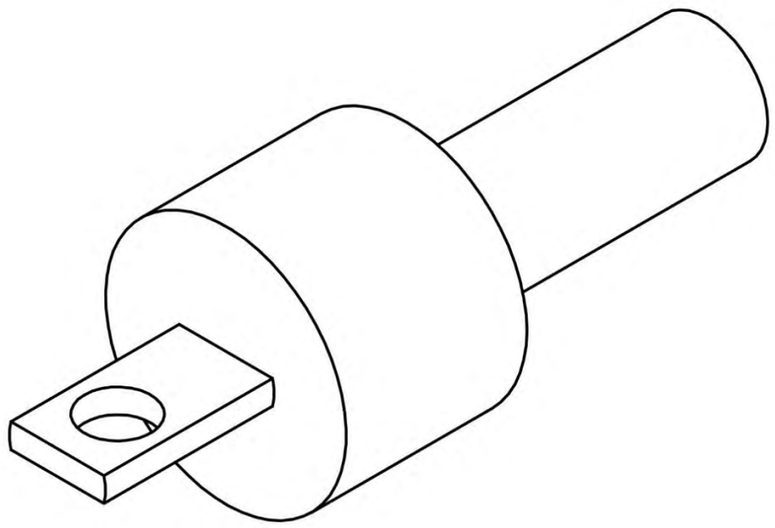
ACABADO SUPERFICIAL	TOLERANCIA GENERAL	MATERIAL
	-	PLA

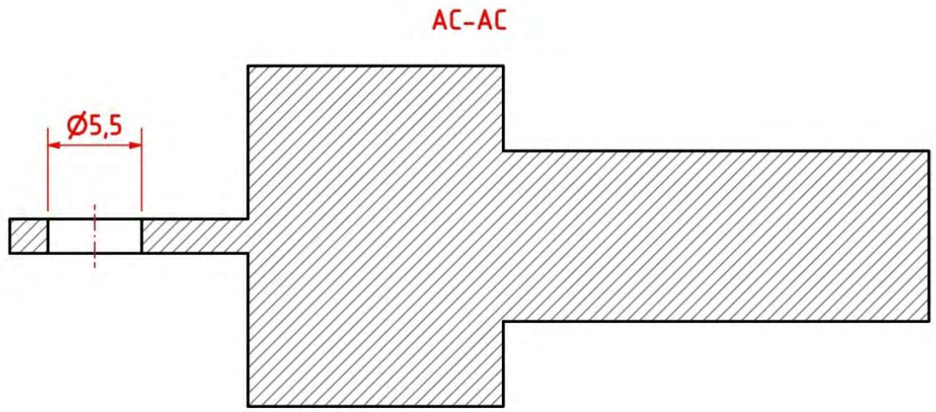
PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ FACULTAD DE CIENCIAS E INGENIERÍA – ING. MECATRÓNICA		
MÉTODO DE PROYECCIÓN	1MTR02 - 0000 - HORARIO10M4 BIELA MANIVELA 1	ESCALA 3:1
20191965	RUIZ FIGUEROA, SEBASTIAN	FECHA: 31.10.2023
REVISADO POR:		LÁMINA: A3

Anexo 70









ACABADO SUPERFICIAL

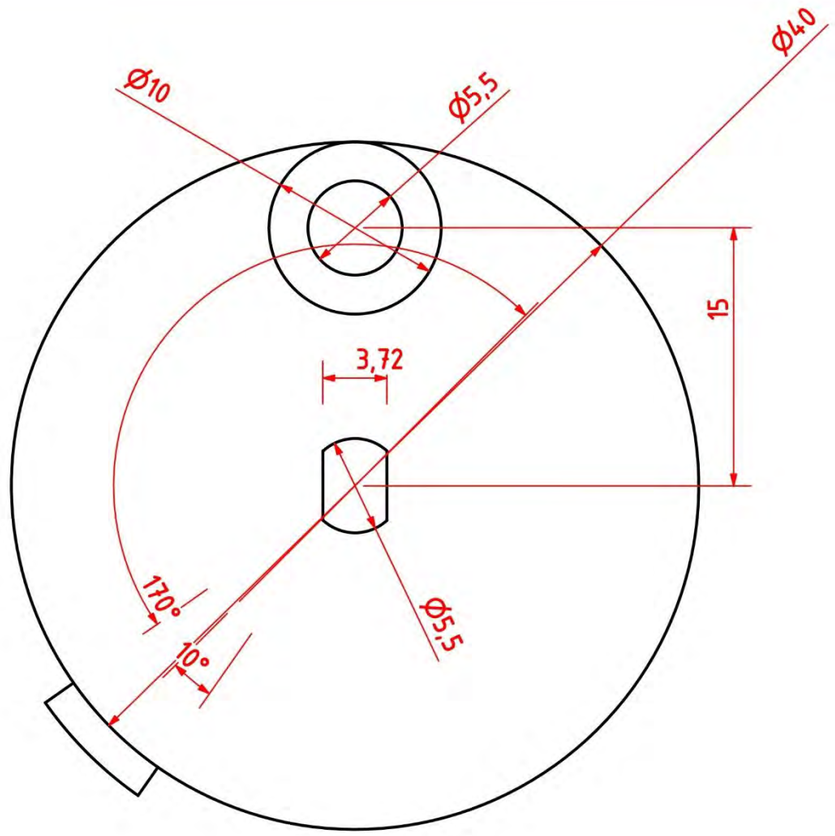
TOLERANCIA GENERAL
-
-

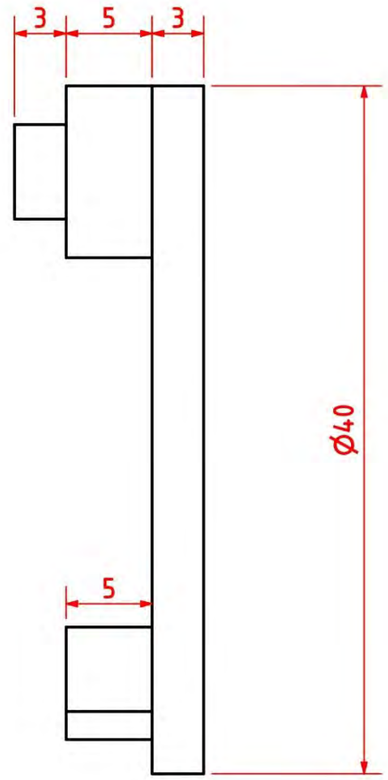
MATERIAL
PLA

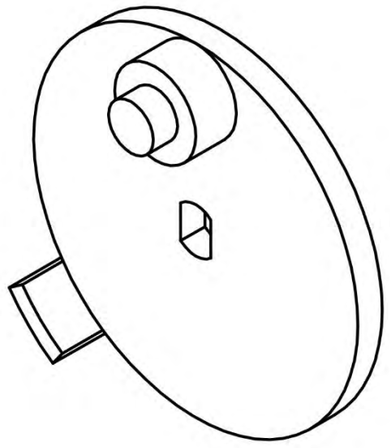
PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA – ING. MECATRÓNICA

MÉTODO DE PROYECCIÓN	1MTR02 - 0000 - HORARIO10M4 BIELA MANIVELA 2	ESCALA 3:1
20191965	RUIZ FIGUEROA, SEBASTIAN	FECHA: 31.10.2023
REVISADO POR:		LÁMINA: A3

Anexo 71



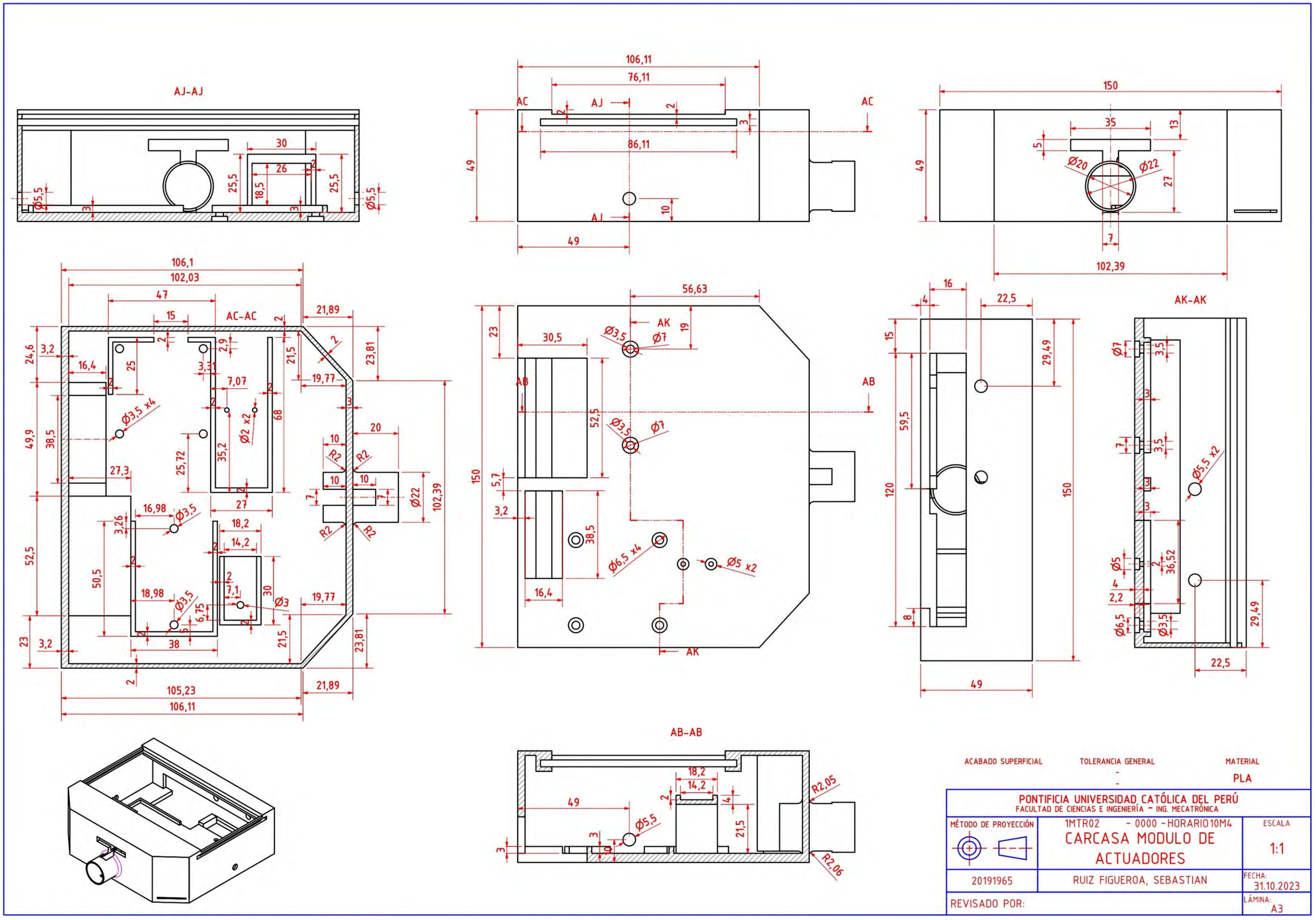




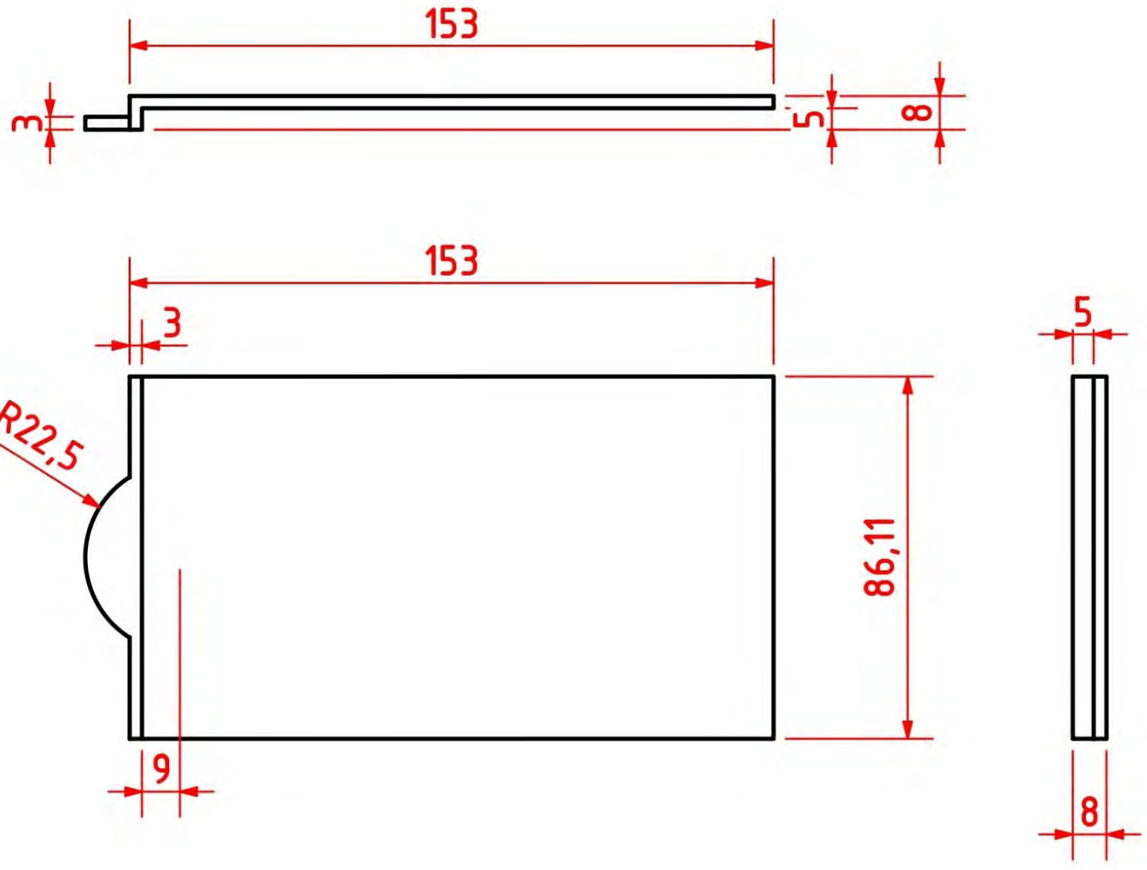
ACABADO SUPERFICIAL	TOLERANCIA GENERAL	MATERIAL
	-	PLA

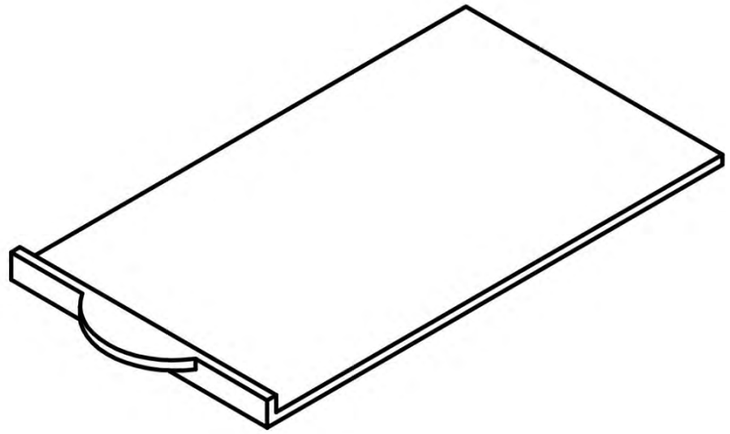
PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ FACULTAD DE CIENCIAS E INGENIERÍA – ING. MECATRÓNICA		
MÉTODO DE PROYECCIÓN	1MTR02 - 0000 - HORARIO10M4 RUEDA BIELA MANIVELA	ESCALA 3:1
20191965	RUIZ FIGUEROA, SEBASTIAN	FECHA: 31.10.2023
REVISADO POR:		LÁMINA: A3

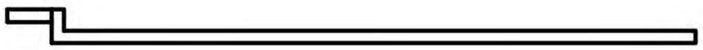
Anexo 72


AJ-AJ
106,11
76,11
AC
AJ
86,11
49
150
35
102,39
106,1
102,03
47
15
AC-AC
21,89
23,81
19,77
68
20
27,3
16,98
18,2
14,2
18,98
6,75
38
105,23
56,63
30,5
AK
AB
52,5
38,6
16,4
16
4
22,5
29,49
59,5
120
AK-AK
36,52
29,49
AB-AB
ACABADO SUPERFICIAL
TOLERANCIA GENERAL
MATERIAL
PLA
PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA – ING. MECATRÓNICA
MÉTODO DE PROYECCIÓN
1MTR02 - 0000 - HORARIO10M4
CARCASA MODULO DE ACTUADORES
ESCALA
1:1
20191965
RUIZ FIGUEROA, SEBASTIAN
FECHA:
31.10.2023
REVISADO POR:
LÁMINA:
A3

Anexo 73







ACABADO SUPERFICIAL

TOLERANCIA GENERAL

-

-

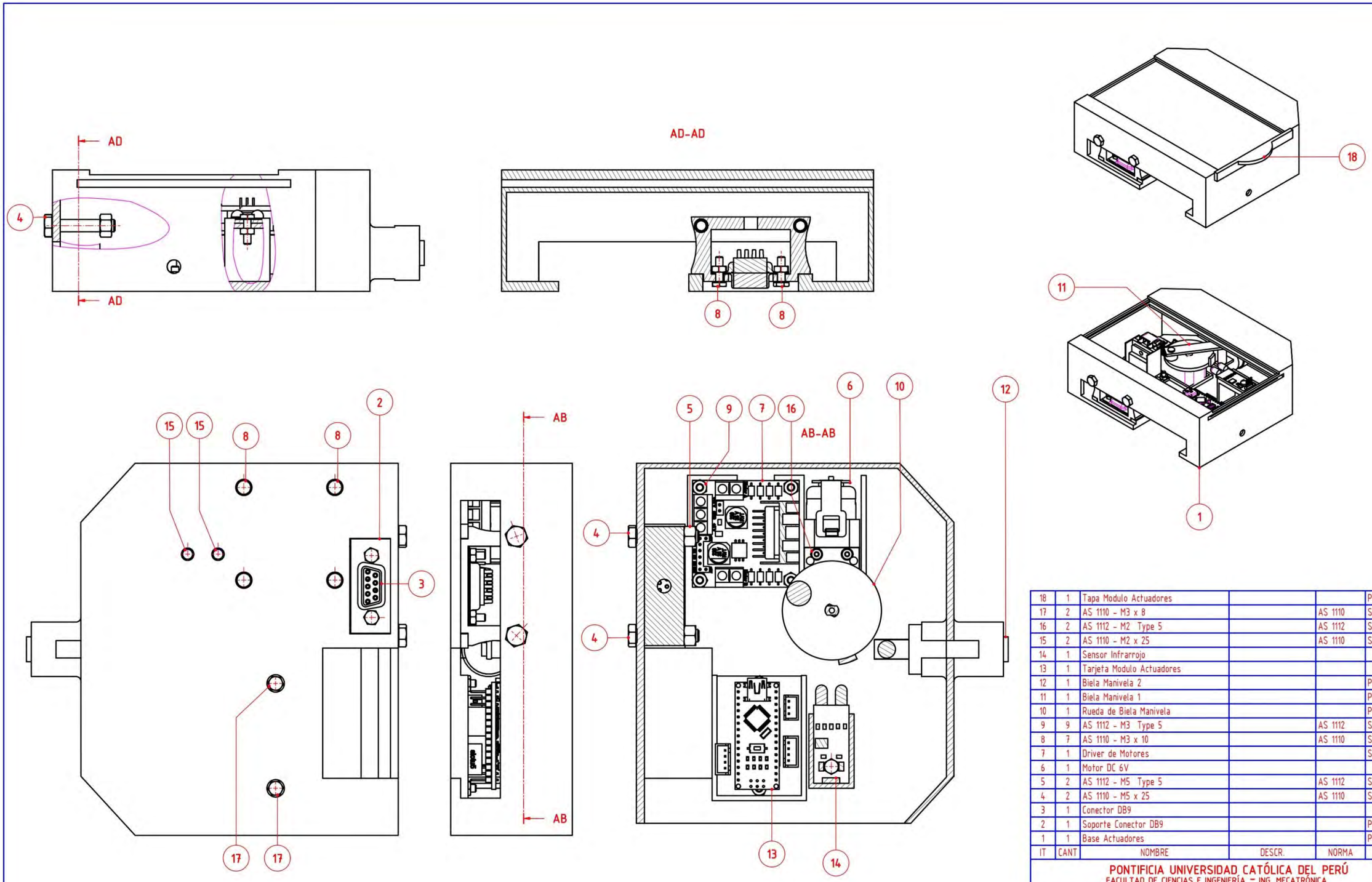
MATERIAL

PLA

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA – ING. MECATRÓNICA

MÉTODO DE PROYECCIÓN	1MTR02 - 0000 - HORARIO10M4 TAPA MODULO ACTUADORES	ESCALA 1:2
20191965	RUIZ FIGUEROA, SEBASTIAN	FECHA: 31.10.2023
REVISADO POR:		LÁMINA: A3

Anexo 74

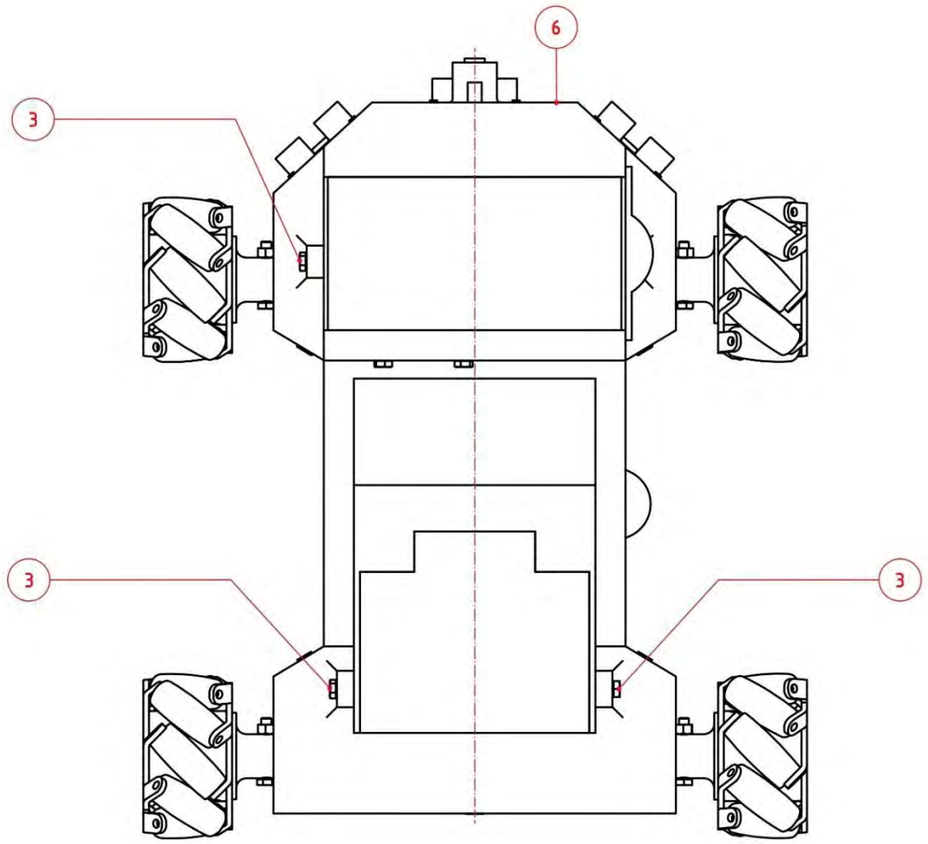


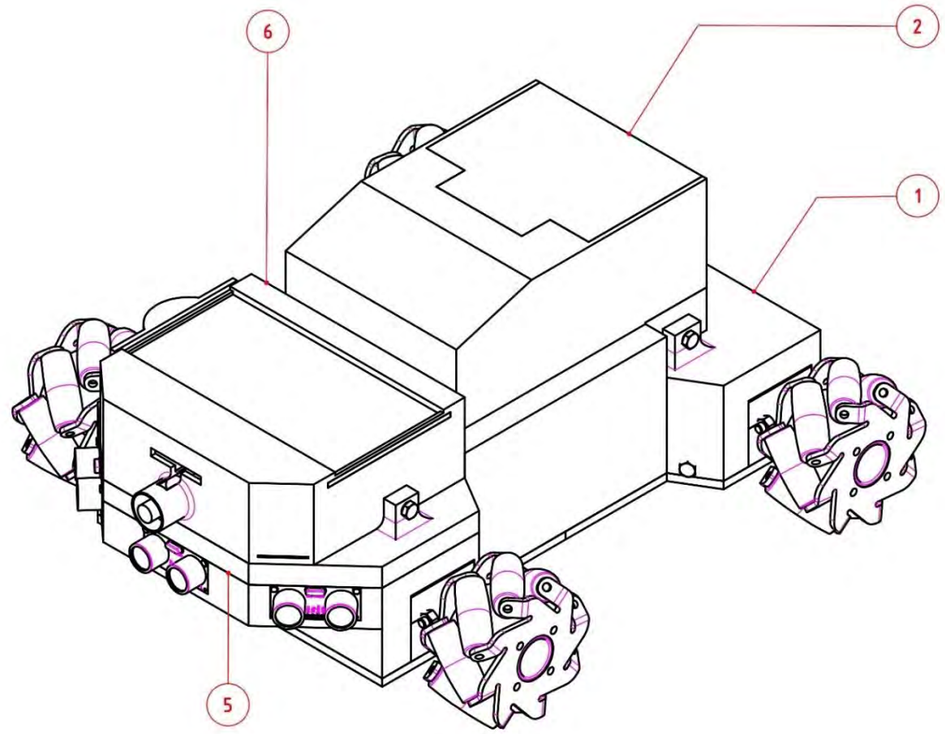
IT	CANT	NOMBRE	DESCR.	NORMA	MATERIAL
18	1	Tapa Modulo Actuadores			PLA
17	2	AS 1110 - M3 x 8		AS 1110	Steel, Mild
16	2	AS 1112 - M2 Type 5		AS 1112	Steel, Mild
15	2	AS 1110 - M2 x 25		AS 1110	Steel, Mild
14	1	Sensor Infrarrojo			
13	1	Tarjeta Modulo Actuadores			
12	1	Biela Manivela 2			PLA
11	1	Biela Manivela 1			PLA
10	1	Rueda de Biela Manivela			PLA
9	9	AS 1112 - M3 Type 5		AS 1112	Steel, Mild
8	7	AS 1110 - M3 x 10		AS 1110	Steel, Mild
7	1	Driver de Motores			Steel
6	1	Motor DC 6V			
5	2	AS 1112 - M5 Type 5		AS 1112	Steel, Mild
4	2	AS 1110 - M5 x 25		AS 1110	Steel, Mild
3	1	Conector DB9			
2	1	Soporte Conector DB9			PLA
1	1	Base Actuadores			PLA

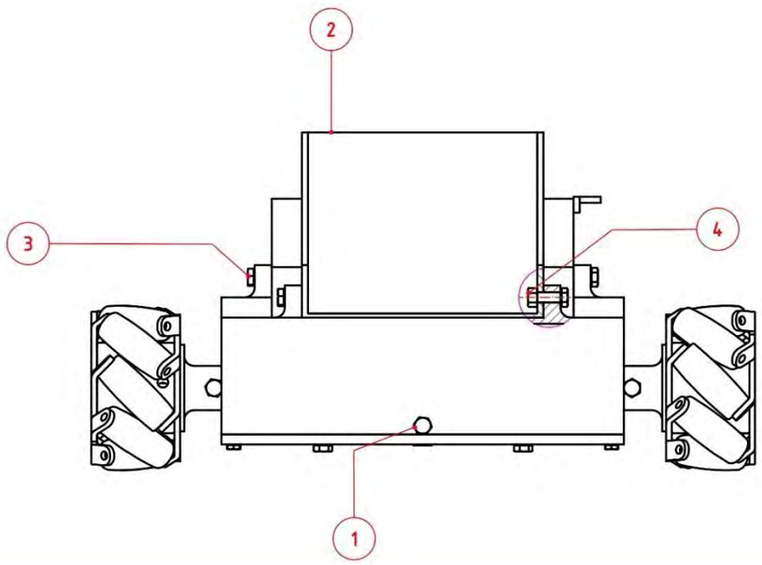
PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA – ING. MECATRÓNICA

MÉTODO DE PROYECCIÓN	1MTR02 - 0000 - HORARIO10M4 PLANO DE ENSAMBLE MODULO DE ACTUADORES	ESCALA 1:1
20191965	RUIZ FIGUEROA, SEBASTIAN	FECHA: 31.10.2023
REVISADO POR:		LÁMINA: A2

Anexo 75







IT	CANT	NOMBRE	DESCR.	NORMA	MATERIAL
6	1	Modulo Actuadores			
5	1	Modulo de Sensores			
4	4	AS 1112 - M5 Type 5		AS 1112	Steel, Mild
3	4	AS 1110 - M5 x 16		AS 1110	Steel, Mild
2	1	Modulo de Control			
1	1	Modulo Principal			

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA – ING. MECATRÓNICA

MÉTODO DE PROYECCIÓN	1MTR02 - 0000 - HORARIO10M4 PLANO DE ENSAMBLE COMPLETO	ESCALA 1:2
20191965	RUIZ FIGUEROA, SEBASTIAN	FECHA: 31.10.2023
REVISADO POR:		LÁMINA: A2

ET LUX IN TENEBRIS LUCET
MCMXVII