UNIVERSIDAD DE TARAPACÁ

FACULTAD DE INGENIERÍA

DEPARTAMENTO DE INGENIERÍA CIVIL EN COMPUTACIÓN E INFORMÁTICA

Ingeniería@
Computación e Informática

Plan de Proyecto
“ICarus”

Alumno(os): Martin Del Solar
Mayling Alvarez
Ivan Collao
Kamila Diaz
Yazuska Castillo

Asignatura: Proyecto I

Profesor: Humberto Urrutia López

Diciembre – 2024

Historial de Cambios

Versión	Descripción	Autor(es)
1.0	Concepción del Documento	Mayling Álvarez Martin Del Solar
1.1	Finalización de ítem 1	Kamila Diaz
1.2	Finalización de ítem 2	Yazuska Castillo
1.3	Finalización de ítem 3	Ivan Collao
1.4	Finalización de ítem 4	Martin Del Solar
1.5	Finalización de ítem 5	Mayling Álvarez
1.6	Finalización de ítem 6	Mayling Álvarez Yazuska Castillo
1.7	Finalización de ítem 7	Martin Del Solar Yazuska Castillo
1.8	Finalización de ítem 8	Martin Del Solar
1.9	Revisión y Finalización del Informe	Martin Del Solar Yazuska Castillo Mayling Álvarez Ivan Collao Kamila Diaz
1.10	Actualización de la conclusión	Iván Collao
2.0	Actualización para entrega final	Todos.

Tabla de Contenidos

1. Panorama General

1.1. Introducción

En este semestre, se evidenciará la labor en equipo realizada para alcanzar el objetivo de la materia de forma colaborativa, brindando una experiencia en ingeniería. Para lograrlo, se utilizará el kit educativo de LEGO Mindstorms Education EV3 para desarrollar un robot que pueda recoger objetos, además de poder movilizarse a través de una interfaz programada por el usuario en Python.

En esta presentación, no solo mostraremos la estructura y progreso de nuestro grupo para cumplir con los requisitos de la materia, sino también compartiremos información sobre la asignación de responsabilidades, la estrategia que hemos elegido y las acciones que estamos tomando para lograr los objetivos del proyecto. También se registrarán las primeras impresiones de este proceso, así como la investigación pertinente que se llevará a cabo a lo largo del semestre.

1.2. Objetivos

1.2.1. Objetivo General

Desarrollar y programar un robot EV3 que sea capaz de movilizarse según una interfaz de python.

1.2.2. Objetivos Específicos

- Experimentar con el Set de Lego Mindstorms Ev3 para la creación del robot.
- Armar y ensamblar un modelo con buena estabilidad, movilidad y un componente encargado para sujetar una pelota.
- Estudiar el sistema operativo de Linux, junto con la librería de Python de EV3, donde se investigará e implementará la instalación de ev3dev.
- Estudiar la librería de tkinter para generar y diseñar una interfaz gráfica apta para el usuario.

1.3. Restricciones

- Se debe programar solo en Python.
- Solo se debe utilizar la plataforma Redmine para los documentos y avance del proyecto.
- Se debe utilizar el Set de Lego Mindstorms EV3.
- Limitación de tiempo para dedicar al proyecto.
- Cantidad de integrantes limitada a solo 5.
- Disponibilidad del robot para codificar y probar.
- Robot debe ser capaz de moverse y tomar objetos con una garra.
- Se debe tener una conexión inalámbrica del robot hacia un servidor estando ambos en la misma red.

1.4. Entregables

- *Bitácoras*
- *Carta Gantt*
- *Informes*
- *Presentaciones*
- *Manual de usuario*
- *Interfaz*
- *Robot*
- *Wiki*

2. Organización del Personal

La organización en un grupo es esencial para el desarrollo de un trabajo, y para ello, es necesario una distribución del trabajo necesario para lograr el objetivo del proyecto.

2.1. Descripción de los Roles

Jefe de proyecto: Representante del equipo, supervisa y organiza el progreso del proyecto.

Ensamblador: Encargado del montaje y el armado de las piezas, monitorea el cumplimiento de las funcionalidades del robot, en conjunto con el programador.

Programador: Encargado del área de la codificación y funcionamiento del robot, en colaboración del ensamblador.

Documentador: Encargado de registrar el avance del proyecto, junto con la redacción de los informes.

Diseñador: Encargado de la creación del logotipo y la estética del proyecto.

2.2. Personal que Cumplirá los Roles

Rol	Responsable	Involucrados
Jefe de proyecto	Yazuska Castillo	Yazuska Castillo
Ensamblador	Kamila Díaz	Kamila Díaz Martin Del Solar Iván Collao Yazuska Castillo Mayling Alvarez
Diseñador	Mayling Alvarez	Mayling Alvarez
Programador	Yazuska Castillo	Yazuska Castillo Mayling Alvarez

Documentador	Martín del Solar	Martin Del Solar Mayling Alvarez

2.3. Métodos de Comunicación

Los principales medios de comunicación que utilizaremos son los siguientes: WhatsApp, que se utilizará para la mensajería, haciendo uso de los grupos que ofrece la plataforma; Discord, que será empleado como servicio de reuniones, aprovechando sus canales de texto y voz.

3. Planificación del Proyecto

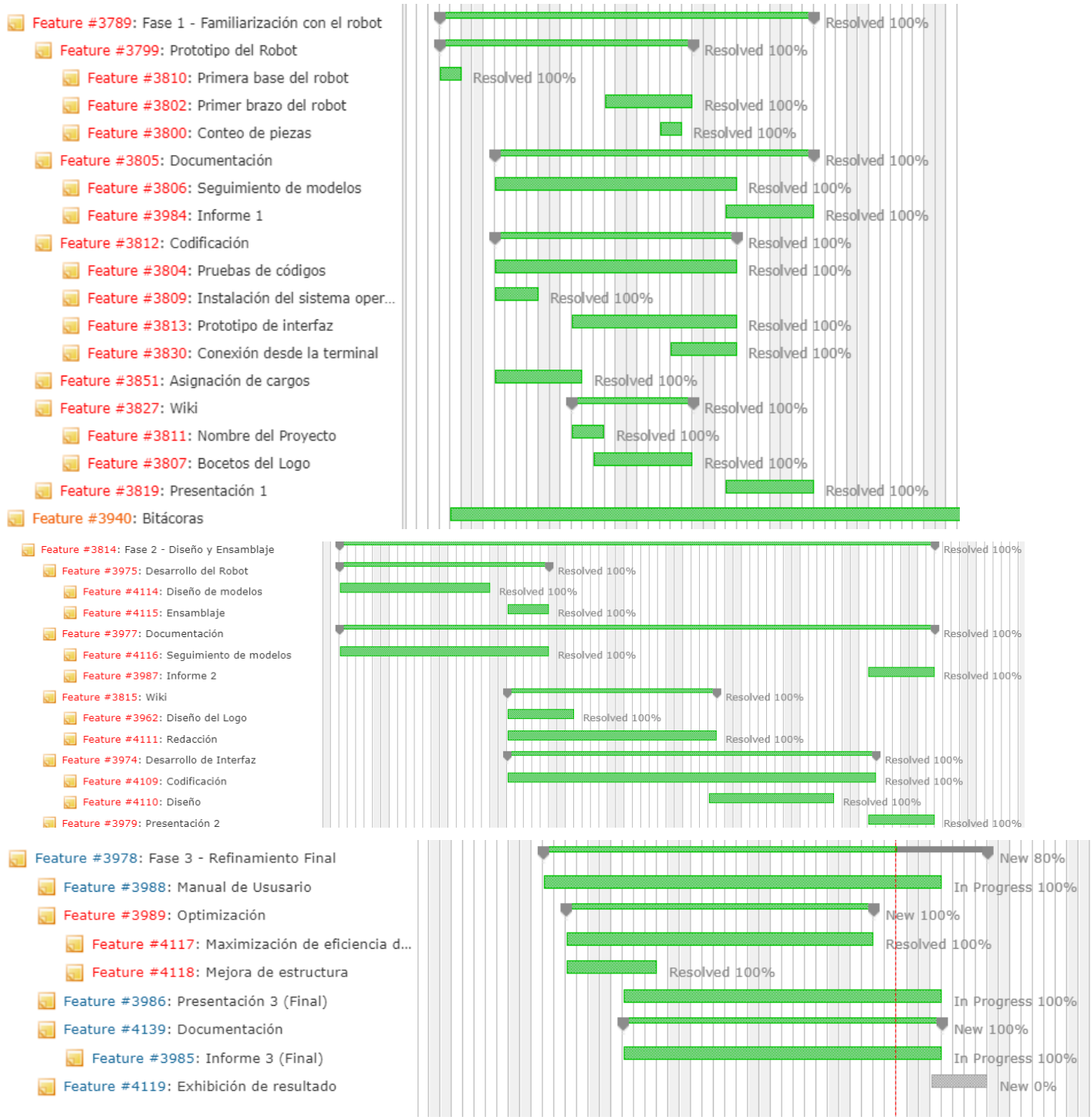
3.1. Actividades

Nombre	Descripción	Responsables	Producto
Preparación para el proyecto	Se realiza una investigación en las plataformas entregadas.	Todo el grupo.	Reprogramación de la tarjeta MicroSD.
Prototipo del robot	Se arma un modelo de prueba del robot, siguiendo una guía básica.	Ivan Collao Yazuska Castillo	Comprensión del uso de MicroPython
Experimentación con el robot	Pruebas de movilidad.	Yazuska Castillo Ivan Collao Mayling Álvarez	Comprensión básica de las librerías
Organización del proyecto	Planificación de roles y asignación	Todo el grupo.	Definición del nombre del proyecto
Primer modelo del robot	Se comienza el armado del robot	Martin Del Solar Kamila Diaz	Confección de garra para el robot.
Término del primer modelo	Se realiza el modelado 3D del primer modelo del robot	Mayling Álvarez	Planificación del movimiento
Conteo de piezas	Se realiza el inventario de piezas usadas	Kamila Diaz	Inventario de piezas
Programación de movimientos.	Se prueban los movimientos de la garra del robot	Martin Del Solar Kamila Díaz	Se consigue la codificación predeterminada del movimiento de la garra
Confección del primer informe de avance	Se realiza el primer informe de avance	Todo el grupo	Primer informe

Término de la producción de las bitácoras para la etapa 1	Se realizaron bitácoras semanales detallando el avance	Martin Del Solar	Bitácoras
Diseño del logo	Subcontratación de diseñador para la confección del logo.	Mayling Álvarez	Logo
Armar el modelo 3D	Se realizó el armado del modelo final, hecho previamente en digital.	Kamila Diaz Yazuska Castillo	Modelo final
Mejoras físicas al modelo final	Se agregaron piezas necesarias al modelo final.	Kamila Diaz Martin Del Solar	Modelo final mejorado.
Codificación de base para la interfaz	Codificación de una interfaz básica.	Yazuska Castillo Ivan Collao	Interfaz base
Diseño de la interfaz	Codificación el diseño oficial de la interfaz	Mayling Álvarez	Interfaz terminada
Redacción de la wiki	Creación de la wiki del proyecto	Mayling Álvarez Kamila Diaz	Wiki terminada
Bitácoras para etapa 2	Seguimiento semanal de avance a través de bitácoras	Martin Del Solar	Bitácoras.
Confección del segundo informe de avance	Se realiza el segundo informe, prolongando el anterior.	Todo el grupo.	Informe 2
Confección de la segunda presentación	Se armó la segunda presentación, en base del informe.	Mayling Álvarez	Segunda presentación.

Realización del Manual de usuario	Se redactó el manual de usuario.	Todo el grupo.	Manual de Usuario
Mejora del robot	Se realizaron pruebas para mejorar el robot	Yazuska Castillo	Robot mejorado.
Confección del informe final y la Presentación final.	Se desarrollan el informe final y la presentación final	Todo el grupo.	Exhibición de resultados finales. (Informe y Presentación)

3.2. Asignación de Tiempo



3.3. Gestión de Riesgos

Se presenta a continuación una tabla que exhibe un desglose de los problemas que se han presentado a lo largo de la primera fase del proyecto. Esta tabla resume el impacto de cada desafío al clasificar el daño en cinco niveles distintos. Cada nivel está asociado con diferentes tipos de daño:

1. Daño catastrófico: Las medidas a tomar en el caso son de forma inmediata, puede provocar que el proyecto se detenga o retrase significativamente, teniendo que volver a empezar desde cero.
2. Daño crítico: Se deben tomar medidas necesarias para resolver el riesgo, debido a que puede provocar que el proyecto se retrase en varias etapas.
3. Daño circunstancial: El riesgo se debe resolver en el momento, debido a que puede retrasar el desarrollo de una etapa base del proyecto.
4. Daño mínimo: El riesgo no es de mayor importancia, es un detalle imprevisto que no necesita mucha atención y se puede resolver en cualquier momento.
5. Daño recurrente: El riesgo no es significativo, pero es reiterativo, retrasa en las sesiones de trabajo, pero no en etapas.

Riesgo	Probabilidad de Ocurrencia	Nivel de impacto	Acción Remedial
Ausencia de piezas.	80%	4	Solicitar las faltantes al administrador de piezas.
El desempeño del robot no es eficiente.	60%	2	Ensamblar un robot más adecuado siguiendo guías en línea o un nuevo diseño adaptándolo a lo requerido.
Incomprensión de fallos con bibliotecas.	30%	1	Volver a *flashear* el microSD y verificar la correcta instalación de éstas, luego actualizarlas para confirmar.

Horario insuficiente para el cumplimiento de tareas en conjunto.	20%	4	Coordinamos los horarios disponibles del personal.
Falta de disponibilidad del equipo para la experimentación con el robot.	60%	2	Solicitar un nuevo ev3 brick para probar códigos y utilizarlo como base para el ensamblaje de componentes por separado.
Personal faltando al horario asignado de trabajo	70%	4	Adelanto de tareas del personal disponible para mayor accesibilidad en caso de que el personal faltante necesite ayuda para terminar a tiempo su encargo.
Descarga de batería del EV3.	50%	5	Utilizar cargador y seguir utilizándolo mientras carga.
Error en la codificación.	60%	5	Corregir errores sintácticos y lógicos en lo posible, de no serlo investigar una solución o explorar otro enfoque.
Recibir equipo defectuoso.	40%	1	Conseguir un reemplazo del equipo con el encargado de las piezas o prescindir de su uso.
Congelación del robot.	60%	5	Esperar 10 minutos por si se logra volver a conectar automáticamente, si no forzar el reinicio del robot.

Dificultades con la conexión wifi.	80%	3	Esperar 10 minutos por si se logra volver a conectar automáticamente, si no cambiar la conexión a una privada.
Atraso en el cumplimiento de tareas.	80%	3	Comunicar al equipo, y utilizar las horas extras disponibles.
Falla de registro en el redmine.	70%	1	Comunicar al administrador de la página para encontrar una solución.
Poca estabilidad en el movimiento del robot.	60%	3	Agregar más piezas e implementar mecanismos físicos.

4. Planificación de los Recursos

4.1. Hardware

- Set Lego Mindstorm EV3.
- Micro SD, del set de Lego Mindstorm, en el cual se podrán ejecutar las instrucciones del robot. (micro Python)
- Computador con el sistema operativo necesario para poder programar las instrucciones para el robot.

4.2. Software

- Sistema operativo Linux, para programar las funciones del robot.
- Redmine, página para la organización del proyecto.
- Visual Studio Code, editor de código.
- Canva.
- Krita.
- LDD (Lego Digital Designer).

4.3. Estimación de Costos

Costo de Hardware:

Producto	Precio
Set Lego Mindstorm(EV3)	$ 1.600.000
Asus vivobook 16X	$ 600.000
Lenovo Thinkpad x390 yoga	$ 1.138.755
Notebook HP ENVY 15-ep1501la (486K5LA) Con Processador Intel Core	$ 779.990
MacBook Pro Retina 13" i5 8GB RAM (128 GB SSD / Plata)	$ 600.000
Notebook Toshiba Tecra Z40 C1410LA P/N PT463U-07P01Y	$ 899.990
Apple iPad" Décima Generación (2024) 128GB Wi-Fi - plateado	$ 799.990
Apple Pencil 1ra Generación	$ 100.990
Tablet samsung galaxy tab s7 fe 12.4" 4gb ram negro de 64gb	$ 739.990
Micro SD	$ 11.990
Total:	$ 6.530.715

Costo de Software:

Producto	Precio
Total :	$ 0

Costo de Trabajador:

Rol	Horas	Horas Extra	Precio / Hora
Jefe de proyecto	24 horas	10 horas	$ 30.000
Programador	70 horas	10 horas	$ 25.000
Ensamblador	60 horas	10 horas	$ 24.000
Diseñador	50 horas	10 horas	$ 23.000
Documentador	70 horas	10 horas	$ 23.000
Total :	-	-	$ 7.920.000

Destacado:

- *La contabilización de las horas trabajadas comienza a partir de la formación del grupo de trabajo.*
- *Para la categorización de las horas de trabajo, se tuvo en cuenta el tiempo de trabajo en clases.*
- *Para la categorización de las horas extras, se tuvo en cuenta el tiempo en las que se trabajó fuera del horario de clase, pero dentro del mismo departamento.*

Total de Costo:

Costo Hardware	$ 6.530.715
Costo Software	$ 0
Costo Empleados	$ 7.920.000
Total :	$ 14.450.715

5. Análisis y diseño.

5.1. Especificación de requerimientos.

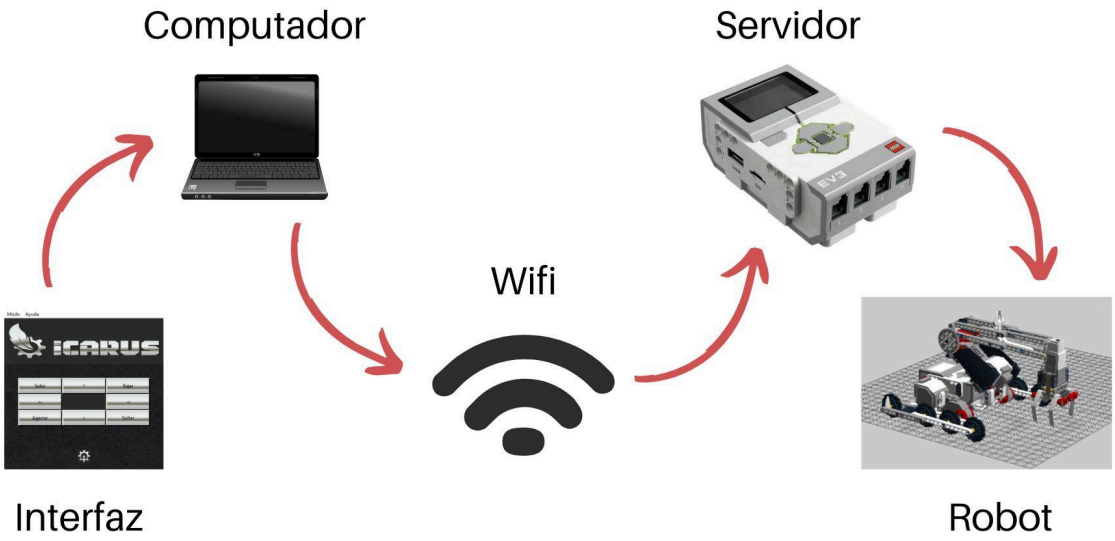
5.1.1. Requerimientos funcionales.

- El sistema debe incluir un robot capaz de conectarse a una red Wi-Fi y de recibir comandos a través de esta conexión para permitir su control remoto.
- Se debe desarrollar una interfaz gráfica en Python con tkinter, que sea intuitiva y fácil de usar, permitiendo al usuario controlar todas las funcionalidades del robot desde una computadora o dispositivo compatible.
- El robot debe ser capaz de desplazarse en diversas direcciones, incluyendo avance, retroceso y giros, para ajustar su ángulo y orientación según las instrucciones del usuario.
- El robot debe contar con una garra motorizada que permita acciones como subir, bajar, abrir y cerrar, ofreciendo precisión en tareas de manipulación de objetos.
- La interfaz debe ofrecer botones o controles específicos para acciones detalladas, como:
    - Control de movimiento: botones para avanzar, retroceder y girar.
    - Control de garra: opciones para elevar o bajar la garra y para abrirla o cerrarla.

5.1.2. Requerimientos no funcionales.

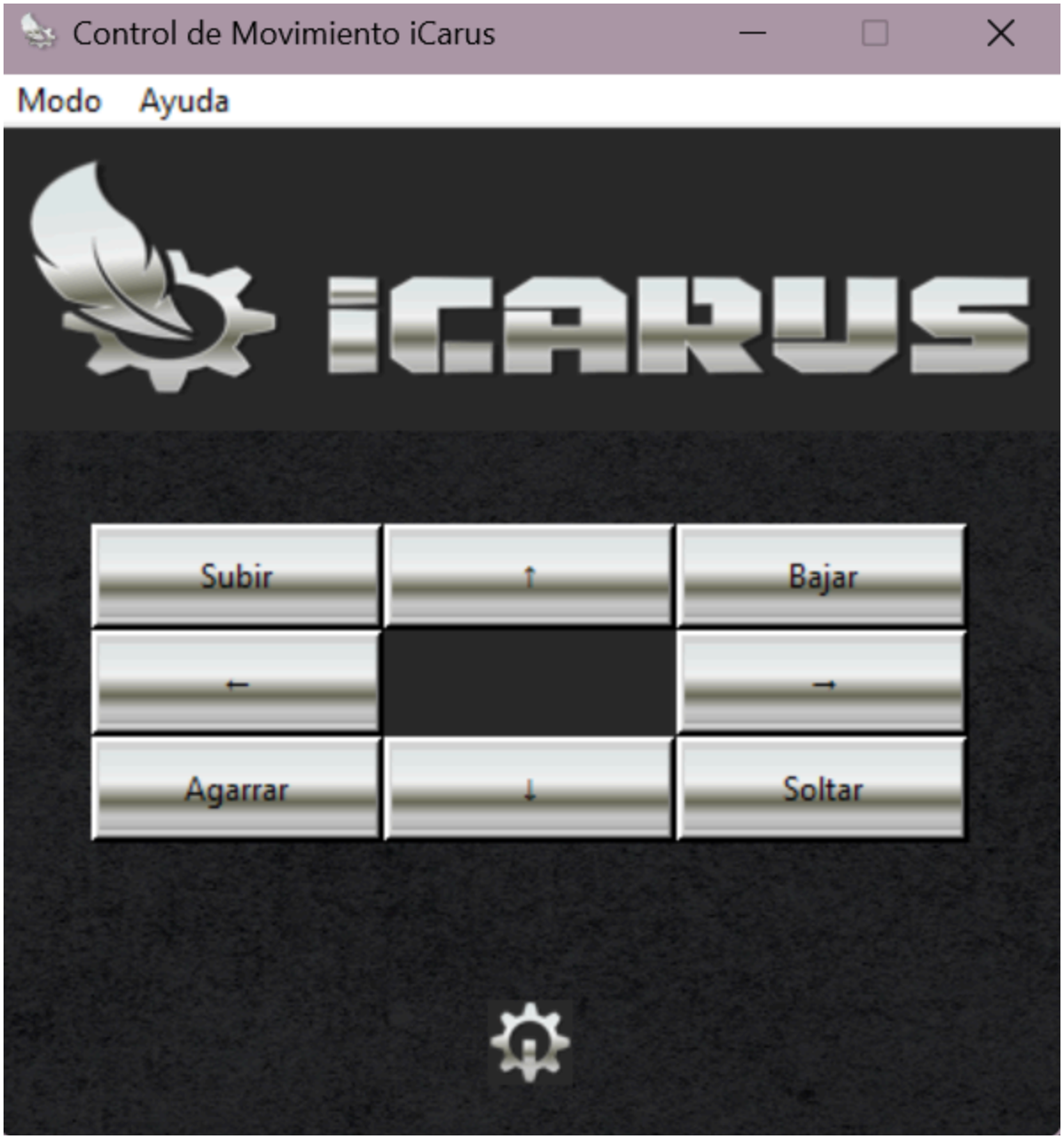
- El proyecto debe incluir un manual detallado con instrucciones completas sobre el funcionamiento integral del robot.
- La interfaz debe ser intuitiva y facilitar el control del robot a cualquier usuario, independientemente de su nivel técnico. Se deben emplear íconos y etiquetas claras para que las funciones sean fáciles de identificar y entender.
- El robot debe ser eficiente en su consumo de energía, permitiendo sesiones de uso prolongadas sin recarga frecuente. Esto implica una gestión óptima de la batería y el uso de componentes de bajo consumo.
- La interfaz debe ofrecer retroalimentación visual al usuario en tiempo real sobre el estado del robot.
- La estructura del robot debe ser duradera y adecuada para entornos de uso moderado, permitiendo que el robot funcione de manera confiable durante un tiempo prolongado sin desgaste prematuro.

5.2. Arquitectura.



1. Tanto el robot como la computadora deben estar conectados a la misma red Wi-Fi.
2. A continuación, se ejecutará la interfaz gráfica que permitirá controlar el robot a distancia.
3. Luego, se debe iniciar el servidor desde la interfaz y establecer la conexión con la computadora del usuario.
4. El usuario podrá controlar el robot una vez que la interfaz gráfica se haya conectado al servidor del robot.
5. Finalmente, el robot ejecutará los movimientos que el usuario envíe a través de la interfaz.

5.3. Interfaz.


Control de Movimiento iCarus
Modo
Ayuda
iCARUS
Subir
↑
Bajar
←
→
Agarrar
↓
Soltar

Resúmen de interfaz:

- Los botones con flechas verticales mueven el robot de arriba hacia abajo.
- Las flechas horizontales rotan el robot.
- El botón “Subir” eleva la garra.
- El botón “Bajar” baja la garra.
- El botón “Agarrar” cierra la garra.
- El botón “Soltar” abre la garra.
- El botón inferior con forma de tuerca conecta y desconecta el servidor.

En el menú superior en “Modo” se pueden cambiar los modos de movimiento, pudiendo seleccionar el modo “Mouse” para modificar los controles a los movimientos del mouse y el menú ayuda redirecciona al usuario según lo solicitado.

6. Implementación

6.1. Fundamentos físicos

El motor de LEGO 45544 es un motor grande, y su velocidad está definida en términos de rotaciones por minuto (RPM). En el caso del motor grande, el rango de velocidades va de 0 a 100, donde 100 es la velocidad máxima en RPM.

Para convertir las rotaciones por minuto a metros por segundo, se debe considerar lo siguiente:

1. Diámetro de la rueda o engranaje: La distancia recorrida por cada vuelta depende del tamaño de la rueda o engranaje conectado al motor, que para este caso es de 40.7 mm.

2. Número de revoluciones por segundo (RPS): 100 en la escala de velocidad del motor corresponde a 100 RPM (rotaciones por minuto), lo que se traduce a:

$$RPS = (100\,[RPM]) \div 60 = 1.67[\,RPS]$$

3. Distancia recorrida por cada revolución: Usamos la fórmula de la circunferencia de un círculo para calcular la distancia que recorre la rueda en cada revolución:

$$Distancia\ por\ revolución = \pi \times diámetro$$

Para una rueda de 40.7 mm de diámetro:

$$Distancia\ por\ revolución = \pi \times 0.0407[m] \approx 0.128[m]$$

4. Velocidad en metros por segundo: Finalmente, multiplicamos las revoluciones por segundo (1.67 RPS) por la distancia recorrida por revolución:

$$Velocidad = 1.67[RPS] \times 0.128[m] \approx 0.213[m/s]$$

Por lo tanto, cuando el motor de LEGO 45544 está funcionando a su máxima velocidad (100 [RPM]), con las ruedas sprocket de 40.7 mm de diámetro, la velocidad sería aproximadamente 0.213 metros por segundo.

El análisis realizado sobre la velocidad del motor LEGO 45544 muestra que, con una rueda sprocket de 40.7 mm de diámetro, el robot alcanza una velocidad de aproximadamente 0.213 m/s cuando el motor está funcionando a su máxima velocidad (100 RPM). Esta velocidad es adecuada para que el robot pueda moverse rápidamente y posicionarse con precisión alrededor de objetos, como las pelotas, sin que el movimiento sea tan rápido como para perder el control. Esto permite que el robot agarre las pelotas de manera eficiente, manteniendo un buen balance entre velocidad y precisión.

Sin embargo, a pesar de que la velocidad calculada es eficiente para las tareas de movimiento y captura, se ha observado que genera un tambaleo en la garra durante el

desplazamiento del robot. Este tambaleo se debe a las fuerzas dinámicas generadas por el movimiento rápido del robot, que provocan oscilaciones en la garra, afectando su estabilidad y precisión en tareas de agarre.

Para mitigar este problema, se especula que un péndulo amortiguado podría ser una solución eficaz. Este péndulo ayudaría a absorber las oscilaciones generadas por el movimiento del robot, actuando como un amortiguador que disipa la energía cinética y reduce el efecto de tambaleo de la garra. El uso de un péndulo permitiría que la garra se mantenga más estable, mejorando su efectividad en la captura de objetos sin perder control o precisión.

En términos de física, el péndulo amortiguado seguiría la ecuación del movimiento oscilatorio amortiguado:

$$\theta(t) = \theta_0 e^{-\gamma t} cos(\omega' t + \phi)$$

Donde γ es el coeficiente de amortiguamiento que controla la cantidad de disipación de energía y ω es la frecuencia angular amortiguada del sistema. La implementación de este sistema de amortiguación permitiría al robot mantener un movimiento más controlado y una garra más estable mientras se desplaza a su velocidad máxima.

Por lo tanto, aunque la velocidad actual del robot es adecuada para tareas de captura de pelotas, el tambaleo de la garra sigue siendo un desafío a resolver. Implementar un péndulo amortiguado es una solución a futuro que podría mejorar la estabilidad general del robot y optimizar la ejecución de sus tareas, garantizando que el robot sea tanto rápido como preciso al manipular objetos.

6.2. Descripción de los programas

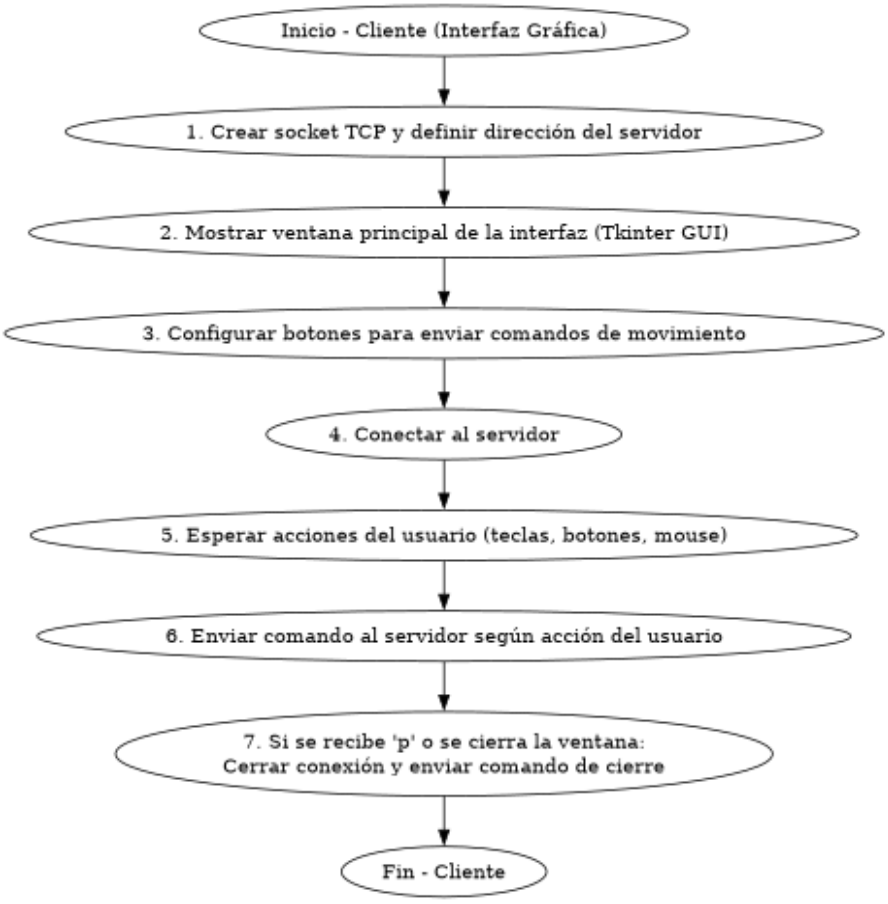
El programa consta de dos partes principales, la que se ejecuta en el robot, el servidor, y la que se ejecuta en la computadora, el cliente. Para establecer una conexión correcta, se debe asegurar que el puerto y la dirección IP asignadas al robot sean las mismas asignadas al cliente.

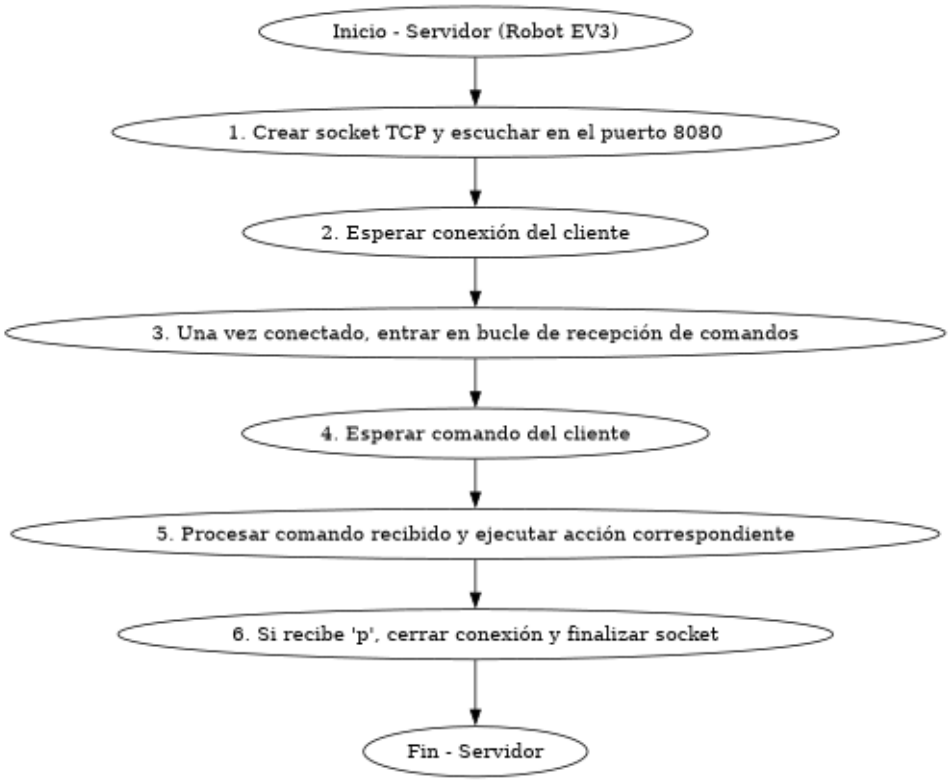
- servidor.py: Es la parte del programa que se ejecuta en el robot, el cual recibe datos desde un cliente para utilizar las funciones de Function para moverse según el comando recibido.

- cliente.py: Inicializa el movimiento de motores para ser utilizados por el servidor de forma simple. Contiene funciones para agarrar, soltar, adelante, atrás, izquierda, derecha.

- Function.py: Este se encarga de conectarse al robot según la ip y el puerto dados, contiene la interfaz para controlar al robot tanto en pantalla como por comandos de teclado.

6.3. Diagramas




Inicio - Servidor (Robot EV3)
1. Crear socket TCP y escuchar en el puerto 8080
2. Esperar conexión del cliente
3. Una vez conectado, entrar en bucle de recepción de comandos
4. Esperar comando del cliente
5. Procesar comando recibido y ejecutar acción correspondiente
6. Si recibe 'p', cerrar conexión y finalizar socket
Fin - Servidor

7. Resultados

7.1. Estado Final del Proyecto

Actualmente, el proyecto ha logrado los siguientes avances:

- Terminación de la versión del robot: La construcción del robot ha sido finalizada en su totalidad.
- Conexión remota exitosa: Se ha establecido correctamente la conexión con el robot.
- Desarrollo de la interfaz gráfica con Tkinter: Se ha implementado una interfaz funcional en Python utilizando la librería Tkinter para facilitar la interacción del usuario con el robot.
- Implementación de funciones de movimiento: El robot puede desplazarse de forma óptima y es capaz de agarrar una pelota de ping-pong.
- Análisis de diagrama de Gantt: revisar y actualizar el plan del proyecto.
- Registros de actividades, informes y presentaciones: Preparar registros, informes y presentaciones para registrar y comunicar el progreso y los resultados alcanzados.
- Desarrollo del servidor: Se diseñó un servidor llamado "Server.py" utilizando la librería "socket" de Python en el entorno EV3Dev para facilitar la comunicación.

7.2. Problemas encontrados y solución utilizada

1. Congelamiento del robot
    - Descripción: de manera inesperada, al terminar una conexión con el cliente, el robot se queda congelado el el menú principal
    - Solución: reinstalar o cambiar el almacenamiento del sistema operativo del robot

2. Exceso en el motor de la garra
    - Descripción: Cuando al robot se le ordena agarrar o soltar múltiples veces desde el cliente, el eje de la garra se desplaza ligeramente, haciendo que la repetición de esto resulte en una garra no 100% funcional y requiriendo el ajuste manual de esta.
    - Solución: implementar una verificación en el programa tal que, el robot no intente agarrar nuevamente cuando la garra está cerrada.

8. Pruebas

8.1 Descripción de pruebas realizadas

Se llevaron a cabo pruebas de manejo y movimiento para evaluar el rendimiento de la garra construida con LEGO Mindstorms EV3. Estas pruebas se centraron en verificar la estabilidad de la garra al sujetar objetos y la velocidad del movimiento durante su funcionamiento. Además de la prueba para la implementación de nuevos sensores.

8.2 Resultados de pruebas finales

Durante las pruebas, se detectaron y corrigieron los siguientes problemas:

- La garra no era lo suficientemente estable, lo que dificultaba el agarre efectivo de los objetos. Este inconveniente se solucionó mejorando el diseño estructural, lo que resultó en un agarre más firme y seguro.
- El movimiento de la garra era demasiado lento, lo que afectaba la eficiencia general del sistema. Para resolver este problema, se incrementó ligeramente la velocidad del motor responsable del movimiento, optimizando su rendimiento.
- La posibilidad de implementar un sensor ultrasónico para que el robot pudiera agarrar objetos sin la necesidad de manejarlo desde la interfaz, no fue eficiente y se descartó.

Estas modificaciones permitieron mejorar tanto la estabilidad como la rapidez del sistema, logrando un funcionamiento más eficaz y confiable.

9. Conclusión

El proyecto del robot Lego Mindstorm EV3 ICarus ha sido concluido exitosamente, cumpliendo con todos los objetivos planteados inicialmente. Se implementó todas las características requeridas, incluyendo la capacidad de movimiento y el control de la garra mediante la interfaz gráfica en Python. Las pruebas que se realizaron demostraron que el robot responde de manera correcta a los comandos emitidos por el cliente, y los ajustes finales en el control de la garra garantizan un desempeño preciso y confiable.

En resumen, el proyecto ha logrado no solo cumplir con los objetivos técnicos y de aprendizaje iniciales, sino que también ha proporcionado grandes lecciones sobre la integración de hardware y su complementación con el software. Estos logros sientan una base para futuros proyectos en el ámbito de la robótica y la programación.

10. Referencias

1. LEGO Group. (2013). LEGO Mindstorms EV3 User Guide. LEGO Education. https://education.lego.com

2. Bagnall, B. (2014). Maximum LEGO EV3: Building Robots with Java Brains. No Starch Press.

3. LEGO Education. (2020). LEGO Mindstorms EV3 Resources. https://education.lego.com/en-us/support/mindstorms-ev3

4. Pybricks Team. (2023). Pybricks Documentation for LEGO Mindstorms EV3. Pybricks. https://docs.pybricks.com

5. GitHub. (2021). ev3dev - Linux for LEGO Mindstorms EV3. https://github.com/ev3dev/ev3dev

6. LEGO Education. (2020). Programming with Python on LEGO Mindstorms EV3. https://education.lego.com/en-us/support/mindstorms-ev3/python

7. Hassenplug, S. (2018). Getting Started with Python for LEGO EV3. https://hassenplug.com/ev3python

8. Lawlor, R. (2016). Learning LEGO Mindstorms EV3 Programming. Apress.

9. Red Hat. (2019). ev3dev Stretch Documentation for LEGO Mindstorms EV3. https://www.ev3dev.org/docs/getting-started/

10. LEGO Group. (2019). LEGO Mindstorms EV3 Education Core Set. LEGO Education.

11. Smolinski, K. (2019). LEGO Mindstorms EV3 Robot Inventor 's Guide. Packt Publishing.

12. Pybricks Team. (2023). Getting Started with Pybricks for EV3. https://pybricks.com/start/ev3/

13. LEGO Group. (2021). LEGO Mindstorms Software Guide. LEGO Education. https://education.lego.com/en-us/

14. Lund, H. H. (2014). RoboCup Jr. Soccer with LEGO Mindstorms EV3. RoboCup Federation.

15. Chalmers, J. (2015). LEGO EV3 Robotics: A Guide to Building and Programming.