UNIVERSIDAD DE TARAPACÁ

FACULTAD DE INGENIERÍA

Departamento de Ingeniería en Computación e Informática

Informe Final
Kimera

Autor(es): Javier Choque Orellana

Juan Pilco Casapia

Leonel García Arocutipa

Aarón Saravia Saravia

Asignatura: Proyecto I

Profesor(es): Leonel Alarcón Bravo

ARICA, 12 diciembre 2022

Historial de Cambios

Fecha	Versión	Descripción	Autor(es)
18/08/2022	1.0	Planificación de cómo será el proyecto.	Juan Pilco Leonel García Diego Alarcón Javier Choque
25/08/2022	1.1	Cambio del proyecto base a un proyecto definitivo.	Juan Pilco Leonel García Diego Alarcón Javier Choque
01/09/2022	1.2	Construcción del robot **Matador**.	Juan Pilco Leonel García
08/09/2022	1.4	Se cambió de nombre a **Kimera**.	Juan Pilco Leonel García Diego Alarcón
15/09/2022	1.5	Construcción del lanzador.	Juan Pico Leonel García Diego Alarcón
22/09/2022	1.6	Construcción del aparato de levantamiento.	Juan Pilco Leonel García
29/09/2022	1.7	Construcción del soporte.	Juan Pilco Leonel García
24/11/2022	1.8	Intento de conexión SSH	Juan Pilco Aarón Saravia
05/12/2022	1.9	Mejora de interfaz y servidor	Juan Pilco Aarón Saravia
15/12/2022	2.0	Corrección del sensor	Juan Pilco Aarón Saravia

Tabla de Contenidos

9. Conclusiones
    9.1 Conclusiones generales
    9.2 Trabajo Futuro

10. Referencias (utilizando el estándar IEEE)

Anexos
    Anexo A: Hardware (diagrama de construcción del robot, componentes principales)
    Anexo B: Software
    Anexo C: Comunicaciones (configuración de comunicación pc-robot considerando RPyC o Socket)

Panorama General

1.1) Introducción:

En este informe se explicará los avances del proyecto final, nombrado "**Kimera**", la organización del equipo, las planificaciones del proyecto, y detallar las funcionalidades del robot o lo que sería el resultado de todo este trabajo.

1.2) Objetivo General:

Desarrollo de un robot llamado "**Kimera**" con la capacidad de apuntar de forma vertical y horizontal detectando con sensor de ultrasonido el objetivo al que disparar como un lanzamisiles.

1.3) Objetivo Específicos:

- Mostrar los resultados finales del proyecto **"Kimera".**
- Usando el lenguaje de programación Python mostrar los últimos avances del proyecto.

1.4) Restricciones:

- **Piezas faltantes:** Hubo problemas con algunas piezas por lo que se fue improvisando o reemplazando con lo que estaba disponible para el cuerpo del robot.
- **Tiempo Estimado:** Dado el tiempo de 4 horas aún seguimos teniendo problemas con el avance ya sea inasistencias de algunos compañeros por pruebas o unas emergencias.
- **Inasistencias de compañeros:** En ciertos días se fue faltando en los días martes o jueves debido a problemas fuera, llevando a retrasos en el desarrollo del proyecto.
- **Motivación del grupo:** Hubo días donde la motivación al armar o programar no era la misma afectando el desarrollo del robot debido a discusiones grupales, sueño o problemas personales.
- **Días suspendidos o recesos:** Dependiendo de la situación, puede suspenderse la reunión para el proyecto afectando el progreso ya sea por actividades, feriados o emergencias.
- **Distancia:** No todos los miembros del grupo viven en la ciudad sino fuera (campo, parcelas, etc.) dificultando su llegada a las clases o juntas llegando más tarde de lo debido.
- **Problemas de conexión:** Como se mencionó anteriormente, hay miembros que viven fuera de la ciudad con una conexión inestable lo cual dificulta las reuniones online o compartir información sobre el proyecto.
- **Cambio de miembros**: A finales hubo eliminación de compañeros debido a la inasistencia perjudicando a la mayoría de grupos con cambios de proyecto o de miembros.

1.5) Entregables:

- **Manual:** Instructivo que indica de forma explicativa la forma correcta de usar el robot.
- **Piezas Lego:** Piezas que fueron usadas y deben mantenerse en buen estado sin perderse mientras se va construyendo el proyecto.
- **Robot (Kimera):** Proyecto que se fue desarrollando dependiendo las reuniones y la función que cumplirá cuando esté terminado.

Organización del Personal

2.1) Descripción de roles:

- **Jefe de proyecto:** Dirige el proyecto del grupo y se encarga de ver el avance del proyecto.
- **Programadores:** Programador que desarrolla el software del proyecto.
- **Organizador:** Documenta y registra los avances del proyecto.
- **Constructores:** Construyen y manipulan el hardware del proyecto.
- **Consultores:** Realiza trabajos de asistencias a los otros roles como relevos en caso de una ausencia.

2.2) Personal asignado:

- **Jefe de proyecto:** Juan Pilco.
- **Programadores:** Leonel García, Juan Pilco, Aarón Saravia.
- **Organizadores:** Javier Choque, Aarón Saravia.
- **Constructores:** Leonel García, Juan Pilco.
- **Consultores:** Javier Choque, Aarón Saravia.

2.3) Mecanismo de comunicación:

Las Apps que se usaron para una mejor organización y comunicación del grupo se divide en dos categorías la principal donde el grupo siempre habla y hay mejor comunicación y las secundarias que son programas que no se usan mucho pero son de emergencia o para dudas.

Entre los secundarios está Whatsapp donde los organizadores pueden hablar debido a que no todos tienen la App descargada no siendo el mejor medio para comunicarse, Telegram se usa mayormente para dudas o fechas siendo el menos usado.

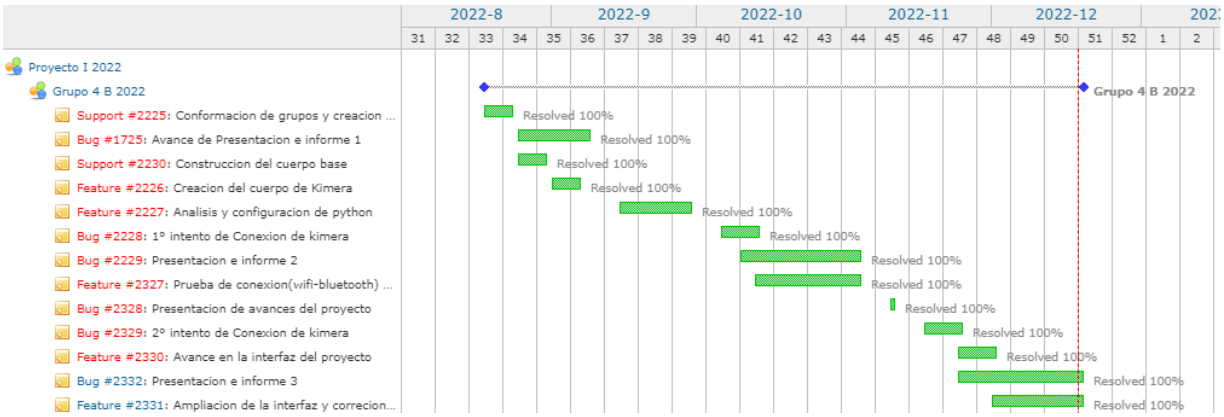
- **Mecanismo principal:** Discord.
- **Mecanismo secundario:** Telegram, Whatsapp.

Planificación del Proyecto

3.1) Actividades:

Nombre	Descripción	Responsables	Producto
Matador	Lanzador de proyectiles	Leonel García Juan Pilco	Robot Lanzamisiles
Matador	Mecanismo que levante el cañón.	Javier Choque Leonel García Juan Pilco	Robot Lanzamisiles
Kimera	Documentación del proyecto	Javier Choque Aarón Saravia	Robot Lanzamisiles
Kimera	Base del robot	Leonel García Juan Pilco	Robot Lanzamisiles
Kimera	Conexión con la computadora	Juan Pilco Javier Choque Aarón Saravia	Robot Lanzamisiles
Kimera	Codificación de la interfaz y servidor	Juan Pilco Aarón Saravia Leonel García	Robot Lanzamisiles
Kimera	Ampliación de la interfaz	Juan Pilco Aarón Saravia	Robot Lanzamisiles
Kimera	Corrección del sensor	Juan Pilco Aarón Saravia Leonel García	Robot Lanzamisiles

3.2) Asignación de tiempo:



3.3) Gestión de riesgos:

Tabla con posibles riesgos que se tienen a consideración al realizar el proyecto. El nivel de impacto se ordena de la siguiente forma:

1. Catastrófico.
2. Crítico.
3. Marginal.
4. Despreciable.

Riesgo	Probabilidad de ocurrencia	Nivel de impacto	Acción remedial
Falta de herramientas.	90%	4	Buscar y reemplazar herramientas faltantes.
Mal estimación de tiempo.	60%	3	Reorganizar avance de tal manera de solapar el tiempo perdido.
Limitaciones de conocimiento.	50%	3	Investigar sobre el tema que genere el conflicto.
Ausencia del personal.	50%	4	Reorganizar la jornada a raíz de la ausencia.
Cliente cambie los requisitos.	30%	2	Estudiar cambios y analizar el impacto a futuro.
Poco compromiso del personal.	20%	3	Reorganizar el personal de tal forma que no haya atrasos en el avance del proyecto.
Dañar el equipo.	20%	2	Reemplazar equipo o herramientas defectuosos por otros de igual uso.
Perdida de avance del proyecto.	10%	1	Buscar archivos guardados sobre el avance.
Personal conflictivo.	10%	3	Reorganizar la estructura interna del equipo y analizar la situación del equipo.
Recibir equipo defectuoso.	10%	2	Reemplazar equipo o herramientas defectuosos por otros de igual uso.
Cancelación de proyecto.	10%	1	Guardar avance del proyecto y abandonarlo.

Planificación de los Recursos

4.1) Recursos (Hardware - Software):

Hardware:

- **Computador:** Tuvimos a disposición un computador fijo para la programación del proyecto.
- **Piezas Lego (Mindstorms ev3):** Piezas lego usadas para la construcción del robot.
- **Controlador maestro (master controller EV3):** Es la cabeza principal para que el robot funcione.
- **Motores y sensores:** Los motores se encargan de darle el movimiento al robot mientras que los sensores sirven para detectar el entorno a su alrededor.

Software:

- **Microsoft office:** Programa para la realización de bitácoras e informes en PPT y la carta Gantt.
- **Redmine:** Página dada por el profesor para subir los avances e informes del proyecto de forma semanal.
- **Discord:** Aplicación usada por el grupo para comunicarse de forma online.
- **Visual Studio Code:** Aplicación que usamos para hacer el programa del proyecto.
- **Diagrams.net:** Página web donde realizamos el diagrama.
- **Putty:** Programa que permite conectarse al master controller EV3.
- **Rpyc:** Programa que se conecta junto a Putty para un mejor control del EV3.

4.2) Estimación de costos:

- **Lego Mindstorm Ev3:** $599.000.
- **Sueldo de programador:** $900.000.
- **Sueldo de Organizador:** $800.000.
- **Ev3dev:** $0.

Análisis – Diseño

5.1) Especificación de Requerimientos:

Funcionales:

- El robot puede disparar a varias direcciones.
- Puede detectar objetos con medidas en centímetros.
- Podemos moverlo para que dispare a cierta dirección.

No funcionales:

- Dispara al presionar el botón disparar.
- Localiza a través de sensores de ultrasonido.
- Gira en horizontal por los motores que hacen girar a los engranajes.

5.2) Arquitectura Propuesta:

Los componentes como los motores y sensores, se manejan mediante el código del servidor, ya que las funciones del EV3 están importadas ahí.

Y el programa Putty hace posible la conexión mediante el protocolo SSH, conexión por la cual controlaremos el robot, utilizando las funciones con una interfaz gráfica.

5.3) Diseño de la Interfaz Usuario:

Al iniciar Kimera posee dos botones iniciales “Controles y Ultrasonido” que lleva al manejo del robot o medir la distancia con el ultrasonido, y el botón Apagar que cierra la ventana de la interfaz.

Implementación

$$y = y_0 + v_{0y}t - \frac{1}{2}gt^2$$

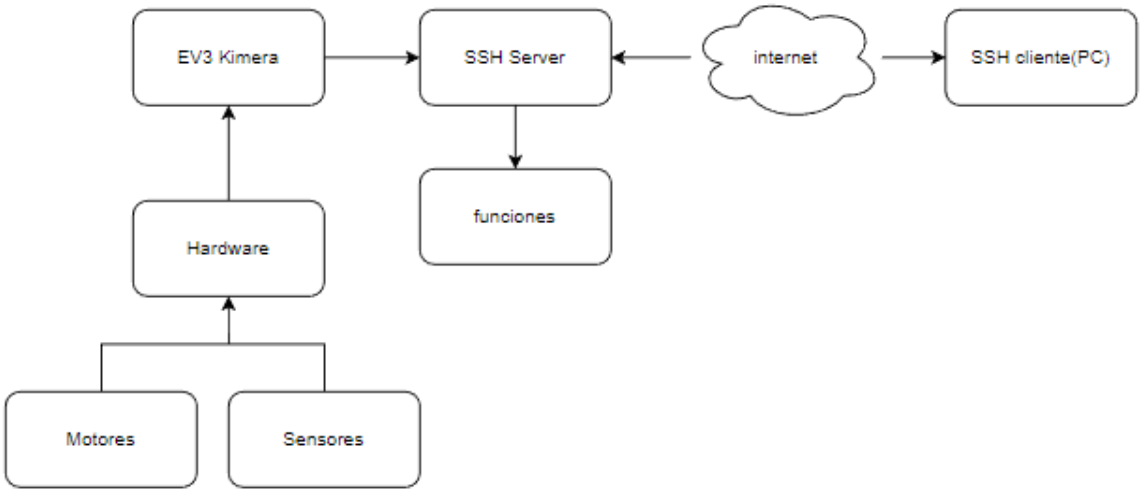
6.1) Fundamentos de proyectiles u otro:

El proyecto se relaciona con el movimiento de proyectil, que es el movimiento de un objeto lanzado, en función de la aceleración con la gravedad. No se alcanzó a implementar esta fórmula en nuestro proyecto.

6.2) Descripción de los programas:

- **Putty:** para establecer una conexión al servidor por protocolo SSH.
- **Rpyc:** mediante la conexión proporciona los comandos necesarios para el funcionamiento de los controladores.
- **Python:** el lenguaje de programación que se utiliza para programar las funciones del robot.
- **Visual Studio Code:** editor de código en donde modificaremos.

6.3) Diagrama:



Pruebas

7.1) Pruebas realizadas:

1) **Testeo del motor EV3 de forma manual:** Testeo del motor en los primeros días para probar su software antes de instalar el Python con la tarjeta de memoria o incluirlo en la construcción del proyecto.

2) **Prueba de motor y sensor:** Prueba del funcionamiento del motor y el sensor una vez hecho una construcción base del proyecto para ver si sus puertos funcionaban o daba un error.

3) **Conexión con Python**: Se intentó establecer una conexión del python al EV3 para modificar la interfaz que sería el controlador junto al servidor que incluye todas las funciones como el disparo o el movimiento.

4) **Movimiento de Kimera con el controlador**: Fue probado el movimiento de la torreta Kimera con una interfaz que da instrucciones para moverse vertical y horizontalmente, o disparar al objetivo que tiene frente.

7.2) Resultados de las pruebas:

1. **Testeo del motor EV3 de forma manual:** Funciona perfectamente.

2. **Prueba de motor y sensor (software):** En un principio daba problemas porque no funcionaban algunos comandos de la clase `ev3dev2.motor` con algunos motores por ej. (run_timed).

3. **Conexión con Python**: Al inicio nos daba error porque el bloque no conectaba al wifi la solución fue cada vez que cambiemos de red reiniciar el bloque.

4. **Movimiento de Kimera con el controlador**: Mediante Putty donde fue codificado por comandos de la clase `ev3dev2.motor` y la interfaz hecha en tkinter por medio de vs Code se hizo los controladores para controlar los movimientos del robot.

Resultados

8.1) Estado final del proyecto:

Contiene una interfaz que muestra los botones que le dará instrucciones a los motores el cual le dará movimiento, ya sea vertical o horizontal, también un botón el cual efectuará el disparo, mediante los sensores también se muestra a qué distancia está el objetivo que tenga enfrente.

8.2) Problemas encontrados y soluciones propuestas:

- **Falta de tiempo:** Debido a que hubo días con receso o feriados el tiempo se fue limitando dejando menos días o tiempo justo para el avance del proyecto "**Kimera**".
    - **Solución:** Organizar juntas fuera del horario de clases para una mejor organización y avanzar de manera eficaz en el proyecto evitando atrasos con el código o su construcción.
- **Problema con la conexión:** Hubo problemas con la conexión debido a que al principio no se conectaba por wifi o bluetooth retrasando la programación al robot.
    - **Solución:** Con la ayuda del profesor se resolvió el problema y pudo conectarse correctamente gracias al programa Putty.
- **Falta de comunicación:** en algunos momentos no supimos entendernos y no supimos organizarnos atrasando todo a última fecha.
    - **Solución:** Entendernos entre todo el grupo mejor y organizar las juntas a la disponibilidad de todos.
- **Atrasos o faltas de compañeros:** En las reuniones a veces llegaban atrasados 2 o 3 miembros del grupo en el horario de clases o en el peor de los casos no venía nadie perjudicando el avance del proyecto.
    - **Solución:** Hablar con los compañeros para que llegaran aun si vienen atrasados o no quisieran venir debido a que la inasistencia puede traer problemas a futuro.

Conclusiones

9.1) Conclusiones generales:

Estamos conformes con el resultado del proyecto, aunque no pudimos implementar las últimas ideas que tuvimos, si logramos completar los puntos generales. También se ha aprendido mucho sobre su programación y armado en este proyecto dándonos una mejor noción para la próxima.

9.2) Trabajo Futuro:

Dado los problemas y las dificultades que tuvo el equipo, no se podría continuar el proyecto, dejándolo para crear uno mejor, usando como base lo aprendido en el proyecto y no caer en los mismos errores que se cometieron y con el cual no habría muchas complicaciones como el anterior.

Referencias

1. *Motor classes — python-ev3dev 2.1.0.post1 documentation*, https://ev3dev-lang.readthedocs.io/projects/python-ev3dev/en/stable/motors.html#medium-ev3-motor

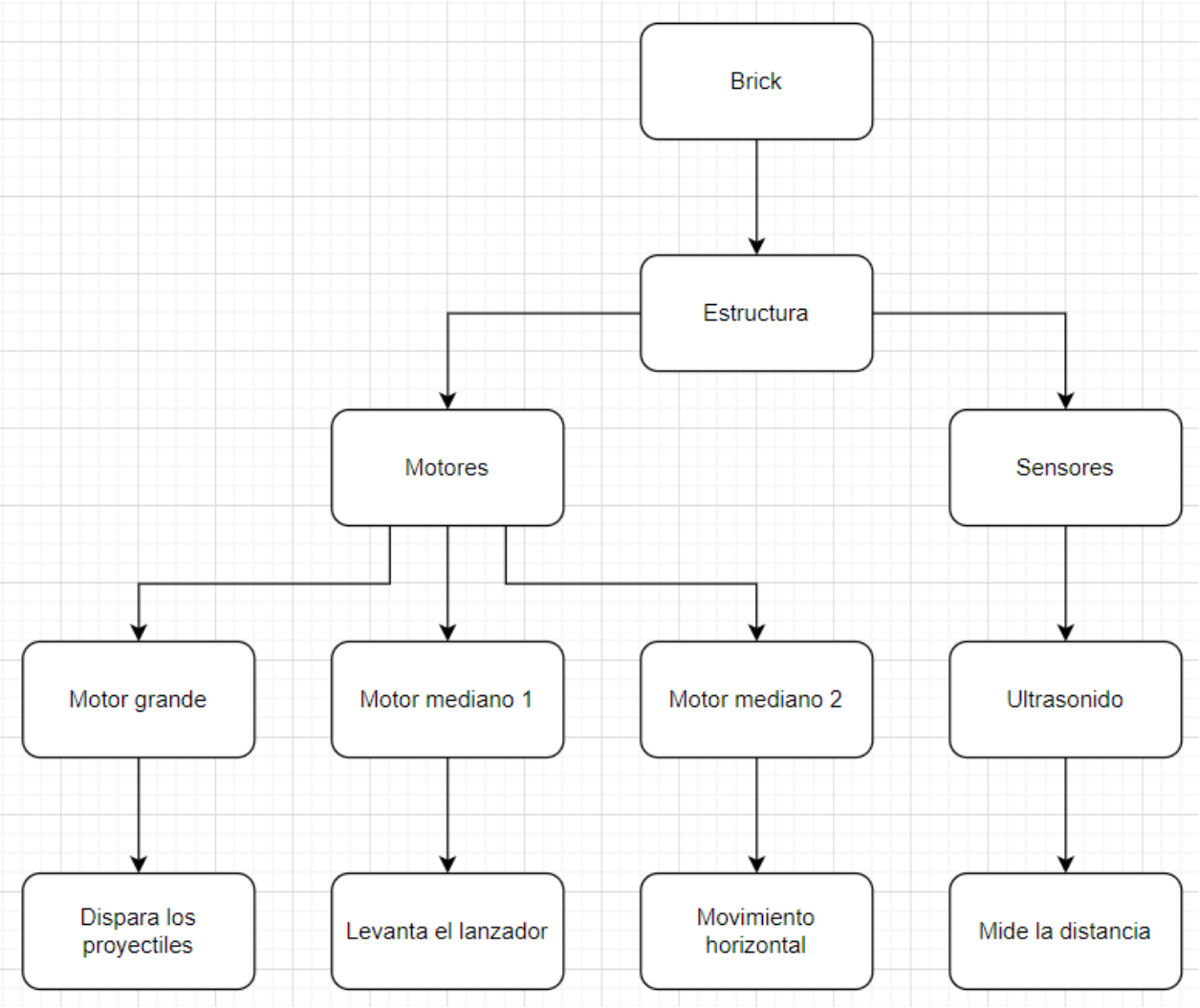
2. *Sensor classes — python-ev3dev 2.1.0.post1 documentation*, https://ev3dev-lang.readthedocs.io/projects/python-ev3dev/en/stable/sensors.html

3. *Next: Documentación del editor de texto Vi,* https://docs.oracle.com/cd/E19620-01/805-7644/6j76klopr/index.html. Accessed 19 December 2022.

4. *“Movimiento de proyectil - Física universitaria volumen 1.”* https://openstax.org/books/f%C3%ADsica-universitaria-volumen-1/pages/4-3-movimiento-de-proyectil.

Anexos

Anexo A: Hardware



Anexo B: Software

```
import sys
from tkinter import *
from tkinter import messagebox
import rpyc

PORT = 12345
IP = '192.168.84.254'

conn = rpyc.connect(IP,port=PORT)

running = False
cm = 0

def controlesKimera():
    def start_up(event):
        global running
        running = True
        # AQUI HACER QUE EL MOTOR SE MUEVA AL PRESIONAR | motor.run_forever(speed_sp=900)
        conn.root.Arriba()
        print("arriba")

    def stop_up(event):
        global running
        #AQUI HACER QUE EL MOTOR SE DETENGA AL SOLTAR | motor.stop(stop_action="hold")
        conn.root.Detener()
        print("parar arriba")
        running = False

    def start_down(event):
        global running
        running = True
        # AQUI HACER QUE EL MOTOR SE MUEVA AL PRESIONAR | motor.run_forever(speed_sp=900)
        print("abajo")
        conn.root.Abajo()

    def stop_down(event):
        global running
        #AQUI HACER QUE EL MOTOR SE DETENGA AL SOLTAR | motor.stop(stop_action="hold")
        print("parar abajo")
        conn.root.Detener()
        running = False

    def start_right(event):
        global running
        running = True
        # AQUI HACER QUE EL MOTOR SE MUEVA AL PRESIONAR | motor.run_forever(speed_sp=900)
        conn.root.Izquierda()
        print("derecha")
    def stop_right(event):
        global running
        #AQUI HACER QUE EL MOTOR SE DETENGA AL SOLTAR | motor.stop(stop_action="hold")
        conn.root.Detener()
        print("parar derecha")
        running = False

    def start_left(event):
        global running
        running = True
        # AQUI HACER QUE EL MOTOR SE MUEVA AL PRESIONAR | motor.run_forever(speed_sp=900)
        conn.root.Derecha()
        print("izquierda")

    def stop_left(event):
        global running
        #AQUI HACER QUE EL MOTOR SE DETENGA AL SOLTAR | motor.stop(stop_action="hold")
        conn.root.Detener()
        print("parar izquierda")
        running = False

    def shoot(event):
        global running
        #AQUI HACER QUE EL SENSOR MIDA LA DISTANCIA | ultrasonicsensor.value()
        conn.root.disp()
        print("disparar")
        running = False

    def stop_shoot(event):
        global running
        #AQUI HACER QUE EL SENSOR MIDA LA DISTANCIA | ultrasonicsensor.value()
        conn.root.Detener()
        print("disparar")
        running = False

    def show_distance(event):
        global running
        global cm
        #AQUI HACER QUE EL SENSOR MIDA LA DISTANCIA | ultrasonicsensor.value()
        print("mostrar distancia")
        print(conn.root.sensor())
        labelMedicion['text'] = conn.root.sensor()
        running = False
```

```
    ventana2 = Toplevel(ventanaPrincipal)
    ventana2.title("Controles y Ultrasonido")
    ventana2.resizable(0,0)
    ventana2.config(bg="DarkGrey")

    labelControles= Label(ventana2, text= "Controles" ,bg= "DarkGrey" ,fg= "Black" ,font= ("Bold", 14) ,width= 7 , height= 2)
    labelControles.grid(row= 0, column= 1, padx= 5, pady= 5)

    arriba= Button(ventana2, text= "^", width= 10, height= 2, bg= "DarkGrey")
    arriba.grid(row= 1, column= 1, padx= 5, pady= 5)
    arriba.bind('<ButtonPress-1>', start_up)
    arriba.bind('<ButtonRelease-1>', stop_up)

    abajo= Button(ventana2, text= "v", width= 10, height= 2, bg= "DarkGrey")
    abajo.grid(row= 3, column= 1, padx= 5, pady= 5)
    abajo.bind('<ButtonPress-1>', start_down)
    abajo.bind('<ButtonRelease-1>', stop_down)

    izquierda= Button(ventana2, text= "<", width= 5, height= 4, bg= "DarkGrey")
    izquierda.grid(row= 2, column= 0, padx= 5, pady= 5)
    izquierda.bind('<ButtonPress-1>', start_left)
    izquierda.bind('<ButtonRelease-1>', stop_left)

    derecha= Button(ventana2, text= ">", width= 5, height= 4, bg= "DarkGrey")
    derecha.grid(row= 2, column= 2, padx= 5, pady= 5)
    derecha.bind('<ButtonPress-1>', start_right)
    derecha.bind('<ButtonRelease-1>', stop_right)

    dispara= Button(ventana2, text= "Disparar", width= 10, height= 4, bg= "DarkGrey")
    dispara.grid(row= 2, column= 1, padx= 5, pady= 5)
    dispara.bind('<ButtonPress-1>', shoot)
    dispara.bind('<ButtonRelease-1>', stop_shoot)

    labelUltrasonido= Label(ventana2, text= "Ultrasonido" ,bg= "DarkGrey" ,fg= "Black" ,font= ("Bold", 14) ,width= 20 , height= 2)
    labelUltrasonido.grid(row= 0, column= 3, padx= 5, pady= 5)
    ultrasonido= Button(ventana2, text="Medir Distancia", width= 12, height= 2, bg= "DarkGrey")
    ultrasonido.grid(row= 1, column= 3, padx= 5, pady= 5)
    ultrasonido.bind('<ButtonPress-1>', show_distance)
    labelMedicion= Label(ventana2, text= "Esperando Medicion..." ,bg= "DarkGrey" ,fg= "Black" ,width= 20 , height= 2)
    labelMedicion.grid(row= 2, column= 3, padx= 5, pady= 5)

def apagar():
    messagebox.showinfo(message="Programa detenido y apagado", title="Notificación")
    ventanaPrincipal.destroy()

#### Programa Principal ####
#Inicializa el cuadro de Tkinter y le pone un titulo
ventanaPrincipal = Tk()
ventanaPrincipal.title("Proyecto Kimera Grupo 4 EV3")
ventanaPrincipal.resizable(0,0)
ventanaPrincipal.config(bg="DarkGrey")

# Pone una etiqueta en la ventana ventanaPrincipal con el titulo del programa.
tituloVentanaPrincipal= Label(text= "Kimera" ,bg= "DarkGrey" ,fg= "Black" ,font= ("Bold", 50), height= 1)
tituloVentanaPrincipal.grid(row= 0, column= 0)

opcion1= Button(ventanaPrincipal, text= "Controles y Ultrasonido", width= 45, height= 2, bg= "DarkGrey", command= controlesKimera)
opcion1.grid(row= 1, column= 0, padx= 5, pady= 5)

opcion2= Button(ventanaPrincipal, text= "Apagar", width= 45, height= 2, bg= "DarkGrey", command= apagar)
opcion2.grid(row= 2, column= 0, padx= 5, pady= 5)

ventanaPrincipal.mainloop()
```

```
import rpyc
from rpyc.utils.server import ThreadedServer
from ev3dev2.motor import Motor
from ev3dev2.motor import MediumMotor
from ev3dev2.sensor.lego import UltrasonicSensor

class MyService(rpyc.Service):
    def exposed_Arriba(self):
        rueda = Motor('outC')
        rueda.run_forever(speed_sp=200)

    def exposed_Abajo(self):
        rueda = Motor('outC')
        rueda.run_forever(speed_sp=-200)

    def exposed_disp(self):
        canon = Motor('outA')
        canon.run_forever(speed_sp=100)

    def exposed_Derecha(self):
        rueda = Motor('outB')
        rueda.run_forever(speed_sp=400)


    def exposed_Izquierda(self):
        rueda = Motor('outB')
        rueda.run_forever(speed_sp=-400)

    def exposed_Detener(self):
        canon = Motor('outA')
        rueda = Motor('outB')
        ruedal = Motor('outC')

        canon.run_forever(speed_sp=0)
        rueda.run_forever(speed_sp=0)
        ruedal.run_forever(speed_sp=0)

    def exposed_sensor(self):
        us= UltrasonicSensor('in1')
        us.mode='US-SI-CM'
        distance=us.value()/10
        print(distance)
        return distance

if __name__=='__main__':
    print("inicio")
    s = ThreadedServer(MyService, port=12345)
    s.start()
```

Anexo C: Comunicaciones

La comunicación pc-robot es mediante protocolo SSH con el programa Putty en el cual tenemos que estar conectados en la misma red Wi-Fi en ambos equipos donde tendremos que introducir la ip y un puerto para ingresar al SSH server en donde importamos Rpyc nos da acceso a los comandos del ev3 y por último en la interfaz dará instrucciones para utilizar las funciones contenidas en el SSH server.