UNIVERSIDAD DE TARAPACÁ

FACULTAD DE INGENIERÍA

Departamento de Ingeniería en Computación e Informática

Informe Final del Proyecto
FLIP-TAC-TOE

Autor(es): Ernesto García

Gustavo Olivares

Angelina Orozco

Benjamín Poblete

Daniel Ramírez

Asignatura: Proyecto 1

Profesor(es): Ricardo Valdivia

ARICA, 09 de enero del 2020

Historial de Cambios

Fecha	Versión	Descripción	Autor(es)
27/08/2019	1.0	Versión preliminar del formato	Benjamín Poblete
04/09/2019	1.1	Revisión y modificación del plan	Angelina Orozco Benjamín Poblete
05/09/2019	1.2	Revisión y modificación del plan	Angelina Orozco Benjamín Poblete
12/09/2019	2.0	Corrección y modificación del plan	Angelina Orozco Benjamín Poblete
24/09/2019	2.1	Corrección y modificación del plan	Angelina Orozco Benjamín Poblete
26/09/2019	2.2	Corrección y modificación del plan	Angelina Orozco Benjamín Poblete
01/10/2019	2.3	Desarrollo del plan de avance	Benjamín Poblete
03/10/2019	2.4	Desarrollo del plan de avance	Benjamín Poblete Daniel Ramírez
08/10/2019	2.5	Desarrollo del plan de avance	Benjamín Poblete
10/10/2019	2.6	Desarrollo del plan de avance	Benjamín Poblete Ernesto García
15/10/2019	2.7	Desarrollo del plan de avance	Benjamín Poblete Daniel Ramírez
16/10/2019	2.8	Desarrollo del plan de avance	Benjamín Poblete Angelina Orozco
11/12/2019	2.9	Corrección y modificación del plan final	Benjamín Poblete Angelina Orozco
16/12/2019	3.0	Corrección y modificación del plan final	Benjamín Poblete Angelina Orozco
20/12/2019	3.1	Corrección y modificación del plan final	Benjamín Poblete Angelina Orozco
06/01/2020	3.2	Corrección y modificación del plan final	Benjamín Poblete Angelina Orozco

Tabla de Contenidos

I. PANORAMA GENERAL

A. INTRODUCCIÓN

En el presente informe se buscará poner en antecedentes acerca de cómo se trabajará en el proyecto “Flip-tac-toe”, este consiste en la creación de un robot hecho con piezas del kit lego Mindstorms EV3 education, que deberá jugar Flip-Tac-Toe. El kit de lego viene con un mini computador denominado “Brick” que servirá como el cerebro del robot, a través de este se ejecutarán los programas de movimientos. Finalmente, Flip-tac-toe es un juego similar al Tic-Tac-Toe (conocido en Chile como “gato”), la diferencia se haya en que los jugadores tienen la posibilidad de dar vueltas las fichas de su contrincante pudiendo convertir a estas en fichas suyas, el juego se gana teniendo tres fichas en una línea. Con respecto a lo anteriormente dicho, el robot construido tendrá que jugar Flip-Tac-Toe junto al robot de otro grupo utilizando una lata que servirá como ficha y bajo reglas establecidas entre todos los demas grupos que trabajen en el proyecto. Durante el juego se debe empezar desde un punto donde se debe volver luego de poner o voltear una ficha, no se podrá botar al robot rival o se deberá volver al inicio, sin embargo, si se pueden chocar entre robots, si se falla al momento de voltear una lata, dejando caer a la misma, se debe volver al punto de inicio, el juego no es por turnos, todo depende de la velocidad de los robots.

B. OBJETIVO GENERAL

Construir un robot lego Mindstorms EV3 education que sea capaz de jugar flip-tac-toe.

C. OBJETIVOS ESPECIFICOS

1. Diseñar al robot para que sea capaz de moverse, tomar latas y girarlas.

2. Programar los movimientos necesarios del robot para lograr las acciones requeridas.

3. Calibrar los movimientos del robot para verificar un correcto funcionamiento.

4. Realizar una interfaz remota para controlar a distancia los movimientos del robot.

5. Integrar y entregar al producto final para poder jugar Flip-Tac-Toe.

D. RESTRICCIONES

1. La programación será realizada en el lenguaje de programación Python.
2. El proyecto debe ser realizado en el plazo de un semestre académico.
3. Trabajar con el kit Lego Mindstorms EV3 education.

E. ENTREGABLES

1. Bitácoras semanales
2. Informe de formulación de proyecto.
3. Presentación de formulación de proyecto.
4. Informe de avance.
5. Presentación de avance.
6. Informe final.
7. Presentación final.
8. Manual de usuario.
9. Wiki del proyecto.
10. Video del producto final.
11. Producto final: Robot Flip-Tac-toe.

II. ORGANIZACIÓN DEL PERSONAL

A. DESCRIPCIÓN DE ROLES

Programadores: Se encargan de desarrollar los programas en Python que permitan al robot moverse y utilizar su garra para que tome latas y las gire.

Diseñadores: Se encargan de diseñar la apariencia del robot de manera que este tenga las partes necesarias para moverse libremente, tomar latas y girarlas.

Documentadores: Se encargan de la documentación del proyecto, de esta forma, realizan las bitácoras del proyecto y formulan los informes requeridos.

Jefe de grupo: Se encarga de la coordinación del trabajo del grupo y de la comunicación con el profesor.

B. PERSONAL QUE CUMPLIRÁ LOS ROLES

Programadores: Gustavo Olivares-Angelina Orozco

Diseñadores: Ernesto García-Daniel Ramírez

Documentadores: Benjamin Poblete-Angelina Orozco

Jefe de grupo: Daniel Ramírez

C. MECANISMOS DE COMUNICACIÓN

El grupo se comunicará principalmente mediante la aplicación de VoIP (siglas en inglés de Voice over IP: 'voz sobre IP'), Discord, en esta podemos notificarnos con mensajes y conversar directamente, siempre que lo necesitemos. De igual manera, si algún integrante no se puede comunicar con la app mencionada, se recurrirá a la aplicación de mensajería WhatsApp.

III. PLANIFICACIÓN DEL PROYECTO

A. ACTIVIDADES

- Crear diseño básico del robot
  Descripción: Se crea la base del robot básico que viene por defecto en las instrucciones del paquete del robot.
  Responsable: Ernesto García
  Producto: Robot básico

- Configuración básica del robot EV3 e instalación del SO ev3dev
  Descripción: Se realizan las configuraciones básicas del robot para que funcione y se instala el SO ev3dev para que el sistema operativo basado en Linux corra en el robot.
  Responsable: Angelina Orozco
  Producto: Robot con sistema operativo integrado

- Buscar diseños de garras para el robot
  Descripción: Se busca diferentes diseños de garra, pinza o brazo para el robot en sitios de internet para que pueda realizar el movimiento importante el cual se basa en el juego Flip-Tac-Toe, que es poder tomar el objeto, en este caso una lata, lo gire o lo suelte.
  Responsable: Benjamín Poblete
  Producto: Idea de cómo realizar un brazo para el robot

- Diseñar la garra del robot
  Descripción: Se diseña la garra, pinza o brazo para el robot utilizando las piezas del kit Mindstorm EV3.
  Responsable: Daniel Ramírez
  Producto: Brazo para el robot

- Buscar diseños de rotación para el robot
  Descripción: Se busca diseños para la rotación de garra, pinza o brazo para el robot en sitios de internet.
  Responsable: Ernesto García
  Producto: Idea de cómo diseñar un sistema de rotación para el brazo

- Crear diseño del sistema de rotación
  Descripción: Se crea el diseño en el cual se basa el sistema de rotación de la garra del robot.
  Responsable: Ernesto García
  Producto: Sistema de rotación para el brazo del robot

- Investigar la programación para los movimientos básicos del robot
  Descripción: Se busca información para poder programar en Python los movimientos básicos del robot.
  Responsable: Gustavo Olivares
  Producto: Conocimientos básicos en Python para programar

- Programar los movimientos básicos del robot
  Descripción: Se realiza los algoritmos en Python para realizar los movimientos básicos del robot.
  Responsable: Gustavo Olivares
  Producto: Algoritmos de movimientos básicos del robot

- Investigar la programación para los movimientos de la garra
  Descripción: Se busca información para poder programar los movimientos de la garra del robot.
  Responsable: Gustavo Olivares
  Producto: Idea de cómo programar el movimiento de la garra del robot

- Programar los movimientos de la garra del robot
  Descripción: Se realizan los algoritmos en Python para realizar los movimientos de la garra del robot.
  Responsable: Gustavo Olivares
  Producto: Algoritmos de movimientos de la garra del robot

- Investigar la programación para la rotación del brazo
  Descripción: Se busca información para poder programar la rotación del brazo.
  Responsable: Gustavo Olivares
  Producto: Ideas de cómo crear un algoritmo para la rotación del brazo

- Programar la rotación del brazo
  Descripción: Se realiza los algoritmos en Python para realizar la rotación del brazo del robot.
  Responsable: Gustavo Olivares
  Producto: Algoritmo de rotación del brazo del robot

- Realizar pruebas de funcionalidad (Primera parte)
  Descripción: Se realizan pruebas de funcionalidad de los movimientos, ya sea, del robot o de la garra.
  Responsable: Gustavo Olivares
  Producto: Ideas y observaciones que permitan mejorar al robot

- Rediseño del robot (Primera parte)
  Descripción: Se rediseña al robot a partir de las pruebas hechas para que se adapte a las necesidades requeridas.
  Responsable: Ernesto García
  Producto: Segunda versión del robot

- Calibrar los movimientos del robot (Primera parte)
  Descripción: Se calibran los movimientos del robot para que funcionen correctamente.
  Responsable: Angelina Orozco
  Producto: Algoritmos de los movimientos del robot ajustados

- Realizar el informe de formulación
  Descripción: Se realiza el primer informe del proyecto que incluye la formulación de cómo se llevará a cabo el proyecto.
  Responsable: Benjamín Poblete
  Producto: Informe de formulación

- Realizar la presentación de la formulación del proyecto
  Descripción: Se realiza la primera presentación del proyecto en la que se expondrá la formulación del proyecto.
  Responsable: Angelina Orozco
  Producto: Presentación de la formulación del proyecto

- Investigar sobre la realización de una interfaz gráfica
  Descripción: Se investiga en diferentes sitios de internet sobre cómo realizar una interfaz para que sea posible interactuar con los movimientos del robot a través de opciones en el computador o de un celular.
  Responsable: Gustavo Olivares
  Producto: Ideas de cómo realizar una interfaz remota para controlar al robot

- Instalar la librería RPyC en ev3dev/Python
  Descripción: Se instala la librería RPyC en ev3dev/Python para el mecanismo de comunicación remota.
  Responsable: Gustavo Olivares
  Producto: Librería RPyC instalada

- Desarrollar el mecanismo de comunicación remota con el robot
  Descripción: Se desarrolla el mecanismo de comunicación remota con el robot.
  Responsable: Gustavo Olivares
  Producto: Mecanismo de comunicación remota

- Desarrollo del código de la interfaz
  Descripción: Se desarrolla en Python el código de la interfaz remota.
  Responsable: Gustavo Olivares
  Producto: Código en Python de la interfaz remota

- Realizar el diseño de la interfaz grafica
  Descripción: Se realiza el apartado visual de la interfaz gráfica.
  Responsable: Gustavo Olivares
  Producto: Diseño visual de la interfaz grafica

- Realizar la documentación de los códigos
  Descripción: Se realiza la documentación del código del servidor y del cliente.
  Responsable: Angelina Orozco
  Producto: Código del servidor y cliente documentado

- Realizar el informe de avance del proyecto
  Descripción: Se realiza el informe de avance del proyecto que involucra el desarrollo que ha tenido el proyecto.
  Responsable: Benjamín Poblete
  Producto: Informe de avance del proyecto

- Realizar la presentación del avance del proyecto
  Descripción: Se realiza la presentación del avance del proyecto en donde se expondrá el progreso que ha tenido el proyecto.
  Responsable: Angelina Orozco
  Producto: Presentación de avance del proyecto

- Realizar pruebas de funcionalidad (Segunda parte)
  Descripción: Se realizan pruebas de funcionalidad de los movimientos a través de la interfaz, ya sea, del robot o de la garra.
  Responsable: Angelina Orozco
  Producto: Ideas y observaciones que permitan mejorar al robot

- Rediseñar al robot (Segunda parte)
  Descripción: Se rediseña al robot a partir de los resultados obtenidos en las pruebas de funcionalidad.
  Responsable: Ernesto García
  Producto: Tercera versión del robot

- Calibrar los movimientos del robot (Segunda parte)
  Descripción: Se calibran los movimientos del robot para su correcto funcionamiento.
  Responsable: Angelina Orozco
  Producto: Algoritmos de los movimientos del robot ajustados

- Se realiza el videoclip del proyecto
  Descripción: Se realiza un video en donde se presenta al robot del proyecto y sus capacidades.
  Responsable: Daniel Ramírez
  Producto: Video clip del proyecto

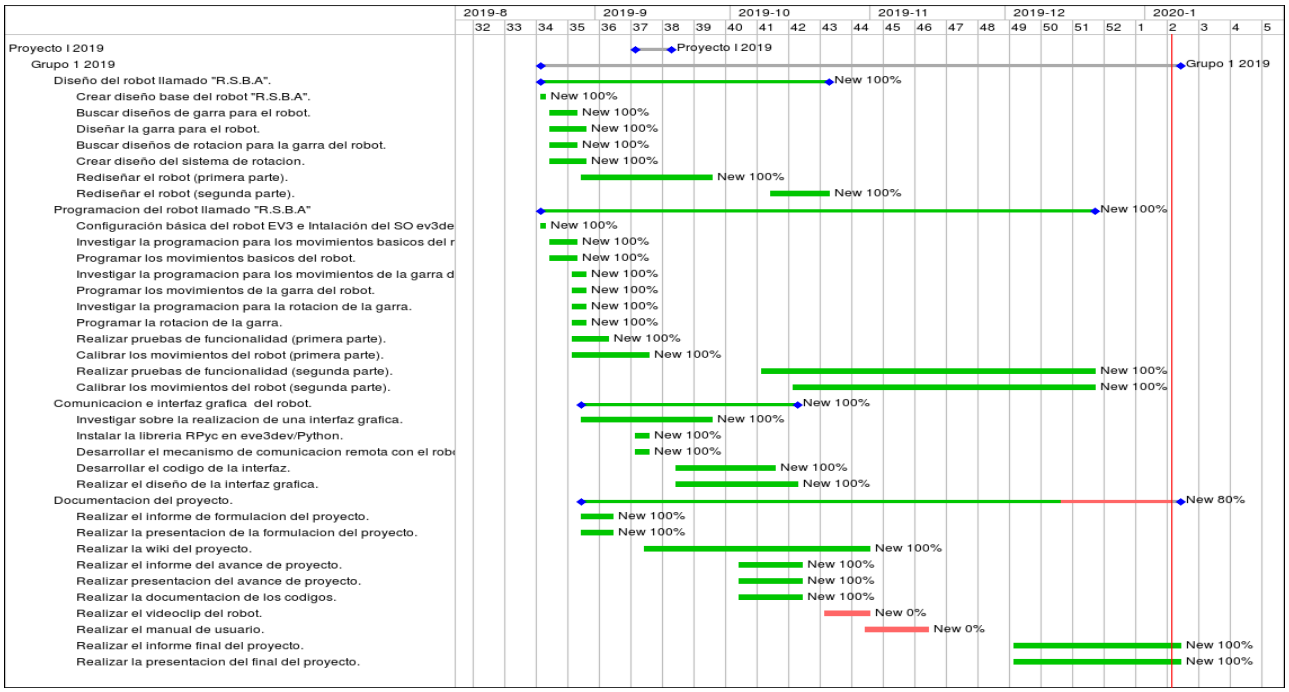
- Realizar la Wiki del proyecto
  Descripción: Se realiza una Wiki de Redmine acerca del proyecto que servirá como una guía de la realización del robot requerido.
  Responsable: Benjamín Poblete
  Producto: Wiki del proyecto

- Realizar el manual de usuario
  Descripción: Se realizará un manual de usuario que permita entender cómo utilizar el robot.
  Responsable: Benjamín Poblete
  Producto: Manual de usuario

- Realizar el informe final del proyecto
  Descripción: Se realiza un informe final que involucra al producto final del proyecto.
  Responsable: Benjamín Poblete
  Producto: Informe final del proyecto

- Realizar la presentación final del proyecto
  Descripción: Se realiza la presentación final en la que se expone al producto final del proyecto.
  Responsable: Angelina Orozco
  Producto: Presentación final

B. ASIGNACIÓN DEL TIEMPO



Como se puede apreciar en la carta Gantt, el apartado de diseño del robot esta completo, los programas de los movimientos están listos, implementados y calibrados, la interfaz gráfica también está terminada, finalmente solo faltaría terminar el video clip del proyecto y el manual de usuario.

C. GESTIÓN DE RIESGOS

RIESGOS	PROBABILIDAD DE OCURRENCIA	NIVEL DE IMPACTO	ACCIÓN REMEDIAL
Las piezas disponibles en el kit no serán suficientes para el diseño.	25%	4	Construir las piezas con una impresora 3D o conseguirse las piezas con otro grupo.
La tarjeta SD se dañe.	60%	1	Comprar una nueva tarjeta SD.
Los programas se pierdan debido a un error en la tarjeta SD.	60%	1	Crear respaldos en distintos pc de los integrantes del grupo.
Un integrante del grupo se encuentre indisponible.	40%	3	Suplir su falta con otro integrante del grupo.
El robot es desarmado por alguna causa externa.	40%	2	Tener imágenes actualizadas del diseño del robot para rearmarlo.
Que ocurra una falla en los motores	60%	1	Reemplazar los motores dañados.

Niveles de impacto:

- 1: Catastrófico
- 2: Crítico
- 3: Marginal
- 4: Despreciable

IV. PLANIFICACIÓN DE LOS RECURSOS

A. RECURSOS HARDWARE-SOFTWARE

- Notebooks
- Tarjeta micro SD clase 10 de 16GB.
- Software de desarrollo Python 3.
- Software de conexión Putty.
- EV3DEV Linux
- EV3DEV 2 (Librerías de funciones del robot en Python)
- Dongle USB WIFI TP LINK
- Kit LEGO Mindstorms EV3 education
- Router
- Microsoft Office

B. ESTIMACIÓN DE COSTOS

Recurso	Valor	Cantidad
Notebooks	$800.000	5
Tarjeta micro SD	$4.000	1
Software de desarrollo Python	De libre acceso	2
Software de conexión Putty	De libre acceso	2
EV3DEV Linux	De libre acceso	2
EV3DEV 2 (Librerías de funciones del robot en Python)	De libre acceso	2
Dongle USB WIFI TP LINK	$10.000	1
Kit LEGO Mindstorms EV3 education	$300.000-$450.000	1
Router	$70.000	1
Microsoft Office	$40.000	5
Sueldo total de cada integrante	$900.000	5

Costo total del proyecto	$9.084.000-$9.234.000

V. ANÁLISIS-DISEÑO

A. ESPECIFICACIÓN DE REQUERIMIENTOS

- **Requerimientos funcionales:**

Código	*Descripción*	*Prioridad*	*Revisión*	*Aceptado*	*Realizado*
1	El robot deberá poder moverse a voluntad del usuario mediante una interfaz en pc/smartphone.	1	26/09/2019	✓	✓
2	El robot debe tener la capacidad de tomar una lata con su brazo al presionar un botón especifico.	1	26/09/2019	✓	✓
3	El robot debe tener la capacidad de rotar una lata en su brazo al presionar un botón especifico.	1	26/09/2019	✓	✓
4	El robot debe tener la capacidad de soltar una lata con su brazo al presionar un botón especifico.	1	10/10/2019	✓	✓
5	El robot debe ser capaz de jugar “Flip-Tac-Toe”.	1	17/10/2019	✓	✓

- **Requerimientos no funcionales:**

Código	*Descripción*	*Prioridad*	*Revisión*	*Aceptado*	*Realizado*
6	La programación debe estar hecha en Python.	1	26/09/2019	✓	✓
7	El robot debe poder tomar una lata en un tiempo, como máximo, de cinco segundos.	2	26/09/2019	✓	✓
8	El robot debe poder girar una lata mientras la toma en un tiempo, como máximo, de cinco segundos.	2	26/09/2019	✓	✓
9	El robot debe estar hecho con piezas del kit lego Mindstorm EV3 education.	1	01/10/2019	✓	✓

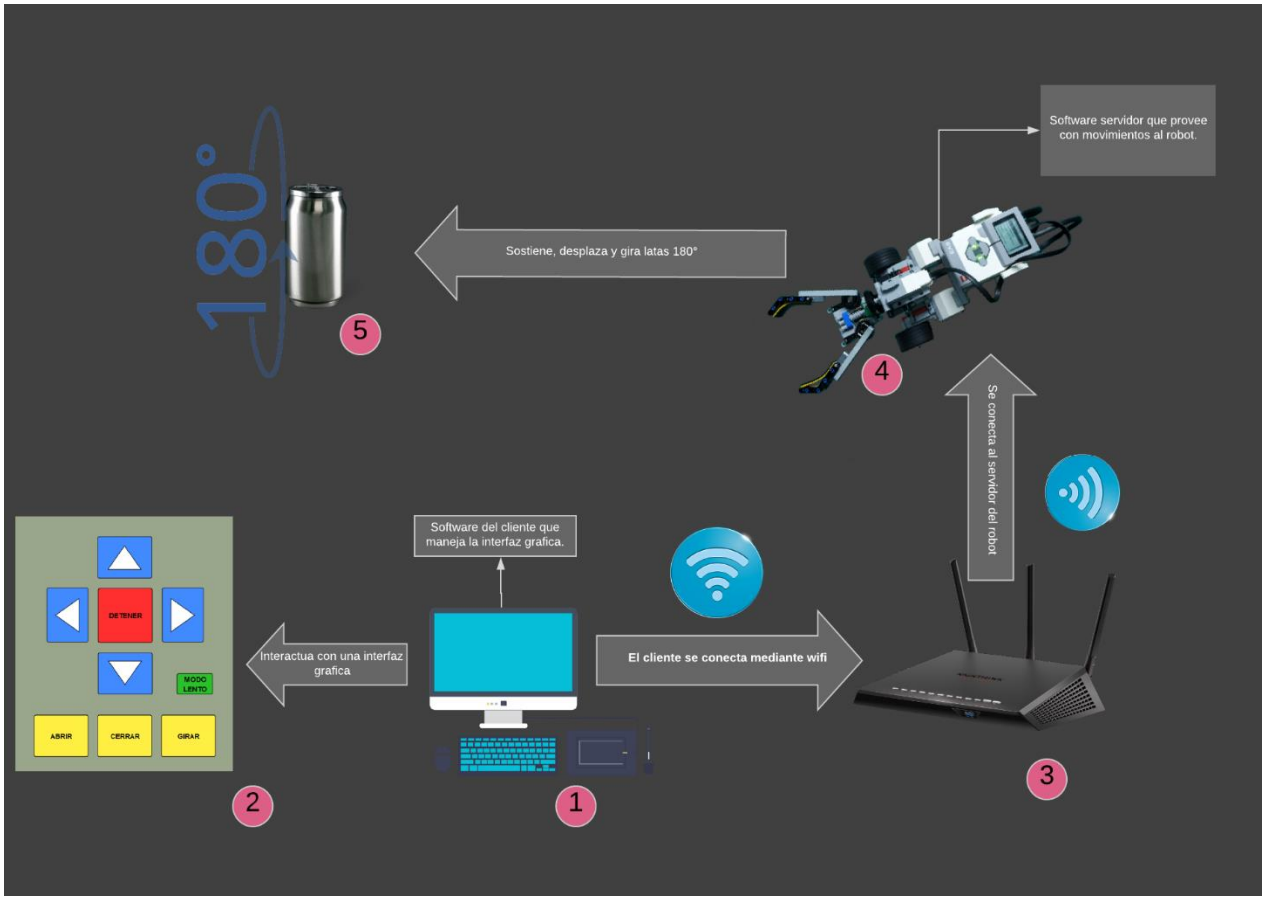
Prioridad:

1: Muy importante

2: Importante

3: No tan importante

B. ARQUITECTURA PROPUESTA



- Punto 1: PC que utiliza un software con el que el cliente puede controlar un robot.
- Punto 2: Interfaz de un software con el que controlar al robot a distancia.
- Punto 3: Router que sirve para la conexión de un servidor entre el PC y el robot mediante la señal WI-FI.
- Punto 4: Robot que realizará los movimientos enviados por el cliente gracias al servidor establecido entre sí mismo y el PC del cliente.
- Punto 5: Objeto con el que el robot deberá interactuar, en este caso, una lata de bebida.

C. DISEÑO DE LA INTERFAZ DE USUARIO

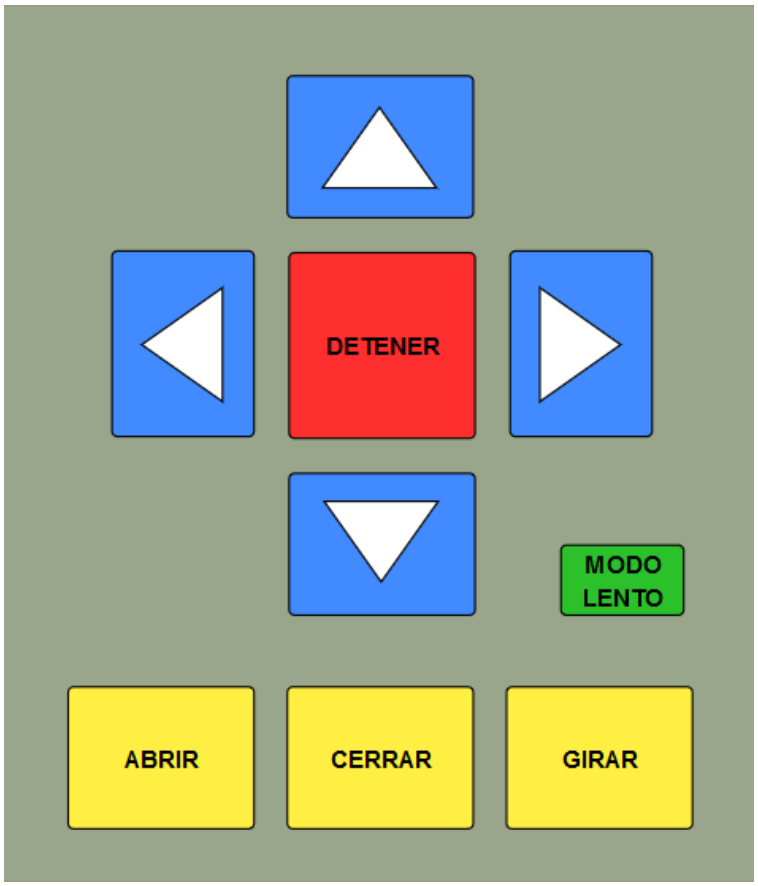


Imagen 1

Esta es la interfaz de movimiento del robot, como se puede ver en la imagen (Imagen 1), en la parte superior hay cuatro botones en forma de cruceta, al ser presionados generarán el movimiento del robot. El botón “Detener” en medio de estos sirve para interrumpir cualquier actividad. Los tres botones inferiores sirven para utilizar la garra, tal como dicen sus nombres uno sirve para abrir la garra y la otra para cerrarla, el ultimo botón sirve para rotar a la misma. Por último, el botón “MODO LENTO” sirve para que los movimientos del robot sean más lentos, de esta manera se puede aumentar la precisión de todos los movimientos.

VI. IMPLEMENTACIÓN

- **Documentación del servidor**

Las siguientes imágenes corresponden a la documentación del programa con los movimientos del robot.

ServidorPruebaDoc

index
/home/robot/ServidorPruebaDoc.py

Modules

rpyc

Classes

rpyc.core.service.Service(builtins.object)
MyService

class **MyService**(rpyc.core.service.Service)

```
    The service base-class. Derive from this class to implement custom RPyC
    services:

    * The name of the class implementing the ``Foo`` service should match the
      pattern ``FooService`` (suffixed by the word 'Service') ::

          class FooService(Service):
              pass

          FooService.get_service_name() # 'FOO'
          FooService.get_service_aliases() # ['FOO']

    * To supply a different name or aliases, use the ``ALIASES`` class attribute ::

          class Foobar(Service):
              ALIASES = ["foo", "bar", "lalaland"]

          Foobar.get_service_name() # 'FOO'
          Foobar.get_service_aliases() # ['FOO', 'BAR', 'LALALAND']

    * Override :func:`on_connect` to perform custom initialization

    * Override :func:`on_disconnect` to perform custom finalization

    * To add exposed methods or attributes, simply define them normally,
      but prefix their name by ``exposed_``, e.g. ::

          class FooService(Service):
              def exposed_add(self, x, y):
                  return x + y

    * All other names (not prefixed by ``exposed_``) are local (not accessible
      to the other party)

    .. note::
       You can override ``_rpyc_getattr``, ``_rpyc_setattr`` and ``_rpyc_delattr``
       to change attribute lookup -- but beware of possible **security implications!**
```

Method resolution order:
MyService
rpyc.core.service.Service
builtins.object

Methods defined here:

exposed_Abrir(self, outga_b)
exposed_Abrir(self,outga_b)

```
Hace abrir la garra del robot.

Parametros
----------

self : MyService
    es una referencia  a la instancia de MyService creada a partir de la clase.

outga_b : string
    es el nombre del motor que se ocupa para cerrar/abrir la garra del robot.
```

exposed_Avanzar(self, outa, outd)
exposed_Avanzar(self,outa,outd)

```
Hace avanzar el robot.

Parametros
----------
self: MyService
    es una refencia a la instancia de MyService creada a partir de la clase.

outa: string
    es el nombre del  motor de la rueda derecha que se va a ocupar.

outd: string
    es el nombre del motor de la rueda izquierda que se va a ocupar.
```

exposed_Cerrar(self, outga_b)
exposed_Cerrar(self,outga_b)

```
Hace cerrar la garra del robot.

Parametros
----------

self : MyService
    es una referencia  a la instancia de MyService creada a partir de la clase.

outga_b : string
    es el nombre del motor que se ocupa para cerrar/abrir la garra del robot.
```

exposed_Derecha(self, outa, outd)
exposed_Derecha(self,outa,outd)

```
Hace girar a la derecha al robot

Paremetros
----------

self : MyService
    es una referencia  a la instancia de MyService creada a partir de la clase.

outa : string
    es el nombre del motor de la rueda derecha del robot que se va a ocupar.

outd : string
    es el nombre del motor de la rueda izquierda del robot que se va a ocupar.
```

```
exposed_Izquierda(self, outa, outd)
    exposed_Izquierda(self,outa,outd)

    Hace girar a la izquierda al robot.

    Parametros
    ----------

    self : MyService
        es una referencia  a la instancia de MyService creada a partir de la clase.

    outa : string
        es el nombre del motor de la rueda derecha del robot que se va a ocupar.

    outd : string
        es el nombre del motor de la rueda izquierda del robot que se va a ocupar.

exposed_Reset(self, outa, outd, outgi_c, outga_b)
    exposed_Reset(self,outa,outd,outgi_c,outga_b)

    Hace retener todos los motores del robot.

    Parametros
    ----------

    self : MyService
        es una referencia  a la instancia de MyService creada a partir de la clase.

    outa : string
        es el nombre del motor de la rueda derecha del robot que se va a ocupar.

    outd : string
        es el nombre del motor de la rueda izquierda del robot que se va a ocupar.

    outgi_c : string
        es el nombre del motor que se ocupa para hacer girar la garra del robot.

    outga_b : string
        es el nombre del motor que se ocupa para cerrar/abrir la garra del robot.
```

```
exposed_Retroceder(self, outa, outd)
    exposed_Retroceder(self,outa,outd)

    Hace retroceder al robot.

    Parametros
    ----------
    self : MyService
        es una referencia  a la instancia de MyService creada a partir de la clase.

    outa : string
        es el nombre del motor de la rueda derecha del robot que se va a ocupar.

    outd : string
        es el nombre del motor de la rueda izquierda del robot que se va a ocupar.

exposed_Subir(self, outgi_c, outga_b)
    exposed_Subir(self,outgi_c,outga_b)

    Hace girar y cerrar la garra del robot.

    Paremtros
    ---------

    self : MyService
        es una referencia  a la instancia de MyService creada a partir de la clase.

    outgi_c : string
        es el nombre del motor que se ocupa para hacer girar la garra del robot.

    outga_b : string
        es el nombre del motor que se ocupa para cerrar/abrir la garra del robot.
```

exposed_elseReset(self, outa, outd)

```
exposed_elseReset(self,outa,outd)

Hace retener los motores de las ruedas del robot.

Parametros
----------

self : MyService
    es una referencia  a la instancia de MyService creada a partir de la clase.

outa : string
    es el nombre del motor de la rueda derecha del robot que se va a ocupar.

outd : string
    es el nombre del motor de la rueda izquierda del robot que se va a ocupar.
```

exposed_slowAbrir(self, outga_b)

```
exposed_slowAbrir(self,outga_b)

Hace abrir lentamente la garra del robot

Parametros
----------

self : MyService
    es una referencia  a la instancia de MyService creada a partir de la clase.

outga_b : string
    es el nombre del motor que se ocupa para cerrar/abrir la garra del robot.
```

exposed_slowAvanzar(self, outa, outd)

```
exposed_slowAvanzar(self,outa,outd)

Hace avanzar el robot lentamente.

Parametros
----------
self: MyService
    es una referencia a la instancia de MyService creada a partir de la clase.

outa:string
    es el nombre del motor de la rueda derecha que se va a ocupar.

outad: string
    es el nombre del motor de la rueda izquierdo que se va a ocupar.
```

exposed_slowCerrar(self, outga_b)

```
exposed_slowCerrar(self,outga_b)

Hace cerrar lentamente la garra del robot.

Parametros
----------

self : MyService
    es una referencia  a la instancia de MyService creada a partir de la clase.

outga_b : string
    es el nombre del motor que se ocupa para cerrar/abrir la garra del robot.
```

exposed_slowDerecha(self, outa, outd)

exposed_slowDerecha(self,outa,outd)

Hace girar lentamente a la derecha robot.

Parametros
---

self : MyService
es una referencia a la instancia de MyService creada a partir de la clase.

outa : string
es el nombre del motor de la rueda derecha del robot que se va a ocupar.

outd : string
es el nombre del motor de la rueda izquierda del robot que se va a ocupar.

exposed_slowIzquierda(self, outa, outd)

exposed_slowIzquierda(self,outa,oud)

Hace girar lentamente a la izquierda al robot.

Parametros
---

self : MyService
es una referencia a la instancia de MyService creada a partir de la clase.

outa : string
es el nombre del motor de la rueda derecha del robot que se va a ocupar.

outd : string
es el nombre del motor de la rueda izquierda del robot que se va a ocupar.

Data descriptors defined here:

__dict__
dictionary for instance variables (if defined)

__weakref__
list of weak references to the object (if defined)

Methods inherited from rpyc.core.service.Service:

__init__(self, conn)
Initialize self. See help(type(self)) for accurate signature.

on_connect(self)
called when the connection is established

on_disconnect(self)
called when the connection had already terminated for cleanup
(must not perform any IO on the connection)

Class methods inherited from rpyc.core.service.Service:

exposed_get_service_aliases = get_service_aliases() from builtins.type
returns a list of the aliases of this service

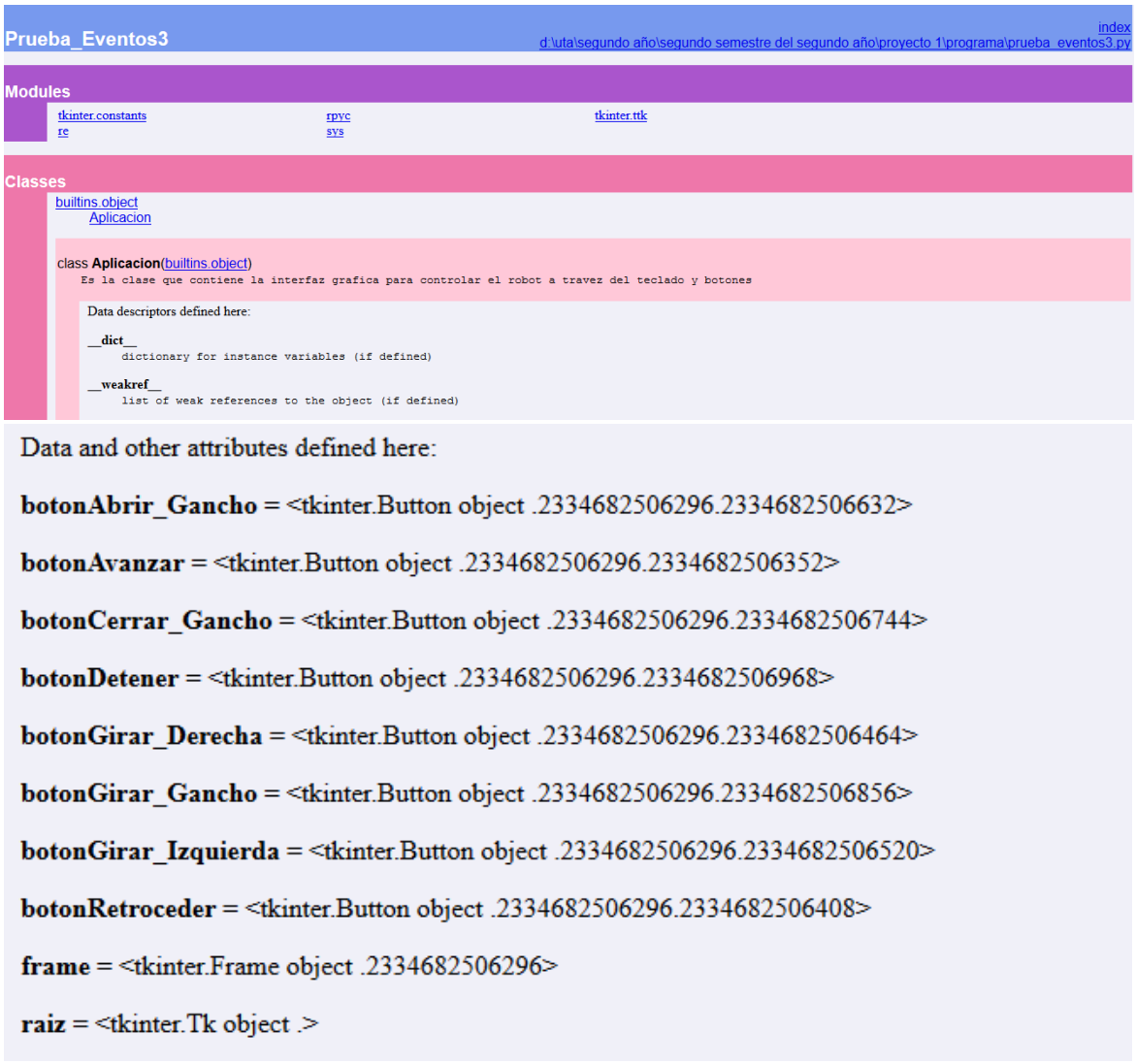
exposed_get_service_name = get_service_name() from builtins.type
returns the canonical name of the service (which is its first
alias)

get_service_aliases() from builtins.type
returns a list of the aliases of this service

get_service_name() from builtins.type
returns the canonical name of the service (which is its first
alias)

- **Documentación del cliente**

Las siguientes imágenes corresponden a la documentación del programa que sirve para ejecutar los movimientos del robot a través de una interfaz.



Prueba_Eventos3

index
d:\uta\segundo año\segundo semestre del segundo año\proyecto 1\programa\prueba_eventos3.py

Modules

tkinter.constants rpyc tkinter.ttk
re sys

Classes

builtins.object
Aplicacion

class **Aplicacion**(builtins.object)
Es la clase que contiene la interfaz grafica para controlar el robot a travez del teclado y botones

Data descriptors defined here:

__dict__
dictionary for instance variables (if defined)

__weakref__
list of weak references to the object (if defined)

Data and other attributes defined here:

botonAbrir_Gancho = <tkinter.Button object .2334682506296.2334682506632>

botonAvanzar = <tkinter.Button object .2334682506296.2334682506352>

botonCerrar_Gancho = <tkinter.Button object .2334682506296.2334682506744>

botonDetener = <tkinter.Button object .2334682506296.2334682506968>

botonGirar_Derecha = <tkinter.Button object .2334682506296.2334682506464>

botonGirar_Gancho = <tkinter.Button object .2334682506296.2334682506856>

botonGirar_Izquierda = <tkinter.Button object .2334682506296.2334682506520>

botonRetroceder = <tkinter.Button object .2334682506296.2334682506408>

frame = <tkinter.Frame object .2334682506296>

raiz = <tkinter.Tk object .>

Functions

Abrir(event)
Abrir(event)

Activa la funcion Abrir del robot a traves de un evento del teclado.

event: evento
recibe un evento del teclado.

Avanzar(event)
Avanzar(event)

Activa la funcion Avanzar del robot a traves de un evento del teclado.

event: evento
recibe un evento del teclado.

Cerrar(event)
Cerrar(event)

Activa la funcion Cerrar del robot a traves de un evento del teclado.

event: evento
recibe un evento del teclado.

Derecha(event)
Derecha(event)

Activa la funcion Derecha del robot a traves de un evento del teclado.

event: evento
recibe un evento del teclado.

Detener(event)
Detenet(event)

Activa la funcion Detener del robot a traves de un evento del teclado.

event: evento
recibe un evento del teclado.

Girar(event)
Girar(event)

Activa la funcion Girar del robot a traves de un evento del teclado.

event: evento
recibe un evento del teclado.

Izquierda(event)
Izquierda(event)

Activa la funcion Izquierda del robot a traves de un evento del teclado.

event: evento
recibe un evento del teclado.

Retroceder(event)
Retroceder(event)

Activa la funcion Retroceder del robot a traves de un evento del teclado.

event: evento
recibe un evento del teclado.

botonAbrir()
BotonAbrir()

Activa la funcion Abrir del robot a traves de un boton que esta dentro de la interfaz grafica.

botonAvanzar()
BotonAvanzar()

Activa la funcion Avanzar del robot a traves de un boton que esta dentro de la interfaz grafica.

botonCerrar()
BotonCerrar()

Activa la funcion Cerrar del robot a traves de un boton que esta dentro de la interfaz grafica.

botonDerecha()
BotonDerecha()

Activa la funcion Derecha del robot a traves de un boton que esta dentro de la interfaz grafica.

botonDetener()
BotonDetener()

Activa la funcion Detener del robot a traves de un boton que esta dentro de la interfaz grafica.

botonGirar()
BotonGirar()

Activa la funcion Girar del robot a traves de un boton que esta dentro de la interfaz grafica.

botonIzquierda()
BotonIzquierda()

Activa la funcion Izquierda del robot a traves de un boton que esta dentro de la interfaz grafica.

botonRetroceder()
BotonRetroceder()

Activa la funcion Retroceder del robot a traves de un boton que esta dentro de la interfaz grafica.

elseDetener(event)
elseDetener(event)

Activa la funcion elseDetener del robot a traves de un evento del teclado.

event: evento
recibe un evento del teclado.

main()
Es la funcion main donde se llama a la clase Aplicacion

slowAbrir(event)
slowAbrir(event)

Activa la funcion slowAbrir del robot a traves de un evento del teclado.

event: evento
recibe un evento del teclado.

slowAvanzar(event)
slowAvanzar(event)

Activa la funcion slowAvanzar del robot a traves de un evento del teclado.

event: evento
recibe un evento del teclado.

slowCerrar(event)
slowCerrar(event)

Activa la funcion slowCerrar del robot a traves de un evento del teclado.

event: evento
recibe un evento del teclado.

slowDerecha(event)
slowDerecha(event)

Activa la funcion slowDerecha del robot a traves de un evento del teclado.

event: evento
recibe un evento del teclado.

slowGirar(event)
slowGirar(event)

Activa la funcion slowGirar del robot a traves de un evento del teclado.

event: evento
recibe un evento del teclado.

slowIzquierda(event)
slowIzquierda(event)

Activa la funcion slowIzquierda del robot a traves de un evento del teclado.

event: evento
recibe un evento del teclado.

slowRetroceder(event)
slowRetroceder(event)

Activa la funcion slowRetroceder del robot a traves de un evento del teclado.

event: evento
recibe un evento del teclado.

Data

ACTIVE = 'active'
ALL = 'all'
ANCHOR = 'anchor'
ARC = 'arc'
BASELINE = 'baseline'
BEVEL = 'bevel'
BOTH = 'both'
BOTTOM = 'bottom'
BROWSE = 'browse'
BUTT = 'butt'
CASCADE = 'cascade'
CENTER = 'center'
CHAR = 'char'
CHECKBUTTON = 'checkbutton'
CHORD = 'chord'
COMMAND = 'command'
CURRENT = 'current'
DISABLED = 'disabled'
DOTBOX = 'dotbox'
E = 'e'
END = 'end'
EW = 'ew'
EXCEPTION = 8
EXTENDED = 'extended'
FALSE = 0
FIRST = 'first'
FLAT = 'flat'
GROOVE = 'groove'
HIDDEN = 'hidden'
HORIZONTAL = 'horizontal'
INSERT = 'insert'
INSIDE = 'inside'
LAST = 'last'

LEFT = 'left'
MITER = 'miter'
MOVETO = 'moveto'
MULTIPLE = 'multiple'
N = 'n'
NE = 'ne'
NO = 0
NONE = 'none'
NORMAL = 'normal'
NS = 'ns'
NSEW = 'nsew'
NUMERIC = 'numeric'
NW = 'nw'
OFF = 0
ON = 1
OUTSIDE = 'outside'
PAGES = 'pages'
PIESLICE = 'pieslice'
PROJECTING = 'projecting'
RADIOBUTTON = 'radiobutton'
RAISED = 'raised'
READABLE = 2
RIDGE = 'ridge'
RIGHT = 'right'
ROUND = 'round'
S = 's'
SCROLL = 'scroll'
SE = 'se'
SEL = 'sel'
SEL_FIRST = 'sel.first'
SEL_LAST = 'sel.last'
SEPARATOR = 'separator'
SINGLE = 'single'
SOLID = 'solid'
SUNKEN = 'sunken'

SW = 'sw'
TOP = 'top'
TRUE = 1
TclVersion = 8.6
TkVersion = 8.6
UNDERLINE = 'underline'
UNITS = 'units'
VERTICAL = 'vertical'
W = 'w'
WORD = 'word'
WRITABLE = 4
X = 'x'
Y = 'y'
YES = 1
wantobjects = 1

VII. RESULTADOS

A. ESTADO ACTUAL DEL PROYECTO

Actualmente el proyecto cuenta con:

- La versión final del robot “R.S.B.A.”.

Esto quiere decir, que el robot cuenta con una garra para tomar y soltar latas que, a su vez, posee un sistema de rotación para voltear dichos objetos.

- Las funciones de movimiento.

Estas ya están implementadas, esto quiere decir que, el robot posee los algoritmos necesarios para moverse y usar su garra (abrir, cerrar y rotar), los cuales son esenciales para jugar Flip-Tac-Toe.

- La conexión de una interfaz al robot.

En otras palabras, se instaló la librería RPyC en ev3dev/Python la cual permite que el robot pueda ser manipulado como si se tratase de un objeto local en el pc.

- La wiki del proyecto.

Esta cuenta con una introducción del proyecto, una presentación de los desarrolladores e imágenes que dan cuenta del avance del proyecto.

- La programación del apartado visual de la interfaz gráfica.

Se tiene listo el movimiento del robot con teclas del computador y existe un apartado visual de la interfaz.

B. PROBLEMAS ENCONTRADOS Y SOLUCIONES PROPUESTAS

PROBLEMAS ENCONTRADOS	SOLUCIONES
La garra del robot estaba muy alta por lo que no le era posible tomar una lata pequeña, sin embargo, si podía tomar una lata de mayor tamaño.	Se cambia el diseño del robot de manera que la garra quede a la altura de la lata que se usará para jugar Flip-Tac-Toe.
En un principio, el sistema de rotación no era capaz de girar la garra pues esta tenía que soportar el peso de la garra y su motor.	Se cambió el diseño del sistema de rotación de manera que la garra gire en torno a su propio eje gracias a engranajes, esto permitió que el sistema no tuviera que soportar los pesos de la garra o el motor de este.
En un principio se intentó desarrollar los algoritmos en Visual Studio, sin embargo, no se pudo pues hubo problemas de compatibilidad con el pc en que se estaba trabajando y luego problemas de cómo trabajar con la plataforma.	Se trabajó en un editor de texto directamente en el robot.
Al momento de rotar la garra, por su diseño, se abría, por lo que soltaba la lata automáticamente una vez terminaba el giro hacia la derecha.	Para arreglar el hecho de que la garra se abriera al momento de rotarla, compensamos lo que se abriese ocupando el sistema de cerrado a la vez

C. CONCLUSIONES

Hasta el momento en que se desarrolla este informe, se puede recalcar que el proyecto ha avanzado bastante bien con respecto a las fechas estipuladas en la carta Gantt, se han tenido algunos problemas que han podido ser resueltos tales como, errores de diseño del robot que han sido solucionados lo más rápido posible, también han existido contratiempos en el desarrollo del código para los movimientos del robot y el desarrollo de la interfaz con la que el usuario podría interactuar con el mismo, sin embargo, el problema se resume en la nula experiencia que poseíamos como equipo para poder trabajar en el lenguaje Python, por lo que no era un gran obstáculo, pues existen variadas fuentes de información para el apoyo en este campo. Con respecto al desarrollo del proyecto, el robot ya está en su diseño final, igualmente, los algoritmos de movimiento de este también están desarrollados por completo y la interfaz gráfica ya se encuentra terminada. En cuanto a la documentación del proyecto, actualmente se cuenta con el informe final del proyecto y está terminada la wiki del proyecto.

VIII. REFERENCIAS

[1] B. S. Marco, «Introducción a la programación con Python,» 15 octubre 2019. [En línea]. Available: http://www.mclibre.org/consultar/python/. [Último acceso: 22 agosto 2019].

[2] R. Hempel, «Motor classes,» 2015. [En línea]. Available: https://ev3dev-lang.readthedocs.io/projects/python-ev3dev/en/stable/motors.html. [Último acceso: 26 septiembre 2019].

[3] ogaworks, «Building Grab and Lift Mindstorms EV3,» 28 noviembre 2016. [En línea]. Available: https://www.youtube.com/watch?v=nTLB9GWRyuo&t=216s. [Último acceso: 22 agosto 2019].

[4] U. d. l. Rioja, «EDITOR VI,» [En línea]. Available: https://www.unirioja.es/cu/enriquez/docencia/Quimica/vi.pdf. [Último acceso: 15 octubre 2019].

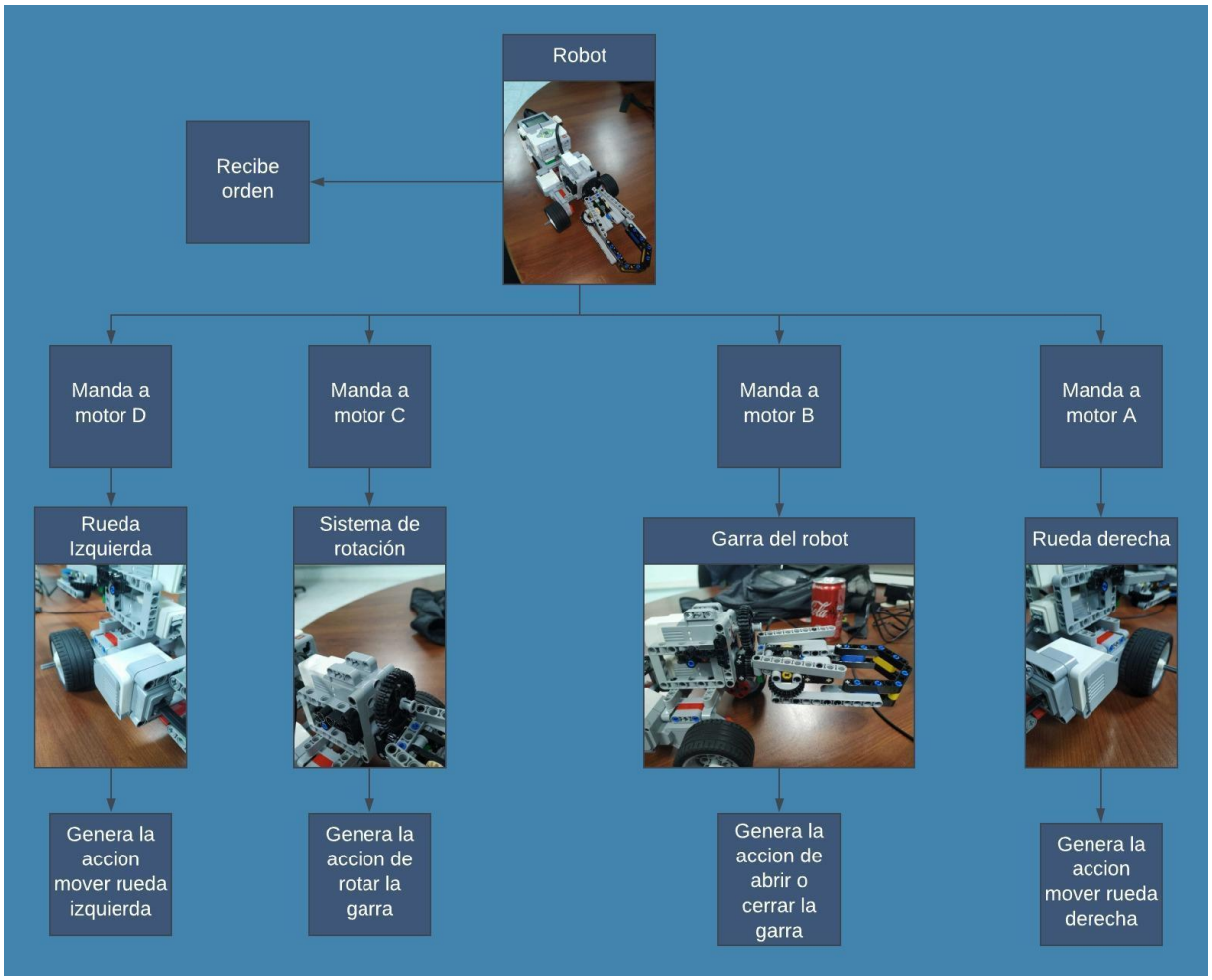
[5] «numpydoc docstring guide¶,» 09 agosto 2019. [En línea]. Available: https://numpydoc.readthedocs.io/en/latest/format.html. [Último acceso: 10 octubre 0219].

[6] D. Fisher, «Python ev3dev - 01 Drive Motors - 02 ev3 Motor API,» 02 octubre 2017. [En línea]. Available: https://www.youtube.com/watch?v=j0-ATIe6pqg. [Último acceso: 26 septiembre 20219].

- **ANEXOS**

ANEXO A: HARDWARE

El siguiente diagrama corresponde al robot construido y a las funciones de cada una de sus partes.



ANEXO B: SOFTWARE

A continuación, se presenta el código necesario de las funciones de movimiento del robot.

```
import rpyc
from rpyc.utils.server import ThreadedServer
from ev3dev2.motor import Motor

class MyService(rpyc.Service):

    """
        Es la clase servidor el cual tiene todas las funciones del robot.

        Parametro
        ---------
        rypc.Service: clase
            MYservice es una subclase de la clase rypc.Service.
    """

       def exposed_Avanzar(self,outa,outd):

        """
        exposed_Avanzar(self,outa,outd)

        Hace avanzar al robot.

        Parametros
        ----------
        self : MyService
            es una referencia a la instancia de MyService creada a partir de
            la clase.

        outa : string
            es nombre del motor de la rueda derecha del robot que se va a
            ocupar.

        outd : string
            es nombre del motor de la rueda izquierdo del robot que se va a
```

```
            ocupar.
        """
        rueda1 = Motor(outa)
        rueda2 = Motor(outd)
        rueda1.run_forever(speed_sp=400)
        rueda2.run_forever(speed_sp=400)
        print("Avanzando")




    def exposed_slowAvanzar(self,outa,outd):

        """
        exposed_slowAvanzar(self,outa,outd)

        Hace avanzar al robot lentamente.

        Parametros
        ----------
        self : MyService
            es una referencia a la instancia de MyService creada a partir de
            la clase.

        outa : string
            es nombre del motor de la rueda derecha del robot que se va a
            ocupar.

        outd : string
            es nombre del motor de la rueda izquierdo del robot que se va a
            ocupar.
        """
        rueda1 = Motor(outa)
        rueda2 = Motor(outd)
        rueda1.run_forever(speed_sp=150)
        rueda2.run_forever(speed_sp=150)
        print("sAvanzar")



    def exposed_Retroceder(self,outa,outd):

        """
        exposed_Retroceder(self,outa,outd)

        Hace retroceder al robot.
```

```
        Parametros
        ----------
        self : Myservice
            es una referencia a la instancia de MyService creada a partir de
            la clase.

        outa : string
            es nombre del motor de la rueda derecha del robot que se va a
            ocupar.




        outd : string
            es nombre del motor de la rueda izquierdo del robot que se va a
            ocupar.
        """
        rueda1 = Motor(outa)
        rueda2 = Motor(outd)
        rueda1.run_forever(speed_sp=-400)
        rueda2.run_forever(speed_sp=-400)
        print("Retrocediendo")



    def exposed_slowRetroceder(self,outa,outd):

        """
        exposed_slowRetroceder(self,outa,outd)

        Hace retroceder al robot lentamente.

        Parametros
        ----------
        self : Myservice
            es una referencia a la instancia de MyService creada a partir de
            la clase.

        outa : string
            es nombre del motor de la rueda derecha del robot que se va a
            ocupar.

        outd : string
            es nombre del motor de la rueda izquierdo del robot que se va a
            ocupar.
```

```
        """
        rueda1 = Motor(outa)
        rueda2 = Motor(outd)
        rueda1.run_forever(speed_sp=-150)
        rueda2.run_forever(speed_sp=-150)
        print("sRetroceder")









    def exposed_Derecha(self,outa,outd):

        """
        exposed_Derecha(self,outa,outd)

        Hace girar a la derecha al robot.

        Parametros
        ----------
        self : Myservice
            es una referencia a la instancia de MyService creada a partir de
            la clase.

        outa : string
            es nombre del motor de la rueda derecha del robot que se va a
            ocupar.

        outd : string
            es el nombre del motor de la rueda izquierdo del robot que se va
            a ocupar.
        """
        rueda1 = Motor(outa)
        rueda2 = Motor(outd)
        rueda1.run_forever(speed_sp=-300)
        rueda2.run_forever(speed_sp=300)
        print("Girando Derecha")

    def exposed_slowDerecha(self,outa,outd):

        """
        exposed_slowDerecha(self,outa,outd)

        Hace girar lentamente a la derecha al robot .
```

```
        Parametros
        ----------
        self : Myservice
            es una referencia a la instancia de MyService creada a partir de
            la clase.

        outa : string
            es nombre del motor de la rueda derecha del robot que se va a
            ocupar.


        outd : string
            es nombre del motor de la rueda izquierdo del robot que se va a
            ocupar.
        """
        rueda1 = Motor(outa)
        rueda2 = Motor(outd)
        rueda1.run_forever(speed_sp=-150)
        rueda2.run_forever(speed_sp=150)
        print("sDerecha")



    def exposed_Izquierda(self,outa,outd):

        """
        exposed_Izquierda(self,outa,outd)

        Hace girar a la izquierda al robot.

        Parametros
        ----------
        self : Myservice
            es una referencia a la instancia de MyService creada a partir de
            la clase.

        outa : string
            es nombre del motor de la rueda derecha del robot que se va a
            ocupar.

        outd : string
            es nombre del motor de la rueda izquierdo del robot que se va a
            ocupar.
        """
```

```
        rueda1 = Motor(outa)
        rueda2 = Motor(outd)
        rueda1.run_forever(speed_sp=300)
        rueda2.run_forever(speed_sp=-300)
        print("Girando Izquierda")


    def exposed_slowIzquierda(self,outa,outd):

        """
        exposed_slowIzquierda(self,outa,outd)

        Hace girar lentamente a la izquierda al robot .

        Parametros
        ----------
        self : Myservice
            es una referencia a la instancia de MyService creada a partir de
            la clase.

        outa : string
            es nombre del motor de la rueda derecha del robot que se va a
            ocupar.

        outd : string
            es nombre del motor de la rueda izquierdo del robot que se va a
            ocupar.
        """
        rueda1 = Motor(outa)
        rueda2 = Motor(outd)
        rueda1.run_forever(speed_sp=150)
        rueda2.run_forever(speed_sp=-150)
        print("sIzquierda")


    def exposed_Cerrar(self,outga_b):

        """
        exposed_Cerrar(self,outga_b)

        Hace cerrar la garra del robot.
```

```
        Parametros
        ----------
        self : Myservice
            es una referencia a la instancia de MyService creada a partir de
            la clase.



        outga_b : string
            es el nombre del motor que se ocupa para cerrar/abrir la garra del
            robot.
        """
        garra_gancho = Motor(outga_b)
        garra_gancho.run_forever(speed_sp=-300)#ABRIR
        print("Cerrar Gancho")

    def exposed_slowCerrar(self,outga_b):

        """
        exposed_slowCerrar(self,outga_b)

        Hace cerrar lentamente la garra del robot.

        Parametros
        ----------
        self : Myservice
            es una referencia a la instancia de MyService creada a partir de
            la clase.

        outga_b : string
            es nombre del motor que se ocupa para cerrar/abrir  la garra del
            robot.

        """
        garra_gancho = Motor(outga_b)
        garra_gancho.run_forever(speed_sp=-200)
        print("sCerrar")

    def exposed_Abrir(self,outga_b):

        """
        exposed_Abrir(self,outga_b)

        Hace abrir la garra del robot.
```

```
        Parametros
        ----------
        self : Myservice
            es una referencia a la instancia de MyService creada a partir de
            la clase.

        outga_b : string
            es nombre del motor que se ocupa para cerrar/abrir la garra del
            robot.

        """
        garra_gancho = Motor(outga_b)
        garra_gancho.run_forever(speed_sp=300) #CERRAR
        print("Abrir Gancho")

    def exposed_slowAbrir(self,outga_b):

        """
        exposed_slowAbrir(self,outga_b)

        Hace abrir lentamente la garra del robot .

        Parametros
        ----------
        self : Myservice
            es una referencia a la instancia de MyService creada a partir de
            la clase.

        outga_b : string
            es nombre del motor que se ocupa para cerrar/abrir la garra del
            robot.

        """
        garra_gancho = Motor(outga_b)
        garra_gancho.run_forever(speed_sp=200)
        print("sAbrir")



    def exposed_Subir(self,outgi_c,outga_b):

        """
        exposed_Subir(self,outgi_c,outga_b)

        Hace girar y cerrar la garra del robot.
```

```
        Parametros
        ----------
        self : Myservice
            es una referencia a la instancia de MyService creada a partir de
            la clase.



       outgi_c : string
            es nombre del motor que se ocupar para hacer girar la garra del
            robot.

        outga_b : string
            es nombre del motor que se ocupa para cerrar/abrir la garra del
             robot.

        """
        garra_girar = Motor(outgi_c)
        garra_gancho = Motor(outga_b)
        garra_girar.run_forever(speed_sp=120) #SUBE
        garra_gancho.run_forever(speed_sp=-90)
        print("Girar Gancho")


    def exposed_slowSubir(self,outgi_c,outga_b):

        """
        exposed_slowSubir(self,outgi_c,outga_b)

        Hace girar y cerrar la garra lentamente del robot.

        Parametros
        ----------
        self : Myservice
            es una referencia a la instancia de MyService creada a partir de
            la clase.

        outgi_c : string
            es nombre del motor que se ocupar para hacer girar la garra del
            robot.

        outga_b : string
            es nombre del motor que se ocupa para cerrar/abrir la garra del
            robot.

        """
```

```
        garra_girar = Motor(outgi_c)
        garra_gancho = Motor(outga_b)
        garra_girar.run_forever(speed_sp=80)
        garra_gancho.run_forever(speed_sp=-90)
        print("sSubir")


    def exposed_Reset(self,outa,outd,outgi_c,outga_b):

        """
        exposed_Reset(self,outa,outd,outgi_c,outga_b)

        Hace retener todos los motores del robot.

        Parametros
        ----------
        self : Myservice
            es una referencia a la instancia de MyService creada a partir de
            la clase.

        outa : string
            es nombre del motor de la rueda derecha del robot que se va a
            ocupar.

        outd : string
            es nombre del motor de la rueda izquierdo del robot que se va a
            ocupar.

        outgi_c : string
            es nombre del motor que se ocupar para hacer girar la garra del
            robot.

        outga_b : string
            es nombre del motor que se ocupa para cerrar/abrir la garra del
            robot.

        """
        rueda1 = Motor(outa)
        rueda2 = Motor(outd)
        garro_girar = Motor(outgi_c)
        garro_gancho = Motor(outga_b)
        rueda1.stop()
        rueda2.stop()
        garro_girar.stop()
        garro_gancho.stop()
        print("Detener")
```

```
    def exposed_elseReset(self,outa,outd):

        """
        exposed_elseReset(self,outa,outd)

        Hace retener los motores de las ruedas del robot.

        Parametros
        ----------
        self : Myservice
            es una referencia a la instancia de MyService creada a partir de
            la clase.

        outa : string
            es nombre del motor de la rueda derecha del robot que se va a
            ocupar.

        outd : string
            es nombre del motor de la rueda izquierdo del robot que se va a
            ocupar.
        """
        rueda1 = Motor(outa)
        rueda2 = Motor(outd)
        rueda1.stop()
        rueda2.stop()
        print("sDetener")

if __name__ == '__main__':
    print("Iniciado")
    s = ThreadedServer(MyService, port=12001)
    s.start()
```

A continuación se presenta el código necesario para que se ejecuten los movimientos del robot a través de una interfaz.

```
import rpyc
from tkinter import *
from tkinter import ttk
from PIL import Image,ImageTk


def Avanzar(event):
    """
    Avanzar(event)

    Activa la funcion Avanzar del robot a traves de un evento del teclado.

    event: evento
        recibe un evento del teclado.
    """
    conn = rpyc.connect('192.168.71.199', port=12001)
    conn.root.Avanzar('outA','outD')

def slowAvanzar(event):
    """
    slowAvanzar(event)

    Activa la funcion slowAvanzar del robot a traves de un evento del teclado.

    event: evento
        recibe un evento del teclado.
    """
    conn = rpyc.connect('192.168.71.199', port=12001)
    conn.root.slowAvanzar('outA','outD')

def Retroceder(event):
    """
    Retroceder(event)

    Activa la funcion Retroceder del robot a traves de un evento del teclado.

    event: evento
        recibe un evento del teclado.
    """
    conn = rpyc.connect('192.168.71.199', port=12001)
    conn.root.Retroceder('outA','outD')
```

```
def slowRetroceder(event):
    """
    slowRetroceder(event)

    Activa la funcion slowRetroceder del robot a traves de un evento del
    teclado.

    event: evento
        recibe un evento del teclado.
    """
    conn = rpyc.connect('192.168.71.199', port=12001)
    conn.root.slowRetroceder('outA','outD')



def Derecha(event):
    """
    Derecha(event)

    Activa la funcion Derecha del robot a traves de un evento del teclado.

    event: evento
        recibe un evento del teclado.
    """
    conn = rpyc.connect('192.168.71.199', port=12001)
    conn.root.Derecha('outA','outD')



def slowDerecha(event):
    """
    slowDerecha(event)

    Activa la funcion slowDerecha  del robot a traves de un evento del teclado

    event: evento
        recibe un evento del teclado.
    """
    conn = rpyc.connect('192.168.71.199', port=12001)
    conn.root.slowDerecha('outA','outD')
```

```
def Izquierda(event):
    """
    Izquierda(event)

    Activa la funcion Izquierda del robot a traves de un evento del teclado.

    event: evento
        recibe un evento del teclado.
    """
    conn = rpyc.connect('192.168.71.199', port=12001)
    conn.root.Izquierda('outA','outD')


def slowIzquierda(event):
    """
    slowIzquierda(event)

    Activa la funcion slowIzquierda del robot a traves de un evento del
    teclado.

    event: evento
        recibe un evento del teclado.
    """
    conn = rpyc.connect('192.168.71.199', port=12001)
    conn.root.slowIzquierda('outA','outD')


def Abrir(event):
    """
    Abrir(event)

    Activa la funcion Abrir del robot a traves de un evento del teclado.

    event: evento
        recibe un evento del teclado.
    """
    conn = rpyc.connect('192.168.71.199', port=12001)
    conn.root.Abrir('outB')
```

```
def slowAbrir(event):
    """
    slowAbrir(event)

    Activa la funcion slowAbrir del robot a traves de un evento del teclado.

    event: evento
        recibe un evento del teclado.
    """
    conn = rpyc.connect('192.168.71.199', port=12001)
    conn.root.slowAbrir('outB')


def Cerrar(event):
    """
    Cerrar(event)

    Activa la funcion Cerrar del robot a traves de un evento del teclado.

    event: evento
        recibe un evento del teclado.
    """
    conn = rpyc.connect('192.168.71.199', port=12001)
    conn.root.Cerrar('outB')



def slowCerrar(event):
    """
    slowCerrar(event)

    Activa la funcion slowCerrar del robot a traves de un evento del teclado.

    event: evento
        recibe un evento del teclado.
    """
    conn = rpyc.connect('192.168.71.199', port=12001)
    conn.root.slowCerrar('outB')
```

```
def Girar(event):
    """
    Girar(event)

    Activa la funcion Girar del robot a traves de un evento del teclado.

    event: evento
        recibe un evento del teclado.
    """
    conn = rpyc.connect('192.168.71.199', port=12001)
    conn.root.Subir('outC','outB')


def slowGirar(event):
    """
    slowGirar(event)

    Activa la funcion slowGirar del robot a traves de un evento del teclado.

    event: evento
        recibe un evento del teclado.
    """
    conn = rpyc.connect('192.168.71.199', port=12001)
    conn.root.slowSubir('outC','outB')


def Detener(event):
    """
    Detenet(event)

    Activa la funcion Detener del robot a traves de un evento del teclado.

    event: evento
        recibe un evento del teclado.
    """
    conn = rpyc.connect('192.168.71.199', port=12001)
    conn.root.Reset('outA','outD','outC','outB')
```

```
def elseDetener(event):
    """
    elseDetener(event)

    Activa la funcion elseDetener del robot a traves de un evento del teclado.

    event: evento
        recibe un evento del teclado.
    """
    conn = rpyc.connect('192.168.71.199', port=12001)
    conn.root.elseReset('outA','outD')


def botonAvanzar():
    """
    BotonAvanzar()

    Activa la funcion Avanzar del robot a traves de un boton que esta dentro d
e la interfaz grafica.

    """
    conn = rpyc.connect('192.168.71.199', port=12001)
    conn.root.Avanzar('outA','outD')


def botonRetroceder():
    """
    BotonRetroceder()

    Activa la funcion Retroceder del robot a traves de un boton que esta dentr
o de la interfaz grafica.

    """
    conn = rpyc.connect('192.168.71.199', port=12001)
    conn.root.Retroceder('outA','outD')
```

```
def botonDerecha():
    """
    BotonDerecha()

    Activa la funcion Derecha del robot a traves de un boton que esta dentro d
e la interfaz grafica.

    """
    conn = rpyc.connect('192.168.71.199', port=12001)
    conn.root.Derecha('outA','outD')


def botonIzquierda():
    """
    BotonIzquierda()

    Activa la funcion Izquierda del robot a traves de un boton que esta dentro
 de la interfaz grafica.

    """
    conn = rpyc.connect('192.168.71.199', port=12001)
    conn.root.Izquierda('outA','outD')


def botonAbrir():
    """
    BotonAbrir()

    Activa la funcion Abrir del robot a traves de un boton que esta dentro de
la interfaz grafica.

    """
    conn = rpyc.connect('192.168.71.199', port=12001)
    conn.root.Abrir('outB')

def botonCerrar():
    """
    BotonCerrar()

    Activa la funcion Cerrar del robot a traves de un boton que esta dentro de
 la interfaz grafica.
```

```
    """
    conn = rpyc.connect('192.168.71.199', port=12001)
    conn.root.Cerrar('outB')



def botonGirar():
    """
    BotonGirar()

    Activa la funcion Girar del robot a traves de un boton que esta dentro de
la interfaz grafica.

    """
    conn = rpyc.connect('192.168.71.199', port=12001)
    conn.root.Subir('outC','outB')


def botonDetener():
    """
    BotonDetener()

    Activa la funcion Detener del robot a traves de un boton que esta dentro d
e la interfaz grafica.

    """
    conn = rpyc.connect('192.168.71.199', port=12001)
    conn.root.Reset('outA','outD','outC','outB')


class Aplicacion():
    """
    Es la clase que contiene la interfaz grafica para controlar el robot a tra
vez del teclado y botones
    """
    raiz=Tk()

    raiz.title("Ejemplo")

    raiz.config(bg="light blue")

    raiz.bind('<Up>',Avanzar)
    raiz.bind('<Shift-Up>',slowAvanzar)

    raiz.bind('<Down>',Retroceder)
    raiz.bind('<Shift-Down>',slowRetroceder)
```

```
    raiz.bind('<Right>',Derecha)
    raiz.bind('<Shift-Right>',slowDerecha)


    raiz.bind('<Left>',Izquierda)
    raiz.bind('<Shift-Left>',slowIzquierda)

    raiz.bind('<q>',Abrir)
    raiz.bind('<Shift-Q>',slowAbrir)

    raiz.bind('<w>',Cerrar)
    raiz.bind('<Shift-W>',slowCerrar)

    raiz.bind('<e>',Girar)
    raiz.bind('<Shift-E>',slowGirar)

    raiz.bind('<r>',Detener)
    raiz.bind('<Shift-R>',elseDetener)

    frame = Frame(raiz, width=600,height=600,bg="light blue")

    imgbotonAvanzar= Image.open('botonAvanzar.png')
    imgbotonRetroceder= Image.open('botonRetroceder.png')
    imgbotonDerecha= Image.open('botonDerecha.png')
    imgbotonIzquierda= Image.open('botonIzquierda.png')
    imgbotonAbrir= Image.open('botonAbrir.png')
    imgbotonCerrar= Image.open('botonCerrar.png')
    imgbotonGirar= Image.open('botonGirar.png')
    imgbotonLento= Image.open('botonLento.png')
    imgbotonDetener= Image.open('botonDetener.png')

    imgbotonAvanzar= ImageTk.PhotoImage(imgbotonAvanzar)
    imgbotonRetroceder= ImageTk.PhotoImage(imgbotonRetroceder)
    imgbotonDerecha= ImageTk.PhotoImage(imgbotonDerecha)
    imgbotonIzquierda= ImageTk.PhotoImage(imgbotonIzquierda)
    imgbotonAbrir= ImageTk.PhotoImage(imgbotonAbrir)
    imgbotonDetener= ImageTk.PhotoImage(imgbotonDetener)
    imgbotonCerrar= ImageTk.PhotoImage(imgbotonCerrar)
    imgbotonGirar= ImageTk.PhotoImage(imgbotonGirar)
    imgbotonLento= ImageTk.PhotoImage(imgbotonLento)
    raiz.resizable(0,0)
```

```
botonAvanzar = Button(frame,text = "Avanzar" ,padx=5,pady=5,
command=botonAvanzar)

botonRetroceder = Button(frame,text = "Retroceder",padx=5,pady=5,
command=botonRetroceder)

botonGirar_Derecha = Button(frame,text = "Girar Derecha",padx=5,pady=5,
command=botonDerecha)

botonGirar_Izquierda = Button(frame,text = "Girar Izquierda",padx=5,pady=5
,command=botonIzquierda)

botonAbrir_Gancho = Button(frame,text = "Abrir Gancho",padx=5,pady=5
,command=botonAbrir)

botonCerrar_Gancho = Button(frame,text = "Cerrar Gancho",padx=5,pady=5
,command=botonCerrar)

botonGirar_Gancho = Button(frame,text = "Girar Gancho",padx=5,pady=5
,command=botonGirar)

botonDetener = Button(frame,text = "Detener",padx=5,pady=5
,command=botonDetener)

botonSlow = Button(frame,text = "Lento",padx=5,pady=5,image=imgbotonLento)



botonAvanzar.grid(row=0,column=1)
botonRetroceder.grid(row=2,column=1)
botonGirar_Derecha.grid(row=1,column=2)
botonGirar_Izquierda.grid(row=1,column=0)
botonAbrir_Gancho.grid(row=4,column=0)
botonCerrar_Gancho.grid(row=4,column=1)
botonGirar_Gancho.grid(row=4,column=2)
botonDetener.grid(row=1,column=1)
botonSlow.grid(row=3,column=2)
```

```
    frame.pack()

    raiz.mainloop()



def main():
    """
    Es la funcion main donde se llama a la clase Aplicacion
    """
    mi_app = Aplicacion()
    return 0

if __name__ == "__main__":
    main()
    pass
```

ANEXO C: COMUNICACIONES

Se utilizó una biblioteca de Python llamada RPyC para la comunicación remota entre el servidor (Robot) y el cliente (Computador) a través de Wifi mediante un dongle USB conectado al robot, utilizando el internet proveniente de un router y una IP del mismo. En la siguiente imagen se muestra un ejemplo de la importación de la biblioteca y su uso.

```
import rpyc
from tkinter import *
from tkinter import ttk
from PIL import Image,ImageTk


def Avanzar(event):
    """
    Avanzar(event)

    Activa la funcion Avanzar del robot a traves de un evento del teclado.

    event: evento
        recibe un evento del teclado.
    """
    conn = rpyc.connect('192.168.71.199', port=12001)
    conn.root.Avanzar('outA','outD')
```