**UNIVERSIDAD DE TARAPACÁ**



**FACULTAD DE INGENIERÍA**

Departamento de Ingeniería en Computación e Informática



**Formulación del Proyecto  
Arturi-toe**

**Autor(es): Javier Mamani**

**Mauricio Mamani**

**Julio Rivera**

**Rodrigo Carvajal**

**Sebastian Lukich**

**Asignatura: Proyecto l**

**Profesor(es): Ricardo Valdivia**

ARICA, DÍA MES AÑO

# Historial de Cambios

|  |  |  |  |
| --- | --- | --- | --- |
| **Fecha** | **Versión** | **Descripción** | **Autor(es)** |
| 13/08/2019 | 1.0 | Enfoque de ideas para crear un robot capaz de cumplir con cada función requerida. | Julio Rivera  Javier Mamani |
| 20/08/2019 | 1.1 | Versión del robot con garra que abre y cierra y motor mediano que la hace girar. Con problemas de equilibrio. | Rodrigo Carvajal |
| 27/08/2019 | 1.2 | Versión del robot con motor grande que hace girar la garra, que abre y cierra. Problemas de equilibrio casi completamente eliminados. | Rodrigo Carvajal |
| 03/09/2019 | 1.3 | Nuevo armado del robot con un brazo y una base más estables. Estructura de la garra arreglada. Además de un entorno a programar más adecuado y amigable. | Rodrigo Carvajal  Julio Rivera |
| 12/09/2019 | 1.4 | El robot tiene interfaz gráfica para sus controles. El manejo de cables y uso de la pantalla del brick se han facilitado | Mauricio Mamani  Julio Rivera |
| 08/10/2019 | 1.5 | Se cambió en el código la dirección de los giros. Se cambió un cable del robot por otro más corto en otra posición | Rodrigo Carvajal |

# Tabla de Contenidos

1. Panorama General
   1. Introducción (contexto)
   2. Objetivo General
   3. Objetivos Específicos
   4. Restricciones
   5. Entregables
2. Organización del Personal

2.1. Descripción de Roles

2.2. Personal que cumplirá los Roles

2.3. Mecanismos de Comunicación

1. Planificación del Proyecto

3.1. Actividades

3.2. Asignación de tiempo (carta Gantt)

3.3. Gestión de Riesgos

1. Planificación de los Recursos

4.1. Recursos Hardware-Software requeridos

4.2. Estimación de Costos (Hardware, Software, Recursos Humanos)

1. Análisis – Diseño

5.1 Especificación de Requerimientos (Funcionales y no Funcionales)

5.2 Arquitectura Propuesta

5.3 Diseño de la Interfaz Usuario

1. Implementación

Descripción de los programas implementados

1. Pruebas

7.1 Descripción de las pruebas realizadas

7.2 Resultados de las pruebas

1. Resultados

8.1 Estado final del proyecto

8.2 Problemas encontrados y soluciones propuestas

8.3 Conclusiones

8.4 Trabajo Futuro

1. Referencias (utilizando el estándar IEEE)

Anexos

Anexo A: Hardware (diagrama de construcción del robot, componentes principales)

Anexo B: Software (código de los programas implementados)

Anexo C: Comunicaciones (configuración de comunicación pc-robot considerando RPyC)

Introducción

LEGO MindStorm es el set de piezas desarrollado por LEGO con fines educativos [1]. Incluye una variedad de piezas y una computadora en forma de ladrillo. El objetivo de tener estos elementos es combinarlos de manera que se tenga un robot funcional a control remoto, facilitando por ejemplo sus ruedas, sensores de proximidad, entre otros artefactos.

Asimismo, el juego FLIP-TAC-TOE es la variación de TIC-TAC-TOE o tres en raya como es más conocido, en donde el jugador tiene la opción de intercambiar el valor de la pieza del oponente. Siendo la “pieza” el elemento físico que representa la X o la O del juego original. Se juega en un espacio llano como el suelo de una habitación pues los participantes deben llevar su pieza desde un suministro de piezas hasta el tablero, lo que implica moverse por la habitación.

Con esto en claro, el presente informe llevará a cabo el registro del desarrollo del robot ARTURI-TOE, un robot construido a base del set LEGO MindStrom capaz de participar en una partida de FLIP-TAC-TOE. A través de una conexión remota [3] el robot recibirá las órdenes indicadas por un código Python [4] con tal de controlar sus movimientos de avanzar, retroceder, girar sobre su eje propio, girar su garra, abrir y cerrar su garra. A su vez, esto necesitará de instalar un sistema operativo compatible con Python en la computadora del robot. Y si bien esta mayor parte del proyecto la cubre un software, se utilizará también hardware tanto para la conexión remota (computadora portátil, USB Dongle, router) como para instalar un sistema operativo compatible (tarjeta MicroSD).

Objetivo General

Construir un robot Lego que sea capaz de responder a las órdenes que le permitan cumplir las reglas del juego FLIP-TAC-TOE mientras es controlado a distancia.

Objetivo Específico

1. Diseñar un modelo de robot que incluya en sus elementos una garra que sea capaz de dar una vuelta en sentido horario y sentido antihorario, de abrir y cerrar agarrando objetos; además de ruedas que sean capaces de dar movilidad al robot hacia adelante y hacia atrás.
2. Implementar un código en Python de controles de movimiento, es decir, código de la movilidad del robot, código del giro de la garra y código de abrir y cerrar la garra.
3. Habilitar la conexión remota por wifi usando USB Dongle para realizar una interfaz de control del robot en la computadora portátil.
4. Realizar las pruebas y ejecución del robot habilitando un espacio para mover al robot, usando latas pequeñas como piezas del juego.
5. Realizar el material de apoyo del proyecto, en específico, informe, manual de uso, video de guía, wiki del proyecto.

Restricciones

* El código de control del robot se desarrollará usando sólo lenguaje Python.
* El diseño del modelo del robot sólo usará combinaciones de las piezas de Lego Mindstorm Ev3 más aquellas mandadas a hacer por la impresora 3D.
* El equipo será conformado por un máximo de 5 integrantes.
* El proyecto se culminará en un tiempo no mayor a un semestre académico.

Entregables

* Se tendrá un robot de Legos funcional armado por completo.
* Se tendrá el software que muestre la interfaz de control del robot.
* Se tendrán informes de avance precediendo el informe final del proyecto.
* Se habilitará una wiki del proyecto.

Descripción de roles

|  |  |  |
| --- | --- | --- |
| Roles | Descripción | Encargados |
| Jefe de Grupo | Este personal está encargado de asignar actividades a sí mismo y a los integrantes, como vea conveniente para cumplir con los tiempos del proyecto. | * Sebastian Lukich |
| Programador | Este personal estará encargado de realizar el código para poder lograr el movimiento del robot y también el desarrollo de una interfaz para poder controlar el robot. | * Julio Rivera * Mauricio Mamani * Javier Mamani |
| Secretario/a | Este personal estará encargado de realizar informes que engloban el desarrollo progresivo del robot, mostrando cada avance del Software. | * Sebastian Lukich * Mauricio Mamani |
| Armador/a | Este personal estará encargado del armado del robot con miras a lograr cumplir con las condiciones del juego FLIP-TAC-TOE. | * Sebastian Lukich * Rodrigo Carvajal * Julio Rivera |

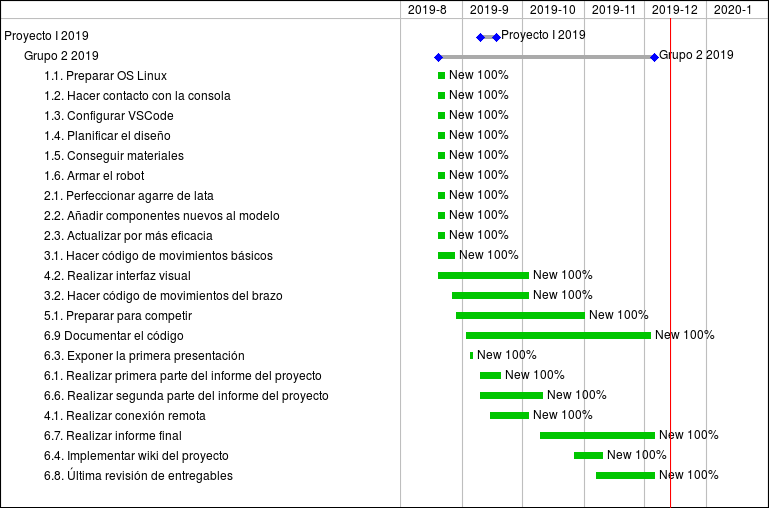
|  |  |
| --- | --- |
| Mecanismos de comunicación | * WhatsApp * Clases * Reuniones semanales |

Actividades

|  |  |  |  |
| --- | --- | --- | --- |
| **Actividad** | **Descripción** | **Responsable** | **Producto** |
| Preparar OS Linux | Bootear una tarjeta SD usando el software Balena Erchen | Sebastian Lukich | Tarjeta SD con Linux funcionando en el Brick del robot LEGO |
| Hacer contacto con la consola | Conectar con el computador mediante el programa PuTTY, una red local y una dirección IPv4 | Mauricio Mamani | Consola de comandos de PuTTY donde se escribe código Python |
| Configurar VSCode | Configurar el path, instalar la extensiones Python, Ev3Controler a VSCode | Mauricio Mamani | Consola más editor de código donde se escribe código Python y se conecta al robot |
| Planificar el diseño | Estudiar diferentes modelos de robots en foros, blogs, YouTube, etc. | Julio Rivera | Diseño del brazo, cuerpo del robot |
| Conseguir materiales | Conseguir las piezas LEGO necesarias para el modelo | Julio Rivera | Set de piezas LEGO Ev3 completo |
| Armar el robot | Armar el robot según el diseño acordado | Rodrigo Carvajal | Robot con motores para las ruedas, para el brazo. Con ruedas y brazo funcionales |
| Perfeccionar agarre de lata | Estudiar al propio modelo del brazo y compararlo con otros mecanismos de agarre/ rotación encontrados en Internet | Rodrigo Carvajal, Julio Rivera | Diseño actualizado más eficiente del brazo |
| Añadir componentes nuevos al modelo | Conseguir las piezas LEGO para el brazo robótico sólo si es necesario | Rodrigo Carvajal | Piezas con forma de garra, engranajes, motor del brazo |
| Actualizar por más eficiencia | Actualizar el brazo del robot por el modelo eficaz | Rodrigo Carvajal | Brazo eficaz que no dificulta el movimiento del robot |
| Hacer código de movimientos básicos | Diseñar mediante Python el código de movimientos de avance, retroceso, dirección del robot en conjunto | Julio Rivera | Código que controle los movimientos básicos del robot funcional |
| Hacer código de movimientos del brazo | Diseñar mediante Python movimientos de agarre, giro del brazo | Julio Rivera | Código que controle los movimientos del brazo del robot funcional |
| Realizar conexión remota | Utilizando RPyC habilitar una conexión remota entre la computadora y el robot | Mauricio Mamani | Conexión a distancia entre Cliente (robot) y Servidor (computadora) |
| Realizar interfaz visual | Usando TKinter y código Python realizar una interfaz visual que se abra en el computador | Mauricio Mamani | Interfaz visual que se abra en el computador y permite controlar al robot mediante teclas clave del teclado |
| Preparar para competir | Probar al robot con los otros elementos del juego: latas, espacio de juego, otro competidor (si es posible); haciendo las últimas modificaciones al código | Sebastian Lukich | Robot listo para ser parte de la partida de FLIP-TAC-TOE |
| Realizar primera parte del informe del proyecto | Poner en un archivo los objetivos generales y específicos más especificaciones del proyecto; buscar informes anteriores al proyecto que sirvan como plantilla | Sebastian Lukich | Avance en digital del Informe del proyecto listo para someterse a correcciones |
| Exponer la primera presentación | Desarrollar en diapositivas un resumen del avance del proyecto hasta el momento y exponerlo para someter el proyecto a evaluación | Sebastian Lukich | Diapositivas en digital con información relevante del proyecto |

|  |  |  |  |
| --- | --- | --- | --- |
| Implementar wiki del proyecto | Escribir una wiki que hable sobre el robot, su uso y su funcionamiento | Sebastian Lukich | Wiki en la web que recopila todo el contenido que hay sobre el proyecto de manera pública |
| Hacer el video | Hacer tomas en video del uso/ funcionamiento del robot, pegarlas en un solo archivo de video, editarlo con fines estéticos y renderizarlo | Rodrigo Carvajal | Video que recopila el contenido del proyecto |
| Realizar segunda parte del informe del proyecto | Poner en un archivo los requerimientos funcionales y no funcionales, la arquitectura propuesta, implementación del diseño y resultados | Sebastian Lukich | Avance en digital del Informe del proyecto listo para someterse a correcciones |
| Realizar informe final | Juntar los diferentes avances del informe en un mismo archivo | Sebastian Lukich | Informe final del proyecto |
| Última revisión de entregables | Frente a los entregables ver si falta alguna cosa | Mauricio Mamani | Lista de entregables completa |

Carta Gantt



Gestión de riesgos

|  |  |  |  |
| --- | --- | --- | --- |
| **Riesgos** | **Probabilidad de ocurrencia** | **Nivel de impacto** | **Acción remedial** |
| Error en el código. | 90% | 3 | Analizar si existen errores de sintaxis o si ya es un error más general, y arreglarlo. |
| Mala estimación del tiempo a desarrollar el proyecto | 40% | 1 | Enfocarse en labores más importantes del proyecto. |
| Diseño del robot puede fallar | 60% | 2 | Buscar modelos que cumplan con las especificaciones necesarias. |
| Robot puede ser desmantelado | 10% | 1 | Tener fotos del robot tomadas con anterioridad con el fin de rearmarlo más fácilmente en el menor tiempo posible. |
| Fallo en tarjeta SD | 10% | 2 | Tener una tarjeta SD de respaldo, y hacer respaldo de la información que ella contiene. |
| Conflicto en equipo | 10% | 3 | Tomar en consideración cada petición de los integrantes, y evaluar si es viable cada idea. |
| Abandono de integrantes | 10% | 1 | Gestionar el equipo, para repartirse las labores del proyecto. |

Planificación de recursos

|  |  |
| --- | --- |
| **Recursos Software** | **Recursos Hardware** |
| * PuTTY * Balena Erchen * Visual Studio Code * EV3DEV * Python 3 y librerías * RPyC * TKinter | * Notebook * Micro SD * Lego MindStorm Ev3 * Dongle Wifi * Router |

Estimación de costos

|  |  |  |  |
| --- | --- | --- | --- |
| **Materiales** | **Costo (pesos chilenos)** | **Unidades** | **Producto** |
| Tarjeta micro SD | 5.000 | 1 | 5.000 |
| Notebook | 350.000 | 2 | 700.000 |
| Lego ev3 MindStorm | 350.000 | 1 | 350.000 |
| Dongle Wifi | 7.400 | 1 | 7.400 |
| Router | 39.990 | 1 | 39.990 |
| Bebida Coca Cola | 350 | 1 | 350 |
| **Total Materiales:** |  |  | 1.102.740 |
| **Recursos humanos** | **Costo (Pesos/hora)** | **Cantidad** | **Duración (h)** |
| Valor hora de los integrantes | 15.000 | 5 | 96 |
| **Total recursos humanos:** |  |  | 7.200.000 |
| TOTAL: |  |  | 8.302.740 |

ANÁLISIS DE DISEÑO

Especificación de requerimientos

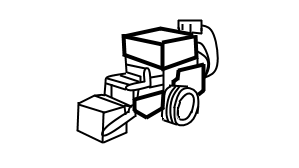
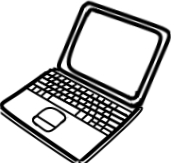
**Funcionales**

1. Se requiere que el robot se acerque a una lata, la alce y pueda llevársela como carga.
2. Se requiere que el robot voltee la lata que carga consigo y la suelte en posición invertida en el suelo.
3. Se requiere que el robot juegue Flip-tac-toe.
4. Se requiere que el robot se pueda manipular remotamente usando interfaz gráfica.

**No funcionales**

1. Se requiere que el código del robot sea diseñado en lenguaje Python.
2. Se requiere que el robot sea armado sólo usando piezas Lego.
3. Se requiere que la lata del juego sea el modelo de lata de bebida más pequeño que se encuentre a la venta.

Arquitectura propuesta



**RPyC**

Software Movimiento:

Abrir y cerrar garra

Hacer girar la garra

Decir mensaje de voz

Avanzar hacia donde mira el robot

Retroceder

Girar garra a la derecha

Girar garra a la izquierda

Frenar robot

Software Interfaz:

Muestra el título “Arturi-toe”

Botón para avanzar

Botón para retroceder

Botón para ir a la izquierda

Botón para ir a la derecha

Botón para frenar

Botón para abrir garra

Botón para cerrar garra

Servidor

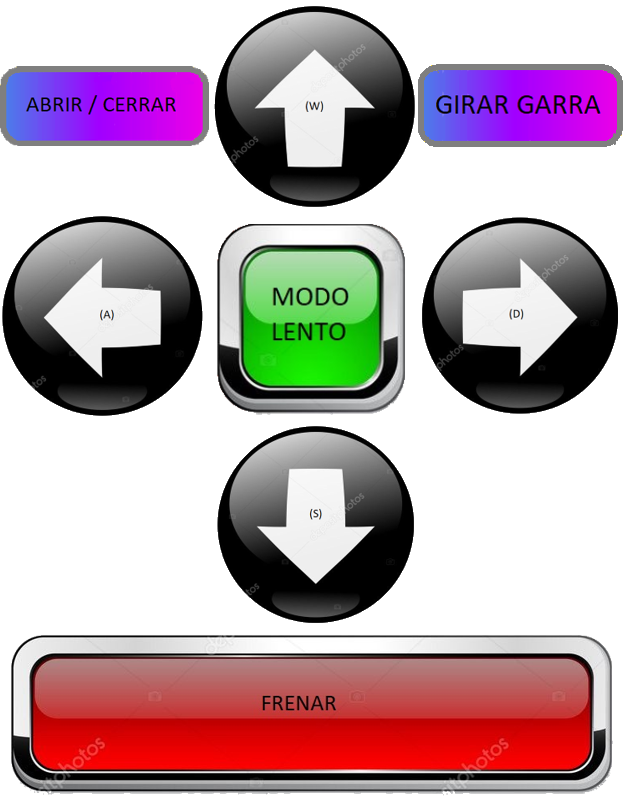
Cliente

**Lego Mindstorm Ev3**

**Portátil**

El cliente envía las órdenes mediante el computador al RPyC y este llama a las funciones que ya están en el robot.

Diseño de la interfaz de usuario

El usuario dispondrá de una interfaz como la que se muestra en la imagen, para poder controlar y comunicarse remotamente con el robot. El funcionamiento de los botones es: 

1. Abrir (cerrar): El robot abre (cierra) su garra mientras se mantenga presionado el botón. Por defecto al prender el robot el botón está en modo “abrir”.
2. Avanzar: El robot avanza mientras se mantenga presionado el botón.
3. Girar: El robot gira su garra 180° grados.
4. Izquierda: El robot gira en su propio eje hacia la izquierda mientras se mantenga presionado el botón.
5. Retroceder: El robot retrocede mientras se mantenga presionado el botón.
6. Derecha: El robot gira en su propio eje hacia la derecha mientras se mantenga presionado el botón.
7. Frenar: Detiene todos los procesos que están en marcha.
8. Modo lento: La misma función de “avanzar” pero con menos velocidad.

IMPLEMENTACIÓN

A través de la herramienta pydoc [4][5].

* Nota: El formato Numpy/Scipy es el que está siendo usado oficialmente en la captura de pantalla. Aclarar que el código está optimizado de tal manera que en algunas partes no hay variables que se usen como argumento o como retorno de las funciones, tan sólo funciones sin argumento que trae la librería.





PRUEBAS

* + - 1. Descripción de las pruebas realizadas

Se prendió al robot, se hizo conexión remota con una computadora portátil y se lo puso a andar por el suelo del laboratorio. Las órdenes de avanzar, retroceder se daban desde la computadora manteniendo presionadas las teclas respectivas. De forma similar se controlaban las funciones de girar derecha, girar izquierda, abrir garra, voltear garra, avanzar lento.

El robot controlado por un miembro del equipo se dispuso en el suelo del laboratorio donde también habían: otros robots de otros equipos de trabajo, latas de bebida vacías y las mesas del laboratorio. Usando los controles de la computadora el miembro del equipo tenía que: llevar al robot desde la mesa hasta la lata, parquearlo cerca para que pudiese tomarla con la garra, efectuar un giro de la lata, y regresarlo al lugar junto a la mesa de donde partió.

* + - 1. Resultados de las pruebas

El robot cumple de manera prometedora el trabajo de ir en dirección hacia la lata y regresar al sitio cerca a la mesa de donde partió. El tiempo de parqueo, por otra parte, se extiende más de lo esperado.

Las funciones de abrir la garra y voltearla con la lata retenida no presentan mayor inconveniente cuando el robot ya está parqueado cerca a la lata.

RESULTADOS

1. Estado actual del proyecto

Los movimientos del robot se controlan única y completamente por el ordenador, a través de una interfaz gráfica hecha con software Python. La garra rota, se abre y cierra por comandos de la interfaz y es capaz de tomar la lata de aluminio y girarla. El robot logra avanzar mientras se presiona el botón de la interfaz destinado a esa función. Asimismo, al soltarlo, se frena; no obstante, hay un botón de frenado redundante. Además, el robot logra girar a la derecha, izquierda, retroceder de manera similar a como funciona el control de avance hacia adelante.

El código en Python actual permite no sólo el disponer de una interfaz gráfica sino también unos controles más intuitivos para el usuario. Cada control tiene asociada una tecla diferente del teclado, con el añadido de que se pueden presionar los botones en pantalla para hacer las mismas funciones que hace ya el teclado. Entre estas funciones: avanzar, retroceder, girar a la izquierda, girar a la derecha, dejar la garra en modo abrir, dejar la garra en modo cerrar, detener la apertura/ cierre de la garra de golpe y, girar la garra.

La conexión que se usa para conectar al cliente -la computadora- con el servidor -el robot- es vía Wifi.

El robot ya se ha probado en un entorno parecido al de Flip-tac-toe con latas y otros robots en el campo.

Existe un manual de usuario que hasta ahora sólo explica los componentes físicos del robot y sus funciones.

Existe una wiki que incluye desde fotos del robot armado en su totalidad hasta una explicación de los componentes de su interfaz. Incluye introducción, objetivos generales, objetivos específicos, gestión de los gastos, riesgos, entre otras cosas que se desarrollan también en el informe.

El informe del proyecto incluye una introducción, objetivos generales, objetivos específicos, actividades, gestión de los gastos, gestión de riesgos, carta Gantt; además del análisis de diseño, resultados, implementación, pruebas realizadas, trabajo a futuro y más referencias en forma de anexos.

1. Problemas encontrados y soluciones propuestas

Si bien la garra desempeña su función de tomar la lata y girarla aún existe dificultad en la tarea de frenar al robot frente a la lata sin chocarla ni tumbarla. Este hecho está conectado al hecho de que al soltar el botón de avance de la interfaz el robot sigue avanzando por unos momentos (menos de 1s). Para hacer frente a esto, se propone resolver los retrasos en la conexión a internet usando no la red Wifi común del laboratorio donde se hará la competencia, sino una red propia del equipo compartida por uno de sus miembros, usando su plan de datos. Otra alternativa sería implementar un botón nuevo de avanzar lento, dejando la velocidad por defecto del robot en rápida, como ya estaba configurada.

Si bien la interfaz que se ha estado usando cumple con tener todos los botones de movimiento, el diseño es poco intuitivo y acaba por dificultar su uso. Por lo que se propone rediseñar desde cero esta interfaz que ya se tenía, para reemplazar por un diseño más intuitivo.

1. Conclusiones

* El robot recibe órdenes de una computadora vía una red wifi. Esto bien lo limita a la velocidad o conexión que le brinda la red y podría mejorarse con una red especial sólo para el robot.
* El robot desempeña satisfactoriamente sus funciones de moverse a lo largo del suelo, agarrar la lata de bebida y girarla. No quedan otros requerimientos funcionales por añadir en los objetivos de este proyecto.
* El robot posee conexión a una interfaz en la computadora que permite de manera intuitiva dar órdenes a cada una de las funciones de movimiento, funciones de uso de la garra. El retraso que se observa al presionar un botón de la interfaz y medir el tiempo de reacción del motor del robot se puede aliviar mejorando la conexión de Wifi.
* Las reglas del juego Flip-tac-toe son:
  1. Se tiene un tablero de 3x3 baldosas (7x7 por el tamaño del robot), donde se jugará.
  2. Cada jugador tiene por objetivo alinear 3 latas a su favor como en el clásico juego “gato”.
  3. Cada jugador tiene una única posibilidad de dar vuelta una lata posicionada por el contrincante, con el fin de que sea considerada para su favor.
  4. El robot puede chocar a su contrincante, pero si lo bota tendrá una penalización.

1. Trabajo futuro

Si bien el contexto del proyecto le da una característica relevante que es lo lúdico, lo que la competencia entre robots es una actividad de ocio; esto no impide que a futuro el conocimiento obtenido en las áreas de robótica quede sin utilizarse. Debido al alcance significativo que tienen los conocimientos de los diagramas cliente-servidor para el área de, por ejemplo, manejo de redes, se profundizará más en el estudio de sus aplicaciones. Se profundizará en el tema a través de lectura comprensiva de papers informativos, así como el consumo de material audiovisual referente al tema encontrado el YouTube.

REFERENCIAS

[1] Grupo lego, Building Instructions & Program Descriptions, [online], <https://education.lego.com/en-us/support/mindstorms-ev3/building-instructions>.

[2] [William's Robot Friends](https://www.youtube.com/channel/UCeS4rfDb0aPnKygJHYRMxqg), 16 abril 2015, Most Simple EV3 Robot Claw, [online], <https://www.youtube.com/watch?v=VEsLKZAAoSc> .

[3] LEGO®, ev3dev is your EV3 re-imagined, [online], <https://www.ev3dev.org/> .

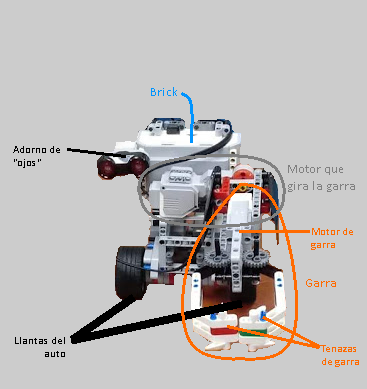
[4] R,H, 2015, Pythom-ev3, [online], <https://ev3dev-lang.readthedocs.io/projects/python-ev3dev/en/stable/spec.html>

[5] J. Mertz, Documenting Python Code: A Complete Guide, [online], https://realpython.com/documenting-python-code/#documenting-your-python-code-base-using-docstrings

[6] Pherkad, 1 febrero 2014, Python 3 para impacientes, [online], https://python-para-impacientes.blogspot.com/2014/02/docstrings.html

ANEXOS

Anexo A: Hardware



Anexo B: Software

Código para el cliente

nuv.py

|  |
| --- |
| import rpyc  from tkinter import \*  conn = rpyc.connect('192.168.70.50', port=65535)  conn.root.speak\_message('holiii')  conn.close()  window = Tk()    window.title("---ARTURITOE---")    window.geometry('350x200')    def avanzar():  conn.root.Avanzar()  print("holaaa")  def retroceder():  conn.root.Retroceder()  def izquierda():  conn.root.Girar\_izq()  def derecha():  conn.root.Girar\_der()  def frenar():  conn.root.Frenar()  def Abrir():  conn.root.Abrir()  def Cerrar():  conn.root.Cerrar()  def girar():  conn.root.Giro\_Garra()  #def salir()    window.bind('w', lambda a : avanzar())  window.bind('s', lambda a : retroceder())  window.bind('a', lambda a : izquierda())  window.bind('d', lambda a : derecha())  window.bind('<space>', lambda a : frenar())  window.bind('q', lambda a : Abrir())  window.bind('e', lambda a : Cerrar())  window.bind('r', lambda a : girar())  #window.bind('<Escape>', lambda a : salir())  btn = Button(window, text="avanzar", command=avanzar)  btn\_2 = Button(window, text="retroceder", command=retroceder)  btn\_3 = Button(window, text="izquierda", command=izquierda)  btn\_4 = Button(window, text="derecha", command=derecha)  btn\_5 = Button(window, text="frenar", command=frenar)  btn\_abrir = Button(window, text="abrir", command=Abrir)  btn\_cerrar = Button(window, text="cerrar", command=Cerrar)  btn\_girar = Button(window, text="girar", command=girar)  btn.grid(column=1, row=0)  btn\_2.grid(column=1, row=1)  btn\_3.grid(column=0, row=1)  btn\_4.grid(column=2, row=1)  btn\_5.grid(column=1, row=2)  btn\_abrir.grid(column=0, row=0)  btn\_cerrar.grid(column=2, row=0)  btn\_girar.grid(column=3, row=0)  window.mainloop() |

Código del servidor

prueba2.py

|  |
| --- |
| #!/usr/bin/env python3  import rpyc  from rpyc.utils.server import ThreadedServer  from ev3dev.ev3 import \*  class Robot(rpyc.Service):  nombre="arturitoe"  giro=True  garra\_abierta=True  rueda\_izq=Motor(OUTPUT\_A)  rueda\_der=Motor(OUTPUT\_D)  garras=MediumMotor(OUTPUT\_B)  girar\_garra=Motor(OUTPUT\_C)      def exposed\_Abrir\_Cerrar(self):  if self.garra\_abierta==True:  self.garras.run\_timed(time\_sp=800,speed\_sp=1080)  self.garra\_abierta=False  else:  self.garras.run\_timed(time\_sp=800,speed\_sp=-1080)  self.garra\_abierta=True    def exposed\_Giro\_Garra(self):  if self.giro==True:  self.girar\_garra.run\_timed(time\_sp=500,speed\_sp=400)  self.giro=False  else:  self.girar\_garra.run\_timed(time\_sp=500,speed\_sp=-400)  self.giro=True  def exposed\_speak\_message(self, msg):  Sound.speak(msg)  def exposed\_Avanzar(self):  self.rueda\_der.run\_forever(speed\_sp=500)  self.rueda\_izq.run\_forever(speed\_sp=500)  def exposed\_Retroceder(self):  self.rueda\_der.run\_forever(speed\_sp=-500)  self.rueda\_izq.run\_forever(speed\_sp=-500)  def exposed\_Girar\_der(self):  self.rueda\_der.run\_forever(speed\_sp=200)  def exposed\_Girar\_izq(self):  self.rueda\_izq.run\_forever(speed\_sp=200)  def exposed\_Frenar(self):  self.rueda\_der.stop()  self.rueda\_izq.stop()    if \_\_name\_\_ == "\_\_main\_\_":  s = ThreadedServer(Robot, port=65535)  s.start()  print("Server up") |

Anexo C: Comunicaciones

|  |
| --- |
| {  "ev3devBrowser.additionalDevices": [  {  "name": "arturitoe",  "ipAddress": "192.168.70.166"    }  ],  "launch": {  "version": "0.2.0",  "configurations": [  {  "name": "Download and Run",  "type": "ev3devBrowser",  "request": "launch",  "program": "/home/robot/${workspaceRootFolderName}/${relativeFile}"  }  ]  },  "ev3devBrowser.password" : "arturitoe"  } |