**UNIVERSIDAD DE TARAPACA**

 **ESCUELA UNIVERSITARIA DE INGENIERIA INDUSTRIAL, INFORMATICA Y DE SISTEMAS**

**AREA INGENIERIA CIVIL EN COMPUTACIÓN E INFORMATICA**

**PLAN DE PROYECTO**

|  |  |
| --- | --- |
| **Asignatura:** | Proyecto I |
| **Alumnos:** | Bravo, Axl  |
|  | Gaete, Jeremy  |
|  | Gavia, Scarlet  |
|  | Zúñiga Romo, Sebastián  |
| **Profesores:** | Ricardo ValdiviaDiego Aracena |

**ARICA - CHILE**

**2018**

**Historial de Cambios**

|  |  |  |  |
| --- | --- | --- | --- |
| Fecha | Versión | Descripción | Auto(es) |
| 9/09/2018 | 1.0 | Informe I finalización  | Sebastián Zuñiga |
| 04/10/2018 | 1.1 | Modificación informe-Restricciones-Canta Gantt | Sebastián Zuñiga |
| 04/10/2018 | 2.0 | Agregación de puntos 5, 6, 7 y 8 del Informe II | Sebastián Zuñiga |
| 20/10/2018 | 2.1 | Realización de la implementación. | Axl Bravo |
| 22/10/2018 | 2.2 | Agregación de resultados 7.1 y 7.2 | Scarlet Gavia |
| 23/10/2018 | 2.3 | Análisis y diseño | Sebastián Zuñiga |
| 22/11/2018 | 2.4 | 7. Descripción de las Pruebas realizadas y resultados de las pruebas. | Scarlet GaviaSebastián Zuñiga |

**Tabla de contenidos**

[1. Panorama General 1](#_Toc531821991)

[**1.1. Introducción (contexto) 1**](#_Toc531821992)

[**1.2. Objetivos General 1**](#_Toc531821993)

[**1.3. Objetivos específicos 1**](#_Toc531821994)

[**1.4. Restricciones 1**](#_Toc531821995)

[**1.5. Entregables 2**](#_Toc531821996)

[2. Organización del Personal 4](#_Toc531821997)

[**2.1. Descripción de Roles 4**](#_Toc531821998)

[**2.2. Personal que cumplirá roles 4**](#_Toc531821999)

[**2.3. Mecanismo de Comunicación 4**](#_Toc531822000)

[3. Planificación del proyecto 5](#_Toc531822001)

[**3.1. Actividades (nombre, descripción, responsable, producto) 5**](#_Toc531822002)

[**3.2. Asignación del tiempo (carta Gantt Redmine) 7**](#_Toc531822003)

[**3.3. Personal-rol asignado 7**](#_Toc531822005)

[**3.4. Gestión de Riesgos (ver plantilla para el tratamiento de los Riesgos) 8**](#_Toc531822006)

[4. Planificación de los Recursos 10](#_Toc531822007)

[**4.1. Recursos Hardware y Software requeridos 10**](#_Toc531822008)

[**4.1.1. Hardware 10**](#_Toc531822009)

[**4.1.2. Software 10**](#_Toc531822010)

[**4.2. Estimación de Costos (Hardware, Software, Recursos Humanos) 10**](#_Toc531822011)

[5. Análisis – Diseño 12](#_Toc531822012)

[5.1. Especificación de Requerimientos (incluyendo método/algoritmo considerados para resolver el cubo Rubik). 12](#_Toc531822013)

[**5.1.1. Requisitos Funcionales 12**](#_Toc531822014)

[**5.1.2. Requisitos No Funcionales 12**](#_Toc531822015)

[**5.2. Arquitectura Propuesta (incluyendo aspectos de comunicación). 13**](#_Toc531822016)

[**5.3. Diseño de la Interfaz Usuario 14**](#_Toc531822017)

[6. Implementación 17](#_Toc531822018)

[**6.1. Descripción de los programas implementados (entradas, salidas, procesos) 17**](#_Toc531822019)

[**6.2. Diagrama de interacción entre programas 21**](#_Toc531822020)

[**8. Resultados 28**](#_Toc531822030)

[**8.1. Estado final del proyecto 28**](#_Toc531822031)

[**8.2. Conclusiones 28**](#_Toc531822032)

[**9. Referencias (estándar IEEE) 29**](#_Toc531822033)

[**Anexo 30**](#_Toc531822034)

[**Código del robot 30**](#_Toc531822035)

Tabla de imágenes

[1.3.2. Imagen Carta Gantt 7](#_Toc531821505)

[1.4.1. Imagen Mov\_frontPrima(): (Movimiento Frontal Inverso) 21](#_Toc531821367)

[1.4.2. Imagen Mov\_backPrima(): (Movimiento hacia atrás inverso) 21](#_Toc531821368)

[1.4.3. Imagen Mov\_LPrima(): (Movimiento Izquierda Inverso) 22](#_Toc531821369)

[1.4.4. Imagen Mov\_RPrima(): (Movimiento Derecha Inverso) 22](#_Toc531821370)

[1.4.5. Imagen Mov\_UPrima(): (Movimiento Cara Superior Inversa) 23](#_Toc531821371)

[1.4.6. Imagen Mov\_L(): (Movimiento Izquierda) 23](#_Toc531821372)

[1.4.7. Imagen Mov\_R(): (Movimiento Derecha) 24](#_Toc531821373)

[1.4.8. Imagen Mov\_U(): (Movimiento Cara superior) 24](#_Toc531821374)

[1.4.9. Imagen SexyMove(): (Movimiento Sexy) 25](#_Toc531821375)

# Panorama General

## Introducción (contexto)

Desde que el hombre vio la posibilidad de usar máquinas en vez de personas comenzó un desarrollo, fue una investigación una evolución. Aunque el concepto de maquinas automatizadas se remonta a la antigüedad, el robot es el aparato más popular en los últimos tiempos o específicamente en el último siglo, capaz de realizar complejos algoritmos. Nuestra finalidad es demostrar de manera tangible la capacidad del robot con respecto a la resolución de algoritmos, específicamente en un cubo Rubik.

## Objetivos General

* Construir, Desarrollar y programar un robot que sea capaz de ejecutar secuencias de algoritmos de cubo Rubik de forma remota y así ayudar a un usuario a armar un cubo Rubik.

## Objetivos específicos

* Realizar el armado del robot.
* Estudiar los algoritmos de cubo Rubik.
* Programar los algoritmos en lenguaje Python.
* Diseñar una interfaz gráfica para el usuario.
* Analizar los resultados.

## Restricciones

* Contar solo con el semestre para hacer el proyecto.
* No encontrar las piezas necesarias para la construcción del robot.
* Lenguaje de programación Python.

## Entregables

|  |  |  |
| --- | --- | --- |
| Identificación Entregable | Descripción entregable | Fecha de entrega |
| Bitácora I | Reporte semanal de tareas | 16 de agosto de 2018 |
| Bitácora II | Reporte semanal de tareas | 23 de agosto de 2018 |
| Bitácora III | Reporte semanal de tareas | 30 de agosto de 2018 |
| Informe I | Entrega informe I | 12 de agosto de 2018 |
| Presentación I | Entrega de Presentación I | 12 de agosto de 2018 |
| Redmine | Actualización de la wiki | 20 de agosto de 2018 |
| Bitácora IV | Reporte semanal de tareas | 6 de septiembre de 2018 |
| Bitácora V | Reporte semanal de tareas | 13 de septiembre de 2018 |
| Bitácora VI | Reporte semanal de tareas | 27 de septiembre de 2018 |
| Bitácora VII | Reporte semanal de tareas | 4 de octubre de 2018 |
| Bitácora VIII | Reporte semanal de tareas | 18 de octubre de 2018 |
| Informe II | Entrega Informe II | 25 de octubre de 2018 |
| Presentación II | Entrega Presentación II | 25 de octubre de 2018 |
| Bitácora IX | Reporte semanal de tareas | 25 de octubre de 2018 |
| Bitácora X | Reporte semanal de tareas | 1 de noviembre de 2018 |
| Bitácora XI | Reporte semanal de tareas | 8 de noviembre de 2018 |
| Bitácora XII | Reporte semanal de tareas | 15 de noviembre de 2018 |
| Bitácora XIII | Reporte semanal de tareas | 22 de noviembre de 2018 |
| Informe II de avance | Entrega del Informe II de avance | 22 de noviembre de 2018 |

# Organización del Personal

## Descripción de Roles

|  |  |
| --- | --- |
| Rol | Descripción |
| Jefe de grupo | Es aquel capaz de tomar decisiones en momentos de alta tensión además de ejercer de líder en ciertos casos. |
| Programador | Es aquella persona que escribe el código para la adaptación de los algoritmos Rubik. |
| Diseñador | Es aquel capaz de armar el robot y realizar presentaciones. |
| Secretario | Es aquella persona capaz de la construcción y redacción del informe |

## Personal que cumplirá roles

|  |  |  |
| --- | --- | --- |
| Actividad | Responsable | Involucrados |
| Documentación del proyecto | Sebastián Zuñiga | Sebastián ZuñigaJeremy Gaete |
| Ensamblado del robot | Scarlet Gavia | Jeremy GaeteAxl BravoScarlet Gavía  |
| Trabajo de código | Axl Bravo | Axl Bravo Jeremy GaeteScarlet Gavia |

## Mecanismo de Comunicación

* Cuentas en redes sociales Grupo WhatsApp del proyecto

# Planificación del proyecto

## Actividades (nombre, descripción, responsable, producto)

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre | Descripción | Responsable | Producto |
| 1. Planteamiento del Problema
 |  |  |  |
| 1. Documentación del Proyecto
 | Trabajo en el plan de proyecto | Sebastián Zuñiga | Informe I |
| * 1. Bitácoras semanales
 | Actualizar Bitácoras cada semana | Jeremy Gaete | Bitácoras |
| 1. Ensamblado de Robot
 |  |  |  |
| * 1. Adquirir piezas del Robot
 |  | Scarlet Gavia |  |
| * 1. Armar Robot
 | Comienza el Armado del robot | Axl Bravo | Construcción del robot |
| 1. Trabajo de Código
 | Programación del robot |  |  |
| * 1. Estudiar Lenguaje de programación
 | Estudio del lenguaje Python | Axl Bravo | Completar 1° etapa de programación |
| * 1. Comprensión y adaptación del código de mind Cuber
 | Estudio y modificación del código mind Cuber | Jeremy Gaete |  |
| * 1. Estudio de la librería ev3dev-lang-python
 | Plugin de Visual Studio | Jeremy Gaete |  |
| 1. Programas del robot
 |  |  |  |
| * 1. Adaptación del Movimiento del Robot
 | Programar movimiento para que haga lo deseado | Jeremy Gaete | Seguir con la programación |
| * 1. Programación de Algoritmos Rubik
 | Realizar los movimientos para generar los algoritmos de cubo Rubik | Scarlet Gavia | Realizar los últimos cambios al robot |
| * 1. Programación de la Interfaz EV3
 |  | Scarlet Gavia | Interfaz gráfica terminada |
| 1. Implementación de la solución de comunicación remota
 |  | Jeremy Gaete |  |
| * 1. Pruebas finales del robot
 | Hacer pruebas finales al robot | Axl Bravo | Finalizar el proyecto con el robot terminado |

## Asignación del tiempo (carta Gantt Redmine)

# 1.3.2. Imagen Carta Gantt



Fuente: (Carta Gantt, 6 de diciembre de 2018)

## Personal-rol asignado

(Carta Gantt)

El equipo consta con:

* 1 jefe de grupo, encargado de liderar y asistir al resto del Equipo (Jeremy Gaete).
* 1 secretario, encargado del informe (Sebastián Zuñiga).
* 1 programador, en cargado de la programación del robot (Axl Bravo).
* 1 diseñador, encargado de lo que es la parte de construcción del robot y de PowerPoint (Scarlet Gavia).

## Gestión de Riesgos (ver plantilla para el tratamiento de los Riesgos)

1. CATASTRÓFICO
2. CRÍTICO
3. MARGINAL
4. DESPRECIABLE

|  |  |  |  |
| --- | --- | --- | --- |
| Riesgo | Probabilidad de Ocurrencia ( % ) | Nivel de impacto | Acciones Remediales |
| Falta de personal | 20 | 2 | Se redistribuirán las tareas para alivianar la carga que implica la falta de un integrante del equipo. |
| Inexperiencia en programación | 25 | 2 | Se deberá tomar más tiempo para el aprendizaje del mismo. |
| Lentitud en la toma de decisiones  | 10 | 3 | El jefe de grupo tomara las decisiones finales en caso de duda. |
| Baja motivación | 25 | 2 | Se deberá animar al compañero en caso de que haya desanimo. |
| Accidentes/Enfermedades | 20 | 3 | Se redistribuirán cargos. |
| Oposición comunitaria | 20 | 1 | Se volverá a discutir los temas o se hará lo que el jefe de grupo decida en otro caso. |
| Falta de servicios básicos entregados | 15 | 3 | Se dará aviso al encargado en caso de que exista dicha falta. |
| Irresponsabilidad del personal | 20 | 2 | La persona será amonestada posteriormente. |
| Indisponibilidad de las herramientas de desarrollo | 10 | 3 | Se consultara con el encargado del lugar para buscar una solución  |
| Perdida de piezas del robot | 10 | 4 | Se hablará el problema como grupo y si se necesitaran mas piezas se consultará al encargado |

# Planificación de los Recursos

## Recursos Hardware y Software requeridos

### **Hardware**

* Robot EV3
* Piezas lego para la construcción del robot
* 1 notebook para la instalación y programación del robot
* Cubo Rubik

### **Software**

* Python
* Visual Studio Code.
* Windows

## Estimación de Costos (Hardware, Software, Recursos Humanos)

Se ha desarrollado como equipo un plan de costos aproximado, que reflejó los gastos que solvento el equipo.

|  |  |  |  |
| --- | --- | --- | --- |
| Elemento | Detalle | Costo por persona | Costo en CLP |
| Cubo Rubik | Cubo Rubik de 6 caras 3x3 | 1500 | 5.500 |
| Robot EV3 | Lego Mindstorms EV3 |  | 490.209 |
| Sueldo Coordinador de grupo | Ganancia mensual del Coordinador |  | 1.500.000 |
| Sueldo secretario | Sueldo mensual del secretario |  | 400.000 |
| Sueldo programador | Sueldo mensual programador |  | 700.000 |
| Sueldo Diseñador | Sueldo mensual diseñador |  | 450.000 |
| Movilización | Costo del uso del microbús diario | 1. – 800
 | 1.040 – 3.400 |
| Total(mensual) |  |  | 3.549.109 |
| Total(semestre) |  |  | 17.745.545 |

# Análisis – Diseño

# **Especificación de Requerimientos (incluyendo método/algoritmo** considerados para resolver el cubo Rubik).

### **Requisitos Funcionales**

|  |  |
| --- | --- |
| NOMBRE | DESCRIPCIÓN DEL REQUISITO |
| Comunicación | Tendrá un menú de opciones necesarias para realizar movimientos básicos además de los algoritmos. |
| El Robot | Podrá armar el cubo mediante los algoritmos. |

### **Requisitos No Funcionales**

|  |  |
| --- | --- |
| NOMBRE | DESCRIPCION DE REQUISITO |
| Tiempo de ejecución | Cada movimiento deberá ejecutarse en un tiempo estimado |
| Manual de usuario | El robot debe contar con su manual de usuario |
| Interfaz grafica | Debe disponer de una Interfaz Grafica |

## Arquitectura Propuesta (incluyendo aspectos de comunicación).

## Diseño de la Interfaz Usuario

Bosquejo de la Interfaz de usuario.


Tendrá 3 opciones:
-Presionar Notación Básica.

-Presionar Movimientos principales.

-Patrones avanzados.





Opción 1°: Notación Submenú con la notación básica del cubo Rubik 3x3

Opción 2: Movimiento principales

Donde se realizan los movimientos netamenente del robot.



Opcion 3: Patrones

Submenú con opciones para armar distintos patrones en el Rubik



# Implementación

## Descripción de los programas implementados (entradas, salidas, procesos)

Primeramente, se dará inicio con las funciones de movimiento principales, dichas funciones de movimiento son las más básicas, estas son por ejemplo rotar la base a la izquierda o derecha, con o sin la garra y cambiar de cara.

Funciones de movimiento principal:

 def giro\_Der\_G(): Función que permite girar la base en sentido horario con la garra encima.

 def giro\_Der(): Función que permite girar la base en sentido horario, sin la garra.

 def giro\_Izq\_G(): Función que permite girar la base en sentido anti horario con la garra encima.

 def giro\_Izq(): Función que permite girar la base en sentido anti horario, sin la garra.

 def g\_subir(): Esta función sirve para levantar la garra, esta posición es la inicial para todos los movimientos.

 def g\_bajar(): Esta función se utiliza para bajar la garra, con esta función es importante para poder realizar los cambios de cara o giros de base.

 def cambio\_cara(): Con esta función podremos cambiar de cara en el cubo, es decir, si el cubo tiene como posición inicial el color rojo de frente y el color blanco en la parte superior, con esta función lograremos que la cara de color quede en la parte superior.

 Movimientos básicos: Estos movimientos son compuestos de los movimientos principales, son utilizados para realizar los movimientos por defecto o estándar del cubo.

Movimientos Básicos

def mov\_L(): Función que permite el movimiento de la parte lateral izquierda del cubo hacia arriba (Definición estándar L).

Función compuesta por: def giro\_Der(), def g\_Bajar(), def cambio\_cara(), def giro\_Izq().

def mov\_LPrima(): Función que permite el movimiento de la parte lateral izquierda del cubo hacia abajo (Definición estándar L’).

Función compuesta por: def giro\_Der(), def g\_Bajar(), def cambio\_cara(), def giro\_Izq\_G(), def giro\_Izq().

def mov\_R(): Función que permite el movimiento de la parte lateral derecha del cubo hacia arriba (Definición estándar R).

Función compuesta por: def giro\_Izq(), def g\_Bajar(), def cambio\_Cara(), def giro\_Izq\_G(), def giro\_Der().

def mov\_RPrima(): Función que permite el movimiento de la parte lateral derecha del cubo hacia abajo (Definición estándar R’).

Función compuesta por: def giro\_Izq(), def cambio\_Cara(), def g\_Bajar(), def giro\_Der\_G(), def giro\_Der(),

def mov\_U(): Función que permite girar la parte superior de la cara frontal en sentido horario (Definición estándar U).

Función compuesta por: def cambio\_Cara(), def g\_Bajar(), def giro\_Izq\_G().

def mov\_UPrima(): Función que permite girar la parte superior de la cara frontal en sentido anti horario (Definición estándar U’).

Función compuesta por: def cambio\_Cara(), def g\_Bajar, def giro\_Der\_G()

def mov\_D(): Cumple la misma función que def giro\_Izq\_G() (definición estándar D)

def mov\_DPrima(): Cumple la misma función que def giro\_Der\_G() (definición estándar D’)

def mov\_front(): Función que permite el movimiento de la cara frontal en sentido horario (definición estándar F)

Función compuesta por: def giro\_Izq(), def g\_Bajar(), def cambio\_Cara(), def giro\_Izq\_G(),def g\_Subir(), def giro\_Der()

def mov\_frontPrima(): Función que permite el movimiento de la cara frontal en sentido anti horario (definición estándar F’)

Función compuesta por: def giro\_Izq(), def g\_Bajar(), def cambio\_Cara(), def giro\_Der\_G(), def g\_Subir(), def giro\_Der()

def mov\_backPrima(): Función que permite el movimiento de la cara trasera en sentido anti horario (definición estándar B’)

Función compuesta por: def g\_Baja(), def cambio\_Cara(), def g\_Bajar(), def giro\_Der\_G(), def g\_Subir().

def mov\_back(): Función que permite el movimiento de la cara trasera en sentido horario (definición estándar B)

Función compuesta por: def g\_Bajar(), def cambio\_Cara(), def giro\_Izq\_G(), def g\_Subir().

 Movimientos complejos: Algoritmos utilizados por gente experimentada que se utilizan para resolver el cubo de una forma más eficaz y eficiente. Estos movimientos son compuestos de los movimientos básicos.

Movimientos complejos

def sexyMove(): Función que permite el movimiento más común llamado sexy move.

Función compuesta por: def mov\_R(), def mov\_U(), def mov\_RPrima(),mov\_UPrima().

## Diagrama de interacción entre programas

# 1.4.1. Imagen mov\_frontPrima(): (Movimiento Frontal Inverso)



# 1.4.2. Imagen mov\_backPrima(): (Movimiento hacia atrás inverso)



# 1.4.3. Imagen mov\_LPrima(): (Movimiento Izquierda Inverso)



# 1.4.4. Imagen mov\_RPrima(): (Movimiento Derecha Inverso)



# 1.4.5. Imagen mov\_UPrima(): (Movimiento Cara Superior Inversa)



# 1.4.6. Imagen mov\_L(): (Movimiento Izquierda)



# 1.4.7. Imagen mov\_R(): (Movimiento Derecha)



# 1.4.8. Imagen mov\_U(): (Movimiento Cara superior)



# 1.4.9. Imagen sexyMove(): (Movimiento Sexy)



# Pruebas

## Descripción de las pruebas realizadas

* Prueba de conexión: Hemos realizado la conexión mediante wi-fi.
* Prueba de giros: Se han realizado múltiples giros de la base para poder configurar una mejor calibración.
* Pruebas básicas: Se probaron los movimientos básicos tales como “R”, “L” o “B”.
* Prueba de algoritmos implementados: Se probaron algoritmos tales como “sexyMove” o “cruzSuperior”, algoritmos compuestos por movimientos básicos

## Resultados de las pruebas

|  |  |
| --- | --- |
| Nombre  | Resultado  |
| Prueba de conexión  | Hemos realizado la conexión de manera exitosa.  |
| Prueba de giros  | Los giros fueron realizados exitosamente, pero cuando se ejecutan en gran cantidad hay problemas de calibración.  |
| Pruebas básicas | Las pruebas de los movimientos básicos se realizaron exitosamente, pero al ser compuestos por movimientos principales tales como los giros, heredan los problemas de calibración |
| Prueba de sexy move | Con ayuda de el usuario, el robot es capaz de ejecutar el algoritmo en su totalidad. |
| Prueba de algoritmos | Se presentan los problemas de calibración ya que estos algoritmos están compuestos por los movimientos básicos. |
|  |  |

## Resultados

## Estado final del proyecto

Nos encontramos en la finalización de nuestro proyecto, hemos completado en su totalidad la interfaz y la programación de los algoritmos, el robot ejecuta todos los algoritmos que se les ordene y dispone de una interfaz sencilla para que el usuario pueda armar su Rubik 3x3 de diversas maneras.

## Conclusiones

 Para finalizar hemos de decir que con esto hemos aprendido a usar el lenguaje de programación Python a un mayor nivel y con algunas librerías en específico, hemos aprendido a crear interfases gráficas en Python y aplicar todo eso a el uso del robot ev3dev, creando así un robot que haga algoritmos para la resolución del cubo Rubik y que un usuario lo pueda disfrutar.

 El robot tiene 12 movimientos básicos y 6 algoritmos para ayudar a armar el cubo, también tiene algoritmos para el cambio de aristas y esquinas de la base superior, con la finalidad de facilitar el armado de cubo al usuario.

* 1. **Trabajo a futuro**

A futuro planeamos actualizar nuestra interfaz gráfica, además de agregar más algoritmos y movimientos básicos como son : “E”, “E’ ”,”S”, “S’ ”, “M”, “M’ ”, ”LW” y “LW’ “, también agregaremos una mayor variedad de algoritmos para facilitar el armado del cubo Rubik y para finalizar planeamos buscar de configurar el robot para otros tipos de cubo.

## Referencias (estándar IEEE)

<http://mindcuber.com/> (Recuperado el 14 de agosto de 2018)

<https://www.ev3dev.org/docs/getting-started/> (recuperado el 6 de septiembre de 2018).

<https://www.youtube.com/watch?v=cqtRqsI6xMc> (Recuperado el 6 de septiembre de 2018).

<https://www.lego.com/es-es/mindstorms/apps/ev3-programmer-app> (Recuperado el 28 de agosto de 2018).

<https://www.youtube.com/watch?v=G2FCfQj-9ig> (Recuperado el 6 de diciembre de 2018).

<https://github.com/eliluminado/Guia-Tkinter> (Recuperado el 6 de diciembre de 2018).

## Anexo

###  **Código del robot**

#!/usr/bin/env: python3

import ev3dev.ev3 as ev3

from time import sleep

m = ev3.LargeMotor('outB')

l = ev3.LargeMotor('outA')

def giro\_Der\_G(): #Mover base con Garra a la Derecha

 m.run\_timed(time\_sp = 500, speed\_sp = 469)

 sleep(1)

 m.run\_timed(time\_sp = 500, speed\_sp = -78)

def giro\_Der(): #Mover base sin Garra a la Derecha

 m.run\_timed(time\_sp = 504, speed\_sp = 355)

 sleep(1)

def giro\_Izq\_G(): #Mover base con garra a la Izquierda

 m.run\_timed(time\_sp = 500, speed\_sp = -469)

 sleep(1)

 m.run\_timed(time\_sp = 502, speed\_sp = 90)

def giro\_Izq(): #Mover base sin garra a la Izquierda

 m.run\_timed(time\_sp = 506, speed\_sp = -355)

 sleep(1)

def g\_Bajar(): #Devolver garra cuando esta arriba

 l.run\_timed(time\_sp = 300, speed\_sp = 250)

 sleep(0.5)

 #l.run\_timed(time\_sp = 450, speed\_sp = -360)

def cambio\_Cara(): #Cambio de cara conservando la garra arriba (NO EDITAR, FUNCIONA BIEN)

 l.run\_timed(time\_sp = 300, speed\_sp = 250)

 sleep(0.5)

 l.run\_timed(time\_sp = 450, speed\_sp = -360)

def g\_Subir(): #Levantar garra

 l.run\_timed(time\_sp = 300, speed\_sp = -250)

def mov\_L():

 giro\_Der()

 sleep(1)

 g\_Bajar()

 sleep(1)

 cambio\_Cara()

 sleep(1)

 g\_Bajar()

 sleep(1)

 giro\_Der\_G()

 sleep(1)

 cambio\_Cara()

 sleep(1)

 g\_Bajar()

 sleep(1)

 cambio\_Cara()

 sleep(1)

 g\_Bajar()

 sleep(1)

 cambio\_Cara()

 sleep(1)

 giro\_Izq()

 sleep(1)

def mov\_LPrima():

 giro\_Der()

 sleep(1)

 g\_Bajar()

 sleep(1)

 cambio\_Cara()

 sleep(1)

 g\_Bajar()

 sleep(1)

 giro\_Izq\_G()

 sleep(1)

 cambio\_Cara()

 sleep(1)

 g\_Bajar()

 sleep(1)

 cambio\_Cara()

 sleep(1)

 g\_Bajar()

 sleep(1)

 cambio\_Cara()

 sleep(1)

 giro\_Izq()

 sleep(1)

def mov\_R():

 giro\_Izq()

 sleep(1)

 g\_Bajar()

 sleep(1)

 cambio\_Cara()

 sleep(1)

 g\_Bajar()

 sleep(1)

 giro\_Izq\_G()

 sleep(1)

 cambio\_Cara()

 sleep(1)

 g\_Bajar()

 sleep(1)

 cambio\_Cara()

 sleep(1)

 g\_Bajar()

 sleep(1)

 cambio\_Cara()

 sleep(1)

 giro\_Der()

def mov\_RPrima():

 giro\_Izq()

 sleep(1)

 cambio\_Cara()

 sleep()

 g\_Bajar()

 sleep(1)

 giro\_Der\_G()

 sleep(1)

 cambio\_Cara()

 sleep(1)

 g\_Bajar()

 sleep(1)

 cambio\_Cara()

 sleep(1)

 g\_Bajar()

 sleep(1)

 cambio\_Cara()

 sleep(1)

 giro\_Der()

 sleep(1)

def mov\_U():

 cambio\_Cara()

 sleep(1)

 g\_Bajar()

 sleep(1)

 cambio\_Cara()

 sleep(1)

 g\_Bajar()

 sleep(1)

 giro\_Izq\_G()

 sleep(1)

 cambio\_Cara()

 sleep(1)

 g\_Bajar()

 sleep(1)

 cambio\_Cara()

 sleep(1)

 g\_Bajar()

 sleep(1)

def mov\_UPrima():

 cambio\_Cara()

 sleep(1)

 g\_Bajar()

 sleep(1)

 cambio\_Cara()

 sleep(1)

 g\_Bajar()

 sleep(1)

 giro\_Der\_G()

 sleep(1)

 cambio\_Cara()

 sleep(1)

 g\_Bajar()

 sleep(1)

 cambio\_Cara()

 sleep(1)

 g\_Bajar()

 sleep(1)

def mov\_D():

 g\_Bajar()

 sleep(1)

 giro\_Izq\_G()

 sleep(1)

 g\_Subir()

 sleep(1)

def mov\_DPrima():

 g\_Bajar()

 sleep(1)

 giro\_Der\_G()

 sleep(1)

 g\_Subir()

 sleep(1)

def mov\_Front():

 giro\_Izq()

 sleep(1)

 giro\_Izq()

 sleep(1)

 g\_Bajar()

 sleep(1)

 cambio\_Cara()

 sleep(1)

 g\_Bajar()

 sleep(1)

 giro\_Izq\_G()

 sleep(1)

 g\_Subir()

 sleep(1)

 giro\_Der()

 sleep(1)

 giro\_Der()

 sleep(1)

 g\_Bajar()

 sleep(1)

 cambio\_Cara()

 sleep(1)

def mov\_FrontPrima():

 giro\_Izq()

 sleep(1)

 giro\_Izq()

 sleep(1)

 g\_Bajar()

 sleep(1)

 cambio\_Cara()

 sleep(1)

 g\_Bajar()

 sleep(1)

 giro\_Der\_G()

 sleep(1)

 g\_Subir()

 sleep(1)

 giro\_Der()

 sleep(1)

 giro\_Der()

 sleep(1)

 g\_Bajar()

 sleep(1)

 cambio\_Cara()

 sleep(1)

def mov\_BackPrima():

 g\_Bajar()

 sleep(1)

 cambio\_Cara()

 sleep(1)

 g\_Bajar()

 sleep(1)

 giro\_Der\_G()

 sleep(1)

 cambio\_Cara()

 sleep(1)

 g\_Bajar()

 sleep(1)

 cambio\_Cara()

 sleep(1)

 g\_Bajar()

 sleep(1)

 cambio\_Cara()

 sleep(1)

 g\_Bajar()

 sleep(1)

 g\_Subir()

 sleep(1)

def mov\_Back():

 g\_Bajar()

 sleep(1)

 cambio\_Cara()

 sleep(1)

 g\_Bajar()

 sleep(1)

 giro\_Izq\_G()

 sleep(1)

 cambio\_Cara()

 sleep(1)

 g\_Bajar()

 sleep(1)

 cambio\_Cara()

 sleep(1)

 g\_Bajar()

 sleep(1)

 cambio\_Cara()

 sleep(1)

 g\_Bajar()

 sleep(1)

 g\_Subir()

 sleep(1)

def sexyMov():

 mov\_R()

 sleep(1)

 mov\_U()

 sleep(1)

 mov\_RPrima()

 sleep(1)

 mov\_UPrima()

 sleep(1)

sexyMov()