UNIVERSIDAD DE TARAPACA

ESCUELA UNIVERSITARIA DE INGENIERIA INDUSTRIAL, INFORMATICA Y DE SISTEMAS

EUIIIS

AREA INGENIERIA CIVIL EN COMPUTACIÓN E INFORMATICA

Ingeniería Computación e Informática

PLAN DE PROYECTO

Asignatura: Proyecto I

Alumnos: Bravo, Axl
Gaete, Jeremy
Gavia, Scarlet
Zúñiga Romo, Sebastián

Profesores: Ricardo Valdivia
Diego 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	Analisis y diseño	Sebastian Zuñiga

Tabla de contenidos

1. Panorama General

1.1. 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.

1.2. Objetivos General

- Construir, Desarrollar y programar un robot que sea capazde ejecutar secuencias de algorimos de cubo Rubik de forma remota.

1.3. Objetivos específicos

- Realizar el armado del robot.
- Estudiar los algoritmos de cubo Rubik.
- Analizar los resultados

1.4. Restricciones

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

1.5. 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
Bitacora IV	Reporte semanal de tareas	6 de septiembre de 2018
Bitacora V	Reporte semanal de tareas	13 de septiembre de 2018
Bitacora VI	Reporte semanal de tareas	27 de septiembre de 2018
Bitacora VII	Reporte semanal de tareas	4 de octubre de 2018
Bitacora 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

2. Organización del Personal

2.1. Descripción de Roles

Programador: Descripcion

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	El aquella persona capaz de la construcción y redacción del informe

2.2. Personal que cumplirá roles

Actividad	Responsable	Involucrados
Documentación del proyecto	Sebastián Zuñiga	Sebastián Zuñiga Jeremy Gaete
Ensamblado del robot	Scarlet Gavia	Jeremy Gaete Axl Bravo Scarlet Gavía
Trabajo de código	Axl Bravo	Axl Bravo Jeremy Gaete Scarlet Gavia

2.3. Mecanismo de Comunicación

- Cuentas en redes sociales Grupo WhatsApp del proyecto

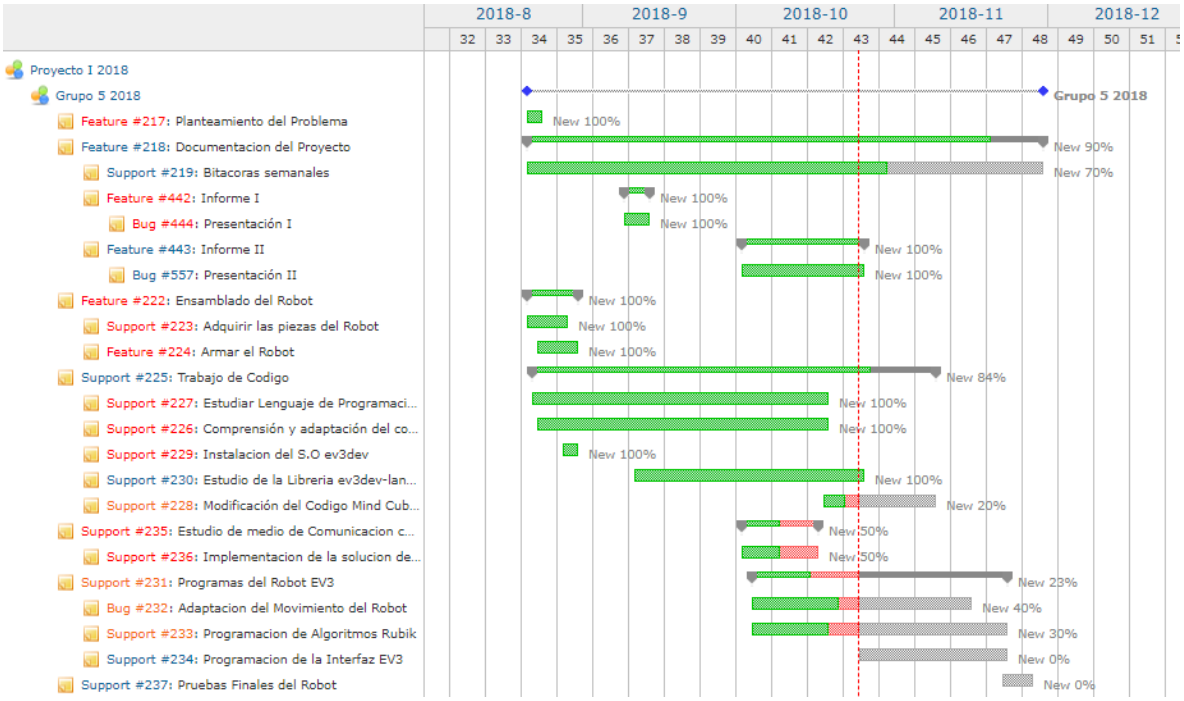
3. Planificación del proyecto

3.1. Actividades (nombre, descripción, responsable, producto)

Nombre	Descripción	Responsable	Producto
1. Planteamiento del Problema			
2. Documentación del Proyecto	Trabajo en el plan de proyecto	Sebastián Zuñiga	Informe I
2.1. Bitácoras semanales	Actualizar Bitácoras cada semana	Jeremy Gaete	Bitacoras
3. Ensamblado de Robot			
3.1. Adquirir piezas del Robot		Scarlet Gavia	
3.2. Armar Robot	Comienza el Armado del robot	Axl Bravo	Construcción del robot
4. Trabajo de Código	Programación del robot		
4.1. Estudiar Lenguaje de programación	Estudio del lenguaje Python	Axl Bravo	Completar 1° etapa de programación
4.2. Comprensión y adaptación del código de mind Cuber	Estudio y modificación del código mind Cuber	Jeremy Gaete	
4.3. Estudio de la librería	Plugin de Visual Studio	Jeremy Gaete	

ev3dev-lang-python			
5. Programas del robot			
5.1. Adaptación del Movimiento del Robot	Programar movimiento para que haga lo deseado	Jeremy Gaete	Seguir con la programación
5.2. Programación de Algoritmos Rubik	Realizar los movimiento para generar los algoritmos de cubo rubik	Scarlet Gavia	Realizar los últimos cambios al robot
5.3. Programación de la Interfaz EV3		Scarlet Gavia	Interfaz grafica terminada
6. Implementación de la solución de comunicación remota		Jeremy Gaete	
6.1. Pruebas finales del robot	Hacer pruebas finales al robot	Axl Bravo	Finalizar el proyecto con el robot terminado

3.2. Asignacion del tiempo (carta Gantt Redmine)



Fuente: (Carta Gantt, 24 de octubre de 2018)

3.3. 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).

3.4. 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

4. Planificación de los Recursos

4.1. Recursos Hardware y Software requeridos

4.1.1. Hardware

- Robot EV3
- Piezas legos para la construcción del robot
- 1 Notebook para la instalación y programación del robot
- Cubo Rubik

4.1.2. Software

- Python
- Visual Studio Code.
- Windows

4.2. 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	5500
Robot EV3	Lego Mindstorms EV3		490209
Sueldo Coordinador de grupo	Ganancia mensual del Coordinador		1500000
Sueldo secretario	Sueldo mensual del secretario		400000
Sueldo programador	Sueldo mensual programador		700000

Sueldo Diseñador	Sueldo mensual diseñador		450000
Movilización	Costo del uso del microbús diario	260– 800	1040 - 3400
Total			3549109

5. Análisis – Diseño

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

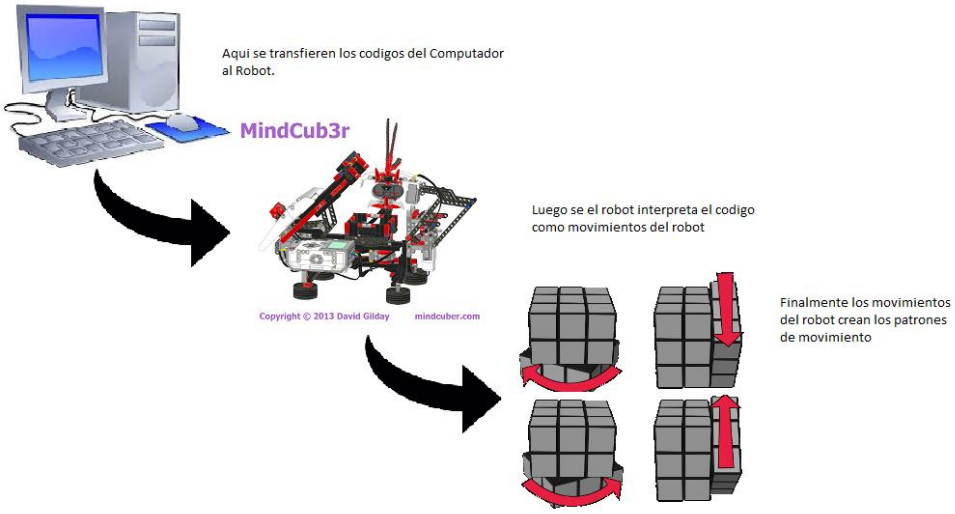
5.1.1. Requisitos Funcionales

NOMBRE	DESCRIPCIÓN DEL REQUISITO
Realizar movimientos del robot	Es la acción que realiza el robot a través de del ingreso de datos.
Ejecutar comandos a través de PuTTY	**PuTTy** es un cliente de red que sirve principalmente para iniciar una sesión remota con otra maquina o servidor.

5.1.2. Requisitos No Funcionales

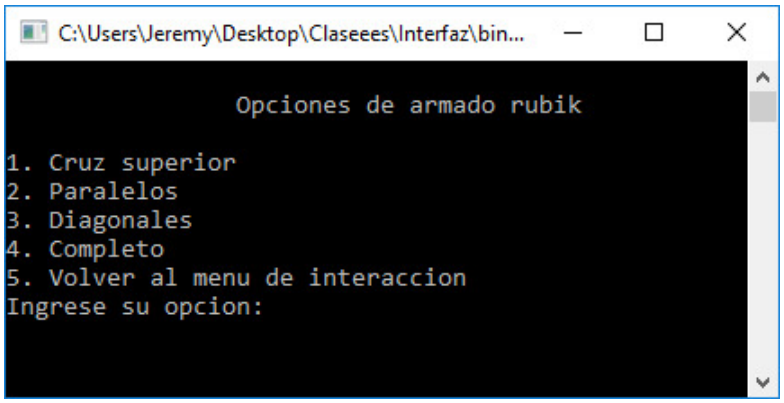
NOMBRE	DESCRIPCION DE REQUISITO
Lenguaje Python	Es una de las limitaciones del proyecto
Armar cubo Rubik	Armado del cubo con los algoritmos anteriormente realizados

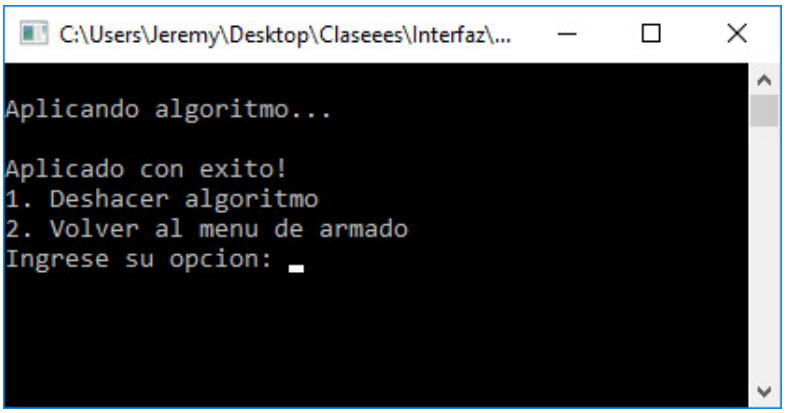
5.2. Arquitectura Propuesta (incluyendo aspectos de comunicación).



5.3. Diseño de la Interfaz Usuario

Bosquejo de la Interfaz de usuario.





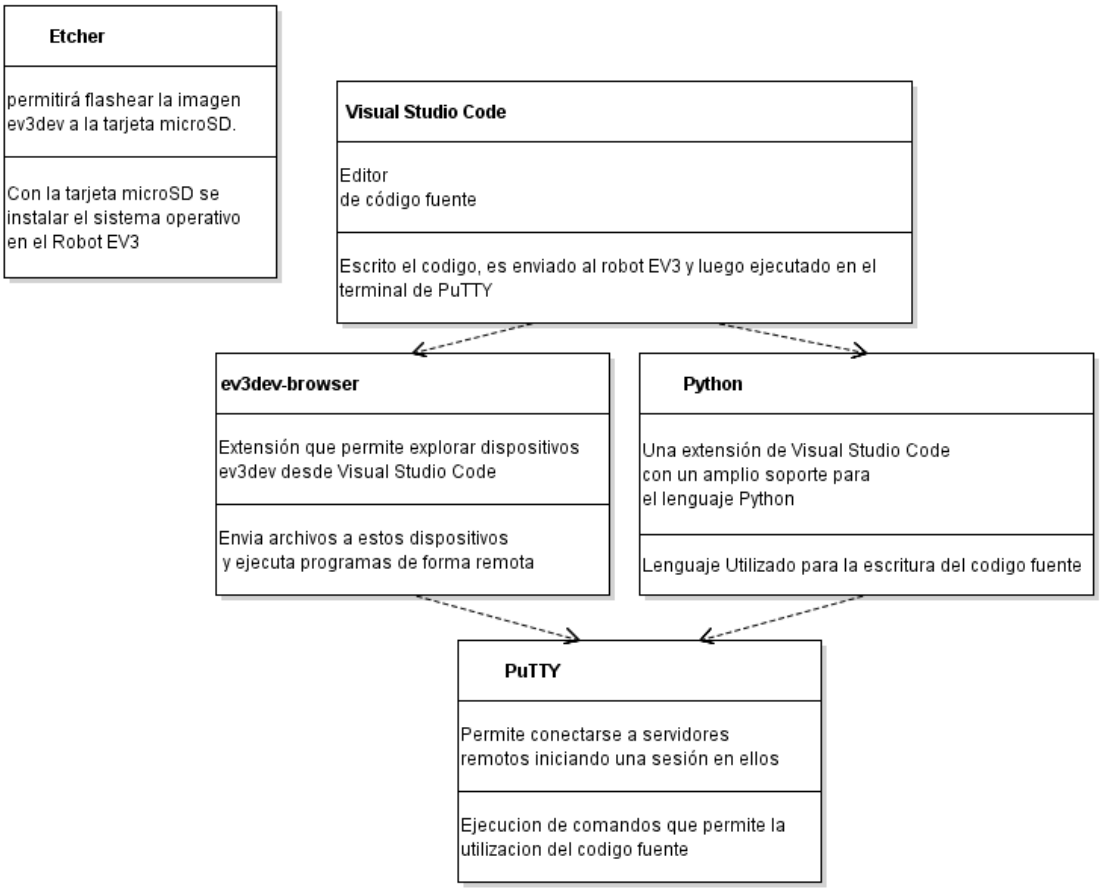
6. Implementación

6.1. Descripción de los programas implementados (entradas, salidas, procesos)

PuTTY es un cliente SSH y Telnet con el que podemos conectarnos a servidores remotos iniciando una sesión que nos permitirá ejecutar comandos. Tales comandos ejecutan el código escrito en Visual Studio Code que es un editor de código fuente, en tal editor de código usamos el lenguaje de programación python para definir una de sus funciones de movimiento, por ejemplo, el Sexy Move.

Una vez escrito el código, es enviado al robot EV3 y luego ejecutado en el terminal de PuTTY mediante el comando: python3 "Nombre_Del_Programa".py.

6.2. Diagrama de interacción entre programas



7. Resultados

7.1. Estado actual del proyecto

Nos encontramos en la realización de los algoritmos necesarios para que el robot realice movimientos como los que son el giro del cubo, girar lo en su eje y además de realizar el cambio de cara del cubo Rubik.

7.2. Problemas encontrados y soluciones propuestas

Un problema fue la conexión del EV3 a el computador, la cual no nos permitía enviar los códigos del computador a el ev3, lo cual lo solucionamos con la ayuda del docente Ricardo Valdivia el cual nos dios la solución de utilizas un cliente SSH y Telnt para poder conectar ambos servidores y poder enviar los códigos al ev3 esto fue mediante PUTTY.

Hubieron problemas con la base del robot que toma al cubo, debido a que esta era demasiado grande como para agarrar al cubo y lo solucionamos rearmando la base, se achico y se anivelo a la comodidad del robot.

7.3. Conclusiones

El robot EV3 es una poderosa herramienta de aprendizaje, hemos aprendido a usar Python sobre él, con el fin de lograr hacer algoritmos para que el robot los ejecute, por lo mismo hemos aprendimos algoritmos nuevos para la solución del cubo Rubik, a todos se nos asignó un rol dentro del equipo y al avanzar el proyecto aprendimos a trabajar en equipo y ayudarnos mutuamente, para así poder llegar a nuestro objetivo.

8. 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)

Anexo

Pseudocodigo del robot

```
#!/usr/bin/env: python3
import ev3dev.ev3 as ev3 #Importar la Biblioteca de EV3
from time import sleep


m = ev3.LargeMotor('outB') #Se Defini m como el motor principal
que movera la garra del robot
l = ev3.LargeMotor('outA') #Se Define l como el motor secundario
que movera la base del cubo


def sexyMove():
    #Cambio de cara conservando la garra arriba
    l.run_timed(time_sp = 300, speed_sp = 250)
    sleep(0.5)
    l.run_timed(time_sp = 450, speed_sp = -360)


    sleep(1)
```

```
#Mover base sin Garra(3 veces)
m.run_timed(time_sp = 500, speed_sp = 361)
sleep(1)
m.run_timed(time_sp = 500, speed_sp = 361)
sleep(1)
m.run_timed(time_sp = 500, speed_sp = 361)
sleep(1)

sleep(1)

#Devolver garra a su posicion inicial cuando esta arriba
l.run_timed(time_sp = 300, speed_sp = 250)
sleep(1)

#Cambio de cara conservando la garra arriba
l.run_timed(time_sp = 300, speed_sp = 250)
sleep(0.5)
l.run_timed(time_sp = 450, speed_sp = -360)

sleep(1)

#Devolver garra a su posicion inicial cuando esta arriba
l.run_timed(time_sp = 300, speed_sp = 250)
sleep(1)

#Mover base con Garra
m.run_timed(time_sp = 500, speed_sp = 469)
```

```
sleep(1)
m.run_timed(time_sp = 500, speed_sp = -78)

#Cambio de cara conservando la garra arriba
l.run_timed(time_sp = 300, speed_sp = 250)
sleep(0.5)
l.run_timed(time_sp = 450, speed_sp = -360)

#Mover base sin Garra
m.run_timed(time_sp = 500, speed_sp = 361)
sleep(1)

#Devolver garra a su posicion inicial cuando esta arriba
l.run_timed(time_sp = 300, speed_sp = 250)
sleep(1)

#Mover base con Garra
m.run_timed(time_sp = 500, speed_sp = 469)
sleep(1)
m.run_timed(time_sp = 500, speed_sp = -78)

sleep(2)

#Encontrar algoritmo para levantar la Garra


#Fin Busqueda de algoritmo para levantar la Garra
```

```
#Mover base sin Garra
m.run_timed(time_sp = 500, speed_sp = 361)
sleep(1)

#Cambio de cara conservando la garra arriba
l.run_timed(time_sp = 300, speed_sp = 250)
sleep(0.5)
l.run_timed(time_sp = 450, speed_sp = -360)

#Devolver garra a su posicion inicial cuando esta arriba
l.run_timed(time_sp = 300, speed_sp = 250)
sleep(1)

#Mover base con Garra(3 veces)
m.run_timed(time_sp = 500, speed_sp = 469)
sleep(1)
m.run_timed(time_sp = 500, speed_sp = -78)

sleep(1)

m.run_timed(time_sp = 500, speed_sp = 469)
sleep(1)
m.run_timed(time_sp = 500, speed_sp = -78)

sleep(1)
```

```
    m.run_timed(time_sp = 500, speed_sp = 469)
    sleep(1)
    m.run_timed(time_sp = 500, speed_sp = -78)

    #Girar la cara del cubo 3 veces y al final conservar la garra
en el cubo
    #Cambio de cara conservando la garra arriba
    l.run_timed(time_sp = 300, speed_sp = 250)
    sleep(0.5)
    l.run_timed(time_sp = 450, speed_sp = -360)

    #Devolver garra a su posicion inicial cuando esta arriba
    l.run_timed(time_sp = 300, speed_sp = 250)
    sleep(1)

    #Cambio de cara conservando la garra arriba
    l.run_timed(time_sp = 300, speed_sp = 250)
    sleep(0.5)
    l.run_timed(time_sp = 450, speed_sp = -360)

    #Devolver garra a su posicion inicial cuando esta arriba
    l.run_timed(time_sp = 300, speed_sp = 250)
    sleep(1)

    #Cambio de cara conservando la garra arriba
    l.run_timed(time_sp = 300, speed_sp = 250)
```

```
sleep(0.5)
l.run_timed(time_sp = 450, speed_sp = -360)

#Devolver garra a su posicion inicial cuando esta arriba
l.run_timed(time_sp = 300, speed_sp = 250)
sleep(1)
#Fin giro de caras

#Mover base con Garra(3 veces)
m.run_timed(time_sp = 500, speed_sp = 469)
sleep(1)
m.run_timed(time_sp = 500, speed_sp = -78)

sleep(1)

m.run_timed(time_sp = 500, speed_sp = 469)
sleep(1)
m.run_timed(time_sp = 500, speed_sp = -78)

sleep(1)

m.run_timed(time_sp = 500, speed_sp = 469)
sleep(1)
m.run_timed(time_sp = 500, speed_sp = -78)

sleep(1)
#Fin Girar base
```

```
#Encontrar algoritmo para levantar la Garra



#Fin Busqueda de algoritmo para levantar la Garra

#Mover base sin Garra
m.run_timed(time_sp = 500, speed_sp = 361)
sleep(1)

#Cambio de cara conservando la garra arriba
l.run_timed(time_sp = 300, speed_sp = 250)
sleep(0.5)
l.run_timed(time_sp = 450, speed_sp = -360)

#Devolver garra a su posicion inicial cuando esta arriba
l.run_timed(time_sp = 300, speed_sp = 250)
sleep(1)

#Fin SexyMove
```