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
Bitacora IX	Reporte semanal de tareas	25 de octubre de 2018
Bitacora 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

2. Organización del Personal

2.1. 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	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

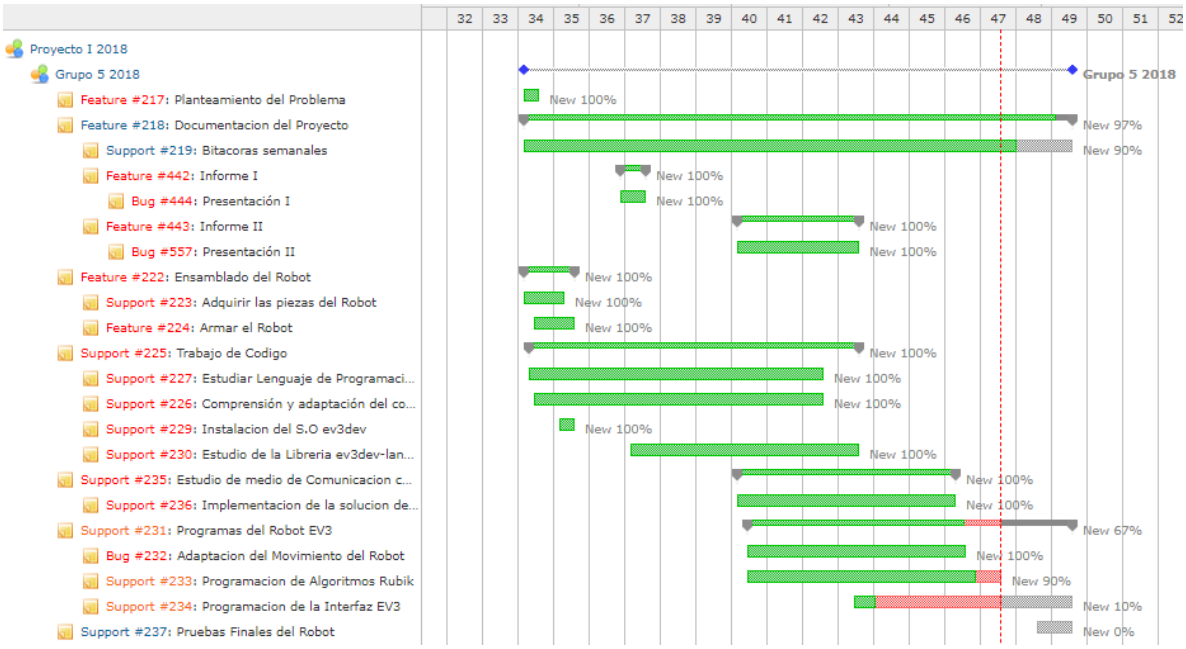
3. Planificación del proyecto

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, 22 de noviembre 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	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	260– 800	1.040 – 3.400
Total(mensual)			3.549.109
Total(semestre)			17.745.545

5. Análisis – Diseño

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

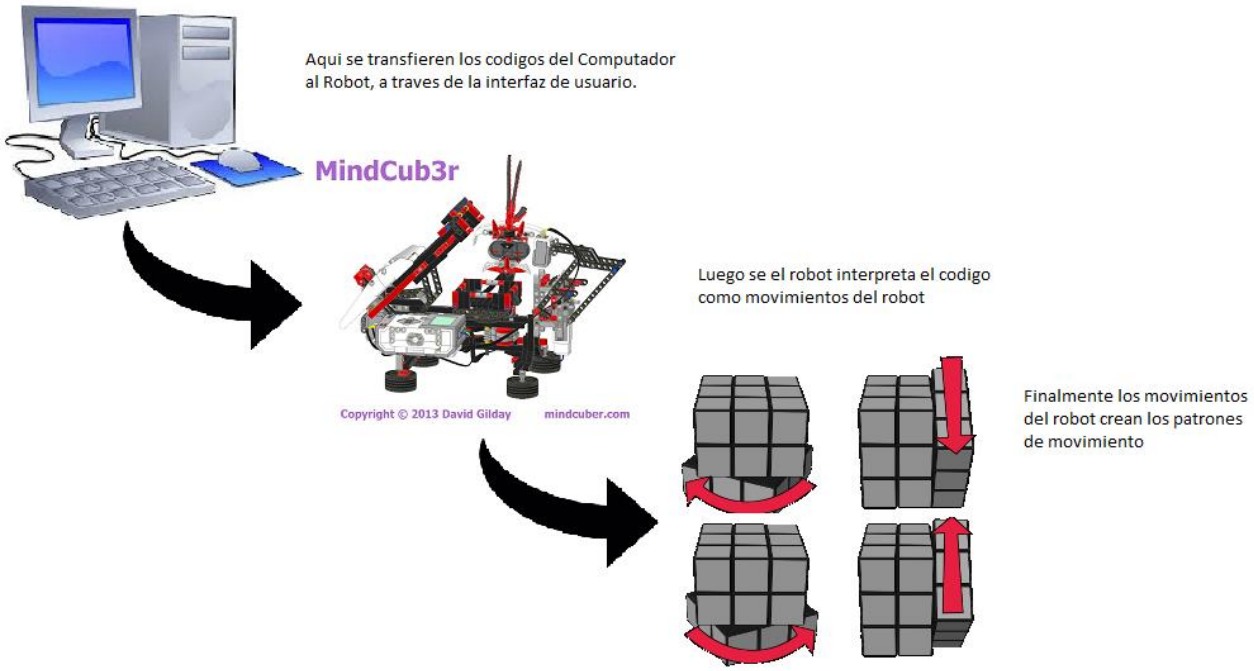
5.1.1. Requisitos Funcionales

NOMBRE	DESCRIPCIÓN DEL REQUISITO
Comunicación	Tendra un menú de opciones necesarias para realizar movimientos básicos además de los algoritmos.
El Robot	Podra armar el cubo mediante los algoritmos. algo

5.1.2. Requisitos No Funcionales

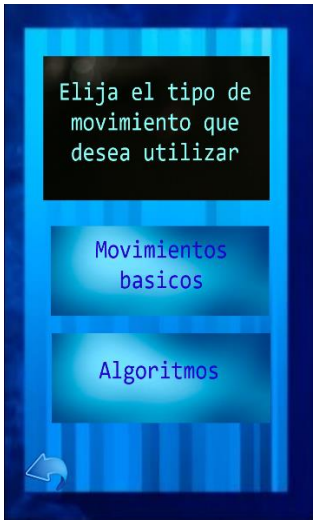
NOMBRE	DESCRIPCION DE REQUISITO
Tiempo de ejecución	Cada movimiento deberá ejecutarse en un tiempo estimado
Manual de usuario	El robot debe contar con su manual de usuario
Interfaz grafica	Debe disponer de una Interfaz Grafica

5.2. Arquitectura Propuesta (incluyendo aspectos de comunicación).



5.3. Diseño de la Interfaz Usuario

Bosquejo de la Interfaz de usuario.



Tenemos 2 opciones:
-Presionar movimientos básicos que stos son los conocidos como “L R U D B F”

-Presionar algoritmos de movimientos que estos son una combinación de movimientos básicos.

Esto nos abrirá 2 opciones

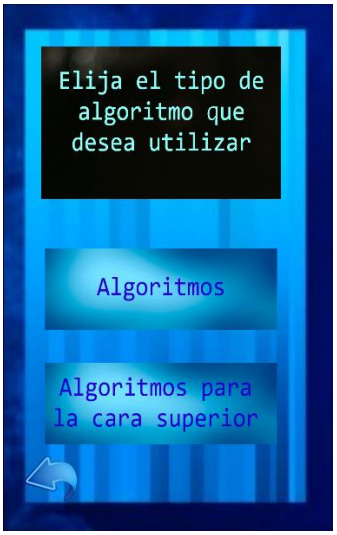
Primera opción:

Movimientos básicos

Tenemos nuevamente 2 opciones:

-Algoritmo para la cara superior.

-Cambiar las aristas o vértices.



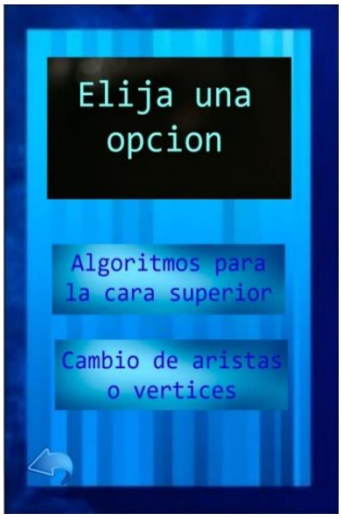
Aquí tenemos 2 opciones de nuevo:

-Algoritmos.

-Algoritmos para la cara superior.

Al presionar “Algoritmos” nos da una lista de algoritmos el cual debemos elegir y después presionar ejecutar.

Al presionar “Algoritmos para la cara superior” tendremos 2 opciones.



Cundo elegimos la opción “Algoritmos para la cara superior” nos saldrá una lista de algoritmos para armar completa la cara superior del cubo.

Cuando elegimos la opción “Cambio de esquinas y aristas ” nos da algoritmos para cambiar las esquinas y aristas de la cara superior del cubo.

6. Implementación

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

Primeramente comenzaremos 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').
Funcion 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)
Funcion 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’)
Funcion 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’)
Funcion 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().

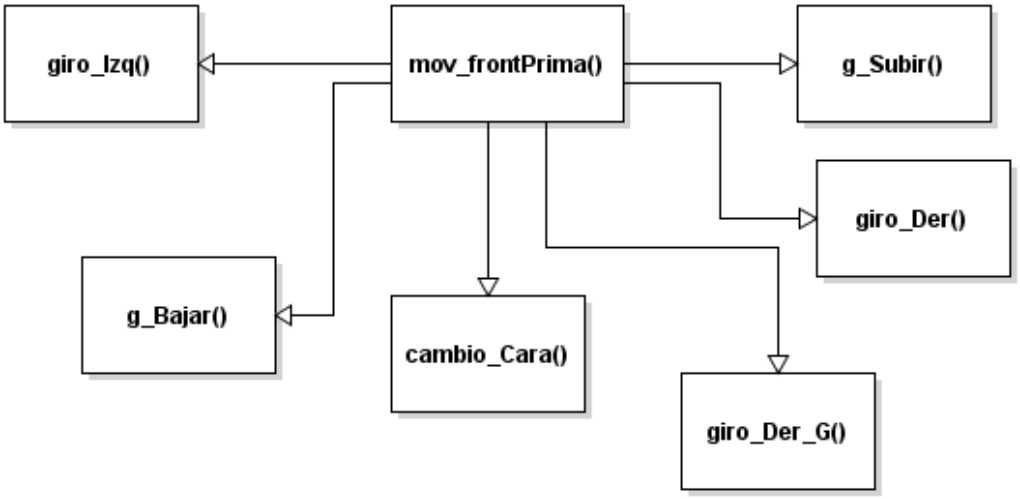
Movimientos complejos: Algoritmos utilizados por gente experimentada que se utilizan para resolver el cubo de una forma más eficaz y eficiente. Estos movimientos son compuestos de los movimientos básicos.

Movimientos complejos

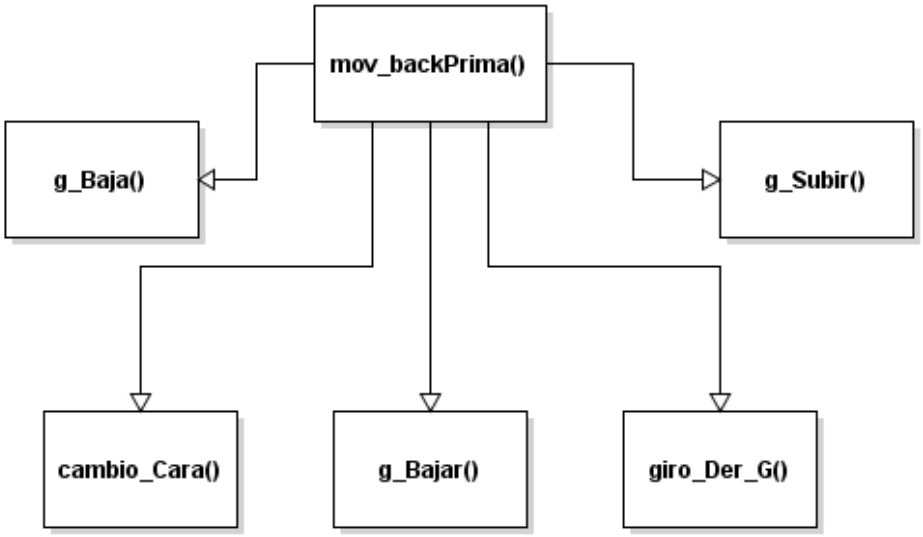
def sexyMove(): Función que permite el movimiento más común llamado sexy move.
Funcion compuesta por: def mov_R(), def mov_U(), def mov_RPrima(),mov_UPrima().

6.2. Diagrama de interacción entre programas

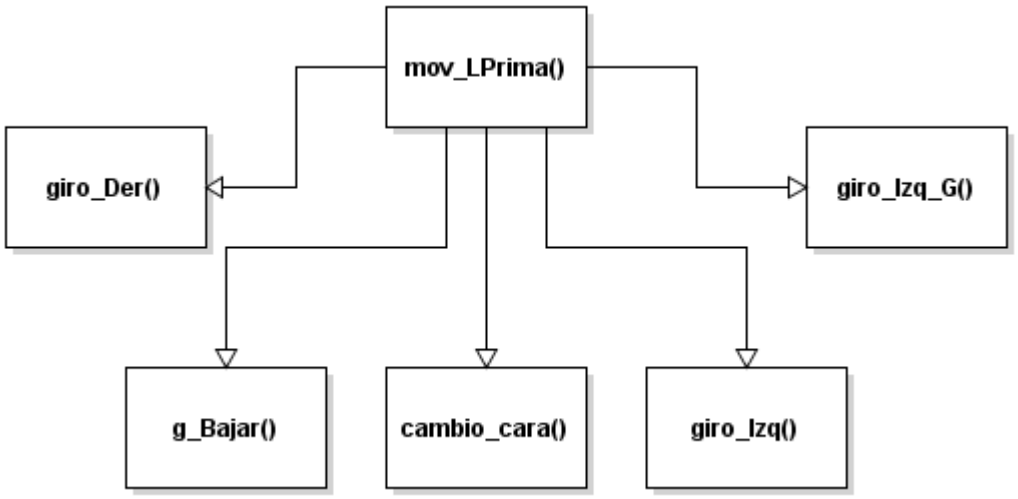
Mov_frontPrima(): (Movimiento Frontal Inverso)



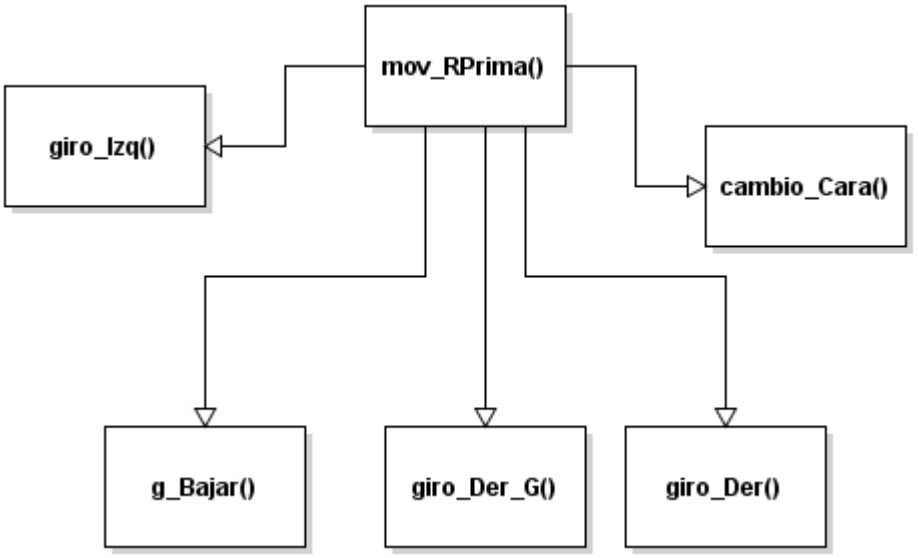
Mov_backPrima(): (Movimiento hacia atrás inverso)



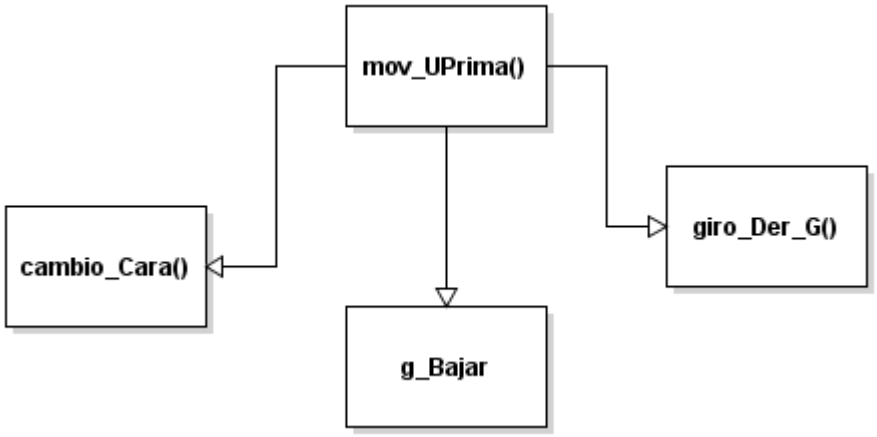
Mov_LPrima(): (Movimiento Izquierda Inverso)



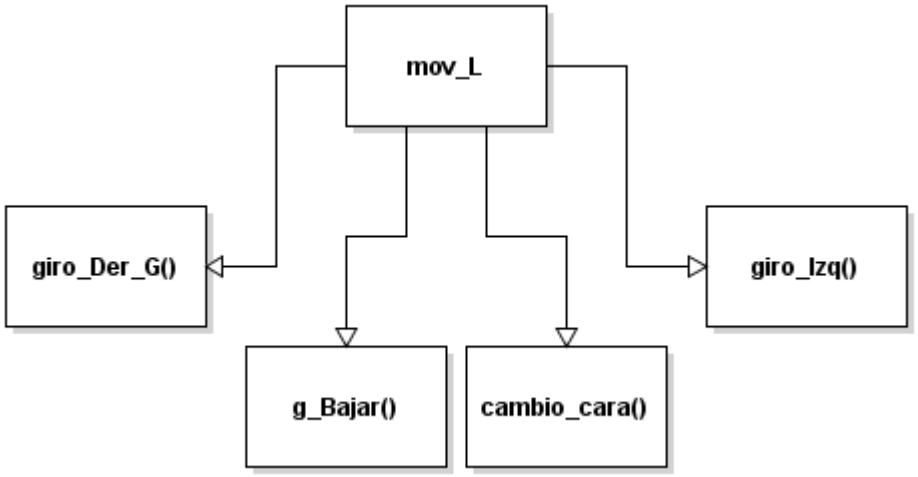
Mov_RPrima(): (Movimiento Derecha Inverso)



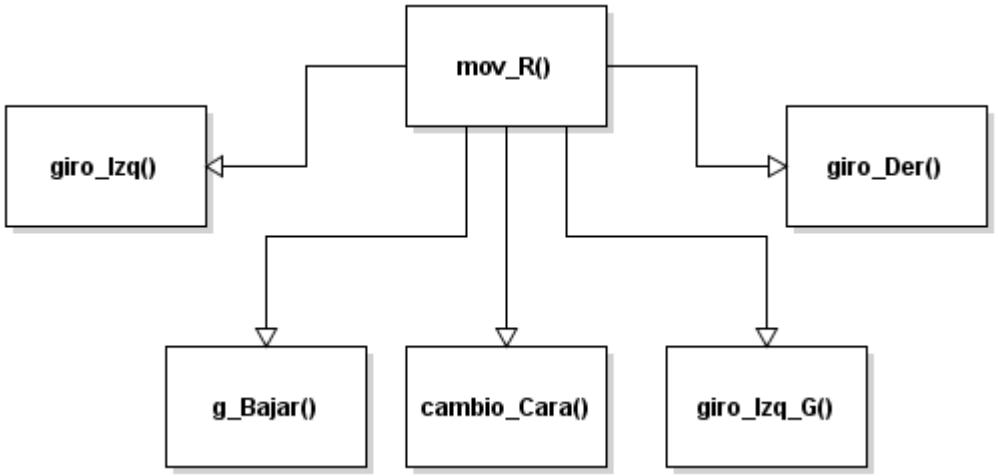
Mov_UPrima(): (Movimiento Cara Superior Inversa)



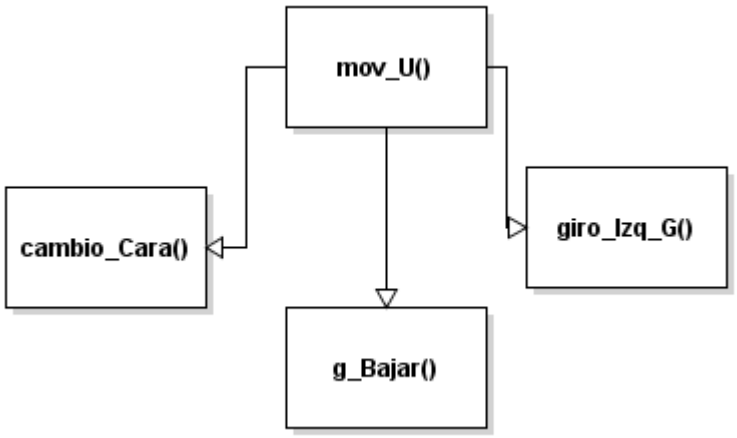
Mov_L(): (Movimiento Izquierda)



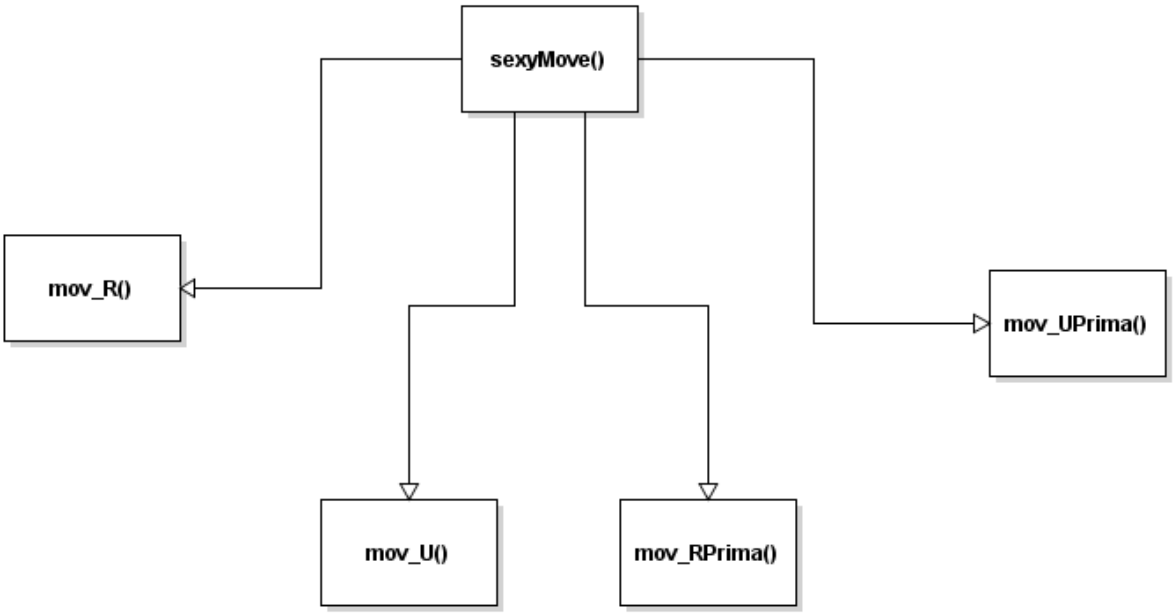
Mov_R(): (Movimiento Derecha)



Mov_U(): (Movimiento Cara superior)



SexyMove(): (Movimiento Sexy)



7. Pruebas

7.1. Descripción de las pruebas realizadas

- Prueba de conexión: se probó la conexión entre el robot y el computador
- Prueba de armado: Se probo mediante códigos al armado del robot para que este no se rompa al ser utilizado.
- Prueba de giros: Se probaron los giros para cuadrar la cantidad de grado por cada giro.
- Pruebas básicas: aquí se prueban los movimientos básicos como son: “L”,”U”,”R”,”B”.”D”,”F”.
- Prueba sexy move: Se probo el sexy move ya que este esta formado por diferentes movimientos básicos.
- Prueba de algoritmos implementados: se probó la combinación de movimientos básicos para ver si los giros cuadraban y no había problema con estos.

7.2. Resultados de las pruebas

Nombre	*Resultado*
Prueba de conexión	La conexión con el robot estuvo bien, el único problema fue que el robot no nos recibía el código
Prueba de armado	No tuvimos problemas con el armado del cubo, con la base si tuvimos un pequeño problema
Prueba de giros	Los giros fueron calibrados para que cada vez que gire este cambie solo una vez la cara
Pruebas básicas	Las pruebas de los movimientos básicos estuvieron bien no hubo ningún tipo de problema
Prueba de sexy move	Ocurría una descalibración al momento de realizar mas de un movimiento básico.
Prueba de algoritmos	El robot realizo bien los algoritmos, el único error fue que se des calibraron los giros, a veces giraban de más o les faltaba girar.

8. Resultados

8.1. Estado final 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.

8.2. Conclusiones

Para finalizar podrías decir que con esto hemos aprendido a usar el lenguaje de programación Python, con algunas librerías en específico, aprendimos a crear interfaz 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.

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 a las personas

8.3. Trabajo a futuro

9. A futuro queremos mejorar nuestra interfaz grafica actualizandola, además de agregar más algoritmos y movimientos básicos como son : “E”, “E’ ”,”S”, “S’ ”, “M”, “M’ ”, ”LW” y “LW’ “, y agregar el sensor de movimiento para que el robot sea capaz de armar el cubo autónomamente, además de convertir los codigos mas eficientes para que los movimientos se realicen mas velozmente, también queremos buscar la manera de utilizar o 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).

Anexo

Codigo 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()
```