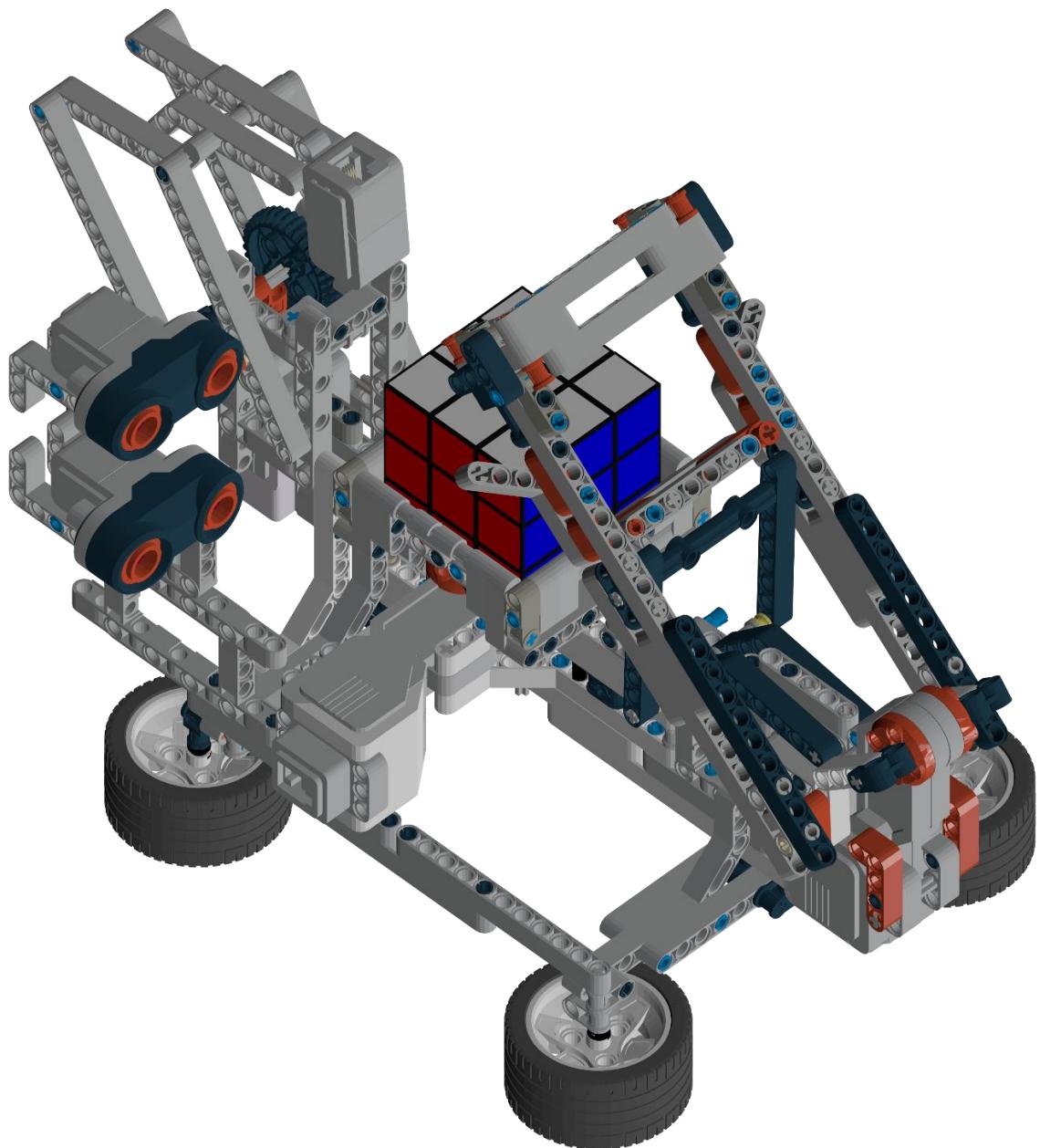


Intel·ligència artificial aplicada al cub de Rubik



Autor: Max Martínez Ruts

Tutor: Xavi Rio

28/11/2016

Índex

Índex.....	1
1. Introducció	3
1.1. Objectius	3
1.2. Experiència amb la programació.....	3
1.3. Perquè programació?.....	4
1.4. Estructura del treball.....	5
2. Cub de Rubik	6
2.1. Història	6
2.2. Funcionament	6
3. El robot.....	7
3.1. Construcció del robot.....	7
3.2. Model 3D.....	8
3.2.1. Procés de modelació	8
3.2.2. Renderització de models.....	9
3.2.3. Animació model 3D	11
3.3. Components	12
3.3.1. Bloc EV3.....	12
3.3.2. Sensor de color.....	14
3.3.3. Sensor de distància.....	18
3.3.4. Motor gran	19
3.3.5. Motor mitjà	21
3.4. Parts mòbils.....	22
3.4.1.1. Base Giratòria.....	22
3.4.1.2. Estructura del sensor de color.....	23
3.4.1.3. Pinça	24
3.4.2. Moviment del sensor de color	25
3.4.3. Gir del cub eix vertical.....	26
3.4.4. Gir cub eix profunditat	27
3.4.5. Gir cara	28
3.4.6. Simulació de motors al GeoGebra.....	29
3.5. Sistema intuïtiu de moviments	31
4. Programació	34
4.1. Software	35
4.2. Notació	35
4.3. Solució del cub de rubik	37
4.3.1. Lectura del cub	37

4.3.1.1.	Detecció del color.....	37
4.3.1.2.	Construcció d'un cub de Rubik numèric.....	37
4.3.1.3.	Ordenació de colors	38
4.3.2.	Girs	39
4.3.2.1.	Girs de cara.....	39
4.3.2.2.	Girs del cub.....	40
4.3.3.	Solució	42
4.3.3.1.	Cross (Creu)	42
4.3.3.2.	F2L (First Two Layers)	43
4.3.3.3.	OLL (Orientation Last Layer).....	44
4.3.3.4.	PLL (Permutation Last Layer).....	45
4.4.	Avantatges i inconvenients del mètode escollit	46
5.	Conclusió	47
5.1.	Reflexió sobre la intel·ligència artificial.....	48
6.	Referències.....	51
6.1.	Webgrafia	51
6.2.	Bibliografia	51
7.	Annex.....	52
7.1.	Lectura_Cub	52
7.2.	Moviments	55
7.3.	Solució	59

1. INTRODUCCIÓ

El treball consisteix en la resolució robotitzada d'un cub de Rubik. Tot i que la part escrita del treball es centrarà en el funcionament del robot (funcionament de motors i sensors, moviments necessaris per a resoldre el cub, etc), el gruix del treball ha estat en la programació del robot i no en la seva elaboració o construcció. Com el títol esmenta, el mètode per el qual el robot resol el cub de Rubik és aplicant una intel·ligència artificial¹ al codi de programació.

1.1. OBJECTIUS

L'objectiu principal i el gran pes del treball ha estat la programació del robot. No obstant, he hagut de realitzar altres feines per a fer el treball, que també m'han servit per aprendre més i per poder fer-ne un ús en un futur.

Una d'aquestes feines ha estat la construcció del robot de Lego, que no formava part de l'objectiu però era necessari per a la realització del treball, per a aconseguir un robot físic i no només un codi de programació. Un altre món en el qual m'he introduït, tot i que ja tenia una mica d'experiència és en la modelació i la animació en 3D que he hagut de fer per tal de poder representar el robot visualment per a facilitar-ne la seva comprensió.

Però com he dit, l'objectiu principal ha estat el de la programació, ja que sempre m'ha fascinat el poder d'aquesta, la gran quantitat d'implementacions que té i que pot tenir en un futur.

1.2. EXPERIÈNCIA AMB LA PROGRAMACIÓ

Vaig començar a programar fa tres anys, primer a nivell educatiu, amb llenguatges de programació visual com el Scratch² i App Inventor³. Després vaig començar a programar PHP⁴ i alguna petita pagina web que em va ensenyar el meu pare. Poc a poc em vaig anar endinsant en la programació i vaig fer un petit videojoc amb Adobe Flash⁵ utilitzant Actionscript⁶. Més endavant vaig començar

¹ **Intel·ligència artificial:** desenvolupament d'algoritmes que permet a una màquina prendre decisions intel·ligents o bé comportar-se com si tingués una intel·ligència semblant a la humana.

² **Scratch:** eina gratuïta de programació per a la introducció a la programació.

³ **App Inventor:** entorn integrat de desenvolupament que permet crear aplicacions mòbils.

⁴ **PHP:** llenguatge de programació que s'utilitza per a generar pàgines web de forma dinàmica.

⁵ **Adobe Flash:** plataforma de software per a la producció d'animacions, videojocs, i aplicacions.

⁶ **Actionscript:** llenguatge de programació que s'utilitza per a les aplicacions d'Adobe Flash.

a utilitzar Unity⁷ per programar videojocs, la qual cosa em va fer adonar-me de la gran poder de la programació, la gran quantitat de possibilitats que ofereix.

1.3. PERQUÈ PROGRAMACIÓ?

La programació en la robòtica no és només l'ordre que es dóna a una màquina, sinó el llenguatge que proporciona la interacció entre la màquina i l'exterior. Entenem per programació el fet d'ordenar a un robot a moure tal motor, o a visualitzar per pantalla una pàgina web, o un videojoc, però la programació pot introduir-se en altres temes quotidians i subtils.

El fet que les tecnologies s'endinsin a les nostres vides, permet que puguem obtenir dades humans en tot moment. Per posar un exemple, quan cerquem per Internet la paraula “Audi”, estem donant una informació a una base de dades; estem interessats per els cotxes, llavors aquesta informació s'aprofita, i els anuncis que apareixen a les pàgines web passen a ser anuncis de cotxes, de motors, etc. Aquest fenomen es coneix com el Big Data, el procés de convertir simples dades en un conjunt d'informació útil a través d'algoritmes. I això és gràcies a la programació.

El que he volgut fer al meu treball és un petit exemple de Big Data. He volgut que a partir d'unes dades com poden ser els colors d'un cub, a través d'una programació d'algoritmes, el robot sigui capaç de convertir aquestes dades en un conjunt d'informació útil, en aquest cas per resoldre el cub. Aquest procés també està molt relacionat amb el tema de la intel·ligència artificial.

És pel meu interès pel tema de la intel·ligència artificial per el qual he volgut profundir el meu treball en la programació.

Cal remarcar que l'objectiu en cap moment ha estat aconseguir un aparell que pugui resoldre un cub de Rubik, sinó entendre com funciona aquesta programació, com convertir dades en informació útil.

Desafortunadament, dedicar el treball a explicar la programació d'un robot pot arribar a ser difícil de representar en un full i complicat, tant d'explicar com d'entendre. És per això que he hagut de dedicar gairebé tota la part escrita del treball al funcionament del robot i dels seus components i una petita part només a explicar per sobre com funciona la programació, evidenciant molt codi de

⁷ **Unity:** plataforma de software per a la producció de videojocs.

programació i passant per sobre d'aspectes importants però que com dic, són realment difícils d'explicar en un full imprès.

1.4. ESTRUCTURA DEL TREBALL

He estructurat el treball en diferents blocs segons l'àmbit que tracten. El primer bloc, dedicat al cub de Rubik és només per introduir i explicar els conceptes bàsics del cub de Rubik. No té cap informació relacionada amb el meu treball però és necessari per a la comprensió del qual.

El segon bloc és el que conté el gruix del treball i és el que explica com funciona el robot. S'explica el funcionament dels sensors i dels motors, i el funcionament físic per el qual el robot és capaç de girar el cub i les seves cares.

Finalment, el tercer bloc està dedicat a la programació, al procés per el qual el robot obté unes dades, les interpreta, i finalment és capaç d'aplicar aquesta interpretació al moviment dels motors per a que puguin resoldre el cub.

2. CUB DE RUBIK

2.1. HISTÒRIA

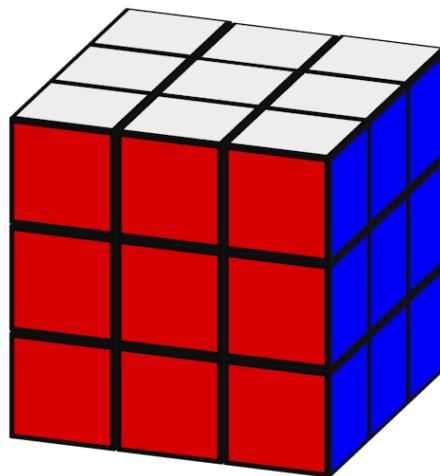
El cub de Rubik és un trencaclosques tridimensional inventat al 1974 per l'escultor i arquitecte húngar Erno Rubik. Fins a l'actualitat, el cub de Rubik ha estat la jocuina més venuda, amb una quantitat de 350 milions d'unitats venudes. El cub de Rubik va guanyar el premi a “Millor jocuina de l'any” el mateix any de ser creat, i com a curiositat, el seu creador no el sabia resoldre.

2.2. FUNCIONAMENT

El cub de Rubik clàssic consta de sis cares amb colors uniformes (blanc, vermell, blau, verd, taronja i groc), i un sistema d'eixos permet girar cada cara independentment, de manera que els colors es barregen.

El cub de Rubik utilitzat és de tipus “3x3x3”, com el que es veu a la imatge, i es compon de 8 vèrtexs, 12 arestes i 6 centres.

Qualsevol moviment que s'aplica al cub, és a dir, la rotació d'una cara, canvia de posició 4 vèrtexs i 4 arestes, però mai el centre de la cara.

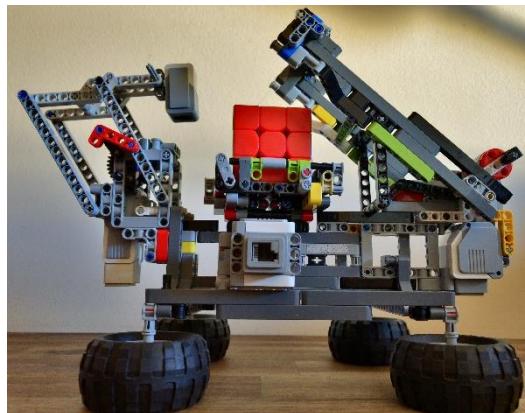


3. EL ROBOT

3.1. CONSTRUCCIÓ DEL ROBOT

El procés de construcció del robot ha estat força senzill. Com he mencionat anteriorment, l'objectiu i el pes del treball ha estat el fet de programar el robot, no de construir-lo.

Per triar el model vaig estar buscant i analitzant alguns models de gent que havia construït un robot per resoldre el cub de Rubik amb Lego. Després d'analitzar-los vaig optar per el model amb el mínim de motors necessaris, i vaig fer una comanda per Internet de les peces necessàries per a la seva construcció. L'institut em va proporcionar els components electrònics necessaris per a la resolució del cub, i vaig poder començar a construir-lo. Finalment vaig fer unes modificacions, per augmentar la seva precisió i augmentar les seves funcions.



3.2. MODEL 3D

El propòsit de la construcció d'un model 3D ha estat poder representar el robot virtualment. D'aquesta manera he pogut fer una animació per representar els moviments del robot i he pogut documentar la informació amb imatges obtingudes de renders⁸ del robot que he modelat

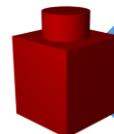
3.2.1. Procés de modelació

Vaig començar a modelar el robot amb un programa anomenat LDD (Lego Digital Design), que permet la modelació en 3D de peces Lego i exportació en format .obj⁹, però al acabar de modelar el robot, em vaig adornar que al exportar el model les peces dels components EV3 no s'exportaven, així que vaig haver de canviar d'editor i vaig fer una recerca. Després d'investigar una mica vaig trobar un programa anomenat LCDCAD, que permet la modelació en 3D de peces Lego i l'exportació en format .ldr¹⁰. Aquest format sí que era compatible amb totes les peces EV3.

Un cop acabada la modelació volia importar el model a Blender, un programa de modelació 3D avançat. Però Blender no pot importar formats .ldr, així que després de cercar una mica vaig trobar un programa anomenat LEOCAD, que permet obrir arxius .ldr i exportar-los com a .obj (format compatible amb Blender), de manera que podia modelar amb .ldr, exportar a .obj sense problemes de compatibilitat i finalment importar aquest model .obj des de Blender i allà editar-lo, animar-lo o renderitzar.



LDCAD
Construcció del
model (.ldr)



LEOCAD
Exportació a .obj



BLENDER
Importació del
model .obj

⁸ Render: Generació d'una imatge d'un model 3D mitjançant el càlcul d'il·luminació

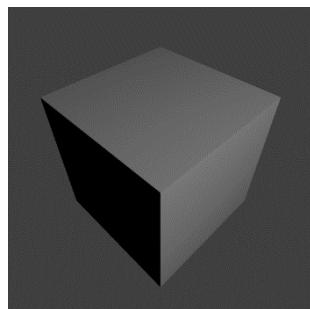
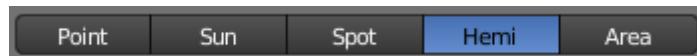
⁹ .obj: Format que representa geometries 3D

¹⁰ .ldr: Format que representa geometries 3D que es fa servir per representar peces de LEGO en 3D

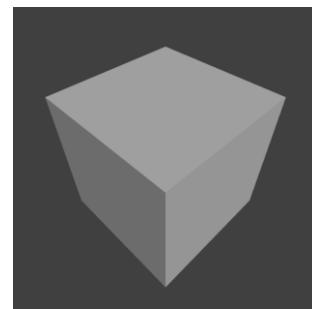
3.2.2. Renderització de models

Els editors 3D més avançats, integren un software per generar imatges del model a partir dels càlculs de paràmetres d'il·luminació i visualització anomenats paràmetres de renderització. Aquests són alguns dels paràmetres amb els quals he renderitzat totes les imatges del model.

Tipus d'il·luminació: Blender ofereix cinc tipus diferents d'il·luminació. La il·luminació predeterminada és la llum puntual, que surt d'un punt cap a totes les direccions, com per exemple la llum d'una bombeta. La il·luminació que he triat jo és la hemisfèrica, una llum que s'emet en una sola direcció, com la llum solar, però que no genera ombres, per tant permet la visualització de l'objecte sencer.

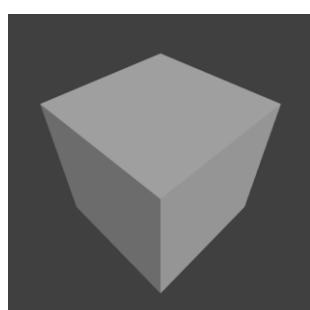


Il·luminació puntual

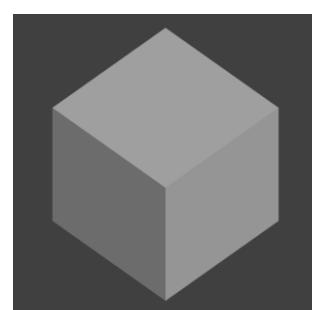


Il·luminació hemisfèrica

Tipus de vista: La vista predeterminada que ofereix Blender és la vista en perspectiva, que dóna profunditat a l'objecte i és més real, però he triat la vista ortogràfica, una vista que no té en compte la profunditat, per tant crea imatges molt més simples, amb més paral·leles, de manera que facilita la visualització de l'objecte.

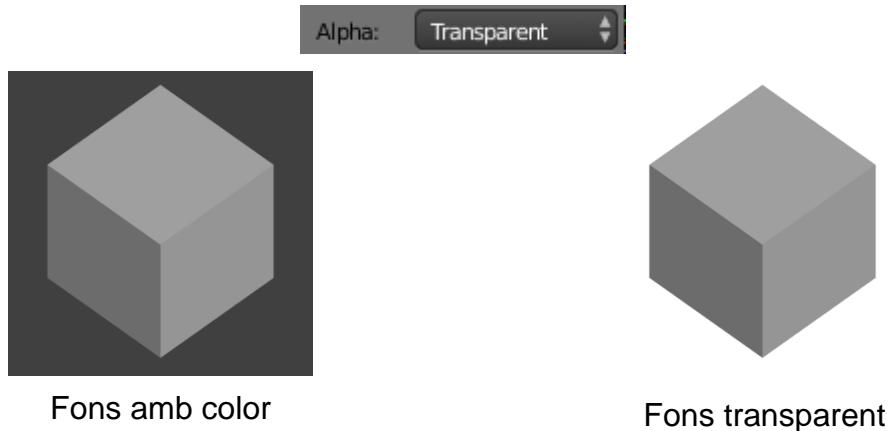


Vista en perspectiva

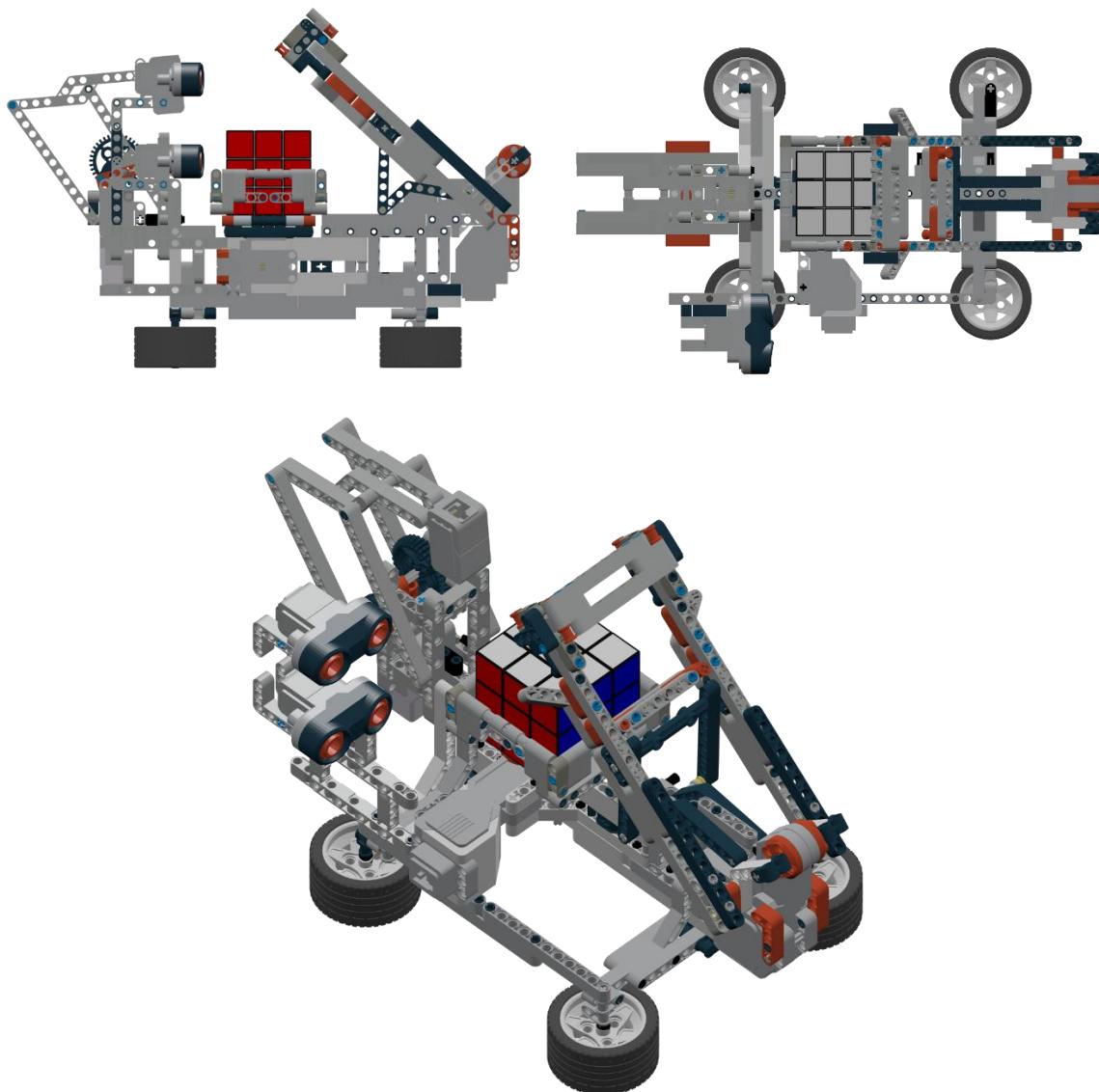


Vista ortogràfica

Tipus de fons: Per últim, Blender ofereix la opció de produir imatges .png amb fons transparents, opció que he aprofitat per obtenir unes imatges més senzilles.



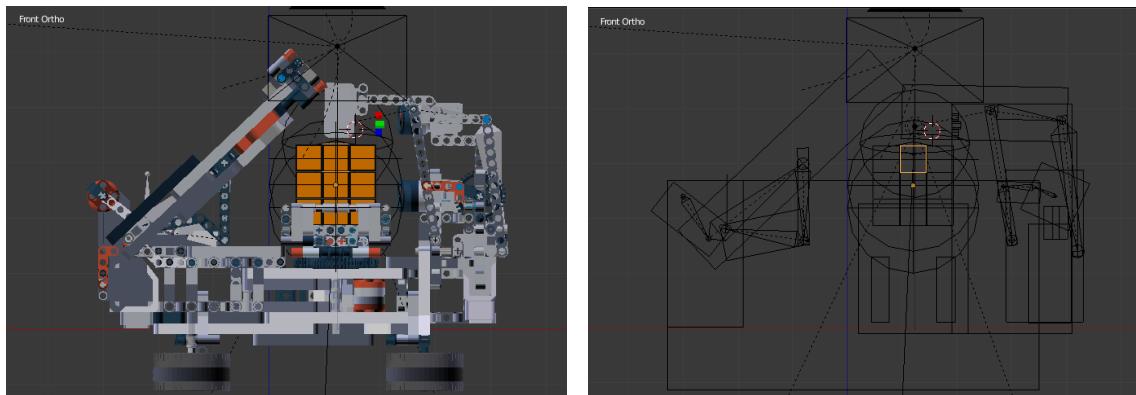
A continuació es mostren uns renders del model:



3.2.3. Animació model 3D

Vaig pensar que, a part d'obtenir imatges del robot, estaria bé fer una animació per tal de representar el moviment del robot i el funcionament dels seus components a la presentació oral d'una manera molt més visual i fàcil d'entendre. És per això que vaig mirar un curs per a la animació de Blender i vaig començar a animar el model 3D.

Per fer-ho, primer de tot vaig crear un esquelet del model. Un esquelet d'un model 3D funciona de la mateixa manera que un esquelet humà. En els éssers humans, les parts del cos i els músculs s'associen i depenen de l'articulació de d'esquelet. De la mateixa manera, al model, un grup de peces de Lego s'adjunten a un "os" de l'esquelet, i el moviment d'aquest proporciona també el moviment de tot el conjunt de peces adjuntades. D'aquesta manera es pot animar el moviment d'aquestes peces només amb el moviment d'un os i no peça per peça. A continuació es mostra una imatge a Blender del model (esquerra) i de l'esquelet del model (dreta).



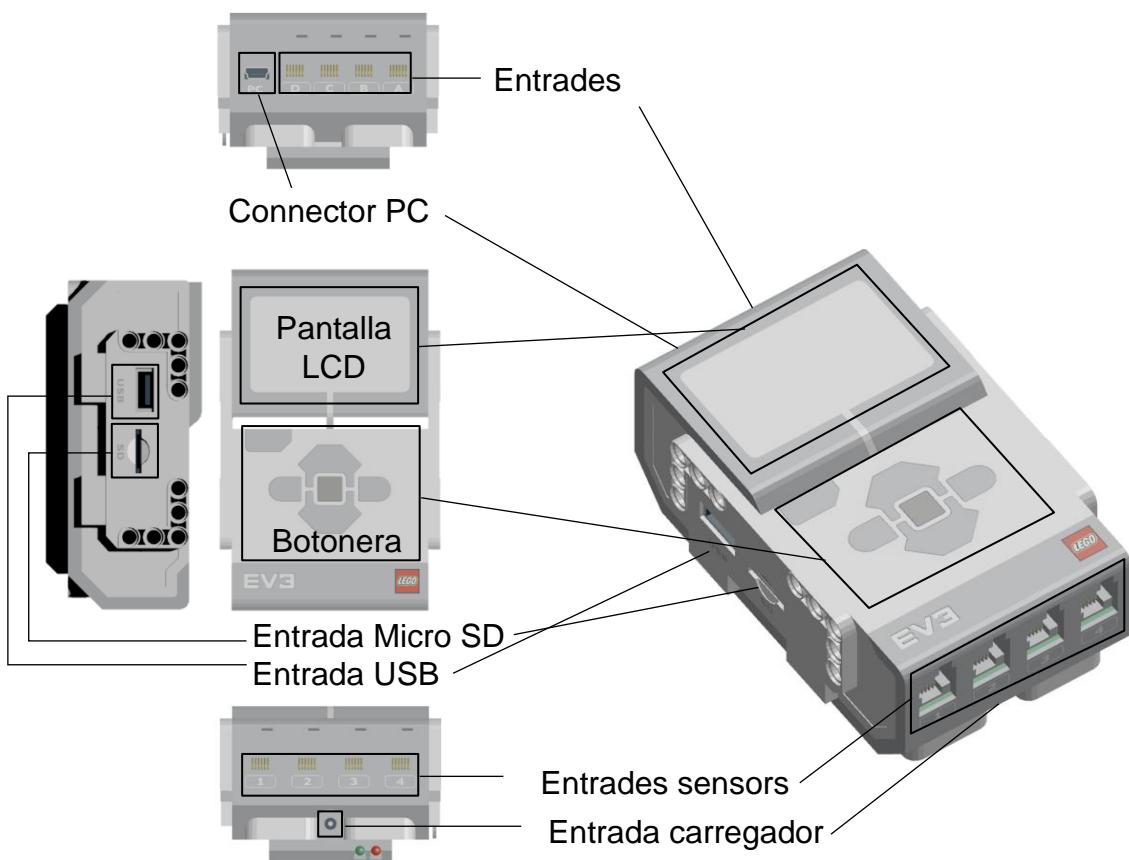
Per tant, amb l'esquelet acabat, vaig poder animar tot el model sense gaires dificultats i amb comoditat. Posteriorment vaig aplicar els mateixos paràmetres de renderització que vaig aplicar a les imatges.

Amb els paràmetres d'animació inicial, cada imatge tardava trenta segons a renderitzar-se i cada segon contenia vint-i-quatre imatges, la qual cosa feia que tardés dotze minuts per renderitzar cada segon d'animació. Vaig investigar una mica per Internet maneres de reduir aquest temps sense reduir la qualitat de l'animació i al final vaig poder optimitzar la renderització a cinc segons per imatge, per tant 2 minuts per segon d'animació, un temps força més agradable.

3.3. COMPONENTS

3.3.1. Bloc EV3

Esquema



Definició

L'EV3 és la tercera generació dels microcontroladors que permeten programar robots de Lego. Aquests microcontroladors van sorgir amb la idea de beneficiar-se'n d'un ús industrial, de control i mesura, però posteriorment Lego va col·laborar amb una empresa de software (LabVIEW), que va permetre als usuaris inexperts fer ús d'aquesta eina.

El microcontrolador consta d'una carcassa de Lego per facilitar la seva muntura, una pantalla LCD per mostrar informació, una botonera, un carregador per subministrar energia, unes entrades per als sensors, unes sortides per als motors i unes connexions de LEGO per muntar el bloc on sigui necessari.

Característiques

- Processador ARM 9 basat en el sistema operatiu Linux
- Quatre entrades per analitzar la informació obtinguda pels sensors a 1000 mostres/segon
- Quatre sortides per executar els motors
- Memòria externa de 16 MB
- Memòria RAM de 64 MB
- Lector de targetes Mini SDHC de 32 MB Botonera il·luminada
- Pantalla de resolució 178x128 que permet la visualització de gràfics i anàlisi d'informació dels sensors
- Altaveu
- Comunicació amb PC per USB, Wi-Fi o Bluetooth
- Alimentat per 6 piles AA o una bateria de 2050 mAh recarregable
- Preu de 189,99 \$

Codi del component utilitzat a la programació

- Interacció amb els botons

Pot executar un codi quan es pressiona la tecla indicada s'executa el següent codi:

```
if(Button."Botó pressionat".isDown()) {  
    "Codi que es vol executar"  
}
```

I per parar el codi fins que es pressiona la tecla desitjada:

```
while(Button.ESCAPE.isUp()) {  
}
```

- Imprimir per pantalla

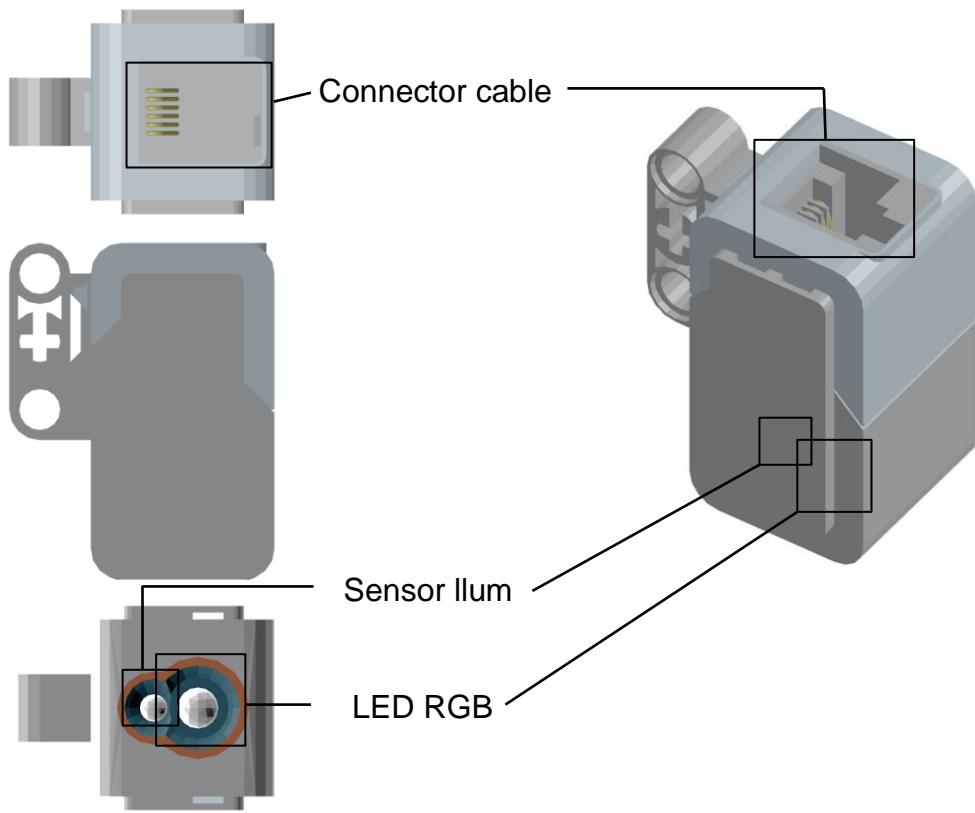
Durant el procés de programació és molt útil visualitzar alguns valors per la pantalla com per exemple valors dels sensors o altres informacions.

El següent codi permet escriure a la pantalla el text desitjat sense salt de línia

```
System.out.print("Text per a imprimir a la pantalla");
```

3.3.2. Sensor de color

Esquema



Definició

El sensor de color EV3 és un sensor capaç d'obtenir valors RGB dels objectes que es troba al davant. També pot servir com a sensor de lluminositat ja que, com s'explicarà a continuació, el sensor consta d'un LED RGB i un sensor de lluminositat. El sensor inclou un software per detectar colors bàsics automàticament, però s'ha utilitzat, ja que aquest sistema no inclou tots els colors del Cub de Rubik i no és precís.

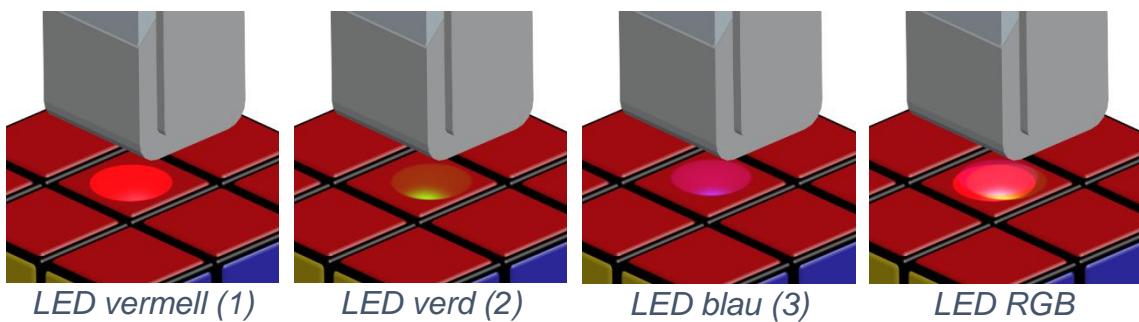
Característiques

- Preu: 39,99 \$
- Freqüència de mostra: 1 kHz (1000 mostres per segon)
- Inclou software per detectar colors bàsics automàticament
- Mesura els valors RGB tant en ambients foscos com clars

Funcionament

El sensor té integrat tres LED; un vermell, un verd i un blau (RGB). La llum d'aquests LEDs, en reflectir-se en l'objecte, torna a arribar al sensor. Aquest sensor calcula la quantitat de llum que ha arribat.

Un objecte vermell, el veiem vermell perquè absorbeix la llum de tots els colors menys la vermel·la, de manera que aquesta es reflecteix; per tant, si il·luminem un objecte vermell amb llum vermel·la es reflecteix la llum (1), en canvi si ho fem amb llum verda o blava no ho fa tant (2 i 3).



Per tant, el sensor de llum pot detectar la quantitat de vermell de l'objecte, ja que si detecta molta llum quan el LED vermell està encès, vol dir que l'objecte és força vermell. El mateix fa amb el verd i el blau, per tant el sensor obté el valor RGB de la superfície que té al davant.

Cal dir, que amb l'escala RGB, es pot representar qualsevol color, per tant aquest sensor pot detectar qualsevol color a partir de la seva quantitat de vermell, de verd i de blau.

Codi del component utilitzat a la programació

- Programació del sensor

Es crea la variable per al port “S3” (port on es connecta el sensor de color) i a continuació adjuntem els valors que s'obtenen d'aquest port a la variable del sensor de color.

A continuació s'indica el tipus de “mostra_color[]”, un array que posteriorment emmagatzemarà els valors RGB.

```
Port s3 = LocalEV3.get().getPort("S3");  
color_sensor = new EV3ColorSensor(s3);  
conjunt_color = color_sensor.getRGBMode();  
mostra_color = new float[conjunt_color.sampleSize()];
```

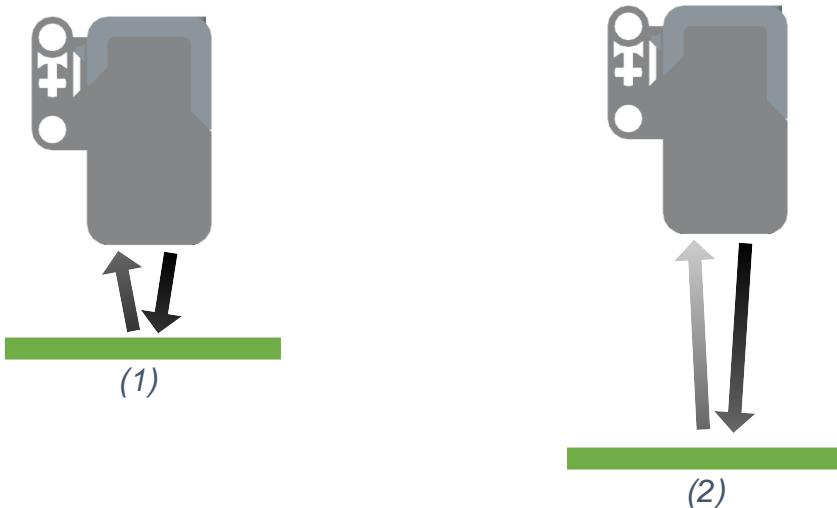
- Adjuntament de valors a les variables

Finalment s'adjunten els valors RGB a l'array mostra_color, de manera que mostra_color[0] = vermell, mostra_color[1] = verd i mostra_color [2] = blau.

```
conjunt_color.fetchSample(mostra_color, 0);
```

- Correcció d'errors

Un cop creades les 3 variables, vaig adornar-me que els valors RGB variaven molt dependent de la distància sensor-superfície; en el cas (1), el sensor està a prop de la superfície i la quantitat de llum que hi arriba és molt alta, per tant els valors RGB es disparen per sobre dels 255 (quantitat màxima), en canvi en el cas (2), el sensor està allunyat i la quantitat de llum qui hi arriba és molt menor.



Per tant, vaig optar per trobar la proporció verd-vermell-blau que tenia cada superfície, ja que el percentatge de color no varia segons la distància sensor-superfície. Els percentatges vaig calcular-los de la següent manera:

$$\text{Vermell \%} = \frac{\text{Vermell}}{\text{Vermell} + \text{Verd} + \text{Blau}} \cdot 100$$

```
float r = mostra_color[0] / (mostra_color[0] + mostra_color[1] +
mostra_color[2]) * 100.0f;
```

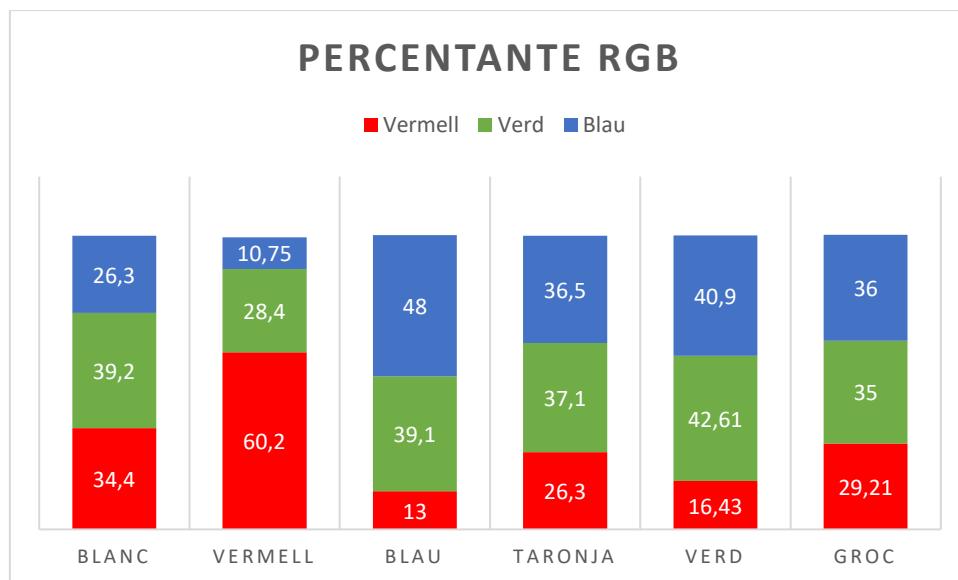
$$Verd \% = \frac{Verd}{Vermell + Verd + Blau} \cdot 100$$

```
float g = mostra_color[1] / (mostra_color[0] + mostra_color[1] +
mostra_color[2]) * 100.0f;
```

$$Blau \% = \frac{Blau}{Vermell + Verd + Blau} \cdot 100$$

```
float b = mostra_color[2] / (mostra_color[0] + mostra_color[1] +
mostra_color[2]) * 100.0f;
```

I els valors obtinguts van ser els següents:

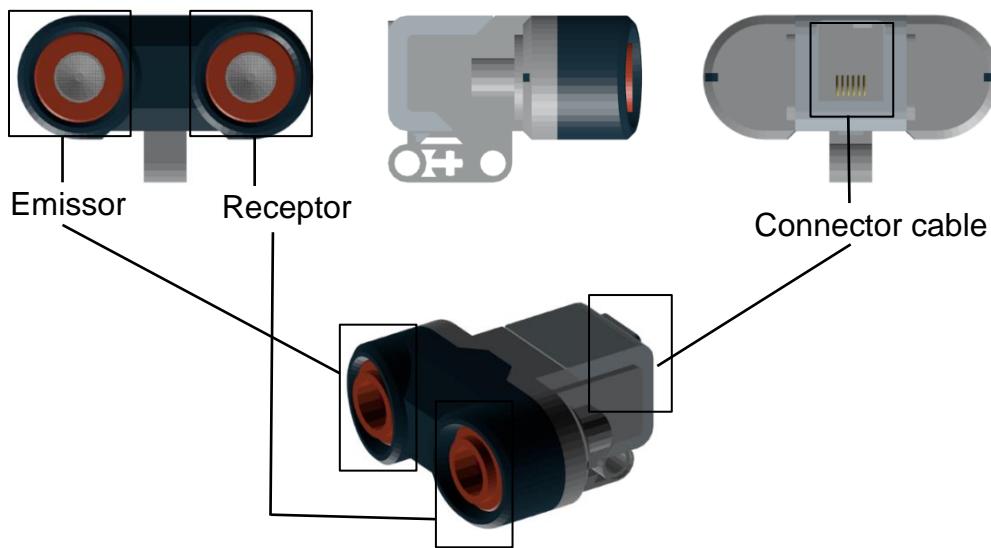


Taula de percentatge de colors

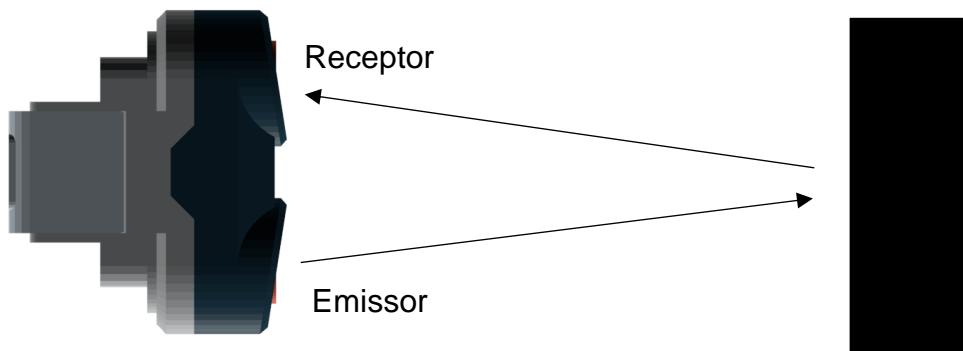
El tant per cent de cada color era força precís, és a dir, no variava per molt que les condicions lumíniques canviessin, per tant, vaig programar el sensor per a que tinguessin un marge d'error d'un 2-3 per cent, ja que el taronja i el groc tenien valors molt semblants i a l'augmentar el marge d'error podria fer que el sensor confongués els colors.

3.3.3. Sensor de distància

Esquema



El funcionament és força senzill. L'emissor emet uns sons de freqüència alta, inapreciable per les oïdes humanes, i aquest viatja a la velocitat del so (330 m/s). A continuació impacta contra l'objecte i rebota fins arribar a l'emissor.

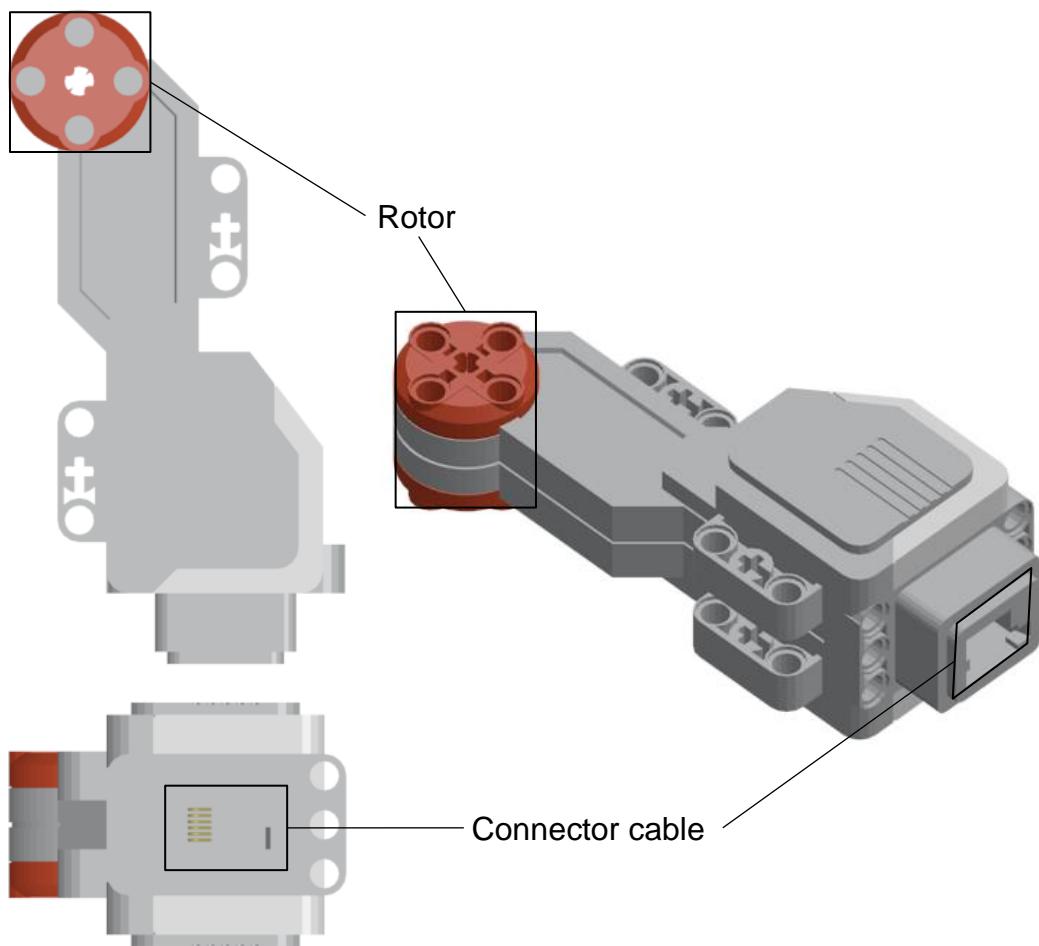


El sensor és capaç de mesurar el temps que triga el so en arribar al receptor, per tant, com la velocitat del so en l'aire és constant, es pot calcular la distància.

$$\text{Distància} = \frac{\text{Velocitat del so a l'aire} \cdot \text{Temps}}{2}$$

3.3.4. Motor gran

Esquema



Definició

El servo motor gran EV3 és un motor amb connexions de LEGO, que permet el seu acoblament fàcil al robot. Aquest motor és ideal per rotar o moure mecanismes amb un pes elevat, ja que el seu moment és força gran. La connexió que veiem a la part frontal, ens permet connectar el motor amb el bloc EV3. Inclou la tecnologia "tacho feedback", que permet saber l'angle instantani del motor.

Característiques

- Nom:EV3 Large Servo Motor
- Preu: 24,95 \$
- Velocitat angular màxima: 160-170 rpm

-Moment estàtic: 20 n/cm

-Moment dinàmic: 40 n/cm

Codi del component utilitzat a la programació

- Informacions del motor

Per fer girar el motor fins que es troba un obstacle o un límit.

```
while (!Motor."Motor".isStalled()) {"Gira el motor"}
```

El següent codi retorna l'angle actual del motor, la qual cosa serveix per corregir errors, és a dir, si l'angle desitjat no és l'angle actual se li donarà una ordre al motor per igual aquests angles.

```
Motor.C.getTachoCount()
```

- Funcions del motor

Girar el rotor fins a l'angle indicat, executant el codi en sèrie¹¹ o en paral·lel¹²

```
Motor.C.rotateTo(int "angle", bool "mode")
```

Indicar la velocitat angular en n/min⁻¹

```
Motor.A.setSpeed(int "velocitat angular");
```

Restaurar l'angle “0” del rotor

```
Motor.A.resetTachoCount();
```

Girar motor sentit horari

```
Motor.B.forward();
```

Girar motor sentit anithorari

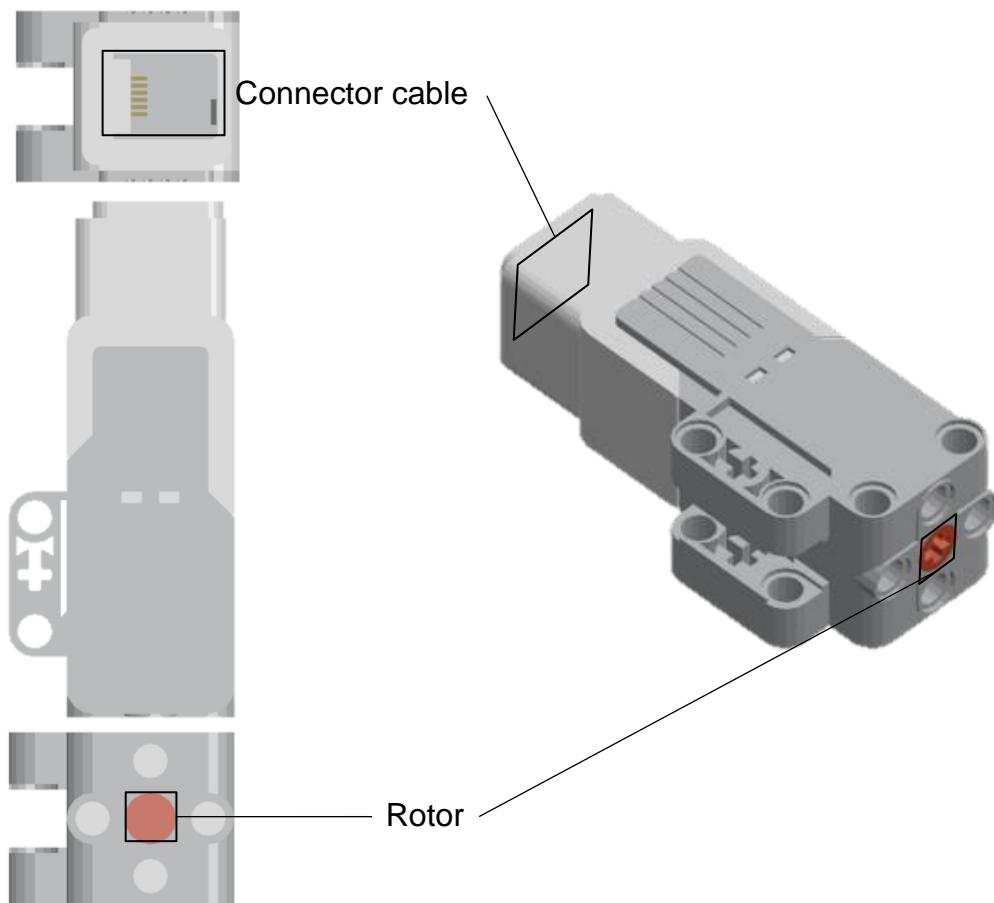
```
Motor.C.backward();
```

¹¹ **Codi en sèrie:** El codi s'executa de manera lineal, és a dir, fins que un motor no acaba de girar, no s'executarà el codi hi ha a continuació.

¹² **Codi en paral·lel:** El codi en paral·lel permet fer girar un motor mentre s'executa la resta del codi.

3.3.5. Motor mitjà

Esquema



Definició

El servo motor mitjà EV3 és el mateix que el servo motor gran però amb dimensions i característiques diferents, però es programa de la mateixa manera.

Característiques:

Nom: EV3 Medium Servo Motor

Preu: 19,95 \$

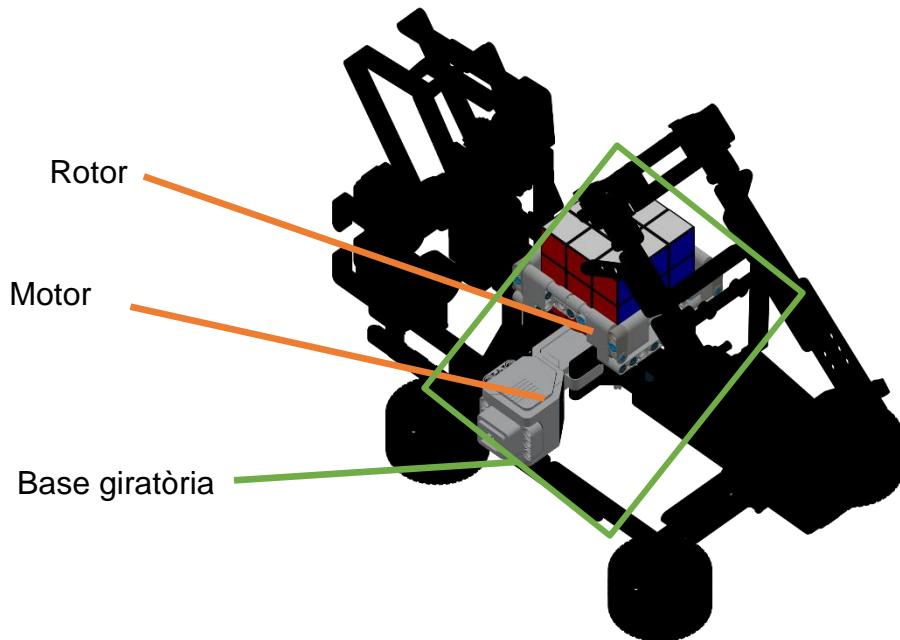
Velocitat angular màxima: 240-250 rpm

Moment estàtic: 8 n/cm

Moment dinàmic: 12 n/cm

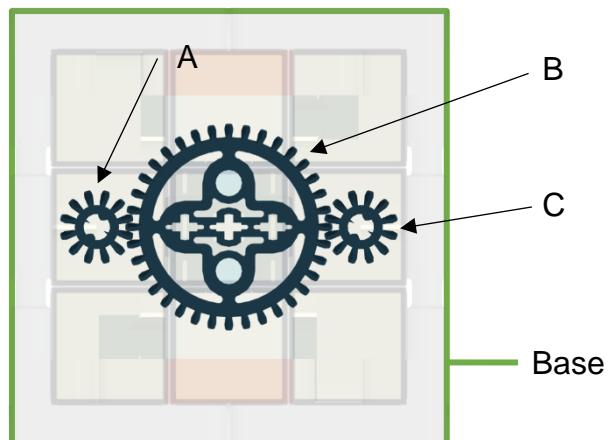
3.4. PARTS MÒBILS

3.4.1.1. Base Giratòria



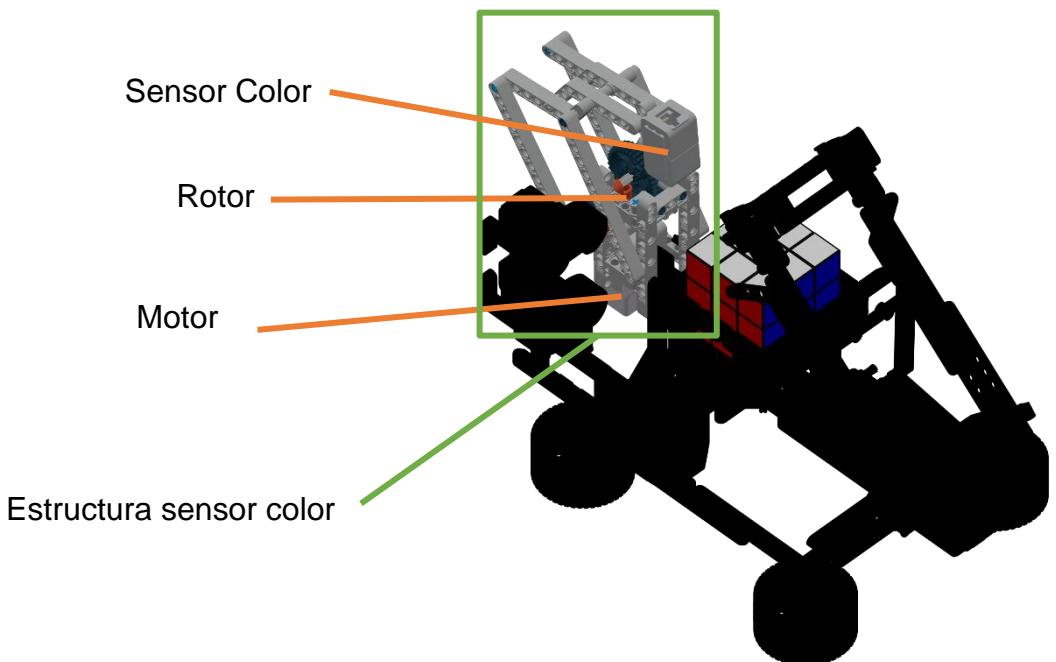
La base giratòria és el conjunt de peces que permet la rotació de la base on es situa el cub de Rubik. Aquesta estructura està formada per el motor, i un sistema d'engranatges connectat a la base.

El sistema d'engranatges està format per l'engranatge motriu (A), que va lligat amb el rotor, l'engranatge conduit (B), que va lligat amb la base, i l'engranatge estabilitzador (C).

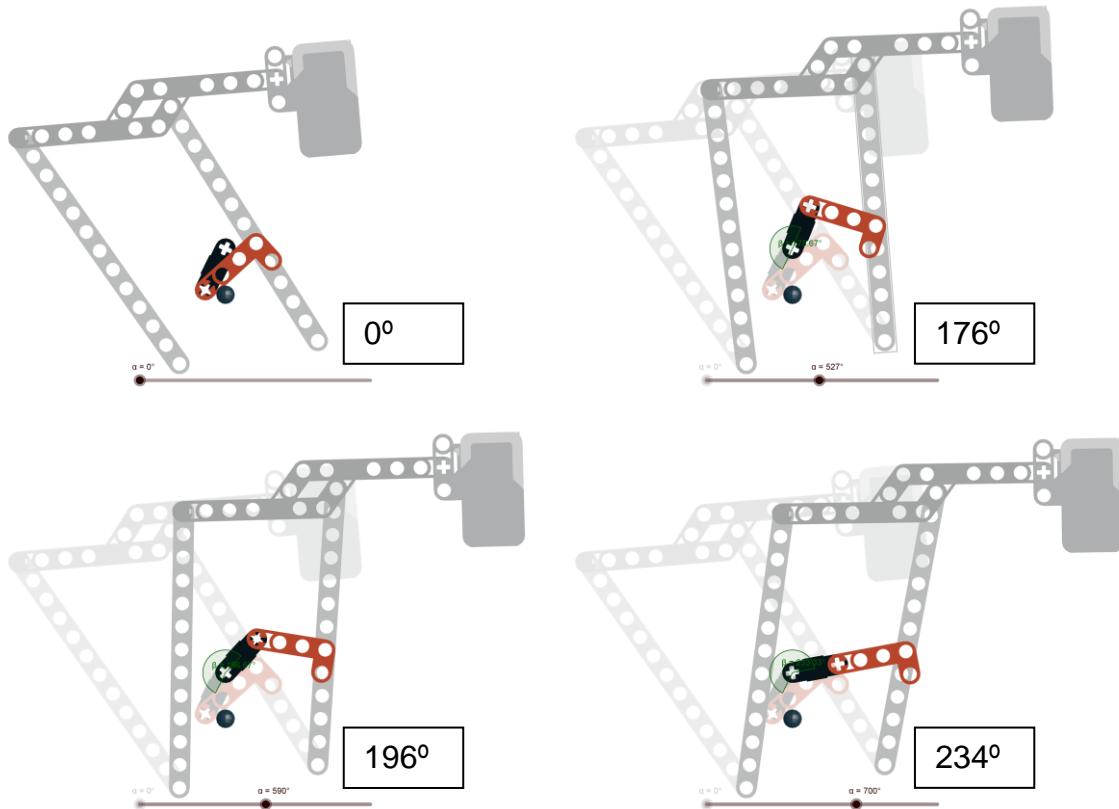


El sistema d'engranatges té una relació de transmissió $i = \frac{Z_1}{Z_2} = \frac{12}{36} = \frac{1}{3}$, on Z és el número de dents de l'engranatge. Per tant, la velocitat angular (W) de la base és $w_2 = w_1 \cdot i = \frac{w_1}{3} rad/s$, de manera que 3 voltes del rotor (engranatge motriu), suposen 1 volta de la base (engranatge conduit), per tant: $angle\ base = \frac{angle\ rotor}{3}$.

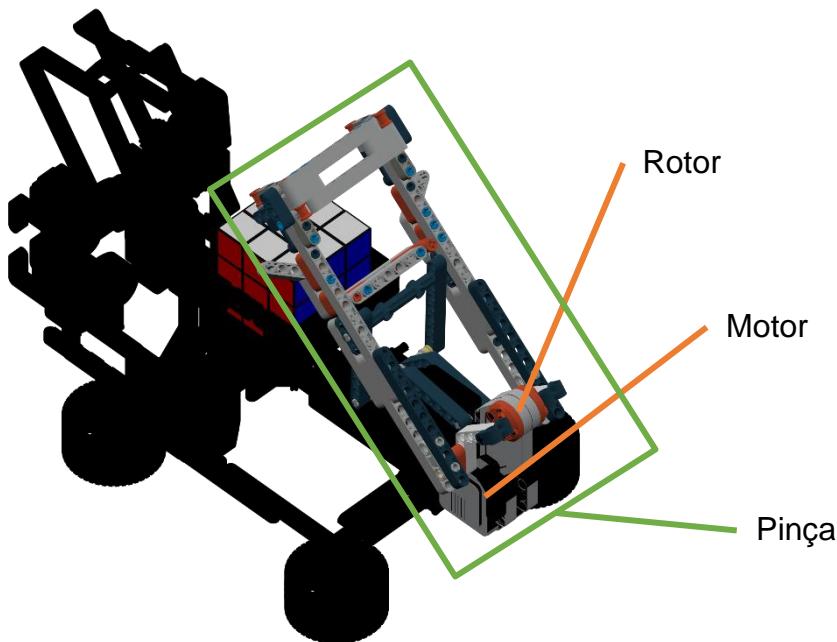
3.4.1.2. Estructura del sensor de color



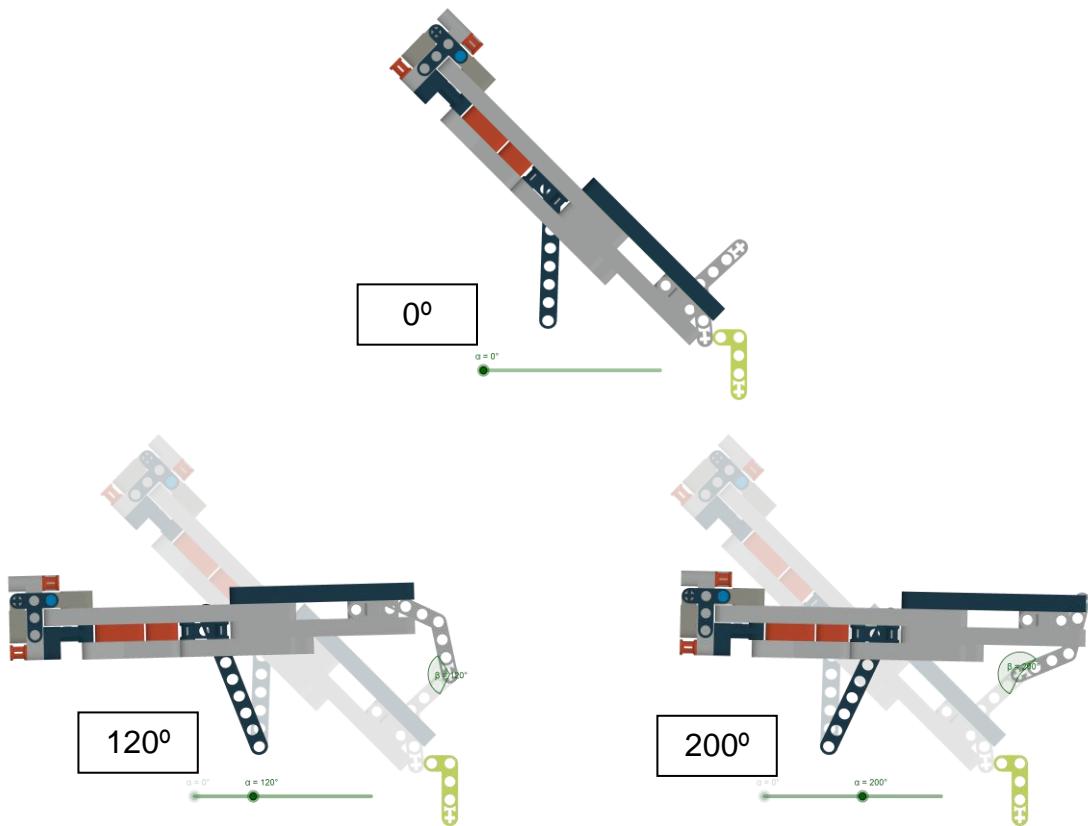
L'estructura del sensor de color és el conjunt de peces que permeten el moviment del sensor de color. La rotació del rotor es transforma en un moviment del sensor a través d'una estructura de peces. A continuació es mostren unes captures del GeoGebra de la posició del sensor segons l'angle del rotor:



3.4.1.3. Pinça



La pinça és el conjunt de peces que permet o bé fixar el cub de Rubik a la base o bé fer-lo girar sobre l'eix Z. A continuació es mostren unes captures del GeoGebra on es pot apreciar la posició de la pinça segons l'angle del rotor

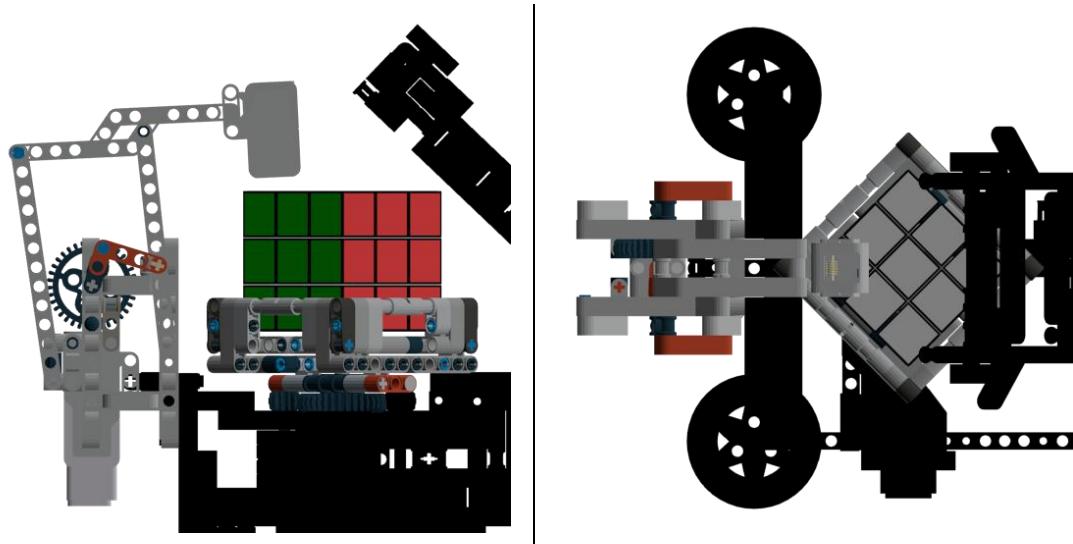


3.4.2. Moviment del sensor de color

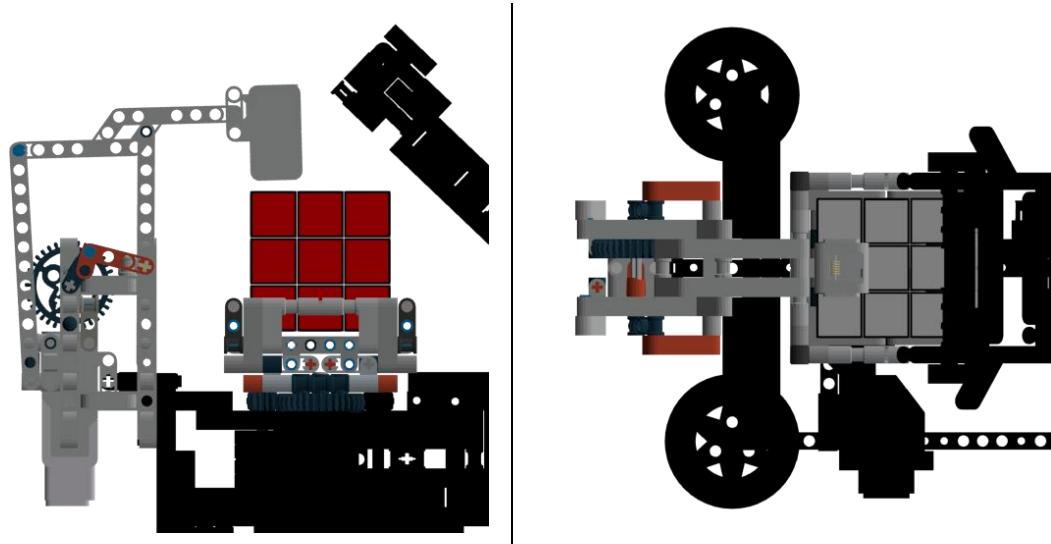
Vista frontal

Vista superior

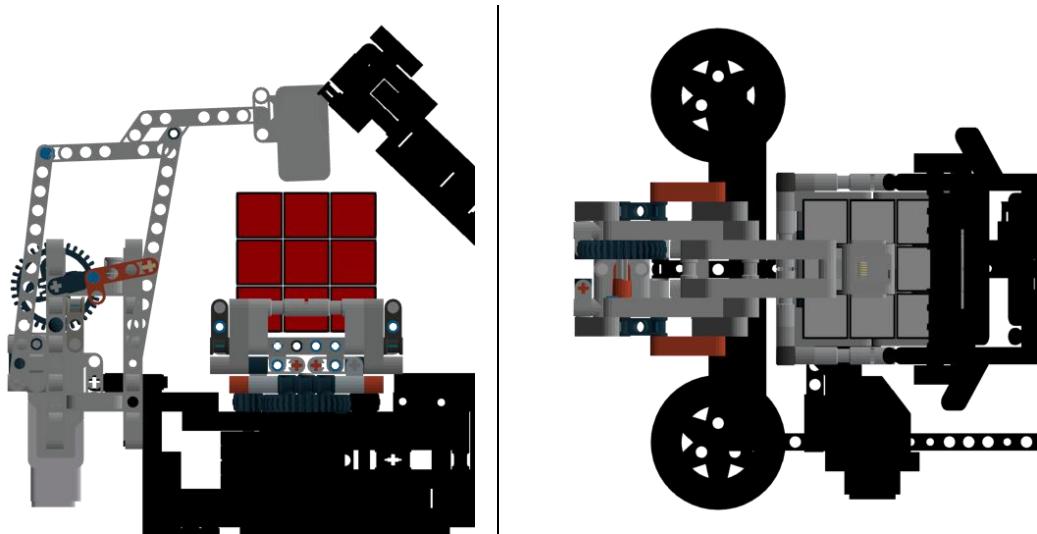
Per detectar el color de les cantonades de cada cara, el rotor de l'estructura del sensor gira fins a un angle de 176° mentre que la base giratòria rota $k * 90^\circ + 45^\circ$ on k és un número de 0 a 3. Per tant, per llegir la primera cantonada la base giratòria es situa a 45° , per a la segona a 135° , per a la tercera a 225° i per a la quarta a 315° .



Per detectar el color dels laterals de cada cara, el rotor de l'estructura del sensor gira fins a un angle de 196° mentre que la base giratòria rota $k * 90^\circ$ on k és un número de 0 a 3. Per tant, per llegir el primer costat, la base es situa a 0° , per al segon a 90° , per al tercer a 180° i per al quart a 270° .

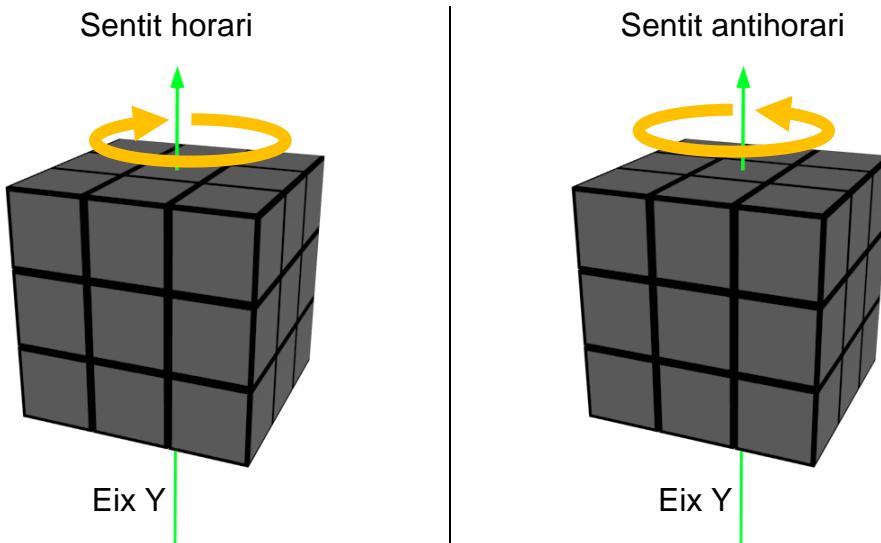


Per detectar el color del centre de cada cara, el rotor de l'estructura del sensor gira fins a un angle de 234º.



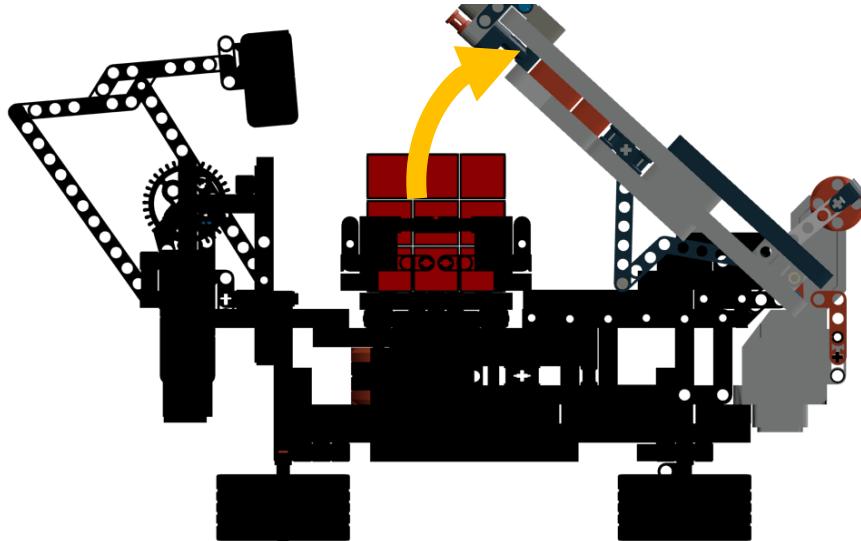
Codi: Lectura_Cub (es troba intercalat en un altre codi)

3.4.3. Gir del cub eix vertical

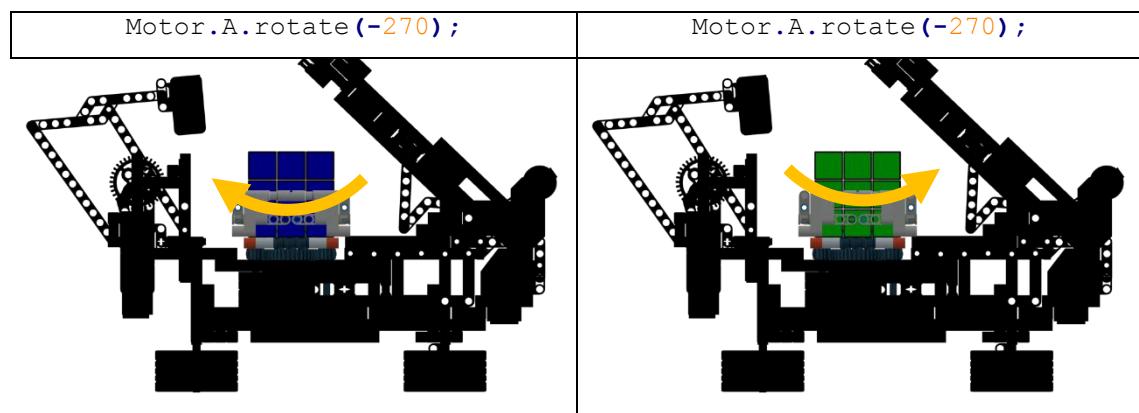


Si la pinça està col·locada al cub, es retira:

```
if(Motor.C.getTachoCount()>5) {  
    Motor.C.rotateTo(0);  
}
```

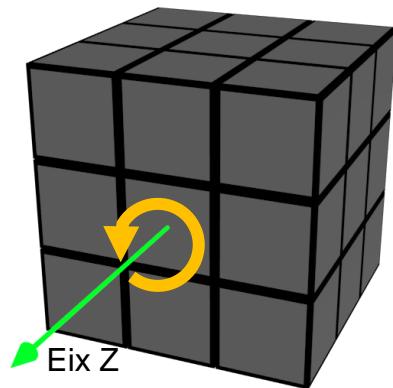


La base gira 90 en el sentit de la direcció:



Codi: Moviments(pàg.) [101]U[113]

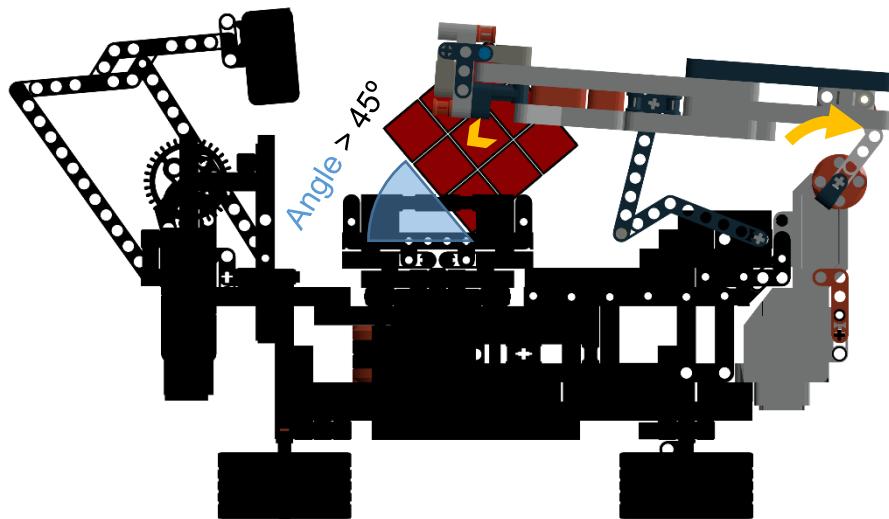
3.4.4. Gir cub eix profunditat



- Procediment

La pinça inclina el cub més de 45 graus, de manera que el centre de gravetat del cub es projecta sobre la cara que es vol situar a la base.

```
Motor.C.rotateTo(200);
```

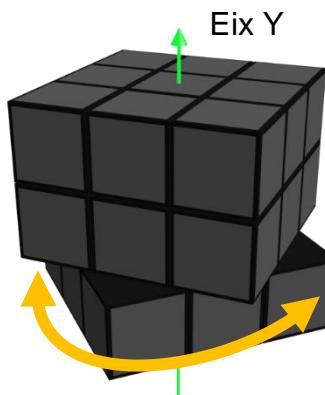


La pinça torna a la posició inicial (pinça sobre el cub), de manera que empenta el cub i el fa acabar de girar.

```
Motor.C.rotateTo(120);
```

Codi: Moviments(pàg. 56) línies [122,123]

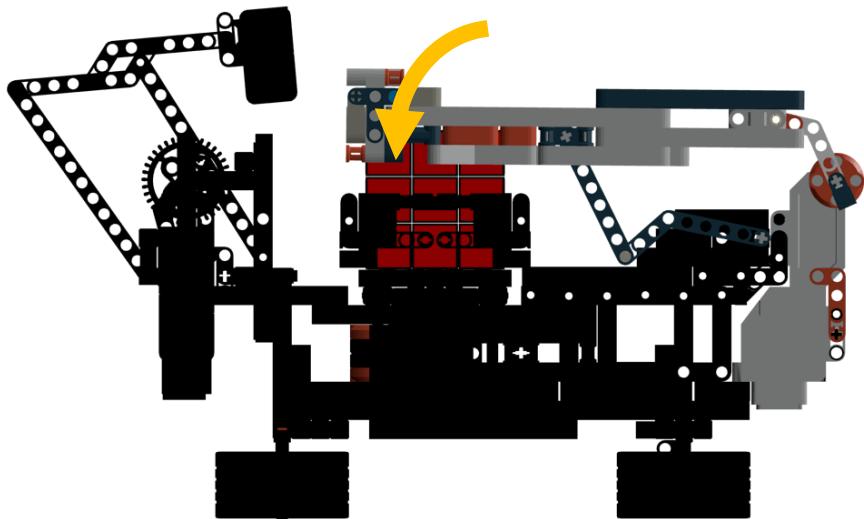
3.4.5. Gir cara



- Procediment

Si la pinça encara no està col·locada al cub de rubik, el motor la col·loca

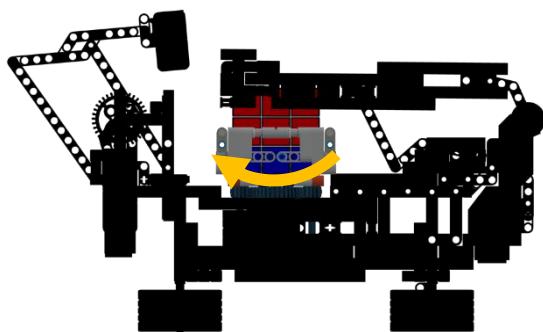
```
if(Motor.C.getTachoCount()<70){  
    Motor.C.rotateTo(120);  
}
```



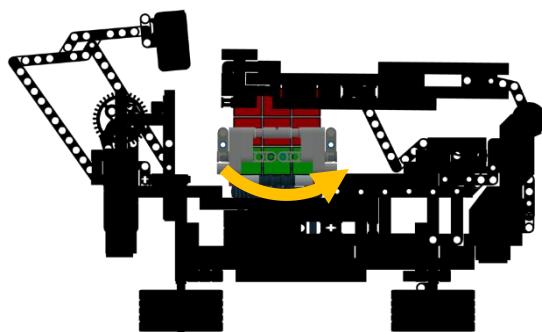
La base gira 90 graus tants cops com la variable (quantitat) demani. Si el paràmetre “sentit” és verdader, la base gira 90 graus en sentit horari, si és fals, ha fa en sentit anithorari

```
for(int i=0;i<quantitat;i++) {
    if(sentit){
        Motor.A.rotate(-3 * 90);
    }else{
        Motor.A.rotate(3 * 90);
    }
}
```

Sentit horari



Sentit antihorari



Codi: Moviments(pàg. 56) línies [81,87]

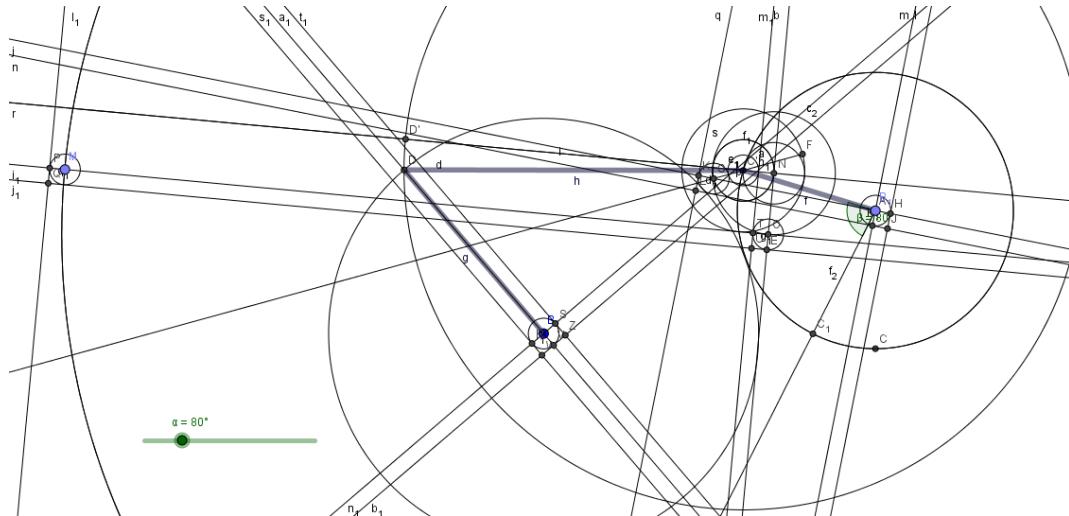
3.4.6. Simulació de motors al GeoGebra

GeoGebra és un software matemàtic interactiu lliure, que consta d'un processador geomètric i un algebraic, on es poden dibuixar i crear formes geomètriques. He aprofitat aquesta eina per fer una simulació del moviment de

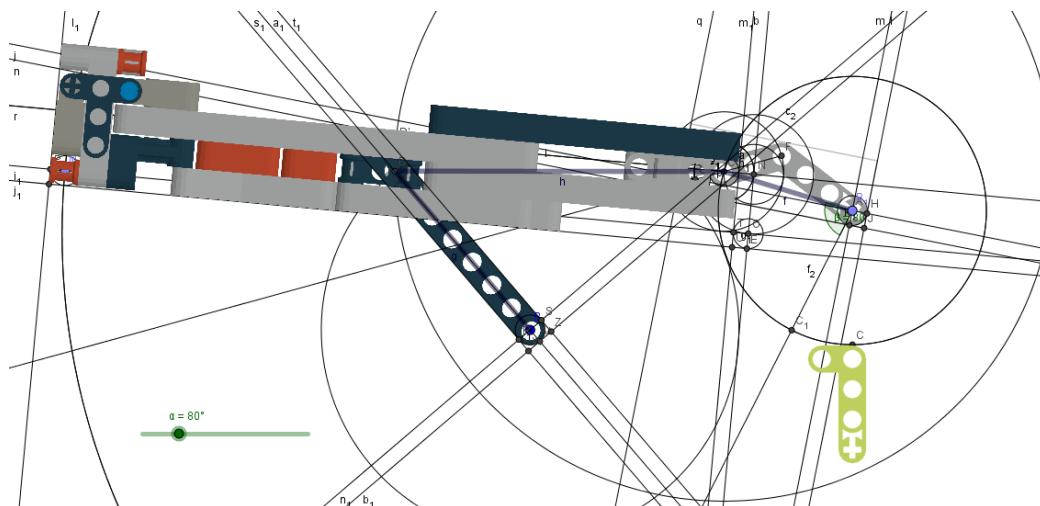
l'estructura de Lego en funció de l'angle del rotor. Aquest és el procediment que he seguit per a construir els simuladors:

Primer de tot he creat un “slider” que conté un valor que pot variar entre 0 i 360 que es vincula a un angle.

Posteriorment he creat l'esquelet de l'estructura, amb principis de dibuix tècnic:



I finalment he adjuntat les imatges de cada peça vinculades als punts estratègics de l'esquelet:

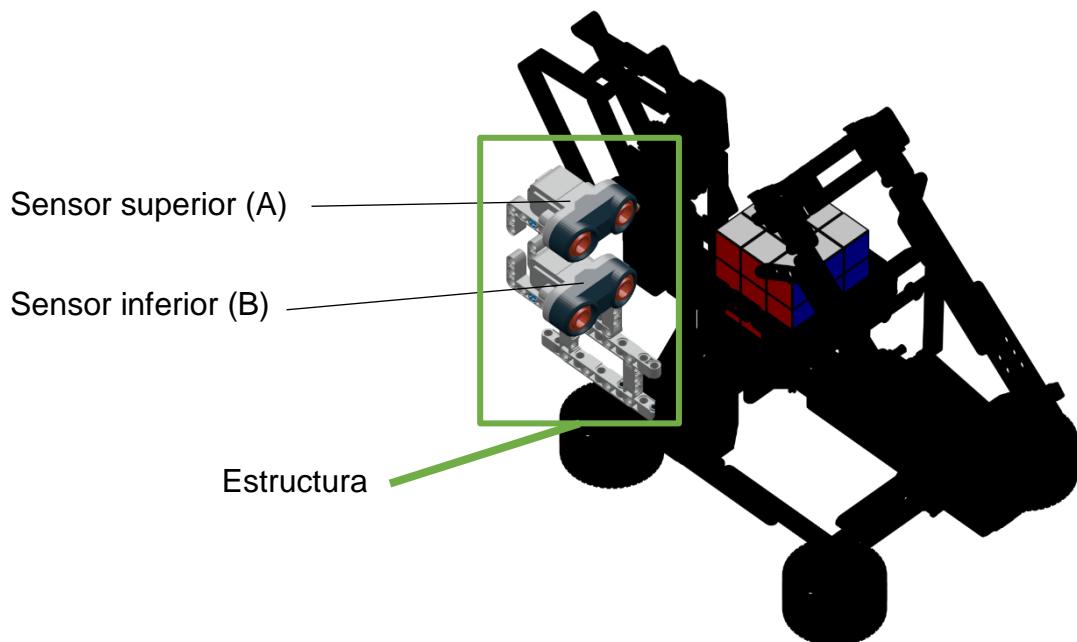


Després he “amagat” tot l'esquelet de manera que només es visualitzen les imatges.

Lliscant el “slider” es pot veure com es mou tota l'estructura en funció de l'angle del rotor. Cosa que m'ha permès saber quin angle havia de girar-lo per tal d'aconseguir qualsevol moviment. Fer aquests simuladors de moviment dels motors m'ha ajudat a esbrinar quin angle havien de rotar els motors per tal d'assolir les posicions adequades.

3.5. SISTEMA INTUÏTIU DE MOVIMENTS

Vaig pensar, que a part de resoldre automàticament el cub de Rubik seria interessant poder aplicar girs i rotacions de cara aprofitant els sensors de distància EV3, així que vaig aprofitar els dos sensors d'ultrasons podria simular tots els moviments que el robot podia fer.



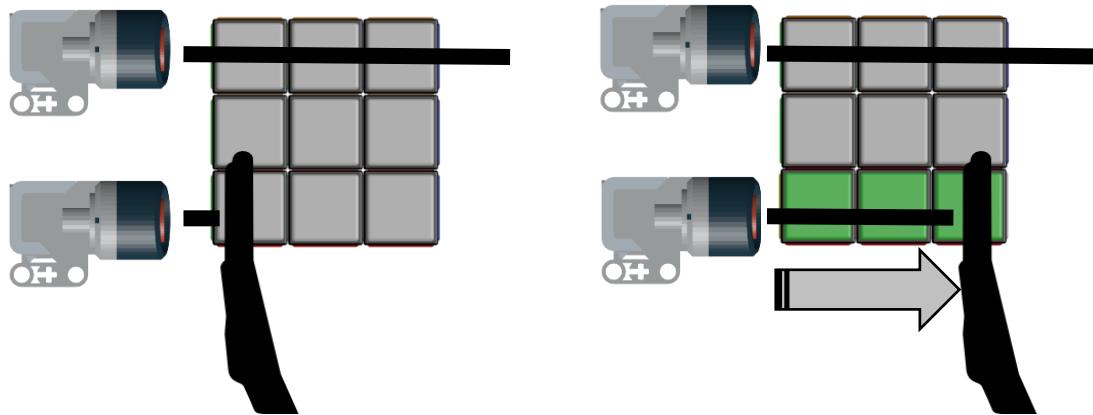
Funcionament

El funcionament és simple; quan qualsevol dels dos sensors detecta una distància propera fa un so, al cap de mig segon, torna a fer un altre so i aquest cop guarda les dues distàncies en variables (distància sensor A inicial i distància sensor B inicial), i al cap de mig segon més, torna a mesurar les distàncies finals i les guarda. Un cop el robot té les quatre variables emmagatzemades, segons com siguin farà els següents moviments:

- **Girar base sentit antihorari:**

Els motors giraran la base del cub cap a la dreta si:

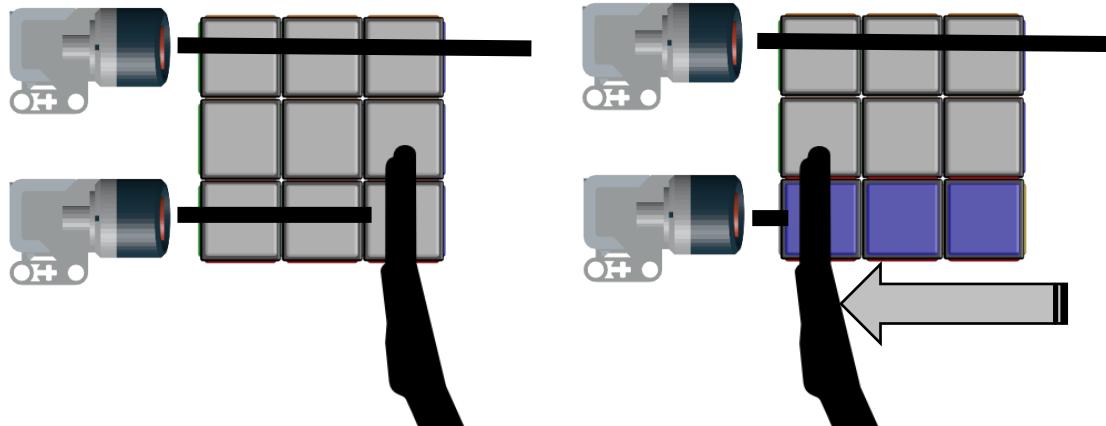
	Distancia inicial	Distancia final
Sensor superior (A)	Infinit	infinit
Sensor inferior (B)	b_1	$b_1 < b_2$



- Girar base sentit horari:**

Els motors giraran el cub en sentit horari si:

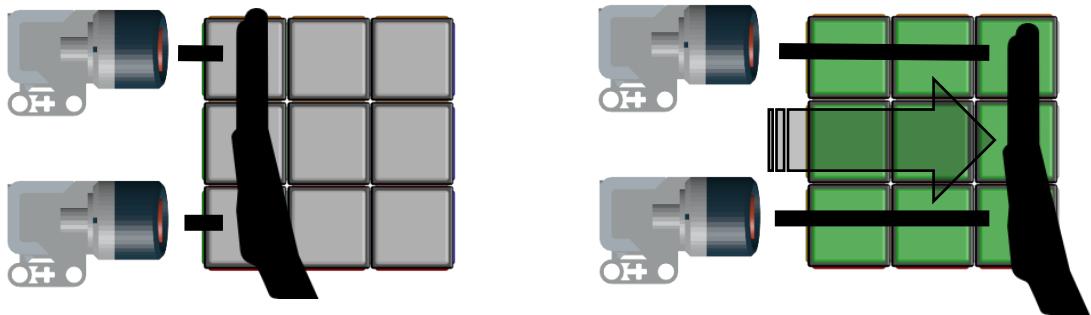
	Inici	final
Sensor superior (A)	Infinit	infinit
Sensor inferior (B)	b_1	$b_1 > b_2$



- Gir cub sentit antihorari**

Els motors giraran el cub cap a la dreta si:

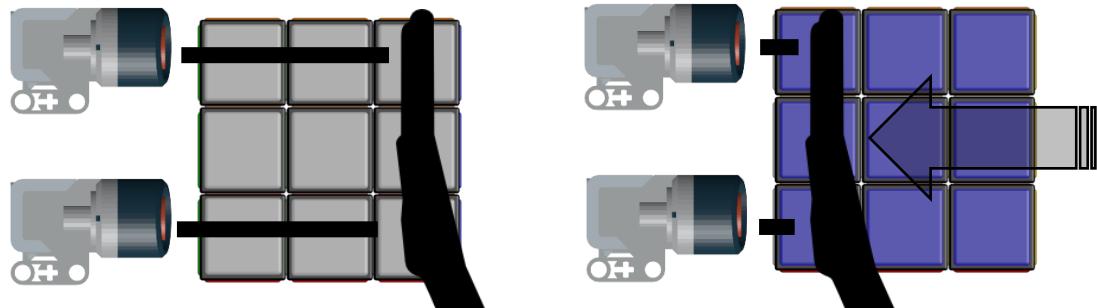
	Inici	final
Sensor superior (A)	a_1	$a_1 < a_2$
Sensor inferior (B)	b_1	$b_1 < b_2$



- Girar cub sentit horari**

Els motors giraran el cub cap a l'esquerra si:

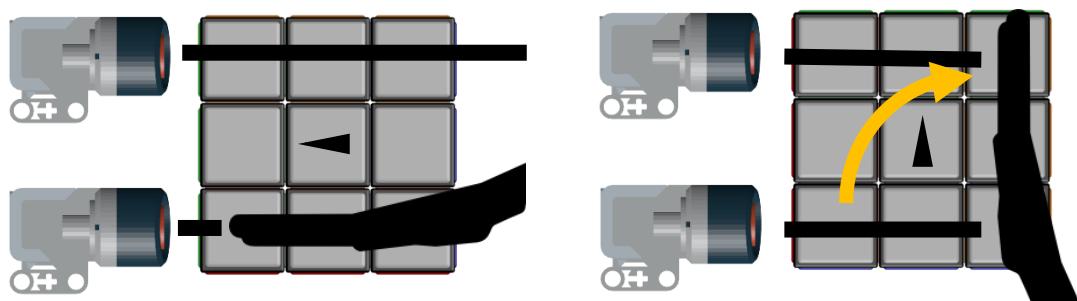
	Inici	final
Sensor superior (A)	a_1	$a_1 > a_2$
Sensor inferior (B)	b_1	$b_1 > b_2$



- Gir eix profunditat**

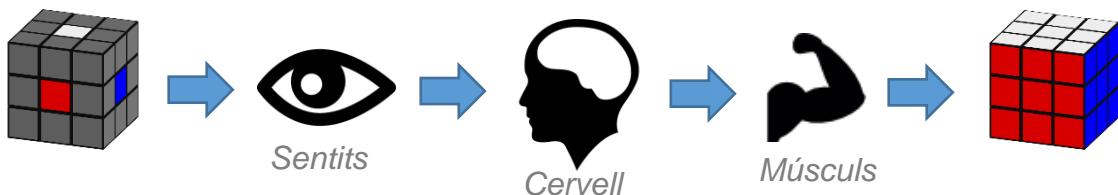
Els motors giraran el cub sobre l'eix Z si:

	Inici	final
Sensor superior (A)	infinit	a_2
Sensor inferior (B)	b_1	$b_1 < b_2$

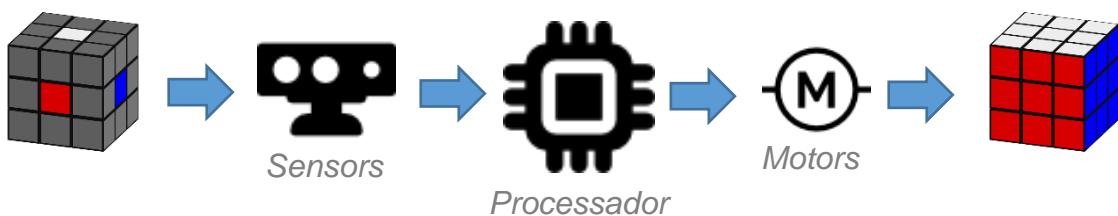


4. PROGRAMACIÓ

Un ésser humà, quan resol un cub de Rubik, sense adonar-se, és capaç de detectar els colors, trobar solucions i investigar el cub gràcies a la intel·ligència proporcionada pel cervell. Aquesta intel·ligència, sumada a la memòria que es requereix per memoritzar alguns algoritmes per resoldre els diferents passos, fan que un humà pugui resoldre un cub sense gaires dificultats, ja que la vista converteix la imatge en dades, el cervell interpreta les dades i finalment els músculs poden efectuar els moviments per a resoldre el cub de Rubik.



En un robot el funcionament és diferent. Un robot té memòria; no intel·ligència, la intel·ligència és artificial, s'ha de simular, per tant la informació és captada pels sensors en forma de nombres, s'envia al processador en forma de nombres, aquest processador calcula una solució a base d'uns algoritmes que simulen la intel·ligència humana d'una manera capaç d'entendre per una màquina, és a dir, amb codi de programació, i finalment s'envia aquesta solució cap als motors en forma de nombres.



És per això que la programació ha estat la part més extensa del treball; perquè he hagut de convertir aquesta intel·ligència automàtica que el nostre cervell ens proporciona, en una intel·ligència artificial, és a dir, en un conjunt d'ordres, amb grans quantitats de nombres que poden arribar a confondre a qui el llegeix, bàsicament perquè qualsevol informació del cub, ja sigui els colors que conté, els moviments que ha d'efectuar el robot, la orientació del cub, o qualsevol altre paràmetre, es tradueix en forma de nombres.

Cal dir també que aquesta programació és difícil d'explicar i d'entendre, ja que conté una gran quantitat de codi per evitar errors, corregir imprecisions o facilitar-ne la interpretació. Aquest codi si no és explicat amb el robot al davant i amb explicacions visuals o animades és complicat d'entendre.

4.1. SOFTWARE

Els components electrònics de Lego Mindstorms es poden programar amb un software que funciona fent “cliks” i arrossegant, pensat per a nens o persones que volen fer un ús inexpert dels components, però aquest software es insuficient per a resoldre un cub de Rubik. Vaig trobar un plugin¹³ d'Eclipse¹⁴ anomenat leJOS, que permet la programació dels components EV3 en codi, de manera que es poden accedir a molts més recursos dels components i es poden fer moltes més operacions per a resoldre el cub. Per fer servir aquest plugin, vaig haver d'instal·lar una targeta MicroSD amb el plugin al bloc EV3, i descarregar-me el plugin a l'Eclipse.

He tingut sort d'utilitzar el plugin leJOS, perquè utilitza el llenguatge Java¹⁵, amb el qual ja havia programat algun cop per a programar algun petit videojoc. LeJOS utilitza Java perquè és un llenguatge senzill, fàcil d'aprendre i perquè està orientat a objectes.

4.2. NOTACIÓ

Al codi de la programació, tot s'indica amb nombres i lletres, per tant s'ha d'establir un llenguatge per poder referir-se als colors, cares o moviments que fa el cub de Rubik.

Colors: per enunciar cada color, es fa servir un número; aquesta és la llista

Color	Número
Blanc	0
Vermell	1
Blau	2

¹³ **Plugin:** Extensió d'un programa per afegir-li una funció

¹⁴ **Eclipse:** Entorn integrat de desenvolupament de codi

¹⁵ **Java:** Llenguatge de programació orientat a objectes

Taronja	3
Verd	4
Groc	5

Aquest patró també serveix per a enunciar les cares, és a dir, cara[1] equival a la cara vermella.

Orientació: per referir-se a les cares segons la seva orientació i no el seu color, es fa servir el següent patró:

Cara	Número
Superior	0
Frontal	1
Dreta	2
Posterior	3
Esquerra	4
Inferior	5

Girs: per referir-se als següents girs es farà servir:

Gir	Sentit	Lletra	Prové de
Cara superior	Horari	U	Up
	Antihorari	u	
Cara frontal	Horari	F	Front
	Antihorari	f	
Cara dreta	Horari	R	Right
	Antihorari	r	
Cara posterior	Horari	B	Back
	Antihorari	b	
Cara esquerra	Horari	L	Left
	Antihorari	l	
Cara inferior	Horari	D	Down
	Antihorari	d	

També cal dir que si després d'aquesta lletra s'afegeix el número “2”, el gir s'efectuarà 2 cops.

4.3. SOLUCIÓ DEL CUB DE RUBIK

4.3.1. Lectura del cub

La lectura del cub és tot el procés per el qual el robot és capaç de transformar el cub de Rubik en una matriu de valors que relaciona posicions i colors. Aquesta matriu és la que més tard s'analitza per solucionar el cub de Rubik.

4.3.1.1. Detecció del color

El sensor, un cop ha fet la lectura RGB, obté una variable “r” referent al tant per cent de vermell, una variable “g” referent al tant per cent de verd, i una variable “b” referent al tant per cent de blau.

Llavors es fa passar el codi per un conjunt de sis “if”, on, en cadascun d'aquests s'avalua si els valors rgb obtinguts s'acosten a algun percentatge dels colors de la “Taula de percentatges rbg” (pàg.).

Si cap d'aquests “if” retorna un valor positiu, el robot suposa que el sensor de color no està posicionat just per sobre del color que es vol llegir, així que va rotant uns graus la base giratòria i avança i retrocedeix el sensor de color fins que aquests obtenen uns valors rgb admesos en alguns d'aquests “if”. Llavors, dins del “if” que hagi retornat verdader, el codi adjudica a aquesta peça el color del “if”.

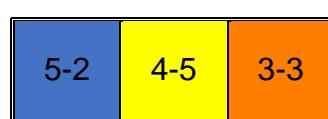
Codi: Lectura_Cub(pàg. 52)

4.3.1.2. Construcció d'un cub de Rubik numèric

El sensor de color, segueix un patró a l'hora de llegir cada cara, és a dir, segueix ordre a l'hora de llegir els colors. Aquest ordre és el que s'aprecia a la imatge de la dreta.

Comença pel centre, avança cap a la cantonada inferior dreta, i a partir d'aquesta avança en sentit antihorari fins a llegir totes les peces de la cara.

5	4	3
6	0	2
7	8	1



Els colors obtinguts s'emmagatzemen en forma de nombre en un array doble “cub[6][9]”, on el primer grau de l'array s'entra amb el nombre del color de la cara i el segon grau s'entra amb el valor de la posició de la peça.

6-0	0-4	2-2
7-5	8-1	1-4

Per exemple, a la imatge de la dreta es pot observar que $cub[4] = \{4,4,2,3,5,2,0,5,1\}$, ja que els colors que s'han llegit a la cara verda (4), són: verd(4), verd(4), blau(2), taronja(3), groc(5), blau(2), blanc(0), groc(5) i vermell(1).

O el que és el mateix: $cub[4][0]=4$, $cub[4][1]=4$, $cub[4][2]=2$, $cub[4][3]=3$, $cub[4][4]=5$, $cub[4][5]=2$, etc.

Codi: Lectura_Cub(pàg. 52) línies (73)U(78)U(83)U(88)U(93)U(98)

4.3.1.3. Ordenació de colors

Un cop es troben tots els colors del cub en la matriu “cub[][],” és important canviar l'ordre per facilitar la solució del cub posteriorment, de manera que es canvia l'ordre intern de cada cara per un altre ordre més lògic i representatiu que avança d'esquerra a dreta i de dalt a baix.

Per exemple, $cub[4]$, que era $\{4,4,2,3,5,2,0,5,1\}$, passa a ser $\{2,5,3,0,4,2,5,1,4\}$



La imatge inferior mostra quins valors hauríem d'entrar a l'array “cub[][]” per tal d'accedir al valor del color de cada peça:

Superior			Dreta			Posterior			Esquerra		
0.0	0.1	0.2									
0.3	0.4	0.5									
0.6	0.7	0.8									
Frontal											
1.0	1.1	1.2	2.0	2.1	2.2	3.0	3.1	3.2	4.0	4.1	4.2
1.3	1.4	1.5	2.3	2.4	2.5	3.3	3.4	3.5	4.3	4.4	4.5
1.6	1.7	1.8	2.6	2.7	2.8	3.6	3.7	3.8	4.6	4.7	4.8
Inferior			5.0	5.1	5.2						
			5.3	5.4	5.5						
			5.6	5.7	5.8						

4.3.2. Girs

4.3.2.1. Girs de cara

Cada cop que s'efectua el gir d'una cara, tant en sentit horari com en sentit antihorari, tots els colors d'aquesta cara (exceptuant el centre), canvien de posició, com també ho fa el conjunt de colors que envolta la cara (corona). A les taules inferiors es pot observar el canvi d'aquests colors tant en sentit horari com antihorari quan es gira la cara superior.

0.0	0.1	0.2
0.3	0.4	0.5
0.6	0.7	0.8
1.0	1.1	1.2
2.0	2.1	2.2
3.0	3.1	3.2
4.0	4.1	4.2

0.6	0.3	0.0
0.7	0.4	0.1
0.8	0.5	0.2
2.0	2.1	2.2
3.0	3.1	3.2
4.0	4.1	4.2
1.0	1.1	1.2

0.2	0.5	0.8
0.1	0.4	0.7
0.0	0.3	0.6
4.0	4.1	4.2
1.0	1.1	1.2
2.0	2.1	2.2
3.0	3.1	3.2

Per tant, cada cop que es vol rotar una cara, cal rotar els valors de la cara i els valors de la corona d'aquesta cara, que depenen de la cara seran uns valors o altres, cosa que fa augmentar el codi, ja que s'ha de programar els valors que es volen canviar per a cada cara per separat.

Codi rotació cara: Moviments(pàg. 58) línies [241,264]

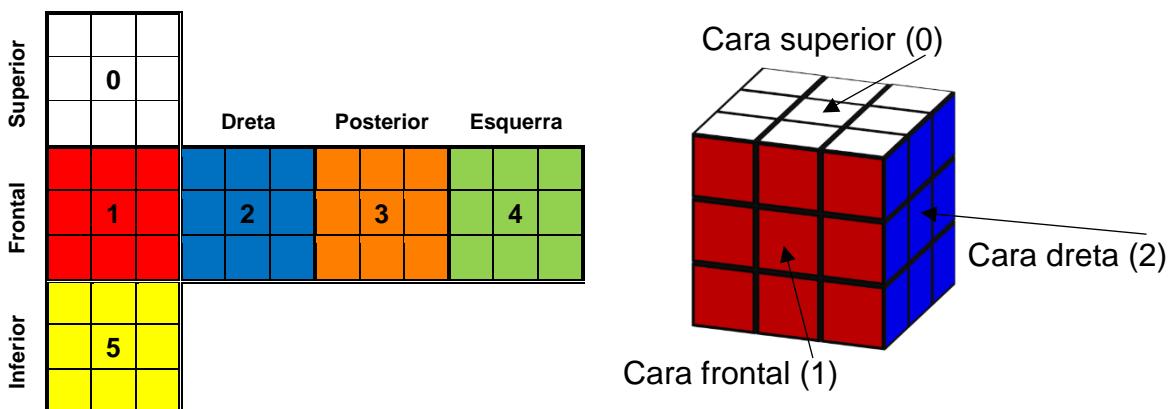
Codi rotació corona: Moviments(pàg. 56) línies [128,239]

4.3.2.2. Girs del cub

Com s'ha explicat anteriorment, el robot és capaç de girar el cub sobre l'eix vertical en sentit horari i antihorari, i sobre l'eix de la profunditat en sentit horari. Aquests moviments no canvien cap peça de lloc, per tant la matriu on es troben els valors dels colors continua intacta. El que sí que canvia és la orientació del cub, per tant cal fer constància de com queda orientat el cub després d'efectuar qualsevol moviment. Per fer-ho, es crea l'array "centres[6]". On

centres[0] = color de la cara superior
 centres[1] = color de la cara frontal
 centres[2] = color de la cara dreta
 centres[3] = color de la cara posterior
 centres[4] = color de la cara esquerra
 centres[5] = color de la cara inferior

Per tant, si el cub estigués orientat com es veu a la imatge inferior dreta, l'array "centres" seria [0,1,2,3,4,5], ja que la posició "0" de l'array l'ocupa el valor 0 (blanc), la posició "1" de l'array l'ocupa el valor "1" (vermell) etc.

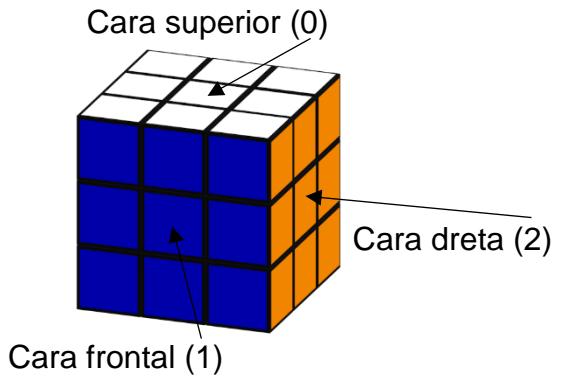
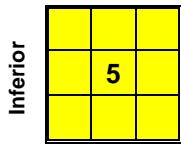


Gir horari sobre l'eix vertical

Si el robot rota el cub 90º en sentit horari respecte de la orientació inicial, llavors l'array "centres" seria [0,2,3,4,1,5], ja que la posició "0" de l'array l'ocuparia el valor "0" (blanc), la posició "1", l'ocuparà el valor "2" (blau), la posició "2" l'ocuparà el valor "3"(taronja) etc.

	0					
Superior						

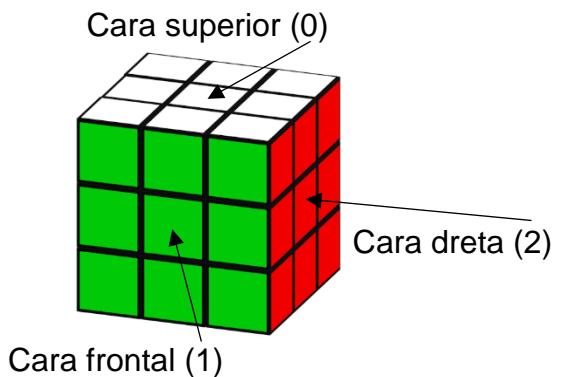
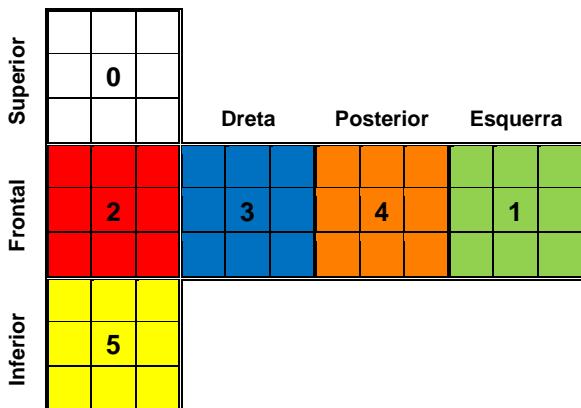
	Dreta	Posterior	Esquerra	
Frontal	4	1	2	3



Codi: Moviments(pàg. 56) línies [108,114]

Gir anitohorari sobre l'eix vertical

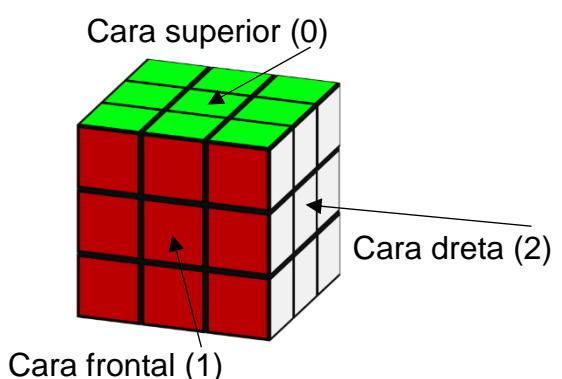
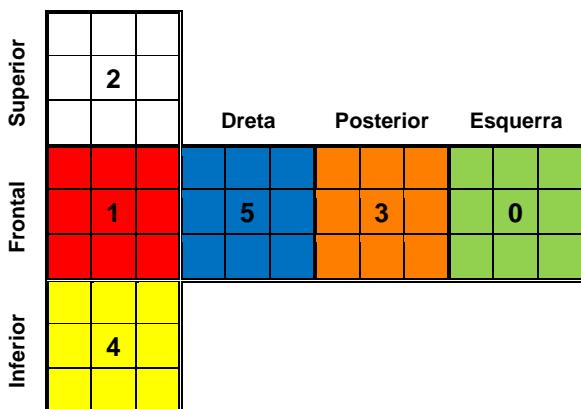
I seguint amb aquest patró, si el cub gira en sentit anitohorari, l'array “centres” serà: [0,4,1,2,3,5].



Codi: Moviments(pàg. 56) línies [95,100]

Gir anitohorari sobre l'eix de profunditat

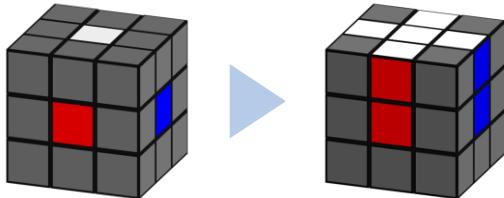
I si el cub gira d'aquesta manera, l'array “centres” serà [4,1,0,3,5,2].



Codi: Moviments(pàg. 56) línies [118,123]

4.3.3. Solució

4.3.3.1. Cross (Creu)

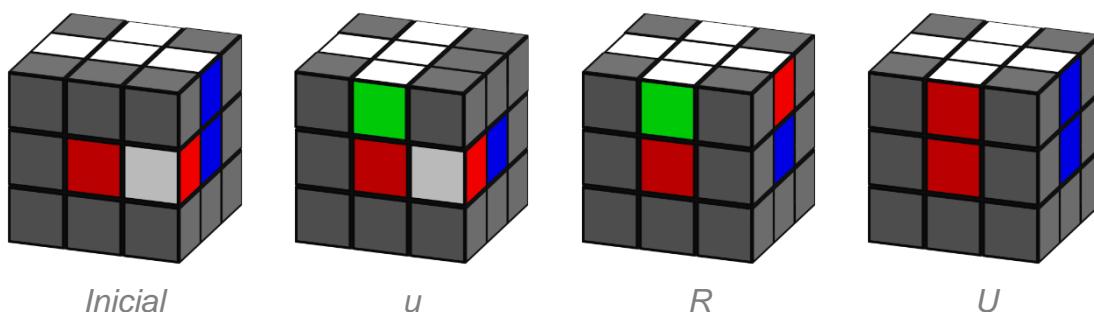


El primer pas, present a gairebé tot els mètodes de resolució existents, és resoldre la creu, que consisteix a resoldre les arestes que uneixen la cara superior amb les cares laterals.

El sistema per resoldre aquest pas és força senzill. El robot resol la creu aresta per aresta, començant per la blanca-vermella (0-1), i acabant per la blanca-verda(0-4).

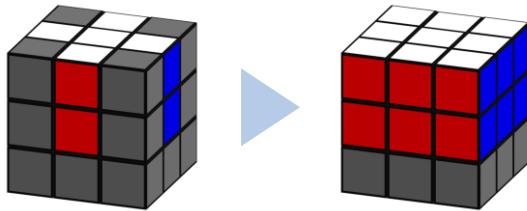
Per resoldre cada aresta, detecta la posició i la orientació en la qual es troba. Aquesta aresta pot estar situada a 12 posicions diferents i pot estar orientada de 2 maneres, per tant, pot estar 24 situacions diferents. Un cop el robot sap en quina situació es troba la aresta, efectua l'algoritme necessari de moviments per tal de portar l'aresta a la seva posició amb la seva orientació pertanyent.

Per trobar la situació en la qual es troba la aresta, el codi passa per un conjunt de 24 “if”, on cadascun està pensat per retornar verdader en una situació de l'aresta diferent, per tant per cada aresta, el codi només entrerà per un “if”, que contindrà el codi necessari per efectuar els moviments necessaris per portar l'aresta al seu lloc. A les imatges inferiors es mostra un exemple del conjunt de girs que ha d'efectuar el robot per tal de portar l'aresta a la seva posició. Cal dir que aquest procediment es repeteix per a les quatre arestes amb un sistema que va canviant el color de l'aresta que es vol solucionar.



Codi: Solució(pàg. 59) línia [25,104]

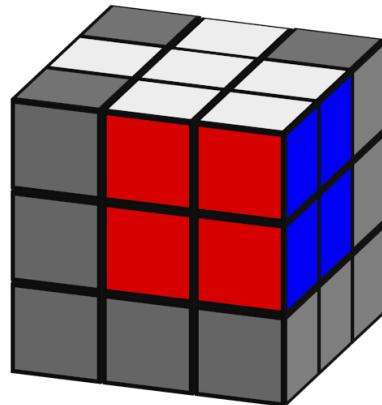
4.3.3.2. F2L (First Two Layers)



El pas F2L (First Two Layers) és el pass amb més programació perquè uneix dos passos en un sol algoritme, és a dir, soluciona les cantonades superiors i les arestes laterals en un sol conjunt de moviments.

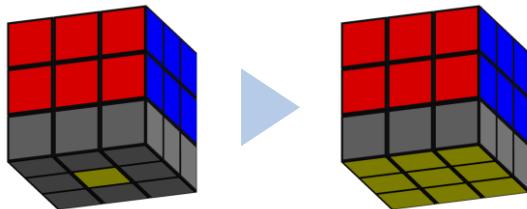
En aquest pas el codi primer resol la cantonada superior i l'aresta que uneix la cara vermella amb la cara blava, i posteriorment farà la blava-taronja, la taronja-verda i la verda-vermella.

Per entendre el procediment, suposem que es vol resoldre la cantonada superior i l'aresta lateral que uneix les cares vermella i blava, com es mostra a la imatge. El funcionament és força similar al de la creu. El codi passa per un conjunt de “if” que detecta on està la cantonada blanc-vermell-blau, i com està orientada, i on està l'aresta vermell-blau, i com està orientada, i a partir d'aquestes dades, el codi proporciona un algoritme que indica al robot quins moviments ha de fer per tal de posicionar aquestes peces al seu lloc, i posteriorment fa el mateix amb les altres peces a les altres cares fins tenir solucionades les dues primeres capes, cal dir que el nombre de situacions d'aquest pas seria força gran, ja que cada cantonada podria estar en 8 posicions orientada de 3 maneres, i cada aresta podria estar en 8 posicions orientada de 2 maneres, el que donaria un total de $8 \cdot 3 \cdot 8 \cdot 2 = 384$ situacions diferents. Però abans de passar per la cadena de “if”, un codi analitza el cub i efectua uns moviment per tal de posicionar les cantonades en llocs estratègics, de manera que aquesta quantitat de situacions acaba reduïda a 56, la qual cosa facilita la seva programació.



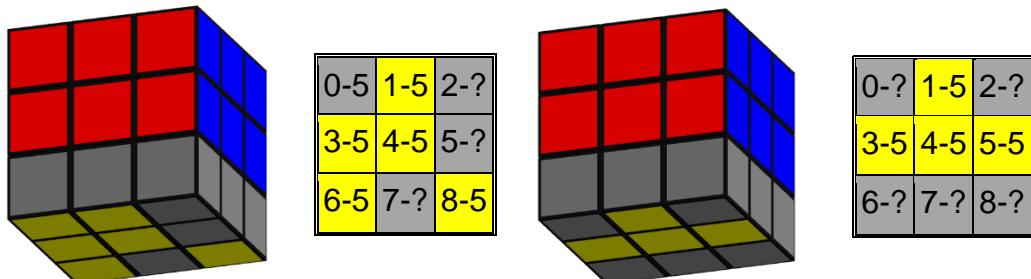
Codi: Solució(pàg. 60) línies [106,309]

4.3.3.3. OLL (Orientation Last Layer)



L'objectiu del pas del OLL (Orientation Last Layer) és el de solucionar tota la cara inferior, tot i que no es resolgui la corona d'aquesta, és a dir les peces que envolten aquesta cara.

El funcionament per resoldre el pas és totalment diferent al dels dos primers passos. Aquest sistema no es basa en detectar les situacions de les peces, sinó la "forma" que té la cara inferior en funció de les peces grogues que s'hi troben, és a dir, el patró que formen les peces grogues que es troben en la cara inferior. I depenent del patró que es trobi s'efectuarà un conjunt de moviments per resoldre la cara. Posem per exemple dos situacions:

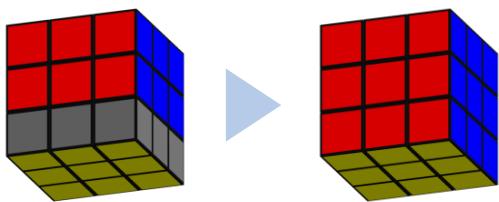


De manera que es crea un array lògic de la cara on cada posició obté valor verdader si la seva peça és groga i fals si és qualsevol altre color, per tant cada patró tindrà un array amb valors diferents, per tant, l'array de la imatge esquerra serà: `array[] = {false, true, false, true, true, false, true, false, false}`, i a la dreta: `array[] = {false, true, false, true, true, false, false, false, false}`. Així, el codi passarà per una cadena de "if" on un d'aquests contindrà el patró del cub i indicarà els moviments que ha d'efectuar el robot per tal de resoldre la cara inferior.

Aquest pas inclou també un gran nombre de situacions diferents, ja que si comptem que la cara té 8 colors que poden ser groc o no, vuit posicions poden estar en dues situacions diferents, per tant el número de situacions possibles és $2^8 = 256$, però hi ha molts casos impossibles o que són miralls o girs d'altres casos, per tant al final el nombre de situacions possibles és de 57.

Codi: Solució(pàg.63) línies [311,506]

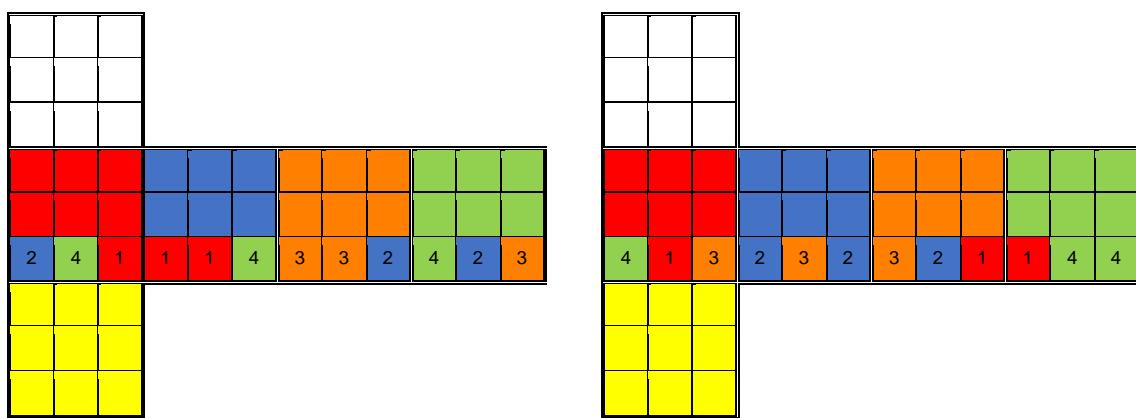
4.3.3.4. PLL (Permutation Last Layer)



L'últim pas per completar el cub de Rubik és el PLL (Permutation Last Layer).

Aquest pas funciona d'una manera semblant al OLL, però en comptes d'analitzar la cara groga, analitza la corona d'aquesta cara, és a dir, tots els colors que l'envolten, i dependent dels colors que hi troba, efectua uns moviments que resoldran el cub.

Un array converteix la corona en una llista de colors, començant per els de la cara vermella, per exemple, l'array de la imatge seria: {2,4,1,1,1,4,3,3,2,4,2,3}, mentre que el de l'esquerra seria {4,1,3,2,3,2,3,2,1,1,4,4}.



Per tant, el codi passarà per una cadena de “if” on en un d’ells l’array coincidirà amb els valors del cub, per tant efectuarà els moviments necessaris per resoldre el cub. Les situacions d’aquest pas serien força grans, ja que hi ha 12 posicions diferents on cadascuna pot ser de quatre colors diferents, per tant seria de $4^{12} = 16777216$ situacions possibles, però aquest número es veu reduït ja que la majoria de possibilitats no són possibles degut a que no hi poden haver més de tres posicions amb el mateix color, ni els colors de les arestes poden ser els mateixos en dues cares, i es pot canviar els colors per uns altres, de manera que aquest nombre al final queda reduït a una quantitat de 21 situacions diferents, és a dir, 21 “if”, on cadascun contindrà l’algoritme per acabar la solució del cub de Rubik.

Codi: Solució(pàg.65) línies [508,663]

4.4. AVANTATGES I INCONVENIENTS DEL MÈTODE ESCOLLIT

Un cop esquematitzat el funcionament de la programació que incorpora el robot, m'agradaria explicar que existeixen mètodes que solucionen el cub de Rubik amb la meitat de moviments i un codi molt més curt, que es basen en la teoria de que qualsevol cub de Rubik pot ser resolt en un màxim de 21 moviments, on cada moviment pot ser el gir d'una cara en sentit horari o antihorari, de manera que hi ha 12 diferents moviments. Amb aquest concepte, es pot suposar que si apliquem a un cub barrejat 21 moviments de totes les maneres possibles, sempre hi haurà una situació en la qual es resolgui el cub. Però aquest nombre de situacions diferents seria $12^{21} = 4,6 \cdot 10^{22}$ on , després hauria de comprovar si el cub es troba solucionat, passant per una cadena immensa de "if". Llavors el nombre d'operacions que hauria de fer un processador per resoldre el cub seria aproximadament de $2,48 \cdot 10^{24}$, un nombre força gran, tot i que un processador d'avui dia podria efectuar amb un parell de segons. Llavors, com és que no he optat per aquest mètode de resolució?

Doncs bé, un humà seria incapàc d'efectuar totes aquestes operacions, per la qual cosa seria incapàc de resoldre el cub amb aquest mètode. Això ja és un indici de que aquest mètode utilitza molt el processament computacional però no la intel·ligència. I, com he dit anteriorment el treball està orientat a la programació intel·ligent i no a la programació de dades massives. Amb el mètode de resolució utilitzat, el codi ha de fer quatre passos, on cadascun passa per cadenes de com a màxim 50 "if", i els ha de repetir uns quants cops, per cada color, però només han de comprovar si un parell o tres de colors estan ben orientats, el que suposa com a molt unes 2500 operacions, de manera que és unes $10 \cdot 10^{20}$ vegades més ràpid de calcular, ja que aquest mètode incorpora un sistema intel·ligent.

Es podria pensar que un parell de segons no són res, i que per tant no suposa cap desavantatge el fet de triar el mètode sense la "intel·ligència artificial", però com he explicat a l'apartat de la intel·ligència artificial, aquesta matriu de dades es fa molt més extensa quan interpretarem dades inorgàniques com ho són les dades humanes, com la veu o la imatge d'una cara. Llavors aquest nombre de situacions possibles augmentaria exponencialment i no linealment, de manera que ens trobaríem amb un número tan gran que un processador podria tardar

segles a calcular. És per això que és tan important d'incorporar un sistema “intel·ligent” per poder optimitzar la solució i rebaixar el nombre d’operacions tant com sigui possible.

5. CONCLUSIÓ

Des d'un punt de vista objectiu, el treball m'ha servit per introduir-me en temes en els que sempre he estat interessat, tots d'àmbit tecnològic. He après a modelar i animar en 3D d'una manera més professional de la que havia experimentat fins el moment, i m'ha agradat molt veure el gran impacte visual que proporcionen aquestes dues aplicacions de la informàtica.

També m'he endinsat una mica més l'ús del GeoGebra. M'he adonat que tot el que puguem veure pot ser representat amb nombres i geometria. Fer els simuladors dels motors m'ha ajudat per entendre les aplicacions de les matemàtiques, les matemàtiques que no veiem però que hi són a tot arreu.

Un altre aspecte important ha estat la comprensió de com funcionen els components electrònics, tant els sensors com els motors. És més important entendre el robot que construir-lo, ja que aprendre és la base del que es pot construir en un futur, en canvi, construir és construir i prou.

El fet d'haver orientat el treball a la programació també ha suposat aprendre molt sobre aquesta. No em refereixo a aprendre un llenguatge o simplement a donar ordres a una màquina de manera ordenada i neta, sinó de veure el ventall de possibilitats que ofereix i que podrà oferir en un futur.

Conèixer noves tecnologies i softwares m'ha semblat molt interessant i m'ha fet adonar-me de l'ampli ventall que ofereix la tecnologia avui dia. Però, personalment crec que el treball de recerca no només és un treball que serveix per aprofundir coneixements. També serveix per adonar-nos de què és el que ens fascina i què és el que realment volem estudiar en un futur. Com he comentat a la introducció, des de fa anys que he estat interessat en l'àmbit de la programació, tot i que sempre l'havia orientat més a la informàtica (fer petits videojocs o programes) i no a la robòtica, com és el cas d'aquest treball.

Després d'haver acabat el treball i haver aprofundit els meus coneixements en la geometria, la modelació i animació 3D i la programació, penso que el més important no és el temari que he après, sinó els valors, o les idees que he adquirit i en les quals em vull endinsar en un futur.

Principalment, aquesta idea ha estat la del valor i la importància de la programació, i específicament de la intel·ligència artificial. La intel·ligència artificial és el procés per el qual es simula l'efecte d'interpretació de dades. Aquesta interpretació avui dia és impossible, ja que els robots només actuen en base de nombres, i no d'impulsos nerviosos que contenen formes intel·ligents, és per això que la única manera de simular una interpretació d'una dada en un robot és a través de la programació.

5.1. REFLEXIÓ SOBRE LA INTEL·LIGÈNCIA ARTIFICIAL

Tot i que un humà pot semblar molt diferent a un robot, a nivell físic, la única cosa que els diferencia és la intel·ligència. Es a dir, tots dos tenen una propietat material, tots dos tenen la capacitat d'emmagatzemar informació, i tots dos tenen la capacitat d'enviar i rebre aquesta informació, ja sigui per capacitat neuronal o computacional. L'única propietat inexistent en un robot és la intel·ligència. És per això que avui dia s'investiga sobre la intel·ligència artificial; perquè és la propietat que li falta a un robot per a que pugui ser extraordinari.

És pel meu interès per la intel·ligència artificial per el qual també he volgut orientar el meu treball a la programació. La intel·ligència artificial no té una definició pròpia, ja que existeixen molts graus d'intel·ligència artificial dependent de la seva complexitat, però en termes generals és el fet de simular una intel·ligència a través d'un codi de programació aconseguint l'efecte de que una màquina pugui entendre i interpretar unes dades i posteriorment realitzar una acció dependent de la interpretació d'aquestes dades.

El primer grau d'intel·ligència artificial es pot veure com una traducció d'una dada i no la interpretació d'aquesta, dependent del que entenguem per interpretar. Aquesta interpretació consta de convertir una dada física en una dada virtual, com per exemple el ratolí del nostre ordinador, que converteix la seva posició física en una dada informàtica, que és interpretada i materialitzada a la nostra pantalla.

El segon grau d'intel·ligència artificial és el de la interpretació de matrius, és a dir, la interpretació d'un conjunt de dades per convertir-les en un altre conjunt de dades. Aquest és el cas del meu robot, que és capaç d'interpretar unes dades (colors que ocupen el cub de Rubik), i segons la ordenació d'aquestes dades és capaç de construir una solució per al cub de Rubik, és a dir, és capaç d'interpretar aquesta matriu de dades. Aquest tipus d'intel·ligència artificial encara es pot explotar més, però sol tenir funcions lúdiques. El meu robot, a nivell pràctic no és més que una joguina.

La interpretació humana és un exemple del cas anterior però força més extens. És el que considero que avui dia és el màxim grau d'intel·ligència artificial assequible, com pot ser el reconeixement facial, o el reconeixement de veu com la coneguda Siri de Apple. Interpretar una qualitat humana no és interpretar un color. Els humans som inorgànics, irregulars, la qual cosa dificulta la nostra conversió en dades, i un cop obtenim aquestes dades, resulta complicat establir els algoritmes per convertir-les en informació útil, informació humana. És per això que aquests sistemes incorporen grans quantitats de codi i d'algoritmes força complicats d'entendre, però no són res més que el codi del meu robot portats a gran escala.

La interacció amb els humans és avui dia impossible, ja que interactuar amb un humà requereix estar al seu nivell intel·lectual, i l'única manera de que una màquina pugui arribar a ser tan intel·ligent com un humà és a través de l'autoaprenentatge, i no l'aprenentatge que li dóna un programador a través del codi. És en aquest grau, en l'autoaprenentatge, on la intel·ligència artificial podria evolucionar exponencialment, ja que l'autoaprenentatge és realment l'únic factor inexistent avui dia en un robot. Aquesta intel·ligència a nivell humà podria suposar moltes utilitats, tant a nivell físic com per exemple el servei d'un robot a una persona discapacitada, tant a nivell intel·lectual, per aprofitar aquesta intel·ligència per investigar noves tecnologies o trobar noves cures a malalties, aprofitant la intel·ligència del mateix ordinador. Cal remarcar que aquest tipus d'intel·ligència avui dia és impossible i en cas que algun dia sigui possible el més segur és que no sigui tal com ens la imaginem. També cal dir que requeriria d'un emmagatzematge de dades i una processament computacional molt més amplis dels que disposem avui dia, però per això està el futur, per a innovar i millorar.

Per comprendre el gran ventall de possibilitats que ofereix la intel·ligència artificial crec que és important portar aquesta a un estat quàntic, fora del nostre món tridimensional, i fer una reflexió.

Els humans cada cop som físicament més dèbils i intel·lectualment més potents. Si ens fixem en milers d'anys enrere, érem més forts, més resistents, teníem dents més esmolades i músculs més forts. El fet que ara siguem més dèbils és perquè la intel·ligència ens ha fet prescindir d'aquestes qualitats físiques que ja no ens són útils perquè amb la intel·ligència les hem pogut substituir. L'agricultura i la ramaderia ens van permetre prescindir de la lluita amb altres animals, i ara ja no necessitem propietats físiques per lluitar.

És un exemple molt més exagerat els dels paràsits, uns animals que van ser prou intel·ligents per descobrir que no necessitaven mobilitat per a poder alimentar-se i sobreviure, i que van anar perdent totes les qualitats físiques fins a ser el que són avui dia. Tot i que els veiem repugnants, és envejable el seu procés de desmaterialització que han efectuat al evolucionar. Avui dia només necessiten passar d'un organisme a un altre per sobreviure. No s'han de preocupar de gaire més.

I és que vist des d'un punt de vista quàntic, el cos no és més que una materialització de l'ànima, o la intel·ligència, que ens serveix per poder interactuar amb un món que som capaços de comprendre en tres dimensions. I és aquest cos el que ens limita, ens limita a un món tridimensional. Si un ésser fos capaç d'aconseguir una intel·ligència sense un cos material lligat, realment seria immensa la seva llibertat, ja que no es veuria limitada a un món tangible. I això seria possible amb la intel·ligència artificial, perquè la intel·ligència artificial seria creada sense un cos lligat. D'aquesta manera, seríem totalment invulnerables a qualsevol aspecte material, és a dir, no tindríem malalties o no necessitaríem trobar un planeta on viure per tota l'eternitat. Viuríem en un servidor, i no en un món.

Tot i que aquest aspecte pugui semblar paranoic i filosòfic, crec que representa molt bé el gran poder que suposaria la intel·ligència artificial. Crec que és una ciència amb un gran futur per davant i amb un ampli ventall d'aplicacions, la majoria de les quals no en som conscients avui dia perquè ni tan sols podem imaginar el gran poder i importància que suposaria aquesta.

6. REFERÈNCIES

6.1. WEBGRAFIA

Informació sobre el cub de Rubik:

https://es.wikipedia.org/wiki/Cubo_de_Rubik

Construcció del model:

<http://www.mindcuber.com>

Informació dels components:

<https://www.lego.com/en-us/mindstorms>

Aprenentatge de modelació i animació en 3D:

<http://www.g-blender.org/foro/>

Informació per a la programació:

<http://www.lejos.org/ev3.php> (4/07/2016)

<https://en.wikipedia.org/wiki/LeJOS> ()

Software utilitzat:

Geogebra (simulacions de motors)

Eclipse (Programació)

Java (Programació)

LDCAD (Modelació en 3D)

LEOCAD (Importació i exportació de models 3D)

Blender (Animació i modelació en 3D)

6.2. BIBLIOGRAFIA

Paral·lelament a la realització del treball, he anat llegint el llibre *Superintelligence* (Nick Bostrom), 2015, que parla sobre la intel·ligència artificial i en el cas concret del reconeixement i interpretació de dades.

7. ANNEX

El codi de programació s'estructura en 3 scripts diferents, el primer (Lectura_Cub) està dedicat a la part de reconeixement dels colors, el segon (Moviments) està dedicat a efectuar els moviments del cub d'una manera intel·ligent, i el tercer (Solució) és el conjunt d'algorítmes i condicions per els quals el robot és capaç de trobar la solució, els moviments que ha d'efectuar per resoldre el cub.

Cal dir que la major part del codi no és explicada al treball i per això pot semblar difícil d'entendre. Algunes parts del codi són mencionades durant el treball, amb el nom del script i les línies en les quals es troba.

A continuació es mostra el codi dels 3 scripts:

7.1. LECTURA_CUB

```
1 package pack;
2 import lejos.hardware.Button;
3 import lejos.hardware.ev3.LocalEV3;
4 import lejos.hardware.motor.Motor;
5 import lejos.hardware.port.Port;
6 import lejos.hardware.sensor.EV3ColorSensor;
7 import lejos.robotics.SampleProvider;
8 import lejos.utility.Delay;
9
10 public class Lectura_Cub {
11
12     EV3ColorSensor color_sensor;
13     SampleProvider conjunt_color;
14     float[] mostra_color;
15     public static int[][] cub = new int[6][9];
16
17     public static void main(String[] args) {
18         new Lectura_Cub();
19     }
20
21     public Lectura_Cub() {
22         Port s3 = LocalEV3.get().getPort("S3");
23         color_sensor = new EV3ColorSensor(s3);
24         conjunt_color = color_sensor.getRGBMode();
25         mostra_color = new float[conjunt_color.sampleSize()];
26         int[] motor_B = {0,173,110,173,110,173,110,173,110};
27         int[] motor_A = {0,135,270,405,540,675,810,945,1080};
28
29         boolean intent;
30         int intent_num;
31         Motor.B.forward();
32         while(!Motor.B.isStalled()) {}
33         Motor.B.stop();
34         Motor.B.rotate(-750);
35         Motor.C.backward();
36         while(!Motor.C.isStalled()) {}
37         Motor.C.stop();
38
39         Motor.A.setSpeed(400);
40         Motor.B.setSpeed(400);
41         Motor.C.setSpeed(450);
42         Motor.A.resetTachoCount();
43         Motor.B.resetTachoCount();
44         Motor.C.resetTachoCount();
45     }
```

```

46   for(int cara =0;cara<6;cara++){
47
48     for(int lloc=0;lloc<9;lloc++){
49       intent = false;
50       intent_num = 0;
51
52       Motor.A.rotateTo(motor_A[lloc],true);
53       Motor.B.rotateTo(motor_B[lloc]);
54       System.out.println(Motor.B.getTachoCount());
55       while(intent == false){
56         if(Button.ESCAPE.isDown()){
57           System.exit(0);
58         }
59         conjunt_color.fetchSample(mostra_color, 0);
60
61         float r =
62         Math.round(mostra_color[0]/(mostra_color[0]+mostra_color[1]+mostra_color[2])*10000.0f)/1
63         00.0f;
64         float g =
65         Math.round(mostra_color[1]/(mostra_color[0]+mostra_color[1]+mostra_color[2])*10000.0f)/1
66         00.0f;
67         float b =
68         Math.round(mostra_color[2]/(mostra_color[0]+mostra_color[1]+mostra_color[2])*10000.0f)/1
69         00.0f;
70
71         intent_num ++ ;
72         if (24.3f<b && b<28.3f && 37.2f<g && g<41.2f && 32.4f<r && r<36.4f ){
73           cub[cara][lloc] = 0;
74           intent = true;
75           //blanc
76         }
77         if (9.75f<b && b<14.75f && 26.4f<g && g<30.4f && 58f<r && r<62f ){
78           cub[cara][lloc] = 1;
79           intent = true;
80           //vermell
81         }
82         if (45f<b && b<51f && 37.1f<g && g<43.1f && 10f<r && r<16f ){
83           cub[cara][lloc] = 2;
84           intent = true;
85           //blau
86         }
87         if (24.3f<r && r<27f && 35.5f<g && g<39.5f && 34.5f<b && b<38f ){
88           cub[cara][lloc] = 3;
89           intent = true;
90           //taronja
91         }
92         if (14.43f<r && r<18.43f && 41.6f<g && g<46.6f && 37.9f<b && b<42.9f ){
93           cub[cara][lloc] = 4;
94           intent = true;
95           //verd
96         }
97         if (27f<r && r<31f && 33f<g && g<37f && 34f<b && b<38f ){
98           cub[cara][lloc] = 5;
99           intent = true;
100          //groc
101        }
102        if(intent == false){
103          if(intent_num%2==0){
104            Motor.B.rotate(-intent_num*3);
105          }
106          if(intent_num%2==1){
107            Motor.B.rotate(intent_num*3);
108          }
109          if(intent_num%4==0){
110            Motor.A.rotate(intent_num);
111          }
112          if(intent_num%4==2){
113            Motor.A.rotate(-intent_num);
114          }
115        }
116        Delay.msDelay(100);
117      }
118    }
119    Motor.B.rotateTo(300);
120
121    if(cara==1||cara==2||cara==3){
122      Motor.A.rotate(270);

```

```
123     }
124     Motor.C.rotateTo(200);
125     Motor.C.rotateTo(0);
126     if(cara==1||cara==2||cara==3||cara==4){
127         Motor.A.rotate(-270);
128     }
129     Motor.A.resetTachoCount();
130 }
131 Motor.C.rotateTo(200);
132 Motor.C.rotateTo(0);
133 Motor.A.rotate(-270);
134 color_sensor.close();
135 Moviments.Adjunta_Colors();
136
137 while(Button.ESCAPE.isUp()){
138 }
139 new Solucio();
```

7.2. MOVIMENTS

```
1 package pack;
2
3 import lejos.hardware.motor.Motor;
4
5 public class Moviments {
6
7     public static int[][] cub = new int[6][9];
8     public static int[][] cub_i = new int[6][9];
9
10    static int[] centres = {0,1,2,3,4,5};
11
12    public static void main(String[] args) {
13        new Moviments();
14    }
15
16    public Moviments(){
17        Adjunta_Colors();
18    }
19
20    public static void Adjunta_Colors(){
21        for(int cara=0;cara<6;cara++){
22            cub[cara][0] = Lectura_Cub.cub[cara][5];
23            cub[cara][1] = Lectura_Cub.cub[cara][4];
24            cub[cara][2] = Lectura_Cub.cub[cara][3];
25            cub[cara][3] = Lectura_Cub.cub[cara][6];
26            cub[cara][4] = Lectura_Cub.cub[cara][0];
27            cub[cara][5] = Lectura_Cub.cub[cara][2];
28            cub[cara][6] = Lectura_Cub.cub[cara][7];
29            cub[cara][7] = Lectura_Cub.cub[cara][8];
30            cub[cara][8] = Lectura_Cub.cub[cara][1];
31            for(int a=0;a<9;a++){
32                System.out.print(cub[cara][a]);
33            }
34        }
35    }
36
37
38    public static void Cara(int color, int quantitat, boolean sentit){
39        Motor.C.setSpeed(450);
40        int lloc = 0;
41        for (int i=0;i<6;i++){
42            if (centres[i]==color){
43                lloc = i;
44            }
45        }
46        switch(lloc){
47            case 0:
48                Frontal();
49                Frontal();
50                break;
51            case 1:
52                Dreta();
53                Frontal();
54                break;
55            case 2:
56                Frontal();
57                break;
58            case 3:
59                Esquerra();
60                Frontal();
61                break;
62            case 4:
63                Dreta();
64                Dreta();
65                Frontal();
66                break;
67            case 5:
68                break;
69        }
70        Base(sentit,quantitat,color);
71    }
72}
```

```

73 static void Base(boolean sentit,int quantitat,int color){
74
75     if(Motor.C.getTachoCount ()<70){
76         Motor.C.rotateTo(120);
77     }
78     for(int i=0;i<quantitat;i++){
79         Corona(color,sentit);
80         RotaCara(color,sentit);
81         if(sentit){
82             Motor.A.rotate(-275);
83             Motor.A.rotate(5);
84         }else{
85             Motor.A.rotate(275);
86             Motor.A.rotate(-5);
87         }
88     }
89 }
90
91 static void Dreta(){
92     if(Motor.C.getTachoCount ()>5){
93         Motor.C.rotateTo(0);
94     }
95     int[] copia = new int[6];
96     System.arraycopy(centres, 0, copia, 0, 6);
97     centres[1] = copia[4];
98     centres[2] = copia[1];
99     centres[3] = copia[2];
100    centres[4] = copia[3];
101    Motor.A.rotate(-270);
102 }
103
104 static void Esquerra(){
105     if(Motor.C.getTachoCount ()>5){
106         Motor.C.rotateTo(0);
107     }
108     int[] copia = new int[6];
109     System.arraycopy(centres, 0, copia, 0, 6);
110     centres[1] = copia[2];
111     centres[2] = copia[3];
112     centres[3] = copia[4];
113     centres[4] = copia[1];
114     Motor.A.rotate(270);
115 }
116
117 static void Frontal(){
118     int[] copia = new int[6];
119     System.arraycopy(centres, 0, copia, 0, 6);
120     centres[0] = copia[4];
121     centres[2] = copia[0];
122     centres[4] = copia[5];
123     centres[5] = copia[2];
124     Motor.C.rotateTo(200);
125     Motor.C.rotateTo(120);
126 }
127
128 static void Corona(int color,boolean sentit){
129     int i;
130     if(sentit){
131         i = -3;
132     }else{
133         i = 3;
134     }
135     switch(color){
136     case 0:
137         int[] copia_0 =
138 {cub[4][2],cub[4][1],cub[4][0],cub[3][2],cub[3][1],cub[3][0],cub[2][2],cub[2][1],cub[2][0],cub[1][2],cub[1][1],cub[1][0]},;
139         cub[4][2] = copia_0[(12+i)%12];
140         cub[4][1] = copia_0[(13+i)%12];
141         cub[4][0] = copia_0[(14+i)%12];
142         cub[3][2] = copia_0[(15+i)%12];
143         cub[3][1] = copia_0[(16+i)%12];
144         cub[3][0] = copia_0[(17+i)%12];
145         cub[2][2] = copia_0[(18+i)%12];
146         cub[2][1] = copia_0[(19+i)%12];
147         cub[2][0] = copia_0[(20+i)%12];
148         cub[1][2] = copia_0[(21+i)%12];
149

```

```

150     cub[1][1] = copia_0[(22+i)%12];
151     cub[1][0] = copia_0[(23+i)%12];
152     break;
153 case 1:
154     int[] copia_1 =
155 {cub[0][6],cub[0][7],cub[0][8],cub[2][0],cub[2][3],cub[2][6],cub[5][2],cub[5][1],cub[5][
156 0],cub[4][8],cub[4][5],cub[4][2]};
157     cub[0][6] = copia_1[(12+i)%12];
158     cub[0][7] = copia_1[(13+i)%12];
159     cub[0][8] = copia_1[(14+i)%12];
160     cub[2][0] = copia_1[(15+i)%12];
161     cub[2][3] = copia_1[(16+i)%12];
162     cub[2][6] = copia_1[(17+i)%12];
163     cub[5][2] = copia_1[(18+i)%12];
164     cub[5][1] = copia_1[(19+i)%12];
165     cub[5][0] = copia_1[(20+i)%12];
166     cub[4][8] = copia_1[(21+i)%12];
167     cub[4][5] = copia_1[(22+i)%12];
168     cub[4][2] = copia_1[(23+i)%12];
169     break;
170 case 2:
171     int[] copia_2 =
172 {cub[0][8],cub[0][5],cub[0][2],cub[3][0],cub[3][3],cub[3][6],cub[5][8],cub[5][5],cub[5][
173 2],cub[1][8],cub[1][5],cub[1][2]};
174     cub[0][8] = copia_2[(12+i)%12];
175     cub[0][5] = copia_2[(13+i)%12];
176     cub[0][2] = copia_2[(14+i)%12];
177     cub[3][0] = copia_2[(15+i)%12];
178     cub[3][3] = copia_2[(16+i)%12];
179     cub[3][6] = copia_2[(17+i)%12];
180     cub[5][8] = copia_2[(18+i)%12];
181     cub[5][5] = copia_2[(19+i)%12];
182     cub[5][2] = copia_2[(20+i)%12];
183     cub[1][8] = copia_2[(21+i)%12];
184     cub[1][5] = copia_2[(22+i)%12];
185     cub[1][2] = copia_2[(23+i)%12];
186     break;
187 case 3:
188     int[] copia_3 =
189 {cub[0][2],cub[0][1],cub[0][0],cub[4][0],cub[4][3],cub[4][6],cub[5][6],cub[5][7],cub[5][
190 8],cub[2][8],cub[2][5],cub[2][2]};
191     cub[0][2] = copia_3[(12+i)%12];
192     cub[0][1] = copia_3[(13+i)%12];
193     cub[0][0] = copia_3[(14+i)%12];
194     cub[4][0] = copia_3[(15+i)%12];
195     cub[4][3] = copia_3[(16+i)%12];
196     cub[4][6] = copia_3[(17+i)%12];
197     cub[5][6] = copia_3[(18+i)%12];
198     cub[5][7] = copia_3[(19+i)%12];
199     cub[5][8] = copia_3[(20+i)%12];
200     cub[2][8] = copia_3[(21+i)%12];
201     cub[2][5] = copia_3[(22+i)%12];
202     cub[2][2] = copia_3[(23+i)%12];
203     break;
204 case 4:
205     int[] copia_4 =
206 {cub[0][0],cub[0][3],cub[0][6],cub[1][0],cub[1][3],cub[1][6],cub[5][0],cub[5][3],cub[5][
207 6],cub[3][8],cub[3][5],cub[3][2]};
208     cub[0][0] = copia_4[(12+i)%12];
209     cub[0][3] = copia_4[(13+i)%12];
210     cub[0][6] = copia_4[(14+i)%12];
211     cub[1][0] = copia_4[(15+i)%12];
212     cub[1][3] = copia_4[(16+i)%12];
213     cub[1][6] = copia_4[(17+i)%12];
214     cub[5][0] = copia_4[(18+i)%12];
215     cub[5][3] = copia_4[(19+i)%12];
216     cub[5][6] = copia_4[(20+i)%12];
217     cub[3][8] = copia_4[(21+i)%12];
218     cub[3][5] = copia_4[(22+i)%12];
219     cub[3][2] = copia_4[(23+i)%12];
220     break;
221 case 5:
222     int[] copia_5 =
223 {cub[1][6],cub[1][7],cub[1][8],cub[2][6],cub[2][7],cub[2][8],cub[3][6],cub[3][7],cub[3][
224 8],cub[4][6],cub[4][7],cub[4][8]};
225     cub[1][6] = copia_5[(12+i)%12];
226     cub[1][7] = copia_5[(13+i)%12];

```

```

227     cub[1][8] = copia_5[(14+i)%12];
228     cub[2][6] = copia_5[(15+i)%12];
229     cub[2][7] = copia_5[(16+i)%12];
230     cub[2][8] = copia_5[(17+i)%12];
231     cub[3][6] = copia_5[(18+i)%12];
232     cub[3][7] = copia_5[(19+i)%12];
233     cub[3][8] = copia_5[(20+i)%12];
234     cub[4][6] = copia_5[(21+i)%12];
235     cub[4][7] = copia_5[(22+i)%12];
236     cub[4][8] = copia_5[(23+i)%12];
237     break;
238 }
239 }
240
241 static void RotaCara(int color, boolean sentit){
242     int[] copia = new int[9];
243     System.arraycopy(cub[color], 0, copia, 0, 9);
244     if(sentit){
245         cub[color][0] = copia[6];
246         cub[color][1] = copia[3];
247         cub[color][2] = copia[0];
248         cub[color][3] = copia[7];
249         cub[color][5] = copia[1];
250         cub[color][6] = copia[8];
251         cub[color][7] = copia[5];
252         cub[color][8] = copia[2];
253     }else{
254         cub[color][0] = copia[2];
255         cub[color][1] = copia[5];
256         cub[color][2] = copia[8];
257         cub[color][3] = copia[1];
258         cub[color][5] = copia[7];
259         cub[color][6] = copia[0];
260         cub[color][7] = copia[3];
261         cub[color][8] = copia[6];
262     }
263 }
264 }
```

7.3. SOLUCIÓ

```
1 package pack;
2
3 import java.util.Arrays;
4
5 import lejos.hardware.Button;
6 import lejos.hardware.motor.Motor;
7 import lejos.utility.Delay;
8
9 public class Solucio {
10     public static int[] color ={0,1,2,3,4,5};
11
12     public static void main(String[] args) {
13         // TODO Auto-generated method stub
14         new Solucio();
15     }
16
17     public Solucio(){
18         Pas_1();
19         Pas_2();
20         Pas_3();
21         Pas_4();
22     }
23
24
25     void Pas_1(){
26
27         for(int i=0;i<4;i++){
28             if(Arrays.equals(arestes_sup(color[1]-1), new int[]{color[0],color[1]})){
29                 System.out.println(1);
30             }
31             if(Arrays.equals(arestes_sup(color[2]-1), new int[]{color[0],color[1]})){
32                 Transforma(new String[]{"R2","d","F2"});
33             }
34             if(Arrays.equals(arestes_sup(color[3]-1), new int[]{color[0],color[1]})){
35                 Transforma(new String[]{"B2","D2","F2"});
36             }
37             if(Arrays.equals(arestes_sup(color[4]-1), new int[]{color[0],color[1]})){
38                 Transforma(new String[]{"L2","D","F2"});
39             }
40             if(Arrays.equals(arestes_mig(color[1]-1), new int[]{color[0],color[1]})){
41                 Transforma(new String[]{"u","R","U"});
42             }
43             if(Arrays.equals(arestes_mig(color[2]-1), new int[]{color[0],color[1]})){
44                 Transforma(new String[]{"U2","B","U2"});
45             }
46             if(Arrays.equals(arestes_mig(color[3]-1), new int[]{color[0],color[1]})){
47                 Transforma(new String[]{"U","L","u"});
48             }
49             if(Arrays.equals(arestes_mig(color[4]-1), new int[]{color[0],color[1]})){
50                 Transforma(new String[]{"F"});
51             }
52             if(Arrays.equals(arestes_inf(color[1]-1), new int[]{color[0],color[1]})){
53                 Transforma(new String[]{"f","u","R","U"});
54             }
55             if(Arrays.equals(arestes_inf(color[2]-1), new int[]{color[0],color[1]})){
56                 Transforma(new String[]{"R","f","r"});
57             }
58             if(Arrays.equals(arestes_inf(color[3]-1), new int[]{color[0],color[1]})){
59                 Transforma(new String[]{"d","R","f","r"});
60             }
61             if(Arrays.equals(arestes_inf(color[4]-1), new int[]{color[0],color[1]})){
62                 Transforma(new String[]{"l","F","L"});
63             }
64             if(Arrays.equals(arestes_sup(color[1]-1), new int[]{color[1],color[0]})){
65                 Transforma(new String[]{"F","u","R","U"});
66             }
67             if(Arrays.equals(arestes_sup(color[2]-1), new int[]{color[1],color[0]})){
68                 Transforma(new String[]{"r","f"});
69             }
70             if(Arrays.equals(arestes_sup(color[3]-1), new int[]{color[1],color[0]})){
71                 Transforma(new String[]{"B","U","L","u"});
72             }
73         }
74     }
75 }
```

```

73     if(Groups.equals(arestes_sup(color[4]-1), new int[]{color[1],color[0]})){
74         Transforma(new String[]{"L","F"});
75     }
76     if(Groups.equals(arestes_mig(color[1]-1), new int[]{color[1],color[0]})){
77         Transforma(new String[]{"f"});
78     }
79     if(Groups.equals(arestes_mig(color[2]-1), new int[]{color[1],color[0]})){
80         Transforma(new String[]{"u","r","U"});
81     }
82     if(Groups.equals(arestes_mig(color[3]-1), new int[]{color[1],color[0]})){
83         Transforma(new String[]{"U","b","U2"});
84     }
85     if(Groups.equals(arestes_mig(color[4]-1), new int[]{color[1],color[0]})){
86         Transforma(new String[]{"U","l","u"});
87     }
88     if(Groups.equals(arestes_inf(color[1]-1), new int[]{color[1],color[0]})){
89         Transforma(new String[]{"F2"});
90     }
91     if(Groups.equals(arestes_inf(color[2]-1), new int[]{color[1],color[0]})){
92         Transforma(new String[]{"d","F2"});
93     }
94     if(Groups.equals(arestes_inf(color[3]-1), new int[]{color[1],color[0]})){
95         Transforma(new String[]{"D2","F2"});
96     }
97     if(Groups.equals(arestes_inf(color[4]-1), new int[]{color[1],color[0]})){
98         Transforma(new String[]{"D","F2"});
99     }
100    System.arraycopy(rota_valors(color,true), 0, color, 0, 6);
101    while(Button.ESCAPE.isUp()){
102    }
103  }
104}
105
106
107 void Pas_2(){
108   int[] colors_2 = {5,1,4,3,2,0};
109   System.arraycopy(colors_2, 0, color, 0, 6);
110   for(int i =0;i<4;i++){
111     if(conte_valors(arestes_mig(color[3]-1),new int[]{color[1],color[2]})){
112       Transforma(new String[]{"B","U","b"});
113     }
114     if(conte_valors(arestes_mig(color[4]-1),new int[]{color[1],color[2]})){
115       Transforma(new String[]{"L","U","l"});
116     }
117     if(conte_valors(arestes_mig(color[1]-1),new int[]{color[1],color[2]})){
118       Transforma(new String[]{"F","u","f"});
119     }
120     if(conte_valors(cantonades_sup(color[3]-1),new int[]{color[5],color[1],color[2]})){
121       Transforma(new String[]{"B","U","b"});
122     }
123     if(conte_valors(cantonades_sup(color[4]-1),new int[]{color[5],color[1],color[2]})){
124       Transforma(new String[]{"L","U2","l"});
125     }
126     if(conte_valors(cantonades_sup(color[1]-1),new int[]{color[5],color[1],color[2]})){
127       Transforma(new String[]{"l","U","L"});
128     }
129     if(conte_valors(cantonades_inf(color[3]-1),new int[]{color[5],color[1],color[2]})){
130       Transforma(new String[]{"U"});
131     }
132     if(conte_valors(cantonades_inf(color[4]-1),new int[]{color[5],color[1],color[2]})){
133       Transforma(new String[]{"U2"});
134     }
135     if(conte_valors(cantonades_inf(color[1]-1),new int[]{color[5],color[1],color[2]})){
136       Transforma(new String[]{"u"});
137     }
138     if(conte_valors(cantonades_sup(color[2]-1),new int[]{color[5],color[1],color[2]})){
139       if(Groups.equals(arestes_inf(color[1]-1), new int[]{color[2],color[1]})){
140         Transforma(new String[]{"u"});
141       }
142       if(Groups.equals(arestes_inf(color[2]-1), new int[]{color[1],color[2]})){
143         Transforma(new String[]{"U"});
144       }
145       if(Groups.equals(arestes_inf(color[3]-1), new int[]{color[1],color[2]})){
146         Transforma(new String[]{"U2"});
147       }
148       if(Groups.equals(arestes_inf(color[3]-1), new int[]{color[2],color[1]})){
149         Transforma(new String[]{"U"});
150       }
151     }
152   }
153 }
```

```

150
151     }
152     if(Groups.equals(arestes_inf(color[4]-1), new int[]{color[1],color[2]})){
153         Transforma(new String[]{"u"});
154     }
155     if(Groups.equals(arestes_inf(color[4]-1), new int[]{color[2],color[1]})){
156         Transforma(new String[]{"U2"});
157     }
158     while(Button.ESCAPE.isUp()){
159     }
160
161     if(Groups.equals(cantonades_sup(color[2]-1), new
162 int[]{color[5],color[2],color[1]})){
163         if(Groups.equals(arestes_mig(color[2]-1), new int[]{color[1],color[2]})){
164             Transforma(new String[]{"R2","U2","F","R2","f","U2","r","U","r"});
165         }
166         if(Groups.equals(arestes_inf(color[1]-1), new int[]{color[1],color[2]})){
167             Transforma(new String[]{"U","R","U","r","u","f","u","F"});
168         }
169         if(Groups.equals(arestes_inf(color[2]-1), new int[]{color[2],color[1]})){
170             Transforma(new String[]{"u","f","u","F","U","R","U","r"});
171         }
172         if(Groups.equals(cantonades_sup(color[2]-1), new
173 int[]{color[2],color[1],color[5]})){
174             if(Groups.equals(arestes_mig(color[2]-1), new int[]{color[2],color[1]})){
175                 Transforma(new String[]{"R2","U2","r","u","R","u","r","U2","r"});
176             }
177             if(Groups.equals(arestes_mig(color[2]-1), new int[]{color[1],color[2]})){
178                 Transforma(new String[]{"f","l","U2","L","F","R","U","r"});
179             }
180             if(Groups.equals(arestes_inf(color[1]-1), new int[]{color[1],color[2]})){
181                 Transforma(new String[]{"f","u","F","U","f","u","F"});
182             }
183             if(Groups.equals(arestes_inf(color[2]-1), new int[]{color[2],color[1]})){
184                 Transforma(new String[]{"R","u","r","U","R","u","r"});
185             }
186         }
187
188         if(Groups.equals(cantonades_sup(color[2]-1), new
189 int[]{color[1],color[5],color[2]})){
190             if(Groups.equals(arestes_mig(color[2]-1), new int[]{color[2],color[1]})){
191                 Transforma(new String[]{"F2","U2","F","U","f","U","F","U2","F"});
192             }
193             if(Groups.equals(arestes_mig(color[2]-1), new int[]{color[1],color[2]})){
194                 Transforma(new String[]{"R","u","r","f","l","U2","L","F"});
195             }
196             if(Groups.equals(arestes_inf(color[1]-1), new int[]{color[1],color[2]})){
197                 Transforma(new String[]{"f","U","F","u","f","U","F"});
198             }
199             if(Groups.equals(arestes_inf(color[2]-1), new int[]{color[2],color[1]})){
200                 Transforma(new String[]{"R","U","r","u","R","U","r"});
201             }
202         }
203
204         if(Groups.equals(cantonades_inf(color[2]-1), new
205 int[]{color[5],color[1],color[2]})){
206             if(Groups.equals(arestes_mig(color[2]-1), new int[]{color[2],color[1]})){
207                 Transforma(new String[]{"U","f","U","F","U","f","U2","F"});
208             }
209             if(Groups.equals(arestes_mig(color[2]-1), new int[]{color[1],color[2]})){
210                 Transforma(new String[]{"U","f","u","F","u","R","U","r"});
211             }
212             if(Groups.equals(arestes_inf(color[1]-1), new int[]{color[1],color[2]})){
213                 Transforma(new String[]{"u","f","U","F"});
214             }
215             if(Groups.equals(arestes_inf(color[1]-1), new int[]{color[2],color[1]})){
216                 Transforma(new String[]{"r","U2","R2","U","R2","U","R"});
217             }
218             if(Groups.equals(arestes_inf(color[4]-1), new int[]{color[1],color[2]})){
219                 Transforma(new String[]{"R2","B","U","b","u","R2"});
220             }
221             if(Groups.equals(arestes_inf(color[4]-1), new int[]{color[2],color[1]})){
222                 Transforma(new String[]{"u","R","U","r","U","R","U","r"});
223             }
224             if(Groups.equals(arestes_inf(color[3]-1), new int[]{color[1],color[2]})){
225                 Transforma(new String[]{"f","u","l","U2","L","u","F"});
226             }
}

```

```

227     if(Groups.equals(arestes_inf(color[3]-1), new int[]{color[2],color[1]})){
228         Transforma(new String[]{"R","U","r"});
229     }
230     if(Groups.equals(arestes_inf(color[2]-1), new int[]{color[1],color[2]})){
231         Transforma(new String[]{"f","l","b","U","L","F"});
232     }
233     if(Groups.equals(arestes_inf(color[2]-1), new int[]{color[2],color[1]})){
234         Transforma(new String[]{"u","R","u","r","U","R","U","r"});
235     }
236 }
237
238     if(Groups.equals(cantonades_inf(color[2]-1), new
239 int[]{color[2],color[5],color[1]})){
240         if(Groups.equals(arestes_mig(color[2]-1), new int[]{color[2],color[1]})){
241             Transforma(new String[]{"b","D","B","u","b","d","B"});
242         }
243         if(Groups.equals(arestes_mig(color[2]-1), new int[]{color[1],color[2]})){
244             Transforma(new String[]{"u","R","U","r","U","f","u","F"});
245         }
246         if(Groups.equals(arestes_inf(color[1]-1), new int[]{color[1],color[2]})){
247             Transforma(new String[]{"U","f","U","F","u","f","u","F"});
248         }
249         if(Groups.equals(arestes_inf(color[1]-1), new int[]{color[2],color[1]})){
250             Transforma(new String[]{"f","U","F","U2","R","U","r"});
251         }
252         if(Groups.equals(arestes_inf(color[4]-1), new int[]{color[1],color[2]})){
253             Transforma(new String[]{"f","u","F"});
254         }
255         if(Groups.equals(arestes_inf(color[4]-1), new int[]{color[2],color[1]})){
256             Transforma(new String[]{"F2","U2","r","F2","R","U2","F2"});
257         }
258         if(Groups.equals(arestes_inf(color[3]-1), new int[]{color[1],color[2]})){
259             Transforma(new String[]{"R2","U","B","u","b","R2"});
260         }
261         if(Groups.equals(arestes_inf(color[3]-1), new int[]{color[2],color[1]})){
262             Transforma(new String[]{"F2","l","u","L","U","F2"});
263         }
264         if(Groups.equals(arestes_inf(color[2]-1), new int[]{color[1],color[2]})){
265             Transforma(new String[]{"F","U2","F2","u","F2","u","f"});
266         }
267         if(Groups.equals(arestes_inf(color[2]-1), new int[]{color[2],color[1]})){
268             Transforma(new String[]{"U","R","u","r"});
269         }
270     }
271
272     if(Groups.equals(cantonades_inf(color[2]-1), new
273 int[]{color[1],color[2],color[5]})){
274         if(Groups.equals(arestes_mig(color[2]-1), new int[]{color[2],color[1]})){
275             Transforma(new String[]{"R","U","r","u","R","U","r","u","R","U","r"});
276         }
277         if(Groups.equals(arestes_mig(color[2]-1), new int[]{color[1],color[2]})){
278             Transforma(new String[]{"f","U","F","R","U2","r"});
279         }
280         if(Groups.equals(arestes_inf(color[1]-1), new int[]{color[1],color[2]})){
281             Transforma(new String[]{"f","U2","F","U","f","u","F"});
282         }
283         if(Groups.equals(arestes_inf(color[1]-1), new int[]{color[2],color[1]})){
284             Transforma(new String[]{"U2","R2","U2","r","u","R","u","R2"});
285         }
286         if(Groups.equals(arestes_inf(color[4]-1), new int[]{color[1],color[2]})){
287             Transforma(new String[]{"u","f","U2","F","u","f","U","F"});
288         }
289         if(Groups.equals(arestes_inf(color[4]-1), new int[]{color[2],color[1]})){
290             Transforma(new String[]{"U2","R","U","r","U","R","u","r"});
291         }
292         if(Groups.equals(arestes_inf(color[3]-1), new int[]{color[1],color[2]})){
293             Transforma(new String[]{"U2","f","u","F","u","f","U","F"});
294         }
295         if(Groups.equals(arestes_inf(color[3]-1), new int[]{color[2],color[1]})){
296             Transforma(new String[]{"U","R","U2","r","U","R","u","r"});
297         }
298         if(Groups.equals(arestes_inf(color[2]-1), new int[]{color[1],color[2]})){
299             Transforma(new String[]{"f","u","f","L","F","l","U2","F"});
300         }
301         if(Groups.equals(arestes_inf(color[2]-1), new int[]{color[2],color[1]})){
302             Transforma(new String[]{"R","U2","r","u","R","U","r"});
303         }
}

```

```

304
305     System.arraycopy(rota_valors(color,true), 0, color, 0, 6);
306     while(Button.ESCAPE.isUp()){
307     }
308   }
309 }
310
311 void Pas_3(){
312   boolean solucio = false;
313   while(solucio==false){
314     if(oll(new int[]{12,13,14,15,16,17,18,19,20})){
315     }
316     if(oll(new int[]{0,2,13,15,16,17,18,19,20})){
317       Transforma(new String[]{"r","U2","R","F","u","r","u","R","U","f"});
318     }
319     if(oll(new int[]{2,6,12,13,15,16,17,18,19})){
320       Transforma(new String[]{"r","f","L","F","R","f","l","F"});
321     }
322     if(oll(new int[]{2,9,12,13,15,16,17,19,20})){
323       Transforma(new String[]{"r","f","l","F","R","f","L","F"});
324     }
325     if(oll(new int[]{0,3,6,13,15,16,17,18,19})){
326       Transforma(new String[]{"R","U","r","U","R","U2","r"});
327     }
328     if(oll(new int[]{2,8,11,13,15,16,17,19,20})){
329       Transforma(new String[]{"l","u","L","u","l","U2","L"});
330     }
331     if(oll(new int[]{2,6,9,11,13,15,16,17,19})){
332       Transforma(new String[]{"R","U2","R2","u","R2","u","R2","U2","R"});
333     }
334     if(oll(new int[]{0,2,6,8,13,15,16,17,19})){
335       Transforma(new String[]{"R","U2","r","u","R","U","r","u","R","u","r"});
336     }
337   //totes les cantonades orientades
338   if(oll(new int[]{4,7,12,13,14,15,16,18,20})){
339     Transforma(new String[]{"L","F","r","f","l","R","U","R","u","r"});
340   }
341   if(oll(new int[]{1,7,12,14,15,16,17,18,20})){
342     Transforma(new String[]{"R","U","r","u","L","r","F","R","f","l"});
343   }
344   //cap aresta orientada
345   if(oll(new int[]{1,4,7,10,12,14,16,18,20})){
346     Transforma(new String[]{"l","R","B","R","B","r","b","L2","R2","F","R","f","l"});
347   }
348   if(oll(new int[]{0,1,3,4,6,7,10,16,18})){
349     Transforma(new String[]{"l","R2","B","r","B","L","U2","l","B","r","L"});
350   }
351   if(oll(new int[]{1,2,4,7,8,10,11,16,20})){
352     Transforma(new String[]{"l","R","b","L","U2","l","b","R","b","R2","L"});
353   }
354   if(oll(new int[]{0,1,4,5,7,10,14,16,18})){
355     Transforma(new String[]{"l","B","L","b","U2","b","U","r","U","R","B"});
356   }
357   if(oll(new int[]{1,4,5,7,9,10,12,14,16})){
358     Transforma(new String[]{"l","B2","R","B","r","B","L2","F2","r","f","R","f","l"});
359   }
360   if(oll(new int[]{0,1,2,4,7,10,16,18,20})){
361     Transforma(new String[]{"F","R","U","r","U","f","U2","f","L","F","l"});
362   }
363   if(oll(new int[]{1,2,4,6,7,9,10,11,16})){
364     Transforma(new String[]{"F","R","U","r","u","f","B","U","L","u","l","b"});
365   }
366   if(oll(new int[]{1,3,4,5,7,9,10,11,16})){
367     Transforma(new String[]{"R","U2","R2","F","R","f","U2","r","F","R","f"});
368   }
369   //forma de T
370   if(oll(new int[]{0,1,7,8,14,15,16,17,20})){
371     Transforma(new String[]{"R","U","r","u","r","F","R","f"});
372   }
373   if(oll(new int[]{1,7,9,11,14,15,16,17,20})){
374     Transforma(new String[]{"F","R","U","r","u","f"});
375   }
376   //forma de I
377   if(oll(new int[]{0,1,3,5,7,8,15,16,17})){
378     Transforma(new String[]{"F","U","R","u","r","U","R","u","r","f"});
379   }
380   if(oll(new int[]{0,3,4,5,8,10,13,16,19})){

```

```

381     Transforma(new String[]{"r","u","R","u","r","U","f","U","F","R"});
382 }
383 if(oll(new int[]{1,3,5,7,9,11,15,16,17})){
384     Transforma(new
385 String[]{"l","b","L","u","r","U","R","u","r","U","R","l","B","L"});
386 }
387 if(oll(new int[]{3,4,5,9,10,11,13,16,19})){
388     Transforma(new String[]{"r","U2","R2","U","r","U","R","U2","b","r","B"});
389 }
390 //forma de L
391 if(oll(new int[]{0,2,4,6,7,8,13,15,16})){
392     Transforma(new String[]{"L","F2","r","f","R","F","r","f","R","f","l"});
393 }
394 if(oll(new int[]{0,1,2,4,6,8,15,16,19})){
395     Transforma(new String[]{"l","B2","R","B","r","b","R","B","r","B","L"});
396 }
397 if(oll(new int[]{0,3,4,5,7,8,13,15,16})){
398     Transforma(new String[]{"r","b","D2","b","D2","B","R2","B","r"});
399 }
400 if(oll(new int[]{0,1,3,4,5,8,15,16,19})){
401     Transforma(new String[]{"R","f","U2","F","U2","F","R2","f","R"});
402 }
403 if(oll(new int[]{2,4,6,7,9,11,13,15,16})){
404     Transforma(new String[]{"F","R","U","r","u","R","U","r","u","f"});
405 }
406 if(oll(new int[]{0,3,5,7,8,10,13,16,17})){
407     Transforma(new String[]{"f","l","u","L","U","l","u","L","U","F"});
408 }
409 //forma de W
410 if(oll(new int[]{0,4,5,7,13,14,15,16,18})){
411     Transforma(new String[]{"R","U","r","U","R","u","r","u","r","F","R","f"});
412 }
413 if(oll(new int[]{1,3,4,8,12,15,16,19,20})){
414     Transforma(new String[]{"r","u","R","u","r","U","R","U","R","b","r","B"});
415 }
416 //forma de C
417 if(oll(new int[]{1,3,7,11,15,16,17,18,20})){
418     Transforma(new String[]{"R","U","R2","u","r","F","R","U","R","u","f"});
419 }
420 if(oll(new int[]{4,9,10,11,13,15,16,19,20})){
421     Transforma(new String[]{"L","U","L","f","l","F","u","l"});
422 }
423 //forma de P
424 if(oll(new int[]{1,9,10,11,14,16,17,19,20})){
425     Transforma(new String[]{"B","U","L","u","l","b"});
426 }
427 if(oll(new int[]{1,3,4,5,12,15,16,18,19})){
428     Transforma(new String[]{"b","u","r","U","R","B"});
429 }
430 if(oll(new int[]{2,4,6,7,12,13,15,16,18})){
431     Transforma(new String[]{"L","U","f","u","l","U","L","F","l"});
432 }
433 if(oll(new int[]{0,7,8,10,13,14,16,17,20})){
434     Transforma(new String[]{"r","u","F","U","R","u","r","f","R"});
435 }
436 //forma de peix
437 if(oll(new int[]{0,1,4,5,14,15,16,18,19})){
438     Transforma(new String[]{"R","b","r","B","U","B","u","b"});
439 }
440 if(oll(new int[]{0,5,7,10,13,14,16,17,18})){
441     Transforma(new String[]{"r","U2","R2","b","r","B","r","U2","R"});
442 }
443 if(oll(new int[]{0,1,4,6,9,14,15,16,19})){
444     Transforma(new String[]{"R","U","r","b","R","B","u","b","r","B"});
445 }
446 if(oll(new int[]{1,2,5,8,10,12,16,17,19})){
447     Transforma(new String[]{"l","u","L","B","l","b","U","B","L","b"});
448 }
449 //forma de quadrat
450 if(oll(new int[]{0,1,3,9,10,16,17,19,20})){
451     Transforma(new String[]{"l","B2","R","B","r","B","L"});
452 }
453 if(oll(new int[]{5,7,8,10,11,13,14,16,17})){
454     Transforma(new String[]{"L","F2","r","f","R","f","l"});
455 }
456 //forma de raig
457

```

```

458     if(oll(new int[]{0,1,5,7,14,15,16,17,18})){
459         Transforma(new String[]{"L","f","l","u","L","U","F","u","l"});
460     }
461     if(oll(new int[]{1,2,7,9,12,15,16,17,20})){
462         Transforma(new String[]{"r","F","R","U","r","u","f","U","R"});
463     }
464     if(oll(new int[]{0,3,4,6,7,13,15,16,18})){
465         Transforma(new String[]{"L","F","r","F","R","F2","l"});
466     }
467     if(oll(new int[]{1,2,4,5,8,12,15,16,19})){
468         Transforma(new String[]{"l","b","R","b","r","B2","L"});
469     }
470     if(oll(new int[]{1,2,4,8,11,15,16,19,20})){
471         Transforma(new String[]{"R","L2","b","L","b","l","B2","L","b","L","r"});
472     }
473     if(oll(new int[]{0,1,3,6,10,16,17,18,19})){
474         Transforma(new String[]{"l","R2","B","r","B","R","B2","r","B","L","r"});
475     }
476     //forma de L
477     Transforma(new String[]{"R","b","r","u","R","B","r","b","U","B"});
478 }
479 if(oll(new int[]{1,2,5,7,8,12,15,16,17})){
480     Transforma(new String[]{"l","B","L","U","l","b","L","B","u","b"});
481 }
482 if(oll(new int[]{1,5,7,8,11,14,15,16,17})){
483     Transforma(new String[]{"L","F","l","R","U","r","u","L","f","l"});
484 }
485 if(oll(new int[]{1,3,6,7,9,12,15,16,17})){
486     Transforma(new String[]{"r","f","R","l","u","L","U","r","F","R"});
487 }
488 if(oll(new int[]{1,6,8,10,12,14,15,16,19})){
489     Transforma(new String[]{"R","u","r","U2","R","U","B","u","b","u","r"});
490 }
491 if(oll(new int[]{1,4,6,8,12,14,15,16,19})){
492     Transforma(new String[]{"l","U","L","U2","l","u","b","U","B","U","L"});
493 }
494 if(oll(new int[]{2,6,7,10,12,13,16,17,18})){
495     Transforma(new String[]{"l","u","L","U","l","U","L","F","U","f","l","u","L"});
496 }
497 if(oll(new int[]{0,4,7,8,13,14,15,16,20})){
498     Transforma(new String[]{"R","U","r","u","R","u","r","f","u","F","R","U","r"});
499 }
500 if(solucio==false){
501     Moviments.Cara(5, 1, true);
502 }
503 while(Button.ESCAPE.isUp()){
504 }
505 }
506 }
507
508 void Pas_4(){
509     boolean solucio=false;
510
511     while(solucio==false){
512         for(int i=0;i<4;i++){
513             if(pll(new
514 int[]{color[1],color[2],color[1],color[4],color[1],color[3],color[2],color[4],color[4],c
515 olor[3],color[3],color[2]})){
516                 Reset();
517                 Transforma(new String[]{"r","F","r","B2","R","f","r","B2","R2"});
518             }
519             if(pll(new
520 int[]{color[4],color[2],color[3],color[2],color[1],color[2],color[1],color[4],color[4],c
521 olor[3],color[3],color[1]})){
522                 Reset();
523                 Transforma(new String[]{"R2","B2","R","F","r","B2","R","f","R"});
524             }
525             if(pll(new
526 int[]{color[1],color[4],color[3],color[2],color[3],color[4],color[3],color[2],color[1],c
527 olor[4],color[1],color[2]})){
528                 Reset();
529                 Transforma(new
530 String[]{"R","b","r","F","R","B","r","F2","l","B","L","F","l","b","L"});
531             }
532             if(pll(new
533 int[]{color[1],color[4],color[1],color[4],color[1],color[4],color[3],color[2],color[3],c
534 olor[2],color[3],color[2]})){

```

```

535     Reset();
536     Transforma(new
537 String[]{"r","L","F","R2","L2","B","R2","L2","F","r","L","D2","R2","L2"});
538   }
539   if(pll(new
540 int[]{color[2],color[4],color[2],color[1],color[3],color[1],color[4],color[2],color[4],c
541 olor[3],color[1],color[3]})){
542     Reset();
543     Transforma(new String[]{"R2","L2","d","R2","L2","U2","R2","L2","d","R2","L2"});
544   }
545   if(pll(new
546 int[]{color[2],color[2],color[2],color[1],color[3],color[1],color[4],color[1],color[4],c
547 olor[3],color[4],color[3]})){
548     Reset();
549     Transforma(new String[]{"R","u","R","U","R","U","R","u","r","u","R2"});
550   }
551   if(pll(new
552 int[]{color[2],color[1],color[2],color[1],color[3],color[1],color[4],color[4],color[4],c
553 olor[3],color[2],color[3]})){
554     Reset();
555     Transforma(new String[]{"r","U","r","u","r","u","r","U","R","U","R2"});
556   }
557
558   if(pll(new
559 int[]{color[2],color[1],color[1],color[4],color[2],color[2],color[1],color[4],color[4],c
560 olor[3],color[3],color[3]})){
561     Reset();
562     Transforma(new String[]{"F2","l","u","L","F2","r","D","r","d","R2"});
563   }
564   if(pll(new
565 int[]{color[2],color[1],color[4],color[4],color[2],color[1],color[1],color[4],color[4],c
566 olor[3],color[3],color[3]})){
567     Reset();
568     Transforma(new String[]{"R","U2","r","u","R","U2","l","U","r","u","L"});
569   }
570   if(pll(new
571 int[]{color[2],color[1],color[4],color[3],color[2],color[1],color[4],color[4],color[4],c
572 olor[3],color[1],color[3]})){
573     Reset();
574     Transforma(new
575 String[]{"R","U","r","u","r","F","R2","u","r","u","R","U","r","f"});
576   }
577   if(pll(new
578 int[]{color[1],color[2],color[3],color[2],color[4],color[1],color[4],color[1],color[4],c
579 olor[3],color[3],color[2]})){
580     Reset();
581     Transforma(new
582 String[]{"r","U2","R","U2","r","F","R","U","r","u","r","f","R2","u"});
583   }
584   if(pll(new
585 int[]{color[1],color[2],color[3],color[2],color[1],color[1],color[4],color[3],color[4],c
586 olor[3],color[4],color[2]})){
587     Reset();
588     Transforma(new
589 String[]{"l","U2","l","U2","L","f","l","u","L","U","L","F","L2","U"});
590   }
591   if(pll(new
592 int[]{color[1],color[2],color[3],color[2],color[3],color[1],color[4],color[4],color[4],c
593 olor[3],color[1],color[2]})){
594     Reset();
595     Transforma(new
596 String[]{"r","U","R","u","R2","f","u","F","U","R","F","r","f","R2","u"});
597   }
598
599   if(pll(new
600 int[]{color[4],color[3],color[2],color[1],color[2],color[4],color[3],color[4],color[3],c
601 olor[2],color[1],color[1]})){
602     Reset();
603     Transforma(new
604 String[]{"U2","F2","d","L","u","L","U","l","D","F2","R","u","r"});
605   }
606   if(pll(new
607 int[]{color[3],color[2],color[3],color[2],color[4],color[1],color[4],color[3],color[2],c
608 olor[1],color[1],color[4]})){
609     Reset();
610     Transforma(new String[]{"F2","D","r","U","r","u","R","d","F2","l","U","L"});
611   }

```

```

612     if(pll(new
613 int[]{color[1],color[2],color[4],color[3],color[1],color[3],color[2],color[3],color[1],c
614 olor[4],color[4],color[2]})){
615         Reset();
616         Transforma(new String[]{"l","u","L","F2","D","r","U","R","u","R","d","F2"});
617     }
618     if(pll(new
619 int[]{color[2],color[4],color[1],color[4],color[2],color[1],color[3],color[4],c
620 olor[3],color[1],color[3]})){
621         Reset();
622         Transforma(new String[]{"R","U","r","F2","d","L","u","l","U","l","D","F2"});
623     }
624     if(pll(new
625 int[]{color[3],color[4],color[1],color[4],color[1],color[2],color[1],color[3],color[4],c
626 olor[2],color[2],color[4]})){
627         Reset();
628         Transforma(new
629 String[]{"r","U","r","u","b","D","b","d","B2","r","b","R","B","R"});
630     }
631     if(pll(new
632 int[]{color[2],color[4],color[3],color[3],color[1],color[4],color[4],color[2],c
633 olor[1],color[1],color[3]})){
634         Reset();
635         Transforma(new
636 String[]{"r","U","r","U","R","U","r","f","R","U","r","u","r","F","R2","u","r","U2","R",
637 "u","r"});
638     }
639     if(pll(new
640 int[]{color[4],color[2],color[2],color[1],color[3],color[2],color[4],color[4],c
641 olor[3],color[1],color[1]})){
642         Reset();
643         Transforma(new
644 String[]{"r","U","R","u","r","f","u","F","R","U","r","F","r","f","R","u","R"});
645     }
646     if(pll(new
647 int[]{color[1],color[3],color[2],color[2],color[4],color[3],color[1],color[1],c
648 olor[4],color[3],color[2]})){
649         Reset();
650         Transforma(new
651 String[]{"R2","u","r","U","R","u","b","r","F","r","f","r","B","R"});
652     }
653     System.arraycopy(rota_valors(color,true), 0, color, 0, 6);
654     Moviments.Cara(5, 1, true);
655 }
656 if(solucio==false){
657     Moviments.Cara(5, 1, true);
658 }
659 while(Button.ESCAPE.isUp()){
660 }
661 }
662 }
663 }
664
//aquest metode es fa servir per saber si dos arrays contenen els mateixos valors, tot
665 i que estiguin desordenats, de manera que es pugui saber la posició de la peça sense
666 saber-ne la orientació.
667 boolean conte_valors (int[] arr1,int[]arr2){
668     //S'ordenen els dos arrays de valors petits a grans, de manera que si conenen els
669     //mateixos valors i s'ordenen, seran arrays iguals.
670     java.util.Arrays.sort(arr1);
671     java.util.Arrays.sort(arr2);
672     //Es retorna verdader si els dos arrays ordenats són iguals, i fals si no ho son
673     return Arrays.equals(arr1, arr2);
674 }
675
676
677 int[] cantonades_sup(int num){
678     int[] resultat = new int[3];
679     switch(num){
680     case 0:
681         int[] resultat_0 = {Moviments.cub[0][8],Moviments.cub[1][2],Moviments.cub[2][0]};
682         System.arraycopy(resultat_0, 0, resultat, 0, 3);
683         break;
684     case 1:
685         int[] resultat_1 = {Moviments.cub[0][2],Moviments.cub[2][2],Moviments.cub[3][0]};
686         System.arraycopy(resultat_1, 0, resultat, 0, 3);
687         break;
688     case 2:

```

```

689     int[] resultat_2 = {Moviments.cub[0][0],Moviments.cub[3][2],Moviments.cub[4][0]};
690     System.arraycopy(resultat_2, 0, resultat, 0, 3);
691     break;
692 case 3:
693     int[] resultat_3 = {Moviments.cub[0][6],Moviments.cub[4][2],Moviments.cub[1][0]};
694     System.arraycopy(resultat_3, 0, resultat, 0, 3);
695     break;
696 }
697 return resultat;
698 }
699
700 int[] cantonades_inf(int num){
701     int[] resultat = new int[3];
702     switch(num){
703 case 0:
704     int[] resultat_0 = {Moviments.cub[1][8],Moviments.cub[2][6],Moviments.cub[5][2]};
705     System.arraycopy(resultat_0, 0, resultat, 0, 3);
706     break;
707 case 1:
708     int[] resultat_1 = {Moviments.cub[2][8],Moviments.cub[3][6],Moviments.cub[5][8]};
709     System.arraycopy(resultat_1, 0, resultat, 0, 3);
710     break;
711 case 2:
712     int[] resultat_2 = {Moviments.cub[3][8],Moviments.cub[4][6],Moviments.cub[5][6]};
713     System.arraycopy(resultat_2, 0, resultat, 0, 3);
714     break;
715 case 3:
716     int[] resultat_3 = {Moviments.cub[4][8],Moviments.cub[1][6],Moviments.cub[5][0]};
717     System.arraycopy(resultat_3, 0, resultat, 0, 3);
718     break;
719 }
720 return resultat;
721 }
722
723 int[] arrestes_sup(int num){
724     int[] resultat = new int[2];
725     switch(num){
726 case 0:
727     int[] resultat_0 = {Moviments.cub[0][7],Moviments.cub[1][1]};
728     System.arraycopy(resultat_0, 0, resultat, 0, 2);
729     break;
730 case 1:
731     int[] resultat_1 = {Moviments.cub[0][5],Moviments.cub[2][1]};
732     System.arraycopy(resultat_1, 0, resultat, 0, 2);
733     break;
734 case 2:
735     int[] resultat_2 = {Moviments.cub[0][1],Moviments.cub[3][1]};
736     System.arraycopy(resultat_2, 0, resultat, 0, 2);
737     break;
738 case 3:
739     int[] resultat_3 = {Moviments.cub[0][3],Moviments.cub[4][1]};
740     System.arraycopy(resultat_3, 0, resultat, 0, 2);
741     break;
742 }
743 return resultat;
744 }
745 int[] arrestes_inf(int num){
746     int[] resultat = new int[2];
747     switch(num){
748 case 0:
749     int[] resultat_0 = {Moviments.cub[1][7],Moviments.cub[5][1]};
750     System.arraycopy(resultat_0, 0, resultat, 0, 2);
751     break;
752 case 1:
753     int[] resultat_1 = {Moviments.cub[2][7],Moviments.cub[5][5]};
754     System.arraycopy(resultat_1, 0, resultat, 0, 2);
755     break;
756 case 2:
757     int[] resultat_2 = {Moviments.cub[3][7],Moviments.cub[5][7]};
758     System.arraycopy(resultat_2, 0, resultat, 0, 2);
759     break;
760 case 3:
761     int[] resultat_3 = {Moviments.cub[4][7],Moviments.cub[5][3]};
762     System.arraycopy(resultat_3, 0, resultat, 0, 2);
763     break;
764 }
765 return resultat;

```

```

766     }
767
768     int[] arrestes_mig(int num){
769         int[] resultat = new int[2];
770         switch(num){
771             case 0:
772                 int[] resultat_0 = {Moviments.cub[1][5],Moviments.cub[2][3]};
773                 System.arraycopy(resultat_0, 0, resultat, 0, 2);
774                 break;
775             case 1:
776                 int[] resultat_1 = {Moviments.cub[2][5],Moviments.cub[3][3]};
777                 System.arraycopy(resultat_1, 0, resultat, 0, 2);
778                 break;
779             case 2:
780                 int[] resultat_2 = {Moviments.cub[3][5],Moviments.cub[4][3]};
781                 System.arraycopy(resultat_2, 0, resultat, 0, 2);
782                 break;
783             case 3:
784                 int[] resultat_3 = {Moviments.cub[4][5],Moviments.cub[1][3]};
785                 System.arraycopy(resultat_3, 0, resultat, 0, 2);
786                 break;
787         }
788         return resultat;
789     }
790
791     int[] rota_valors(int[] cares,boolean sentit){
792         int[] resultat = new int[6];
793         if(sentit){
794             resultat[0] = cares[0];
795             resultat[1] = cares[2];
796             resultat[2] = cares[3];
797             resultat[3] = cares[4];
798             resultat[4] = cares[1];
799             resultat[5] = cares[5];
800         }else{
801             resultat[0] = cares[0];
802             resultat[1] = cares[4];
803             resultat[2] = cares[1];
804             resultat[3] = cares[2];
805             resultat[4] = cares[3];
806             resultat[5] = cares[5];
807         }
808         return resultat;
809     }
810
811
812     void Transfoma(String[] hola){
813         int color_ = 0;
814         int quantitat = 1;
815         boolean sentit = false;
816         for (int i=0;i<hola.length;i++){
817             switch(hola[i].charAt(0)){
818                 case 'U':
819                     color_ = color[0];
820                     sentit = true;
821                     break;
822                 case 'F':
823                     color_ = color[1];
824                     sentit = true;
825                     break;
826                 case 'R':
827                     color_ = color[2];
828                     sentit = true;
829                     break;
830                 case 'B':
831                     color_ = color[3];
832                     sentit = true;
833                     break;
834                 case 'L':
835                     color_ = color[4];
836                     sentit = true;
837                     break;
838                 case 'D':
839                     color_ = color[5];
840                     sentit = true;
841                     break;
842                 case 'u':

```

```

843         color_ = color[0];
844         sentit = false;
845         break;
846     case 'f':
847         color_ = color[1];
848         sentit = false;
849         break;
850     case 'r':
851         color_ = color[2];
852         sentit = false;
853         break;
854     case 'b':
855         color_ = color[3];
856         sentit = false;
857         break;
858     case 'l':
859         color_ = color[4];
860         sentit = false;
861         break;
862     case 'd':
863         color_ = color[5];
864         sentit = false;
865         break;
866     }
867     if(hola[i].length()>1){
868         quantitat = 2;
869     }else{
870         quantitat=1;
871     }
872     Moviments.Cara(color_, quantitat, sentit);
873 }
874 }
875
876 boolean oll (int[] valors){
877     int[] copias =
878 {Moviments.cub[3][6],Moviments.cub[3][7],Moviments.cub[3][8],Moviments.cub[4][6],Movimen-
879 ts.cub[4][7],Moviments.cub[4][8],Moviments.cub[1][6],Moviments.cub[1][7],Moviments.cub[1]
880 ][8],Moviments.cub[2][6],Moviments.cub[2][7],Moviments.cub[2][8],Moviments.cub[5][8],Mov-
881 iments.cub[5][7],Moviments.cub[5][6],Moviments.cub[5][5],Moviments.cub[5][4],Moviments.c-
882 ub[5][3],Moviments.cub[5][2],Moviments.cub[5][1],Moviments.cub[5][0]};
883     boolean algoritme = true;
884     for(int i = 0;i<9;i++){
885         if(copias[valors[i]]!=5){
886             algoritme = false;
887         }
888     }
889     return algoritme;
890 }
891
892 boolean pll (int[] valors){
893     int[] copias =
894 {Moviments.cub[3][6],Moviments.cub[3][7],Moviments.cub[3][8],Moviments.cub[4][6],Movimen-
895 ts.cub[4][7],Moviments.cub[4][8],Moviments.cub[1][6],Moviments.cub[1][7],Moviments.cub[1]
896 ][8],Moviments.cub[2][6],Moviments.cub[2][7],Moviments.cub[2][8]};
897     boolean algoritme = true;
898     for(int i = 0;i<12;i++){
899         if(copias[i]!=valors[i]){
900             algoritme = false;
901         }
902     }
903     return algoritme;
904 }
905
906 void Reset(){
907     color[0] = 5;
908     color[1] = 1;
909     color[2] = 4;
910     color[3] = 3;
911     color[4] = 2;
912     color[5] = 0;
913 }
914 }
```

