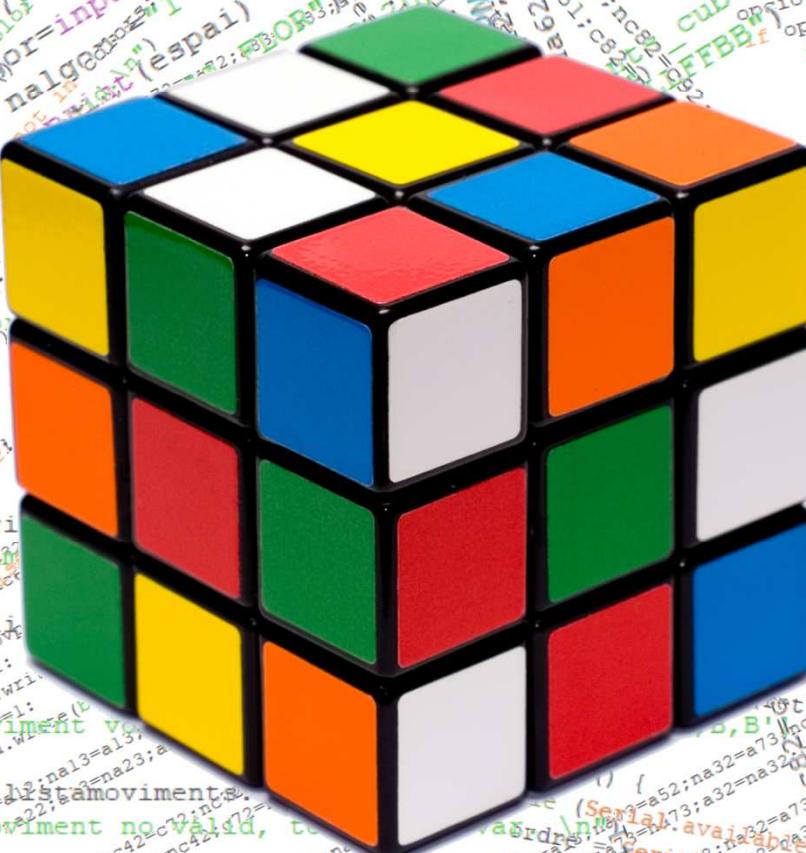


# Programació, disseny i muntatge d'un robot capaç de resoldre el cub de Rubik



Marín Gargallo, Pol – 2n Batx A

Damunt Masip, Xavier

19/12/2018

## Índex

Introducció.....	3
Per què vaig escollir aquest tema?.....	3
Què esperava aprendre?.....	4
Procés tecnològic .....	5
Detecció de colors .....	6
Espectre de colors.....	9
Resolució virtual del cub.....	10
Moviment del cub.....	11
Diferents apartats del programa.....	14
Procés de resolució del cub virtual .....	19
Resolució física del cub.....	27
Muntatge.....	28
Impressió 3D .....	29
Pressupost.....	30
Propostes de continuïtat.....	30
Conclusions .....	31
Webgrafia .....	32
Agraïments .....	33

## Annex

Annex A. Estudi del número de moviments.....	2
Annex B. Mètode Fridrich.....	4
Annex C. Com instal•lar mòduls al Python.....	5
Annex D. Funcionament dels motors pas a pas .....	9

## Introducció

### Per què vaig escollir aquest tema?

Des de feia un temps, m'agradava passar l'estona amb cubs de Rubik. M'havia comprat uns quants de diversos colors i formes. Sempre m'havia semblat una jocuina molt interessant, ja que no només és entretinguda sinó que també et fa pensar. Moltes vegades quan no tenia res a fer o m'avorria, agafava un cub i passava l'estona.

Un dia, a l'estiu de l'any passat, estava pensant sobre batxillerat mentre tenia un cub a la mà. Sabia que venia una nova etapa molt important en la meva vida acadèmica, i havia una cosa que em cridava especialment l'atenció: el treball de recerca. No sabia de que el volia fer, l'únic que tenia clar era que havia de ser sobre algun tema que m'agradés molt, i que per molt que estigui moltes hores amb el mateix no em cansés massa, o almenys se'm fes més amè. Doncs vaig pensar que seria interessant fer-ho sobre el cub de Rubik, que em sembla molt interessant; sent una jocuina tan senzilla amagava moltes curiositats, matemàtiques i possibilitats.

Havia vist algun vídeo de robots que el resolien, alguns fins i tot en poc més d'un segon. Em semblava impressionant i vaig pensar en com molaria fer una màquina així. Sincerament, no em veia capaç de fer-ho, tots els projectes eren de d'universitaris o d'enginyers ja consagrats. No vaig pensar més.

Uns mesos és tard, ja cursant primer de batxillerat, una de les assignatures que més em van agradar va ser tecnologia, però específicament aquells dies que feiem programació amb Python. Sempre m'havia cridat molt l'atenció el món de la programació i els llenguatges informàtics, però mai havia sabut com començar. Gràcies a aquestes classes vaig poder endinsar-me en un món que em va enganxar des del moment en que ho vaig provar. Sempre que podia provava de fer coses noves amb el Python, no eren gran cosa, però m'encantava. Quan havíem d'entregar deures de programació no només em sortien amb facilitat sinó que sempre li acabava posant coses de més perquè m'entretenia.

Un dia, vam començar a parlar a classe sobre el Treball de Recerca, i ens van ensenyar les propostes que feien des del centre. No vaig trobar cap tema que em crides especialment l'atenció. Havia algun tema interessant però estava convençut de que si els agafava m'acabarien avorrint i amargant l'estiu i el principi de segon. Jo volia un tema que m'entusiasmés en tal mesura que no m'importe passar-me hores i hores dedicant-m'hi. M'agradaria utilitzar i aprendre nous llenguatges de programació, però quins? per a fer què? Podia aprofitar el treball per a dur a terme la idea del cub de Rubik , però sonava massa idí·lica.

Me'n recordo d'un dia, als passadissos, amb els meus amics, parlant sobre el famós 'TdR' que en unes setmanes començaria a formar part del nostre dia a dia. Imaginàvem el que ens encantaria fer, fantasiejàvem una mica. I jo ho vaig dir: "Doncs a mi m'encantaria poder fer una màquina que resolgués el cub de Rubik, però seria una currada brutal". A tots ens semblava una idea increïble, però poc realista. No em veia capaç de fer-ho. Quant més pensava, més inconvenients i dificultats aparentment insalvables se m'ocorrien.

Tot i així, no em vaig desanimar, inclús em vaig apropar un dia al Xavi, qui més tard va esdevenir el meu tutor del treball, i li vaig dir que agafaria la opció de '*Programació amb Python 3*' però que tenia una idea en ment que anava més enllà. Pensava que tot quedaria en això, una idea, un altre projecte que em feia il·lusió i no arribava a res. M'imaginava que m'acabaria dient que tries una altra opció menys esbojarrada i més realista.

El principi del treball va ser una mica difús. Em vaig posar a programar amb Python, i em vaig plantejar un repte, programar un cub. Només un cub. 6 cares, 9 peces per cara. I que es pogués moure, simular el moviment al menys. Quan el vaig començar tampoc vaig pensar que s'acabaria convertint de veritat en el meu treball. Realment, ja tenia mig assumit que acabaria fent una altra cosa, potser més fàcil i assequible. El que volia fer era simplement reptar-me a mi mateix. No sabia com començar, així que vaig anar provant coses i maneres de fer. Vaig trobar una manera que més o menys funcionava, és la que vaig tirar endavant i explico en aquest treball. Python no era la millor opció per a fer això ja que no està tan orientat a objectes com altres llenguatges com Java per exemple. Tot i així, simulava el moviment. I tarda rere tarda, vaig fer-lo més o menys funcional en moviments. Cada cop veia menys difús el treball i vaig anar avançant.

Va arribar un punt que vaig decidir que sí que ho faria sobre el cub de Rubik, i vaig posar-me més seriosament i vaig ensenyar al Xavi el que havia fet. Poquet a poquet, dia rere dia, el programa amb Python va anar agafant forma.

Molts mesos més tard, amb tot ja fet, sento una forta emoció al provar tot els programes que havia fet i resoldre el cub per primera vegada al menjador de casa. Ho havia aconseguit!

## Què esperava aprendre?

Vaigaprofitar aquest projecte com a excusa per a reptar-me a mi mateix. Era conscient de que era un projecte bastant ambiciós, i que cobria moltes àrees de la tecnologia. El principal que volia aprendre era nous llenguatges de programació, i ser capaç de millorar en els que ja coneixia. També esperava aprendre a controlar motors i a utilitzar Arduino.

## Procés tecnològic

La idea que tenia per dur a terme el projecte va ser segmentar-ho en tres parts, la primera encarregada de detecció de color, la segona resolució virtual del cub i la tercera en físic. També havia de pensar com les connectaria entre elles.

El procés tecnològic està format principalment de tres etapes:

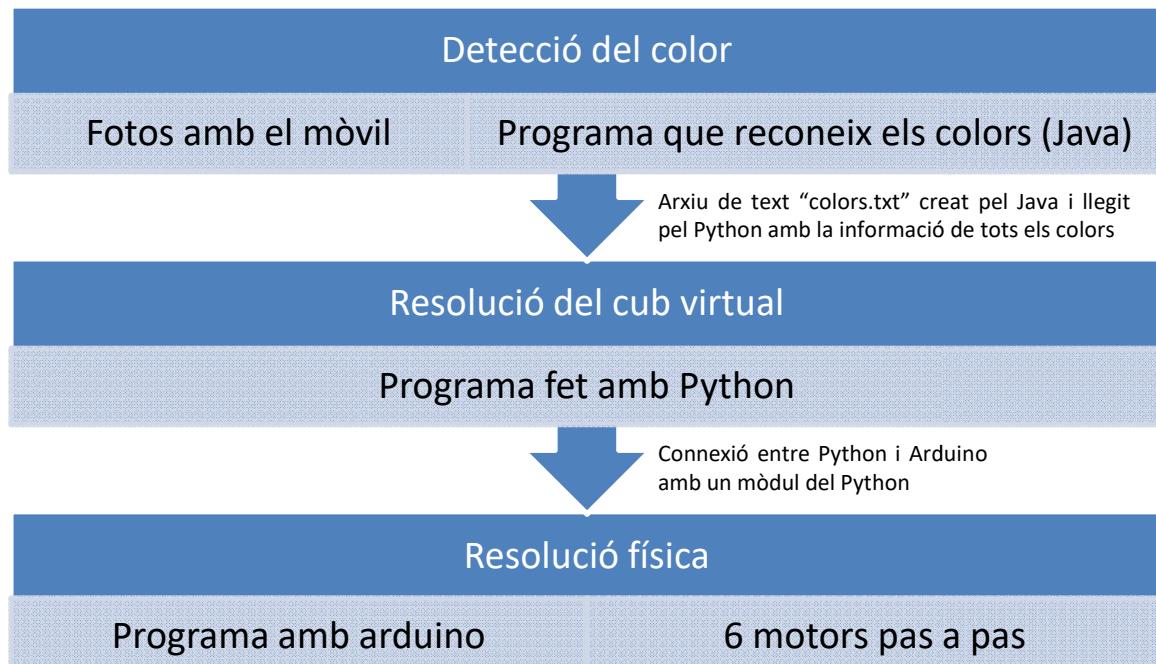
- Detecció dels colors
- Resolució virtual del cub
- Resolució física del cub

El procés de detecció de color és el que s'encarrega de la obtenció d'informació, en aquest cas es tracta dels colors de cada una de les cares.

La resolució virtual s'encarrega de, tenint en compte la distribució de colors, donar les indicacions necessàries per poder resoldre el cub.

La resolució física és la fase en la que partint dels moviments establerts en la fase anterior, es resol el cub físic.

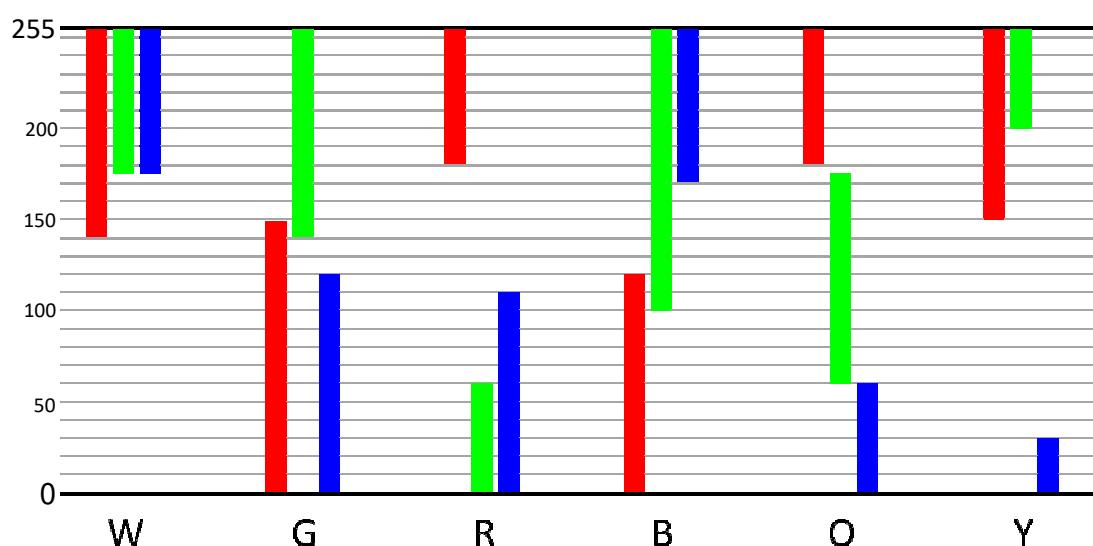
Apart d'aquests tres processos principals, el procés tecnològic consta de més processos intermediaris i secundaris, com podrien ser el fer les fotos de les cares del cub o transmetre les dades entre els programes.



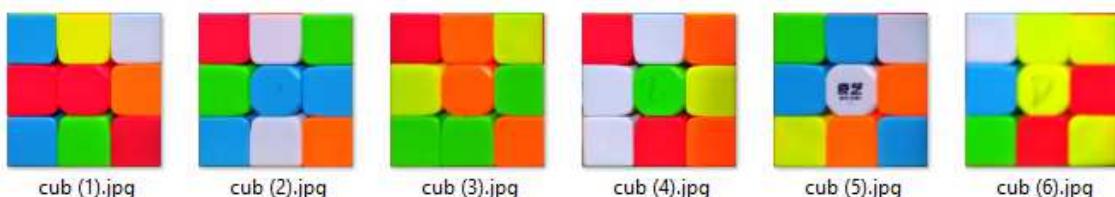
## Detecció de colors

Aquesta part del procés la vaig fer amb el programa Processing, utilitzant el llenguatge de Java. Aquest programa agafa 6 imatges (una de cada cara), de cada cara detecta el RGB de 9 punts diferents (cadascun dels quadradets) i depenen del respectiu RGB el classifica com un color o un altre dels 6 possibles (blanc, vermell, blau, taronja, verd i groc).

	Blanc (W)	Verd (G)	Vermell (R)	Blau (B)	Taronja (O)	Groc (Y)
R	140-255	0-149	180-255	0-120	180-255	150-255
G	175-255	140-255	0-60	100-255	60-175	200-255
B	175-255	0-120	0-110	170-255	0-60	0-30



Les imatges que rep el programa són com aquestes:



Estan fetes amb un dispositiu mòbil, utilitzant la aplicació *Office Lens*. Amb aquesta aplicació, la imatge s'ajusta sola a la forma del cub, no cal retallar res. A més es poden fer totes les fotos seguides dins del mateix bloc, a l'hora de guardar el conjunt de fotos, es diu que es guardin en format d'imatge i no PDF i es posa el nom “cub” ; automàticament el programa les guarda totes amb aquests noms. S'han de fer en un ordre prèviament establert (R,B,O,G,W,Y) per a que el programa al obrir-les funcioni correctament. Aquestes fotos seran guardades a la carpeta “Almacenamiento interno/Pictures/Office Lens” del dispositiu mòbil, el qual ha d'estar connectat per cable al ordinador per al ràpid traspàs de les fotografies.

Vista de la aplicación Office Lens des de la pantalla de l'Smartphone



Aquest és el codi del programa. Quan l'executes ensenya una imatge que acaba de crear amb els colors de les fotos de les cares prèviament fetes.

```

1 PImage cara;
2 String col;
3 int mult;
4
5 int EW=0;
6 int ER=0;
7 int EB=0;
8 int EO=0;
9 int EG=0;
10 int EY=0;
11
12 void setup() {
13     size(220, 315);
14     Line(0,155,width,155);
15 }
16
17 void draw() {
18     loadPixels();
19     for (int i = 1; i <= 6; i++) {
20         for (int j = 0; j < 3; j++) {
21             for (int k = 0; k<3; k++) {
22                 cara = loadImage("cub ("+i+").jpg");
23
24                 float R1 = red(cara.pixels[((cara.width/6)+cara.width/24+j*cara.width/3)+cara.width*(cara.height/6+k*cara.height/3)]);
25                 float R2 = red(cara.pixels[((cara.width/6)-cara.width/24+j*cara.width/3)+cara.width*(cara.height/6+k*cara.height/3)]);
26                 float R0 = red(cara.pixels[((cara.width/6)+j*cara.width/3)+cara.width*(cara.height/6+k*cara.height/3)]);
27                 float R = (R0+R1+R2)/3;
28
29                 float G0 = green(cara.pixels[((cara.width/6)+cara.width/24+j*cara.width/3)+cara.width*(cara.height/6+k*cara.height/3)]);
30                 float G1 = green(cara.pixels[((cara.width/6)-cara.width/24+j*cara.width/3)+cara.width*(cara.height/6+k*cara.height/3)]);
31                 float G2 = green(cara.pixels[((cara.width/6)+j*cara.width/3)+cara.width*((cara.height/6+cara.height/15)+k*cara.height/3)]);
32                 float G = (G0+G1+G2)/3;
33
34                 float B0 = blue(cara.pixels[((cara.width/6)+cara.width/24+j*cara.width/3)+cara.width*(cara.height/6+k*cara.height/3)]);
35                 float B1 = blue(cara.pixels[((cara.width/6)-cara.width/24+j*cara.width/3)+cara.width*(cara.height/6+k*cara.height/3)]);
36                 float B2 = blue(cara.pixels[((cara.width/6)+j*cara.width/3)+cara.width*(cara.height/6+k*cara.height/3)]);
37                 float B = (B1+B2+B0)/3;
38
39                 if (j==1 & k==1) {
40                     mult = 1;
41                 } else {
42                     mult = 0;
43                 }
44             }
45
46             fill(R,G,B);
47             if (i < 5) {
48                 ellipse(10+55*(i-1)+15*j,60+ 15*k,10+2*mult,10+2*mult);
49             }
50             if (i == 5) {
51                 ellipse(10+15*j,10+15*k,10+2*mult,10+2*mult);
52             }
53             if (i == 6) {
54                 ellipse(10+15*j,110+15*k,10+2*mult,10+2*mult);
55             }
}

```

Declaració d'algunes variables

Extreu el vermell, verd i blau de cadascun dels quadradets de cada cara.

Agafa en tres punts diferents de cada quadrat i fa la mitja per a tenir un millor resultat

Aquesta part no és necessària, l'únic que fa és que els centres de cada cara es vegin més grans que els altres quadradets.

Es crea un conjunt de cercles, cadascun representa un quadrat del cub. Cada cercle té el color que s'ha extret directament de la imatge.

Gràcies al `for()` he pogut fer tot el procés una vegada i que es repeteixi per a cada quadradet en comptes de haver de repetir tot el procés per a cada un.

Hi ha tres `for()`, un per cada cara i els altres dos per a les columnes i files.

Quan executes aquests programa, es crea un arxiu de text anomenat “*colors.txt*” que, com el nom indica, té escrits tots els colors del cub, per a ser llegit pel programa que ho resoldrà.

## Programació, disseny i muntatge d'un robot capaç de resoldre el cub de Rubik

```

56   fill(0);
57   if (R>=140 &&G>=175 &&B>=175) {
58     col="W";
59     EW++;
60     fill(255);
61   }
62   if (R>=180 && G>=60 && G<=175 && B<=60) {
63     col="O";
64     EO++;
65     fill(255,130,0);
66   }
67   if (R<=120 && G>=100 && B>=170) {
68     col="B";
69     EB++;
70     fill(0,0,255);
71   }
72   if (R>=150 && G>=200 && B<=30) {
73     col="Y";
74     EY++;
75     fill(255,255,0);
76   }
77   if (R<=149 && G>=140 && B<=120) {
78     col="G";
79     EG++;
80     fill(0,255,0);
81   }
82   if (R>=180 && G<=60 && B<=110) {
83     col="R";
84     ER++;
85     fill(255,0,0);
86   }
87
88   if (i < 5) {
89     ellipse(10+55*(i-1)+15*j,160+60+ 15*k,10+2*mult,10+2*mult);
90   }
91   if (i == 5) {
92     ellipse(10+15*j,160+10+15*k,10+2*mult,10+2*mult);
93   }
94   if (i == 6) {
95     ellipse(10+15*j,160+110+15*k,10+2*mult,10+2*mult);
96   }
97 }
98 }
99 }
100
101 if (EW>=8 && EY==9 && EG==9 && ER==9 && EB==9 && EO==9) {
102   stroke(0,255,0);
103   strokeWeight(5);
104   noFill();
105   ellipse(width-30,height-30,30,30);
106 }
107 else {
108   stroke(255,0,0);
109   strokeWeight(5);
110   line(width-45,height-45,width-10,height-10);
111   line(width-10,height-45,width-45,height-10);
112 }
113 saveFrame("Detecció color.jpg");
114 noLoop();
115 }

```

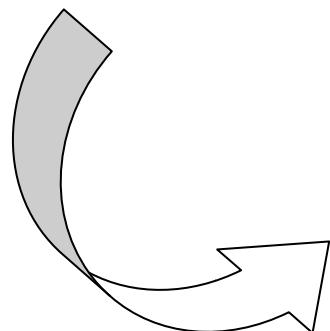
Depenent del rang del vermell, verd i blau de cada quadradet, es diu el seu color pertinent entre els sis possibles.

A més, cada cop que surt un color, el compta, per a més tard veure si ha funcionat bé.

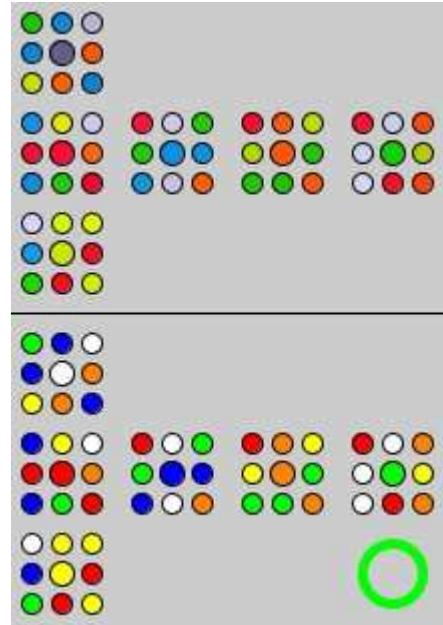
Es torna a fer el mateix dibuix d'abans, però ara amb el color preestablert si s'ha detectat bé. Si falla i no pot reconèixer el color, es veurà de color negre.

Compta el número resultant de quadrets que hi ha de cada color.  
Si el programa ha funcionat bé, hauria d'haver 9 de cada un.  
(Al blanc he posat que hi hagi com a mínim 8, perquè la cara amb el blanc central, al tenir el logo en negre, a vegades no el detecta bé.)

Guarda la imatge creada

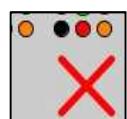


Colors detectats pel programa



Colors directament extrets de les fotos.

El cercle significa que en principi pot ser que sigui correcta la detecció, ja que hi ha el mateix nombre de tots els colors.  
En cas contrari, es mostra una creu vermella.



Imatge generada pel programa

## Espectre de colors

Aquestes són representacions dels espectres de colors que reconeix el programa. En cada color es mostra tots els possibles colors que el programa podria detectar com que fos d'aquell color. Al estar el color separat en 3 parts (red, green & blue), es mostra la gama de colors en una representació tridimensional. Cada dimensió correspon a un d'aquests colors primaris. En l'eix x, el vermell va del seu mínim al seu màxim. En l'eix y, el verd. I com era impossible fer una representació tridimensional del color, he utilitzat el temps com a tercera dimensió. Al GIF es mostra el rang de cada color del RGB, i el blau en moviment. Aquestes animacions també estan fetes amb Java. Per a veure els GIF en moviment, escaneja el codi QR.

```

1 int Rr1 = 140;
2 int Rg1 = 175;
3 int Rb1 = 175;
4 int Rb1i=Rb1;
5 int Rr2 = 255; //115
6 int Rg2 = 255; //80
7 int Rb2 = 255; //80
8 int count;
9 String name;
10 int canvi=0;
11 int wr=0;
12 int capt=1;
13 int tocapt=1;
14
15 void setup() {
16   size(400, 450);
17   background(0);
18   stroke(255, 100);
19   strokeWeight(1);
20   stroke(255, 0, 150);
21   line(16, 3*(Rg2-Rg1)+60, 3*(Rr2-Rr1)+29, 3*(Rg2-Rg1)+60);
22   line(15, 3*(Rr2-Rr1)-57, 15, 3*(Rg2-Rg1)+63);
23   line(3*(Rr2-Rr1)-30, 3*(Rg2-Rg1)+57, 3*(Rr2-Rr1)+30, 3*(Rg2-Rg1)+65);
24   point((15+((Rr2/255)*(3*(Rr2-Rr1)+15))), 3*(Rg2-Rg1)+58);
25   noStroke();
26   fill(255, 0, 0);
27   rect((Rr1+((Rr2-Rr1)+15))/255, 3*(Rg2-Rg1)+58, (Rr2-Rr1)+1*(3*(Rr2-Rr1)+15)/255, 5);
28   strokeWeight(1);
29   stroke(0, 255, 0, 150);
30   line(15, 3*(Rg2-Rg1)+90, 3*(Rr2-Rr1)+30, 3*(Rg2-Rg1)+90);
31   line(15, 3*(Rg2-Rg1)+87, 15, 3*(Rg2-Rg1)+93);
32   line(3*(Rr2-Rr1)+30, 3*(Rg2-Rg1)+87, 3*(Rr2-Rr1)+30, 3*(Rg2-Rg1)+93);
33   noStroke();
34   fill(0, 255, 0);
35   rect((15+((Rg1+3*(Rr2-Rr1)))/255, 3*(Rg2-Rg1)+88, (Rg2-Rg1)+1*(3*(Rr2-Rr1)+15)/255, 5);
36   stroke(255);
37   line(15, 3*(Rg2-Rg1)+25, 3*(Rr2-Rr1)+17, 3*(Rg2-Rg1)+25);
38   line(15, 3*(Rg2-Rg1)+22, 15, 3*(Rg2-Rg1)+28);
39   line(3*(Rr2-Rr1)+17, 3*(Rg2-Rg1)+22, 3*(Rr2-Rr1)+17, 3*(Rg2-Rg1)+28);
40   line(3*(Rr2-Rr1)+30, 3*(Rr2-Rr1)+25, 3*(Rg2-Rg1)+17);
41   line(3*(Rr2-Rr1)+22, 15, 3*(Rr2-Rr1)+28, 15);
42   line(3*(Rr2-Rr1)+22, 3*(Rg2-Rg1)+17, 3*(Rr2-Rr1)+28, 3*(Rg2-Rg1)+17);
43   fill(255);
44   text("round(Rr1)", 5, 3*(Rg2-Rg1)+41);
45   text("round(Rr2)", 3*(Rr2-Rr1)+8, 3*(Rg2-Rg1)+41);
46   text("RED", (3*(Rr2-Rr1)+10)/2, 3*(Rg2-Rg1)+41);
47   text("round(Rg1)", 3*(Rr2-Rr1)+32, 20);
48   text("round(Rg2)", 3*(Rr2-Rr1)+32, 3*(Rg2-Rg1)+22);
49   text("\n\n\n", 3*(Rr2-Rr1)+32, 110);
50
51 void draw() {
52   for (int i = 5; i<(Rr2-Rr1)+6; i++) {
53     for (int j = 5; j<(Rg2-Rg1)+6; j++) {
54       noStroke();
55       fill(Rr1-i-5, Rg1+j-5, Rb1);
56       rect(i-5, j-5, 3, 3);
57       if ((Rb1-1)>255 & wr==1) {
58         print((i-5)+" "+(j-5)+" "+round(Rr1+i-5)+" "+round(Rg1+j-5)+" "+round(Rb1));
59         print("\n");
60       }
61     }
62   }
63 }
64
65 fill(0);
66 noStroke();
67 rect(0, 3*(Rg2-Rg1)+110, 800, 800);
68
69 noStroke();
70 fill(0, 255);
71 ellipse((15+Rb1*(3*(Rr2-Rr1)+15))/255, 3*(Rg2-Rg1)+120, 5, 5);
72
73 fill(255);
74 text("R: "+round(Rr1)+" - "+round(Rr2), 15, 3*(Rg2-Rg1)+150);
75 text("G: "+round(Rg1)+" - "+round(Rg2), 15, 3*(Rg2-Rg1)+170);
76 text("B: "+round(Rb1)+" - "+round(Rb2)+" ", 15, 3*(Rg2-Rg1)+190);
77
78 strokeWeight(1);
79 stroke(0, 0, 255, 150);
80 line(15, 3*(Rg2-Rg1)+120, 3*(Rr2-Rr1)+30, 3*(Rg2-Rg1)+120);
81 line(15, 3*(Rg2-Rg1)+117, 15, 3*(Rg2-Rg1)+123);
82 line(3*(Rr2-Rr1)+30, 3*(Rg2-Rg1)+117, 3*(Rr2-Rr1)+30, 3*(Rg2-Rg1)+123);
83
84 if (capt==1 && tocapt==1) {
85   count+=1;
86   name="Frames/blancos_posibles_"+(count)+".jpg";
87   saveFrame(name);
88 }
89 if (Rb1==Rb2 & canvi==0) {
90   Rb1++;
91 }
92 if (Rb1>Rb1i & canvi==1) {
93   Rb1--;
94 }
95 if (Rb1==Rb1i) {
96   canvi=1;
97 }
98 if (Rb1!=Rb1i) {
99   canvi=0;
100 }
101 }
102 }

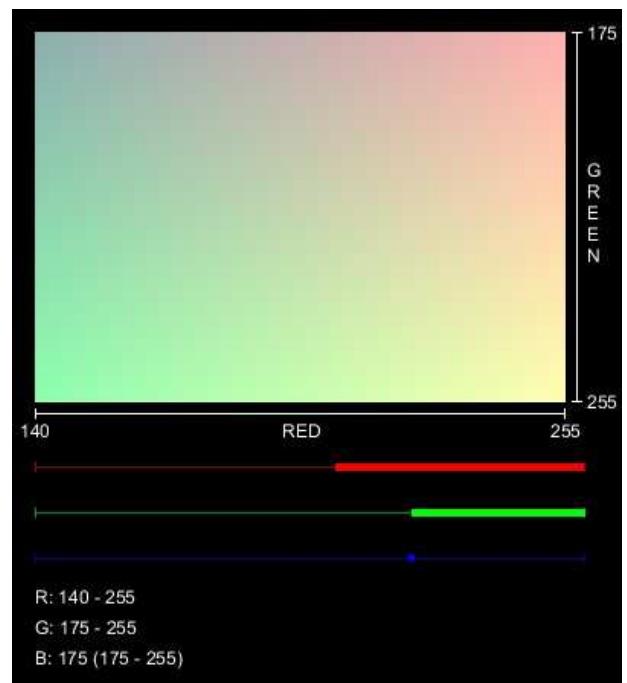
```

De dreta a Esquerra augmenta el vermell, i de dalt a avall. el verd.

Llegenda, a sota de les tres línies.

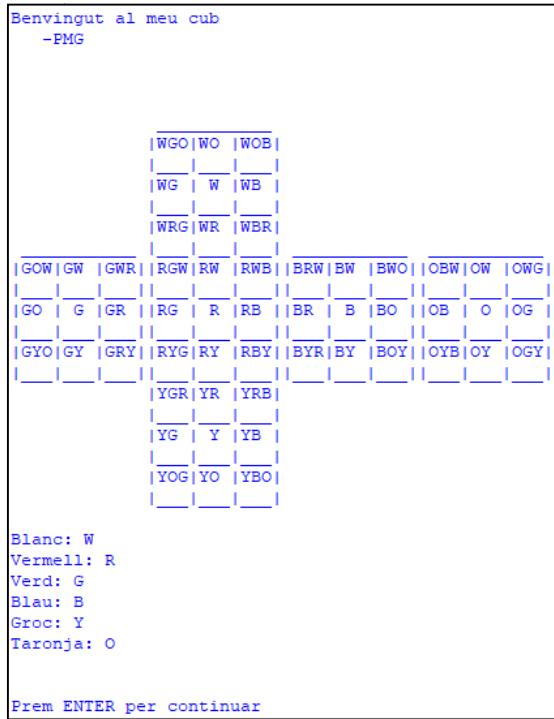
Guardar totes les captures, amb les que després he fet un GIF amb la eina online: [www.ezgif.com](http://www.ezgif.com)

Aquest és el programa del blanc, però tots els altres són exactament iguals, només canvia els números del principi de mínim i màxim de Red, Blue & Green.

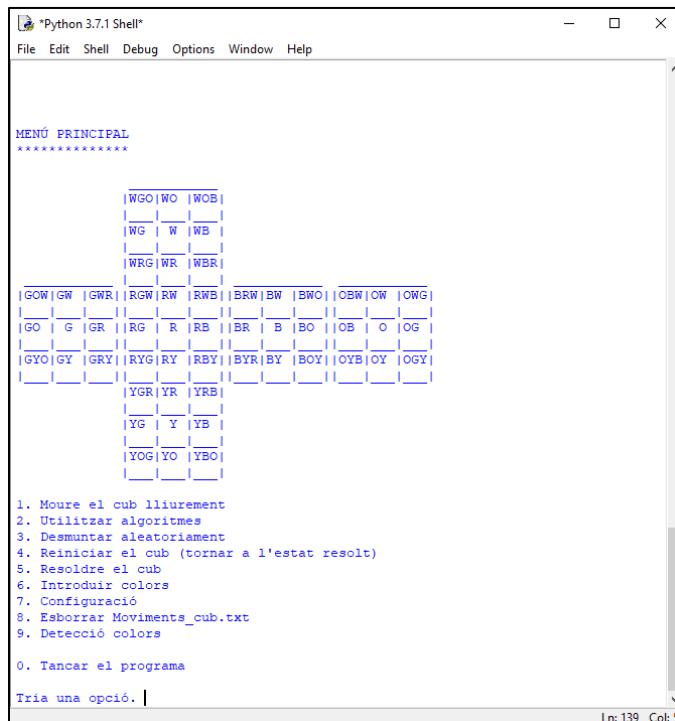


## Resolució virtual del cub

Aquesta part del procés està feta amb Python. Consta d'un programa bastant extens amb moltes opcions. Quan obres el programa es veu això:



Després d'aquesta primera pantalla introductòria, quan premes ENTER s'obre el menú principal.



Aquest és el menú principal, pots entrar a diferents apartats i fer diferents coses. A “*Tria una opció.*” escrius un número i et porta al apartat seleccionat. Si poses ‘0’ es tanca el programa.

## Moviment del cub

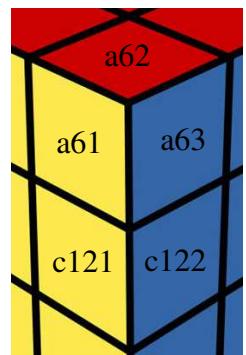
Aconseguir que el cub es mogués, encara que no s'aprecia molt visualment, va ser una tasca que em va portar molt temps. Per començar, vaig fer una representació plana del cub.

	a31	c31	a41																
	c21	W	c41																
	a11	c11	a21																
a32	c22	a13	a12	c12	a23	a22	c42	a43	a42	c32	a33								
c61	G	c52	c51	R	c82	c81	B	c72	c71	O	c62								
a73	c102	a52	a53	c92	a62	a63	c122	a82	a83	c112	a72								
			a51	c91	a61														
			c101	Y	c121														
			a71	c111	a81														

Vaig crear una variable per a cada cel·la, cadascuna amb una lletra i un número. Totes les cantonades tenen la lletra 'a' i les aretes tenen la 'c'.

Els 6 centres no tenen variables, simplement la lletra del color que representen: W (White – blanc), G (Green – Verd), R (Red – Vermell), B (Blue – Blau), O (Orange – Taronja) i Y (Yellow – Groc).

A les cantonades, el número significa quina cantonada ocupa i quina de les tres cares és segons un sistema que m'he inventat. Hi ha 8 cantonades físiques, unes 24 cel·les de cantonades en total. El primer número va del 1 al 8 (quina cantonada), i el segon varia del 1 al 3 (quina cara de la cantonada). Com a referència he agafat totes les cel·les número 1 de la seva cantonada les que estaven en la cara blanca i les de la cara groga. La número 2 i 3 venien donades per sentit horari.



A les aretes, el sistema és el mateix. Però hi ha més aretes, són 12 en total, 36 cel·les d'arestes. El primer número va del 1 al 12 (per això hi ha algunes amb un número de 3 xifres). El segon varia entre l'1 i el 2.

Per tant, cada peça física tindrà les seves dues o tres (depenent si es aresta o cantonada) cel·les amb el primer número igual i el segon diferent per a cada cara.

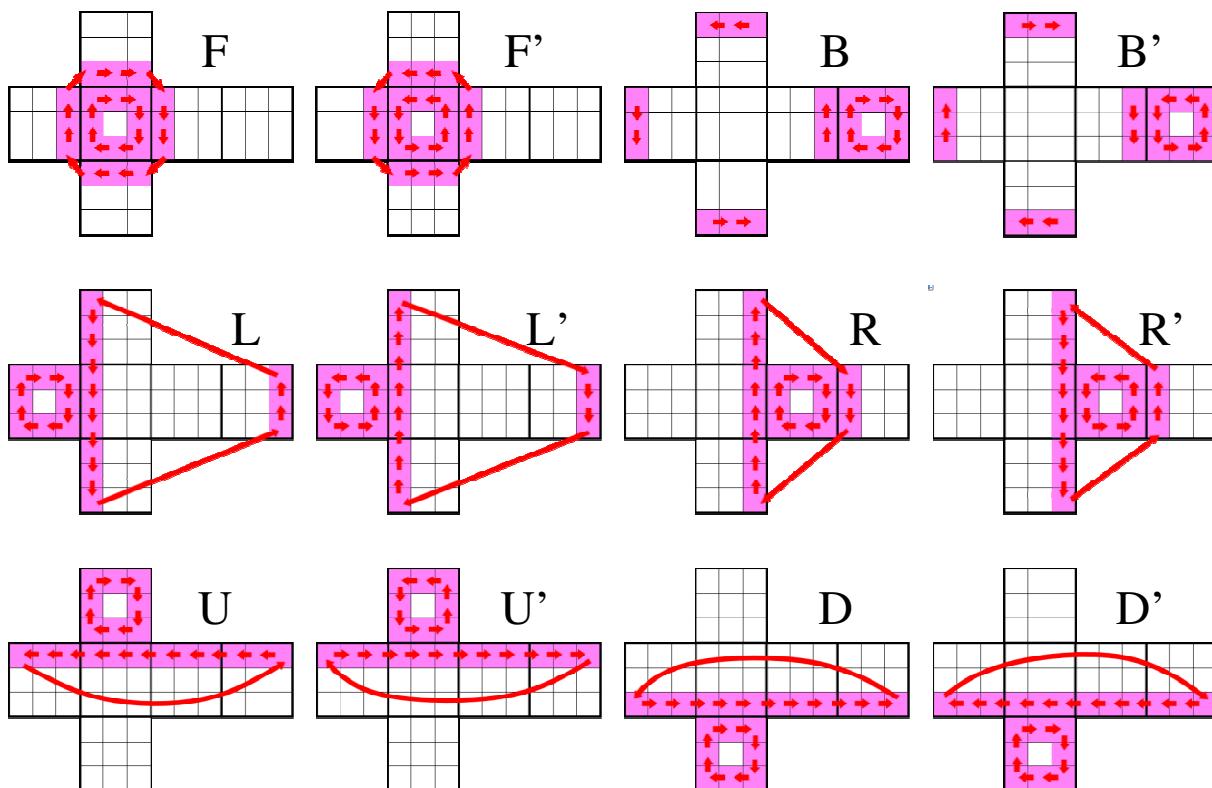
És molt important que aquestes peces vagin tota l'estona juntes en tots els moviments, ja que en el cub físic, són peces enganxades que es mouen sempre juntes.

Un cop feta l'estructura base del cub, cal fer-lo capaç de moure. Per a això vaig crear el que serien els 12 moviments principals. Aquests moviments s'anomenen amb lletres, fent referència al punts cardinals: U (Up), L (Left), R (Right), F (Front), D (Down) i B (Back). Vaig prendre com a referència la cara vermella com a frontal.

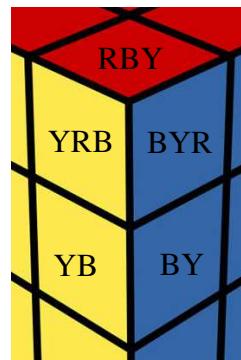
Cadascun d'aquests moviments pot ser en sentit horari o antihorari, per indicar-ho, U serà sentit horari, U' antihorari. Aquest és el mateix codi que s'utilitza normalment al argot dels cubs, per exemple a tutorials i manuals. Doncs, els moviments són els següents:

U	Cara blanca en sentit horari
L	Cara verda en sentit horari
F	Cara vermella en sentit horari
R	Cara blava en sentit horari
B	Cara taronja en sentit horari
D	Cara groga en sentit horari
U'	Cara blanca en sentit antihorari
L'	Cara verda en sentit antihorari
F'	Cara vermella en sentit antihorari
R'	Cara blava en sentit antihorari
D'	Cara taronja en sentit antihorari
B'	Cara groga en sentit antihorari

Ja sabem quins són els moviments que ha de fer, falta aconseguir que els faci. Per fer-ho, aquest era el canvi de variables que havia de fer per a cada moviment:



Un cop el cub es capaç de moure's, vaig fer l'assignment de color. Cada casella té més d'una lletra de nom. Les cantonades tenen 3 lletres, la primera representa el color de la cara, i les altres dues els colors de les cares de la mateixa cantonada en sentit horari. Les arestes tenen dues lletres, la primera representa el color de la cara, i la segona el color de l'altra cara de la mateixa aresta.



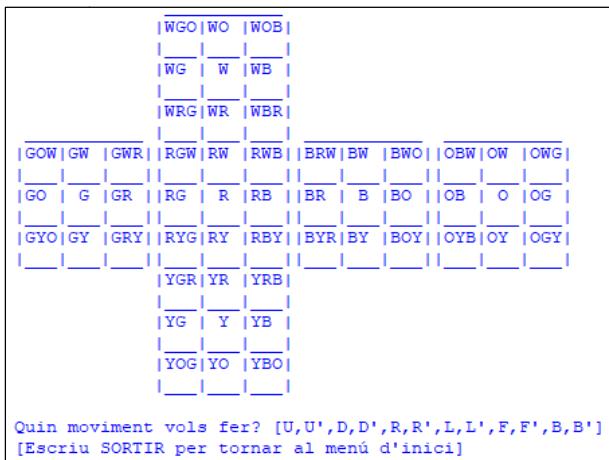
			WGO	WO	WOB							
			WG	W	WB							
			WRG	WR	WBR							
GOW	GW	GWR	RGW	RW	RWB	BRW	BW	BWO	OBW	OW	OWG	
GO	G	GR	RG	R	RB	BR	B	BO	OB	O	OG	
GYO	GY	GRY	RYG	RY	RYB	BYR	BY	BOY	OYB	OY	OGY	
			YGR	YR	YRB							
			YG	Y	YB							
			YOG	YO	YBO							

Aquesta és la base del programa, una simulació bidimensional de totes les cares del cub, amb la representació del seu color, i una simulació de moviment.

## **Diferents apartats del programa**

### **1. Moure el cub lliurement**

En aquest menú pots fer el moviment que vulguis. Si escrius “sortir” tornes al menú principal



### **2. Utilitzar algoritmes**

Aquí pots utilitzar algoritmes, que són conjunts de moviments ja preestablerts.

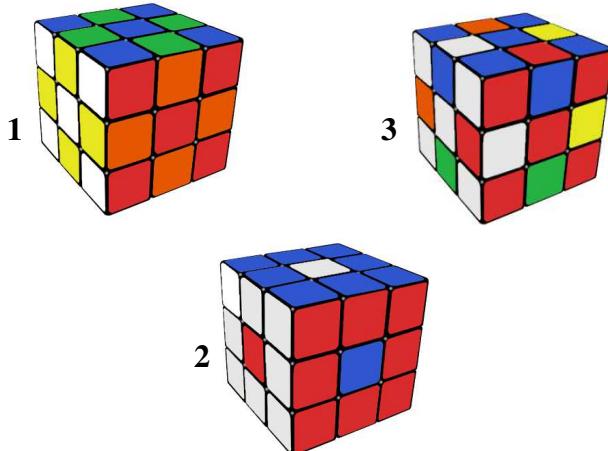
The image shows a terminal window displaying a 3x3x3 Rubik's cube state and a list of algorithms:

```
|WGO|WO |WOB| | | | | | | | | |
|_ _ _|_ _ |
|WG | W |WB |
|_ _ _|_ _ |
|WRG|WR |WBR|
|_ _ _|_ _ |
|GOW|GW |GWR| RGW|RW |RWB| BRW|BW |BWO| OBW|OW |OWG|
|_ _ _|_ _ _|_ _ _|_ _ _|_ _ _|_ _ _|_ _ _|_ _ _|_ _ |
|GO | G |GR | RG | R |RB | BR | B |BO | OB | O |OG |
|_ _ _|_ _ _|_ _ _|_ _ _|_ _ _|_ _ _|_ _ _|_ _ _|_ _ |
|GYO|GY |GRY| RYG|RY |RBY| BYR|BY |BOY| OYB|OY |OGY|
|_ _ _|_ _ _|_ _ _|_ _ _|_ _ _|_ _ _|_ _ _|_ _ _|_ _ |
|YGR|YR |YRB|
|_ _ _|_ _ |
|YG | Y |YB |
|_ _ _|_ _ |
|YOG|YO |YBO|
|_ _ _|_ _ |
```

Selecciona l'algoritme que vols fer:

1. La flor
2. Canvi de centres
3. Gir d'arestes
4. Tornar enrere

Tria. |



### **3. Desmuntar aleatoriament**

La opció 3 del menú crea una sèrie de moviments de manera aleatòria entre els 12 possibles. Al menú de configuració pots escollir el número de moviments que vols que generi.

### **4. Reiniciar el cub (tornar a l'estat resolt)**

Aquesta opció simplement torna el cub a l'estat resolt sense fer res, es reinicia.

## 5. Resoldre el cub

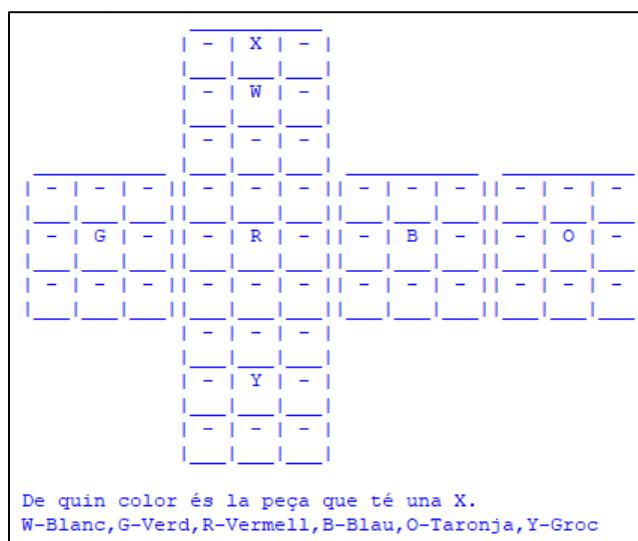
Aquesta opció porta el cub de l'estat actual desmuntat a l'estat resolt utilitzant els moviments prèviament vistos. A l'apartat següent s'explica com ho fa.

## 6. Introduir colors

D'entrada aquest apartat té dos subapartats.

1. Els colors de totes les peces
2. Canviar algun color

Al primer apartat, et surten totes les peces en blanc. Cada cop que introduceixes un color dient la lletra del color, apareix una X en la següent casella. La casella que té la X és de la qual has de dir el color.



Al segon apartat, és només per modificar el color d'alguna casella puntual, per si ens hem equivocat a l'apartat anterior, o simplement volem canviar algun color. Es mostra el cub, amb el color respectiu de cada casella. S'han de posar les coordenades per a escollir la casella que es vol canviar, per exemple G5. Un cop poses les coordenades de la casella que vols canviar, et surt una X a aquella casella i pots triar el color.

	A	B	C	D	E	F	G	H	I	J	K	L
1							W	W	W			
2							W	W	W			
3							W	W	W			
4	G	G	G	R	R	R	B	B	B	O	O	O
5	G	G	G	R	R	R	X	B	B	O	O	O
6	G	G	G	R	R	R	B	B	B	O	O	O
7												
8												
9												

De quin color és la cel·la amb la X (casella G5)? (W,G,R,B,O,Y)

	A	B	C	D	E	F	G	H	I	J	K	L
1							W	W	W			
2							W	W	W			
3							W	W	W			
4	G	G	G	R	R	R	B	B	B	O	O	O
5	G	G	G	R	R	R	X	B	B	O	O	O
6	G	G	G	R	R	R	B	B	B	O	O	O
7												
8												
9												

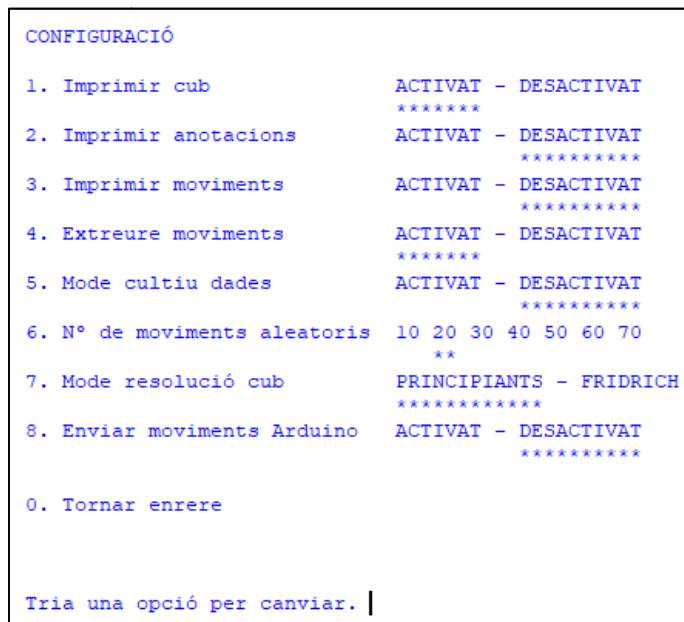
Quina peça vols canviar? (Escríu SORTIR per tornar al menú INTRODUIR COLORS)

Quan ja has introduït tots els colors en qualsevol de les dues opcions, el nom que indica el color de cada cel·la (format per 2 o 3 números) es creat automàticament, agafant el color de la cel·la pertinent i el demés colors de la mateixa peça.

No recomano utilitzar molt aquesta opció, donat que amb un sol error el cub és impossible de resoldre i el programa no ho detectarà, però tampoc serà resolt.

## 7. Configuració

Aquesta opció obre un altre menú on pots triar entre varies opcions per a utilitzar el programa com millor prefereixis. Al obrir-lo es veu el següent. Aquestes són les configuracions que venen per defecte quan obres el programa, per canviar-les, només cal que indiquis quina vols canviar.



### 1. Imprimir cub

Quan aquesta opció està activada, cada cop que es fa un moviment (ja sigui movent el cub lliurement, desmuntant-lo al azar, resolent-lo...) s'imprimeix la imatge del cub per veure quins moviments s'estan fent. És una opció molt útil per exemple al moure el cub lliurement, però no quan es resol sol, perquè va molt lent i fa perdre molt de temps.

### 2. Imprimir anotacions

Aquesta opció la utilitzava molt quan estava programant i encara no havia acabat, per veure si tot anava bé, i si el programa fallava trobar abans i més fàcilment el que no funcionava.

### 3. Imprimir moviments

Quan s'està resolent el cub, et diu quins moviments està fent. Era molt útil per quan estava creant el programa i fallava saber per on anava i que ha passat.

#### 4. Extreure moviments

Qualsevol moviment que fa l'imprimeix en un arxiu de text, per veure quins moviments ha fet. En un principi m'anava a servir per passar els moviments al Arduino, però al final no ho he utilitzat per això. Tots els moviments es guarden al arxiu “Moviments\_cub.txt”, i si no existeix es crea.

#### 5. Mode cultiu de dades

Aquest mode em permet desmuntar el cub al azar i muntar-lo per verificar que el programa no fallava. A més, em guardava en un arxiu de text el número de moviments que cada prova havia utilitzat per resoldre el cub. Quan l'actives, totes les altres opcions es desactiven, no són compatibles; i si les tornes a activar, el mode de cultiu es desactiva. Utilitzant aquesta opció, vaig fer el procediment de desordenar i muntar el cub unes 50.000 vegades aproximadament. (Veure annex A)

#### 6. Nº de moviments aleatoris

Pots seleccionar quants moviments aleatoris vols que es facin a l'hora de desordenar. El que volia veure era si fent més moviments per desordenar, es resolia en més moviments. Vaig poder observar que si que el moviments per desordenar i els que calen per ordenar tenen una relació, però a partir de 30 ja no es nota la diferència, ja que el cub ja està bastant desordenat.

#### 7. Mode resolució cub

Aquesta opció no es funcional, és un projecte que volia fer per a optimitzar al màxim el número de moviments que calen per arribar a l'estat resolt. He fet el programa amb el mètode de principiants (explico com ho he fet al següent apartat); i volia implementar el mètode Fridrich (explicació mètode Fridrich al Annex B) que utilitza un número considerablement més baix de moviments que el mètode de principiants. Aquesta opció ajudaria a que el procés fos més ràpid, però requereix de molt de temps, queda com a futur projecte d'una segona part.

#### 8. Enviar moviments Arduino

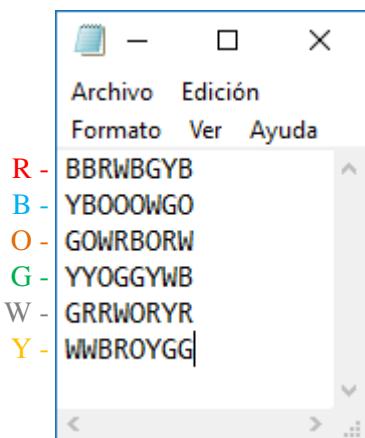
Aquí pots dir si vols o no enviar qualsevol moviment que es faci al programa al Arduino per a que els motors els realitzin. Això serveix per a tenir sempre el cub en el mateix estat al Python i físicament; així es pot tenir més controlat i no cal fer cap mena d'ajustaments. Per fer això vaig utilitzar un mòdul de Python anomenat serial. (Com instal·lar mòduls al Python al Annex C)

## 8. Esborrar Moviments\_cub.txt

Triant aquesta opció l'únic que fas és eliminar l'arxiu “Moviments\_cub.txt”.

## 9. Detecció colors

Aquesta opció el que fa és obrir l'arxiu de text “colors.txt” creat pel Java, llegir-lo i col·locar els colors al seu lloc.

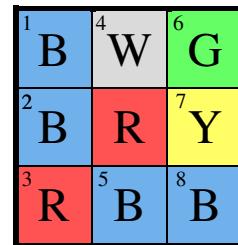


```
R - BBRWBGYB  
B - YB000WGO  
O - GOWRB0RW  
G - YYOGGGYWB  
W - GRRWORYR  
Y - WWBROYGG
```

Aquesta és una possible mostra del arxiu que crea el Java. Hi ha 6 files, una per cada cara. A cada fila 8 lletres, una per cada quadret de la cara (sense comptar el centre que sempre és el mateix).

El Python ja sap com llegir-ho, la primera línia és per al vermell i les lletres van en aquest ordre:

BBRWBGYB  
1 2 3 4 5 6 7 8



L'ordre en el que l'he programat és el mateix en el que s'han de fer les fotos amb l'*Office Lens* per a que funcioni correctament. Aquest ordre és: Vermella, blava, taronja, verda, blanca i groga.

També cal tenir en compte un detall molt important a l'hora de fer les fotos amb l'*Office Lens*: l'orientació. Totes les fotos han de ser fetes amb la mateixa orientació amb la que estan al programa. És a dir, quan es fan les de la cara vermella, blava, taronja i verda, han de tenir sempre a dalt la cara blanca i a baix la cara groga. Si no es fan així, el programa de detecció de color funcionarà igual, però les posarà totes malament, el qual desembocarà molt probablement en que no sigui possible resoldre el cub. I si, per poc possible que sigui, és un estat possible i l'acaba resolent el Python, és impossible que es resolgui bé el cub físic.

## Procés de resolució del cub virtual

Al començar aquest procés, tenim el cub en un estat desordenat. El que busquem és que es resolgui utilitzant els moviments ja vistos. Per fer-ho, vaig utilitzar el mètode de principiants, que era el que jo coneixia i el més senzill. Com que a la representació del cub que vaig fer al Python el vermell estava al centre, vaig fer que el programa comences a resoldre el cub per la cara vermella.

Aquest mètode consisteix en anar per nivells, primer es fa la creu de la primera cara, després es posen les cantonades, després el segon nivell, i per acabar la ultima cara i l'últim nivell.

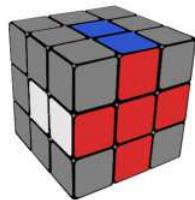
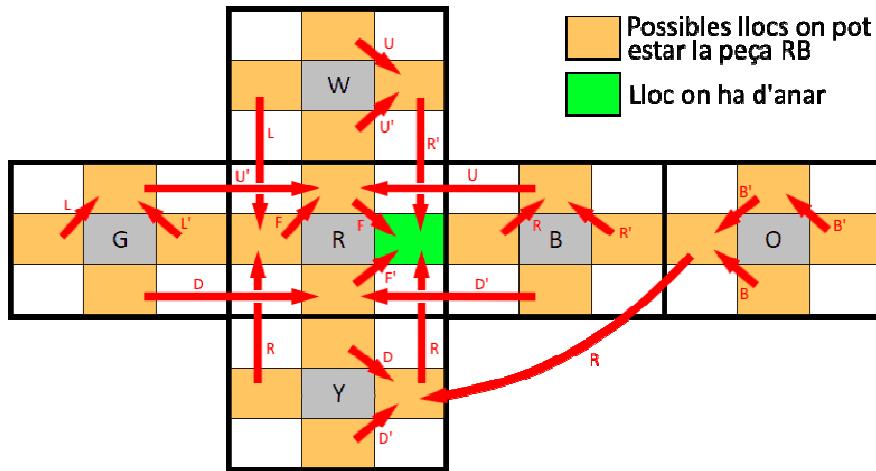


Aquest mètode és relativament senzill, la creu del principi és molt fàcil d'orientar, i d'allà cap a endavant només cal seguir una sèrie d'algoritmes. Però fer-ho a la vida real amb un cub de veritat és una cosa, amb un cub virtual és molt diferent. I al utilitzar la cara vermella com a inicial, en comptes de la blanca que és la típica vaig haver d'adaptar tots els algoritmes.

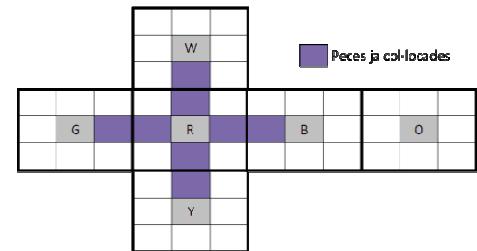
Per començar, el programa busca les peces de la creu vermella. La primera que busca és la que hauria d'estar a la casella c12, la RW. És el mateix que buscar la c11 (WR) que és la mateixa peça.

a31	c31	a41													
c21	W	c41													
a11	c11	a21													
a32	c22	a13	a12	c12	a23	a22	c42	a43	a42	c32	a33				
c61	G	c52	c51	R	c82	c81	B	c72	c71	O	c62				
a73	c102	a52	a53	c92	a62	a63	c122	a82	a83	c112	a72				
			a51	c91	a61										
			c101	Y	c121										
			a71	c111	a81										

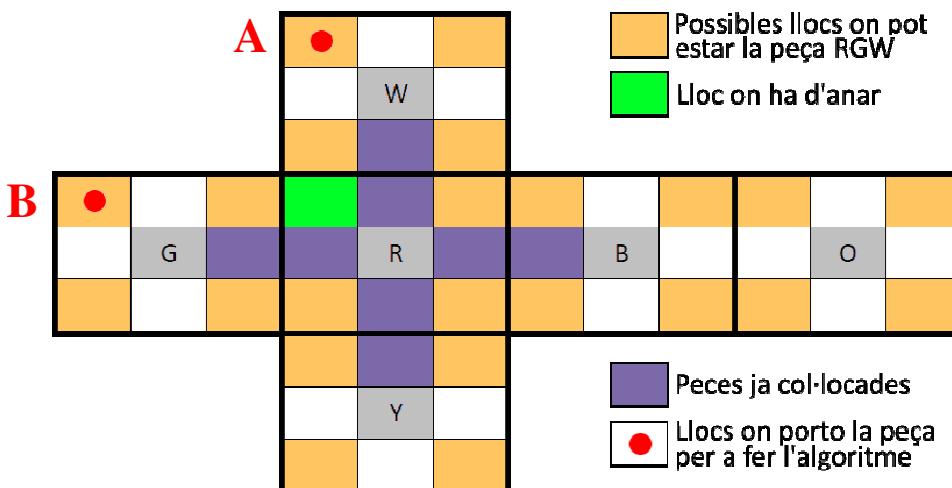
Ara mateix, la RW pot estar a tants llocs com arrestes hi ha, el Python la busca i depenent d'on sigui fa un moviment o un altre. Per a que funcioni, tots els passos que ha de fer han de ser moviments dels 12 establerts abans. Per a aconseguir-ho, vaig fer que seguis un “camí” fins arribar a la cel·la destí. Busca entre totes les possibles opcions on podria estar la peces que busquem, i amb els moviments la porta fins a la cel·la c82.



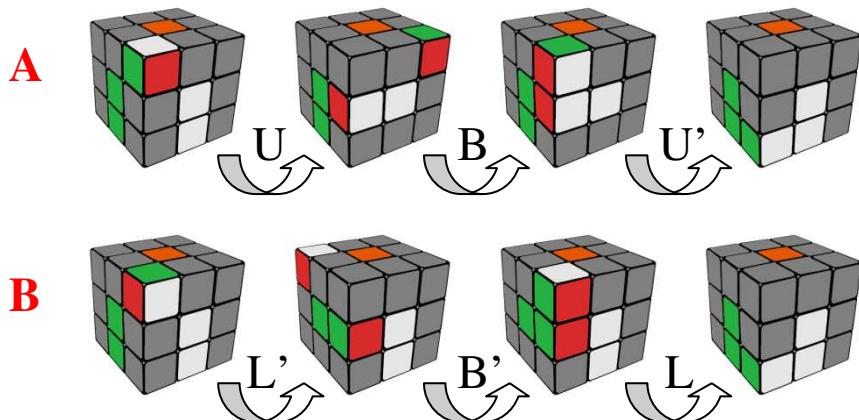
Així es col·loca la primera peça. Amb les quatre de la creu (RW, RG i RY) fa el mateix, i les acaba col·locant les quatre sense moure'n cap. Al acabar aquesta fase, ja tenim la creu de la cara inicial feta. Ara cal posar les cantonades del primer nivell.



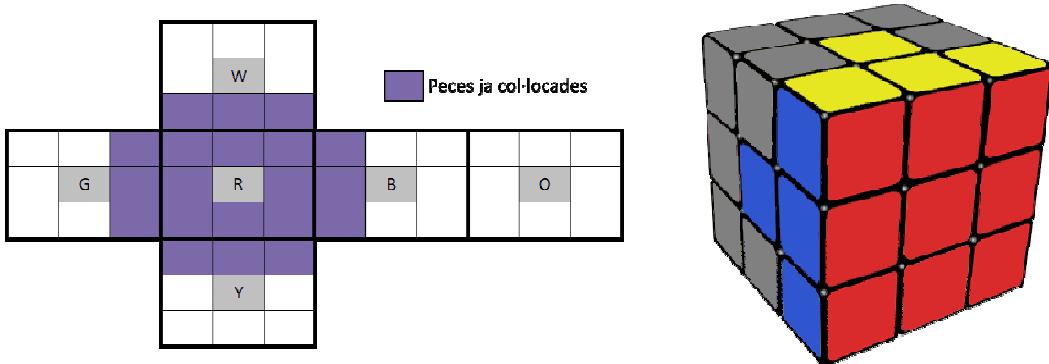
Per posar les cantonada, comença per la cantonada RGW que ha d'anar a cel·la a12. Si no hi ha hagut sort, i la peça no està ja ben col·locada per casualitat, es porta la RGW fins a qualsevol dels dos punts marcats com s'ha fet al pas anterior.



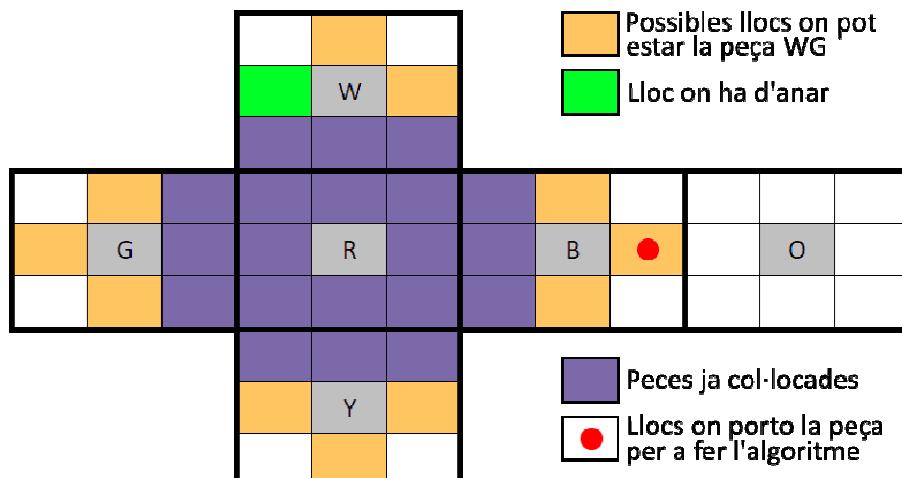
Un cop ja està a qualsevol d'aquests dos punts, es fa un algoritme molt curt per ficar-la a on ha d'anar.



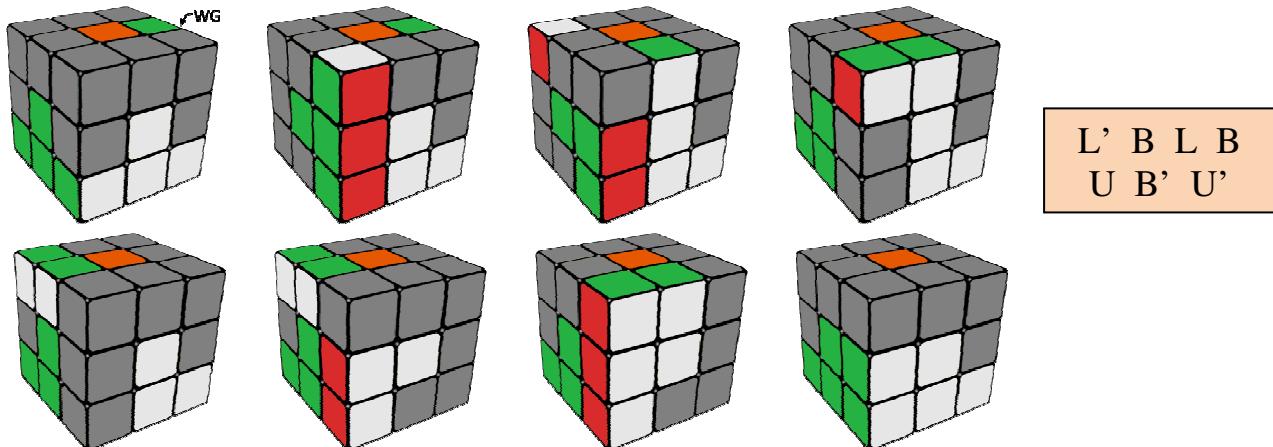
Fa el mateix en cada cantonada del primer nivell. A aquesta alçada el programa ja ha resolt el primer nivell, que en principi és menys complex però té moltes més possibilitats, per tant, m'ha ocupat molt el codi per fer només el primer nivell. El cub ara mateix està així:



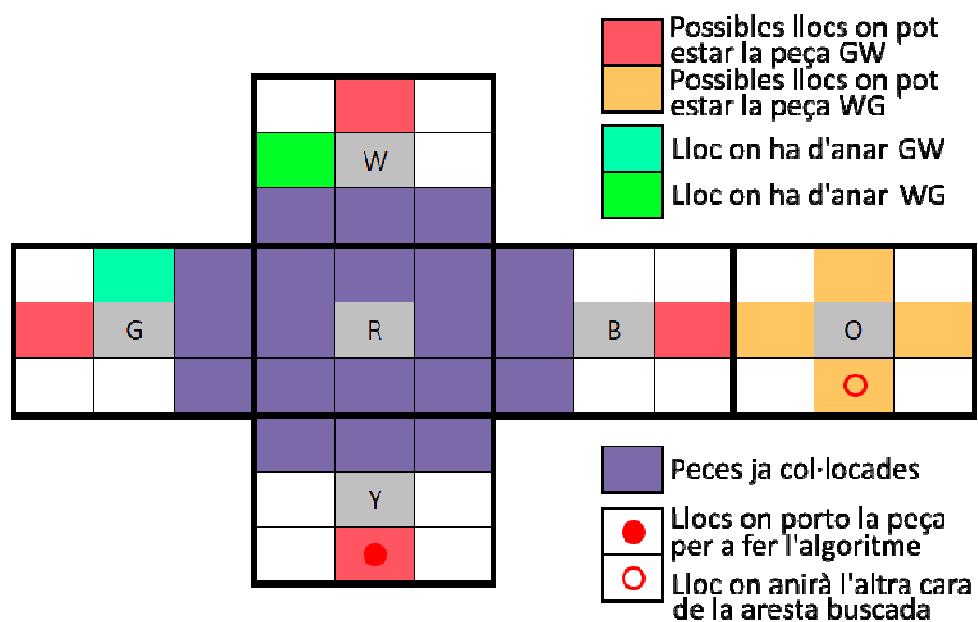
El següent pas que realitza el programa és buscar les arestes que han d'anar al segon nivell. La primera que col·loca és la peça WG a la cel·la c21. Si la peça que busca no està a la cara taronja fa el següent:



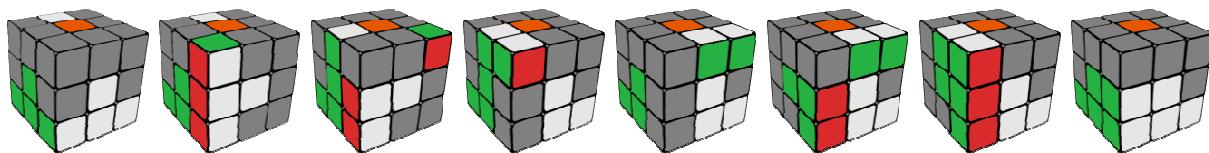
Un cop està en la posició marcada, fa un algoritme per a posar-la al lloc desitjat sense modificar la cara que ja està resolta.



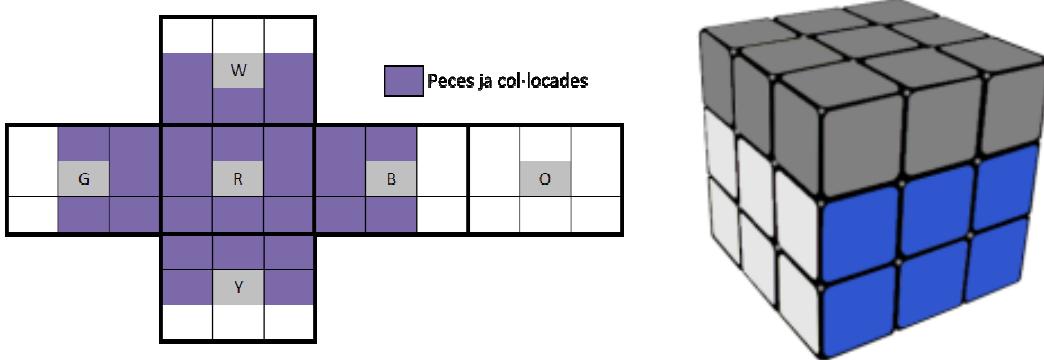
D'altra banda, si la aresta que busca es troba a la cara taronja, busca la peça GW que es l'altra cara de la mateixa aresta. Doncs fa el que seria l'algoritme mirall de l'anterior; els moviments són els mateixos però invertits, el programa el que entén és el següent:



L'algoritme que realitza és el següent: U B' U' B' L' B L



Repeteix el mateix procés per a totes les peces, llavors el cub ja està resolt fins al segon nivell.



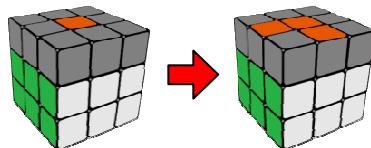
Per fer la cara superior, començarà només orientant totes les cares taronges deixant així tota la cara del mateix color, sense importar el tercer nivell.

A partir d'aquí el procediment a seguir és menys intuïtiu i més mecànic. Només cal seguir ordres i executar algoritmes, és pràcticament impossible guiar-se per la intuïció i acabar resolent el cub.

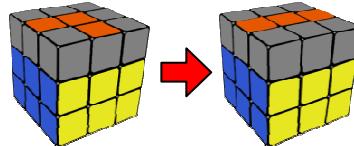
Per això el programa a partir d'aquí té majoritàriament algoritmes més llargs i menys situacions possibles.

Ara es centra només en el color de la cara de dalt. Per fer això li dic que només es fixi en la primera lletra de cada peça, la qual reflexa el seu color, el que busca és que totes siguin taronges. Abans d'això es busca la creu taronja. Per aconseguir la cara taronja només cal fer un algoritme, que és: L U B U' B' L'. Es pot aplicar en tres situacions:

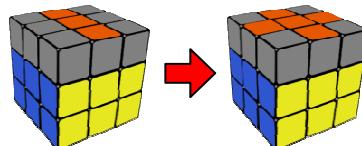
- Si la cara de taronja no h ha cap orientada. Al fer l'algoritme ens quedaran dues arestes ben orientades en forma d'L.



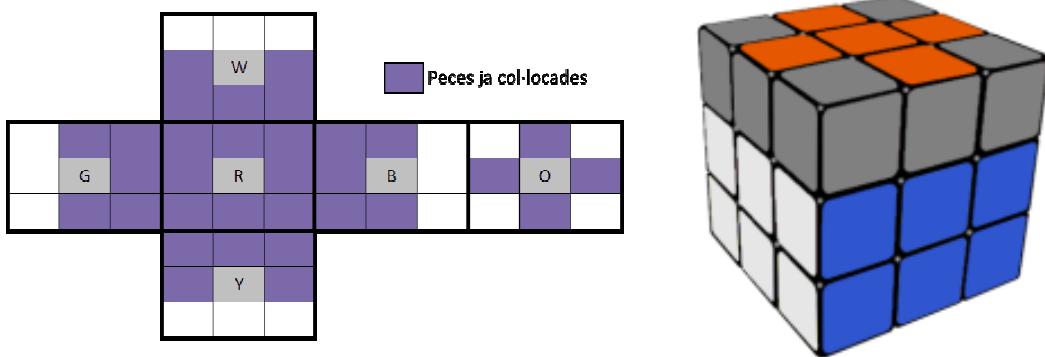
- Se hi ha dues orientades formant una L. La L es situa amb un taronja a sobre de la blava i un a sobre de la groga per a que l'algoritme funcioni. Al acabar l'algoritme queden dues peces ben orientades, però formant una línia recta.



- Si hi ha dues orientades formant una recta. La recta és situa sobre la cara blanca i la groga, es fa l'algoritme i ja tenim la creu feta.



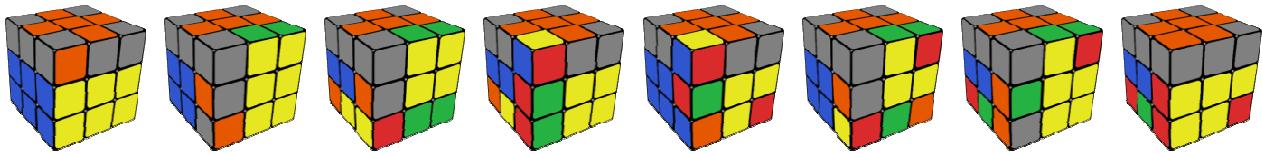
L'estat actual del cub és aquest:



L'algoritme per col·locar bé les arestes és encara més fàcil.

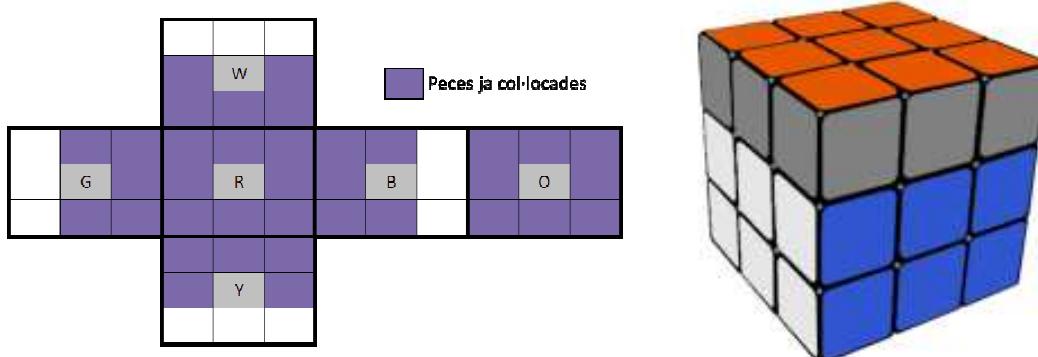
El programa agafa només les arestes que estan malament col·locades i fa el següent algoritme:

$2(D' F' D F)$

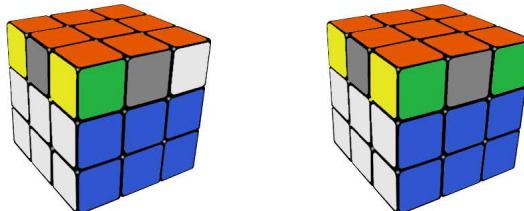


Ara mateix, s'ha pogut col·locar la cantonada orientada cap amunt, però em mogut el que ja havíem fet. Aparentment és així, però aquest algoritme està fet per a que quan el repeteixes 3 vegades torna al estat original. Per tant, el que fa ara el cub és moure només la cara taronja per posar la següent cantonada taronja mal orientada a la cel·la a83 i repeteix l'algoritme. Quan es fan aquest moviments, es mou la peça en sentit antihorari vist des de fora, així que si el taronja està a la cel·la a82, s'haurà de fer dues vegades. Tal i com està creat el cub de Rubik, es impossible (si està tot bé) que hi hagi només una cantonada malament; sempre hi haurà suficients malament com per fer aquest algoritme un número de vegades igual a un múltiple de 3.

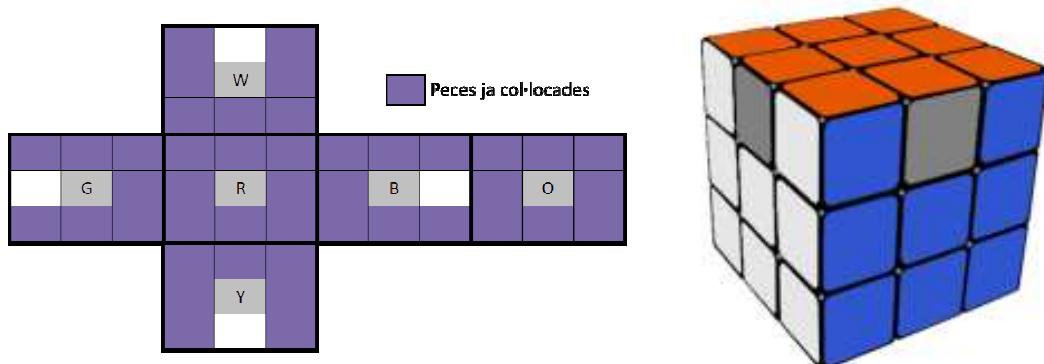
Al acabar, ja estarà tota la cara taronja bé, però no tot el tercer nivell.



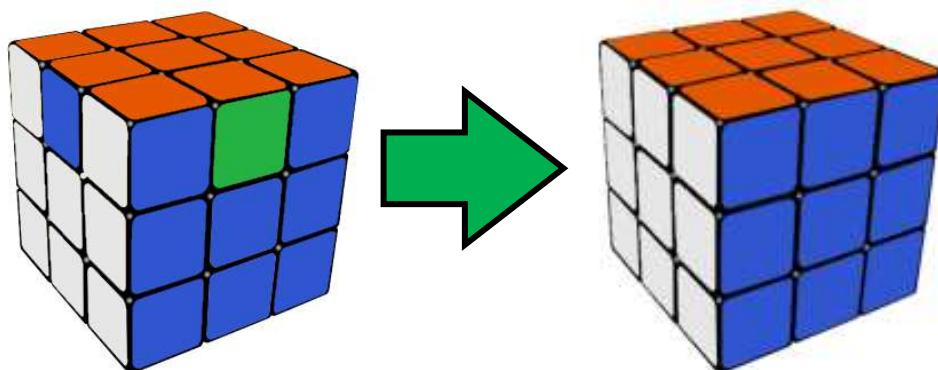
Per a casi acabar, busca si hi ha dues cantonades que estiguin a la mateixa cara i tinguin el mateix color, és a dir, que ja estiguin ben colocades. Si totes estan ben col·locades passarà al últim pas, si només dues estan bé col·locades les situarà a la cara verda i farà el següent algoritme. I si no n'hi ha cap, el fa igualment. El que fa aquest algoritme es canviar de posició les dues cantonades oposades a des d'on es comença l'algoritme: U R' U 2L U' R U 2L 2U



Per fi, ha arribat a l'últim pas, ja està tot ben col·locat, només falta acabar de posar bé les ultimes arrestes.



Per a aquest pas, l'únic que fa es fer un algoritme que serveix per rotar tres arrestes. Si les quatre estan malament, fa l'algoritme sense fer res més; doncs una es col·locarà i repetirà el procés. I si només



**I ja estaria!! El cub ja s'ha resolt!**

Al acabar, el programa mira que tot estigui bé (cada peça al lloc que li toca) i imprimeix el missatge següent:

```
|WGO|WO |WOB| | | | | | | | | | | | |
|_ _ _|_ _ |
|WG | W |WB|
|_ _ _|_ _ |
|WRG|WR |WBR|
|_ _ _|_ _ |
|GOW|GW |GWR||RGW|RW |RWB||BRW|BW |BWO||OBW|OW |OWG|
|_ _ _|_ _ _|_ _ _||_ _ _|_ _ _|_ _ _||_ _ _|_ _ _|_ _ _||_ _ _|_ _ _|_ _ |
|GO | G |GR ||RG | R |RB ||BR | B |BO ||OB | O |OG|
|_ _ _|_ _ _|_ _ _||_ _ _|_ _ _|_ _ _||_ _ _|_ _ _|_ _ _||_ _ _|_ _ _|_ _ |
|GYO|GY |GRY||RYG|RY |RBY||BYR|BY |BOY||OYB|OY |OGY|
|_ _ _|_ _ _|_ _ _||_ _ _|_ _ _|_ _ _||_ _ _|_ _ _|_ _ _||_ _ _|_ _ _|_ _ |
|YGR|YR |YRB|
|_ _ _|_ _ |
|YG | Y |YB|
|_ _ _|_ _ |
|YOG|YO |YBO|
|_ _ _|_ _ |

cub resolt!
Comprobació, està bé!
Han calgut 158 moviments!
```

Aquesta és segurament la part que se m'ha fet més llarga programant. Havia d'estar constantment amb el cub a les mans comprovant tots els algoritmes, pensant totes les possibilitats, fent-me esquemes... I quan fallava després d'estar molta estona fent-lo era desesperant. És una part molt pesada d'explicar, però més encara de fer. Aquest codi m'ha ocupat unes 1600 línies per a que funcioni perfectament.

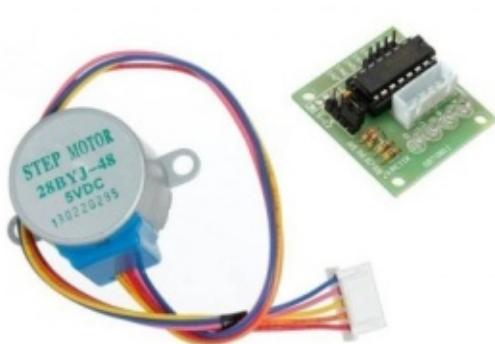
## Resolució física del cub

Sincerament, quan vaig començar el projecte no m'esperava arribar tant lluny com he arribat. Sabia que era un projecte molt ambiciós i tot i que em veia capaç no les tenia totes. Al principi inclús no sabia si podria fer programa que resolgués el cub amb Python. Però treballant molt, i tirant-me moltes hores programant i buscant informació al final si que vaig arribar; inclús superant les meves expectatives inicials. És per això que la part física tampoc la tenia molt planificada des de el principi.

Quan vaig començar, vaig mirar varies opcions. Tenia pensat utilitzar Arduino, ja que es una eina senzilla i molt útil. No l'havia utilitzat mai i realment em feia molta il·lusió perquè havia escoltat a parlar de projectes fets amb aquesta placa i havia vist alguns vídeos on la utilitzaven.

Al principi, vaig provar d'utilitzar un servo motor. Tot i ser la opció més senzilla, econòmica i la que tenia més a mà; va ser ràpidament descartada ja que els servos com a màxim poden fer girs de 360°. I si volia resoldre un cub de Rubik no hauria de estar tan limitat amb els girs. Encara que no em servien, em van servir per fer les meves primeres proves amb l'Arduino i poder veure poc a poc com funcionava, fent-me familiar amb el seu llenguatge. Em va semblar molt senzill controlar un servo motor amb l'Arduino, només necessitava un pin per enviar informació, i dos per l'alimentació (5V i GND); apart, el codi era molt senzill.

Un cop descartats els servos no sabia quina altra opció em serviria. No tenia ni idea de que podia utilitzar amb Arduino, ni com; així que vaig començar a buscar vídeos de màquines que resolien el cub de Rubik ja fetes. La gran majoria, per no dir totes, utilitzaven motors pas a pas.



Va ser llavors quan vaig decidir comprar-me motors pas a pas per provar-los. L'error que vaig cometre va ser comprar els més barats i petits. Pensava que podrien moure la cara del cub, però no només no podien per el parell tan petit que tenien, sinó que a demés giraven molt lentament.

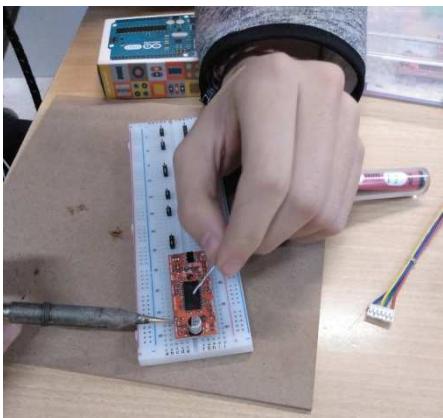
El nom d'aquest motor petit és 28BYJ-48 (annex E), és el motor pas a pas més bàsic i simple, per fer-lo funcionar vaig utilitzar el driver amb el que venia. Al ser un motor unipolar, tenia 5 cables, 4 per les bobines i 1 per l'alimentació, el qual els bipolars no utilitzen.. El driver necessitava dos pins per alimentació i 4 pins per a controlar els moviments.

Tot i que per al projecte final no em va servir, almenys vaig poder practicar una mica i provar el llenguatge que utilitza Arduino. També vaig aprendre a utilitzar la protoboard, les connexions amb els cables i els pins de la placa Arduino. Inclús en el cas de que tinguessin suficient força, si havia

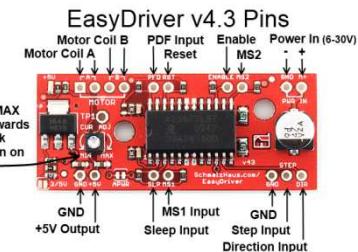
d'utilitzar 6 motors amb aquests drivers necessitaria 24 pins, i l'Arduino UNO (el model que jo utilitzava) no en té tants.

Més tard, vaig comprar uns altres drivers amb els qual podia controlar el servo només amb 2 pins (un per indicar el sentit de gir i l'altre els passos que ha de donar). A més aquest driver és compatible amb qualsevol motor pas a pas; és a dir, si em comprava uns altres també els podria utilitzar amb aquell driver.

Un cop ja havia acabat el Python i el Java, ja només em calia acabar aquesta part del procés i em vaig comprar 6 motors més grans i potents. Aquests nous motors (annex F) eren bipolars, per tant només tenien 4 cables, no calia un cinquè ni sisè extra per a l'alimentació.

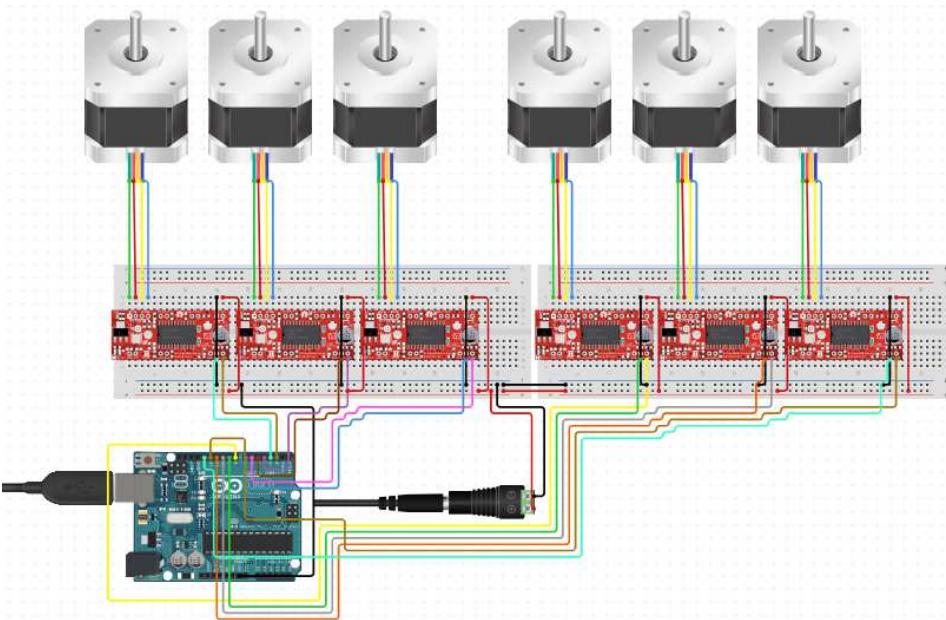


Per poder utilitzar els driver vaig haver de soldar els pins que em feien falta. Encara que el driver té moltes entrades per a pins, amb els motors només vaig haver de soldar els dos pins d'alimentació (connectats a una font d'alimentació), els 4 pins per connectar el motor (2 per cada bobina) i els 3 que anaven connectats al Arduino (GND, pin d'informació STEP i pin d'informació DIR).



## Muntatge

Vaig muntar el circuit amb Arduino tal i com es mostra a la imatge. Cada motor connectat a un driver, i els drivers connectats tots a la fond d'alimentació i a la placa d'Arduino.



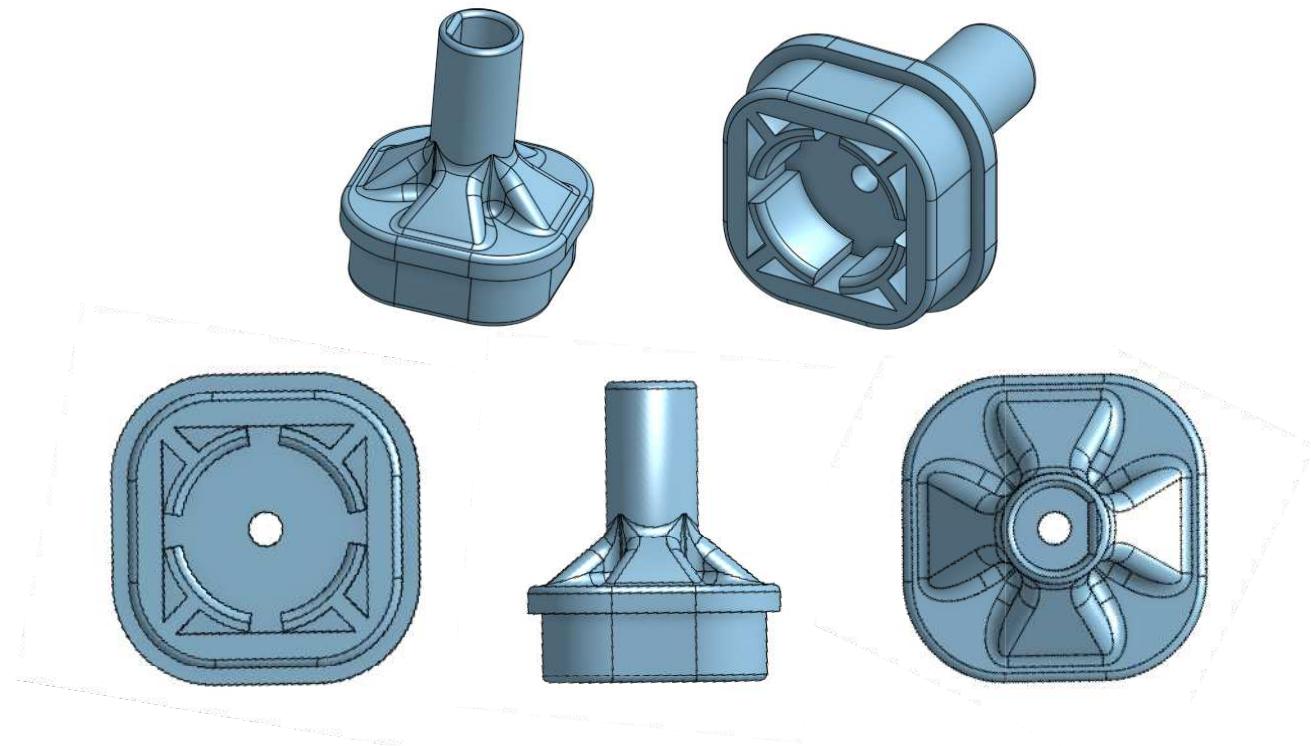
Al apartat següent explico com ho vaig fer per connectar els motors amb el cub.

Ara l'únic que falta es crear un suport per a que cada motor estigui a la seva cara corresponent.

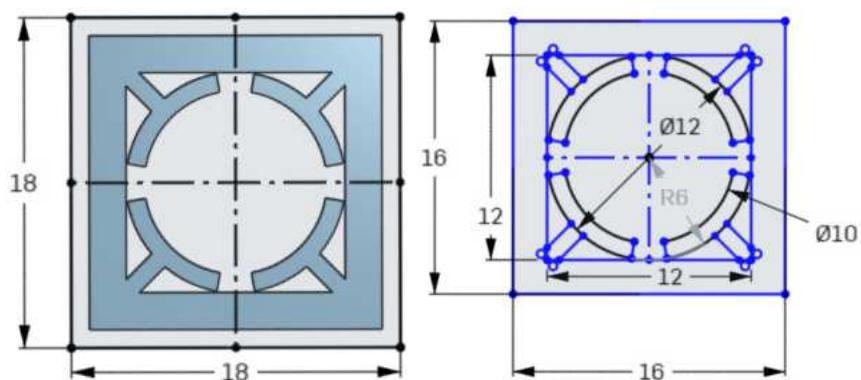
## Impressió 3D

Per a poder moure la cara del cub amb el motor pensava en posar el motor d'alguna manera enganxat al centre i que fes girar. Aquesta peça d'unió la vaig fer amb la ajuda d'un company amb disseny 3D. Vam utilitzar la eina online <https://cad.onshape.com>

Aquests són unes imatges de com es veia la peça al programa:



I les mesures:



Gràcies a aquesta peça he pogut transmetre el moment del motor a la cara del cub. Sembla una part una mica insignificant, però sense això no hagués pogut aconseguir el meu objectiu.

## Pressupost

Aquest és un recompte dels diners que he invertit en aquest treball.

Producte	Quantitat	Preus per unitat	Total
Arduino UNO	1	20,00 €	20,00 €
Protoboard	2	2,00 €	4,00 €
Pack 40 jumper wires	2	0,83 €	1,66 €
Stepper Motor + wire	1	13,99 €	13,99 €
Pack 5 Stepper motor	1	39,99 €	39,99 €
Pack 5 stepper motor wires	1	8,00 €	8,00 €
Pack 5 Easy Drivers	1	15,77 €	15,77 €
Easy Driver	1	6,99 €	6,99 €
			<b>Total</b>
			<b>110,40 €</b>

Per a que fos completament funcional, també caldia la font d'alimentació però no he considerat necessari posar-la. A mi me l'ha subministrat l'institut. També m'han deixat una placa Arduino UNO, però aquesta si que he considerat necessari apuntar-la, ja que és molt important en el projecte i forma part del robot en sí, no només de l'alimentació elèctrica.

## Propostes de continuïtat

M'agradaria poder acabar bé el que per temps o per coneixement no he pogut fer, o no tan bé com m'agradaria. Un cas clar d'això seria l'estructura que he utilitzat per aguantar els motors. Si que es veritat que era el que menys m'importava i que tal i com està, encara que no sigui molt elaborat funciona que és el que importa. Igualment, si fes una segona part del treball, definitivament l'estructura seria el meu principal objectiu.

Un altre objectiu seria acabar la opció d'utilitzar el mètode Fridrich per a resoldre el cub al Python, ja que reduiria bastant el número de moviments necessaris per resoldre el cub. Inclús estaria bé implementar més mètodes. En definitiva, qualsevol cosa que contribueixi a resoldre el cub en menys moviments o més ràpid seria un avenç.

Un altre punt que m'hagués fet molta il·lusió haver fet, però que se'm va ocórrer molt tard és fer una aplicació de mòbil i controlar el robot des del telèfon. Però no he tingut ni temps, ni coneixements per fer-ho.

Jo crec que aquests serien les principals propostes que jo faria per afegir més coses al treball. Inclús no descarto fer-ho algun dia com a projecte a nivell personal.

## **Conclusions**

Jo estic molt orgullós del que he aconseguit. Quan vaig començar, literalment, no sabia per on començar. Així que vaig començar per el que més o menys dominava, el Python. Durant l'estiu vaig dedicar moltes hores, inclús diria que més de les que sembla, a fer el Python. Al principi del treball no tenia molt clar si ho podria fer, però em veia capaç. Sabia que hauria d'esforçar-me molt i dedicar-hi molt de temps; però també sabia que era un tema que m'enganyava i que ho faria encantat.

Pel camí he tocat moltes àrees que ja tenia pensat que em farien falta, i també moltes que no m'ho esperava però m'han calgut. Des de programació amb 3 llenguatges diferents (2 de nous), soldar, ús de motors i font d'alimentació, impressió 3D... Sincerament estic molt content del que he après durant tot el projecte.

Durant la recta final, quan ja havia acabat les dues primeres parts i només em faltava fer la part física, ja començava a veure el meu objectiu més clar. Però es que abans d'això, a l'estiu i principis de curs, tenia masses fronts oberts i no veia molt clar el poder acabar-ho tal i com volia.

He d'admetre que durant l'estiu no vaig treballar tant com m'hagués agradat ni com m'hagués convingut. Només vaig fer la part Python, que la vaig acabar pràcticament a l'estiu. Entre això i simulacres he hagut de treballar molt a última hora i estic convençut que el resultat hagués sigut molt millor si ho hagués preparat amb més temps. Tot i així, estic molt content.

Sobre els objectius que volia assolir amb el treball i el que esperava aprendre, considero que he fet bastant, inclús més del que en un principi esperava. El que principalment buscava era aprendre nous llenguatges de programació i aprendre una mica de robòtica a ser possible amb Arduino. Crec que no només he complert els principals objectius, sinó que a més he après moltes més coses.

Crec que ha valgut la pena totes les tardes i nits que he passat davant de la pantalla del ordinador per a arribar a un objectiu que en un principi era aparentment impossible.

## Webgrafia

He visitat moltes webs per buscar com podia fer certes coses programant, però sobretot he vist molts vídeos de gent que feia coses similars per entendre millor com funciona alguna funció o simplement aprendre una mica d'un llenguatge nou. Per exemple, el cas de Java, m'he vist molts vídeos d'un canal que es diu '*The Coding Train*', que es un canal fa molts vídeos curtets resolent *Coding Challenges* principalment amb Java i p5.js utilitzant el programa *Processing*.

També he utilitzat dues pàgines molt importants són <https://github.com> i <https://stackoverflow.com>, d'on he pogut trobar solució a problemes que tenia amb errors dels programes. A aquestes dues pàgines la gent puja codis que ha fet, i la comunitat s'ajuda mútuament amb problemes que tenen. Stackoverflow està més orientat a gent que busca ajuda amb alguna funció que no li funciona. Fan un post explicant el seu problema i ensenyant la part del codi que falla, i la gent respon amb codis alternatius que podrien funcionar, corregint errors, donant consells... D'altra banda Github és un lloc on la gent pública codis que han fet, programes ja creats per ensenyar-los al món i a altres programadors. Aquestes dues pàgines m'han ajudat bastant en molts aspectes, hi ha molta informació sobre codi que l'he extret d'aquestes dues webs, inclús em vaig crear un compte de Github.

Informació sobre com fer certes coses amb Java:

- <https://geekytheory.com/como-crear-y-modificar-ficheros-con-java>
- [https://processing.org/reference/createWriter\\_.html](https://processing.org/reference/createWriter_.html)

Informació sobre els motors pas a pas:

- [https://es.wikipedia.org/wiki/Motor\\_paso\\_a\\_paso](https://es.wikipedia.org/wiki/Motor_paso_a_paso)
- <https://www.digikey.es/products/en/motors-solenoids-driver-boards-modules/stepper-motors/179?FV=ffe000b3&quantity=0&ColumnSort=1000011&page=1&k=stepper+motor&pageSize=25>
- <http://www.jss-motor.com/product.html>

Ús Arduino, drivers i motors:

- <https://es.paperblog.com/arduino-tutorial-22-control-de-un-motor-paso-a-paso-28-byj-48-4072450/>
- <http://adam-meyer.com/arduino/easydriver/>
- <https://www.circuito.io/app?components=9238,9238,9238,9238,9238,9238,9442,11021>

Mètode Fridrich:

- <http://coscorronderazon.blogspot.com/2009/11/metodo-fridrich-para-cubo-de-rubik-3x3.html>
- <https://blog.kubekings.com/algoritmos-de-oll-fridrich/>

Instal·lar mòduls al Python:

- [https://www.w3schools.com/python/python\\_pip.asp](https://www.w3schools.com/python/python_pip.asp)

Crear GIFs online:

- <https://ezgif.com/maker>

## **Agraïments**

Aquest treball no hagués sigut possible sense l'ajuda i el recolzament incondicional dels meus pares. Que tot i mostrar-se una mica escèptics sobre el projecte que pretenia fer el seu fill, sempre em van donar suport i tot i que no entenien que feia sempre m'ajudaven en el que podien. També per haver sigut els '*sponsors*' del projecte.

També vull agrair al meu tutor, en Xavi Damunt, per haver confiat en mi i haver-me ajudat sempre que ha pogut. Segurament un altre professor m'hagués dit que escollís millor un tema menys rebuscat i més senzill. Ell en canvi, no em va tallar les ales.

Una altra persona a la que vull donar-li les gràcies especialment, és a un company de classe i gran amic que m'ha ajudat molt en una part clau del projecte en la que sense la seva ajuda no ho hagués pogut aconseguir. Em va donar un molt generós cop de mà a l'hora de dissenyar les peces que havien de connectar el motor amb el cub, i vam acabar fent un disseny senzillament genial. Gràcies de tot cor, Alex!

També vull donar les gràcies a en David, un altre amic, que em va fer l'enorme favor d'imprimir-me les peces.

# **Annex**

## Annex A. Estudi del número de moviments

Apart de poder resoldre el cub, una altra cosa que volia aconseguir amb el Python, és poder fer una miqueta d'estudi sobre com funciona el programa de bé.

Utilitzant la opció de cultiu de dades, es pot desmuntar al azar el cub i tornar a muntar les vegades que vulguis i de manera automàtica. No cal dir-li res al programa, només al principi quantes vegades es vol que es faci aquest procés. Al activar-se, s'obre la opció 3 del menú (desordenar al azar) i la 5 alternativament i de manera automàtica. Això em va verificar que en principi el programa no falla, ja que he resolt el cub més de 50.000 vegades i no m'ha donat cap classe de problema.

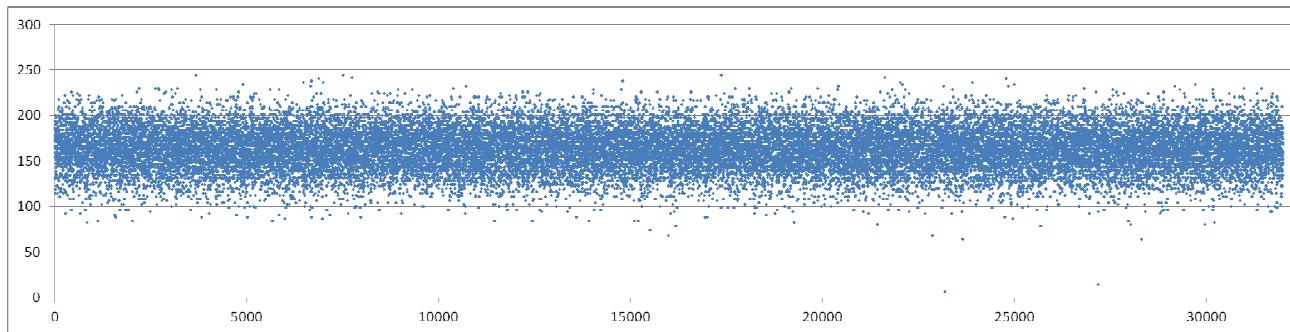
A demés, aquesta opció, compta el número de moviments que es fan per resoldre el cub en cadascuna de les vegades i les guarda en l'arxiu “Número\_moviments.txt”.

Així és com es veu quan s'acciona aquesta opció:

```
Mode cultiu a punt de començar
Quantes vegades vols executar-ho?
Possibles cultius: 5, 10, 20, 50, 100, 200, 500, 1000, 5000, 14000. 0 per sortir
del mode cultiu
```

Quan comença l'únic que es veu en pantalla és el número de resolucions que porta que va canviant a gran velocitat. I quan arriba al número programat, torna al menú principal del programa.

Utilitzant aquesta eina, vaig obtenir un document més de 50.000 números. Amb aquests número vaig fer una gràfica, per veure quin és el mínim i el màxim possible de moviments per resoldre el cub, i quins números són més comuns.



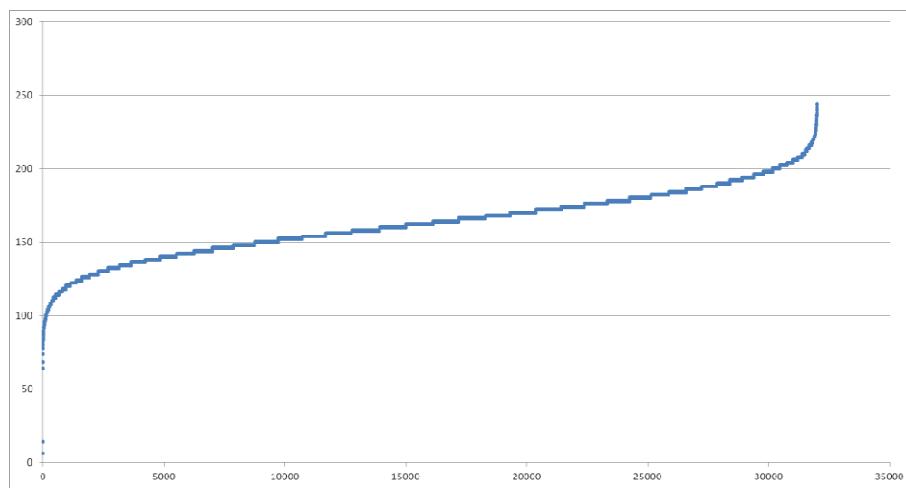
L'Excel admet com a màxim 32.000 dades en una gràfica, així que vaig arribar al màxim i no em va deixar posar més. Igualment, ja tenim una molt bona gràfica.

D'entrada veiem que cap punt passa de 250 i que la majoria estan entre 110 i 210 més o menys. Entre 80 i 110 també hi i ha bastants però no tants, igual que entre 210 i 240. També podem

apreciar la improbabilitat estadística de que et surti per casualitat un estat pràcticament resolt que necessiti només menys de 50 moviments. D'entre 32000 punts, només 2 estan per sota de 50.

D'aquesta selecció de 32000 mostres, la mitjana és 162,76681. Tenint en compte que la mitjana de moviments requerits pel mètode de principiants ronda els 160, podríem dir que el programa està bastant ben optimitzat, no fa molts moviments innecessàries, i és bastant fidel a aquest mètode. El valor més alt és 244, i el més petit és 6.

També vaig crear una altra gràfica. Les dades eren les mateixes, però en aquest cas estaven ordenades de menor a major.



El que podem interpretar d'aquesta gràfica és la quantitat de resultats que hi ha hagut per cada número de moviments aleatoris al desmuntar. Com més petit és el pendent, és a dir, més “plana” està la gràfica, significa que hi ha més mostres que utilitzen aquest número de moviments. I com més pendent té, menys resultats hi ha amb aquell número de moviments.

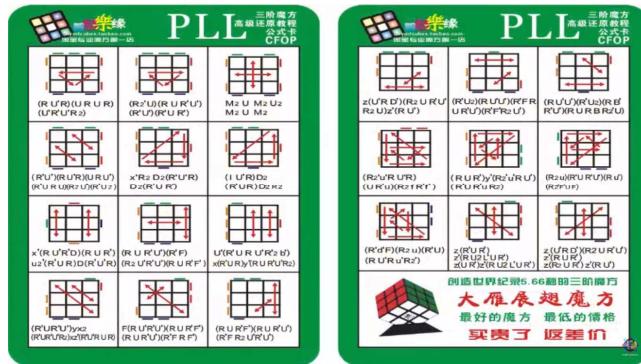
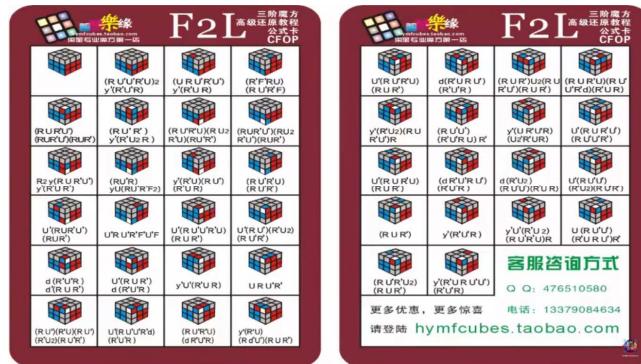
Una característica de la gràfica que em va sobtar una mica al principi, es que la línia no es contínua i regular, sinó que fa com una escala. Al principi pensava que era degut a que havia masses dades i no carregava bé. Però després em vaig adonar de que no, simplement és així perquè només hi ha números enters a la gràfica. Al no haver decimals, fa un “saltet” cada cop que canvia de número.

En conclusió, el programa utilitza el mètode de principiants i resol el cub en un mínim teòric de 6 (encara que segurament, tot i que molt improbable pugui ser encara menor) i un màxim teòric de 244. El més probable, és que el resolgui entre 140 i 190 més o menys.

## Annex B. Mètode Fridrich

El mètode de resolució del cub de Rubik Fridrich és un mètode creat per la enginyera elèctrica txecoslovaca Jessica Fridrich. El que aconsegueix aquest mètode és reduir significativament el número de moviments , empleats en la resolució; això sí, hi ha molts més algoritmes, i més específics i complexos. El procediment és el mateix que al mètode de principiants, comença per una cara, completa el primer nivell, el segon i finalment la última cara. Però respecte a aquest mètode, el mètode Fridrich presenta una àmpliament major dificultat i varietat d'algoritmes.

Té 3 parts: First 2 Layers (F2L), Orientation of the Last Layer (OLL) i Permutation of the Last Layer (PLL). En les següents imatges es mostren els algoritmes de cada fase.



## Annex C. Com instal·lar mòduls al Python

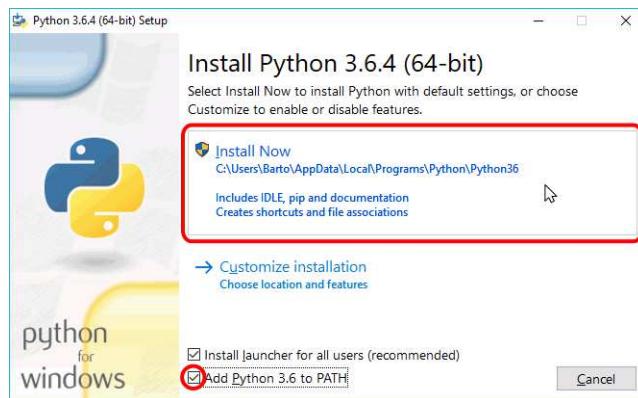
Per a varies opcions que vaig utilitzar al Python calien mòduls. Un mòdul és com una extensió del programa que et permet fer certes coses i utilitzar certes funcions que sense aquest mòdul no es podrien fer. Per a utilitzar un mòdul, al codi del Python l'has d'importar primer amb el comandament 'import'.

```
import random
```

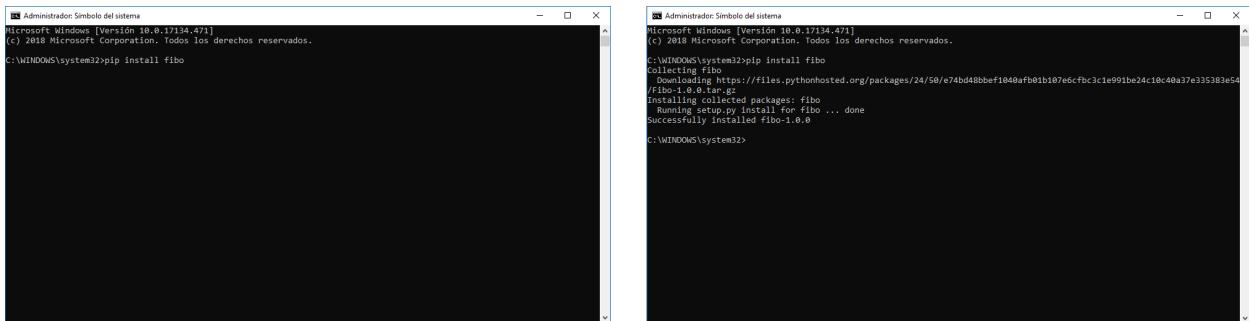
Un cop importat ja es pot utilitzar, si es tracta d'utilitzar sense importar-lo dona error.

Molts d'aquests mòduls, sobretot els més utilitzats i d'ús més genèric ja estan instal·lats per defecte. Però per a poder controlar l'Arduino des del Python em calia un mòdul anomenat serial i l'havia d'instal·lar. Per fer-ho s'havia de fer des de el CMD (símbol del sistema) executat com a administrador i utilitzant la funció pip, però no em funcionava. Vaig estar molt temps provant i no m'anava. Després d'uns dies per fi vaig trobar l'error: quan em vaig instal·lar el Python no vaig acceptar la opció que et deixa utilitzar nous comandaments al símbol del sistema que implementa Python.

Doncs, per a poder instal·lar un mòdul el primer és tenir el Python ben instal·lat. A l'hora d'instal·lar s'ha d'acceptar la opció "Añadir Python 3.6 al PATH".



Ara ja podia descarregar-me mòduls des del CMD. Havia d'utilitzar el següent comandament:

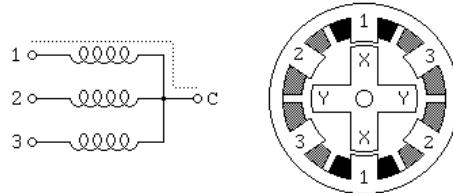


Ara ja podem utilitzar el nou mòdul!

## Annex D. Funcionament dels motors pas a pas

El motor pas a pas és un motor que utilitza l'electromagnetisme per crear moviment circular. Són motors molt precisos, però no són els que tenen el major parell precisament. S'utilitzen en robòtica, impressores digitals, drons, etc.

El seu funcionament es basa en les bobines que té al voltant d'una peça en forma d'estrella. Crea un camp magnètic en dos pols i fa que s'alineïn. Al imantar el parell següent es torna a alinear girant molt poquet. Fa aquest procés successivament per crear el moviment.



Durant el treball, he utilitzat motors pas a pas monopolars i bipolaris, la diferència entre els dos és principalment l'alimentació. Els motors unipolars poden tenir 5 o 6 cables de sortida, un per l'alimentació i els altres 4 per les bobines, també són més simples de controlar. El bipolar, en canvi, només en té 4 normalment.

Taula d'ordre de les fases d'un motor bipolar (sentit horari)

Pas	A+	A-	B+	B-	Imatge
1	+	-			
2	+	-	+	-	
3			+	-	
4	-	+	+	-	
5	-	+			
6	-	+	-	+	
7			-	+	
8	+	-	-	+	