Introducció

Ja se el que posare, tinc un petit esbos, informació sobre la historia del cub, qui el va inventar, etc... Curiositats com que es la joguina mes venuda de la historia... Una mica de estudi matematic: els possibles estats que te, el numero de deu...

Durant aquesta setmana faig la intro i te la envio

Procés tecnològic

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.

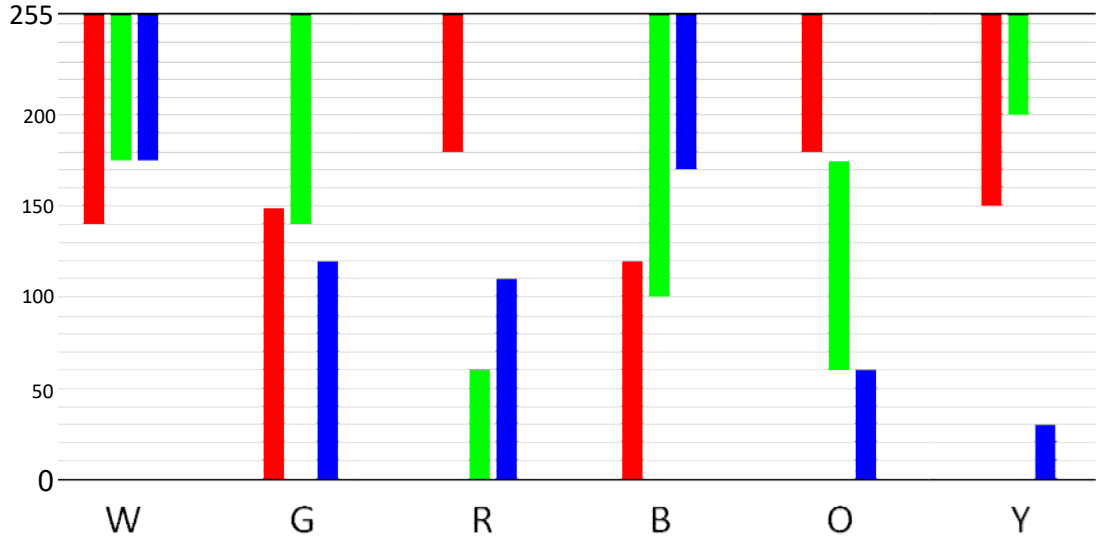
Apart d'aquests tres processos principals, el procés tecnològic consta de més processos intermediaris i secundaris, com podrien ser el fer les fotos de les cares del cub o transmetre les dades entre els programes.

Encara he de posar mes coses aqui

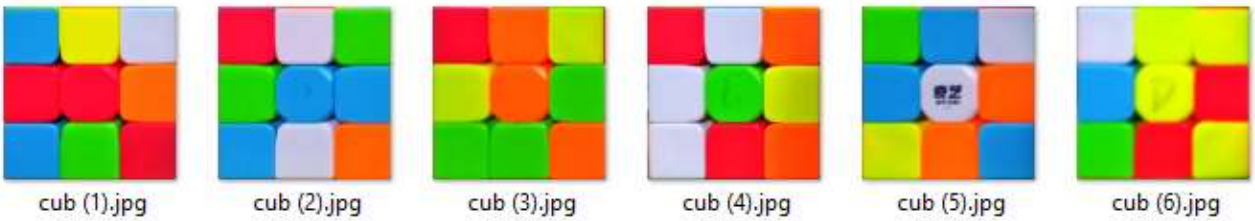
Detecció de colors

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

Deteccio_color

```
PImage cara;
String col;
int mult;

int EW=0;
int ER=0;
int EB=0;
int EO=0;
int EG=0;
int EY=0;

void setup() {
  size(220, 315);
  line(0,155,width,155);
}

void draw() {
  loadPixels();
  for (int i = 1; i <= 6; i++) {
    for (int j = 0; j < 3; j++) {
      for (int k = 0; k<3; k++) {
        cara = loadImage("cub ("+i+").jpg");

        float R1 = red(cara.pixels[((cara.width/6)+cara.width/24+j*cara.width/3)+cara.width*(cara.height/6+k*cara.height/3)]);
        float R2 = red(cara.pixels[((cara.width/6)-cara.width/24+j*cara.width/3)+cara.width*(cara.height/6+k*cara.height/3)]);
        float R0 = red(cara.pixels[((cara.width/6)+j*cara.width/3)+cara.width*(cara.height/6+k*cara.height/3)]);
        float R = (R0+R1+R2)/3;

        float G0 = green(cara.pixels[((cara.width/6)+cara.width/24+j*cara.width/3)+cara.width*(cara.height/6+k*cara.height/3)]);
        float G1 = green(cara.pixels[((cara.width/6)-cara.width/24+j*cara.width/3)+cara.width*(cara.height/6+k*cara.height/3)]);
        float G2 = green(cara.pixels[((cara.width/6)+j*cara.width/3)+cara.width*((cara.height/6+cara.height/15)+k*cara.height/3)]);
        float G = (G0+G1+G2)/3;

        float B0 = blue(cara.pixels[((cara.width/6)+cara.width/24+j*cara.width/3)+cara.width*(cara.height/6+k*cara.height/3)]);
        float B1 = blue(cara.pixels[((cara.width/6)-cara.width/24+j*cara.width/3)+cara.width*(cara.height/6+k*cara.height/3)]);
        float B2 = blue(cara.pixels[((cara.width/6)+j*cara.width/3)+cara.width*(cara.height/6+k*cara.height/3)]);
        float B = (B1+B2+B0)/3;

        if (j==1 && k==1) {
          mult = 1;
        }
        else {
          mult = 0;
        }

        fill(R,G,B);
        if (i < 5) {
          ellipse(10+55*(i-1)+15*j,60+ 15*k,10+2*mult,10+2*mult);
        }
        if (i == 5) {
          ellipse(10+15*j,10+15*k,10+2*mult,10+2*mult);
        }
        if (i == 6) {
          ellipse(10+15*j,110+15*k,10+2*mult,10+2*mult);
        }
```

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 estar més segurs

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 te 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.

```
        fill(0);
        if (R>=140 &&G>=175 &&B>=175) {
          col="W";
          EW++;
          fill(255);
        }
        if (R>=180 && G>=60 && G<=175 && B<=60) {
          col="O";
          EO++;
          fill(255,130,0);
        }
        if (R<=120 && G>=100 && B>=170) {
          col="B";
          EB++;
          fill(0,0,255);
        }
        if (R>=150 && G>=200 && B<=30) {
          col="Y";
          EY++;
          fill(255,255,0);
        }
        if (R<=149 && G>=140 && B<=120) {
          col="G";
          EG++;
          fill(0,255,0);
        }
        if (R>=180 && G<=60 && B<=110) {
          col="R";
          ER++;
          fill(255,0,0);
        }

        if (i < 5) {
          ellipse(10+55*(i-1)+15*j,160+60+ 15*k,10+2*mult,10+2*mult);
        }
        if (i == 5) {
          ellipse(10+15*j,160+10+15*k,10+2*mult,10+2*mult);
        }
        if (i == 6) {
          ellipse(10+15*j,160+110+15*k,10+2*mult,10+2*mult);
        }
      }
    }
  }

  if (EW>=8 && EY==9 && EG==9 && ER==9 && EB==9 && EO==9) {
    stroke(0,255,0);
    strokeWeight(5);
    noFill();
    ellipse(width-30,height-30,30,30);
  }
  else {
    stroke(255,0,0);
    strokeWeight(5);
    line(width-45,height-45,width-10,height-10);
    line(width-10,height-45,width-45,height-10);
  }
  saveFrame("Detecció color.jpg");
  noLoop();
}
```

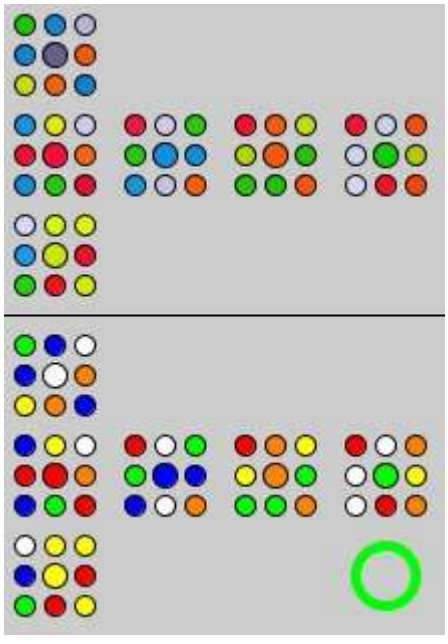
Depenent del rang del vermell, verd i blau de cada quadradet, es diu el seu color pertinent entre els sis possibles.

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.)

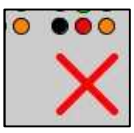
Guarda la imatge creada



Colors directament extrets de les fotos.

Colors detectats pel programa

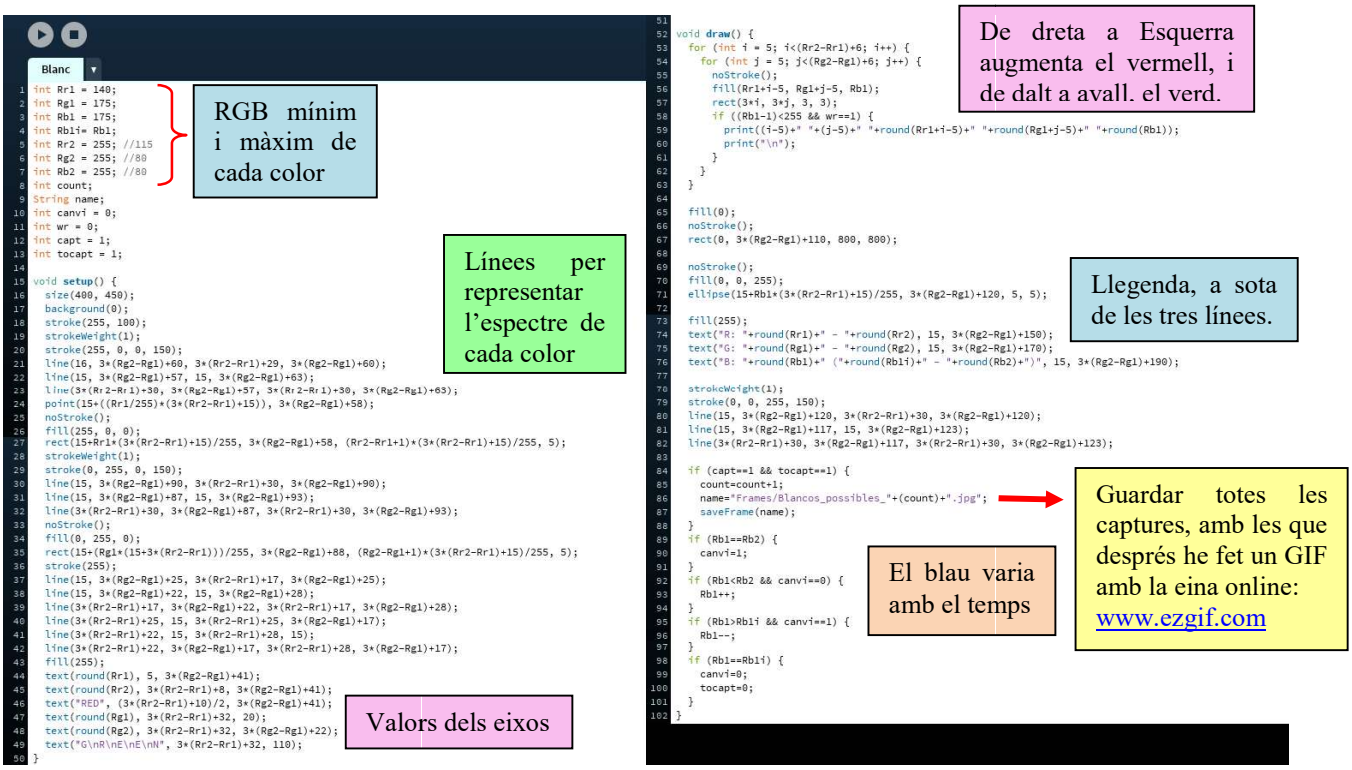
El cercle significa que en principi pot ser que sigui correcta la detecció, ja que hi ha el mateix número de tots els colors.
En cas contrari, es mostra una creu vermella.



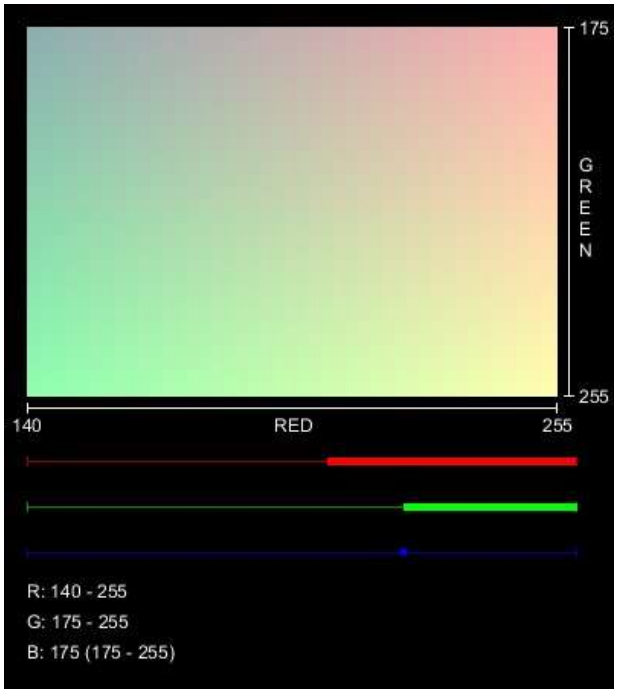
Imatge generada pel programa

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.



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 i menús.

Fare algo tipo lo de l'apartat anterior, pero com que el codi es molt llarg, només amb el que es veu quan executes el programa, i un esquema del que he fet per que es resolgui sol, ja que no l'he buscat enlloc, l'he fet jo

I per acabar, fare un apartat extra, parlant de l'arduino, els servos (primera opcio que vaig descartar), els stepper motors (els primers que vaig utilitzar i els nous), i tot el que he fet per a que funcioni