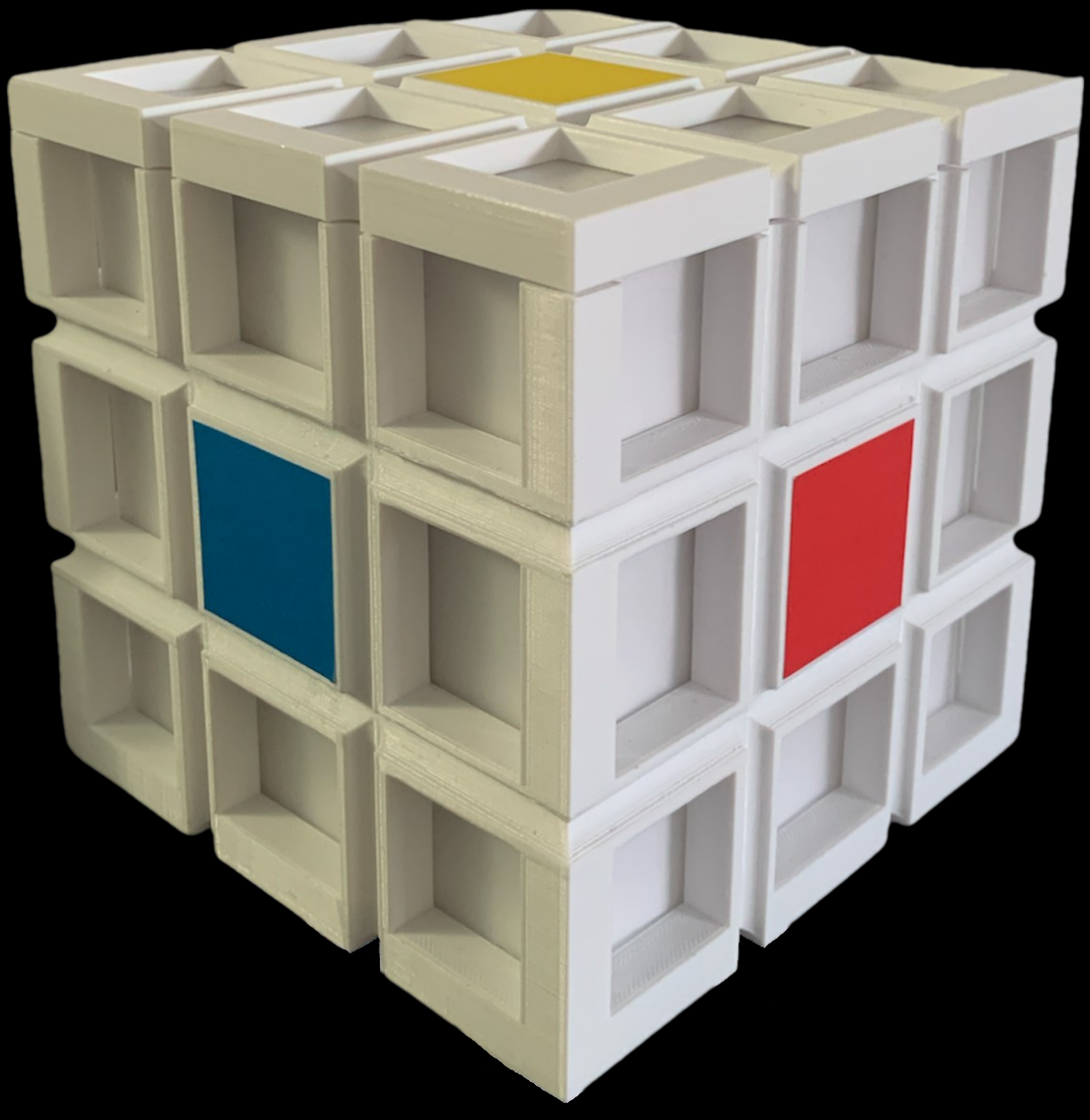
Treball de Recerca

El cub de Rubik

Una solució tecnològica-visual



F2L

AGRAÏMENTS

En primer lloc m'agradaria agraïr enormement l'ajuda i consell que m'han donat en tots els aspectes relacionats amb la programació i l'elaboració del codi.

Vull donar les gràcies també al meu tutor de recerca que m'ha acompanyat durant tot el procés i m'ha ajudat amb tot el que ha pogut, sobretot en els aspectes més formals del treball.

També m'agradaria agraïr als amics que m'han cedit algun material del qual jo no disposava.

Finalment m'agradaria donar les gràcies a la meva família per haver confiat en mi i en el meu projecte, haver refermat les meves idees i ajudat a veure algunes coses des de noves perspectives.

RESUMEN

Partiendo del interés por los métodos de resolución del cubo de Rubik que sirven para solucionarlo de una manera fácil, el presente trabajo ha tratado de conseguir un prototipo de cubo de Rubik que ayude a resolver el puzzle a cualquier persona sin importar su nivel.

Gracias a la impresión 3d se ha construido un prototipo que, mediante la iluminación de las piezas que deben moverse en cada paso, muestra la resolución del rompecabezas. Se trata de una caja grande con forma de cubo de Rubik 3x3, en el interior de la cual se halla una placa base Raspberry Pi conectada a varios LEDs. A través de estos se genera una ayuda visual para mostrar los pasos de resolución a seguir desde un cubo de Rubik normal.

Para un correcto funcionamiento se ha creado un programa informático con el lenguaje de programación Python que transforma la solución del cubo en señales lumínicas a través de la placa Raspberry Pi para indicar a los usuarios los distintos pasos a seguir. Para ello se utiliza un programa que permite introducir la situación inicial del cubo y genera una de las soluciones más óptimas, siguiendo el algoritmo de resolución de Kociemba.

En conclusión, se ha conseguido crear un prototipo que dé ayuda a la resolución del cubo de Rubik de una manera intuitiva.

ABSTRACT

Starting from an interest in the resolution methods of Rubik's cube and in order to solve the cube in an easy way, the intention of this research work was to create a prototype of a Rubik's cube that can help anyone to solve this puzzle regardless of their level.

Thanks to 3d printing, a prototype that shows the resolution of this brainteaser by illuminating the pieces that have to be moved in each step was built. Broadly speaking, this prototype is a big box with the shape of a Rubik's cube 3x3 in whose interior there is a Raspberry Pi motherboard that is connected to several LEDs which give some visual help. Its purpose is showing the resolution steps to follow from a normal Rubik's cube.

A computer programe was also created with Python programming language which transforms the solution of the cube into light signals through the Raspberry Pi motherboard to indicate the users the different steps to follow. In order to to get this solution, firstly it is necessary to use a program in which the initial situation of the cube can be entered in order to obtain the most optimal solution following the Kociemba's resolution algorithm.

In conclusion, a prototype that helps to solve the Rubik's cube in an intuitive way was created.

Keywords: Rubik's cube, Python, Raspberry Pi, 3d printing, Rubik's cube solution, easy resolution for the Rubiks cube, light solution, visual Rubik's cube solution, Kociemba's algorithm, optimal solution for the Rubik's cube, motherboard.

Índex

Índex de Figures

0. INTRODUCCIÓ

El cub de Rubik és un trencaclosques que, qui més qui menys, coneix i segurament ha provat de resoldre alguna vegada, però no tothom ha estat capaç d'aconseguir-ho, probablement a causa de l'enorme quantitat de configuracions que permet aquest puzle i de la falta de coneixements del funcionament, girs i translacions. Algunes persones que no són capaces d'aconseguir l'objectiu final a vegades busquen ajuda per arribar-hi. Avui en dia, aquesta ajuda la poden trobar fàcilment a través de plataformes digitals d'una manera tan fàcil com pot ser YouTube. Però la manera més personalitzada, i en conseqüència més entenedora, continua essent que alguna persona que té els coneixements per resoldre el cub et guiï i te n'ensenyi. És per aquest motiu que he encarat el meu treball a la construcció d'una espècie de caixa amb la forma d'un cub de Rubik que a través d'uns LEDs interiors et guiï cap a la solució mitjançant la il·luminació de les peces que s'hauran de moure.

Un primer objectiu del treball va ser aconseguir fer funcionar un programa creat prèviament per un desenvolupador extern per tal de poder determinar una solució del cub en un estat desordenat per tal de poder-lo resoldre.
El segon objectiu proposat consistia en dissenyar un programa informàtic que transformés aquesta solució donada pel programa en una senyal que permetés als LEDs engegar-se i apagar-se de la manera adient per tal d'ensenyar els moviments a seguir per aconseguir resoldre el cub. Per realitzar aquest pas va ser necessari aprendre un llenguatge de programació.
Un tercer objectiu va ser el disseny de la capsa amb forma de cub on hi ha tots els elements que permeten el correcte funcionament del projecte, com poden ser LEDs i plaques base, i que es va imprimir amb una impressora 3d.
Un quart objectiu imprescindible va ser comprendre com és possible enviar senyals a diferents LEDs per aconseguir que només s'engeguessin els que volguéssim nosaltres.

Però l'objectiu final va ser aconseguir crear un aparell que et mostrés la solució de qualsevol situació del cub d'una manera que fos intuïtiva de resoldre per a una persona que no sap massa com resoldre'n un.

Personalment, vaig triar aquest treball ja que tinc una gran afició en aquest trencaclosques en totes les seves versions. Cadascuna d'aquestes versions t'ajuda a entendre i a tenir una visió més àmplia per enfocar els problemes que et pots trobar més endavant en altres trencaclosques, desenvolupant així una manera de pensar que et permet resoldre trencaclosques d'una manera més senzilla aplicant els coneixements apresos. D'altra banda, percebo que hi ha bastanta gent que no sap com arribar a la solució i deixen de banda aquest joc que a més a més de tenir una capacitat per passar una bona estona, ens ajuda a millorar la nostra intel·ligència espacial, tal com van concloure Maryam Delavar, Iryna V. Kolesnikova, Beheshte Rahimzade, Mahdi Ghahhari, Alimorad Mosapuor i Ali Moradi al seu estudi "Development of mental rotation ability at primary school level" (2018). És per aquest motiu que penso que si hi hagués algun aparell que per si sol o amb molt poca ajuda t'ajudés a resoldre'n un, hi hauria molta més gent que descobriria i gaudiria d'aquest magnífic puzle.

Penso que aquest treball pot tenir una rellevància ja que pot ser una eina de motivació per a aquelles persones que els fa una mica de por o respecte enfrontar-se al cub de Rubik, i d'aquesta manera acostar una mica més la possibilitat d'una millora en la intel·ligència espacial de les persones. A més a més, podria ser una eina que donés confiança per intentar fer un progrés cap a la solució o simplement cap a la comprensió dels moviments del cub sense tenir la por que es quedi el cub sense resoldre la resta de la vida, ja que sabries que en tot moment podries tornar a la posició inicial.

Per realitzar aquest treball s'ha treballat a través del programa de programació Python, les plaques base "Raspberry Pi", el programa de modelat 3d "Autodesk Fusion 360" per dissenyar el cub i el programa "Cura" per preparar la peça per imprimir-la en 3d.

Primerament, em centraré en explicar els orígens, el funcionament, mètodes de resolució i diferents estats possibles del cub de Rubik. A continuació explicaré els components electrònics utilitzats i faré una petita pinzellada al món de la impressió en 3d.

Tot seguit, parlaré de com ha estat el procés de creació del meu prototip de cub, des del disseny a l'impressió en 3d, passant per la creació del codi i programació per acabar amb el muntatge i tests.
Finalment, recolliré els resultats per analitzar i extreure les conclusions d'aquest projecte.

MARC TEÒRIC

1. EL CUB DE RUBIK

1.1. Origen i història

El cub de Rubik és la joguina més venuda de tota la història, i és que aquest ingeniós trencaclosques inventat per l'hongarès i professor de la universitat de Budapest, Ernő Rubik l'any 1975 va ser capaç de deixar marca en les vides de la gent de l'època i d'avui en dia.

Quan Rubik va crear el seu famós cub, no ho va pas fer amb la intenció de crear una joguina o un trencaclosques, sinó que ho va fer perquè els seus alumnes no entenien del tot bé com s'aguantaven les dovelles[1] d'un arc de mig punt. Ell, per explicar això, va inventar-se una estructura de 6 eixos per fer-ne un símil. La peça que estava unida a cada eix (els centres) representaven les claus[2] mentre que les altres peces representaven les dovelles. Quan va deixar el cub als alumnes perquè poguessin veure com s'aguantaven aquests van girar algunes cares, i quan li van tornar el cub a Ernő va fixar-se que el patró de les peces de fusta havia quedat desfet i va intentar resoldre'l. Va ser així com es va adonar que havia creat un trencaclosques realment difícil i no va ser capaç de resoldre'l. Un dia va decidir pintar les cares de diferents colors i va portar varis cubs als seus alumnes. Actualment els colors del cub són blanc, groc, verd, blau, vermell i taronja i sempre estan disposats de la mateixa manera. Si ens fixem amb parelles oposades de colors trobem blanc-groc, verd-blau i vermell-taronja. Va ser d'aquesta manera com l'any 1975 va patentar el "Cub Màgic" i va tenir un gran èxit, sobretot a Hongria.

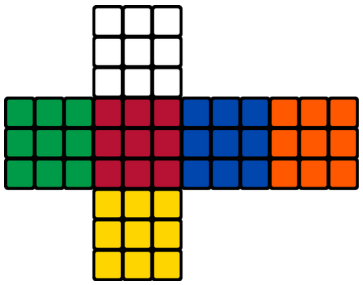


Figura 1. Representació plana del cub de Rubik amb els seus colors.
Font: Wikimedia Commons

1 Cadascun dels blocs de pedra en forma trapezoïdal que formen arcs, voltes...
2 Dovella central d'un arc.

Un temps més tard, un americà, representant d'una companyia de joguines, li va comprar la patent del cub i això fa que a principis dels anys 80 es comencés a comercialitzar mundialment. Va ser així com es va convertir en la joguina més venuda de la història.

Des d'aquell moment el cub ha anat millorant pel que fa al gir i han aparegut nous cubs amb més o menys peces per cada cara, amb diferents formes i fins i tot s'ha adaptat per a persones amb discapacitats visuals. Més recentment també s'han creat cubs intel·ligents que mitjançant Bluetooth es connecten als dispositius mòbils o tauletes i permeten veure exactament què és el que passa al cub, t'ensenyen a resoldre'l mitjançant lliçons interactives, pots competir en línia amb altres persones a veure qui fa el millor temps o fins i tot pots jugar a mini-jocs interactius. És rellevant destacar que tota aquesta varietat que trobem avui en dia és gràcies a que els drets d'autor es van acabar l'any 2000 i altres fabricants van començar a poder fabricar els seus propis cubs de Rubik.

1.2. Configuracions del cub de Rubik

El cub de Rubik, malgrat la seva aparença senzilla, té un nombre gegant de possibles configuracions. Primerament, hem d'observar com està format, és a dir, quin tipus de peces té. A cada cara hi ha 9 quadres, el central dels quals manté la seva posició relativa amb els altres centres[3] facis les rotacions que facis. Per tant, podem descartar aquesta peça a l'hora de mirar les possibles configuracions. Ara ens fixem en les peces perifèriques i veiem que hi ha 2 tipus de peça, les arestes i les cantonades. Aquestes peces però, no estan formades per un sol color i comparteixen un altre color si són arestes i dos altres colors si són cantonades. Els colors d'una peça és impossible que varïn mai. Així doncs, si agafem tot el cub en global veiem 12 arestes i 8 cantonades. A més a més, cal afegir que quan fem qualsevol tipus de rotació les arestes van a un lloc on hi havia una altre aresta i les cantonades a un lloc on hi havia una cantonada, per tant, és impossible que es barregin.

3 Peça que està al quadre central de cada cara.

A partir d'aquestes observacions ja podem començar a fer els primers càlculs per a esbrinar quantes configuracions té el cub de Rubik. Com que tenim 8 cantonades o vèrtexs que poden estar en 8 posicions diferents fem el següent raonament: La primera cantonada podrà estar en qualsevol dels 8 llocs disponibles, la segona en qualsevol dels 7 restants, la tercera en les 6 que queden i així fins que ens quedem sense, per tant, podem expressar les possibles posicions de les cantonades com a 8!. A més a més, cada vèrtex està format per 3 colors, per tant haurem de multiplicar per 3^8 aquest 8! ja que cadascuna de les 8 cantonades podrà estar col·locada de 3 maneres diferents.

Tot seguit farem el mateix amb les arestes, per tant, seguint el mateix raonament esmentat amb els vèrtexs podem dir que les possibles confuguracions de les cantonades són 12!. I com que aquestes estan formades per a 2 colors, haurem de multiplicar per 2^{12} el 12! tal i com hem fet anteriorment amb els vèrtexs.

Si operem amb aquestes dades obtenim una cosa així:

$$8! \cdot 3^8 \cdot 12! \cdot 2^{12} = 5.19 \times 10^{20}$$

Però aquestes no són les configuracions totals que té el cub ja que algunes d'aquestes no són possibles. Per exemple, és impossible tenir tot el cub resolt menys una cantonada que estigui girada, i el mateix passa amb les arestes. Per tant hem de dividir aquest nombre per 3 i per 2. I encara haurem de tornar a dividir per 2 ja que 2 arestes no poden estar intercanviades sense que la resta del cub s'immuti. I finalment, amb aquestes noves normes ja podem operar i obtenim la següent xifra:

$$\frac{8! \cdot 3^8 \cdot 12! \cdot 2^{12}}{3 \cdot 2 \cdot 2} = 4.325 \times 10^{19}$$

Per tant, en un cub de Rubik de 3x3 [4] tenim un total de 43.252.003.274.489.856.000 possibles configuracions (aproximant, 43 trilions de configuracions possibles).

Per fer-nos una idea de la gran quantitat que és aquest nombre podem plantejar un seguit de situacions que ens poden ajudar a veure-ho.

Primerament, suposarem que cada segon fem un moviment i aconseguim una configuració que encara no havíem fet. Per tant passarem aquests 43 trilions de segons que necessitaríem per a aconseguir passar per totes les possibles configuracions en anys i obtenim la quantitat de 1,36 bilions d'anys, és a dir, 100 vegades més que l'edat de l'univers. Per tant, si haguéssim començat a moure el cub just a l'instant que es va produir el Big Bang encara ara estaríem movent el cub.

Una altre hipotètica situació que també va molt bé per il·lustrar com d'enorme és aquest nombre és imaginar-nos que agafem tants cubs com configuracions té i els posem un al costat de l'altre. Si poguéssim fer això amb cubs estàndards de 56 mm aconseguiríem cobrir una distància de 256 anys llum.

1.3. Mètodes de resolució

Tal i com és fàcil d'imaginar, amb la gran quantitat de possibles configuracions del cub, no hi ha una sola manera de resoldre'l, sinó més aviat al contrari. Trobem una gran quantitat de mètodes per aconseguir arribar a l'objectiu final del trencaclosques, cadascun bo per algunes coses i més deficient per d'altres. Tot seguit explicaré alguns dels mètodes més destacats, ja sigui per la seva simplicitat, per la seva possibilitat de resoldre el cub en escassos segons o per resoldre el cub en molts pocs moviments.

4 El cub de Rubik més conegut de tots, el que té un total de 9 peces a cada cara, 3 a cada costat.

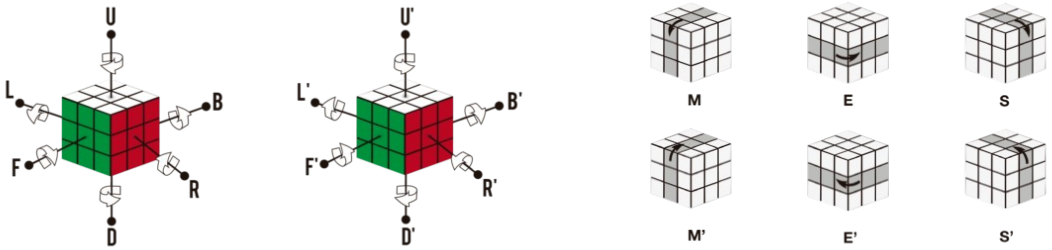
1.3.1. Notació

Per tal de millorar la comprensió i classificació de quin tipus de moviment trobem en el cub i d'aquesta manera permetre una millor transmissió d'informació quan volem referir-nos a uns moviments concrets, sobretot utilitzats a l'hora d'aprendre nous algoritmes[5], hi ha establerta una notació.

Aquesta notació anomena cada cara segons la seva posició amb la lletra inicial en majúscula d'aquesta (en anglès), de tal manera que tenim aquestes cares: U (Up), D (Down), R (Right), L (left), F (front) i B (back).

A més a més, per tal de definir cap a quin costat cal girar la cara s'utilitza el sentit de les agulles del rellotge. Si el moviment és horari s'escriu la lletra sola sense res (U), mentre que si és un moviment antihorari s'escriu la lletra amb una cometa a dalt a la dreta (U') i en aquest cas es llegiria "U prima".

També hi ha un altre tipus de moviments que són molt menys freqüents però que també val la pena mencionar. Es tracta dels girs de les capes intermitges i s'anomenen: M, per designar la capa que es troba entre les capa R i L; E per a la capa d'entre la U i D; i S per a la capa que es troba entre la F i B. D'aquestes 3 capes la més utilitzada és la M.



Figures 2-6. Representació dels moviments del cub. Font: Kubekings

5 Conjunt de passos a seguir de manera ordenada per tal d'arribar a un objectiu final a partir d'una situació inicial concreta.

1.3.2. Mètode per a principiants

Aquest és un dels mètodes de resolució més fàcils i senzills de recordar, i és per aquest motiu que ha adoptat aquest nom, ja que és el mètode més utilitzat les primeres vegades en que s'aprèn a resoldre el cub. És un mètode bastant intuïtiu i els pocs algoritmes que té són molt curts i fàcils de recordar. De fet, aquest mètode és la simplificació d'un altre més complex anomenat Fridrich.

La resolució s'estructura en aconseguir anar construint el cub capa a capa, començant des de baix i fins a dalt.

El primer que s'ha d'aconseguir és formar una creu, que es fa d'una manera intuïtiva, en una cara, de tal manera que les arestes coincideixin en colors amb les dues cares que toca.

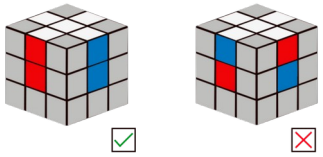


Figura 7. Representació de la creu blanca amb les arestes al lloc corresponent. Font: Kubekings

Tot seguit es posen els vèrtex de la cara de tal manera que tinguem una cara complerta i que els colors perifèrics també encaixin. Després, com que ja tindrem una capa complerta, començarem a completar la següent capa. Per completar aquesta o bé necessitem 2 algoritmes molt semblants i senzills o bé una gran intuïció que malauradament no tothom disposa. El que fem en aquest pas és posicionar al seu lloc les diferents arestes que formen la segona capa.

A continuació, haurem de resoldre l'última capa. Aquí és on els algoritmes ja tenen un pes bastant important ja que és molt fàcil desfer tot el progrés aconseguit si no es vigila una mica. El primer que haurem de fer per resoldre aquesta capa és fer una creu a la cara superior. Per això necessitarem un o dos algoritmes, depenent de la velocitat i quantitat de passos que es vulguin fer, tot i que varien molt poc i són fàcils de memoritzar. En segon lloc, haurem d'ordenar les arestes per tal de que ja tinguin la relació de colors que hauran de tenir quan el cub estigui resolt. Per a fer això serà necessari un altre algoritme.

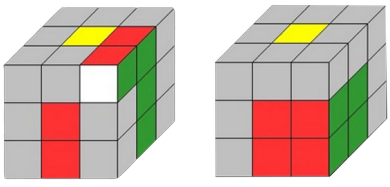
Ja per anar acabant haurem de permutar[6] les cantonades i serà necessari un altre algoritme. Un cop estiguin al lloc correcte, i ja per acabar, les haurem d'orientar mitjançant un seguit de moviments que no es poden considerar ni algoritme.

En resum, aquest mètode és molt útil per a persones que s'inicien en aquest extens món, ja que només consta d'uns 6 algoritmes molt senzills que són molt fàcils d'aprendre i entendre, que sol ser una part molt important en el procés d'aprenentatge.

1.3.3. Mètode CFOP

El mètode CFOP (de les sigles en anglès de *Cross, F2L*[7]*, OLL*[8]*, PLL*[9]*;* que són els passos a seguir d'aquest mètode) o també conegut com a mètode Fridrich degut al cognom de la seva inventora principal, Jessica Fridrich, és un mètode basat en el sistema de resolució de capa per capa, de la mateixa manera que el mètode per a principiants.

Per a començar, com la majoria dels mètodes que utilitzen aquest sistema que hem esmentat anteriorment, es fa una creu en una cara d'una manera intuïtiva, igual que el mètode per a principiants. Tot seguit, es completen les peces que falten de les dues cares, és a dir, s'inserten les arestes per paquets formats per un vèrtex i una aresta que tenen continuïtat en els colors que els formen.



Figures 8 i 9. Representació de com s'inserta el bloc vèrtex-aresta.
Font: Fandom (Rubik's Cube Wiki); modificada amb la supressió del fons

6 Intercanviar la posició de certes peces
7 Les sigles en anglès de *First 2 Layers* (2 primeres capes)
8 Les sigles en anglès de *Orientation of the Last Layer* (Orientació de l'última capa)
9 Les sigles en anglès de *Permutation of the Last Layer* (Permutació de l'última capa)

Aquest pas es pot fer de varies maneres. Una primera manera és saber diferenciar 3 situacions genèriques i a partir d'aquí i amb una mica d'agilitat i domini del cub anar col·locant les peces d'aquestes capes. En canvi, la segona manera és més ràpida ja que et permet diferenciar moltes més solucions però té l'inconvenient que és necessari aprendre's 41 algoritmes per a les 41 diferents situacions que et pots trobar. A més a més, com a punt negatiu, veiem que seguint aquest procediment entens bastant menys el funcionament del cub que si ho fas de la primera manera.

A continuació i un cop ja hem completat les dues primeres capes ens centrarem en l'última. Per a resoldre aquesta, primer farem OLL i després PLL.

En primer lloc, haurem d'aprendre a distingir un total de 57 situacions diferents. Es poden reconèixer més o menys fàcilment ja que dibuixen uns patrons que són bastant reconeixibles i per ajudar-nos també els hi donem un nom que il·lustren una mica el patró perquè sigui més fàcil distingir, reconèixer i referir-se a aquests. De mitjana, amb un bon cub i un bon entrenament, pots fer aquesta part de la resolució en uns escassos 3 segons.

Hi ha una possibilitat que és mesclar el mètode per principiants i aquest per a fer OLL i així no haver de tenir la necessitat d'aprendre't els 57 algoritmes que són necessaris. Per fer aquest submètode primer es forma una creu tal i com hem explicat al mètode per a principiants i després s'apliquen els algoritmes de OLL. Com a conseqüència de fer aquesta reducció, normalment es tardarà una mica més de temps per a completar l'objectiu. Aquest submètode és anomenat OLL reduït o *2-Look OLL*.

El següent i últim pas que cal fer és permutar les peces de l'última capa, és a dir, fer PLL. En aquest pas haurem de distingir un total de 21 situacions, i per tant, haurem d'aprendre 21 algoritmes. Aquest pas és una mica més complex de distingir ja que és impossible veure més de dues cares d'un cub alhora i per tant necessites extreure tota o quasi tota la informació de dues cares. Aquí, els algoritmes són més llargs i per tant, una mica més difícils de recordar, tot i que

també hi ha alguns patrons que ens poden ajudar a identificar les diferents situacions. En aquest pas també trobem que hi ha la possibilitat de sacrificar uns segons en la resolució per tal d'aprendre'ns molts menys algoritmes. En aquest cas concret passaríem de necessitar 21 algoritmes a tan sols 5 bastant senzills, aplicant així el submètode de PLL reduït o *2-Look PLL*. Si agafem les mateixes condicions d'abans, és a dir, un bon cub i molta pràctica en uns 4 segons podem passar de tenir tota la capa desordenada a completar el cub per complet.

Aquest mètode és probablement el més utilitzat perquè és molt ràpid, té una àmplia varietat d'algoritmes per triar en cada situació per tal de poder escollir el que més prefereixes o t'és més fàcil, no necessita una gran comprensió del cub i el seu funcionament i és relativament fàcil d'aprendre. De fet el record del món actual (2021) el va fer el xinès Yusheng Du utilitzant aquest mètode i fent l'increïble temps de 3,47 segons.

Com a punts negatius del mètode, trobem que té una gran quantitat d'algoritmes que fan que si n'aprenguéssim un cada dia tardaríem uns dos mesos i mig a aprendre el mètode complet, ja que el total d'algoritmes són 78 sense tenir en compte F2L. Si es busca la velocitat, és un mètode que requereix una quantitat de moviments bastant més gran que altres mètodes, és necessària una bona inspecció prèvia a la resolució i molta experiència si es vol anar ràpid a construir la creu, i és necessari fer rotacions del cub sencer per a desenvolupar aquesta manera de fer el trencaclosques.

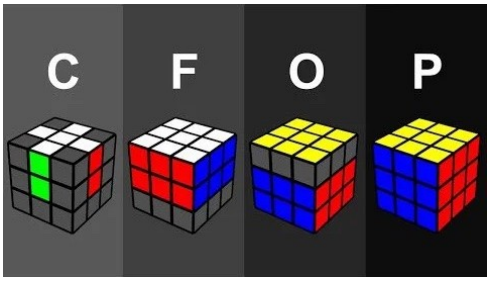


Figura 10. Relació del nom del mètode amb els passos a seguir.
Font: Reddit (Usuari: u/mchljhnvlr)

1.3.4. Algoritme de Kociemba

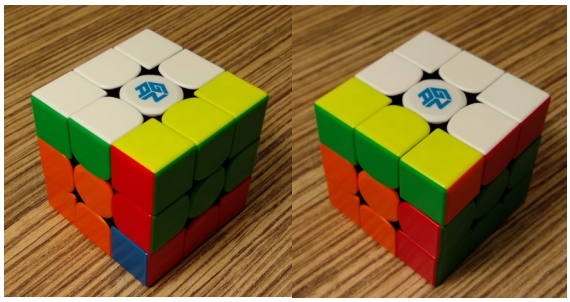
El primer que cal saber d'aquest algoritme és que és un algoritme computacional, és a dir, que està pensat per a la resolució de cubs de Rubik 3x3 mitjançant un ordinador. Aquest algoritme, creat per Herbert Kociemba és similar a un mètode de resolució per ordinador creat anteriorment anomenat "Algoritme de Thistlethwaite", que consisteix en orientar correctament les arestes en primer lloc, tot seguit s'orienten les cantonades, després es fan un seguit de moviments que es coneixen com a "Reducció de Mitja Volta" per preparar el cub pels moviments finals que són exclusivament girs de 180º i finalment s'acaba de resoldre. L'avantatge de l'algoritme de Kociemba respecte a aquest últim esmentat i explicat, sobretot és la disminució de passos necessaris en la solució per aconseguir el cub resolt.

Aquesta disminució ha estat possible gràcies a la millora de l'emmagatzematge de la memòria dels ordinadors i la seva velocitat de processament. L'algoritme de Kociemba està enfocat en dues fases: la primera consisteix en resoldre el cub amb una certa propietat i la segona, en resoldre la resta del cub. Si posem com a exemple F2L (vist anteriorment al mètode CFOP) com a la nostra propietat, el primer pas seria resoldre les dues primeres capes i la segona part consistiria en fer LL[10]. Si s'aconsegueix resoldre F2L amb el mínim de moviments possibles la solució serà relativament curta. Tot i això, podria ser-ho més si fent algun moviment més per F2L després ens n'estalviéssim més per a LL, fins i tot podent arribar a l'omissió de qualsevol algoritme de LL, ja que la capa ja qudaria resolta.

10 De les sigles en anglès de *Last Layer* (Última capa)

L'algoritme de Kociemba funciona d'aquesta manera descrita. La diferència rau en que no utilitza el mètode esmentat en l'exemple ja que és un mètode molt poc eficient si l'objectiu és resoldre el cub amb el mínim nombre de moviment possibles, ja que després de fer F2L només podríem fer moviments en l'última capa de manera lliure sense alterar, almenys parcialment (cosa que suposa un afegit de moviments), la part ja resolta del cub.

Aquest algoritme es basa en la teoria de grups. Un grup és un conjunt al qual se li aplica una operació. Per a que un grup sigui un grup però, és necessari que aquest complexi certes propietats. La primera d'elles és la llei interna (en el paral·lelisme del cub seria que qualsevol moviment que faci segueix estant al cub i no es trenca, per exemple), la segona és l'associativa, que tal com passa en les matemàtiques, algunes operacions no es veuen afectades (3 + (8 – 4) = 7 // 3 + 8 - 4 = 7) mentre que d'altres si que ho fan, com per exemple la resta (3 – (8 – 4) = -1 // 3 – 8 – 4 = -9). La tercera és la propietat neutra (que en el paral·lelisme amb el cub seria l'estat resolt), la quarta és l'oposat (per un moviment U el seu oposat seria U' en l'exemple del cub) i la cinquena i última és la commutativitat, que és l'ordre en que es fan les operacions (no és el mateix fer U R2, que R2 U, tal com veiem a la imatge.



Figures 11 i 12. Representació de la propietat associativa. Al cub de l'esquerra se li ha aplicat la seqüència de moviments U R2 i al de la dreta R2 U
Font pròpia

El primer pas que realitza l'algoritme és introduir el cub en un grup anomenat G1, que té les propietats següents: les cantonades i arestes estan orientades correctament i les arestes de la capa intermèdia es troben ja en aquesta.

Des d'aquest punt es pot resoldre el cub com si fos un altre trencaclosques anomenat Domino, que és un 3x3x2, és a dir, un cub de 3 quadrats de costat per 3 quadrats de costat i 2 d'altura, tal com es veu a la imatge. En aquest punt els moviments que es fan són sempre girs de 180º en totes les cares exceptuant la inferior i la superior.

Figura 13. Cub Domino (3x3x2)
Font original: Ruwix.es

2. COMPONENTS ELECTRÒNICS

2.1. Placa Raspberry Pi

"Raspberry Pi" són una sèrie de petites computadores d'una sola placa (SBC[11]) creades i desenvolupades per l'empresa ·"Raspberry Pi Foundation" amb l'associació de Broadcom.

Es va iniciar com un projecte per portar la informàtica bàsica a les escoles dels països en desenvolupament. Però degut al gran èxit que va tenir van ampliar el mercat i van començar a pensar plaques pensades per a la robòtica. La seva gran quantitat d'usos i la seva possibilitat d'utilitzar USB i HDMI l'han fet una placa utilitzada en una gran varietat de sectors i sent útil per a petits projectes creats per aficionats als ordinadors i a l'electrònica.

Té un software de codi obert anomenat Raspbian, que és una variant del sistema operatiu Linux. Hi ha un total de 5 famílies de plaques i cada placa està formada per uns components específics, fent-la més útil per certes coses.

Aquestes plaques s'anomenen microcontroladors[12] i estan formades per un seguit de microprocessadors[13]. Quan s'ha programat de certa manera la placa, aquesta és capaç d'enviar senyals a certs components electrònics i obtenir informació de l'entorn mitjançant una gran varietat de sensors que poden anar des de detectors de llum fins a sensors de rotació (per posar alguns exemples) i que en connectar-se a la placa permeten que aquesta pugui actuar i tenir control sobre diferents objectes com poden ser llums, motors o actuadors en general. A més a més, al tenir incorporat el software Raspbian pot funcionar com un ordinador senzill.

11 De l'anglès "Single-board computer"
12 És un ordinador de petites dimensions que es troba en un xip.
13 Processador informàtic que processa dades en un sol circuit integrat (conjunt de circuits electrònics)

2.1.1. Microcontrolador

Tal com hem dit anteriorment, un microcontrolador és un circuit integrat compacte que és capaç de realitzar unes accions específiques. Està format per una combinació d'una o varies CPU (unitat central de procés) juntament amb una memòria i un seguit de perifèrics d'entrada i sortida programables, que tenen la funció de rebre o enviar informació.

El processador o CPU podria ser considerat com el cervell del dispositiu, ja que és l'encarregat de processar totes les dades que rep i enviar les diferents directrius per tal de que el microcontrolador pugui fer la seva funció. Per poder fer això necessita fer un seguit d'operacions bàsiques d'aritmètica i lògica.

La memòria serveix per a emmagatzemar les dades que rep el processador i que utilitza per a respondre a les instruccions que se li han programat que portés a terme. Trobem dos tipus de memòria, cadascuna especialitzada en emmagatzemar un tipus d'informació específica. La memòria del programa és l'encarregada de guardar la informació que el programa necessitarà per a l'execució de les instruccions que són necessàries per a la CPU. Aquesta memòria no és volàtil i per tant es manté sense necessitat d'una font d'energia. L'altra memòria que trobem és la memòria de dades, que emmagatzema temporalment les dades que rep. I, al contrari que la memòria del programa, aquesta és una memòria volàtil que si deixa de tenir una font d'alimentació que li proporcioni energia perdrà les dades que tenia guardades.

Els perifèrics d'entrada i sortida (E/S) són els encarregats de comunicar el processador amb el món exterior. Els ports d'entrada reben informació que és enviada a la memòria perquè el processador la pugui utilitzar i aquest envia unes dades que són enviades als ports de sortida que estan connectats als dispositius que realitzaran l'acció pertinent. Els cables o pistes pels quals aquesta informació viatja per la placa s'anomenen busos.

Principalment, un microcontrolador té la funció d'interpretar les dades que rep del seu entorn mitjançant els perifèrics d'entrada i sortida per a controlar una funció específica del dispositiu en que es troba. Quan rep informació, aquesta és emmagatzemada a la memòria, i un cop allà, el processador pot accedir a aquesta informació i fer-la servir per realitzar una certa acció. Però per fer això, primer ha de desxifrar les dades a través de la informació que té guardada a la seva memòria de programa i aplicar les accions convenients en cada cas. Tot seguit, la informació per efectuar aquestes accions és enviada un altre cop als perifèrics de sortida en aquest cas, i d'aquesta manera les dades aconsegueixen sortir del microcontrolador.

2.2. Díodes emissors de llum

Un díode LED (de les sigles en anglès Light Emitting Diode) o DEL (díode electroluminescent) és un dispositiu semiconductor que emet llum incoherent[14] d'espectre reduït quan se'n polaritza[15] de forma directa la unió PN i és travessat per corrent elèctric. Una unió PN està formada per dos materials semi conductors que estan en contacte. Quan circula corrent en el sentit directe (del positiu al negatiu), els electrons es mouen utilitzant els buits disponibles en ambdós semiconductors, mentre que quan ho fan en el sentit contrari, els electrons comencen ocupant els buits fins que queden plens i el corrent s'atura.

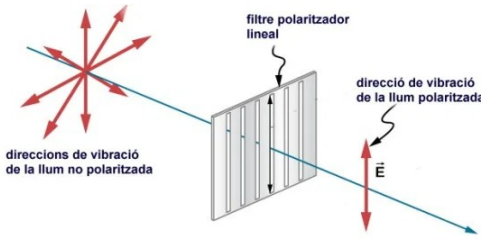


Figura 14. Polarització de la llum en la direcció vertical.
Font: Infominer

14 Radiació electromagnètica que produeix ones de manera independent.
15 El fenòmen de la polarització consisteix en deixar passar una direcció de la llum, fent que les altres direccions quedin absorbides.

Els díodes LEDs més comuns tenen el material semiconductor encapsulat en una coberta de plàstic. Una part de la coberta és plana i serveix per indicar el càtode[16], que és la terminal més curta, mentre que a l'altre costat trobem l'ànode[17], que té la terminal més llarga.

Hi ha diversos tipus de LEDs. Alguns emeten llum infraroja, que per tant no podem veure, d'altres llum visible, i uns altres llum ultraviolada.

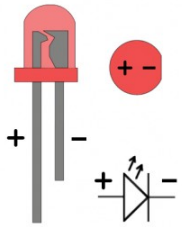


Figura 15. Esquema d'un díode LED.
Font: Thermino

2.3. Resistències

Una resistència elèctrica és un component electrònic que té la funció d'oferir resistència al pas de corrent elèctric per tal de limitar el voltatge d'un circuit.

El seu objectiu consisteix en mantenir un valor de resistència fix malgrat la variació de voltatge i d'altres factors com podria ser la temperatura ambiental. Aquesta és una situació ideal que no es dona en el dia a dia. Tot i això, el resultat final acaba sent una resistència que ofereix poca variació del seu valor i que sempre està dins d'un interval conegut.

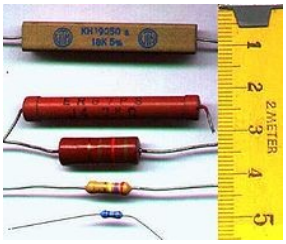


Figura 16. Diferents tipus de resistència.
Font: Viquipèdia

16 Terminal negativa del LED
17 Terminal positiva del LED

Els materials dels quals poden estar fetes les resistències són variats i depenen del valor final que vulguem que tingui la resistència. Malgrat que el material més utilitzat en les resistències és el carboni degut a la seva resistivitat[18] relativament alta, també podem trobar resistències fetes amb filferro recobert amb una pel·lícula metàl·lica, o amb ceràmica, per exemple.

La resistència és una magnitud i les seves unitats són els ohms (Ω). Aquesta magnitud que es defineix com al quocient entre el voltatge i la intensitat. Aquesta relació queda establerta mitjançant la llei d'Ohm, la formula de la qual diu: $R=\frac{V}{I}$ en el qual R és la resistència expressada en ohms (Ω), V el voltatge expressat en volts (V) i I la intensitat expressada en amperes (A).

Una altra manera d'expressar aquesta fórmula molt comuna és $V=I\cdot R$, una fórmula en que es veu clar que la resistència és una constant entre el voltatge i la intensitat tal i com es pot apreciar en la següent gràfica.

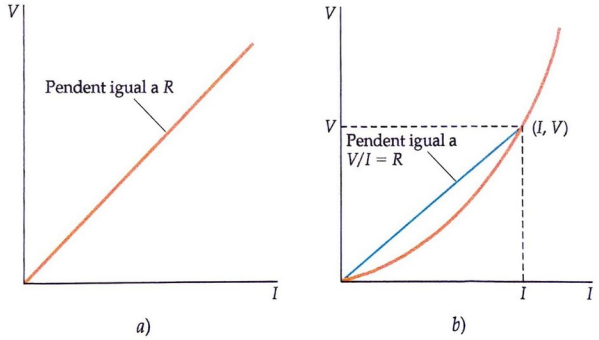


Figura 17. Gràfica de V en funció de I.
En a) el voltatge és proporcional al corrent i el pendent de la recta és igual a R. Aquest cas és ideal ja que en realitat la resistència varia amb la temperatura, un factor que augmenta a mesura que augmenta el corrent elèctric.
En b) veiem que la resistència és la recta que uneix l'origen amb el punt (I,V), ja que el voltatge no és proporcional a la intensitat.

Font: Tipler, P.A., Mosca, G. (2008). *Física per a la ciència i la tecnologia: Volum 2: Electricitat i magnetisme. La llum. Física moderna* (2a ed.). : Nova York: W.H. Freeman and company, New York and Basingstoke
Traducció per Editorial Reverté

18 Resistència específica d'un material conductor. Es diferencia de la resistència perquè aquesta última és proporcional a la longitud del fil conductor i inversament proporcional a la seva secció, és a dir, que varia els seus valors segons aquests dos factors. Les unitats de la resistivitat són $\Omega\cdot m$ i es simbolitza amb ρ.

Per distingir el valor de les diferents resistències s'ha inventat un sistema de colors que està imprès en cada resistència. Les barres de colors estan impreses de tal manera que hi ha un grup de 3 o 4 ratlles agrupades amb una petita separació entre elles i una altra ralla més allunyada. Sempre es comença a llegir per la ratlla que està més a prop de l'extrem de la resistència i que forma part d'aquest grup de 3 o 4 ralles. L'última barra que forma aquest grup és sempre el multiplicador i la ratlla que està més apartada indica la tolerància. Per identificar el valor d'una resistència cal consultar la taula següent per extreure'n el valor correcte.

Figura 18. Exemple de resistència. Font: Viquipèdia

Color	1r dígit	2n dígit	3r dígit	Multiplicador	Tolerància
Negre	0	0	0	10^0	
Marró	1	1	1	10^1	1%
Vermell	2	2	2	10^2	2%
Taronja	3	3	3	10^3	
Groc	4	4	4	10^4	
Verd	5	5	5	10^5	
Blau	6	6	6	10^6	
Violat	7	7	7	10^7	
Gris	8	8	8	10^8	
Blanc	9	9	9	10^9	
Daurat				10^{-1}	5%
Platejat				10^{-2}	10%
Sense color					20%

Per tenir un exemple ens fixarem en la resistència de la imatge anterior. El primer que identifiquem és el grup de 3 tires de color. Com que comencem a llegir els colors per la part més externa de la resistència el primer color que trobem és el groc, per tant el primer dígit de la resistència és 4. El següent color és el violat per tant el número que segueix és el 7. L'última ratlla d'aquest grup és sempre el multiplicador per tant sabem que hem de multiplicar el valor per 10^2, és a dir, per 100. La barra que està sola ens indica la tolerància que en aquest cas és del 5%. Per tant, aquesta resistència és de 4700 Ω que amb la tolerància del 5% podem obtenir uns valors d'entre 4465 Ω i 4935 Ω.

3. IMPRESSIÓ EN 3D

Els ésser humans sempre hem volgut tenir a l'abast eines i màquines que ens ajudin a fer la nostra vida més fàcil i a optimitzar el temps al màxim. La invenció de l'impremta va suposar un gran estalvi de temps en el procés de copiar llibres. Més endavant, amb l'arribada dels ordinadors i les impressores, un ja podia fer el mateix procés que les impremtes des de casa d'una manera molt senzilla, ràpida i econòmica, permetent així una llibertat considerable a l'hora de plasmar un text amb una lletra universal sobre un paper. Podem pensar que la impressió en 3d podrà suposar un canvi similar al de les impressores que tothom té a casa, ja que permetrà a l'usuari idear un objecte concret i a través del disseny i una impressora 3d, poder ser capaç de crear aquest objecte. Això permetrà que cada vegada més gent pugui fabricar objectes que poden ser útils pel dia a dia des de casa, essent així molt més específics i personalitzats del que sovint poden acostumar a ser els objectes que podem comprar en una botiga o per internet.

Però, tal com passa també amb les impressores normals, que n'hi ha de diferents tipus (per exemple amb tinta, làser,...), també trobem diferents models d'impressora 3d. Alguns d'aquests estan pensats per un ús més personal, i per tant són més econòmiques i petites, d'altres per a un ús més comercial, algunes altres es fan servir en les indústries per a fabricar peces específiques fins i tot s'està investigant per imprimir menjar i teixits humans.

3.1. Impressió per addició

Aquest tipus d'impressores 3d són les més utilitzades, sobretot en l'àmbit domèstic. El seu funcionament consisteix en afegir capes de material successivament per tal d'anar formant un objecte.

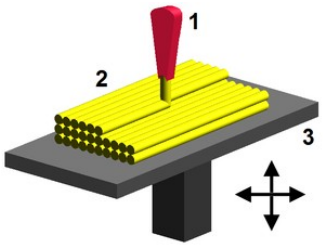


Figura 19. Il·lustració del funcionament d'adició de material d'una impressora 3d.
Font: Xataca

En aquest tipus d'impressió es poden utilitzar diferents materials, cada un amb unes propietat específiques. N'utilitzarem un o un altre depenent de l'ús que li vulguem donar i del tipus d'impressora que tinguem a la nostra disposició, ja que no totes són capaces d'imprimir amb els mateixos materials.

L'àcid polilàctic o PLA és segurament el material més utilitzat en la impressió domèstica. És un termoplàstic rígid que per ser utilitzat es fon mitjançant un extrusor, una màquina relativament petita que és l'encarregada de fondre el filament de PLA en aquest cas per tal de que es pugui dipositar en capes. Per fer aquest pas la impressora consta de 3 eixos mòbils. Hi ha dos models diferents de la disposició dels eixos. Un d'aquests té l'extrusor (1) que es mou de costat sobre un eix que també té la possibilitat de moure's en l'eix vertical i el llit (2) es mou endavant i endarrere. En canvi, l'altre model varia lleugerament, ja que l'extrusor es mou en l'eix x i y i és el llit el que es mou en l'eix z, anant baixant a mesura que es van dipositant capes de filament.

Figura 20. Primer model d'impressora esmentat

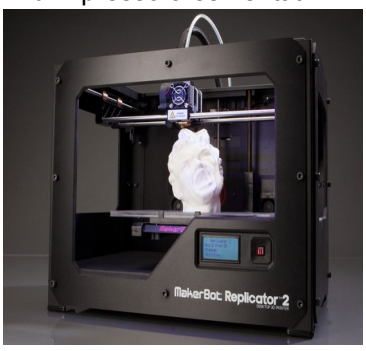


Figura 21. Segon model d'impressora esmentat. Font: Enter.co

Un altre material també molt utilitzat és la resina ABS o acrilonitril butadiè estirè és un termoplàstic amorf que és molt utilitzat en l'automoció i en altres usos domèstics. Per exemple, les famoses peces de la marca Lego estan fetes d'aquest material.

També es poden utilitzar altres materials com poden ser metalls, menjar o fins i tot teixits humans, ja que aquesta és l'única manera amb què és possible imprimir amb materials orgànics.

Aquest tipus d'impressió és molt versàtil ja que pot imprimir una gran varietat de formes amb la simple acció d'anar disposant capes de material unes sobre les altres. Tot i això, un dels problemes importants que té és que en alguns casos (sobretot quan en algunes zones es supera un angle de 45º respecte la perpendicular al pla d'impressió) és necessari imprimir unes estructures anomenades suports. Un suport simplement és una part impresa amb el mateix material de la peça que serveix per tal de que, quan per culpa de les limitacions de l'impressora no és possible imprimir bé una zona, el material tingui una estructura provisional on recolzar-se i així evitar malformacions de la figura impresa. L'estructura del suport té una densitat de farcit molt baixa per tal de gastar el mínim de material possible. Aquests suports s'han de retirar de manera manual un cop s'ha acabat d'imprimir i en alguns casos si es vol un millor acabat perquè aquests suports han deixat petites marques s'hi pot aplicar un tractament específic per cada material per tal d'aconseguir un tacte més fi o l'eliminació de la percepció de les capes, per posar alguns exemples.

3.2. Impressió per compactació

Més conegut pel seu nom en anglès (selective laser sintering o SLS), aquest tipus d'impressió consisteix a compactar el material amb el que s'imprimirà la peça. Aquest material es trobarà polvoritzat a una temperatura pròxima a la de fusió i mitjançant un làser s'escalfarà la capa de material amb la forma d'una de les moltes capes que tindrà l'objecte per tal d'aconseguir que quedi dura respecte a la resta de material. Un cop s'ha fet aquest procés, s'aplica una altra capa de material a sobre i es repeteixen els mateixos passos. Un cop s'ha acabat, es trenca el material que no s'ha endurit i ens queda a les nostres mans l'objecte desitjat. A més a més, com un aspecte important a destacar, tot el material que no s'ha utilitzat estrictament per a fer la peça es pot reutilitzar per a següents usos.

MARC PRÀCTIC

4. CONSTRUCCIÓ DEL CUB DE RUBIK

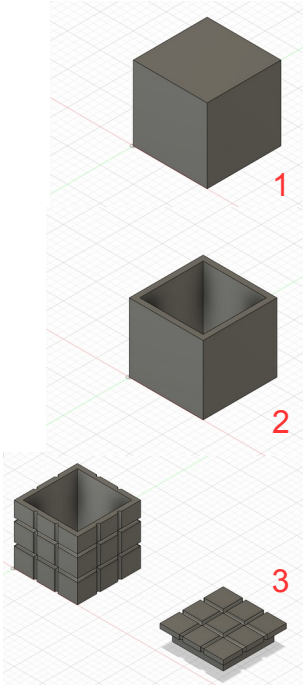
En aquest projecte no s'ha construït un cub funcional en que es poden fer girs, sinó una espècie de capsa amb la forma d'un cub de Rubik en la que en el seu interior es troba el sistema de llums LED's i la placa Raspberry Pi que són els encarregats de mostrar la solució mitjançant un patró de llums.

4.1. Disseny del cub

Per poder fer un bon disseny del cub el primer que es va haver de triar va ser un bon programa de modelatge 3d. En aquest cas el programa escollit va ser Autodesk Fusion 360. Aquest programa pot ser una mica menys intuïtiu que algun altre però quan es tracta de dissenyar certs objectes funciona millor i amb una mica de pràctica el seu potencial pot ser considerable.

Abans de començar cal explicar que la major part del procés es va fer amb les dimensions d'un cub de Rubik estàndard, per tal de poder dimensionar bé el projecte i evitar deformacions. Més tard ja s'escalaria a la mida desitjada.

El primer que calia fer era dibuixar un cub que em serviria com a base per començar a treballar (1). A partir d'aquí va tocar buidar el cub (2) per anar aconseguint ja aquesta idea de capsa que estava buscant. Un cop vaig tenir això, el següent que vaig fer va ser dona-l'hi un aspecte de cub de Rubik i crear la tapa (3). Per tal d'evitar que quedés un cub deformat en l'eix z, va ser necessari retallar una mica la part superior per així aconseguir que quan s'hi afegís la tapa quedés un cub perfectament dimensionat.



Figures 22-24. Procés del disseny del cub.
Font pròpia

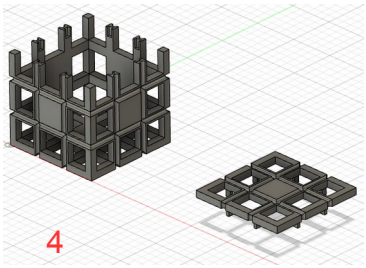
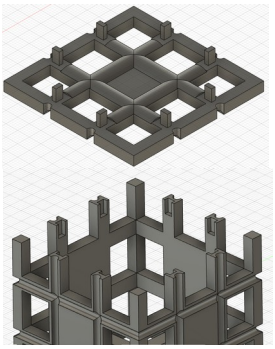


Figura 25. Procés del disseny del cub.
Font pròpia

Com que volem un cub que tingui unes obertures per on la llum dels díodes LED es pugui veure, va ser necessari dissenyar-les (4). Per fer-ho, es pot realitzar de dues maneres. El primer disseny consisteix en un cub amb parets sòlides exceptuant les cantonades, que seran les obertures on hi estaran disposats els LEDs. Tanmateix, l'altra opció, que ha sigut l'escollida, consta d'unes obertures en totes les peces que formen la perifèria de la cara del cub, és a dir, les 4 arestes i 4 vèrtexs. Aquest disseny té certs avantatges respecte al primer esmentat. Un d'ells és un estalvi de material important, ja que no s'ha d'imprimir en les arestes. A més a més, el fet de que les obertures estiguin disposades d'aquesta manera dona una homogeneïtat al cub que no podem observar tan en la primera proposta. Un petit inconvenient però, és que en aquests forats que deixem s'hi ha d'imprimir uns suports, i tal com hem explicat anteriorment, és un material que un cop la peça s'ha acabat d'imprimir es retira i es llença a les escombraries. Aquest va ser un factor important a valorar perquè per un costat en la primera opció s'utilitza bastant més material que en la segona, mentre que en la segona s'imprimeix menys quantitat de PLA però hi ha una quantitat considerable d'aquest que es llença. Finalment em vaig decantar per imprimir els suports i assumir aquesta petita pèrdua de material.

Després de realitzar això, calia buscar una manera de fer que la tapa i el cos del cub poguessin estar junts d'una manera més o menys fixa evitant que amb poc moviment caigués, però que per altra banda també fos relativament fàcil de posar i treure per si cal revisar qualsevol cosa del seu interior en un futur o simplement per introduir fàcilment els LEDs en el procés de muntar-l'ho.

La solució més simple i senzilla que vaig trobar va ser la de fer unes petites entrades a les parts que separen els forats que marquen les peces i que estan en contacte amb la zona superior. Paral·lelament, a la tapa va ser necessari que hi hagués una part sortint que encaixés en aquestes ranures.



Figures 26 i 27. Encaix de la tapa amb el cub
Font pròpia

Tot seguit vaig decidir fer una petita impressió a una escala més reduïda, de tal manera que el cub tingués la mida d'un cub de Rubik estàndard, és a dir, 56 mm de costat. Aquesta part del procés va anar bé per corregir petits errors del disseny i millorar alguns aspectes. Un d'ells, per exemple, va ser que els sortints que havien de fer encaixar la tapa amb el cub formaven una quadrícula sota la tapa que suposava una pèrdua d'espai i una malbaratament de material. Així doncs, vaig poder rectificar el disseny abans de tornar a fer una impressió a l'escala bona.

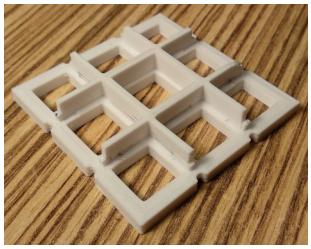


Figura 28. Impressió de prova amb la quadrícula.
Font pròpia

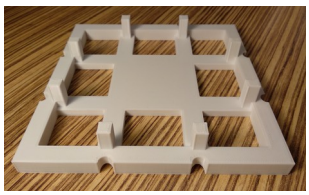
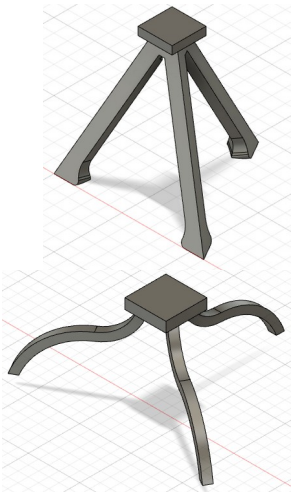


Figura 29. Impressió final amb la quadrícula ja eliminada.
Font pròpia

Un cop arribat a aquest punt vaig decidir escalar els nostres models 3d per començar a treballar amb les mesures finals. La primera intenció va ser de fer un cub que tingués 12 cm de costat, però després de prendre algunes mides vaig veure de seguida que aquestes mesures serien insuficients i faltaria espai a l'interior del cub, principalment, per encabir-hi la placa Raspberry Pi. És per aquest motiu que vaig passar dels 12 cm de costat als 15 cm i d'aquesta manera poder aconseguir un espai més gran en l'interior. A més a més, per evitar algun altre problema d'espai que hi pogués haver, vaig reduir el gruix de la tapa en la seva part que dona a l'interior del cub, deixant així la perifèria de la mateixa mida per mantenir unes proporcions homogènies en l'exterior del cub.

Tot seguit vaig pensar que seria una bona idea tenir el cub una mica elevat del terra i vaig començar a dissenyar un peu que aixecaria el cub de la superfície en que es trobés. Aquesta part no va ser fàcil i abans no vaig aconseguir el disseny definitiu hi van haver dos antecessors. El primer d'ells tenia un disseny que no el feia massa atractiu i que donava un aspecte rígid. Com que no m'acabava de fer el pes vaig decidir buscar un altre disseny diferent. Va ser per això que vaig començar a modelar la segona versió. Aquesta versió era més estètica i tenia unes potes corbes. Els problemes en aquest disseny van arribar juntament amb la impressió en 3d. Es va imprimir amb uns acabats no massa bons degut a la seva forma i vaig poder comprovar que les seves potes eren massa primes ja que en el moment d'intentar retirar els suports generats per l'impressora 3d una de les potes va trencar-se. Això va ser el causant de l'últim canvi en el disseny d'aquesta part. Aquesta darrera modificació va ser la que va aportar una transformació més gran. En comptes de tres potes com havia fet fins aquell moment vaig optar per un suport que tingués quatre potes. A més a més, com que sabia que els cables havien de sortir de dins del cub vaig reservar un petit espai per on poguessin passar. Cal destacar que per les formes que té aquesta peça hagués sigut bastant difícil de fer amb algun altre programa de disseny 3d que no fos el Fusion 360.



Figures 30 i 31. Primers dissenys del peu
Font pròpia

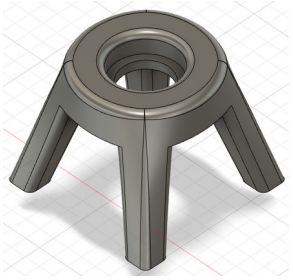


Figura 32. Disseny final del peu.
Font pròpia

També vaig pensar que seria una bona idea poder girar el cub horitzontalment sobre un eix que estigués situat just en aquest peu. Per aquest motiu vaig dibuixar una peça que servís d'unió entre el cub i el suport. Aquesta peça consta de dues parts, una de quadrada que encaixa amb una petita depressió feta a la part inferior del cub i una altra de cilíndrica que és la que permet que el cub pugui girar de manera independent al suport. L'orifici central, tal com he comentat anteriorment serà el que permetrà el pas de cables.

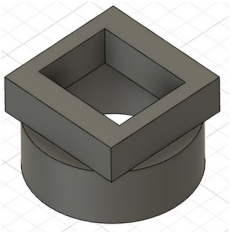


Figura 33. Peça que permet el moviment del cub.
Font pròpia

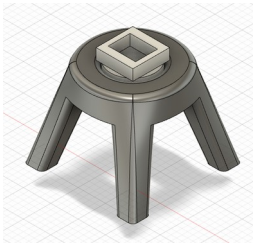


Figura 34. Unió entre el peu i la peça rotatòria.
Font pròpia

Per acabar cal fer esment a la part per on hauran de passar els cables. Les obertures es trobaran una sota de l'altre i així els cables podran connectar els components de dins del cub amb l'exterior. Aquests forats, tal i com hem anat dient, es troben al cub, al peu i a la peça que uneix el cub amb el peu.

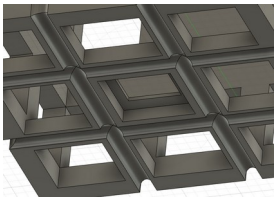


Figura 35. Depressió feta a la part inferior del cub.
Font pròpia

4.2. Impressió 3d del cub

Un cop tots els dissenys ja estaven ben definits i acotats s'havien de començar a imprimir. Primerament calia exportar els models 3d en .stl , un format que permet emmagatzemar la descripció d'un objecte 3d. Les sigles STL provenen de l'anglès "Standard Triangle Language"[19] degut al seu funcionament, ja que divideixen la figura 3d en petits triangles per definir la seva geometria i mida, però mai factors com poden ser el color o la textura. Aquest arxiu STL s'ha d'obrir amb un programa específic que realitzarà un procés anomenat

19 Llenguatge estàndard de triangles

laminació, que consisteix en dividir l'estructura del model desitjat a imprimir en petites làmines. El gruix d'aquestes làmines pot variar lleugerament, donant lloc a impressions més o menys definides i amb més o menys qualitat. Normalment es considera una mida estàndard 0.2 mm. Una conseqüència de les diferents altures de capa és el consum de material i el temps d'impressió. Com menor és aquesta, més material es gasta i més temps tarda a finalitzar la impressió. Per fer una petita comparació, podem agafar de model el cos del cub d'aquest projecte, és a dir, el cub sense la tapa ni el peu. Entre una capa de 0.2 mm de gruix amb una de 0.28 mm hi ha aproximadament 12 hores i 32 grams de material de diferència. És per aquest motiu que en aquest treball de recerca s'ha imprès tots els dissenys amb una altura de capa de 0.28 mm, ja que això ha suposat un estalvi de material, temps i, en conseqüència, d'energia considerable. A més a més, al ser un objecte de dimensions grans respecte a l'altura de capa, s'ha pogut imprimir a aquesta qualitat sense que el disseny perdés resolució. Per definir tot això i altres factors com per exemple la densitat de farcit de les figures i els suports o el patró amb el qual s'imprimiran els models i suports (tal com es pot veure en les imatges), he utilitzat un programa anomenat "Cura".

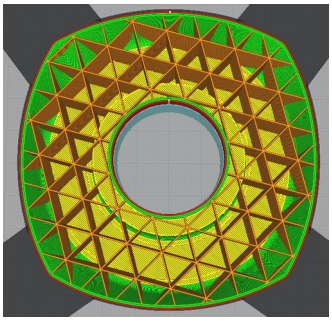


Figura 36. Farcit amb patró triangular.
Font pròpia

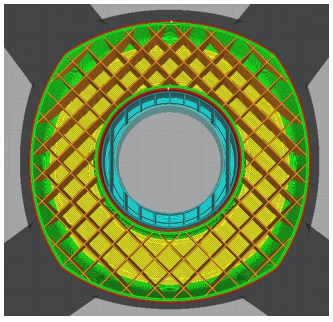


Figura 37. Farcit amb patró de reixeta per la figura i amb lineal pel suport (blau).
Font pròpia

En aquest programa, a més a més de les característiques esmentades anteriorment, també s'hi pot decidir la temperatura de l'extrusor, del llit de la impressora, la velocitat d'impressió de la primera capa i de la resta del model, el gruix de les parets, el tipus d'adherència al llit, etc.

Una vegada tots aquests paràmetres ja estan definits el programa exporta aquest fitxer STL en un altre format que serà el que podrà llegir la impressora, el format GCODE.

Des d'aquest punt ja es pot començar el procés d'impressió en si mateix. En el meu cas imprimiré amb una impressora d'addició de PLA. Un cop la màquina s'ha engegat és important calibrar el llit d'impressió. Aquest és un procés que per anar bé s'ha de fer cada vegada que es vol imprimir quelcom, tot i que si es fa sempre no s'ha de variar gran cosa. Per fer-ho, la impressora té unes posicions preestablertes en diferents llocs del llit en les que l'extrusor es posa quasi a tocar el llit perquè puguis calibrar la impressora. El mètode més comú per calibrar-la consisteix en passar un full de paper entre el llit i la punta de l'extrusor i notar que rasca però que pot desplaçar-se. Per aconseguir anar anivellant el llit simplement s'han de fer girar unes petites rodes que trobem a la part inferior del llit. Aquest és un mètode que al principi costa bastant trobar la mesura adient però que amb una mica d'experiència t'acaba ajudant a fer una calibració bastant bona abans de començar a imprimir. Tot i això, no sempre és perfecte i és recomanable estar pendent de com imprimeix la primera capa i anar ajustant el llit segons com siguin les traces de filament que es van imprimint.

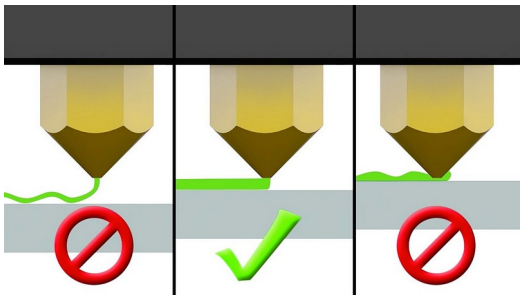


Figura 38. Il·lustració de la mida ideal entre llit i extrusor segons com surt el filament.
Font: Kuttercraft

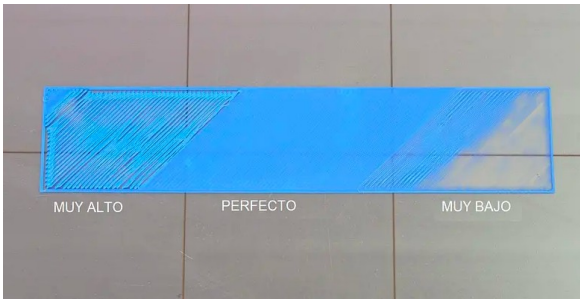


Figura 39. Comparació entre una extrusió alta, a la mida perfecte i baixa.
Font: Minkafab

Una altra estratègia que es pot seguir per assegurar una bona primera capa sobre la qual poder construir una bona figura amb les mínimes irregularitats és netejar el llit amb un drap moll i després eixugar-ho per tal de netejar qualsevol resta de material que hagi pogut quedar d'impressions prèvies que puguin malmetre la nostra peça. Normalment una figura s'acaba espatllant degut a que una part del filament no s'ha adherit bé al llit i al no fer-ho pot ser arrossegat amb la resta de filament que l'extrusor va disposant o pot resultar amb una part de la figura que estigui elevada cap amunt, amb una possibilitat de generar problemes que arruïnin la peça al cap d'alguna hora, cosa que no sol ser gaire agradable i que en la majoria de casos implica tornar a començar la impressió des del principi. També s'hi pot aplicar una laca per millorar l'adherència tot i que en aquest cas no s'ha utilitzat.

La primera peça que vaig imprimir va ser el cos general del cub. Un dels motius pels quals va ser la primera va ser per assegurar que tindria el suficient filament per acabar la peça i no es quedaria a mig fer arruïnant així la impressió i havent llençat una gran quantitat de material. Al ser una peça de dimensions tan grans, el temps i materials utilitzats eren bastant elevats. Estem parlant d'un temps d'impressió d'unes 25 hores i mitja aproximadament i 315 grams de material, tot i que només 237 grams formen la figura en si i els 78 grams restants van ser destinats a la generació de suports.

Aquesta peça es va haver de començar a imprimir tres vegades. La primera d'elles tenia un petit problema en el disseny que per sort va poder ser detectat en les primeres capes i ràpidament vaig poder parar la impressió per retocar el disseny abans de tornar-hi. A més a més d'aquest error que va sortir es va fer una revisió a tota la resta de la figura per evitar trobar-se amb cap problema més endavant que perjudiqués la peça. I si la primera peça havia durat poca estona abans no sortís el primer error, la segona encara va tardar menys. Quan tot just estava imprimint la primera capa una mala adherència del material al llit de la impressora va ser el causant de la detenció de la impressió.

El material que no es va disposar correctament va ser arrossegat pel nou material que la màquina anava fonent i va suposar una pèrdua de la forma requerida.

Finalment, la tercera i última impressió va ser la definitiva i després de 25 hores i escaig la peça estava impresa a la perfecció. Només calia treure tots els suports generats. Per fer això vaig utilitzar unes alicates per arrencar els suports i un gúbia petita per arrancar les parts de més petites i de més difícil accés on les alicates no em servien.

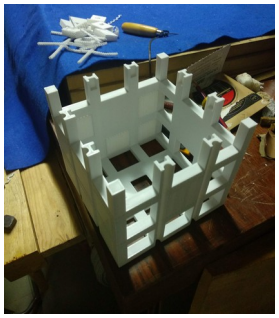
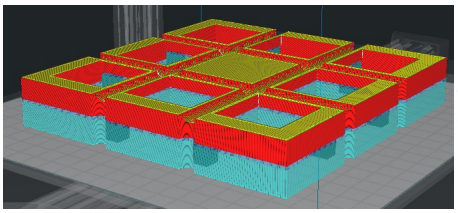


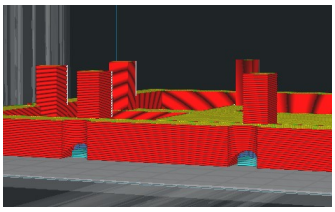
Figura 40. Extracció dels suports del cub. Font pròpia.

La segona peça que es va imprimir va ser la tapa del cub. Seguint la mateixa estratègia comentada anteriorment, l'ordre d'impressió també va estar determinat una mica per la quantitat de filament restant. En aquest cas va semblar que en quedava una quantitat suficient com per imprimir la peça sencera i així va ser. Després de 4 hores i 47 minuts d'impressió la peça estava a punt per treure de la impressora. Per imprimir aquesta tapa es van utilitzar 70 grams de material, dels quals molt pocs es van utilitzar per generar un suport. El fet de que s'utilitzés tan poc material per a la impressió d'aquesta part es pot justificar de dues maneres. La primera és sense cap mena de dubte la disposició d'impressió, és a dir, com estava col·locada la peça en el moment de la impressió. Per evitar gastar material innecessàriament era important imprimir la tapa de tal manera que la part exterior quedés a sota, tocant el llit. D'aquesta manera s'aconsegueix estalviar tots els suports que s'haurien generat si la peça hagués estat girada 180º en l'eix x o y, ja que tal com havia comentat prèviament aquesta consta d'uns petits sortints que serveixen per unir la tapa i això suposaria un aixecament de la major part de la peça que, per tant, hauria de portar molts més suports. De fet, la diferència és tant important que estem parlant de 44 grams de material de suport de diferència. La segona justificació, que està estretament lligada amb la primera, és que els únics llocs on ha sigut

necessari generar suports ha sigut en les petites ranures que té el cub i que serveixen per definir la forma per tal d'identificar ràpidament que es tracta d'un cub de Rubik.



Figures 41 i 42. Diferència de suports (blau) segons la col·locació de la peça. Font pròpia.



Finalment, les últimes parts a imprimir-se van ser les que formen el peu per sostenir el cub a una certa altura. Aquesta part possiblement va ser la més conflictiva. Tal com ja he dit, el disseny del peu es va veure alterat diverses vegades. El segon disseny, que semblava que seria l'escollit, va resultar ser molt dèbil i això va implicar aquest últim canvi de disseny. La impressió en aquest cas va ser conjunta entre el peu i la peça que permetria el moviment giratori del cub. Aquesta peça no havia estat inclosa en els models previs degut a que va ser una idea d'última hora que es va poder portar a terme gràcies a aquesta necessitat de crear un nou peu pensat d'una manera totalment diferent. Per fer aquesta impressió es van gastar 77 grams de material i es va estar imprimint 4 hores i 50 minuts. Tot i que la impressió va ser bona els problemes no van tardar a aparèixer. La peça petita que seria l'encarregada de permetre la rotació del cub sobre del peu era una mica massa petita i feia que el cub es bellugués massa quan se'l sotmetia a una rotació. Per aquest motiu es va haver de retocar les mides de la peça en qüestió perquè encaixés bé. No obstant, en aquest segon intent tampoc vaig aconseguir la mida correcte i la peça va quedar excessivament gran. Per tant, vaig haver de repetir el procés seguit anteriorment un altre cop. Aquesta vegada vaig aconseguir una mida intermedia que permetia un bon encaix i rotació sense que el cub es mogués en excés. Per sort, aquestes impressions en que simplement s'imprimia aquesta peça sola només tardaven 1 hora i gastaven 12 grams de material, que en comparació amb totes les altres estructures impreses anteriorment són uns temps i material utilitzat molt petits.

Per acabar, penso que és interessant fer un recompte de tot el material utilitzat per imprimir aquest projecte. El cub amb la tapa i el peu (és a dir, el projecte complet) pesa 376 grams. Però això no és el total de material gastat en la impressió, ja que en aquest pes no s'hi compten els suports impresos, que fan un total de 88 grams. Però encara podem treure unes altres mesures, ja que hi ha hagut certes impressions que s'han hagut de descartar per alguns motius ja esmentats anteriorment que sumen un total de 41 grams més. I si a més a més hi afegim el prototip que va servir per detectar errors i fer una primera prova d'impressió hi hem de sumar uns altres 37 grams.

Per tant, sumant el conjunt de material imprès veiem que s'ha imprès un total de 542 grams dels quals 166 corresponen als residus i al prototip.

També podem fer un petit recompte del temps d'impressió. Si en fixem en el temps net de impressió podem veure que el cos del cub va tardar 25 hores i mitja a imprimir-se, la tapa 4 hores i 47 minuts i el peu i la peça que permet la rotació van tardar unes altres 4 hores i 49 minuts. Però com que no només s'ha imprès això també podem sumar els altres temps. El primer peu que es va imprimir i que va resultar ser massa fràgil va tardar 2 hores i 48 minuts a imprimir-se i les dues peces que serveixen per aconseguir girar el cub sobre el suport van tardar uns 57 minuts més cadascuna. A més a més, els problemes en les primeres impressions del cub van fer perdre 45 minuts i el prototip va tardar 4 hores i 32 minuts més a imprimir-se. Per tant la impressora va estar imprimint un total de 35 hores i 6 minuts d'impressió de peces bones i 9 hores i 59 minuts peces que no van arribar a ser útils, la suma del qual fa 45 hores i 5 minuts d'impressió sense tenir en compte els temps de preparació, d'extracció de suports i altres possibles retards.

4.3. Muntatge dels components electrònics i retocs finals del cub

Aquesta part que semblava que hauria de ser senzilla va ser de les que més esforç s'hi va haver d'aportar. A més a més, es van haver de fer un seguit de coses que requerien concentració i traça per evitar un desastre que arruïnés tot el procés portat. Per sort vaig aconseguir anar passant per les diferents etapes i finalment el cub va quedar del tot muntat i a punt per funcionar.

Primerament vaig pensar que seria una bona idea enganxar una mena de pantalles translúcides que deixessin passar una mica la llum dels LEDs però que evitessin veure l'interior, de tal manera que tots el cables que hi trobaríem quedarien amagats proporcionant sobretot una millora important en la part estètica del cub. Al principi, quan tenia el prototip en petit imprès, vaig pensar que podria ser una bona idea enganxar una mica de paper vegetal a les parts interiors del cub, però finalment vaig veure que era massa prim i es veia el que hi havia al seu interior, per tant va ser necessari buscar una alternativa. Buscant què podria fer servir vaig trobar unes portades d'unes guies telefòniques que em podien servir ja que eren més opaques que el paper vegetal. Tot i això, pel meu gust encara es veia massa l'interior. I aquí va ser on, amb l'ajuda de la família, vam pensar que seria una bona idea enganxar un tros de paper juntament amb les portades i vam provar com funcionaven. Va resultar que donaven l'efecte que estàvem buscant i per tant va ser l'opció escollida.

Es van prendre mides de les parets interiors del cub i em vaig posar a tallar les làmines, tant a les portades com en fulls de paper. Va ser un procés que va ser més llarg del que esperava, ja que la impressió en 3d va generar petites irregularitats no observables a simple vista però que va suposar que cada peça havia de ser lleugerament diferent tot adaptant-se a l'espai requerit.

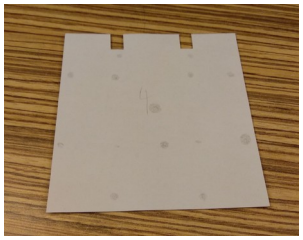


Figura 43. Exemple d'una làmina utilitzada. Font pròpia.

Un cop vaig tenir totes les làmines tallades vaig enganxar primer el full de paper amb la portada de plàstic i més tard ho vaig enganxar tot a les parets del cub amb cianoacrilat[20] per aconseguir una bona adhesió i evitar que en algun moment futur es poguessin desenganxar.

Tot seguit vaig procedir a crear els adhesius que definirien el color de cada cara per a permetre la correcta orientació del cub barrejat amb el del treball i així assegurar un correcte procés de resolució. El primer que vaig haver de fer va ser mesurar els centres del cub. Vaig observar que les seves dimensions eren de 42x42 mm i com que volia una petita distància entre l'adhesiu i el límit del centre vaig decidir que farien 38x38 mm, aconseguint així 2 mm de marge a cada costat. Després de fer aquests petits amidaments era l'hora de comprar els adhesius. Tot buscant vaig trobar uns fulls adhesius de colors que vaig veure que em podrien servir i els vaig comprar. Un cop a casa vaig marcar un quadrat de 38x38 mm a cada full adhesiu i els vaig retallar amb una cisalla per aconseguir un tall el màxim de net possible. Finalment, l'últim pas era la col·locació d'aquests al cub. Per fer-ho es van marcar unes guies amb llapis que marcaven les diagonals del centre per intentar tenir els adhesius el màxim de centrats possible. Quan va arribar el moment d'enganxar-los només va fer falta estar atent amb el patró de colors perquè coincidís amb el patró que podem trobar en qualsevol cub de Rubik.

Un altre detall que vaig aplicar al cub va ser la col·locació d'unes gotes de silicona antilliscants adhesives a les potes del peu per evitar que quan es girés una mica el cub no rellisqués. D'aquesta manera s'aconsegueix que el cub quedi quiet durant la resolució.

Arribats a aquest punt, la part externa del cub ja estava acabada i podia començar a preparar la part interna en que hi anirien tots els LEDs i la Raspberry Pi. El primer que vaig fer va ser dissenyar com seria el circuit elèctric. A cada cantonada trobaríem una "breadbord" o placa de proves, una placa plena de petits forats que mitjançant algunes connexions entre ells

20 Més conegut com a "Super Glue"

formen un seguit de circuits fixos en que s'hi poden connectar diferents elements. L'avantatge d'aquesta placa de proves és que fixa diferents elements d'una manera bastant bona tot introduint pins de cables o les "potes" de diferents elements com LEDs o resistències a pressió sense la necessitat de soldar res. Tenint en compte la meva inexperiència en l'àmbit de les soldadures vaig pensar que aquesta seria una bona manera que em permetria connectar LEDs i resistències sense haver de soldar. Normalment aquestes plaques de proves acostumen a fer més de 15 cm de llarg i més de 5 cm d'ample, però per internet en vaig trobar unes de petites de 4,5 cm de llarg i 3,5 cm d'ample, per la qual cosa eren ideals pel meu treball. Va ser aquí on vaig muntar el circuit i vaig provar l'orientació dels LEDs per tal que enfoquessin la cara que s'havia d'il·luminar. Alguns d'aquests va ser necessari tallar-los perquè les potes quedaven massa llargues i no enfocaven del tot bé el lloc que havien d'il·luminar. Aquí vaig haver de vigilar de diferenciar el pol negatiu i el positiu deixant el pol positiu amb la pota més llarga per així evitar algun problema a l'hora de la col·locació. També vaig tallar les potes de les resistències perquè quedessin el màxim d'avall que fos possible i així molestessin el mínim als LEDs.

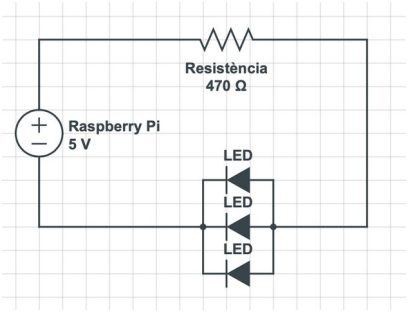


Figura 44. Esquema elèctric del circuit que trobem a cada cantonada.
Creació pròpia feta a Circuit Lab

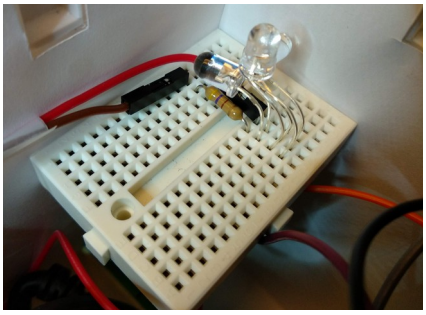


Figura 45. Circuit elèctric muntat a la breadboard d'una cantonada.
Font pròpia.

Una vegada va estar tot muntat va arribar el moment d'enganxar cada breadboard dins del cub. Per fer-ho i assegurar que no caurien també es va utilitzar cianoacrilat com a les pantalles instal·lades a les parets. Amb cura es va anar pintat els costats de cada placa de proves i es va anar enganxant al lloc corresponent amb cura, vigilant bastant per evitar fer malbé les pantalles encarregades de difuminar la llum. Aconseguir enganxar les 8 breadboards al cub va ser un procés delicat però que va sortir amb èxit.

El següent que calia fer era connectar tots els cables a la placa Raspberry Pi, cosa que significava que ens acostàvem a la finalització de la part física del treball. El primer que vaig fer va ser introduir la placa dins del cub evitant que cap cable quedés enrotllat a algun lloc i després fos difícil la seva alliberació. Paral·lelament vaig fer passar els cables d'alimentació i de les sortides USB pel forat inferior del cub, facilitant així el procés del muntatge. Tot seguit vaig començar a ajuntar els cables amb les sortides i entrades de la placa base. Mentre ho feia anava apuntant en quina sortida estava connectat cada cable per després poder-ho introduir al programa informàtic i així tenir classificat sobre quina cantonada actuava cada sortida de la placa.

Arribats a aquest punt pensava que ja estava quasi tot muntat i que havia sigut un èxit, però la part més feixuga encara havia d'arribar. L'últim que calia fer era connectar el cable HDMI a la placa i fer-lo passar pel forat de la cara inferior del cub per poder-lo endollar en un monitor i així poder fer funcionar el programa. Quan vaig fer entrar el cable pel forat vaig veure que l'espai era molt reduït i que seria difícil de connectar-lo a la placa. Degut a això havia de posar un cable que fos flexible per doblar-lo el màxim possible a l'interior del cub. Finalment en vaig trobar un i quan el vaig haver endollat semblava que la placa s'havia quedat encallada per culpa de la pressió que exercia el cable sobre la paret del cub, tot immobilitzant la placa. Al cap d'una estona i molt esforç vaig ser capaç d'aconseguir posar-ho tot bé, aconseguint així la finalització total de la part física.

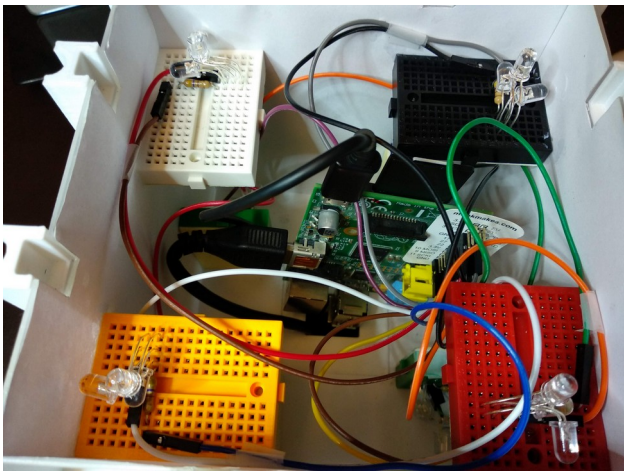


Figura 46. Aspecte de l'interior del cub ja acabat. Font pròpia.

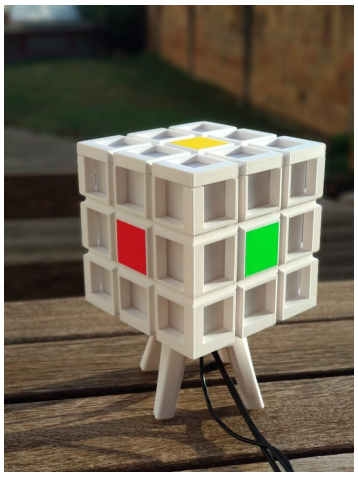


Figura 47. Aspecte de l'exterior del cub ja acabat. Font pròpia.

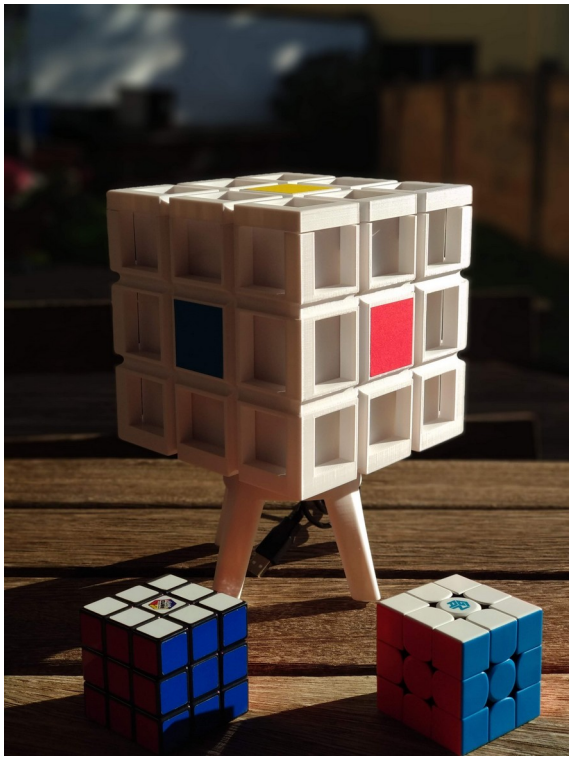


Figura 48. Comparació de la mida del cub acabat amb dos cubs de Rubik normals. Font pròpia.

5. PROGRAMACIÓ DEL CODI

Aquesta és una part del treball que és fonamental per a l'hora d'aconseguir un bon resultat final. Podem considerar la part de la programació com el cervell de tot el projecte, ja que de fet s'encarregarà de controlar tot el que passa. Per a crear el programa s'ha utilitzat el llenguatge de programació *Python*.

5.1. Funcionament del procés de resolució

Per tal que aquest projecte funcioni correctament s'hauran de seguir uns passos que seran necessaris per a preparar el programa i garantir una bona resolució. El funcionament del cub serà el següent: primerament s'introduirà la situació del cub de Rubik que l'usuari tingui barrejat, tot seguit s'haurà d'obrir la terminal de l'ordinador i executar el programa creat, introduint la solució donada anteriorment. A partir d'aquí s'haurà de mirar el cub imprès en 3d i seguir els patrons que generaran els llums LEDs amb cub barrejat. Per poder controlar una mica els llums es podran prémer unes tecles d'un teclat per tal de poder repetir la seqüència mostrada o passar al següent moviment a realitzar.

Comencem per explicar la part de la generació de la solució d'un cub de Rubik. En aquesta part crec important agraïr a l'usuari de GitHub "hkociemba" per penjar a internet un programa realitzat amb Python en el que pots introduir la solució de manera manual o mitjançant una càmera, ja sigui la mateixa de l'ordinador o bé un mòdul per utilitzar amb una placa Raspberry Pi, i que et genera la solució del cub seguint l'algoritme de Kociemba, creat per ell mateix. Trobar aquest programa va suposar un gran pas en la definició del treball, ja que d'altra manera hagués sigut molt difícil d'aconseguir arribar a l'objectiu final del treball. A més a més, per una persona que no havia programat mai a la vida com era el meu cas hagués sigut quasi impossible de fer, i més tenint en compte el temps limitat del projecte.

Per poder fer funcionar el programa vaig recórrer a un amic que em va explicar algunes coses que desconeixia. Va ser amb ell que vam aconseguir fer-lo funcionar.

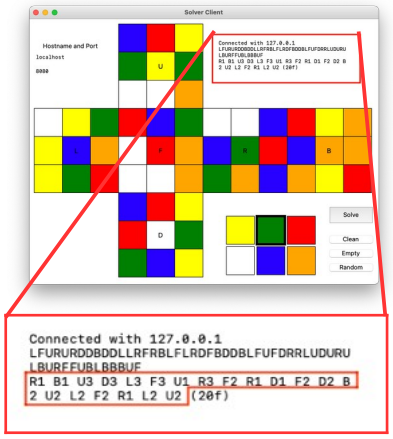


Figura 49. Posició on trobem la solució del cub. Font pròpia.

El primer que vam haver de fer va ser iniciar un petit servidor des de l'ordinador d'una manera molt senzilla, ja que només havíem d'executar un fitxer des de la terminal. Aquest va ser un primer pas important perquè a partir d'aquí ja es podia introduir la situació del cub barrejat al programa de manera manual i et retornava la solució. La solució la podem trobar en la part superior dreta i és el tros en que hi ha una lletra en majúscula acompanyada d'un nombre, tal com es veu a la figura 49.

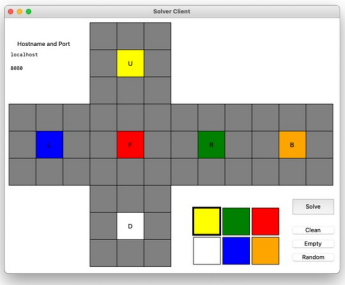


Figura 50. Posicionament dels colors del programa. Font pròpia.

Per poder resoldre el cub seguint els passos donats és important posicionar el cub de la mateixa manera en la que el programa posiciona el desplegable del cub, és a dir, el vermell a davant i el groc a la cara superior. Això és una cosa que s'ha hagut de tenir en compte a l'hora de programar, però sobretot a l'hora de posicionar els LEDs, ja que és important que s'encengui el llum correcte o resultaria inútil.

A partir d'aquest pas hem d'executar el programa creat expressament per aquest projecte. Per fer-ho haurem de tenir la placa Raspberry Pi connectada a un monitor i a la corrent. Després d'encendre la placa serà necessari obrir la terminal (que la podem trobar a la barra superior de la pantalla o bé pressionant les tecles ctr + alt + T alhora) i realitzar un seguit d'ordres per tal d'obrir el programa. El primer que haurem d'escriure serà "cd Desktop", una ordre que el que fa és posicionar-se a la carpeta Escriptori. Després haurem d'executar el

fitxer i per fer-ho haurem d'escriure el següent: "sudo python3 TdR.py". Al fer això li estem dient que obri el fitxer TdR.py amb el programa Python i amb drets d'administrador. La partícula python3 serveix per obrir el fitxer amb la versió 3 de Python i la sudo significa que es pot obrir el fitxer amb drets d'administrador, cosa que permet realitzar certes accions que d'altra manera no es podrien fer.

Tot seguit el programa ens demanarà escriure la solució que haurem aconseguit prèviament. Després d'escriure-la i prémer la tecla "enter" s'il·luminaran un seguit de cantonades en ordre, tot indicant el moviment que s'haurà d'imitar al cub barrejat. Abans de fer el moviment ensenyat a través dels llums serà necessari orientar el cub imitant la posició dels colors del cub del treball. Si es considera necessari tornar a visualitzar el moviment que s'haurà de seguir serà tant senzill com prémer la lletra "r" del teclat que estarà connectat a la placa Raspberry Pi i es tornarà a mostrar la seqüència de llums. Si el moviment ja s'ha fet correctament només serà necessari prémer la tecla "espai" per passar al següent moviment. D'aquesta manera s'hauria d'anar seguint fins a veure que el cub fa uns parpellejos finals en que s'il·lumina tot el cub. Arribats a aquest punt el cub hauria d'estar resolt, i si no ho està significarà que l'usuari s'ha equivocat en algun pas.

5.2. Creació del programa

En aquest projecte la part del programa creat per mi mateix té un pes important ja que és un programa únic que està pensat expressament per a fer la funció que li correspon. En aquesta part del projecte segurament és en la que s'hi ha destinat més hores de treball, ja que a l'hora de crear un programa es necessita un progrés que acostuma a ser lent i ple de problemes que s'han d'anar solucionant mica en mica. A més a més, tampoc ha estat una part fàcil de fer i en podem destacar dos motius. El primer d'ells és que en el moment de començar el treball jo no havia programat mai, per tant va ser difícil en aquest sentit perquè vaig haver d'anar assimilant mica en mica els diferents conceptes

que podria necessitar per a programar. El segon d'ells va ser la dificultat de trobar una manera en que el programa pogués funcionar d'una manera correcta i controlada tot facilitat així l'experiència de l'usuari a l'hora de fer servir el programa.

El primer pas que va ser necessari fer, fins i tot abans de començar a aprendre a programar, va ser escollir el programa amb que es faria el projecte. Des del principi i durant molt de temps semblava que el programa escollit seria Arduino. Arduino són unes plaques similars a les Raspberry Pi que tenen el seu propi llenguatge de programació, que era en el que tenia pensat treballar ja que pensava utilitzar una placa Arduino per a realitzar el projecte. Tot i això, després d'un temps buscant possibles altres solucions que ajudessin a encaminar millor el treball vaig descobrir les plaques Raspberry Pi amb les quals es podia programar amb el llenguatge de programació Python. Va ser en aquest moment que vaig decidir fer un canvi i centrar la meva feina en aquesta nova direcció. Per tant, al tenir la primera part (i potser la més important) decidida vaig buscar cursos de programació amb Python. La gran majoria d'ells eren cursos que estaven pensats com a una extraescolar i per aquest motiu acostumaven a començar al setembre o octubre. Finalment vaig trobar un curs pel YouTube que em va semblar que era prou adient per a mi i vaig decidir fer-lo. Aquest curs em va ajudar a aprendre els conceptes bàsics de la programació i d'aquesta manera vaig evitar estar molt perdut, sobretot al principi.

Quan vaig començar a escriure el codi del programa sabia una mica com ho volia fer però no sabia ben bé com fer-ho. Per tirar endavant, sobretot al començament, vaig necessitar una mica d'ajuda d'un professional. La seva ajuda va ser summament útil i només puc fer que agrair-li enormement el fet de que m'ajudés. Primerament vaig haver d'importar unes llibreries que farien funcionar el programa i que servirien per a poder interactuar amb un teclat, permetre gestionar les accions mitjançant temps o controlar les entrades i sortides de la Raspberry Pi. Cal dir que aquesta última funció es va implementar més tard ja que al principi vaig fer el programa amb un ordinador

normal per comoditat a l'hora de treballar i va ser més endavant quan vaig introduir tots aquests canvis per a fer funcionar la placa al mateix temps que traslladava el programa del meu ordinador a la Raspberry Pi, ja que d'altra manera no hagués pogut fer les proves necessàries a l'ordinador perquè el programa no pot funcionar correctament sense la placa si aquest conté aquesta llibreria.

El següent que vaig programar va ser que l'usuari introduís la solució al programa i que aquesta es guardés com a una variable. A partir d'aquesta variable es va desenvolupar la resta del programa. Per a poder separar cada pas de la solució vam haver de "tallar" la solució en els diferents moviments individuals que s'haurien d'anar seguint. El programa és capaç de distingir això perquè entre un pas i el següent hi ha un espai de separació, el qual és utilitzat per indicar on cal tallar la solució per obtenir els diferents moviments. Després el programa agafa aquests trossets un a un i fa quelcom segons li hagi programat. Com que aquests passos individuals sempre segueixen un patró va ser relativament fàcil separar les diferents informacions que contenen. Un d'aquests fragments sempre està format d'una lletra que indica la cara que s'haurà de girar i un nombre que indica els girs de 90º que s'hauran de fer en sentit horari. Si agafem com a exemple U2 voldrà dir que s'haurà de girar la cara superior 180º. Gràcies a aquest patró vaig poder crear una variable que agafava sempre la lletra i una altra que es quedava amb el nombre. Des d'aquí vaig poder dir al programa que si la lletra era tal i el nombre tal altre, havia de fer un seguit d'ordres. Aquestes ordres consisteixen en encendre i apagar els llums de manera ordenada per aconseguir un efecte de moviment quan estàs mirant el cub. Això es va haver de repetir per cada possible moviment que es pugui fer al cub.

Per a poder interactuar una mica amb el cub vaig haver de fer ús de la biblioteca "keyboard" ja que permet utilitzar el teclat com a font d'entrada d'informació. La primera idea que tenia pensada era que quan premessis la tecla espai et mostrés la seqüència de llums del moviment que tocava fer en

bucle de tal manera que quan haguessis fet el moviment al cub podries tornar a prémer l'espai i et mostraria el següent moviment a fer en bucle fins que tornessis a realitzar el mateix procés. Després de moltes proves vaig veure que la manera en que el funcionament s'acostava més al descrit no era gens pràctica i encara menys vàlida. Per tant s'havia de buscar una nova solució. Aquesta solució consisteix en fer que el patró de llums que s'encén només ho faci un sol cop i que si l'usuari creu necessari tornar a veure-ho pugui prémer la lletra "r" i es torni a repetir. Això va ser un pas important perquè va permetre desencallar un problema que feia temps que durava. Per fer això vaig haver de declarar una variable[21] amb un valor que en aquest cas va ser 1. Si el valor d'aquesta variable era el 1 es podia executar l'ordre d'encendre i apagar els LEDs. Al finalitzar la seqüència de llums es canvia el valor d'aquesta de tal manera que s'evita que es torni a repetir el procés perquè el valor necessari per a iniciar-se ja no és el mateix. I aquí és on entra la part interessant en que l'usuari pot decidir què fer. Si prem la lletra "r" aquest valor de la variable tornarà a ser 1 i per tant es podran repetir un altre cop les ordres, mentre que si prem la tecla "espai" farà un *break*[22] i es repetirà el procés amb la següent part de la solució fins a acabar.

Al finalitzar tots els moviments el cub farà pampalluguejarà 3 cops per indica el fi de la resolució. Aquesta part va ser una incorporació d'última hora però que vaig pensar que seria una manera per indicar que s'havia acabat la resolució millor que no pas una simple absència de llum. A més a més a la terminal també apareixerà un missatge que indicarà el final de la resolució.

El procés de fer el programa no va sortir a la primera i hi van haver alguns problemes que es van haver d'anar solucionant mica en mica. El primer problema que em vaig trobar va ser el de poder ser capaç de dir al programa que diferenciés les diferents parts de la solució. Aquí em va ser molt útil l'ajuda que em va proporcionar aquest programador i em va permetre avançar. Més

21 Dit del fet de crear una nova variable.
22 Ordre que permet fer un tall en el codi i que en aquest cas és útil per indicar quan s'ha acabat de fer un moviment en el cub.

endavant, quan encara estava fent proves sense tenir en compte els LEDs vaig haver de trobar una manera d'aturar el bucle que havia creat quan premia la tecla espai. El problema, que de fet em va acompanyar fins al final, va ser el causant de la solució adoptada al final. Per intentar solucionar això i aconseguir que fos com a mi m'hagués agradat m'hi vaig passar moltes hores i finalment no va poder ser ben bé igual, tot i que la solució trobada s'hi acosta.

Altres problemes van sorgir quan vaig provar de fer funcionar el programa a la placa Raspberry Pi. El primer d'ells va ser un error que va portar bastants maldecaps comparats amb el problema en si, que era insignificant. Quan feia les meves proves per comprovar que el programa funcionava em sortia un error estrany que està associat a un caràcter estrany, normalment degut a copiar i enganxar certes coses. Però aquest no era el cas i al final va resultar ser un accent en la frase en que es demana a l'usuari d'introduir la solució. És per aquest motiu que es va haver de treure l'accent i acceptar que hi hauria una petita falta d'ortografia degut a les limitacions del programa. Una possible explicació de que passi això és que la placa està configurada en anglès i al ser una llengua que no té accents detecti aquests com un caràcter estrany. Mentre realitzava alguna prova també em vaig trobar algun problema espontani que es va solucionar amb la simple acció de tancar i tornar a obrir el programa.

Durant el procés també va ser important anar decidint algunes coses sobre el funcionament del cub. Una d'elles va ser decidir la quantitat de LEDs que es trobarien a cada cara. Vaig haver de descartar la possibilitat de posar llums a les cantonades i arestes per la falta de sortides de la Raspberry Pi i per això es van quedar simplement les cantonades. Una altra qüestió que va sorgir va ser el patró que seguirien els llums a l'hora de mostrar les solucions. Aquí hi havien dues opcions sobre la taula. La primera que es va pensar i que ha sigut la que ha perdurat consisteix en mostrar el sentit del gir encenent una a una les diferents cantonades d'una cara, tot creant un moviment circular. En els casos en que s'ha de fer un gir de 180º s'il·lumina cada cantonada dos cops tot fent un pampallugueig. En canvi, la segona opció consistia en il·luminar solament

una peça d'inici i una de fi, de tal manera que sabries que hauries de portar la peça il·luminada en primer lloc a la il·luminada posteriorment. Hi van haver dos motius que estan una mica relacionats que van fer decantar la balança a favor de la primera opció. El primer que vaig veure va ser que en els moviments de 90º es podria confondre quina cara s'havia de girar, ja que el fet de que dues cares comparteixin una aresta pot portar confusió sobre quina és la cara que s'ha de girar. Això s'hagués pogut solucionar si només s'encenguessin els LEDs de la peça d'una cara i no la peça sencera (és a dir, amb 3 cares il·luminades). I aquí va ser on va aparèixer el segon motiu pel qual es va descartar la segona opció. Això comentat recentment es podia solucionar si s'aconseguís una entrada a la placa per cada LED, i aquí va ser on va mancar la proposta, ja que s'haguessin necessitat un total de 24 entrades i només disposàvem de 10.

D'aquesta manera i després de superar els problemes que van anar sortint durant el procés vaig acabar el programa que finalment va ocupar 732 línies de codi.

6. CONCLUSIONS

La valoració final d'aquest treball és molt positiva. El cub en la seva versió final compleix plenament la funció pel qual va ser creat. D'una manera intuïtiva permet a l'usuari seguir el moviment que indiquen els llums del cub que s'il·luminen seguint certs patrons. Penso que la principal utilitat d'aquest projecte és tenir un recolzament en la resolució d'un cub de Rubik per a aquelles persones que encara estan aprenent com funciona i com resoldre'l. Amb aquest cub s'aconsegueix generar la suficient confiança a algú inexpert en els cubs de Rubik per a intentar resoldre'n un amb la tranquil·litat de saber que en qualsevol moment el podrà tornar a solucionar. Crec que això és molt útil ja que algunes vegada no t'atreveixes a provar de resoldre algun trencaclosques així per la por de no poder-lo tornar a tenir resolt en molt de temps. Personalment he experimentat aquesta sensació i estic segur que si hagués tingut un aparell que fos capaç d'ajudar-me a arribar a la solució m'hagués atrevit més a provar de resoldre un puzle d'aquestes característiques.

Fent una recapitulació dels objectius proposats inicialment veiem que s'han complert tots. El primer objectiu proposat va ser fer funcionar un programa creat per un desenvolupador extern que et donava una solució per resoldre el cub de Rubik un cop havies introduït la situació en que es trobava. Malgrat fer-lo funcionar correctament, una possible millora seria intentar aconseguir que el programa aconseguís un reconeixement dels colors de cada cara mitjançant una càmera.

El segon objectiu és un dels objectius que més èxit ha tingut en relació a les expectatives. La creació del programa que s'encarrega d'enviar les senyals necessàries als llums LED per poder mostrar a l'usuari la solució ha estat un èxit. Malgrat les dificultats trobades degut a la meva inexperiència en el món de la programació el programa té un funcionament força fàcil que permet a

tothom utilitzar-lo. Si volguéssim buscar un aspecte de millora del programa es podria intentar d'aconseguir que els llums mostressin el moviment indicat d'una manera repetitiva sense la necessitat de prémer una tecla com ho fa actualment, i que al prémer l'espai del teclat et mostrés el següent moviment que caldria fer. Tot i això, seria necessari aprendre a programar bastant més ja que no és senzill de programar un comportament així.

El tercer objectiu consistia en la creació d'una capsa en forma de cub de Rubik que tingués un espai a l'interior per instal·lar un sistema de LEDs i la placa Raspberry Pi que permetessin ensenyar la solució a l'usuari. Segurament aquesta és la part de la qual estic més orgullós del resultat final obtingut. L'aspecte aconseguit és molt bo i a la part interior ha sigut possible posar-hi tots els elements necessaris pel bon funcionament del projecte. Al ser la primera vegada que construïa un projecte així hi ha alguns petits detalls que podrien millorar-se de cara a una segona versió. El primer d'ells seria fer el cub una mica més gran, ja que al moment de posar el cable HDMI al seu interior va quedar bastant ajustat. El segon detall simplement fa referència a l'aspecte exterior, i és que entre les pantalles que vaig enganxar per difuminar la llum i el cub hi ha una petita escletxa que fa veure que els acabats es podrien millorar en aquest aspecte. Per solucionar això s'hauria de fer una petita modificació en el disseny de tal manera que cada cara estigués una mica enfonsada deixant lloc per posar les pantalles opaques.

El següent objectiu era comprendre com enviar senyals als LEDs per tal de controlar quin LED s'encenia i quin deixava de fer-ho. Aquest va ser un objectiu que va poder-se resoldre d'una manera ben senzilla malgrat que al principi ho veia com a un objectiu difícil d'assolir malgrat que era totalment necessari. Per poder-lo assolir simplement va fer falta una mica d'investigació que em va permetre trobar una solució senzilla, utilitzant els diferents pins d'entrada i sortida de la placa Raspberry Pi.

Finalment, l'últim objectiu proposat va ser aconseguir un aparell que et mostrés la solució d'un cub de Rubik d'una manera intuïtiva mitjançant llums. Aquest objectiu final s'ha pogut assolir i a més a més s'ha fet amb èxit, aconseguint que amb molt pocs coneixements del cub de Rubik puguis ser capaç de resoldre'l tot seguint les indicacions donades pels LEDs del cub.

Fent un anàlisi dels resultats podem veure que la part més difícil i menys intuïtiva és la preparació, ja que és necessari tenir instal·lats alguns programes a l'ordinador per poder començar. Tot i això, amb una explicació de com fer-ho no hauria de ser massa difícil de preparar-ho tot. Malgrat això, l'objectiu final del projecte s'ha pogut assolir i qualsevol pot ser capaç resoldre aquest famós trencaclosques. De cara a possibles millores es podria intentar perfeccionar els programes i àdhuc provar de fusionar-los, tot donant a l'usuari una millor experiència, més simple i amb menys requeriments.

Aquest treball va començar encarat d'una manera diferent al resultat final i es va anar perfilant mica en mica a mesura que la investigació avançava i anaven sorgint nous problemes que degut al temps limitat del projecte i a la meva inexperiència en alguns àmbits van haver d'anar-se descartant i buscant solucions alternatives. Malgrat això, el fonament de la primera idea ha seguit dempeus i ha perdurat fins al final.

Finalment, podem dir que aquest treball, a més a més de ser exitós ha sigut útil per mi ja que m'ha ajudat a aprendre a organitzar-me la feina jo mateix i a lluitar per perseguir una idea, donant el màxim de tu per intentar que el resultat sigui el millor possible. És per aquest motiu que estic molt content dels resultats obtinguts en aquest treball.

7. REFERÈNCIES BIBLIOGRÀFIQUES

Tipler, P. A., i Mosca, G. M. (2020). *Física per a la ciència i la tecnologia. Volum 2: Electricitat i magnetisme. La llum. Física moderna* (2a ed.). Reverté.

Wikipedia. (2021) *Ernő Rubik.* Recuperat 9 gener 2021, des de https://en.wikipedia.org/wiki/Ernő_Rubik

Smithsonian Magazine. (2021) *A Brief History of the Rubik's Cube*. Recuperat 9 gener 2021, des de www.smithsonianmag.com/innovation/brief-history-rubiks-cube-180975911

Wikipedia. (2021) *Optimal solutions for Rubik's Cube*. Recuperat 14 març 2021, des de https://en.wikipedia.org/wiki/Optimal_solutions_for_Rubik%27s_Cube

TV3, (2016). *La genialitat del cub de Rubik* [Vídeo]. CCMA. https://www.ccma.cat/tv3/alacarta/quequicom/la-genialitat-del-cub-de-rubik/video/5630304/

World Cube Association. (2021). *Records | World Cube Association*. Recuperat 14 març 2021, des de https://www.worldcubeassociation.org/results/records

Ruwix. (2018). *New Rubik's Cube record: Yusheng Du - 3.47 seconds*. Recuperat 14 març 2021, des de https://ruwix.com/blog/yusheng-du-record-347/

Tomé, R. [Asociación Proyecto Cometas Divulgativas]. (2018, November 21). *Teoría de grupos y el cubo de Rubik* [Video]. Consultat 14 març 2021, des de https://www.youtube.com/watch?v=8xdzuDyXicE&t=1579s

Wikipedia. (2021). *Jessica Fridrich*. Recuperat 17 març 2021, des de https://en.wikipedia.org/wiki/Jessica_Fridrich

Wikipedia. (2021). *CFOP method*. Recuperat 17 març 2021, des de https://en.wikipedia.org/wiki/CFOP_method

Speedsolving.com Wiki. (2021). *CFOP method*. Recuperat 17 març 2021, des de https://www.speedsolving.com/wiki/index.php/CFOP_method

Speedsolving.com Wiki. (2021). *Kociemba's Algorithm*. Recuperat 17 març 2021, des de https://www.speedsolving.com/wiki/index.php/Kociemba%27s_Algorithm

Speedsolving.com Wiki. (2021). *Human Thistlethwaite Algorithm*. Recuperat 17 març 2021, des de https://www.speedsolving.com/wiki/index.php/Human_Thistlethwaite_Algorithm

Speedsolving.com Wiki. (2021). *Thistlethwaite's Algorithm*. Recuperat 18 març 2021, des de https://www.speedsolving.com/wiki/index.php/Thistlethwaite%27s_algorithm

Speedsolving.com Wiki. (2021). *Corner Orientation*. Recuperat 18 març 2021, des de https://www.speedsolving.com/wiki/index.php/Corner_Orientation

Wikipedia. (2021). *Microprocessor*. Recuperat 6 abril 2021, des de https://en.wikipedia.org/wiki/Microprocessor

Wikipedia. (2021). *Microcontroller*. Recuperat 6 abril 2021, des de https://en.wikipedia.org/wiki/Microcontroller

Lutkevich, B. (2019). *Microcontroller (MCU)*. Recuperat 6 abril 2021, des de https://internetofthingsagenda.techtarget.com/definition/microcontroller

Viquipèdia. (2021). *Díode emissor de llum*. Recuperat 17 juny 2021, des de https://ca.wikipedia.org/wiki/D%C3%ADode_emissor_de_llum

Departamento de Tecnología Electrónica de la Universidad de Vigo. (2016). *El diodo*. Recuperat 19 setembre 2021, des de http://dte_recursos.webs.uvigo.es/recursos/multimedia/potencia/ac-dc/archivos/diodo.htm

Viquipèdia. (2021). *Polarització electromagnètica*. Recuperat 17 de juny 2021, des de https://ca.wikipedia.org/wiki/Polaritzaci%C3%B3_electromagn%C3%A8tica

Viquipèdia. (2021). *Impressora 3D*. Recuperat 17 juny 2021, des de https://ca.wikipedia.org/wiki/Impressora_3D#Les_impressores_3D_de_l%C3%A0ser

Impresoras3dblog. (2013). *¿Cuál es el funcionamiento de una impresora 3D?* Recuperat 17 juny 2021, des de https://impresoras3dblog.wordpress.com/tag/impresoras-3d-adicion/

Fabara, S. (2014). *Así es el futuro de las impresoras 3D*. Recuperat 17 juny 2021, des de https://www.enter.co/especiales/hogar-digital/hogar-del-futuro-con-impresoras-3d/

Scientificprotocols. (2016, August 15). *Unión PN de semiconductores*. [Vídeo]. YouTube. https://www.youtube.com/watch?v=uL6OgPpx1dc

Wikipedia. (2021). *Raspberry Pi*. Recuperat 17 juny 2021, des de https://en.wikipedia.org/wiki/Raspberry_Pi

Stack Overflow. (2021). Consultes vàries. Recuperat del 14 agost 2021 al 8 setembre 2021, des de https://stackoverflow.com

Viquipèdia. (2020). *STL*. Recuperat 30 agost 2021, des de https://ca.wikipedia.org/wiki/STL

Viquipèdia. (2021). *Resistència elèctrica (component)*. Recuperat 1 setembre 2021, des de https://ca.wikipedia.org/wiki/Resist%C3%A8ncia_el%C3%A8ctrica_(component)

Autor desconegut. *La Resistencia*. (2019). Recuperat 1 setembre 2021, des de http://www.videorockola.com/proyectos-electronicos/curso-de-electronica-basica/electronica-basica-la-resistencia/

Inglés, E. (2015). *Que no se te resistan las resistencias*. Recuperat 1 setembre 2021, des de https://quevieneipv6.com/que-no-se-te-resistan-las-resistencias/

La Geekipedia De Ernesto. (2019, July 15). *Curso Python desde cero - Curso de programación profesional*. [Vídeo]. YouTube. https://www.youtube.com/watch?v=mENHDQ8SLsI&list=PLyvsggKtwbLW1j0d5yaCkRF9Axpdlhsxz

Programiz. (Any d'última actualització desconegut). *Python sleep() Function (With Examples)*. Recuperat 4 setembre 2021, des de https://www.programiz.com/python-programming/time/sleep

Briceño, G. (2020). *Python: Como se implementa una sentencia switch-case*. Recuperat 7 setembre 2021, des de https://www.clubdetecnologia.net/blog/2017/python-como-se-implementa-una-sentencia-switch-case/

Circuit Lab. (2021). *CircuitLab*. Recuperat 12 setembre 2021, des de https://www.circuitlab.com/editor/#?id=7pq5wm&from=homepage

Kociemba, H. (2013). *Solve Rubik's Cube with Cube Explorer*. Recuperat 26 maig 2021 http://kociemba.org/cube.htm

Kociemba, H. (2019). *GitHub - hkociemba/RubiksCube-TwophaseSolver: Solve Rubik's Cube in less than 20 moves on average with Python*. Recuperat 2 juny 2021, des de https://github.com/hkociemba/RubiksCube-TwophaseSolver

Delavar, M., Kolesnikova, I. V., Rahimzade, B., Ghahhari, M., Mosapuor, A., i Moradi, A. (2018). *Development of Mental Rotation Ability at Primary School Level*. Published. Recuperat 28 octubre 2021, des de http://yadda.icm.edu.pl/yadda/element/bwmeta1.element.psjd-883e231f-1771-4ed8-a5af-778b236bf450/c/WSN_101__2018__77-88.pdf

ANNEXOS

Codi del programa creat:

```
import keyboard
import time
import RPi.GPIO as GPIO
import warnings
warnings.filterwarnings("ignore", category=RuntimeWarning)

#Associar reds de les cantonades amb el pin corresponent. Sigles dels colors de la cantonada en angles
GPIO.setmode(GPIO.BCM)
ygr = 17
ybr = 22
ybo = 7
ygo = 4
wgr = 18
wgo = 24
wbo = 25
wbr = 8

#Preparar els pins com a sortida
GPIO.setup (ygr, GPIO.OUT)
GPIO.setup (ybr, GPIO.OUT)
GPIO.setup (ybo, GPIO.OUT)
GPIO.setup (ygo, GPIO.OUT)
GPIO.setup (wgr, GPIO.OUT)
GPIO.setup (wgo, GPIO.OUT)
GPIO.setup (wbo, GPIO.OUT)
GPIO.setup (wbr, GPIO.OUT)

#Inici del programa
st = raw_input("Si us plau, introdueix la solucio: ")
arr = st.split(" ")
print(arr)
for x in arr:

     i=1

     while True:
          letter= x[0]
          it= x[1]

          #Moviment U
          if letter == "U":
               if it == "1":
                    while i==1:
                         GPIO.output(ygr, True)
                         time.sleep(1)
                         GPIO.output(ygr, False)
                         time.sleep(1)
                         GPIO.output(ybr, True)
                         time.sleep(1)
                         GPIO.output(ybr, False)
                         time.sleep(1)
                         GPIO.output(ybo, True)
                         time.sleep(1)
                         GPIO.output(ybo, False)
                         time.sleep(1)
                         GPIO.output(ygo, True)
                         time.sleep(1)
                         GPIO.output(ygo, False)

                         i += 1

                    if keyboard.is_pressed("r"):
                         i=1
```

```
        if keyboard.is_pressed("spacebar"):
            break

if it == "2":
    while i==1:
        GPIO.output(ygr, True)
        time.sleep(0.3)
        GPIO.output(ygr, False)
        time.sleep(0.3)
        GPIO.output(ygr, True)
        time.sleep(0.3)
        GPIO.output(ygr, False)
        time.sleep(1)
        GPIO.output(ybr, True)
        time.sleep(0.3)
        GPIO.output(ybr, False)
        time.sleep(0.3)
        GPIO.output(ybr, True)
        time.sleep(0.3)
        GPIO.output(ybr, False)
        time.sleep(1)
        GPIO.output(ybo, True)
        time.sleep(0.3)
        GPIO.output(ybo, False)
        time.sleep(0.3)
        GPIO.output(ybo, True)
        time.sleep(0.3)
        GPIO.output(ybo, False)
        time.sleep(1)
        GPIO.output(ygo, True)
        time.sleep(0.3)
        GPIO.output(ygo, False)
        time.sleep(0.3)
        GPIO.output(ygo, True)
        time.sleep(0.3)
        GPIO.output(ygo, False)

        i += 1

    if keyboard.is_pressed("r"):
        i=1
    if keyboard.is_pressed("spacebar"):
        break

if it == "3":
    while i==1:
        GPIO.output(ygr, True)
        time.sleep(1)
        GPIO.output(ygr, False)
        time.sleep(1)
        GPIO.output(ygo, True)
        time.sleep(1)
        GPIO.output(ygo, False)
        time.sleep(1)
        GPIO.output(ybo, True)
        time.sleep(1)
        GPIO.output(ybo, False)
        time.sleep(1)
        GPIO.output(ybr, True)
        time.sleep(1)
        GPIO.output(ybr, False)

        i += 1

    if keyboard.is_pressed("r"):
        i=1
    if keyboard.is_pressed("spacebar"):
        break
```

```
#Moviment D
if letter == "D":
     if it == "1":
          while i==1:
               GPIO.output(wgr, True)
               time.sleep(1)
               GPIO.output(wgr, False)
               time.sleep(1)
               GPIO.output(wgo, True)
               time.sleep(1)
               GPIO.output(wgo, False)
               time.sleep(1)
               GPIO.output(wbo, True)
               time.sleep(1)
               GPIO.output(wbo, False)
               time.sleep(1)
               GPIO.output(wbr, True)
               time.sleep(1)
               GPIO.output(wbr, False)

               i += 1

          if keyboard.is_pressed("r"):
               i=1
          if keyboard.is_pressed("spacebar"):
               break

     if it == "2":
          while i==1:
               GPIO.output(wgr, True)
               time.sleep(0.3)
               GPIO.output(wgr, False)
               time.sleep(0.3)
               GPIO.output(wgr, True)
               time.sleep(0.3)
               GPIO.output(wgr, False)
               time.sleep(1)
               GPIO.output(wgo, True)
               time.sleep(0.3)
               GPIO.output(wgo, False)
               time.sleep(0.3)
               GPIO.output(wgo, True)
               time.sleep(0.3)
               GPIO.output(wgo, False)
               time.sleep(1)
               GPIO.output(wbo, True)
               time.sleep(0.3)
               GPIO.output(wbo, False)
               time.sleep(0.3)
               GPIO.output(wbo, True)
               time.sleep(0.3)
               GPIO.output(wbo, False)
               time.sleep(1)
               GPIO.output(wbr, True)
               time.sleep(0.3)
               GPIO.output(wbr, False)
               time.sleep(0.3)
               GPIO.output(wbr, True)
               time.sleep(0.3)
               GPIO.output(wbr, False)

               i += 1

          if keyboard.is_pressed("r"):
               i=1
          if keyboard.is_pressed("spacebar"):
               break
```

```
     if it == "3":
          while i==1:
               GPIO.output(wgr, True)
               time.sleep(1)
               GPIO.output(wgr, False)
               time.sleep(1)
               GPIO.output(wbr, True)
               time.sleep(1)
               GPIO.output(wbr, False)
               time.sleep(1)
               GPIO.output(wbo, True)
               time.sleep(1)
               GPIO.output(wbo, False)
               time.sleep(1)
               GPIO.output(wgo, True)
               time.sleep(1)
               GPIO.output(wgo, False)

               i += 1

          if keyboard.is_pressed("r"):
               i=1
          if keyboard.is_pressed("spacebar"):
               break

#Moviment R
if letter == "R":
     if it == "1":
          while i==1:
               GPIO.output(ygr, True)
               time.sleep(1)
               GPIO.output(ygr, False)
               time.sleep(1)
               GPIO.output(ygo, True)
               time.sleep(1)
               GPIO.output(ygo, False)
               time.sleep(1)
               GPIO.output(wgo, True)
               time.sleep(1)
               GPIO.output(wgo, False)
               time.sleep(1)
               GPIO.output(wgr, True)
               time.sleep(1)
               GPIO.output(wgr, False)

               i += 1

          if keyboard.is_pressed("r"):
               i=1
          if keyboard.is_pressed("spacebar"):
               break

     if it == "2":
          while i==1:
               GPIO.output(ygr, True)
               time.sleep(0.3)
               GPIO.output(ygr, False)
               time.sleep(0.3)
               GPIO.output(ygr, True)
               time.sleep(0.3)
               GPIO.output(ygr, False)
               time.sleep(1)
               GPIO.output(ygo, True)
               time.sleep(0.3)
               GPIO.output(ygo, False)
               time.sleep(0.3)
               GPIO.output(ygo, True)
               time.sleep(0.3)
```

```
                    GPIO.output(ygo, False)
                    time.sleep(1)
                    GPIO.output(wgo, True)
                    time.sleep(0.3)
                    GPIO.output(wgo, False)
                    time.sleep(0.3)
                    GPIO.output(wgo, True)
                    time.sleep(0.3)
                    GPIO.output(wgo, False)
                    time.sleep(1)
                    GPIO.output(wgr, True)
                    time.sleep(0.3)
                    GPIO.output(wgr, False)
                    time.sleep(0.3)
                    GPIO.output(wgr, True)
                    time.sleep(0.3)
                    GPIO.output(wgr, False)

                    i += 1

               if keyboard.is_pressed("r"):
                    i=1
               if keyboard.is_pressed("spacebar"):
                    break

          if it == "3":
               while i==1:
                    GPIO.output(ygr, True)
                    time.sleep(1)
                    GPIO.output(ygr, False)
                    time.sleep(1)
                    GPIO.output(wgr, True)
                    time.sleep(1)
                    GPIO.output(wgr, False)
                    time.sleep(1)
                    GPIO.output(wgo, True)
                    time.sleep(1)
                    GPIO.output(wgo, False)
                    time.sleep(1)
                    GPIO.output(ygo, True)
                    time.sleep(1)
                    GPIO.output(ygo, False)

                    i += 1
               if keyboard.is_pressed("r"):
                    i=1
               if keyboard.is_pressed("spacebar"):
                    break

     #Moviment L
     if letter == "L":
          if it == "1":
               while i==1:
                    GPIO.output(ybr, True)
                    time.sleep(1)
                    GPIO.output(ybr, False)
                    time.sleep(1)
                    GPIO.output(wbr, True)
                    time.sleep(1)
                    GPIO.output(wbr, False)
                    time.sleep(1)
                    GPIO.output(wbo, True)
                    time.sleep(1)
                    GPIO.output(wbo, False)
                    time.sleep(1)
                    GPIO.output(ybo, True)
                    time.sleep(1)
                    GPIO.output(ybo, False)
```

```
            i += 1
        if keyboard.is_pressed("r"):
            i=1
        if keyboard.is_pressed("spacebar"):
            break

if it == "2":
    while i==1:
        GPIO.output(ybr, True)
        time.sleep(0.3)
        GPIO.output(ybr, False)
        time.sleep(0.3)
        GPIO.output(ybr, True)
        time.sleep(0.3)
        GPIO.output(ybr, False)
        time.sleep(1)
        GPIO.output(wbr, True)
        time.sleep(0.3)
        GPIO.output(wbr, False)
        time.sleep(0.3)
        GPIO.output(wbr, True)
        time.sleep(0.3)
        GPIO.output(wbr, False)
        time.sleep(1)
        GPIO.output(wbo, True)
        time.sleep(0.3)
        GPIO.output(wbo, False)
        time.sleep(0.3)
        GPIO.output(wbo, True)
        time.sleep(0.3)
        GPIO.output(wbo, False)
        time.sleep(1)
        GPIO.output(ybo, True)
        time.sleep(0.3)
        GPIO.output(ybo, False)
        time.sleep(0.3)
        GPIO.output(ybo, True)
        time.sleep(0.3)
        GPIO.output(ybo, False)

        i += 1

    if keyboard.is_pressed("r"):
        i=1
    if keyboard.is_pressed("spacebar"):
        break

if it == "3":
    while i==1:
        GPIO.output(ybr, True)
        time.sleep(1)
        GPIO.output(ybr, False)
        time.sleep(1)
        GPIO.output(ybo, True)
        time.sleep(1)
        GPIO.output(ybo, False)
        time.sleep(1)
        GPIO.output(wbo, True)
        time.sleep(1)
        GPIO.output(wbo, False)
        time.sleep(1)
        GPIO.output(wbr, True)
        time.sleep(1)
        GPIO.output(wbr, False)

        i += 1
```

```
            if keyboard.is_pressed("r"):
                 i=1
            if keyboard.is_pressed("spacebar"):
                 break

#Moviment F
if letter == "F":
     if it == "1":
          while i==1:
                 GPIO.output(ybr, True)
                 time.sleep(1)
                 GPIO.output(ybr, False)
                 time.sleep(1)
                 GPIO.output(ygr, True)
                 time.sleep(1)
                 GPIO.output(ygr, False)
                 time.sleep(1)
                 GPIO.output(wgr, True)
                 time.sleep(1)
                 GPIO.output(wgr, False)
                 time.sleep(1)
                 GPIO.output(wbr, True)
                 time.sleep(1)
                 GPIO.output(wbr, False)

                 i += 1

          if keyboard.is_pressed("r"):
                 i=1
          if keyboard.is_pressed("spacebar"):
                 break

     if it == "2":
          while i==1:
                 GPIO.output(ybr, True)
                 time.sleep(0.3)
                 GPIO.output(ybr, False)
                 time.sleep(0.3)
                 GPIO.output(ybr, True)
                 time.sleep(0.3)
                 GPIO.output(ybr, False)
                 time.sleep(1)
                 GPIO.output(ygr, True)
                 time.sleep(0.3)
                 GPIO.output(ygr, False)
                 time.sleep(0.3)
                 GPIO.output(ygr, True)
                 time.sleep(0.3)
                 GPIO.output(ygr, False)
                 time.sleep(1)
                 GPIO.output(wgr, True)
                 time.sleep(0.3)
                 GPIO.output(wgr, False)
                 time.sleep(0.3)
                 GPIO.output(wgr, True)
                 time.sleep(0.3)
                 GPIO.output(wgr, False)
                 time.sleep(1)
                 GPIO.output(wbr, True)
                 time.sleep(0.3)
                 GPIO.output(wbr, False)
                 time.sleep(0.3)
                 GPIO.output(wbr, True)
                 time.sleep(0.3)
                 GPIO.output(wbr, False)

                 i += 1
```

```
            if keyboard.is_pressed("r"):
                i=1
            if keyboard.is_pressed("spacebar"):
                break


        if it == "3":
            while i==1:
                GPIO.output(ybr, True)
                time.sleep(1)
                GPIO.output(ybr, False)
                time.sleep(1)
                GPIO.output(wbr, True)
                time.sleep(1)
                GPIO.output(wbr, False)
                time.sleep(1)
                GPIO.output(wgr, True)
                time.sleep(1)
                GPIO.output(wgr, False)
                time.sleep(1)
                GPIO.output(ygr, True)
                time.sleep(1)
                GPIO.output(ygr, False)

                i += 1

            if keyboard.is_pressed("r"):
                i=1
            if keyboard.is_pressed("spacebar"):
                break

    #Moviment B
    if letter == "B":
        if it == "1":
            while i==1:
                GPIO.output(ygo, True)
                time.sleep(1)
                GPIO.output(ygo, False)
                time.sleep(1)
                GPIO.output(ybo, True)
                time.sleep(1)
                GPIO.output(ybo, False)
                time.sleep(1)
                GPIO.output(wbo, True)
                time.sleep(1)
                GPIO.output(wbo, False)
                time.sleep(1)
                GPIO.output(wgo, True)
                time.sleep(1)
                GPIO.output(wgo, False)

                i += 1

            if keyboard.is_pressed("r"):
                i=1
            if keyboard.is_pressed("spacebar"):
            break

        if it == "2":
            while i==1:
                GPIO.output(ygo, True)
                time.sleep(0.3)
                GPIO.output(ygo, False)
                time.sleep(0.3)
                GPIO.output(ygo, True)
                time.sleep(0.3)
                GPIO.output(ygo, False)
                time.sleep(1)
```

```
                    GPIO.output(ybo, True)
                    time.sleep(0.3)
                    GPIO.output(ybo, False)
                    time.sleep(0.3)
                    GPIO.output(ybo, True)
                    time.sleep(0.3)
                    GPIO.output(ybo, False)
                    time.sleep(1)
                    GPIO.output(wbo, True)
                    time.sleep(0.3)
                    GPIO.output(wbo, False)
                    time.sleep(0.3)
                    GPIO.output(wbo, True)
                    time.sleep(0.3)
                    GPIO.output(wbo, False)
                    time.sleep(1)
                    GPIO.output(wgo, True)
                    time.sleep(0.3)
                    GPIO.output(wgo, False)
                    time.sleep(0.3)
                    GPIO.output(wgo, True)
                    time.sleep(0.3)
                    GPIO.output(wgo, False)

                    i += 1

               if keyboard.is_pressed("r"):
                    i=1
               if keyboard.is_pressed("spacebar"):
                    break

          if it == "3":
               while i==1:
                    GPIO.output(ybo, True)
                    time.sleep(1)
                    GPIO.output(ybo, False)
                    time.sleep(1)
                    GPIO.output(ygo, True)
                    time.sleep(1)
                    GPIO.output(ygo, False)
                    time.sleep(1)
                    GPIO.output(wgo, True)
                    time.sleep(1)
                    GPIO.output(wgo, False)
                    time.sleep(1)
                    GPIO.output(wbo, True)
                    time.sleep(1)
                    GPIO.output(wbo, False)

                    i += 1

               if keyboard.is_pressed("r"):
                    i=1
               if keyboard.is_pressed("spacebar"):
                    break

     print("")

print("Felicitats! Has resolt el cub de Rubik")

GPIO.output(ybo, True)
GPIO.output(ybr, True)
GPIO.output(ygo, True)
GPIO.output(ygr, True)
GPIO.output(wbo, True)
GPIO.output(wbr, True)
GPIO.output(wgo, True)
GPIO.output(wgr, True)
```

```
time.sleep(0.5)
GPIO.output(ybo, False)
GPIO.output(ybr, False)
GPIO.output(ygo, False)
GPIO.output(ygr, False)
GPIO.output(wbo, False)
GPIO.output(wbr, False)
GPIO.output(wgo, False)
GPIO.output(wgr, False)
time.sleep(0.7)

GPIO.output(ybo, True)
GPIO.output(ybr, True)
GPIO.output(ygo, True)
GPIO.output(ygr, True)
GPIO.output(wbo, True)
GPIO.output(wbr, True)
GPIO.output(wgo, True)
GPIO.output(wgr, True)
time.sleep(0.5)
GPIO.output(ybo, False)
GPIO.output(ybr, False)
GPIO.output(ygo, False)
GPIO.output(ygr, False)
GPIO.output(wbo, False)
GPIO.output(wbr, False)
GPIO.output(wgo, False)
GPIO.output(wgr, False)
time.sleep(0.7)

GPIO.output(ybo, True)
GPIO.output(ybr, True)
GPIO.output(ygo, True)
GPIO.output(ygr, True)
GPIO.output(wbo, True)
GPIO.output(wbr, True)
GPIO.output(wgo, True)
GPIO.output(wgr, True)
time.sleep(1.5)
GPIO.output(ybo, False)
GPIO.output(ybr, False)
GPIO.output(ygo, False)
GPIO.output(ygr, False)
GPIO.output(wbo, False)
GPIO.output(wbr, False)
GPIO.output(wgo, False)
GPIO.output(wgr, False)

GPIO.cleanup()
```

Lloc de descàrrega del programa que dona la solució del cub de Rubik:

https://github.com/hkociemba/RubiksCube-TwophaseSolver.git

F2L

Treball de Recerca