Treball de Fi de Grau/Màster

Grau en enginyeria en tecnologies industrials

DISSENY I CONSTRUCCIÓ D'UN INSTRUMENT PORTÀTIL DETECTOR DELS COMPONENTS RGB DE LLUM INCIDENT I ENREGISTRAMENT AL NÚVOL DELS COMPONENTS DETECTATS.

MEMÒRIA

Autor: Carlos Álvaro García
Director: Rosa Rodriguez Montañés

Convocatòria: Febrer 2018

ETSEIB
Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona

Resum

Aquest projecte consisteix en el disseny i construcció d'un dispositiu portàtil electrònic basat en un microcontrolador PIC (Peripheral Interface Controller). Aquest dispositiu és capaç de captar un feix de llum reflectit per un objecte mitjançant un sensor electrònic digital (que desglossa la intensitat dels components RGB presents en el feix) i, a través d'un mòdul Bluetooth, transmetre informació sobre els components RGB a un dispositiu mòbil Smartphone. Per altra banda, aquest projecte també presenta i detalla el desenvolupament de serveis en línia i un cloud (o núvol) personalitzat per a enregistrar a internet els components capturats.

Més concretament, el dispositiu portàtil presenta una bateria capaç de carregar-se amb una entrada MicroUSB, que alimentarà els diversos components presents en el sistema (sensor digital, mòdul BlueTooth, microcontrolador). El microcontrolador PIC és programat amb llenguatge C.

El telèfon SmartPhone du una aplicació nativa Android desenvolupada en Java i Kotlin que servirà de pont entre el dispositiu Portàtil i els serveis i el núvol -on s'emmagatzemarà la informació cromàtica rebuda-.

Així doncs, en aquest document es detallen els components que s'han utilitzat per a la realització de tots els sistemes esmentats, així com els passos que s'han seguit per implementar-los. També s'explica la programació d'aquests, tant la del microcontrolador, la de l'aplicació Android i la dels serveis en línia.

Sumari

1. Glossari

RGB(C): (de l'anglès *Red, Green, Blue [clear]*): És la composició del color en termes de la intensitat dels colors primaris de la llum: el vermell, el verd i el blau. Addicionalment, es normal parlar del component Clear, referit a la intensitat de la llum incident (sense descompondre).

Java: És el llenguatge més usat de programació natiu pels sistemes Android.

Kotlin: És el segon llenguatge més usat de programació natiu pels Sistemes Android.

Android: Sistema operatiu basat en Java, el més present en el món per a dispositius mòbils –Smartphones, tabletes,...-.

Node.js: Llenguatge basat en *javascript* ideat per a desenvolupar aplicacions mòbils escalables, fent èmfasi en les connexions amb servidors.

Frontend: Capa de software d'una aplicació amb la que interactua l'usuari.

Backend: Capa de software d'una aplicació formada pel conjunt d'elements, connexions i serveis amb els que no interactua l'usuari directament.

JSON: Acrònim de JavaScript Object Notation, «notació d'objecte de JavaScript») és un format de text creat per a l'intercanvi de dades comprensible i còmode d'operar per a humans. JSON és un subconjunt de la notació literal d'objectes de JavaScript que, a dia d'avui, per la seva àmplia adopció com a alternativa a XML, es considera un format de llenguatge independent.

API: (de l'anglès *Application programming Interface*): Conjunt de mètodes i funcionalitats que una determinada llibreria aporta per connectar dos sistemes de programació diferents.

Amazon Web Services (AWS): Conjunt de serveis i eines d'internet que permeten el desenvolupament de xarxes i serveis on-line.

NoSQL: Tipus de base de dades moderna que no funciona solament amb llenguatge SQL, sinó que presenta una estructura més flexible i permet fer desenvolupaments eficaços amb conjunts de dades complicats i poc jeràrquics.

Slave: Element electrònic que es comunica amb un màster, a petició d'aquest.

Master: Element electrònic que es comunica amb un slave i que comanda les comunicacions.

Baud rate**:** Quantitat de bauds –unitat de senyal- en una comunicació. Un baud pot contenir diversos bits.

UART (de l'anglès *Universal Asynchronous Receiver-Transmitter*): Nom que comunament rep l'EUSART.

EUSART (de l'anglès *Enhanced Universal Synchronous Asynchronous Receiver- Transmitter*): Dispositiu que converteix dades de paral·lel a sèrie o de sèrie a paral·lel per tal de transmetre o rebre informació, respectivament.

I/O (de l'anglès *Input/Output*): Entrada/sortida.

Microcontrolador: Circuit encapsulat integrat que inclou una petita CPU, unitats de memòria, ports d'entrada i sortida i perifèrics.

Perifèric: Aparell o dispositiu auxiliar i independent connectat a la CPU.

PIC (de l'anglès *Peripheral Interface Controller*): Família de microcontroladors fabricats per *Microchip Technology Inc*.

2. Prefaci

2.1. Origen del projecte i Motivació

La motivació per la realització d'aquest projecte neix dels mals resultats aconseguits per l'autor en mesclar pintura, intentant obtenir un color prèviament vist .

Si l'enginyeria és aquell conjunt de tècniques que permeten l'aplicació dels coneixements científics a la indústria, les comunicacions, l'agricultura, l'aeronàutica, les obres publiques… per tal de solucionar un problema o una carència existent, aquest projecte és precisament una aplicació directa d'aquest principi. L'autor del projecte presenta aquest treball com un prototip per una solució holística dedicada a aquelles persones que, a l'intentar reconèixer o recordar el color d'un objecte tangible, es troben que no són capaces d'encertar amb la identificació d'aquest color, ja sigui per a dibuixar una flor que han vist, per pintar una paret o fer un disseny.

Part de la motivació neix en l'assignatura optativa Taller d'Electrònica cursada a l'ETSEIB, on un conjunt de tècniques i mòduls electrònics són presentats per tal de que l'alumne copsi i s'aproximi al prototipatge electrònic, fet que va motivar a l'autor a intentar aplicar els coneixements apresos a resoldre un problema real amb aquelles tecnologies i mètodes coneguts llavors.

2.2. Requeriments previs

Per la realització d'aquest projecte cal ser coneixedor de diverses eines i tecnologies existents, majoritàriament de l'àmbit de l'electrònica i la programació. Més concretament, cal coneixements sobre electricitat i funcionament de mòduls electrònics, així com protocols de comunicació i serialització, i programació de baix nivell C.

També cal conèixer tecnologies de Software, especial atenció en la programació de dispositius mòbils (*Java i Kotlin*) a través *d'Android Studio*. També cal disposar de coneixements sobre tecnologia de *backends* informàtics, de base de dades tipus noSQL i cal conèixer eines de creació de serveis on-line, amb la posterior generació d'*APIs*. Per gestionar les bases de dades noSQL cal conèixer llenguatges per manipular bases de dades (*Node.js*, per exemple). Per aquest projecte cal conèixer eines per crear, en definitiva, un núvol privat (amb Amazon Web Services, per exemple).

3. Introducció

3.1. Objectius del projecte

L'objectiu principal és prototipar un dispositiu funcional electrònic capaç d'identificar els components RGB de la llum incident sobre aquest i amb un notable nivell d'autonomia –és a dir, disposar de bateria i ser recarregable-. El dispositiu està controlat per un microcontrolador PIC i l'usuari s'hi connecta a través d'un Smartphone amb sistema Android. A més a més, les dades es guarden al núvol.

D'aquest objectiu principal se'n desprenen altres igual de crítics, com la prova de diferents sensors digitals identificadors de components RGB i la posterior tria del més adequat per al projecte. Altre objectiu és el desenvolupament d'una aplicació Android en codi natiu Java i Kotlin i, així, incidir i aprofundir en la programació en aquests llenguatges.

L'altre objectiu és crear serveis on-line que permetin emmagatzemar al núvol tots els colors mesurats i, així, tenir-los disponibles des de qualsevol aparell Android amb connexió a internet.

3.2. Abast del projecte

Una de les limitacions d'aquest projecte és el pressupost invertit en el material del projecte, fet que repercuteix directament en la qualitat dels sensors captadors de llum en RGB –un dels mòdul més rellevants del projecte-, limitant així la precisió i resolució dels resultats obtinguts.

Per reduir l'abast del projecte també s'ha optat per desenvolupar l'aplicació Smartphone sols per dispositius portàtils Android (els més comuns a Espanya **[1]**).

Es pretén fer també un backend per aplicació mòbil amb prestacions funcionals però no completes com s'esperaria d'una aplicació ready-to-sell, per exemple, no es té l'ambició de dotar la aplicació amb un sistema d'usuaris (creació de comptes personals) o de fer consideracions estètiques alhora de dissenyar-la.

4. Components del dispositiu Portàtil

El dispositiu portàtil desenvolupat en aquest projecte presenta diversos elements i mòduls electrònics interconnectats. En aquest capítol se'n detallen tres dels més importants, que són el microcontrolador, el sensor de color considerat i el mòdul Bluetooth. A més a més, es presenta informació referent a l'alimentació del sistema.

L'element del sistema més determinant sobre la qualitat dels resultats obtinguts és el sensor que capta la llum i obté la intensitat dels diferents components en RGB. Aquest sensor ha d'estar controlat i comandat per el microcontrolador, que és un dels altres elements clau del sistema. El microcontrolador també comanda el mòdul Bluetooth per establir connexions amb l'exterior.

4.1. Microcontrolador PIC16f690

S'ha triat utilitzar en el dispositiu portàtil dissenyat en aquest projecte el microcontrolador PIC16F690 de *Microchip Technology Inc.* per la seva versatilitat, capacitat, mida i preu que cobreixen perfectament els requeriments del projecte. A més, amb aquest microcontrolador s'ha treballat amb anterioritat en alguna de les assignatures del grau. Per tant, aquest element comandarà el sensor i les comunicacions via Bluetooth amb l'exterior del dispositiu portàtil.

4.1.1. Característiques

A la següent taula es mostren les característiques generals, extretes del Datasheet del fabricant **[2].**

Device	Program Memory	Data Memory		I/O	10-bit A/D (ch)	Comparators	Timers 8/16-bit	SSP	ECCP+	EUSART
	Flash (words)	SRAM (bytes)	EEPROM (bytes)							
PIC16F690	4096	256	256	18	12	2	2/1	Yes	Yes	Yes

4.1.1-1 Taula de generalitats del microcontrolador pic16f690

Entre les característiques més rellevants d'aquest microcontrolador es destaca que:

- Té una memòria de programa de 4096 paraules, on una paraula és un grup de 14 bits que conformen una instrucció bàsica del microcontrolador. Això permet elaborar programes de complexitat suficient com per encabir el projecte.

- Disposa d'un total de 20 pins, 18 dedicats a la comunicació exterior i 2 a l'alimentació. La comunicació es fa efectiva a través de 3 ports A B i C; de 5, 4 i 8 pins cadascun, respectivament. En la següent imatge es desglossen els pins dels que disposa el PIC16f690.

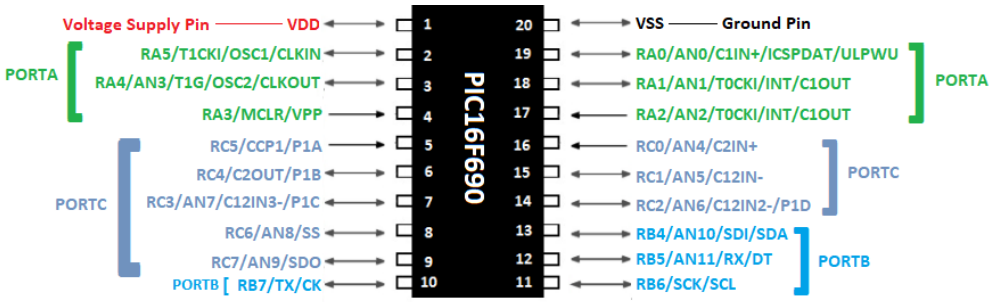


Figura 4.1.1-2. Pinout *del PIC16F690*

- Disposa del perifèric d'entrada/sortida anomenat UART (Universal Asynchronous Receiver Transmitter) –EUSART segons el fabricant-, que és un mòdul que utilitza un protocol de comunicació que permet connectar i transmetre dades bidireccionalment entre els dos elements connectats, de manera assíncrona –sense rellotge-, emmagatzemant la informació en una sèrie de registres que utilitza a tal efecte. En aquest PIC16F690, la comunicació pel mòdul UART es fa efectiva a través dels pins 12 i 10, configurats com a senyals RX (*receiver*) i TX (*transmitter*), respectivament. Aquest tipus de connexió s'empra en la comunicació entre el PIC16F690 i el mòdul Bluetooth present en el dispositiu portàtil. La comunicació està estandarditzada i es disposa de llibreries que faciliten l'ús d'aquesta.

- Disposa de perifèric d'entrada/sortida I2C (Inter-Integrated Circuit), que és un bus de dades sèrie bidireccional síncron que permet transmetre informació a través dels pins 11 i 13, configurats com SCL (rellotge) i SDA (dades), respectivament. S'ha fet especial èmfasi en aquest protocol en aquest treball degut a que no es disposa de llibreries públiques per facilitar les comunicacions entre sensor i microcontrolador (a través de protocol I2C) i, per tant, s'ha desenvolupat investigació referent a aquest bus. Per a més informació sobre aquest bus en aquest projecte, veieu els punts 4.2.2 i 6.3.

- Disposa de servei d'interrupcions. Un servei d'interrupció consisteix, generalment, en un bloc de codi associat a una interrupció de hardware, és a dir, un fragment de codi que s'executa quan es compleix una condició lògica elèctrica de baix nivell. El servei

d'interrupció, llavors, permet aturar el programa que s'estigués executant en aquell moment en detriment del codi del servei d'interrupció. Aquesta rutina es coneix com ISR, *(Interrupt Service Routine*). Per aquest projecte, es treballa amb una interrupció que es desencadena en prémer un polsador, per tal de que, en aquell moment que l'usuari ho vulgui, s'executi un codi que permeti identificar el color que es vulgui reconèixer. Per a poder-ho fer, caldrà habilitar les interrupcions modificant els registres corresponents, com es s'explicita en l'apartat 6.4

4.2. Sensor RGB

Un sensor és un dispositiu que converteix senyals físiques en senyals elèctriques. Més concretament, pel projecte present es treballa amb sensors RGB digitals que, a trets generals, són uns mòduls electrònics que presenten una matriu de 4 conjunts de fotodíodes, cadascun amb diferents filtres òptics pels components Red, Green i Blue i clear; respectivament. Els fotodíodes, sensibles a la llum incident, estan distribuïts en el sensor junt amb els filtres òptics per rebre llum reflectida per un objecte i poder-ne extreure i determinar la intensitat de diferents freqüències –és a dir, de diferents colors primaris-.

El mercat presenta una varietat àmplia de sensors capaços de mesurar el color -entre altres funcions- de la llum incident. En aquest projecte es comparen dos sensors digitals de baix cost, l' APDS9960 i el TCS34725. A continuació se'n detallen algunes característiques importants per a la realització del treball.

4.2.1. Comparativa general dels sensors APDS9960 i TCS34725

Aquests dos sensors presenten un seguit de característiques que els fan rellevants respecte altres sensors del mercat, especialment pel preu. . A la Figura 4.2.1-1 es mostren les fotografies de les plaques de circuit imprès que els contenen i han estat utilitzades en aquest projecte.

Figura 4.2.1-1 Sensor APS9960 i TC34725 d'Adafruit

Es presenta a continuació la taula *4.2.1-1* amb un desglossament de trets generals d'aquests sensors, més concretament dels sensors APDS9960 i TCS34725 muntats i venuts per *Adafruit*[1].

	APDS9960	TCS34725
Protocol comunicació	Digital, i2c	Digital, i2c
Voltatge entrada	3.3-5v	3.3-5v
Filtres	UV i IR	UV i IR
Altres funcions	Detecció proximitat Detecció gestos	Detecció proximitat Detecció gestos
Detecció temperatura color	Sí	Sí
LED blanc incorporat	No	Sí
Preu	7.5€	7.95€
Mida	17.8mm*17.8mm	20.44mm*20.88mm

Taula 4.2.1-1 Característiques generals dels sensors APDS i TCS

Com s'observa, aparentment són sensors de característiques generals similars. Per determinar el sensor més adient per incloure'l en el dispositiu portàtil, es fa una comparativa funcional.

4.2.2. Protocol I2C

El protocol I2C s'aplica sobre un bus sèrie de dades que requereix de dos línies per transmetre informació, una que actua com a rellotge (SCL) i serveix per regularitzar l'enviament i recepció de dades i, l'altre, la línia de dades (SDA), que és aquella per on es transmeten les dades. El protocol funciona de tal forma que hi ha un controlador *(Master*) que escriu i llegeix registres d'un perifèric (*Slave*).

Aquest protocol el va desenvolupar Philips[2] al 1992 i és àmpliament utilitzat per comunicar controladors i perifèrics.

Com s'ha esmentat anteriorment, els dos sensors es comuniquen usant aquest protocol i el PIC16f690 és compatible amb aquest bus. Més concretament, el PIC16f690 actua com a *Master*, enviant per la línia SCL el seguit de bits que actuen com a rellotge i, per la SDA, es reben i s'envien les dades. Aquestes dades s'han d'enviar i rebre seguint el protocol detallat

[1] Adafruit Industries és una companyia de hardware open-source que dissenya, manufactura i ven components electrònics diversos. Consulteu el seu web a http://www.adafruit.com

[2] Per a més información sobre aquest protocol, podeu consultar el web https://www.i2c-bus.org/

a continuació i esquematitzat en la Figura 4.2.2-1.

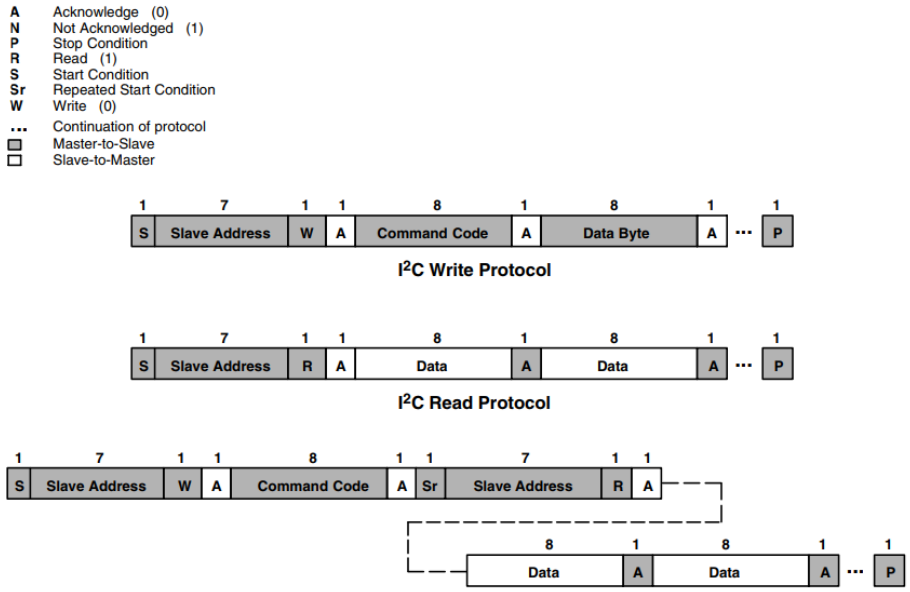
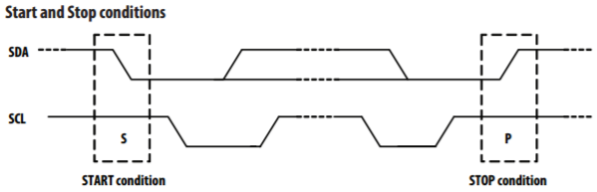


Figura 4.2.2-1 Format del Protocol I2C. S'observa en la llegenda que hi ha diversos elements que es repeteixen en el protocol, com l'Start (S), o el Acknowledge (A). S'observa també la quantitat de bits que conformen cada element just amunt de cada element.

Per iniciar la comunicació, es necessita una condició de Start, que es dóna quan el bit per la línia SDA, que inicialment val 1, realitza un flanc de baixada a 0 de forma simultània a un bit amb valor 1 en la línia de rellotge (Figura 4.2.2.-2).



4.2.2.-2 Condicions d'Start i Stop de la comunicació

Per explicitar el funcionament del protocol, es detalla un segment de l'operativa d'aquest: Mitjançant el PIC16f690 s'escriu una seqüència[3] com la proposada en la última línia de la imatge 4.2.2-1. És a dir, una seqüència START -> SLAVE (l'adreça identificadora del perifèric. Per l'APDS9960, 0x39) -> W (És a dir, un 0, com diu la llegenda del protocol). Llavors s'espera la resposta del perifèric. En cas de que tot sigui correcte, ha de retornar una A -Acknowledge, és a dir, un 0-.

En les següents imatges es mostra la línia SDA (senyal en la part superior de les Figures

[3] Per veure la programació que hi ha darrere d'aquesta operativa, consultar l'apartat 6.3

4.2.2-3 i 4.2.2-4) i la línia SCL (part inferior de les mateixes Figures) vistes amb un oscil·loscopi. Es mostra com el perifèric sols respon amb un ACK (0) en cas d'enviar una adreça de SLAVE correcta. Per aquest exemple, es connecta el PIC i el perifèric –en aquest cas, un APDS9960, amb adreça SLAVE amb valor 0x39, tal com detalla el seu DataSheet **[3].**

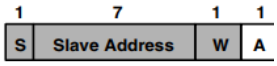


Figura 4.2.2-2 Fragment de l'operativa en el I2C

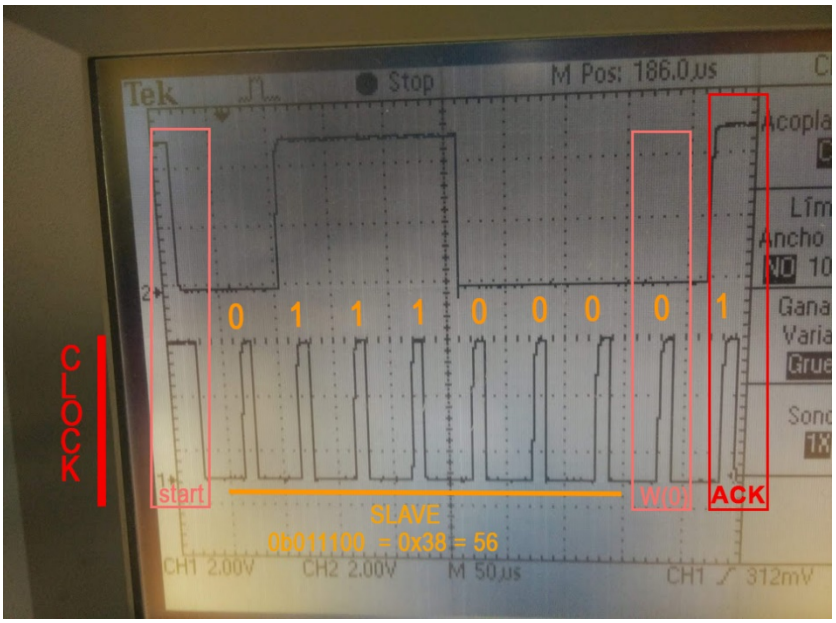


Figura 4.2.2-3 Enviament d'un 0x38 com a SLAVE (valor erroni)

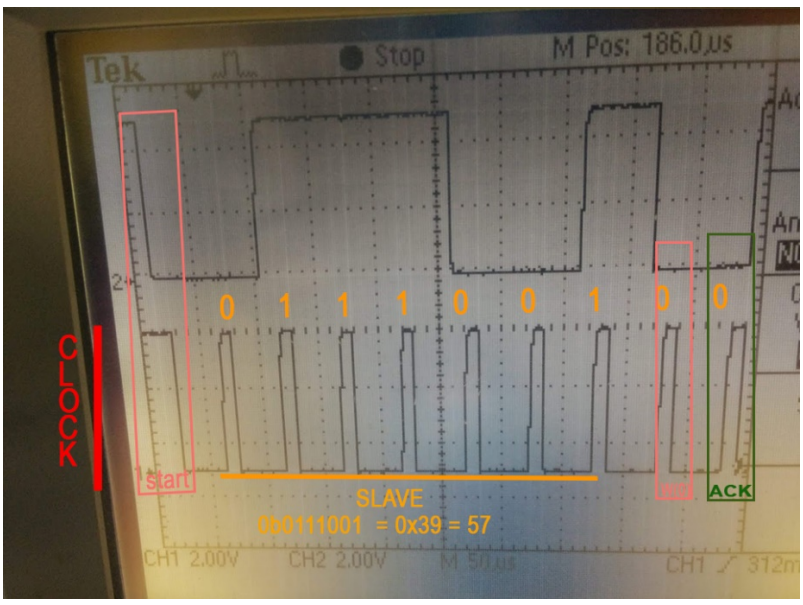


Figura 4.2.2-4 Enviament d'un 0x39 com a SLAVE

Tal com mostren les imatges, s'ha provat d'enviar un 0x38 –una adreça no correcta a propòsit- per comprovar el funcionament del protocol. Els resultats són els esperats, el ACK que ha de retornar el perifèric no es produeix. Quan s'envia un 0x39 (l'adreça correcta) el perifèric respon perfectament. Per tant, s'ha comprovat que el protocol I2C permet una comunicació bilateral entre controlador i perifèric i que és correcta.

4.2.3. Comparativa de resultats entre l'APDS9960 i el TCS34725

En aquest apartat es presenta part del resultat experimental comparatiu derivat d'utilitzar els dos sensors per mesurar diferents colors -llums reflectides per objectes- en unes mateixes condicions. Primerament, es presenten les condicions de treball. Els sensors són col·locats dins un recipient de parets negres per aïllar-lo de llum intrusiva que pugui introduir error en la mesura (Figura *4.2.3-1*).. També s'introdueix un LED blanc neutre al recipient per tal de que existeixi llum perquè l'objecte la pugui reflectir.

Figura 4.2.3-1, recipient amb un LED apagat i encès, respectivament

Seguidament, es presenten tres objectes *de diferents colors*[4] que seran mesurats pels sensors. Aquests tres objectes són làmines de goma EVA (Figura 4.2.3-2)..

[4] Cal recordar que no existeix tal cosa com 'el color d'un objecte'. Les fotos de les figures 4.2.3-2 mostren uns colors determinats perquè les fotos s'han realitzat en unes condicions de llum determinades, diferents a les condicions d'operació del sensor, i diferents a les que opera l'ull humà.

4.2.3-2 Fotografies de les superfícies de goma EVA

Per disminuir la varietat cromàtica que aprecia l'ull en les diferents fotografies i poder comparar millor amb els sensors, tractem la imatge amb un processador d'imatge per obtenir la mitjana del color de les fotografies. Utilitzem, en aquest cas, el Software *Adobe Photoshop cs6*, i apliquem un filtre *de Average Blur.*

El resultat de les transformacions és aquest:

4.2.3-3 Average *dels colors de les superficies de goma Eva*

Per les proves dels sensors, s'ha utilitzat un Arduino Uno, que és una plataforma de desenvolupament fàcil i ràpida. En aquest capítol no es detallarà el funcionament de l'Arduino ni de la comunicació amb els sensors, tot i que el protocol de comunicació és l'I2C, el mateix utilitzat amb el microcontrolador PIC16F690.

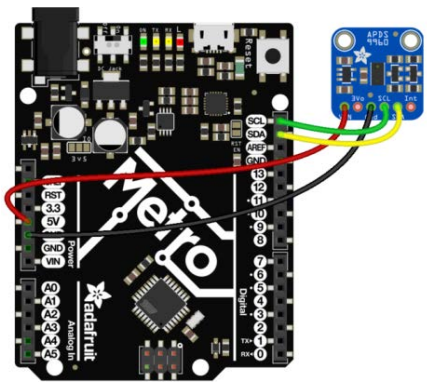


Figura 4.2.3-4 Aquí veiem les connexions entre una placa de desenvolupament –Tipus Arduino Uno- i un sensor RGB, en aquest cas, l'apds9960 (Cal recordar que les connexions pels dos sensors són iguals).

Es mostra en la següent taula alguns dels resultats de les proves. S'han mesurat 10 vegades cada color amb cada sensor i s'ha obtingut el valor mitjà. Dins el rectangle del color corresponent, s'indiquen els components amb l'ordre RGB en una escala del 0 al 255.

.

Mitjana cromàtica de l'objecte	R 179,47,52		G 108,174,105		B 40,77,155	
Color mesurat pels sensors	TC 171,47,46	APDS 149,66,83	TC 85,179,85	APDS 56,184,94	TC 40,79,137	APDS 26,85,184

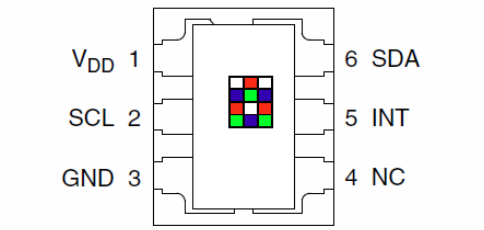
Taula 4.2.3-1 Colors obtinguts amb els dos sensors comparats amb els dels objectes.

Es comprova que els valors obtinguts pel sensor TC són lleugerament millors que els obtinguts pel sensor APDS, especialment pel component vermell. S'observa que el sensor APDS presenta valors baixos pel component vermell de la llum i valors alts pels components blaus, desvirtuant així els colors obtinguts, exagerant els blaus (comparat amb el TC) per a tots els colors i minvant els vermells..

Per tant, es considera descartar el sensor APDS9960 i utilitzar el TCS34725 pel muntatge final.

4.2.4. El Sensor TCS4725

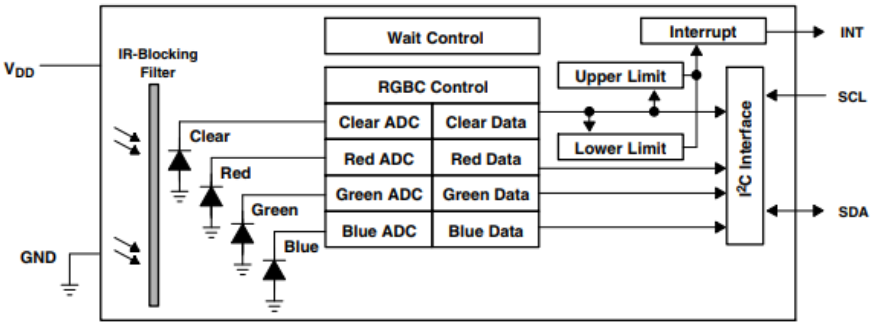
El sensor TCS34725 és un sensor amb resposta digital que compta amb una matriu que bloca rajos infrarojos i que està formada per 3x4 fotodíodes amb filtres per cada component RGBC.A la Figura *4.2.4-1* es mostra el seu Pinout.



4.2.4-1 Sensor TCS34725 i matriu de fotodíodes

També disposa de 4 convertidors Analògic-digital (ADC), registres de dades, una màquina d'estats i una interfície I2C. Aquest sensor, llavors, és capaç de dur a terme funcions com, per xemple, retornar digitalment a través dels pins SDA i SCL (protocol I2C) les components RGBC (intensitat de cada component) de la llum incident sobre ell, integrant el corrent que

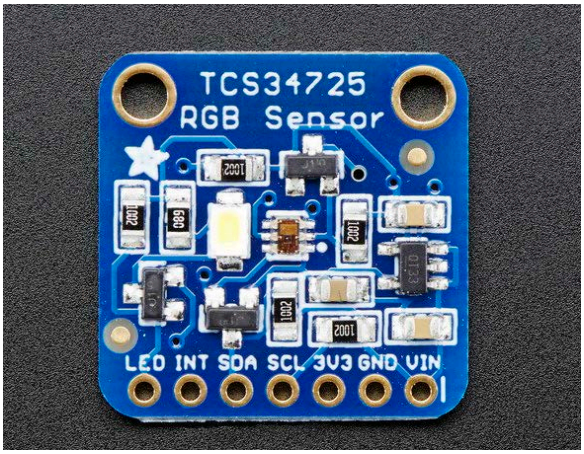
passa pels fotodíodes. També té serveis d'interrupcions -activacions del sensor a través de certes condicions relacionades amb llindars relatius a la intensitat de llum rebuda-, capacitat per operar com a detector de gestos i sensor de color de temperatura; tot i que aquestes 3 últimes prestacions no són usades pel projecte. Aquestes funcionalitats es troben detallades al datasheet del sensor **[4]**.



4.2.4-2 Diagrama explicatiu amb caixes del Sensor TCS34725

El sensor té registre per modificar el temps d'integració i el guany de les dades que retorna, a través dels registres **RGBC Control** com detalla el diagrama 4.2.4-2. A l'apartat 6.4 es detalla amb més concreció alguns aspectes sobre els registres del sensor.

En aquest projecte, s'opta pel sensor muntat en circuit imprès per *Adafruit* (Figura *4.2.4-3)*, que ho fa en una placa on l'entrada de voltatge pot ser de 3.3V-5VDC i on hi ha un pin extra per activar un LED Neutral de 4150K integrat, cosa que simplifica el muntatge del dispositiu final, que requereix d'una il·luminació blanca per detectar la llum que es reflecteix en els objectes.



4.2.4-3 Sensor TCS34725 muntat per Adafruit, amb els pins SDA i SCL per la comunicació I2C. El pin GND i VIN estan connectats als senyals de terra i l'alimentació 3.3-5V, respectivament.

4.3. Mòdul Bluetooth HC-05

Els mòduls Bluetooth són components electrònics que permeten comunicar-se entre ells mitjançant tecnologies RFID, és a dir, sense necessitat d'emprar cables, creant així petites xarxes sense fil que faciliten transmissió de dades i sincronització de dispositius. És normal que la majoria d'Smartphones i ordinadors vinguin equipats amb aquests mòduls; també estan presents en molts perifèrics (altaveus, teclats, ratolins...).

Per aquest projecte es proposa utilitzar un mòdul Bluetooth HC-05 (Figura 4.3-1) per enviar dades des del Microcontrolador fins a l'Smartphone. Per a fer-ho, s'utilitza el bloc intern de comunicació UART per transmetre dades a través dels pins TX i RX del Microcontrolador fins als pins RX i TX del mòdul Bluetooth, que, posteriorment, els enviarà mitjançant una xarxa RFID a l'smartphone.

El mòdul emprat en aquest projecte és el BlueTooth HC-05 del fabricant *Strat-Num, INC* que presenta, per un baix cost, prestacions suficients per encabir-se en el projecte i amb un baix consum elèctric. **[5]**

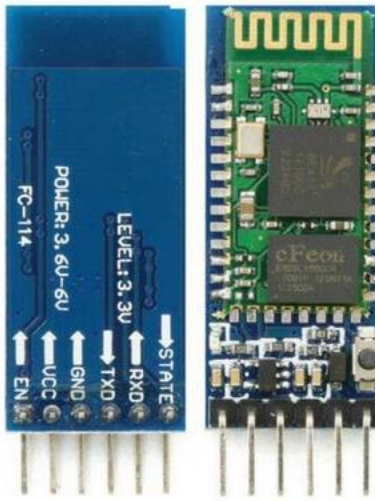


Figura 4.3-1 El mòdul Bluetooth presenta pins TX i RX per on es fa efectiva la comunicació, el pin GND correspon al terra elèctric i el pin VCC correspon a l' alimentació a 3.3-5V.

4.4. Alimentació

L'alimentació del dispositiu portàtil es vol configurar com a recarregable, és a dir, sense utilitzar piles ni elements que siguin considerats d'un sol ús. Per a recarregar el dispositiu en aquest dispositiu, llavors, s'empren 3 elements diferents que es comenten a continuació.

4.4.1 Bateria

En aquest cas es tria per motius econòmics, de disponibilitat i sostenibilitat, utilitzar una bateria d'un Smartphone en desús i reciclar així la seva utilitat. Es tracta de la bateria BL-5K de Nokia, amb una capacitat de 1200mAh i una sortida de 3.7V de la qual se'n mostra una fotografia.

Figura 4.4.1-1 Veiem que la bateria disposa de 3 entrades. Començant per la dreta: Alimentació positiva, terra i sensor de temperatura.

4.4.2 Adaptador microUSB-bateria

Per a efectuar la càrrega de la bateria, s'ha de suplir energia des de l'exterior. Per connectar la bateria amb l'exterior, es decideix adaptar la càrrega de la bateria a través d'un port microUSB, un port comunament usat per molts dispositius electrònics (Smartphones, per exemple). Es decideix utilitzar un panell amb port microUSB i amb circuit integrat protector (p.e.: es desconnecta quan la bateria està carregada) i que disposi de pins de sortida soldables. S'utilitza el mòdul TP4056 de Houyu Electronics **[6]** il·lustrat a la Figura *4.4.2-1*.

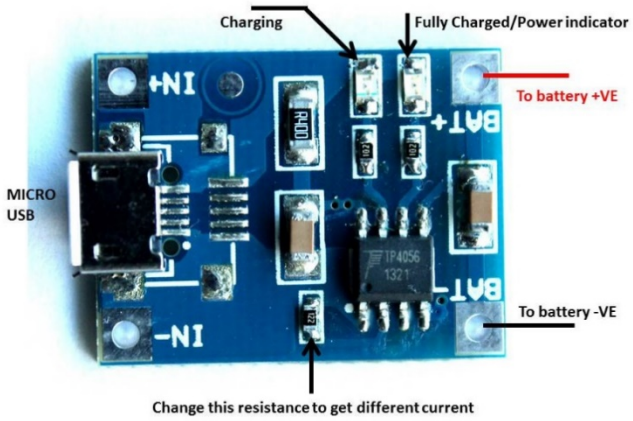
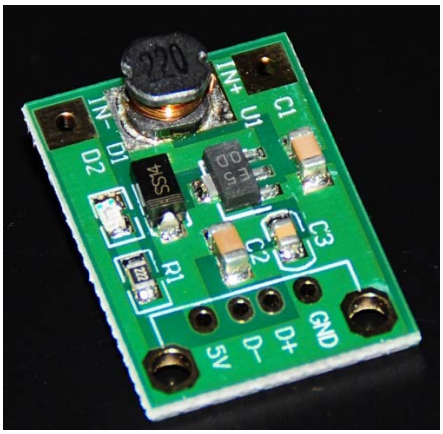


Figura 4.4.2-1 Veiem, d'esquerra a dreta, l'entrada microUSB de 4.5-5.5V. Una resistència ajustable per aconseguir diferents corrents de sortida per alimentar la bateria. Els LEDS indicadors d'estat de càrrega de la bateria. I, finalment, els pins preparats per ser soldats i que alimentaran la bateria.

4.4.3 Step-up i regulador de voltatge

Finalment, l'energia, abans d'arribar al circuit digital del dispositiu, es fa passar per un Step-UP de voltatge, que incrementarà els 3.7V de la bateria fins a 5V i també els estabilitzarà i els regularà, mantenint un voltatge constant pel funcionament òptim del dispositiu. En aquest cas, s'utilitza el Step-up DC-DC5V de Hyelesiontek **[7]** (Figura 4.4.3-1) per oferir les prestacions necessàries.



4.4.3-1 El Step UP té pints IN+ i IN- per on ha de rebre 1-5V. La Sortida, GND i 5V són els parell de pins que alimentaran el circuit digital final del dispositiu.

4.5 Interruptors

Per tenir control sobre l'alimentació i activació del circuit, s'integren dos elements interruptors que es comenten seguidament.

4.5.1 Interruptor d'alimentació

Per desconnectar el circuit digital de la part del circuit de potència, es col·loca un interruptor que aïlli (o connecti) la bateria del regulador de voltatge, immediatament connectat al circuit digital.
S'utilitza un Rocker Switch SPST **[8]** de dos posicions per la seva idoneïtat de mida i ergonomia.

Figura 4.5.1-1. Switch de dos posicions emprat en el projecte

4.5.2 Polsador desencadenador d Interrupcions

Com s'ha esmentat anteriorment en l'apartat 4.1.1, el Microcontrolador disposa de servei d'interrupcions, que permet executar un codi quan es dóna alguna condició elèctrica en una entrada concreta del PIC16F690. Més concretament pel projecte que es du a terme, en el moment en el que en una entrada seleccionada es connecta a un 0 digital, s'acciona aquest servei d'interrupció. Per a fer-ho, es disposa un polsador que, en prémer-se, connectaria l'entrada del PIC amb el 0 digital.

S'utilitza un *11.535.P/A pulsador empotrable* d'*Electro DH* per la seva idoneïtat de mida i d'ergonomia.

4.5.2-1 Polsador normalment obert utilitzat en el projecte

5. Arquitectura del dispositiu portàtil

En aquest capítot es detalla les connexions entre els diferents elements del dispositiu portàtil, es detalla l'elaboració de la carcassa sobre la que s'acomoden els diferents elements electrònics i se'n detalla l'aspecte final del muntatge físic..

5.1. Carcassa

La carcassa ha d'allotjar i protegir els elements electrònics.

5.1.1. Consideracions generals

Es considera un tipus de carcassa similar a la d'una llanterna, que ha de complir les següents condicions:

- Capaç d'allotjar tots els elements electrònics, inclosa la bateria, que té una mida de 8x8x1.5cm.
- Ha de tenir una obertura per on pugui sortir llum provinent del LED del sensor i l'obertura ha de ser capaç de rebre la llum que l'objecte reflecteix, per a que incideixi sobre el sensor.
- Ha d'amagar els dispositius electrònics i evitar contacte directe de l'usuari amb ells.
- Ha de comptar amb una ranura per on pugui entrar un connector tipus MicroUSB per alimentar la bateria.
- Ha de comptar amb ranures on poder col·locar els elements de control de l'usuari: L'Interruptor i el polsador.
- Ha d'evitar arestes vives per ergonomia i seguretat.

Per a complir totes les característiques, es mesuren tots eles elements electrònics i de control dels que es disposa i es procedeix a fer un disseny iteratiu.

Es decideix fer la carcassa amb tecnologia d'impressió 3D, concretament amb tecnologia FFF (de l'anglès *fused filament Fabrication*), usant com a matèria prima el plàstic PLA, per ser un termoplàstic barat i biodegradable, derivat de recursos renovables, fent així que la fabricació de la carcassa s'obtingui per un preu assequible, de forma ràpida, comptant amb ordres de magnitud de precisió i de resistència de material suficients pel prototip que es vol dur a terme.

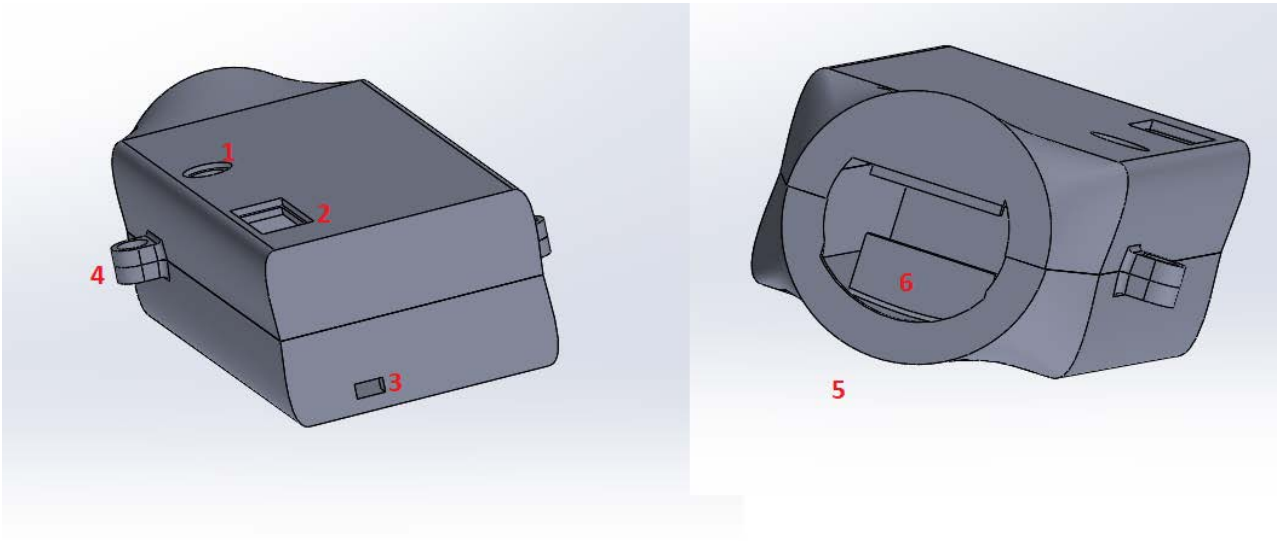
5.1.2. Model 3D

Es dissenya un model 3D a partir de SolidWorks, amb les següents consideracions

específiques de la tecnologia amb la que sap que es fabricarà:

- En impressió 3D per FFF amb PLA, els forats en el material es contrauen dècimes de mil·límetres; a tenir en compte alhora d'escollir les cotes i les toleràncies adequades per ajustar els diferents elements.
- Els voladissos són difícils d'imprimir ja que requereixen d'imprimir material de suport; a evitar si és possible.
- En peces altes, el fenomen d'histèresis pot aparèixer a mesura que s'avança verticalment en la impressió; és millor imprimir peces horitzontals.

El disseny de la carcassa es fragmenta en dos parts perquè, d'altra forma, alguna paret hauria quedat en voladís, dificultant i encarint la impressió. L'assemblatge és el mostrat a la següent Figura:



Figuta 5.1.2.-1 Carcassa del dispositiu portàtil. En vermell: 1) Ranura ajustada pel polsador 2) Ranura ajustada per l'interruptor 3) Ranura per passar el connector MicroUSB i alimentar la bateria 4) Solcs per connectar les dues peces de la carcassa amb passadors 5) Obertura per on surt la llum 6) Paret on fixar el sensor en verticalitat

En les següents dues figures es mostren els plànols - en mida reduïda- amb cotes funcionals de les dues peces de les carcasses. Per consultar a mida completa el plànol tècnic, visitar l'**annex**.

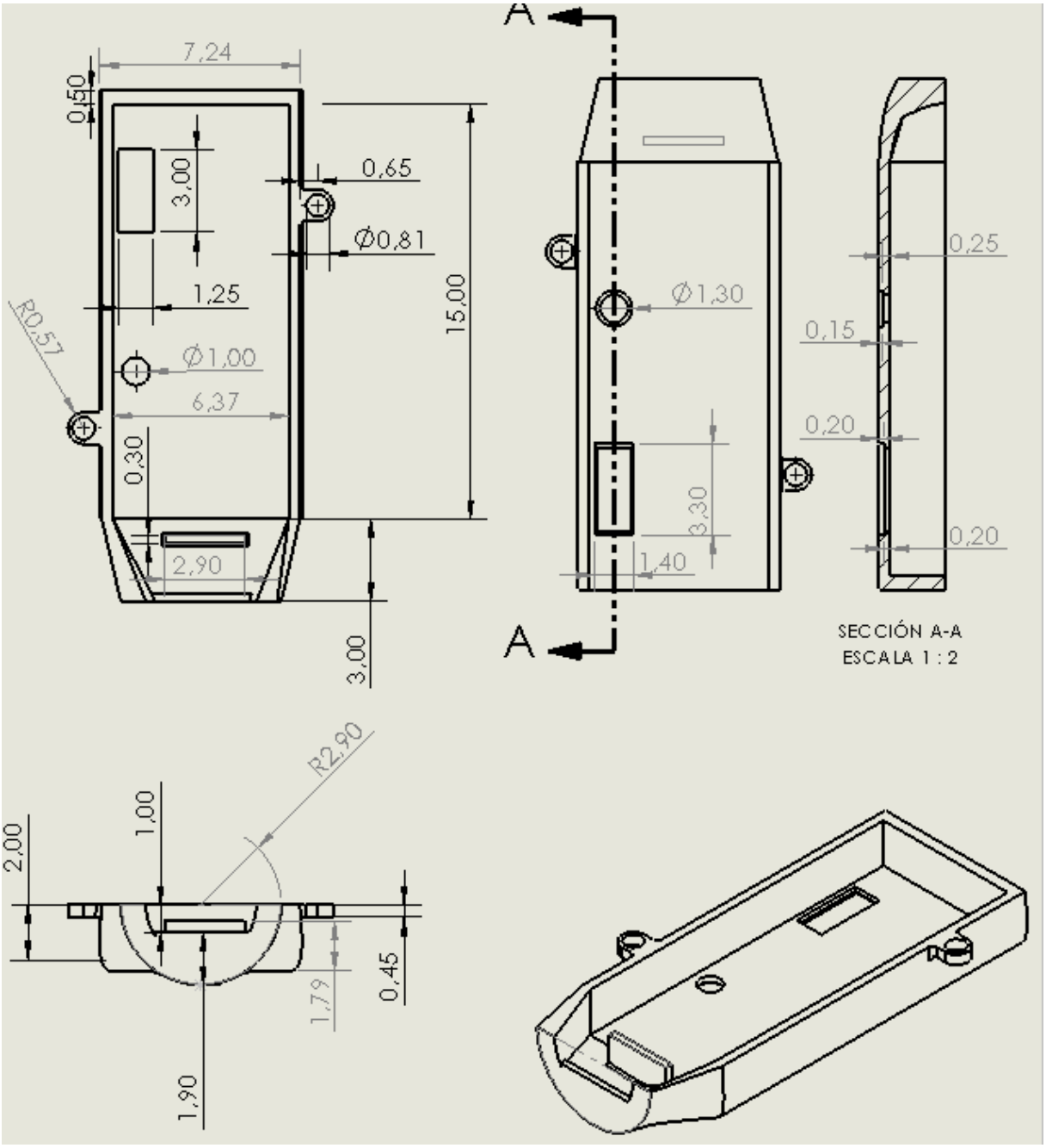


Figura 5.1.2-2 Miniatura del plànol tècnic de la part superior de la carcassa

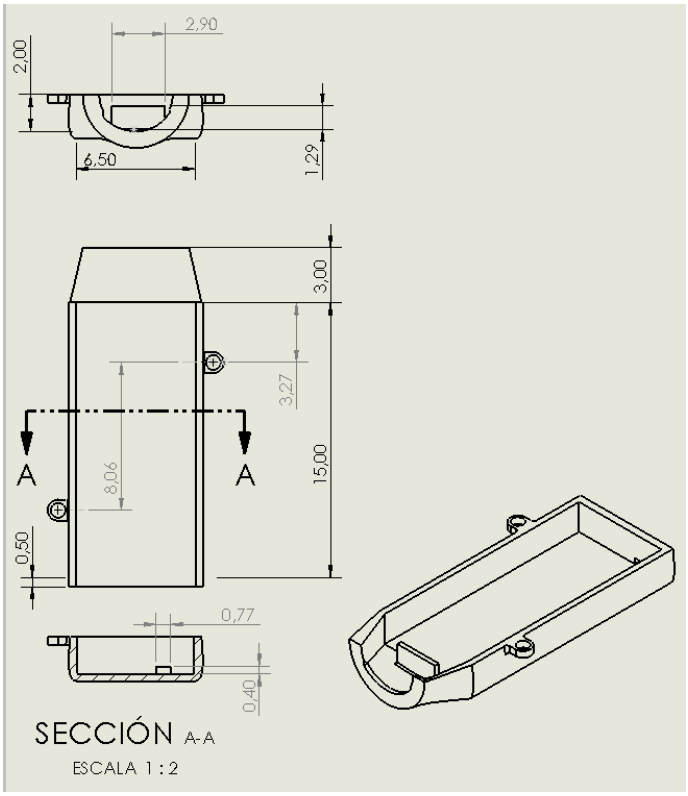


Figura 5.1.2-3 Miniatura del plànol tècnic de la part inferior de la carcassa

La impressió 3D es realitza a través del servei ***3DHUBS*** **[9]** i el resultat és el mostrat a la Figura següent:

Figura 5.1.2-4. Carcassa impresa junt amb interruptor i polsador per referenciar la mida.

5.2 Circuit d'alimentació

L'alimentació del circuit es basa en tres elements electrònics i un element de control que es detallen en aquesta secció.

Un mòdul microUSB rep energia de l'exterior a través d'un connector adequat i la dirigeix a la bateria per a que es pugui carregar. El corrent de càrrega és prop dels 0.7A. Seguidament, aquesta bateria dóna energia a 3.7V a un mòdul step-up que l'apujarà fins a 5V i la mantindrà regular per alimentar el circuit. Aquesta última connexió es fa a través d'un interruptor que connecta i desconnecta la bateria de la resta d'elements digitals (i de l'step-up). En el següent diagrama es mostren les connexions.

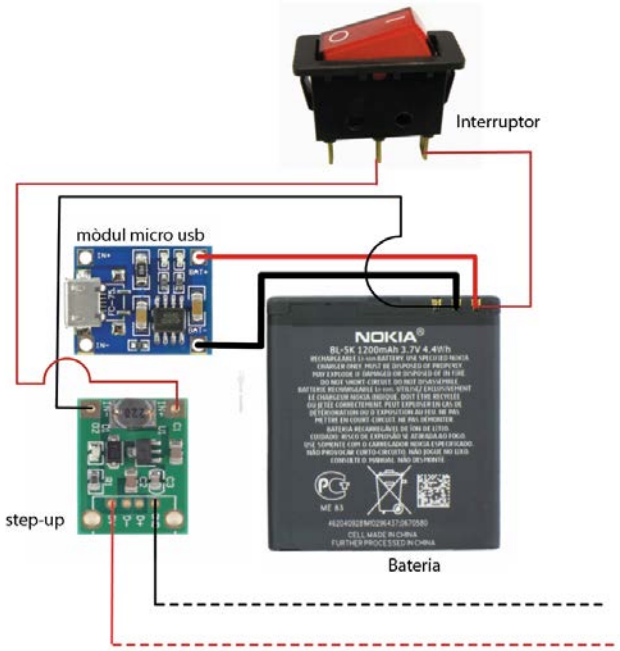


Figura 5.2-1. Diagrama d'elements i connexions de la part del circuit de potència

S'agrupen aquests elements –exceptuant l'interruptor- en una única placa PCB i certes connexions es solden i es fixen mentre que, d'altres, es prolonguen amb cables de diferents terminacions en funció de la seva utilització i comoditat a l'hora de muntar i desmuntar el circuit (Figura 5.2-2).

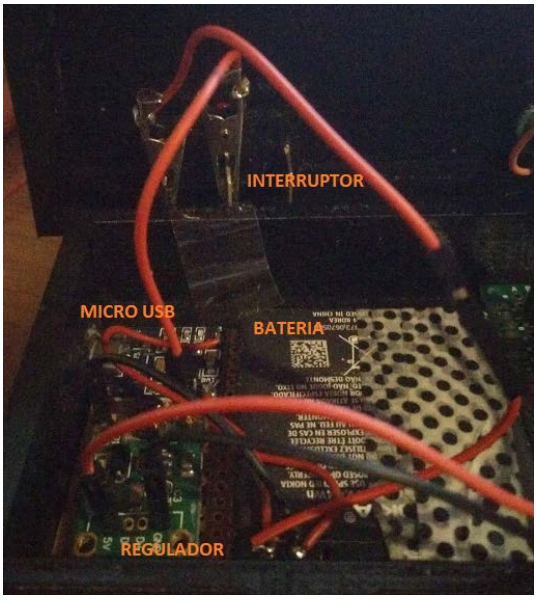


Figura 5.2-2. Elements i connexions de la part del circuit de potència muntats sobre una placa PCB, en la carcassa.

5.3 Circuit Digital

El circuit digital dissenyat en aquest projecte es configura com un conjunt de 4 elements: El PIC16F690, el sensor TCS34725, el Mòdul Bluetooth HC-05 i el Polsador per activar serveis d'interrupcions. La Figura 5.3-1 il·lustra l'interconnexionat del disseny final.

L'element central del circuit digital és el PIC16F690, on 7 pins són emprats pel funcionament del sistema tal i com es llista a continuació.

- El pin 1 i 20, per l'alimentació.
- El pin 13 i 11, per establir la comunicació amb el sensor.
- El pin 12 i 10, per establir la comunicació amb el Bluetooth.
- El Pin 17, per atendre les interrupcions.

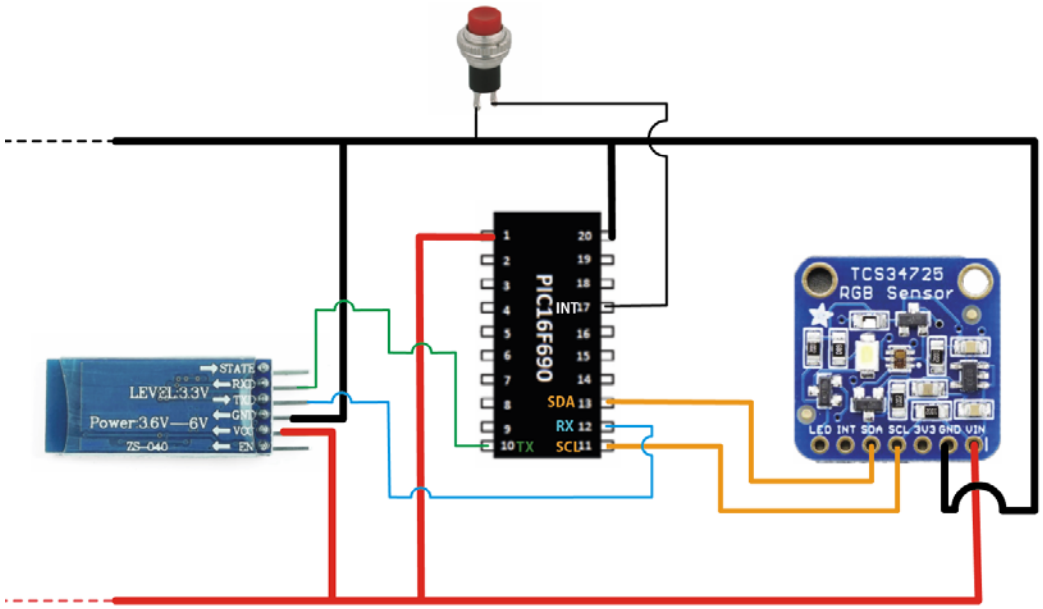


Figura 5.3-1. Diagrama d'elements i connexions de la part del circuit digital

Aquests elements –exceptuant l'interruptor i el sensor- es solden en una placa de proves de fibra de vidre. Per certes connexions, es solden connectors de tipus femella a la placa per tal de facilitar muntatge i desmuntatge del circuit (p.e. en les entrades SDA i SCL del PIC) . La següent Figura il·lustra part de l'aspecte final de la placa.

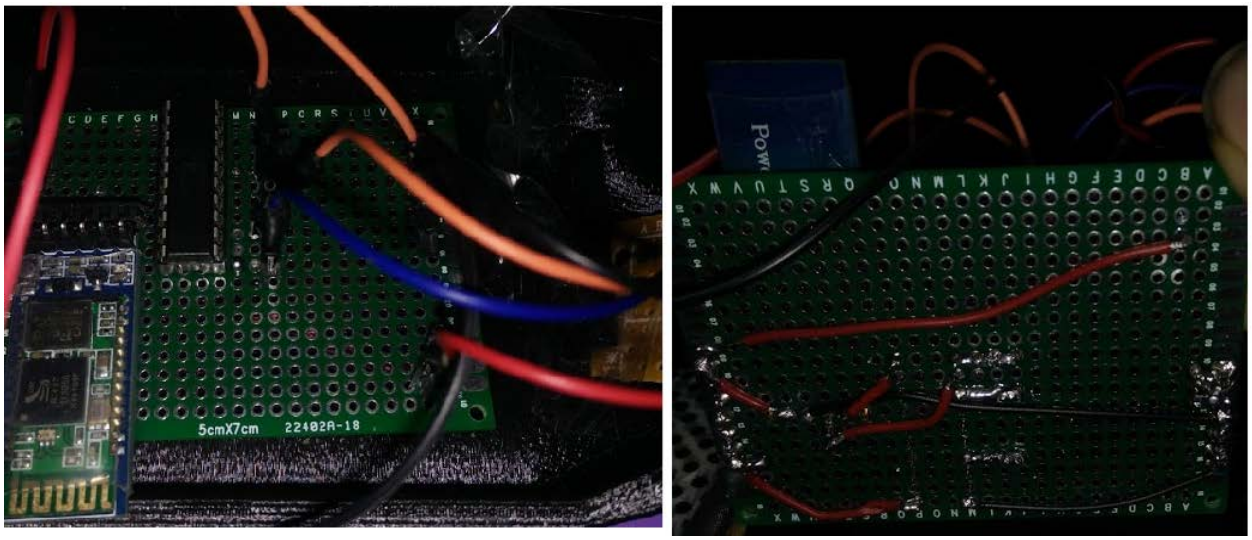


Figura 5.3-2. Anvers i revers de la placa, després de soldar-li els components electrònics.

5.3 Arquitectura completa

Es disposa i es fixen tots els elements electrònics i mecànics sobre la carcassa amb les següents connexions:

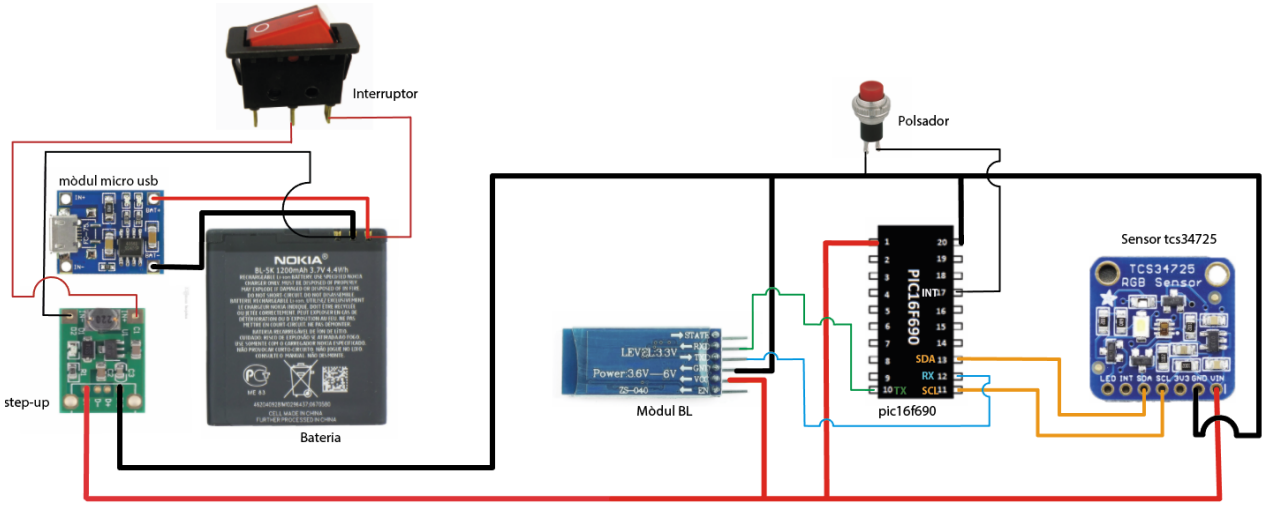


Figura 5.3-1. Diagrama de totes les connexions del circuit

S'asseguren la major part d'elements sobre la carcassa amb el següent resultat:

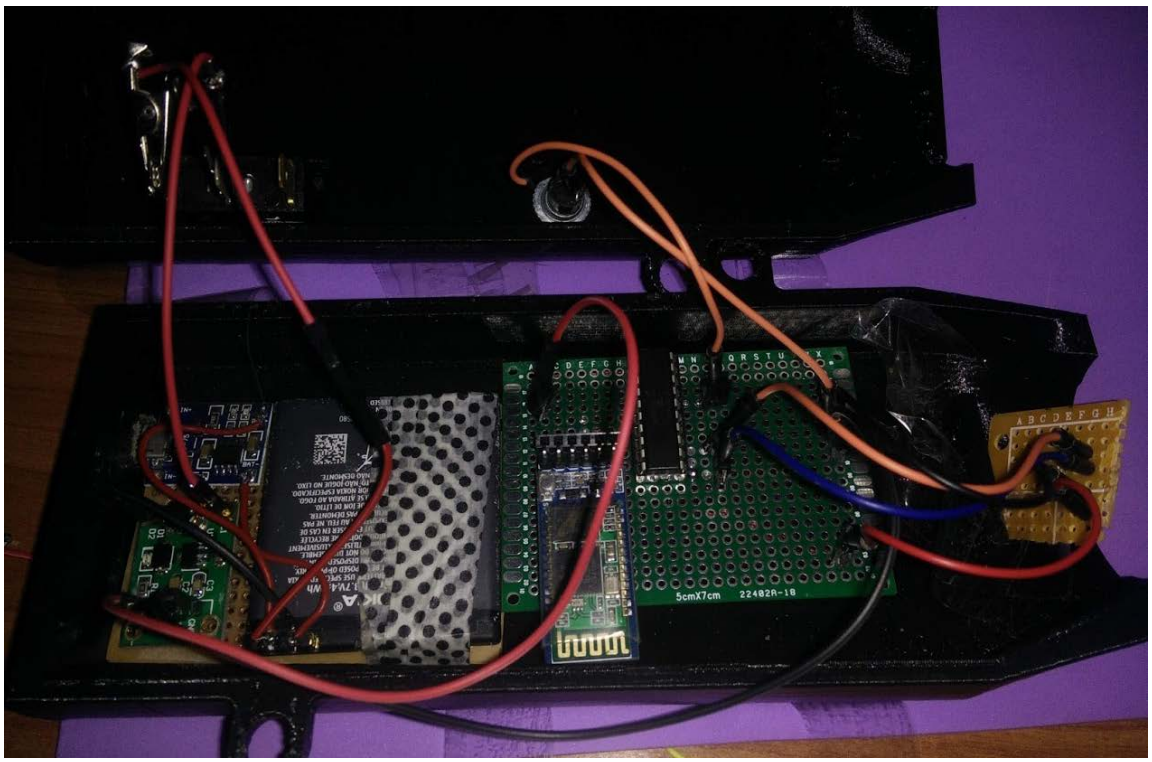


Figura 5.3-2. Arquitectura completa del dispositiu portàtil

6. Programació del dispositiu portàtil

El dispositiu portàtil té només un element per ser programat: El microcontrolador. S'utilitza un IDE per desenvolupar el codi de la programació, concretament el MPLAB **[10].** Per fer efectiva la programació, s'utilitza un programador pel microcontrolador. Això és, concretament, el PICKit 3.5. En aquest capítol es detallen les característiques de la part del projecte relacionada amb la programació del microcontrolador PIC.

6.1. Software MPLAB i programador PICKit

El MPLAB és un programa gratuït tipus IDE, és a dir, un Entorn de Desenvolupament Integrat (de l'anglès *Integrated Development Environtment*), una aplicació informàtica –per Windows, en aquest cas- que proporciona serveis i eines per facilitar el desenvolupament o la programació d'un software o hardware. En aquest cas, està orientat a desenvolupar programació de diferents dispositius de la marca MicroChip, com el PIC16F690 usat en aquest projecte, a través de codi escrit en llenguatge C. El PICKit és una interfície física electrònica que permet carregar la programació escrita en el MPLAB a la memòria ROM del PIC16F690.

Per a fer-ho, es connectarà el PICKit **[11]** a l'ordinador a través d'un port USB i també al Microcontrolador, a partir d'uns pins especialment dedicats a la programació com es mostra a la següent figura..

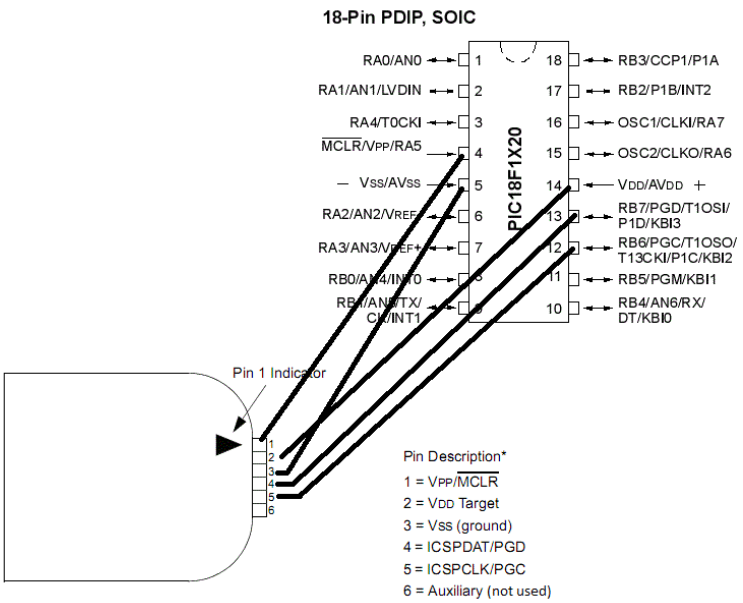


Figura 6.1.-1 Exemple de connexió entre un programador PICKit i un PIC de la marca MicroChip

6.2. Generalitats funcionals

El microcontrolador ha de complir les següents tasques:

- Establir comunicacions amb el sensor.
- Establir comunicacions amb el mòdul Bluetooth.
- Atendre l'usuari del dispositiu portàtil i, quan aquest premi un polsador, identificar el color de la llum que incideix sobre el sensor. Per això, cal:
    - Atendre interrupcions generades pel polsador.
    - Sol·licitar al sensor el valor de les components RGBC incidents sobre ell.
    - Adequar la informació rebuda pel sensor.
- Transmetre la informació del sensor, un cop adequada, al mòdul Bluetooth.

Per tant, el Microcontrolador ha d'estar generalment en un estat de bucle (de repòs) després d'haver-se configurat i sols actiu en el moment d'atendre la interrupció del polsador, enregistrant el color.

A la Figura 6.3-1 es mostra el diagrama de flux de les accions generals realitzades pel programa del microcontrolador PIC del projecte.

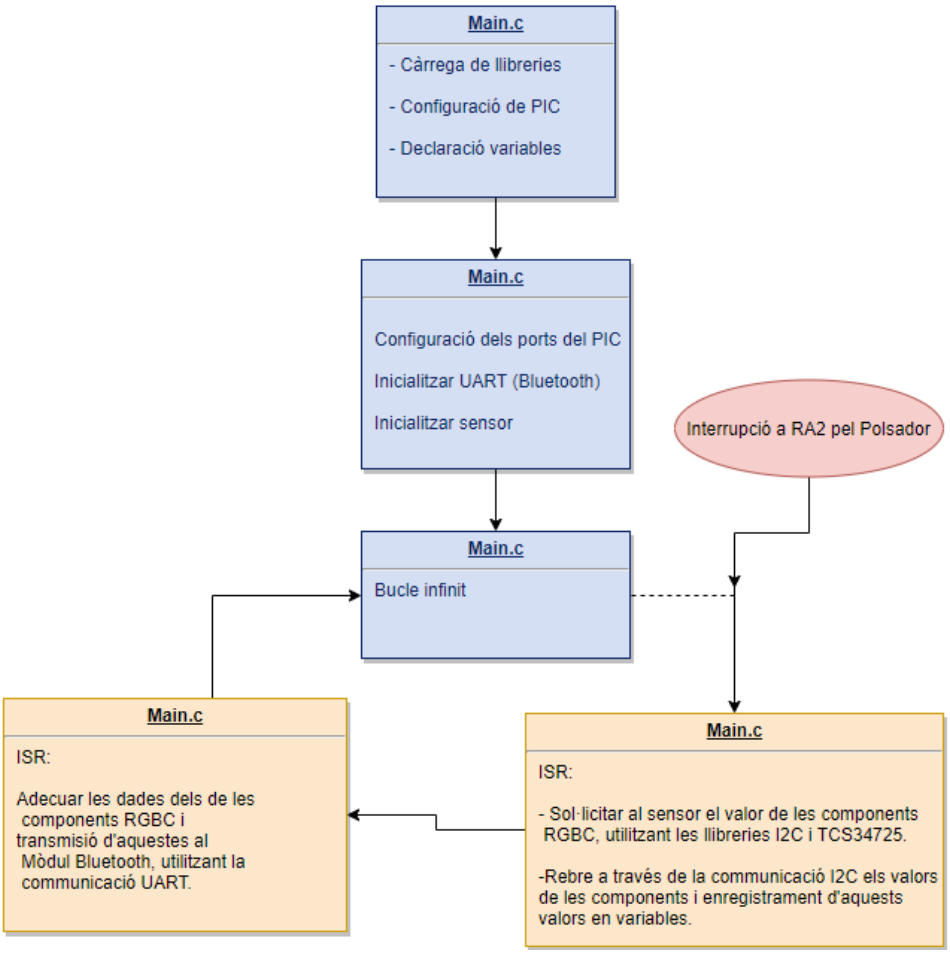


Figura 6.3-1 Diagrama de flux d'accions generals del programa enregistrat en el PIC16F690

6.3. Llibreries

Les llibreries són fitxers de suport que faciliten l'ordenació de les diverses funcions i línies de codi executades per un programa, que normalment estan dedicades a realitzar una tasca o conjunt de tasques que tenen contingut en comú.

Pel PIC16F690 i la funcionalitat requerida en aquest projecte s'utilitzen diverses llibreries:

- **UART.h**: Llibreria que agrupa un seguit de funcions que permeten la inicialització i la comunicació efectiva entre el microcontrolador i el Mòdul Bluetooth. És una llibreria àmpliament utilitzada i fàcilment descarregable de forma gratuïta.

- **I2C.h**: Llibreria que agrupa un seguit de funcions que permeten la comunicació efectiva dins el marc del protocol I2C. Concretament, defineix les diferents accions que pot dur a terme el protocol I2C. És una llibreria àmpliament utilitzada i fàcilment

descarregable de forma gratuïta.

- **TCS43725.h i TCS43725.c:** Llibreria que permet establir la comunicació de forma còmoda entre el sensor de color i el PIC16F690. Aquesta llibreria és inèdita i per tant s'ha creat especialment per aquest projecte.

Sobre les dues primeres llibreries no es fa cap comentari ja que són extensament utilitzades, tot i que es poden consultar íntegrament en un dipòsit accessible a través del següent codi QR, així com la tercera llibreria. Addicionalment, aquesta tercera llibreria també és consultable en l'annex de la memòria.

6.3.1. TCS43725.h

Un arxiu tipus .h agrupa un seguit de declaracions, ja siguin de funcions o de valors de variables.

En aquest cas, es declaren les adreces de diferents registres útils per configurar el sensor com es mostra parcialment a la *Figura 6.3.1-1*. Particularment, entre d'altres, es declaren les adreces on s'emmagatzema el temps d'integració del sensor, el guany del sensor, el ID assignat al sensor pel fabricant, les components RGBC enregistrades pel sensor i les adreces principals '*Slave Adress*' del sensor -bàsiques per poder comunicar-se-. Aquests valors es troben disponibles en el datasheet del fabricant **[4].**

També es declaren les funcions d'inicialització del sensor i les funcions per poder-se comunicar amb el sensor a través del protocol I2C, que són detallades en el punt 6.3.2.

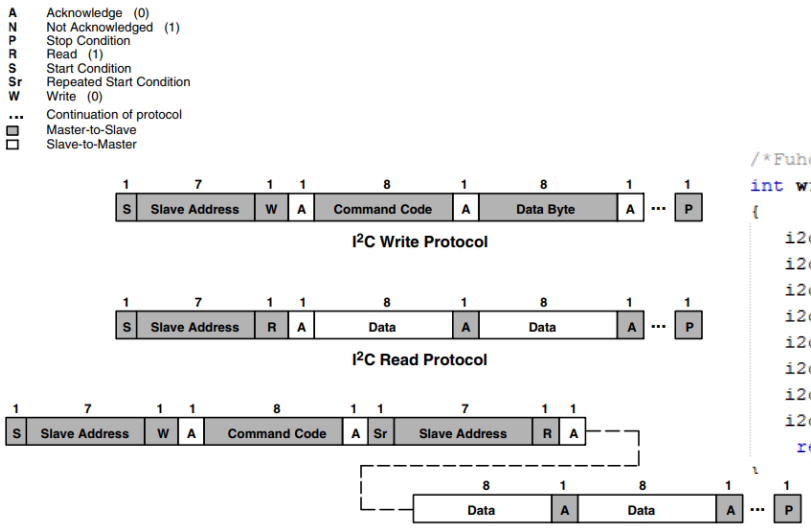
```
#define TCS34725_CDATAL          0x14    /* Clear channel data */
#define TCS34725_CDATAH          0x15
#define TCS34725_RDATAL          0x16    /* Red channel data */
#define TCS34725_RDATAH          0x17
#define TCS34725_GDATAL          0x18    /* Green channel data */
#define TCS34725_GDATAH          0x19
#define TCS34725_BDATAL          0x1A    /* Blue channel data */
#define TCS34725_BDATAH          0x1B
bool initialize();
int wireWriteDataByte(unsigned char , unsigned char );
unsigned char wireReadDataByte(unsigned char);
```

Figura 6.3.1-1 Extracte del codi de la llibreria TCS34725.h. Es veu en la figura la declaració de les adreces (en hexadecimal) on es registren les components RGBC del sensor. També 3 funcions són declarades.

6.3.2. TCS43725.c

En aquest fitxer de la llibreria es determina el seguit d'accions que el microcontrolador ha d'executar per fer la comunicació amb el sensor. Com s'ha esmentat en el punt 4.2.2, el protocol I2C requereix un format concret per funcionar correctament.

El protocol d'escriptura (write) consisteix en la seqüència d'accions de la següent manera (Figura 6.3.2-1): 1) Start. 2) Adreça del sensor en mode W (escriptura). 3) Acknowledge 4) Byte de comanda (registre a ser sobreescrit) 5) Acknowledge 6) Dada (Byte) a escriure en el registre 7) Acknowledge 8) Aturada del protocol




```
/*Fuhció per escriure un byte determinat al registre determinat*/
int wireWriteDataByte(unsigned char reg, unsigned char val)
{
   i2c_start();
   i2c_out_byte(TCS34725_ADDRESS_W);
   i2c_ack();
   i2c_out_byte(reg | TCS34725_COMMAND_BIT);
   i2c_ack();
   i2c_out_byte(val);
   i2c_ack();
   i2c_stop();
    return 1;
```

Figura 6.3.2-1. Protocol I2C. Comparació entre el diagrama teòric per escriure en un registre i la funció programada per escriure en un registre.

Es completa també la funció per llegir un registre determinat i també la funció per inicialitzar

el sensor (*Figura 6.3.2-3*). Aquesta última sols comprova si l'identificador del sensor és correcte (si el valor retornat pel sensor coincideix amb l'especificat pel fabricant).

```
/*Llegir un byte del registre que s'ordeni*/
unsigned char wireReadDataByte(unsigned char reg)
{
    char val = 0;
    i2c_start();
    i2c_out_byte(TCS34725_ADDRESS_W);
    i2c_ack();
    i2c_out_byte(reg | TCS34725_COMMAND_BIT);
    i2c_ack();
    i2c_start();
    i2c_out_byte(TCS34725_ADDRESS_R);
    i2c_ack();
    val = i2c_in_byte();
    i2c_nack();
    i2c_stop();
    return(val);

}
```

Figura 6.3.2-2. Funció programada per llegir un byte d'un determinat registre.

```
/*Es comprova si el sensor es communica correctament:
 Si el registre que emmagatzema la ID del sensor és diferent de 0x44 (el valor
 que el fabricant determina, torna False)
 */
bool initialize()
{
    unsigned char id=0;
    id= wireReadDataByte(TCS34725_ID);
    if(id!= 0x44) {
        return false;
    }
return true;
}
```

Figura 6.3.2-3. Funció per inicialitzar el sensor TCS utilitzat.

6.4. Programa principal

El programa principal –main.C- conté el codi que executarà primerament el microcontrolador i del qual es desprenen totes les crides a altres arxius i biblioteques. Així, el que cal fer és, primerament, carregar la resta de llibreries a utilitzar, declarar variables i configurar el microcontrolador com es llista a continuació.

```
#include <pic16f690.h>
#include <xc.h>
#define _XTAL_FREQ 4000000
#include  <stdio.h>
#include  <stdbool.h>
#include "uart.h"
#include "TCS34725.h"
```

Figura 6.4-1. Càrrega i importació de diferents llibreries a usar pel programa.

Es desactiven també totes les opcions del microcontrolador exceptuant l'oscil·lador intern, ja que aquest actuarà com a rellotge per permetre el funcionament seqüencial del PIC. També es declaren variables.

```
#pragma config FOSC = INTRCIO
#pragma config WDTE = OFF
#pragma config PWRTE = OFF
#pragma config MCLRE = OFF
#pragma config CP = OFF
#pragma config CPD = OFF
#pragma config BOREN = OFF
#pragma config IESO = OFF
#pragma config FCMEN = OFF

//variables a usar més tard
char buf[16];
unsigned int cdat;
unsigned int id;
int t;
int LECTURA;
```

Figura 6.4-2. Configuració del microcontrolador i declaració variables.

Es configuren els ports fent una posada a 0 i es configuren els ports que actuen com a ports d'entrada (RA2 per atendre interrupcions). També es desactiva la possibilitat de treballar en analògic, en detriment de la lògica digital.

```
//tots els ports es posen a 0 inicialment
PORTA = 0;
PORTB = 0;
PORTC = 0;

TRISA = 0b00000100; //RA2 com a entrada (pendent d'interrupcions)
TRISB = 0b00000000; //tots els pins com a sortida
TRISC = 0b00000000; // tots els pins com a sortida

//com que es vol treballar digitalment, ANSEL i ANSELH a 0
ANSEL = 0X00; // digital
ANSELH = 0X00; // digital
```

Figura 6.4-3. Configuració de ports del Microcontrolador

Es configuren les opcions per atendre interrupcions. També s'habilita el pull-up intern que

protegeix i assegura el funcionament correcte de la interrupció, evitant possibles corrents elèctrics no desitjats. Això s'aconsegueix amb la connexió feble (resistència elèctrica elevada) al valor lògic 1 dels pins definits com a entrada, evitant voltatges indefinits en el cas de nodes flotants elèctricament.

```
INTF = 0; // Reset del flag de les interrupcions
INTE = 1; // Habilitació les interrupcions externes
PEIE = 1; //Habilitació de interrupcions de perifèrics
GIE = 1; // Habilitació interrupcions globals
// Habilitació de els pull-up interns, per no tenir problemes amb
// els possibles punts flotants quan es connecten les interrupcions externes
OPTION_REG = 0;
```

Figura 6.4-4. Configuració relativa a les interrupcions del microcontrolador.

Es configuren els pins que han de servir com a transmissor i receptor del microcontrolador per comunicar-se amb el Bluetooth, i s'inicialitza la comunicació fent ús de la llibreria UART.h, amb la funció UART_Init(*Baud-rate*) com es mostra a continuació.

```
//configuració BLUETOOTH
    TRISBbits.TRISB5=1;     // Habilitem el pin 5B com a entrada (RX)
    TRISBbits.TRISB7=0;      // Habilitem el pin 5B com a entrada (RX)
    UART_Init(9600);// inicialització de la comunicació UART amb el Bluetooth
```

Figura 6.4-5. Configuració relativa a la comunicació UART

Seguidament, s'inicialitza i es configura el sensor. Primerament, es comprova que es reconegui aquest i, en cas afirmatiu, s'habilita l'oscil·lador intern i també s'habilita el convertidor Analògic-Digital. Posteriorment, es defineix un temps d'integració de 240ms i un guany de 4 (valor predeterminat pel fabricant) que són valors suficients pels requeriments del projecte.

```
/*En cas d'identificar el sensor, el configurem*/
if(initialize) {

     wireWriteDataByte(TCS34725_ENABLE, TCS34725_ENABLE_PON);
          __delay_ms(10);
           wireWriteDataByte(TCS34725_ENABLE,  0b00000011);
           __delay_ms(10);

//inicialització dels paràmetres del sensor
wireWriteDataByte(TCS34725_ATIME, TCS34725_WT_240MS); // es defineix el temps
//d'integració com 240ms
    __delay_ms(10);
     wireWriteDataByte(TCS34725_CONTROL, TCS34725_GAIN_4X); //s'afegeix un
     //guany canònic al valor captat pel sensor (valor per defecte)
    }
```

Figura 6.4-6. Configuració del sensor i inicialització d'aquest

El servei d'interrupció (ISR) –activat al prémer un polsador que connecta el terra amb el pin RA2, és a dir, que connecta el pin al 0 digital-, primerament deshabilita la possibilitat d'atendre una altra interrupció fins acabar aquest servei i, després, declara variables per allotjar els valors de 8 bits que el sensor retorna quan se li sol·licita la informació dels components RGBC. Es creen dues variables per cada component, ja que el sensor és capaç de captar informació amb suficient resolució com per treballar amb dades de 16 bits, però la transmissió es fa per busos de 8 bits, per tant, es separa cada valor de component en 8 bits de menys pes i 8 bits de més pes. A continuació es mostra aquesta part del programa.

```
void interrupt isr()
{


int C_data_L, C_data_H;
int R_data_L, R_data_H;
int G_data_L, G_data_H;
int B_data_L, B_data_H;
  if (INTF) //si el flag de les interrupcions val 1
  {
     //es deshabiliten interrupcions momentàneament
    INTE = 0;
__delay_ms(20);
```

Figura 6.4-7. Declaració variables locals de la funció interrupció

Seguidament, es sol·licita al sensor per I2C els valors de les components RGBC i, tal com s'ha mencionat amb anterioritat, aquest procés triga, com a mínim 240 ms, degut a que s'ha seleccionat un temps d'integració igual a aquest valor.

```
//es sol·liciten els valors dels registres corresponents a les components RGBC
    C_data_L = wireReadDataByte(TCS34725_CDATAL);
    C_data_H = wireReadDataByte(TCS34725_CDATAH);
    R_data_L = wireReadDataByte(TCS34725_RDATAL);
    R_data_H = wireReadDataByte(TCS34725_RDATAH);
    G_data_L = wireReadDataByte(TCS34725_GDATAL);
    G_data_H = wireReadDataByte(TCS34725_GDATAH);
    B_data_L = wireReadDataByte(TCS34725_BDATAL);
    B_data_H = wireReadDataByte(TCS34725_BDATAH);
    //es deixa el microcontrolador en espera fins passat el temps d'intregració
```

Figura 6.4-8. Sol·licitació dels valors de les components

Seguidament, comença la transmissió de les dades obtingudes cap al mòdul Bluetooth. El bluetooth emetrà al dispositiu que tingui emparellat la mateixa seqüència de bits rebuda mitjançant tecnologia RFID. Es comença escrivint un '-' (45 en ASCII) com a inici del missatge. S'utilitza la funció de llibreria UART_Write(byte) per fer-ho.

```
//s'escriu un nombre arbitrari al BL (a l'smarphone) com a 'hand-shake'
  UART_Write(45);
  //s'obté el valor en 16 bits del registre de Clear
cdat=C_data_L+(C_data_H << 8);
```

Figura 6.4-9 En aquesta imatge es veu com s'obté un valor total cdat *(valor de la component Clear) en format 16 bits. El valor s'obté sumant el valor de menys pes d'una component amb el valor de més pes (corregut 8 posicions) de la mateixa component.*

Seguidament, s'envia el missatge al mòdul Bluetooth amb la informació referent a les components enregistrades. S'utilitza la funció de llibreria Uart_Write_Text(String) per enviar cadenes de text. El missatge, llavors, s'escriu com text i els valors de les components en 16 bits es transformen primerament en un String (una cadena de caràcters) a través d'un buffer declarat a l'inici del programa principal com es mostra a continuació.

```
    UART_Write_Text("Color:(");
  __delay_ms(3);
  sprintf(buf,"%d",cdat);
  UART_Write_Text(buf);
  __delay_ms(3);
  UART_Write_Text(",");
sprintf(buf,"%d",(R_data_L+(R_data_H << 8)));
  __delay_ms(3);
   UART_Write_Text(buf);
    __delay_ms(3);
  UART_Write_Text(",");
   __delay_ms(3);
  sprintf(buf,"%d",(G_data_L+(G_data_H << 8)));
   UART_Write_Text(buf);
    __delay_ms(3);
   UART_Write_Text(",");
    __delay_ms(3);
sprintf(buf,"%d",(B_data_L+(B_data_H << 8)));
   UART_Write_Text(buf);
    __delay_ms(3);
    UART_Write_Text(")");
```

Figura 6.4-10. Transmissió del missatge al mòdul Bluetooth, és a dir, a l'Smartphone

Finalment, el missatge queda amb el format **"-Color:(**Valor de clear*, *Valor de vermell*, *Valor de verd*, *Valor de blau**)"**. Un cop s'ha transmès el missatge, es finalitza el servei d'interrupcions i s'habilita altra vegada.

7. Backend de l'aplicació i núvol

El núvol és aquella xarxa on-line on s'emmagatzemen i/o executen un conjunt de serveis i arxius.
En el marc d'aquest treball es desitja emmagatzemar un registre de colors reconeguts per l'usuari així que, naturalment, s'haurà de disposar d'una **base de dades** que atengui a aquesta funcionalitat. Per manipular una base de dades en línia (afegir elements, catalogar elements…) cal un **servei en línia** que executi **funcions**. Aquestes funcions han de ser accessibles sols pels usuaris del dispositiu portàtil i que disposin d'una aplicació mòbil (que actua com a pont entre les funcions al núvol i el dispositiu físic construït). La connexió entre aplicació mòbil i les **funcions** es fa, normalment, a través d'una API.

API fa referència a l'expressió *Application Programming Interface* (la traducció és Interfície de Programació d'Aplicacions). El concepte fa referència als processos, les **funcions** i els **mètodes** que ofereix una determinada biblioteca de programació com a capa d'abstracció perquè sigui emprada per un altre programa informàtic. En aquest cas, la biblioteca de funcions al núvol ha de ser usada per l'aplicació mòbil desenvolupada amb Kotlin i Java.

Aquest sistema multiplataforma connectat es completa amb elements de seguretat –tokens-, que són objectes informàtics efímers que serveixen per validar (o rebutjar) connexions, atenent a un seguit d'indicacions sobre permisos tant en el frontend com en el núvol de l'aplicació.

Per desenvolupar aquest marc, es treballa amb Amazon Web Services (AWS), un conjunt de recursos i serveis destinats a allotjar i executar programes, bases de dades i fitxers en línia. Algunes empreses que basen el seu funcionament en AWS són, entre d'altres, *AirBnB, Netlflix, tinder o SoundCloud.* En la Figura 7-1 es presenten dades econòmiques referents als serveis en el núvol.

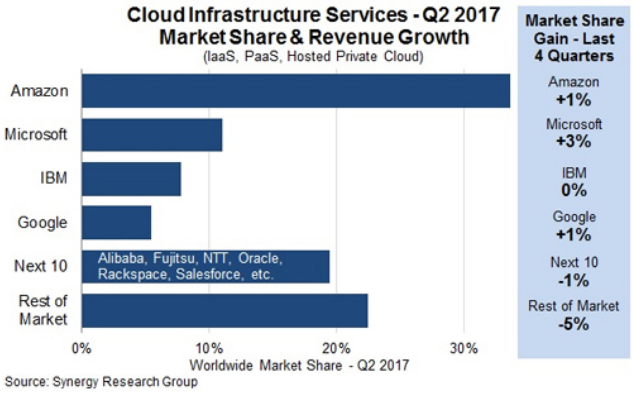


Figura 7-1.L'empresa que té més tràfic i més benefici pels seus serveis de Cloud no és Google ni Microsoft, sinó que és AWS i amb una diferència considerable, com mostren les dades de la figura.

Tal com s'ha esmentat anteriorment, el backend online de l'aplicació necessita:

- Una base de dades on poder emmagatzemar el registre de colors. En Aquest cas, s'utilitza una Base de dades dinàmica, a través del **servei Amazon DynamoDB.**
- Dues funcions per maniuplar la base de dades: Una per afegir colors registrats i una altra per recuperar de forma ordenada els registres. En aquest cas, s'utilitza el servei **AWS Lambda**. També cal un seguit de normes de seguretat per les funcions, a través **d'AWS IAM.**
- Un protocol de conexió entre l'Smartphone i les funcions. En aquest cas, una API a través de l' **Amazon API Gateway.**
- Un servei que atengui els permisos dels usuaris, en aquest cas, a través **d'Amazon Cognito.**

Aquest conjunt de recursos que ara es detallen, conformen el backend i núvol de l'aplicació mostrat en la següent figura..

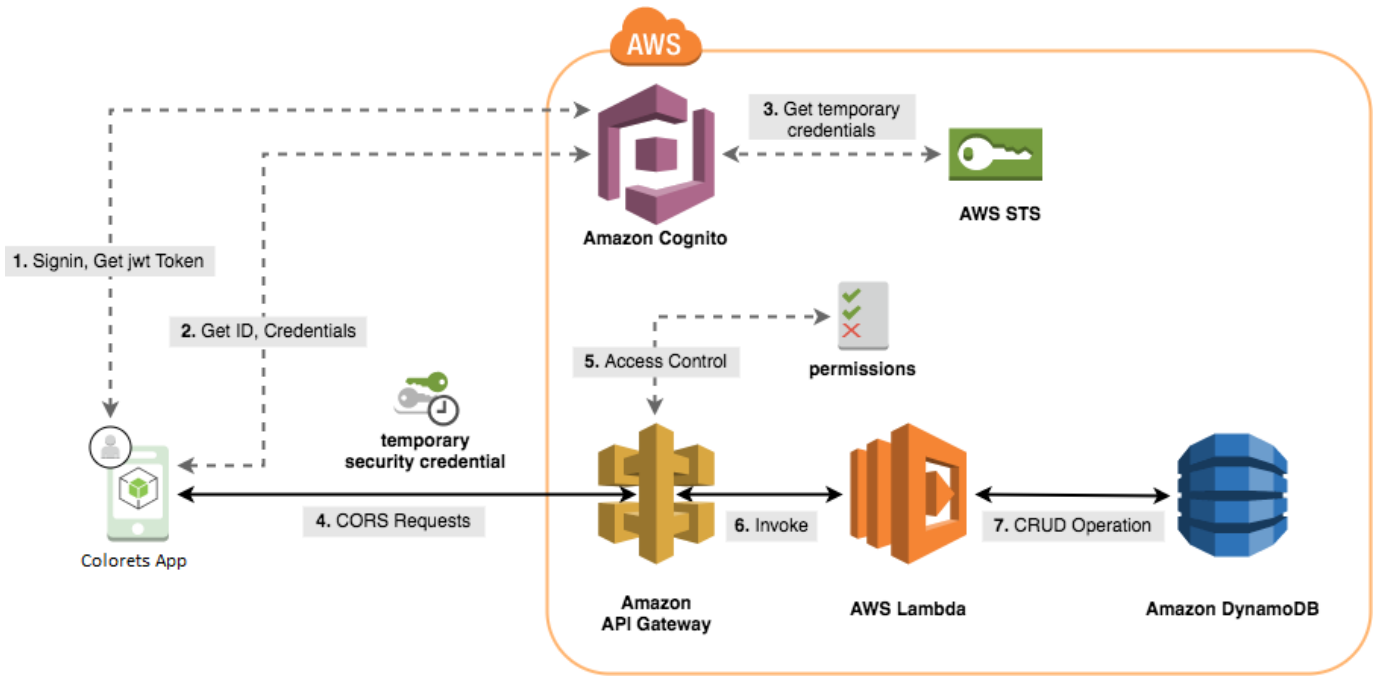


Figura 6.1-1 Diagrama de connexions entre els diferents elements del backend de l'aplicació

7.1. DynamoDB

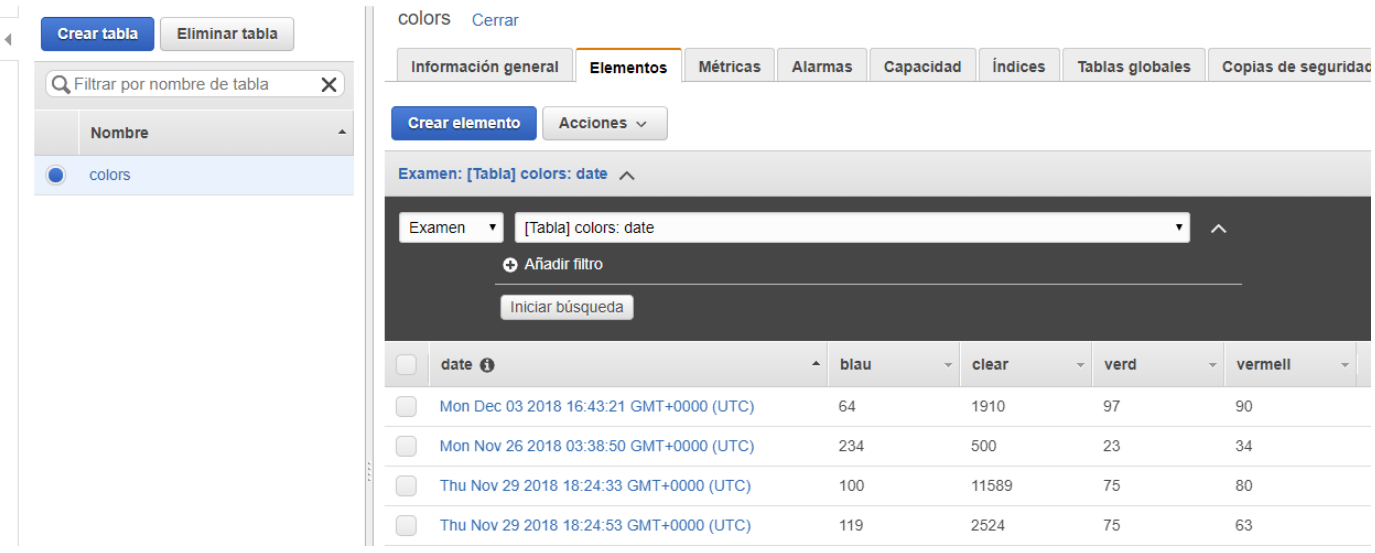
Una base de dades dinàmica (noSQL) com la DynamoDB té una estructura d'emmagatzematge d'elements del tipus clau-valor, fet que permet l'escalabilitat horitzontal més eficientment que una base de dades relacional i no necessita operar amb llenguatge SQL. Aquest tipus de base de dades -prenent el model relacional SQL com a referència- permet afegir columnes a certes files en concret i no a totes les files de la taula; permetent també fer determinades consultes no possibles amb un model relacional estàndard. Aquestes possibilitats han fet que l'ús d'aquestes bases de dades s'estengui notòriament per aplicacions en temps real i de bigData, permetent recuperar i agregar dades més econòmicament que en una base de dades tradicional.

L'estructura de les dades a emmagatzemar en el projecte és prou simple com per encabir-se en una base de dades tradicional, però en cas de que l'aplicació creixés i tingués molts usuaris i prestacions, és millor triar una base de dades d'aquest tipus, que permetria afegir noves columnes de dades i crear esquemes potencialment útils pel futur.

L'estructura a emmagatzemar és:

- Index principal: Marca ISO de la data en la que s'ha fet el registre del color (date).
- Index secundaris: Valor de Clear, Vermell, Verd i Blau en RGB.

En la Figura *6.1-1* veiem la interfície de l'AWS (DynamoDB) i diferents elements registrats.



6.1-1 Elements de la Base de dades

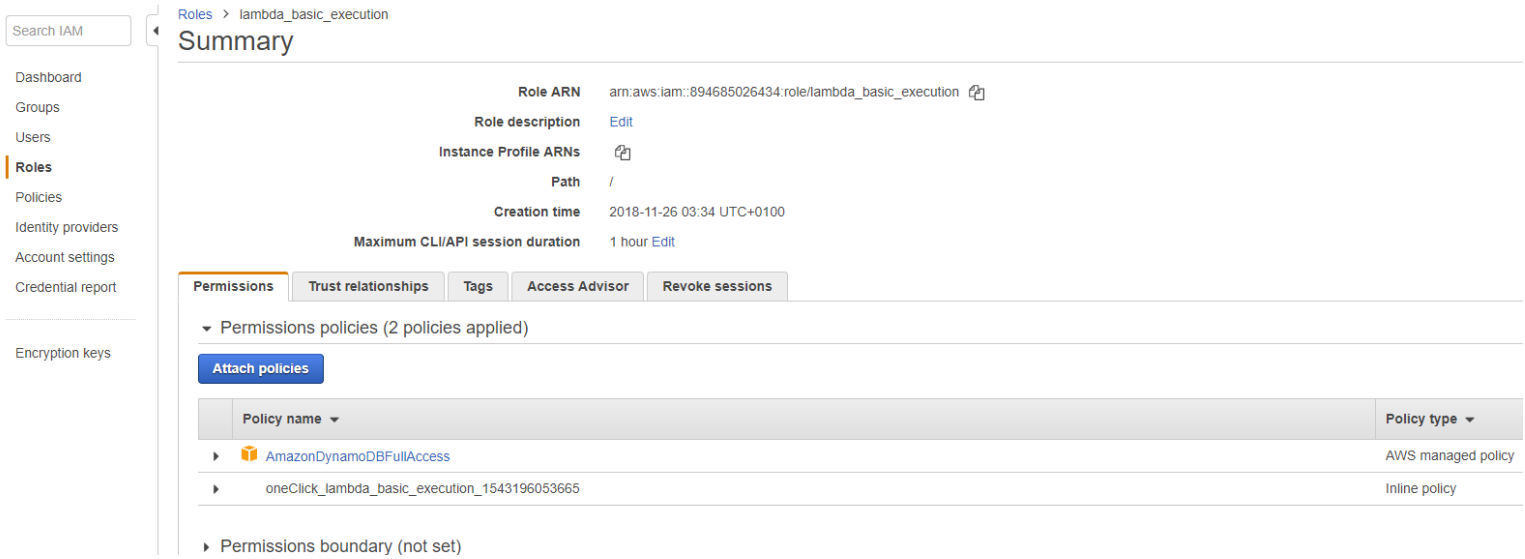
7.2. AWS Lambda

AWS Lambda és aquella part dels serveis de AWS que permet executar codi que respongui a **esdeveniments** sense aprovisionar ni administrar servidors. Això té l'avantatge de que només es paga pel temps informàtic que es consumeixi atès que no es cobra quan el codi no s'està executant i no requereix, com s'ha dit, d'un manteniment de servidor.

El propòsit de Lambda és el de fer més simple la construcció d'aplicacions *on-demand* que responguin a informació nova i a esdeveniments. Una instància Lamda respon en mil·lisegons a un esdeveniment. Permet executar codi en NODE.JS, Python, Java, Go i C#.

Abans d'esmentar breument la creació d'una funció Lambda i el codi usat, es crea un rol d'execució per la funció. Un rol d'execució serveix per delimitar a una funció AWS Lambda l'accés a diferents serveis, fent la funció més segura i robusta, evitant possibles càrregues econòmiques no desitjades per part de l'aplicació. En aquest projecte, sols interessa que la funció tingui accés a base de dades DynamoDB.

Es crea un rol a través de l'AWS IAM i s'anomena *Lambda_basic_execution,* se li afegeix capacitat per llegir, editar i omplir una base de dades DynamoDB, és a dir, se li afegeix *AmazonDynamoDBFullAcces.*



7.2-1 Rol d'execució per la funció AWS Lambda

Seguidament, es creen la funcions. Concretament, es crea *SetColor* per afegir un element

a la base de dades i *GetColors* per recuperar els elements de la base de dades. Ambdues es creen amb Node.JS per ser un llenguatge que coneix l'autor, tot i que podrien funcionar amb Python o qualsevol altre llenguatge disponible (Figures 7.2-2 i 7.2-3).

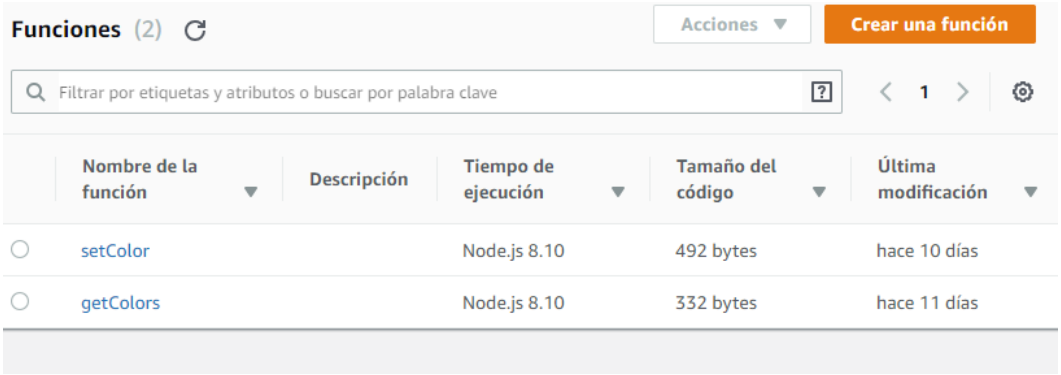
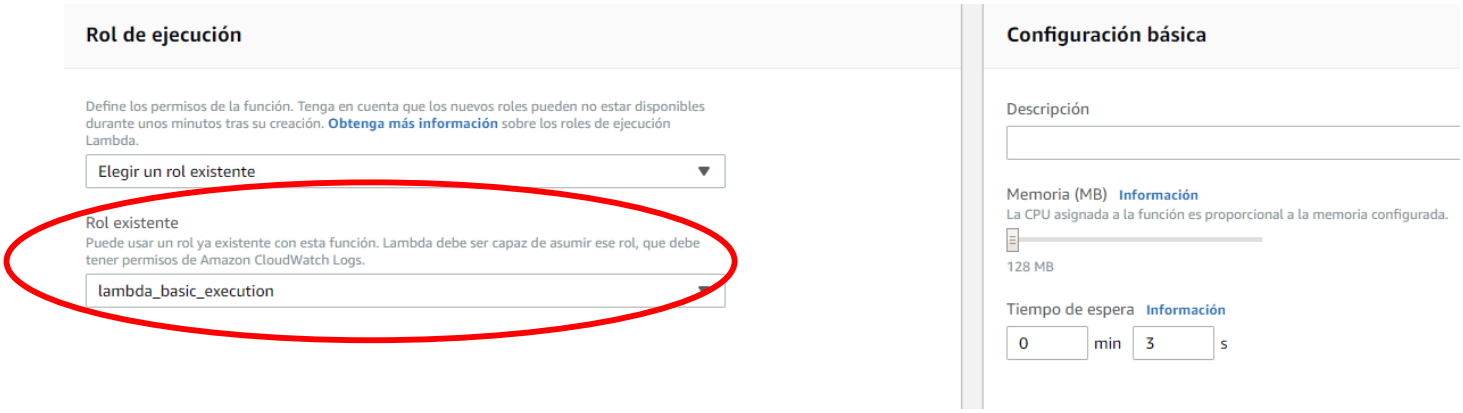


Figura 7.2-2 Funcions creades

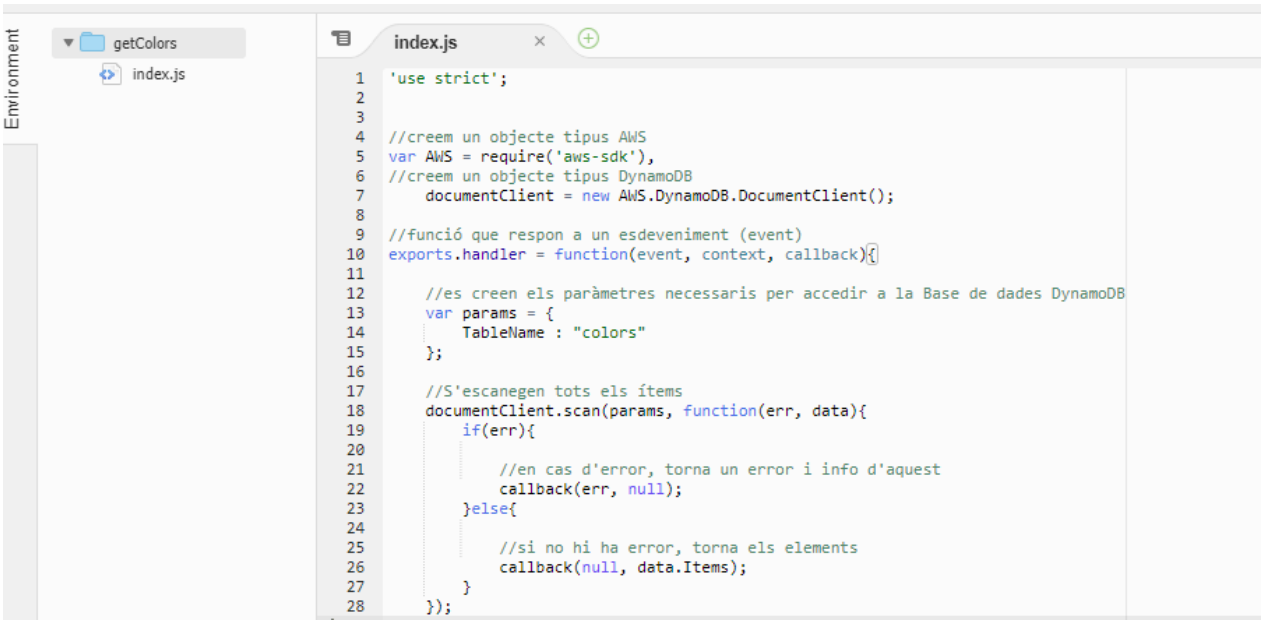


7.2-3 S'assigna en ambdues funcions el rol d'execució creat anteriorment

Les funcions esmentades poden consultar-se íntegrament a l'annex de la memòria o al següent dipòsit, a través del codi QR mostrat a continuació.

.

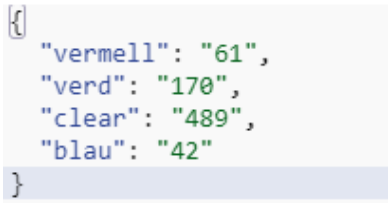
La funció getColors és la següent:



7.2-4 Funció getColors

Cal mencionar que l'esdeveniment que rep la funció (i al que respon) ha de ser un objecte tipus JSON. JSON (acrònim de JavaScript Object Notation, «notació d'objecte de JavaScript») és un format de text creat per a l'intercanvi de dades. JSON és un subconjunt de la notació literal d'objectes de JavaScript que, a dia d'avui, per la seva àmplia adopció com a alternativa a XML, es considera un format de llenguatge independent.

El format concret de l'objecte JSON ha de ser el següent (els valors numèrics són exemple):



7.2-4 Esquema JSON de l'esdeveniment

La funció setColor és la següent i depèn de la informació rebuda (esdeveniment *event*):

setColor
index.js

index.js

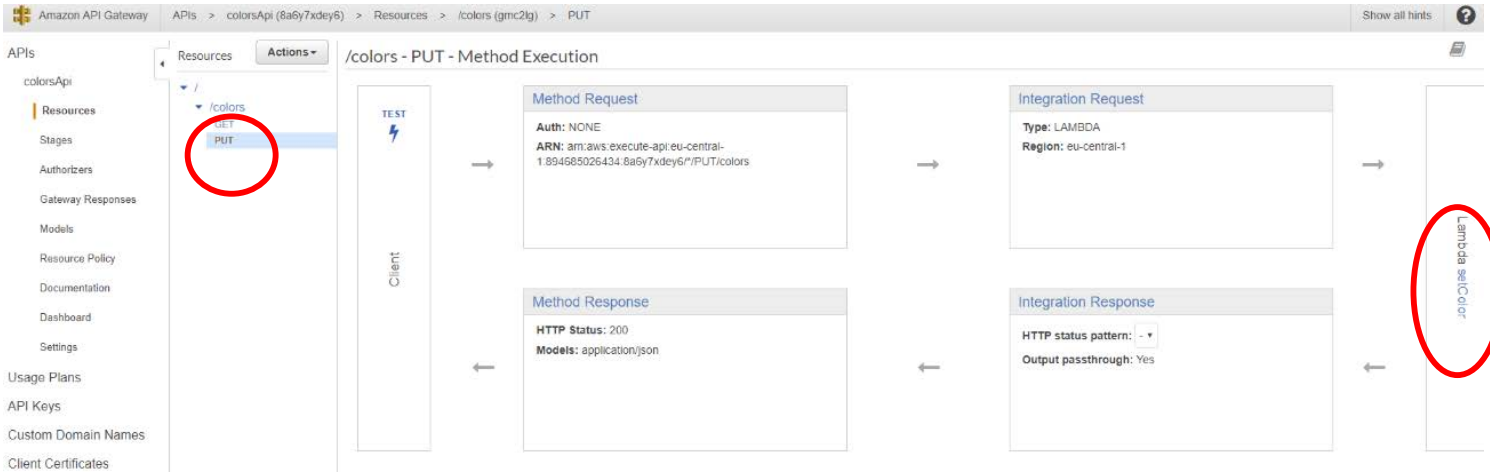
```
var AWS = require('aws-sdk');
exports.handler = (event, context, callback) => {

var ddb = new AWS.DynamoDB({});

//creació de l'objecte tipus DATA per enregistrar el moment del reconeixment del color
var data1= new Date();

//transformació a string
var data2= data1.toString();

//paràmetres que s'afegiran a la base de dades com a item. Veiem que els colors
//venen dictats per l'esdeveniment (EVENT)
var params = {


  Item: {
    'vermell' : {N: event.vermell},
    'blau': {N: event.blau},
    'verd' : {N: event.verd},
    'clear' : {N: event.clear},
    'date' : {S: data2} ,
  },
  TableName: 'colors'
};

// crida la base de dades i afegeix l'item
ddb.putItem(params, function(err, data) {
  if (err) {

     callback(null, err);
  } else {
    //en cas de que no hi hagi problema, torna un OK
    callback(null, 'succeess');
  }
});
```

7.2.-5 Funció setColor

7.3 API Gateway

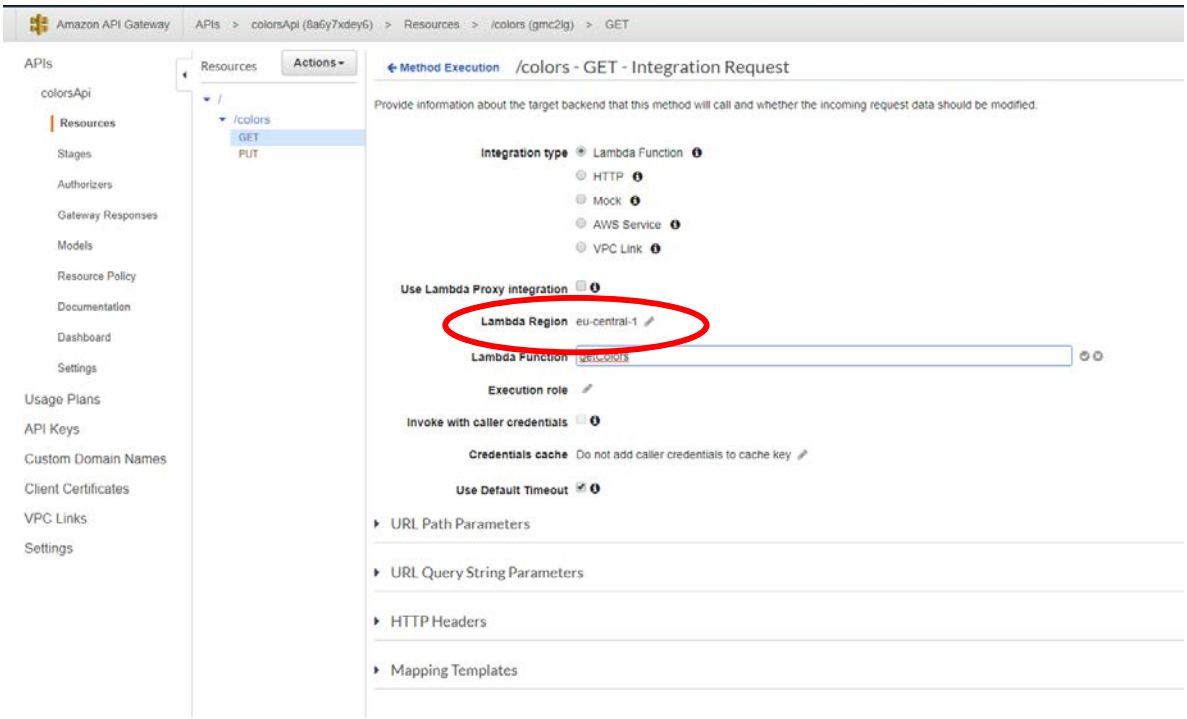
L'amazon API Gateway és un servei de AWS que permet als desenvolupadors crear, publicar, mantenir, monitoritzar i protegir una API a qualsevol escala, connectant així diversos serveis, programes, aplicacions i dispositius. Pel projecte es crea una API (***ColorsApi***) que serveixi com a interfície entre l'Smartphone i les funcions **AWS Lambda**. Es crea una API que, concretament**, tingui un recurs Anomenat *Colors*** i disposi de dos mètodes, ***Get i Put***, per agafar els elements de la base de dades o per col·locar-ne elements, respectivament. A cada mètode se li assigna una funció AWS Lambda de les creades anteriorment.

En la següent imatge veiem la interfície de l'API a l'AWS, concretament pel mètode PUT. La resposta és un objecte JSON amb el registre de tots els colors dins la base de dades. La integració (Integration request) de l'API es fa a través de la funció LAMBDA setColor, com es veu a la dreta de la següent imatge:



7.3-1 Vista general del mètode PUT de l'API, el recurs Colors

Més detalladament, aquí veiem desplegat la configuració de la integració del mètode (Integration request) GET. Obviament és del tipus funció Lambda.



7.3-2 Definició de la execució del mètode GET de l'API

7.4. Amazon Cognito

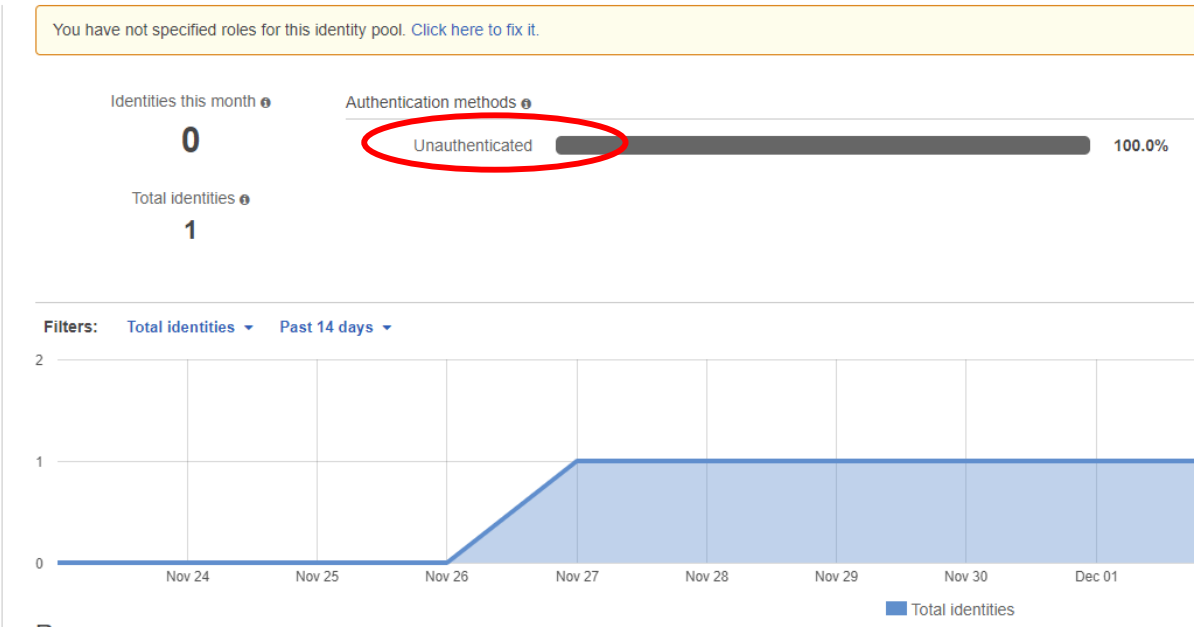
La creació d'usuaris és rellevant per la majoria d'aplicacions mòbils, ja sigui perquè pot resultar

interessant poder guardar preferències o dades d'un usuari i que es mantinguin quan l'usuari canvia de dispositiu. També pot ser rellevant dotar a certs usuaris de certs permisos, permeten accedir a certs serveis només a un grup d'usuaris. En l'aplicació desenvolupada en aquest projecte es desitja que l'usuari pugui emmagatzemar els seus colors registrats i els pugui recuperar des de qualsevol dispositiu sempre i quan accedeixi al seu compte personal.

Tanmateix, per ara, per aquest projecte, sols es crea **un espai d'identitats** (anomenat *eu-central-1:45d0c236-1a35-455f-beda-873d2d01edf3*), **el que correspon a totes aquelles persones que accedeixen a l'aplicació mòbil i no facin cap procés d'autenticació** –no tinguin cap compte d'usuari de l'aplicació- i, per tant no es considera, per ara, la creació de comptes personals.

A qualsevol **usuari no autenticat** a l'aplicació (és a dir, **que accedeixi a l'espai d'identitats** *eu-central-1:45d0c236-1a35-455f-beda-873d2d01edf3)* se'l dota de capacitat per accedir a la Api creada anteriorment, *ColorsApi*. És a dir, se'l proveirà amb un Token vàlid per poder accedir temporalment als serveis. Com es pot intuir, l'Amazon Cognito és un potent gestor d'identitats i usuaris.

En la següent imatge veiem l'Identity Pool (espai d'identitats) creat per l'aplicació, on es mostra que, de moment, sols existeix una identitat (òbviament, la de l'autor usant el seu Smartphone personal per desenvolupar el projecte) i que aquesta és Unauthenthicated (no autenticada, és a dir, que no pertany a cap compte individual).

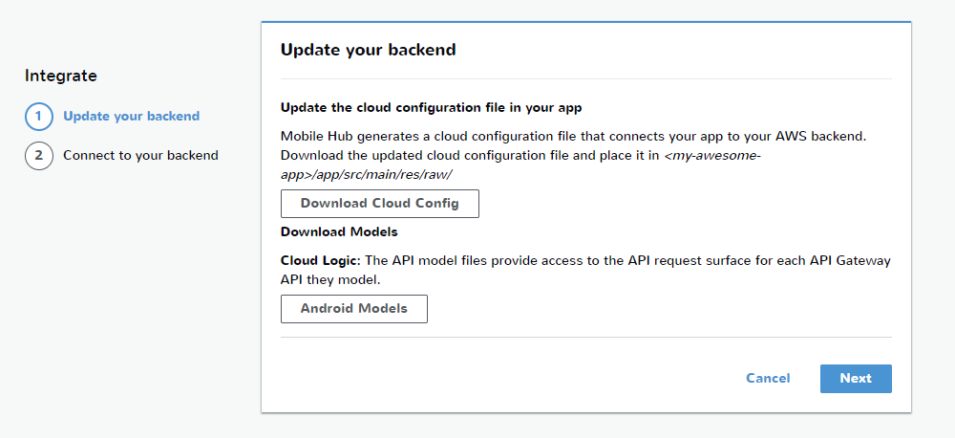


7.4-1 Vista general del'espai d'identitats de l'aplicació

7.5. De l'AWS a l'aplicació Mòbil

Per connectar l'API Gateway amb l'aplicació mòbil i poder accedir als serveis des de l'Smartphone, calen unes funcions i un protocol que els relacioni. AWS permet gestionar aquest codi (protocol, funcions) mitjançant el ***AWS Mobile Hub***, un servei que, entre altres coses, ofereix models i biblioteques per a descarregar i utilitzar en el desenvolupament de l'aplicació mòbil. Concretament, interessa poder disposar de codi que faciliti la crida de *l'API Gateway*, i que faciliti la transmissió i recepció de dades entre l'API Gateway i l'Smartphone.

Es crea un Hub Mobile per dispositius tipus Android i se li assigna la API anteriorment creada. Això permet descarregar els models de l'API (fitxers base que identifiquen i creen un marc per les comunicacions Android-API Gateway).



7.5-1 Pantalla de descàrrega dels models de l'API de l'aplicació en el AWS Mobile Hub. Tot i que es poden escriure manualment aquests models, s'opta per la creació automàtica que fa l'AWS Mobile Hub i la seva descàrrega.

8. Frontend de l’aplicació amb Android Studio

El frontend d’una aplicació són aquelles capes de software amb les que interactua l’usuari directament.

8.1. Generalitats

El frontend és, per aquest projecte, l’aplicació per Smartphone desenvolupada amb Java i Kotlin, a través del IDE Android Studio. La raó de desenvolupar una aplicació amb dos llenguatges alhora es fonamenta en que hi ha nombroses biblioteques i funcions natives disponibles per pròpies de Java, útil per desenvolupar les parts de codi més estandarditzades, com per exemple les parts de codi relacionades amb la comunicació Bluetooth. Kotlin, tanmateix, és un llenguatge més còmode, més modern i més eficient, tot i que no té tantes biblioteques disponibles.

En els apartats següents es detalla la interfície i parts del codi (tant sols les parts més crítiques, ja que el total de codi present supera les 1,200 línies) que conformen l’aplicació mòbil. En qualsevol cas, es pot consultar íntegrament en un depòsit seguint el següent codi QR o en l’annex (hi consten els arxius més rellevants).

8.1. Activitats i diagrama de flux

En els següents apartats es mostra la interfície gràfica de l’aplicació i el flux a través d’elles, explicant les funcionalitats de l’aplicació.

8.1.1. Activitat principal

L'activitat principal (Main Activity) és l'activitat que mostra l'aplicació a l'encendre-la.

Figura 8.1.1.-1 Activitat Principal

En aquesta activitat, hi ha dos línies de text (1 i 2 de la figura 8.1.1.-1): La primera mostra la informació que rep el dispositiu Bluetooth i la segona mostra l'estat de la connexió amb aquest. Un conjunt de botons estàtics permeten:

- **3:** Consultar la llista de dispositius emparellats amb l'Smartphone a través del mòdul Bluetooth, desplegant la llista **8** i mostrant aquests dispositius. En cas de tenir el Bluetooth desactivat, un *Toast* (notificació flotant) d'Android avisa de que aquest està desactivat.
- **4**: Començar a buscar nous dispositius Bluetooth no emparellats, llençant el gestor propi de Bluetooth de cada dispositiu Android.
- **5:** Aquest botó fa un canvi de pantalla i va a la biblioteca de colors registrats i guardats en el núvol.
- **6:** Aquest botó és informatiu. Canvia de color i canvia el text que hi té a sobre quan el dispositiu portàtil envia un missatge (amb les components RGBC reconegudes) a través del Bluetooth i l'smartphone el rep.
- **7:** Un cop es té en el botó **6** la informació sobre el color rebut, aquest botó número 7 permet connectar-se amb AWS i, mitjançant l'API, fer una crida per afegir el color a la biblioteca de colors.
- **8:** Element tipus llista ListView que es genera dinàmicament en funció dels dispositius emparellats que té l'Smartphone.

8.1.2. Activitat Biblioteca

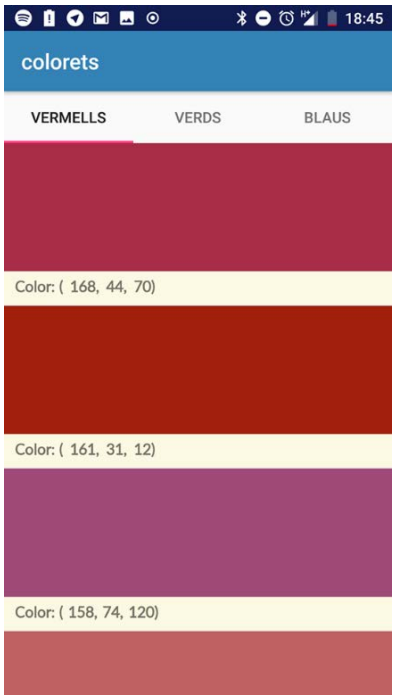


Figura 8.1.2-1. Activitat Biblioteca

La segona activitat, la biblioteca, es presenta amb 3 pestanyes (3 fragments d'activitat) on es separen els colors registrats a la biblioteca en funció del seu color predominant. Aquesta forma de presentar-ho sols vol explicitar que l'aplicació té capacitat per tractar les dades registrades al núvol i té capacitat per ordenar-les. Una altra forma d'ordenar-les podria ser separar els colors clars dels foscos, saturats o no saturats...

Per desplaçar-se entre pestanyes, es pot clicar cadascuna d'elles o bé fer *Swipe* (gest tàctil d'arrossegament horitzontal).

Per tornar enrere (a l'activitat principal) cal prémer el botó físic de l'smartphone conegut com *Back*.

8.1.3. Diagrama de flux

El en el següent diagrama de flux es presenta l'experiència de l'usuari usant l'aplicació, concretament des **d'obrir l'aplicació fins a connectar-se amb el Bluetooth**. L'usuari llença l'activitat principal i **sol·licita consultar els dispositius emparellats** prèviament (Pantalla 1 de la figura 8.1.3-1). Cal que el Bluetooth estigui connectat; altrament, un *toast* (notificació flotant) avisa que el Bluetooth no està habilitat en el dispositiu. Es genera dinàmicament una llista *listview* a la part inferior de la pantalla amb elements seleccionables (cadascun dels dispositius disponibles per a connectar-se). Al fer click sobre un s'estableix una connexió Bluetooth. En cas d'una connexió fallida amb el dispositiu seleccionat, s'expressa aquesta situació a través del text "Status". Si la connexió és correcta, s'expressa aquesta condició també a través del text "status", tal com es veu a la pantalla 3 de la figura 8.1.3-1.

Figura 8.1.3-1. Flux de les pantalles per la connexió Bluetooth

Seguidament, s'exposa el flux de pantalles que fa efectiu el registre dels colors: Quan s'està connectat al Bluetooth del dispositiu portàtil, aquest dispositiu portàtil és capaç de desencadenar canvis en la interfície gràfica (canvi del color del botó central) quan es reconeix un color. Des de que es prem el polsador i es genera la interrupció en el PIC fins que es manifesta el color registrat en el Smartphone, hi ha un lapse de temps de centenars de ms.

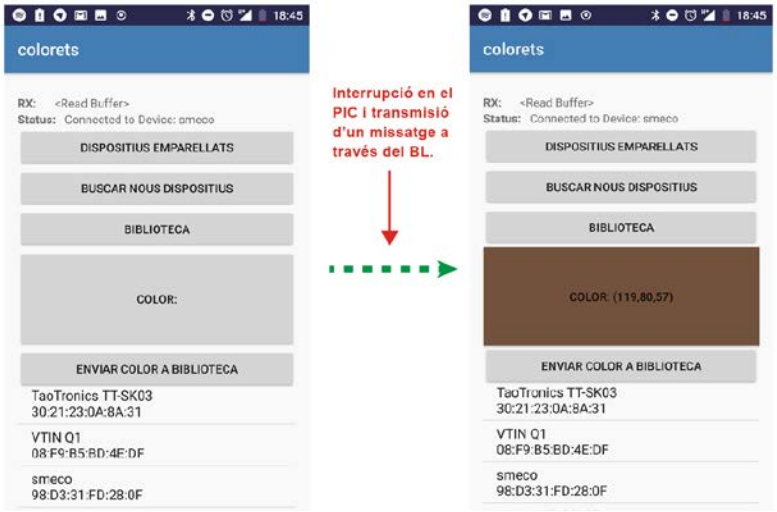


Figura 8.1.3-2. S'ha fet un reconeixement de color d'una capsa de cartró.

Un cop ja es té un color reconegut i mostrat en la pantalla principal, es pot optar per registrar-

lo en el núvol. Per això, cal prémer el botó “enviar color a biblioteca” (aquest botó no desencadena cap canvi en la interfície gràfica, però executa el servei que registra el color al núvol).

Per consultar la biblioteca de colors registrats, cal polsar el tercer botó i, aquest, desencadenarà un canvi d’activitat.

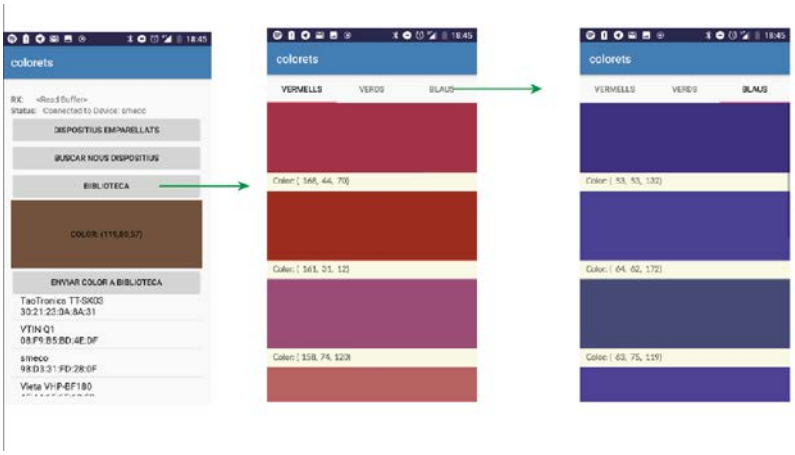


Figura 8.1.3-3 Flux per accedir a la biblioteca i moviment entre pestanyes dins aquesta.

8.2. Interfície gràfica

Per desenvolupar les interfícies gràfiques de l’aplicació s’ha utilitzat l’Android Studio per generar fitxers tipus XML enllaçats amb el codi font, una forma còmoda per produir elements gràfics mitjançant codi XML i un editor visual. Normalment s’associa un fitxer XML a cada pantalla –activitat de l’aplicació-. També s’associa un fitxer XML per cada element que es generi dinàmicament i que s’encabeix dins un conjunt d’elements (per exemple, per definir l’estil de cada element que conforma una llista que s’omple dinàmicament).

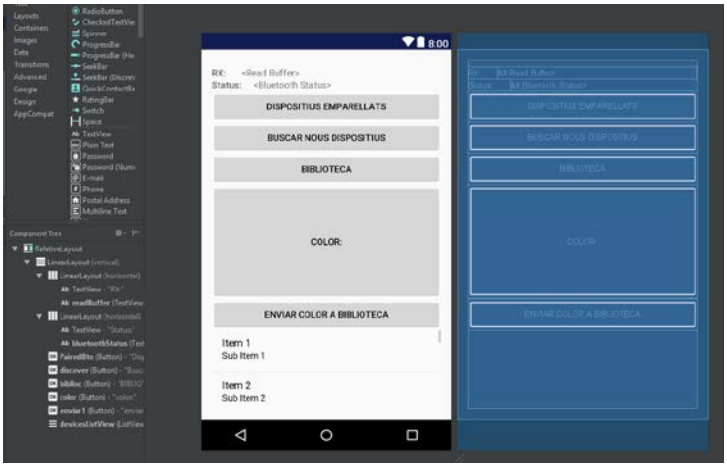


Figura 8.2-1 Editor visual de l'Android Studio, visualitzant l'arxiu activity_main.xml. Es visualitza l'activitat principal de l'aplicació. Un conjunt de textos i botons estàtics omplen la pantalla. A la part inferior, una llista d'elements (item 1, item 2...) conformen una vista tipus ListView amb elements que s'han de generar dinàmicament (per això, en la pantalla dreta de la imatge no apareix res, ja que sols hi apareixen elements estàtics com els botons).

La biblioteca de colors és una activitat (diferent a la principal) formada per 3 *Fragments* (les 3 pestanyes per cada color són un tipus d'activitat especial d'Android anomenada *fragment*). Cada pestanya és, simplement, una vista tipus *ListView* amb diferents elements (cada color examinat).

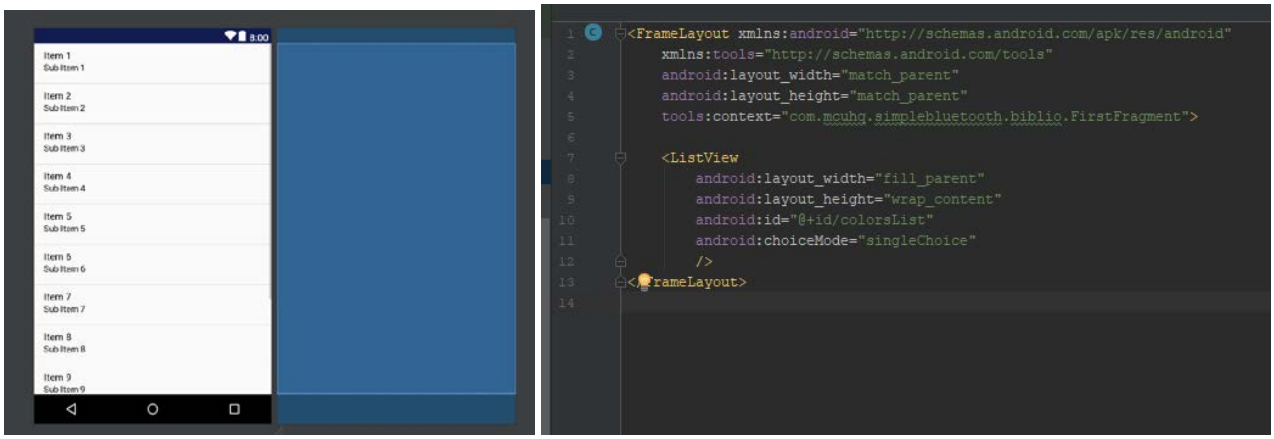


Figura 8.2-2. Fragment que es repetirà 3 vegades en la biblioteca, una vegada per cada color primari. Arxiu fragment_first.xml. A l'esquerra, l'editor visual d'Android. A la dreta, el codi XML que defineix el fragment i el ListView

El tipus d'element *item* que omple la llista ha de definir-se també en un fitxer XML. És útil aquest tipus de construcció perquè, més endavant, quan s'hagin de generar dinàmicament els elements de la llista a partir d'un Array d'elements recuperats de la base de dades al núvol,

es podrà tenir un model definit, referenciable i editable.

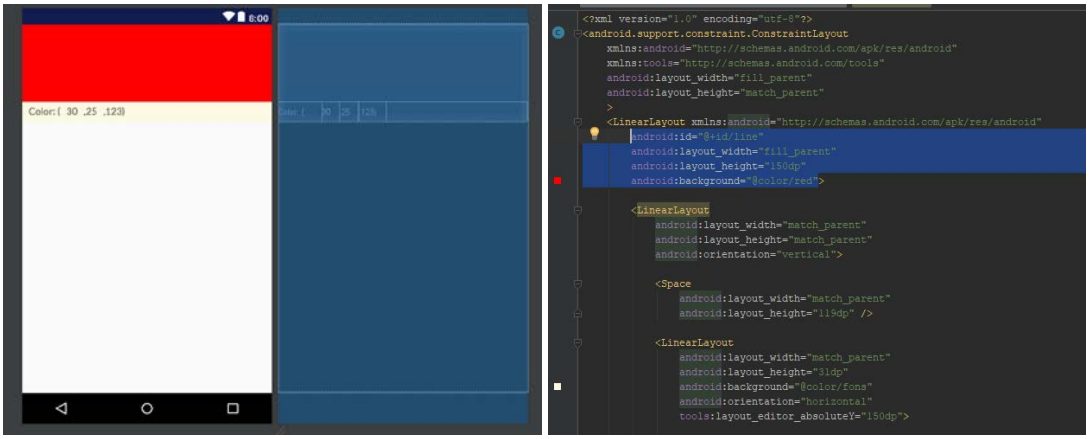


Figura 8.2-3. Prototip d'element de la llista de colors de la biblioteca. Arxiu item_color.xml. A l'esquerra, l'editor visual d'Android. A la dreta, el codi XML que defineix part de l'element.

A la figura 8.2-3 s'ha ressaltat una de les propietats de l'element construït, concretament, el **linearLayout** superior (el rectangle pintat [de color vermell per defecte] de la figura 8.2-3). Com es comprova, hi ha unes línies amb el codi:

```
android:id="@+id/line"
android:background="@color/red"
```

Es defineix un identificador *(+id*) per poder-se referir programàticament a aquesta propietat LinearLayout, i poder canviar atributs (tal com el *background color*) de forma dinàmica.

8.3. Arxius font

En aquest apartat es mostra part del codi usat per generar l'aplicació. Tot i que s'ha intentat detallar de manera entenedora, és recomanat estar familiaritzat amb la programació orientada a objectes i, més concretament, **amb la programació d'Android i AWS [12] per copsar els següents apartats amb profunditat.**

8.3.0. Desglossament d'arxius

Es disposa en el següent diagrama els arxius font de l'aplicació, amb un breu detall de cada fitxer rellevant present. Per ordre d'aparició (salvades excepcions):

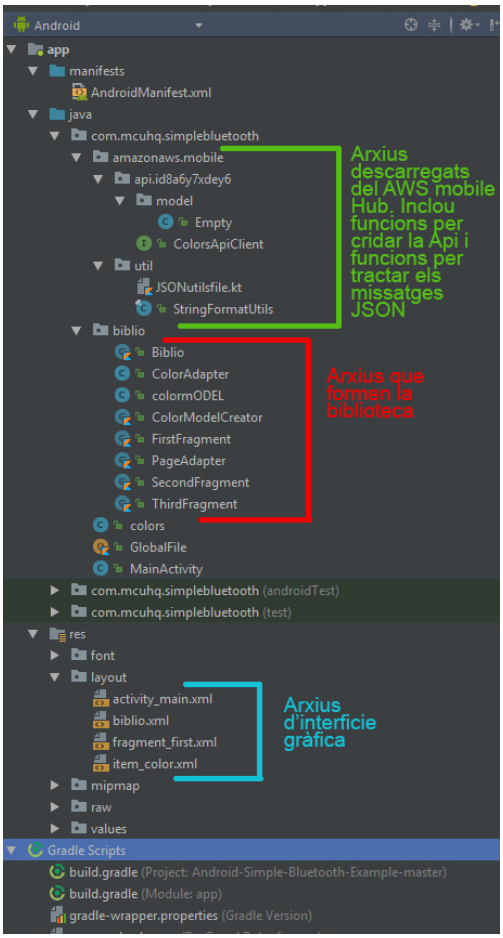


Figura 8.3-1. Desglossament d'arxius de l'aplicació en l'android Studio.

AndroidManifest.xml: Fitxer clau. Defineix nom de l'app, activitat principal, permisos de l'aplicació (internet, Bluetooth...).

Carpeta Amazonaws.mobile: Inclou les funcions necessàries per invocar l'API Gateway d'amazon a través de codi.

Carpeta Biblio: Conforma els arxius font de la biblioteca.

- **Biblio:** Activitat pantalla de la biblioteca. Té associat l'arxiu biblio.xml. Carrega les diverses activitats-pestanya. (Programat amb kotlin).
- **PageAdapter:** Classe que es dedica a col·locar i adaptar diferents fragments (FirstFragment, SecondFragment i ThirdFragment) en una activitat, concretament en la biblioteca. (kotlin).
- **FirstFragment,SecondFragment i ThirdFragment:** Cada pestanya de la biblioteca (una per color primari). Tenen associats l'arxiu fragment_first.xml. Allotgen elements visuals adaptats pel ColorAdapter en Listviews (kotlin).
- **colormODEL:** Classe-objecte simple que sols té 4 variables privades (valor R,G,B,C) i mètodes per accedir a les variables. (programat amb java).
- **ColorAdapter:** Classe que es dedica a generar elements visuals dinàmicament a partir d'un Array d'objectes *colormODEL*, poblant així els ListView (les llistes de colors) de les pestanyes. És a dir, adapta un Array. Cada element visual generat dinàmicament té associat l'arxiu item_color.xml (programat amb java).
- **ColorModelCreator:** Classe que transforma l'objecte JSON rebut del núvol (que conté tots els elements de la base de dades) en un Array de *colormODELS*. (kotlin)

Colors: Classe-objecte que transforma la cadena de text rebuda de la comunicació Bluetooth i que obté les components RGBC, guardant-les en variables. (Kotlin)

GlobalFile: Objecte tipus *Singleton* (objecte que pot ser usat i cridat des de qualsevol arxiu, no cal declarar-lo, ja que sols existeix una instància i és global) que conté com a variables diversos objectes relatius a l'ús de l'Amazon Web Services. També conté la funció-crida a l'Api Gateway **per recuperar tots els colors registrats** en el núvol (*Api**Get**CallWithPath)* i que un cop rebuts els allotjarà com a variable pròpia de l'objecte GlobalFile en format Array de

ColormODEL's, a través d'un ColorModelCreator. També conté **la funció-crida que afegeix** un element a la base de dades (*Api**Put**CallWithPath*). (kotlin)

MainActivity: Arxiu clau. Activitat-pantalla principal que instància el objecte GlobalFile, que fa crides a l'Api Gateway i que conté la part de codi relativa al Bluetooth. (java)

Carpeta res.layout: Conté els fitxers XML per definir interfícies gràfiques.

Carpeta res.raw: Conté el fitxer de configuració de l'AWS on s'especifica, entre altres coses, l'espai d'identitats de l'aplicació d'amazon Cognito '*eu-central-1:45d0c236-1a35-455f-beda-873d2d01edf3*', com es menciona en l'apartat 7.4.

Gradle Scripts: Conté arxius que configuren les llibreries utilitzades, les dependències i repositoris.

Build.gradle: Arxiu clau**.** Arxiu que defineix, entre altres coses, dependències a incloure en l'aplicació. S'han inclòs dependències d'Amazon Web Service i de Kotlin.

8.3.1. AWS i biblioteca. Codi.

En aquest apartat es detalla breument la inicialització i el funcionament de la connexió amb l'AWS en la capa de software frontend de l'aplicació, concretament es detalla la part de codi que descarrega elements del núvol. Per consultar el codi íntegrament, podeu seguir aquest QR mostrat a continuació que obre, directament, l'arxiu MainActivity.java de l'aplicació en un dipòsit o consultar l'Annex a aquesta memòria.

En la inicialització de l'aplicació (al MainActivity), s'inicialitza el AWSMobileClient, una classe que permet establir connexió amb el núvol. Més concretament, s'encarrega de proveir, entre altres coses, un credencial temporal –token- per poder accedir com a usuari no-autenticat (però identificat) als serveis d'AWS, com s'ha esmentat en l'apartat 7.4. Seguidament,

s'executa la funció ApiGetCallWithPath del GlobalFile, que obtindrà tots els elements registrats a la base de dades del núvol.

```
   AWSMobileClient.getInstance().initialize(this, new AWSStartupHandler()
{
            @Override
            public void onComplete(AWSStartupResult awsStartupResult) {
                Log.d("log", "AWSMobileClient is instantiated and you are
connected to AWS!");
            }
        }).execute();
GlobalFile.INSTANCE.apiGetCallWithPath("colors");
```

Figura 8.3.1- 1 Fragment de codi d'inicialització del MainActivity.

Aquest arxiu GlobalFile ja s'ha esmentat que permet cridar l'API Gateway, proveint un *Path* (una ruta) que es refereix al recurs de l'API que es vol cridar. En l'apartat 7.3 s'ha fet menció a que el recurs s'ha anomenat "*Colors*". Aquest missatge rebut al fer la crida arriba en format JSON i conté, en un sol string, tot els elements registrats a la base de dades. Es transforma l'String JSON en un Array d'objectes més fàcils de manipular (colormODEL's) utilitzant el ColorModelCreator. Per tant, la funció ApiGetCallWithpath ha de declarar un colorModelCreator.

```
object GlobalFile {

    public var list1=ArrayList<colormODEL>()
private var apiClient: ColorsApiClient? = null

   public fun apiGetCallWithPath(path: String) {
       var modelcreator: ColorModelCreator?=null;
            apiClient =
ApiClientFactory().credentialsProvider(AWSMobileClient.getInstance().c
redentialsProvider).build(ColorsApiClient::class.java)
```

Figura 8.3.1-2 Fragment del codi de definició del GlobalFile. Es declara una llista de colormODEL's al principi. La funció ApiGetcallWith té com a variables pròpies un colorModelCreator i un client API (apiClient) que obté credencials (a partir del AWSMobileClient instanciat en el MainActivity).

Més endavant de la funció, es modela una sol·licitud. Aquesta sol·licitud *request* ha d'incloure el Path i el mètode de sol·licitud, ja sigui un GET o un PUT, que són els disponibles com es menciona en l'apartat 7.3, en la creació dels mètodes de l'API.

```
val request = ApiRequest(ColorsApiClient::class.java.simpleName)
        .withPath(path)
        .withHttpMethod(HttpMethodName.GET)
        .withHeaders(headers)
        .withParameters(parameters)
```

Figura 8.3.1-3 Model de Sol·licitud al GlobalFile. El Path és una variable que s'explicita en la

crida de la funció.

La sol·licitud s'utilitza per **executar** la crida a l'API i s'obté la resposta. Es crea una variable *responseData* que té com a valor la resposta de la crida a l'API. El ColorModelCreator prèviament instanciat transforma la resposta (objecte String amb format de JSON) en una llista de colormODELs fent tractament de JSON i Strings, a través de la funció createColors, retornant i assignant la llista de colormODEL's a la variable list1 del fitxer GlobalFile .

```
val response = apiClient?.execute(request)
val responseContentStream = response?.getContent()
if (responseContentStream != null) {
    val responseData = IOUtils.toString(responseContentStream)
   list1=modelcreator.createColors(responseData);
```

Figura 8.3.1-4 Execució de la sol·licitud al servidor al GlobalFile.

La definició del colormODEL és la següent:

```
public class colormODEL  {
    private int r, g, b, c;
    public colormODEL(int r1, int g1, int b1, int c1) {
      r=r1; g=g1; b=b1; c=c1; }
    public int getr() {        return r;   }
    public int getb() {        return b;}
    public int getg() {        return g;    }
    public int getc() {        return c;    }
```

Figura 8.3.1-5 Definició de colormODEL

Un cop es disposa del GlobalFle amb la llista *list1* completa, quan es carregui l'activitat de la biblioteca podrà omplir-se amb els elements de la base de dades. A l'activitat de la biblioteca, es carreguen les 3 pestanyes amb els 3 fragments (FirstFragment, SecondFragment i ThirdFragment).

Detallem part del codi del fragment del FirstFragment (Color vermell): El fragment ha d'iterar l'array de colormODEL's list1 del GlobalFile, comprovar per cada element si el component vermell és més gran que el verd i el blau i, en cas afirmatiu, utilitzar un **ColorAdapter** *adapterC* (instanciat a l'inici del FirstFragment) per agregar un element visual a la listView del fragment, a través de la funció add, a la que se li passa cada element ***i*** tipus colormODEL).

```
for(i in GlobalFile.list1.orEmpty()) {
    if(i.getr()>= i.getb() && i.getr()>=i.getg()) {
            adapterC.add(i);}
}
```

Figura 8.3.1-5 Iteració en un Fragment sobre la llista de colors descarregats del núvol i tractats.

A continuació es detalla part del codi de la funció add del ColorAdatper. Com s'ha mencionat en l'apartat 8.2, cada element del ListView -definit a l'arxiu item_color.xml- disposa d'una propietat tipus LinearLayout, que té un Id i un color de fons (background).

```
android:id="@+id/line"
android:background="@color/red"
```

El ColorAdapter genera instancies (elements) que poblen el ListView i pot referenciar les propietats dels elements (a través de l'identificador Id) per modificar els seus atributs.

```
LinearLayout ly= (LinearLayout) convertView.findViewById(R.id.line);
ly.setBackgroundColor(Color.argb(255,colormodel.getr(),colormodel.getg(),
colormodel.getb()));
```

Figura 8.3.1-6 Modificació de la propietat LinearLayout pertanyent a un element generat dinàmicament tipus item_color.xml. El ColorAdapter modifica atributs de l'element

8.3.2. Connexió Bluetooth i AWS. Codi.

En Aquest apartat es detalla la part de codi d'inserció d'un element a la base de dades del núvol. També es mostra reduïdament el codi que permet la connexió Bluetooth amb el dispositiu portàtil i la comunicació que existeix. S'omet en aquest apartat la major part del codi, especialment d'aquelles parts basades en llibreries i classes d'Android, com els BluetoothSocket (rutes entre dispositius connectats), els ConnectedThread (fils d'execució que estan atents a entrades de missatges Bluetooth), o els Handler de missatges (funcions que reaccionen a l'entrada de missatges quan un fil d'execució ho requereix) **[13] [14] [15]**

Un cop es fa efectiva la connexió Bluetooth entre el dispositiu portàtil i l'Smartphone, aquest últim pot rebre i processar missatges transmesos amb Bluetooth. Per fer-ho, s'ha definit un Handler (una funció que respon a una interrupció de rebuda de missatge). El missatge enviat pel dispositiu portàtil ha de tenir el format **"-Color:(*Valor de clear*, *Valor de vermell*, *Valor de verd*, *Valor de blau*)"** com s'ha mencionat en l'apartat 6.4 . Aquesta funció comprova que el missatge sigui correcte i no hagin aparegut interferències. Primerament, comprova que el primer caràcter sigui un '-', que el setè sigui un '(" i que existeixi un tancament de parèntesi ')'. En cas de ser correcte, utilitza una instància anomenada **color** d'un objecte de la classe **Colors** per transformar el missatge rebut, a través de la funció **getcols**.

```
mHandler = new Handler(){
    public void handleMessage(android.os.Message msg){
        if(msg.what == MESSAGE_READ){
            String readMessage = null;

            try {

                readMessage = new String((byte[]) msg.obj, "UTF-8");
```

```
                if(readMessage.charAt(0)=='-')
                {
                    if(readMessage.charAt(7)=='(')
                    {
                        if(readMessage.indexOf(')')>0) {
                            color.getcols(readMessage);
```

Figura 8.3.2-1. Detall del codi per gestionar missatges Bluetooth, al MainActivity

L'objecte **color** és de la classe **colors** i té variables **b,r,g,c** on s'emmagatzemarà el color rebut i transformat a RGBC, a través de la funció **getcols** definida en la següent imatge.

```
public class colors {
public int b,r,g,c;
    public float c1,r1,g1,b1;
    public void getcols(String str) {
String sub;
int indxofp;
```

*Figura 8.3.2-2. Detall del codi de la classe **colors***

Aquesta funció **getcols** bàsicament tracta la cadena de caràcters obtenint una sub-cadena de caràcters amb la informació relativa als valors de les components RGBC i, seguidament, separa la informació i la ordena en un Array.

```
indxofp = str.indexOf(')');
String[] strl;
sub = str.substring(8, indxofp);
strl = sub.split(",");
c1 = Float.parseFloat(strl[0]);
```

*Figura 8.3.2-3. Detall del tractament del missatge enviat pel dispositiu portàtil. S'obté un array **str1** on cada element representa una component RGBC. La 1a component str1[0] és el valor de Clear. Codi present a la classe **colors.***

Per obtenir el color en RGBC estàndard de 8 bits (valors entre 0 i 255), s'obté la part proporcional de cada component RGB respecte el clear i es multiplica per 255, seguidament es fa un arrodoniment i s'emmagatzema com a variable pública de la classe **colors.**

```
r1 = Float.parseFloat(strl[1]) / c1 * 255;
g1 = Float.parseFloat(strl[2]) / c1 * 255;
b1 = Float.parseFloat(strl[3]) / c1 * 255;
b = Math.round(b1);
r = Math.round(r1);
g = Math.round(g1);
```

*Figura 8.3.2-4. Detall de la transformació del missatge i obtenció de l'estàndard de RGB de 8 bits, a la funció **getcols** de la classe **colors**.*

Després d'aplicar la funció **getcols**, es pot modificar la interfície gràfica de l'aplicació, mostrant en el botó **col** (el botó central i més gran del MainActivity) el color reconegut pel sensor.

```
  color.getcols(readMessage);

  col.setBackgroundColor(Color.argb(255, color.r, color.g, color.b));
```

*Figura 8.3.2-5. Detall de l'ús de l'objecte **color** del tipus **colors** en la funció **Handler** del **MainActivity**.*

Un cop es disposa d'un objecte **color** de la classe **colors**, ja es pot afegir a la base de dades de la biblioteca, prement el botó "enviar color a la biblioteca". Aquest botó té lligat un *Listener* que fa una crida al GlobalFile, concretament a la funció apiPutCallWithPath, que requereix dos paràmetres: La ruta del recurs de l'Api i l'element a enviar en format JSON. Per obtenir aquest element en format JSON, s'utilitza la classe JSONUtilsFile, concretament la funció fillCommand, que s'ha programat per rebre un objecte tipus **colors** i que retorna un objecte String adaptat a l'estructura JSON. El format de l'objecte està definit en la figura 7.2-4 .

```
enviarBiblio.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

GlobalFile.INSTANCE.apiPutCallWithPath("/colors",js.fillCommand(color));

    }
});
```

Figura 8.3.2-6. Detall de la crida a l'API, desencadenada al fer click en un botó. Codi present al MainActivity.

```
class JSONUtilsFile {

    fun fillCommand(color: colors) : String {
        val contactsObj = JSONObject()
        var stringProducts = ""
        try {
            Log.e("String products", stringProducts)

            contactsObj.put("vermell",color.r.toString())
            contactsObj.put("verd", color.g.toString())
            contactsObj.put("blau", color.b.toString())
            contactsObj.put("clear", color.c1.toString())
        } catch (e: JSONException) {
            e.printStackTrace() }
        val jsonStr = contactsObj.toString()
        Log.e("CONTACTS", jsonStr) //

        return jsonStr    }}
```

Figura 8.3.2-7. Funció fillCommand a la classe JSONUtilsFile

9. Resultat final i projecció de futur

En aquest capítol es comenten alguns aspectes dels resultats obtinguts en finalitzar el dispositiu portàtil dissenyat i construït en aquest projecte. També es presenten les possibles millores aplicables al dispositiu obtingut.

9.1. Generalitats

EL consum d'energia del dispositiu portàtil, a grans trets, es desglossa en:

- El consum del Bluetooth HC-05, que depèn de l'operació, consumeix entre 20-40mA
- El consum del PIC16F690, que quan està en el loop infinit, consumeix menys de 1mA i pot pujar fins els 2-3mA.
- El sensor TCS34725, que juntament amb el LED que té integrat, consumeixen entorn 14mA.

Per tant, hi ha un consum d'uns 57mA considerant el màxim de cada rang mencionat anteriorment. Això implica que el dispositiu portàtil construït presenta una autonomia de 21 hores de funcionament continu, ja que disposa d'una bateria de 1200mAh recarregable amb USB. Aquest dispositiu pot ser emprat en diferents entorns ja que està assemblat robustament.

El consum d'energia també es controla eficaç i còmodament connectant i desconnectant la bateria amb l'interruptor.

(a)

(b)

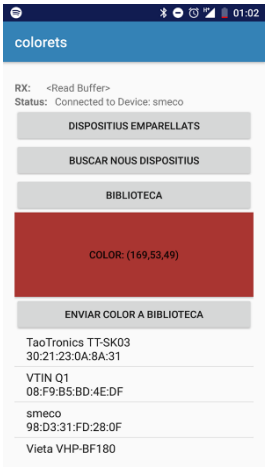
Figura 9.1-1 Dispositiu portàtil assemblat i finalitzat, abans (a) i després (b) de canviar l'estat de l'interruptor.

. El disseny d'aquest dispositiu no ha contemplat consideracions estètiques. Tampoc ha contemplat especialment l'ergonomia de la carcassa; com a conseqüència, pot no resultar còmode sostenir-lo sols amb una mà i prémer el botó. La mida de l'aparell podria reduir-se considerablement ja que l'element més voluminós -la bateria- és innecessàriament gran. També pot comprimir-se tot l'assemblatge electrònic utilitzant PCBs especialment creades per connectar els diversos elements presents.

Figures 9.1-3. Dispositiu portàtil assemblat i finalitzat sostingut per una mà.

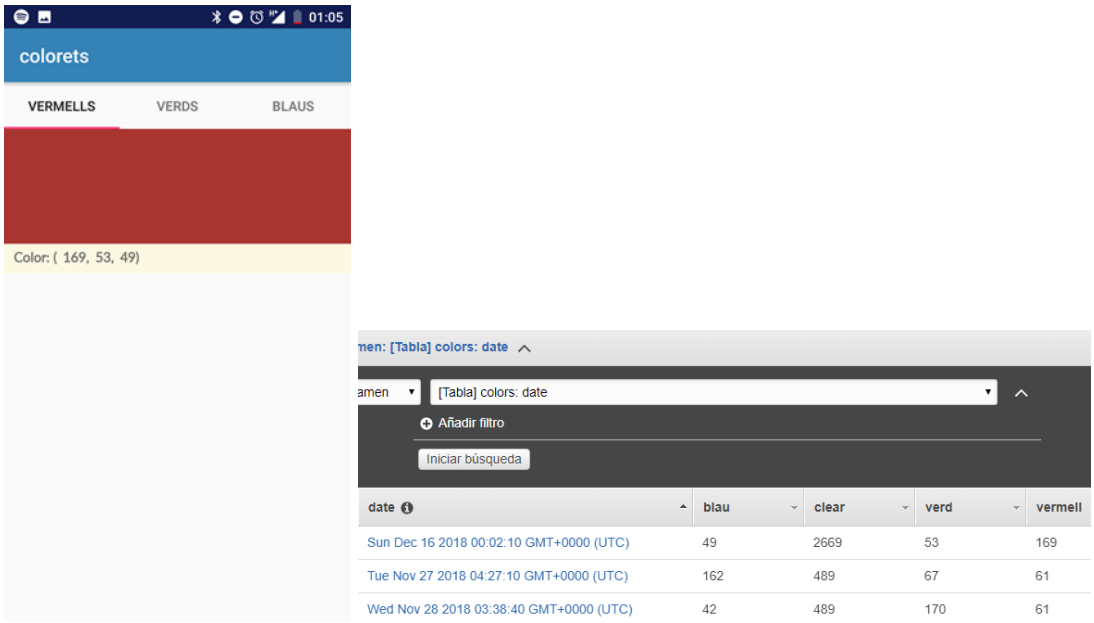
En qualsevol cas, els elements electrònics utilitzats (exceptuant la bateria) i les connexions presents podrien ser una configuració vàlida per un dispositiu final no prototípic.

L'aplicació a l'Smartphone resulta altament prototípica, tampoc s'han fet consideracions estètiques ni ergonòmiques alhora de dissenyar-la. Tanmateix, disposa de les funcions principals i aquestes estan programades robustament i amb un llenguatge eficient, modern i amb projecció de futur com és Kotlin –majoritàriament-. Les latències en les connexions Bluetooth del dispositiu portàtil i l'smartphone són baixes (pocs centenars de ms).



Figures 9.1-4. Passa menys d'un segon des de que es prem el polsador del dispositiu portàtil en la figura 9.1-2 i s'actualitza el botó central a l'aplicació amb el color reconegut.

L'aplicació podria ser millorada i completada en el futur afegint possibilitat de creació de comptes personals i enregistrament d'aquests en el núvol. Això permetria tenir un compte personal per accedir des de qualsevol dispositiu Android. En general, la aplicació també pot completar-se amb altres funcionalitats i millorant interfície gràfica i menús de navegació. També podria considerar-se el desenvolupament per dispositius Apple.



9.1-5 i 9.1-6. A l'esquerra, biblioteca de l'aplicació amb el color enregistrat prèviament i, a la dreta, visualització de l'element (el primer) en la base de dades del núvol.

La biblioteca a l'aplicació podria millorar-se altament, presentant pestanyes o buscadors de colors en funció de diverses capes de filtres: Color majoritari, colors clars/foscos o colors molt/poc saturats per fer còmode la recerca de colors en la paleta registrada fins aleshores. També es planteja en un futur afegir la localització de cada color reconegut, permetent ordenar els colors examinats en funció de la posició geogràfica, o inclòs mostrant els diferents elements sobre un mapa. Aquesta funcionalitat seria fàcil d'implementar mitjançant l'API del GPS de l'Smartphone Android i l'API de GoogleMaps en Android, per exemple.

És fàcil escalar la base de dades horitzontalment ja que és tipus noSQL, fet que permetria afegir columnes noves (tals com columnes per contenir latitud/longitud assignades a cada

color enregistrat), i seria fàcil afegir noves funcionalitats a l'App a través de l'API creada, connectant-la amb nous recursos i mètodes. Això és degut a que el backend que s'ha creat per l'aplicació és altament robust, ràpid, segur, escalable, modern i multiplataforma. En definitiva, un backend complet i professional.

9.2. Resultats cromàtics

Els colors reconeguts distenen certa mesura del que aprecia l'ull humà especialment en els colors més clars. Això és degut, en part, a que el sensor no està totalment aïllat i rep llum rebotada de les parets de la pròpia carcassa, entre d'altres. També sembla que el sensor no capta especialment bé els colors propers al groc (colors amb baixa component blava i component similar de vermell i verd), enfosquint-los considerablement.

La resta de colors, especialment els primaris, són captats amb encert, com els vermells i els blaus i els colors resultants de la suma d'aquests dos.

Una altre consideració és que els colors dels objectes translúcids (on la llum que emet el LED parcialment travessa l'objecte i no rebota) no són especialment ben reconeguts per l'aparell. Tampoc es reconeix bé el color de certes superfícies corbes on la llum rebota en una direcció que no la fa incidir sobre el sensor. És a dir, la quantitat de llum incident sobre el sensor és un factor clau alhora de valorar la qualitat dels resultats.

És clar que la precisió i resolució del l'aparell podria ser millorada utilitzant un sensor més professional (i més car), que disposés d'una matriu de fotodíodes més gran i més sensible.

10.Pressupost

Es desglossa el pressupost en la següent taula:

COSTOS DE MATERIALS			
Concepte	**Quantitat**	**Preu unitari (€)**	**Preu total (€)**
Protoboard	1	18,00	18,00
Microcontrolador PIC16F690	1	2,70	2,70
Programador PICkit3	1	19,99	19,99
Bateria Nokia BK	1	15,95	15,95
Sensor TCS34725	1	7,99	7,99
Micro USB port	1	0,95	0,95
Mòdul Bluetooth HC-05	1	4,90	4,90
Regulador step-up	1	3,95	3,95
Plaques PCB	2	1,00	2,00
Interurptor	1	1,85	1,85
Polsador	1	1,20	1,20
Impressió 3D	1	32,85	32,85
Passadors roscats	2	0,40	0,80
Amazon Web Service	-	1,00	1,00
Cost total de materials			**74,37**

COSTOS DE PERSONAL			
Concepte	**Preu/hora (€/h)**	**Hores (h)**	**Preu total (€)**
Treball estudiant	15	370	5500
Tutories	50	10	500
Cost total de personal			**6000**
COST TOTAL DEL PROJECTE			**6074,37**

11. Conclusions

Aquest projecte tenia l'ambició de dissenyar i prototipar un aparell funcional per reconèixer els colors dels objectes i s'han complert els objectius. Aquesta ambició era altament eclèctica, ja que requeria coneixements d'electrònica, disseny, prototipatge, programació i, tanmateix, s'ha pogut desenvolupar el projecte en menys de 4 mesos i dedicant un temps limitat.

Per altra banda, també s'ha pogut crear una aplicació mòbil amb software i tècniques professionals sense utilitzar cap framework o eina per prototipar aplicacions mòbils, aprofundint en el coneixement de desenvolupament de software per Smartphones i encabint-lo en un projecte majoritàriament de hardware-electrònica. A més, s'ha completat amb un backend complet i modern utilitzant tècniques i eines de tecnologia puntera en aquest àmbit.

Per tant, admetent que els resultats són bons amb les eines utilitzades, es considera aquest projecte altament satisfactori.

Per acabar...

Aquest treball s'ha fonamentat en el model RGB, un model que racionalitza, ordena, talla i cataloga el color com el resultat de la suma de 3 variables. Més concretament, les variables són l'amplitud de les ones electromagnètiques de tres rajos de llum filtrats, on cada raig és d'un color primari (i cada color primari es diferència perquè les seves ones electromagnètiques tenen assignades unes freqüències determinades).

Aquesta catalogació fomenta una concepció individualitzada de l'experiència del color, l'objectiva com a element aïllat del seu entorn, la disgrega de l'univers i l'etiqueta; lligant cada element únic i irrepetible (cada color) amb uns números concrets: els valors de 3 variables.

El primer model de color apareix en plena era moderna ó, més concretament, a l'any 1931, amb la creació del **CIE 1931 RGB color space**. Cinc anys abans, apareixeria a França la **AFNOR**, la *Associació Francesa de la Normalització* i, 16 anys més tard, la categorització seria una activitat flagrantment globalitzada amb la fundació de la **ISO**, la major associació per l'estandardització i normalització, que ja incorpora avui dia 162 països membres.

És natural, llavors, que la nostra forma de concebre el color estigui altament impactada pels plantejaments, estructures i corrents de pensament de l'Europa modernista; la modernitat, tal com la defineix Focault, és "*una etapa de la història on [...] es prioritza l'individualisme, la llibertat [universal], on es té fe en un inevitable progrés científic i tecnològic, on [succeeix] la racionalització...".*

Entenem llavors avui dia cada color com un element normalitzat i universal, categoritzat, individual, irrepetible i racionalitzat, definible a través d'un conjunt de números. Es manifesta fins a quin punt aquesta concepció ha penetrat en l'imaginari comú que, si repassem el diccionari de la **Real Lengua Española**, veiem que defineix el color com la *"Sensación producida por los rayos luminosos que impresionan los órganos visuales y* ***que depende de la longitud de onda****",* explicitant així i fent èmfasi en un **lligam de un color amb un número concret** (longitud d'ona). Ni el diccionari de 1925, 1884, 1817 o 1780 considera fer una relació entre color i un número o planteja algun tipus de racionalització, més bé fan relacions de *color i cos*, com en l'edició de 1884 on part de la definició fa "*Los colores toman nombre de los objetos ó sustancias que los presentan" o,* a l'any 1780, *"comunmente se entiende por COLOR lo que hace visible la superficie de los cuerpos";* concepcions del color més holístiques i més corpòries, menys estrictes i menys compromeses amb la individualitat de la identitat de cada color. És a dir, antigament hi havia una concepció del color no normalitzada.

Percebre, com a fenomen, és una experiència corpòria, altament subjectiva i holística. És natural, llavors, que en la concepció categoritzada i racionalitzada que tenim del color ens sorprengui que dos representacions d'un únic color RGB siguin aparentment diferents.

Els quadrats petits inscrits a l'esquerra i a la dreta són del mateix color RGB.

La majoria de persones dirien que els dos quadrats inscrits són de diferent color. El color es podria considerar una entelèquia de la percepció. La sensació produïda per la llum que impressiona els òrgans visuals depèn flagrantment del seu entorn, del cos (no estrictament del cervell) i història de la persona que percep el color, no solament dels òrgans visuals. Per exemple, la sensació que percep una persona amb cromatofòbia no depèn estrictament de la longitud d'ona, més bé depèn de la representació visual que fa aquella persona en funció de diverses variables. A les figures d'amunt, una persona amb cromatofòbia podria desencadenar desassossec envers un dels quadrats i no per l'altre, tot i ser la mateixa longitud d'ona.

Qui sap si, més endavant, es podrà catalogar el color en funció de la sensació produïda en una persona i no en funció de longituds d'ona.

Tanmateix, per ara, el model RGB és una eina útil i altament eficaç per certs propòsits d'enregistrament de colors i reproducció d'aquests, i és innegable que (tret d'estar relacionat amb el món de les belles arts o de la impressió i, per tant, amb el model CMYK, **C**yan, **M**agenta, **Y**ellow and **K**ey) és difícil pensar en color i no lligar-ho a models RGBs, tot i que és una concepció del color que data de menys de 100 anys.

Agraïments

S'agraeix a la tutora d'aquest treball, la Dra. Rosa Rodríguez, que sempre s'ha disposat per ajudar en el desenvolupament d'aquest projecte, humana i amigablement. Gràcies per la calor i els coneixements compartits.

S'agraeix també al Roger Soriano els consells sobre l'estructura de l'aplicació mòbil, valuosos i que han servit com a drecera en la realització del projecte.

Bibliografia

[1] IT USER, *Android lidera el mercado de los sistemas operativos móviles en España. [https://www.ituser.es/movilidad/2017/03/android-lidera-el-mercado-de-los-sistemas-operativos-moviles-en-espana, 12 setembre 2018]*

[2] Microchip Inc. *PIC16F690 Datasheet. 2*015. [*http://ww1.microchip.com/downloads/en/DeviceDoc/41262E.pdf, 20 de setembre 2018]*

[3] Avago Technologies. *APDS-9960 Datasheet*, 2013. [https://cdn.sparkfun.com/assets/learn_tutorials/3/2/1/Avago-APDS-9960-datasheet.pdf, 3 d'octubre de 2018]

[4] Texas Advanced OptoElectronics Solutions. *TCS3472 COLOR LIGHT-TO-DIGITAL CONVERTER with IR FILTER.* 2012. *[https://cdn-shop.adafruit.com/datasheets/TCS34725.pdf, 3 octubre de 2018].*

[5] Strat-num. *BTM-5 Bluetooth Wireless TTL Master/Slave Transceiver Module Datasheet.* 2011. [http://www.strat-num.fr/IMG/pdf/bluetooth-module-btm5-datasheet.pdf, 10 octubre de 2018].

[6] NanJing Top Power ASIC Corp, *TP4056 1A Standalone Linear Li-Ion Battery Charger with Thermal Regulation in SOP-8*, 2016 [http://www.haoyuelectronics.com/Attachment/TP4056-modules/TP4056.pdf 3 de novembre de 2018]

[7] E-visiontek Store. *Especificaciones del articulo*, 2018 [https://es.aliexpress.com/store/product/5V-DC-DC-Converter-Step-Up-Power-Supply-DC-DC-Booster-Boost-Buck-Converter-Board-Step/1543304_32635991770.html, 3 de novembre de 2018]*.

[8] Rabtron, *Description.* Data desconeguda. [http://shop.rabtron.co.za/catalog/rocker-switch-spst-with-light-220v-p-6775.html, 10 novembre de 2018]

[9] Robin Brockotter. *3D Printing Geometry restrictions. 2018. [https://www.3dhubs.com/knowledge-base/3d-printing-geometry-restrictions, 17 de novembre]*

[10] Microchip Technology Inc. MPLAB XC8 C COMPILER User's Guide, 2012.

[11] Microhip Technology Inc. *MPLAB XC8 C COMPILER User's Guide*, 2013. [http://ww1.microchip.com/downloads/en/DeviceDoc/52116A.pdf, 6 de novembre de 2018]

[12] Amazon Web Services, AWS Documentation, 2018. [https://docs.aws.amazon.com/index.html#lang/es_es, 24 de novembre de 2018]

[13] Android, *Developer Documentation*. 2018 [https://developer.android.com/reference/android/bluetooth/BluetoothSocket, 19 de novembre de 2018]

[14] Android, *Developer Documentation. 2018 [*https://developer.android.com/guide/topics/connectivity/bluetooth?hl=es-419 19 de novembre de 2018]

[15] Android, *Developer Documentation. 2018 [*https://developer.android.com/reference/android/os/Handler?hl=es-419 19 de novembre de 2018]