UNIVERSITAT DE BARCELONA

Pràctiques en empresa

QuantumlabUB: Quantum Computation

Autor: Josep Bosch Monjo

Supervisors:
Bruno Juliá Díaz
Carles Calero Borrallo
Montserrat Guilleumas Morell

Departament de Física Quàntica i Astrofísica
Universitat de Barcelona
Grau de Física
Octubre 2020 - Juny 2021

Índex

1 Introducció

1.1 QuantumlabUB

QuantumlabUB és un projecte començat al 2018 i en el que estudiants del Grau de Física de la Universitat de Barcelona a l'assignatura de *Pràctiques en empresa* creen programes que intenten fer simulacions i demostracions de diferents aspectes de la mecànica quàntica. Aquests estan supervisats pels professors Carles Calero, Montserrat Guilleumas i Bruno Juliá.

Aquests programes busquen ensenyar resultats de certs problemes de la mecànica quàntica de manera no només qualitativa, sinó que també s'intenten obtenir simulacions quantitativament coherents amb la realitat.

Els programes realitzats en aquest projecte estan pensats per ser portats a xerrades i fires on s'arribi a un públic de tots els nivells de coneixements, interessat en aprendre més sobre el món de la quàntica. Per tant, es busca crear aplicacions visualment atractives, intuitives, interactives i que permetin explicar amb certa facilitat algun punt de la mecànica quàntica. A més, estan pensades perquè vagin acompanyades d'una breu xerrada divulgativa sobre el concepte que pretenen donar a conèixer.

De fet teníem la intenció d'anar a la Festa de la Ciència de Barcelona a presentar els nostres projectes, però per la situació de COVID que limitava els aforaments no vam poder portar tots els programes que s'havíen fet.

1.2 Metodologia de les pràctiques

Aquestes pràctiques s'han fet durant el període lectiu del Grau de Física del curs 2020-2021. Durant aquests mesos hem estat 3 estudiants que hem portat a terme cada un el nostre projecte.

El funcionament era que cada alumne anava desenvolupant el seu programa des de zero pel seu compte i un cop a la setmana ens reuníem els tres alumnes i els tres tutors per comentar els nostres avanços i els possibles problemes que havíem tingut, així com discutir cap a on enfocar el nostre projecte. Cada un de nosaltres feiem una presentació del nostre programa explicant en què havíem estat treballant durant la setmana anterior. Aquestes reunions s'hauríen fet de forma presencial si no hagués estat per la situació de COVID que ho ha dificultat, com a conseqüència totes les reunions s'han hagut de fer de forma telemàtica.

Els alumnes teníem llibertat per organitzar-nos el treball i desenvolupar l'aplicació, podíem triar quin aspecte treballar i per on seguir en cada moment, seleccionant les idees que ens semblaven més intressants després de discutir-les a les reunions, on els professors també ens feien propostes i donaven la seva

opinió sobre la viabilitat de les idees compartides. Entre els alumnes també ens donàvem feedback i vam estar provant els programes dels altres per tal de comprovar que no hi haguessin errors i per veure quines parts podien costar més d'entendre a algú que l'utilitzava per primer cop. També era útil parlar entre nosaltres ja que en algunes ocasions algun de nosaltres volia implementar algun element que algú altre ja havia utilitzat en el seu programa.

El programa ha estat desenvolupat en *Python* i l'entorn gràfic s'ha fet utilitzant la llibreria *Kivy*. A mesura que hem avançat anàvem pujant el nostre codi a GitHub, que permet descarregar fàcilment el codi i guarda les diferents versions del programa al llarg del desenvolupament. El codi de cada un està dins d'una carpeta del GitHub comú de *QuantumlabUB* on hi ha tots els projectes fets al llarg dels anys.

1.3 Projecte

Existeixen diverses empreses que estan desenvolupant ordinadors quàntics cada vegada més potents. En determinades tasques com per exemple factoritzar un nombre gran, aquest tipus d'ordinadors presenta un gran avantatge davant els ordinadors convencionals pel que es refereix a poder computacional. Els ordinadors quàntics es basen en els principis de la mecànica quàntica i el seu funcionament difereix molt del d'un ordinador clàssic, per tant els seus circuits i algoritmes també seran molt diferents.

Revisant els projectes fets en anys anteriors vaig veure que molts es centraven en resoldre equacions (Schrödinger, Gross-Pitaevskii, etc.) i fer simulacions a partir d'aquestes utilitzant mètodes numèrics. No hi havia cap programa enfocat a la informació quàntica o la computació quàntica, que són temes que m'interessen bastant. A més molta gent a qui estan destinats aquests programes ha sentit a parlar d'ordinadors quàntics però la majoria no saben com funcionen ni per què poden servir. A part també és important mencionar que últimament està creixent molt l'interès per la computació quàntica en l'àmbit de la investigació, en els darrers anys han aparegut moltes noves empreses en aquest sector i algunes grans empreses com Google, IBM o Microsoft també han començat a ficar-se en el desenvolupament d'ordinadors quàntics. Per aquestes raons vaig decidir centrar el meu projecte en la computació quàntica, ja que a més també m'interessava molt aprendre més sobre el tema i fent aquest programa era una manera d'aconseguir-ho.

El meu programa té com a objectiu ensenyar com funciona la computació quàntica i com es programa un ordinador quàntic, permetent a l'usuari crear els seus propis circuits de forma intuïtiva i ensenyant-li algunes aplicacions i algoritmes útils.

1.3.1 IBM Quantum Experience

Existeix un dissenyador de circuits quàntics online per principiants anomenat *IBM Quantum Experience* en el qual m'he bastat bastant, però té alguns inconvenients que he intentat evitar en el meu programa:

- S'ha d'haver mirat com funciona cada porta quàntica prèviament. Aquesta mateixa pàgina té un document on estan explicades totes les portes. El problema que veig és que no és interactiu ni intuïtiu per a una persona que no està familiaritzada amb la mecànica quàntica ni té un nivell alt de matemàtiques.
- Un altre problema de crear aquest tipus de circuits amb estats de diversos qubits és que a mesura que vas pujant els qubits totals, algunes característiques que et permeten visualitzar els estats deixen d'estar disponibles. Quan arribes a 9 qubits cap d'aquestes característiques està disponible i l'únic que pots fer és enviar el circuit a un simulador que et donarà un histograma amb els resultats obtinguts. Aquest procés tarda un segons i els resultats només es poden veure des d'una altra pàgina de resultats, el que fa que no sigui una forma àgil de visualitzar els canvis en el nostre circuit.

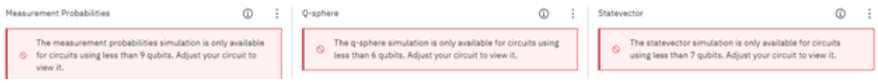


Figure 1: La visualització de la taula de probabilitats, la Q-esfera i el vector d'estat no estan disponible per circuits de 9, 6 i 7 qubits respectivament.

- No apareixen exemples de circuits o algoritmes de manera que et surti el circuit explicat per parts i el puguis manipular o hi puguis interactuar. Si vols algun exemple d'algoritme s'ha de buscar a part i construir-lo tu mateix.
- Les portes que pots utilitzar en els teus circuits estan limitades a les més típiques. Tens la opció de fer portes personalitzades però s'ha de fer escrivint codi en llenguatge QASM.

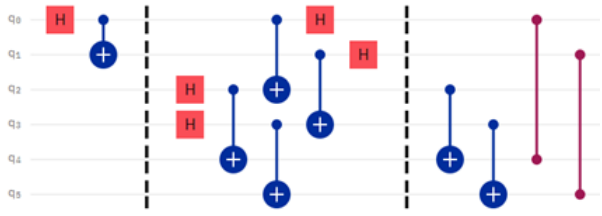


Figure 2: Exemple d'un circuit creat amb el IBM Quantum Composer.

1.4 Introducció als circuits quàntics

El funcionament dels ordinadors quàntics és molt diferent del dels ordinadors clàssics i per tant els seus algoritmes també. La informació dels ordinadors clàssics està codificada en binari, utilitzant cadenes de 0's i 1's, on cada un d'aquests nombres és la unitat mínima d'informació i s'anomena *bit*. En canvi, la unitat mínima d'informació quàntica la podem associar als qubits, aquests qubits es tracten de sistemes quàntics amb dos estats d'energia accessibles, per fer-ho més fàcil podem anomenar aquests dos estats com $|0\rangle$ i $|1\rangle$. Al ser sistemes quàntics, es regeixen per la mecànica quàntica i això significa que aquests dos estats poden estar superposats, és a dir, un qubit pot estar en una superposició $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ on es compleix que per normalització $|\alpha|^2 + |\beta|^2 = \text{Prob}(0) + \text{Prob}(1) = 1$.

L'estat dels qubits es pot modificar mitjançant uns operadors unitaris que anomenarem *portes quàntiques*, si tenim més d'un qubit podem definir portes quàntiques que afectin a diversos d'aquests qubits i així poder aprofitar-nos d'un altre concepte molt interessant de la mecànica quàntica com és l'entrellaçament.

Una porta molt utilitzada és l'anomenada porta de Hadamard, que s'aplica a un sol qubit i ve representada per la matriu

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{1}$$

que si s'aplica per exemple sobre un estat $|0\rangle$ el porta a l'estat $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = |+\rangle$.

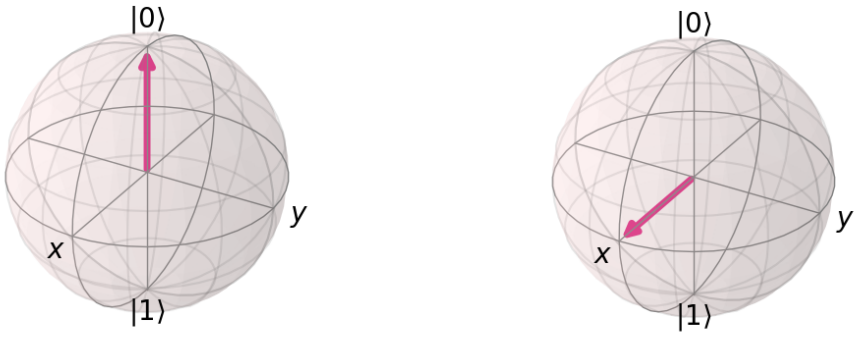


Figure 3: Efecte de la porta Hadamard sobre un estat apuntant en la direcció Z vist en l'esfera de Bloch. Podem veure que en aquest cas l'estat passarà a apuntar en la direcció X.

A partir d'aquí, es poden crear algoritmes amb aquestes portes quàntiques que permeten resoldre certs problemes molt més ràpidament que els ordinadors

clàssics, ja que un ordinador quàntic pot fer actuar un algoritme sobre una superposició de molts estats i que aquest faci que només "sobrevisqui" l'estat solució del problema, fent que l'amplitud de la resta d'estats s'anul·li i per tant augmentant la probabilitat de mesurar l'estat solució. La construcció d'aquests algoritmes no és trivial ja que no tenen res a veure amb algortimes clàssics.

La manera de representar els circuits és dibuixant una línia horitzontal per a cada qubit. Les portes que es volen aplicar a sobre els qubits es posen a sobre d'aquestes línies horitzontals, de manera que l'ordre d'aplicació serà d'esquerra a dreta.

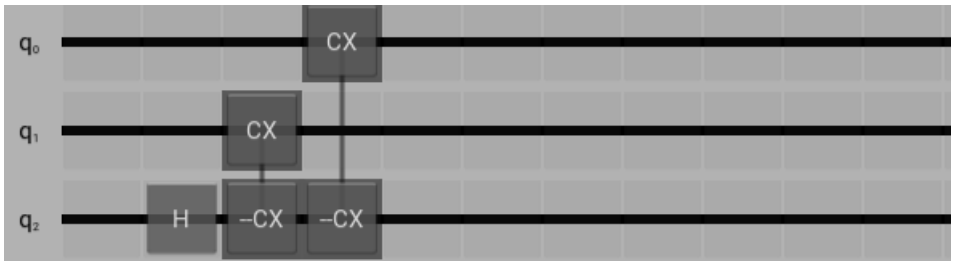


Figure 4: Circuit de 3 qubits, primer s'aplica una porta Hadamard sobre el qubit 2, després una porta NOT sobre el qubit 1 controlada per el qubit 2 i finalment una altra porta NOT sobre el qubit 0 controlada per el qubit 2. El resultat d'aquest circuit és una superosició dels estats $|000\rangle$ i $|111\rangle$.

De fet, aquests ordinadros quàntics basats en portes no són la única manera de fer computació quàntica, però són dels més ambiciosos ja que permeten resoldre un major nombre de problemes diferents, el problema és que també són els més difícils de construir sense obtenir grans errors. També existeixen uns dispositius anomenats *Quantum Annealers* que en lloc de manipular l'estat dels qubits com es fa en el model de portes, el que fa és buscar l'estat fonamental d'un sistema que es sap que és solució del problema que es vol resoldre. L'avantatge és que són més fàcils de construir i l'inconvenient és que no poden resoldre tots els problemes que un ordinador de portes és capaç de resoldre, a part que els algoritmes tampoc funcionen igual. Tot i així, aquest projecte està pensat completament per computació quàntica basada en portes.

2 Guia d'usuari

Les diferents pantalles del programa estan basades en el dissenyador de circuits per a un nombre arbitrari de qubits, aquest té tres panells principals:

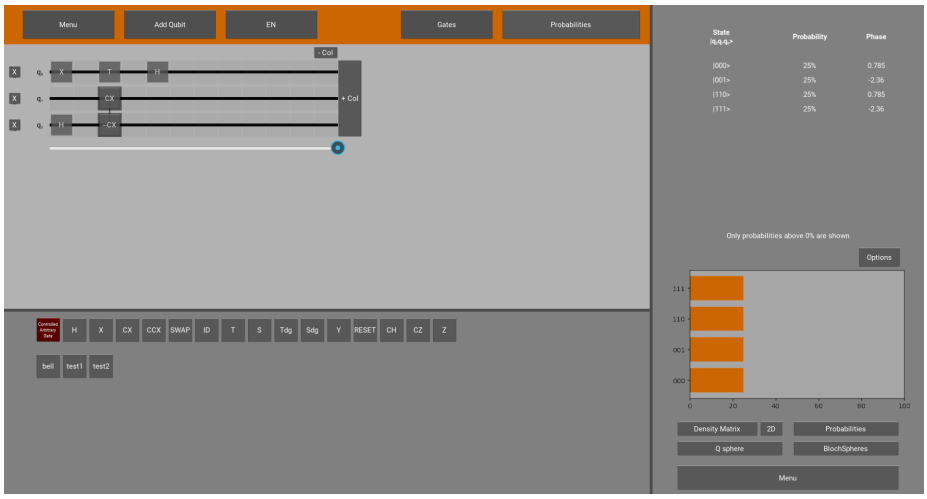


Figure 5: Dissenyador de circuits, podem veure el circuit amb els diferents qubits (cada línia horitzontal correspon a un qubit), a la dreta el panell de probabilitats i a baix el panell de portes.

2.0.1 Panell de probabilitats

Aquest panell es mostra a la dreta de la pantalla i indicarà les probabilitats d'obtenir cada estat possible després d'aplicar el nostre circuit als qubits i altres característiques interessants. Es pot mostrar i amagar amb un botó *Probabilitites* de la barra superior. Les dades d'aquest panell s'actualitzen automàticament cada cop que es fa un canvi en el circuit. Els diferents elements que hi trobem són:

- Llistat d'estats en forma de taula, juntament amb la probabilitat de mesurar-los i la fase corresponent. En cas de tenir molts estats, permet fer scroll cap abaix per poder-los veure tots.

- Botó *"Options"* que obra un menú on es pot ajustar el llindar de probabilitat a partir del qual es mostren els estats en la taula i en la gràfica. Una altra opció és ajustar la direcció de mesura de cada un dels qubits modificant els angles θ i ϕ, o directament seleccionant la direcció d'un dels tres eixos principals X, Y i Z. També es pot veure la direcció triada en l'esfera de Bloch (en una finestra emergent) clicant en el botó "Show", la direcció predeterminada és en l'eix Z.
  En el panell sortirà un missatge indicant el llindar de probabilitats i un missatge avisant-nos de que algun qubit no s'està mesurant en la direcció Z si és el cas.

- Gràfica dels possibles estats resultants de mesurar l'estat final del circuit, on l'amplitud de les barres indica la probabilitat de mesurar-lo. El llindar de probabilitat també s'aplica en aquesta gràfica.

- Figures addicionals que es generen al clicar els botons de sota la gràfica. Aquestes figures es generen directament mitjançant *Qiskit* i no són afectades pel llindar de les opcions. Es poden representar la matriu densitat en 3D i en 2D, les probabilitats, la Q-sphere i les esferes de Bloch, totes s'obren en finestres emergents.

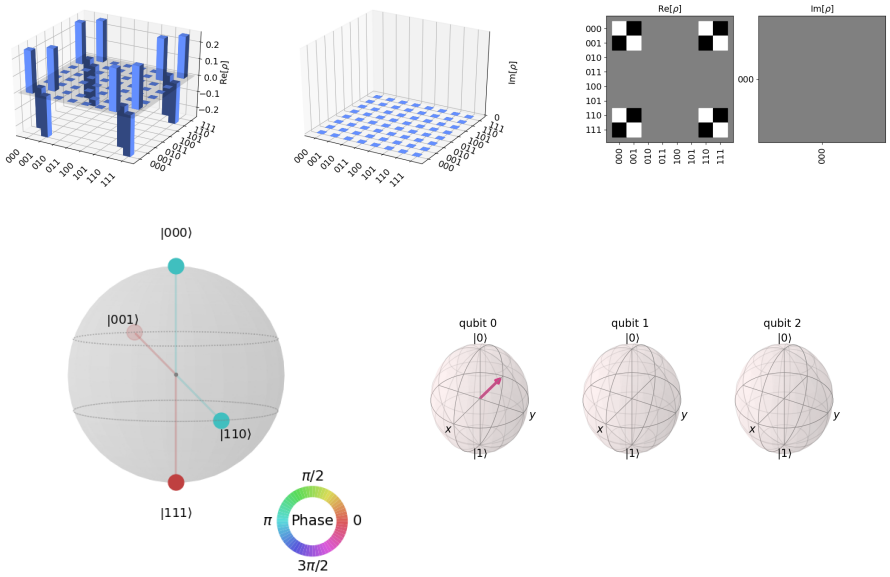


Figure 6: Algunes de les figures que es generen amb els botons del panell de probabilitats. A dalt tenim la representació de la matriu densitat, en 3D i en 2D, i a baix a l'esquerra la Q-esfera i a la dreta les esferes de Bloch de cada un dels qubits, per els qubits 1 i 2 no apareix la fletxa ja que estan entrellaçats i no es pot definir un estat independent per cada un d'ells.

2.0.2 Panell de portes

Aquest panell es mostra a sota de la pantalla i ens mostrarà totes les portes que podem aplicar al nostre circuit, igual que el panell de probabilitats es pot amagar mitjançant el botó *Gates* de la barra superior.

El primer botó que hi trobem anomenat *"Controlled Arbitrary Gate"* es diferència per ser vermell i permet crear portes controlades (o no) arbitràries. Al clicar-hi s'obrirà un menú on primer haurem de seleccionar el nombre de qubits de control i el nombre de qubits sobre els que aplicar la porta (*target*) que volem. Si triem un únic qubit *target* ens sortirà la opció de triar els angles de rotació que volem que apliqui la porta i un llistat amb les portes de un qubit disponibles, tant les comunes com les personalitzades. Si volem aplicar

una rotació hem de triar les angles i apretar *"Apply"*, si volem aplicar una porta ja feta simplement cliquem a sobre d'aquesta. Si en canvi hem triat més d'un *target qubit*, només tindrem la opció de triar una porta ja feta. S'ha de tenir en compte també que quan tinguem qubits de control, a la hora d'aplicar la porta en el circuit haurem de clicar primer els qubits de control i després els *target qubits*.

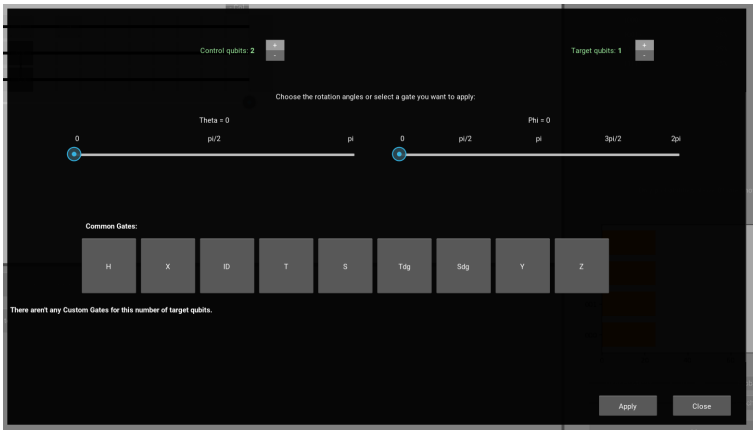


Figure 7: Menú per crear portes arbitràries controlades.

Al costat d'aquest botó vermell, ens apareixeran la resta de portes quàntiques més comunes. Una mica més avall ens sortirà una llistat de les portes personalitzades que tinguem fetes en el nostre ordinador. En cas de tenir moltes portes, el panell permet fer scroll per poder-les veure totes. Per el cas de les portes personalitzades, fent doble click sobre elles surt en una pantalla emergent un esquema del circuit equivalent a aquella porta. En cas d'aplicar-la s'hauran de clicar per ordre els qubits las que volem que s'apliquin, tinguent en compte que s'ha de fer en ordre, començant per el q_0 de l'esquema.

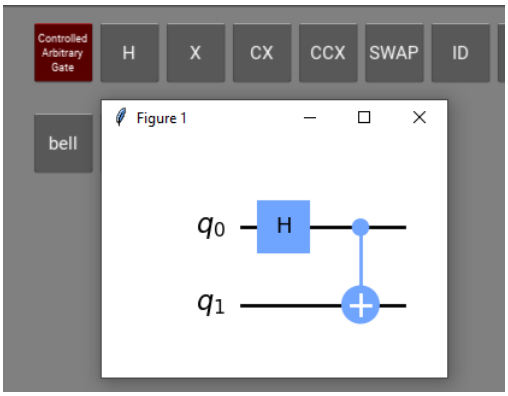


Figure 8: Esquema mostrat per a les portes personalitzades al fer-hi doble click.

2.0.3 Panell del circuit

Aquí és on hi ha el nostre circuit, apareixen els nostres qubits i els hi podrem aplicar les portes quàntiques que vulguem. El circuit s'ajusta al tamany de la pantalla automàticament depenent de si els panells de probabilitats i de portes s'estan mostrant o no.

A la barra superior apareix un botó *Add Qubit* que afegirà un qubit nou al nostre circuit. A l'esquerra de cada un d'ells apareix un botó amb una X que esborrarà el qubit i totes les portes que se li apliquin. En cas d'esborrar un qubit, tots els que hi hagi per sota pujaran i es renombraran per tal que quedin tots junts i ordenats.
En cas de voler aplicar una porta simplement s'ha d'arrossegar des del panell de portes fins al lloc del circuit on es vulgui aplicar. Per les portes de múltiples qubits s'ha de tenir en compte que s'han de clicar un qubit rere l'altre, en la mateixa columna. En cas de que es cliqui fora abans de seleccionar tots els qubits necessaris, la porta desapareixerà. També es poden arrossegar les portes a un altre lloc clicant-les durant aproximadament 1 segon i arrossegant-la a un altre lloc, si s'arrosseguen a fora del circuit s'esborraran.

A la dreta del tot tenim dos botons, *"+ Col"* i *"- Col"* que afegeixen i esborren una columna respectivament. En cas de que les files o columnes sobresurtin de la zona del circuit es podran veure fent scroll, tant vertical com hortizontalment.

A sota els qubits apareix un slider que per defecte està col·locat després de la última columna. El podem moure a qualsevol alçada del circuit i el que farà serà donar-nos els resultats en el panell de probabilitats en aquell punt del circuit, és a dir, només s'aplicarà el circuit des de l'esquerra del tot fins al punt seleccionat amb l'slider. Això permet veure millor com van progressant els estats de mesura dels qubits al llarg del circuit sense haver de modificar-lo.

Ara bé, en el circuit principal trobarem diferentes pantalles a les que podrem accedir:

Figure 9: Menú principal que apareix quan s'executa el programa.

2.1 Circuit Builder

Aquesta pantalla inclou totes les caracterítisques del dissenyador de circuits descrites anteriorment. La pantalla està pensada perquè l'usuari pugui crear els seus circuits i fer proves amb les seves portes i els seus algoritmes.

2.2 Custom Gates

Aquesta pantalla està pensada per fer portes personalitzades que poden consistir en un conjunt de portes aplicades sobre un o diversos qubits que volem tenir comptactades en una porta per si per exemple utilitzem amb molta freqüència. Un altre ús és per fer petits circuits que s'aplicaràn després en un circuit més gran. Per fer portes personalitzades més simples com per exemple rotacions arbitràries o controlades ja hi ha un altre menú en el dissenyador de circuits.
Al clicar en aquest botó el primer que ens sortirà és un menú amb la llista de portes personalitzades que tinguem guardades en la carpeta del nostre programa, per tant aquestes portes es guarden si es tanca i torna a obrir l'aplicació. A sota apareixen diferents botons:

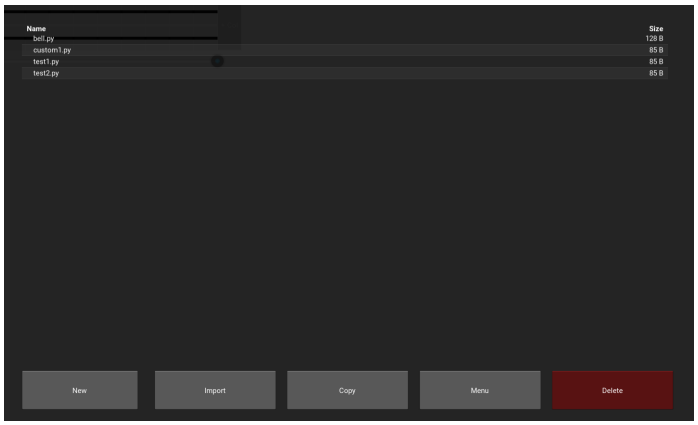


Figure 10: Menú que apareix per a gestionar les portes personalitzades disponibles.

"New" que primer ens demanarà que donem un nom a la nostra porta i després ens obrirà el dissenyador per tal que fem una porta personalitzada nova.

El botó *"Import"* que després d'haver clicat sobre una porta de la llista, ens l'obrirà en el dissenyador podent-la modificar.

El botó *"Copy"* que ens demanarà un nom que li volem donar a la còpia que es crearà de la porta que tinguem seleccionada.

"Delete" esborrarà permanentment la porta que tinguem seleccionada.

Un cop a dins del dissenyador, el funcionament és igual que el *Circuit Builder*, la unica diferència que hi ha és que aqui no apareixeran el llistat de portes personalitzades, per tal d'evitar que se n'utilitzi una dins una altra, i que aquesta alhora sigui utilitzada dins la primera, creant un efecte recursiu.

2.3 Demos

En aquesta pantalla es mostren circuits i algoritmes interessants. Al obrir la pantalla apareix una llista dels diferents circuits que hi ha, al obrir-ne un apareix el dissenyador amb el circuit construit. En aquest cas, el panell de portes no apareix ja que aquí la idea no és construir un circuit, sinó veure'n un de ja fet i intentar entendre com funciona. Per això en el lloc on hi havia el panell de portes ara apareix un text descriptiu que es pot anar passant amb les fletxes d'endavant i endarrere.

El text de sota pretèn explicar el circuit, en el cas dels algoritmes es comença explicant el problema que intenten resoldre, després es comenta com resoldria el problema un ordinador clàssic i finalment es passa a explicar l'algoritme quàntic. En cada pas de l'explicació de l'algoritme es ressalta la part de l'algoritme de la que s'està parlant en el panell del circuit.

Cada una de les demostracions estan personalitzades individualment i poden incloure elements extra que permetin ajudar a entendre millor el circuit que es vol explicar en cada cas.

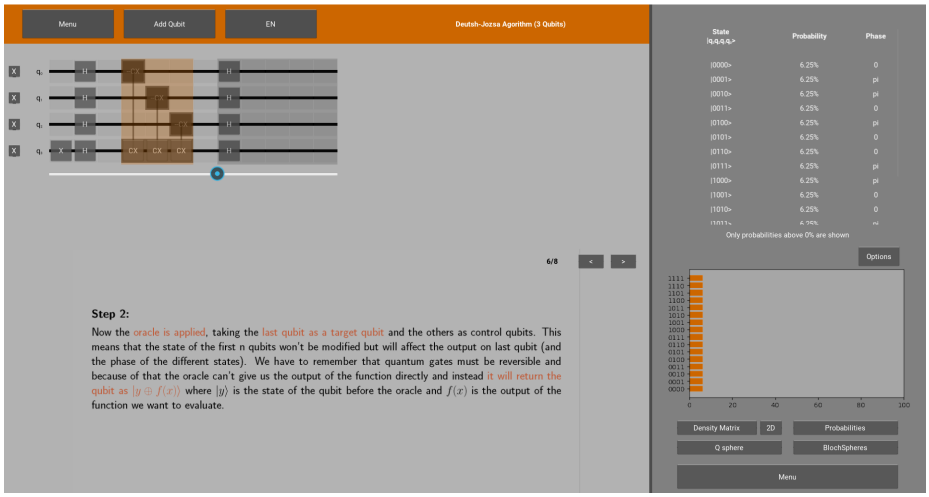


Figure 11: Pantalla de la demostració de l'algoritme de Deutsch-Jozsa per a 3 qubits. A baix en lloc d'haver-hi el panell de portes hi ha una explicació que es pot anar passant endavant i endarrere amb les fletxes. A la figura el text està explicant la segona part del circuit, que es pot veure ressaltada de color groc en el circuit. L'slider de sota els qubits està fent que els resultats mostrats en el panell de probabilitats siguin en aquell punt, just després del pas que s'està explicant en el text.

3 Desenvolupament del programa

Durant els mesos en els que he estat treballant en el programa, he anat comptabilitzant les hores de treball que anava fent cada setmana. Al ser unes pràctiques bastant autònomes, hi ha hagut molta flexibilitat a la hora d'organitzar el volum de feina, podent per exemple dedicar-li menys hores durant les èpoques d'exàmens. En total he fet unes 240 hores en les que s'inclou el temps dedicat a llegir sobre tot el que ha estat necessari per portar a terme el projecte, tant per la part de la idea com per la part de programació, i el temps dedicat a programar i documentar l'aplicació.

3.1 Primeres idees

Quan vaig decidir fer el meu programa relacionat amb la computació quàntica, el primer que vaig haver de fer va ser investigar sobre el tema per tal de poder definir una idea més concreta per començar a desenvolupar-la. Les primeres setmanes de les pràctiques les vaig dedicar a aprendre sobre les portes i els circuits

quàntics i a familiaritzar-me amb el seu funcionament, ja que eren uns conceptes que no havia estudiat mai.

Un altre aspecte amb el que vaig haver de treballar aquest primer període va ser en l'ús de la llibreria *Kivy* i en aprendre com funciona la *programació orientada a objectes*, ja que és una forma de programar bastant diferent a com s'aprèn a programar en la carrera de Física. Aquesta part també va ser molt important abans de definir la idea del programa, ja que em va ajudar a conèixer les limitacions d'aquesta llibreria i la dificultat de dissenyar certs aspectes de la aplicació.

Un cop vaig començar a definir la idea, vaig veure que exisiten diferents llibreries de Python que permetien programar circuits quàntics a partir de portes. Vaig decidir implementar una d'aquestes llibreries ja que per un costat m'estalviava tota la feina de fer el càlcul dels estats del circuit després d'aplicar les portes del circuit, i per l'altra em permet exportar el codi fet amb una d'aquests frameworks per si el vols utilitzar en un altre lloc.

Després d'investigar una mica sobre les diferents opcions, vaig decidir decantarme per la llibreria *Qiskit*, creada per IBM, ja que era la que millor documentada estava i existia la opció d'executar els codis fets en aquest llenguatge en un ordinador quàntic real.

D'aquesta manera, en el meu programa en lloc de fer els càlculs d'aplicar totes les portes sobre els qubits, el que havia de fer era d'alguna manera convertir el circuit fet en el dissenyador en codi *Qiskit*.

3.2 Evolució del programa

Durant tot el procés de programació, degut a totes les dificultats i problemes que m'han anat sorgint, la idea del programa i els objectius s'han anat modificant lleugerament. Per començar, a la idea inicial vaig incloure alguns elements que no he acabat posant en el programa per diverses raons: algunes coses no tenien gaire sentit o no aportaven res útil, altres idees eren molt difícils d'incloure en el programa i altres no les he afegit en el programa per falta de temps ja que hi ha hagut moltes coses que m'han portat bastant més temps del que pensava. Per altra banda, també he afegit algunes coses que a la idea inicial no tenia pensades ja hi he anat pensant al llarg de la programació o me les han proposat els tutors en alguna de les reunions.

A part de les idees, la forma d'implementar-les moltes vegades l'anava improvitzant ja que, sobretot al principi, cada cosa que volia fer requeria d'una recerca primer sobre els diferents elements del *Kivy* amb els que podia fer-ho per acabar triant el que creia que em portaria menys problemes i que em faria la feina més fàcil. Tot i així, moltes vegades la forma de trobar la millor (o única) forma de programar les coses per tal que funcioni tot bé era mitjançant

prova/error.

Un resum dels passos que vaig seguir per fer el programa, amb els principals avanços i els problemes que més em va costar solucionar seria el següent:

El programa el vaig començar creant una graella personalitzada per el panell del circuit de manera que pugués editar qualsevol cel·la lliurement ja que vaig estar intentant fer-ho de moltes altres maneres que haurien estat més senzilles, però no em donaven la llibertat d'edició que jo necessitava. Per tant, vaig haver de crear la graella a partir d'ajuntar elements individuals que corresponíen a les cel·les. Un cop la vaig tenir més o menys feta vaig afegir els panells de portes i de probabilitats amb els botons per ensenyar-los o amagar-los. Després vaig continuar amb els botons per afegir i esborrar files i columnes de la graella i afegint la funció de scroll, a part d'ajustar-ne el tamany en funció de si els altres dos panells s'estaven mostrant o no.

Un cop vaig tenir la graella i els panells fets, vaig treballar una mica en la forma de guardar tota la informació i vaig començar definint unes quantes portes de prova per implementar la funció d'arrossegar-les des del panell de portes fins al panell del circuit i que es quedin a la cel·la on s'han deixat anar. Aquí el problema que tenia era que Kivi a vegaes treballa amb coordenades locals i altres vegades amb coordenades globals, cosa que feia que quan deixava anar una porta en un lloc de la graella, aquesta ho detectés en un altre punt. Després vaig continuar afegint la funció per portes de múltiples qubits i la funció d'arrossegar les portes del circuit per poder-les moure de lloc. Un cop fet tot això vaig acabar d'arreglar la part de la matriu que guarda la informació del circuit i vaig afegir alguns elements gràfics.

Després de tenir feta tota la graella i la funció d'arrossegar les portes, vaig començar a treballar amb el Qiskit i vaig fer un codi que em convertia la meva matriu del circuit en codi Qiskit i vaig posar que s'executés cada cop que es feia un canvi en el circuit. Vaig estar afegint elements al panell de probabilitats per tal de mostrar els resultats del circuit cada cop que feia un canvi en aquest.

Un cop ja tenia el dissenyador de circuits més o menys funcional havent modificat alguns elements i havent afegit algunes funcionalitats més, vaig començar amb la pantalla de *Custom Gates*. Vaig estar mirant com fer aquestes portes personalitzades i vaig decidir crear un arxiu per a cada una. Fer que s'importessin no només al principi, sinó que s'actualitzessin cada cop que n'afegia o en modificava alguna, em va portar bastant problemes ja que no trobava la manera de fer que s'actualitzessin al moment després d'un canvi. Quan ho vaig aconseguir vaig crear tot el menú on surt el llistat de portes amb les opcions d'importar-les, borrar-les, crear-ne de noves, etc.

Vaig continuar afegint les opciones del panell de probabilitats i el menú de portes controlades arbitràries. Això últim em va portar bastant temps de re-

cerca ja que no trobava la manera d'afegir qubits de control a qualsevol porta. Després de provar moltes maneres de fer-ho, vaig trobar una funció de Qiskit que em va ajudar a aconseguir el que volia fer.

El següent que vaig fer va ser començar a treballar amb la pantalla *Demos*, primer buscant algoritmes que podia explicar, vaig buscar la manera de guardar-los amb tota la informació que necessitava i vaig decidir fer una carpeta amb un arxiu per a cada circuit. El panell de portes el vaig amagar i en el seu lloc vaig posar un lloc on afegir la explicació sobre el circuit.

Vaig estar escrivint les explicacions dels algoritmes, afegint alguna figura per intentar que fossin el més intuïtives possible, i tinguen en compte que tindríen el circuit construit al costat. Primer volia posar un text que permetés fer scroll vertical per poder-lo llegir tot bé, pero l'scroll donava problemes i feia que el programa anés molt lent. Després de fer moltes proves vaig acabar per dividir-lo en parts estàtiques i posar fletxes per anar passant endavant i endarrere el text. A més vaig fer que "s'il·luminés" la part del circuit de la que s'estava parlant en cada moment en el text que s'estava mirant. En el dissenyador comú a totes les pantalles també vaig afegir l'slider a sota els qubits que permet mirar els resultats en qualsevol punt del circuit.

Al acabar tot això, i també durant tot el procés, vaig estar millorant alguns aspectes visuals i arreglant errors més petits, així com modificar alguna funció per millorar-la d'alguna manera.

4 Demos

Aquests són algunes de les explicacions dels algoritmes de la pantalla *Demos*.

4.1 Deutsch-Jozsa Algorithm

If we have a hidden function (we can't solve it analitically but we can check the output for a given input) that is either **constant** (returns always 0 or always 1) or **balanced** (returns 0 for half of the inputs and 1 for the other half), we can build the Deutsch-Jozsa Algorithm in order to **determine which type of function we have.**

Example of the two possible function types:

Constant	
Input	**Output**
000	0
001	0
010	0
011	0
100	0
101	0
110	0
111	0

Balanced	
Input	**Output**
000	1
001	1
010	0
011	1
100	0
101	0
110	0
111	1

The way to find the solution in a classical computer is by checking the output for the different inputs. As soon as we get the two possible outputs we can guarantee that the function is balanced.

The problem is that when we start getting the same output for the different inputs tested, the probability of the function being constant increases as we check for more inputs, but we will have to check at least half plus one, $\frac{N}{2} + 1$ of the inputs to be 100% sure that it is a constant function.

In this example with 3 bits (or qubits) there are only 8 different inputs, but if we imagine an input of 10 bits, the number of unique values grows exponentially to 1024.

A quantum computer using the Deutsch-Jozsa Algorithm can solve this problem with a single iteration, it will only have to test 1 input.

The Quantum Algorithm

This algorithm was first proposed in 1992 and is one of the first algorithms to solve a problem faster than a classical machine.

It consists in an algorithm that when applied to n input qubits (+1 additional) you will measure the state $|0\rangle^n$ with **100% probability if the function is constant** and with **0% probability if the function is balanced.**

The short explanation is that the outputs from the **constant function make a constructive interference** for the amplitude of this state since all outputs are the same, while for a **balanced function** it will result in a **destructive interference** between the outputs 0 and 1 and therefore that state won't be found. Now let's move on to a little more detailed explanation.

For the implementation of this algorithm we will need n qubits for the input (describing 2^n states) + 1 more qubit where the oracle output will be applied.

Step 1:

We will have to start applying a Hadamard gate to the first n qubits starting

at the state $|0\rangle$ in order to create a superposition of all possible states. For the last qubit we will have to bring it to the state $|1\rangle$ and then apply a Hadamard gate to get the superposition $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. The global state we have at this point is described by

$$\frac{1}{\sqrt{2^{n+1}}} \sum_{x=0}^{2^n-1} |x\rangle(|0\rangle - |1\rangle)$$

Where $\frac{1}{\sqrt{2^{n+1}}}$ is a normalization factor and $\sum_{x=0}^{2^n-1} |x\rangle$ is the superposition of the first n qubits.

Step 2:

Now the oracle is applied, taking the last qubit as a target qubit and the others as control qubits. This means that the state of the first n qubits won't be modified but will affect the output on last qubit (and the phase of the different states). We have to remember that quantum gates must be reversible and because of that the oracle can't give us the output of the function directly and instead it will return the qubit as $|y \oplus f(x)\rangle$ where $|y\rangle$ is the state of the qubit before the oracle and $f(x)$ is the output of the function we want to evaluate.

$\|y\rangle$	$f(x)$	$\|y \oplus f(x)\rangle$
0	0	0
0	1	1
1	0	1
1	1	0
0	$f(x)$	$f(x)$
1	$f(x)$	$1 \oplus f(x)$
$(\|0\rangle - \|1\rangle)$	0	$(\|0\rangle - \|1\rangle)$
$(\|0\rangle - \|1\rangle)$	1	$-(\|0\rangle - \|1\rangle)$

We can see that the output $f(x)$ only has an effect if it is 1 and just adds a global phase that changes the sign of the initial state.

The global state after the oracle will be

$$\frac{1}{\sqrt{2^{n+1}}} \sum_{x=0}^{2^n-1} |x\rangle(|f(x)\rangle - |1 \oplus f(x)\rangle) \quad = \quad \frac{1}{\sqrt{2^{n+1}}} \sum_{x=0}^{2^n-1} (-1)^{f(x)}|x\rangle(|0\rangle - |1\rangle)$$

Step 3:

The last step is applying again a Hadamard gate to each one, if no oracle was applied, they would return to the initial state, but since the oracle has modified the relative phases of the different states, it may not come back to the $|0\rangle^n$ state.

From here it's easy to see that if we have a sum of N states and half of them give $f(x) = 0$ and the other half give $f(x) = 1$, the first ones will add a positive

amplitude and the last ones will add a negative amplitude to the initial state, ending with an amplitude equal to 0 and a probability of 0% of measuring the state $|000...\rangle$.

If instead we have a constant function, the amplitude will result in +1 or -1, measuring the state $|000...\rangle$ on the first n qubits with a probability of 100%.

4.2 Grover's Algorithm

This algorithm solves the problem of finding an element in an unsorted database. It could be for example an element from a list with a specific property we want, or finding an entry on an alphabetically ordered phone book by the phone number. We will see later that this last example isn't very useful in practice.

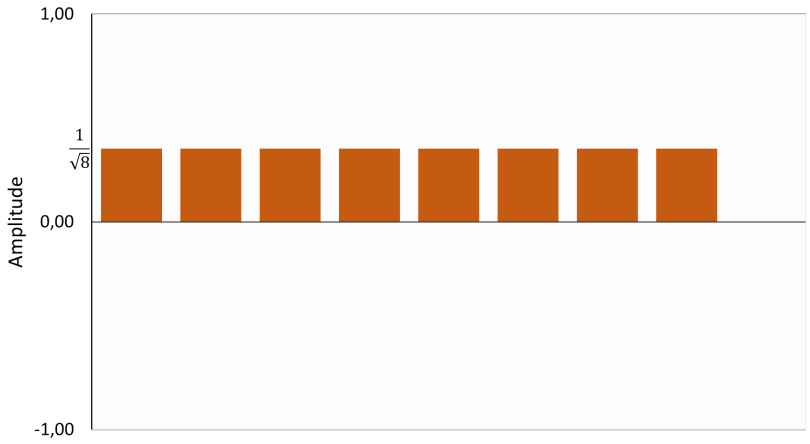
For a classical computer, it is solved by checking the entries one by one until it finds the entry we want. The solution can be found on any number of iterations, from finding it on the first attempt to the last. On average, we will have to check $\frac{N}{2}$ entries on a list of N elements before finding the solution.

If we have a very large database, this way of solving the problem can become quite slow. The Grover's Algorithm on a quantum computer can solve this problem in $\sqrt{N}$ steps, making it way more faster than a classical computer for a large value of N.

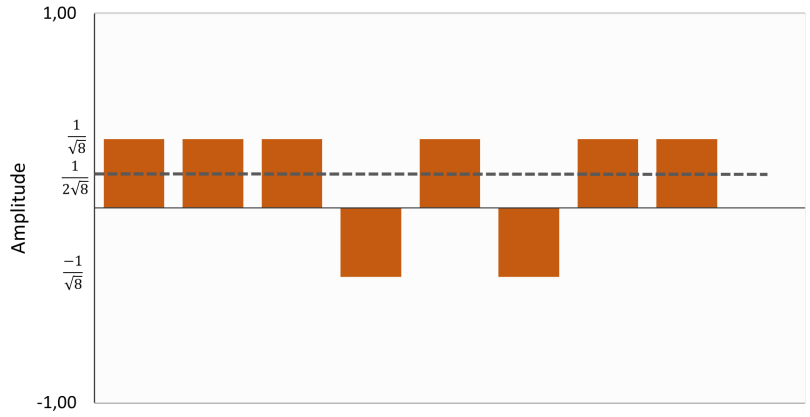
The Quantum Algorithm

Before starting with the algorithm, we will have to assign to each entry in the list a quantum state of our qubits that we can measure.

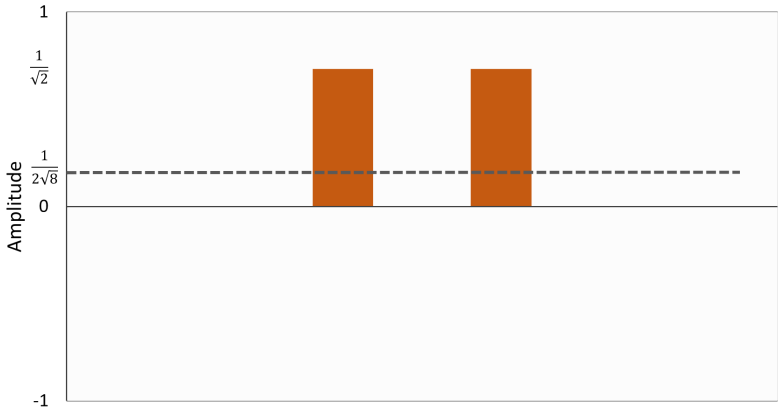
We start with n qubits in the $|0\rangle$ state, the first thing that has to be done is creating a superposition of all possible states by applying a Hadamard gate to all qubits. At this moment we have the same probability of measuring all states since all amplitudes are equal.



The second step is to apply the oracle to the qubits. The oracle is a quantum circuit that reflects the global state with respect to an orthogonal state of the wanted state. It is easier to understand if we look at the amplitudes diagram where what happens is that the sign of the amplitude we want to measure changes.



Now we have to apply a circuit called Amplification, which what basically does is reflect the global state with respect to the initial equal superposition of all states. If we look at the amplitudes, we see that this corresponds to reflecting all amplitudes with respect to the average amplitude (marked with a dash line).



By doing this, the amplitude of the selected state will increase and all other amplitudes will decrease, making the wanted state the most probable to measure.

In our example we have 8 states, so the initial amplitude for all states is $\frac{1}{\sqrt{8}}$. Then we select 2 states and after the first reflection the average amplitude becomes $\frac{6\frac{1}{\sqrt{8}}+2\frac{-1}{\sqrt{8}}}{8} = \frac{1}{2\sqrt{8}}$ that is exactly half of the amplitude of the unwanted states. For that reason, in the next reflection, the amplitude of all these states will become 0 and the amplitudes of the states selected will grow to $\frac{1}{\sqrt{2}}$ that is equivalent to a probability of measuring these states of 50% each.

However, not always the probability of measuring the other stats will be 0 and we will have to apply the Oracle+Amplification around $\sqrt{\frac{N}{M}}$ times, where N is the number of states and M is the number of solutions.

As we said before, the phone book example is not a good practical example for this algorithm. That's because in order to build the oracle, we would have to "mark" the element we want and therefore we would already know the answer of the problem.

Instead, we could use this algorithm to solve problems where the solution has to satisfy a set of conditions. Some examples could be Sudoku or problems like

- *John can only meet either on Monday, Wednesday or Thursday*
- *Catherine cannot meet on Wednesday*
- *Anne cannot meet on Friday*
- *Peter cannot meet neither on Tuesday nor on Thursday*

When can the meeting take place?

These are called Boolean Satisfiability Problems (SAT).

5 Futures millores

He acabat aquest programa amb una versió funcional i que es pot utilitzar perfectament, però també he acabat amb una llista de propostes per millorar el programa. Són coses que m'hauria agradat intentar implementar en cas que les pràctiques haguessin durat més temps.

- Millorar l'aspecte visual afegint icones per als botons i portes.
- Afegir més demostracions d'algoritmes famosos i interessants i afegir-hi més elements per interactuar-hi i poder manipular els estats inicials abans d'aplicar els circuits.
- Afegir una pantalla amb explicacions de conceptes bàsics de mecànica quàntica, d'una manera interactiva, per tal que les persones que no hi estiguin familiaritzades puguin entendre millor el funcionament dels ordinadors i circuits quàntics.
- Incloure un selector d'idioma per poder traduir tots els menus i les explicacions de les demostracions.
- Un altre mode més interactiu: *Mode Objectiu*
  Aquest mode és el més semblant a un joc ja que és molt interactiu i has d'intentar aconseguir un objectiu. Et donen un estat inicial i un estat final al que has d'intentar arribar, la teva tasca consisteix en aplicar portes quàntiques per modificar l'estat donat i aconseguir l'estat demanat.
  - Per tal que el joc no sigui molt difícil s'inclouran nivells de dificultat a més d'un botó de *"Pista"* que ens pot ajudar amb alguna informació com per exemple quines són les portes que necessitarem en el nostre circuit.
  - La forma de comprovar si hem encertat serà amb un botó de *"Comprovar"*, una opció és que ens doni una puntuació indicant quant ens hem acostat a l'objectiu en lloc de Correcte/Incorrecte.
  - Hi haurà una opció de *"Previsualitzar els resultats"* que ens permetrà veure el resultat en temps real mentre construïm el circuit sense necessitat de prémer el botó de *"Comprova"*. Si pel contrari tenim aquesta opció desactivada, haurem de construir el nostre circuit "a cegues" i no veurem el resultat del nostre circuit fins que no clickem *"Comprova"*.

  A més, el joc es dividirà en dificultats per tal de que es pugui jugar independentment del nivell d'intuició que es tigui a la hora de construir circuits quàntics:

  - **Fàcil:** tindrem uns certs qubits i una probabilitat objectiu, aquí les fases no es tindran en compte.
  - **Mitjà:** tindrem alguns qubits més que en el nivell anterior i algunes portes més complexes. Seguirem tenint una probabilitat objectiu.
  - **Difícil:** a partir dels nostres qubits haurem d'aconseguir un estat objectiu, incloent probabilitats i fases.

- Ja que el programa consta d'un entorn gràfic que permet dissenyar un circuit que després és convertit a codi Qiskit, es podria incloure la opció de que el circuit es converteixi en algun altre llenguatge de programació de circuits quàntics com *Qibo, Cirq*, etc.

6 Conclusions

Durant el transcurs d'aquestes pràctiques he après molt en diferents aspectes. Pel que fa al tema principal del projecte que he desenvolupat, la computació quàntica, puc dir que he après molt, els meus coneixements d'ara són molt majors als de l'inici de les pràctiques.

Una altra habilitat en la que he millorat ha estat en la programació ja que el programa fet és molt diferent a tot el que hem fet al Grau de Física. A part de la millora en Python en general, també he après a utilitzar llibreries enfocades en la computació quàntica com ha estat amb Qiskit.

Degut a treballar de forma autònoma en un projecte del que jo mateix definia els objectius he vist que el més probable és que durant el desenvolupament d'un projecte sorgeixin problemes que et retrassin a la hora d'aconseguir els objectius proposats, i que aquests s'han hagut d'anar adaptant segons com avançava el programa i segons les noves idees que anaven apareixent. He comprovat com d'important és organitzar i ser constant en un projecte d'almneys uns quants mesos de durada. Gràcies a haver fet reunions setmanals també he après a presentar el programa en el que estava treballant, a més de rebre feedback i tenir-lo en compte per el progrés del projecte. A més he après que era important compartir el programa i donar-lo a provar a altres persones ja que una cosa que pot semblar fàcil d'entendre o intuïtiva per la persona que ha fet l'aplicació pot ser tot el contrari per una persona que no n'estigui gens familiaritzada.

El que m'emporto d'aquestes pràctiques és, a banda del coneixement, una gran experiència que no hauria pogut guanyar fent alguna altra assignatura del grau, així que estic molt satisfet d'haver cursat l'assignatura de Pràctiques en Empresa. També estic content amb el programa que m'ha quedat al final d'aquestes pràctiques i de fet m'agradaria que el programa no es quedés aquí, sinó que espero poder acabar d'incloure els aspectes mencionats anteriorment que no m'ha donat temps a afegir durants aquestes pràctiques i poder portar-lo a alguna xerrada o algun institut per exposar-lo quan la situació de la pandèmia

s'hagi calmat una mica més.

Per acabar, m'agradaria agraïar als nostres tutors Bruno, Carles i Muntsa per guiar-nos al llarg d'aquest projecte en el que hem après tant. També als meus companys Marina i Lluc que estaven portant el seu projecte paral·lelament i amb els que ens hem estat ajudant durant el desenvolupament del programa.

References

[1] IBM Quantum Composer, https://quantum-computing.ibm.com/composer

[2] Qiskit textbook, https://qiskit.org/textbook/preface.html

[3] Qiskit documentation, https://sooluthomas.github.io/testTranslation/

[4] Kivy documentation, https://kivy.org/doc/stable/

[5] Michael A. Nielsen and Isaac L. Chuang, *Quantum Computation and Quantum Information*, New York, 2000

[6] Qiskit Course *Introduction to Quantum Computing and Quantum Hardware*, https://qiskit.org/learn/intro-qc-qh