

EXchange  
tensions

COINsolidation.

Tallinn, Estonia. (E-residency)

# White Paper.

versión 1.0.0

April 2021.

COINsolidation.org es una marca registrada de COINsolidation International, bajo licencia de uso libre y comercial. Términos y condiciones de uso en: [www.Coinsolidation.org](http://www.Coinsolidation.org)

COINsolidation International se fusiono con [www.OpenQbit.com](http://www.OpenQbit.com) para la cooperación de tecnología basada en mecánica cuántica (Seguridad Cuántica & Computación Cuántica) esta fusión permite usar, compartir y realizar reingeniería de la tecnología desarrollada por OpenQbit Inc. (Estonia, E-residency)

## Contenido

1. Introducción.....	3
2. Seguridad Computación cuántica.....	16
3. Creación de dispositivo “Hardware” de un QRNG (Quantum Random Number Generator). ....	21
4. ¿Qué es Prueba de Cuanto (PQu - Proof Quantum)? .....	27
5. Algoritmo para la creación de una dirección universal consolidada (CUA). ....	30
Proyecto y solución por COINsolidation. ....	34
6. Creación de App CUA (Consolidated Universal Address) en 15 minutos. ....	35
7. Crear tu Exchange de criptomonedas Ethereum en Android solo en 15 minutos. ....	39
8. Roadmap COINsolidation. ....	42
9. COINsolidation Token (CUAG) – ICO DISTRIBUTION PLAN. ....	43
10. Características generales del token COINsolidation: .....	44
11. Conceptos básicos aplicados en plataformas de Blockchain. ....	45
12. ¿Qué es la programación Blockly? .....	48
13. Anexo “Código para algoritmo CUA” .....	48
14. Términos. ....	48

## 1. Introducción.

### COINsolidation

#### Algoritmos oneCKey (one Consolidated private Key) y CUA (Consolidated Universal Address).

By Guillermo Vidal [vidal@coinsolidation.org](mailto:vidal@coinsolidation.org)  
[www.coinsolidation.org](http://www.coinsolidation.org) / [www.coinsolidation.io](http://www.coinsolidation.io)

**Resumen:** Un algoritmo para consolidar la llave privada de direcciones de diferentes blockchains, utilizadas en la actualidad en el área financiera (criptomonedas), aplicaciones y sectores diversos. Proponemos dos algoritmos el primero que llamamos **oneCKey (one Consolidated private Key)** se aplique para consolidar llaves privadas (Una llave privada para N direcciones de diferentes tecnológicas blockchain). Lo anterior se usa para crear sistemas más eficientes en administración de llaves privadas. Este sistema se en conjuntos de series para ser, la primera Serie A1 o serie "Génesis" aplicamos oneCKey en los blockchains Bitcoin y Ethereum una relación de 1:2 (una llave privada para generar dos direcciones) y posteriormente tendremos Series AX y BX donde habrá relaciones de 1: N (una llave privada para N direcciones). Usamos otro algoritmo llamado **CUA (Consolidated Universal Address)** para la creación de direcciones universales que llamaremos direcciones consolidadas. El algoritmo CUA se desglosa en tres tipos de consolidación de direcciones las cuales son; **CUA** "Consolidated Universal Address" una dirección formada de Criptomoneda con Criptomoneda, **DAC** "Direct Address Consolidated" una dirección formada de Token con Token y la **HAC** "Hibric Address Consolidated" una dirección formada de Token con Criptomoneda.

#### INTRODUCCION.

Actualmente 2021 se tiene una tendencia al uso de sistemas blockchain en diferentes sectores como financiero (criptomonedas) y emergentes tanto públicos como privados. En las diferentes tecnologías de blockchain se manejan la generación de sus propias direcciones de uso común para las operaciones de depósitos de sus activos correspondientes y cada dirección administra sus activos de forma independiente por medio de su llave privada única referenciada a una sola dirección de depósito pública, en la actualidad se tiene solamente una relación 1:1 (una dirección para depósito

referenciada a una sola y única llave privada para transferencias de activos) por lo que la administración de varias direcciones se ve complicada al tener siempre una cantidad de N-direcciones para depósito de activos referenciadas a N-llaves privadas de transferencia de activos, una relación N:N. Donde N es la cantidad de direcciones creadas por un usuario en diferentes criptomonedas, siempre se cumple esta relación de N: N.

Por otra parte, también consideremos que la administración se complica para las llaves privadas de igual forma con la relación N: N, es decir un usuario que tenga N direcciones de depósito tendrá de manera directamente proporcional N llaves privadas para administrar. Al aplicar el algoritmo oneCKey se creará una relación de 1: N, una relación de una llave privada para varias direcciones de depósito de diferentes tecnologías blockchain (consolidamos las llaves privadas a solo una que administrará N direcciones de depósito). Por ultimo también se tiene la necesidad de que cada usuario al querer desear realizar un deposito o una transferencia de activos necesita saber la dirección destino, la propuesta es aplicar el segundo algoritmo llamado CUA – “Consolidated Universal Address”, una dirección que consolida dos o más tecnologías blockchain dando como resultado una sola dirección.

Todo lo anterior debe estar funcionando con los sistemas de criptografía que actualmente usa cada criptomoneda, al consolidar la llave primaria se utiliza la validación y la criptografía actual de cada blockchain nativo de la criptomoneda a usar, así cuando se desarrolle cada serie con relación 1: N+1 se aplicara los mismos protocolos de criptografía y hash actuales, esto es consolidación en seguridad.

Después de realizar un estudio sobre el basto mundo de las criptomonedas más importantes y sobresalientes del mercado, hemos encontrado que hay dos tendencias fundamentales para la generación de direcciones y llaves privadas. Las dos curvas de criptografía usado tipo ECC (*Elliptic curve cryptography*) que cubren el 95% de las criptomonedas actuales “más sobresalientes”, son: I. La Curva criptográfica secp256k1 usado por Bitcoin y Ethereum. (**Series A**).

II. La Curva criptográfica Ed25519 tiene una tendencia a ser usada Es una de las curvas ECC más rápidas y no está cubierta por ninguna *patente*. (**Series B**).

Ejemplo:

Name	Type	Signing alg	Curve	Hash	Address encoding	Address hash
<b>Bitcoin</b>	UTXO	ECDSA	secp256k1	SHA-256	base58, bech32	SHA-256, RIPEMD-160
<b>Ethereum</b>	account	ECDSA	secp256k1	Keccak-256 *	none (just hex) *	last 20B of Keccak-256 *
<b>XRP</b>	account	ECDSA *	secp256k1 *	first half of SHA-512	base58 with different alphabet *	SHA-256, RIPEMD-160
<b>Litecoin</b>	UTXO	ECDSA	secp256k1	SHA-256 *	base58, bech32	SHA-256, RIPEMD-160
<b>EOS</b>	account	ECDSA	secp256k1	SHA-256	none *	none *
<b>Bitcoin Cash</b>	Same as Bitcoin *					
<b>Stellar</b>	account	EdDSA	ed25519	SHA-256 and SHA-512 in EdDSA *	base32	none
<b>Binance Coin</b>	Ethereum ERC-20 token *					
<b>Tether</b>	Bitcoin Omni layer / Ethereum ERC-20 token					
<b>TRON</b>	UTXO	ECDSA	secp256k1	SHA-256	base58	last 20 bytes of Keccak-256 *
<b>Cardano</b>	UTXO	EdDSA	ed25519	none and SHA-512 in EdDSA *	base58	none
<b>Monero</b>	UTXO *	<i>it's complicated*</i>	ed25519	Keccak-256 *	base58	Keccak-256 *
<b>IOTA</b>	UTXO	Winternitz one time signature scheme	-	Curl, Kerl *	none	Kerl
<b>Dash</b>	UTXO	ECDSA	secp256k1	SHA-256 *	base58	SHA-256, RIPEMD-160
<b>Maker</b>	Ethereum ERC-20 token					
<b>NEO</b>	account	ECDSA	secp256r1	SHA-256	base58	SHA-256, RIPEMD-160
<b>Ontology</b>	account	ECDSA	nist256p1	3x SHA-256	base58	SHA-256, RIPEMD-160
<b>Ethereum Classic</b>	Same as Ethereum					
<b>NEM</b>	account	EdDSA	ed25519	none and Keccak-256 in EdDSA *	base32	Keccak-256, RIPEMD-160
<b>Zcash</b>	UTXO	ECDSA, zk-SNARKs *	secp256k1, Jubjub *	SHA-256	base58, bech32	SHA-256, RIPEMD-160
<b>Tezos</b>	account	EdDSA, ECDSA *	ed25519, secp256k1, secp256r1	BLAKE2 and SHA-512 in EdDSA *	base58	BLAKE2

## Criptografía para Serie A o Serie “Génesis”.

La serie A o serie “Génesis” es la primera consolidación de llaves privadas. Esta serie está formada por la consolidación de las 2 principales criptomonedas y 2 Tokens (Bitcoin, Ethereum, OAP y COINsolidation).

La criptografía aplicada en ambos casos es ECDSA (*Elliptic Curve Digital Secure Algorithm*) que es la que utilizan actualmente en su generación de llaves públicas y privadas tanto Bitcoin como Ethereum.

La generación de la llave privada consolidada integrada por ambos blockchain Bitcoin y Ethereum se puede aplicar en los siguientes casos (3 opciones).

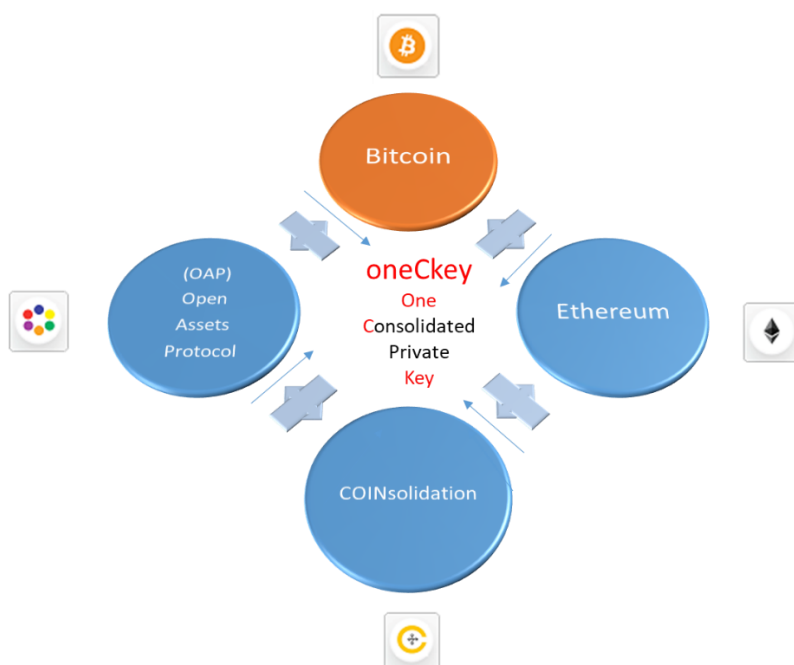
1.- Generación de una llave privada consolidada a partir de QRNG (*Quantum Random Number Generator*)



Generamos un numero aleatorio de una fuente QRNG, proponemos la generación del uso mecánica óptica cuántico mediante una foto con formato RAW para producir la

suficiente entropía. Esto se realiza usando el lente del Smartphone, aplicamos hash (SHA254) y hacemos la conversión a hexadecimal para crear la llave privada consolidada, esta la aplicamos el algoritmo oneCKey-secp256k1 para generar las respectivas direcciones de depósito en cada blockchain Bitcoin y Ethereum. Aplicamos a la dirección Bitcoin formato del protocolo OAP (Open Assets Protocol) para generar un ExoToken. Usando oneCKey nos dará una sola llave privada para dos diferentes direcciones de depósito de diferentes blockchain como son Bitcoin y Ethereum, así como la generación de un Token OAP (Open Assets Protocol) para ser usado por el usuario según sus intereses, abriendo una posibilidad de ampliar sus perspectivas de negocios.

2.- Generación de una llave privada consolidada a partir de una dirección de Bitcoin existente. Se aplica la llave privada Bitcoin al algoritmo oneCKey-secp256k1 para generar una dirección Ethereum y aplicamos a dirección Bitcoin formato del protocolo OAP (Open Assets Protocol) para generar ExoToken.



Ejemplo:

Llave privada Bitcoin existente (en esta aplicamos oneCKey para crear otras direcciones):

**28a0f97c6921e43872eb0640af41a54b9bde57c71cf4efe0db9d829f8b2cf645**

Dirección Bitcoin:

**18XmTwfTeurjKQ8i1rEQT1DAx8BDjdR96A**

Dirección Ethereum:

**0x9c789b22758c85f456dca3ac02e1fb00a059a4e**

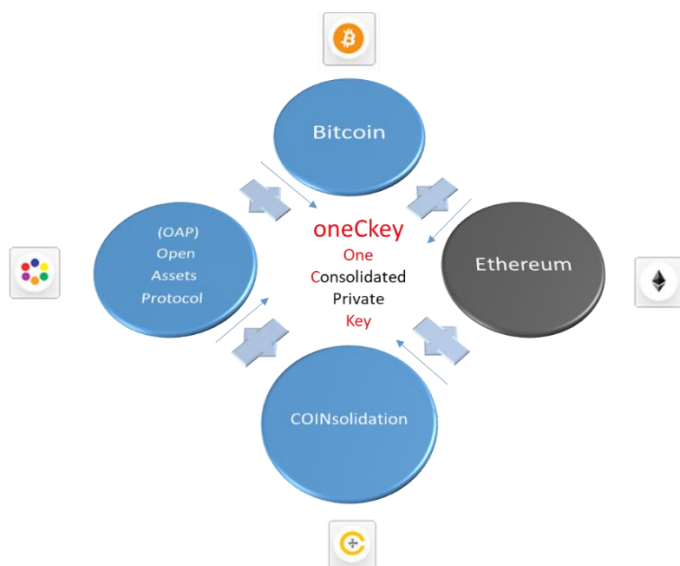
Dirección OAP (Open Assest Protocol) basada en la anterior dirección Bitcoin.

**akjVei7Uo8PkRBeJ54ULb6s7kHjMPhs8UjG**

Dirección Token COINsolidation:

**0x9d08c0ac0f2fdf078c883db6fa617b15776e4b41**

3.- Generación de una llave privada consolidada a partir de una dirección de Ethereum existente. Se aplica la llave privada Ethereum al algoritmo oneCKey-secp256k1 para generar dirección Bitcoin y aplicamos a dirección Bitcoin formato del protocolo OAP (Open Assets Protocol) para generar ExoToken.





## Criptografía para Serie A1.

Donde interviene un total de 5 criptomonedas y 2 Tokens.



En este caso tendremos 5 opciones para crear la oneCKey, podemos usar cada llave privada ya existente de las diferentes criptomonedas soportadas.

## Criptografía para Serie A2.

Ejemplo: Si tomamos la llave privada de DASH aplicando oneCKey generaremos 5 direcciones basadas en la llave privada de DASH y los dos Tokens también dependerán de esta. Una relación de 1:7 ( 1 llave privada consolidada para 7 direcciones).

Llave privada de DASH (aplicamos oneCKey para crear otras direcciones).

**20ba723de1fdeee66e927e30fdb3ada74ae23bfdb370da378f301ce4dbf27312**

Dirección DASH:

**XtAGtBbnzykDSwoWUSjgQUF4gG1h3uyYKW**

Dirección Bitcoin:

**1JUS3vwu3GXdJ1CvcZRTYwZGqvRzwT7FEk**

Dirección Ethereum:

**0x9d4a5854955c8e498e61eaaae7d3846917381f5b**

Dirección OAP:

**akUSKJ6mEWkRKAFNHfBXeCoTrBXcAyavXnS**

Dirección COINsolidation Token:

**0x9d08c0ac0f2fdf078c883db6fa617b15776e4b41**

Dirección Litecoin

**LchPK9Fj7vmgYou5nhQkpxd348oHAJ3aSu**

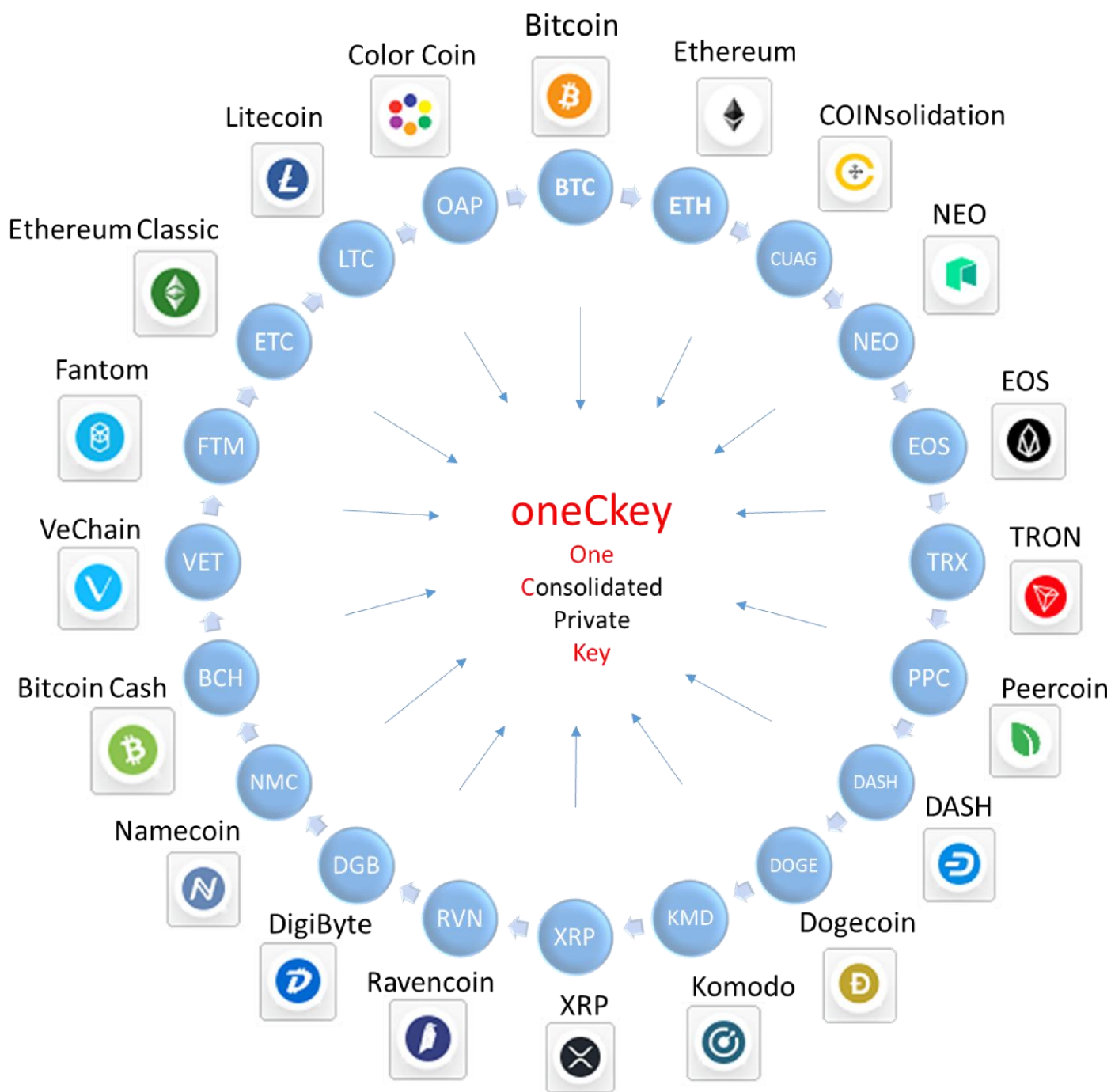
Dirección Dogecoin:

**DNcXbBtYLgRuq1PXM9R26hisj4AJGJj4pT**

Lo anterior podemos aplicarlo de la misma forma para cualquier dirección ya existente de las criptomonedas antes mencionadas y soportadas por oneCKEY en la serie B. Nos da como resultado una llave privada para usarla en 5 criptomonedas y 2 Tokens.

## Criptografía para Serie A3.

La serie A3 se aplica a 18 criptomonedas y 2 Tokens. Lo que sería una relación 1: 20 (una llave privada consolidada para uso común en 20 direcciones).



Lo anterior podemos aplicarlo de la misma forma para cualquier dirección ya existente de las criptomonedas antes mencionadas y soportadas por oneCKey en la serie A3.

Ejemplo: A la llave privada de Bitcoin aplicando oneCKey generaremos 18 direcciones con una sola llave privada y dos Tokens de esta. Una relación de 1:20 (una llave privada consolidada para 20 direcciones).

Llave privada de Bitcoin (aplicamos oneCKey para crear otras direcciones).

**d2783ceae51a0c74fb733640f7128092dba3093605232b2b76fb102dca092d69**

1. Dirección Bitcoin:  
1N67xWPN9F1sEMXparSpzT24erp5F2p6LK <https://www.blockchain.com/explorer>
2. Dirección Ethereum:  
0x91efb31bcd0bd12a088f9625344bbe92c1543bc3 <https://etherscan.io/>
3. Dirección COINsolidation:  
0x9d08c0ac0f2fdf078c883db6fa617b15776e4b41 <https://etherscan.io/>
4. Dirección Wallet EOS:  
5KQyKH8ssQkwsmUWffvrbsMRf5wh8iHofZGvkHou5h1HYCqJYp
5. Dirección TRON:  
TPGr8Vp78vNcfhSvnpp7EnucaBKEBXns7h (The account is not activated.)  
<https://trx.tokenview.com/>
6. Dirección Peercoin:  
PVgJ7UnDCAW4DCCavvmMfLzLGbyxJ4HRpP <https://blockbook.peercoin.net/>
7. Dirección DASH:  
Xwmxnm3G6xETPJ8QJm3qyhrVCPmECp3dt <https://explorer.dash.org/insight/>
8. Dirección Dogecoin:  
DSEDVmL1Sev9mMiRKSSPYDBfXzYNYadkdV <https://blockchair.com/dogecoin>
9. Dirección Komodo:  
RWNK32Gek4pSJMmu242Rx5yMGR8GfprvY69 <https://kmdexplorer.io/>
10. Dirección XRP:  
r4afxWP49Er1NMXF2iSFzTpheiFnEpFaLK (The account is not activated.)  
<https://bithomp.com/explorer/>
11. Dirección Ravencoin:  
RWNK32Gek4pSJMmu242Rx5yMGR8GfprvY69  
<https://ravencoin.network/>

12. Dirección DigiByte:  
DSEDVmL1Sev9mMiRKSSPYDBFxzYNYadkdV <https://digibyteblockexplorer.com/>
13. Dirección Namecoin:  
NHfVA9tM4d7RktnKrfmQCyAyP6D8AQ2rA5  
<https://www.cryptoground.com/namecoin-block-explorer>
14. Dirección Bitcoin Cash:  
1N67xWPN9F1sEMXparSpzT24erp5F2p6LK (New format to see explorer site)  
<https://explorer.bitcoin.com/bch> New format address:  
bitcoincash:qrn49p6cy49tzgatt9gz9xz3gf8qxl9tu800hul9m
15. Dirección VeChain:  
0x91efb31bcd0bd12a088f9625344bbe92c1543bc3 <https://explore.vechain.org>
16. Dirección Fantom:  
0x91efb31bcd0bd12a088f9625344bbe92c1543bc3 <https://explorer.fantom.network/>
17. Dirección Ethereum Classic:  
0x91efb31bcd0bd12a088f9625344bbe92c1543bc3 <https://etcblockexplorer.com/>
18. Dirección Litecoin:  
LgK5DihCDuFvVADyKzS8GU5ps5BMGWfCXX <https://blockchair.com/litecoin>
19. Dirección Color coin:  
akY41CgChciuZ6bhBdUZ1eJvdzTzF9Sizv3 <https://blockchainexplorer.lykke.com/>
20. WIF (Wallet Import Format) - NEO use secp256r1, Do not use secp256k1).  
Para obtener dirección importar WIF sitio: <https://neotracker.io/wallet/open-wallet> usa WIF como Private Key.  
  
WIF (Wallet Import Format) NEO:  
  
L4GqU2DmXVryD5JRwgphv6yF9q6pQ93FAyBTJE5BzshaFnwXPB4H  
Dirección NEO:  
AKkcm37QeMxZpFfRrrLGeqFRgkFzMPH9Gn  
  
<https://neotracker.io/>
21. Dirección ExoCrypto:  
Exo41CgChciuZ6bhBdUZ1eJvdzTzF9Sizv3 <https://exoCrypto.com>

Resumen de cryptomonedas soportadas y clasificadas por las dos curvas ECC.  
(secp256k1 & Ed25519)

Serie B1 (Ed25519) – Q42021

1. Polkadot
2. Cardano
3. Monero
4. COINsolidation\*\*\*
5. Stellar
6. Algorand
7. IOTA
8. Elrond
9. Decred
10. Nano
11. Horizen
12. Siacoin
13. Stacks
14. Lisk
15. Qtum
16. Waves 17. ....Others.

Serie A3 (secp256k1) – Ready

1. Bitcoin\*
2. OAP\*\*\*
3. Ethereum\*\*
4. COINsolidation\*\*\*
5. XRP
6. NEO
7. EOS
8. TRON
9. Dash
10. Bitcoin Cash\*
11. Ethereum Classic\*\*
12. Litecoin
13. Dogecoin
14. VeChain\*\*
15. DigiByte
16. Fantom\*\*
17. Ravencoin
18. Komodo
19. Peercoin
20. Namecoin

(\*) Criptomonedas Bitcoin con la misma dirección ya que el algoritmo para generarlas es el mismo únicamente cambia el blockchain que es diferentes para las transacciones de cada una.

(\*\*) Criptomonedas Ethereum con la misma dirección ya que el algoritmo para generarlas es el mismo únicamente cambia el blockchain que es diferentes para las transacciones de cada una.

(\*\*\*) Tokens generados y dependientes de la llave primaria consolidada.

Roadmap Serie A4 incluye Serie A3 + Zcash, Binance BNB, Filecoin, Tether, Tezos, Cosmos, Zilliqa y Avalanche. Q4-2021

Roadmap inicia Serie B1 con Curve Ed25519. (Polkadot, Cardano, Monero, Stellar, Algorand, IOTA, Elrond, Algorand, y Waves) liberación Q42021.

**NOTA:** Pueden variar los crypto-activos de cada serie dependiendo del comportamiento del mercado de criptomonedas puede la lista ser modificada, según la liquidez de cada activo.

Seguridad implementada en oneCKey, algoritmos PQC - (Post-Quantum Cryptography) embebido en el repositorio donde se almacena la llave consolidada generada de forma local en cada Smartphone o en su caso una llave ya existente introducida por el usuario.

Para la codificación segura de la oneCKey hemos implementado una combinación de algoritmos de seguridad PQC compuesta con: AES-CGM + chacha20poly1305.

Chacha20poly1305: <https://tools.ietf.org/html/rfc7539>

AES-CGM: <https://tools.ietf.org/html/rfc5288>

U = operador de aplicación de algoritmos PQC.

Cifrado de oneCKey = U (AES-CGM ( chacha20poly1305 (OneCKey)))

## 2. Seguridad Computación cuántica.

### ¿Cómo funciona la computación cuántica? <sup>(2)</sup>

La transformación digital está generando cambios en el mundo más rápido que nunca. ¿Creerías que la era digital está por acabarse? Ya se ha señalado la **alfabetización digital** como un área donde el conocimiento abierto y las oportunidades accesibles para aprender sobre la tecnología son urgentes para abordar las brechas en el desarrollo social y económico. Aprender de los conceptos claves de la era digital se volverá aún más crítico ante la inminente llegada de otra nueva ola tecnológica capaz de transformar los modelos existentes con una velocidad y potencia asombrosa: las **tecnologías cuánticas**.

En este artículo, comparamos los conceptos básicos de la computación tradicional y la computación cuántica; y también comenzamos a explorar su aplicación en otras áreas relacionadas.

¿Qué son las tecnologías cuánticas?

A lo largo de la historia, el ser humano ha ido desarrollando tecnología a medida que ha ido entendiendo el funcionamiento de la naturaleza a través de la ciencia. Entre los años 1900 y 1930, el estudio de algunos fenómenos físicos que aún no estaban bien entendidos dio lugar a una nueva teoría física, la **Mecánica Cuántica**. Esta teoría describe y explica el funcionamiento del mundo microscópico, hábitat natural de moléculas, átomos o electrones. Gracias a ella no solo se ha conseguido explicar esos fenómenos, sino que ha sido posible entender que la realidad subatómica funciona de forma completamente contra intuitiva, casi mágica, y que en el mundo microscópico tienen lugar sucesos que no ocurren en el mundo macroscópico.

Entre estas **propiedades cuánticas** se incluyen la superposición cuántica, el entrelazamiento cuántico y el teletransporte cuántico.


- La **superposición cuántica** describe cómo una partícula puede estar en diferentes estados a la vez.
- El **entrelazamiento cuántico** describe cómo dos partículas tan separadas como se desee pueden estar correlacionadas de forma que, al interactuar con una, la otra se entera.
- El **teletransporte cuántico** utiliza el entrelazamiento cuántico para enviar información de un lugar a otro del espacio sin necesidad de viajar a través de él.

Las tecnologías cuánticas son basadas en estas propiedades cuánticas de la naturaleza subatómica.



En este caso, hoy en día el entendimiento del mundo microscópico a través de la Mecánica Cuántica nos permite inventar y diseñar tecnologías capaces de mejorar la vida de las personas. Hay muchas y muy diferentes tecnologías que utilizan fenómenos cuánticos y, algunas de ellas, como el láser o las imágenes por resonancia magnética (IRM), llevan ya entre nosotros más de medio siglo. Sin embargo, actualmente estamos presenciando una revolución tecnológica en áreas como la computación cuántica, la información cuántica, la simulación cuántica, la óptica cuántica, la metrología cuántica, los relojes cuánticos o los sensores cuánticos.

¿Qué es la computación cuántica? Primero, hay que entender la computación clásica.



**FIGURA 1.**  
Ejemplos de caracteres en lenguaje binario.

Caracter	Bits
7	111
A	01000001
\$	00100100
:)	0011101000101001

Para entender cómo funcionan los computadores cuánticos es conveniente explicar primero cómo funcionan los computadores que utilizamos a diario, a los que nos referiremos en este documento como computadores digitales o clásicos. Estos, al igual que el resto de los dispositivos electrónicos como tabletas o teléfonos móviles, utilizan bits como unidades fundamentales de memoria. Esto significa que los programas y aplicaciones están codificados en bits, es decir, en lenguaje binario de ceros y unos. Cada vez que interactuamos con cualquiera de estos dispositivos, por ejemplo, pulsando una tecla del teclado, se crean, destruyen y/o modifican cadenas de ceros y unos dentro de la computadora.

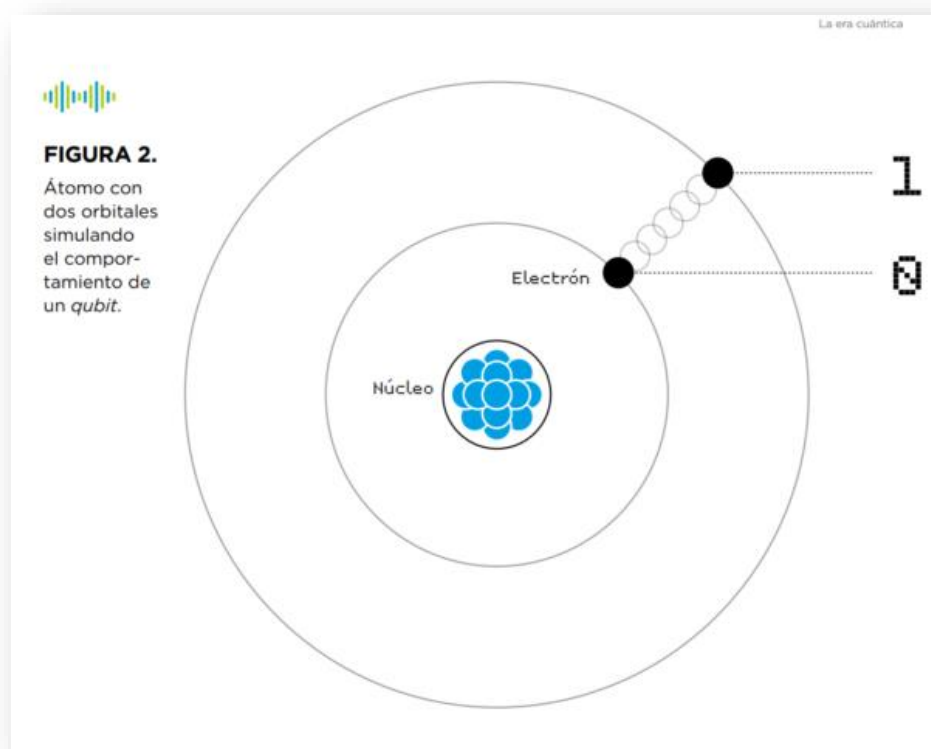
La pregunta interesante es, ¿qué son físicamente estos ceros y unos dentro de la computadora? Los estados cero y uno de los bits se corresponden con corriente eléctrica que circula, o no, a través de unas piezas microscópicas denominadas transistores, que actúan como interruptores. Cuando no circula corriente, el transistor está “apagado” y se corresponde con un bit 0, y cuando circula está “encendido” y se corresponde con un bit 1.

De forma más simplificada, es como si los bits 0 y 1 se correspondiesen con huecos, de manera que un hueco vacío es un bit 0 y un hueco ocupado por un electrón es un bit 1. Es por este motivo que estos dispositivos se llaman electrónicos. A modo de ejemplo, en la

figura 1 se muestra la escritura en lenguaje binario de algunos caracteres. Ahora que tenemos una idea de cómo funcionan los computadores actuales, tratemos de entender cómo funcionan los cuánticos.

### De los bits a qubits

La unidad fundamental de información en computación cuántica es el quantum bit o qubit. Los qubits son, por definición, sistemas cuánticos de dos niveles -ahora veremos ejemplos- que al igual que los bits pueden estar en el nivel bajo, que se corresponde con un estado de baja excitación o energía definido como 0, o en el nivel alto, que se corresponde con un estado de mayor excitación o definido como 1. Sin embargo, y aquí radica la diferencia fundamental con la computación clásica, los qubits también pueden estar en cualquiera de los infinitos estados intermedios entre el 0 y el 1, como por ejemplo un estado que sea mitad 0 y mitad 1, o tres cuartos de 0 y un cuarto de 1. Este fenómeno se conoce como superposición cuántica y es natural en sistemas cuánticos.



Algoritmos cuánticos, computación exponencialmente más poderosa y eficiente

El propósito de los computadores cuánticos es aprovechar estas propiedades cuánticas de los *qubits*, como sistemas cuánticos que son, para poder correr algoritmos cuánticos que utilizan la superposición y el entrelazamiento para ofrecer una capacidad de procesamiento mucho mayor que los clásicos. Es importante indicar que el verdadero cambio de paradigma

no consiste en hacer lo mismo que hacen las computadoras digitales o clásicas -las actuales-, pero más rápido, como de forma errónea se puede leer en muchos artículos, sino que los algoritmos cuánticos permiten realizar ciertas operaciones de una manera totalmente diferente que en muchos casos resulta ser más eficiente -es decir, en mucho menos tiempo o utilizando muchos menos recursos computacionales-.

Veamos un ejemplo concreto de lo que esto implica. Imaginemos que estamos en Bogotá y queremos saber cuál es la mejor ruta para llegar a Lima de entre un millón de opciones para llegar ( $N=1.000.000$ ). De cara a poder utilizar computadoras para encontrar el camino óptimo necesitamos digitalizar 1.000.000 opciones, lo que implica traducirlas a lenguaje de bits para el computador clásico y a *qubits* para el computador cuántico. Mientras que una computadora clásica necesitaría ir uno por uno analizando todos los caminos hasta encontrar el deseado, una computadora cuántica se aprovecha del proceso conocido como paralelismo cuántico que le permite considerar todos los caminos a la vez. Esto implica que, si bien la computadora clásica necesita del orden de  $N/2$  pasos o iteraciones, es decir, 500.000 intentos, la computadora cuántica encontrará la ruta óptima tras solo  $\sqrt{N}$  operaciones sobre el registro, es decir, 1.000 intentos.

En el caso anterior la ventaja es cuadrática, pero en otros casos es incluso exponencial, lo que significa que con  $n$  *qubits* podemos obtener una capacidad computacional equivalente a  $2^n$  bits. Para ejemplificar esto, es frecuente contar que con unos 270 qubits se podrían tener más estados base en un computador cuántico -más cadenas de caracteres diferentes y simultáneas- que el número de átomos en el universo, que se estima en torno a  $10^{80}$ . Otro ejemplo, es que se estima que con un computador cuántico de entre 2000 y 2500 *qubits* se podría romper prácticamente toda la criptografía utilizada hoy en día (la conocida como criptografía de clave pública).

¿Por qué es importante saber sobre la tecnología cuántica?

Estamos en un momento de transformación digital en el que distintas tecnologías emergentes como blockchain, inteligencia artificial, drones, Internet de las cosas, realidad virtual, 5G, impresoras 3D, robots o vehículos autónomos tienen cada vez más presencia en múltiples ámbitos y sectores. Estas tecnologías, llamadas a mejorar la calidad de vida del ser humano acelerando el desarrollo y generando impacto social, avanzan hoy en día de manera paralela. Solo en contadas ocasiones vemos compañías desarrollando productos que exploten combinaciones de dos o más de estas tecnologías, como blockchain y IoT o drones e inteligencia artificial. Si bien están destinadas a converger generando así un impacto exponencialmente mayor, la etapa inicial de desarrollo en que se encuentran y la escasez de desarrolladores y personas con perfiles técnicos hacen que las convergencias sean aún una tarea pendiente.

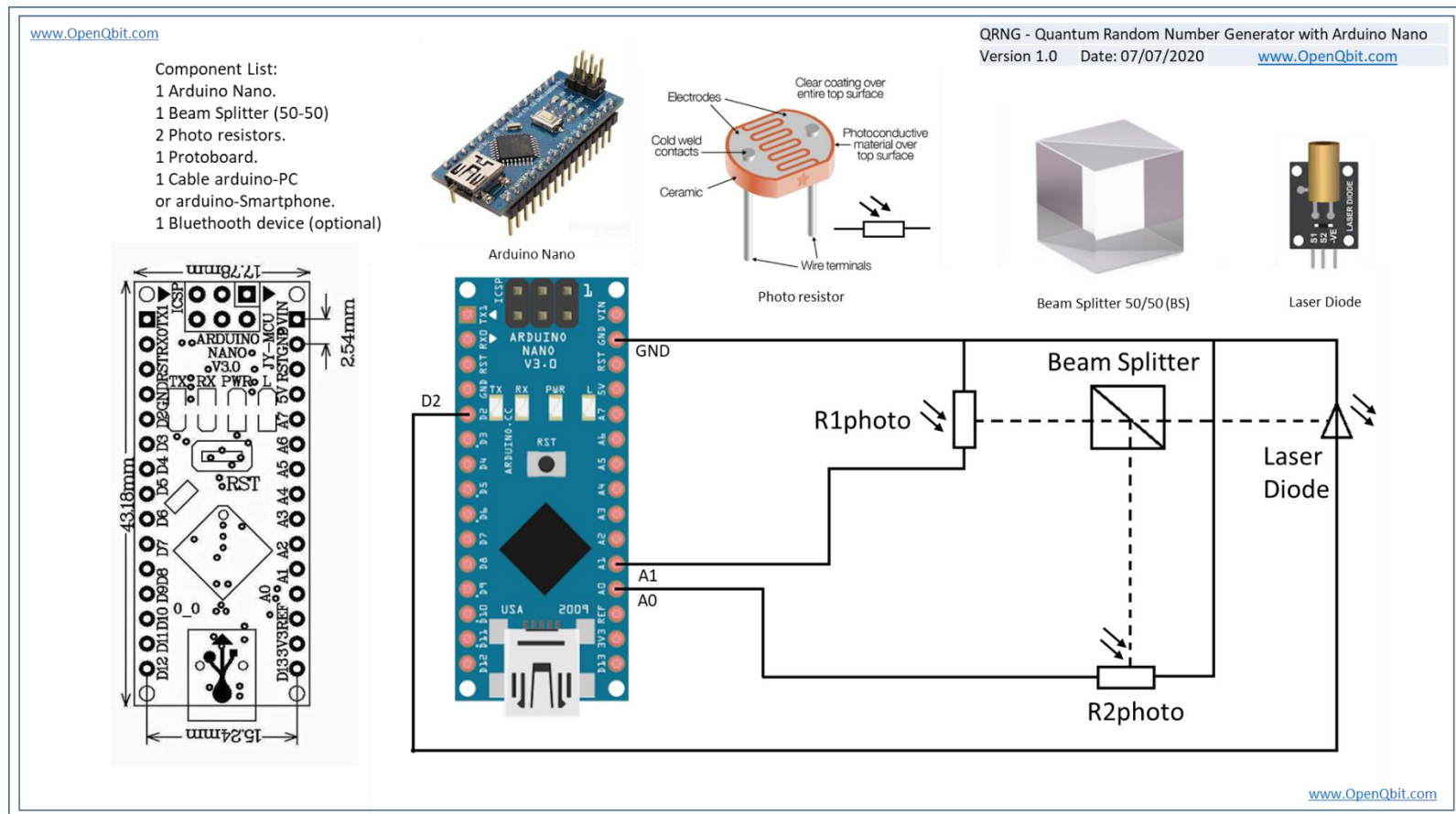
De las tecnologías cuánticas, por su potencial disruptivo, se espera que no solo converjan con todas estas nuevas tecnologías, sino que tengan una influencia transversal en prácticamente la totalidad de ellas. La computación cuántica amenazará la autenticación, intercambio y almacenamiento seguro de datos, teniendo un impacto mayor en aquellas tecnologías en las que la criptografía tiene un rol más relevante, como ciberseguridad o blockchain, y un impacto negativo menor pero también a considerar en tecnologías como 5G, IoT o drones.

### **¿Quieres practicar la computación cuántica?**

En la red hay ya disponibles decenas de simuladores de computadores cuánticos con diferentes lenguajes de programación ya existentes como C, C++, Java, Matlab, Máxima, Python o Octave. También, lenguajes nuevos como Q#, lanzado por Microsoft. Se puede explorar y jugar con una maquina cuántica virtual a través de plataformas como la de IBM y la de Rigetti.

### 3. Creación de dispositivo “Hardware” de un QRNG (Quantum Random Number Generator).

Ahora crearemos un dispositivo físico “Hardware” para generar números aleatorios cuánticos (QRNG) con componentes económicos y que se puedan armar fácilmente en casa y bajo costo aproximadamente \$35 USD.



## QRNGv1.0.ino

Software  
Program to arduino nano.

```
/* OpenQbitQRNG Firmware V1.0
 *Author: Guillermo Vidal
 *Copyright © 2020 OpenQbit, Inc.
 *License: MIT
 */
```

```
int triggerQ = 2; // This pin will pulse our quantum circuit
int QuA0Pin = A0; // This pin measures the horizontal polarized photons
int QuA1Pin = A1; // This pin measures the vertically polarized photons
float Qu0 = 0;
float Qu1 = 0;
```

```
void setup() {
  // Just setting up triggerPin and serial connection
  pinMode(triggerQ, OUTPUT); // sets the digital pin 2 as output
  Serial.begin(9600);
}
```

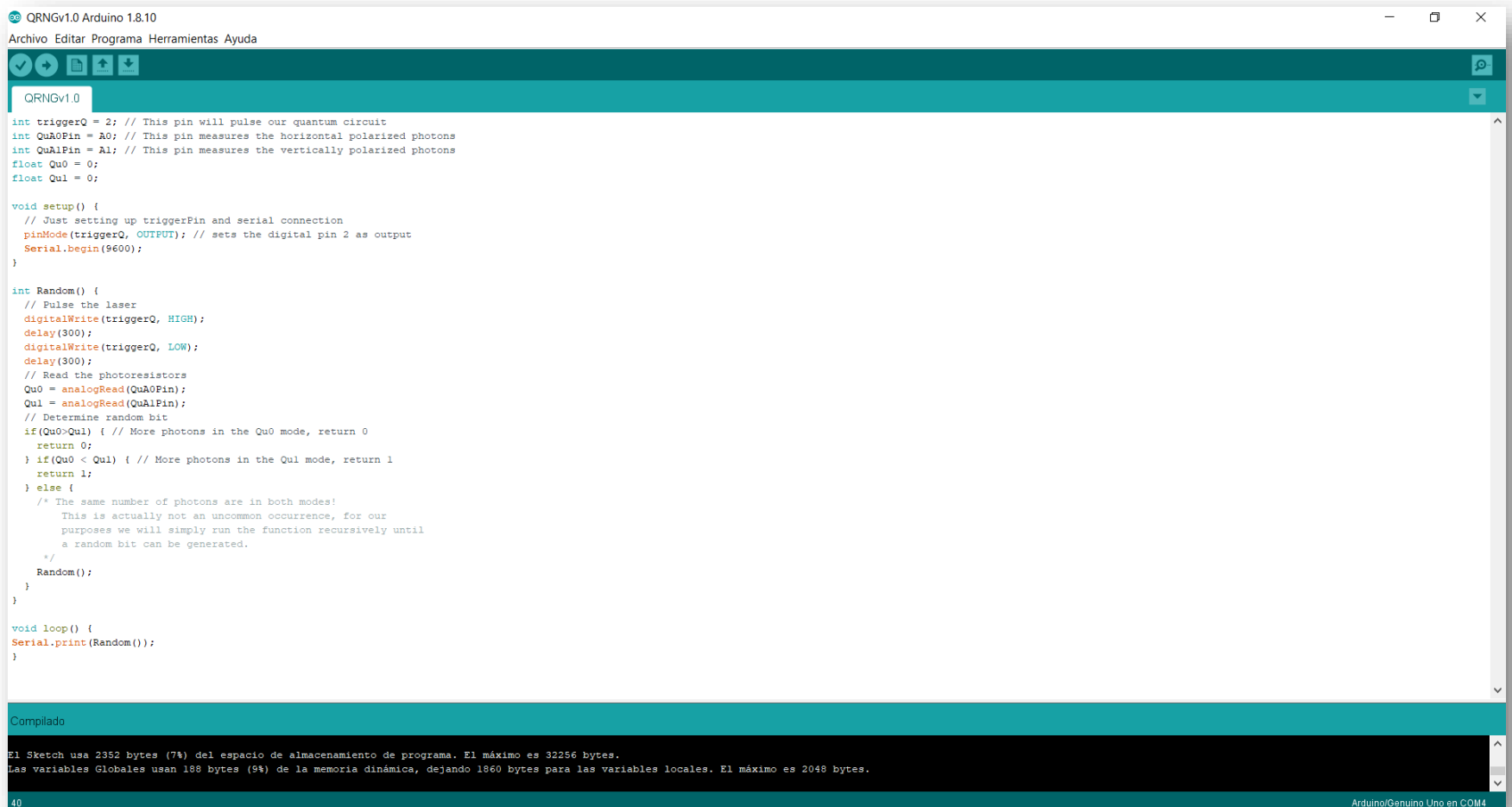
```
int Random() {
  // Pulse the laser
  digitalWrite(triggerQ, HIGH);
  delay(300);
  digitalWrite(triggerQ, LOW);
  delay(300);
  // Read the photoresistors
  Qu0 = analogRead(QuA0Pin);
  Qu1 = analogRead(QuA1Pin);
  // Determine random bit
  if(Qu0>Qu1) { // More photons in the Qu0 mode, return 0
    return 0;
  } if(Qu0 < Qu1) { // More photons in the Qu1 mode, return 1
    return 1;
  } else {
    /* The same number of photons are in both modes!
     This is actually not an uncommon occurrence, for our
     purposes we will simply run the function recursively until
     a random bit can be generated.
    */
    Random();
  }
}
```

```
void loop() {
  Serial.print(Random());
}
```

## Output console

0010110101011110101011010.....

Compilando el programa QRNGv10.ino y subiendo a arduino nano....



The screenshot shows the Arduino IDE interface with the file 'QRNGv1.0' open. The code is written in C++ and implements a quantum random number generator. It uses two analog pins (A0 and A1) to measure horizontal and vertical polarized photons. The setup function initializes the trigger pin (2) as an output and starts the serial connection at 9600 baud. The Random function pulses the laser, reads the photoreistors, and determines a random bit based on the photon counts. The loop function prints the random bit to the serial monitor.

```
QRNGv1.0

int triggerQ = 2; // This pin will pulse our quantum circuit
int QuA0Pin = A0; // This pin measures the horizontal polarized photons
int QuA1Pin = A1; // This pin measures the vertically polarized photons
float Qu0 = 0;
float Qu1 = 0;

void setup() {
  // Just setting up triggerPin and serial connection
  pinMode(triggerQ, OUTPUT); // sets the digital pin 2 as output
  Serial.begin(9600);
}

int Random() {
  // Pulse the laser
  digitalWrite(triggerQ, HIGH);
  delay(300);
  digitalWrite(triggerQ, LOW);
  delay(300);
  // Read the photoreistors
  Qu0 = analogRead(QuA0Pin);
  Qu1 = analogRead(QuA1Pin);
  // Determine random bit
  if(Qu0>Qu1) { // More photons in the Qu0 mode, return 0
    return 0;
  } if(Qu0 < Qu1) { // More photons in the Qu1 mode, return 1
    return 1;
  } else {
    /* The same number of photons are in both modes!
       This is actually not an uncommon occurrence, for our
       purposes we will simply run the function recursively until
       a random bit can be generated.
    */
    Random();
  }
}

void loop() {
  Serial.print(Random());
}
```

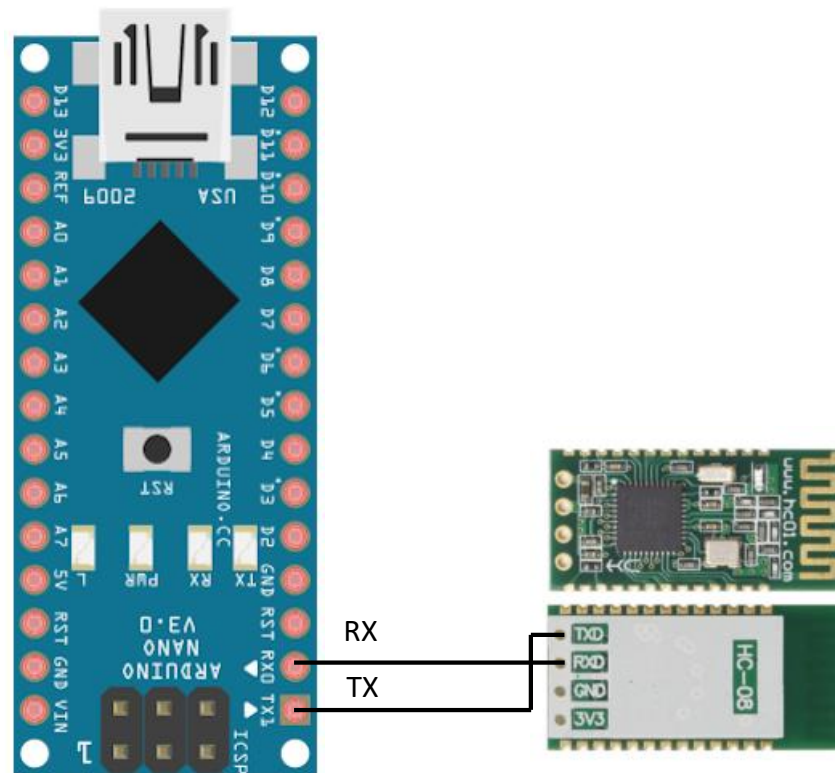
Compilado

El Sketch usa 2352 bytes (7%) del espacio de almacenamiento de programa. El máximo es 32256 bytes.  
Las variables Globales usan 188 bytes (9%) de la memoria dinámica, dejando 1860 bytes para las variables locales. El máximo es 2048 bytes.

40 Arduino/Genuino Uno en COM4

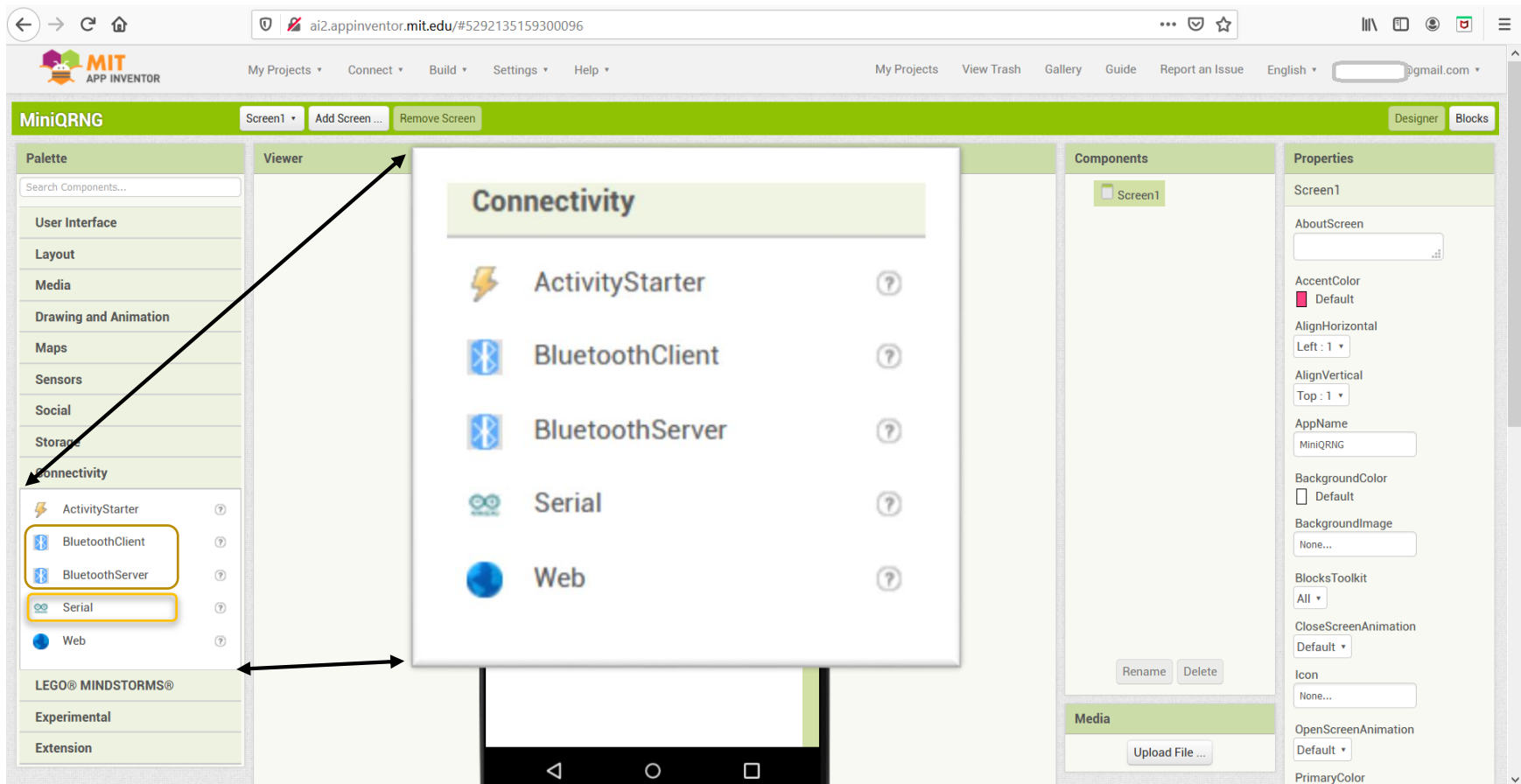
Hay dos formas para comunicarse con el arduino nano, una es mediante el puerto Serial y la otra es a través de una conexión Bluetooth.

Para la conexión bluetooth es muy simple solamente tenemos que comprar el modulo HC-08 o uno similar y conectarlo de la siguiente forma:





Para poder conectar App Inventor con Arduino se puede usar los siguientes componentes Serial o Bluetooth:





Ahora ya compilado y cargado el programa QRNGv10.ino solo falta comunicarnos con el arduino nano para guardar los datos (números aleatorios cuánticos) estos estarán en formato binario, sin embargo, los datos obtenidos se pueden pasar fácilmente a otro formato como el hexadecimal o decimal según sea el requerimiento final.

Por ultimo para ver ejemplo de como son la conexión serial o por medio de bluetooth enseguida se muestran unas ligas de referencia.

Recordemos que todo es a través de programación Blockly para ser probada con App Inventor este ya cuenta con bloques para la comunicación con arduino de forma serial u otro sistema tipo blockly podrá ser a través de bluetooth similar en línea.

[http://kio4.com/appinventor/9A0\\_bluetooth\\_RXTX.htm](http://kio4.com/appinventor/9A0_bluetooth_RXTX.htm)

<http://kio4.com/appinventor/index.htm#bluetooth>

<https://community.appinventor.mit.edu/>

Para revisar todo el proyecto completo de diseño y uso de extensiones de QRNG (Quantum Random Number Generator). Revisar el manual de usuario en:

<https://github.com/COINsolidation/UserGuide>



#### 4. ¿Qué es Prueba de Cuanto (PQu - Proof Quantum)?

PoQu. - “Proof of Quantum” es un algoritmo de consenso desarrollada para Mini BlocklyChain y COINsolidation, esta prueba es una variante de la Prueba de Trabajo (PoW - “Proof of Work”) que funciona de la siguiente forma.

La Prueba de Cuanto (PoQu) al inicio se ejecuta con el mismo algoritmo de la “Prueba de Trabajo” (PoW) se fundamenta en poner a trabajar el procesador del dispositivo (PC, Servidor, Tableta o Teléfono móvil) para obtener una cadena de caracteres que es un acertijo matemático llamado “hash”.

Recordemos que un “hash” es un algoritmo o proceso matemático que al introducir una frase o algún tipo de información digital como archivos de texto, programa, imagen, video, sonido u otro tipo diverso de información digital nos da como resultado un carácter alfanumérico que representa la firma digital que lo representa de forma única y no repetible del dato, el algoritmo de “hash” es unidireccional esto quiere decir que al introducir un dato para obtener su firma “hash” su proceso inverso no podemos realizarlo, al tener una firma “hash” no podemos saber de qué información fue obtenido esta propiedad nos da un ventaja de seguridad para procesar la información que enviamos por internet. ¿Cómo funciona? imaginemos enviar cualquier tipo de información a través de canales no seguros y acompañar dicha información con su respectivo “hash origen”, el receptor al recibir la información podrá sacar el “hash” de la información recibida lo llamaremos “hash destino” y cotejarlo con el “hash origen” si ambos “hashes” son iguales podemos confirmar que la información no ha sido alterada en el canal que se envió, es solo un ejemplo en donde se usa actualmente este tipo de proceso de seguridad de información.

En la actualidad se tiene diferentes tipos de algoritmos o procesos tipo “hash” estos difieren en el nivel de seguridad. Los más utilizados o conocidos son: MD5, SHA256 y SHA512.

Ejemplo de SHA256:

Tenemos una cadena o frase como sigue: “Mini BlocklyChain es modular”

Si aplicamos un “hash” del tipo SHA256 a la cadena anterior nos dará el siguiente “hash”.

**f41af7e61c3b02fdd5e5c612302b62a2dd52fcb38f9de97cb2afd827e8804db8**

La anterior cadena alfanumérica es la firma que representa la frase del ejemplo anterior



Para más ejemplo podemos usar en internet el sitio:

<https://emn178.github.io/online-tools/sha256.html>

En el caso del algoritmo de “Prueba de Trabajo” (PoW) trabaja usando poder de cómputo para poder obtener un “hash” predefinido.

Imaginemos que tenemos el “hash” anterior que sacamos de la cadena “Mini BlocklyChain es modular”.

**f41af7e61c3b02fdd5e5c612302b62a2dd52fcb38f9de97cb2afd827e8804db8**

A este “hash” en su inicio ponemos el parámetro de dificultad que es simplemente poner ceros “0” en el inicio, es decir si decimos que la dificultad es de 4 tendrá “0000” + “hash” a este le llamaremos “hash semilla”

**0000 f41af7e61c3b02fdd5e5c612302b62a2dd52fcb38f9de97cb2afd827e8804db8**

Ahora teniendo en cuenta que sabemos la información de entrada que es la cadena: “Mini BlocklyChain es modular” le agregamos al final de la cadena un número empezando desde cero “0” y sacamos su hash a este lo llamaremos “hash nonce”:

**f41af7e61c3b02fdd5e5c612302b62a2dd52fcb38f9de97cb2afd827e8804db80**

Sacamos “hash nonce”:

**7529f3ad273fc8a9eff12183f8d6f886821900750bb6b59c1504924dfd85a7c8**

Después realizamos una comparación del nuevo “hash nonce” con el “hash semilla” si son iguales el nodo que primero encuentre la igualdad ganará la ejecución de procesar la transacción en curso. Como podemos ver este proceso está basado en probabilidad y fuerza de cómputo que tenga el dispositivo lo que da a la prueba de trabajo “Proof of Work” una equidad de consenso para todos los nodos.

En caso de no coincidir el “hash semilla” con el “hash nonce”, se incrementa la dificultad en uno y se vuelve a sacar el “hash nonce”, el número que se esté incrementando le llamaremos número “nonce”, se compara con el “hash semilla” hasta que coincidan o sean iguales.

Como podemos ver el número “nonce” o incremento es el que ayudara a obtener el “hash” de igualdad.

Teniendo como base el algoritmo de “Prueba de Trabajo” (PoW), el algoritmo Prueba de Cuanto (PoQu) se basa en obtener el número “nonce” como lo hace PoW y usando una



dificultad de nivel mínimo que va desde 1 a 5, este sirve únicamente para que el dispositivo móvil gane el derecho de ser un candidato para ganar el consenso.

La Prueba de Cuanto (PoQu), se activa cuando el teléfono móvil a terminado el PoW mínimo y gana el pase para obtener un numero de probabilidad en el sistema de QRNG.

El QRNG (Quantum Random Number Generator) es un Generador de números cuánticos aleatorios, este sistema está basado en generar verdaderos números aleatorios basados en la mecánica cuántica es el sistema más seguro en la actualidad para generar este tipo de números. Para más detalles ver “Seguridad Computación Cuántica” en el índice 3.

COINsolidation puede implementar los dos tipos de conceso PoW mínimo y PoQu.

La Prueba de Cuanto (PoQu) se basa en obtener el número “nonce” este número en la Prueba de Cuanto (PoQu) es conocido como “Magic Number” con este el sistema “Peer to Peer” confirmara si es correcto el número y posteriormente se obtendrá un numero aleatorio con el pool de servidores QRNG de COINsolidation. Este número aleatorio se registrará en todos los nodos, se creará una lista que contendrá **((Sumatoria Nodos) /2)) +1** y de esta lista se escogerá el que tenga el mayor porcentaje de probabilidad para ser el candidato ganador del consenso (PoQu) y este ejecutará la cola de transacción en curso.

El algoritmo Prueba de Cuanto (PoQu) también usa prueba del **NIST** (National Institute of Standards and Technology) para asegurarnos que los números aleatorios del QRNG son verdaderamente números aleatorios.

<https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-22r1a.pdf>

En COINsolidation tenemos implementado un bloque para la PoW y un bloque para el caso de la PoQu. Estos bloques usan un tipo de hash: SHA256 para su uso libre, para uso comercial se tiene un SHA512 y otros tipos de hash según sea el requerimiento.

Para ver más detalles del concepto de HASH ver:

[https://es.wikipedia.org/wiki/Funcion\\_hash](https://es.wikipedia.org/wiki/Funcion_hash)

NOTA: La Prueba de Trabajo (PoW) usada en los teléfonos móviles solamente pueden usar una dificultad máxima de 5 ya que el procesamiento matemático de estos dispositivos no es dedicado como los servidores o PC. El algoritmo (PoW) únicamente lo usamos para obtener la oportunidad de obtener su pase o permiso para entrar al sistema del Generador Cuántico de Números Aleatorios (QRNG – Quantum Random Number Generator) y con este ejecutar el algoritmo de la Prueba de Cuanto (PoQu). Ver uso de (PoQu) en Mini BlockyChain:

<https://github.com/openqbit-diy/MiniBlocklyChain>



## 5. Algoritmo para la creación de una dirección universal consolidada (CUA).

Actualmente las fusiones de empresas están al día, ya sea por un bien económico, tecnológico o de mercado.

Presentamos el primer modelo de fusión de criptoactivos o crypto-tokens basado en un algoritmo para crear una dirección consolidada que se usa y genera en el ambiente de COINsolidation.

Creamos tres tipos de direcciones consolidadas.

- **CUA** (Consolidated Universal Address) esta se usa para consolidar y crear un nuevo ExoToken (activo) para ser usado por el usuario. La combinación sera Cryptocurrency con Cryptocurrency. En el caso de la CUA se tiene la primera serie creada llamada CUAG (Coinsolidated Universal Address Genesis).
- **HAC** (Hibric Address Consolidated) se usa cuando necesitamos consolidar una direccion referente a una cryptocurrency y/o token y una direccion normal de tranferencia de activos.
- **DAC** (Dual Address Consolidated) esta se usa para administrar y consolidar dos direcciones normales de Token un mismo blockchain o de dos diferentes tecnologias, son direcciones simples de tranferencia de activos referidos a Token-Token.

Empecemos por ver las ventajas de la CUA.

Una dirección CUA está formada por la dirección del token COINsolidation (dirección estática) y un token adicional conocido como “Colored Coin” (dirección variable). En este caso podemos ver que las direcciones CUA siempre estarán formados por direcciones de algún tipo de combinación de activo (Criptomonedas o tokens).

En nuestro caso cuando consolidamos el token COINsolidation y un token OAP lo conoceremos como la

“CUA génesis” o CUAG (Consolidated Universal Address Génesis).

El token COINsolidation esta creado en el blockchain Ethereum y usa el standard ERC20 (Ethereum Request for Comments 20).



El token “Colored coin” está basado y creado en el blockchain de Bitcoin y usa el standard Open Asset Protocol (OAP).

Empecemos a revisar cual es el potencial y beneficio de consolidar direcciones.

- Para los usuarios que creen una CUA se podrá crear un token (OAP) que puede personalizar el usuario creador de la CUA, el usuario tendrá la posibilidad de tener su propio token o crypto activo para que lo pueda usar en la creación, soporte o expansión de su(s) negocios, de una forma simple y sencilla tendrá un activo en el mundo de los crypto-tokens.
- Para las empresas que creen una CUA podrán tener un token (OAP) que podrán usar para crear valor en su cadena de suministros o usar el activo en transacciones de liquidez basadas en el soporte económico de sus activos y pasivos de la empresa.
- Para las criptomonedas y tokens existentes al crear una CUA, podrán usar su dirección que identifica su activo y al consolidarla con el token (OAP) podrán crecer su demanda ofreciendo a sus inversionistas actuales y futuros su propio token para sus usuarios.

Nuestras direcciones universales consolidadas CUA (Consolidated Universal Address) son creadas utilizando el siguiente algoritmo:

Paso 1.- Seleccionamos las direcciones en Bitcoin y Ethereum.

Dirección Bitcoin – (BTC). – dirección A1

**1Hjx3CanChCytqVz7vek1SSvN1momghJ42**

Dirección Ethereum – (ETH) – dirección A2

**0x9d08c0ac0f2fdf078c883db6fa617b15776e4b41**

Paso 2.- Se obtiene el SHA512(address String-Texto) de cada dirección quitando el primer elemento de cada dirección y se toma de SHA512 los dos caracteres situados en los lugares 120 al 121 de cada hash operación simbolizada con “U”. Números verificadores.

U(SHA512(Hjx3CanChCytqVz7vek1SSvN1momghJ42)) = **bf**

U(SHA512(x9d08c0ac0f2fdf078c883db6fa617b15776e4b41)) = **28**



Paso 3.- Se toma el primer elemento (carácter) de cada dirección empezando por la dirección con menor tamaño en número de elementos y se obtiene la cadena “10”.

$A_{10}[0] = 1$

$A_{20}[0] = 0$

Paso 3.- Se les obtiene el SHA512 sin los elementos del paso 3 y se toman solo los cuatro caracteres situados en el lugar 120 al 123.

$U(\text{SHA512}(\text{Hjx3CanChCytqVz7vek1SSvN1momghJ42x9d08c0ac0f2fdf078c883db6fa617b15776e4b41})) = 140c$

Paso 4.- Se concatenan los caracteres de cada dirección uno a uno iniciando por la dirección que tenga menos caracteres que la componga, en caso de tener la misma cantidad de caracteres la concatenación puede iniciar de cualquier dirección.

Address 1 =  $A_{10}[0], A_{11}[1], A_{12}[2], A_{13}[3], A_{14}[4], \dots, A_{1N}[n], A_{1N+1}[n+1]$ .

Address 2 =  $A_{20}[0], A_{21}[1], A_{22}[2], A_{23}[3], A_{24}[4], \dots, A_{2N}[n], A_{2N+1}[n+1]$ .

Concatenación de direcciones:

$A_{10}[0] + A_{20}[0] + A_{10}[1] + A_{20}[1] + A_{11}[2] + A_{22}[2] + \dots A_{1N+1}[n+1] + A_{2N+2}[n+1]$ .

\*\*Últimos caracteres que no se puedan concatenar se ponen al principio de la cadena y el paso 3 se anexa al final de la cadena concatenada.

**776e4b41** Hxj9xd30C8acn0CahcC0yft2qfVdzf70v7e8kc18S8S3vdNb16mfoam6g1h7Jb4125 10

Paso 4.- Se agrega al principio el número de caracteres la cadena que NO se pudieron concatenar en el paso 4.

**8776e4b41** Hxj9xd30C8acn0CahcC0yft2qfVdzf70v7e8kc18S8S3vdNb16mfoam6g1h7Jb412510

Paso 5.- Se agrega al inicio de la cadena dos enteros verificadores XX que nos ayudaran a verificar si la diferencia de las cadenas (resta) la de mayor tamaño menos la menos, siempre debe de dar un numero entero positivo este par de enteros nos apoyara para evitar errores en la concatenación. En caso de que sea la diferencia sea menor o igual a “9” el numero verificador será “00” en caso de ser mayor a “9” la diferencia se marcar en estos dos dígitos.

En nuestro caso de la generación entre Bitcoin y Ethereum siempre los dos dígitos

verificadores serán “00”.

**00**8776e4b41Hxj9xd30C8acn0CahcC0yft2qfVdzf70v7e8kc18S8S3vdNb16mfoam6g1h7Jb412510





Paso 6.- Se concatenan los dos pares de verificadores del paso 2 de cada dirección al principio de la cadena resultado del paso 3 en el mismo orden A1 + A2.

**bf28**008776e4b41Hxj9xd30C8acn0CahcC0yft2qfVdzf70v7e8kc18S8S3vdNb16mfoam6g1h7Jb412510

Paso 7.- Se integra al final de la cadena los 4 dígitos del SHA512 del paso 3.

bf28008776e4b41Hxj9xd30C8acn0CahcC0yft2qfVdzf70v7e8kc18S8S3vdNb16mfoam6g1h7Jb412510**140c**

Paso 8.- Se integra la identificación de **CUA** (Consolidated Universal Address “Genesis”) Dirección Universal Consolidada al inicio de la dirección creada en el paso 5.

**CUA**fd55008776e4b41Hxj9xd30C8acn0CahcC0yft2qfVdzf70v7e8kc18S8S3vdNb16mfoam6g1h7Jb412510140c

\*\*En el caso de la consolidación de direcciones Bitcoin y Ethereum dará una dirección formada por **90** caracteres hexadecimales.

**NOTA:** el tamaño de las direcciones CUA, HAC y DAC pueden variar en cada caso dependiendo de las direcciones que las integren.



## Proyecto y solución por COINsolidation.

En la actualidad se tiene diferentes tipos de Blockchain orientados a activos de diferentes características, esto conlleva a tener una infinidad de tipos de direcciones de uso diario tener que llevar un control exhaustivo para no cometer errores de transferencia.

Por otra parte, el mundo de las criptomonedas y tokens está limitado a expertos en finanzas o en su caso expertos en tecnología de blockchain, para que una persona promedio le es difícil aventurarse en la creación de su propia criptomoneda o token.

Las dos anteriores problemáticas la hemos solucionado en COINsolidation realizando los siguientes puntos y/o herramientas que hemos creado.

Para el punto del control de direcciones de diferentes blockchain, creamos un algoritmo donde consolida (une) dos o más direcciones en sus diferentes combinaciones dando como resultado una sola dirección de tipo CUA, HAC y/o DAC.

Con esta solución en vez de estar enviando dos direcciones del mismo blockchain o diferente solo se trabajará con una dirección consolidada.

Para la segunda problemática hemos usado la metodología de programación llamada Blockly, esta es una herramienta visual en donde no se necesitan grandes conocimientos de programación y cualquier persona promedio o empresa podrá crear sus propias aplicaciones sin tener que invertir costosos equipos de desarrollo, tiempo y dinero.

Hemos creado las extensiones (módulos) de solo instala y úsalos para crear aplicaciones móviles, en 15 minutos. Ejemplo tu propio Exchange de criptomonedas o desarrollar tu propia moneda (token) en cuestión de minutos. Todo lo anterior usando seguridad en tu información de última generación llamada PQC (Post-Quantum Cryptography).

Solo instala las extensiones en cualquier herramienta de uso libre como Appventor, AppyBuider, Thunkable, Kondular u otras y en minutos podrás entrar al mundo de las criptomonedas y creación de tokens todo en la palma de tu mano.

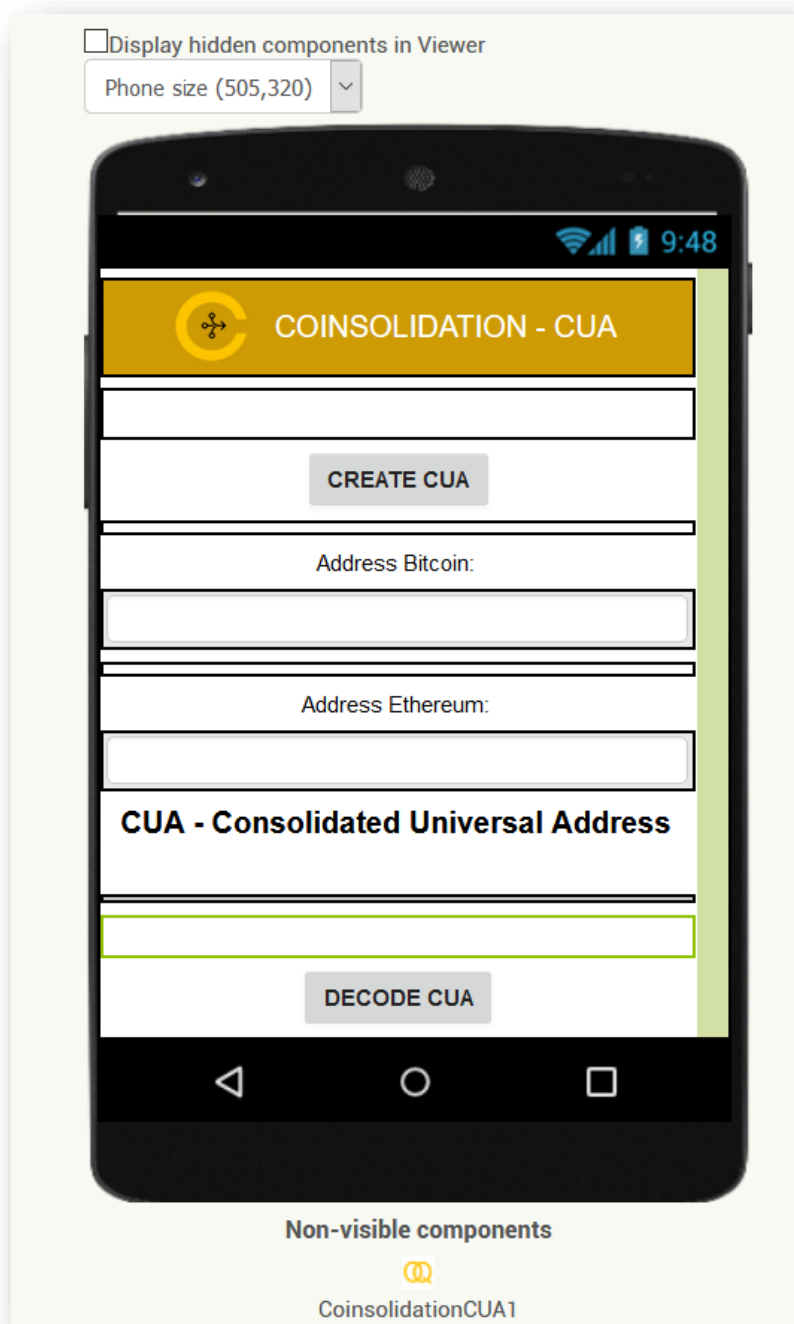
Por ultimo COINsolidation está creando el uso de seguridad cuántica (software y hardware) de bajo costo que podrá usarse en la protección de datos informáticos en casa. En la actualidad las tecnologías basadas en Computación cuántica y seguridad tiene alto costo solo corporativos con alto nivel financiero pueden crearlos y usarlo. Sin embargo, en COINsolidation creemos que las nuevas tecnologías deberán estar al alcance de todos, la equidad de uso del Blockchain y la Computación cuántica debe ser para todos, creamos software de uso libre (criptomonedas) y hardware de bajo costo (seguridad cuántica).



## 6. Creación de App CUA (Consolidated Universal Address) en 15 minutos.

\*App para Bitcoin y Ethereum coins (BTC-ETH)

Diseño pantalla 5 minutos en <https://appinventor.mit.edu/>





Uso de extensión **CoinsolidatioCUA.AIX** (5 minutos).

```
when GenerateCUA .Click
do
  set addressCUA . Text to call CoinsolidationCUA1 .CoinsolidationEncodeCUA_BTC_ETH
  hexAddressBitcoin InputAddressBitcoin . Text
  hexAddressEthereum InputAddressEthereum . Text

when DecodeCUA .Click
do
  call CoinsolidationCUA1 .CoinsolidationDecodeCUA_BTC_ETH
  hexAddressCUA InputAddressCUA . Text

when CoinsolidationCUA1 .OutPutAddress
  bitcoinStr ethereumStr checkBitcoin checkEthereum
do
  set addressBitcoin . Text to get bitcoinStr
  set addressEthereum . Text to get ethereumStr
  set verifyBitcoin . Text to get checkBitcoin
  set verifyEthereum . Text to get checkBitcoin
```



Creamos la aplicación en **Menu > Build > App** (provide QR code for .apk) - (5 minutos).

when **GenerateCUA** .Click

do

set **addressCUA** .Text to call **CoinsolidationCUA1** .CoinsolidationEncodeCUA\_BTC\_ETH

hexAddressBitcoin **InputAddressBitcoin** .Text

hexAddressEthereum **InputAddressEthereum** .Text

when **DecodeCUA** .Click

do

call **CoinsolidationCUA1** .CoinsolidationDecodeCUA\_BTC\_ETH

hexAddressCUA **InputAddressCUA** .Text

when **CoinsolidationCUA1** .OutPutAddress

do

set **addressBitcoin** .Text to get **bitcoinStr**

set **addressEthereum** .Text to get **ethereumStr**

set **verifyBitcoin** .Text to get **checkBitcoin**

set **verifyEthereum** .Text to get **checkBitcoin**

⚠ 0

⚠ 0

Show Warnings

CUA Progress Bar

35%

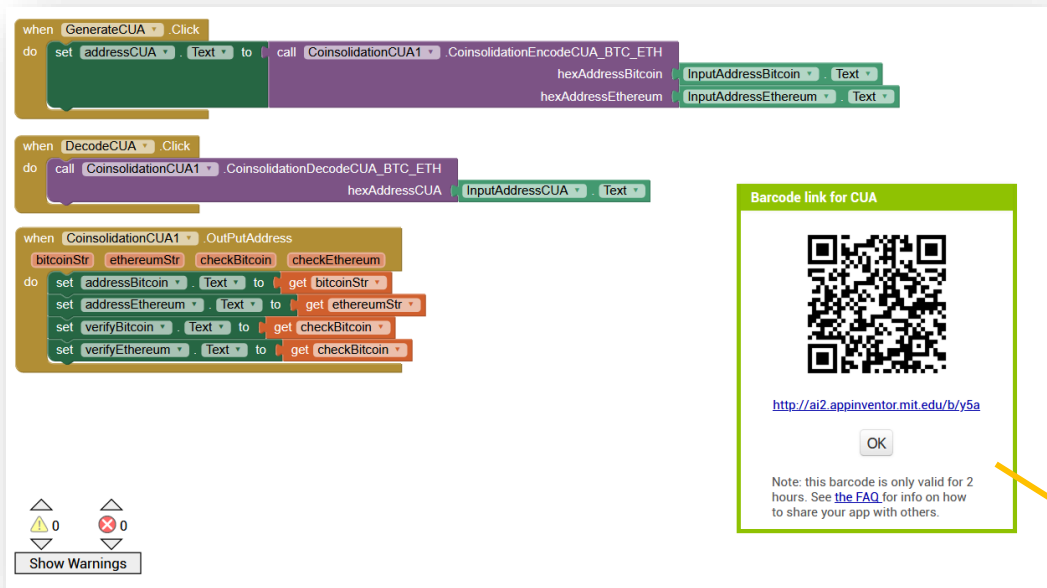
Compiling part 2 (please wait)

COINsolidation.org

Página 37 | 48



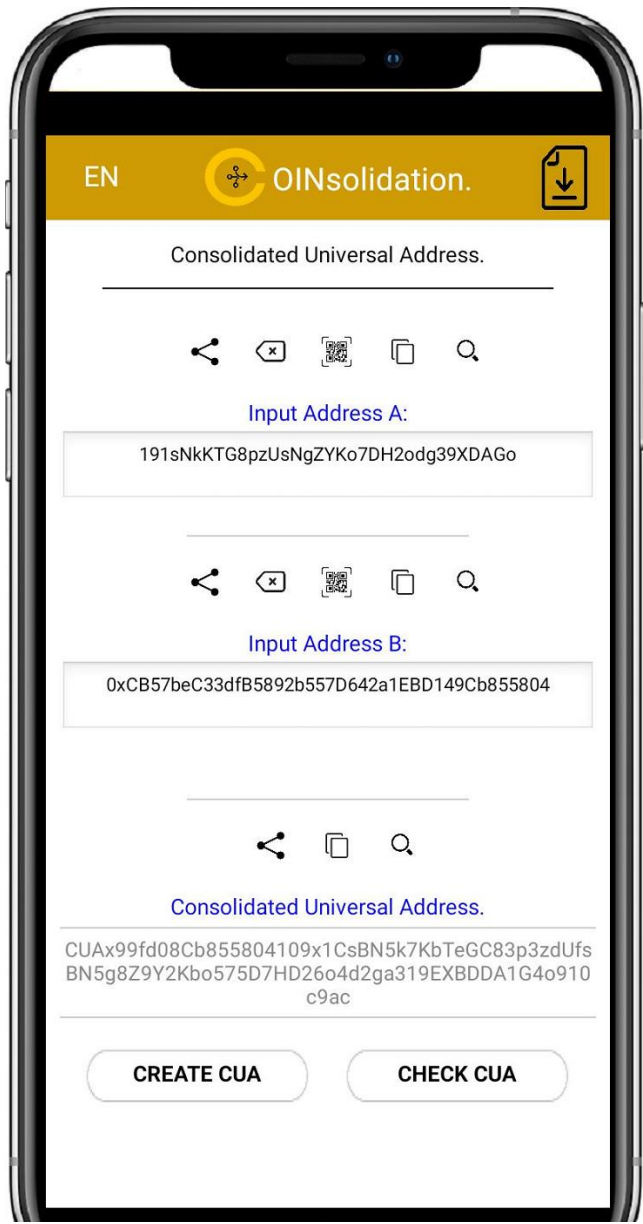
Instalamos la aplicación en el teléfono móvil desde el QR usando la aplicación para Android de AppInventor (MIT AI2 Companion) - <https://play.google.com/store/apps/details?id=edu.mit.appinventor.aicompanion3>



**NOTA:** La aplicación archivo APK listo para instalarse se encuentra en el siguiente repositorio:

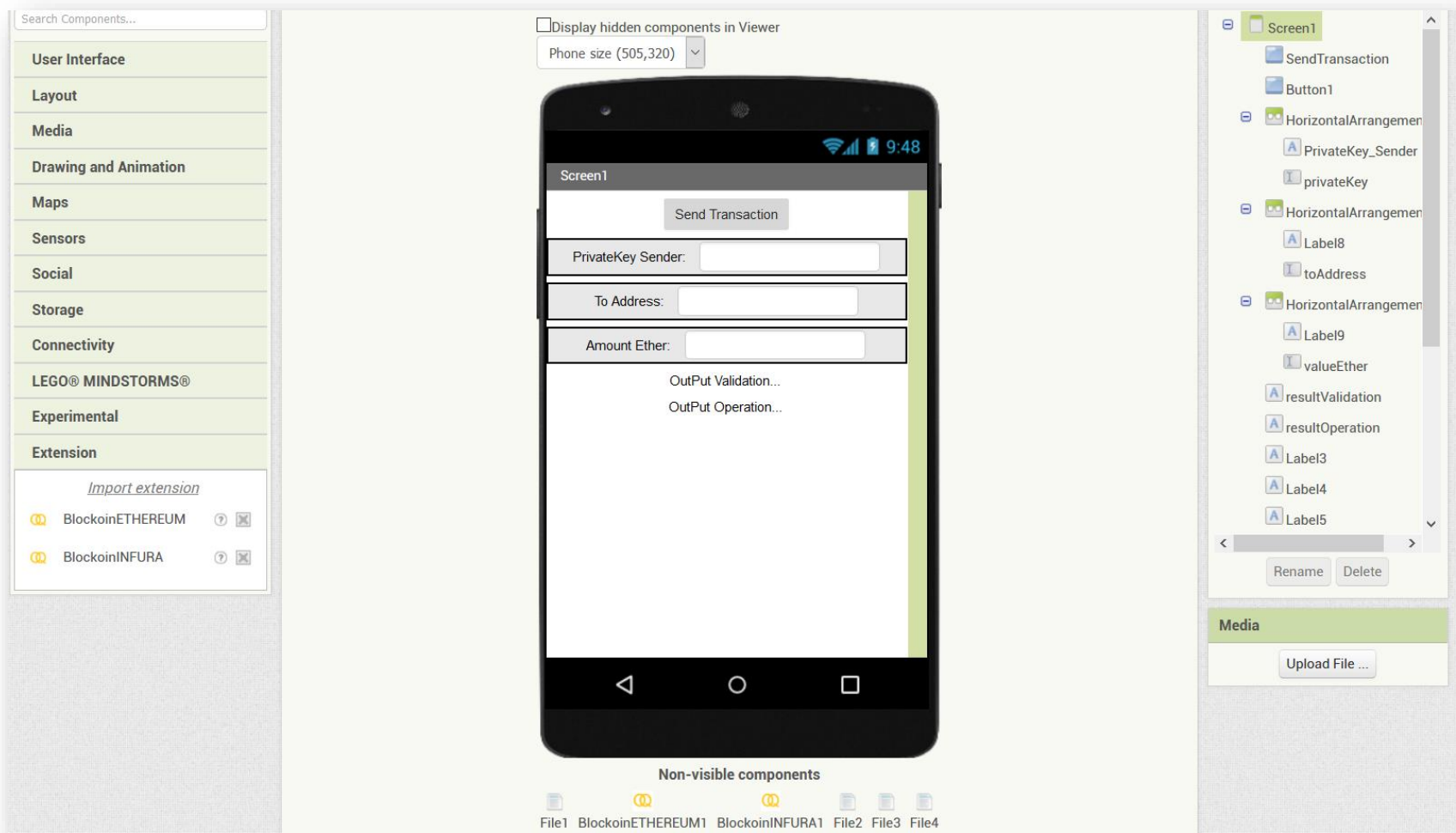
<https://github.com/COINsolidation/App>

Para revisar el Código Java para la generación de extensión CUA e implementar algoritmo para la generación de dirección universales consolidadas revisar el Anexo “Código para algoritmo CUA” o consultar la liga de código: <https://github.com/COINsolidation/source>





7. Crear tu Exchange de criptomonedas Ethereum en Android solo en 15 minutos.  
Diseño en App Inventor (Screen). – 5 minutos.





Bloques de función (eth\_SendTransactionEasy) y evento (OutPutSendTransactionEasy). – 5 minutos

```
when SendTransaction .Click
do
  call BlockoinETHEREUM1 .eth_sendTransactionEasy
    hexPrivateKeySender privateKey .Text
    toAddress toAddress .Text
    valueEther valueEther .Text
```

Datos de entrada:

**PrivateKey:** Llave primaria de la dirección del remitente.

**toAddress:** Dirección hexadecimal del receptor.

**valueEther:** Dar la cantidad de Ether que serán enviados.

```
when BlockoinETHEREUM1 .OutputSendTransactionEasy
  transactionValidationE transactionOperationE
do
  set resultValidation .Text to get transactionValidationE
  call File1 .SaveFile
    text get transactionValidationE
    fileName "/trasactionValidation.txt"
  set resultOperation .Text to get transactionOperationE
  call File1 .SaveFile
    text get transactionOperationE
    fileName "/trasactionOperation.txt"
```

Guardar los resultados en archivos de texto:  
Función File1: Archivo **trasactionValidation.txt**

Guardar los resultados en archivos de texto:  
Función File2: Archivo **trasactionValidation.txt**

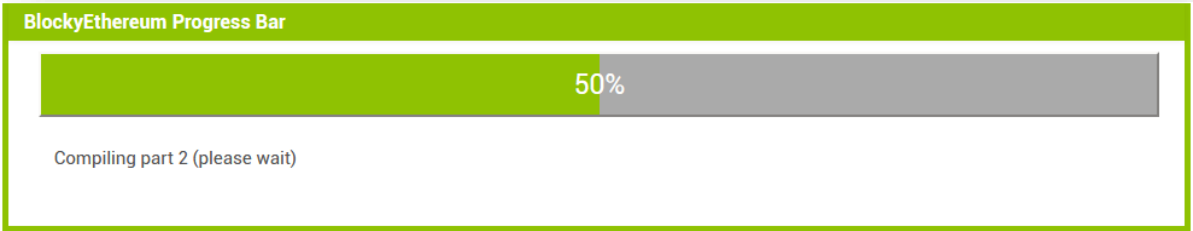
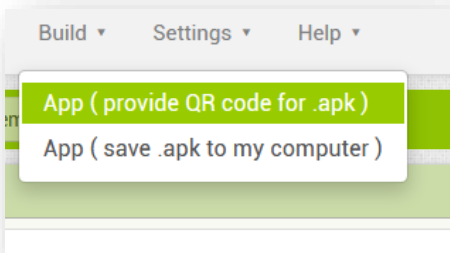
\*\*Mayor detalle ver Guía de usuario Extensión Ethereum Exchange (EEE) en el repositorio: <https://github.com/COINsolidation/userguide>

\*\*Repositorio de extensiones COINsolidation: <https://github.com/coinsolidation/Extesions-Cryptocurrencies> o OpenQbit (Blockchain & Quantum Computing) <https://github.com/openqbit-diy>





Compilamos, generamos archivo APK para instalarlo en el dispositivo Android. – 5 minutos



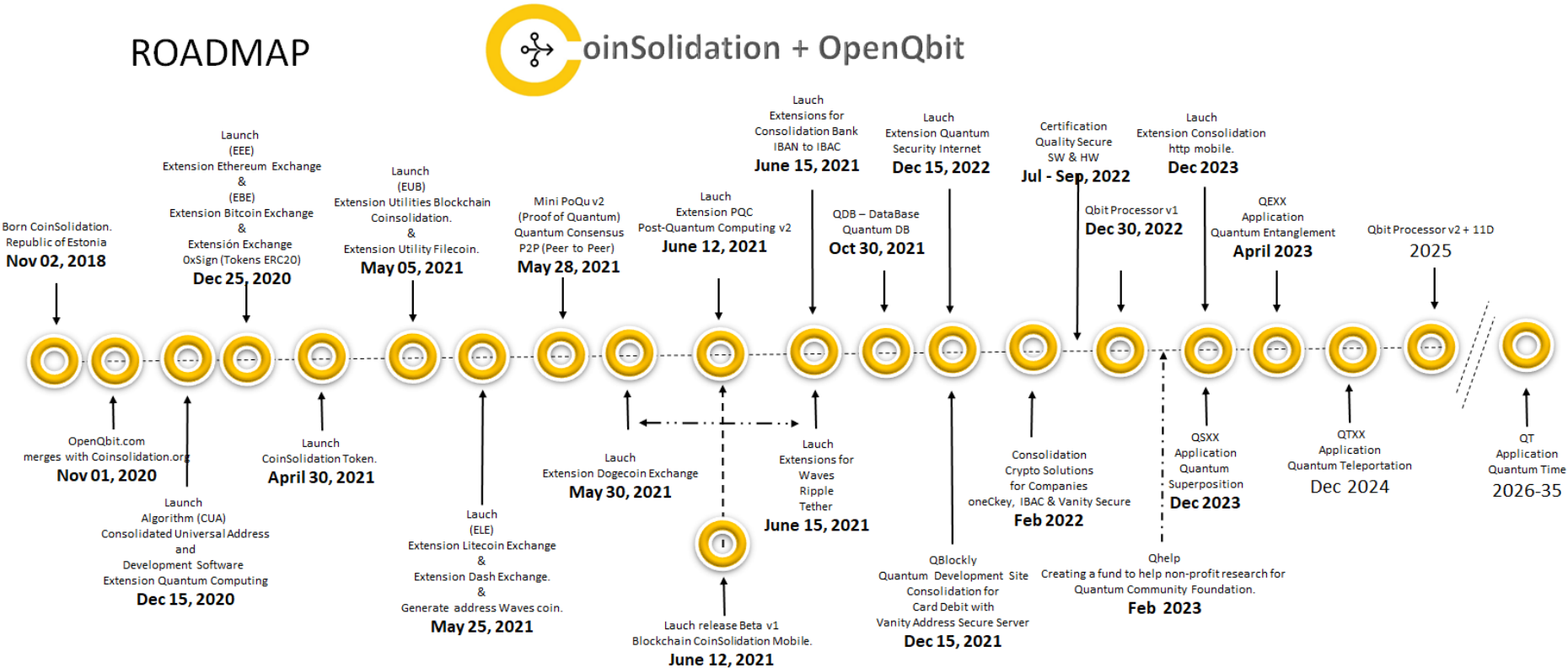
NOTA: Al ejecutar la transacción esta tardara de liberar el botón “Send Transaction” en un tiempo aproximado de 6 a 8 segundos. Debido al tiempo de conexión con la red Ethereum.

Para más detalles de la extensión EEE - (Extensión Ethereum Exchange). Ver el manual de usuario de EEE en la liga:

<https://github.com/COINsolidation/UserGuide>



8. Roadmap COINsolidation.



\*OpenQbit.com merges with COINsolidation.org (Nov 01, 2020) / OpenQbit specializes in Quantum Computing and Security Quantum.  
\*El procesador cuántico versión 1 estará usando compuertas lógicas cuánticas básicas para uso en casa.



9. COINsolidation Token (CUAG) – ICO DISTRIBUTION PLAN.

El ICO está dividido en tres etapas:

EXchange  
tensions

The private sale	\$ 0.01 USD	(30/April 2020 - 30/May 2021)	HARD CAPITAL:	\$ 28,000,000.00 USD
ICO FIRST PHASE	\$ 0.03 USD	(31/May 2021 - 28/Jun 2021)	FUNDRAISING GOAL:	\$ 2,800,000.00 USD
ICO SECOND PHASE	\$ 0.05 USD	(1/Jul 2021 - 31/Jul 2021)	SOFT CAPITAL:	\$ 280,000.00 USD

CoinSolidation TOKEN DISTRIBUTION		
	%	TOKENS
TOKEN SALE	70	28,000,000,000.00
TEAM AND DEVELOPMENT	10	4,000,000,000.00
ADVISORS	5	2,000,000,000.00
PARTNERS	5	2,000,000,000.00
EXCHANGES MARKET	1.5	600,000,000.00
MARKETING	5	2,000,000,000.00
COINsolidation FOUNDATION	0.5	200,000,000.00
BLOCKLY DEVELOPER COMMUNITIES	0.5	200,000,000.00
DEVELOPMENT AND RESEARCH OF QUANTUM COMPUTING	1.5	600,000,000.00
TOTAL SUPPLY 100%		40,000,000,000.00

0x9d08c0ac0f2fdf078c883db6fa617b15776e4b41	COINsolidation TOKEN
0xbbf57DE98c59B4C304C9d15BC5FAb01304aeCD97	ICO ADDRESS
0xa646c054394f85257E18D56Cf5c6b5E603447470	COINsolidation OPERATION ADDRESS



## 10. Características generales del token COINsolidation:

Nombre: COINsolidation

Símbolo: CUAG – (Consolidated Universal Address Genesis).

Totales de tokens creados: 40,000,000,000.00

Numero d decimales: 18

País de Lanzamiento: Estonia

Sitio oficial: [www.COINsolidation.org](http://www.COINsolidation.org)

Empresa: COINsolidation International.

Fecha de lanzamiento: 30 de Abril de 2021

Algoritmo de direcciones: Consolidated Universal Address (CUA).

Seguridad empleada: PQC (Post-Quantum Cryptography) basada en computación cuántica.

Propuesta tecnológica: Extensiones para sistemas Blockly para uso de cryptomonedas e implementación de seguridad cuántica.

Sociedades o acuerdos tecnológicos (fusión):

Empresa: OpenQbit Inc.

Industria: Computación cuántica y PQC (Post-Quantum Cryptography).

Sitio oficial: [www.OpenQbit.com](http://www.OpenQbit.com)





## 11. Conceptos básicos aplicados en plataformas de Blockchain.

### ¿Qué es un blockchain?

La blockchain generalmente se asocia con el Bitcoin y otras criptomonedas, pero estas son solo la punta del iceberg ya que no solo se usa únicamente para el caso de dinero digital, sino puede ser usado para cualquier información que pueda tener un valor para usuarios y/o empresas. Y es que esta tecnología, que tiene sus orígenes en 1991, cuando Stuart Haber y W. Scott Stornetta describieron el primer trabajo sobre una cadena de bloques asegurados criptográficamente, no fue notoria hasta 2008, cuando se hizo popular con la llegada del bitcoin. Pero actualmente su utilización está siendo demandada en otras aplicaciones comerciales y se proyecta un crecimiento en el futuro mediano en varios mercados, como el de las instituciones financieras o el de Internet de las Cosas IoT entre otros sectores.

La cadena de bloques, más conocida por el término en inglés blockchain, es un registro único, consensado y distribuido en varios nodos (dispositivos electrónicos como PC, smartpohones, tabletas, etc) de una red. En el caso de las criptomonedas, podemos pensarlo como el libro contable donde se registra cada una de las transacciones.

Su funcionamiento puede resultar complejo de entender si profundizamos en los detalles internos de su implementación, pero la idea básica es sencilla de seguir.

En cada bloque se almacena:

- 1.- una cantidad de registros o transacciones válidas,
- 2.- información referente a ese bloque,
- 3.- su vinculación con el bloque anterior y el bloque siguiente a través del hash de cada bloque –un código único que sería como la huella digital del bloque.

Por lo tanto, **cada bloque tiene un lugar específico e inamovible dentro de la cadena**, ya que cada bloque contiene información del hash del bloque anterior. La cadena completa se guarda en cada nodo de la red que conforma la blockchain, por lo que **se almacena una copia exacta de la cadena en todos los participantes de la red**.

### ¿Qué es una dirección o cuenta dentro de la plataforma blockchain Ethereum?

Es una cadena de 42 caracteres en la plataforma Ethereum que representan un número en base hexadecimal, en donde serán depositados o enviados los activos definidos en Ethereum. En otras plataformas de blockchain el número de caracteres de la cuenta o dirección puede ser diferente, ejemplo:



0x5d2Acdb34c279Aa6d1e94a77F7b18aB938BFb2bB

### **¿Qué es una criptomoneda?**

Es una moneda digital o virtual diseñada para funcionar como medio de intercambio. Utiliza criptografía (seguridad digital) para asegurar y verificar transacciones, así como para controlar la creación de nuevas unidades de una criptomoneda particular.

### **¿Qué es un token?**

Los tokens son activos digitales que se pueden usar dentro del ecosistema de un proyecto determinado.

La principal distinción entre tokens y criptomonedas es que las primeras requieren otra plataforma blockchain (no propia) para funcionar. Ethereum es la plataforma más común para crear tokens, principalmente debido a su función de contratos inteligentes. Los tokens creados en la blockchain de Ethereum generalmente se conocen como tokens ERC-20 aunque se tiene otros tipos de token más especializados por ejemplo el token ERC-721 usado principalmente para activos coleccionables (tarjetas, uso en video-juegos, obras de arte, etc.).

### **¿Qué es un Exchange?**

Un exchange de criptomonedas es el punto de encuentro donde se realizan los intercambios de estas a cambio de dinero fiat o de otras criptomonedas. En estas casas de cambio online donde se genera el precio de mercado que marca el valor de las criptomonedas en base a la oferta y demanda.

### **¿Qué es Exchange Rates?**

Son las tarifas del valor de un Ether o de otra criptomoneda en monedas de circulación de cada país.

### **¿Qué es una transacción?**

Es la ejecución o transferencia de algún tipo de activo no tangible que se le puede dar un valor pre-establecido dentro del sistema de Ethereum y que posteriormente puede ser cambiado a un valor tangible para una empresa o persona.

### **¿Qué es txHash?**

Es un numero hexadecimal que ayuda a rastrear el resultado a detalle de cada transacción.

### **¿Qué tipos de transacciones hay?**



Se tienen dos tipos, un es la transacción “offline” esta crea sin la necesidad de tener conexión a la red principal de Ethereum se pueden almacenar hasta que se escoja conectarse a la red de Ethereum y liberar la transacción, tienen la ventaja de seguridad ya que toda la transacción se procesa fuera de línea lo que impide cualquier anomalía que pudiera estar en la red de conexión. La otra transacción es la “online” la cual siempre se necesita estar conectado a internet con las ventajas y desventajas de seguridad que trae implícita.

### ¿Qué es una dirección en un Blockchain?

Una dirección o cuenta esta compuesta por tres partes, la dirección, la llave pública y la llave privada, estas dos llaves son una cadena de números y caracteres en formato hexadecimal que se ocupan para enviar y recibir (activos) o ether (moneda digital).

La llave primaria nunca debe ser compartida con nadie ya que es la que autoriza la liberación del saldo (firma las transacciones) que se tenga en la cuenta.

La llave publica es conocida por todo el público y es compartida a cualquier persona ya que es la referencia para confirmar que la transacción es correcta tanto el valor como a quien se envía.

Ejemplos de componentes de dirección de red Ethereum:

```
{  
  "private":  
    "429a043ea6393b358d3542ff2aab9338b9c0ed928e35ec0aed630b93adb14a1c",  
  "public":  
    "049b4b7e72701a09d3ee09165bba460f2549494a9d9fd7a95aaac57c2827eac162fd9e105b  
    2461cd6594ca8ca6a8daf10fe982f918be1b0060c87db9cfbcd289a8",  
  "address": "88ab6dcecc3603c7042f4334fc06db8e8d7062d5"  
}
```



## 12. ¿Qué es la programación Blockly?

**Blockly** es una **metodología de programación visual** compuesto por un sencillo conjunto de comandos que podemos combinar como si fueran las piezas de un rompecabezas. Es una herramienta muy útil para el que quiera **aprender a programar** de una forma intuitiva y simple o para quien ya sabe programar y quiera ver el potencial de este tipo de programación. Esta basada en el lenguaje JavaScript y fue desarrollada por la compañía de Google y MIT.

Blockly es una forma de programación en donde no se necesita algún antecedente de ningún tipo de lenguaje de computación o informática, esto se debe a que únicamente es unir bloques gráficos como si estuviéramos jugando lego o un rompecabezas, solo se necesita tener un poco de lógica y ¡Listo!!!

Cualquiera puede crear programas para teléfonos móviles (smartphones) sin meterse con esos lenguajes de programación difíciles de entender, solo arma bloques de forma gráfica de forma sencilla, fácil y rápida de crear.

## 13. Anexo “Código para algoritmo CUA”.

Referencia a Github: <https://github.com/coinsolidation/source>

## 14. Términos.

Términos y condiciones de uso ver en el sitio de [www.coinsolidation.org](http://www.coinsolidation.org) o <https://github.com/coinsolidation/Terms>

Soporte con uso comercial.

[support@coinsolidation.org](mailto:support@coinsolidation.org)

Ventas uso comercial empresarial de blockchain.

[sales@coinsolidation.org](mailto:sales@coinsolidation.org)

Información legal y preguntas o dudas de licenciamiento.

[legal@coinsolidation.org](mailto:legal@coinsolidation.org)

Redes sociales:

Twitter: <https://twitter.com/ecoinsolidation>

Facebook: <https://www.facebook.com/coinsolidation>