



EXchange  
tensions

COINsolidation.

Tallinn, Estonia. (E-residency)

# White Paper.

version 1.0.0

December 2020.

COINsolidation.org is a registered trademark of COINsolidation International, licensed for free and commercial use. Terms and conditions of use at: [www.Coinsolidation.org](http://www.Coinsolidation.org)

COINsolidation International merged with [www.OpenQbit.com](http://www.OpenQbit.com) for the cooperation of technology based on quantum mechanics (Quantum Security & Quantum Computing). This merger allows the use, sharing and re-engineering of technology developed by OpenQbit Inc. (Estonia, E-residency)

## Content

|  |    |
|--|----|
| 1. Introduction. ....  | 3  |
| 2. Security Quantum computing. ....  | 16 |
| 3. Hardware device creation of a QRNG (Quantum Random Number Generator).....       | 20 |
| 4. What is Proof Quantum (PQu - Proof Quantum)? .....                              | 26 |
| 5. Algorithm for the creation of a Consolidated Universal Address (CUA). ....      | 28 |
| Project and solution by COINsolidation.....  | 32 |
| 6. Creation of App CUA (Consolidated Universal Address) in 15 minutes. ....        | 33 |
| 7. Create your Ethereum cryptocurrency exchange on Android in just 15 minutes..... | 38 |
| 8. Roadmap COINsolidation. ....  | 41 |
| 9. COINsolidation Token (CUAG) - ICO DISTRIBUTION PLAN. ....                       | 42 |
| 10. General characteristics of the COINsolidation token:.....                      | 43 |
| 11. Basic concepts applied to Blockchain platforms.....                            | 44 |
| 12. What is Blockly programming? .....   | 47 |
| 13. Annex "Code for CUA algorithm". ....   | 47 |
| 14. Terms. ....  | 47 |

## 1. Introduction.

### 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)

**Abstract:** An algorithm to consolidate the private key of addresses of different blockchains, currently used in the financial area (cryptocurrencies), applications and various sectors. We propose two algorithms the first one we call **oneCKey (one Consolidated private Key)** is applied to consolidate private keys (One private key for addresses of different blockchain technologies). This is used to create more efficient private key management systems. This system is in sets of series to be, the first Series A1 or "Genesis" series we apply oneCKey on Bitcoin and Ethereum blockchains with a ratio of 1:2 (one private key to generate two addresses) and later we will have Series AX and BX where there will be ratios of 1:N (one private key for N-addresses). We use another algorithm called **CUA (Consolidated Universal Address)** for the creation of universal addresses that we will call consolidated addresses. The CUA algorithm is broken down into three types of address consolidation which are; **CUA** "Consolidated Universal Address" an address formed from Cryptocurrency with Cryptocurrency, **DAC** "Direct Address Consolidated" an address formed from Token with Token and the **HAC** "Hybrid Address Consolidated" an address formed from Token with Cryptocurrency.

## INTRODUCTION.

Currently 2021 there is a trend towards the use of blockchain systems in different sectors such as financial (cryptocurrencies) and emerging public and private sectors. The different blockchain technologies manage the generation of their own common addresses for the deposit operations of their corresponding assets and each address manages its assets independently by means of its unique private key referenced to a single public deposit address, currently there is only a 1:1 (one deposit address

referenced to one and only one private key for asset transfers) so that the management of multiple addresses is complicated by always having a number of N-asset deposit addresses referenced to N-private keys for asset transfers, an N:N relationship. Where N is the number of addresses created by a user in different cryptocurrencies, this N:N relationship always holds.

On the other hand, let's also consider that the management is complicated for private keys in the same way with the N: N ratio, i.e. a user who has N escrow addresses will have directly proportional N private keys to manage. Applying the oneCKey algorithm will create a 1:N ratio, a ratio of one private key for several escrow addresses of different blockchain technologies (we consolidate the private keys to only one that will manage N escrow addresses). Finally, there is also the need that each user wishing to make a deposit or an asset transfer needs to know the destination address, the proposal is to apply the second algorithm called CUA - "Consolidated Universal Address", an address that consolidates two or more blockchain technologies resulting in a single address.

All of the above must be working with the cryptography systems currently used by each cryptocurrency. When consolidating the primary key, the validation and current cryptography of each native blockchain of the cryptocurrency to be used is used, so that when each series is developed with a 1:N+1 ratio, the same cryptography and hashing protocols will be applied, i.e. consolidation in security.

After surveying the vast world of the most important and outstanding cryptocurrencies on the market, we have found that there are two fundamental trends for the generation of addresses and private keys. The two used ECC (*Elliptic curve cryptography*) curves that cover 95% of the current "most outstanding" cryptocurrencies are: I. The secp256k1 cryptographic curve used by Bitcoin and Ethereum. (**Series A**).

II. The Ed25519 cryptographic curve has a tendency to be used as one of the fastest ECC curves and is not covered by any *patents*. (**B series**).

Example:

| 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                        |

## Cryptography for Series A or "Genesis" series.

The A series or "Genesis" series is the first consolidation of private keys. This series consists of the consolidation of the 2 main cryptocurrencies and 2 tokens (Bitcoin, Ethereum, OAP and COINsolidation).

The cryptography applied in both cases is ECDSA (*Elliptic Curve Digital Secure Algorithm*), which is currently used in the generation of public and private keys by both Bitcoin and Ethereum.

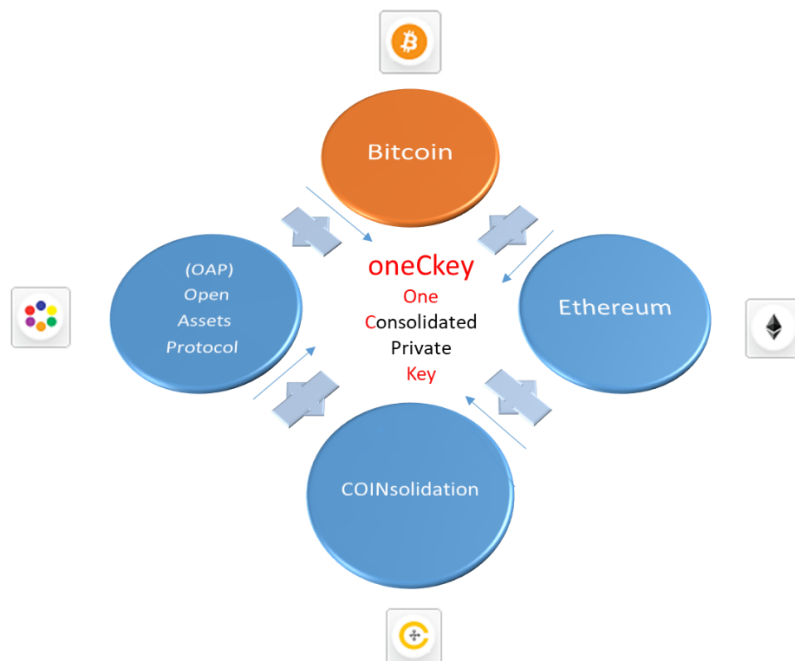
The generation of the consolidated private key integrated by both the Bitcoin and Ethereum blockchain can be applied in the following cases (3 options).

1.- Generation of a consolidated private key from QRNG (*Quantum Random Number Generator*)



We generate a random number from a QRNG source, we propose the generation of the optical-aquatic mechanics using a RAW format photo to produce sufficient entropy. This is done using the smartphone lens, we apply hash (SHA254) and make the conversion to hexadecimal to create the consolidated private key, this we apply the oneCKey-secp256k1 algorithm to generate the respective deposit addresses in each Bitcoin and Ethereum blockchain. We apply the Bitcoin address in OAP (Open Assets Protocol) format to generate an ExoToken. Using oneCKey will give us a single private key for two different deposit addresses of different blockchains such as Bitcoin and Ethereum, as well as the generation of an OAP (Open Assets Protocol) token to be used by the user according to their interests, opening a possibility to expand their business prospects.

2.- Generation of a consolidated private key from an existing Bitcoin address. The Bitcoin private key is applied to the oneCKey-secp256k1 algorithm to generate an Ethereum address and we apply the Bitcoin address in OAP (Open Assets Protocol) format to generate ExoToken.



Example:

Existing Bitcoin private key (in this one we apply oneCKey to create other addresses):

**28a0f97c6921e43872eb0640af41a54b9bde57c71cf4efe0db9d829f8b2cf645**

Bitcoin address:

**18XmTwfTeurjKQ8i1rEQT1DAx8BDjdR96A**

Ethereum address:

**0x9c789b22758c85f456dca3ac02e1fb00a059a4e**

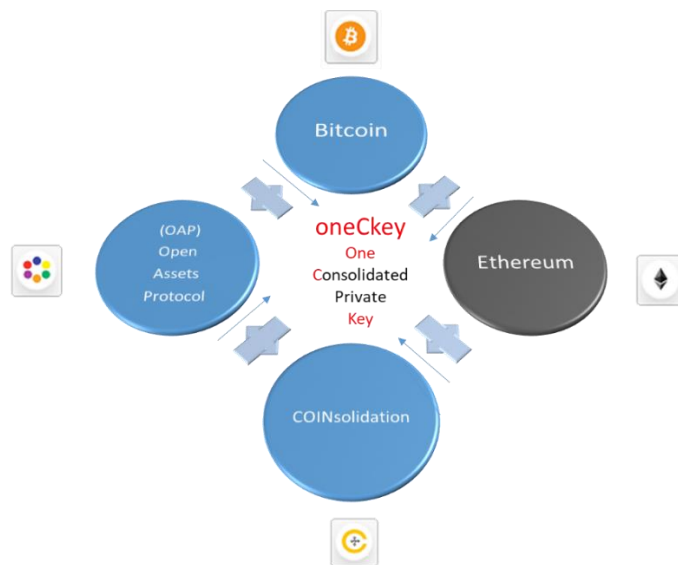
OAP (Open Assest Protocol) address based on the previous Bitcoin address.

**akJVVeI7Uo8PkRBeJ54ULb6s7kHjMPHs8UjG**

Address Token COINsolidation:

**0x9d08c0ac0f2fdf078c883db6fa617b15776e4b41**

Generation of a consolidated private key from an existing Ethereum address. The Ethereum private key is applied to the oneCKey-secp256k1 algorithm to generate a Bitcoin address and we apply the OAP (Open Assets Protocol) format to the Bitcoin address to generate an ExoToken.





## Cryptography for A1 Series.

A total of 5 cryptocurrencies and 2 tokens are involved.



In this case we will have 5 options to create the oneCKey, we can use every existing private key of the different supported cryptocurrencies.

## Cryptography for A2 Series.

Example: If we take the DASH private key by applying oneCKey we will generate 5 addresses based on the DASH private key and the two tokens will also depend on it. A ratio of 1:7 (1 consolidated private key for 7 addresses).

DASH private key (we apply oneCKey to create other addresses).

20ba723de1fdeee66e927e30fdb3ada74ae23bfdb370da378f301ce4dbf27312

DASH Directorate:

XtAGtBbnzykDSwoWUSjgQUF4gG1h3uyYKW

Bitcoin address:

1JUS3vwu3GXdJ1CvcZRTYwZGqvRzwT7FEk

Ethereum address:

**0x9d4a5854955c8e498e61eaaae7d3846917381f5b**

Address OAP:

**akUSKJ6mEWkRKAFNHfBXeCoTrBXcAyavXnS**

Address COINsolidation Token:

**0x9d08c0ac0f2fdf078c883db6fa617b15776e4b41**

Litecoin address

**LchPK9Fj7vmgYou5nhQkpxd348oHAJ3aSu**

Address Dogecoin:

**DNcXbBtYLgRuq1PXM9R26hisj4AJGJJj4pT**

The above can be applied in the same way for any existing address of the above mentioned cryptocurrencies supported by oneCKEY in the B series. This results in a private key for use in 5 cryptocurrencies and 2 tokens.

## Cryptography for A3 Series.

The A3 series applies to 18 cryptocurrencies and 2 tokens. This would be a 1:20 ratio (one consolidated private key for common use in 20 directions).



The above can be applied in the same way for any existing address of the aforementioned cryptocurrencies supported by oneCKey in the A3 series.

Example: By applying oneCKey to the Bitcoin private key, we will generate 18 addresses with a single private key and two tokens from it. A ratio of 1:20 (one consolidated private key for 20 addresses).

Bitcoin private key (we apply oneCKey to create other addresses).

**d2783ceae51a0c74fb733640f7128092dba3093605232b2b76fb102dca092d69**

1. Bitcoin address:  
1N67xWPN9F1sEMXparSpzT24erp5F2p6LK <https://www.blockchain.com/explorer>
2. Ethereum address:  
0x91efb31bcd0bd12a088f9625344bbe92c1543bc3 <https://etherscan.io/>
3. Address COINsolidation:  
0x9d08c0ac0f2fdf078c883db6fa617b15776e4b41 <https://etherscan.io/>
4. EOS Wallet Address:  
5KQyKH8ssQkwsuUWffvrbsMRf5wh8iHofZGvkHou5h1HYCqJYp
5. Address TRON:  
TPGr8Vp78vNcfhSvnpp7EnucaBKEBXns7h (The account is not activated.)  
<https://trx.tokenview.com/>
6. Address Peercoin:  
PVgJ7UnDCAW4DDCCavvmMfLzLGbyxJ4HRpP <https://blockbook.peercoin.net/>
7. DASH Directorate:  
Xwmxm3G6xETPJ8QJm3qyhrVCPmECp3dt <https://explorer.dash.org/insight/>
8. Address Dogecoin:  
DSEDVml1Sev9mMiRKSSPYDBFxzYNYadkdV <https://blockchair.com/dogecoin>
9. Address Komodo:  
RWNK32Gek4pSJMmu242Rx5yMGR8GfprvY69 <https://kmdexplorer.io/>
10. XRP address:  
r4afxWP49Er1NMXF2iSFzTpeiFnEpFaLK (The account is not activated.)  
<https://bithomp.com/explorer/>
11. Address Ravencoin:  
RWNK32Gek4pSJMmu242Rx5yMGR8GfprvY69  
<https://ravencoin.network/>

12. Address DigiByte:  
DSEDVmL1Sev9mMiRKSSPYDBfXzYNYadkdV <https://digibyteblockexplorer.com/>
13. Address Namecoin:  
NHfVA9tM4d7RktnKrfmQCyAndP6D8AQ2rA5  
<https://www.cryptoground.com/namecoin-block-explorer>
14. Address Bitcoin Cash:  
1N67xWPN9F1sEMXparSpzT24erp5F2p6LK (New format to see explorer site)  
<https://explorer.bitcoin.com/bch> New format address:  
bitcoincash:qrn49p6cy49tzgatt9gz9xz3gf8qxl9tu800hul9m
15. Address VeChain:  
0x91efb31bcd0bd12a088f9625344bbe92c1543bc3 <https://explore.vechain.org>
16. Address Fantom:  
0x91efb31bcd0bd12a088f9625344bbe92c1543bc3 <https://explorer.fantom.network/>
17. Ethereum Classic address:  
0x91efb31bcd0bd12a088f9625344bbe92c1543bc3 <https://etcblockexplorer.com/>
18. Litecoin address:  
LgK5DihCDuFvVVADyKzS8GU5ps5BMGWfCXX <https://blockchair.com/litecoin>
19. Address Color coin:  
akY41CgChciuZ6bhBdUZ1eJvdzTzF9Sizv3 <https://blockchainexplorer.lykke.com/>
20. WIF (Wallet Import Format) - NEO use secp256r1, Do not use secp256k1).  
To get address import WIF site: <https://neotracker.io/wallet/open-wallet> uses WIF as Private Key.  
  
WIF (Wallet Import Format) NEO:  
  
L4GqU2DmXVryD5JRwgphv6yF9q6pQ93FAyBTJE5BzshaFnwXPB4H  
Address NEO:  
AKkcm37QeMxZpFfRrrLGeqFRgkFzMPH9Gn  
  
<https://neotracker.io/>
21. Address: ExoCrypto:  
Exo41CgChciuZ6bhBdUZ1eJvdzTzF9Sizv3 <https://exoCrypto.com>

Summary of cryptocurrencies supported and classified by the two ECC curves.  
(secp256k1 & Ed25519)

|  |   |  |
|--|---|--|
| <p>A3 Series (secp256k1) - Ready</p> <ol style="list-style-type: none"> <li>1. Bitcoin*</li> <li>2. OAP***</li> <li>3. Ethereum**</li> <li>4. COINsolidation***</li> <li>5. XRP</li> <li>6. NEO</li> <li>7. EOS</li> <li>8. TRON</li> <li>9. Dash</li> <li>10. Bitcoin Cash</li> <li>11. Ethereum Classic**</li> <li>12. Litecoin</li> <li>13. Dogecoin</li> <li>14. VeChain**</li> <li>15. DigiByte</li> <li>16. Fantom**</li> <li>17. Ravencoin</li> <li>18. Komodo</li> <li>19. Peercoin</li> <li>20. Namecoin</li> </ol> | <p>(*) Bitcoin cryptocurrencies with the same address as the algorithm for generating them is the same, only the blockchain changes, which is different for each one's transactions.</p> <p>(**) Ethereum cryptocurrencies with the same address, as the algorithm for generating them is the same, only the blockchain changes, which is different for the transactions of each one.</p> <p>(***) Tokens generated and dependent on the consolidated primary key.</p> <p>Roadmap Serie A4 incluye Serie A3 + Zcash, Binance BNB, Filecoin, Tether, Tezos, Cosmos, Zilliqa y Avalanche. Q4-2021</p> <p>Roadmap starts Series B1 with Curve Ed25519. (Polkadot, Cardano, Monero, Stellar, Algorand, IOTA, Elrond, Algorand, and Waves) release Q42021.</p> | <p>B1 series (Ed25519) -Q42021</p> <ol style="list-style-type: none"> <li>1. Polkadot</li> <li>2. Cardano</li> <li>3. Monero</li> <li>4. COINsolidation***</li> <li>5. Stellar</li> <li>6. Algorand</li> <li>7. IOTA</li> <li>8. Elrond</li> <li>9. Decred</li> <li>10. Nano</li> <li>11. Horizen</li> <li>12. Siacoin</li> <li>13. Stacks</li> <li>14. Lisk</li> <li>15. Qtum</li> <li>16. Waves 17. ....Others.</li> </ol> |
|--|---|--|

**NOTE:** The crypto-assets of each series may vary depending on the behaviour of the cryptocurrency market and the list may be modified, depending on the liquidity of each asset.

Security implemented in oneCKey, PQC - (Post-Quantum Cryptography) algorithms embedded in the repository where the consolidated key generated locally in each Smartphone or an existing key entered by the user is stored.

For the secure encryption of the oneCKey we have implemented a combination of PQC security algorithms consisting of: AES-CGM + chacha20poly1305.

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

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

U = PQC algorithm application operator.

Encryption of oneCKey = U (AES-CGM ( chacha20poly1305 (OneCKey)))

## 2. Security Quantum computing.

### How does quantum computing work? <sup>(2)</sup>

Digital transformation is changing the world faster than ever before. Would you believe that the digital age is about to end? **Digital literacy** has already been identified as an area where open knowledge and accessible opportunities to learn about technology are urgently needed to address gaps in social and economic development. Learning about the key concepts of the digital age will become even more critical with the imminent arrival of another new wave of technology capable of transforming existing models with astonishing speed and power: **quantum technologies**.

In this article, we compare the basic concepts of traditional computing and quantum computing; and we also begin to explore their application in other related areas.

What are quantum technologies?

Throughout history, humans have developed technology as they have come to understand the workings of nature through science. Between the 1900s and 1930s, the study of some physical phenomena that were not yet well understood gave rise to a new physical theory, **Quantum Mechanics**. This theory describes and explains the workings of the microscopic world, the natural habitat of molecules, atoms and electrons. Not only has it been able to explain these phenomena, but it has also made it possible to understand that the subatomic reality works in a completely counter-intuitive, almost magical way, and that events take place in the microscopic world that do not occur in the macroscopic world.

These **quantum properties** include quantum superposition, quantum entanglement and quantum teleportation.

- **Quantum superposition** describes how a particle can be in different states at the same time.
- **Quantum entanglement** describes how two particles as far apart as desired can be correlated in such a way that, when interacting with one, the other becomes whole.
- **Quantum teleportation** uses quantum entanglement to send information from one place in space to another without the need to travel through space.

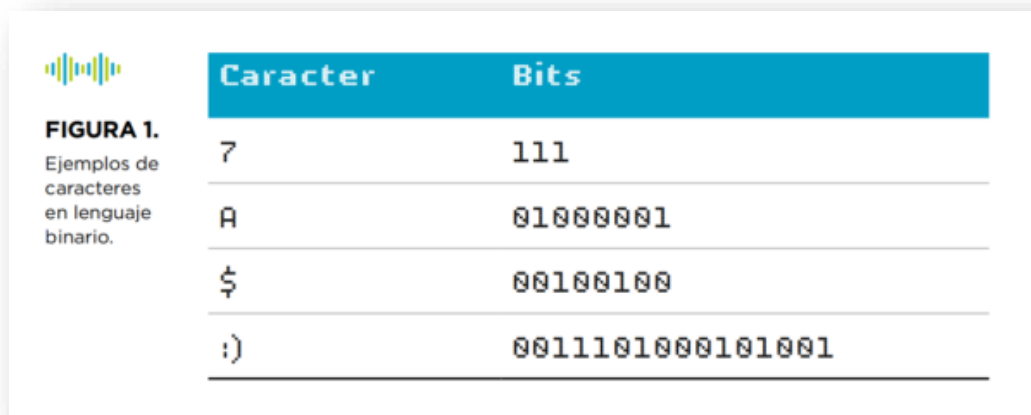
Quantum technologies are based on these quantum properties of subatomic nature.

In this case, today, understanding the microscopic world through quantum mechanics allows us to invent and design technologies that can improve people's lives. There are many different technologies that use quantum phenomena, and some of them, such as lasers or magnetic resonance imaging (MRI), have been around for more than half a century. However, we are currently witnessing a technological revolution in areas such as quantum



computing, quantum information, quantum simulation, quantum optics, quantum metrology, quantum clocks and quantum sensors.

What is quantum computing? First, you have to understand classical computing.



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

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

To understand how quantum computers work, it is useful to first explain how the computers we use every day, referred to in this document as digital or classical computers, work. These, like all other electronic devices such as tablets or mobile phones, use bits as their fundamental units of memory. This means that programs and applications are encoded in bits, i.e. in a binary language of zeros and ones. Every time we interact with any of these devices, for example by pressing a key on the keyboard, strings of zeros and ones are created, destroyed and/or modified within the computer.

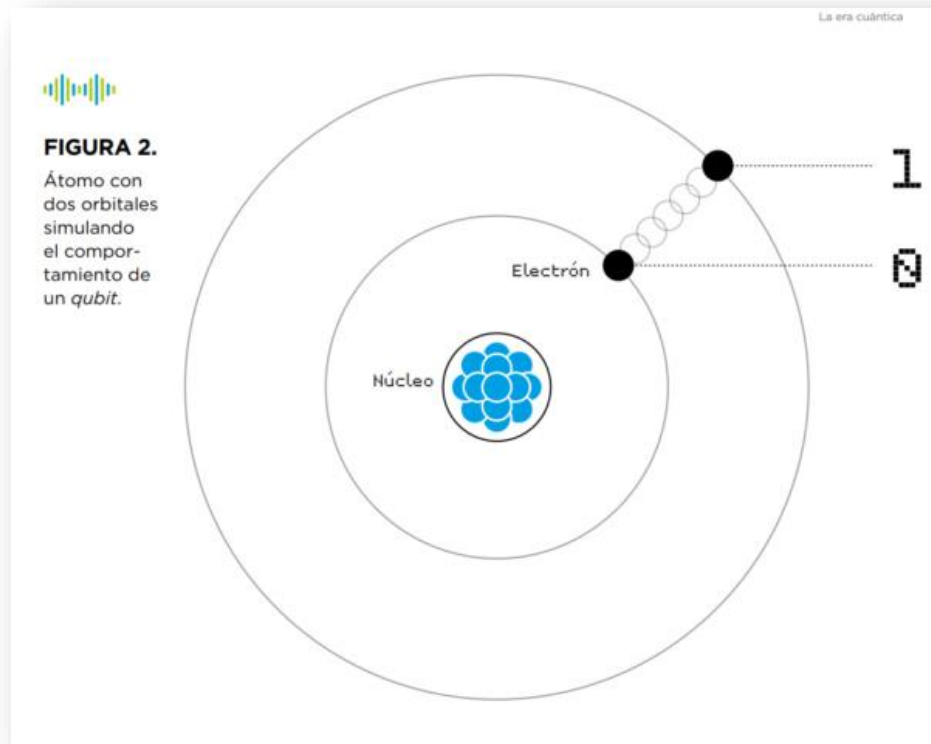
The interesting question is, what are these zeros and ones physically inside the computer? The zero and one states of the bits correspond to electrical current flowing, or not, through microscopic parts called transistors, which act as switches. When no current is flowing, the transistor is "off" and corresponds to a bit 0, and when it is flowing, it is "on" and corresponds to a bit 1.

In a more simplified form, it is as if the bits 0 and 1 correspond to holes, so that an empty hole is a bit 0 and a hole occupied by an electron is a bit 1. As an example, figure 1 shows how some characters are written in binary language. Now that we have an idea of how today's computers work, let's try to understand how quantum computers work.

### From bits to qubits

The fundamental unit of information in quantum computing is the quantum bit or qubit. Qubits are, by definition, two-level quantum systems - we will now look at examples - which, like bits, can be at the low level, which corresponds to a state of low excitation or energy defined as 0, or at the high level, which corresponds to a state of higher excitation or defined

as 1. However, and here lies the fundamental difference with classical computing, qubits can also be in any of an infinite number of intermediate states between 0 and 1, such as a state that is half 0 and half 1, or three quarters of 0 and one quarter of 1. This phenomenon is known as quantum superposition and is natural in quantum systems.



Quantum algorithms, exponentially more powerful and efficient computing

The purpose of quantum computers is to take advantage of these quantum properties of *qubits*, as quantum systems, to be able to run quantum algorithms that use superposition and entanglement to offer a much higher processing capacity than classical ones. It is important to point out that the real paradigm shift does not consist of doing the same thing that digital or classical computers -the current ones- do, but faster, as many articles erroneously state, but that quantum algorithms allow certain operations to be performed in a totally different way that in many cases turns out to be more efficient -that is, in much less time or using much less computational resources-.

Let's look at a concrete example of what this implies. Let's imagine that we are in Bogotá and we want to know which is the best route to Lima out of a million options to get there ( $N=1,000,000$ ). In order to be able to use computers to find the optimal route, we need to digitise 1,000,000 options, which implies translating them into bit language for the classical computer and into *qubits* for the quantum computer. While a classical computer would need

to go through all the paths one by one until it finds the desired one, a quantum computer takes advantage of a process known as quantum parallelism that allows it to consider all paths at once. This implies that, while the classical computer needs on the order of  $N/2$  steps or iterations, i.e. 500,000 attempts, the quantum computer will find the optimal path after only  $\sqrt{N}$  operations on the register, i.e. 1,000 attempts.

In the above case the advantage is quadratic, but in other cases it is even exponential, meaning that with  $n$  *qubits* we can obtain a computational capacity equivalent to  $2^n$  bits. To exemplify this, it is often said that with about 270 qubits you could have more base states in a quantum computer - more different and simultaneous strings of characters - than the number of atoms in the universe, which is estimated to be about  $10^{80}$ . Another example is that it is estimated that with a quantum computer of between 2000 and 2500 *qubits* you could break practically all the cryptography used today (known as public key cryptography).

Why is it important to know about quantum technology?

We are at a time of digital transformation in which different emerging technologies such as blockchain, artificial intelligence, drones, the Internet of Things, virtual reality, 5G, 3D printers, robots and autonomous vehicles are increasingly present in multiple fields and sectors. These technologies, which are set to improve the quality of human life by accelerating development and generating social impact, are currently advancing in parallel. Only rarely do we see companies developing products that exploit combinations of two or more of these technologies, such as blockchain and IoT or drones and artificial intelligence. While they are destined to converge and thus generate exponentially greater impact, the early stage of development they are in and the shortage of developers and people with technical profiles mean that convergences are still a pending task.

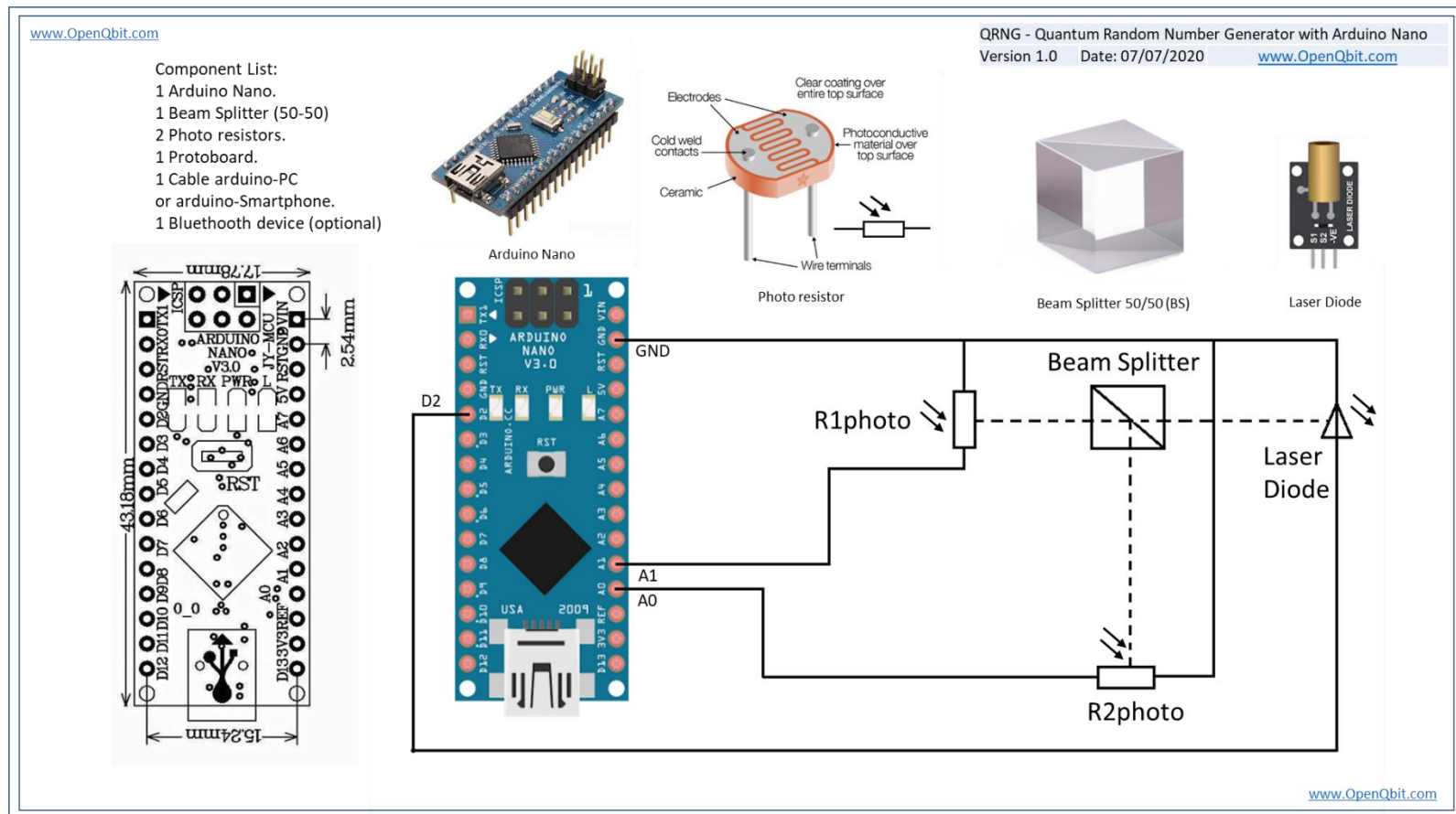
Due to their disruptive potential, quantum technologies are expected not only to converge with all these new technologies, but also to have a cross-cutting influence on practically all of them. Quantum computing will threaten the authentication, exchange and secure storage of data, having a greater impact on those technologies in which cryptography plays a more relevant role, such as cybersecurity or blockchain, and a lesser negative impact but also to be considered in technologies such as 5G, IoT or drones.

### **Do you want to practice quantum computing?**

Dozens of quantum computer simulators are already available on the web with different existing programming languages such as C, C++, Java, Matlab, Maxima, Python or Octave. Also, new languages such as Q#, launched by Microsoft. It is possible to explore and play with a virtual quantum machine through platforms such as IBM's and Rigetti's.

### 3. Hardware device creation of a QRNG (Quantum Random Number Generator).

We will now create a physical hardware device to generate quantum random numbers (QRNG) with inexpensive components that can be easily assembled at home and at a low cost of approximately \$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);
}
```

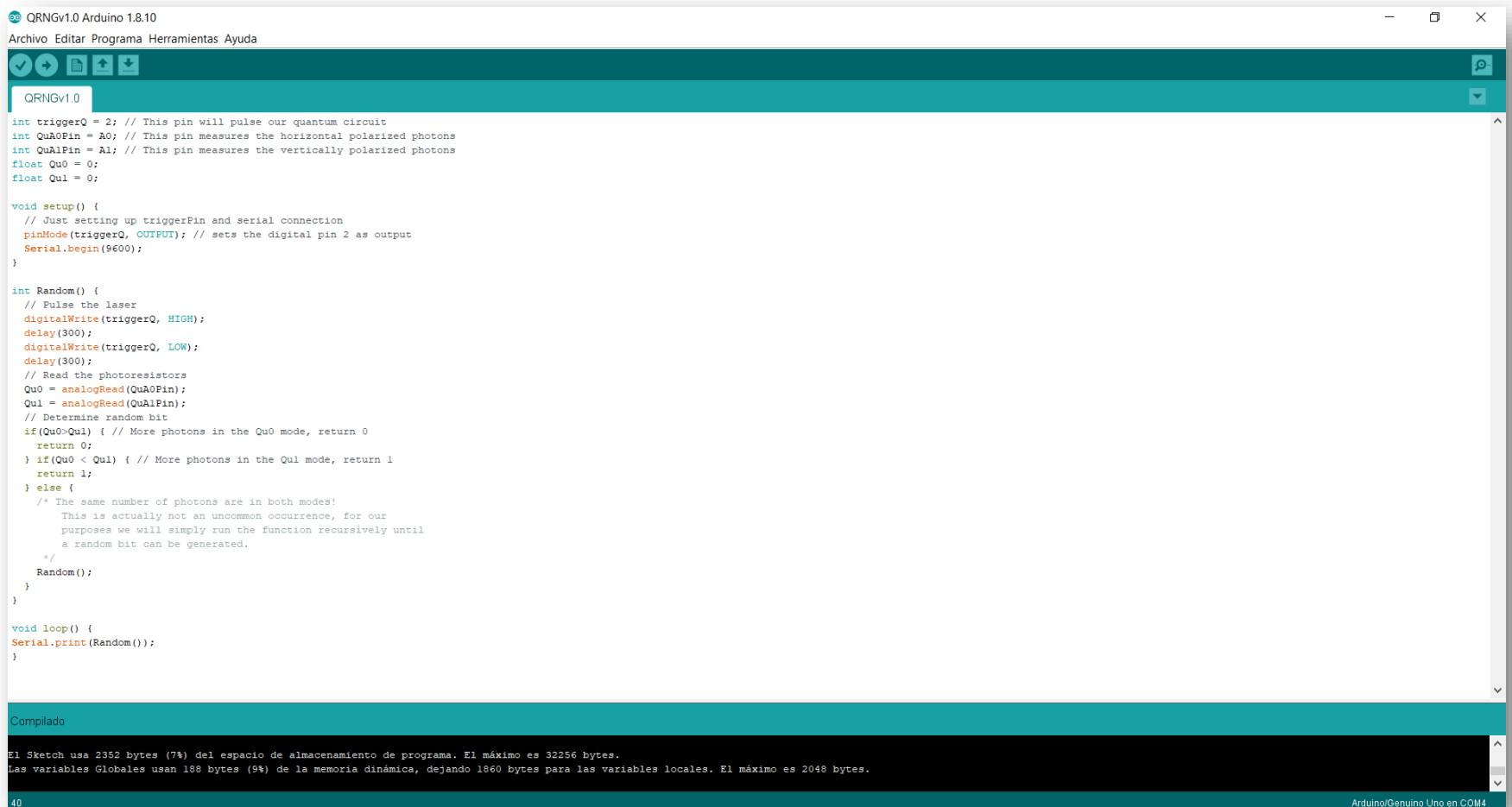
```
int Random() {
  // Pulse the laser
  digitalWrite(triggerQ, HIGH);
  delay(300);
  digitalWrite(triggerQ, LOW);
  delay(300);
  // Read the photoresistors
  Qu0 = analogRead(QuA0Pin);
  Qu1 = analogRead(QuA1Pin);
  // Determine random bit
  if(Qu0>Qu1) { // More photons in the Qu0 mode, return 0
    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.....

Compiling QRNGv10.ino program and uploading to 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 'Random()' function pulses a laser (pin 2) and reads the sensor values to determine a random bit. The 'loop()' function prints the generated 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 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());
}
```

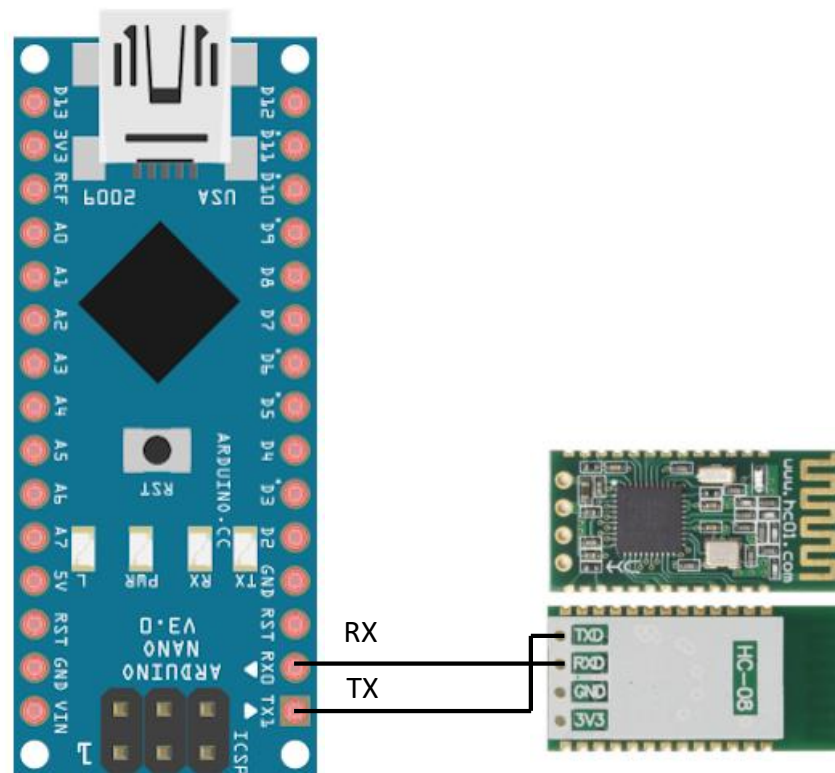
Compilado

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.

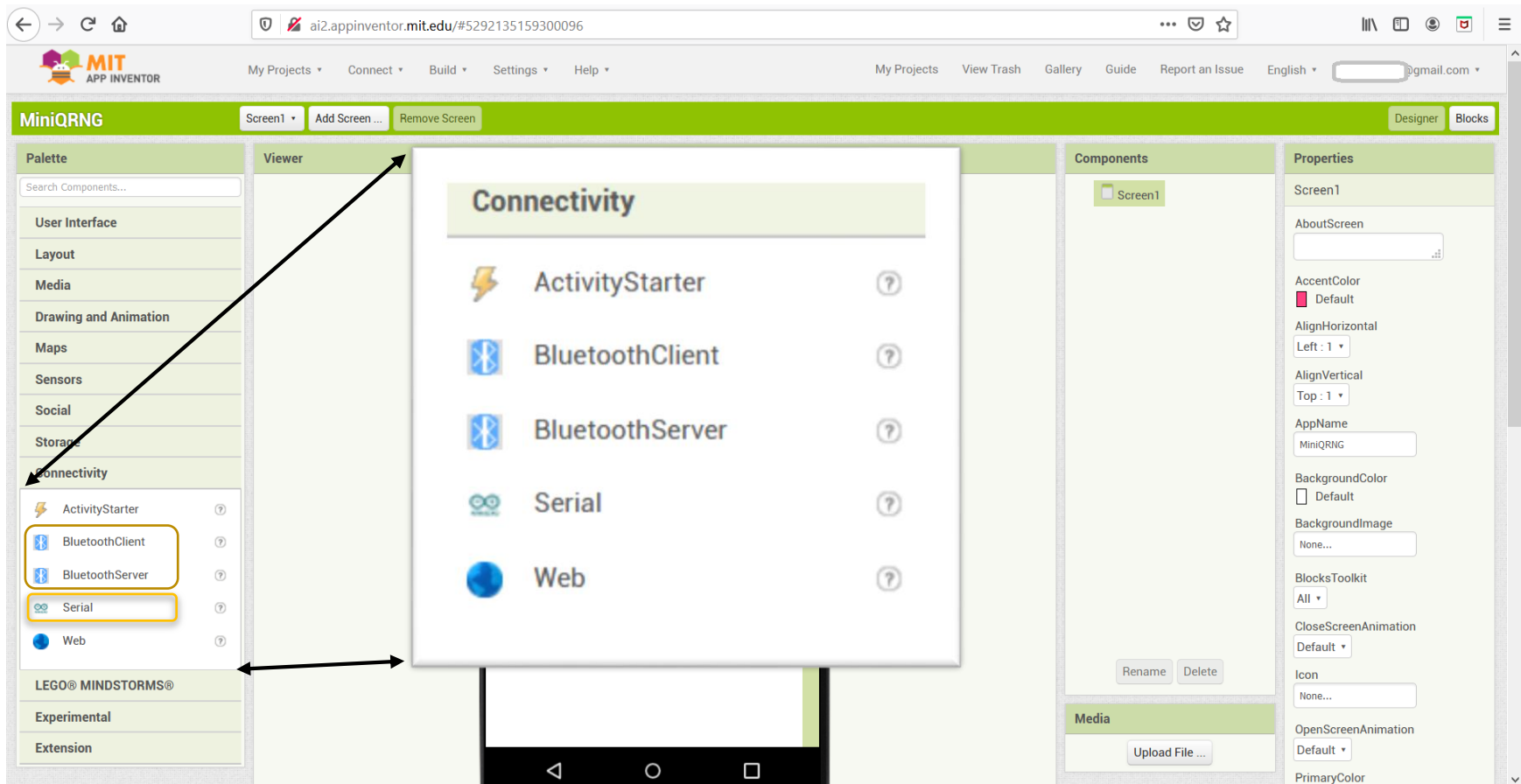
40 Arduino/Genuino Uno en COM4

There are two ways to communicate with the arduino nano, one is through the Serial port and the other is through a Bluetooth connection.

For the bluetooth connection it is very simple, just buy the HC-08 module or a similar one and connect it in the following way:



In order to connect App Inventor with Arduino you can use the following components Serial or Bluetooth:







Now that the QRNGv10.ino program is compiled and loaded, we only need to communicate with the arduino nano to save the data (quantum random numbers) in binary format, however, the data obtained can be easily converted to another format such as hexadecimal or decimal depending on the final requirement.

Finally, to see an example of a serial or bluetooth connection, some reference links are shown below.

Recall that everything is through Blockly programming to be tested with App Inventor this already has blocks for communication with arduino serially or other blockly type system may be through bluetooth similar online.

[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/>

To review the complete project design and use of QRNG (Quantum Random Number Generator) extensions. Check the user manual at:

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



#### 4. What is Proof Quantum (PQu - Proof Quantum)?

PoQu. - Proof of Quantum is a consensus algorithm developed for Mini BlocklyChain and COINsolidation, this proof is a variant of Proof of Work (PoW) that works as follows.

The Proof of Quantum (PoQu) at the beginning is executed with the same algorithm as the "Proof of Work" (PoW) and is based on putting the processor of the device (PC, Server, Tablet or Mobile Phone) to work to obtain a string of characters that is a mathematical riddle called a "hash".

Remember that a "hash" is an algorithm or mathematical process that when entering a phrase or some type of digital information such as text files, program, image, video, sound or any other type of digital information gives us as a result an alphanumeric character that represents the digital signature that represents it in a unique and non-repeatable form of the data, The hash algorithm is unidirectional, which means that when we introduce a piece of data to obtain its hash signature, we cannot carry out the inverse process, as we have a hash signature, we cannot know what information it was obtained from. This property gives us a security advantage when processing the information we send over the internet. How does it work? Imagine sending any type of information through non-secure channels and accompanying this information with its respective "source hash", the receiver on receiving the information can extract the "hash" of the information received we will call it "destination hash" and compare it with the "source hash" if both "hashes" are the same we can confirm that the information has not been altered in the channel that was sent, this is just an example where this type of information security process is currently used.

Currently there are different types of hash algorithms or processes, which differ in the level of security. The most widely used or known are: MD5, SHA256 and SHA512.

Example of SHA256:

We have a string or sentence as follows: "Mini BlocklyChain is modular".

Applying a hash of type SHA256 to the above string will give the following hash.

**f41af7e61c3b02fdd5e5c612302b62a2dd52fcb38f9de97cb2afd827e8804db8**

The above alphanumeric string is the signature representing the sentence in the example above.



For more examples we can use the website:

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

In the case of the "Proof of Work" (PoW) algorithm, it works by using computational power in order to obtain a predefined hash.

Let's imagine we have the above hash that we took from the string "Mini BlocklyChain is modular".

**f41af7e61c3b02fdd5e5c612302b62a2dd52fcb38f9de97cb2afd827e8804db8**

At the beginning of this "hash" we put the difficulty parameter which is simply to put zeros "0" at the beginning, i.e. if we say that the difficulty is 4, it will have "0000" + "hash" and we will call this "seed hash".

**0000 f41af7e61c3b02fdd5e5c612302b62a2dd52fcb38f9de97cb2afd827e8804db8**

Now taking into account that we know the input information which is the string: "Mini BlocklyChain is modular" we add at the end of the string a number starting from zero "0" and we extract its hash which we will call "hash nonce":

**f41af7e61c3b02fdd5e5c612302b62a2dd52fcb38f9de97cb2afd827e8804db80**

We take out "hash nonce":

**7529f3ad273fc8a9eff12183f8d6f886821900750bb6b59c1504924dfd85a7c8**

We then perform a comparison of the new nonce hash with the seed hash. If they are equal, the node that first finds the equality will win the execution of processing the current transaction. As we can see this process is based on probability and computational strength of the device which gives the Proof of Work a fairness of consensus for all nodes.

In case the seed hash does not match the nonce hash, the difficulty is increased by one and the nonce hash is drawn again, the number being incremented is called the nonce number, and compared to the seed hash until they match or are the same.

As we can see, the nonce or increment is the number that will help to obtain the equality hash.

Based on the "Proof of Work" (PoW) algorithm, the Proof of Quantum (PoQu) algorithm is based on obtaining the "nonce" number as PoW does and using a minimum difficulty level ranging from 1 to 5, it serves only for the mobile device to earn the right to be a candidate to win the consensus.



The Proof of Quantum (PoQu), is activated when the mobile phone has completed the minimum PoW and earns the pass to obtain a probability number in the QRNG system.

The QRNG (Quantum Random Number Generator) is a Quantum Random Number Generator, this system is based on generating true random numbers based on quantum mechanics and is currently the most secure system for generating such numbers. For more details see "Quantum Computing Security" in Table of Contents 3.

COINsolidation can implement both minimum PoW and PoQu concession types.

The PoQu test is based on obtaining the number "nonce" this number in the PoQu test is known as "Magic Number" with this the "Peer to Peer" system will confirm if the number is correct and then a random number will be obtained with the COINsolidation QRNG server pool. This random number will be registered in all the nodes, a list will be created containing  $((\text{Sum of Nodes}) / 2) + 1$  and from this list the one with the highest percentage of probability to be the consensus winning candidate (PoQu) will be chosen and it will execute the current transaction queue.

The Proof of Quantum (PoQu) algorithm also uses **NIST** (National Institute of Standards and Technology) testing to ensure that QRNG random numbers are truly random numbers.

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

In COINsolidation we have implemented a block for the PoW and a block for the PoQu case. These blocks use one type of hash: SHA256 for free use, for commercial use we have SHA512 and other types of hashes depending on the requirement.

For more details on the HASH concept see:

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

NOTE: The Proof of Work (PoW) used on mobile phones can only use a maximum difficulty of 5 as the mathematical processing of these devices is not dedicated like servers or PCs. The (PoW) algorithm is only used to get the opportunity to get your pass or permission to enter the Quantum Random Number Generator (QRNG) system and with this to run the Proof of Quantum (PoQu) algorithm. See use of (PoQu) in Mini BlocklyChain:

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

## 5. Algorithm for the creation of a Consolidated Universal Address (CUA).

Mergers of companies are now commonplace, whether for economic, technological or market reasons.



We present the first crypto-asset or crypto-token fusion model based on an algorithm to create a consolidated address that is used and generated in the COINsolidation environment.

We create three types of consolidated addresses.

- **CUA** (Consolidated Universal Address) is used to consolidate and create a new ExoToken (asset) to be used by the user. The combination will be Cryptocurrency with Cryptocurrency. In the case of CUA you have the first series created called CUAG (Coinsolidated Universal Address Genesis).
- **HAC** (Hibric Address Consolidated) is used when we need to consolidate a cryptocurrency and/or token address and a normal asset transfer address.
- **DAC** (Dual Address Consolidated) is used to manage and consolidate two normal token addresses from the same blockchain or from two different technologies, they are simple asset transfer addresses referred to as Token-Token.

Let's start by looking at the advantages of the CUA.

A CUA address is made up of the address of the COINsolidation token (static address) and an additional token known as a "Coloured Coin" (variable address). In this case we can see that CUA addresses will always be made up of addresses of some combination of assets (cryptocurrencies or tokens).

In our case, when we consolidate the COINsolidation token and an OAP token, we will know it as the

"CUA genesis or CUAG (Consolidated Universal Address Genesis).

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

The Colored coin token is based on the Bitcoin blockchain and uses the Open Asset Protocol (OAP) standard.

Let's start by reviewing the potential and benefits of consolidating addresses.

- For users who create a CUA, a token (OAP) can be created that can be personalised by the user creating the CUA, the user will have the possibility of



having his own token or crypto-asset so that he can use it in the creation, support or expansion of his business(es), in a simple and easy way he will have an asset in the world of crypto-tokens.

- For companies that create a CUA, they will be able to have an OAP token that they can use to create value in their supply chain or use the asset in liquidity transactions based on the economic support of their company's assets and liabilities.
- For existing cryptocurrencies and tokens, by creating a CUA, you can use your address that identifies your asset and by consolidating it with the token (OAP) you can grow your demand by offering your current and future investors their own token for your users.

Our Consolidated Universal Addresses (CUA) are created using the following algorithm:

Step 1.- Select the Bitcoin and Ethereum addresses.

Bitcoin Address - (BTC). - address A1

**1Hjx3CanChCytqVz7vek1SSvN1momghJ42**

Ethereum address - (ETH) - address A2

**0x9d08c0ac0f2fdf078c883db6fa617b15776e4b41**

Step 2.- The SHA512(address String-Text) of each address is obtained by removing the first element of each address and taking from SHA512 the two characters located in places 120 to 121 of each hash operation symbolised by "U". Verifier numbers.

$U(\text{SHA512}(\text{Hjx3CanChCytqVz7vek1SSvN1momghJ42})) = \text{bf}$

$U(\text{SHA512}(\text{x9d08c0ac0ac0f2fdf078c883db6fa617b15776e4b41})) = \text{28}$

Step 3.- The first element (character) of each address is taken starting with the address with the smallest number of elements and the string "10" is obtained.

$A_{10}[0] = 1$

$A_{20}[0] = 0$

Step 3.- The SHA512 is obtained without the elements of step 3 and only the four characters from 120 to 123 are taken.



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

Step 4.- The characters of each address are concatenated one by one, starting with the address that has the least number of characters, if they have the same number of characters, the concatenation can start from any address.

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

Address concatenation:

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

Last characters that cannot be concatenated are put at the beginning of the string and step 3 is appended to the end of the concatenated string.

**776e4b41** Hxj9xd30C8acn0CahcC0yft2qfVdzf70v7e8kc18S8S3vdNb16mfoam6g1h7Jb4125 **10**

Step 4.- The number of characters in the string that could NOT be concatenated in step 4 is added at the beginning.

8776e4b41Hxj9xd30C8acn0CahcC0yft2qfVdzf70v7e8kc18S8S3vdNb16mfoam6g1h7Jb412510

Step 5.- At the beginning of the string two XX integer verifiers are added to help us to verify if the difference of the strings (subtraction) the bigger one minus the smaller one, must always give a positive integer, this pair of integers will help us to avoid errors in the concatenation. In case the difference is less than or equal to "9" the verifier number will be "00" in case it is greater than "9" the difference will be marked in these two digits.

In our case of the generation between Bitcoin and Ethereum always the two digits

verifiers shall be "00".

008776e4b41Hxj9xd30C8acn0CahcC0yft2qfVdzf70v7e8kc18S8S3vdNb16mfoam6g1h7Jb412510

Step 6.- The two pairs of verifiers from step 2 in each direction are concatenated at the beginning of the string resulting from step 3 in the same order  $A_1 + A_2$ .

bf28008776e4b41Hxj9xd30C8acn0CahcC0yft2qfVdzf70v7e8kc18S8S3vdNb16mfoam6g1h7Jb412510

Step 7.- The 4 digits of the SHA512 from step 3 are integrated at the end of the string.

**bf28008776e4b41Hxj9xd30C8acn0CahcC0yft2qfVdzf70v7e8kc18S8S3vdNb16mfoam6g1h7Jb412510140c**



Step 8.- The **CUA** (Consolidated Universal Address "Genesis") ID is integrated at the beginning of the address created in step 5.

CUAfd55008776e4b41Hxj9xd30C8acn0CahcC0yft2qfVdzf70v7e8kc18S8S3vdNb16mfoam6g1h7Jb412510140c

In the case of Bitcoin and Ethereum address consolidation, it will give an address consisting of **90** hexadecimal characters.

**NOTE:** the size of the CUA, HAC and DAC addresses may vary from case to case depending on the addresses that comprise them.

## Project and solution by COINsolidation.

Currently there are different types of Blockchain oriented to assets of different characteristics, this leads to having an infinite number of types of addresses for daily use and having to keep an exhaustive control to avoid making transfer errors.





On the other hand, the world of cryptocurrencies and tokens is limited to financial experts or blockchain technology experts, so it is difficult for the average person to venture into the creation of their own cryptocurrency or token.

We have solved both of the above problems in COINsolidation by implementing the following points and/or tools that we have created.

For the point of address control of different blockchain addresses, we create an algorithm where we consolidate (join) two or more addresses in their different combinations resulting in a single address of type CUA, HAC and/or DAC.

With this solution, instead of sending two addresses from the same or different blockchain, only one consolidated address will be used.

For the second problem we have used the programming methodology called Blockly, this is a visual tool where no great programming knowledge is needed and any average person or company will be able to create their own applications without having to invest expensive development equipment, time and money.

We have created the extensions (modules) just install and use them to create mobile applications, in 15 minutes. Example your own cryptocurrency exchange or develop your own currency (token) in a matter of minutes. All this using state-of-the-art data security called PQC (Post-Quantum Cryptography).

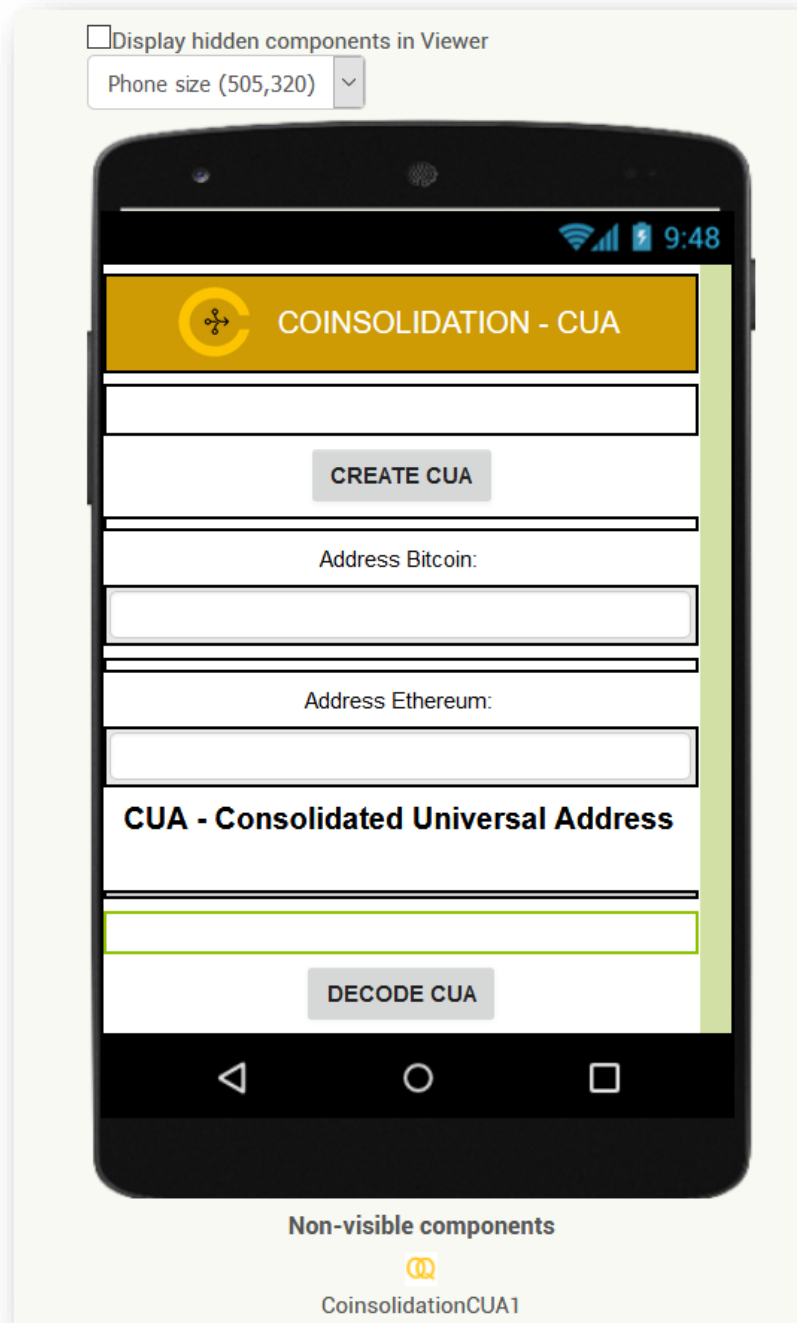
Just install the extensions in any free-to-use tool such as Appinventor, AppyBuilder, Thunkable, Kondular or others and in minutes you can enter the world of cryptocurrencies and token creation all in the palm of your hand.

Finally, COINsolidation is creating the use of low-cost quantum security (software and hardware) that can be used to protect computer data at home. At present, technologies based on quantum computing and security are very expensive and only corporations with a high financial level can create and use them. However, in COINsolidation we believe that new technologies should be available to everyone, the fairness of use of Blockchain and Quantum Computing should be for everyone, we create free to use software (cryptocurrencies) and low cost hardware (quantum security).

## **6. Creation of App CUA (Consolidated Universal Address) in 15 minutes.**

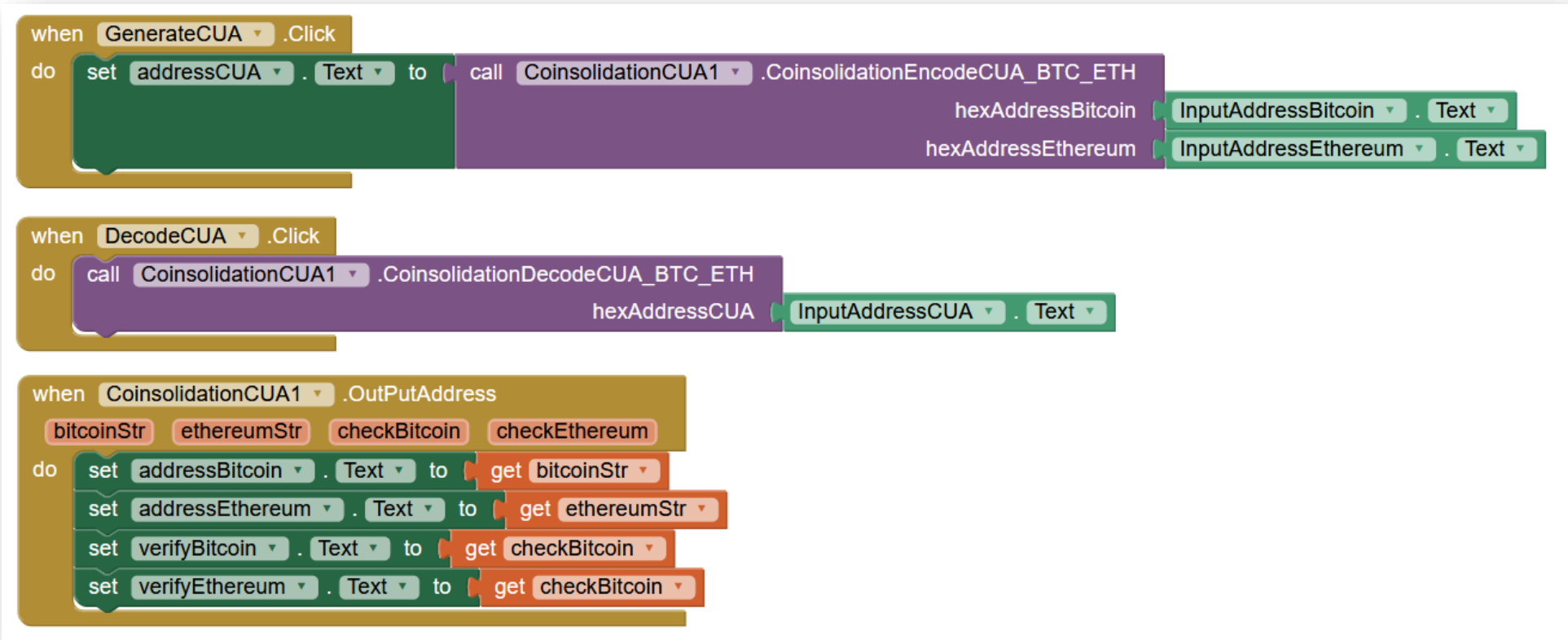
App for Bitcoin and Ethereum coins (BTC-ETH)

5-minute screen design at <https://appinventor.mit.edu/>





Use of CoinsolidatioCUA.AIX extension (5 minutes).





Create the application in **Menu > Build > App** (provide QR code for .apk) - (5 minutes).

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

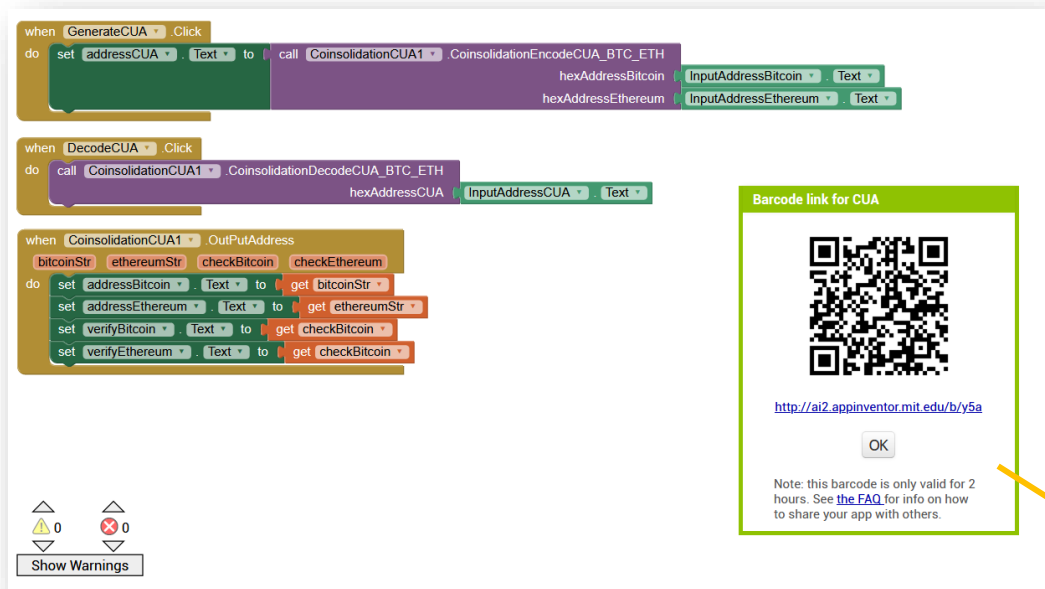
35%

Compiling part 2 (please wait)

🗑



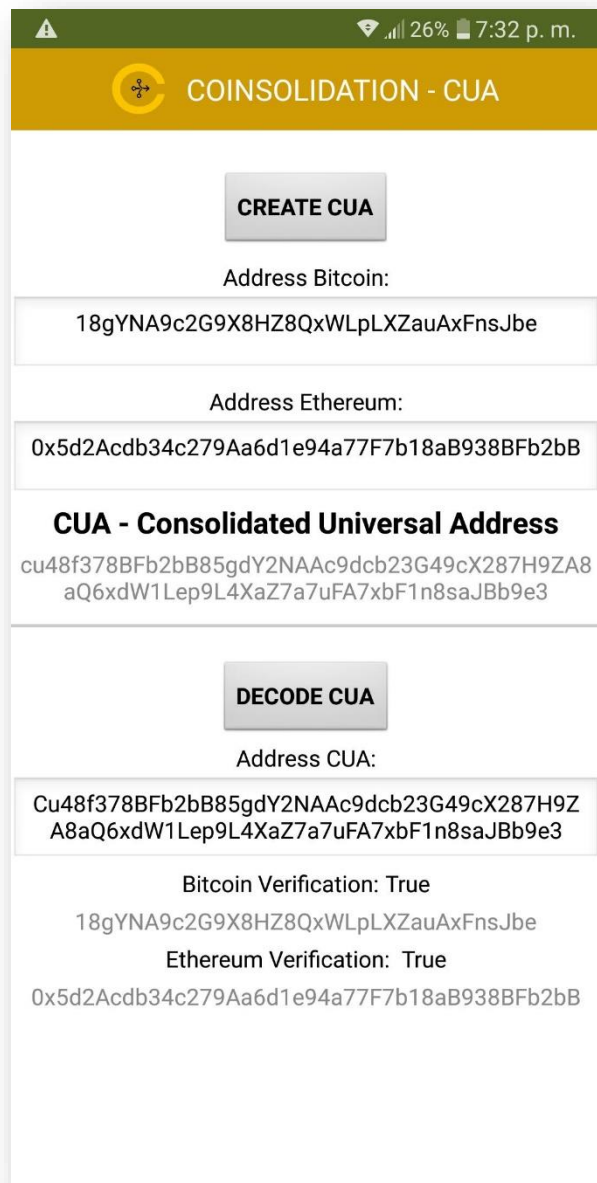
Install the application on the mobile phone from the QR using the AppInventor Android application (MIT AI2 Companion) - <https://play.google.com/store/apps/details?id=edu.mit.appinventor.aicompanion3>



**NOTE:** The application APK file ready to install can be found in the following repository:

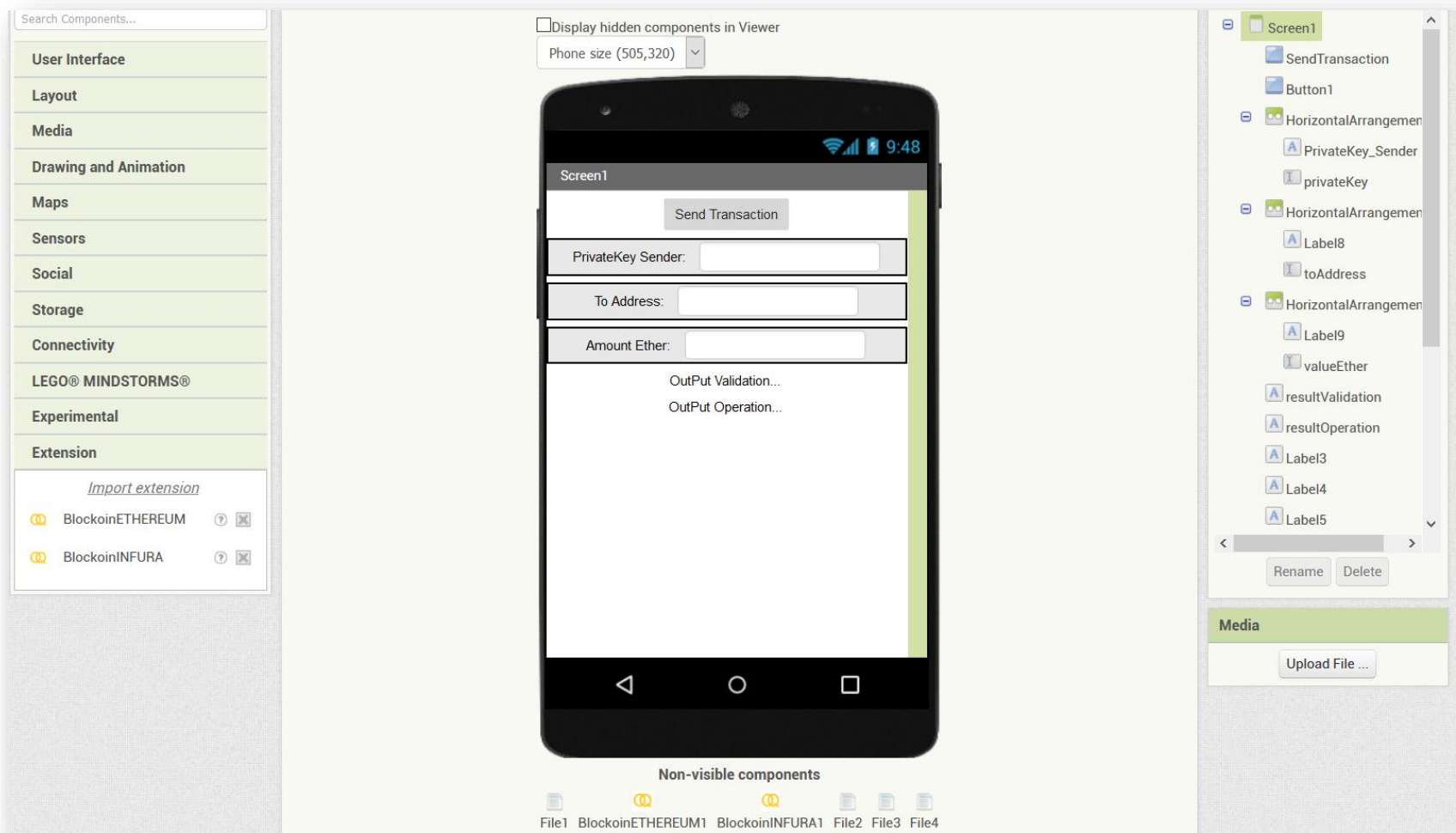
<https://github.com/COINsolidation/App>

To review the Java code for the CUA extension generation and implementation algorithm for the generation of consolidated universal addresses see the Annex "CUA Algorithm Code" or consult the code link: <https://github.com/COINsolidation/source>





7. Create your Ethereum cryptocurrency exchange on Android in just 15 minutes.  
Design in App Inventor (Screen). - 5 minutes.





Function (eth\_SendTransactionEasy) and event (OutPutSendTransactionEasy) blocks. - 5 minutes

when SendTransaction .Click

do

call BlockoinETHEREUM1 .eth\_sendTransactionEasy

hexPrivateKeySender privateKey . Text

toAddress toAddress . Text

valueEther valueEther . Text

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 "

Input data:

PrivateKey: Primary key of the sender's address.

toAddress: Hexadecimal address of the receiver.

valueEther: Give the amount of Ether to be sent.

Save the results in text files:

File1 function: File **trasactionValidation.txt**

Save the results in text files:

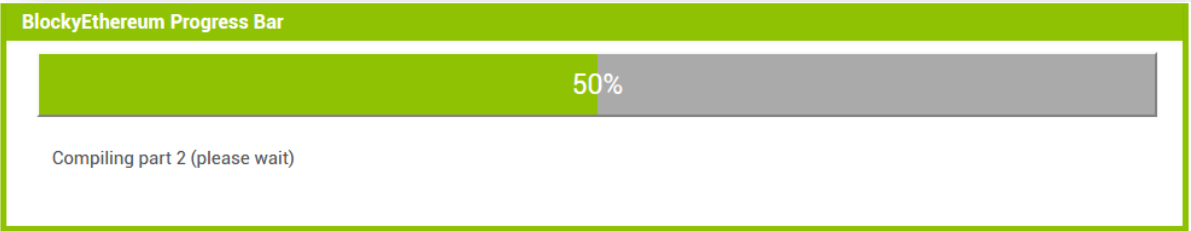
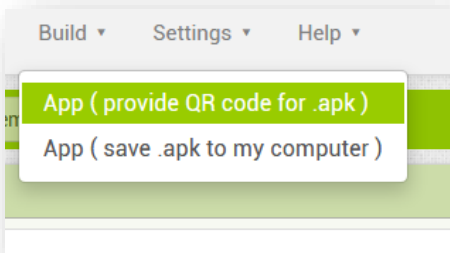
Function File2: File **trasactionValidation.txt**

For more details see the Ethereum Exchange Extension (EEE) User Guide in the repository: <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>



Compile, generate APK file to install it on the Android device. - 5 minutes



NOTE: When executing the transaction it will take approximately 6 to 8 seconds to release the "Send Transaction" button. Due to the connection time to the Ethereum network.

For more details on the EEE extension - (Ethereum Exchange Extension). See the EEE user manual at the link:

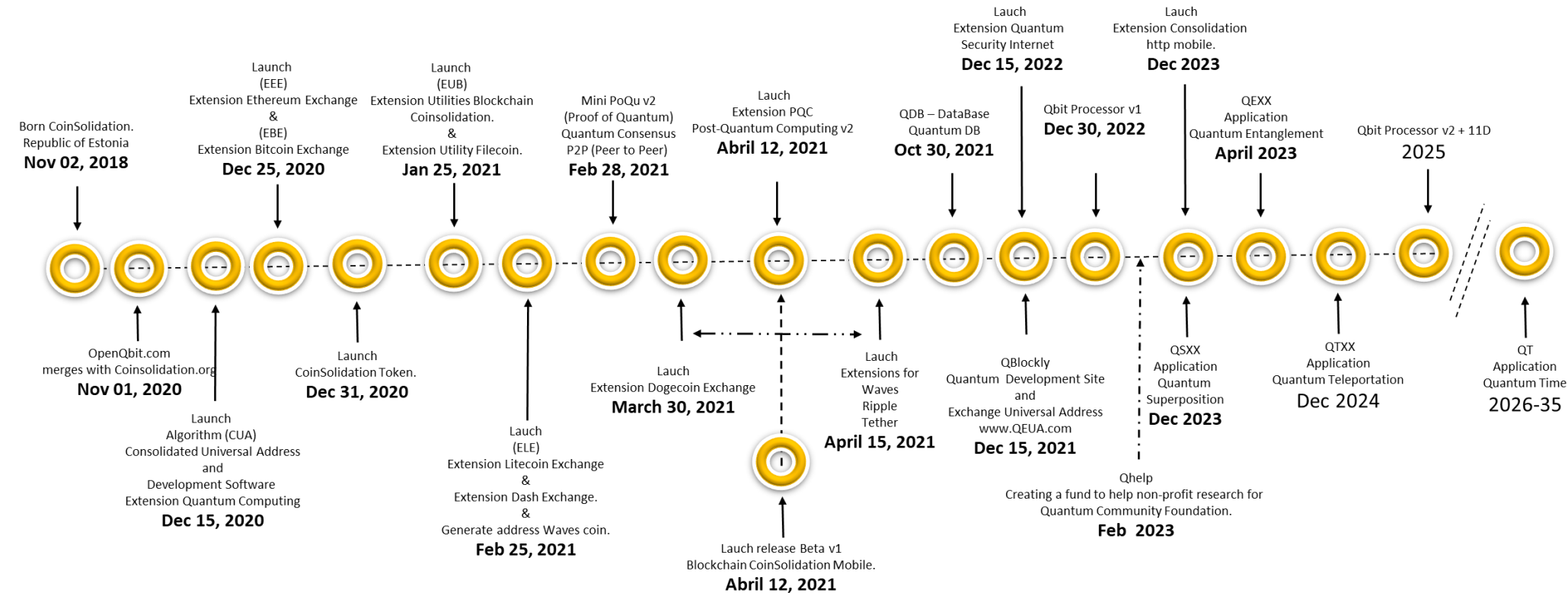
<https://github.com/COINsolidation/UserGuide>





8. Roadmap COINsolidation.

ROADMAP



\*OpenQbit.com merges with COINsolidation.org (Nov 01, 2020) / OpenQbit specializes in Quantum Computing and Security Quantum. The quantum processor version 1 will be using basic quantum logic gates for home use.



EXchange  
tensions

## 9. COINsolidation Token (CUAG) - ICO DISTRIBUTION PLAN.

The ICO is divided into three stages:

|                  |             |                               |                                    |
|------------------|-------------|-------------------------------|------------------------------------|
| 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)   | SOFT CAPITAL: \$ 280,000 USD       |
| ICO SECOND PHASE | \$ 0.05 USD | (1/Jul 2021 - 31/Jul 2021)    |                                    |

| 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. General characteristics of the COINsolidation token:

Name: COINsolidation

Symbol: CUAG - (Consolidated Universal Address Genesis).

Total tokens created: 40,000,000,000.00

Number of decimals: 18

Country of Launch: Estonia

Official website: [www.COINsolidation.org](http://www.COINsolidation.org)

Company: COINsolidation International.

Launch date: April 30, 2021

Addressing algorithm: Consolidated Universal Address (CUA).

Security employed: PQC (Post-Quantum Cryptography) based on quantum computing.

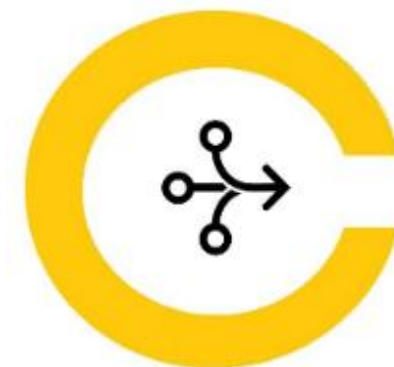
Technological proposal: Extensions for Blockly systems for the use of cryptocurrencies and implementation of quantum security.

Partnerships or technology agreements (merger):

Company: OpenQbit Inc.

Industry: Quantum computing and PQC (Post-Quantum Cryptography).

Official website: [www.OpenQbit.com](http://www.OpenQbit.com) / [www.coinsolidation.org](http://www.coinsolidation.org)





## 11. Basic concepts applied to Blockchain platforms.

### What is a blockchain?

The blockchain is generally associated with Bitcoin and other cryptocurrencies, but these are only the tip of the iceberg as it is not only used for digital money, but can be used for any information that may be of value to users and/or companies. This technology, which has its origins in 1991, when Stuart Haber and W. Scott Stornetta described the first work on a cryptographically secured blockchain, was not noticed until 2008, when it became popular with the arrival of bitcoin. But its use is now in demand in other commercial applications and is projected to grow in the medium future in various markets, such as financial institutions and the Internet of Things IoT among other sectors.

The blockchain, better known by the English term blockchain, is a single, consensual and distributed record in several nodes (electronic devices such as PCs, smartphones, tablets, etc.) of a network. In the case of cryptocurrencies, we can think of it as the ledger where each transaction is recorded.

How it works can be complex to understand if we delve into the internal details of its implementation, but the basic idea is simple to follow.

In each block is stored:

- 1.- a number of valid records or transactions,
- 2.- information relating to that block,
3. linking it to the previous block and the next block through the hash of each block – a unique code that would be like the block's fingerprint.

Therefore, **each block has a specific and immovable place within the chain**, as each block contains information from the hash of the previous block. The entire chain is stored on each node of the network that makes up the blockchain, so **an exact copy of the chain is stored on all network participants**.

### What is an address or account within the Ethereum blockchain platform?

It is a string of 42 characters in the Ethereum platform that represents a number in hexadecimal base, where the assets defined in Ethereum will be deposited or sent. In other blockchain platforms the number of characters in the account or address can be different, e.g:

**0x5d2Acdb34c279Aa6d1e94a77F7b18aB938BFb2bB**



### **What is a cryptocurrency?**

A digital or virtual currency designed to function as a medium of exchange. It uses cryptography (digital security) to secure and verify transactions, as well as to control the creation of new units of a particular cryptocurrency.

### **What is a token?**

Tokens are digital assets that can be used within the ecosystem of a given project.

The main distinction between tokens and cryptocurrencies is that the former require another (not their own) blockchain platform to function. Ethereum is the most common platform for creating tokens, mainly because of its smart contract function. Tokens created on the Ethereum blockchain are generally known as ERC-20 tokens, although there are other more specialised types of tokens, for example the ERC-721 token used mainly for collectible assets (cards, use in video games, artwork, etc.).

### **What is an Exchange?**

A cryptocurrency exchange is the meeting point where cryptocurrencies are exchanged for fiat money or other cryptocurrencies. These online exchanges are where the market price is generated, which determines the value of cryptocurrencies based on supply and demand.

### **What is Exchange Rates?**

These are the rates of the value of an Ether or other cryptocurrency in the currency of circulation of each country, for example, as of the date of the creation of this manual, one Ether has a value in US dollars of \$430.94.

### **What is a transaction?**

It is the execution or transfer of some type of non-tangible asset that can be given a pre-set value within the Ethereum system and can subsequently be changed to a tangible value for a company or person.

### **What is txHash?**

It is a hexadecimal number that helps to track the detailed result of each transaction.

### **What types of transactions are there?**

There are two types, one is the "offline" transaction which is created without the need to be connected to the main Ethereum network, they can be stored until you choose to connect to the Ethereum network and release the transaction, they have the advantage of security



as the entire transaction is processed offline which prevents any anomaly that could be in the network connection. The other transaction is the "online" which always needs to be connected to the internet with the advantages and disadvantages of security that it brings.

### **What is an address on a Blockchain?**

An address or account is composed of three parts, the address, the public key and the private key, these two keys are a string of numbers and characters in hexadecimal format that are used to send and receive (assets) or ether (digital currency).

The primary key should never be shared with anyone as it is the one that authorises the release of the balance (signs transactions) held in the account.

The public key is known to the public and is shared to anyone as it is the reference to confirm that the transaction is correct both in value and to whom it is sent.

Examples of Ethereum network address components:

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



## 12. What is Blockly programming?

**Blockly** is a **visual programming methodology** composed of a simple set of commands that we can combine as if they were the pieces of a puzzle. It is a very useful tool for those who want to **learn to program in** an intuitive and simple way or for those who already know how to program and want to see the potential of this type of programming. It is based on the JavaScript language and was developed by Google and MIT.

Blockly is a form of programming where you don't need any background in any kind of computer language or computer science, this is because you only need to join graphic blocks as if you were playing lego or a puzzle, you just need to have a little logic and that's it!

Anyone can create programmes for mobile phones (smartphones) without messing with those hard-to-understand programming languages, just put together blocks in a graphical way that is simple, easy and quick to create.

## 13. Annex "Code for CUA algorithm".

Github reference: <https://github.com/coinsolidation/source>

## 14. Terms.

Terms and conditions of use see on the website [www.coinsolidation.org](http://www.coinsolidation.org) or <https://github.com/coinsolidation/Terms>

Support with commercial use.

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

Sales commercial business use of blockchain.

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

Legal information and licensing questions or concerns.

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

Social networking:

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

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

