



Instalación, configuración & administración.

Manual usuario

versión 1.0 Beta

Junio 2020.

MiniQRNG es una marca registrada de OpenQbit Inc, bajo licencia de uso libre y comercial.
Términos y condiciones de uso en: www.OpenQbit.com

Contenido

1.	Introducción.....	3
2.	¿Qué es la programación Blockly?.....	4
3.	¿Qué es Termux?.....	4
4.	¿Qué es Mini QRNG ?.....	4
5.	Configuración de almacenamiento “storage” dentro de Termux.....	8
6.	Instalación de servidor SSH (Secure Shell).....	9
7.	Configuración de servidor SSH en teléfono móvil (smartphone).....	10
8.	Ambientes Blockly (App Inventor, AppyBuilder y Thunkable).	16
9.	Definición y uso de bloques en Mini QRNG.....	17
10.	Creacion de dispositivo “Hardware” de un QRNG.....	28
11.	Anexo “Computación Cuántica con OpenQbit”.....	33
12.	Licenciamiento y uso de software.	38

1. Introducción.

La criptografía de hoy se basa en secuencias de números aleatorios. Los generadores de números pseudoaleatorios actualmente en uso parecen proporcionar secuencias de bits aleatorias, pero en realidad estas secuencias de bits tienen ciertos patrones, por lo que existe el riesgo de ser pirateado o de alguna forma manipulados para que la información que viaja por las redes públicas y privadas.

La integración de fuentes de entropía física en generadores de números aleatorios es el método más común para superar esta amenaza de seguridad. Sin embargo, la física clásica es causal, por lo tanto, la imprevisibilidad de una secuencia de bits generada con la física clásica no se puede probar.

La física cuántica, por otro lado, es aleatoria por esencia. Los números generados por un generador de números aleatorios cuánticos (QRNG – Quantum Random Number Generator) no se pueden predecir: QRNG es demostrablemente impredecible. Entonces, si se usa un generador de números aleatorios cuánticos en un sistema de seguridad, incluso una supercomputadora rápida con una operación aritmética rápida no puede predecir las secuencias de bits aleatorias utilizadas por este sistema de seguridad.

La física cuántica usa métodos basados en un concepto fundamental llamada entropía.

TIPOS DE ENTROPÍA

Hay dos tipos generales de fuentes de entropía que pueden medirse para generar números aleatorios verdaderos. El primer tipo incluye un proceso físico que es difícil o imposible de medir o demasiado computacionalmente intenso para predecir, o ambos. Esta es una fuente de caoticentropía. Un ejemplo común conocido por la mayoría de la gente es una máquina de lotería. Se coloca un conjunto de bolas numeradas secuenciales en una cámara y se mezclan constantemente haciendo girar la cámara o soplando aire a través de la cámara. Se permite que varias de las bolas caigan fuera de la cámara y los números marcados en las bolas representan el sorteo de la lotería. El sorteo es aleatorio debido a la gran cantidad de interacciones entre las bolas y la cámara que resulta en un número rápidamente creciente de posibles movimientos de cada bola. No solo la complejidad de estas interacciones es extremadamente alta, no hay una forma aparente de observar o medir con precisión todas las variables internas de las bolas, la cámara y el flujo de aire. Un segundo tipo muy diferente de fuente de entropía es la mecánica cuántica. Muchas partículas microscópicas u ondas, como los fotones, los electrones y los protones tienen propiedades mecánicas cuánticas que incluyen la rotación, la polarización, la posición y el momento. Dada la configuración adecuada para producir estas partículas, los valores específicos de su rotación o polarización, por ejemplo, no solo son desconocidos y teóricamente impredecibles, sino que se determinan físicamente hasta que se realiza una medición.

2. ¿Qué es la programación Blockly?

Blockly es un **lenguaje 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.

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.

3. ¿Qué es Termux?

Termux es un emulador de terminal Android y una aplicación de entorno Linux que funciona directamente sin necesidad de enrutamiento o configuración. Un sistema base mínimo se instala automáticamente.

Usaremos Termux por su estabilidad y fácil instalación y manejo, sin embargo, se puede usar un ambiente instalado de Linux Ubuntu para Android.

En este ambiente de Linux tendrá el “core” de los procesos de comunicación del MiniQRNG.

4. ¿Qué es Mini QRNG ?

Mini QRNG es Software y Hardware que incluye tres soluciones tecnológicas para poder crear QRNGs (Quantum Random Number Generators) – Generadores de números cuánticos aleatorios. Clasificados como sigue:

a.- QRNG API. - Gnerador de números cuánticos aleatorios obtenidos desde servidores externos.

b.- MiniQRNG / Software. - Generador de números cuánticos aleatorios obtenido usando las propiedades físicas (cuántica) de la cámara del teléfono móvil.

c.- MiniQRNG / Hardware. - Generador de números cuánticos obtenido usando hardware basado en las propiedades de la física cuántica de un laser. Mas adelante decimos como armarlo en casa a un bajo costo.

1. Instalación y configuración de Terminal Termux.

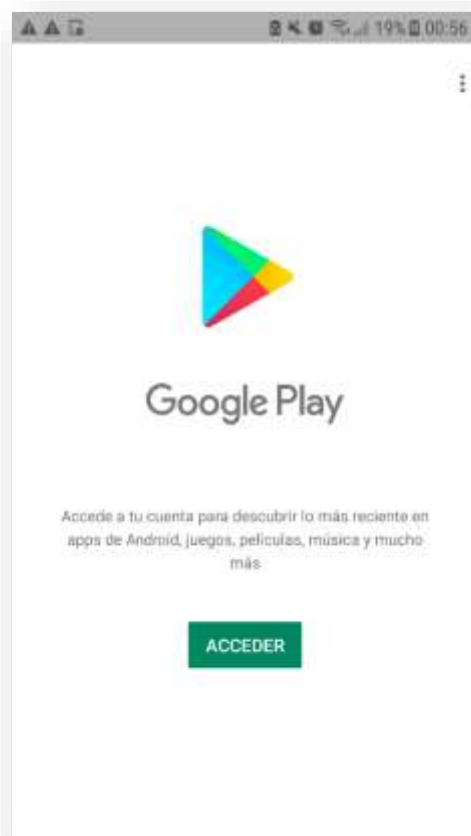
Necesitamos primero un ambiente Linux ya que todo sistema Android está basado en Linux por seguridad y flexibilidad en herramientas, usaremos la terminal “Termux” que contiene dicho ambiente en donde instalaremos la(s) herramienta(s) que nos ayudaran para la creación de QRNGs.

Termux es un emulador de Linux donde instalaremos los paquetes necesarios para crear de números cuanticos.

Una de las principales ventajas al usar Termux es que podrás instalar programas sin tener que “rootear” el móvil (Smartphone) esto asegura que por esta instalación no se pierde ningún tipo de garantía del fabricante.

Instalación de Termux.

Desde tu móvil ve a la aplicación icono de Google Play (play.google.com).



Busca por aplicación “Termux”, selecciónala e iniciar el proceso de instalación.



Inicio de la aplicación Termux.

Después de iniciar tendremos que ejecutar los siguientes dos comandos para realizar actualizaciones del emulador del sistema operativo Linux:

\$ apt update

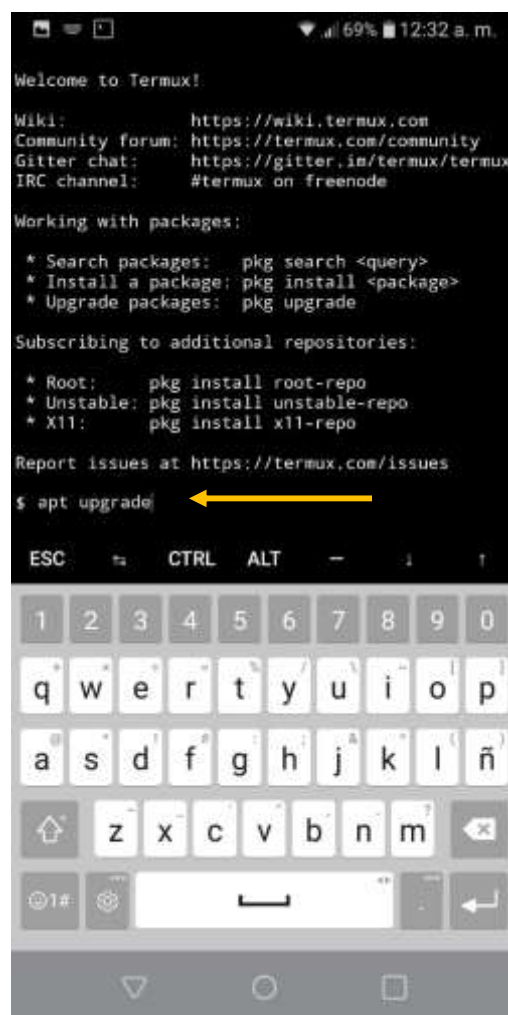
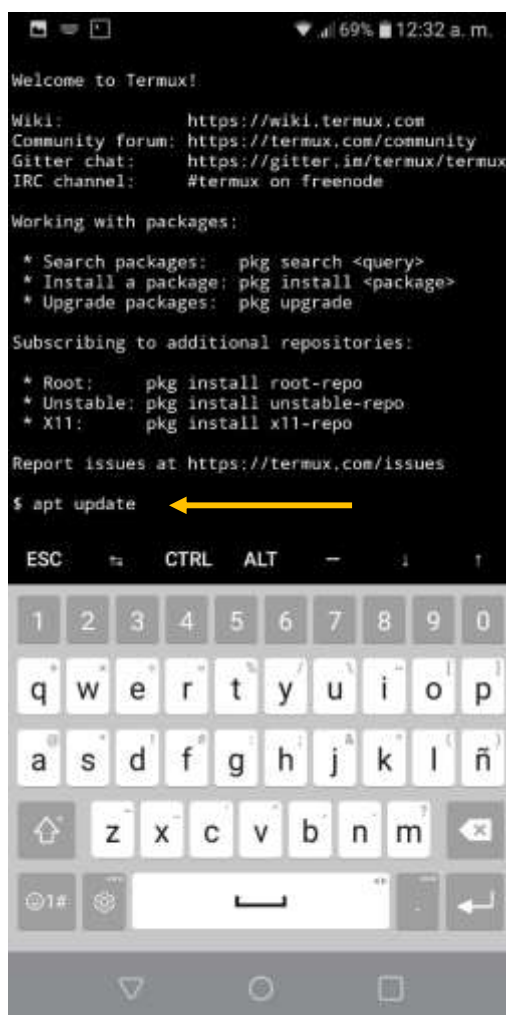
\$ apt upgrade

Confirmar todas las opciones Y(Yes)....

Inicio de Termux

\$ apt update

\$ apt upgrade



5. Configuración de almacenamiento “storage” dentro de Termux.

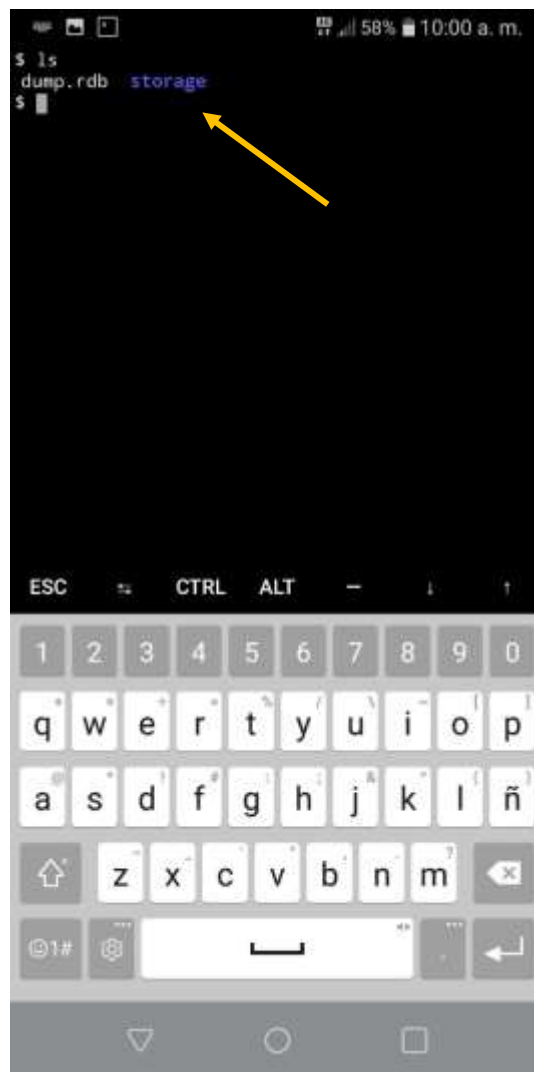
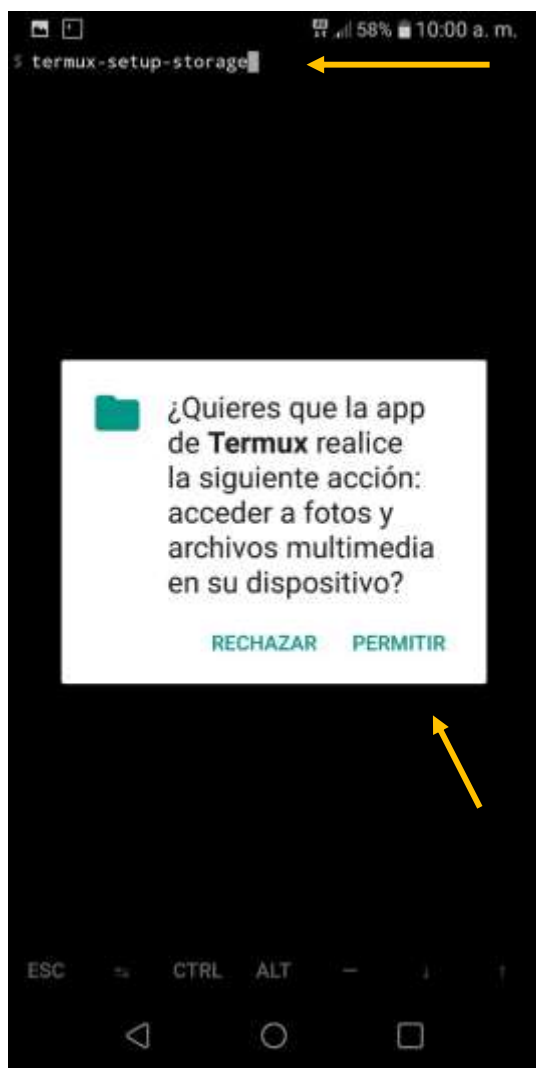
Después de haber realizado el update y upgrade del sistema Termux, empezaremos con la configuración de como poder ver el almacenamiento interno del teléfono en el sistema de Termux esto os ayudara a poder realizar intercambio de información entre Termux y nuestra información que tenemos en el teléfono.

Esto lo podemos realizar de una forma sencilla y rápida ejecutando el siguiente comando en una terminal de Termux.

\$ termux-setup-storage

Al ejecutar el anterior comando nos aparece una ventana pidiendo la confirmación de la creación de un **storage** virtual (directorio) en Termux damos click botón “PERMITIR” y creara el enlace al almacenamiento del teléfono. Verificamos dando el comando:

\$ ls



6. Instalación de servidor SSH (Secure Shell).

```
$ apt install openssh
```

```
$ apt install sshpass
```

```
$ apt install openssh
```



```
$ apt install openssh
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
krb5 ldns libdb libedit termux-auth
The following NEW packages will be installed:
krb5 ldns libdb libedit openssh termux-auth
0 upgraded, 6 newly installed, 0 to remove and 0
not upgraded.
Need to get 2255 kB of archives.
After this operation, 11.9 MB of additional disk
space will be used.
Do you want to continue? [Y/n] Y
Get:1 https://dl.bintray.com/termux/termux-packa
ges-24 stable/main arm libdb arm 18.1.32-4 [465
kB]
Get:2 https://dl.bintray.com/termux/termux-packa
ges-24 stable/main arm krb5 arm 1.18.1 [839 kB]
24% [2 krb5 131 kB/839 kB 16%]
```

```
$ apt install sshpass
```



```
$ apt install sshpass
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
sshpass
0 upgraded, 1 newly installed, 0 to remove and 0
not upgraded.
Need to get 7158 B of archives.
After this operation, 57.3 kB of additional disk
space will be used.
0% [Working]
```

Hemos terminado con la instalación de la red de comunicaciones para servidor SSH localhost en móvil Smartphone.

7. Configuración de servidor SSH en teléfono móvil (smartphone).

Habilitaremos el servidor de SSH en el teléfono móvil para poder conectarnos desde nuestra PC al móvil y poder trabajar de una forma más rápida y cómoda, así mismo nos servirá para comprobar que el servicio del servidor SSH en el móvil funciona correctamente ya que este lo utilizaremos en la comunicación con Mini QRNG.

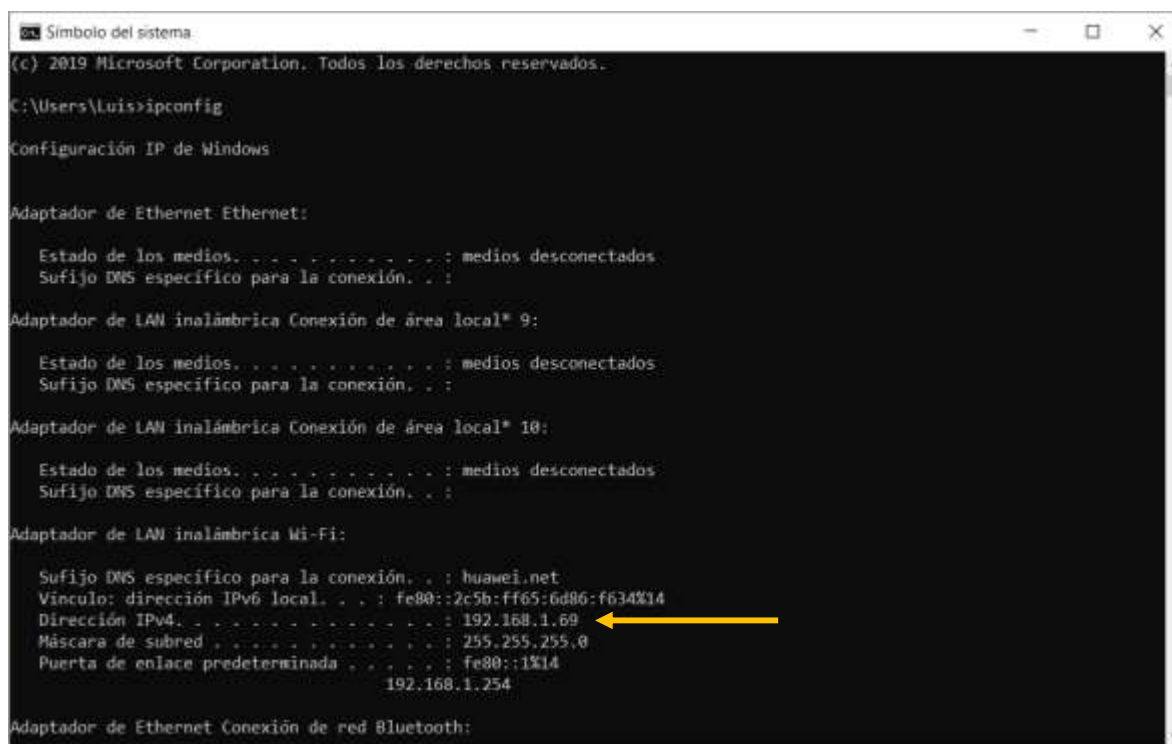
Lo primero que tenemos que hacer es conectar **a la misma red WiFi** el móvil y la PC para que se puedan ver. Las IPs o direcciones deben ser similares a 192.168.XXX.XXX los valores XXX son números variables que se asignan aleatoriamente en cada equipo.

Este ejemplo se probó en un móvil LG Q6 y una PC con Windows 10 Home.

Revisar la IP o dirección que tiene la PC conectada al WiFi deberemos abrir una terminal en Windows.

En el panel inferior donde está la lupa de buscar escribir cmd y presionar la tecla Enter. Se abrirá una terminal y en esta escribimos el comando:

```
C:\Users\nombre_usuario> ipconfig
```



```
(c) 2019 Microsoft Corporation. Todos los derechos reservados.

C:\Users\Luis>ipconfig

Configuración IP de Windows

Adaptador de Ethernet Ethernet:

    Estado de los medios. . . . . : medios desconectados
    Sufijo DNS específico para la conexión. . . :

Adaptador de LAN inalámbrica Conexión de área local* 9:

    Estado de los medios. . . . . : medios desconectados
    Sufijo DNS específico para la conexión. . . :

Adaptador de LAN inalámbrica Conexión de área local* 10:

    Estado de los medios. . . . . : medios desconectados
    Sufijo DNS específico para la conexión. . . :

Adaptador de LAN inalámbrica Wi-Fi:

    Sufijo DNS específico para la conexión. . . : huawei.net
    Vínculo: dirección IPv6 local. . . : fe80::2c5b:ff65:6d86:f634%14
    Dirección IPv4. . . . . : 192.168.1.69
    Máscara de subred . . . . . : 255.255.255.0
    Puerta de enlace predeterminada . . . : fe80::1%14
                                      192.168.1.254

Adaptador de Ethernet Conexión de red Bluetooth:
```

Nos mostrara la IP que tiene asignada la PC en este es la 192.168.1.69 sin embargo esta los más probable es que sea diferente en cada caso.

NOTA: debe tomarse la dirección donde dice “Dirección IPv4” no confundir con la Puerta de enlace.

Ahora en el caso del teléfono móvil en la terminal de Termux debemos teclear el siguiente comando para saber cómo se llama nuestro usuario que usaremos para conectarnos al servidor SSH que tiene nuestro teléfono, ejecutamos el siguiente comando:

\$ whoami

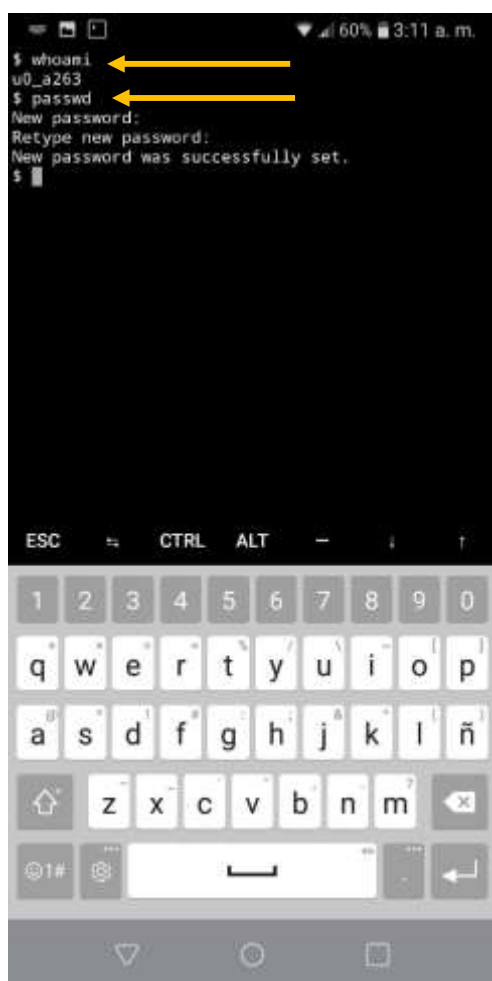
Posteriormente debemos darle un password a este usuario por lo que tenemos que ejecutar el siguiente comando:

\$ passwd

Nos pedirá que tecleemos un password y damos Enter, nuevamente nos pide el password confirmamos le damos el mismo y damos Enter, si ha sido exitosamente **“New password was successfully set”** en caso de marcar un error es posible que el password no se haya tecleado correctamente. Volver a realizar el procedimiento.

Y después para saber que IP tenemos en Termux tecleamos el siguiente comando, la IP esta después de la palabra **“inet”**:

\$ ifconfig -a



```
$ whoami
u0_a263
$ passwd
New password:
Retype new password:
New password was successfully set.
$
```



```
e 0
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>
> mtu 1500
    inet 192.168.1.68 netmask 255.255.255.0
    broadcast 192.168.1.255
    inet6 fe80::257:c1ff:fee6:3051 prefixlen 64 scopeid 0x20<link>
    unspec 00-00-00-00-00-00-00-00-00-00-00-00 txqueuelen 1000 (UNSPEC)
    RX packets 908745 bytes 947916536 (904.0 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 601034 bytes 93496881 (89.1 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

$
```

Ahora es momento de iniciar el servicio del servidor de SSH en el teléfono para que pueda recibir sesiones desde la PC. Ejecutamos el siguiente comando en la terminal de Termux, este comando no arroja ningún resultado.

\$ sshd



Ahora tendremos que instalar un programa en la PC que se comunicara con el servidor SSH del teléfono desde la PC.

Tenemos que ir a la página <https://www.putty.org>

Seleccionar donde se encuentra el enlace “You can download PuTTY here”



Download PuTTY

PuTTY is an SSH and telnet client, developed originally by Simon Tatham for the with source code and is developed and supported by a group of volunteers.

You can download PuTTY [here](#).

Below suggestions are independent of the authors of PuTTY. They are *not* to be seen a



Bitvise SSH Client

Bitvise SSH Client is an SSH and SFTP client for Windows. It is developed and supported prof supports all features supported by PuTTY, as well as the following:

- graphical SFTP file transfer;
- single-click Remote Desktop tunneling;
- auto-reconnecting capability;
- dynamic port forwarding through an integrated proxy;
- an FTP-to-SFTP protocol bridge.

Bitvise SSH Client is **free to use**. You can [download it here](#).

Escoger la versión de 32 bit, no importa si tu sistema es de 64 bits funcionara bien.

Download PuTTY: latest release

[Home](#) | [FAQ](#) | [Feedback](#) | [Licence](#) | [Updates](#) | [Mirrors](#)
Download: [Stable](#) | [Snapshot](#) | [Docs](#) | [Contact](#)

This page contains download links for the latest released version of PuTTY. Currently this is 0.73, released on 2019-09-29.

When new releases come out, this page will update to contain the latest, so this is a good page to bookmark or link to. Alternative

Release versions of PuTTY are versions we think are reasonably likely to work well. However, they are often not the most up-to-date. You can also download the [development snapshots](#), to see if the problem has already been fixed in those versions.

Package files

You probably want one of these. They include versions of all the PuTTY utilities.

(Not sure whether you want the 32-bit or the 64-bit version? Read the [FAQ entry](#).)

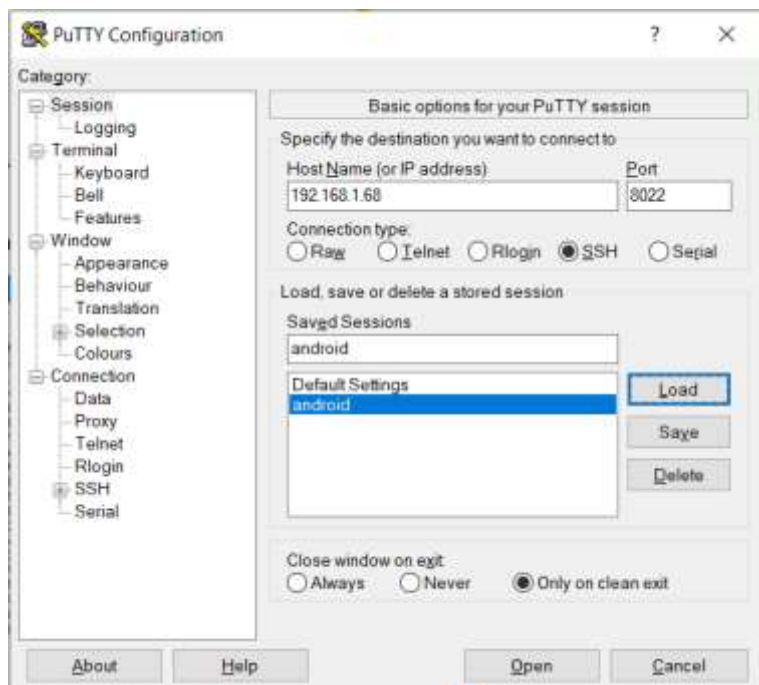
MSI ('Windows Installer')

32-bit:	putty-0.73-installer.msi	(or by FTP)	(signature)
64-bit:	putty-64bit-0.73-installer.msi	(or by FTP)	(signature)

Unix source archive

.tar.gz:	putty-0.73.tar.gz	(or by FTP)	(signature)
----------	-----------------------------------	-------------	-------------

Ya que se haya bajado en tu PC ejecútalo e instalado con las opciones por default. Después inicia la aplicación de PuTTY.



En esta sesión introduciremos los datos de nuestro servidor Openssh que instalamos en el teléfono móvil.

Introducir la IP del teléfono móvil.

HostName or IP address:

192.168.1.68 (ejemplo IP)

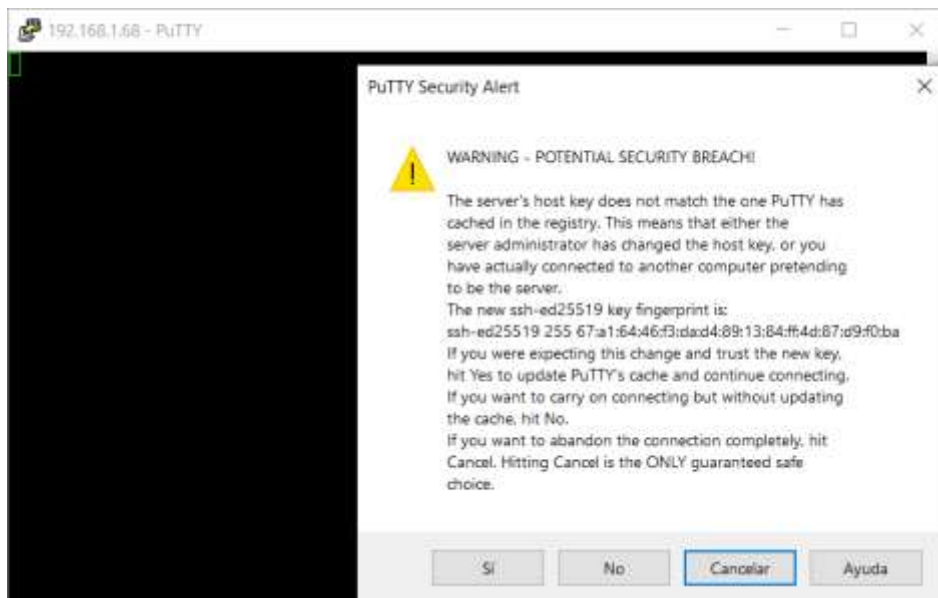
Port:

8022 (Puerto por default del servidor SSH del móvil).

Podemos dar un nombre de la sesión en "Saved Sessions" y damos click en el botón Save.

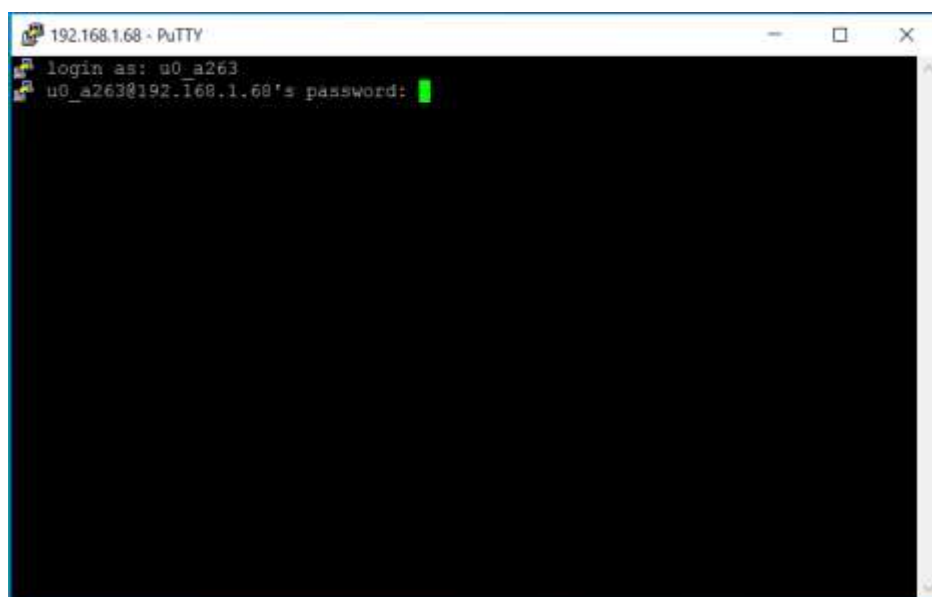
Posteriormente en la parte inferior presionamos para abrir una conexión al servidor dando el botón “Open”.

En la PC al conectarse por primera vez nos **pedirá por única vez** que confirmen la llave de cifrado de información le damos confirmación en el botón de “Si”.

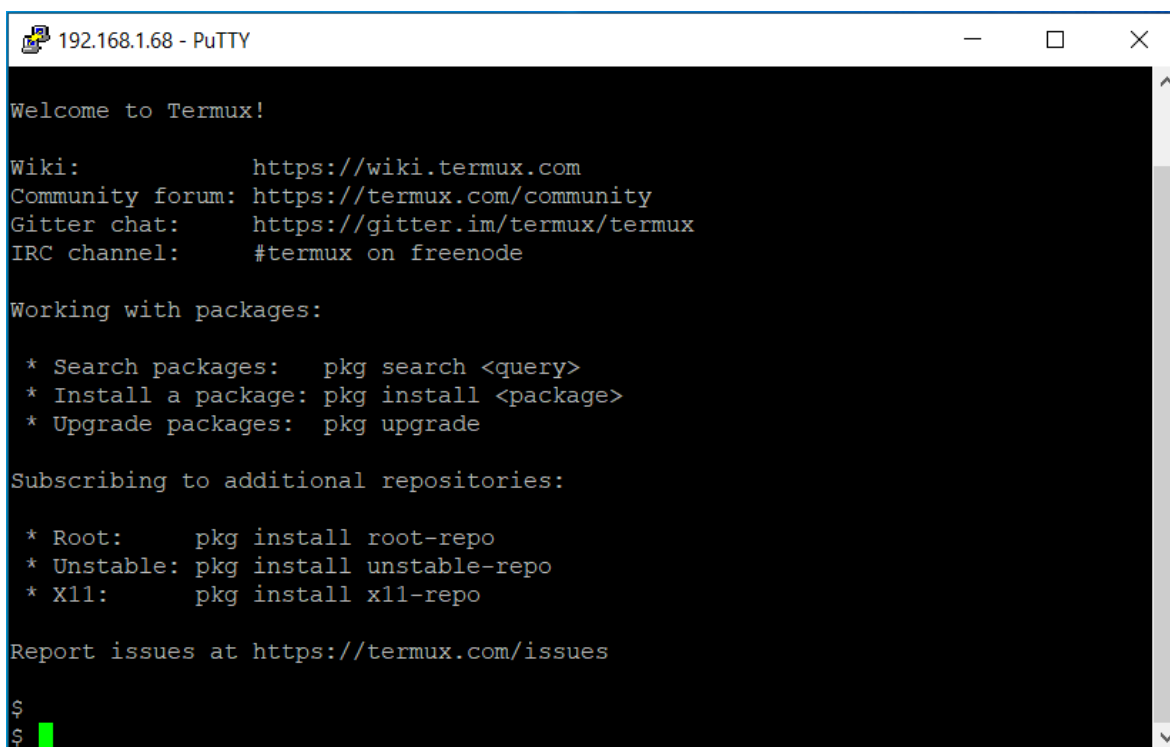


Posteriormente nos pedirá el usuario con el que nos vamos a conectar. Utilizaremos la información que sacamos con anterioridad (usuario y password).

En el **Login as:** debemos introducir nuestro usuario y dar Enter, después nos pedirá el password le damos nuevamente el botón de Enter.



Si los datos fueron los correctos, estaremos en una sesión de SSH (Secure Shell) realizada desde la PC (Cliente) en el teléfono (Servidor SSH).



```
192.168.1.68 - PuTTY
Welcome to Termux!

Wiki:           https://wiki.termux.com
Community forum: https://termux.com/community
Gitter chat:    https://gitter.im/termux/termux
IRC channel:    #termux on freenode

Working with packages:

* Search packages:  pkg search <query>
* Install a package: pkg install <package>
* Upgrade packages: pkg upgrade

Subscribing to additional repositories:

* Root:    pkg install root-repo
* Unstable: pkg install unstable-repo
* X11:     pkg install x11-repo

Report issues at https://termux.com/issues

$
$
```

NOTA IMPORTANTE: Recordemos que la IP (dirección) de la PC y la IP (dirección) teléfono móvil conectados en la misma WiFi estarán cambiando probablemente cada vez que nos desconectemos y volvamos a conectar por lo que hay que volver a verificar que direcciones tienen cada dispositivo, esto nos asegura el éxito de la conexión entre dispositivos por medio del servidor de SSH del teléfono y PC (Cliente).

8. Ambientes Blockly (App Inventor, AppyBuilder y Thunkable).

App Inventor es un entorno de desarrollo de software creado por Google Labs para la elaboración de aplicaciones destinadas al sistema operativo Android. El usuario puede, de forma visual y a partir de un conjunto de herramientas básicas, ir enlazando una serie de bloques para crear la aplicación. El sistema es gratuito y se puede descargar fácilmente de la web. Las aplicaciones creadas con App Inventor son muy fáciles de crear debido a que no se necesita conocimiento de ningún tipo de lenguaje de programación.

Todos los ambientes actuales que usan la tecnología de Blockly como son AppyBuilder y Thunkable entre otros tienen su versión gratuita, su forma de uso puede ser a través de internet en sus diversos sitios o también puede ser instalado en casa.

Los bloques que forman la arquitectura de Mini BloclChain han sido probados en App inventor y AppyBuilder sin embargo por su optimización de código deberán funcionar en las otras plataformas.

Versiones de por internet:

App Inventor.

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

AppyBuilder.

<http://appybuilder.com/>

Thunkable.

<https://thunkable.com/>

Versión para ser instalada en tu equipo (PC):

<https://sites.google.com/site/aprendeappinventor/instala-app-inventor>

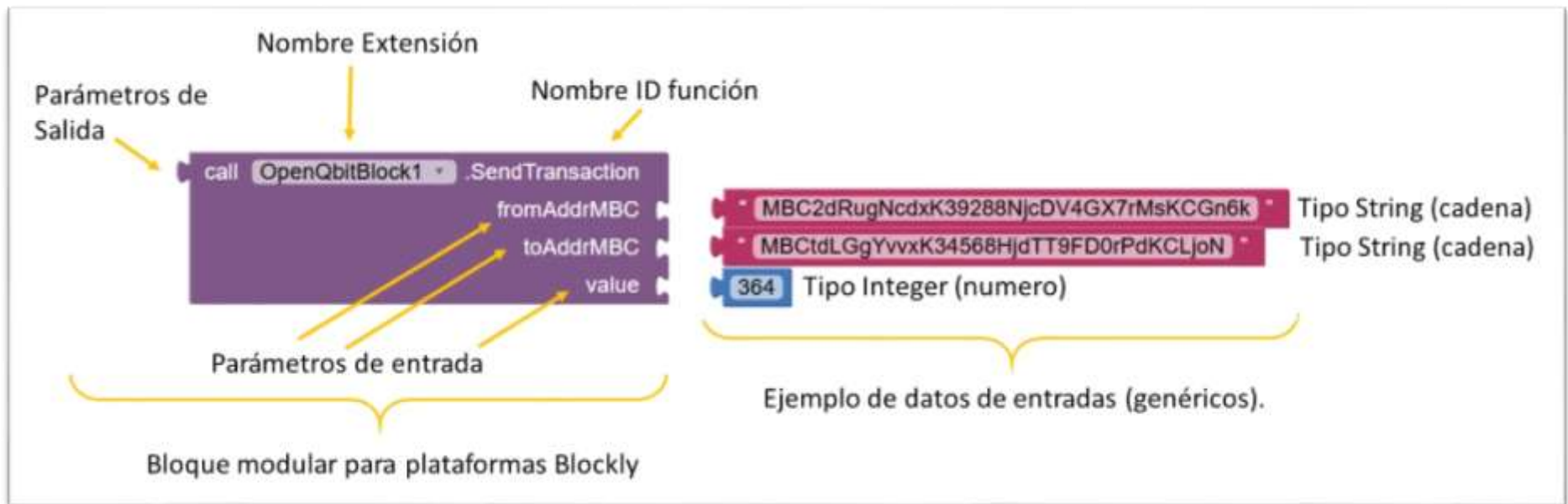
Ambiente para desarrolladores de bloques Blockly.

<https://editor.appybuilder.com/login.php>

9. Definición y uso de bloques en Mini QRNG.

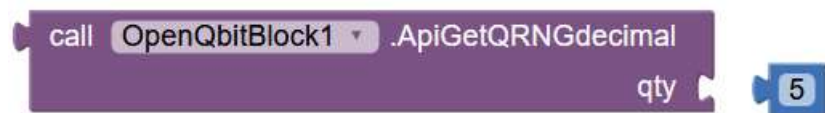
Empezaremos con explicar la distribución de los datos que tendrá todos los bloques, su sintaxis de uso y configuración.

En el siguiente ejemplo podemos ver un bloque modular y sus parámetros de entrada y salida, así como los tipos de datos de entrada, estos datos pueden ser de tipo String (cadena de caracteres) o Integer (numero entero o decimal). Mostramos como es su uso y configurarlo para su funcionamiento adecuado.



Cada bloque modular tendrá su descripción y se nombrará en caso de tener alguna(s) dependencia(s) obligatoria(s) u opcional(es) de otros bloques usados como parámetros de entrada se anunciará el proceso de integración. Comencemos con los bloques de la extensión OpenQbitQRNGwithSSH.

Bloque para generar números cuánticos aleatorios decimal – (ApiGetQRNGdecimal)



Parámetros de entrada: **qty** <Integer>

Parámetros de salida: Da la cantidad “qty” de números aleatorios cuánticos decimales introducidos en la entrada los números están dentro del rango de 0 y 1 en formato JSON.

Ejemplo:

qty = 5; output: {"result": [0.5843012986202495, 0.7746497687824652, 0.05951126805960929, 0.1986079055812694, 0.03689783439899279]}

Descripción: API de generador de números cuánticos aleatorios decimal (QRNG – Quantum Random Number Generator).

Bloque para generar números cuánticos aleatorios decimal – (ApiGetQRNGdecimal)



Parámetros de entrada: **qty** <Integer>, **min** <Integer>, **max** <max>

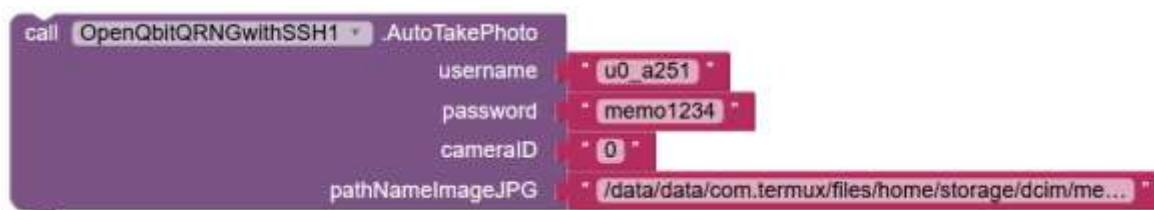
Parámetros de salida: Da la cantidad “qty” de números aleatorios cuánticos enteros introducidos en la entrada los números están dentro del rango de min y max en formato JSON.

Ejemplo:

qty = 8, min = 1, max = 100; output: {"result": [3, 53, 11, 2, 66, 44, 9, 78]}

Descripción: API de generador de números cuánticos aleatorios enteros (QRNG – Quantum Random Number Generator).

Bloque para tomar una foto de forma automática - (AutoTakePhone)



Parámetros de entrada: **username** <String>, **password** <String>, **cameraID** <String>, **pathNameImageJPG** <String>.

Dependencia obligatoria: Para usar este bloque debe cumplir dos dependencias de software; instalar en la terminal Termux el modulo de Termux-API este modulo contiene el proceso para tomar fotos de forma automática y subir el servidor SSH que anteriormente ya se instalo.

Parámetros de salida: Nos entrega una foto (imagen) con formato JPG en la ruta (path) especificada. En la ruta debe

Descripción: Crea una foto con formato JPG de forma automática sin intervención del usuario.

Para la instalacion de Termux-API se debe ejecutar el siguiente comando en la Terminal Termux:

`$ pkg install termux-api`

```

$ pkg install termux-api
Ign:2 https://dl.bintray.com/grimler/game-packag
es-24 games InRelease
Ign:3 https://dl.bintray.com/grimler/science-pac
kages-24 science InRelease
Ign:1 https://dl.bintray.com/termux/termux-packa
ges-24 stable InRelease
Get:5 https://dl.bintray.com/grimler/game-packag
es-24 games Release [5344 B]
Get:6 https://dl.bintray.com/grimler/science-pac
kages-24 science Release [6191 B]
Get:4 https://dl.bintray.com/termux/termux-packa
ges-24 stable Release [8255 B]
Get:7 https://dl.bintray.com/grimler/game-packag
es-24 games Release.gpg [475 B]
Get:8 https://dl.bintray.com/grimler/science-pac
kages-24 science Release.gpg [475 B]
Get:9 https://dl.bintray.com/termux/termux-packa
ges-24 stable Release.gpg [821 B]
0% [8 Release.gpg gpgv 6191 B]

```

```

Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages will be upgraded:
  termux-api
1 upgraded, 0 newly installed, 0 to remove and 7
4 not upgraded.
Need to get 21.2 kB of archives.
After this operation, 4096 B of additional disk
space will be used.
Get:1 https://dl.bintray.com/termux/termux-packa
ges-24 stable/main arm termux-api arm 0.50-1 [21
.2 kB]
Fetched 21.2 kB in 1s (18.7 kB/s)
(Reading database ... 25317 files and directorie
s currently installed.)
Preparing to unpack .../termux-api_0.50-1_arm.de
b ...
Unpacking termux-api (0.50-1) over (0.50) ...
Setting up termux-api (0.50-1) ...
$

```

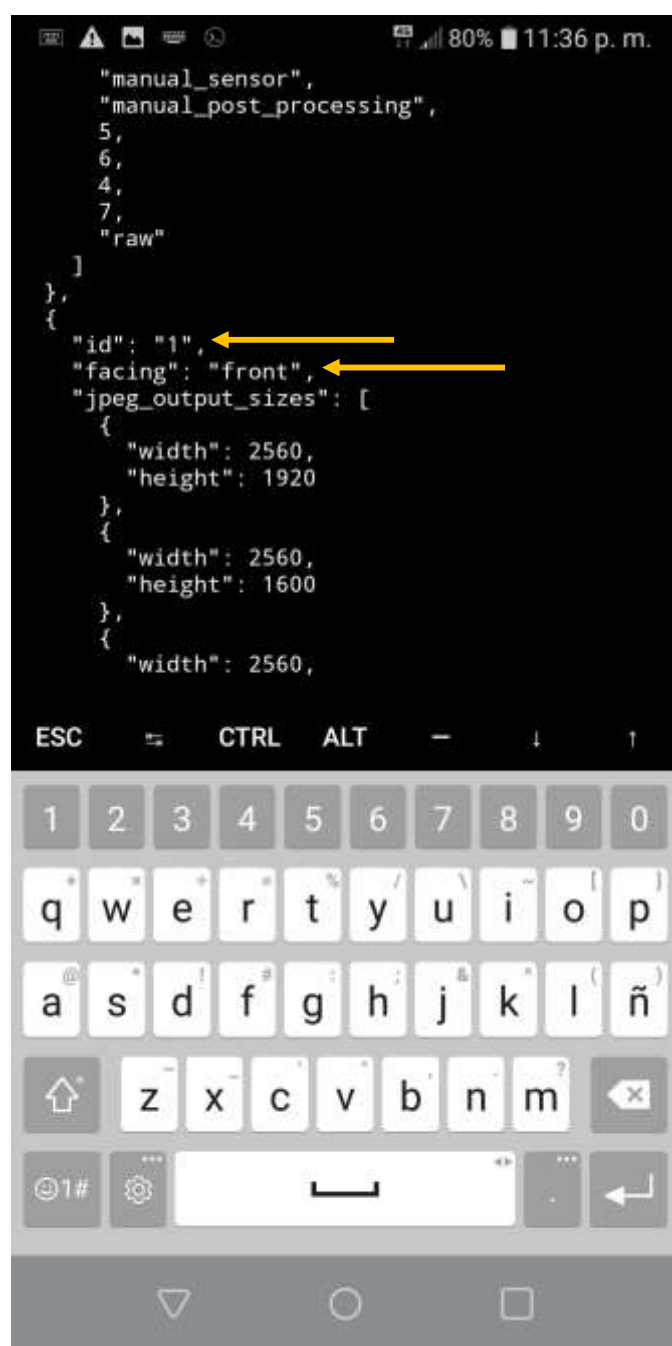
Para saber número, cuantos y posición de IDs (Identificadores de lentes fotográficos) que tiene tu dispositivo móvil (Smartphone) ejecutar el siguiente comando en la terminal Termux.

\$ termux-camera-info



```
$ termux-camera-info
[
  {
    "id": "0",
    "facing": "back",
    "jpeg_output_sizes": [
      {
        "width": 4160,
        "height": 3120
      },
      {
        "width": 4160,
        "height": 2340
      },
      {
        "width": 4160,
        "height": 2080
      },
      {
        "width": 3264,
        "height": 2448
      },
      {

```



```
    "manual_sensor",
    "manual_post_processing",
    5,
    6,
    4,
    7,
    "raw"
  ],
  {
    "id": "1",
    "facing": "front",
    "jpeg_output_sizes": [
      {
        "width": 2560,
        "height": 1920
      },
      {
        "width": 2560,
        "height": 1600
      },
      {
        "width": 2560,
```

En nuestro ejemplo el smarphone LG Q6 que usamos tiene dos IDs “0” en la parte trasera (back) y “1” en la parte fronta (front).

Ahora probemos que funciona la API para tomar una foto usando el ID “0” del lente trasero y le damos un nombre arbitrario en nuestro caso test.jpg

Recordemos que el API solo entrega fotos en formato JPG ejecutamos el siguiente comando:

```
$ termux-camera-photo -c 0 test.jpg
```

El anterior comando nos deberá haber creado un archivo con el nombre test.jpg de forma automática, si es así ya podemos usar el bloque (AutoTakePhoto), no olvidemos iniciar nuestro servidor local de SSH con el comando: **\$ sshd**

NOTA: En la variable pathNameImageJPG se debe considerar que la ruta dentro de la terminal de Termux para acceder al storage del Smartphone debe usar:

```
/data/data/com.termux/files/home/storage/dcim/example.jpg
```

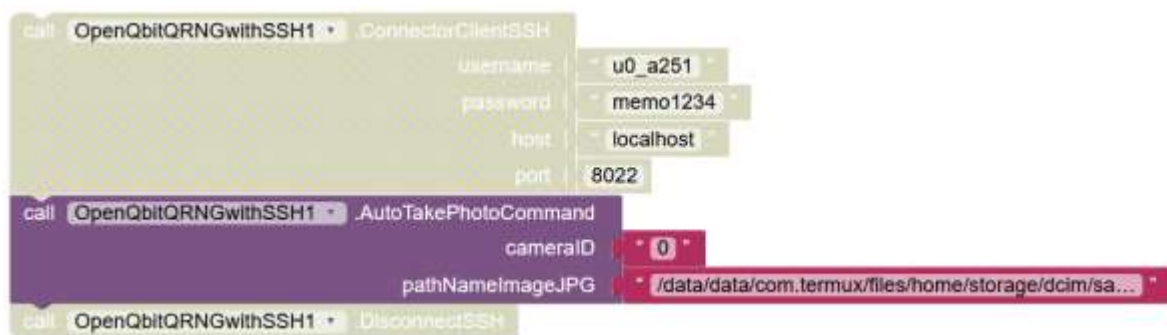
La anterior ruta en android sería igual a:

```
/mnt/sdcard/dcim/example.jpg
```

Sin embargo, debemos recordar que en la terminal de Termux la ruta válida para poder ver el almacenamiento del teléfono móvil siempre se debe considerar la ruta por default:

```
/data/data/com.termux/files/home/storage
```

Bloque para tomar una foto de forma automática SOLO COMANDO - (AutoTakePhoneCommand)



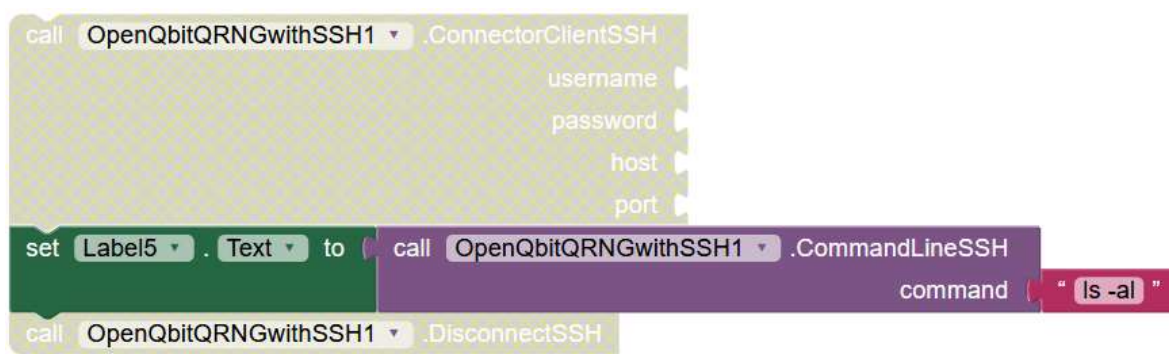
Parámetros de entrada: **cameraID** <String>, **pathNameImageJPG** <String>.

Dependencia obligatoria: Bloque (ConnectorClientSSH), Bloque (DisconnectSSH).

Parámetros de salida: Nos entrega una foto (imagen) con formato JPG en la ruta (path) especificada.

Descripción: Crea una foto con formato JPG de forma automática sin intervención del usuario. Sin embargo, la diferencia en comparación con el bloque (AutoTakePhoto), este bloque solo contiene el comando para crear la foto y se necesita antes el bloque para conectarse al servidor SSH (ConnectorClientSSH) y después usar el bloque (DisconnectSSH).

Bloque para ejecutar comando en la terminal de Termux – (**CommandLineSSH**)



Parámetros de entrada: **command** <String>

Dependencia obligatoria: Bloque (**ConnectorClientSSH**), Bloque (**DisconnectSSH**).

Parámetros de salida: Ejecuta el comando introducido en a terminal Termux.

Descripción: Se ejecuta un comando introducido y se necesita antes el bloque para conectarse al servidor SSH (**ConnectClientSSH**) y después usar el bloque (**DisconnectSSH**).

Bloque para conectarse a un servidor SSH remoto o local – (**ConnectorClientSSH**).

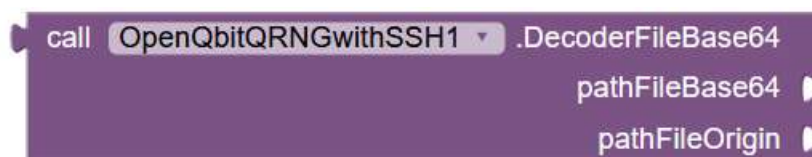


Parámetros de entrada: **username** <string>, **password** <string>, **host** <string>, **port**<integer>

Parámetros de salida: Si es exitosa la conexión con el servidor ssh de la terminal Termux nos entrega un mensaje; **"Connect SSH"**, en caso de no ser exitosa nos entrega un mensaje **NULL**.

Descripción: Bloque de comunicación para conectar a servidor SSH elegido a la terminal Termux, vía protocolo de comunicación SSH (Secure Shell).

Bloque para decodificar un archivo con algoritmo Base64 (**DecoderFileBase64**).

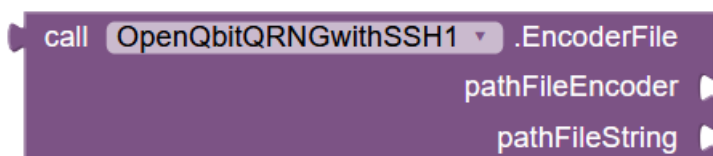


Parámetros de entrada: **pathFileBase64** <String>, **pathFileOrigin** <String>

Parámetros de salida: Archivo origen que se introdujo en el bloque (**EncoderFileBase64**).

Descripción: Se convierte un archivo en formato Base64 al archivo original que fue introducido en el bloque (**EncoderFileBase64**).

Bloque nos convierte un archivo en formato Base64 – (**EncoderFile**)

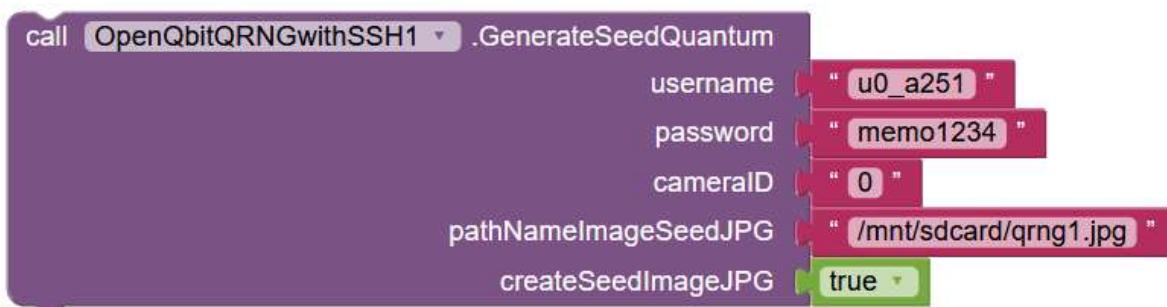


Parámetros de entrada: **pathFileOrigin** <String>, **pathFileBase64** <String>

Parámetros de salida: Archivo codificado en Base64.

Descripción: Se convierte un archivo origen de cualquier formato a un archivo en formato Base64. Los nombres de los archivos pueden ser arbitrarios y escogidos por el usuario.

Bloque para generar QRNG (Quantum Random Number Generator) – (**GenerateSeedQuantum**)



Parámetros de entrada: **username** <String>, **password** <String>, **cameraID** <String>, **pathNameImageJPG** <String>. **createSeedImageJPG** <boolean>

Si el valor booleano es “True” al ejecutarse el bloque cada vez se creará una nueva Imagen JPG semilla con el nombre de la ruta introducida. Si el valor booleano es “Falso” desactivamos la opción de sacar una Imagen (foto) JPG y podemos indicar de forma manual en donde esta una imagen a nuestra selección pudiendo ser de cualquier formato.

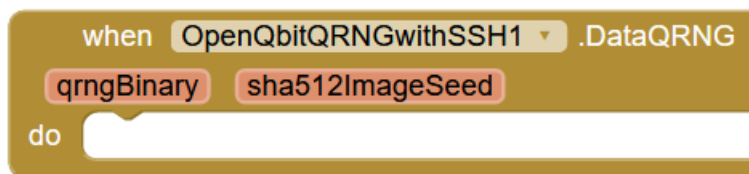
NOTA: Los mejores resultados para genrar un QRNG es basado en una image con formato “RAW”. Ejemplo formato DNG.

Dependencia obligatoria: Se necesita tener instalado el Termux-API ya antes mencionada su instalación en el bloque (**AutoTakePhoto**).

Parámetros de salida: Se ejecuta el evento (**DataQRNG**) y nos entrega dos valores:

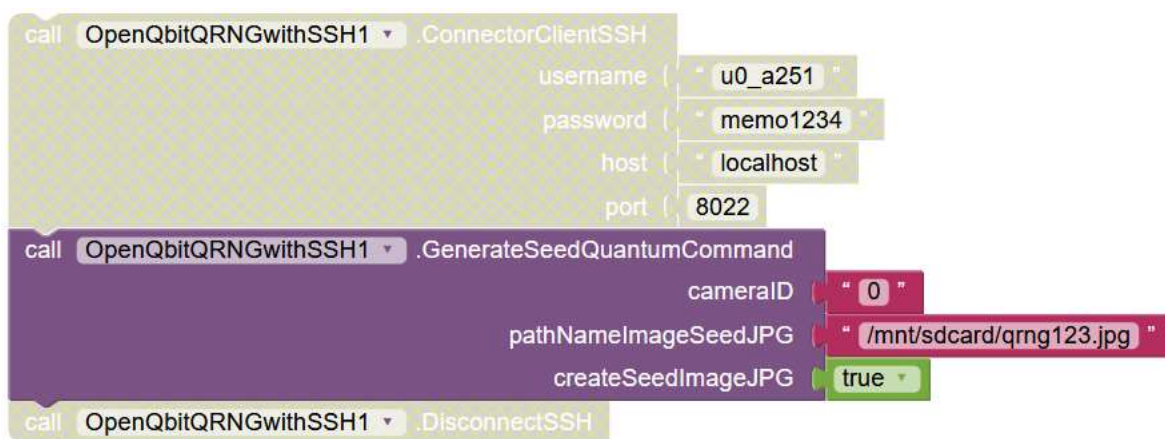
qrngBinary.- Una cadena de números binarios aleatorios.

Sha512ImageSeed.- Sha512 de la imagen JPG semilla de donde salieron los números aleatorios.



Descripción: Genera nuevos aleatorios cuánticos (QRNG) mediante el sensor óptico de la cámara del teléfono móvil el algoritmo esta basado en la captura de foto aleatoria y a esta se le aplica el algoritmo para entregar una cadena de números binarios.

Bloque para generar QRNG (Quantum Random Number Generator) – (GenerateSeedQuantumCommand).



Parámetros de entrada: **cameraID** <String>, **pathNameImageJPG** <String>. **createSeedImageJPG** <booleano>

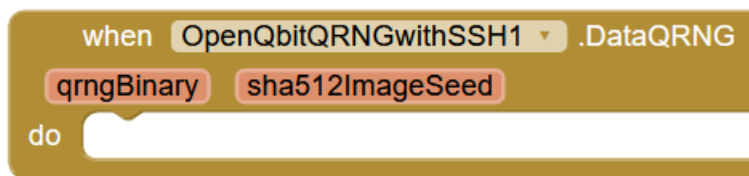
Si el valor booleano es "True" al ejecutarse el bloque cada vez se creará una nueva Imagen JPG semilla con el nombre de la ruta introducida. Si el valor booleano es "Falso" desactivamos la opción de sacar una Imagen (foto) JPG y podemos indicar de forma manual en donde esta una imagen a nuestra selección pudiendo ser de cualquier formato.

Dependencia obligatoria: Bloque (**ConnectorClientSSH**), Bloque (**DisconnectSSH**).

Parámetros de salida: Se ejecuta el evento (**DataQRNG**) y nos entrega dos valores:

qrngBinary.- Una cadena de números binarios aleatorios.

Sha512ImageSeed.- Sha512 de la imagen JPG semilla de donde salieron los números aleatorios.



Descripción: Genera nueros aleatorios cuánticos (QRNG) mediante el sensor óptico de la cámara del teléfono móvil el algoritmo esta basado en la captura de foto aleatoria y a esta se le aplica el algoritmo para entregar una cadena de números binarios.

Sin embargo, la diferencia en comparación con el bloque (**GenerateSeedQuantum**), este bloque solo contiene el comando para ejecutar el algoritmo de reacion de QRNG y se necesita antes el bloque para conectarse al servidor SSH (**ConnectClientSSH**) y después usar el bloque (**DisconnectSSH**).

Bloque para obtener la entropía de Shannon en una Image (foto) – (**GetShannonEntropyFile**).



Parámetros de entrada: **username** <String>, **password** <String>, **pathFileImage** <String>

Dependencia obligatoria: Se necesita instalar en la terminal de Termux el modulo de Shannon_entropy.

Parámetros de salida: Entrega la entropía de una imagen

Ejemplo:

Output: 8.94596789873

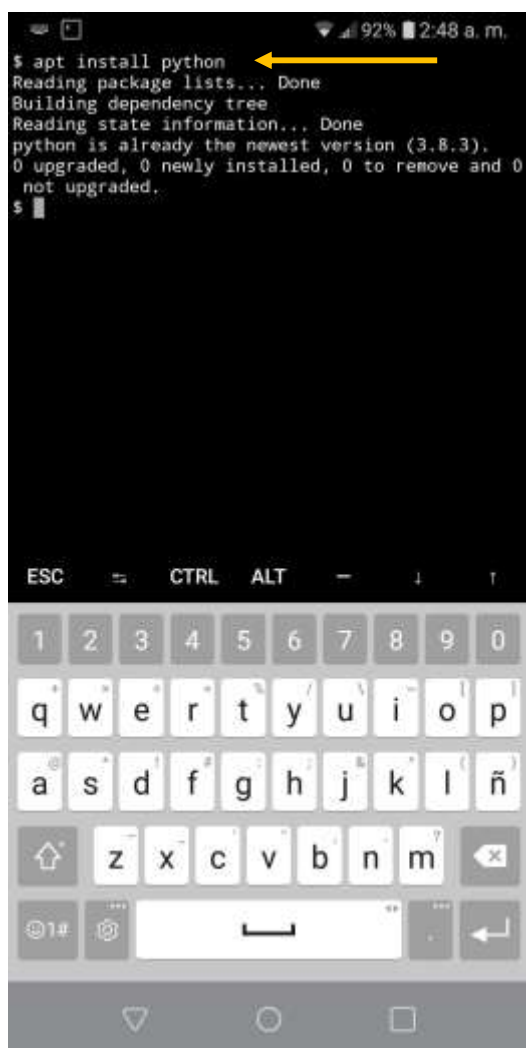
Descripción: Nos entrega la entropía de una imagen. La entropía es el parámetro fundamental para la generación de números aleatorios de buena calidad, mientras mayor sea la entropía dara mejores resultados.

Para la instalación del modulo de entropía de Shannon, primero necesitamos instalar el paquete de Python y posteriormente instalar el modulo de Pillow y Shannon_entropy con los siguientes comandos en la terminal de Termux.

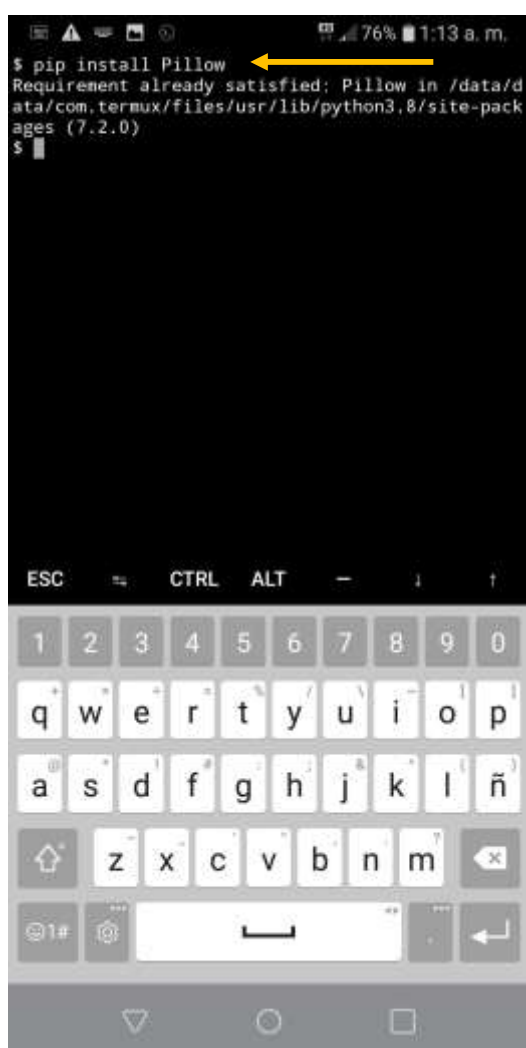
\$ apt install Python

\$ pip install Pillow

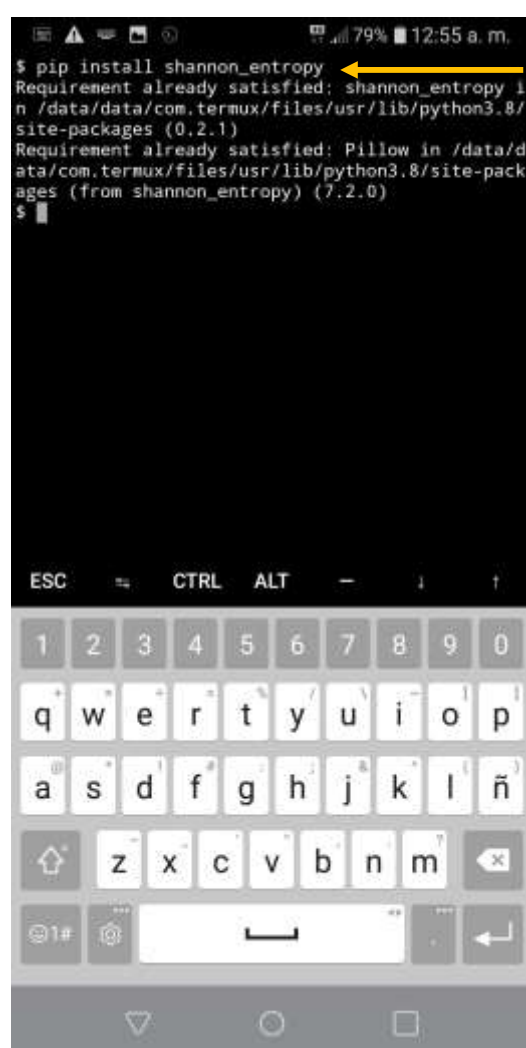
\$ pip install Shannon_entropy



```
$ apt install python
Reading package lists... Done
Building dependency tree
Reading state information... Done
python is already the newest version (3.8.3).
0 upgraded, 0 newly installed, 0 to remove and 0
not upgraded.
$
```



```
$ pip install Pillow
Requirement already satisfied: Pillow in /data/d
ata/com.termux/files/usr/lib/python3.8/site-pack
ages (7.2.0)
$
```



```
$ pip install shannon_entropy
Requirement already satisfied: shannon_entropy i
n /data/data/com.termux/files/usr/lib/python3.8/
site-packages (0.2.1)
Requirement already satisfied: Pillow in /data/d
ata/com.termux/files/usr/lib/python3.8/site-pack
ages (from shannon_entropy) (7.2.0)
$
```

Posteriormente tenemos que crear en el directorio “Home” de Termux el siguiente archivo Python llamado “entropy.py” con el siguiente código dentro.

```
From PIL import Image

Import math

From Shannon_entropy import *

Import sys

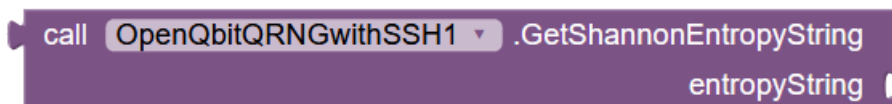
Img=Image.open(sys.argv[1])

print(Shannon_entropy(img))
```

Guardamos el archivo y ya tenemos nuestro ambiente para usarlo con el bloque (GetShannonEntropyFile).

Tip: De hecho, con esta instalación de Python puedes crear tus propios programas en este lenguaje y ejecutarlos a través del bloque (ConnectorClientSSH).

Bloque para obtener entropía Shannon de una cadena de caracteres – (GetShannonEntropyString).



Parámetros de entrada: **entropyString** <String>

Parámetros de salida: Entrega la entropía de una cadena de caracteres.

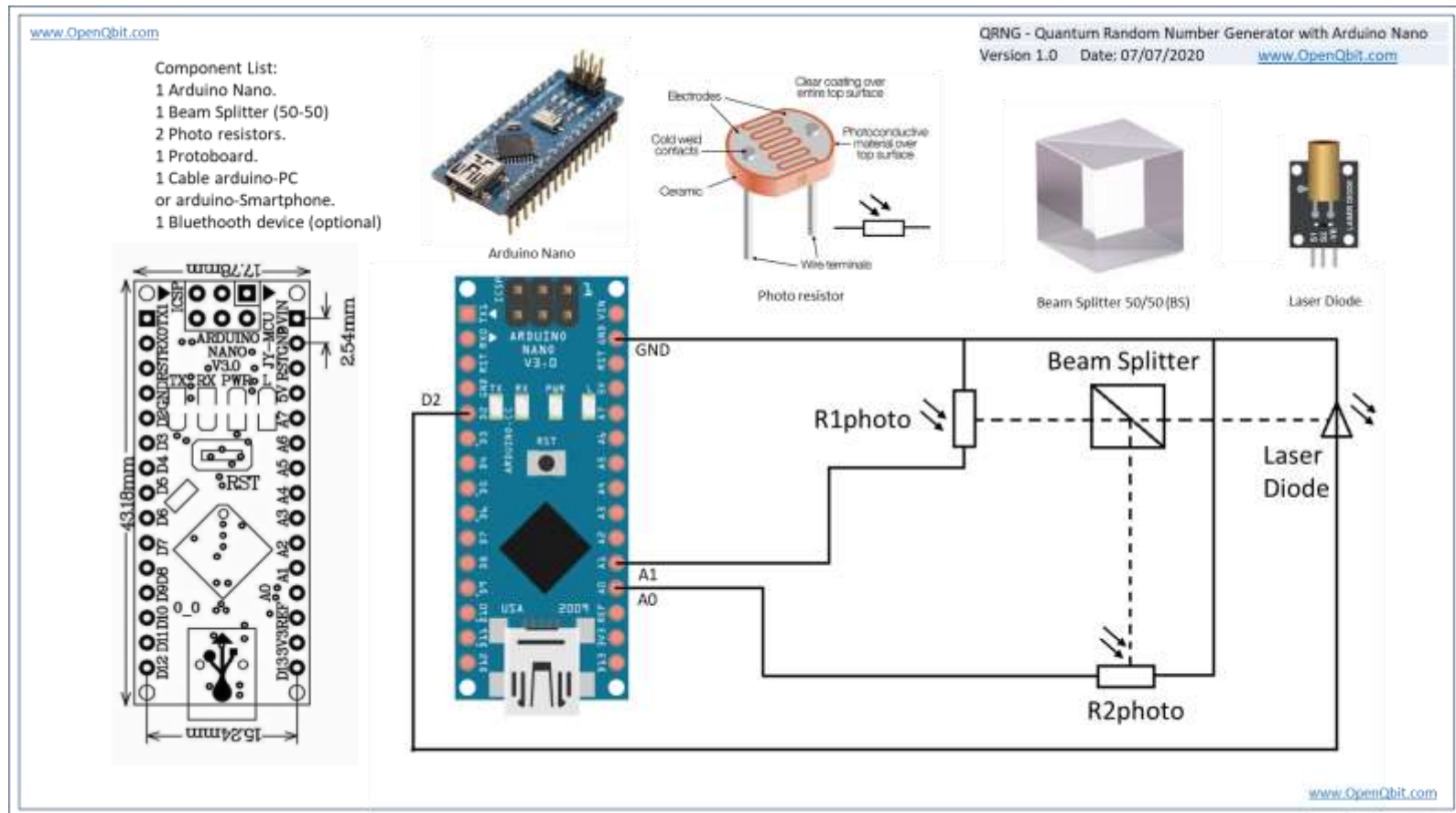
Ejemplo:

Output: 5.76002345671

Descripción: Nos entrega la entropía de una cadena de caracteres. La entropía es el parámetro fundamental para la generación de números aleatorios de buena calidad, mientras mayor sea la entropía dara mejores resultados.

10. Creacion de dispositivo “Hardware” de un QRNG.

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



www.OpenQbit.com

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());
}

```

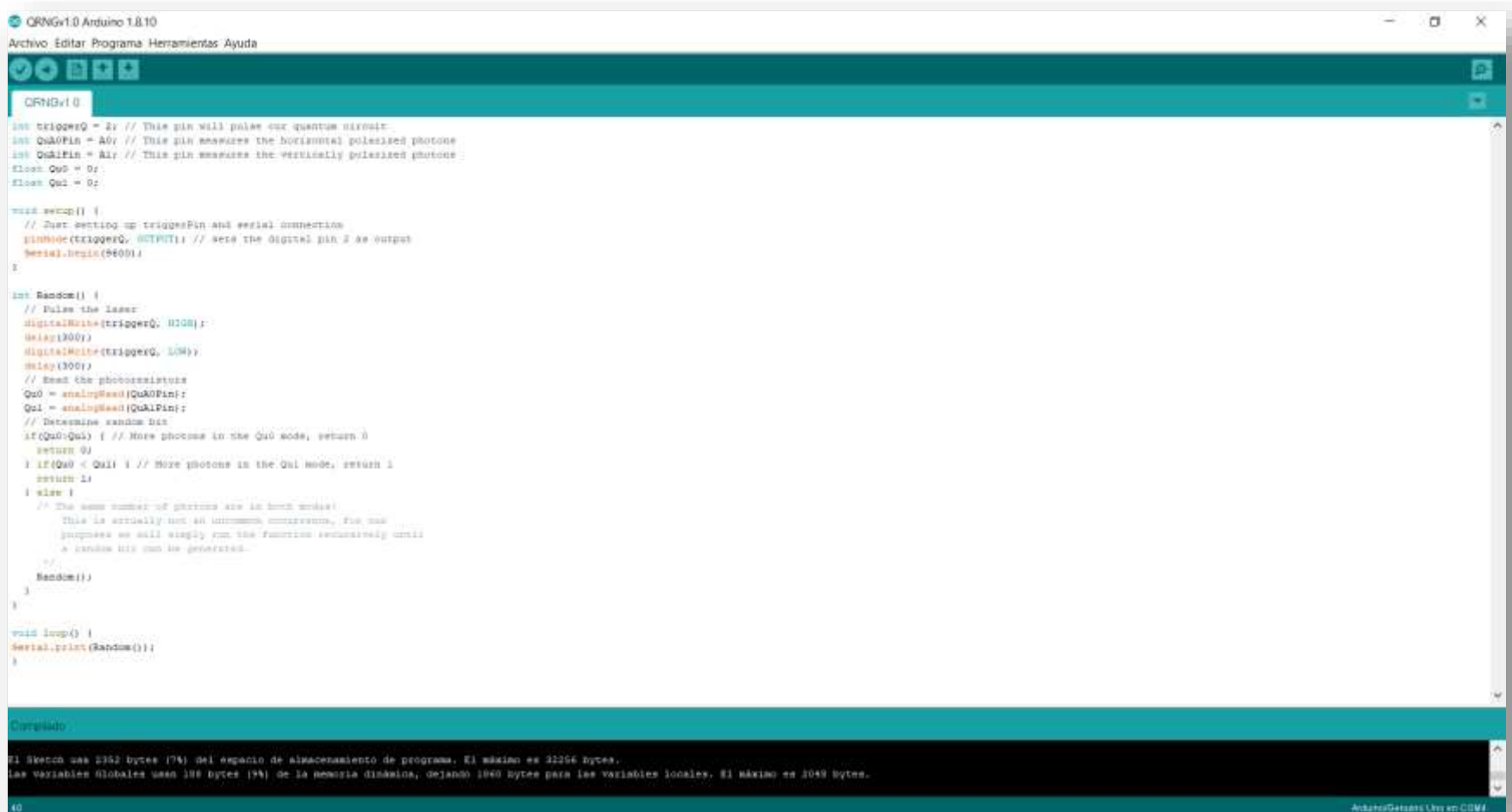
Output console

0010110101011110101011010.....

QRNG - Quantum Random Number Generator with Arduino Nano
Version 1.0 Date: 07/07/2020 www.OpenQbit.com

www.OpenQbit.com

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



```
QRNGv10 Arduino 1.8.10
Archivo Editar Programa Herramientas Ayuda

QRNGv10

int triggerQ = 2; // This pin will pulse our quantum circuit.
int Qa0Pin = A0; // This pin measures the horizontal polarised photons
int Qa1Pin = A1; // This pin measures the vertically polarised photons
float Qa0 = 0;
float Qa1 = 0;

void setup() {
  // Just setting up triggerPin and serial connection
  pinMode(triggerQ, OUTPUT); // sets the digital pin 2 as output
  Serial.begin(5000);
}

int Random() {
  // Pulse the laser
  digitalWrite(triggerQ, HIGH);
  delay(300);
  digitalWrite(triggerQ, LOW);
  delay(300);
  // Read the photoreistors
  Qa0 = analogRead(Qa0Pin);
  Qa1 = analogRead(Qa1Pin);
  // Determine random bit
  if(Qa0 > Qa1) { // More photons in the Qa0 mode, return 0
    return 0;
  } else if(Qa0 < Qa1) { // More photons in the Qa1 mode, return 1
    return 1;
  } else {
    // The same number of photons are in both modes!
    // This is actually not an uncommon occurrence, but our
    // purposes we will simply run the function repeatedly until
    // a random bit can be generated.
    return Random();
  }
}

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

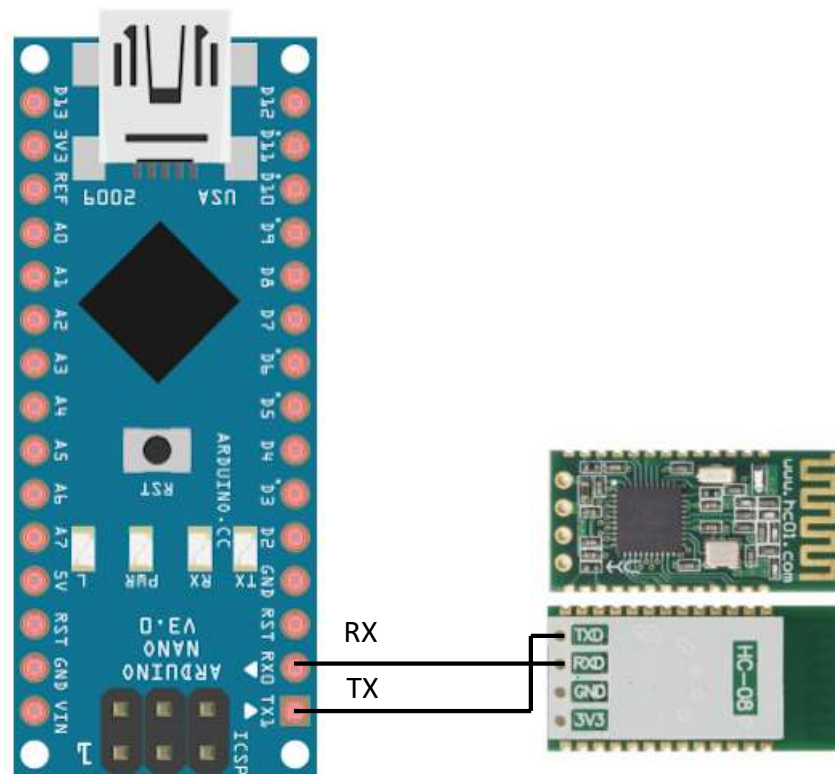
Compilado

El Sketch usa 2352 bytes (7%) del espacio de almacenamiento de programa. El máximo es 32256 bytes.
Las variables Globales usan 198 bytes (1%) de la memoria dinámica, dejando 1840 bytes para las variables locales. El máximo es 2048 bytes.

40 ArduinoGetters Ltd en CDM

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:



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 tarves de bluetooth similar en línea.

http://kio4.com/appinventor/9A0_bluetooth_RXTX.htm

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

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

11. Anexo “Computación Cuántica con OpenQbit”.

¿Cómo funciona la computación cuántica? ⁽²⁾

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

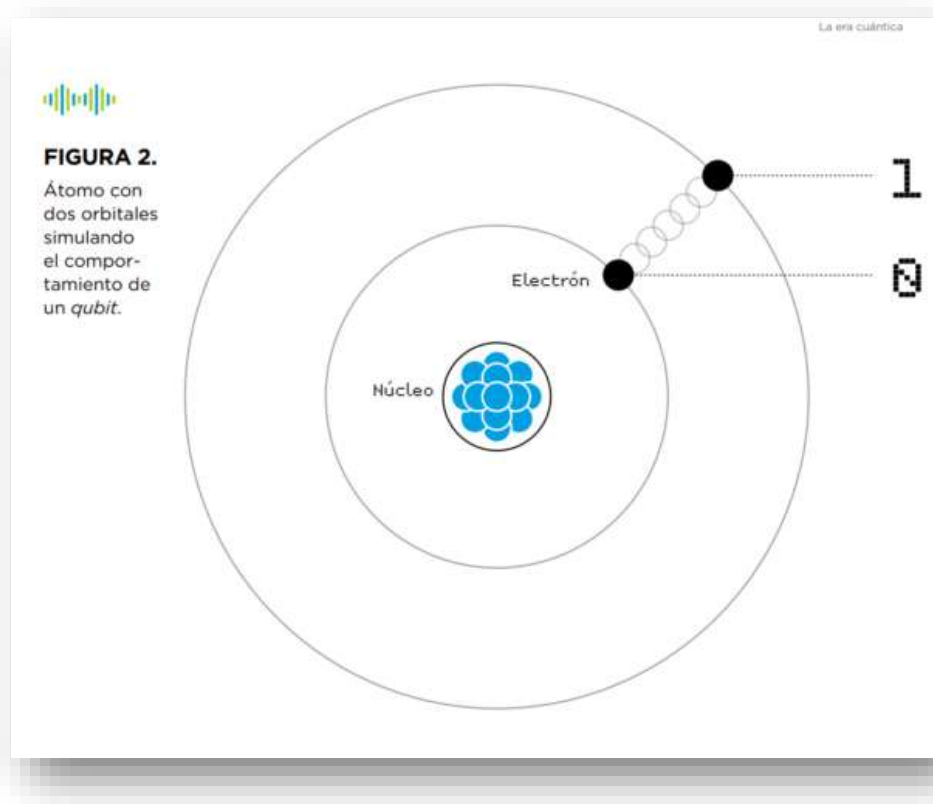
Para entender cómo funcionan los computadores cuánticos es conveniente explicar primero cómo funcionan los computadores que utilizamos a diario, a los que nos referiremos en este documento como computadores digitales o clásicos. Estos, al igual que el resto de los dispositivos electrónicos como tabletas 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.

Mini QRNG es creado por OpenQbit.com compañía que se enfoca en desarrollar tecnología basada en computación cuántica para diferentes tipos de sectores tanto privado como público.

Por qué Mini QRNG es diferente a los demás QRNG, simplemente porque el sistema fue creado para ser modular y se puede armar fácilmente en casa con un costo bastante económico.

(1) <https://blogs.iadb.org/conocimiento-abierto/es/como-funciona-la-computacion-cuantica/>

12. Licenciamiento y uso de software.

Android

<https://source.android.com/setup/start/licenses>

Termux

<https://github.com/termux/termux-app/blob/master/LICENSE.md>

Node

<https://raw.githubusercontent.com/nodejs/node/master/LICENSE>

Python

<https://www.python.org/download/releases/2.7/license/>

OpenSSH

<https://www.openssh.com/features.html>

Putty SSH

<https://www.chiark.greenend.org.uk/~sgtatham/putty/licence.html>

MIT App Inventor 2 Companion y App Inventor Blockly

<https://appinventor.mit.edu/about/termsofservice>

Extensiones externas:

JSOINTOOLS

<https://thunkableblocks.blogspot.com/2017/07/jsontools-extension.html>

Licenciamiento versiones opensource y comercial de sistema Mini QRNG consultar la página oficial <http://www.openqbit.com>

Mini QRNG, Mini BlocklyChain, MiniBlockly, BlocklyCode, MiniBlockMiniChain, QBlockly son marcas registradas por OpenQbit.

Mini QRNG es dominio público.

Todo el código y la documentación en Mini QRNG ha sido dedicado al dominio público por los autores. Todos los autores de códigos y representantes de las empresas para las que trabajan han firmado declaraciones juradas dedicando sus contribuciones al dominio público y los originales de esas declaraciones juradas se almacenan en una caja fuerte en las oficinas principales de OpenQbit México. Cualquier persona es libre de publicar, usar o distribuir las extensiones de Mini QRNG (OpenQbit) original, ya sea en forma de código fuente o como binario compilado, para cualquier propósito, comercial o no comercial, y por cualquier medio.

El párrafo anterior se aplica al código y la documentación entregables en Mini QRNG aquellas partes de la biblioteca de Mini QRNG que realmente agrupa y envía con una aplicación más grande. Algunos scripts utilizados como parte del proceso de compilación (por ejemplo, los

scripts de "configuración" generados por autoconf) pueden estar incluidos en otras licencias de código abierto. Sin embargo, nada de estos scripts de compilación llega a la biblioteca entregable final de Mini QRNG, por lo que las licencias asociadas con esos scripts no deberían ser un factor en la evaluación de sus derechos para copiar y usar la biblioteca Mini QRNG.

Todo el código entregable en Mini QRNG ha sido escrito desde cero. No se ha tomado ningún código de otros proyectos o de Internet abierto. Cada línea de código puede rastrearse hasta su autor original, y todos esos autores tienen dedicaciones de dominio público en el archivo. Por lo tanto, la base de código Mini QRNG es limpia y no está contaminada con código con licencia de otros proyectos de código abierto, no contribución abierta

Mini QRNG es de código abierto, lo que significa que puede hacer tantas copias como desee y hacer lo que quiera con esas copias, sin limitación. Pero Mini QRNG no es de contribución abierta. Para mantener Mini QRNG en el dominio público y garantizar que el código no se contamine con contenido patentado o con licencia, el proyecto no acepta parches de personas desconocidas. Todo el código en Mini QRNG es original, ya que ha sido escrito específicamente para su uso por Mini QRNG. No se ha copiado ningún código de fuentes desconocidas en Internet.

Mini QRNG es de dominio público y no requiere una licencia. Aun así, algunas organizaciones quieren pruebas legales de su derecho a usar Mini QRNG. Las circunstancias donde esto ocurre incluyen lo siguiente:

- Su empresa desea indemnización por reclamos de infracción de derechos de autor.
- Está utilizando Mini QRNG en una jurisdicción que no reconoce el dominio público.
- Está utilizando de Mini QRNG en una jurisdicción que no reconoce el derecho de un autor a dedicar su trabajo al dominio público.
- Desea tener un documento legal tangible como evidencia de que tiene el derecho legal de usar y distribuir de Mini QRNG.
- Su departamento legal le dice que debe comprar una licencia.

Si alguna de las circunstancias anteriores se aplica a usted, OpenQbit, la compañía que emplea a todos los desarrolladores de Mini QRNG, le venderá una Garantía de título para Mini QRNG. Una garantía de título es un documento legal que afirma que los autores reclamados de Mini QRNG son los verdaderos autores, y que los autores tienen el derecho legal de dedicar el Mini QRNG al dominio público, y que OpenQbit se defenderá enérgicamente contra los reclamos de licenciamiento. Todos los ingresos de la venta de las garantías de título de Mini QRNG se utilizan para financiar la mejora continua y el soporte de Mini QRNG.

Código contribuido

Para mantener Mini QRNG completamente libre y libre de derechos de autor, el proyecto no acepta parches. Si desea hacer un cambio sugerido e incluir un parche como prueba de concepto, sería genial. Sin embargo, no se ofenda si reescribimos su parche desde cero. El tipo de licenciamiento no-comercial u opensource quien lo use en esta modalidad y alguna similar sin compra de soporte ya sea uso individual o corporativo no importando el tamaño de empresa se rigira por las siguientes premisas legales.

Renuncia de garantía. A menos que lo exija la ley aplicable o acordado por escrito, el Licenciante proporciona el Trabajo (y cada El Colaborador proporciona sus Contribuciones) "TAL CUAL", **SIN GARANTÍAS O CONDICIONES DE NINGÚN TIPO**, ya sea expreso o implícito, incluidas, entre otras, cualquier garantía o condición de TÍTULO, NO INFRACCIÓN, COMERCIALIZACIÓN O APTITUD PARA UN PROPÓSITO PARTICULAR. Usted es el único responsable de determinar el correcto uso o redistribuir el Trabajo y asumir cualquier riesgo asociado con su ejercicio de permisos bajo esta Licencia.

Cualquier pérdida económica o de algún tipo por el uso de este software el afectado no tendrá opción de pago de ningún tipo. Todo litigio legal las partes se someterán a tribunales únicamente en la jurisdicción de la Ciudad de México, país México.

Para soporte, uso y licenciamiento comercial se tendrá que realizar un acuerdo o contrato establecido entre OpenQbit o su corporativo y la parte interesada.

Los términos y condiciones de distribución comercialización pueden cambiar sin previo aviso, dirigirse al portal oficial de www.openqbit.com para ver cualquier modificación de cláusulas de soporte y licenciamiento no-comercial y comercial.

Cualquier persona, usuario, entidad privada, publica de cualquier índole legal o de cualquier parte del mundo quien simplemente use el software acepta sin condicionantes las clausulas establecidas en este documento y las que se puedan modificar en cualquier momento en el portal de www.openqbit.com sin previo aviso pudiendo ser aplicadas a discreción de OpenQbit en uso no-comercial o comercial.

Cualquier duda e información de Mini QRNG dirigirse a la comunidad de App Inventor o a las comunidades de diversos sistemas Blockly como son: AppBuilder, Trunkable, etc. y/o al correo opensource@openqbit.com por la demanda de preguntas puede tardar la respuesta de 3 a 5 días hábiles.

Soporte con uso comercial.

support@openqbit.com

Ventas uso comercial.

sales@openqbit.com

Información legal y preguntas o dudas de licenciamiento.

legal@openqbit.com

