

Resumen General

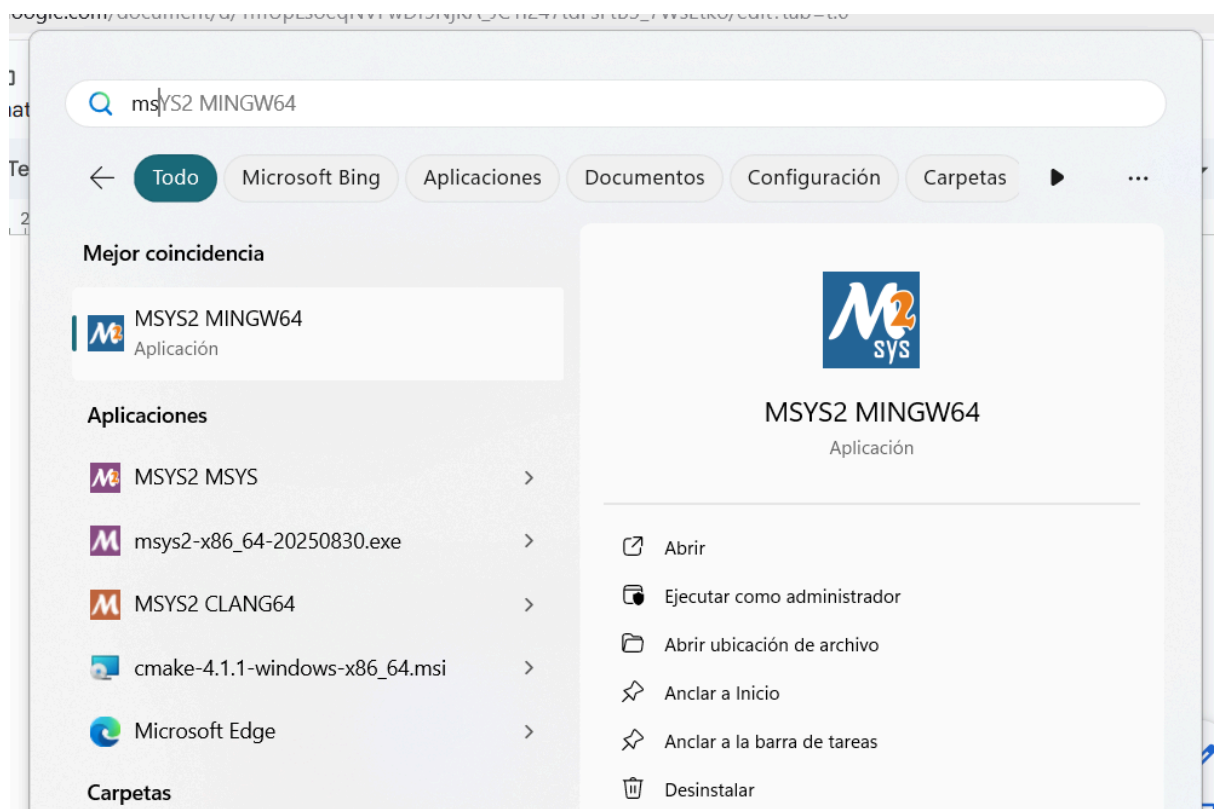
Los pasos que vamos a seguir son:

1. **Instalar el Entorno Base (MSYS2):** Prepararemos la terminal y el sistema base de compilación.
2. **Instalar el Editor de Código (VSCodium):** Instalaremos un editor de texto moderno para manejar los archivos del proyecto.
3. **Compilar el Programa:** Generamos los archivos **.exe**.
4. **Crear el Paquete Portable:** Juntaremos los **.exe** con todas sus librerías (**.dll**) para que funcionen por sí solos.

Fase 1: Instalar el Entorno Base (MSYS2)

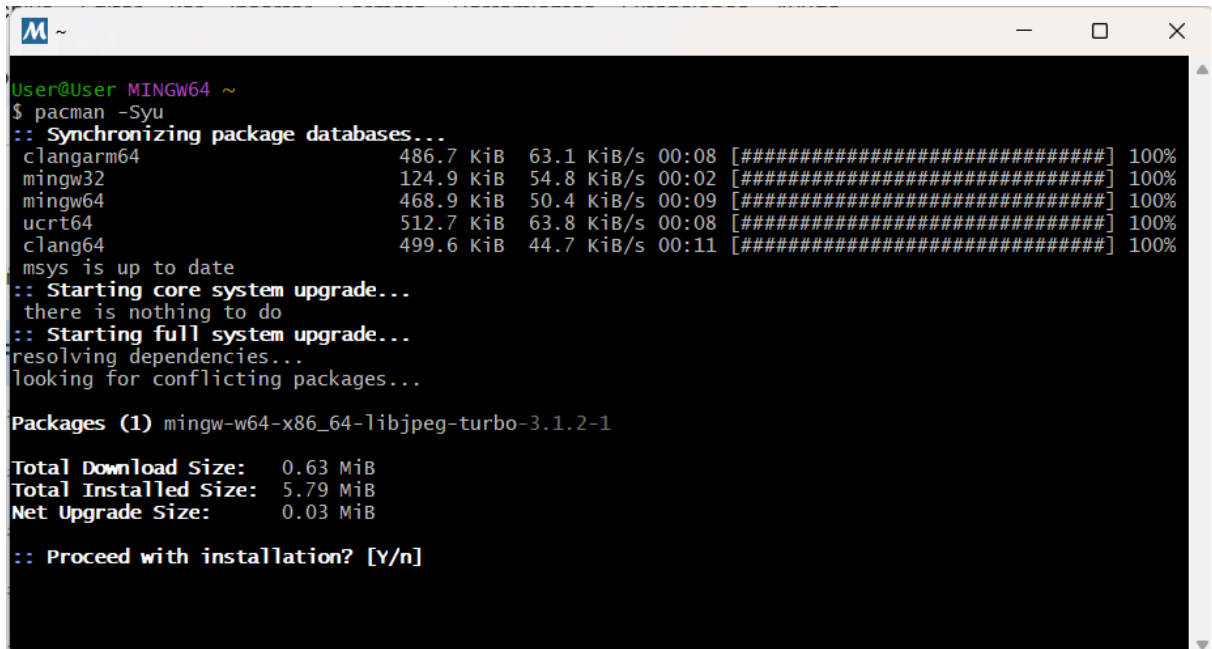
MSYS2 nos proporcionará una potente terminal estilo Linux dentro de windows y un gestor de paquetes (**pacman**) para instalar todo lo demás fácilmente.

1. **Descargar MSYS2:** Ve a la página oficial y descarga el instalador.
 - **Enlace:** msys2.org
2. **Instalar MSYS2:** Ejecuta el instalador y déjalo en la ruta por defecto (**C:\msys64**).
3. Actualizar el Sistema:
 - a. En el menú de inicio de Windows, busca y abre la terminal MSYS2 MINGW64 (la del icono azul, no la morada, la azul).



b. Dentro de la terminal, escribe el siguiente comando y presiona Enter. Escribe Y para confirmar cuando te pregunte.

`pacman -Syu`



```
User@User MINGW64 ~
$ pacman -Syu
:: Synchronizing package databases...
clangarm64             486.7 KiB   63.1 KiB/s   00:08 [#####] 100%
mingw32                124.9 KiB   54.8 KiB/s   00:02 [#####] 100%
mingw64                468.9 KiB   50.4 KiB/s   00:09 [#####] 100%
ucrt64                 512.7 KiB   63.8 KiB/s   00:08 [#####] 100%
clang64                499.6 KiB   44.7 KiB/s   00:11 [#####] 100%
msys is up to date
:: Starting core system upgrade...
there is nothing to do
:: Starting full system upgrade...
resolving dependencies...
looking for conflicting packages...

Packages (1) mingw-w64-x86_64-libjpeg-turbo-3.1.2-1

Total Download Size:   0.63 MiB
Total Installed Size:  5.79 MiB
Net Upgrade Size:      0.03 MiB

:: Proceed with installation? [Y/n]
```

c. Importante: Es normal que la terminal se cierre sola. Vuelve a abrirla y ejecuta `pacman -Syu` una vez más para asegurar que todo está actualizado.

Seguimos en la terminal de **MSYS2 MINGW64** para instalar todo lo necesario para compilar el proyecto.

Copia y pega el siguiente comando completo en la terminal. Es una sola línea larga.

```
pacman -S mingw-w64-x86_64-toolchain mingw-w64-x86_64-cmake git
mingw-w64-x86_64-libraw mingw-w64-x86_64-opencv mingw-w64-x86_64-cli11
mingw-w64-x86_64-wxwidgets3.2-msw mingw-w64-x86_64-gettext
mingw-w64-x86_64-eigen3
```

Presiona Enter. Acepta todas las opciones por defecto (presionando Enter) y escribe Y para confirmar la instalación. Esto tardará varios minutos. (Si algún paquete falla, vuelve a introducir el comando entero, no te preocupes, solo bajará lo que estaba pendiente o falló)

Fase 2: Instalar el Editor de Código (VSCodium) y clonar el repositorio (si no está ya clonado)

VSCodium es una versión 100% de código abierto de Visual Studio Code, un editor excelente para programar.

1. **Descargar VSCodium:** Ve a la página oficial de descargas

Enlace: vscodium.com

Busca la última versión para Windows y descarga el instalador .exe (System Install). Ejecutar e instalar con los valores propuestos por defecto.

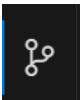
2. Ahora tenemos 2 opciones:

A) Ya tenemos el repositorio clonado con GIT en nuestro ordenador (solo hay que abrir la carpeta y decirle que confiamos en los autores)

MI CONSEJO. Si tenéis ya la carpeta con el repositorio clonado, movedla a:

C:\msys64\home\User\DynaRange

B) No tenemos el repositorio clonado, hay que clonarlo.

Si no lo tenemos clonado hay que ir a la izquierda, elegir el icono  y darle a “clone repository” (en la pantalla de Welcome también hay un enlace a ‘clone repository’). La dirección del repositorio a clonar es:

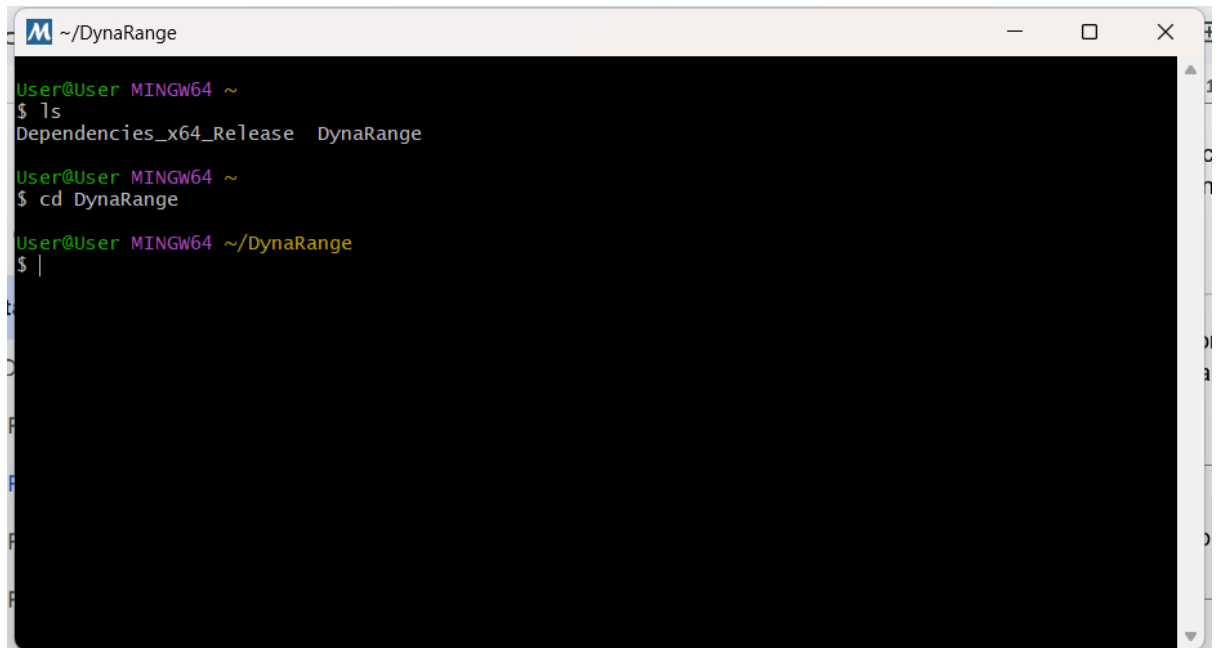
<https://github.com/hurodal/DynaRange.git>

elegimos la carpeta (C:\msys64\home\User\DynaRange) donde queremos que se clone y esperamos que se descargue y le decimos que confiamos en los autores.

Fase 3: Compilar el Programa

Desde la consola MSYS2 (la del icono azul), vamos a la carpeta DynaRange (en el explorador de archivos sería la carpeta C:\msys64\home\User\DynaRange, pero

cuando abrimos el terminal MSYS2 azul ya nos deja en C:\msys64\home\User)

A screenshot of a terminal window titled '~ /DynaRange'. The prompt is 'User@User MINGW64 ~'. The user enters '\$ ls', and the output is 'Dependencies_x64_Release DynaRange'. Then the user enters '\$ cd DynaRange', and the prompt changes to 'User@User MINGW64 ~/DynaRange'. The user then enters '\$ |' and the prompt returns to '\$ |'.

```
~ /DynaRange
User@User MINGW64 ~
$ ls
Dependencies_x64_Release DynaRange

User@User MINGW64 ~
$ cd DynaRange

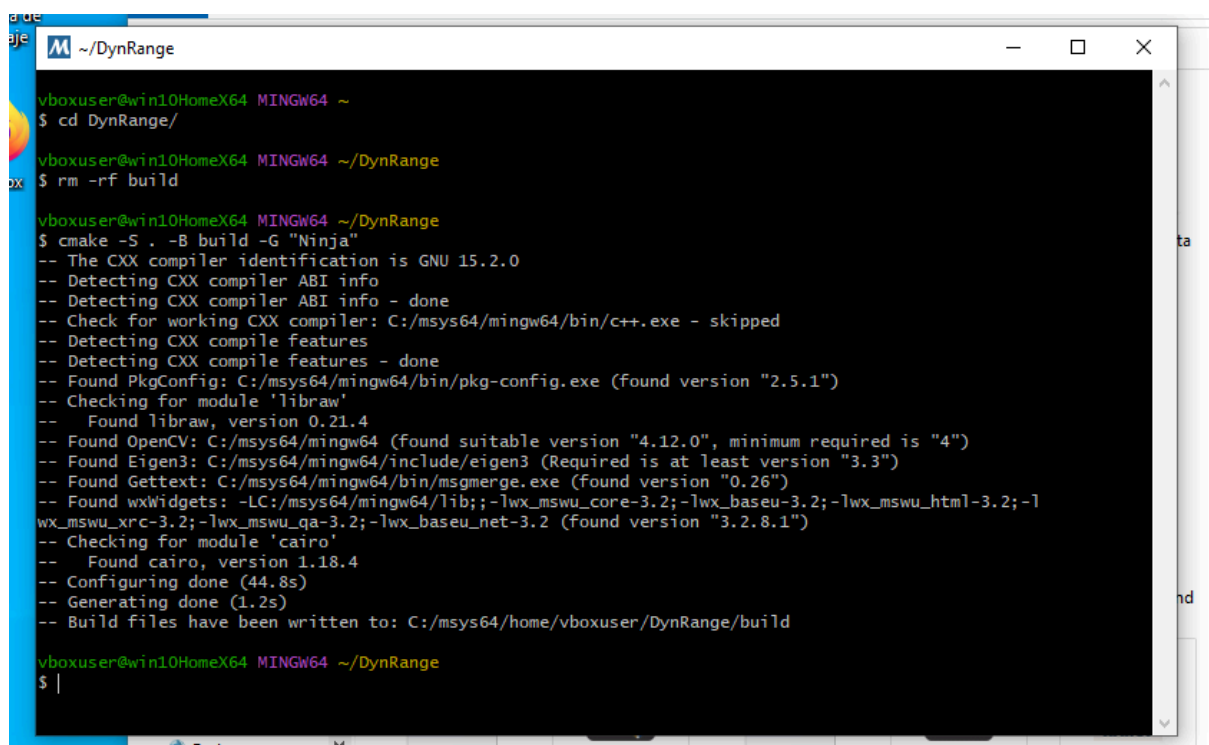
User@User MINGW64 ~/DynaRange
$ |
```

Así que solo tenemos que hacer un -> cd DynaRange

(Nota, en esta consola Linux podemos listar los ficheros con el comando -> ls

Una vez ahí, hay que ejecutar estos dos comandos:

cmake -S . -B build -G "Ninja"

A screenshot of a terminal window titled '~ /DynRange'. The prompt is 'vboxuser@win10HomeX64 MINGW64 ~'. The user enters '\$ cd DynRange/'. The prompt changes to 'vboxuser@win10HomeX64 MINGW64 ~/DynRange'. The user enters '\$ rm -rf build'. The prompt changes to 'vboxuser@win10HomeX64 MINGW64 ~/DynRange'. The user enters '\$ cmake -S . -B build -G "Ninja"'. The output shows the cmake configuration process, including detecting the CXX compiler (GNU 15.2.0), checking for working CXX compiler, detecting CXX compile features, and finding various libraries like libraw, OpenCV, Eigen3, Gettext, wxWidgets, and cairo. The output ends with 'Build files have been written to: C:/msys64/home/vboxuser/DynRange/build'. The user then enters '\$ |' and the prompt returns to '\$ |'.

```
~ /DynRange
vboxuser@win10HomeX64 MINGW64 ~
$ cd DynRange/

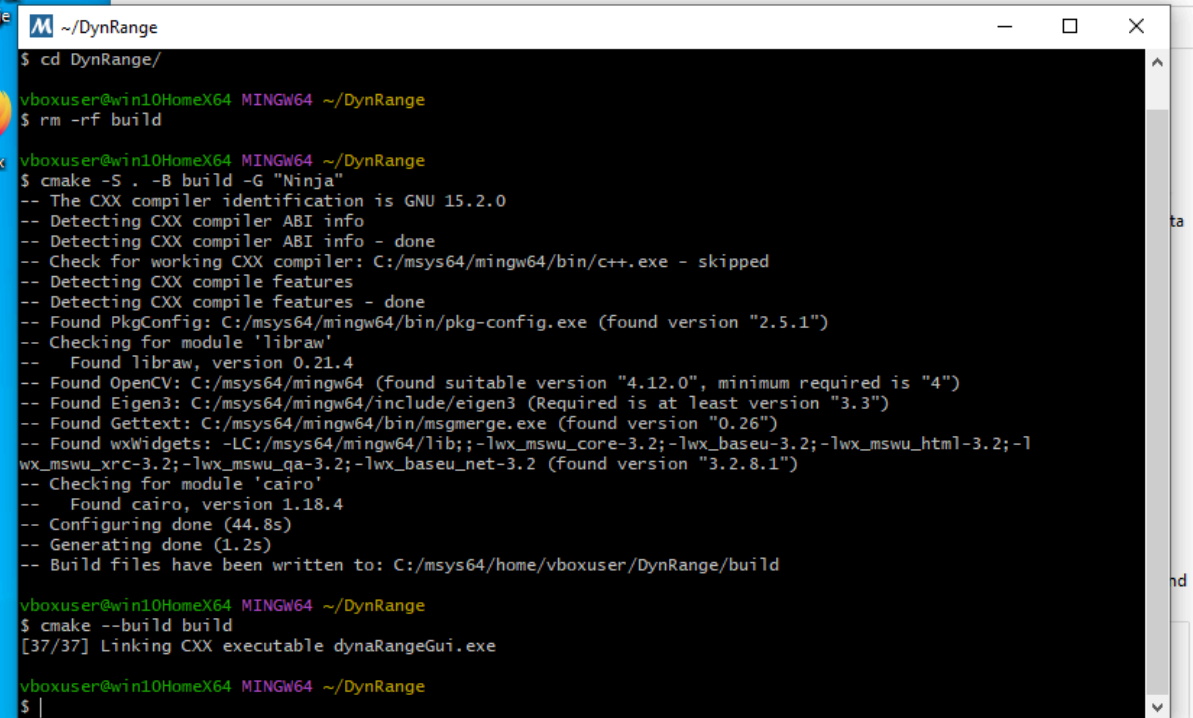
vboxuser@win10HomeX64 MINGW64 ~/DynRange
$ rm -rf build

vboxuser@win10HomeX64 MINGW64 ~/DynRange
$ cmake -S . -B build -G "Ninja"
-- The CXX compiler identification is GNU 15.2.0
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: C:/msys64/mingw64/bin/c++.exe - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Found PkgConfig: C:/msys64/mingw64/bin/pkg-config.exe (found version "2.5.1")
-- Checking for module 'libraw'
-- Found libraw, version 0.21.4
-- Found OpenCV: C:/msys64/mingw64 (found suitable version "4.12.0", minimum required is "4")
-- Found Eigen3: C:/msys64/mingw64/include/eigen3 (Required is at least version "3.3")
-- Found Gettext: C:/msys64/mingw64/bin/msgmerge.exe (found version "0.26")
-- Found wxWidgets: -LC:/msys64/mingw64/lib;-lwx_mswu_core-3.2;-lwx_baseu-3.2;-lwx_mswu_html-3.2;-lwx_mswu_xrc-3.2;-lwx_mswu_qa-3.2;-lwx_baseu_net-3.2 (found version "3.2.8.1")
-- Checking for module 'cairo'
-- Found cairo, version 1.18.4
-- Configuring done (44.8s)
-- Generating done (1.2s)
-- Build files have been written to: C:/msys64/home/vboxuser/DynRange/build

vboxuser@win10HomeX64 MINGW64 ~/DynRange
$ |
```

y luego

cmake --build build



```
~/DynRange
$ cd DynRange/
vboxuser@win10HomeX64 MINGW64 ~/DynRange
$ rm -rf build
vboxuser@win10HomeX64 MINGW64 ~/DynRange
$ cmake -S . -B build -G "Ninja"
-- The CXX compiler identification is GNU 15.2.0
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: C:/msys64/mingw64/bin/c++.exe - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Found PkgConfig: C:/msys64/mingw64/bin/pkg-config.exe (found version "2.5.1")
-- Checking for module 'libraw'
-- Found libraw, version 0.21.4
-- Found OpenCV: C:/msys64/mingw64 (found suitable version "4.12.0", minimum required is "4")
-- Found Eigen3: C:/msys64/mingw64/include/eigen3 (Required is at least version "3.3")
-- Found Gettext: C:/msys64/mingw64/bin/msgmerge.exe (found version "0.26")
-- Found wxWidgets: -LC:/msys64/mingw64/lib;-lwx_mswu_core-3.2;-lwx_baseu-3.2;-lwx_mswu_html-3.2;-lwx_mswu_xrc-3.2;-lwx_mswu_qa-3.2;-lwx_baseu_net-3.2 (found version "3.2.8.1")
-- Checking for module 'cairo'
-- Found cairo, version 1.18.4
-- Configuring done (44.8s)
-- Generating done (1.2s)
-- Build files have been written to: C:/msys64/home/vboxuser/DynRange/build
vboxuser@win10HomeX64 MINGW64 ~/DynRange
$ cmake --build build
[37/37] Linking CXX executable dynaRangeGui.exe
vboxuser@win10HomeX64 MINGW64 ~/DynRange
$ |
```

En la consola se mostrará cómo se va compilando el proyecto, y ya está, en la carpeta build/ ya tienes los dos ejecutables.

Fase 4: Crear el Paquete Portable e Instalador

Esta es la fase final para crear el paquete que puedes distribuir.

1. **Crear Carpeta Portable:** En tu Escritorio (o en tu carpeta del repositorio), crea una carpeta llamada **dynaRangePortable** (si no existe). Copia los archivos **dynaRange.exe** y **dynaRangeGui.exe** desde la carpeta **build** a esta nueva carpeta.
2. **Copiar las DLLs (en el repositorio ya están copiadas):**

Copia todas las DLLs necesarias desde **C:\msys64\mingw64\bin** a tu carpeta **dynaRangePortable**. (repito, en el repositorio que has clonado ya están las dll en dicho directorio, no hay que hacerlo)

3. **Instalar NSIS:** Descarga e instala NSIS desde [su web oficial](#).
4. **Organizar Archivos:** Asegúrate de que en tu Escritorio tienes estas tres cosas juntas:
 - La carpeta **dynaRangePortable** (con los **.exe** y las DLLs).
 - El archivo **crear_instalador_windows.nsi**.

- El archivo [icono_app.ico](#).
- 5. **Compilar el Instalador:** Abre el programa NSIS y pincha en el pequeño enlace que pone **"Compile NSIS Script"** que aparece bajo **"Compiler"** y busca el fichero que se llama "crear_instalador.windows"

¡Listo! El archivo [dynaRangeInstaller.exe](#) que aparecerá en tu Escritorio/Carpeta es el instalador final y profesional, listo para distribuir.