macOS Setup Tool Chain for Apple Silicon

Diego Hernandez Gomez / Piguaso

macOS Setup Tool Chain for Apple Silicon

Esta guía detalla los pasos para configurar la toolchain de desarrollo para el Raspberry Pi Pico en macOS con procesadores Apple Silicon.

Paso 1: Instalar Homebrew

Homebrew es un gestor de paquetes muy útil para instalar software en macOS. Si no tienes Homebrew instalado, puedes instalarlo ejecutando el siguiente comando en tu terminal:

```
/bin/bash -c "$(curl␣-fsSL␣https://raw.githubusercontent.com/Homebrew/install/
    HEAD/install.sh)"
```

Paso 2: Descargar e Instalar la Toolchain de ARM

A continuación, descarga e instala la herramienta ARM GNU Toolchain compatible con Apple Silicon desde el siguiente enlace:

- Página de descarga de ARM GNU Toolchain

Descarga directamente el paquete específico para computadoras Apple Silicon:

- https://developer.arm.com/-/media/Files/downloads/gnu/13.3.rel1/binrel/arm-gnu-toolchain-13.3.rel1-darwin-arm64-arm-none-eabi.pkg

Abre el archivo `.pkg` y sigue las instrucciones del instalador para completar la instalación.

Verificar la instalación

Después de la instalación, verifica que la toolchain se haya instalado correctamente en la carpeta `/Applications`. Ejecuta el siguiente comando en la terminal para comprobar la ubicación:

```
ls /Applications/ArmGNUToolchain/13.3.rel1/arm-none-eabi
```

Paso 3: Preparar el Entorno

Con Homebrew instalado, ejecuta los siguientes comandos en la terminal para instalar `git` y `python3`.

```
# Instalar git
brew install git

# Instalar python3
brew install python3
```

Paso 4: Crear Directorio de Trabajo e Instalar el SDK y Ejemplos

1. Crear el directorio para el SDK

```
mkdir pico
cd pico
```

2. Clonar el SDK desde GitHub

```
git clone -b master https://github.com/raspberrypi/pico-sdk.git
```

3. Inicializar submódulos en el SDK

```
cd pico-sdk
git submodule update --init
cd ..
```

4. Clonar los ejemplos

```
git clone -b master https://github.com/raspberrypi/pico-examples.git
```

Paso 5: Configurar Variables de Entorno

Configura las variables de entorno necesarias para el SDK y la toolchain.

```
export PICO_SDK_PATH=../../pico-sdk
export PICO_TOOLCHAIN_PATH=/Applications/ArmGNUToolchain/13.3.rel1/arm-none-eabi
```

Paso 6: Crear un Proyecto de Ejemplo

1. Crear el directorio para el proyecto

```
mkdir test
cd test
```

2. Copiar el archivo necesario de configuración

```
cp ../pico-sdk/external/pico_sdk_import.cmake .
```

3. Crear el archivo fuente en C (`test.c`)

```
nano test.c
```

Este código de ejemplo en `test.c` enciende y apaga el LED en intervalos definidos. Puedes ajustar el tiempo de encendido y apagado del LED cambiando los valores en `LED ON` y `LED OFF`.

```
#include "pico/stdlib.h"
#include "pico/cyw43_arch.h"

int main() {
    stdio_init_all();
    if (cyw43_arch_init()) {
        printf("Wi-Fi␣init␣failed");
        return -1;
    }
    while (true) {
        cyw43_arch_gpio_put(CYW43_WL_GPIO_LED_PIN, 1); // LED ON
        sleep_ms(250); // Tiempo LED ON
        cyw43_arch_gpio_put(CYW43_WL_GPIO_LED_PIN, 0); // LED OFF
        sleep_ms(250); // Tiempo LED OFF
    }
}
```

4. Crear el archivo `CMakeLists.txt`

```
nano CMakeLists.txt
```

Inserta el siguiente contenido en `CMakeLists.txt`:

```
cmake_minimum_required(VERSION 3.13)
include(pico_sdk_import.cmake)
project(test_project C CXX ASM)
set(CMAKE_C_STANDARD 11)
set(CMAKE_CXX_STANDARD 17)
pico_sdk_init()
add_executable(test test.c)
pico_add_extra_outputs(test)
target_link_libraries(test pico_cyw43_arch_none pico_stdlib)
```

Paso 7: Compilación del Proyecto

1. Crear el directorio de compilación

```
mkdir build
cd build
```

2. Configurar la compilación para la tarjeta `pico_w`

```
cmake -DPICO_BOARD=pico_w ..
```

3. Compilar el proyecto

```
make
```

Paso 8: Finalización

Si la compilación fue exitosa, se generará un archivo `.uf2` en el directorio `build`. Puedes cargar este archivo en tu Raspberry Pi Pico W manteniendo presionado el botón `BOOTSEL` mientras lo conectas a tu computadora.