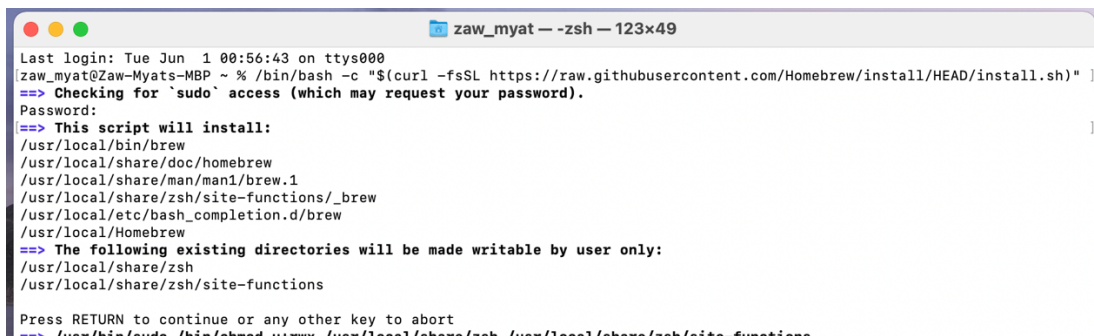


Raspberry Pi Pico အား C Language ဖြင့် ပရိုဂရမ် ရေးရန်ပြင်ဆင်ခြင်း

Apple macOS မှာ install လုပ်တဲ့ပုံစံက Linux နဲ့ဆင်တူပါတယ်။ ကျွန်တော်ကတော့ macOS user ဖြစ်တဲ့အတွက်ကြောင့် macOS မှာ ဘယ်လိုမျိုး install လုပ်မလဲဆိုတာကို ပြောပြပါမယ်။ macOS မှာ install လုပ်ဖို့အတွက် ပထမဦးဆုံးလိုအပ်တာကတော့ “Homebrew” ဖြစ်ပါတယ်။ အဲဒီတော့ Homebrew ကို အရင်ဆုံး install လုပ်ရပါမယ်။ “Homebrew” တင်ဖို့ရန်အတွက် Terminal ကိုသွားပါတယ်။ Terminal ကိုမသွားခင် သင့်ရဲ့ Mac ကို အင်တာနက်နဲ့ချိတ်ဆက်ထားဖို့လိုအပ်ပါမယ်။

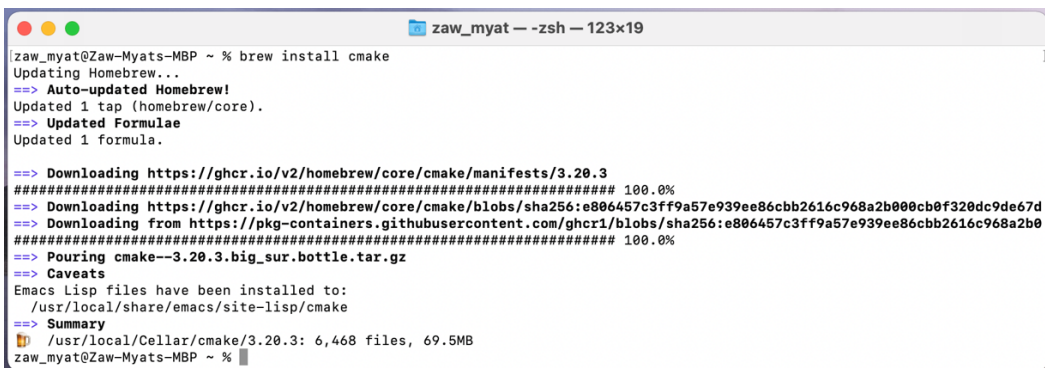
```
$ /bin/bash -c "$(curl -fsSL  
https://raw.githubusercontent.com/Homebrew/install/master/install.sh)"
```



```
zaw_myat — zsh — 123x49
Last login: Tue Jun 1 00:56:43 on ttys000
zaw_myat@Zaw-Myats-MBP ~ % /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
==> Checking for `sudo` access (which may request your password).
Password:
==> This script will install:
/usr/local/bin/brew
/usr/local/share/doc/homebrew
/usr/local/share/man/man1/brew.1
/usr/local/share/zsh/site-functions/_brew
/usr/local/etc/bash_completion.d/brew
/usr/local/Homebrew
==> The following existing directories will be made writable by user only:
/usr/local/share/zsh
/usr/local/share/zsh/site-functions
Press RETURN to continue or any other key to abort
--> /usr/bin/sudo /bin/chmod u+wx /usr/local/share/zsh /usr/local/share/zsh/site-functions
```

အဲဒီနောက် Toolchain ကို install လုပ်ပါမယ်။

```
$ brew install cmake
$ brew tap ArmMbed/homebrew-formulae
```



```
zaw_myat — zsh — 123x19
zaw_myat@Zaw-Myats-MBP ~ % brew install cmake
Updating Homebrew...
==> Auto-updated Homebrew!
Updated 1 tap (homebrew/core).
==> Updated Formulae
Updated 1 formula.

==> Downloading https://ghcr.io/v2/homebrew/core/cmake/manifests/3.20.3
##### 100.0%
==> Downloading https://ghcr.io/v2/homebrew/core/cmake/blobs/sha256:e806457c3ff9a57e939ee86cbb2616c968a2b00cb0f320dc9de67d
==> Downloading from https://pkg-containers.githubusercontent.com/ghcr1/blobs/sha256:e806457c3ff9a57e939ee86cbb2616c968a2b0
##### 100.0%
==> Pouring cmake--3.20.3.big_sur.bottle.tar.gz
==> Caveats
Emacs Lisp files have been installed to:
  /usr/local/share/emacs/site-lisp/cmake
==> Summary
📦 /usr/local/Cellar/cmake/3.20.3: 6,468 files, 69.5MB
zaw_myat@Zaw-Myats-MBP ~ %
```

```
$ brew install arm-none-eabi-gcc
```

```
zaw_myat -- zsh -- 123x23
[zaw_myat@Zaw-Myats-MBP ~ % brew install arm-none-eabi-gcc
=> Installing arm-none-eabi-gcc from armbed/formulae
=> Downloading https://developer.arm.com/-/media/Files/downloads/gnu-rm/9-2019q4/RC2.1/gcc-arm-none-eabi-9-2019-q4-major-m
=> Downloading from https://armkeil.blob.core.windows.net/developer/Files/downloads/gnu-rm/9-2019q4/RC2.1/gcc-arm-none-eab
##### 100.0%
Warning: Your Xcode (12.4) is outdated.
Please update to Xcode 12.5 (or delete it).
Xcode can be updated from the App Store.

Warning: A newer Command Line Tools release is available.
Update them from Software Update in System Preferences or run:
  softwareupdate --all --install --force

If that doesn't show you any updates, run:
  sudo rm -rf /Library/Developer/CommandLineTools
  sudo xcode-select --install

Alternatively, manually download them from:
  https://developer.apple.com/download/more/.
You should download the Command Line Tools for Xcode 12.5.

📦 /usr/local/Cellar/arm-none-eabi-gcc/9-2019-q4-major: 6,009 files, 533.4MB, built in 28 seconds
zaw_myat@Zaw-Myats-MBP ~ %
```

အဲဒီနောက် Pico SDK နဲ့ Example code တွေကိုတင်ရမှာဖြစ်ပါတယ်။ pico-examples ဆိုတဲ့ repository (<https://github.com/raspberrypi/pico-examples>) မှာ Raspberry Pi Pico အတွက် example application တွေပါဝင်ပြီးတော့ အဲဒီ pio-examples repository ကို pico-sdk (<https://github.com/raspberrypi/pico-sdk>) သုံးပြီးရေးထားတာဖြစ်ပါတယ်။ အဲဒီ repository ၂ ခု ကို Mac ရဲ့တစ်နေရာရာမှာ clone လုပ်ဖို့လိုအပ်ပါတယ်။

```
$ cd ~/
$ mkdir pico
$ cd pico
```

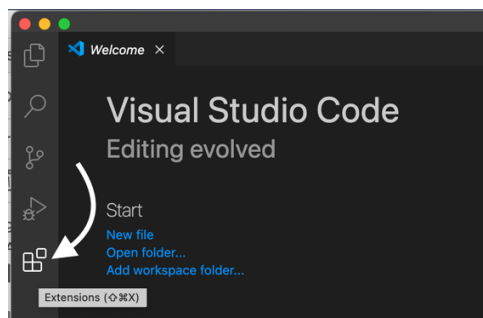
အရင်ဆုံး clone လုပ်ထားတဲ့ပိုင်တွေကို သိမ်းမယ့် folder (pico) ကို mkdir နဲ့တည်ဆောက်လိုက်ပါတယ်။ အဲဒီနောက် pico-sdk နဲ့ pico-examples ဆိုတဲ့ git repository (၂) ခုကို အောက်ပါအတိုင်း terminal မှာ clone လုပ်လိုက်ပါတယ်။

```
$ git clone -b master https://github.com/raspberrypi/pico-sdk.git
$ cd pico-sdk
$ git submodule update --init
$ cd ..
$ git clone -b master https://github.com/raspberrypi/pico-examples.git
```

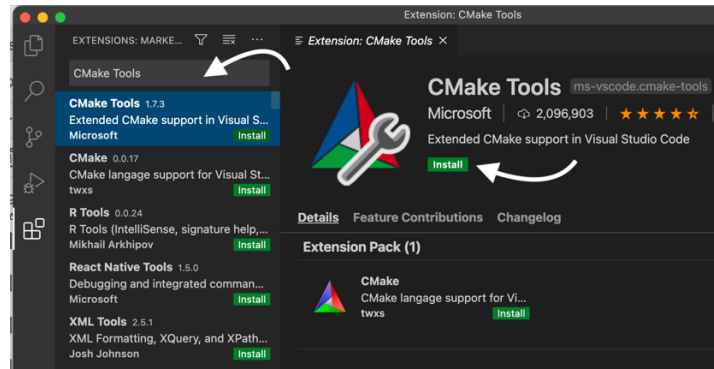
```
pico -- zsh -- 123x25
zaw_myat@Zaw-Myats-MBP Downloads % mkdir pico
zaw_myat@Zaw-Myats-MBP Downloads % cd pico
zaw_myat@Zaw-Myats-MBP pico % git clone -b master https://github.com/raspberrypi/pico-sdk.git
Cloning into 'pico-sdk'...
remote: Enumerating objects: 3539, done.
remote: Counting objects: 100% (695/695), done.
remote: Compressing objects: 100% (249/249), done.
remote: Total 3539 (delta 537), reused 478 (delta 421), pack-reused 2844
Receiving objects: 100% (3539/3539), 1.60 MiB | 1.02 MiB/s, done.
Resolving deltas: 100% (1741/1741), done.
zaw_myat@Zaw-Myats-MBP pico % cd pico-sdk
zaw_myat@Zaw-Myats-MBP pico-sdk % git submodule update --init
Submodule 'tinyusb' (https://github.com/raspberrypi/tinyusb.git) registered for path 'lib/tinyusb'
Cloning into '/Users/zaw_myat/Downloads/pico/pico-sdk/lib/tinyusb'...
Submodule path 'lib/tinyusb': checked out '11c23f88bf42f64ce14b8a7b0b2a4e207dc4dd12'
zaw_myat@Zaw-Myats-MBP pico-sdk % cd ..
zaw_myat@Zaw-Myats-MBP pico % git clone -b master https://github.com/raspberrypi/pico-examples.git
Cloning into 'pico-examples'...
remote: Enumerating objects: 683, done.
remote: Counting objects: 100% (122/122), done.
remote: Compressing objects: 100% (95/95), done.
remote: Total 683 (delta 48), reused 51 (delta 23), pack-reused 561
Receiving objects: 100% (683/683), 2.25 MiB | 2.80 MiB/s, done.
Resolving deltas: 100% (226/226), done.
zaw_myat@Zaw-Myats-MBP pico %
```

Raspberry Pi Pico အတွက် C Program တွေရေးဖို့ရန် Visual Studio Code (VSCode) ကိုအသုံးပြုပါမယ်။ VS Code ဟာ Windows, Linux, Mac သုံးခုစလုံးအတွက်အလုပ်လုပ်နိုင်တဲ့ Cross-platform IDE တစ်ခုဖြစ်ပါတယ်။ အခုကတော့ MacOS အတွက် download ဆွဲပြီး install ပြုလုပ်လိုက်ပါ။

Install ပြုလုပ်ပြီးတဲ့အချိန်မှာ VSCode ကိုဖွင့်ပြီး VSCode ရဲ့ ဘယ်ဘက်ချမ်း Toolbar ပေါ်က “Extensions” icon လေးကိုနှိပ်လိုက်ပါမယ်။



Search box မှာ “CMake Tools” ဆိုပြီး ရှာလိုက်ပါ။ ပြီးရင် “Install” ခလုတ်ကိုနှိပ်ပြီး CMake Tools ကိုတင်ပါမယ်။



အခုဆက်လုပ်ရမှာကတော့ “PICO_SDK_PATH” environment variable ကိုသတ်မှတ်ပေးရမှာပါ။ ဒါကြောင့် terminal ကနေတစ်ဆင့် စောနက git clone လုပ်ထားတဲ့ “pico-examples” ဆိုတဲ့ ဖိုဒါကို navigate လုပ်ပါမယ်။ အဲဒီနောက် “mkdir .vscode” ဆိုတဲ့ command နဲ့ “.vscode” အမည်နဲ့ directory အသစ်တစ်ခုတည်ဆောက်လိုက်ပါမယ်။ အသစ်ဆောက်လိုက်တဲ့ “.vscode” directory ထဲကို “cd .vscode” နဲ့ဝင်ပြီးတော့ ဖိုင်အသစ်တစ်ခုထပ်ထည့်ရပါမယ်။

```

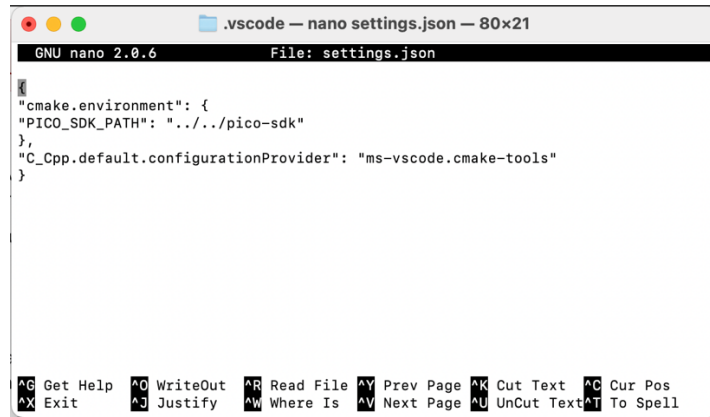
zaw_myat@Zaw-Myats-MBP pico-examples % mkdir .vscode
zaw_myat@Zaw-Myats-MBP pico-examples % ls
CMakeLists.txt      flash                pwm
LICENSE.TXT         gpio                reset
README.md           hello_world         rtc
adc                 i2c                 spi
blink               ide                 system
clocks              interp              timer
cmake               multicore            uart
divider             pico_sdk_import.cmake  usb
dma                 picoboard            watchdog
example_auto_set_url.cmake  pio
zaw_myat@Zaw-Myats-MBP pico-examples % cd .vscode
zaw_myat@Zaw-Myats-MBP .vscode % ls
zaw_myat@Zaw-Myats-MBP .vscode %

```

ထည့်ရမယ့်ဖိုင်က Pico SDK ရဲ့ location ကို CMake Tools တွေ့ဆုံပြောပြမယ့် “settings.json” ဖိုင်ဖြစ်ပါတယ်။

\$ nano settings.json

Nano editor နဲ့ ရေးပြီး “settings.json” ဖိုင်ဆောက်ဖို့ရန်အတွက် “nano settings.json” ဆိုပြီး terminal မှာ ရိုက်လိုက်ပါတယ်။

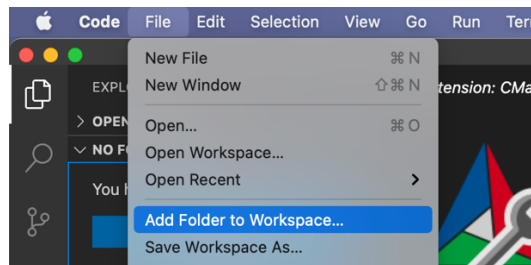


```
GNU nano 2.0.6 File: settings.json
{
  "cmake.environment": {
    "PICO_SDK_PATH": "../../pico-sdk"
  },
  "C_Cpp.default.configurationProvider": "ms-vscode.cmake-tools"
}
```

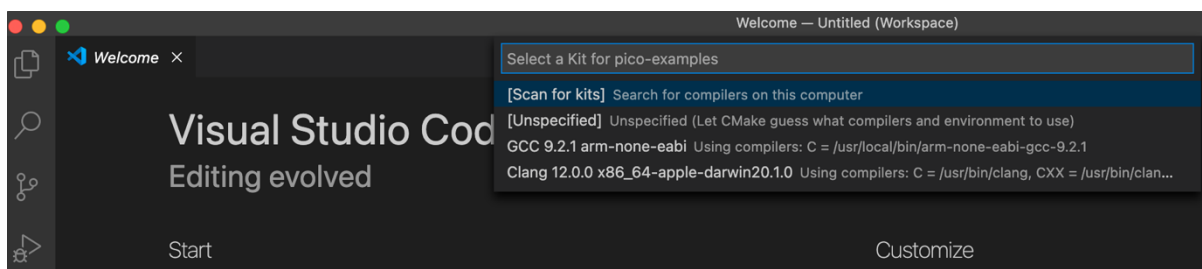
Nano editor မှာ အခုလိုရေးပြီး “Save” ပါမယ်။

```
{
  "cmake.environment": {
    "PICO_SDK_PATH": "../../pico-sdk"
  },
  "C_Cpp.default.configurationProvider": "ms-vscode.cmake-tools"
}
```

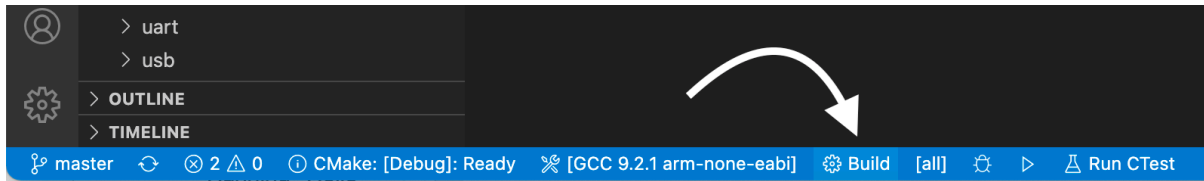
အဲဒီနောက် VSCode ရဲ့ File Menu ထဲက “Add Folder to Workspace...” ကိုဝင်ပြီး “pico-examples” ဖိုဒါ ကို navigate လုပ်ဖွင့်လိုက်ပါမယ်။



ပြီးနောက် အောက်ပါပုံအတိုင်း “GCC arm-none-eabi” compiler ကိုရွေးပေးလိုက်ပါ။



အရှိရှင်းဆုံး LED blink project ကို “Build လုပ်ပြီးစမ်းကြည့်ပါမယ်။” “Build” ကိုနှိပ်ပါ။



အောက်ပါပုံအတိုင်း “Build” လုပ်တဲ့လုပ်ငန်းစဉ်ပြီးဆုံးတဲ့နောက်မှာ “build” directory အသစ်တစ်ခု ထပ်ဆောက်ဖို့ လိုပါသေးတယ်။

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
[build] [100%] Building C object adc/dma_capture/CMakeFiles/adc_dma_capture.dir/Users/zaw_myat/Down
[build] [100%] Building C object adc/dma_capture/CMakeFiles/adc_dma_capture.dir/Users/zaw_myat/Down
[build] [100%] Building C object adc/dma_capture/CMakeFiles/adc_dma_capture.dir/Users/zaw_myat/Down
[build] [100%] Building C object adc/dma_capture/CMakeFiles/adc_dma_capture.dir/Users/zaw_myat/Down
[build] [100%] Linking CXX executable pio_ws2812_parallel.elf
[build] [100%] Linking CXX executable adc_dma_capture.elf
[build] [100%] Built target pio_ws2812_parallel
[build] [100%] Built target pio_ws2812
[build] [100%] Built target adc_dma_capture
[build] Build finished with exit code 0

: Ready  [GCC 9.2.1 arm-none-eabi]  Build  [all]  ▶
```

အပေါ်မှာ ဖန်တီးခဲ့တဲ့ “pico” directory ထဲကို ဝင်ပါမယ်။ အဲဒီမှာ “pico-examples” နဲ့ “pico-sdk” ဆိုပြီး နှစ်ခုရှိတဲ့အနက် “pico-examples” ထဲကို ဝင်လိုက်ပါ။

```
$ cd pico-examples
$ mkdir build
$ cd build
```

အဲဒီနောက် “cmake ..” နဲ့ CMake build directory ကိုပြင်ဆင်လိုက်ပါတယ်။

```
$ cmake ..
```



```
build — -zsh — 87x22
zaw_myat@Zaw-Myats-MBP pico-examples % mkdir build
mkdir: build: File exists
zaw_myat@Zaw-Myats-MBP pico-examples % cd build
zaw_myat@Zaw-Myats-MBP build % cmake ..
PICO_SDK_PATH is /Users/zaw_myat/Downloads/pico/pico-sdk
PICO platform is rp2040.
PICO compiler is
PICO_GCC_TRIPLE defaulted to arm-none-eabi
Using regular optimized debug build (set PICO_DEOPTIMIZED_DEBUG=1 to de-optimize)
PICO target board is pico.
Using board configuration from /Users/zaw_myat/Downloads/pico/pico-sdk/src/boards/include/boards/pico.h
TinyUSB available at /Users/zaw_myat/Downloads/pico/pico-sdk/lib/tinyusb/src/portable/raspberrypi/rp2040; adding USB support.
Compiling TinyUSB with CFG_TUSB_DEBUG=1
-- Could NOT find Doxygen (missing: DOXYGEN_EXECUTABLE)
ELF2UF2 will need to be built
PIOASM will need to be built
-- Configuring done
-- Generating done
-- Build files have been written to: /Users/zaw_myat/Downloads/pico/pico-examples/build
zaw_myat@Zaw-Myats-MBP build %
```

ဒီအဆင့်ပြီးသွားရင်တော့ CMake ဟာ “pico-examples” directory tree တစ်ခုလုံးအတွက် build area တစ်ခု တည်ဆောက်လို့ပြီးသွားပါပြီ။ “make” လို့ခေါ်ပြီးတော့ example application အားလုံးကို build လိုက်လို့ရပါတယ်။ ဒါပေမယ့် “blink” example လေးတစ်ခုကိုဘဲ build လုပ်ကြည့်ပါမယ်။

ဒါကြောင့် လက်ရှိရောက်နေတဲ့ directory ကနေ “blink” directory ကို “cd blink” နဲ့ ပြောင်းလိုက်ပါတယ်။

```
$ cd blink
```

```
$ make -j4
```

“make” အစား “make -j4” ကို သုံးတဲ့အခါမှာ အလုပ်လေးခုကို တပြိုင်တည်းဆောင်ရွက်တာဖြစ်တဲ့အတွက် speed ပိုမြန်စေပါတယ်။ “make -j4” နဲ့ blink ပရောဂျက်လေးကို build လုပ်လိုက်ပါတယ်။

```
PIOASM will need to be built
-- Configuring done
-- Generating done
-- Build files have been written to: /Users/zaw_myat/Downloads/pico/pico-examples/build
zaw_myat@Zaw-Myats-MBP build % ls
CMakeCache.txt      elf2uf2             pioasm
CMakeFiles          flash              pwm
Makefile            generated          reset
adc                 gpio               rtc
blink               hello_world        spi
clocks              i2c                system
cmake               interp             timer
cmake_install.cmake multicore           uart
compile_commands.json pico-sdk            usb
divider            picoboard          watchdog
dma                 pio
zaw_myat@Zaw-Myats-MBP build % cd blink
zaw_myat@Zaw-Myats-MBP blink % make -j4
[Scanning dependencies of target bs2_default]
[ 0%] Performing build step for 'ELF2UF2Build'
[ 0%] Built target bs2_default
[ 0%] Built target bs2_default_padded_checksummed_asm
[Consolidate compiler generated dependencies of target elf2uf2]
[100%] Built target elf2uf2
[ 0%] No install step for 'ELF2UF2Build'
[ 0%] Completed 'ELF2UF2Build'
[ 0%] Built target ELF2UF2Build
[Scanning dependencies of target blink]
[Consolidate compiler generated dependencies of target blink]
[100%] Built target blink
zaw_myat@Zaw-Myats-MBP blink %
```

Build လုပ်ပြီးတဲ့အခါ debugger ကနေအသုံးပြုမယ့် “.elf” ဖိုင်နဲ့ Raspberry Pi Pico ရဲ့ USB Mass Storage ထဲကို ဆွဲထည့်လို့ရမယ့် “.uf2” ဖိုင်တွေထွက်လာမှဖြစ်ပါတယ်။ “blink” ပရိုဂရမ်လေး ကတော့ Raspberry Pi Pico မှာပါပြီးသား GP25 က LED ကို မှိတ်တုတ်မှိတ်တုတ်ဖြစ်စေမှာဖြစ်ပါတယ်။

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

int main() {
    #ifndef PICO_DEFAULT_LED_PIN
    #warning blink example requires a board with a regular LED
    #else
        const uint LED_PIN = PICO_DEFAULT_LED_PIN;
        gpio_init(LED_PIN);
        gpio_set_dir(LED_PIN, GPIO_OUT);
        while (true) {
            gpio_put(LED_PIN, 1);
            sleep_ms(250);
            gpio_put(LED_PIN, 0);
            sleep_ms(250);
        }
    #endif
}
```