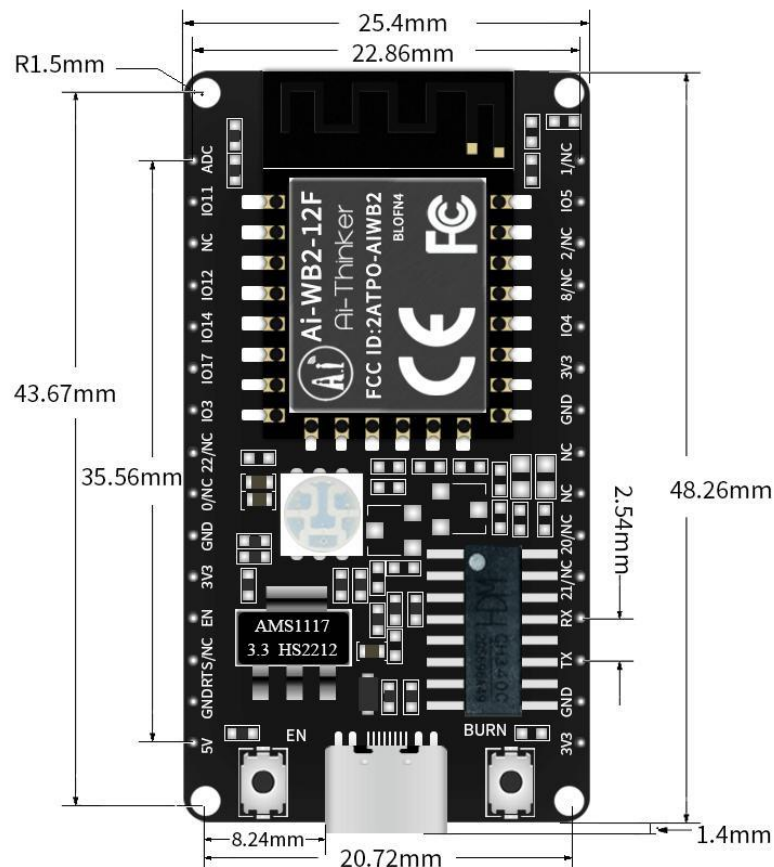


# AI-WB2-12F-KIT DEVELOPMENT BOARD TUTORIAL (FOR WINDOWS USERS)

# OVERVIEW

The ***Ai-WB2-12F Development Board*** is one of the latest modules of the ***Ai-WB2*** series developed by Shenzhen Ai-Thinker Technology Co., Ltd. It uses ***Buffalo Lab's BL602 IoT chip*** as the core processor, which supports Wi-Fi 802.11b/g/n and BLE 5.0 protocol. Therefore, this module can be widely used in mobile devices, wearable electronic devices, smart home, and many other fields.

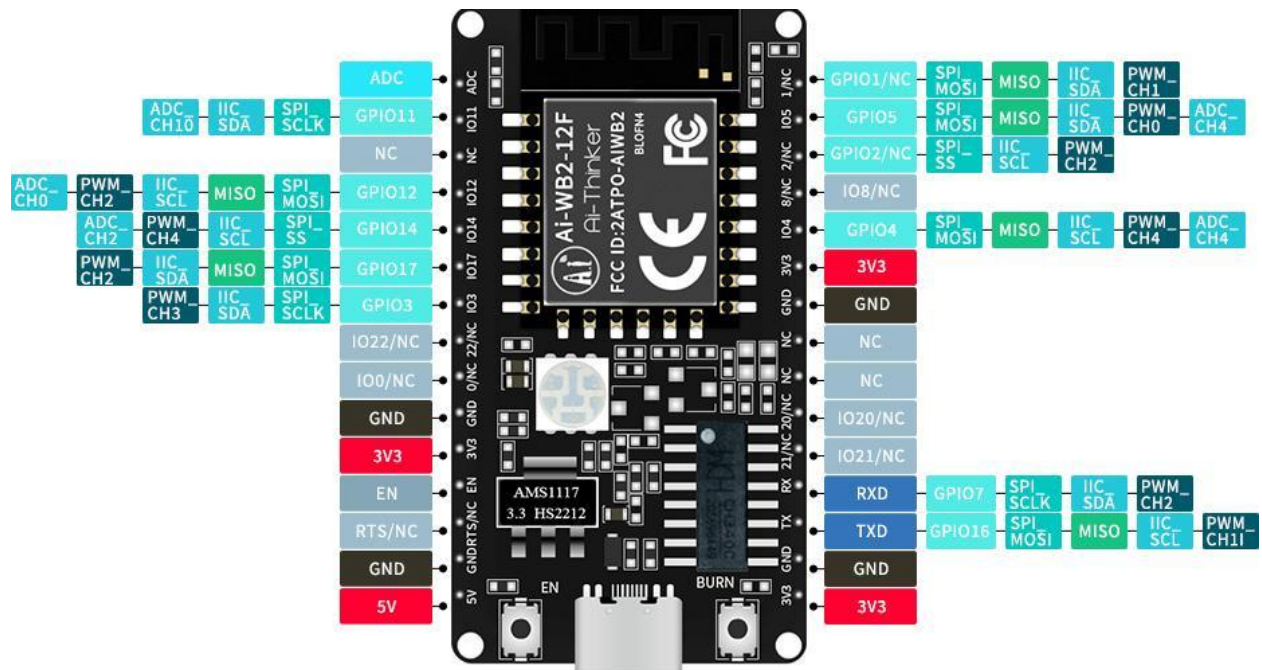
## APPEARANCE



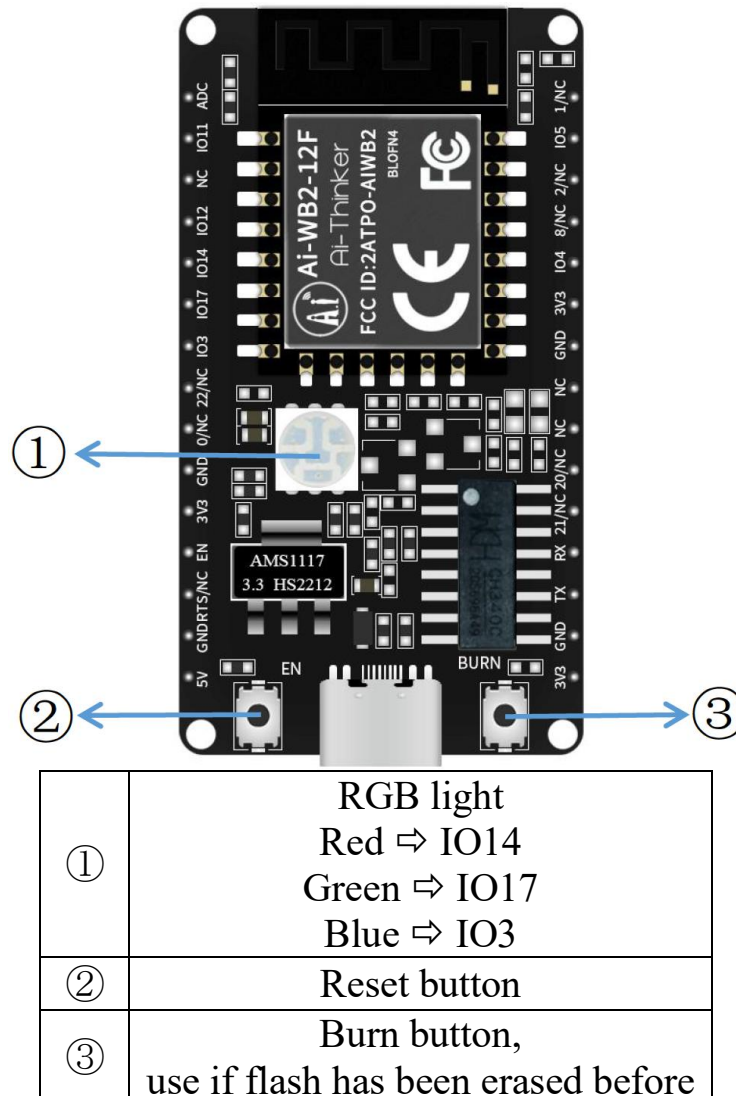
*Front view*



*Back view*



*Pinouts*



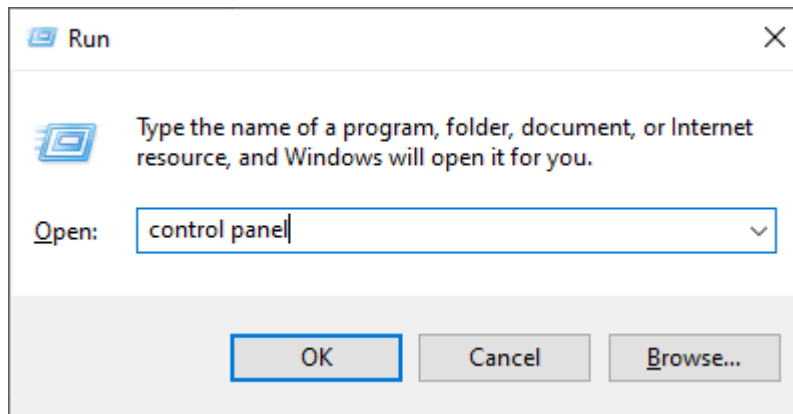
## PREREQUISITES

- Windows 8.1 or newer, with appropriate environment variables set (details in the next step).
- [MSYS2](#).
- A tool for UART communication, such as [Termite](#).
- [Eclipse IDE for Embedded C/C++ Developers](#).
- [Software Development Kit for Ai-WB2 Series](#).

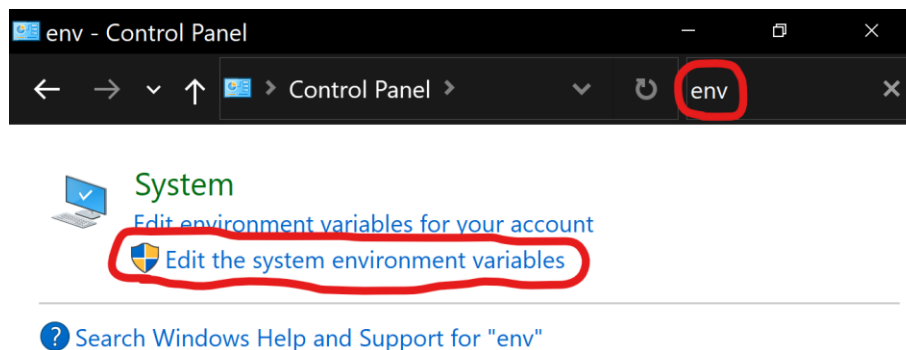
If you have already met all requirements, feel free to skip the rest of these steps below.

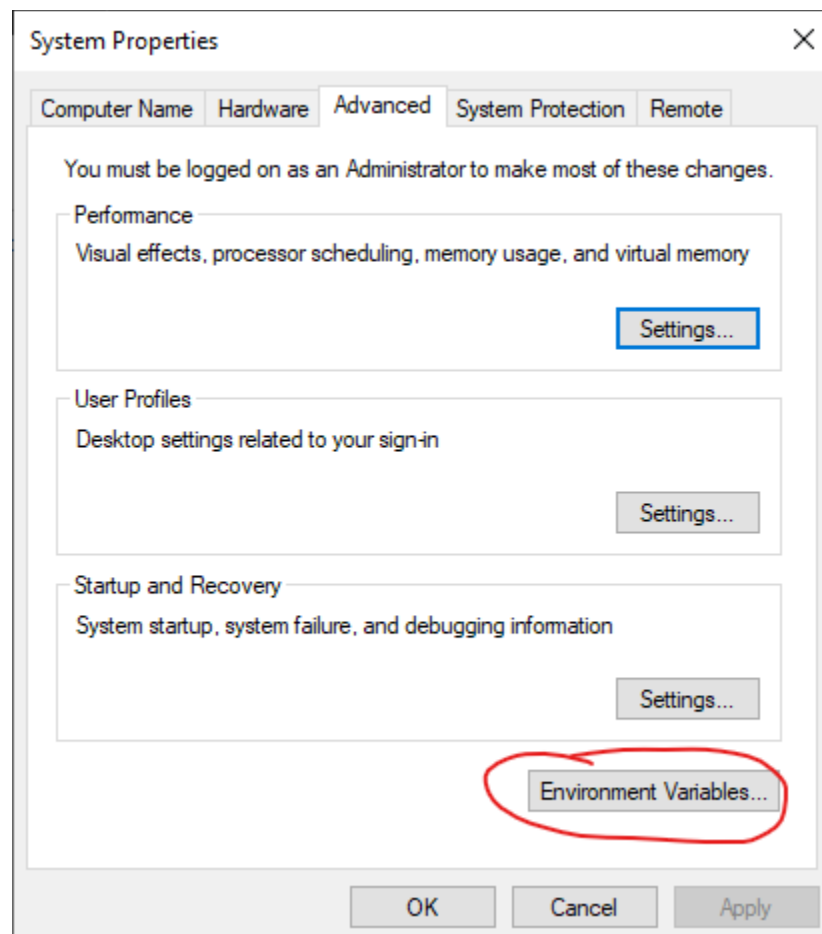
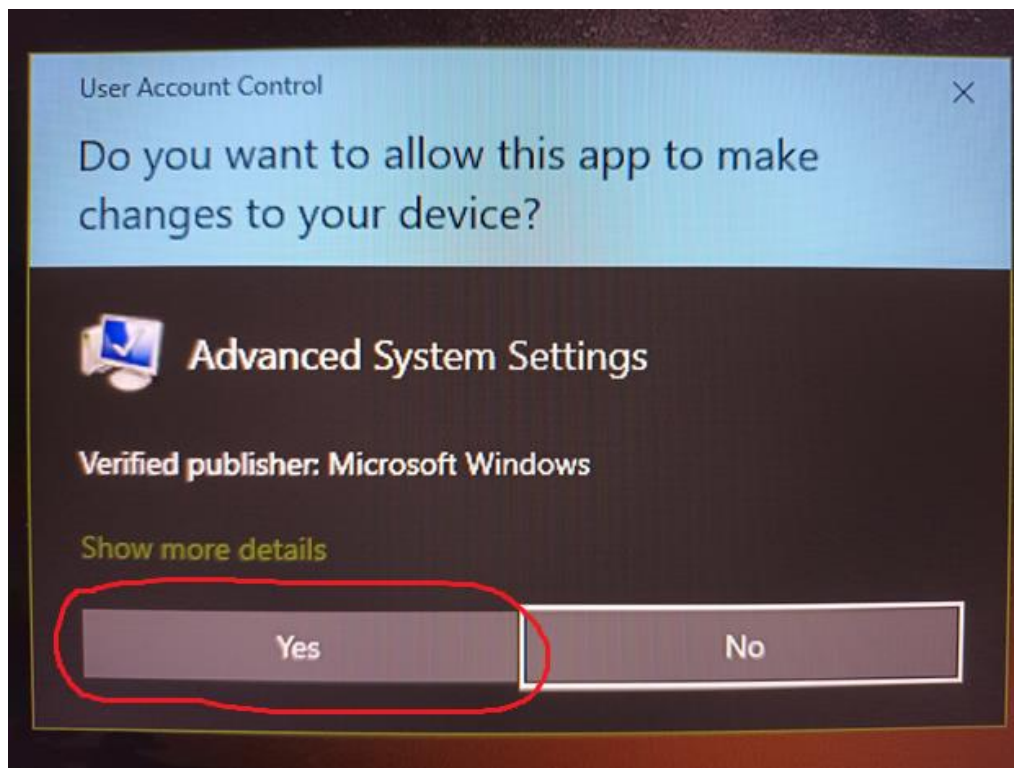
## SETTING ENVIRONMENT VARIABLES

- Press Win + R, in the text box type *control panel*:

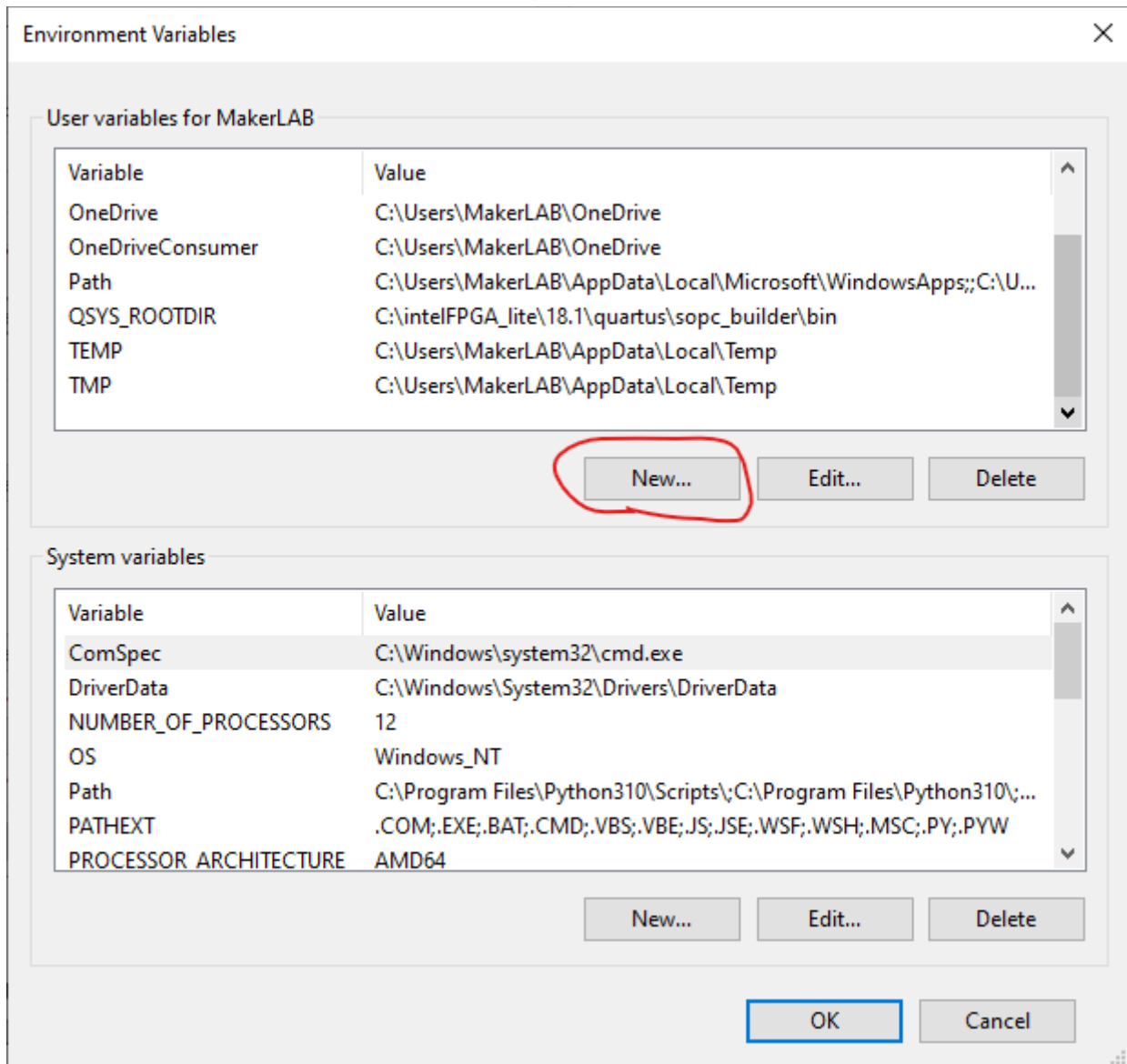


- Type *env* in the search box, and choose *Edit the system environment variables*. Click *Yes* if a prompt screen appears. The *System Properties* window will open after that, click *Environment variables...* at the bottom.





- You can see two regions, *User variables for {your computer username}*, and *System variables*. On *User variables* field, click *New...*, then type **MSYS2\_PATH\_TYPE** for *Variable name*, and **inherit** for *Variable value*. Click *OK* when done.



New User Variable

Variable name: MSYS2\_PATH\_TYPE

Variable value: inherit

Browse Directory... Browse File... OK Cancel

Environment Variables

User variables for MakerLAB

Variable	Value
HPPAuseHwCtrls	false
MSYS2_PATH_TYPE	inherit
OneDrive	C:\Users\MakerLAB\OneDrive
OneDriveConsumer	C:\Users\MakerLAB\OneDrive
Path	C:\Users\MakerLAB\AppData\Local\Microsoft\WindowsApps;C:\U...
QSYS_ROOTDIR	C:\intelFPGA_lite\18.1\quartus\sopc_builder\bin
TEMP	C:\Users\MakerLAB\AppData\Local\Temp

New... Edit... Delete

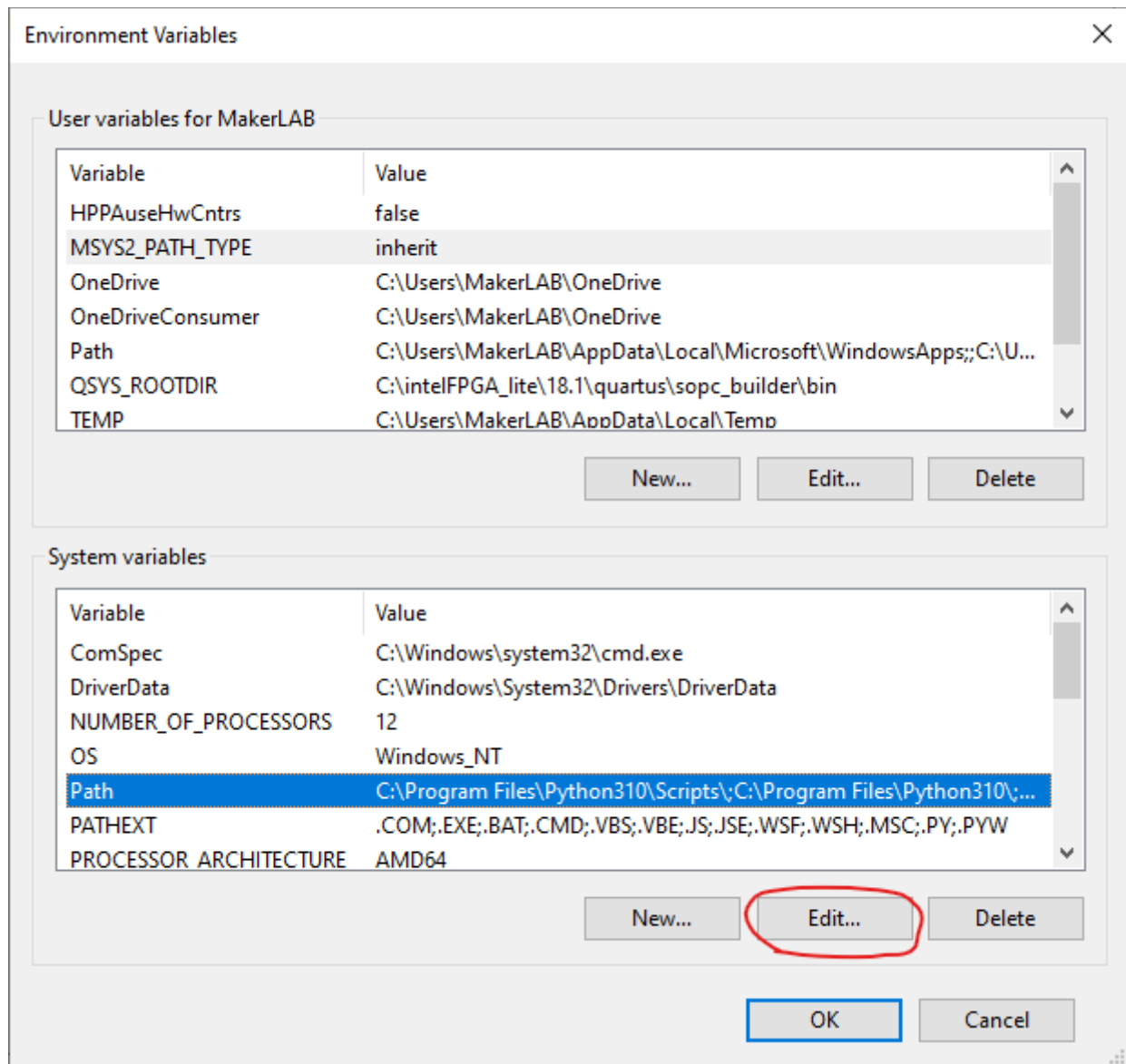
System variables

Variable	Value
ComSpec	C:\Windows\system32\cmd.exe
DriverData	C:\Windows\System32\Drivers\DriverData
NUMBER_OF_PROCESSORS	12
OS	Windows_NT
Path	C:\Program Files\Python310\Scripts\;C:\Program Files\Python310\...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC;.PY;.PYW
PROCESSOR_ARCHITECTURE	AMD64

New... Edit... Delete

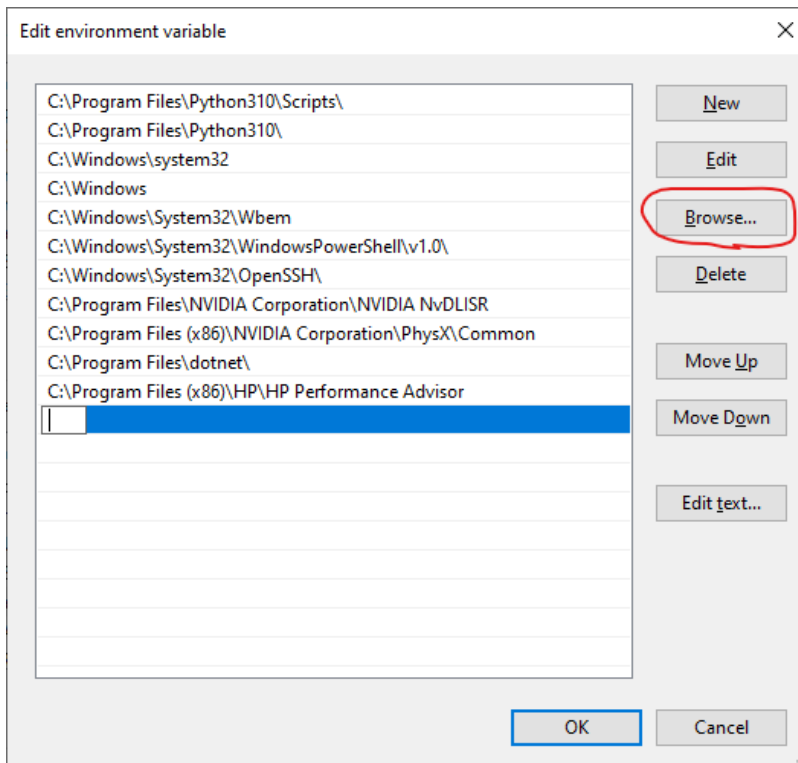
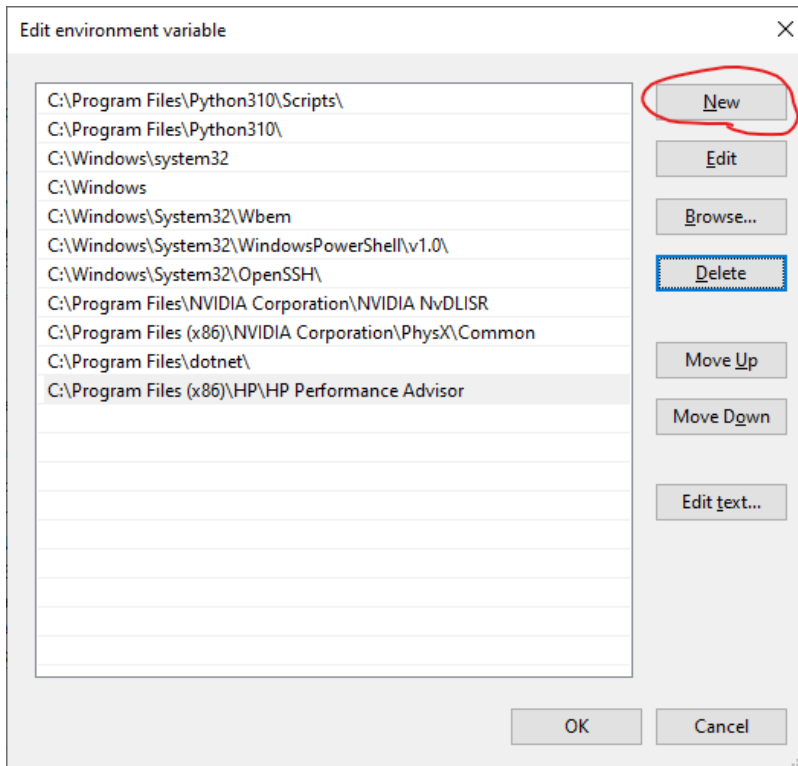
OK Cancel

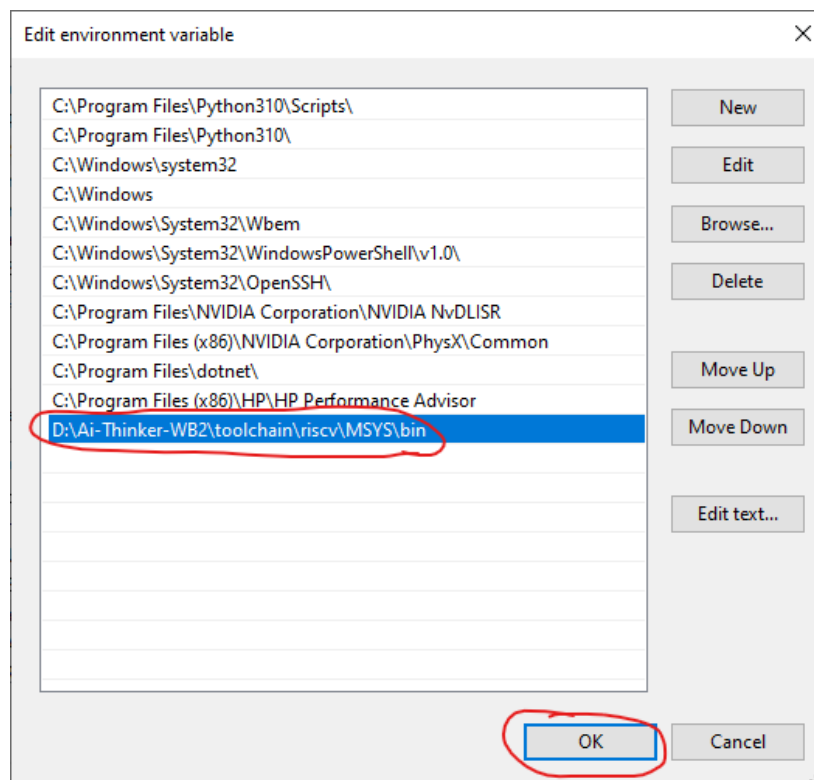
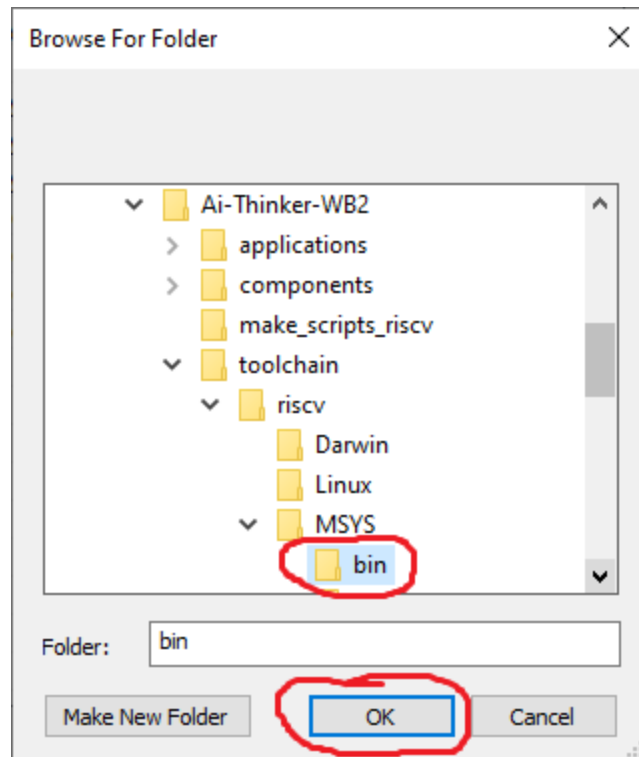
- Next, on *System variables*, choose *Path*, and click *Edit...*:





- Click *New*, then *Browse...*, and select the `\toolchain\riscv\MSYS\bin` folder from the downloaded SDK. Then click *OK* to save the directory, and *OK* again to go back to the environment variables window.

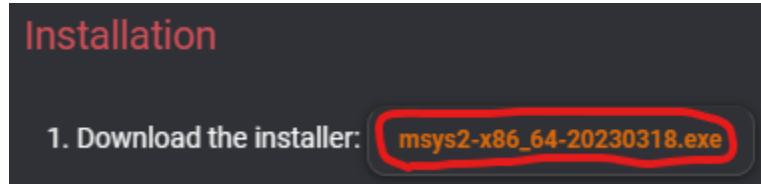




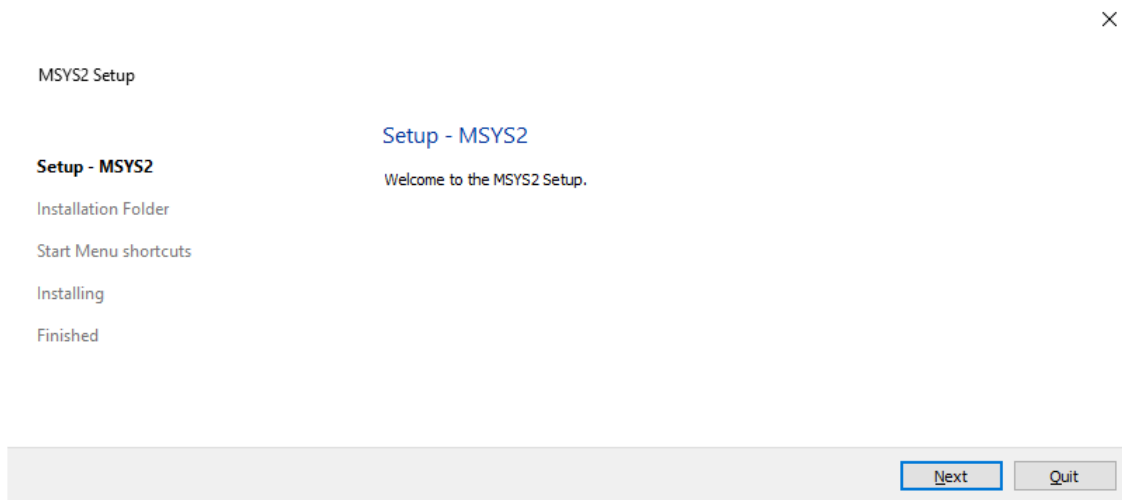
- Double check the newly added user variable and system path. When done, click *OK* two more times to save all changes and close the *Environment Variables* and *System Properties* window.

## INSTALLING MSYS2 AND MAKE

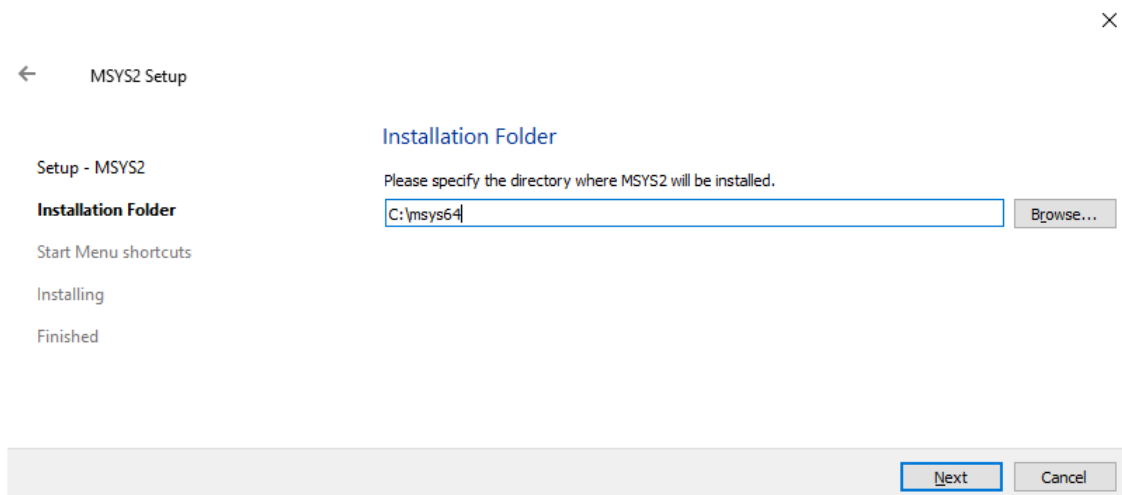
- Go to <https://msys2.org>. Scroll down and click the download link under *Installation* section.



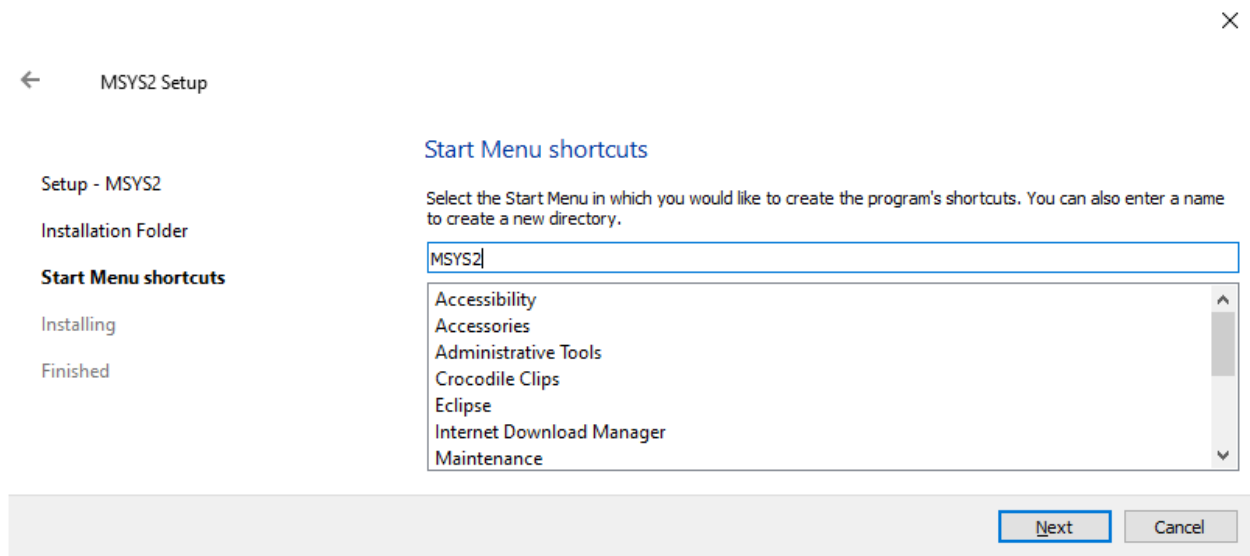
- Wait for the download to finish, then run the installation.



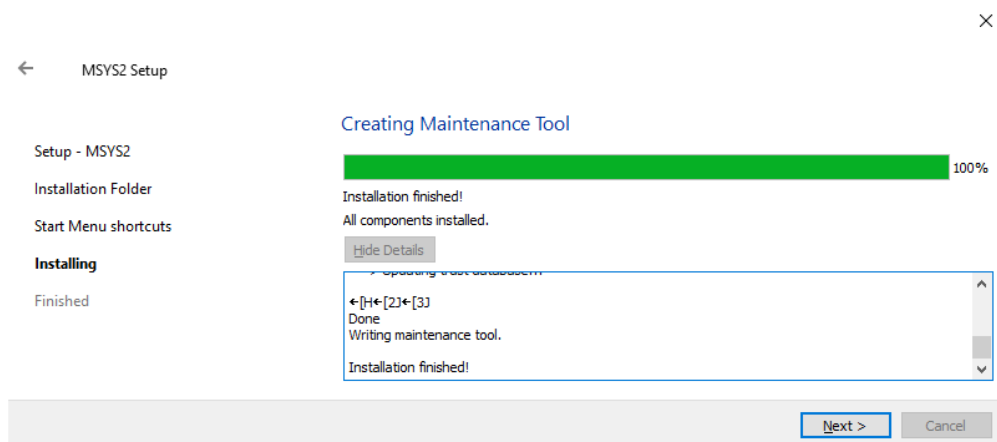
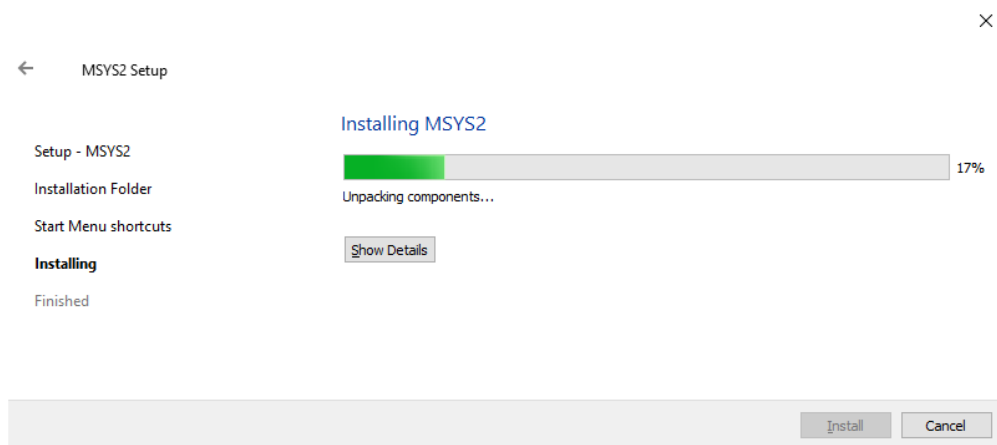
- Click *Next*. Specify where MSYS2 will be installed, default is *C:\msys64*.



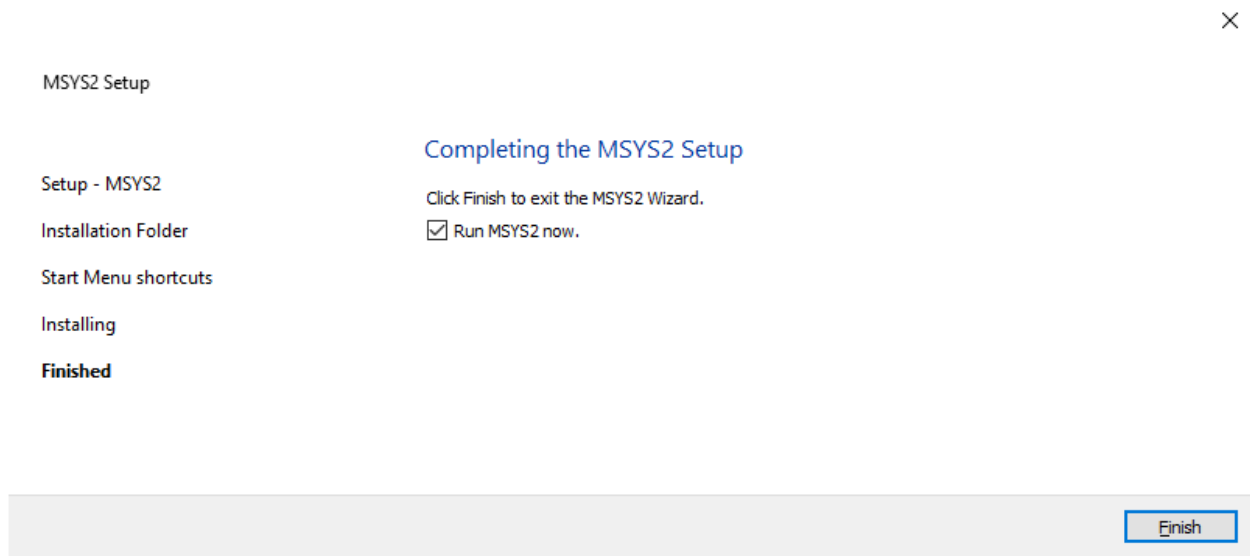
- *Next.* Leave this part unchanged.



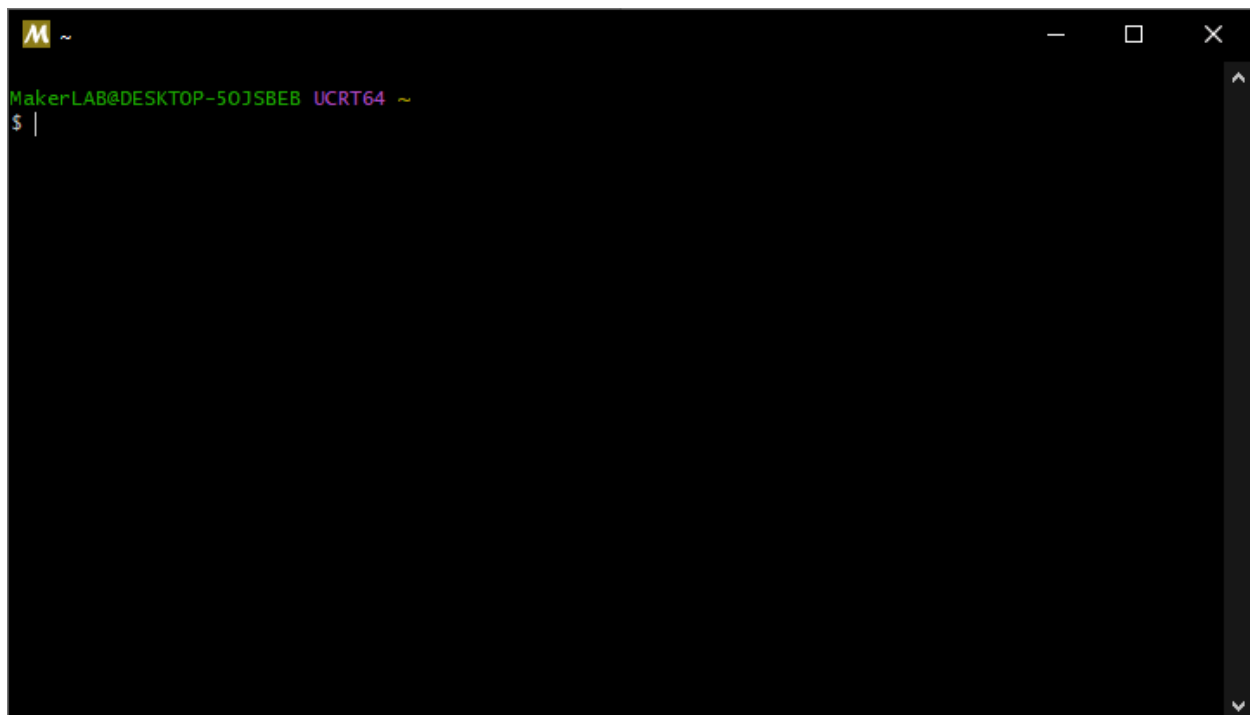
- *Next.* Wait for installation to complete.



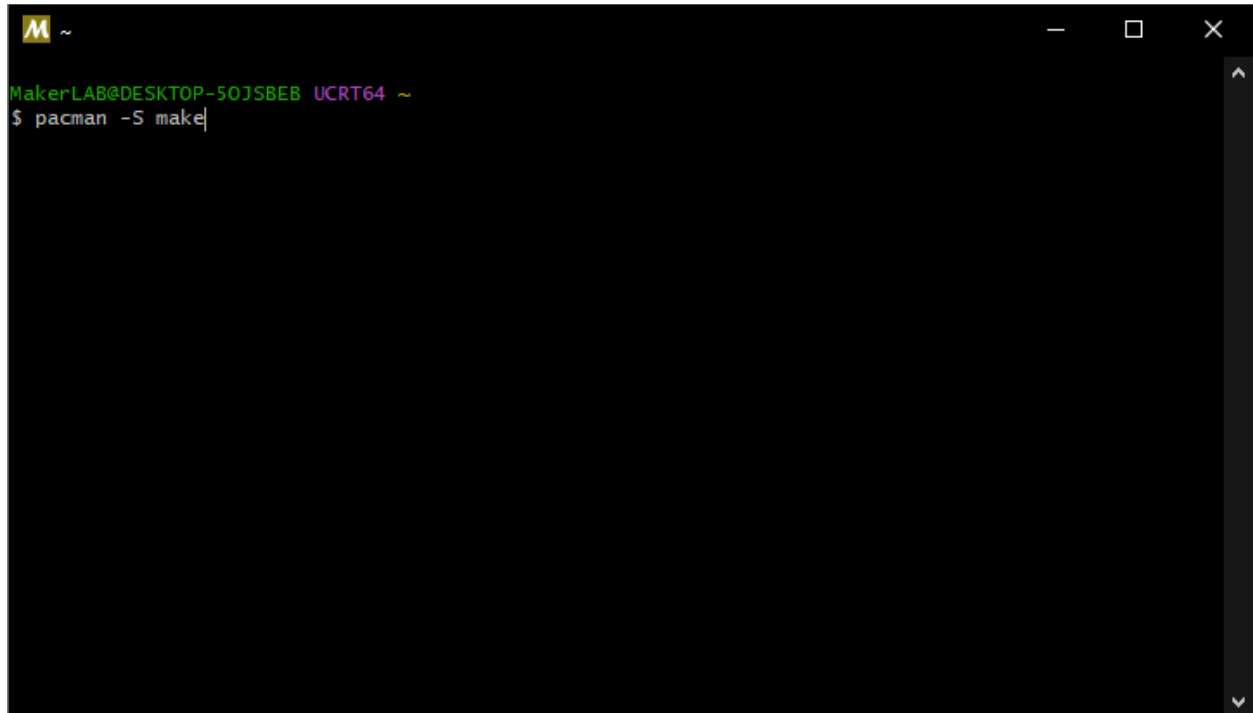
- *Next.* Leave the checkbox checked, and click *Finish* to launch MSYS2.



- A black terminal window will appear:

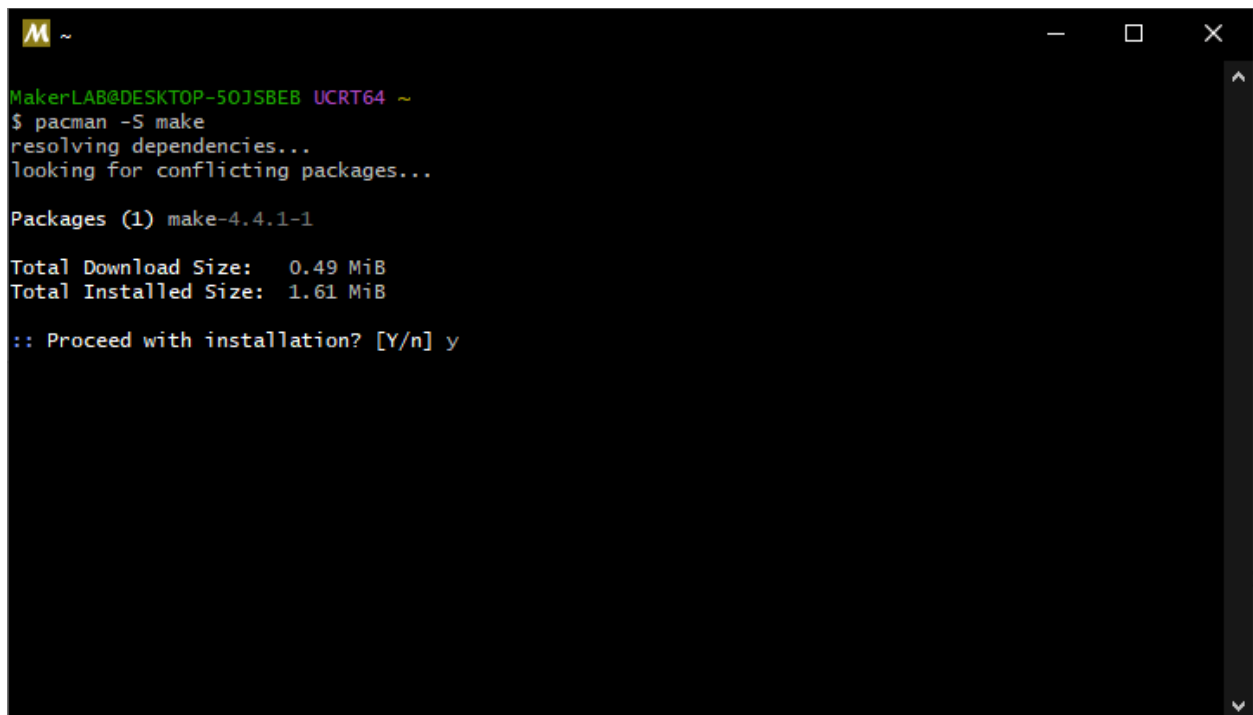


- Inside the terminal, type **pacman -S make**:



```
M ~
MakerLAB@DESKTOP-50JSBEB UCRT64 ~
$ pacman -S make
```

- Press Enter, then type **y** and Enter again when prompted:



```
M ~
MakerLAB@DESKTOP-50JSBEB UCRT64 ~
$ pacman -S make
resolving dependencies...
looking for conflicting packages...

Packages (1) make-4.4.1-1

Total Download Size: 0.49 MiB
Total Installed Size: 1.61 MiB

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

```
M ~
MakerLAB@DESKTOP-5OJSBEB UCRT64 ~
$ pacman -S make
resolving dependencies...
looking for conflicting packages...

Packages (1) make-4.4.1-1

Total Download Size: 0.49 MiB
Total Installed Size: 1.61 MiB

:: Proceed with installation? [Y/n] y
:: Retrieving packages...
make-4.4.1-1-x86_64 505.9 KiB 193 KiB/s 00:03 [#####] 100%
(1/1) checking keys in keyring [#####] 100%
(1/1) checking package integrity [#####] 100%
(1/1) loading package files [#####] 100%
(1/1) checking for file conflicts [#####] 100%
(1/1) checking available disk space [#####] 100%
:: Processing package changes...
(1/1) installing make [#####] 100%
:: Running post-transaction hooks...
(1/1) Updating the info directory file...

MakerLAB@DESKTOP-5OJSBEB UCRT64 ~
$
```

- You have successfully installed *make*. You may close this window now.

## INSTALLING TERMITE

There are many programs that can transmit and receive UART signals. In this tutorial, we will take a look at Termiter, a Serial terminal by CompuPhase.

- Go to [https://compuphase.com/software\\_termite.htm](https://compuphase.com/software_termite.htm), then scroll down and click on the download link under *Downloads & license*.

### Downloads & license

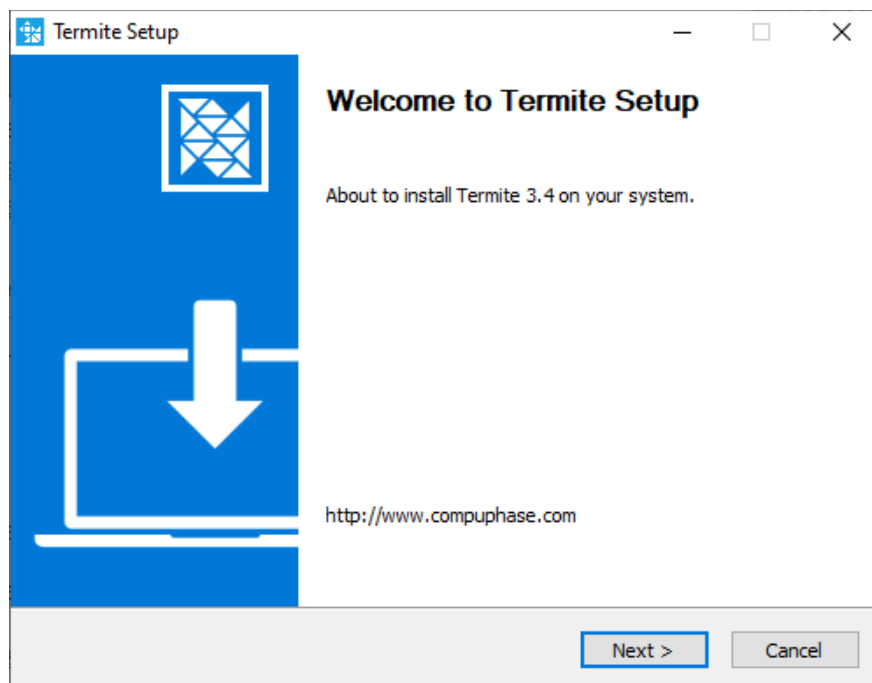
Termiter 3.4 is copyrighted software that is free for personal and commercial use. You may use it and distribute it without limitations. You may however not remove or conceal the copyright. There are no guarantees or warranties whatsoever; use it at your own risk.

The example plug-in filters are distributed under the [zlib/libpng \(open source\) license](#).

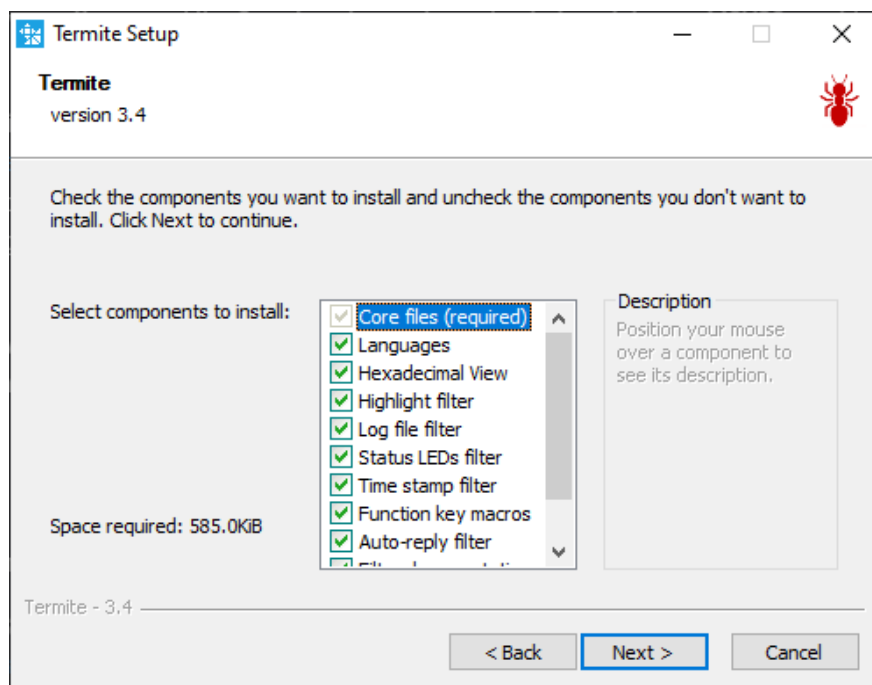
#### Termiter version 3.4 - complete setup (332 KiB)

A self-extracting setup that contains Termiter and all plug-ins and documentation (but excluding the source code for these plug-ins). This setup is provided to conveniently and quickly install Termiter. (Note that Termiter does not *require* installation, but it may just be *convenient* to do so.)

- Wait for the download to finish, then run the installation.

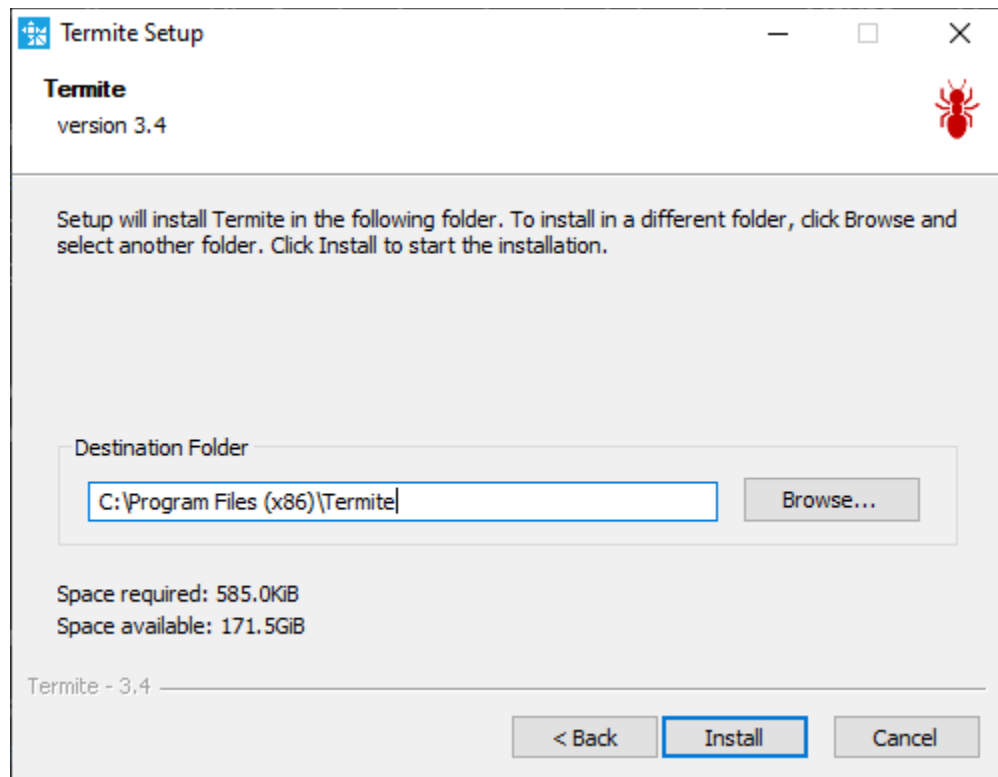


- *Next*, shows the list of features that will be installed. Check/uncheck the features you want/don't want.

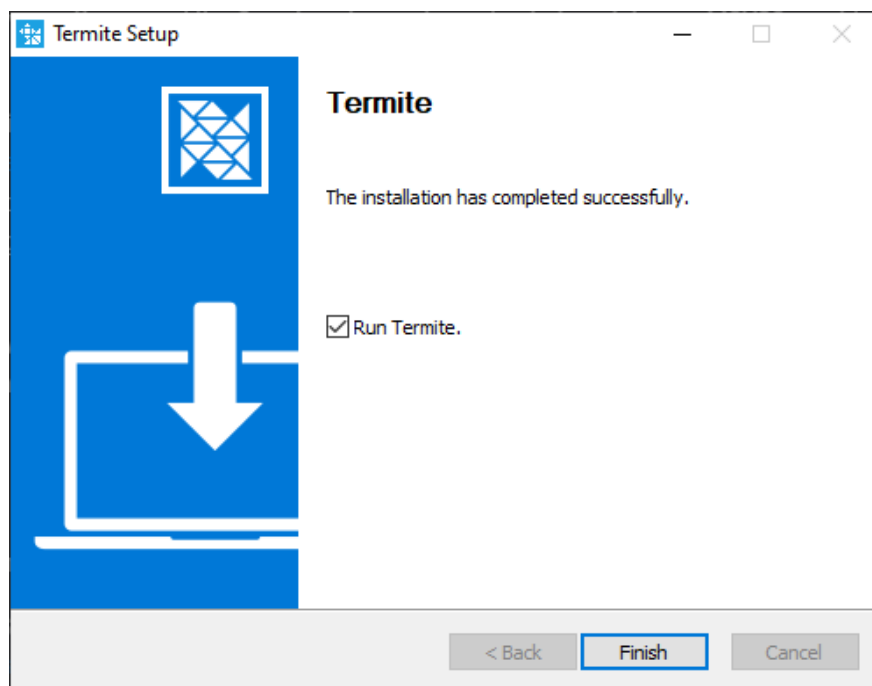




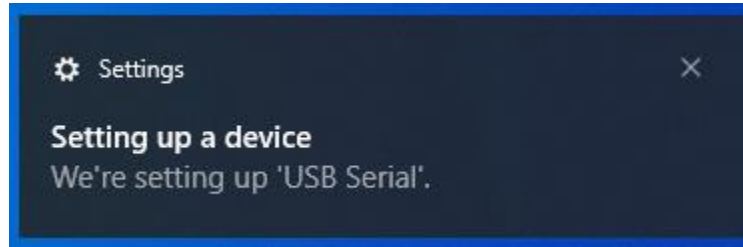
- Next, choose an install location, default is *C:\Program Files (x86)\Termite*.



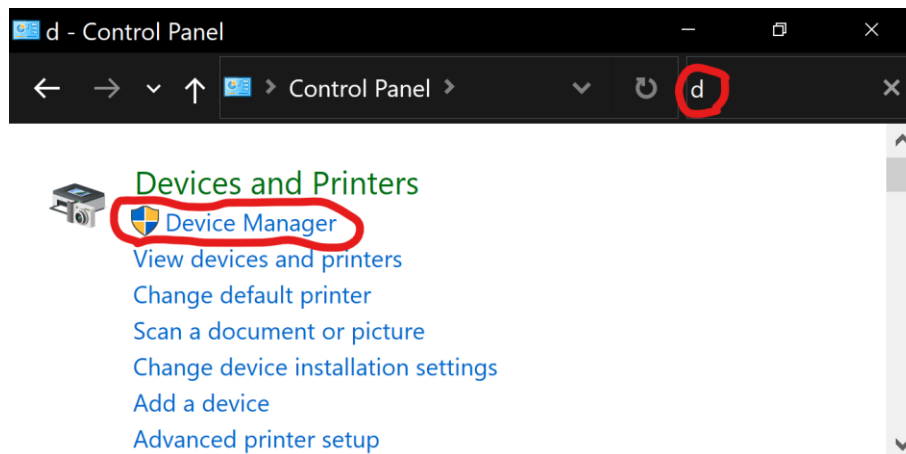
- Click *Install*, and wait for installation to complete.



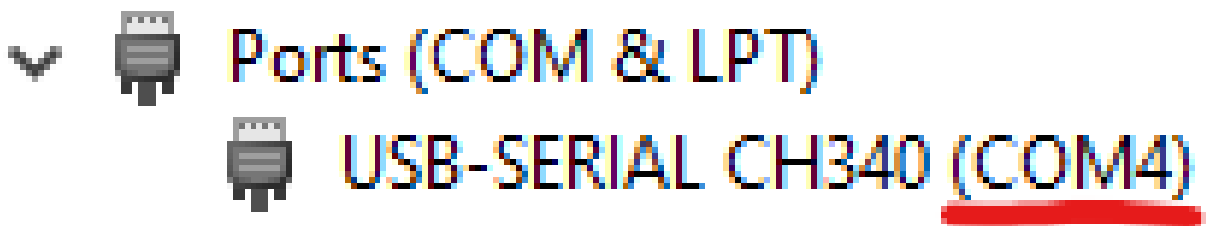
- Leave *Run Terminate* checkbox checked, but **do not click *Finish* yet**. Instead, **plug in your Ai-WB2-12F module now**. If this is the first time you connect the module to your computer, you will get this notification:



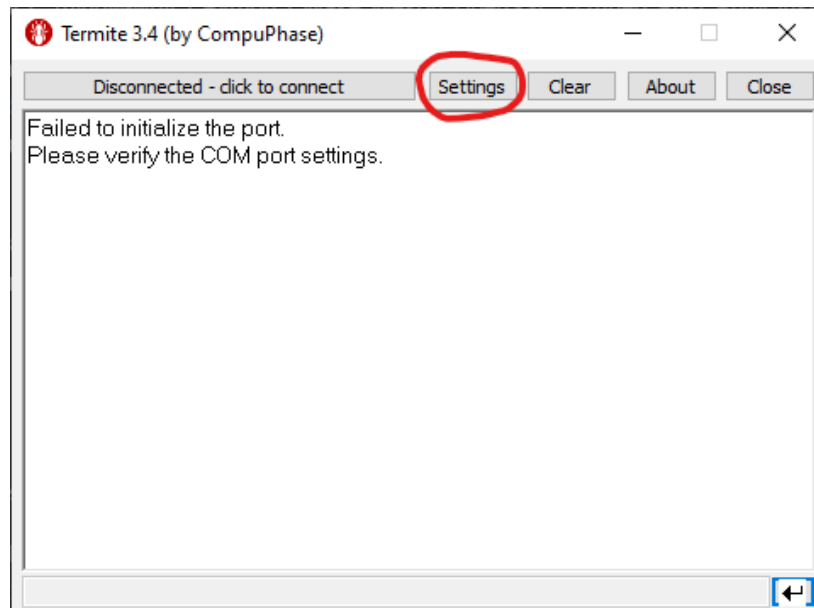
- Wait for about a minute, then go to *Control Panel* again, find and open *Device Manager*:



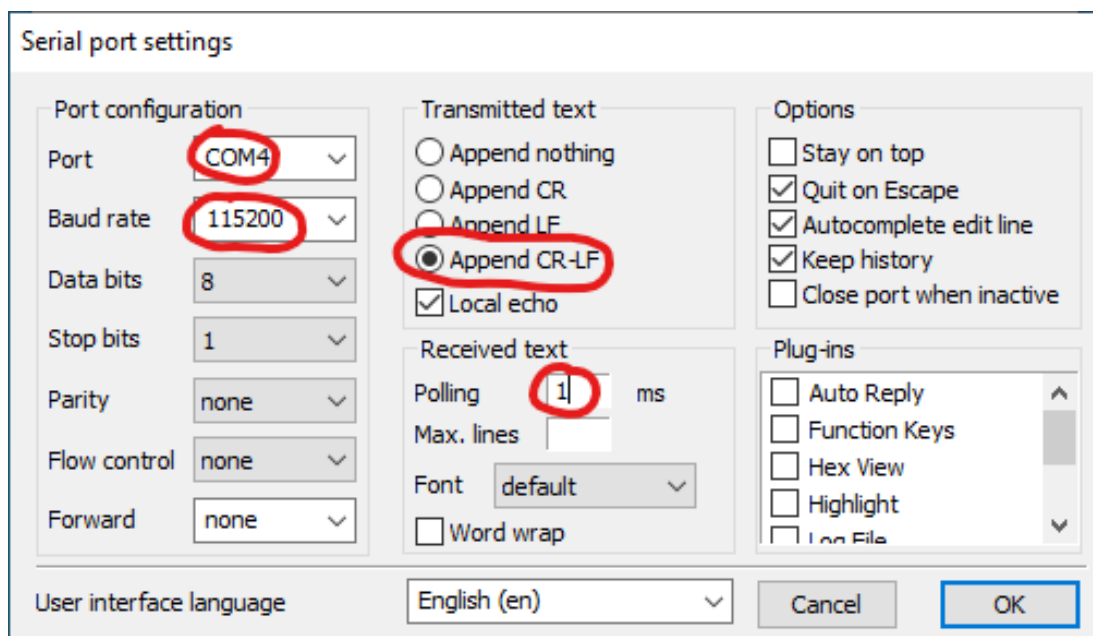
- In the list of drivers, find *Port (COM & LPT)*, expand it, you will see *USB-SERIAL CH340 (COMx)*, where *x* is a number assigned by your computer. Note down this **COMx**, which will be used quite often later.



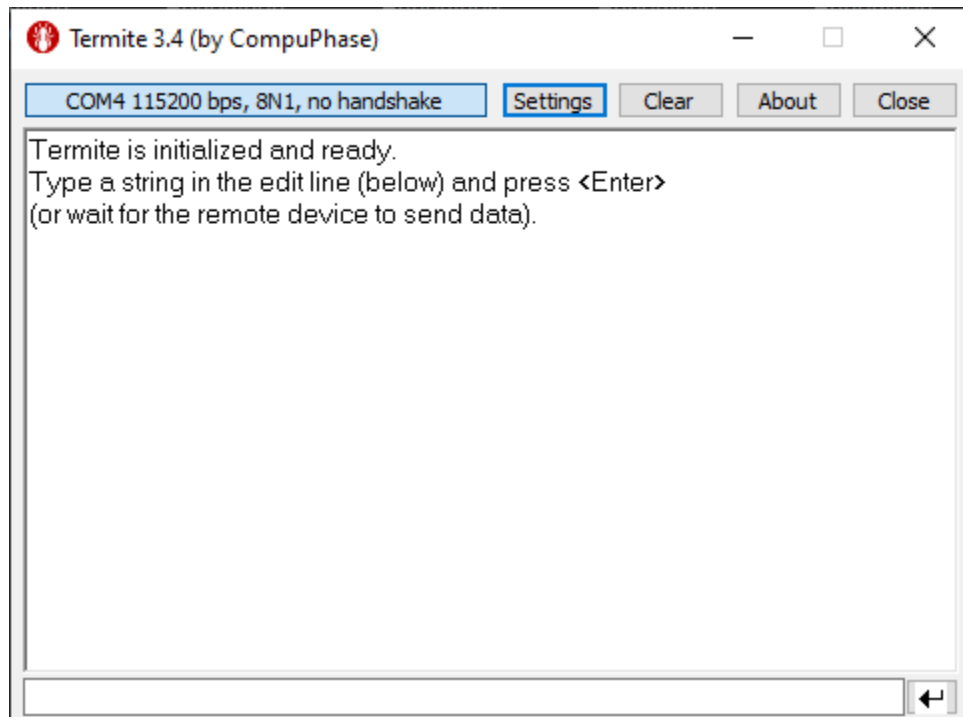
- You can now click *Finish* on the Terminate installation window. Terminate will be launched shortly.



- Click *Settings*, then change the following:
  - *Port* to the previously noted **COMx**.
  - *Baud rate* to 115200, Ai-WB2-12F's default.
  - *Transmitted text* to *Append CR-LF*, for future convenience.
  - *Polling* to 1 ms. Not required, but the faster, the better.

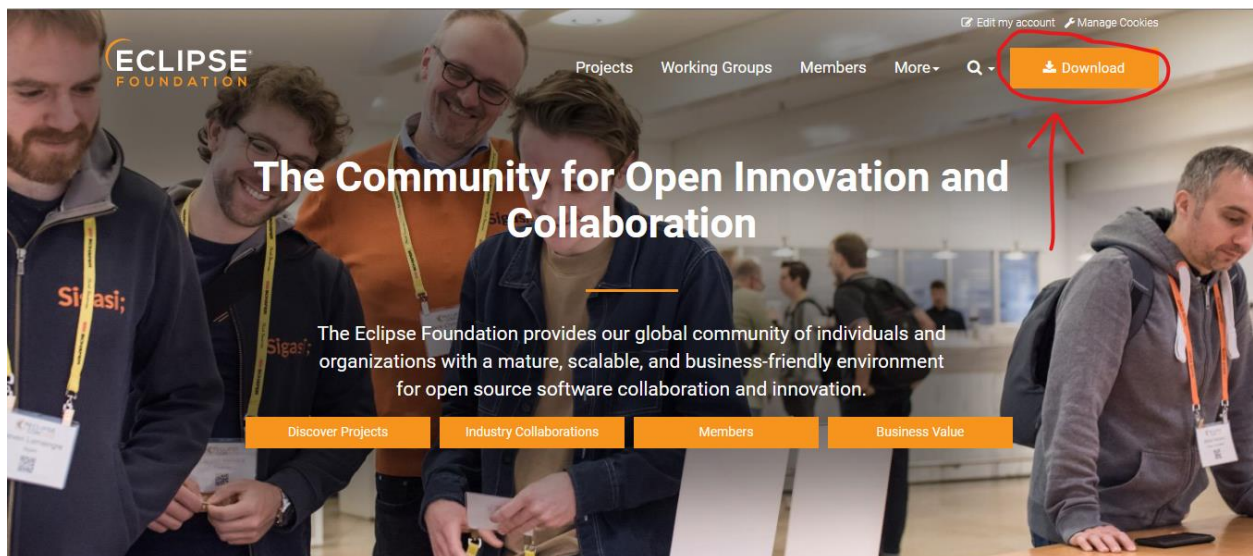


- Double check everything, and click *OK*. Terminate should be initialized and ready now, with COMx and baud rate displayed at the top left corner.




## INSTALLING ECLIPSE

- Go to <https://eclipse.org>, and download the installer.



The Eclipse Installer 2023-03 R now includes a JRE for macOS, Windows and Linux.

OpenJDK Runtimes




### Get **Eclipse IDE 2023-03**

Install your favorite desktop IDE packages.

**Download x86\_64**

[Download Packages](#) | [Need Help?](#)



### TEMURIN

by ADOPTIUM

The Eclipse Temurin™ project provides high-quality, TCK certified OpenJDK runtimes and associated technology for use across the Java™ ecosystem.

**Download Now**

[Learn More](#)

All downloads are provided under the terms and conditions of the [Eclipse Foundation Software User Agreement](#) unless otherwise specified.

**Download**

**Download from:** Japan - Japan Advanced Institute of Science and Technology ([https](https://))

**File:** [eclipse-inst-jre-win64.exe](#) [SHA-512](#)


[>> Select Another Mirror](#)


- Run the installer. Scroll down and choose *Eclipse IDE for Embedded C/C++ Developers*:


eclipseinstaller by Oomph

type filter text

Git client, XML Editor, Maven and Gradle integration

 **Eclipse IDE for Enterprise Java and Web Deve**  
Tools for developers working with Java and Web applications, including a Java IDE, tools for JavaScript, TypeScript, JavaServer...

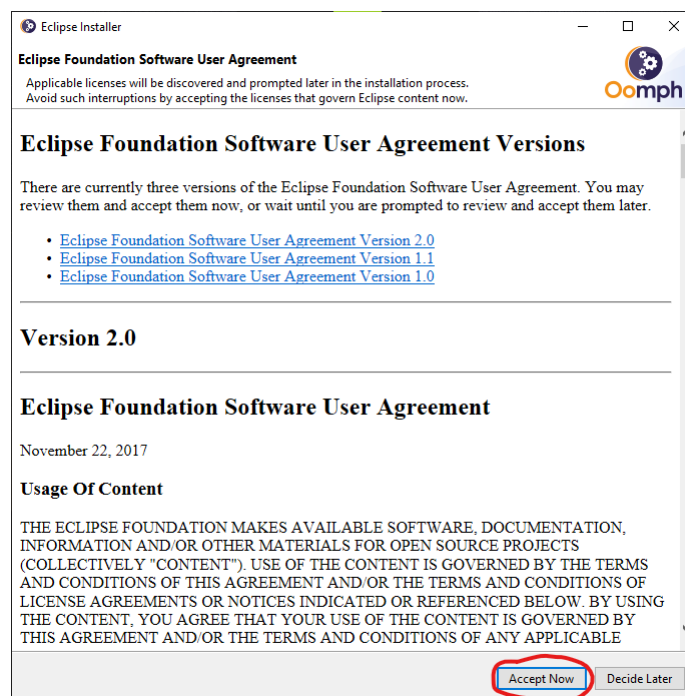
 **Eclipse IDE for C/C++ Developers**  
An IDE for C/C++ developers.

 **Eclipse IDE for Embedded C/C++ Developers**  
An IDE for Embedded C/C++ developers. It includes managed cross build plug-ins (Arm and RISC-V) and debug plug-ins...

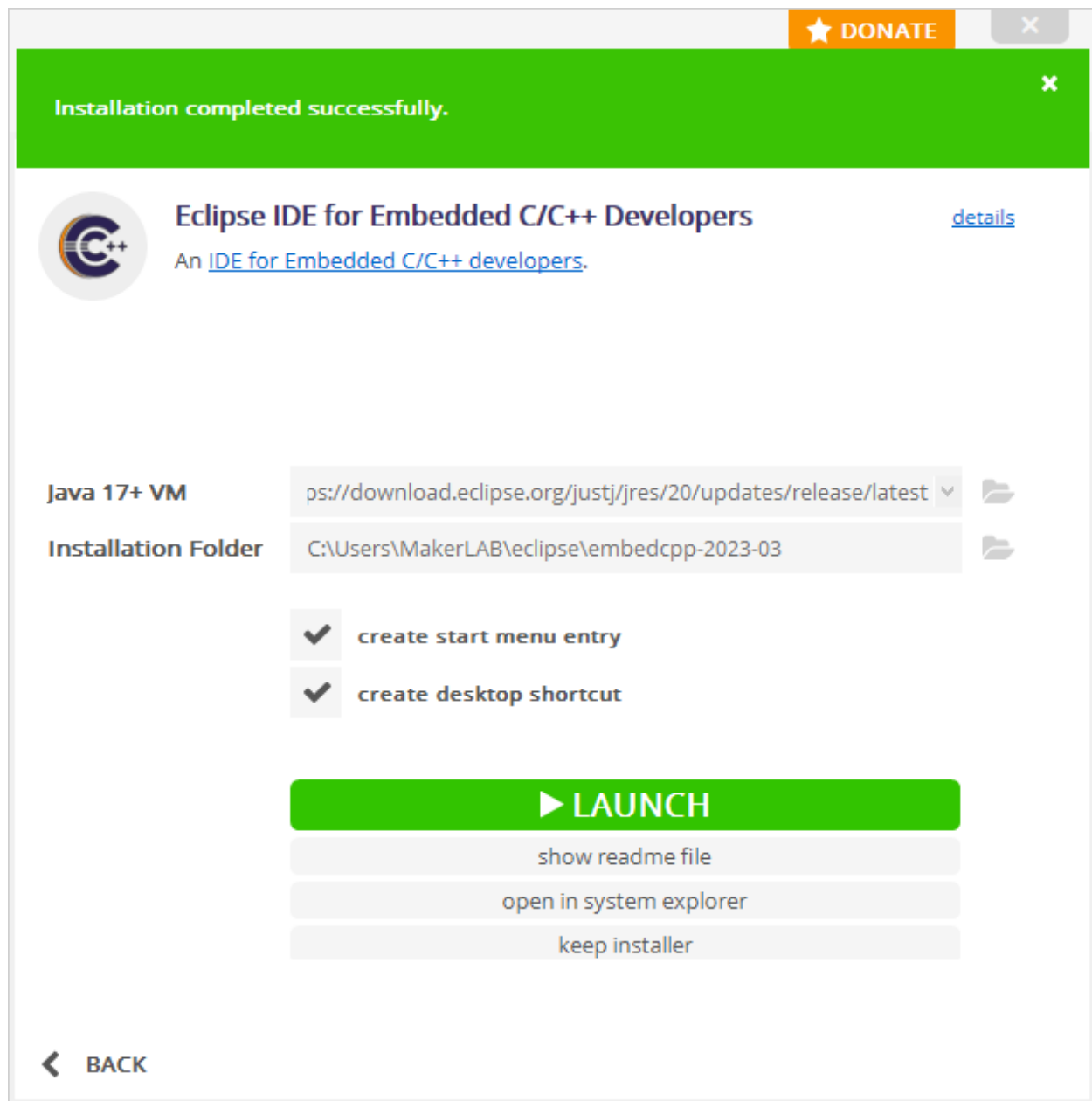
- Choose an *Installation Folder*. For *Java 17+ VM*, any version is fine. I personally prefer the latest version.



- Click *Install*, then click *Accept Now* when the *User Agreement* window appears. Wait for installation to finish.

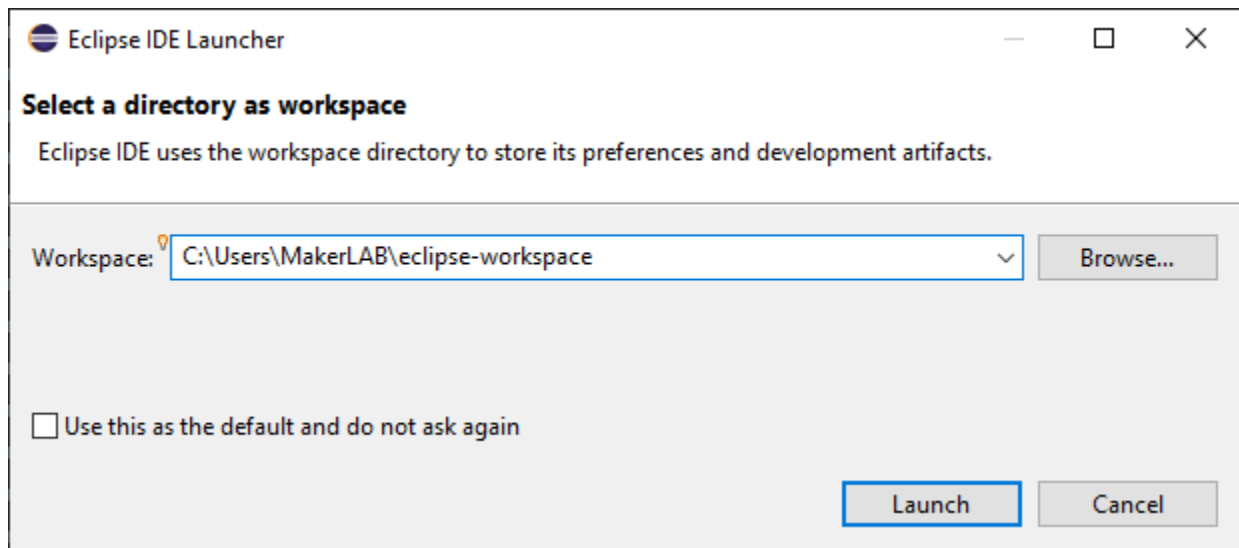


- Installation complete. Click *Launch* to run Eclipse right away. The *Launcher* window will appear.

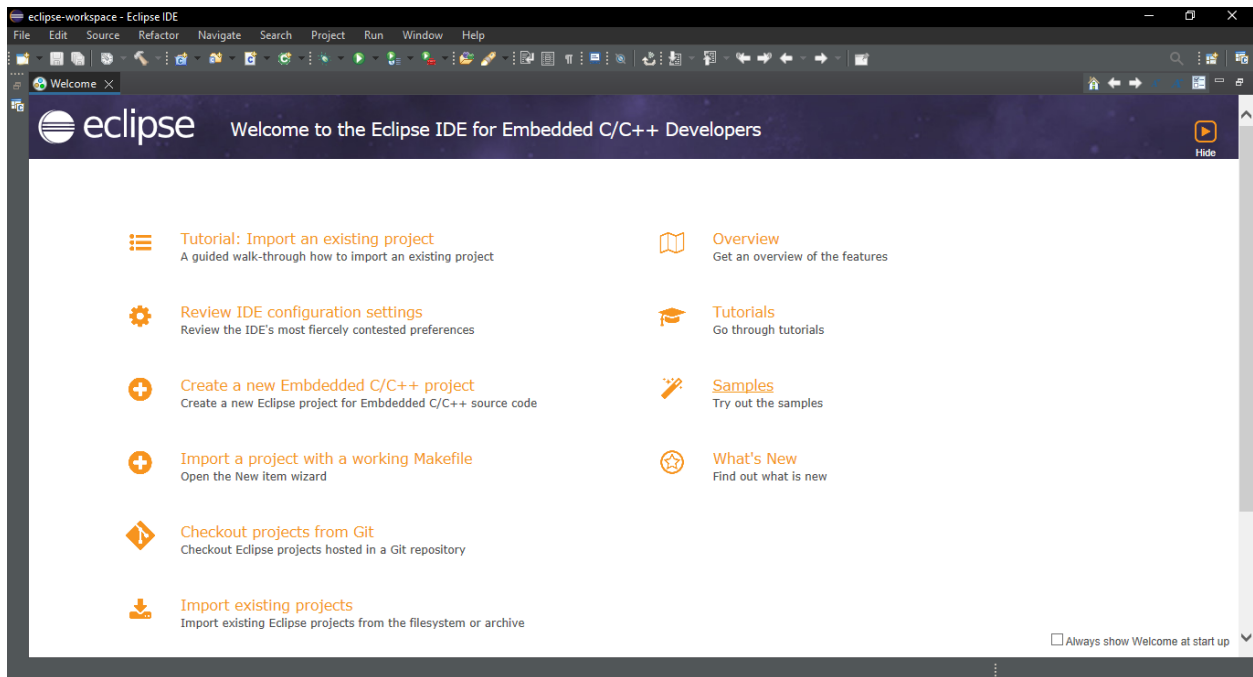




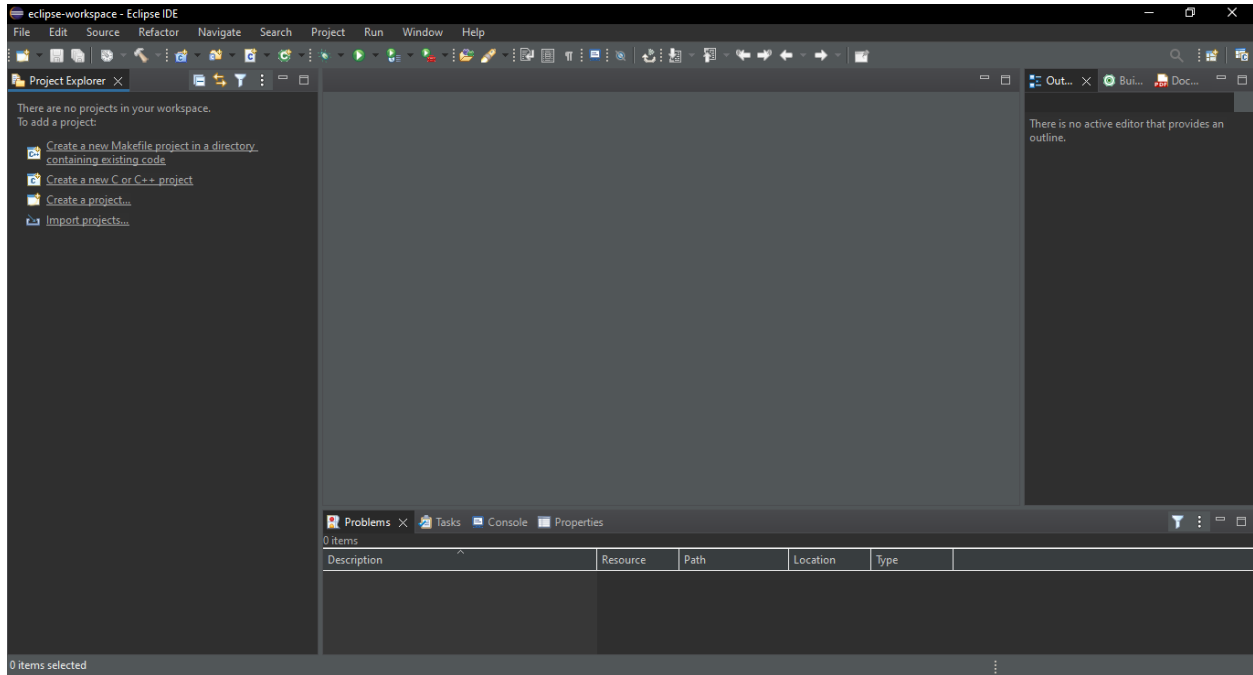
- Choose a workspace directory, default is *C:\Users\{your\_computer\_username}\eclipse-workspace*. Then click *Launch*.



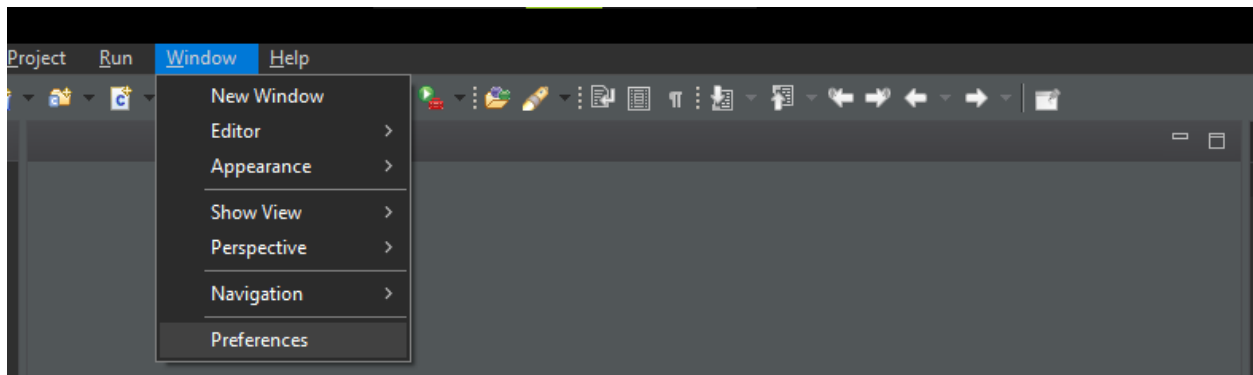




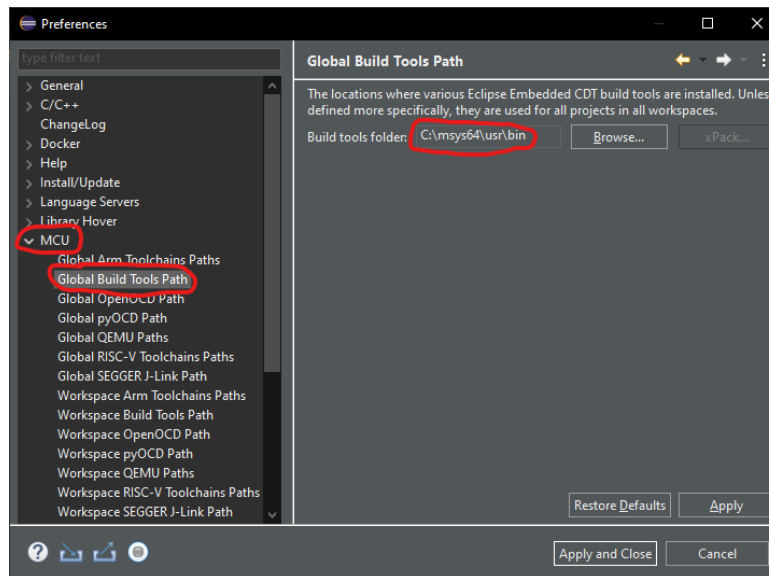
- After closing the *Welcome* screen, this will be your main work area:



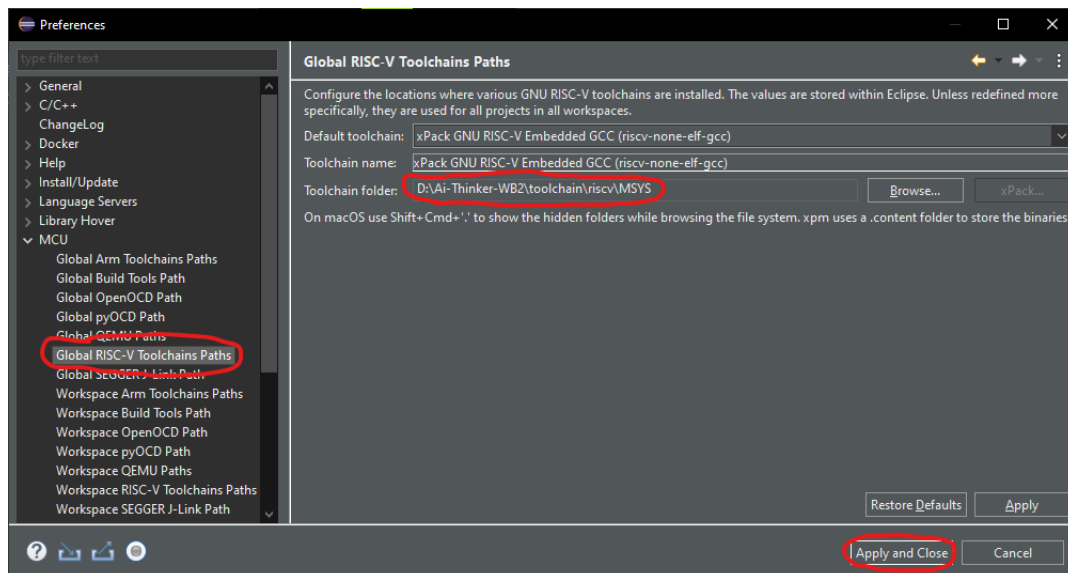
- We need to set up a few more things to get Eclipse ready. Go to *Window* -> *Preferences*:



- Under *MCU*, go to *Global Build Tools Path*. In *Build tools folder*, select `\msys64\usr\bin` from where you installed MSYS2 earlier.



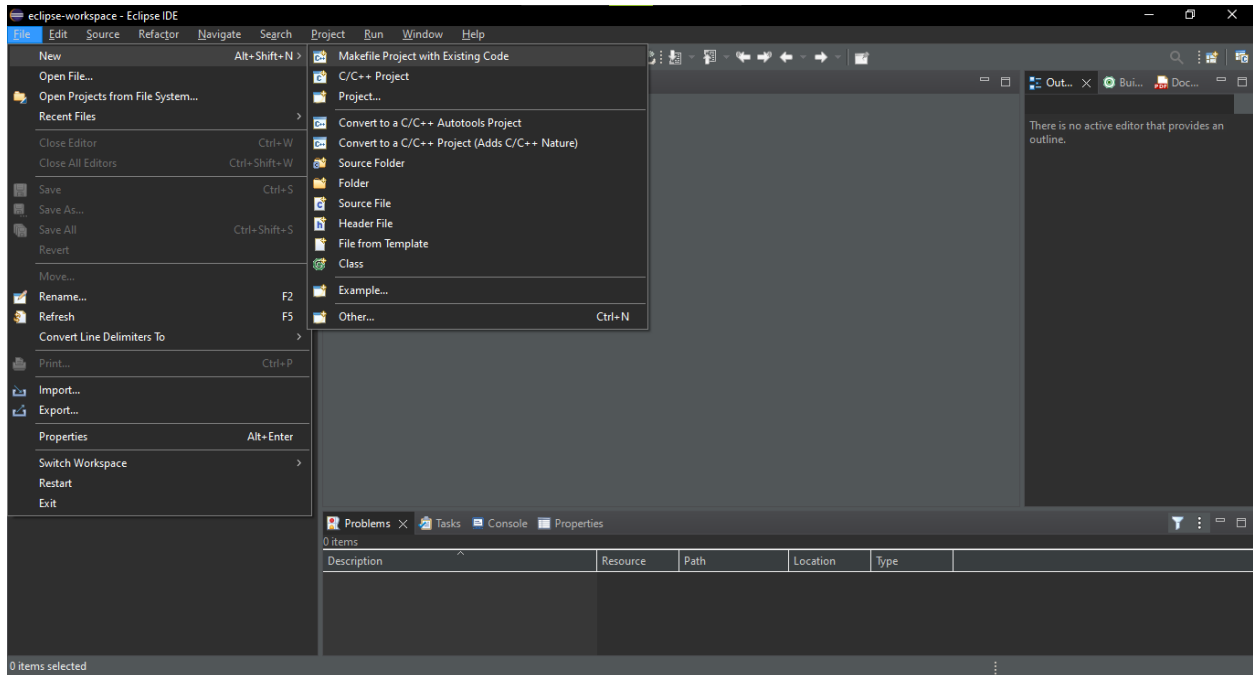
- Also under *MCU*, go to *Global RISC-V Toolchains Paths*. In *Toolchain folder*, select `{SDK_location}\toolchain\riscv\MSYS`, and click *Apply and Close*.



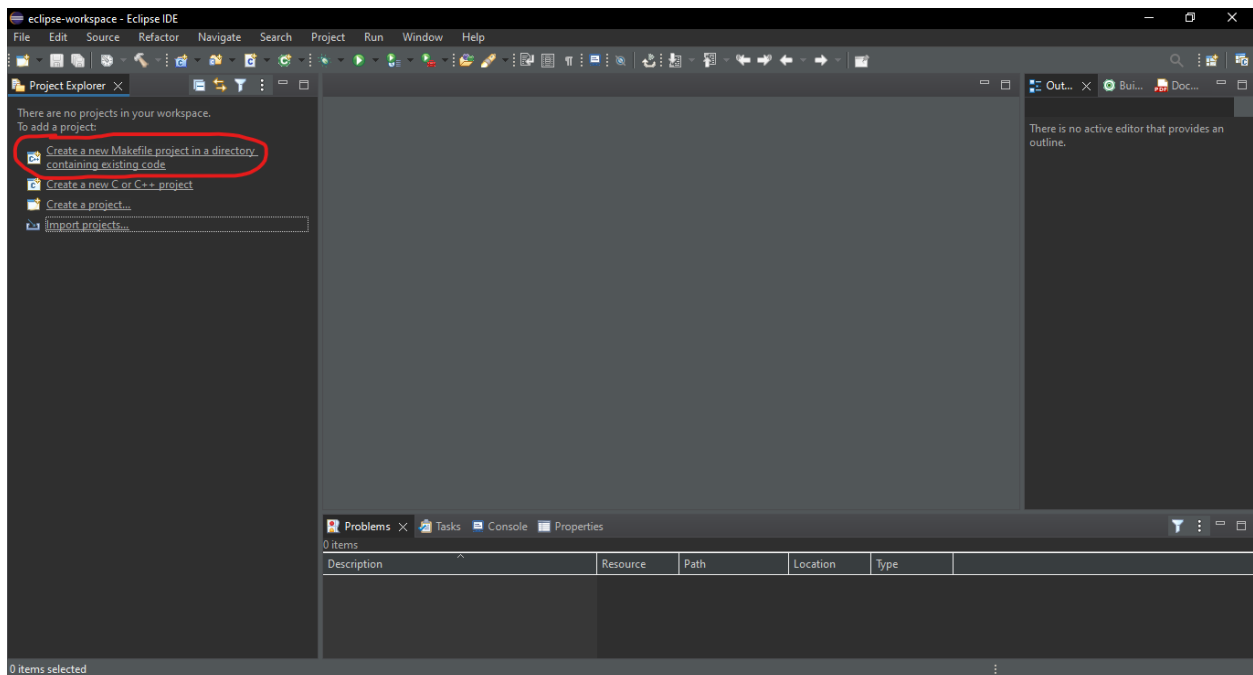
This wraps up our initial setup for Eclipse. You are now ready to create your very first project.

# YOUR FIRST PROJECT: BLINKING AN LED

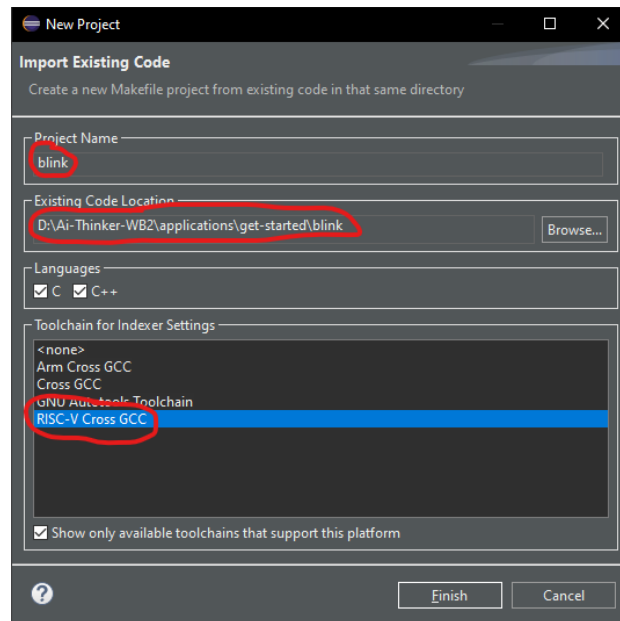
- Go to *File -> New -> Makefile Project with Existing Code:*



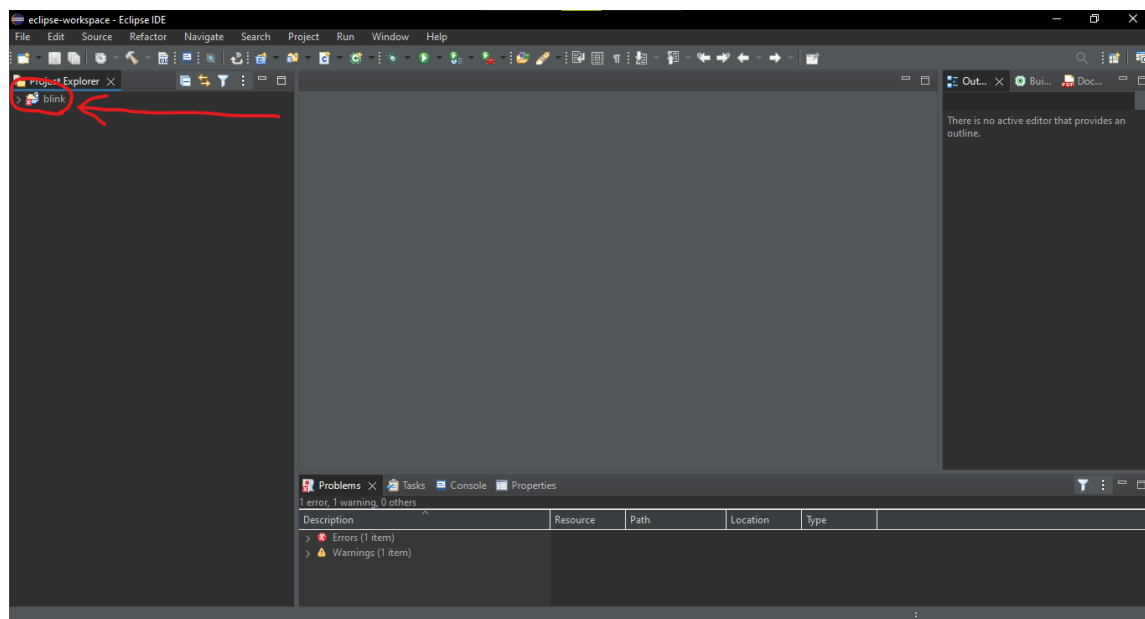
Alternatively, just click *Create a new Makefile project in a directory containing existing code* at the top left corner. Either way will open the same *New Project* window.



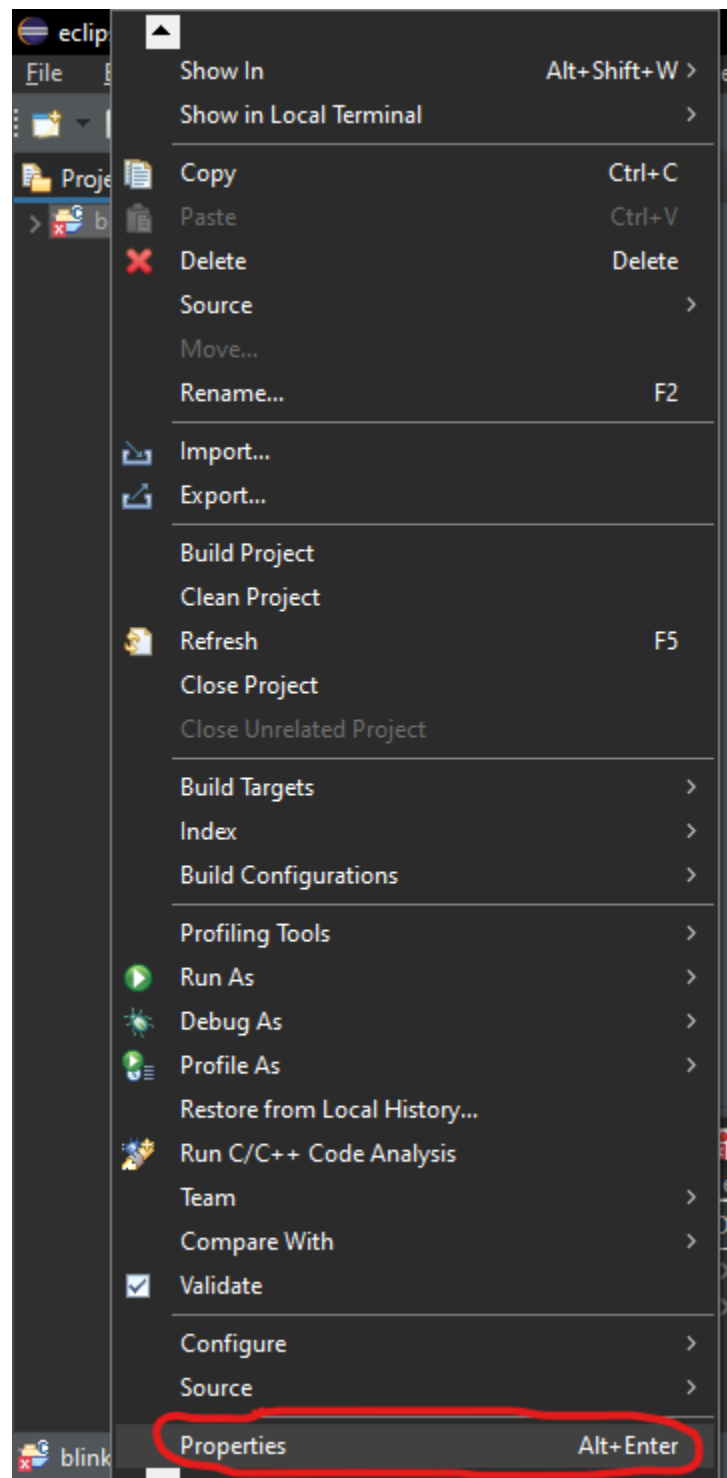
- On the *New Project* window, set *Existing Code Location* to {where you saved the SDK}\Ai-Thinker-WB2\applications\get-started\blink. Then give your project a name, and set *Toolchain for Indexer Settings* to *RISC-V Cross GCC*.



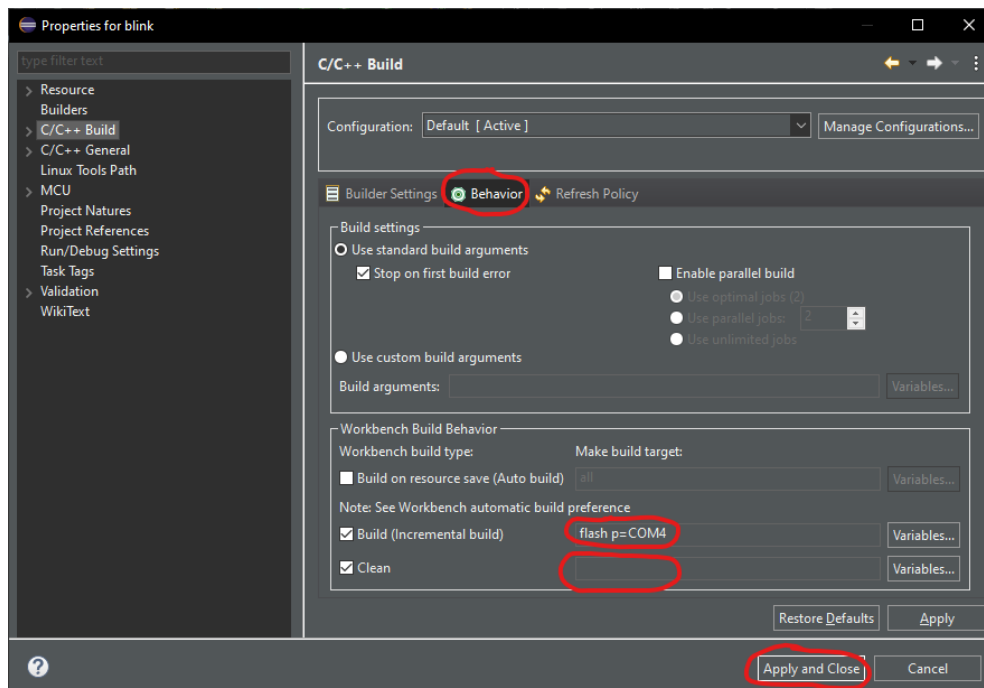
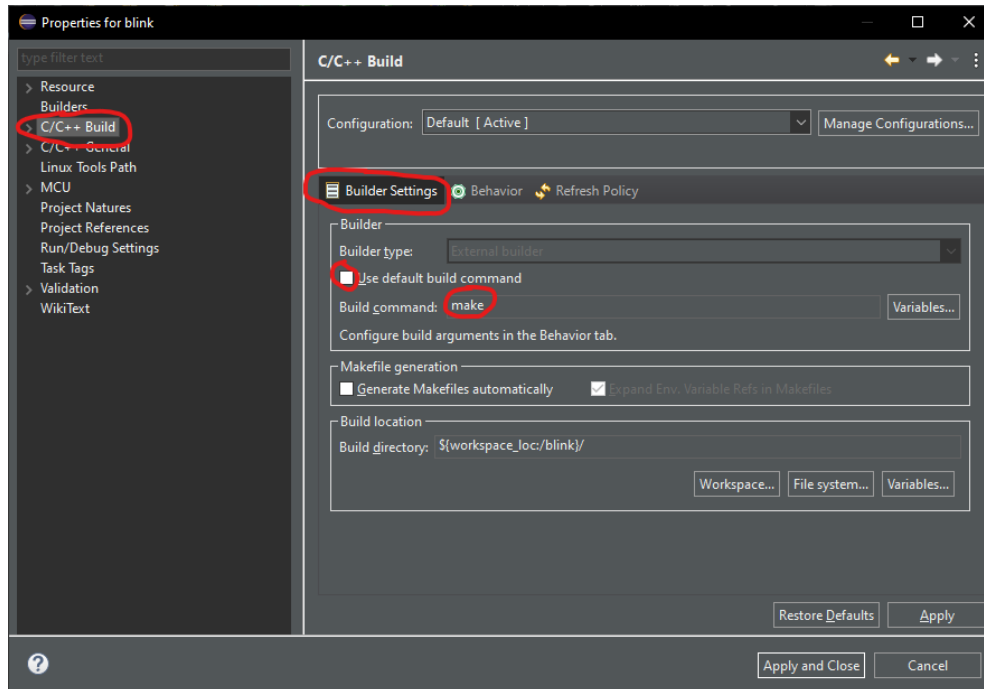
- Click *Finish*. You will see the project folder being created to the left. You may see a few errors and warnings under *Problems*, which you can safely ignore for now.



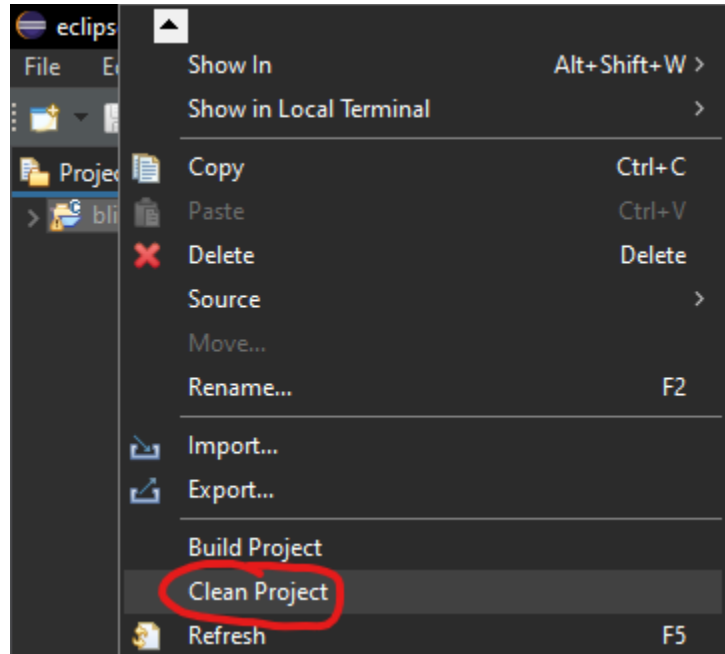
- Right click the project folder, and click *Properties* (or simply select the folder and press Alt + Enter).



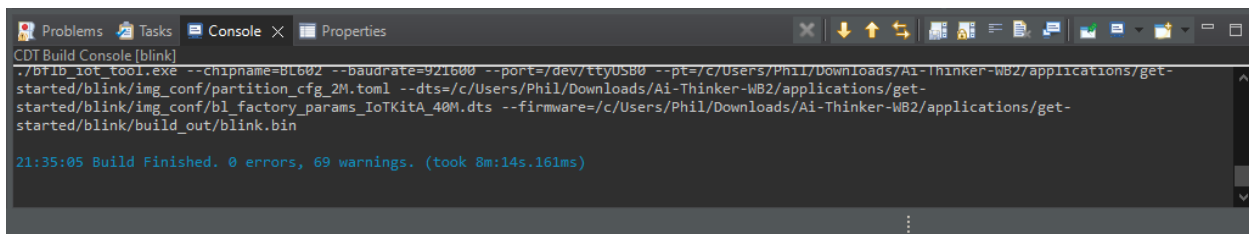
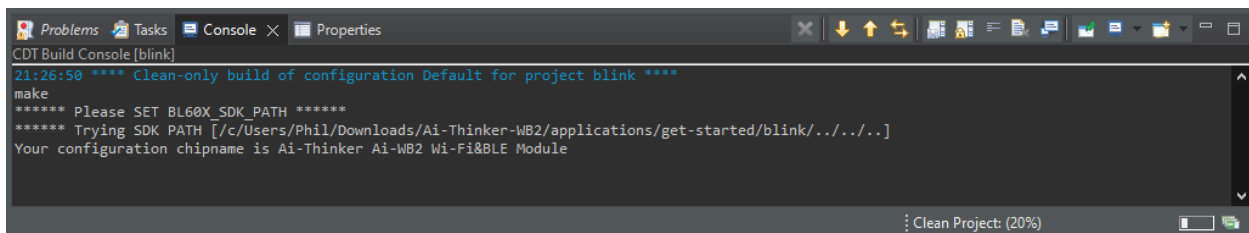
- Click on *C/C++ Build*. Under *Builder Settings*, uncheck *Use default build command*, and type **make** in *Build command*. Under *Behavior*, empty the *Clean* textbox, and change *Build (Incremental build)* to **flash p=COMx**. After clicking *Apply and Close*, *Clean Project* will compile the project, and *Build Project* will both compile the project and download it to your Ai-WB2-12F.



- Right click the project folder again, and click *Clean Project*.

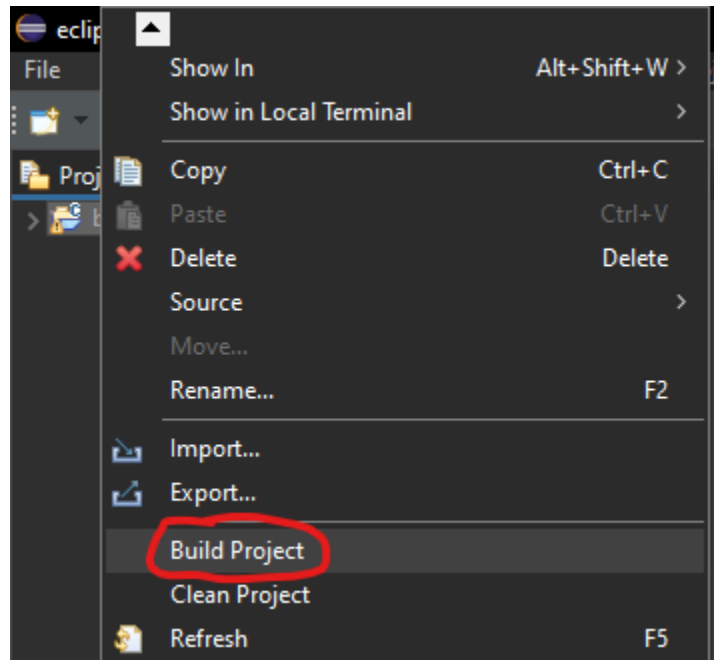


- Since this is the first time you compile the project, it will take around 6 to 8 minutes to finish. You can observe the progress in the *Console* at the bottom.

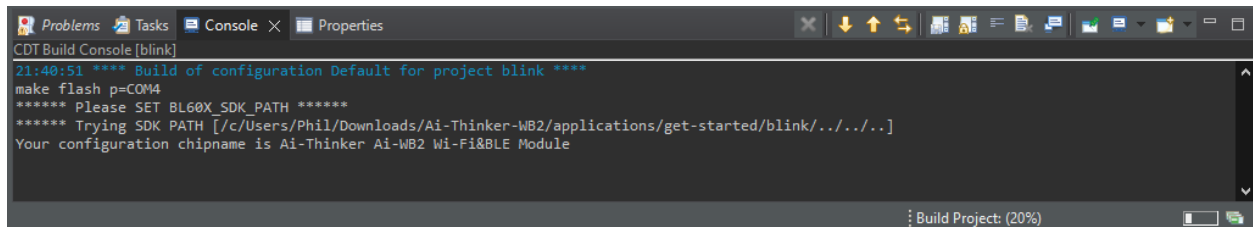




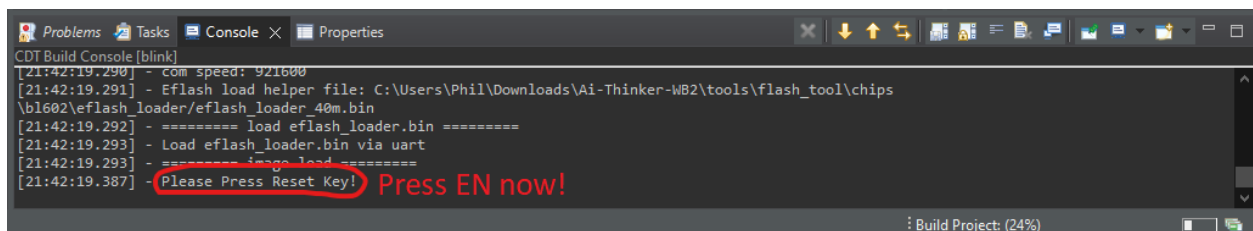
- You have successfully compiled your first project. To actually see this project in action on your Ai-WB2-12F module, right click on the project folder, and click *Build Project*.



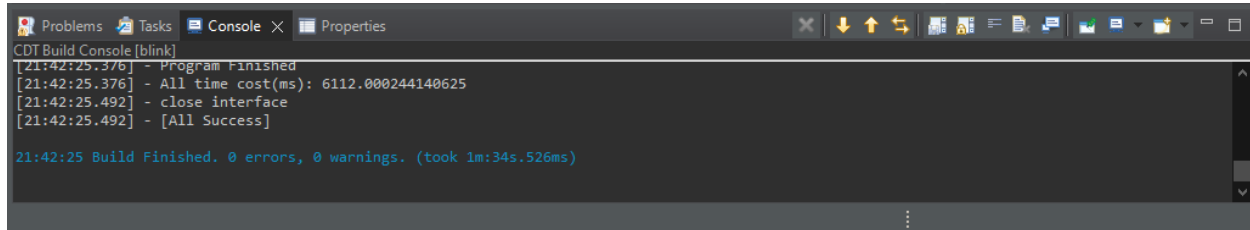
- Since you have already compiled the project once, the progress is now much faster, only taking a few seconds.



- Put your hand on the *EN* button on the module, and press it as soon as the console outputs *Please Press Reset Key!* (See picture below).



- Wait for a few more seconds. When the console output is the same as below, you have successfully downloaded the project to your Ai-WB2-12F module.



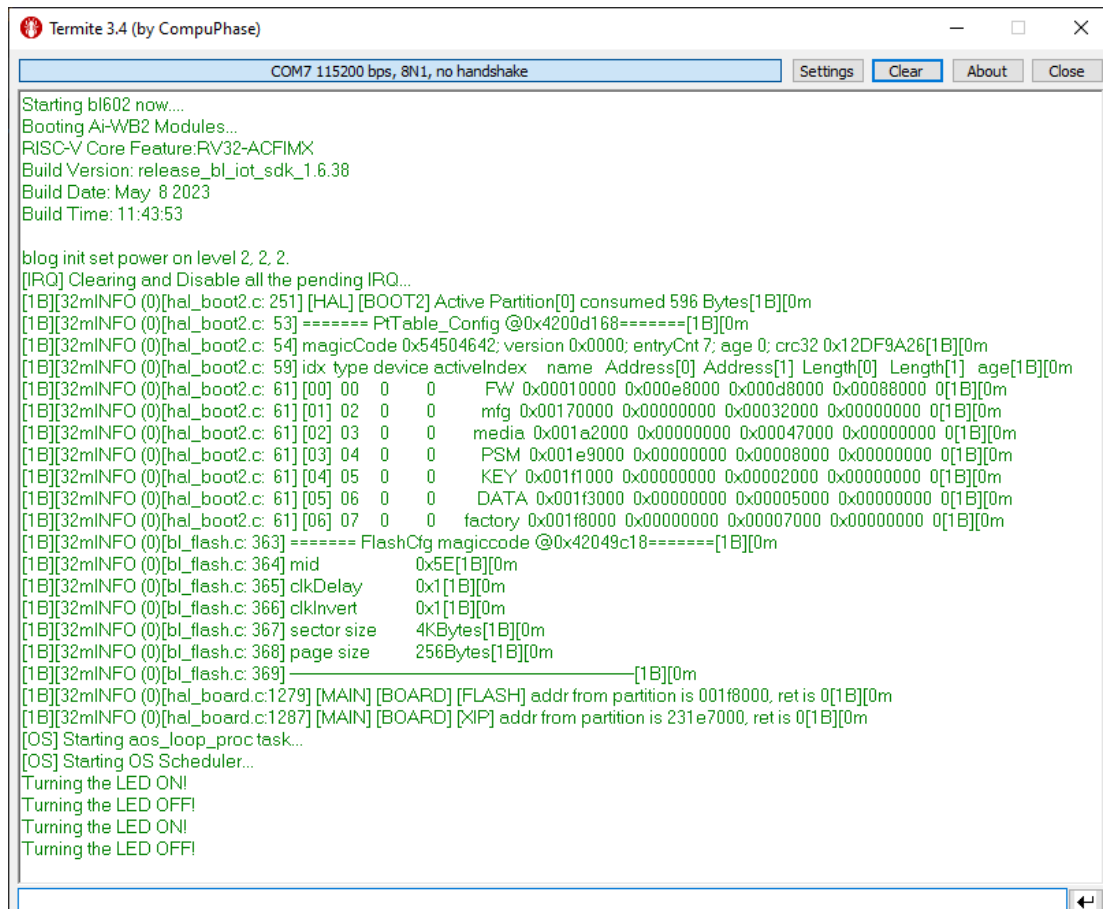
```

Problems Tasks Console X Properties
CDT Build Console [blink]
[21:42:25.376] - Program Finished
[21:42:25.376] - All time cost(ms): 6112.000244140625
[21:42:25.492] - close interface
[21:42:25.492] - [All Success]

21:42:25 Build Finished. 0 errors, 0 warnings. (took 1m:34s.526ms)

```

- However, you will not see the code immediately in action yet. Remember *Termite*? Open it now, and press *EN* again. On your Ai-WB2-12F, you will see a blue LED blinking on and off every second, and on *Termite*, you will see a bunch of green text like below. Every time the blue LED turns on, the module will also put *Turning the LED ON!* on the *Termite* screen, then *Turning the LED OFF!* when the blue LED turns off.



```

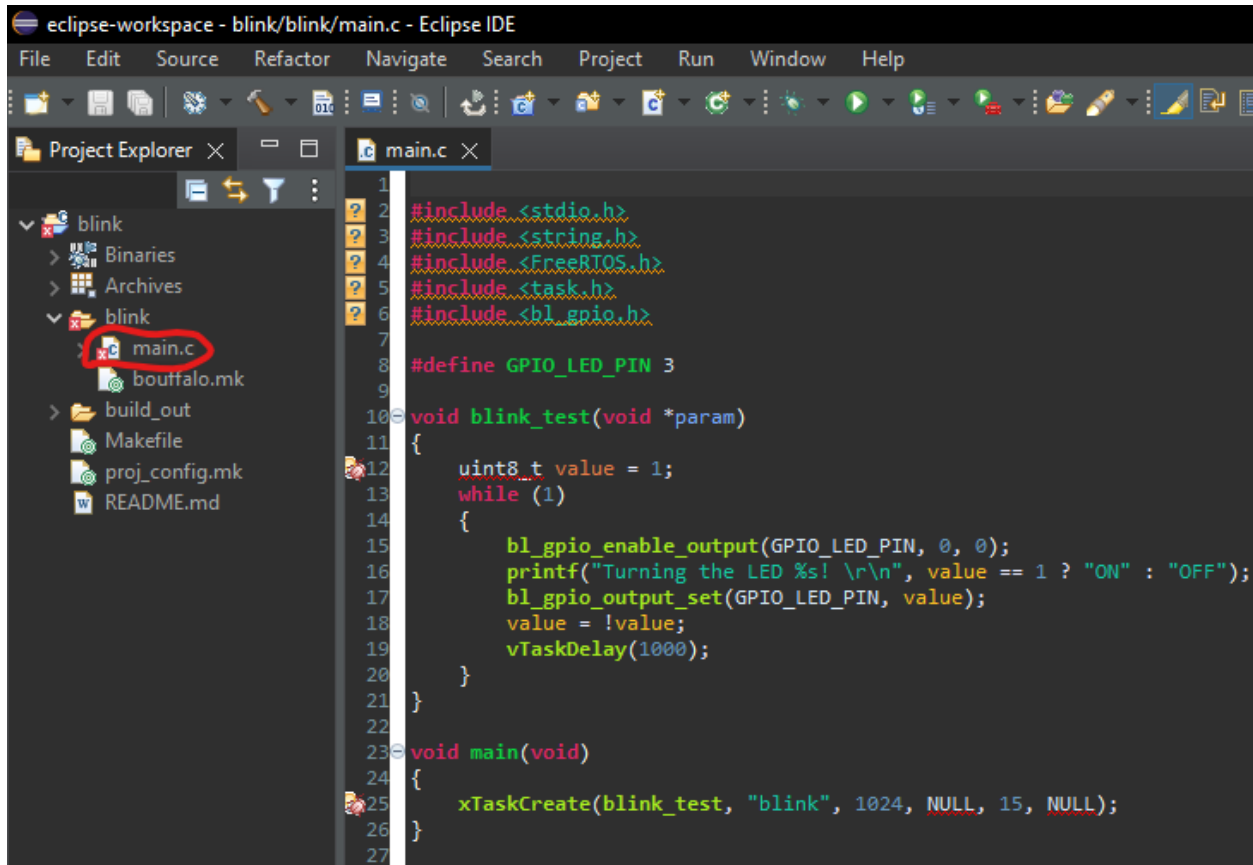
Termite 3.4 (by CompuPhase)
COM7 115200 bps, 8N1, no handshake
Settings Clear About Close

Starting bl602 now....
Booting Ai-WB2 Modules...
RISC-V Core Feature:RV32-ACFIMX
Build Version: release_bl_iot_sdk_1.6.38
Build Date: May 8 2023
Build Time: 11:43:53

blog init set power on level 2, 2, 2.
[IRQ] Clearing and Disable all the pending IRQ...
[1B][32miNFO (0)[hal_boot2.c: 251] [HAL] [BOOT2] Active Partition[0] consumed 596 Bytes[1B][0m
[1B][32miNFO (0)[hal_boot2.c: 53] ===== PTable_Config @0x4200d168=====[1B][0m
[1B][32miNFO (0)[hal_boot2.c: 54] magicCode 0x54504642; version 0x0000; entryCnt 7; age 0; crc32 0x12DF9A26[1B][0m
[1B][32miNFO (0)[hal_boot2.c: 59] idx type device activeIndex name Address[0] Address[1] Length[0] Length[1] age[1B][0m
[1B][32miNFO (0)[hal_boot2.c: 61] [00] 00 0 0 FW 0x00010000 0x000e8000 0x000d8000 0x00088000 0[1B][0m
[1B][32miNFO (0)[hal_boot2.c: 61] [01] 02 0 0 mfg 0x00170000 0x00000000 0x00032000 0x00000000 0[1B][0m
[1B][32miNFO (0)[hal_boot2.c: 61] [02] 03 0 0 media 0x001a2000 0x00000000 0x00047000 0x00000000 0[1B][0m
[1B][32miNFO (0)[hal_boot2.c: 61] [03] 04 0 0 PSM 0x001e9000 0x00000000 0x00008000 0x00000000 0[1B][0m
[1B][32miNFO (0)[hal_boot2.c: 61] [04] 05 0 0 KEY 0x001f1000 0x00000000 0x00002000 0x00000000 0[1B][0m
[1B][32miNFO (0)[hal_boot2.c: 61] [05] 06 0 0 DATA 0x001f3000 0x00000000 0x00005000 0x00000000 0[1B][0m
[1B][32miNFO (0)[hal_boot2.c: 61] [06] 07 0 0 factory 0x001f8000 0x00000000 0x00007000 0x00000000 0[1B][0m
[1B][32miNFO (0)[bl_flash.c: 363] ===== FlashCfg magiccode @0x42049c18=====[1B][0m
[1B][32miNFO (0)[bl_flash.c: 364] mid 0x5E[1B][0m
[1B][32miNFO (0)[bl_flash.c: 365] clkDelay 0x1[1B][0m
[1B][32miNFO (0)[bl_flash.c: 366] clkInvert 0x1[1B][0m
[1B][32miNFO (0)[bl_flash.c: 367] sector size 4KBytes[1B][0m
[1B][32miNFO (0)[bl_flash.c: 368] page size 256Bytes[1B][0m
[1B][32miNFO (0)[bl_flash.c: 369] -----[1B][0m
[1B][32miNFO (0)[hal_board.c:1279] [MAIN] [BOARD] [FLASH] addr from partition is 001f8000, ret is 0[1B][0m
[1B][32miNFO (0)[hal_board.c:1287] [MAIN] [BOARD] [XIP] addr from partition is 231e7000, ret is 0[1B][0m
[OS] Starting aos_loop_proc task...
[OS] Starting OS Scheduler...
Turning the LED ON!
Turning the LED OFF!
Turning the LED ON!
Turning the LED OFF!

```

Congratulation, you have successfully run your very first project on the *Ai-WB2-12F* by Ai-Thinker. If you are curious about the code, you can expand the project folder, then expand the *blink* folder inside. The main code will be in *main.c*.

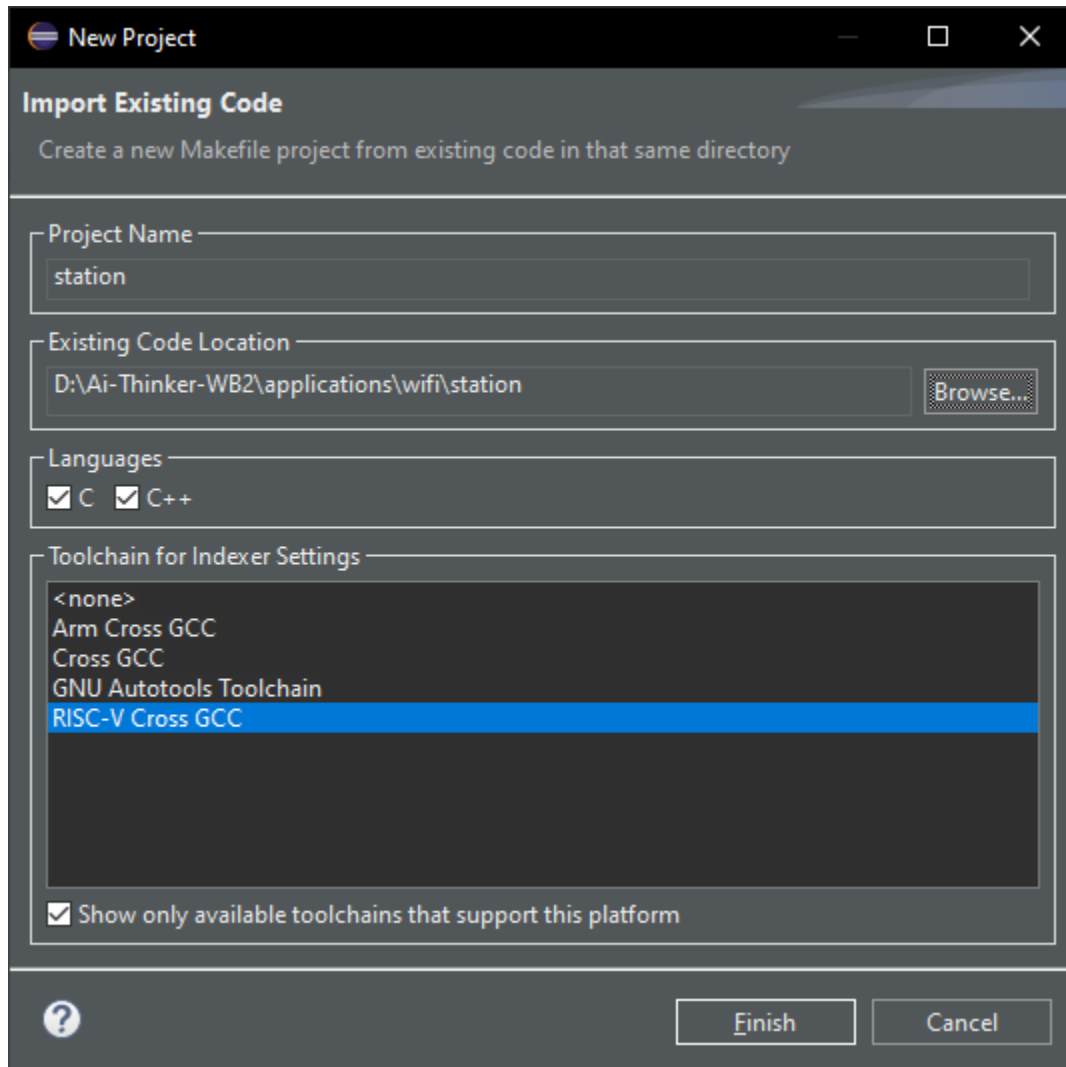


The screenshot shows the Eclipse IDE interface. The Project Explorer on the left displays the project structure: a 'blink' folder containing 'Binaries', 'Archives', 'main.c' (highlighted with a red circle), 'boutfalo.mk', 'build\_out', 'Makefile', 'proj\_config.mk', and 'README.md'. The main editor window shows the content of 'main.c'.

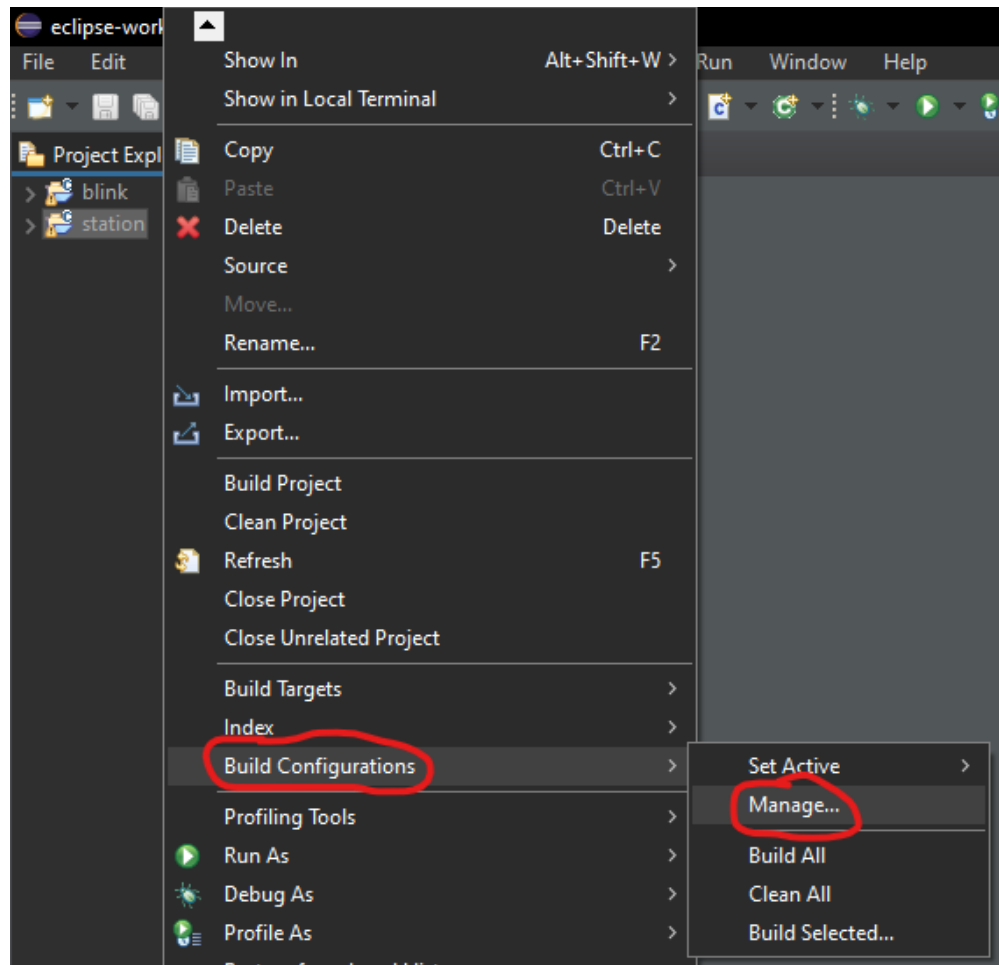
```
1
2 #include <stdio.h>
3 #include <string.h>
4 #include <FreeRTOS.h>
5 #include <task.h>
6 #include <bl_gpio.h>
7
8 #define GPIO_LED_PIN 3
9
10 void blink_test(void *param)
11 {
12     uint8_t value = 1;
13     while (1)
14     {
15         bl_gpio_enable_output(GPIO_LED_PIN, 0, 0);
16         printf("Turning the LED %s! \r\n", value == 1 ? "ON" : "OFF");
17         bl_gpio_output_set(GPIO_LED_PIN, value);
18         value = !value;
19         vTaskDelay(1000);
20     }
21 }
22
23 void main(void)
24 {
25     xTaskCreate(blink_test, "blink", 1024, NULL, 15, NULL);
26 }
27
```

## FEATURE TEST: WI-FI

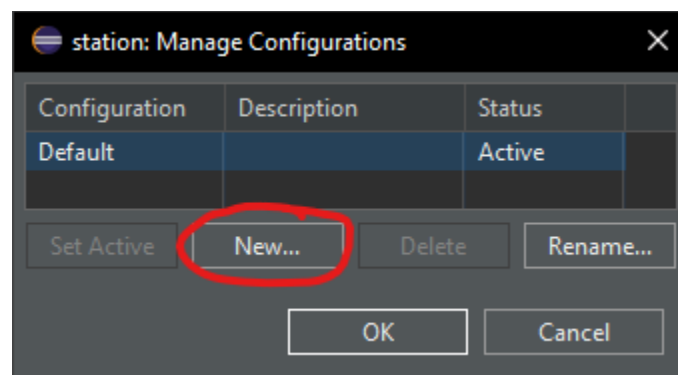
- Let's test one of Ai-WB2-12F's main features: Wi-Fi connection. With *Eclipse* still open, go to *File -> New -> Makefile Project with Existing Code*, and choose `\wifi\station` in the *get-started* folder.



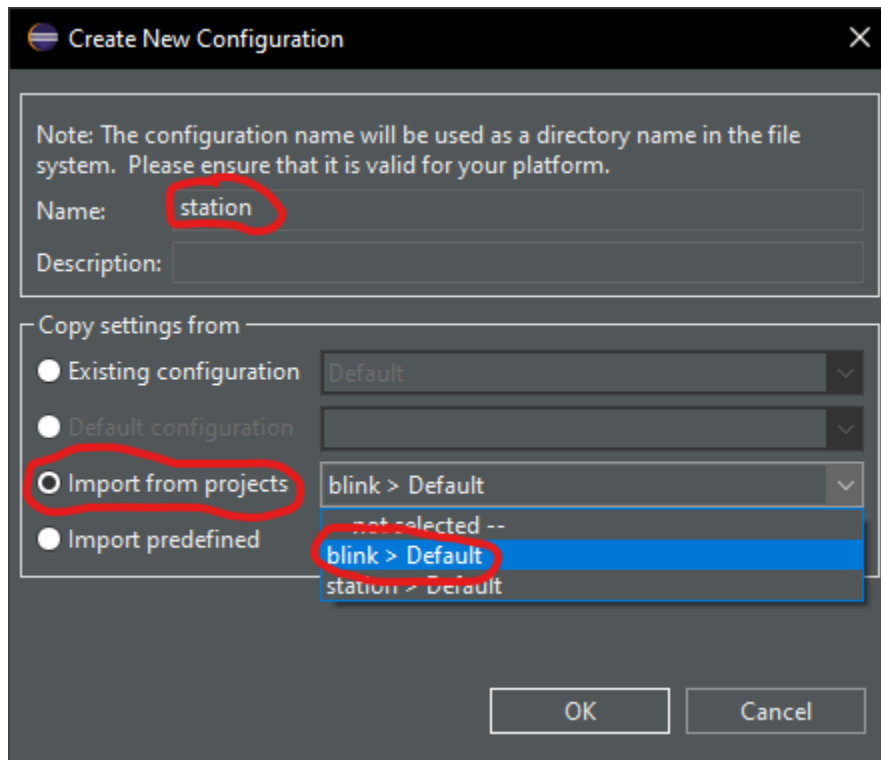
- Unfortunately, build configurations must be set separately for each project. To save time, you can copy previous configurations from the *blink* project. Right click on the *station* project, then go to *Build Configuration* -> *Manage...*



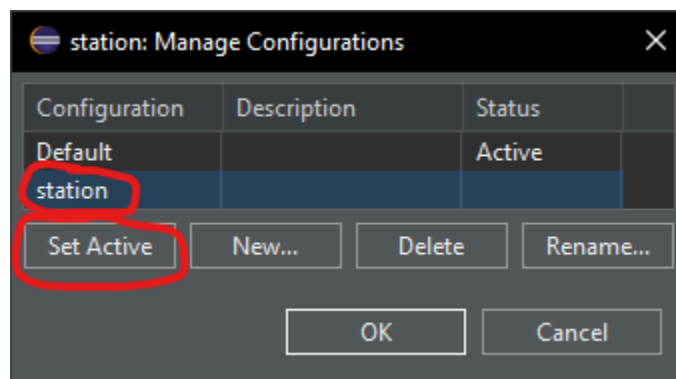
- In the *Manage Configurations* window, click *New...*



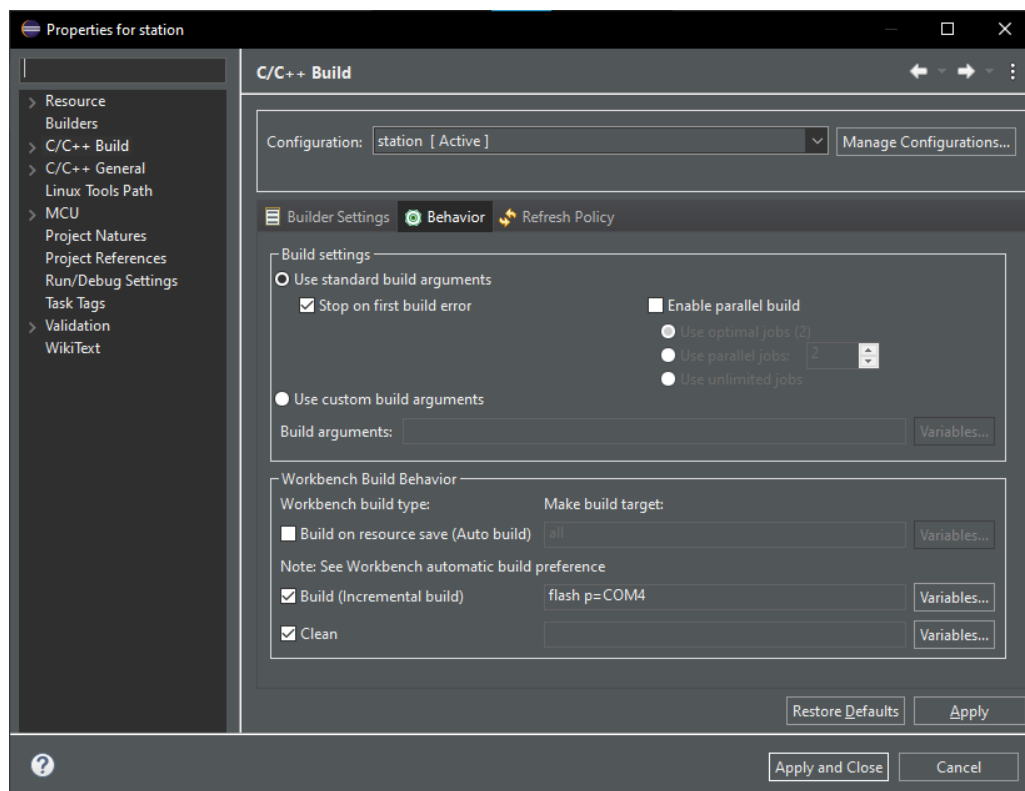
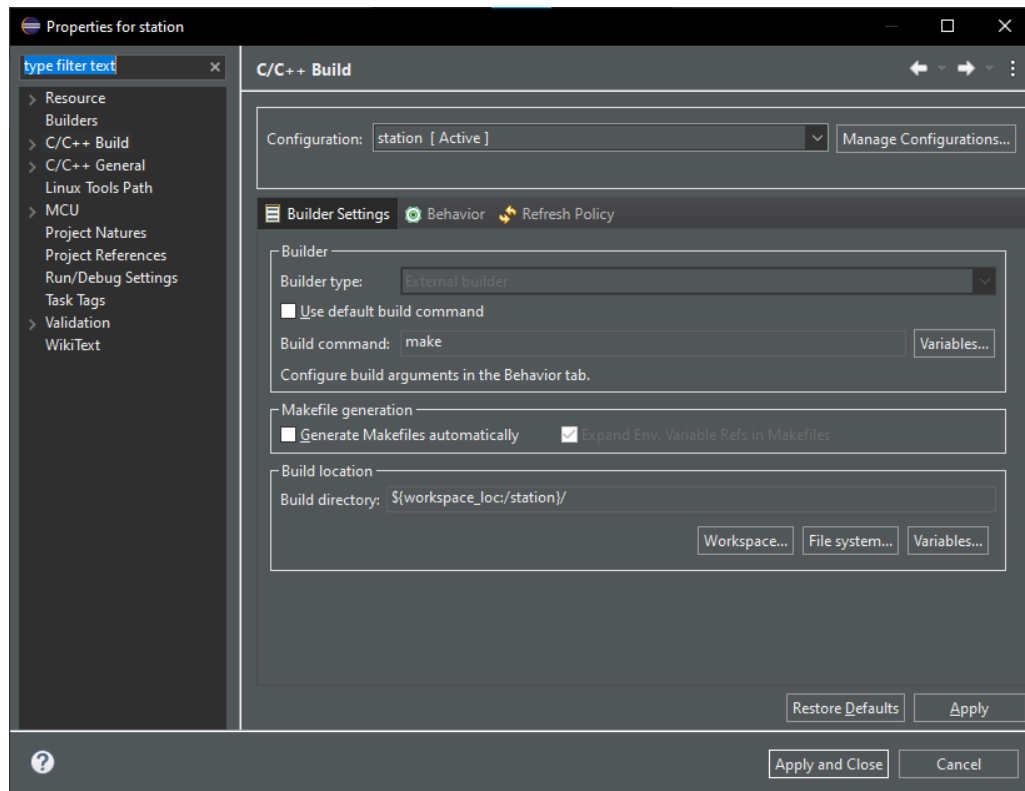
- Give the new configuration a name, then in *Copy settings from*, choose *Import from projects*, and choose *blink > Default* from the list.



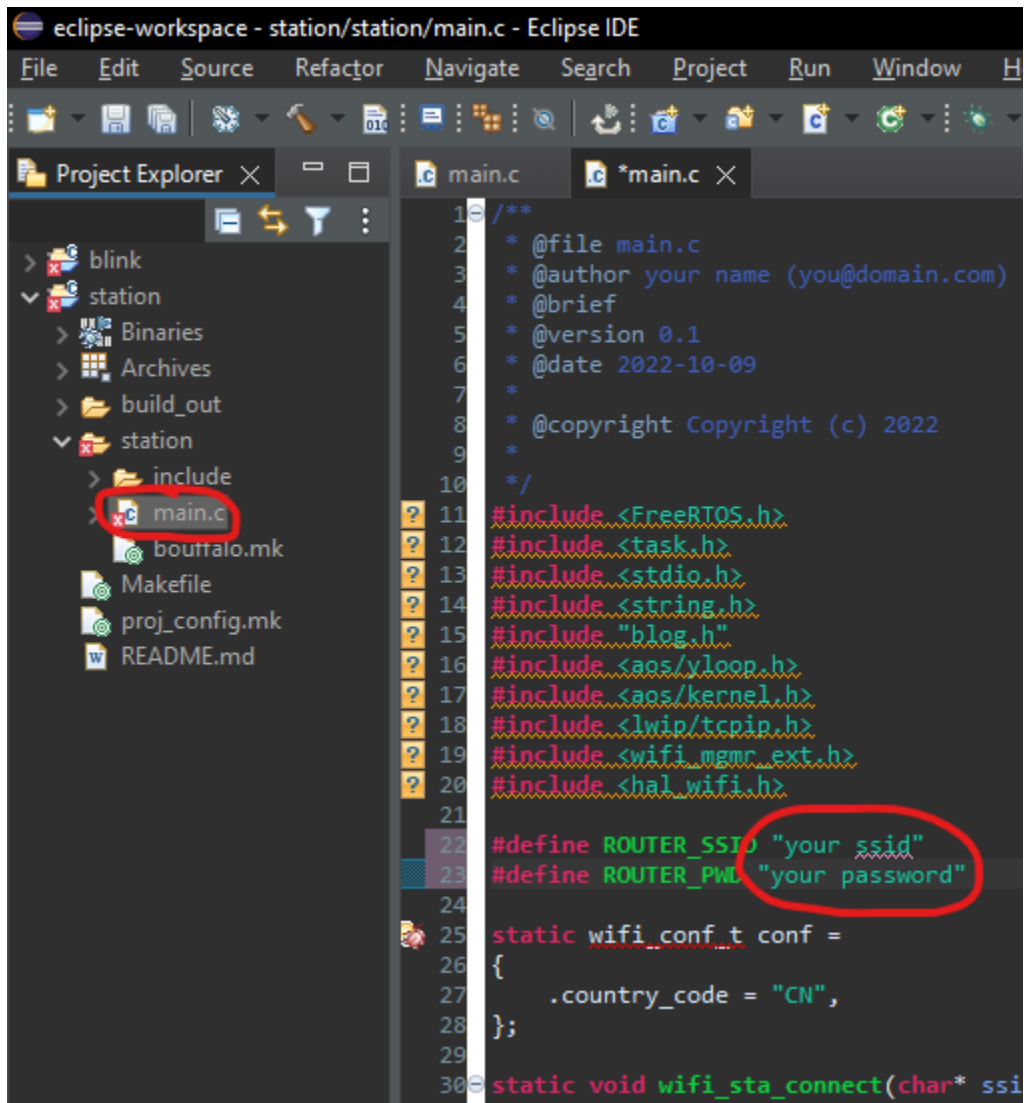
- Click *OK*, you will see the new configuration below *Default*. Select it and click on *Set Active*.



- The *station* project now has the same build configurations as the *blink* project. You can double check that by going to *Properties*:



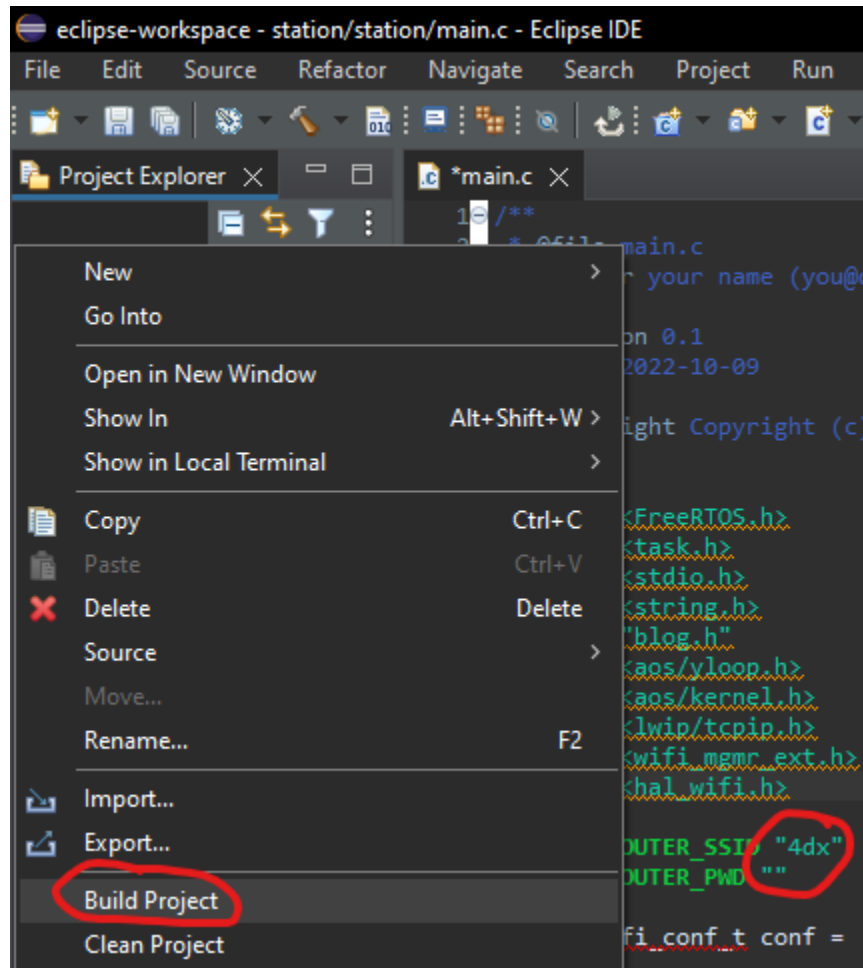
- Next, we need to adjust a few things in the main code. Expand the project, then expand the *station* folder, and double click on *main.c*. Change *your ssid* to your nearest Wi-Fi SSID, and *your password* to that Wi-Fi's password, or `""` if the Wi-Fi is open.



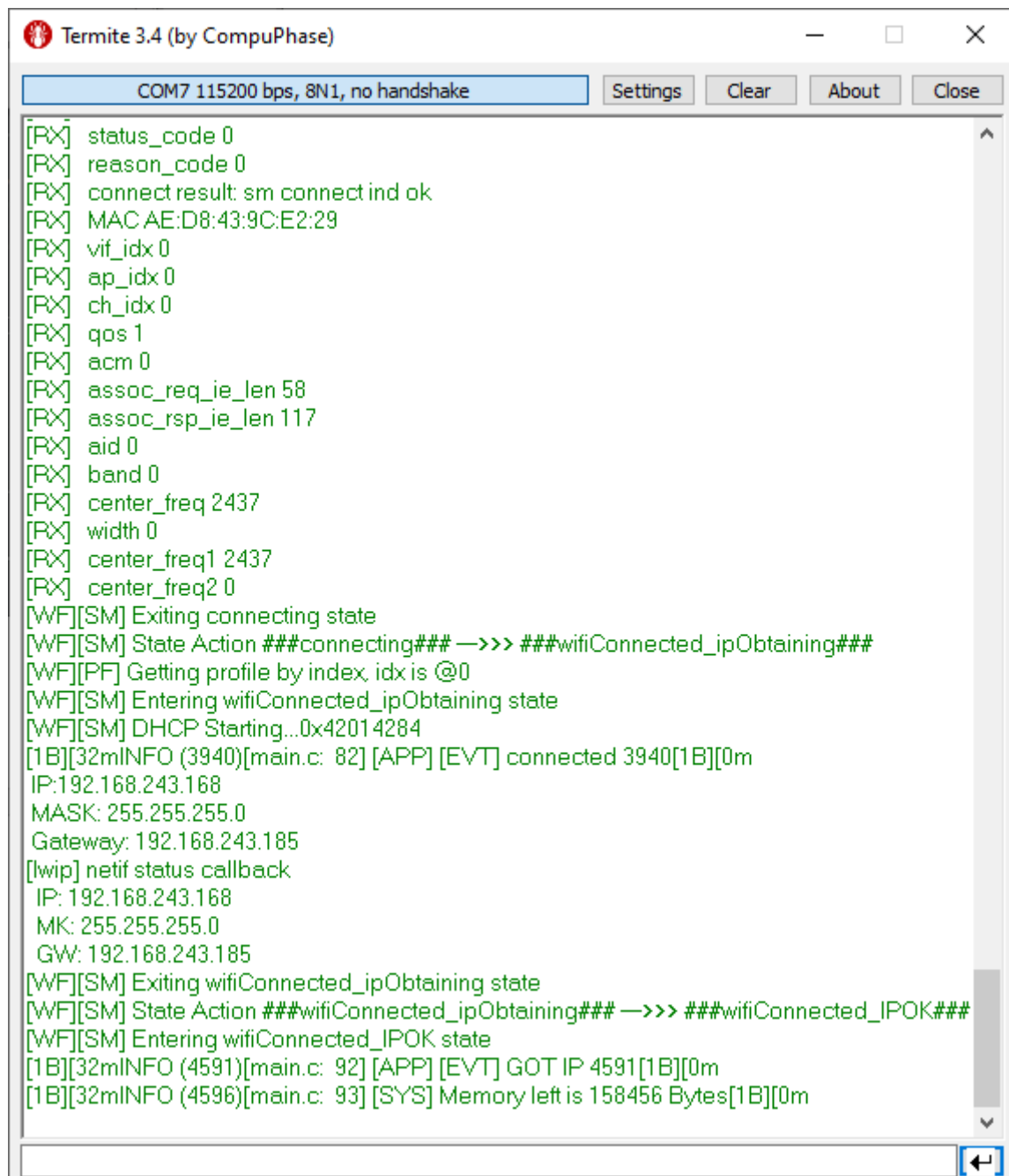
```
1 /**
2  * @file main.c
3  * @author your name (you@domain.com)
4  * @brief
5  * @version 0.1
6  * @date 2022-10-09
7  *
8  * @copyright Copyright (c) 2022
9  *
10 */
11 #include <FreeRTOS.h>
12 #include <task.h>
13 #include <stdio.h>
14 #include <string.h>
15 #include "blog.h"
16 #include <aos/vloop.h>
17 #include <aos/kernel.h>
18 #include <lwip/tcpip.h>
19 #include <wifi_mgmr_ext.h>
20 #include <hal_wifi.h>
21
22 #define ROUTER_SSID "your ssid"
23 #define ROUTER_PWD "your password"
24
25 static wifi_conf_t conf =
26 {
27     .country_code = "CN",
28 };
29
30 static void wifi_sta_connect(char* ssi
```



- Right click the project folder, and to save time, click *Build Project* directly, no need to click *Clean Project* first. Once again, wait for the right time to press *EN* on the Ai-WB2-12F module to download the code.



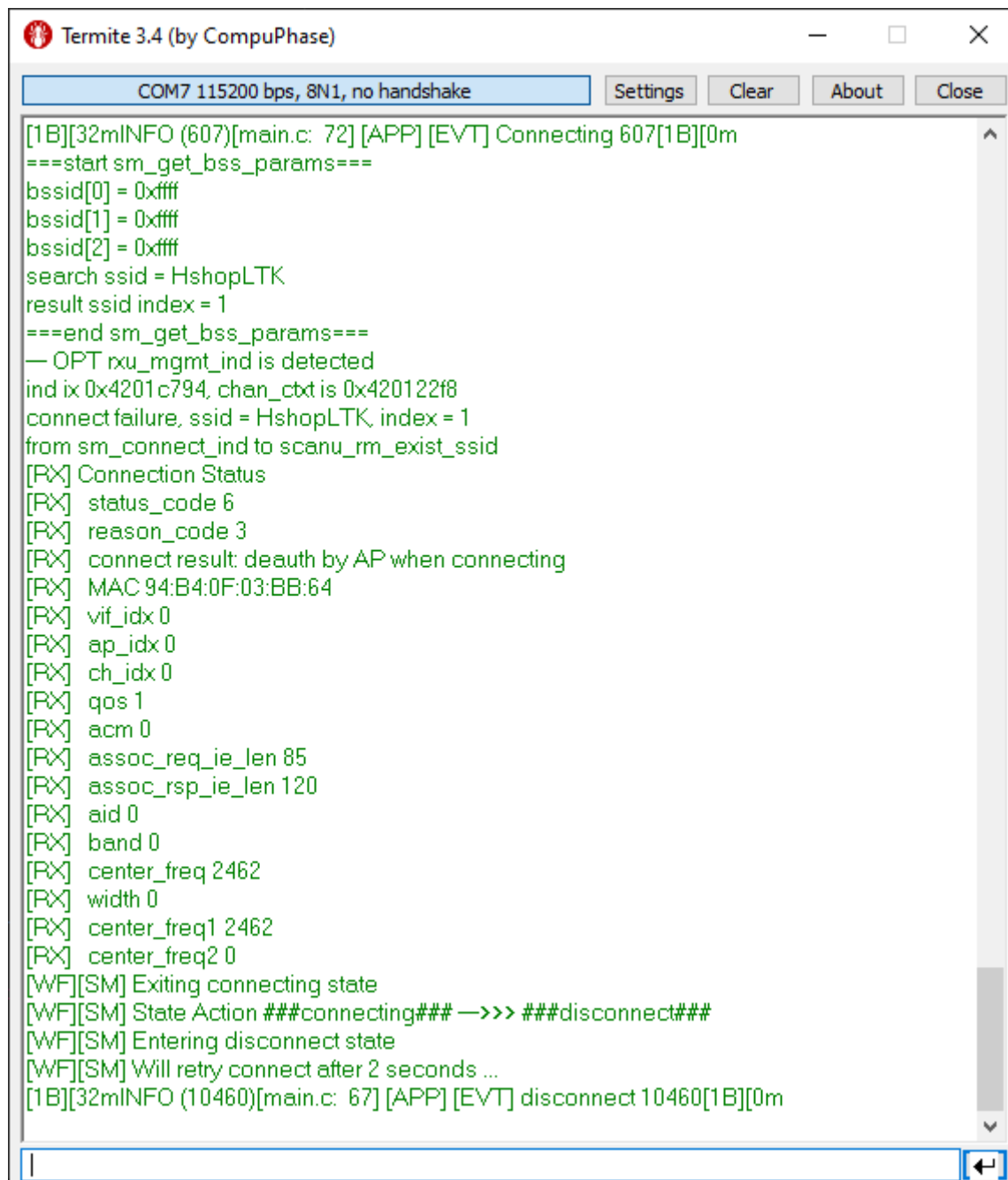
- Open *Termite* and press *EN* again. If the provided SSID and password is correct, the final lines of the output will look like this:



The screenshot shows the Termite 3.4 terminal window. The title bar reads "Termite 3.4 (by CompuPhase)". Below the title bar is a status bar with "COM7 115200 bps, 8N1, no handshake" and buttons for "Settings", "Clear", "About", and "Close". The main area contains a log of network-related events in green text. The log shows the device connecting to a network, receiving an IP address of 192.168.243.168, and successfully obtaining an IP address. The final lines of the log indicate the device is now connected and has memory left.

```
[RX] status_code 0
[RX] reason_code 0
[RX] connect result: sm connect ind ok
[RX] MAC AE:D8:43:9C:E2:29
[RX] vif_idx 0
[RX] ap_idx 0
[RX] ch_idx 0
[RX] qos 1
[RX] acm 0
[RX] assoc_req_ie_len 58
[RX] assoc_rsp_ie_len 117
[RX] aid 0
[RX] band 0
[RX] center_freq 2437
[RX] width 0
[RX] center_freq1 2437
[RX] center_freq2 0
[WF][SM] Exiting connecting state
[WF][SM] State Action ###connecting### ->>> ###wifiConnected_ipObtaining###
[WF][PF] Getting profile by index, idx is @0
[WF][SM] Entering wifiConnected_ipObtaining state
[WF][SM] DHCP Starting...0x42014284
[1B][32mINFO (3940)[main.c: 82] [APP] [EVT] connected 3940[1B][0m
IP:192.168.243.168
MASK: 255.255.255.0
Gateway: 192.168.243.185
[lwip] netif status callback
IP: 192.168.243.168
MK: 255.255.255.0
GW: 192.168.243.185
[WF][SM] Exiting wifiConnected_ipObtaining state
[WF][SM] State Action ###wifiConnected_ipObtaining### ->>> ###wifiConnected_IPOK###
[WF][SM] Entering wifiConnected_IPOK state
[1B][32mINFO (4591)[main.c: 92] [APP] [EVT] GOT IP 4591[1B][0m
[1B][32mINFO (4596)[main.c: 93] [SYS] Memory left is 158456 Bytes[1B][0m
```

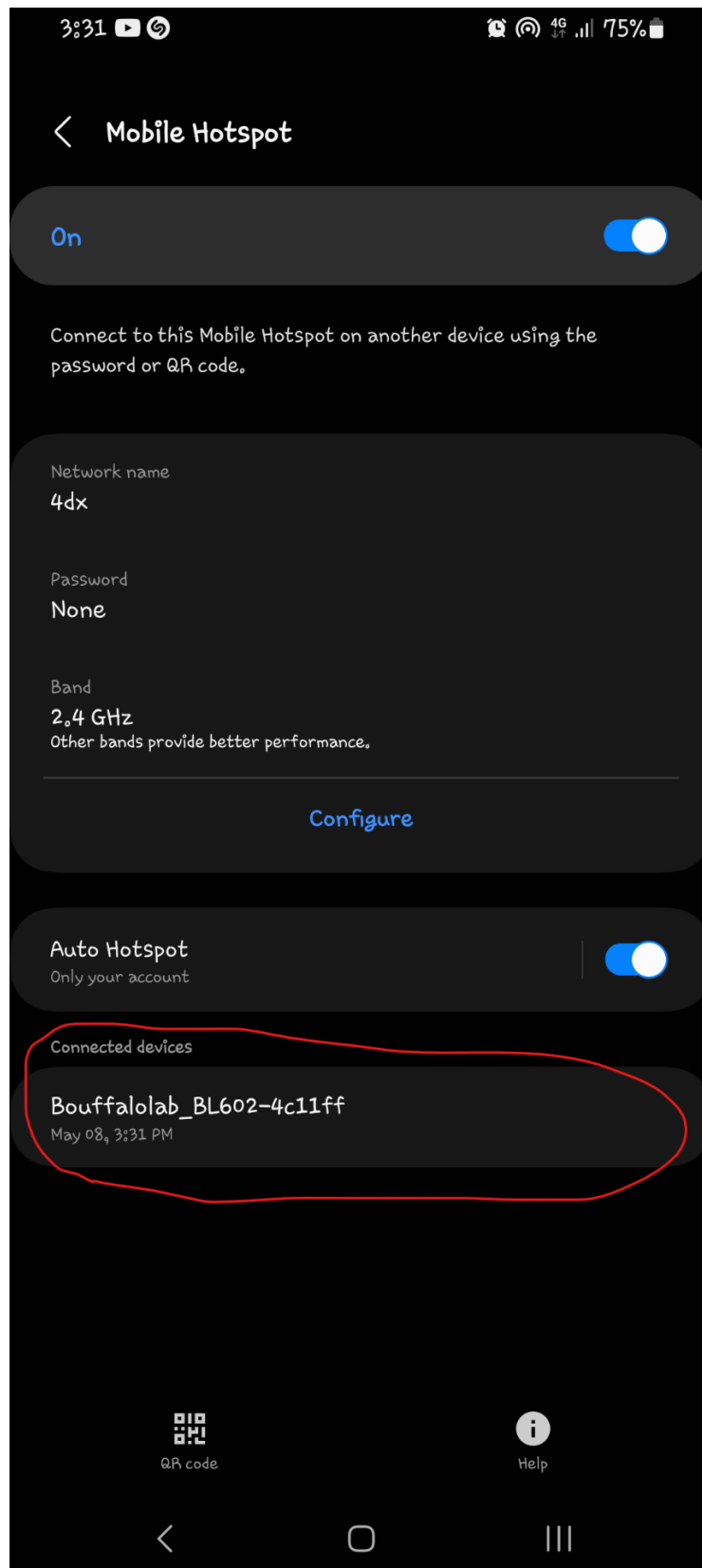
- Otherwise, if either information is incorrect, then it will infinitely attempt to reconnect every 2 seconds:



```
Termite 3.4 (by CompuPhase)
COM7 115200 bps, 8N1, no handshake [Settings] [Clear] [About] [Close]

[1B][32mINFO (607)[main.c: 72] [APP] [EVT] Connecting 607[1B][0m
===start sm_get_bss_params===
bssid[0] = 0xffff
bssid[1] = 0xffff
bssid[2] = 0xffff
search ssid = HshopLTK
result ssid index = 1
===end sm_get_bss_params===
— OPT rxu_mgmt_ind is detected
ind ix 0x4201c794, chan_ctxt is 0x420122f8
connect failure, ssid = HshopLTK, index = 1
from sm_connect_ind to scanu_rm_exist_ssid
[RX] Connection Status
[RX] status_code 6
[RX] reason_code 3
[RX] connect result: deauth by AP when connecting
[RX] MAC 94:B4:0F:03:BB:64
[RX] vif_idx 0
[RX] ap_idx 0
[RX] ch_idx 0
[RX] qos 1
[RX] acm 0
[RX] assoc_req_ie_len 85
[RX] assoc_rsp_ie_len 120
[RX] aid 0
[RX] band 0
[RX] center_freq 2462
[RX] width 0
[RX] center_freq1 2462
[RX] center_freq2 0
[WF][SM] Exiting connecting state
[WF][SM] State Action ###connecting### ->>> ###disconnect###
[WF][SM] Entering disconnect state
[WF][SM] Will retry connect after 2 seconds ...
[1B][32mINFO (10460)[main.c: 67] [APP] [EVT] disconnect 10460[1B][0m
```

- If you use your phone as a Wi-Fi hotspot, you will see the core chip's info:



## CONCLUSION

Through this tutorial, you have set up suitable environment on your computer in order to successfully utilize the *Ai-Thinker's Ai-WB2-12F module* through two example projects: blinking an LED, and connecting to a Wi-Fi network. Feel free to leave a comment below if you have any questions.