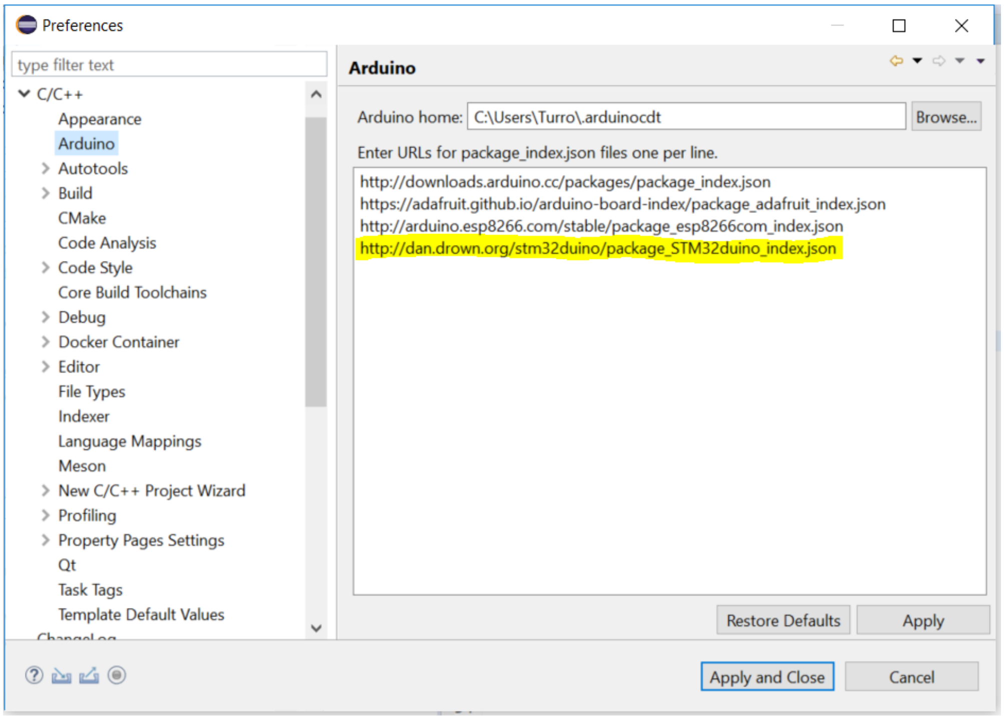
How to setup Eclipse ide to compile, flash and debug Gevino (aka arduino zero) and generic STM32F103 (aka blue pill)

1) Setup Eclipse as Arduino IDE:

   I followed instructions reported here : [Lean Eclipse IDE Setup]

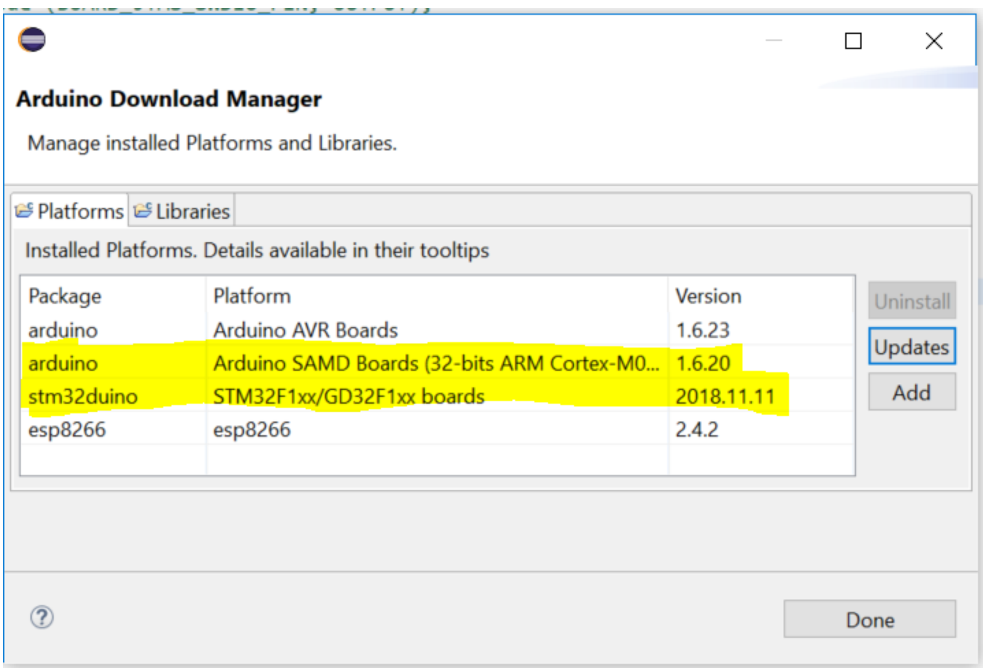
2) Install also arduino samd boards support



Arduino home: C:\Users\Turro\.arduinocdt

Enter URLs for package_index.json files one per line.

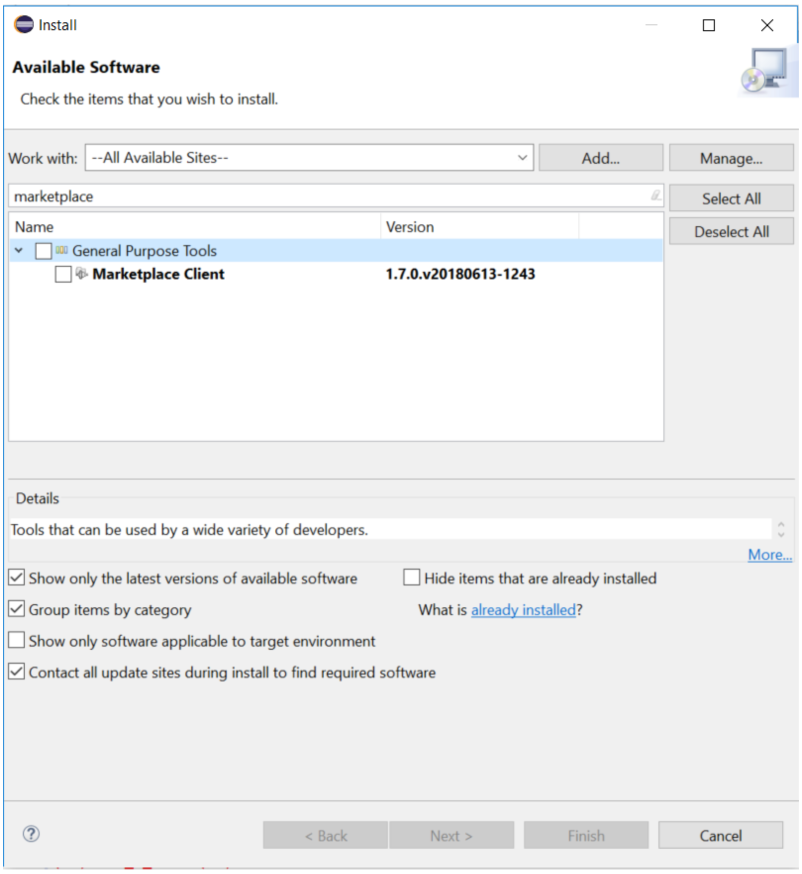
```
http://downloads.arduino.cc/packages/package_index.json
https://adafruit.github.io/arduino-board-index/package_adafruit_index.json
http://arduino.esp8266.com/stable/package_esp8266com_index.json
http://dan.drown.org/stm32duino/package_STM32duino_index.json
```



Installed Platforms. Details available in their tooltips

Package	Platform	Version
arduino	Arduino AVR Boards	1.6.23
arduino	Arduino SAMD Boards (32-bits ARM Cortex-M0...	1.6.20
stm32duino	STM32F1xx/GD32F1xx boards	2018.11.11
esp8266	esp8266	2.4.2

3) Install marketplace through help->install new software…



Install

Available Software

Check the items that you wish to install.

Work with: --All Available Sites--

marketplace

Name	Version
□ General Purpose Tools	
□ **Marketplace Client**	**1.7.0.v20180613-1243**

Details

Tools that can be used by a wide variety of developers.

☑ Show only the latest versions of available software

□ Hide items that are already installed

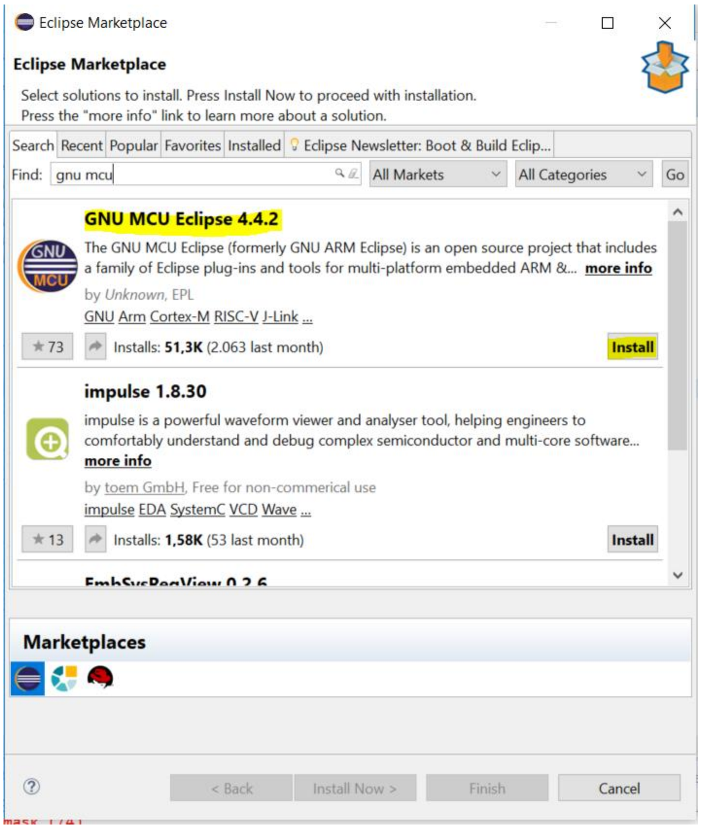
☑ Group items by category

What is already installed?

□ Show only software applicable to target environment

☑ Contact all update sites during install to find required software

4) In help->marketplace install GNU MCU Eclipse Plugin



Eclipse Marketplace

Eclipse Marketplace

Select solutions to install. Press Install Now to proceed with installation.
Press the "more info" link to learn more about a solution.

Find: gnu mcu | All Markets | All Categories | Go

GNU MCU Eclipse 4.4.2

The GNU MCU Eclipse (formerly GNU ARM Eclipse) is an open source project that includes a family of Eclipse plug-ins and tools for multi-platform embedded ARM &... **more info**

by *Unknown*, EPL

GNU Arm Cortex-M RISC-V J-Link ...

★ 73 Installs: **51,3K** (2.063 last month) **Install**

impulse 1.8.30

impulse is a powerful waveform viewer and analyser tool, helping engineers to comfortably understand and debug complex semiconductor and multi-core software... **more info**

by toem GmbH, Free for non-commerical use

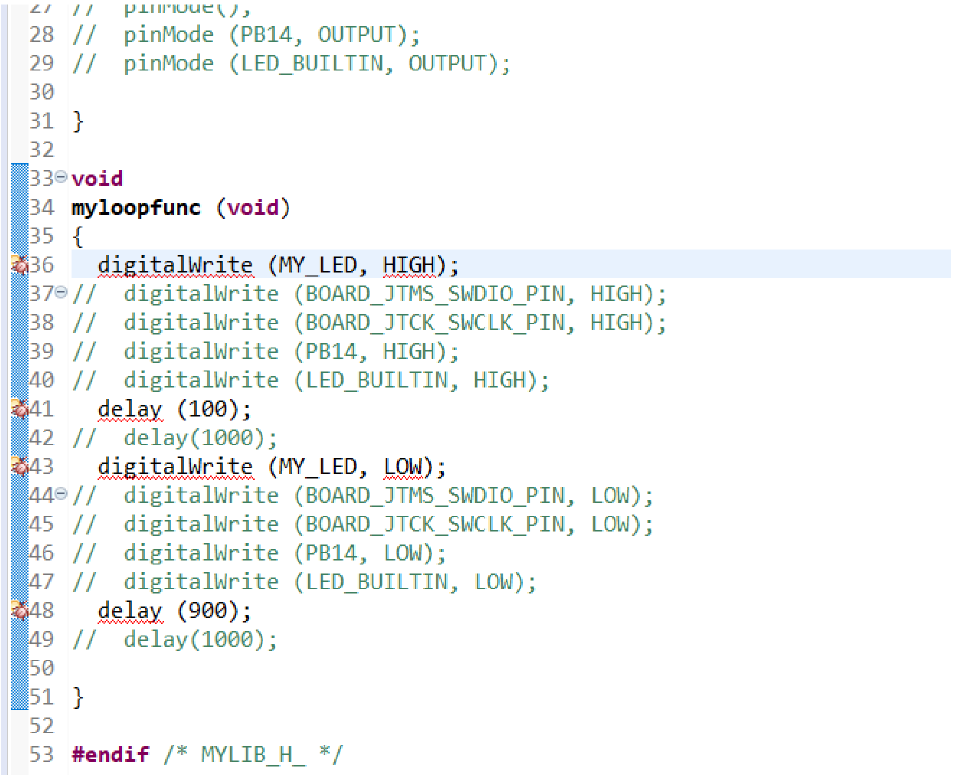
impulse EDA SystemC VCD Wave ...

★ 13 Installs: **1,58K** (53 last month) **Install**

Marketplaces

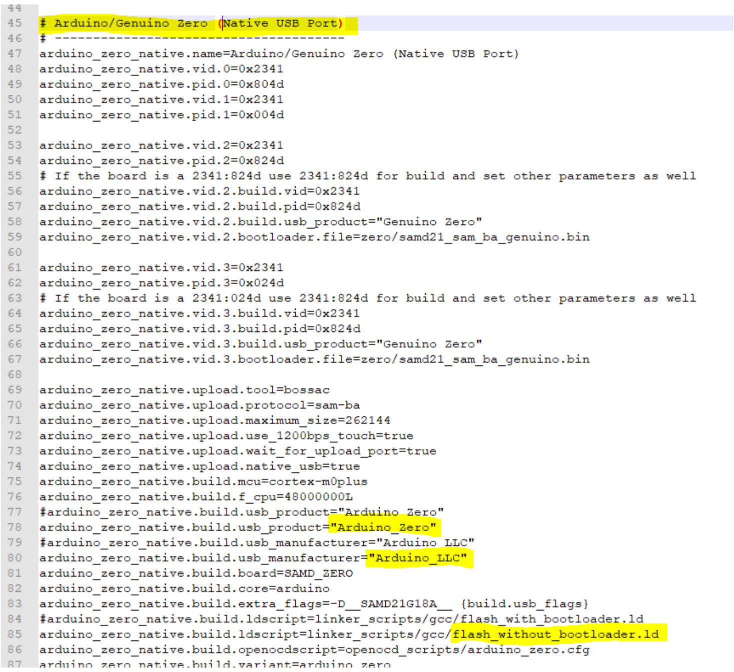
5) Gevino: edit boards.txt

In order to avoid failures in the code indexer which is IMO one of the main benefit on using eclipse instead of arduino ide (i.e. autocompletion of the code, finding declarations,etc…)



```
//   pinMode();
//   pinMode (PB14, OUTPUT);
//   pinMode (LED_BUILTIN, OUTPUT);

}

void
myloopfunc (void)
{
   digitalWrite (MY_LED, HIGH);
//   digitalWrite (BOARD_JTMS_SWDIO_PIN, HIGH);
//   digitalWrite (BOARD_JTCK_SWCLK_PIN, HIGH);
//   digitalWrite (PB14, HIGH);
//   digitalWrite (LED_BUILTIN, HIGH);
   delay (100);
//   delay(1000);
   digitalWrite (MY_LED, LOW);
//   digitalWrite (BOARD_JTMS_SWDIO_PIN, LOW);
//   digitalWrite (BOARD_JTCK_SWCLK_PIN, LOW);
//   digitalWrite (PB14, LOW);
//   digitalWrite (LED_BUILTIN, LOW);
   delay (900);
//   delay(1000);

}

#endif /* MYLIB_H_ */
```

edit the samd boards.txt as below:

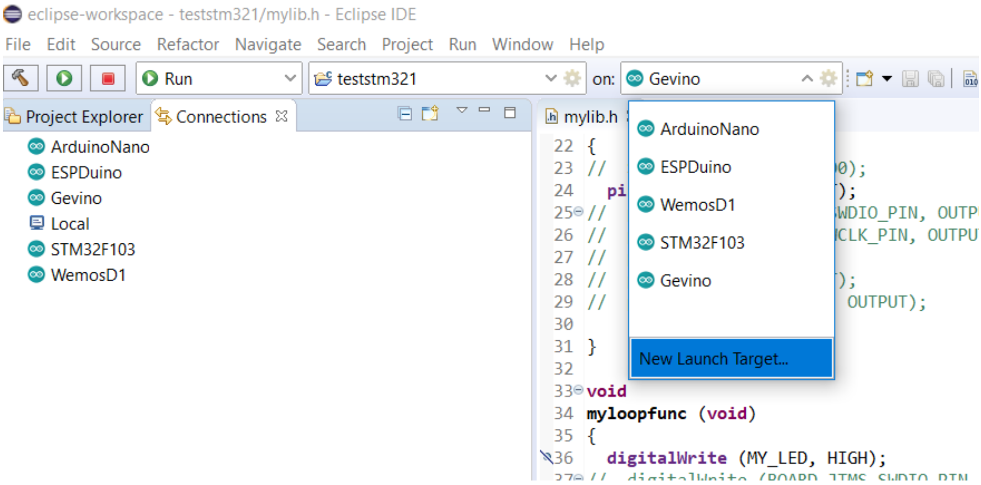


```
# Arduino/Genuino Zero (Native USB Port)
# --------------------------------------
arduino_zero_native.name=Arduino/Genuino Zero (Native USB Port)
arduino_zero_native.vid.0=0x2341
arduino_zero_native.pid.0=0x804d
arduino_zero_native.vid.1=0x2341
arduino_zero_native.pid.1=0x004d

arduino_zero_native.vid.2=0x2341
arduino_zero_native.pid.2=0x824d
# If the board is a 2341:824d use 2341:824d for build and set other parameters as well
arduino_zero_native.vid.2.build.vid=0x2341
arduino_zero_native.vid.2.build.pid=0x824d
arduino_zero_native.vid.2.build.usb_product="Genuino Zero"
arduino_zero_native.vid.2.bootloader.file=zero/samd21_sam_ba_genuino.bin

arduino_zero_native.vid.3=0x2341
arduino_zero_native.pid.3=0x024d
# If the board is a 2341:024d use 2341:824d for build and set other parameters as well
arduino_zero_native.vid.3.build.vid=0x2341
arduino_zero_native.vid.3.build.pid=0x824d
arduino_zero_native.vid.3.build.usb_product="Genuino Zero"
arduino_zero_native.vid.3.bootloader.file=zero/samd21_sam_ba_genuino.bin

arduino_zero_native.upload.tool=bossac
arduino_zero_native.upload.protocol=sam-ba
arduino_zero_native.upload.maximum_size=262144
arduino_zero_native.upload.use_1200bps_touch=true
arduino_zero_native.upload.wait_for_upload_port=true
arduino_zero_native.upload.native_usb=true
arduino_zero_native.build.mcu=cortex-m0plus
arduino_zero_native.build.f_cpu=48000000L
#arduino_zero_native.build.usb_product="Arduino Zero"
arduino_zero_native.build.usb_product="Arduino_Zero"
#arduino_zero_native.build.usb_manufacturer="Arduino LLC"
arduino_zero_native.build.usb_manufacturer="Arduino_LLC"
arduino_zero_native.build.board=SAMD_ZERO
arduino_zero_native.build.core=arduino
arduino_zero_native.build.extra_flags=-D__SAMD21G18A__ {build.usb_flags}
#arduino_zero_native.build.ldscript=linker_scripts/gcc/flash_with_bootloader.ld
arduino_zero_native.build.ldscript=linker_scripts/gcc/flash_without_bootloader.ld
arduino_zero_native.build.openocdscript=openocd_scripts/arduino_zero.cfg
arduino_zero_native.build.variant=arduino_zero
```

Edit the ldscript only if You have a valid cmsis-dap debug adapter

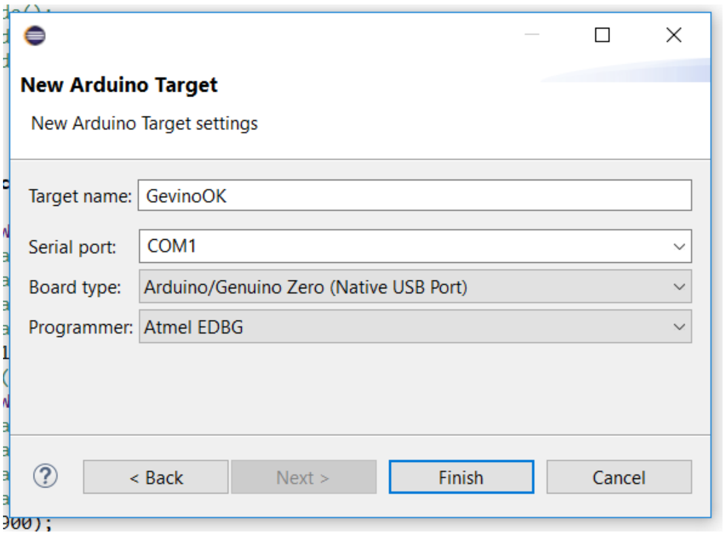
Save the file and restart eclipse to let the ide reading the new setting

6) Create an Arduino connection



```
{
//                          0);
  pi                        );
//                          WDIO_PIN, OUTP
//                          CLK_PIN, OUTPU
//
//                          );
//                           OUTPUT);

}

void
myloopfunc (void)
{
  digitalWrite (MY_LED, HIGH);
```

Like this for Gevino:



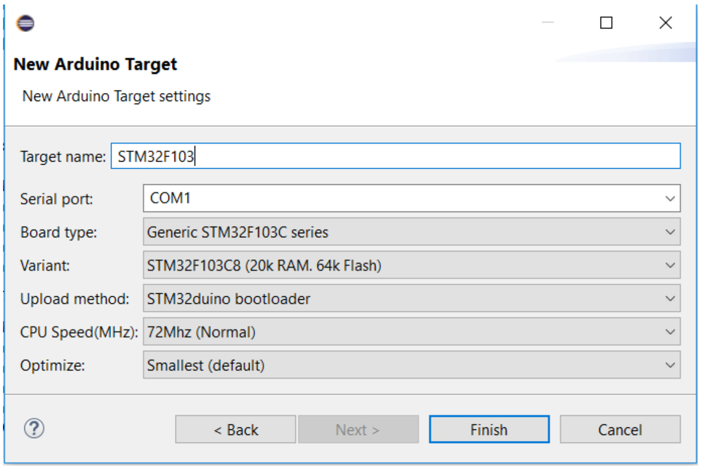
Target name: GevinoOK

Serial port: COM1

Board type: Arduino/Genuino Zero (Native USB Port)

Programmer: Atmel EDBG

Like this for STM32F103:



Target name: STM32F103

Serial port: COM1

Board type: Generic STM32F103C series

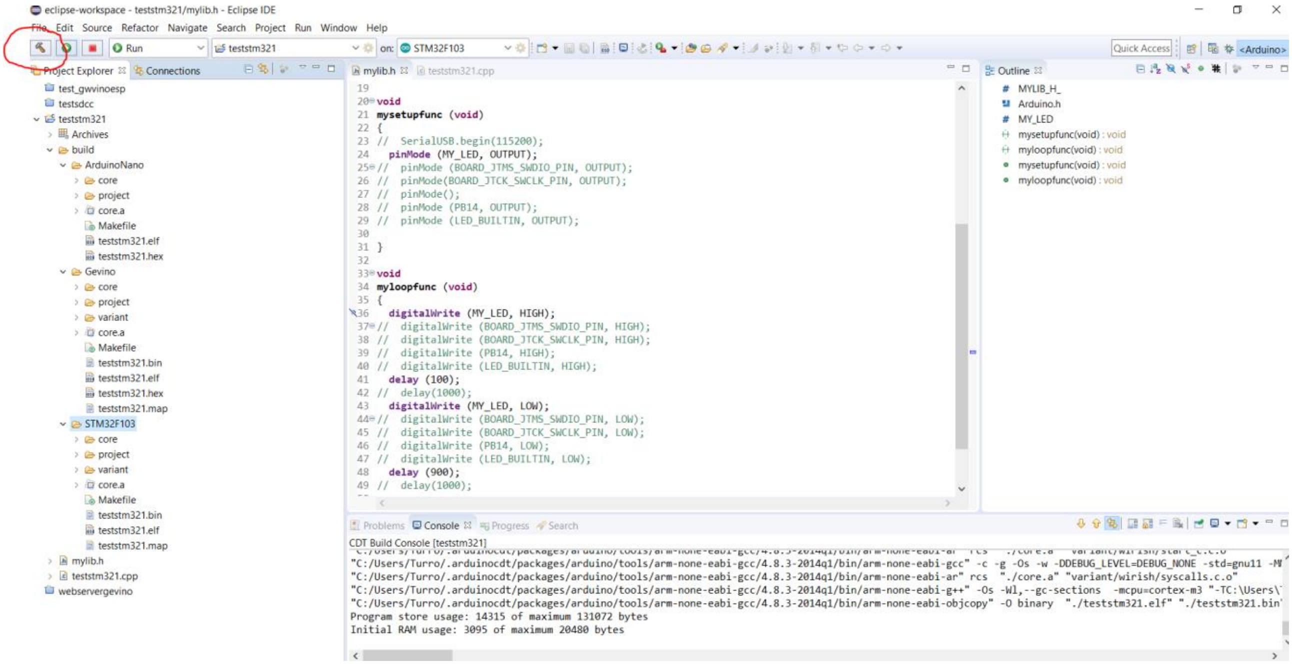
Variant: STM32F103C8 (20k RAM. 64k Flash)

Upload method: STM32duino bootloader

CPU Speed(MHz): 72Mhz (Normal)

Optimize: Smallest (default)

This is a really useful benefit of arduino plugin as the same code can be compiled (click the hammer button) for different target just by setting the launcher

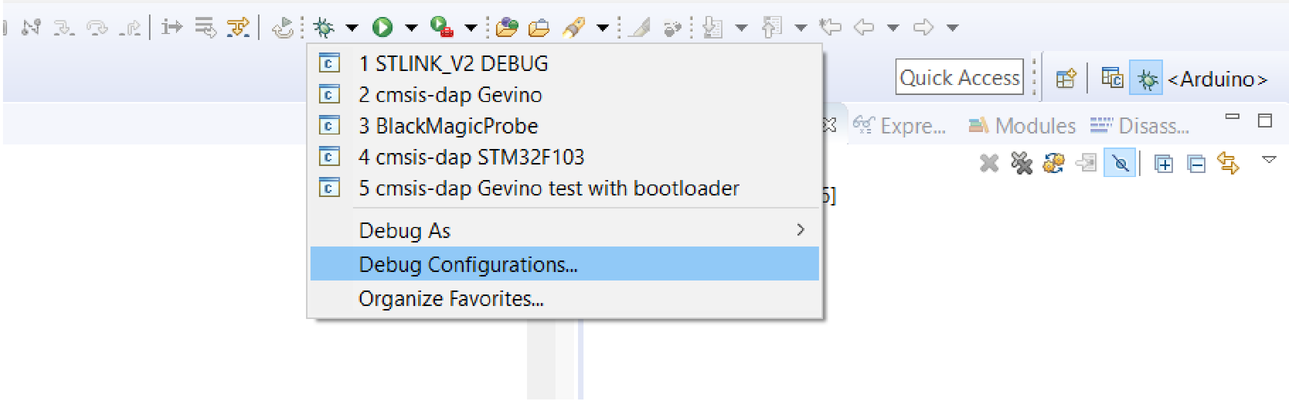


```

void
mysetupfunc (void)
{
//  SerialUSB.begin(115200);
  pinMode (MY_LED, OUTPUT);
//  pinMode (BOARD_JTMS_SWDIO_PIN, OUTPUT);
//  pinMode(BOARD_JTCK_SWCLK_PIN, OUTPUT);
//  pinMode();
//  pinMode (PB14, OUTPUT);
//  pinMode (LED_BUILTIN, OUTPUT);

}

void
myloopfunc (void)
{
  digitalWrite (MY_LED, HIGH);
//  digitalWrite (BOARD_JTMS_SWDIO_PIN, HIGH);
//  digitalWrite (BOARD_JTCK_SWCLK_PIN, HIGH);
//  digitalWrite (PB14, HIGH);
//  digitalWrite (LED_BUILTIN, HIGH);
  delay (100);
//  delay(1000);
  digitalWrite (MY_LED, LOW);
//  digitalWrite (BOARD_JTMS_SWDIO_PIN, LOW);
//  digitalWrite (BOARD_JTCK_SWCLK_PIN, LOW);
//  digitalWrite (PB14, LOW);
//  digitalWrite (LED_BUILTIN, LOW);
  delay (900);
//  delay(1000);
```

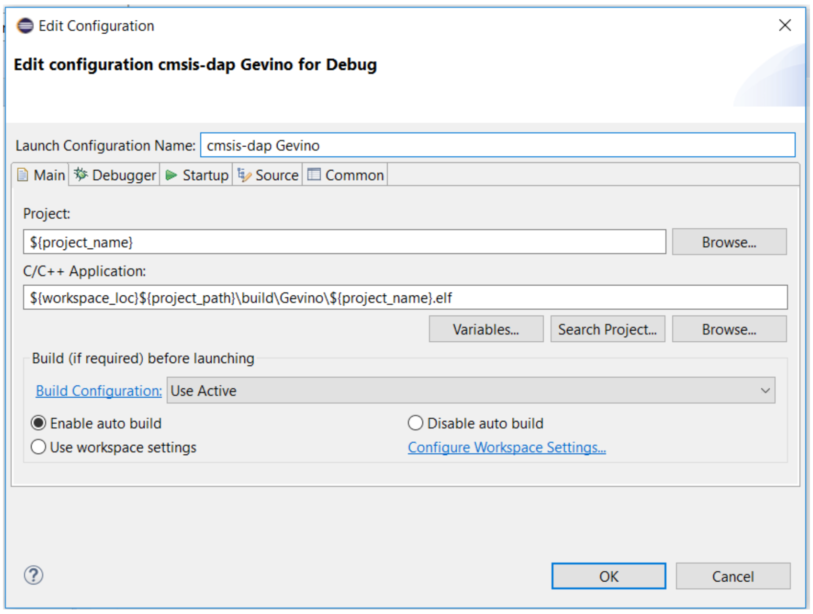
```
"C:/Users/Turro/.arduinocdt/packages/arduino/tools/arm-none-eabi-gcc/4.8.3-2014q1/bin/arm-none-eabi-gcc" -c -g -Os -w -DDEBUG_LEVEL=DEBUG_NONE -std=gnu11 -MI
"C:/Users/Turro/.arduinocdt/packages/arduino/tools/arm-none-eabi-gcc/4.8.3-2014q1/bin/arm-none-eabi-ar" rcs  "./core.a" "variant/wirish/syscalls.c.o"
"C:/Users/Turro/.arduinocdt/packages/arduino/tools/arm-none-eabi-gcc/4.8.3-2014q1/bin/arm-none-eabi-g++" -Os -Wl,--gc-sections  -mcpu=cortex-m3 "-TC:\Users\
"C:/Users/Turro/.arduinocdt/packages/arduino/tools/arm-none-eabi-gcc/4.8.3-2014q1/bin/arm-none-eabi-objcopy" -O binary  "./teststm321.elf" "./teststm321.bin
Program store usage: 14315 of maximum 131072 bytes
Initial RAM usage: 3095 of maximum 20480 bytes
```

The following make sense only if You have a valid CMSIS-DAP debugger adapter

7) Now create the debug configuration
Switch to Debug perspective and create a new debug configuration



For Gevino create an OpenOCD Debug configuration as below:



Edit configuration cmsis-dap Gevino for Debug

Launch Configuration Name: cmsis-dap Gevino

Project: ${project_name}

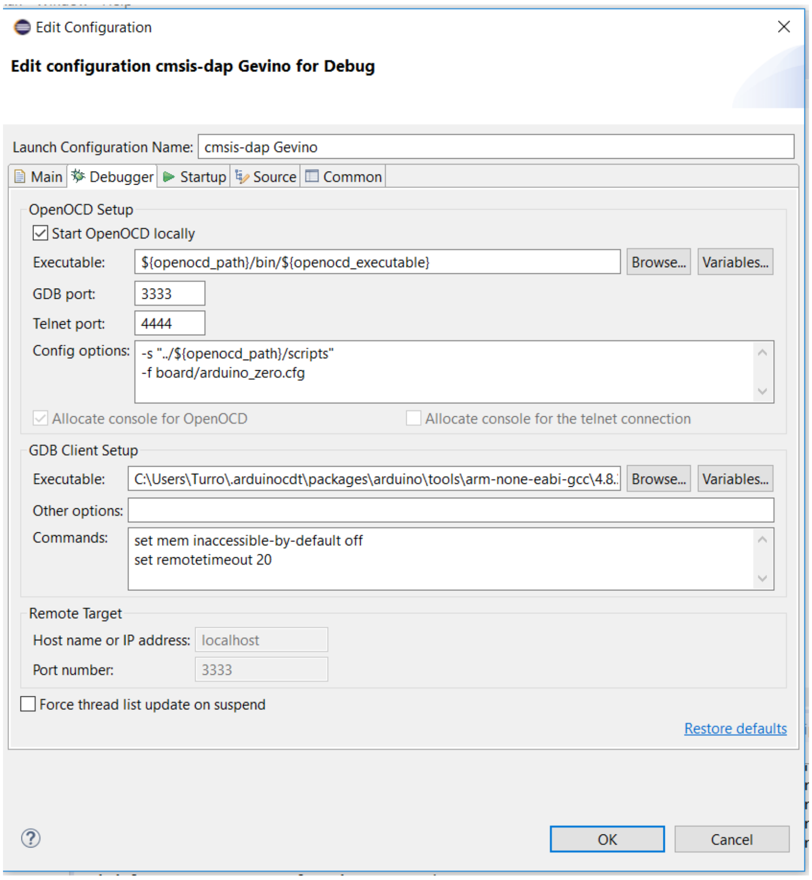
C/C++ Application: ${workspace_loc}${project_path}\build\Gevino\${project_name}.elf

Build (if required) before launching

Build Configuration: Use Active

● Enable auto build ○ Disable auto build

○ Use workspace settings



Edit configuration cmsis-dap Gevino for Debug

Launch Configuration Name: cmsis-dap Gevino

OpenOCD Setup

☑ Start OpenOCD locally

Executable: ${openocd_path}/bin/${openocd_executable}

GDB port: 3333

Telnet port: 4444

Config options:

```
-s "../${openocd_path}/scripts"
-f board/arduino_zero.cfg
```

☑ Allocate console for OpenOCD □ Allocate console for the telnet connection

GDB Client Setup

Executable: C:\Users\Turro\.arduinocdt\packages\arduino\tools\arm-none-eabi-gcc\4.8.

Other options:

Commands:

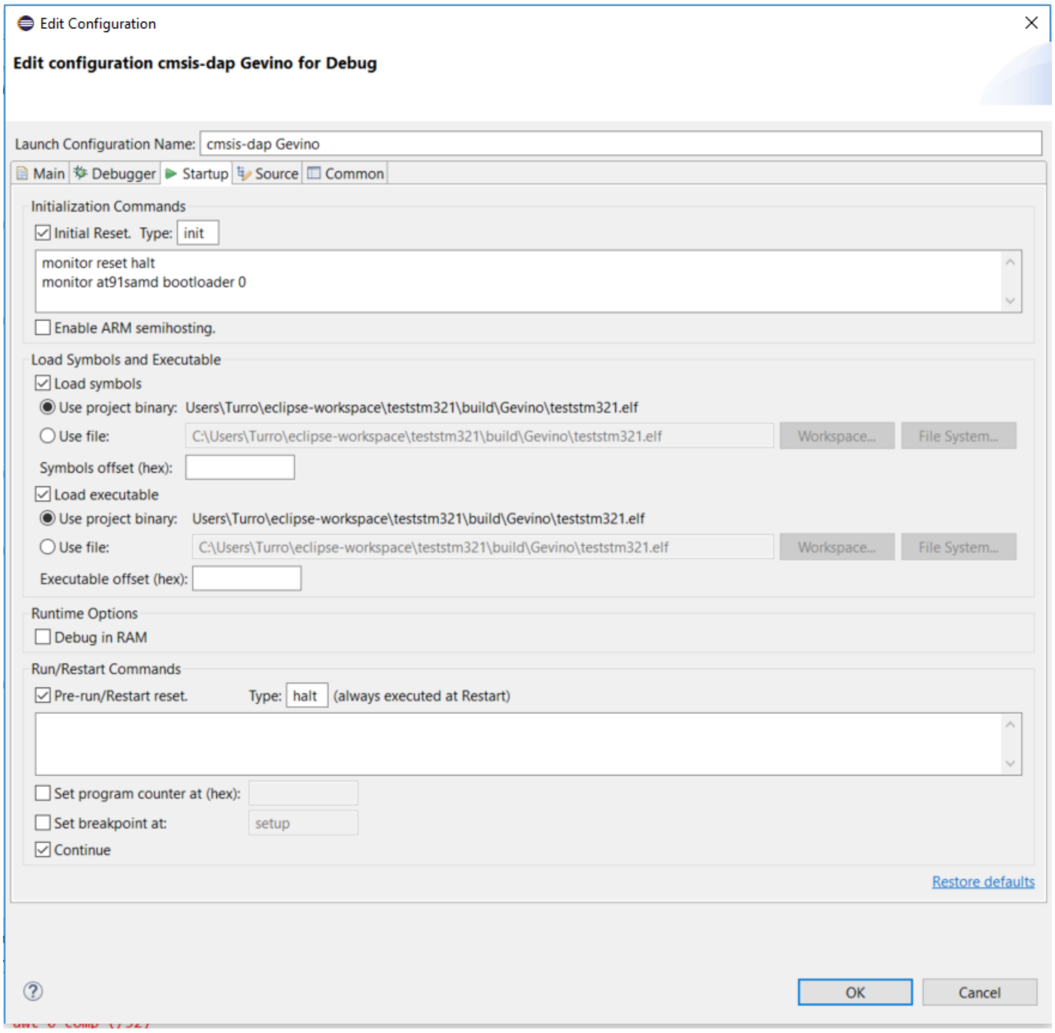
```
set mem inaccessible-by-default off
set remotetimeout 20
```

Remote Target

Host name or IP address: localhost

Port number: 3333

□ Force thread list update on suspend



Edit Configuration

Edit configuration cmsis-dap Gevino for Debug

Launch Configuration Name: cmsis-dap Gevino

Initialization Commands

☑ Initial Reset. Type: init

```
monitor reset halt
monitor at91samd bootloader 0
```

□ Enable ARM semihosting.

Load Symbols and Executable

☑ Load symbols

● Use project binary: Users\Turro\eclipse-workspace\teststm321\build\Gevino\teststm321.elf

○ Use file: C:\Users\Turro\eclipse-workspace\teststm321\build\Gevino\teststm321.elf

Symbols offset (hex):

☑ Load executable

● Use project binary: Users\Turro\eclipse-workspace\teststm321\build\Gevino\teststm321.elf

○ Use file: C:\Users\Turro\eclipse-workspace\teststm321\build\Gevino\teststm321.elf

Executable offset (hex):

Runtime Options

□ Debug in RAM

Run/Restart Commands

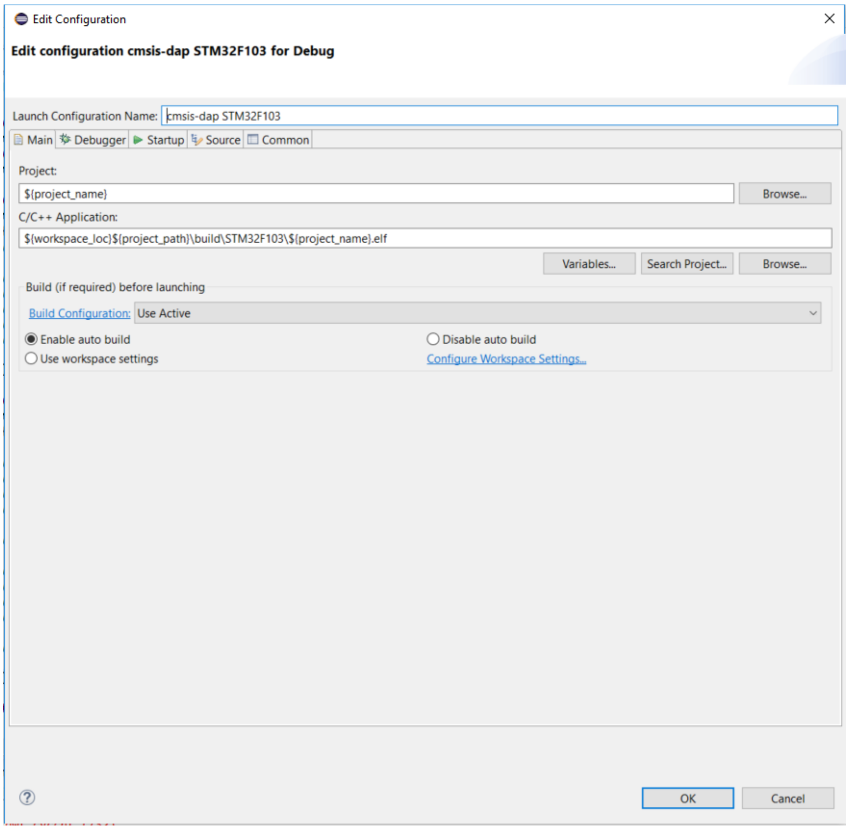
☑ Pre-run/Restart reset. Type: halt (always executed at Restart)

□ Set program counter at (hex):

□ Set breakpoint at: setup

☑ Continue

For STM32F103 create an OpenOCD Debug configuration as below:



Edit Configuration

Edit configuration cmsis-dap STM32F103 for Debug

Launch Configuration Name: cmsis-dap STM32F103

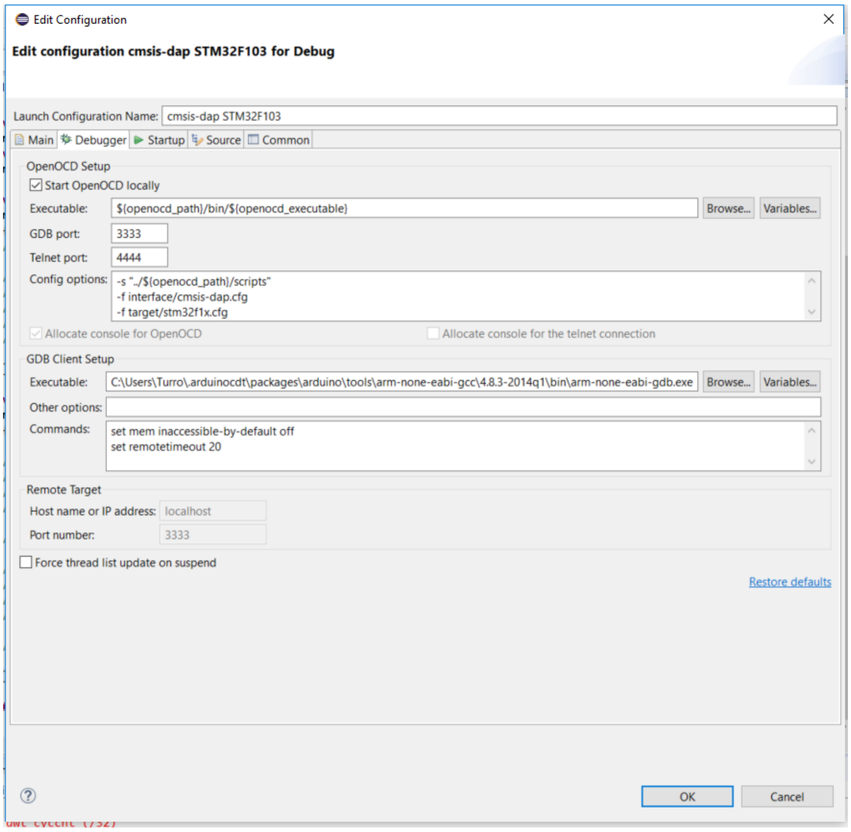
Project:
${project_name}

C/C++ Application:
${workspace_loc}${project_path}\build\STM32F103\${project_name}.elf

Build (if required) before launching

Build Configuration: Use Active

● Enable auto build
○ Disable auto build
○ Use workspace settings



Edit Configuration

Edit configuration cmsis-dap STM32F103 for Debug

Launch Configuration Name: cmsis-dap STM32F103

OpenOCD Setup

☑ Start OpenOCD locally

Executable: ${openocd_path}/bin/${openocd_executable}

GDB port: 3333

Telnet port: 4444

Config options:

```
-s "../${openocd_path}/scripts"
-f interface/cmsis-dap.cfg
-f target/stm32f1x.cfg
```

☑ Allocate console for OpenOCD
□ Allocate console for the telnet connection

GDB Client Setup

Executable: C:\Users\Turro\.arduinocdt\packages\arduino\tools\arm-none-eabi-gcc\4.8.3-2014q1\bin\arm-none-eabi-gdb.exe

Other options:

Commands:

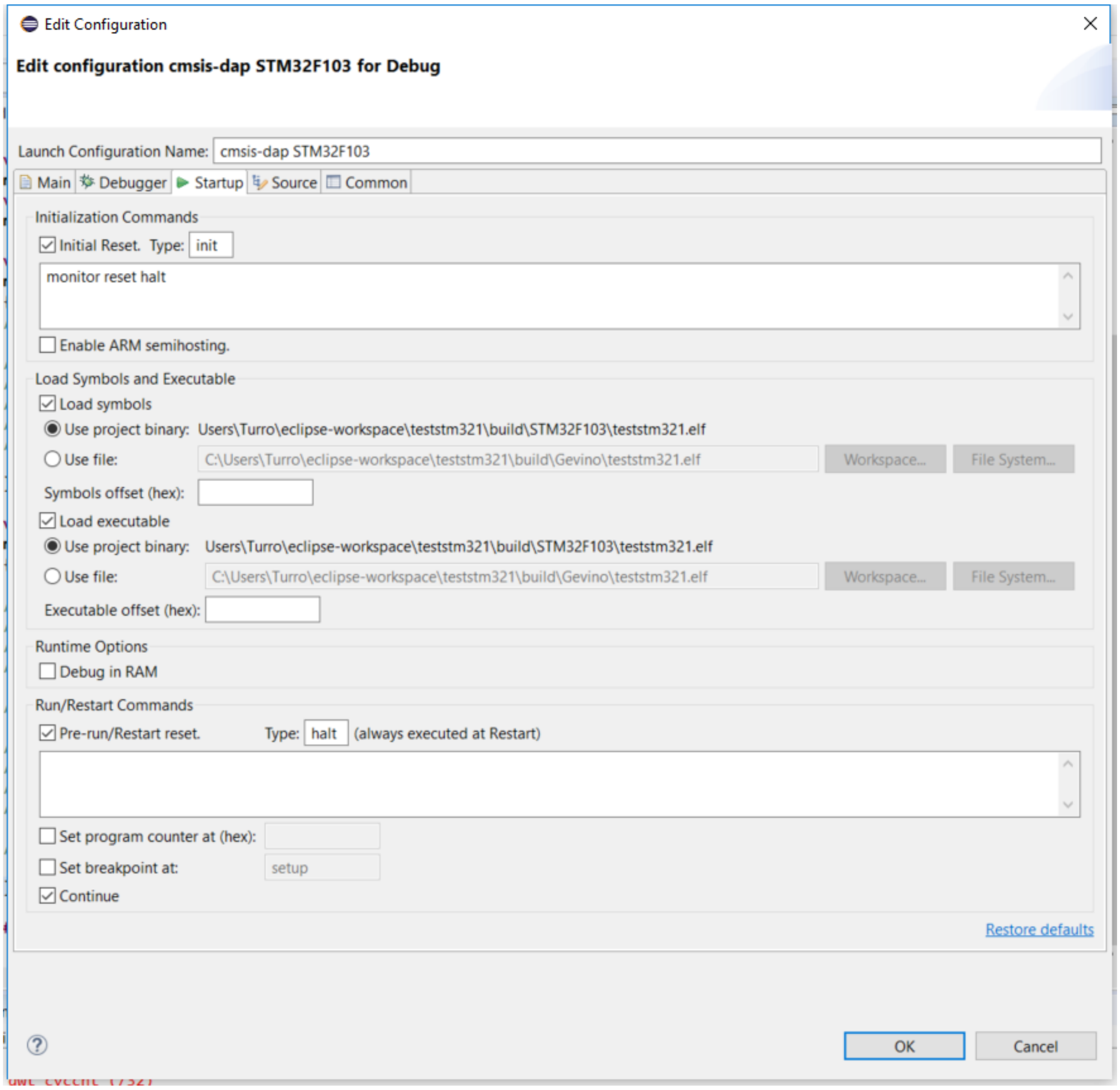
```
set mem inaccessible-by-default off
set remotetimeout 20
```

Remote Target

Host name or IP address: localhost

Port number: 3333

□ Force thread list update on suspend



Edit Configuration

Edit configuration cmsis-dap STM32F103 for Debug

Launch Configuration Name: cmsis-dap STM32F103

Initialization Commands

☑ Initial Reset. Type: init

```
monitor reset halt
```

☐ Enable ARM semihosting.

Load Symbols and Executable

☑ Load symbols

● Use project binary: Users\Turro\eclipse-workspace\teststm321\build\STM32F103\teststm321.elf

○ Use file: C:\Users\Turro\eclipse-workspace\teststm321\build\Gevino\teststm321.elf

Symbols offset (hex):

☑ Load executable

● Use project binary: Users\Turro\eclipse-workspace\teststm321\build\STM32F103\teststm321.elf

○ Use file: C:\Users\Turro\eclipse-workspace\teststm321\build\Gevino\teststm321.elf

Executable offset (hex):

Runtime Options

☐ Debug in RAM

Run/Restart Commands

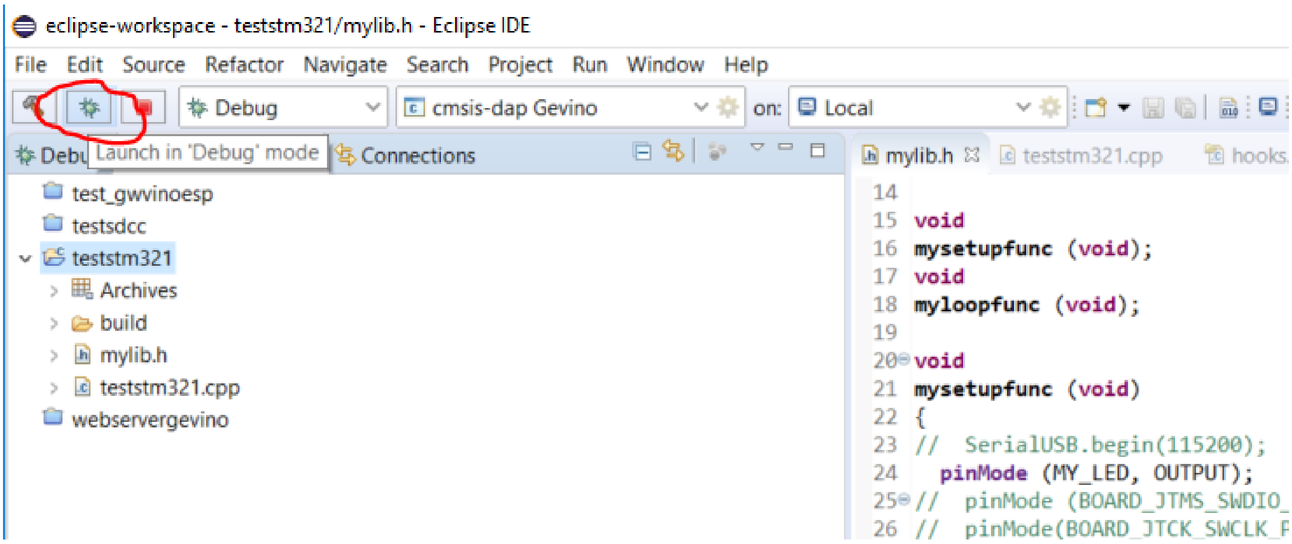
☑ Pre-run/Restart reset. Type: halt (always executed at Restart)

☐ Set program counter at (hex):

☐ Set breakpoint at: setup

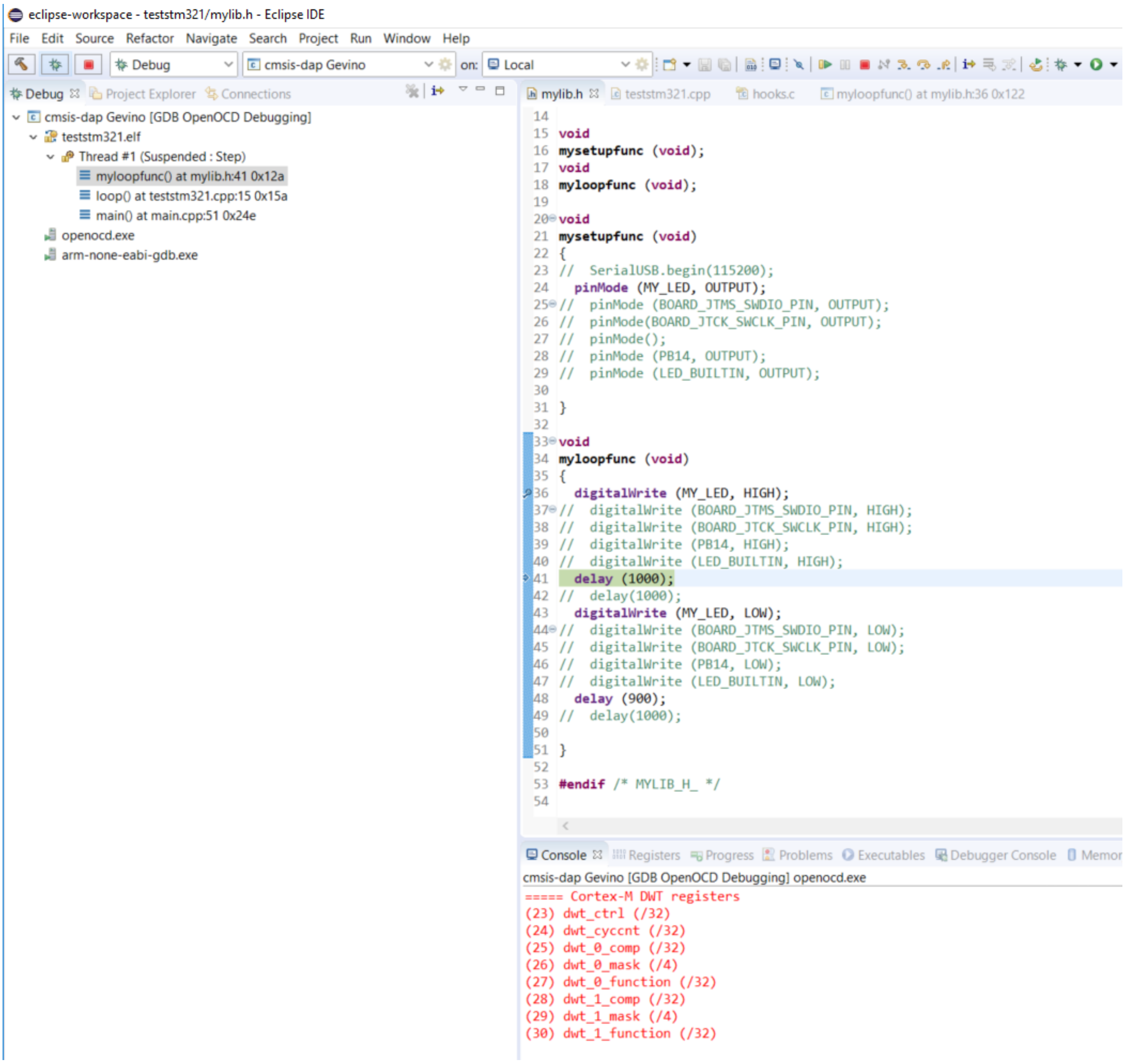
☑ Continue

That’s it, be sure the current project is selected then switch to the debug configuration You need, by clicking debug button You flash the code and now You can debug with variables watching, instruction steps and so.



eclipse-workspace - teststm321/mylib.h - Eclipse IDE

```

void
mysetupfunc (void);
void
myloopfunc (void);

void
mysetupfunc (void)
{
//  SerialUSB.begin(115200);
  pinMode (MY_LED, OUTPUT);
//  pinMode (BOARD_JTMS_SWDIO_
//  pinMode(BOARD_JTCK_SWCLK_F
```



```
cmsis-dap Gevino [GDB OpenOCD Debugging]
  teststm321.elf
    Thread #1 (Suspended : Step)
      myloopfunc() at mylib.h:41 0x12a
      loop() at teststm321.cpp:15 0x15a
      main() at main.cpp:51 0x24e
  openocd.exe
  arm-none-eabi-gdb.exe
```

```

void
mysetupfunc (void);
void
myloopfunc (void);

void
mysetupfunc (void)
{
//  SerialUSB.begin(115200);
  pinMode (MY_LED, OUTPUT);
//  pinMode (BOARD_JTMS_SWDIO_PIN, OUTPUT);
//  pinMode(BOARD_JTCK_SWCLK_PIN, OUTPUT);
//  pinMode();
//  pinMode (PB14, OUTPUT);
//  pinMode (LED_BUILTIN, OUTPUT);

}

void
myloopfunc (void)
{
  digitalWrite (MY_LED, HIGH);
//  digitalWrite (BOARD_JTMS_SWDIO_PIN, HIGH);
//  digitalWrite (BOARD_JTCK_SWCLK_PIN, HIGH);
//  digitalWrite (PB14, HIGH);
//  digitalWrite (LED_BUILTIN, HIGH);
  delay (1000);
//  delay(1000);
  digitalWrite (MY_LED, LOW);
//  digitalWrite (BOARD_JTMS_SWDIO_PIN, LOW);
//  digitalWrite (BOARD_JTCK_SWCLK_PIN, LOW);
//  digitalWrite (PB14, LOW);
//  digitalWrite (LED_BUILTIN, LOW);
  delay (900);
//  delay(1000);

}

#endif /* MYLIB_H_ */

```

```
cmsis-dap Gevino [GDB OpenOCD Debugging] openocd.exe
===== Cortex-M DWT registers
(23) dwt_ctrl (/32)
(24) dwt_cyccnt (/32)
(25) dwt_0_comp (/32)
(26) dwt_0_mask (/4)
(27) dwt_0_function (/32)
(28) dwt_1_comp (/32)
(29) dwt_1_mask (/4)
(30) dwt_1_function (/32)
```

How to get a CMSIS-DAP:

I had a couple of st-link v2 clones, I found quite easy (and cheap) converting one of them to CMSIS-DAP by following these instructions: cmsis-dap-on-a-cheap-st-link-v2-mini-adapter

When converted to CMSIS-DAP it works even on STM32F103.

I tried BlackMagicProbe, it works quite well on Gevino but I wasn't able to get it working on STM32F103.