

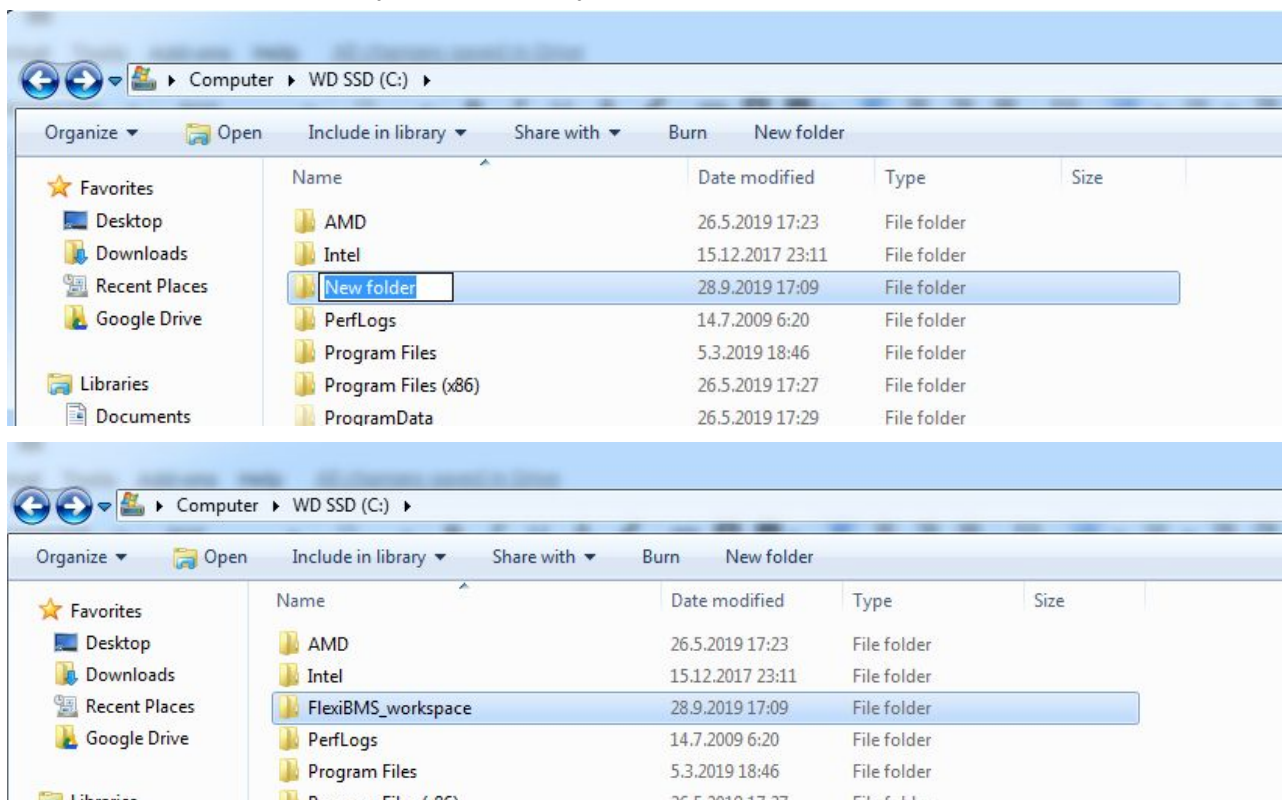
How to setup coding IDE for FlexiBMS

Programs (all free, might need account creation):

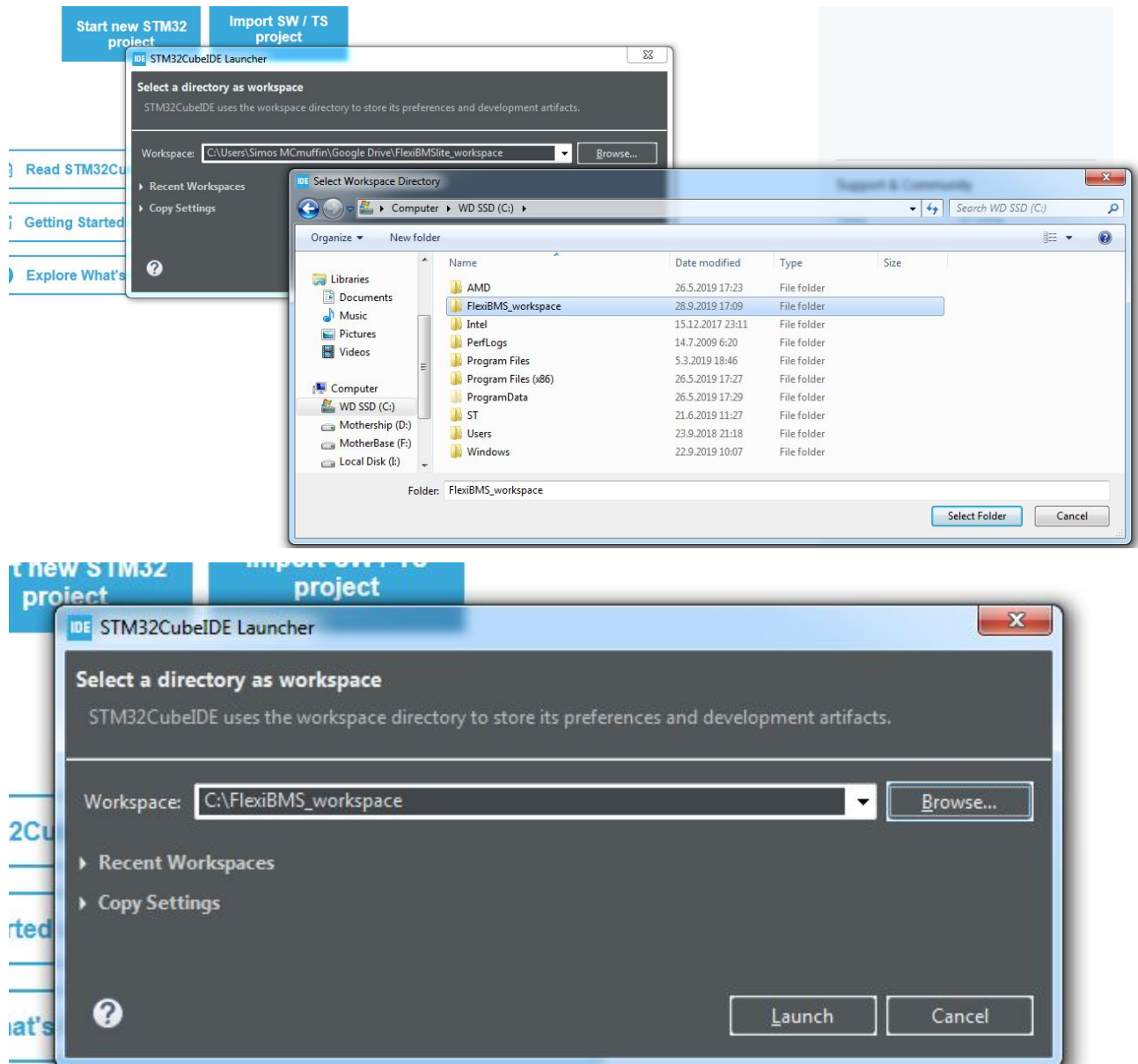
- **STM32cubeIDE, *mandatory*, programming IDE**
 - <https://www.st.com/en/development-tools/stm32cubeide.html>
- STM32 ST-LINK Utility, *optional*, useful flashing and testing tool, can test SWD connection to MCU and flash .hex .bin -files and do chip erases
 - <https://www.st.com/en/development-tools/stsw-link004.html>
- Git, *optional*, needed/recommended if you're going to be committing code to github
 - <https://www.git-scm.com/downloads>

Step-by-step

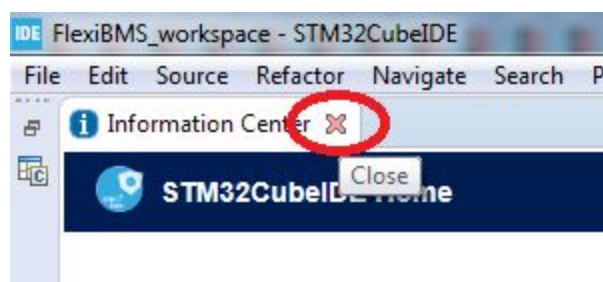
1. Download and install STM32cubeIDE (hereinafter called just "IDE") and optionally STM32 ST-LINK utility and Git.
2. Before launching IDE, let's create a workspace folder for it. (I'm creating it in the root of C:\ for demonstration purposes, you can create yours for example in the IDE installation folder)



- Open IDE and if it's your first time launching it, it should ask for the workspace folder, navigate to the created folder and select it. If you skipped this or have already selected an earlier workspace you can change it from **File -> Switch Workspace -> Other**.



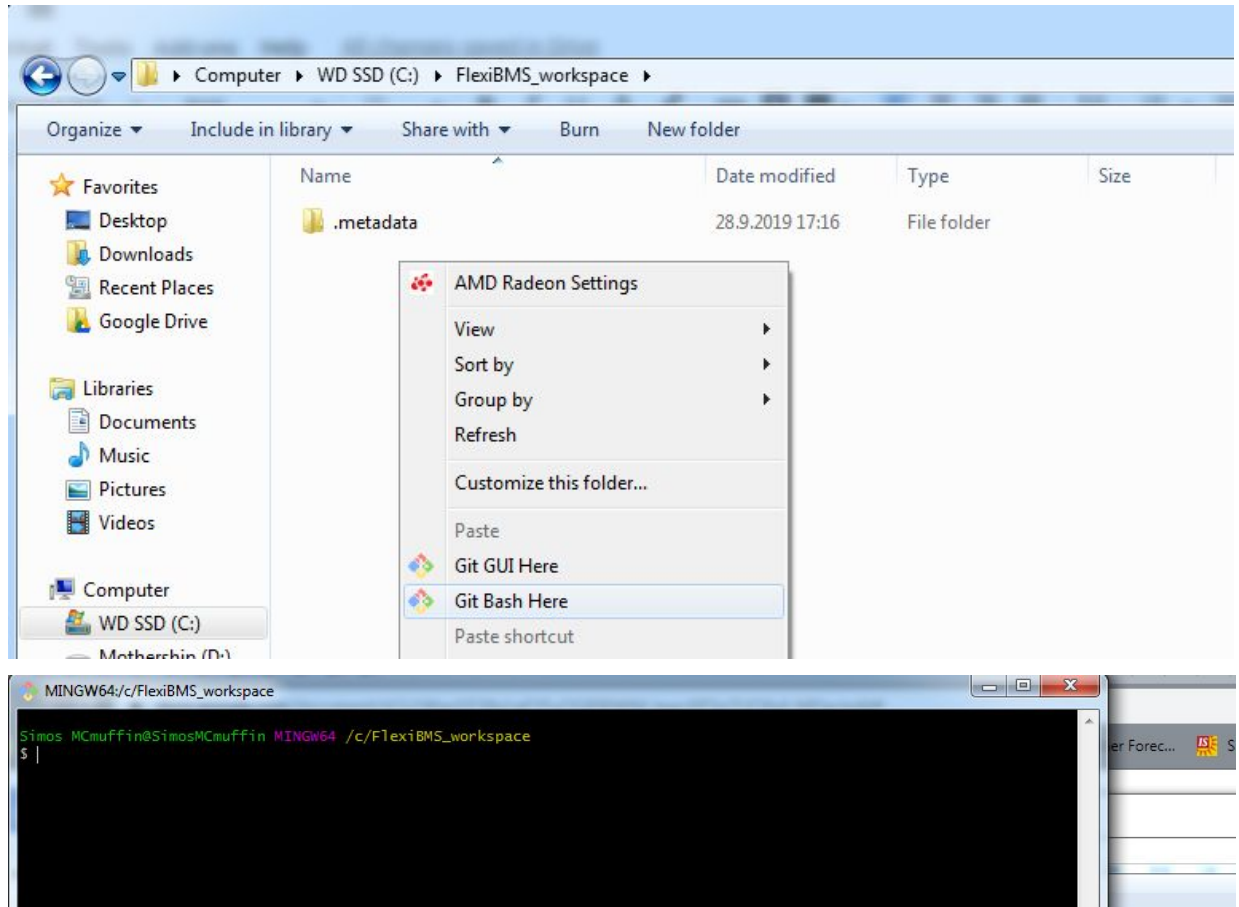
- Close the information center view.



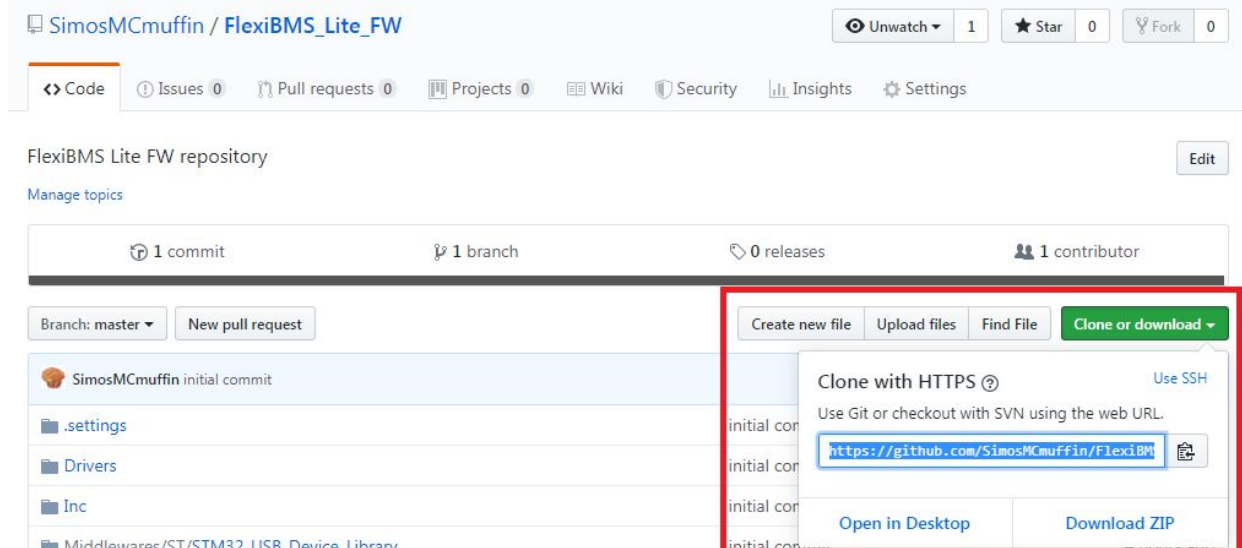
- Now, we need to get the project files from the github repo, so we have something to import into the IDE, so head over to https://github.com/SimosMCmuffin/FlexiBMS_Lite_FW and keep it open.

Using Git:

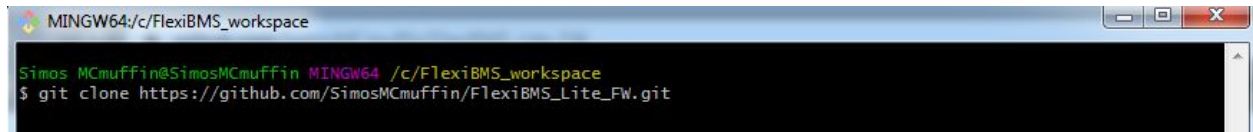
- Go to the IDE workspace folder and open up Git Bash.



- Get the cloning link from the github page.



3. Enter “git clone LINK” in the git bash and enter.

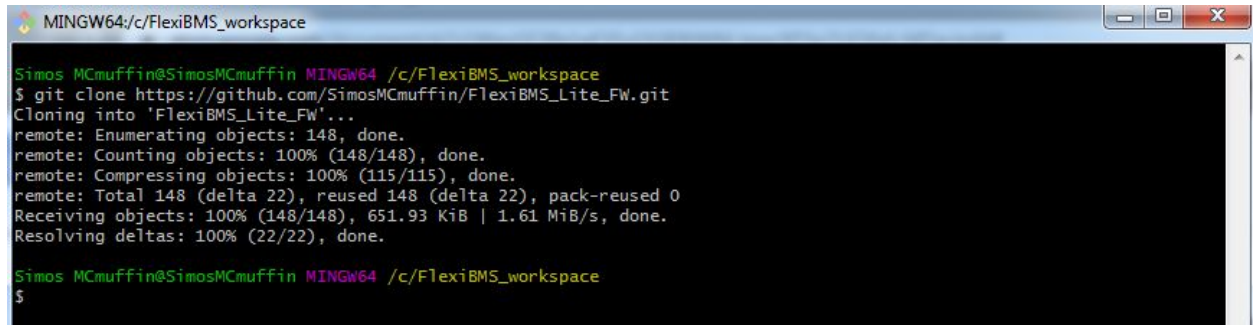


```

MINGW64/c/FlexiBMS_workspace
Simos MCmuffin@SimosMCmuffin MINGW64 /c/FlexiBMS_workspace
$ git clone https://github.com/SimosMCmuffin/FlexiBMS_Lite_FW.git

```

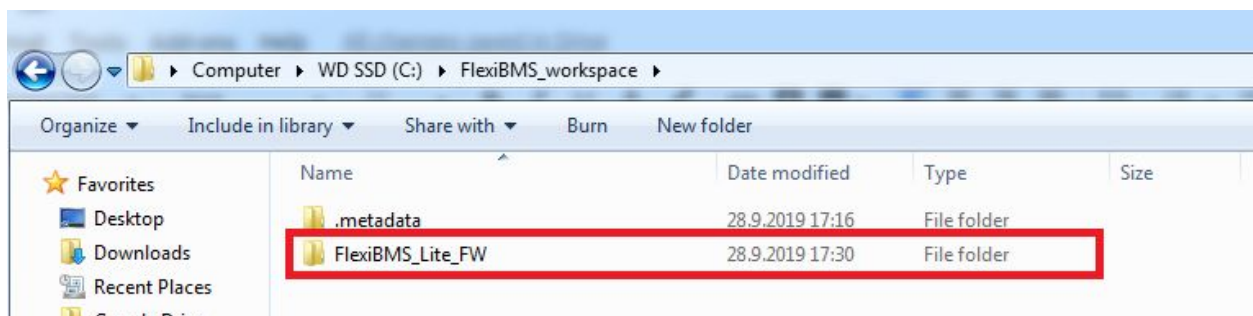
4. This will download the github repo and create a folder for it.



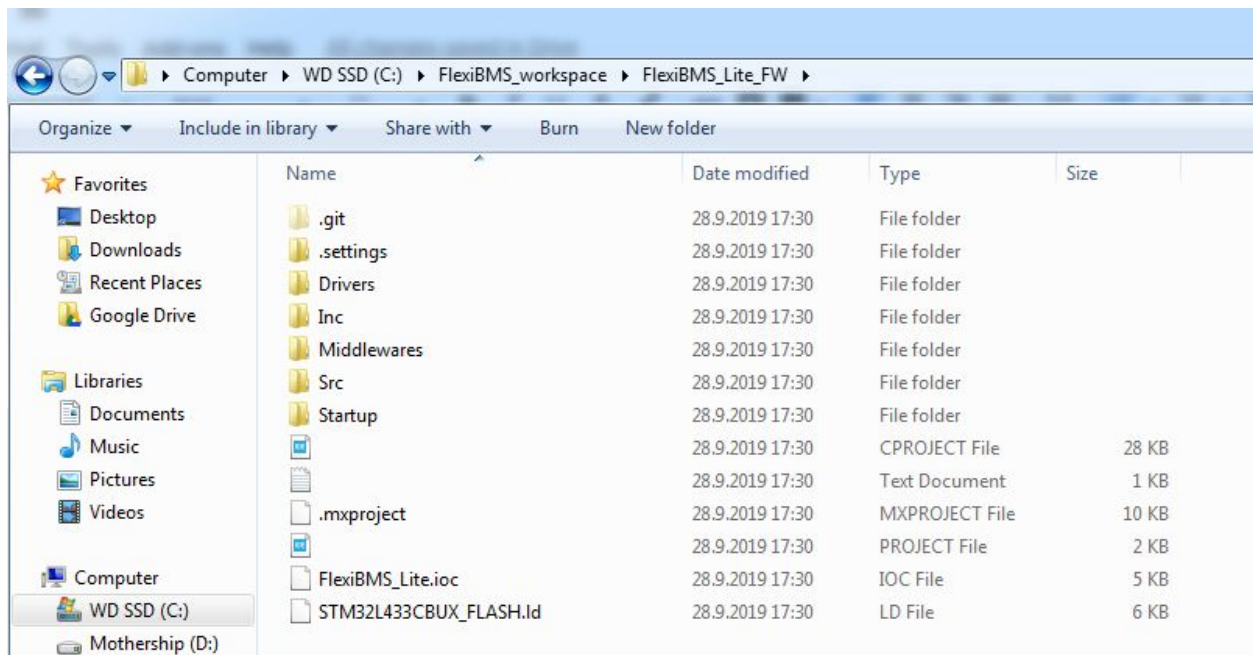
```

MINGW64/c/FlexiBMS_workspace
Simos MCmuffin@SimosMCmuffin MINGW64 /c/FlexiBMS_workspace
$ git clone https://github.com/SimosMCmuffin/FlexiBMS_Lite_FW.git
Cloning into 'FlexiBMS_Lite_FW'...
remote: Enumerating objects: 148, done.
remote: Counting objects: 100% (148/148), done.
remote: Compressing objects: 100% (115/115), done.
remote: Total 148 (delta 22), reused 148 (delta 22), pack-reused 0
Receiving objects: 100% (148/148), 651.93 KiB | 1.61 MiB/s, done.
Resolving deltas: 100% (22/22), done.
Simos MCmuffin@SimosMCmuffin MINGW64 /c/FlexiBMS_workspace
$

```

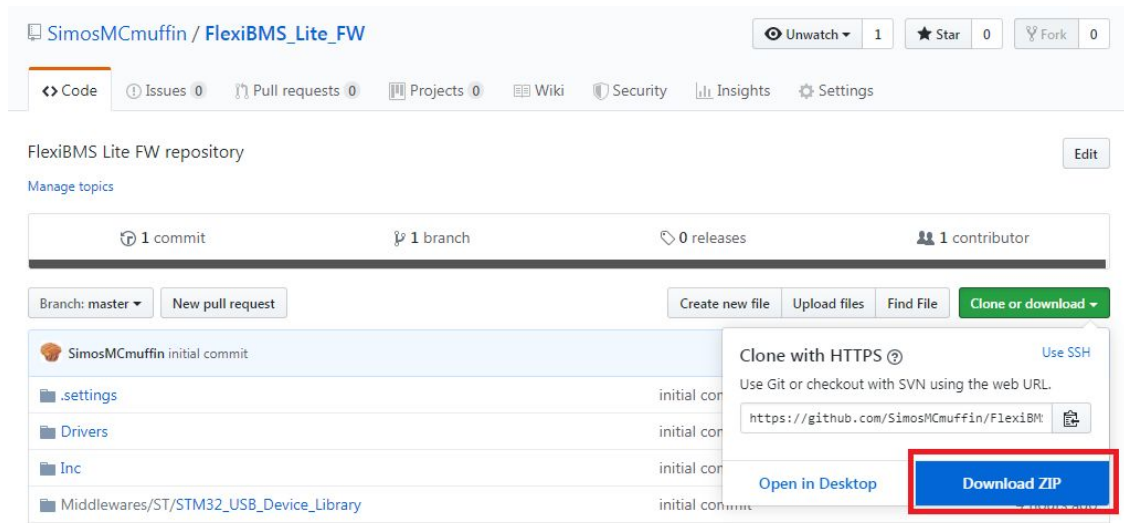


5. Project files should be in the created folder and you're done.

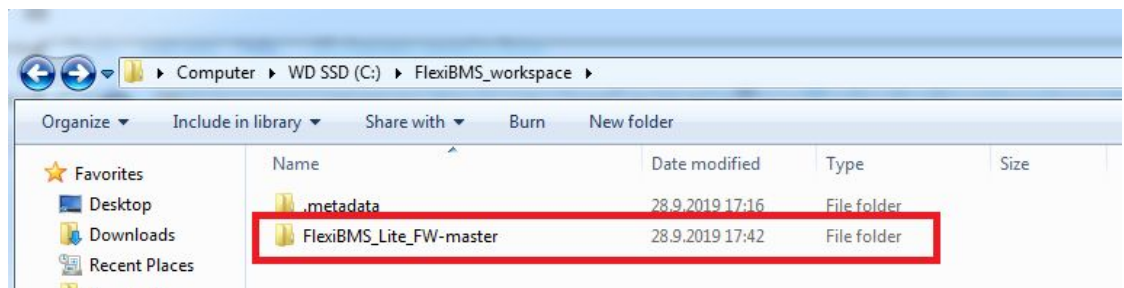


Manually downloading:

1. Go to the github page and download the source files as a .ZIP.



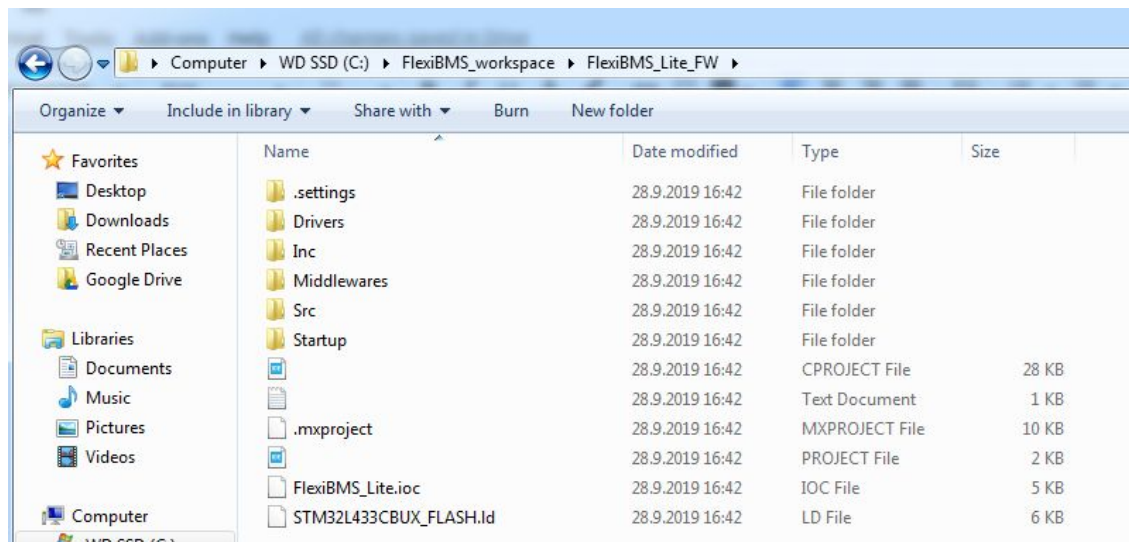
2. Extract the files and move the extracted **folder** over to IDE workspace folder.



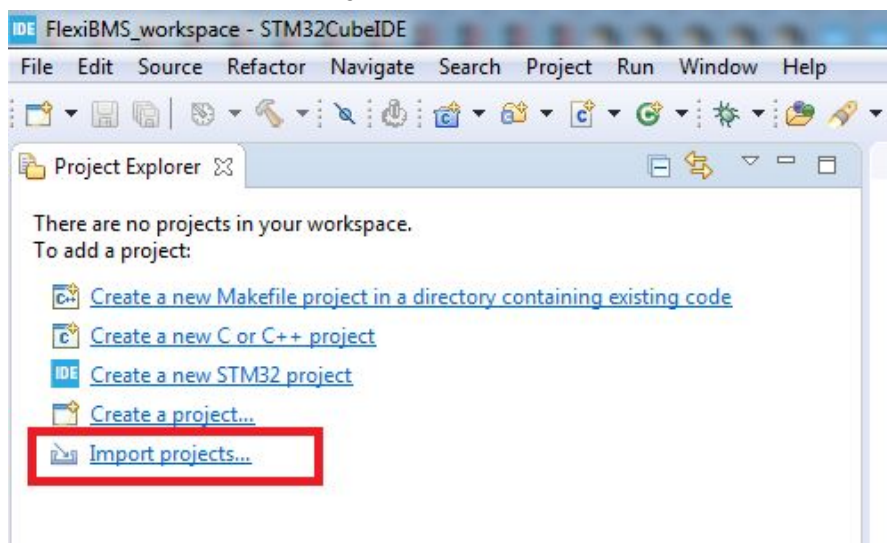
3. Remove the "-master" end from the folder name.



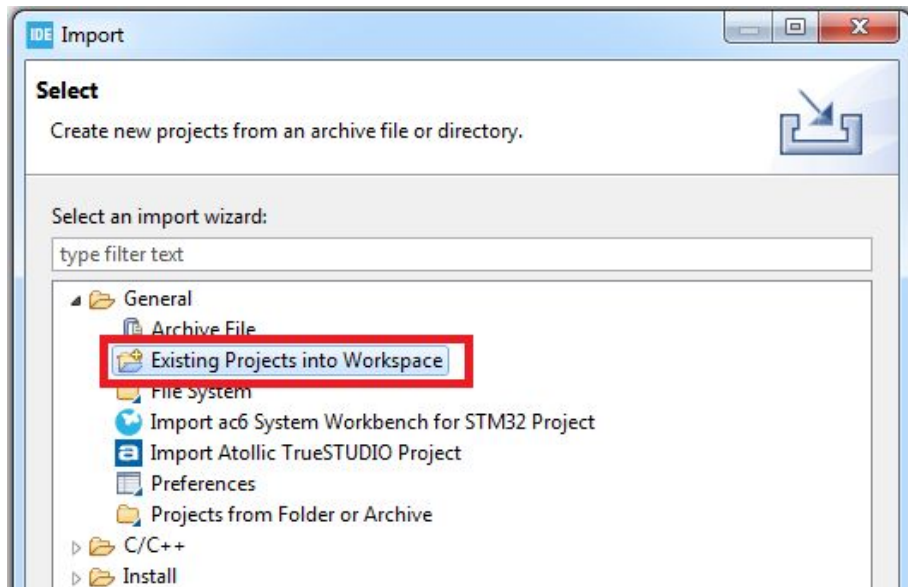
4. Project files are now present. Done.



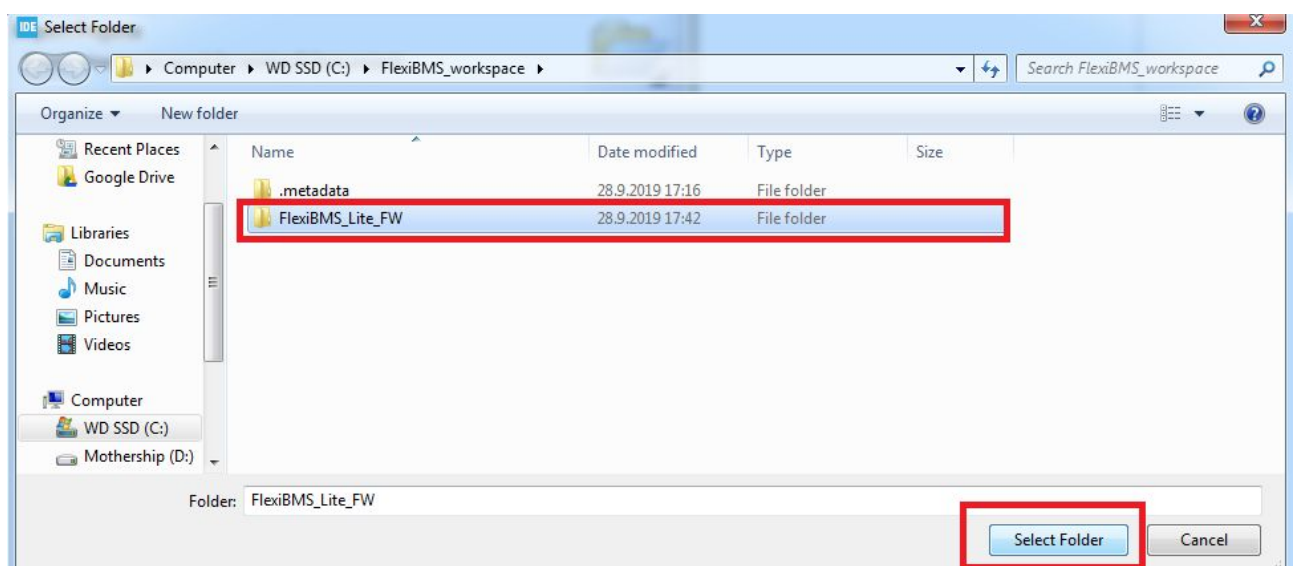
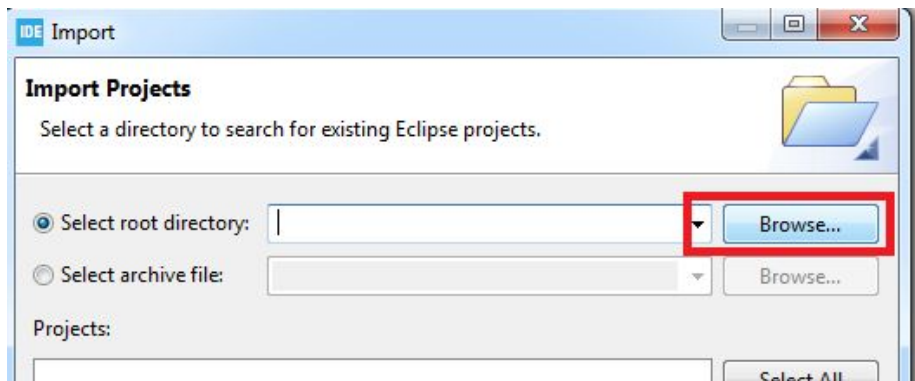
6. Now we have project files, so we can import the project into the IDE. So jump back to the IDE and select **import projects**.



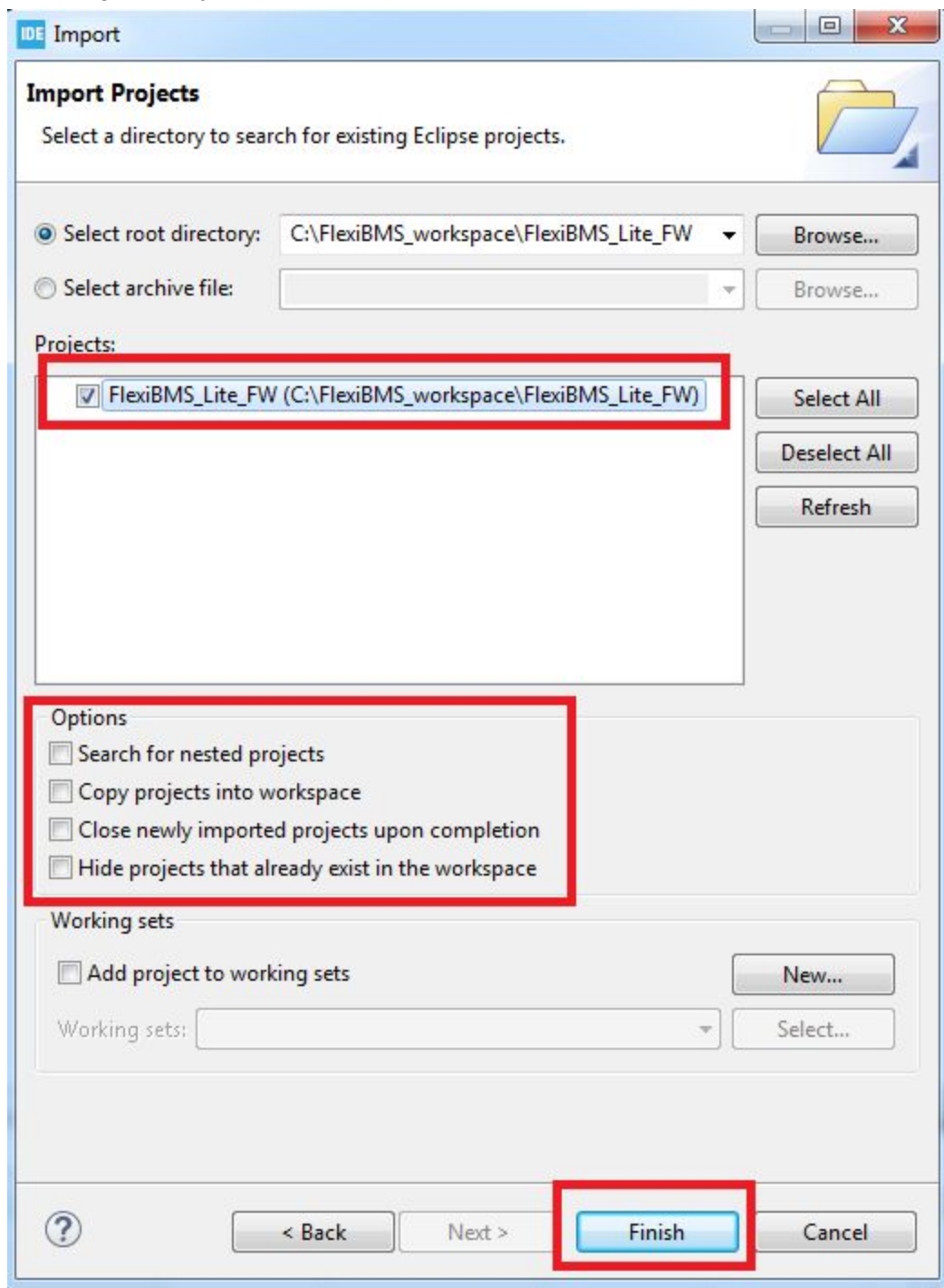
7. General -> Existing Projects into Workspace



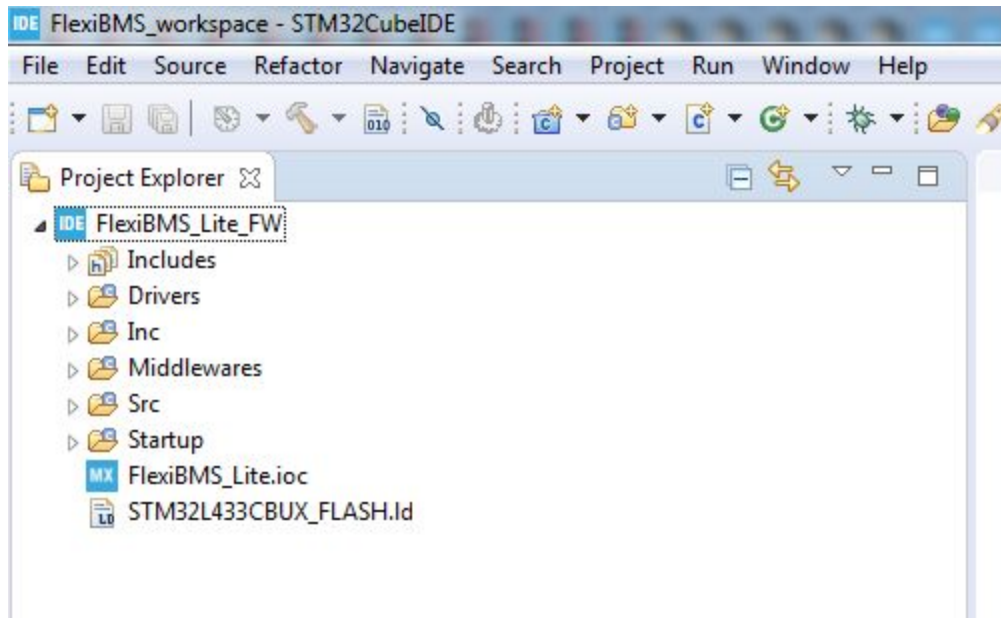
8. In the “Select root directory” navigate to the IDE workspace folder and select it.



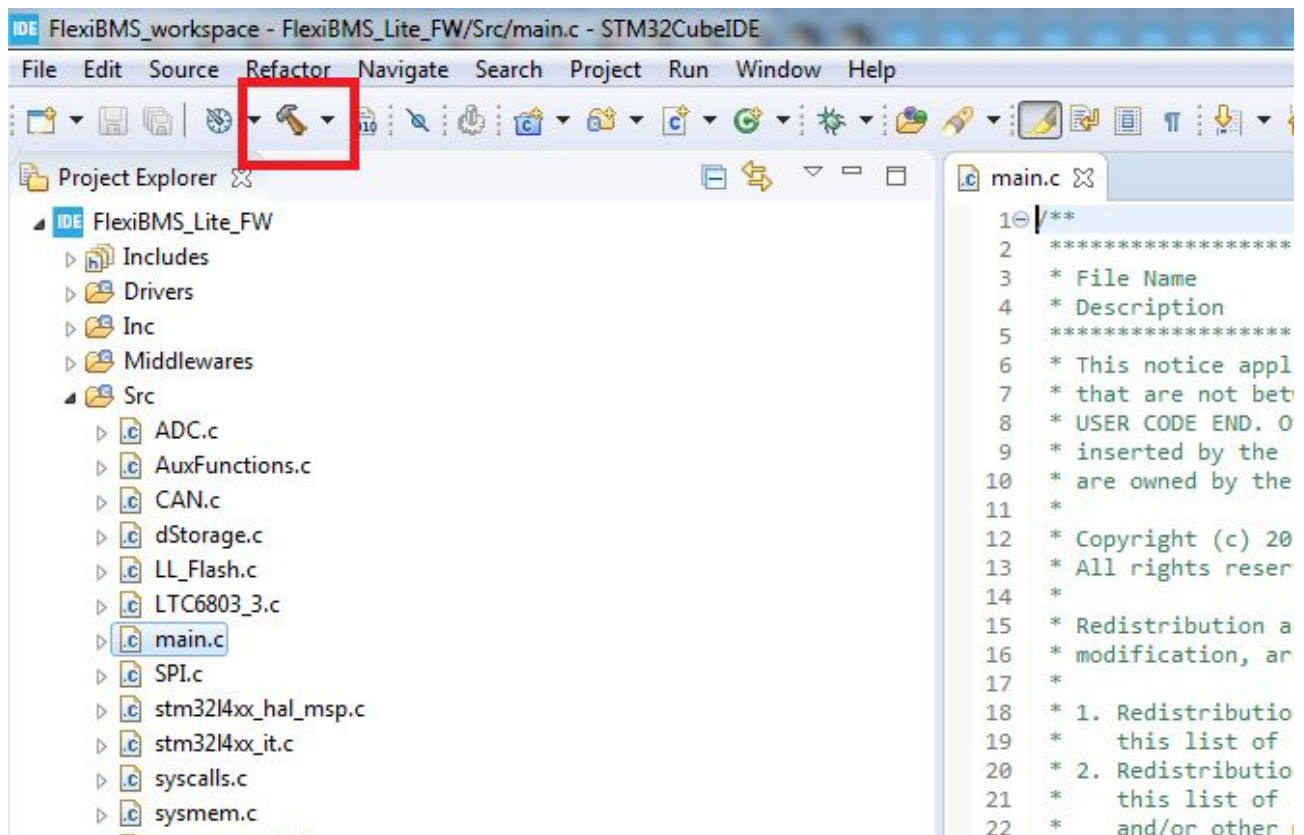
9. The FlexiBMS_Lite_FW should show up in the project listing meaning that it has been detected as an importable project. Select it leave the options un-checked and Finish importing the project.



10. The project should now be in the project explorer.



11. Now the critical part, does it compile? Inside the project structure go to Src-folder and open Main.c file and then click the hammer icon in the top to try building the project.



12. If all goes well, it should compile successfully with only warnings and without errors.

```

35  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
36  * PARTICULAR PURPOSE AND NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY
37  * RIGHTS ARE DISCLAIMED TO THE FULLEST EXTENT PERMITTED BY LAW. IN NO EVENT
38  * SHALL STMICROELECTRONICS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
39  * INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
40  * LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA,
41  * OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
42  * LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
43  * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
44  * EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

```

CDT Build Console [FlexiBMS_Lite_FW]

```

arm-none-eabi-objdump -h -S FlexiBMS_Lite_FW.elf > "FlexiBMS_Lite_FW.list"
text      data      bss      dec      hex filename
47136     508      6408     54052    d324 FlexiBMS_Lite_FW.elf
Finished building: default.size.stdout

Finished building: FlexiBMS_Lite_FW.list

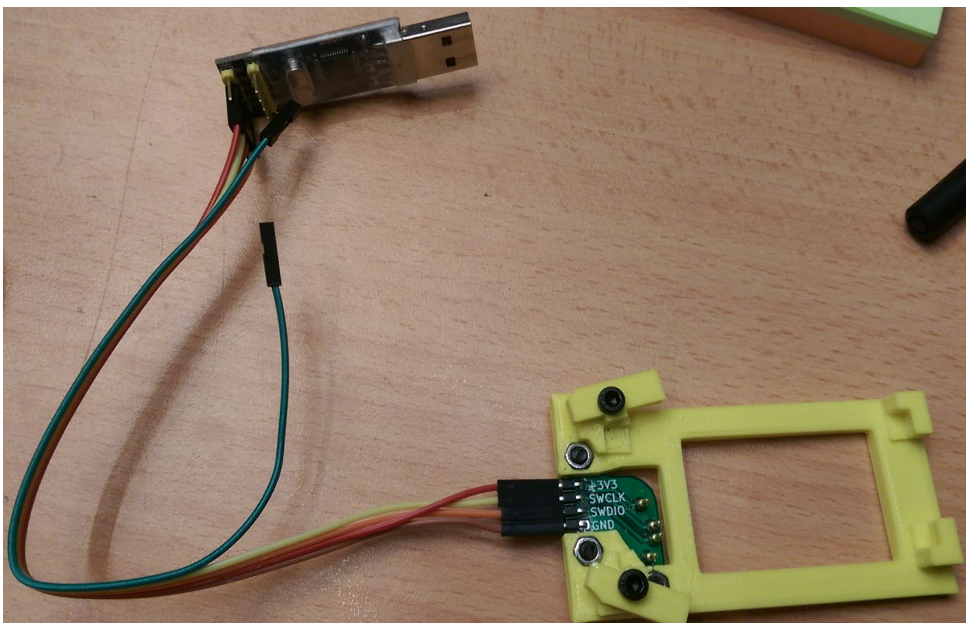
18:13:36 Build Finished. 0 errors, 7 warnings. (took 5s.734ms)

```

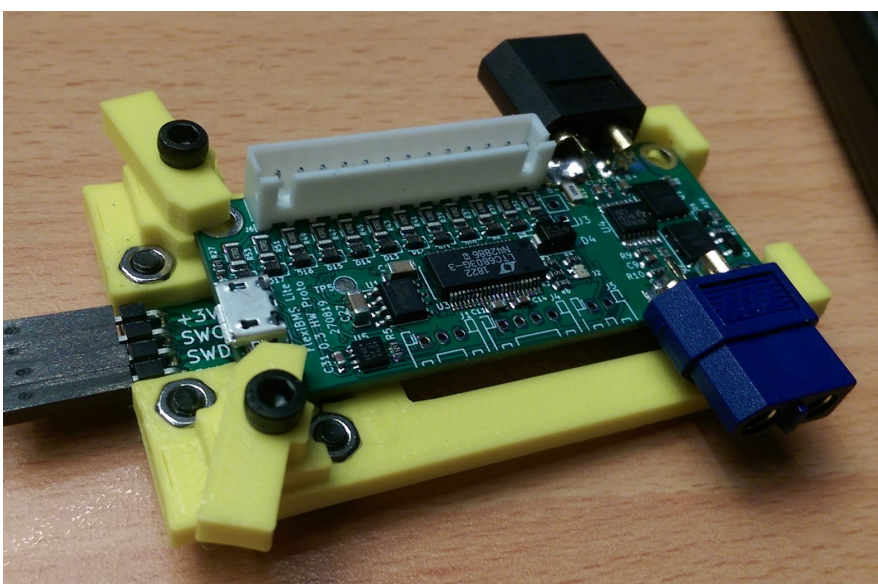
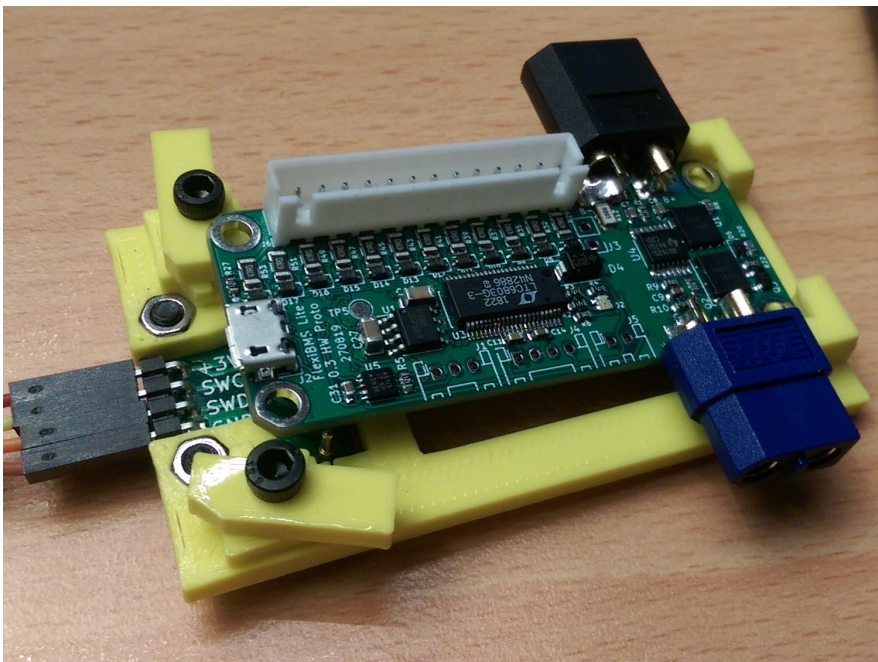
13. If it did compile, then congratulations you have successfully downloaded the project source files and compiled them. Now you can start writing and modifying the code, making changes, testing your code and committing working code.

Connecting programmer to MCU and to the PC

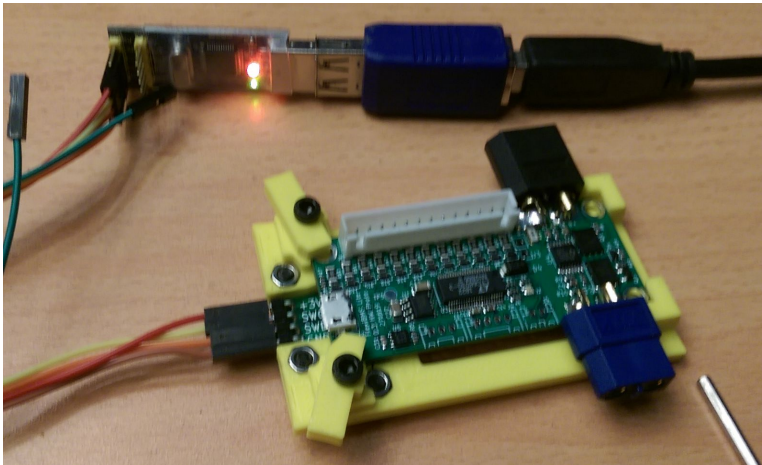
1. Connect ST-LINK to the programming jig, if no battery is connected you'll have to provide +3V3 through the programmer.



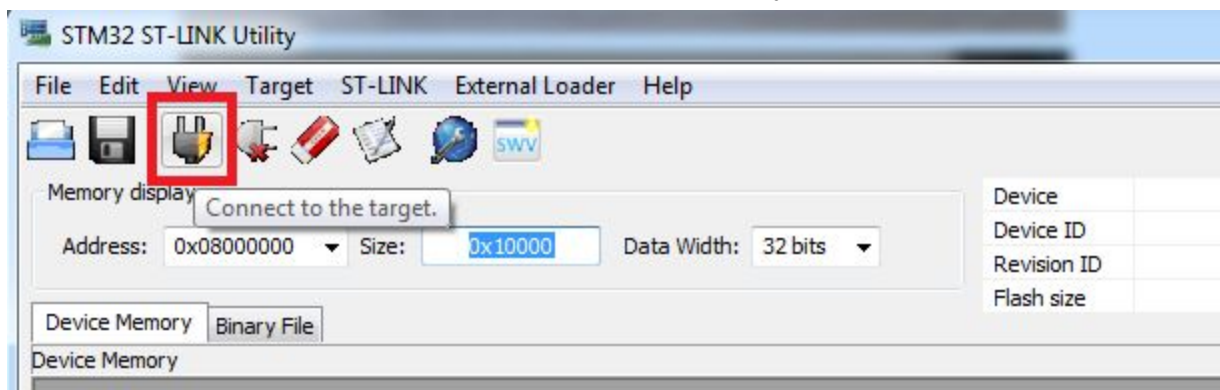
2. Insert the BMS into the jig and lock it down.



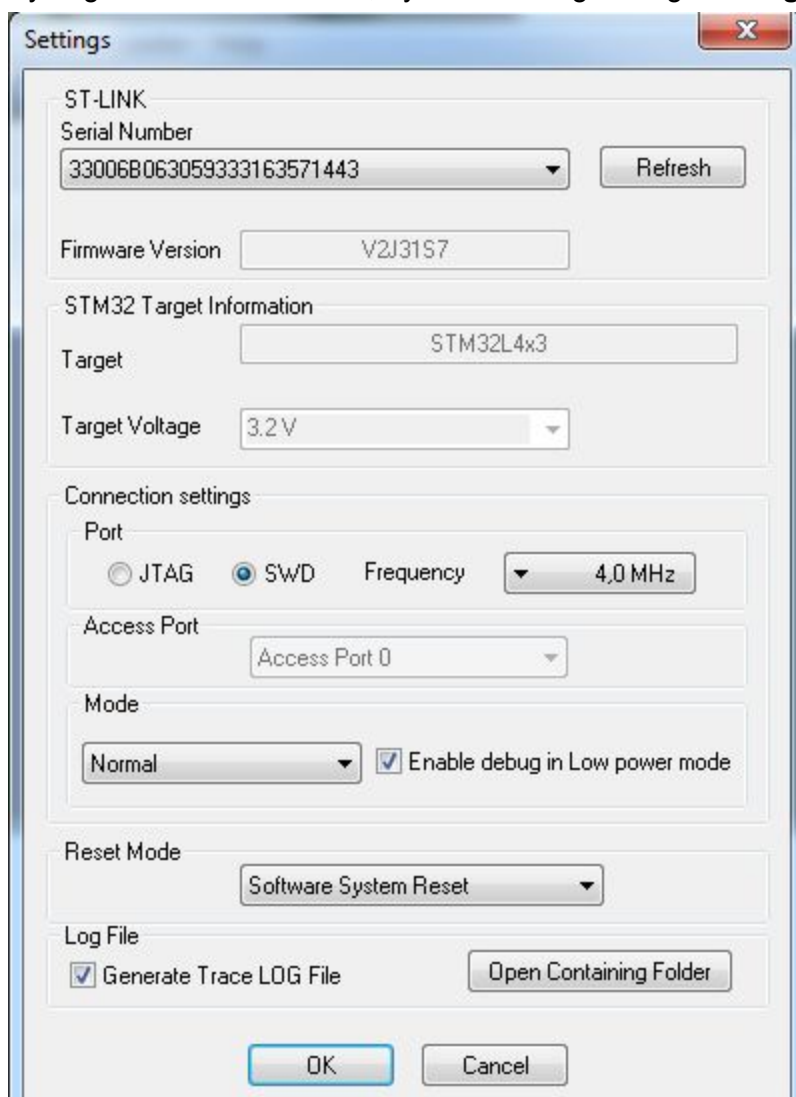
3. Connect programmer to PC.



4. To easily verify that the connection is okay, I like to use STM32 ST-LINK Utility to check that I can connect to the MCU and that it's detected correctly.

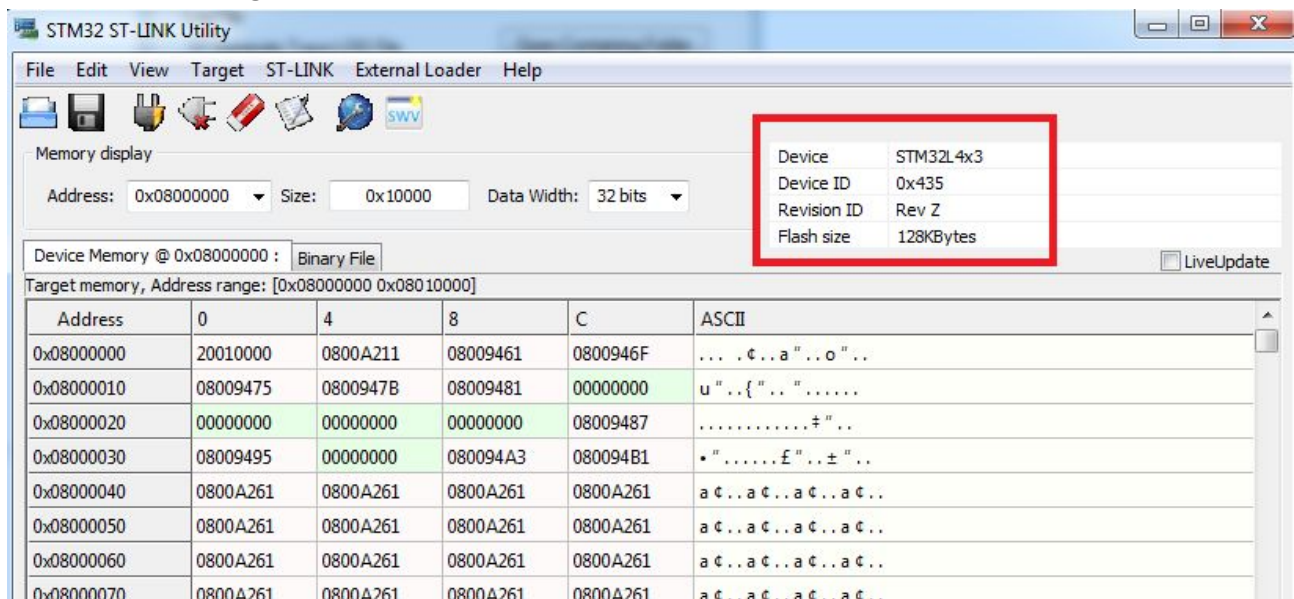


If you get a connection error, try the following settings in **Target -> Settings**



If you still can't get connected, try opening up the locks on the programming jig, pulling and pushing the BMS couple times up and down against the Pogo-pins in case there's a bad connection or some surface oxidation. If still no-bueno then check that the programmer is supplying the +3V3 to the board, if no battery is connected.

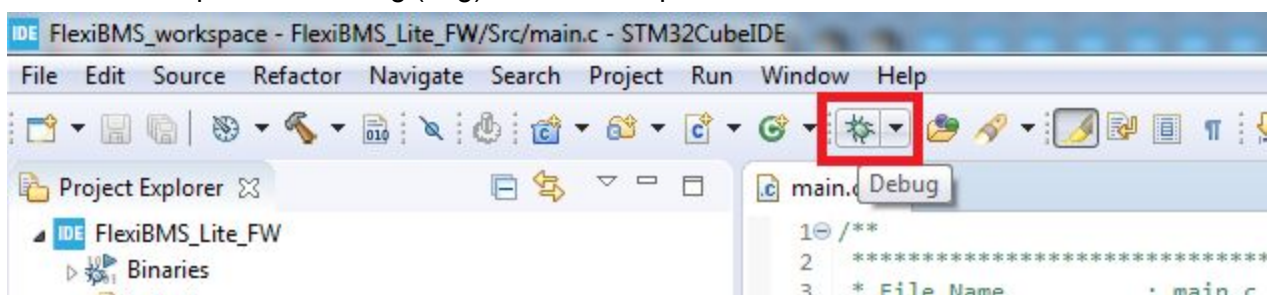
5. Once you're able to connect to the MCU you'll see some data populating the screen and some basic information about the MCU. **Congratulations, you now have a working SWD (Serial-Wire Debug) connection to the MCU.**



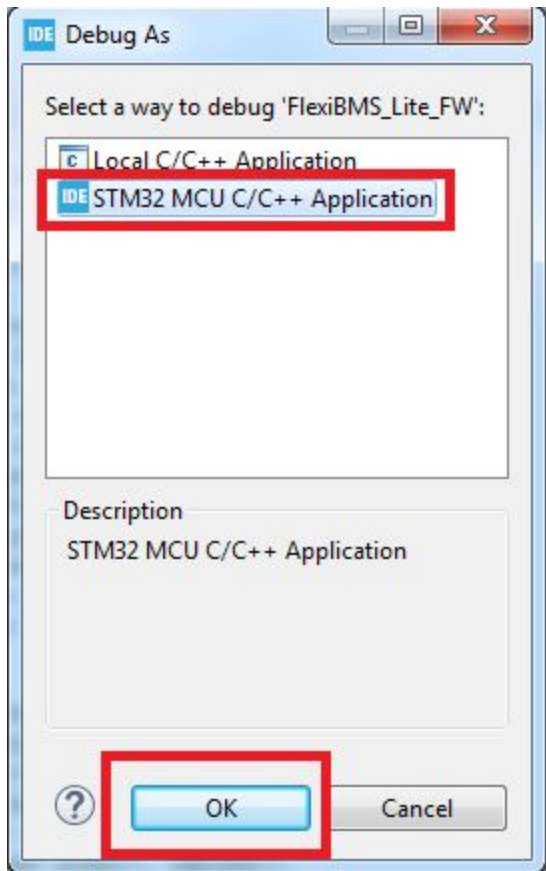
Debugging inside IDE

Once you have verified that your connection to the MCU is okay, you can hop into the IDE and if everything is working as it should and the code compiles successfully via the build (hammer) icon.

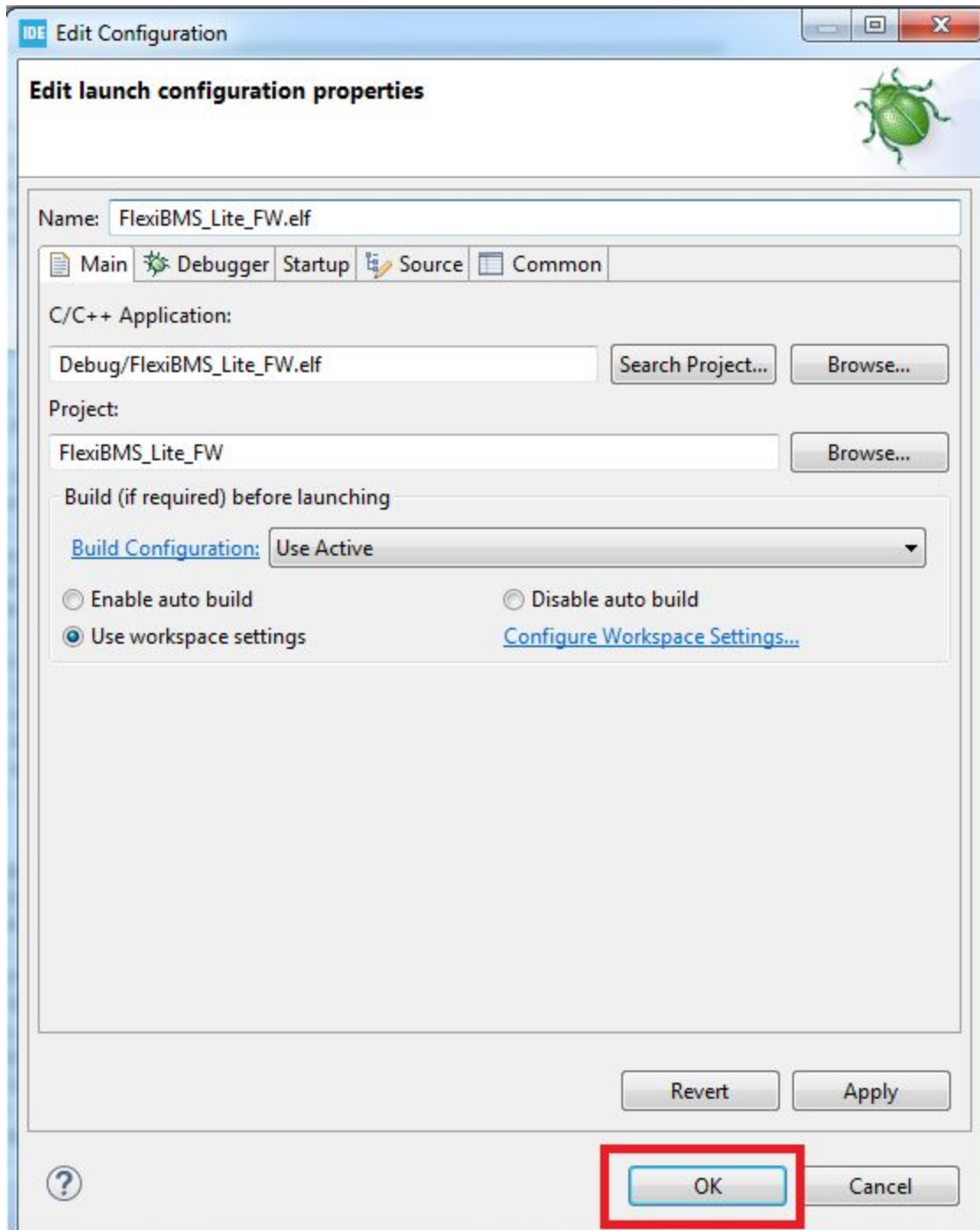
Go ahead and press the debug (bug) icon at the top.



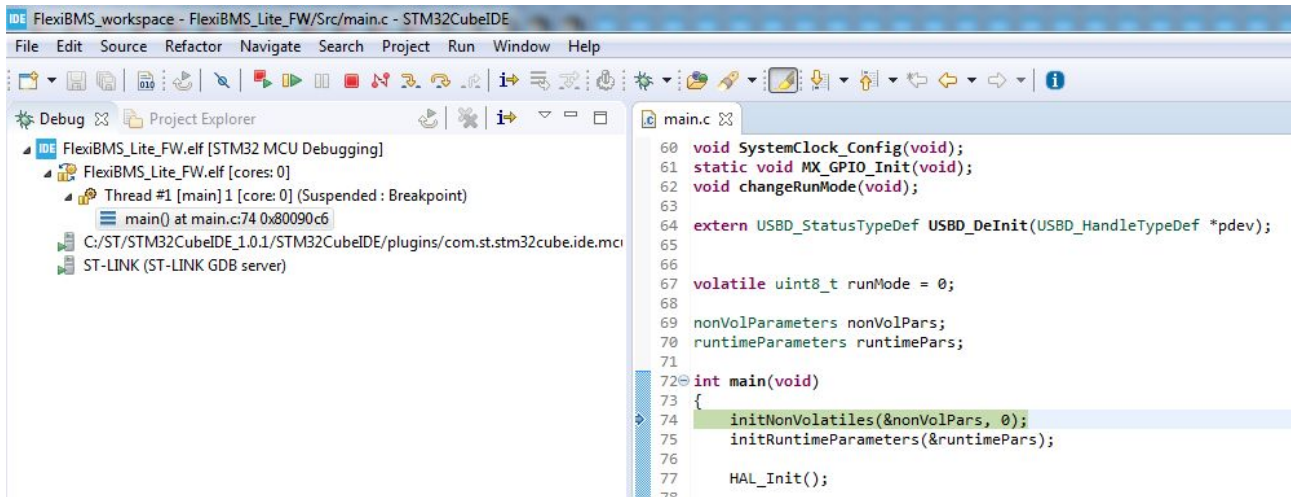
If your project has been freshly imported into the IDE then you might need to initialize the debugging settings. We want to debug as “STM32 MCU C/C++ Application”.



All the default settings should work, so go ahead and just press OK.



The IDE will ask if you'll allow to switch to debugging perspective, allow it and you should be presented a following view.



You are now debugging the project inside the IDE and are able to execute code line-by-line, set breakpoints in the code, check HW register values and manipulate them if needed and just generally test the code.