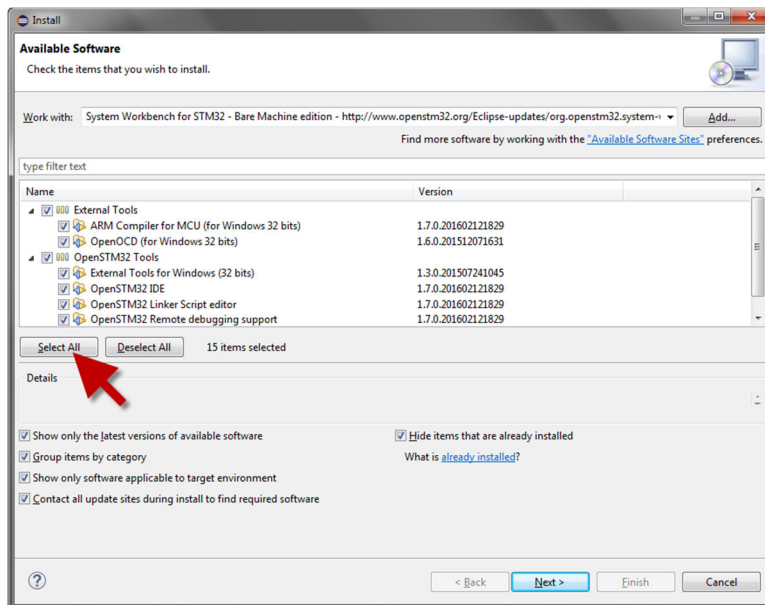


- You should select all of the OpenSTM32 tools on the new window showing available programs and click "Next>"



- Accept the license and System Workbench for STM32 - Bare Metal edition will install itself in your Eclipse setup.
- Restart Eclipse for the tools to be available (Eclipse will automatically suggest the restart.)

6.3 STM USB DRIVERS

- If you have any issues installing the STM drivers you can manually install them as following:
- Install the USB drivers needed by the board from the STMicroelectronics website:
<http://www.st.com/web/en/catalog/tools/PF260219>
 - Scroll down to the bottom of the site
 - Under "Sample & Buy" click the download button
 - The file will be a **ZIP** file
 - Extract all files and **stlink_winusb_install.bat** to install the drivers
 - This will take some time to install...

7 ENVIRONMENT SETUP

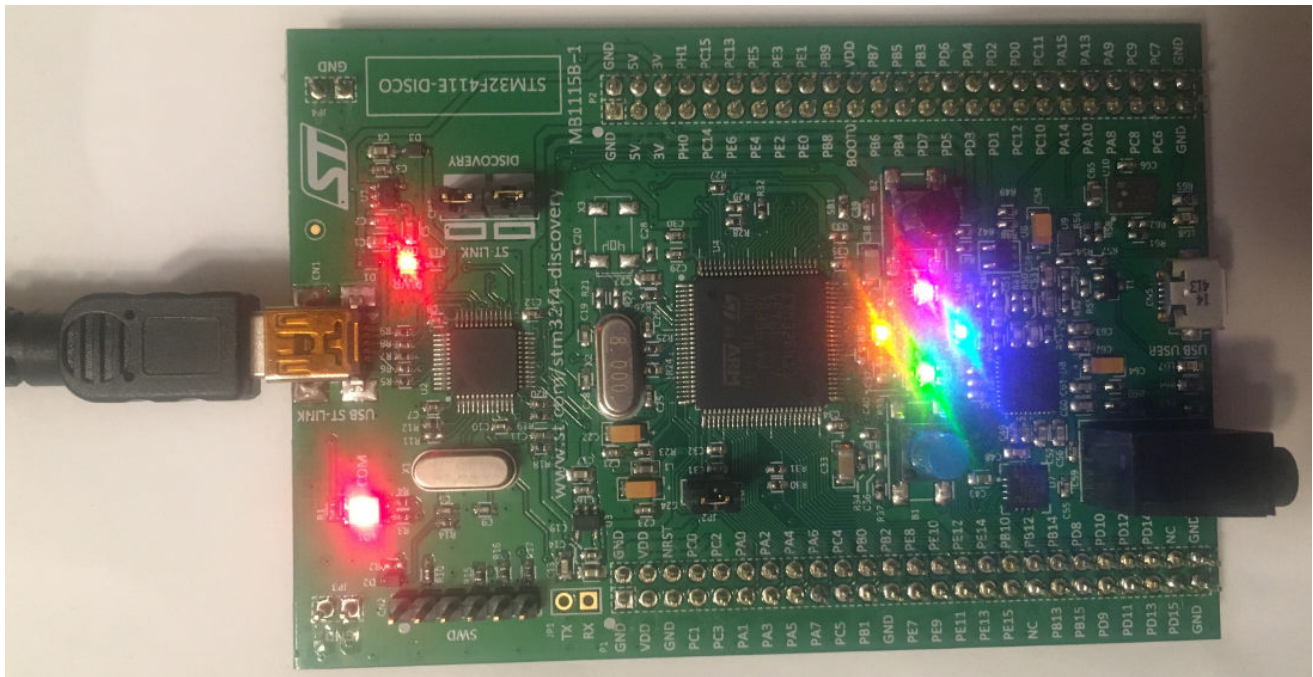
Until now you should have successfully downloaded and installed Eclipse.

Also you should have installed from Eclipse installer the USB drivers needed for your board.

You should also have installed from within Eclipse IDE, the STM32 tool set along with the openOCD and compilers.

7.1 CONNECTING THE ST BOARD

Plug the USB cable to your board



The LEDs on your board should now be blinking.

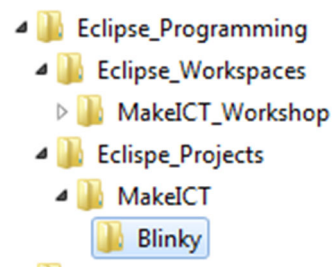
- Push the **BLUE** button and the LEDs will go out.
 - Shake the board and the LEDs will light up in response to the movements.
 - Push the blue button again and the LEDs will blink

Now this is the initial program the development board comes pre installed. We will now create a new project in Eclipse and download our first program Blinky, the hardware equivalent to “Hello World”.

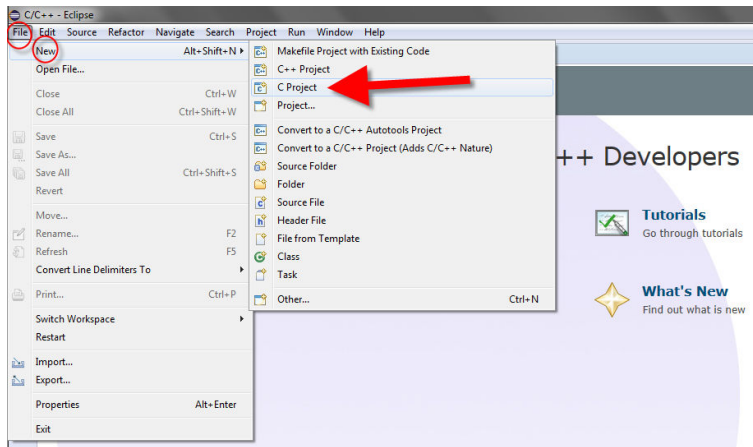
The next steps will guide you to set up the details for the specific board that we are using on Eclipse.

Once we complete this successfully we will have a tool chain that we can use to start writing our code

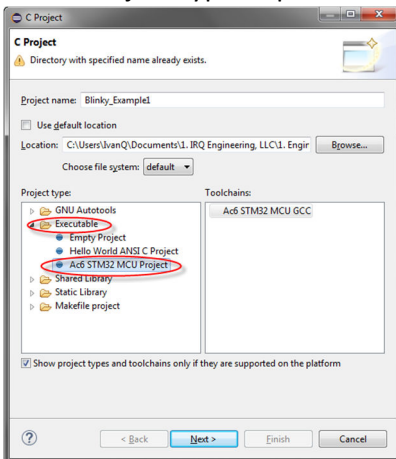
- Before you start anything create the following folder structure:



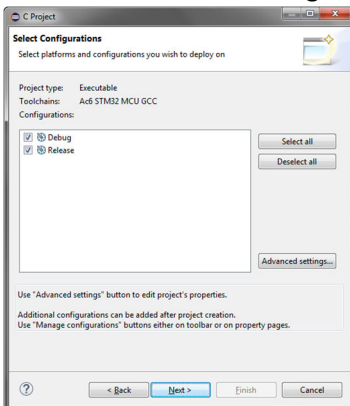
- Start new project:



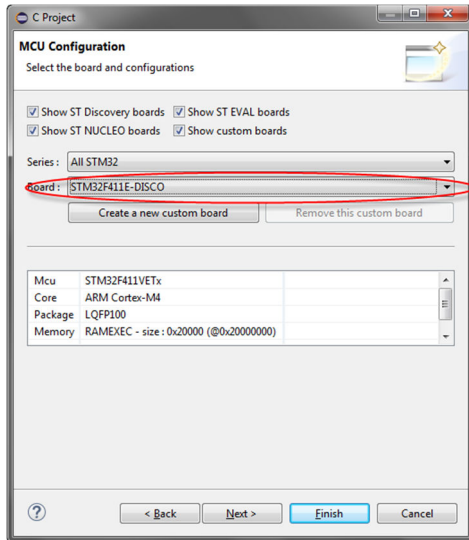
- Under Project Type: expand Executable and click on Ac6 STM32 MCU Project



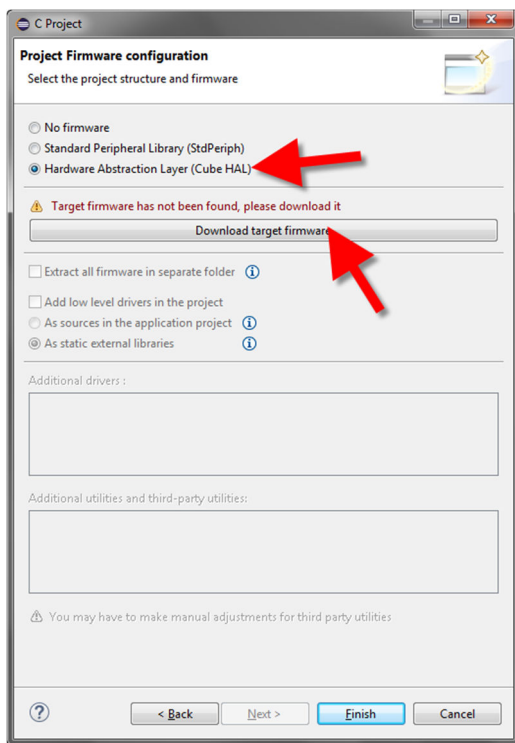
- Click next on the following screen



- Under the Board option find and select STM32F411E-DISCO

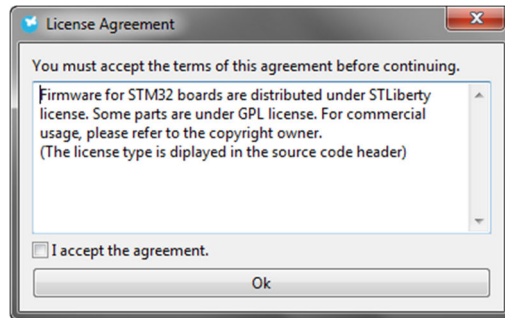


- Make sure that you click “NEXT”, and then select HAL (Hardware Abstraction Layer)”



- Note that the first time there will be a warning that says: “Target firmware has not been found, please download it”

- Click on the download button and accept the terms and conditions



- After clicking OK, the bar at the bottom will start to process the Firmware download. This will take some time.

- Note if the system errors out, try it again as the connection can be lost from time to time

- What have we done here:

- We requested Eclipse to download and use the HAL libraries.

- HAL = Hardware Abstraction Layer

- HAL means that STMicro is providing you with a set of names so that you do not have to poke at the controller registers in order to set up things
- HAL allows to port the code between different microcontrollers, such that you do not have to rewrite code if the register names changes
- HAL is somewhat equivalent to what Arduino does to allow faster programming

- There was also the option to use SPL

- SPL= Standard Peripheral Library: This library is rather interesting for learning as it is more hands on and down to the lower level of the registers
- SPL certainly helps you understand better the microcontroller inner workings
- While SPL is great STMicro has moved to HAL therefore we will use it as part of this class

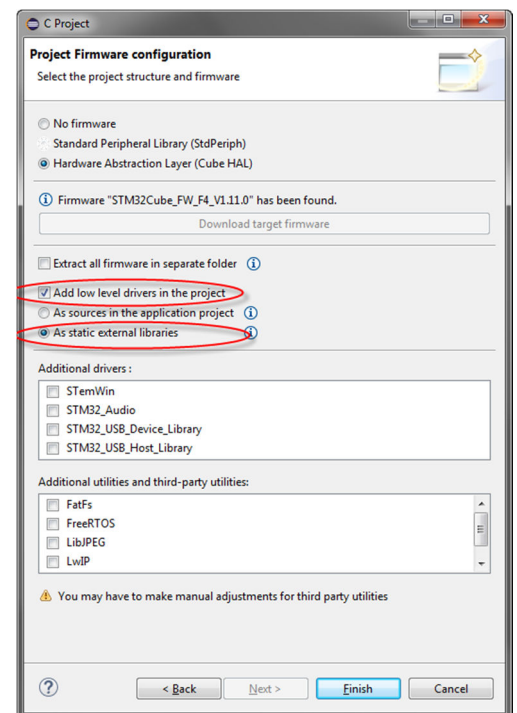
- Select “Add low level drivers in project” and “As static external libraries”.

- Click Finish

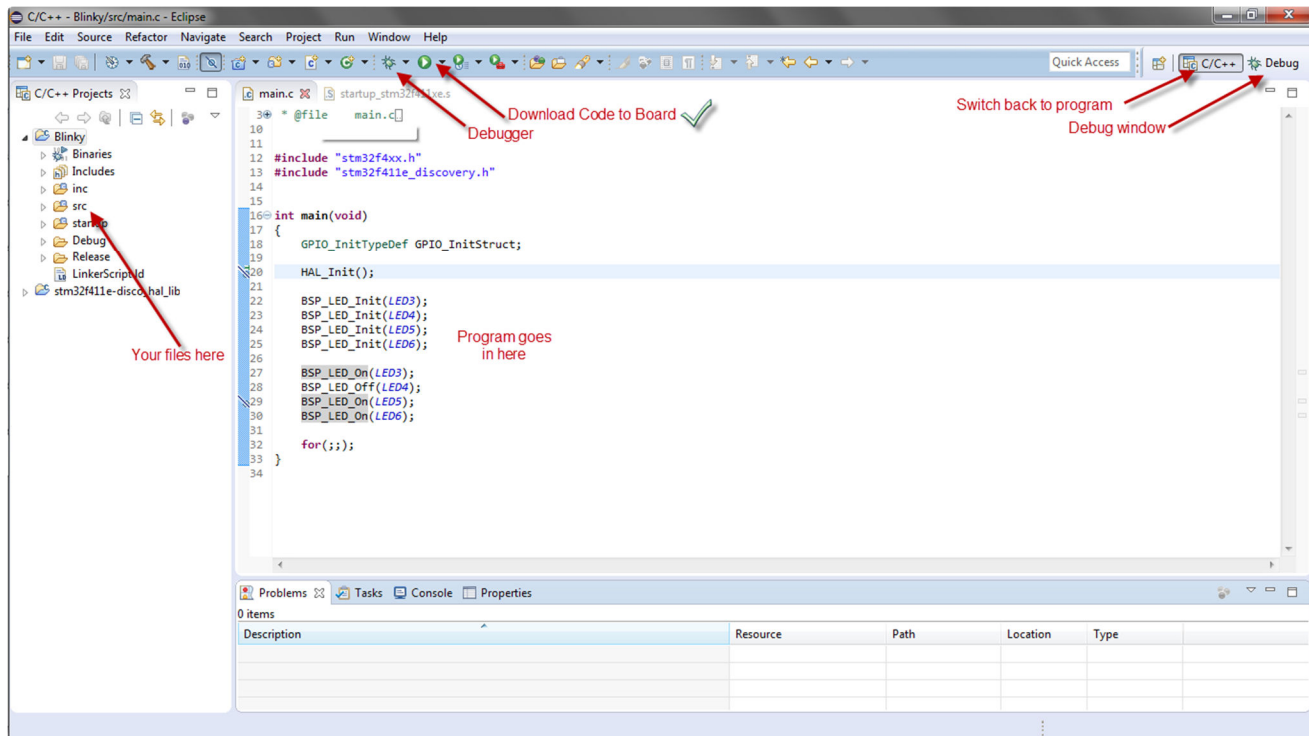
You might notice that Eclipse did not show anything after it is done. This is because some versions of Eclipse requires that you show the Workspace.

If you run into that follow this steps:

- Got to **Window >> Perspective >> Open Perspective>**
 - New window: click on **<C/C++>**
- Go to **Window >> Show View >> Project Explorer**
- Close the welcome screen for Eclipse



7.2 ECLIPSE QUICK AND DIRTY GUIDE



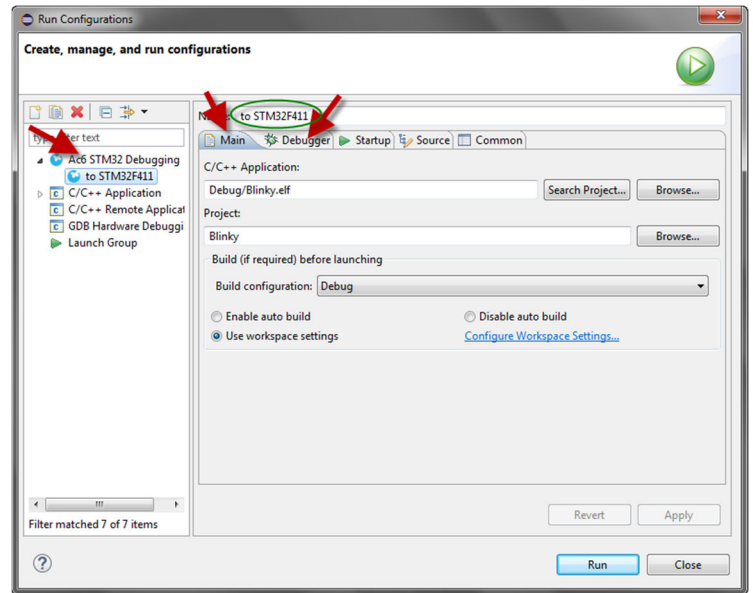
Some things to keep in mind and straight as you run into issues. OpenSTM32 workbench is the package that provided you access to the following:

- To Enable you to code:
 - STM32 Devices database and libraries
 - HAL – Hardware Abstraction Layer (http://www.st.com/st-web-ui/static/active/ip/resource/technical/document/user_manual/DM00105879.pdf)
 - Source code editor
 - Use the Linaro-provided version of arm-none-gnueabi;
- To enable you to Compile
 - Linker script generator
 - Building tools (GCC-based cross compiler, assembler, linker)
 - Quickly build it and generate the ELF file, binary
- To enable you to Debug
 - Debugging tools OpenOCD (modified to support SMT32 boards)
 - Debugging tools GDB

- Debug configuration to do the connection with **ST-Link/V2** to debug on chip
- Flash programming tools

To check out more details go to **Run >> Run Configurations...**

Note that your Main and Debug settings are under tabs when you pick the Ac6 STM32 Debugging configuration entry.



8 LET PLAY “HELLO WORLD” — A.KA. BLINKY

In hardware and embedded programming the first program to ensure you have set up the tool chain correctly, you program Blinky.

Blinky is a simple program that will say Hello to the world by blinking an LED. From that you could potentially be able to blink all other LEDs as well as get input from the switch.

Blinky consist of the following code:

```
/**
*****
 * @file    main.c
 * @author  Ac6
 * @version V1.0
 * @date    01-December-2013
 * @brief   This function initializes then turns on all of the LEDs
*****
 */

#include "stm32f4xx.h"
#include "stm32f411e_discovery.h"

#define ONE_SECOND(1000U) /* 1000 milliseconds */
#define FOREVERfor (;;)/ * A macro to make infinite loops a little more readable */

int main(void) {

    HAL_Init();/* Initialize the hardware abstraction layer */

    BSP_LED_Init(LED3);/* Initialize the LEDs using the routines from
stm32f401_discovery.h */
    BSP_LED_Init(LED4);
    BSP_LED_Init(LED5);
    BSP_LED_Init(LED6);

    FOREVER {
        BSP_LED_Off(LED3);/* LED3 is orange */
        BSP_LED_On(LED4);/* LED4 is green */
        BSP_LED_Off(LED5);/* LED5 is red */
        BSP_LED_On(LED6);/* LED6 is blue */

        HAL_Delay(ONE_SECOND);/* HAL_Delay is a routine that wastes time for the
specified many milliseconds */

        BSP_LED_On(LED3);
        BSP_LED_Off(LED4);
        BSP_LED_On(LED5);
        BSP_LED_Off(LED6);

        HAL_Delay(ONE_SECOND);
    }
}
```


This code uses the HAL libraries. Notice that is not the only way to write a program. The HAL environment gives us a premade (Arduino like) libraries.