Embedded Training– MakeICT Workshop

1 Project Goal

Come learn with use as we introduce programming skills via a project and program embedded systems with the vision of understanding the hardware and software details that Arduino would hide. We will effectively contrast the Arduino IDE to the way we perform the same activities, with the goal of learning how it is done and not rely only on someone else library

The workshop end goal is not to just prototype a system, but rather learn as we go prototyping a system and understanding and having full control of the activities being performed.

2 Workshop Agenda

Week 1: Introduction

- Introduction
- What is embedded
- What is Arduino
- What is the project
- What are we doing
- Activity: Define your project- block diagram it
- Quick overview of C structure Main()
- Quick overview of Variables
- Quick overview of Functions
- Tools
- Activity: Sudo Code the behavior

Week 2:

3 Project Definition

My plan is to create a balancing robot to use as a platform to learn C as embedded language. The balancing robot hardware will use:

- Processor: STM32F411E-DISCO
- Serial Dongle: http://www.digikey.com/product-detail/en/ftdi-future-technology-devices-international-ltd/UMFT234XD-WE/768-1175-ND/3904924

- Motor Drivers: http://www.amazon.com/StepStick-DRV8825-Stepper-Driver-Printer/dp/B00S3Q9YZA/ref=sr_1_1?s=hi&ie=UTF8&qid=1458786809&sr=1-1&keywords=A4988
- Motor: http://www.amazon.com/Stepper-Motor-Bipolar-64oz-Printer/dp/B00PNEQI7W/ref=cm_cd_al_qh_dp_t
- 1 battery: http://www.amazon.com/Turnigy-1300mAh-20C-Lipo-Pack/dp/B0072AEKY8/ref=sr_1_21?s=toys-and-games&ie=UTF8&qid=1458788066&sr=1-21&keywords=11.1V+LiPO
- 1 battery charger:
- 1 power switch: http://www.amazon.com/s/ref=nb_sb_noss_2?url=search-alias%3Dtoys-and-games&field-keywords=rc+switch
- Wire
- Soldering
- 3D printed body / Laser cut body

4 WHAT IS EMBEDDED PROGRAMMING?

What is a program? A program is a set of logical instructions that together execute to deliver some sort of function or activity. A program run in a target environment, like computers, servers, mobile phones or dedicated boards/microcontrollers. Depending on the target you use to program there are different languages available to code programs.

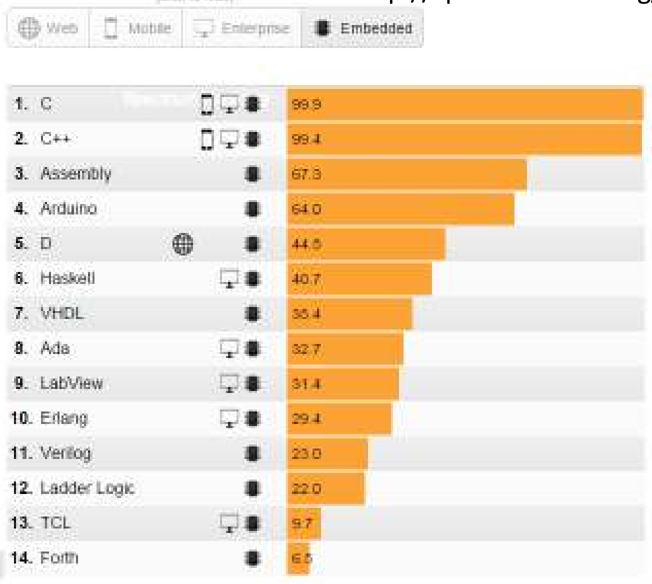
An embedded device can be consider one that has a single function, might not be upgradable, or might not be easily accessible. Some example of embedded devices are your microwave, your A/C controller, your Fitbit, etc. While a smart phone I do not consider an embedded device, it is certainly a collection of embedded functions (like temperature monitoring, battery charging, etc)

Embedded programing refers to the type of programming done on dedicated boards or embedded devices using microcontrollers or microprocessors that do not have enough features to run like computers or servers or mobile phones. Arduino environment can be considered embedded programming that targets the programming of Atmel microprocessor.

What language is worth learning? Well there is not a straight answer to that, there is a language that will work well depending of the application. For example I need to send serial data really quick to test out a device? I could use an Arduino and use it as a test tool.

We will use C for this course as it is the most relevant program with in Embedded. C++ is another great option which can be investigated.

Source: http://spectrum.ieee.org/

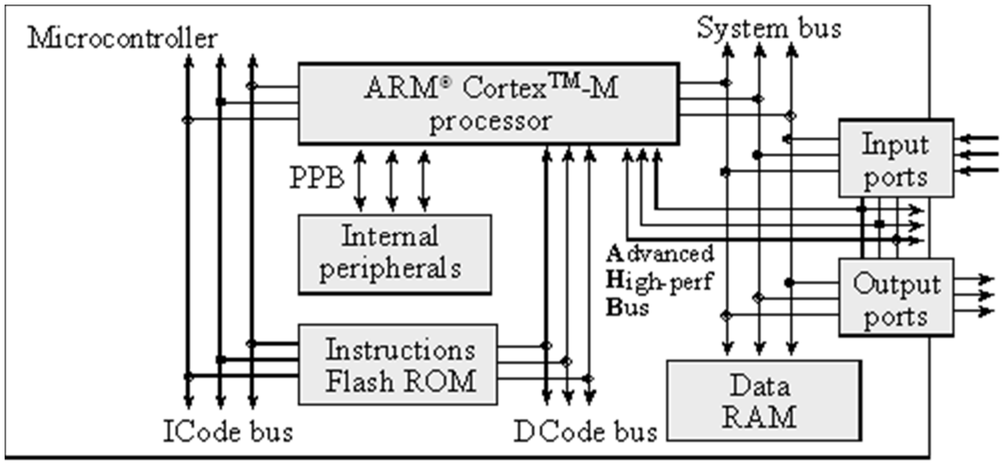


5 A Little Bit About Microcontrollers

A microcontroller is a programmable device, which upon power up it stars executing commands starting on the top of its memory map. It executes the instruction on each clock cycle, and increases a counter to the next address in the memory map.

<<Add figure of actual microprocessor might help>>

A simplified block diagram of a microcontroller based on the ARM® Cortex™-M processor.



It is a Harvard architecture because it has separate data and instruction buses.

- Instructions are fetched from flash ROM using the ICode bus.
- Data are exchanged with memory and I/O via the system bus interface.
- On the Cortex-M4 there is a second I/O bus for high-speed devices like USB.
- There are many sophisticated debugging features utilizing the DCode bus.
- The nested vectored interrupt controller (NVIC) manages interrupts, which are hardware-triggered software functions.
    - Some internal peripherals, like the NVIC communicate directly with the processor via the private peripheral bus (PPB).
    - The tight integration of the processor and interrupt controller provides fast execution of interrupt service routines (ISRs), dramatically reducing the interrupt latency.

The computer can store information in **RAM** by writing to it, or it can retrieve previously stored data by reading from it.

- RAMs are **volatile**; meaning if power is interrupted and restored the information in the RAM is lost.
- Most microcontrollers have **static RAM** (SRAM) using six metal-oxide-semiconductor field-effect transistors (MOS or MOSFET) to create each memory bit.

Information is programmed into **ROM** using techniques more complicated than writing to RAM.

- ROMs are nonvolatile; meaning if power is interrupted and restored the information in the ROM is retained.
- Some ROMs are programmed at the factory and can never be changed.
- A Programmable ROM (PROM) can be erased and reprogrammed by the user
    - The erase/program sequence is typically 10000 times slower than the time to write data into a RAM

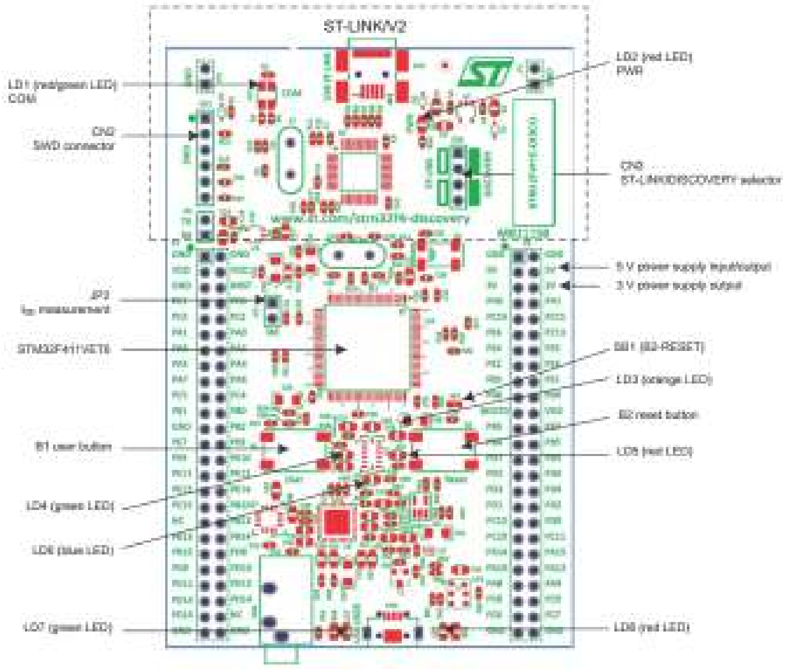
    - Electrically erasable PROM (EEPROM) are both erased and programmed with voltages.
        - We cannot program ones into the ROM.
        - We first erase the ROM, which puts ones into the entire memory
        - Then we program the zeros as needed.
- **Flash ROM** is a popular type of EEPROM.
    - In regular EEPROM, you can erase and program individual bytes.
    - Flash ROM must be erased in large blocks.
    - Because flash is smaller than regular EEPROM, most microcontrollers have a large flash into which we store the software.

For all the systems in this class, we will store instructions and constants in flash ROM and place variables and temporary data in static RAM.

5.1 Our Board

The manual for our board can be found at:
http://www2.st.com/content/ccc/resource/technical/document/user_manual/e9/d2/00/5e/15/46/44/0e/DM00148985.pdf/files/DM00148985.pdf/_jcr_content/translations/en.DM00148985.pdf

The power of our board consists of the following:



- Microcontroller: ARM4 STM32F411VET6 microcontroller featuring
    - 512 KB of Flash memory (**ROM**)
    - 128 KB of **RAM**
- **Debug**: On-board ST-LINK/V2 with selection mode switch to use the kit as a standalone STLINK/V2
- Board **power supply**: through USB bus or from an external 5 V supply voltage
- **Gyro**: L3GD20, ST MEMS motion sensor, 3-axis digital output gyroscope.
- **Accelerometer**: LSM303DLHC, ST MEMS system-in-package featuring
    - a 3D digital linear acceleration sensor
    - and a 3D digital magnetic sensor.
- **Microphone**: MP45DT02, ST MEMS audio sensor, omnidirectional digital microphone
- **Speaker** Output: CS43L22, audio DAC with integrated class D speaker driver
- 8 **LEDs**: Power, USB connection, Orange, Green, Red and Blue
- 2 Pushbuttons: (user and reset)
- USB OTG with micro-AB connector
- Extension header for LQFP100 I/Os for a quick connection to the prototyping board and an easy probing

6 SYSTEM SETUP

The set up will consist of a “text editor” interface called Eclipse. We will improve Eclipse with a set of support for our ARM processor by downloading a STM32 package from www.openstm32.org. Also, we will use OpenOCD, as the On Chip Debuger which comes as part of the STM32 package.

While you download and install the software make sure to listen to http://embedded.fm/episodes/2013/6/25/8-kidnapped-and-blindfolded

6.1 ECLIPSE

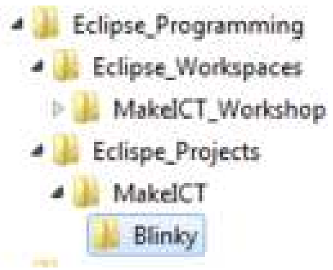
Eclipse is the interface you will use to program and it will hold the files you need for your board. In order to use eclipse you will use a Workspace.

A workspace is a concept of grouping together:

- a set of (somehow) related projects: For example all files using board SMT32F411E-DISCO
- some configuration pertaining to all these projects: The HAL files you want to leverage
- some settings for Eclipse itself

This happens by creating a directory and putting inside it (you don't have to do it, it's done for you) files that manage to tell Eclipse these information. All you have to do explicitly is to select the folder where these files will be placed. And this folder doesn't need to be the same where you put your source code - preferentially it won't be.

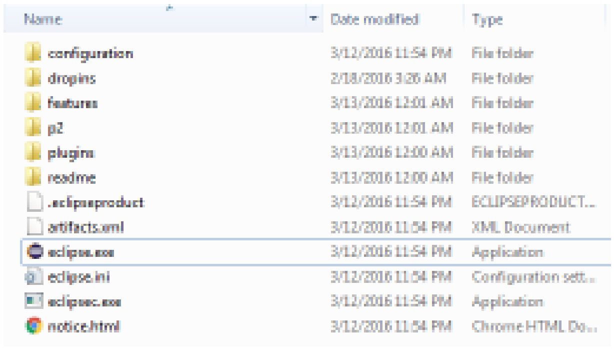
To do so for this workshop we will use the following configuration:



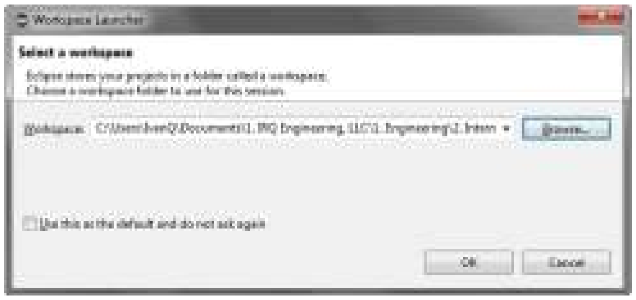
But in general the folder structure you want to follow should be of the following characteristics:

1. Create a folder for your projects:
   `/projects`
2. Create a folder each project, and group the project's sub-projects inside of it:
   `/projects/proj1/subproj1_1`
   `/projects/proj1/subproj1_2`
   `/projects/proj2/subproj2_1`
3. Create a separate folder for your workspaces:
   `/eclipse-workspaces`
4. Create workspaces for your projects:
   `/eclipse-workspaces/proj1`
   `/eclipse-workspaces/proj2`

- Make sure that you have Java installed (http://www.java.com/en/download/manual.jsp0)
    - Use the default folder
- Start by downloading Eclipse from http://www.eclipse.org/downloads/
- Extract the .zip file contents to the ***C:\Program Files\Eclipse*** (if using Win-7 64Bit) or equivalent
    - This will take about 10 min
- Go to ***C:\Program Files\Eclipse*** and run eclipse.exe
- On the pop up window create a desktop. Note the folder selected to understand where to look for your files.
    - Note that you should not include spaces on your folder. Keep names simple



Name	Date modified	Type
configuration	3/12/2016 11:54 PM	File folder
dropins	2/18/2016 3:26 AM	File folder
features	3/13/2016 12:01 AM	File folder
p2	3/13/2016 12:01 AM	File folder
plugins	3/13/2016 12:00 AM	File folder
readme	3/13/2016 12:00 AM	File folder
.eclipseproduct	3/12/2016 11:54 PM	ECLIPSEPRODUCT...
artifacts.xml	3/12/2016 11:54 PM	XML Document
eclipse.exe	3/12/2016 11:54 PM	Application
eclipse.ini	3/12/2016 11:54 PM	Configuration sett...
eclipsec.exe	3/12/2016 11:54 PM	Application
notice.html	3/12/2016 11:54 PM	Chrome HTML Do...



Workspace Launcher

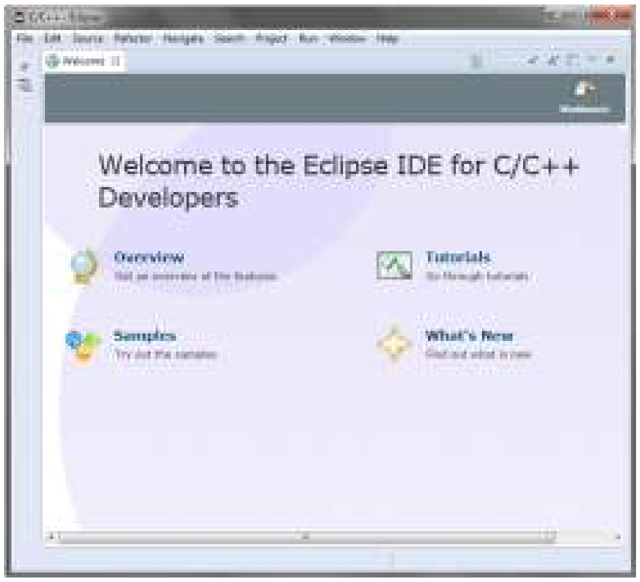
Select a workspace

Eclipse stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: C:\Users\IvanQ\Documents\1. PRQ Engineering, LLC\1. Engineering\2. Intern

☐ Use this as the default and do not ask again

- Click OK
- And you should have Eclipse installed and Running:
- Make sure to create a shortcut to your desktop
- Familiarize yourself with the interface



Welcome to the Eclipse IDE for C/C++ Developers

Overview
Get an overview of the features

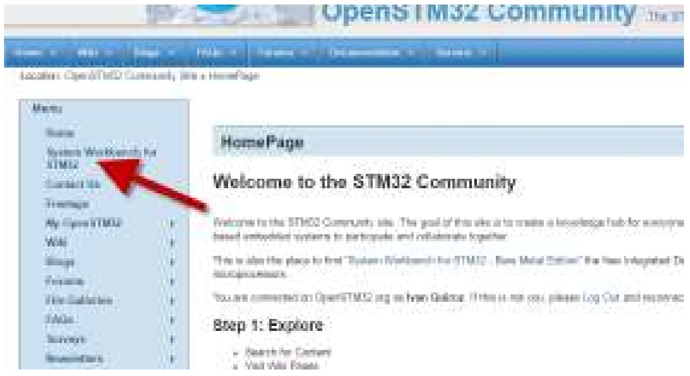
Tutorials
Go through tutorials

Samples
Try out the samples

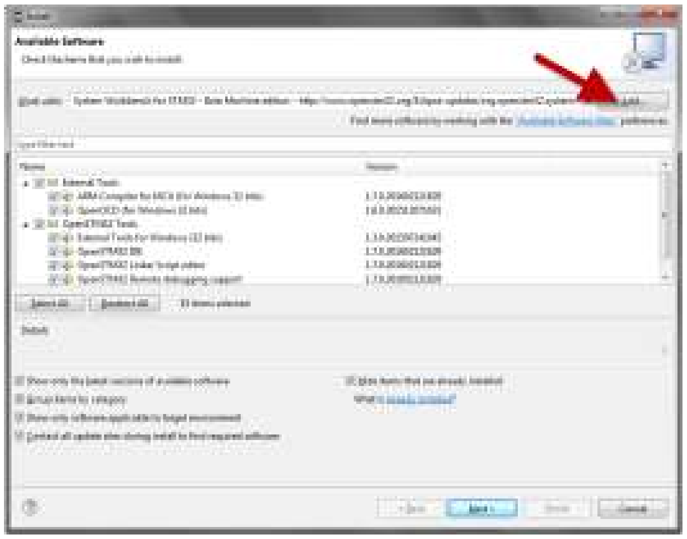
What's New
Find out what is new

File Edit Source Refactor Navigate Search Project Run Window Help
Welcome

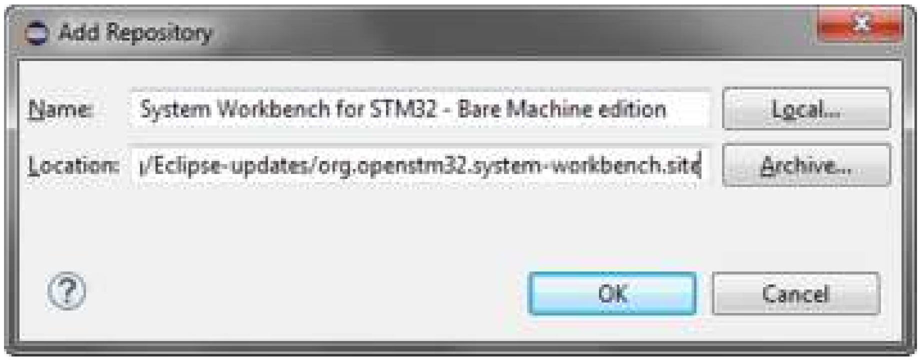
6.2 Installing OpenSTM32



- *Note if you have a Mac you can use the following: http://www.erikandre.org/2015/09/getting-started-with-openstm32-on-osx.html*
- Have your email on hand and readily available
- Go to http://www.openstm32.org/tiki-register.php and register to download it
- After your register, please make sure to check your email, as it will be needed to complete registration.
- Log in at: http://www.openstm32.org/tiki-login_scr.php
- Click "**System Workbench for STM32**"
- Find "[Installing System Workbench for STM32 from Eclipse](#)"
    - Installation of System Workbench for STM32 - Bare edition will be done through the standard Eclipse installer.
    - You should Start Eclipse
        - then open menu "**Help >> Install New Software**";
        - this will open the "Available Software" dialog:
    - Click on "Add:"



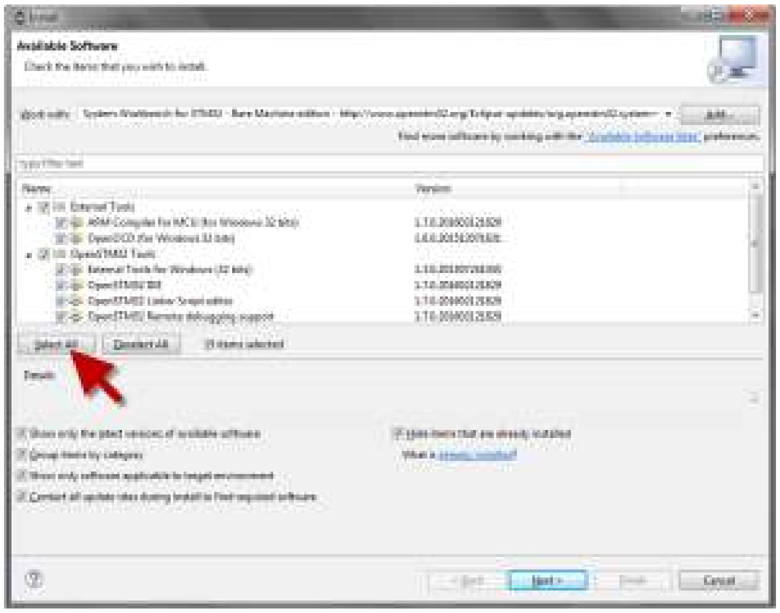
    - Give a name to the update site (System Workbench for STM32 - Bare Machine edition)
    - Set the location to http://www.openstm32.org/Eclipse-updates/org.openstm32.system-workbench.site



Name: System Workbench for STM32 - Bare Machine edition
Location: :/Eclipse-updates/org.openstm32.system-workbench.site

    - Then click "OK" to create the update site

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



Install

Available Software

Check the items that you wish to install.

Work with: System Workbench for STM32 - Bare Machine edition - http://www.openstm32.org/Eclipse-updates/org.openstm32.system- [illegible]

Find more software by working with the "Available Software Sites" preferences.

Name	Version
External Tools	
ARM Compiler for MCU (for Windows 32 bits)	[illegible]
OpenOCD (for Windows 32 bits)	[illegible]
OpenSTM32 Tools	
External Tools for Windows (32 bits)	[illegible]
OpenSTM32 IDE	[illegible]
OpenSTM32 Linker Script editor	[illegible]
OpenSTM32 Remote debugging support	[illegible]

Select All | Deselect All | [illegible] items selected

Details

☑ Show only the latest versions of available software
☑ Group items by category
☑ Show only software applicable to target environment
☑ Contact all update sites during install to find required software
☑ Hide items that are already installed
What is already installed?

< Back | Next > | Finish | Cancel

  - 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

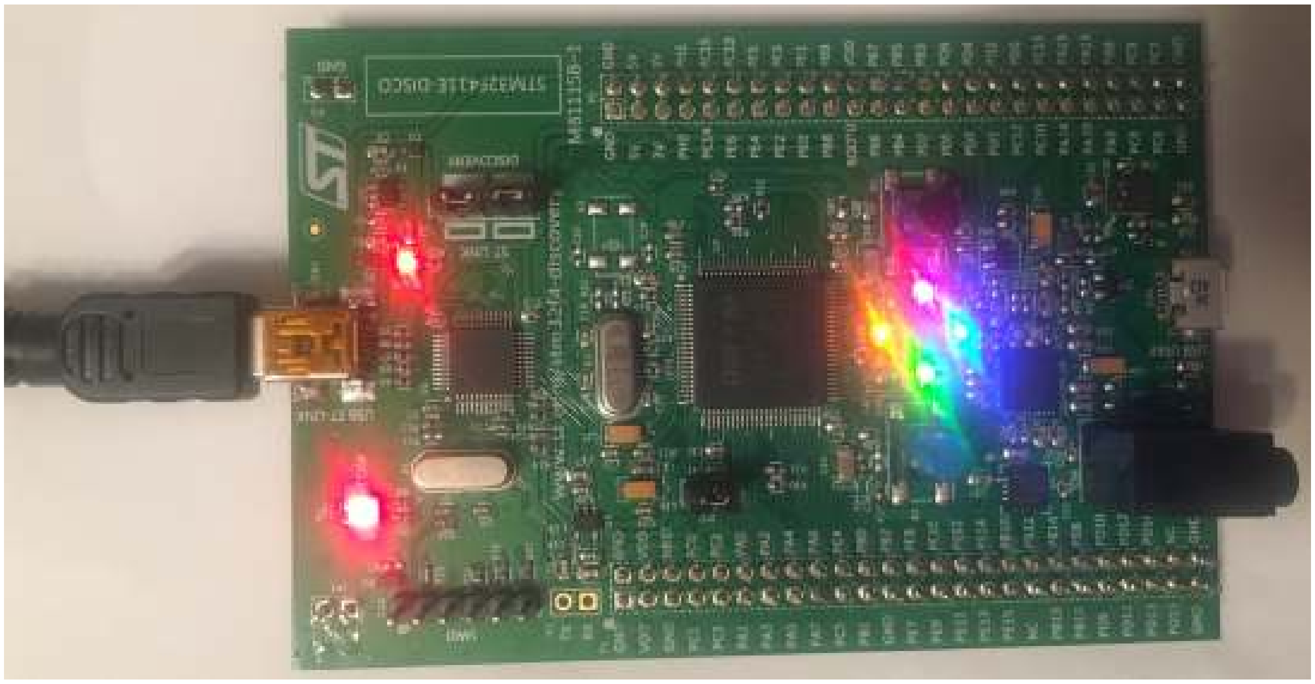
Until now you should have successfully downloaded and installed Eclipse.

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.

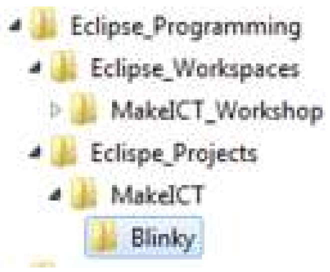
- Push the **BLUE** button and the LEDs will go out.
    - Shake the board and the LEDs will light up in response to the movements.
    - Push the blue button again and the LEDs will blink

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.

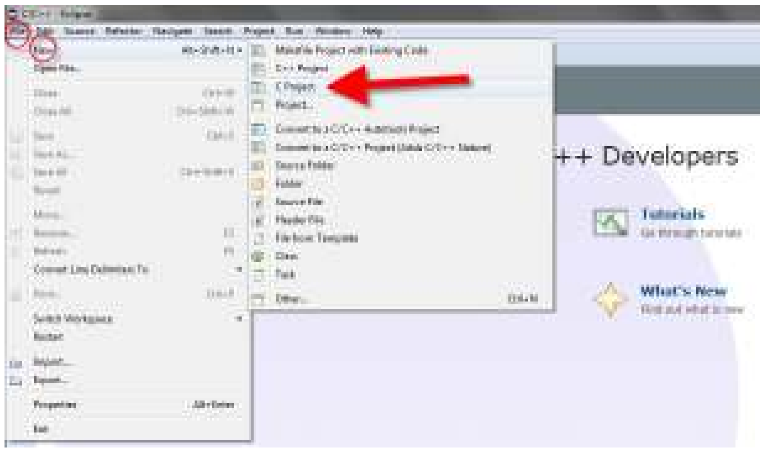
Once we complete this successfully we will have a tool chain that we can use to star writing our code

- Before you start anything create the following folder structure:

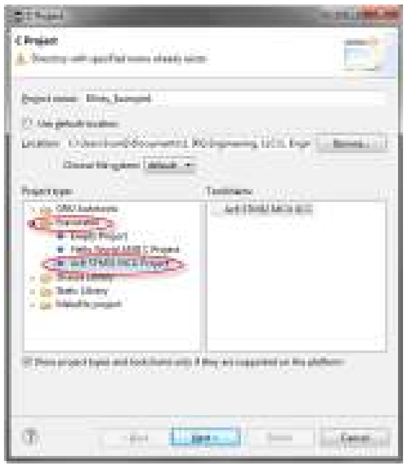


```
Eclipse_Programming
  Eclipse_Workspaces
    MakeICT_Workshop
  Eclispe_Projects
    MakeICT
      Blinky
```

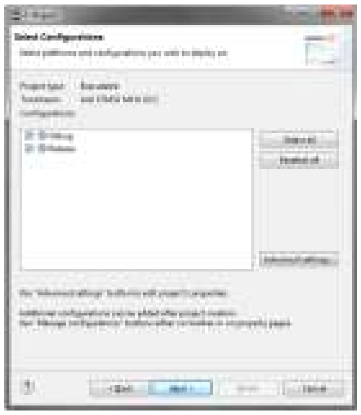
- Start new project:



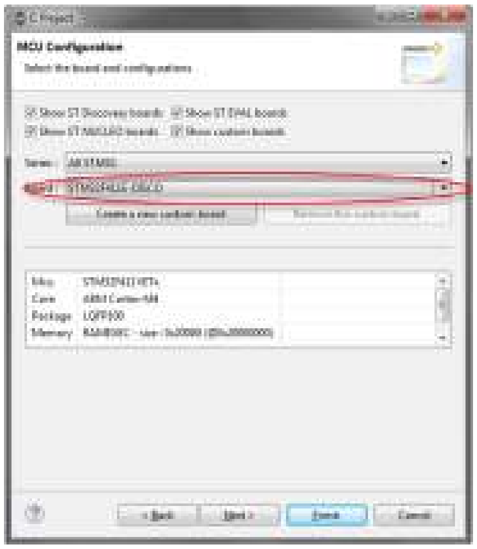
- Under Project Type: expand Executable and click on Ac6 STM32 MCU Project



- Click next on the following screen



- Under the Board option find and select STM32F411E-DISCO



C Project

MCU Configuration

Select the board and configurations

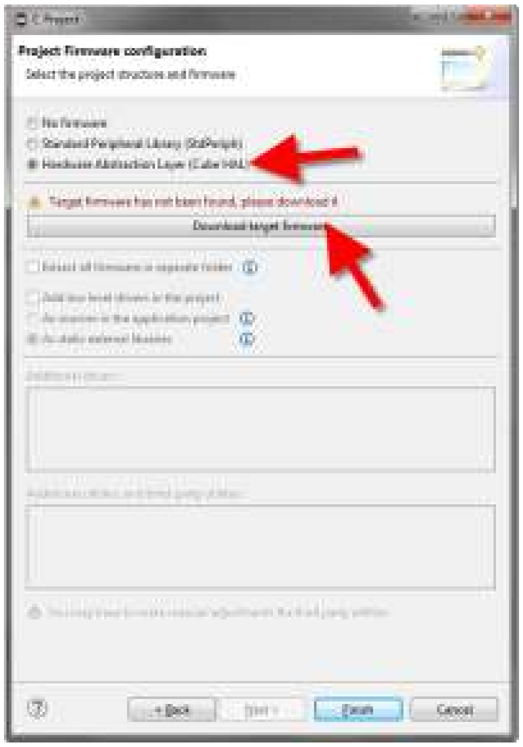
[✓] Show ST Discovery boards [✓] Show ST EVAL boards
[✓] Show ST NUCLEO boards [✓] Show custom boards

Series: All STM32
Board: STM32F411E-DISCO

Create a new custom board | Remove this custom board

Mcu	STM32F411VETx
Core	ARM Cortex-M4
Package	LQFP100
Memory	RAM/BSS - size : 0x20000 (@0x20000000)

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



C Project

Project Firmware configuration

Select the project structure and firmware

( ) No firmware
( ) Standard Peripheral Library (StdPeriph)
(●) Hardware Abstraction Layer (Cube HAL)

Target firmware has not been found, please download it

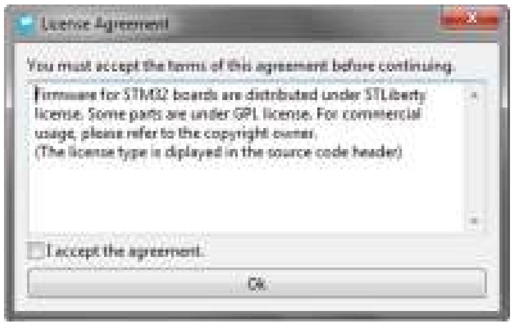
Download target firmware

[ ] Extract all firmware in separate folder

[ ] Add low level drivers in the project
( ) As sources in the application project
(●) As static external libraries

- 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 bottom and accept the terms and conditions



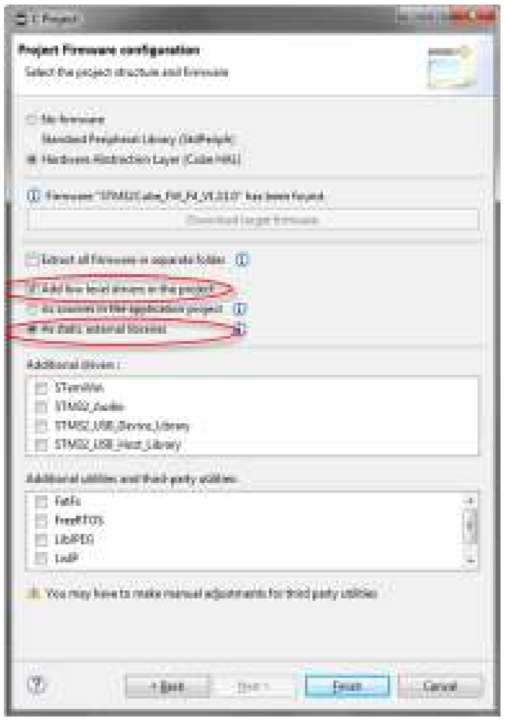
License Agreement

You must accept the terms of this agreement before continuing.

Firmware for STM32 boards are distributed under STLiberty license. Some parts are under GPL license. For commercial usage, please refer to the copyright owner.
(The license type is diplayed in the source code header)

I accept the agreement.

- 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



C Project

Project Firmware configuration

Select the project structure and firmware

No firmware
Standard Peripheral Library (StdPeriph)
[✓ Hardware Abstraction Layer (Cube HAL)]

Firmware "STM32Cube_FW_F4_V1.21.0" has been found

Extract all firmware in separate folder
[✓ Add low level drivers in the project]
As sources in the application project
[✓ As static external libraries]

Additional drivers :
STemWin
STM32_Audio
STM32_USB_Device_Library
STM32_USB_Host_Library

Additional utilities and third-party utilities:
FatFs
FreeRTOS
LibJPEG
LwIP

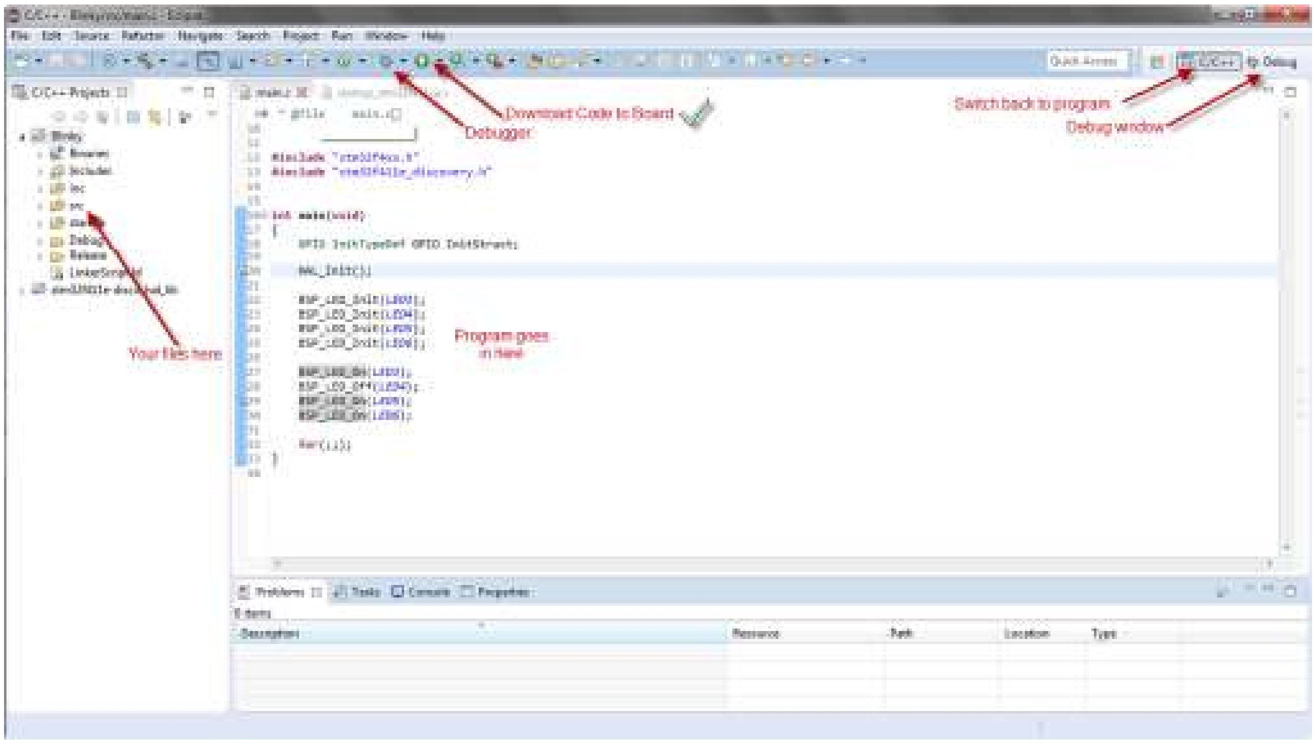
You may have to make manual adjustments for third party utilities

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

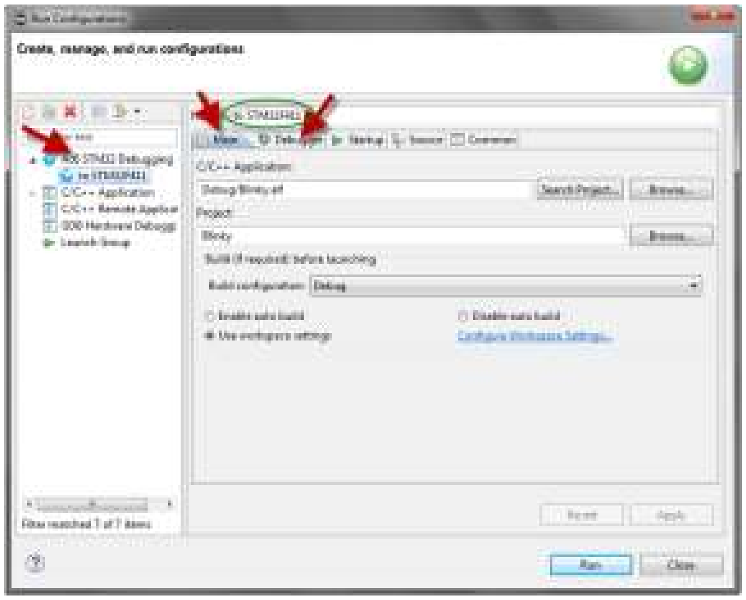
7.2 ECLIPSE QUICK AND DIRTY GUIDE



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

int main(void)
{
    GPIO_InitTypeDef GPIO_InitStruct;

    HAL_Init();

    BSP_LED_Init(LED3);
    BSP_LED_Init(LED4);
    BSP_LED_Init(LED5);
    BSP_LED_Init(LED6);

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

    for(;;);
}
```

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
  - 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 programing tools



Name: STM32F411
C/C++ Application: Debug/Blinky.elf
Project: Blinky
Build (if required) before launching
Build configuration: Debug
Enable auto build
Disable auto build
Use workspace settings
Configure Workspace Settings...
Filter matched 7 of 7 items

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);
	}
}
```