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.

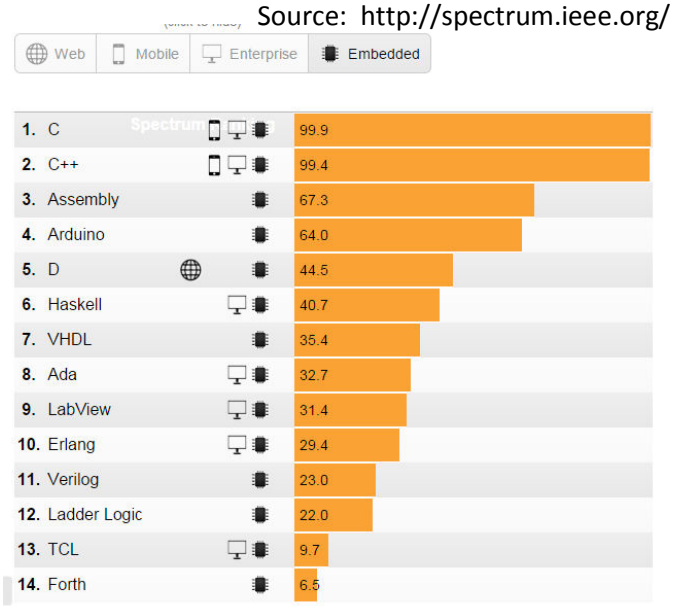
An embedded device can be consider one that has a single function, might not be upgradable, or might not be easily accessible. Some example of embedded devices are your microwave, your A/C controller, your Fitbit, etc. While a smart phone I do not consider an embedded device, it is certainly a collection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

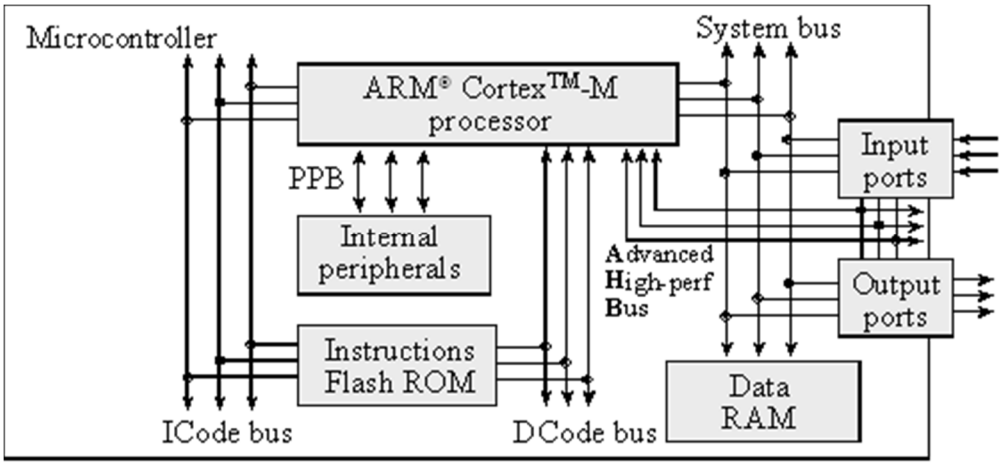


5 A LITTLE BIT ABOUT MICROCONTROLLERS

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

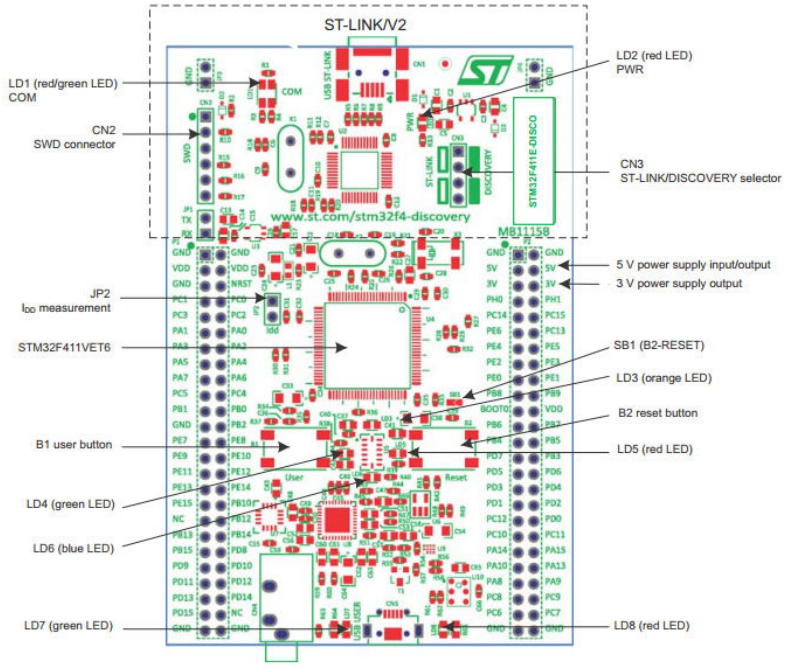
- Electrically erasable PROM (EEPROM) are both erased and programmed with voltages.
  - We cannot program ones into the ROM.
  - We first erase the ROM, which puts ones into the entire memory
  - Then we program the zeros as needed.
- **Flash ROM** is a popular type of EEPROM.
  - In regular EEPROM, 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

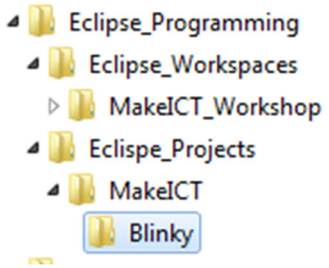
Eclipse is the interface you will use to program and it will hold the files you need for your board. In order to use eclipse you will use a Workspace.

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.

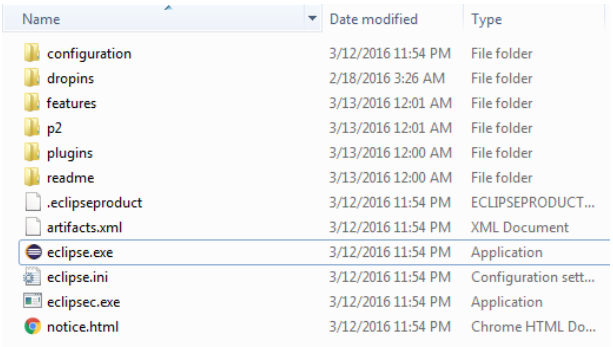
To do so for this workshop we will use the following configuration:

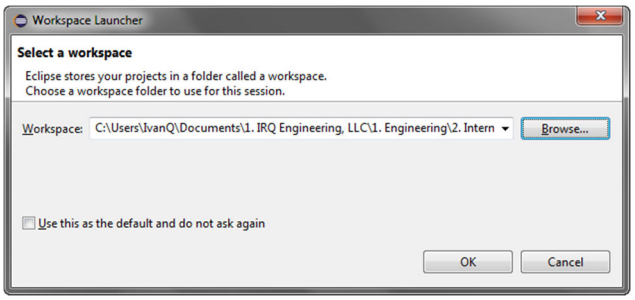


But in general the folder structure you want to follow should be of the following characteristics:

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



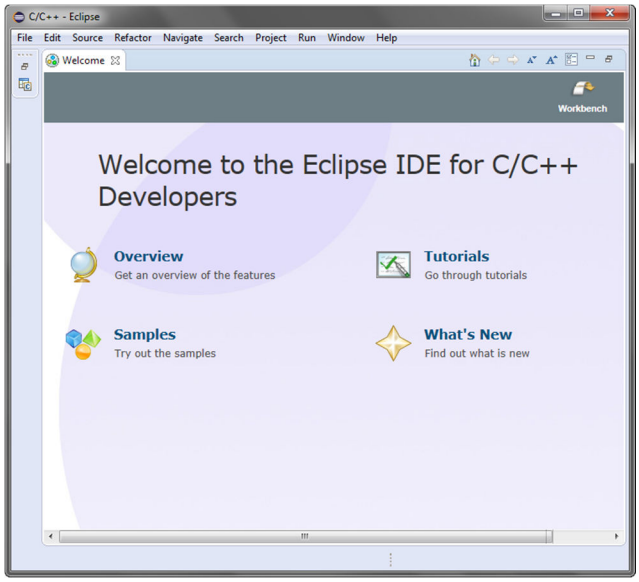


- Click OK

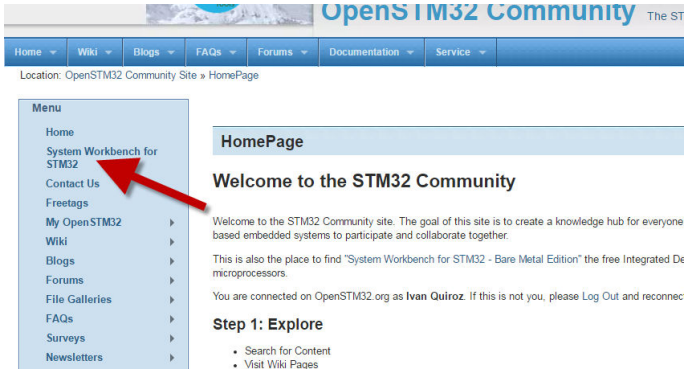
- And you should have Eclipse installed and Running:

- Make sure to create a shortcut to your desktop

- Familiarize yourself with the interface



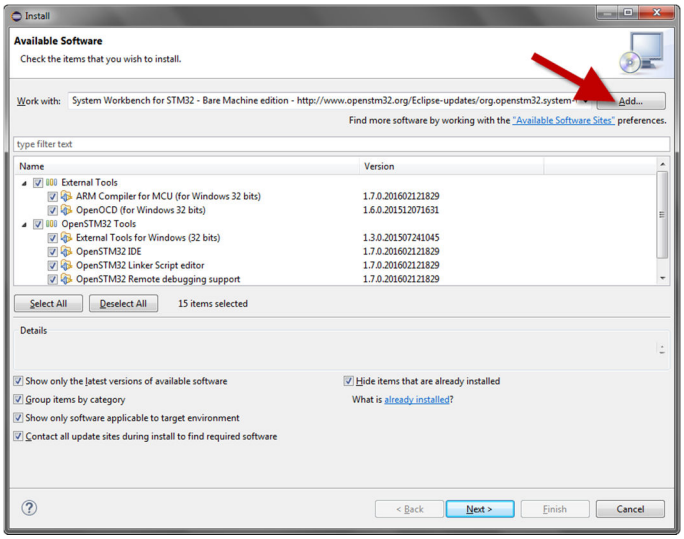
6.2 INSTALLING OPENSTM32



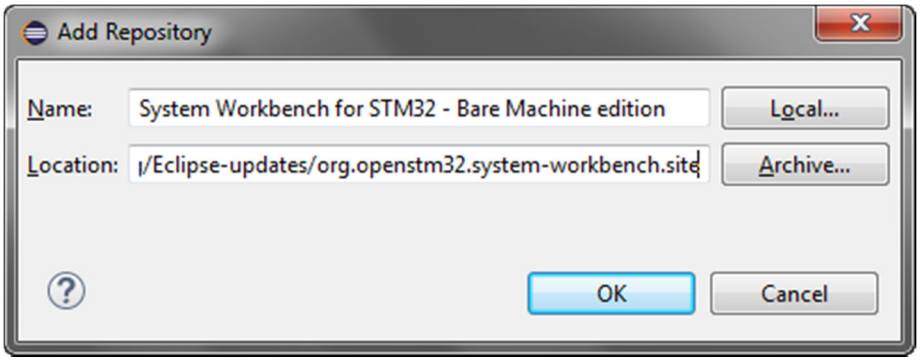
- *Note if you have a Mac you can use the following: http://www.erikandre.org/2015/09/getting-started-with-openstm32-on-osx.html*
- Have your email on hand and readily available
- Go to http://www.openstm32.org/tiki-register.php and register to download it
- After your register, please make sure to check 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



    - Then click "OK" to create the update site