CG2271 Real Time Operating system AY 2017/18

Lab 1 – Introduction to Arduino (3% of your final grade)

Lab Lesson: Wednesday 30 Aug and 6 Sept, Demo: Wednesday 13 Sept

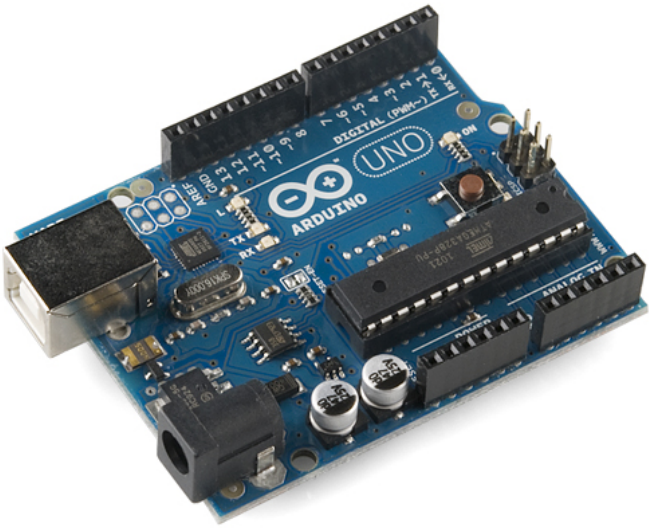
1. Introduction

In this lab you will be introduced to the programming environment for the Arduino platform. The Arduino software integrated development environment (IDE) is sufficient to program the platform. However, freeRTOS (which we will use in the second half of the module) needs Eclipse IDE. Eclipse IDE contains a base workspace and an extensible plug-in system for customizing the environment. We will extend Eclipse with AVR plugin that contains the tools and settings for developing C/C++ programs for ATMEL AVR series of embedded processors. The Arduino platform contains ATMEL AVR ATmega328P processor core. The Eclipse IDE together with AVR plugin and Arduino software provides a powerful integrated development environment for developing and downloading software onto Arduino platform.

In this lab, you will be introduced to set up Eclipse environment with AVR plugin and Arduino software, creating circuits on the breadboard connecting LEDs with the micro-controller, and writing simple software that runs on the Arduino platform to control the LEDs.

2. Downloading and Installing the tools

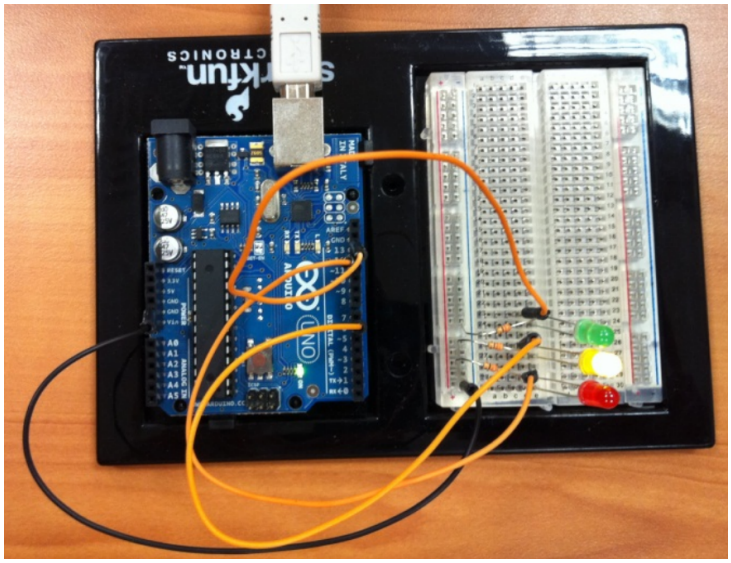
Your primary development platform will be the Arduino platform. Arduino is an open-source hardware/software platform that is designed to make building embedded systems simple. Each team of 2 persons is provided with a Sparkfun Inventor Kit, made up of an Arduino Uno processor board, a breadboard, wires and assorted electronic components to help you build up your projects. The Arduino Uno board is shown below:



The top row of sockets are labeled “0” to “13”, then “GND” and “AREF”. The pins labeled “0” to “13” are digital input/output pins, while GND provides electrical ground. The AREF pin provides a reference voltage for the analog to digital converters.

The bottom row is labeled “A5” to “A0”, then Vin, GND, GND, 5V, 3.3V and RESET. The A5 to A0 inputs are analog inputs, while Vin is to power the board from an external source. GND provides electrical ground; 5V and 3.3V provide a 5-volt and 3.3-volt electrical source, and RESET, when pulled low, causes the board to reboot. There is also a RESET button on the board for this purpose.

You will assemble your circuits on the breadboard that is provided. Connections are made between the breadboard and the Arduino using the wire jumpers provided. An example circuit is shown below:



Note that you should ALWAYS color code your wires. e.g. black for GND, red for 5V/3.3V, green for inputs, orange for outputs, etc. This makes your circuits much easier to debug.

Arduino is also an open-source software standard, comprising of a simple software development environment, tools to upload written programs to the Arduino board, and a set of libraries that simplify access to the hardware. For the first three labs we will be using the Eclipse IDE with AVR plugin, so you will need to download the needed software from this link:

Arduino software 1.6.7
https://www.arduino.cc/en/Main/Software

Eclipse IDE for C/C++ Developers
http://www.eclipse.org/downloads/packages/eclipse-ide-cc-developers/mars1

WinAVR

http://sourceforge.net/projects/winavr/

Arduino software provides all compilation tools except for the make tool. So you will need to install WinAVR so that you have access to a make tool within Eclipse IDE. Download the software and follow the instructions for installation, including installing the Atmel USB driver.

*For **Mac** user, please install **CrossPack for AVR® Development** instead of WinAVR:
https://www.obdev.at/products/crosspack/index.html

Read through this page to gain an overview of the Arduino Project.

http://arduino.cc/en/Guide/HomePage

Follow the platform specific instructions to try out the board. You may use the Arduino software directly without Eclipse IDE for this purpose. For Windows users:

http://arduino.cc/en/Guide/Windows

I will assume that you have installed the Arduino software in c:\. The software used in this report is installed in "c:\program files\arduino". Modify the instructions to suit your actual installation.

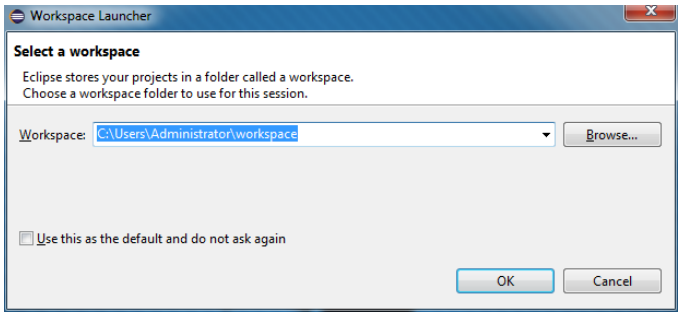
GNU Win32

http://sourceforge.net/projects/gnuwin32/files/make/3.81/make-3.81.exe

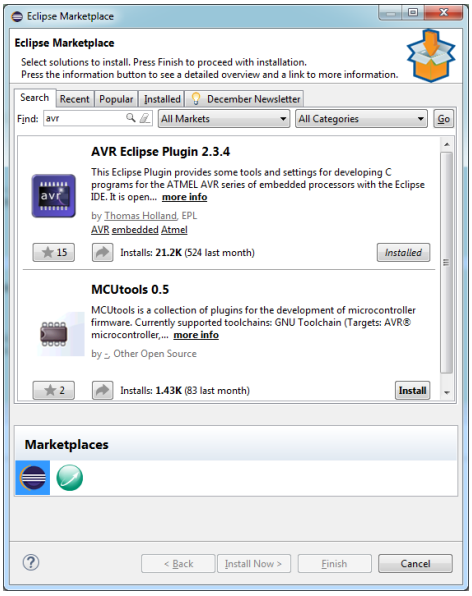
GNU Make will be used to compile the project.

3. Installing AVR Plugin for Eclipse

You should decompress your downloaded Eclipse package first, and click eclipse.exe to get Eclipse run. If it is your first time to use Eclipse, a dialog box will be shown and let you choose a workspace. Browse to a directory and click "OK", then select "Use this as the default and do not ask again" and click "OK".



From Eclipse go to “Help->Eclipse Marketplace”, enter “AVR” in find area and click search. Click “Install” to install “AVR Eclipse Plugin 2.3.4”.



4. Creating a new Arduino project in Eclipse

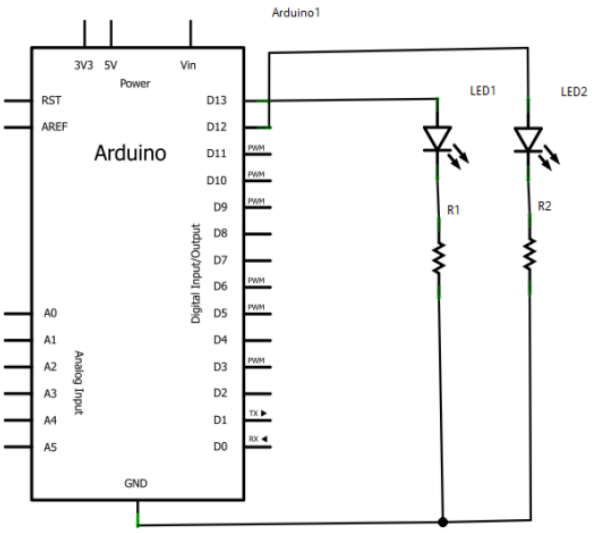
We will now create a simple Blinky program. We will build a circuit with two LEDs (one red and one green), and alternately flash the two LEDs.

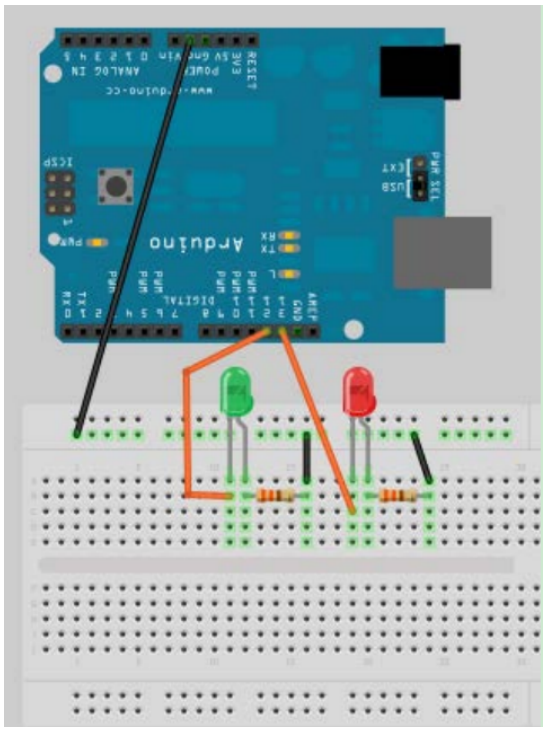
4.1 Building the Circuit

To assemble this circuit you will need:

Component	Quantity
LED	1 red 1 green (or yellow)
Resistor 330 ohm (body has orange-orange-brown stripes)	2
Wires	3 black 2 orange

The circuit diagram is shown below, followed by the layout on the breadboard.



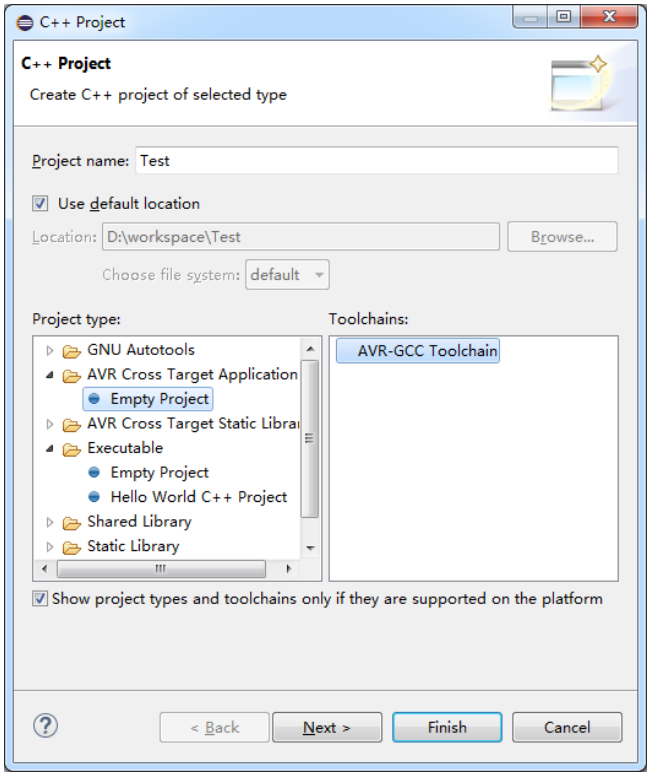


The positive (longer) legs of the red and green (or yellow) LEDs are connected to digital pins 13 and 12 respectively on the Arduino, while the negative (shorter) legs are each connected to a 330 ohm resistor (orange-orange-brown). The other leg of the resistor is connected to GND via the common ground line on the breadboard. The resistors are important to prevent burning the LEDs.

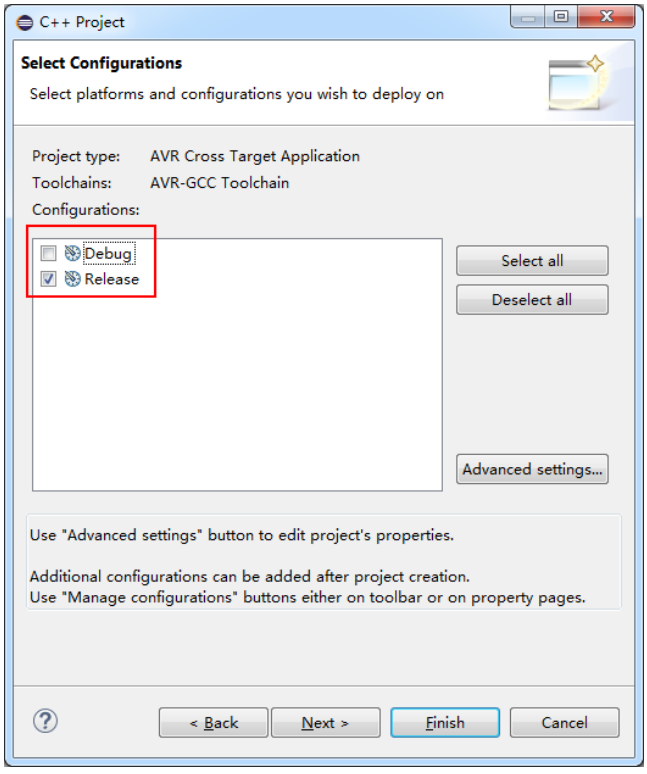
NOTE: DO NOT CUT ANY OF THE WIRES YOU ARE GIVEN!

4.2 Creating a New Test project

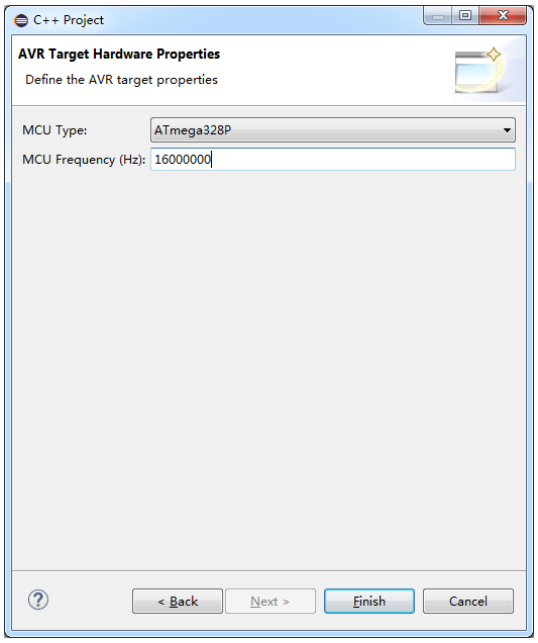
From Eclipse go to “File -> New ->C++ project”. In the “C++ Project” page, click “AVR Cross Target Application” and select “Empty Project”. Enter a project name and click “Next”.



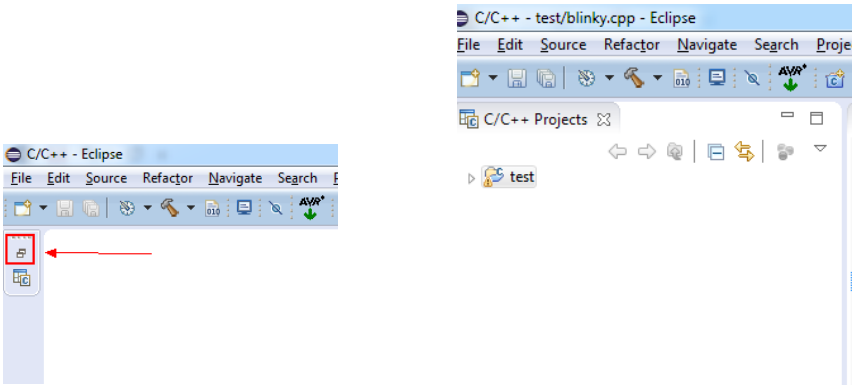
The Arduino SDK with Eclipse cannot build debug version, so deselect the “Debug” configuration and click “Next”.



In the “AVR Target Hardware Properties” page, set the “MCU type” to “ATmega328P” and “MCU Frequency” to 16000000Hz. Once done click “Finish”, now you have an empty Arduino project.



You can browse your projects list by clicking the button shown below:



4.3 Importing Arduino Core Files to Your Project

Now we need to import the Arduino core files from the Arduino SDK folder. In project navigator, right click on your project and select “New ->Folder”, click “Advanced”, select “link to alternate location” and browse directory to

“<arduino_base_path>\ hardware\arduino\avr\cores\arduino”,

Click “Finish” button to end the folder linking.

Do the same for linking “standard” folder to

“<arduino_base_path>\hardware\arduino\variants\standard”

For **Mac**, the directories are:

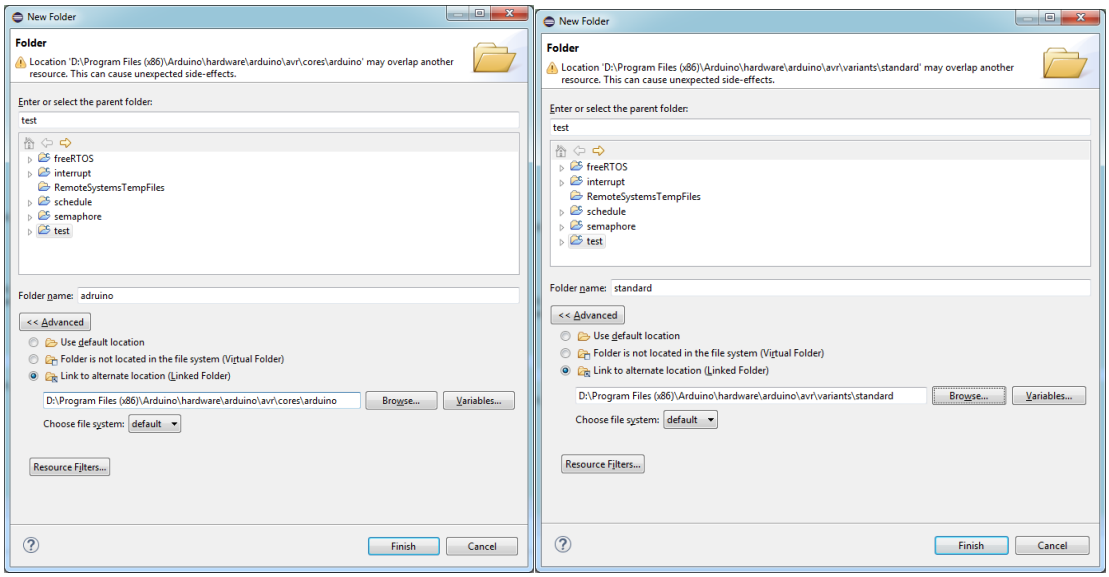
“/Applications/Arduino.app/Contents/Java/hardware/arduino/avr/cores/Arduino”

“/Applications/Arduino.app/Contents/Java/hardware/arduino/avr/variants/standard”

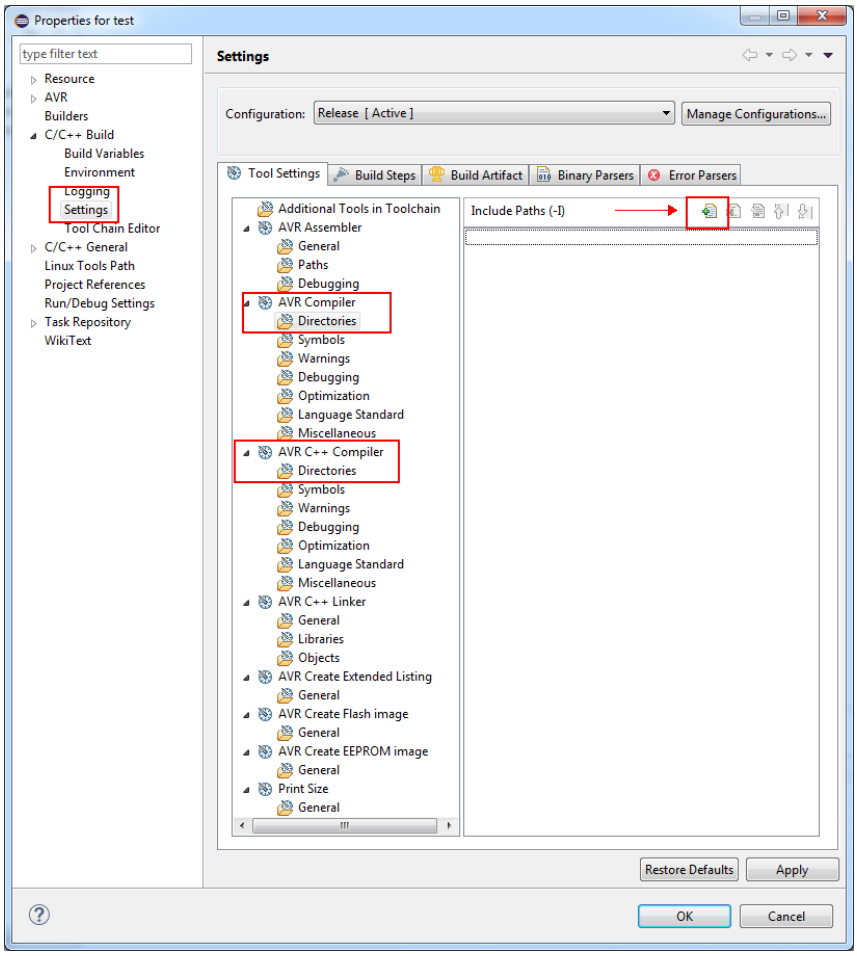
In my case, as the picture shows, the two directories are:

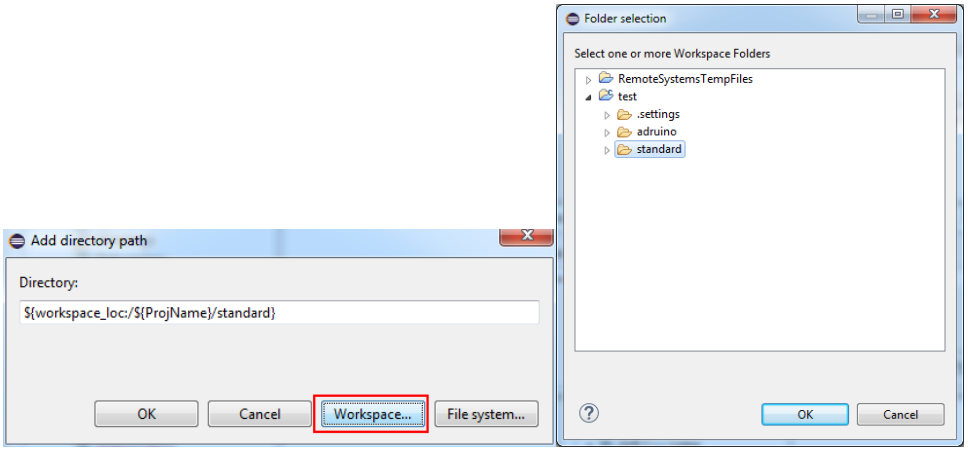
“D:\Program Files (x86)\Arduino\ hardware\arduino\avr\cores\arduino”

“D:\Program Files (x86)\Arduino\hardware\arduino\avr\variants\standard”

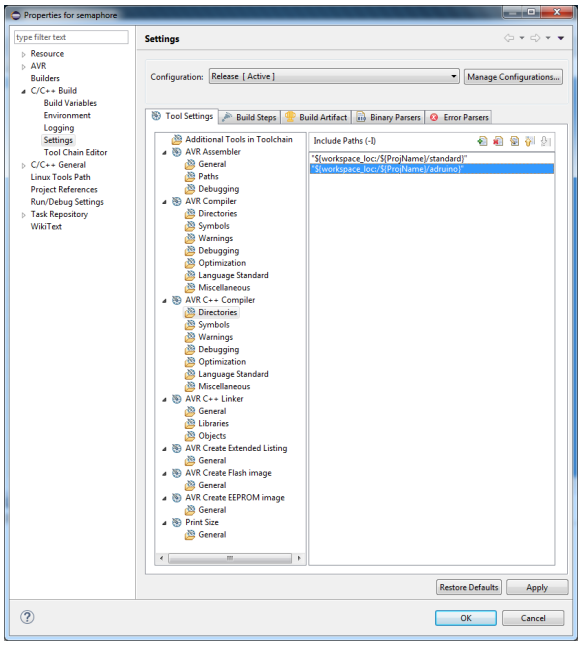


Then we will let compiler know that the linked directories need to be compiled. In Eclipse, right click your project and select “Properties -> C/C++ Build”. In “Setting -> Tool Settings” section, click “AVR Compiler -> Directories”, click “plus” button to add “arduino” and “standard” folders to “AVR Compiler -> Directories” and “AVR C++ Compiler -> Directories”.

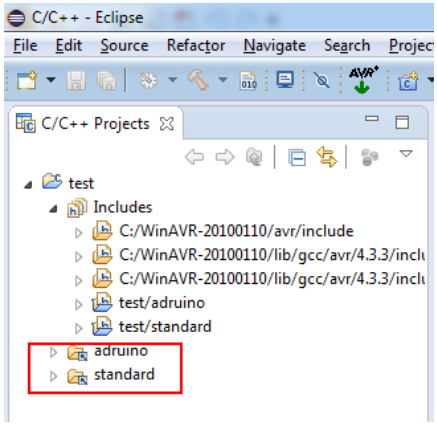




If you have done this step, it should look like below:

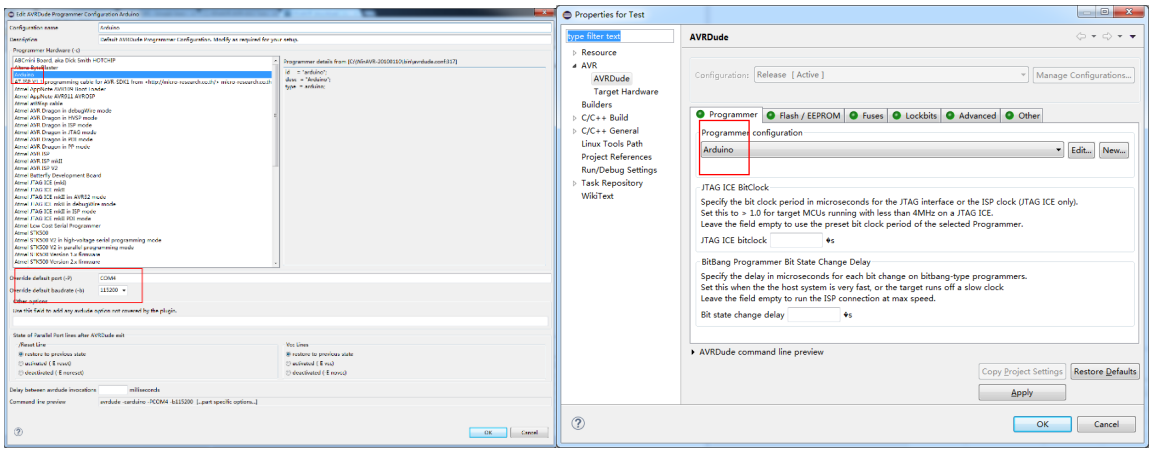


When you finish all the steps, the structure of this project will look as shown below.



4.4 Configuring AVR Build Setting

Open “Preferences” from Window menu, select “AVR->AVRDude” and click “Add” on the right side. In the opened window, select “Arduino” from the “Programmer Hardware” list. Enter a name in the “Configuration name” field above the list. Enter a serial port number to which your Arduino is connected into the "Override default port" box below the list. In my case, the serial port connected to Arduino is COM4. You can check your serial port number (with name “Arduino Uno”) at “Device Manager” in your PC.



Click "AVR/Paths", select "AVR-GCC" and Click "Edit". Set Path source to "Custom", Then Browse directory to

<arduino_base_path>\Arduino\hardware\tools\avr\bin

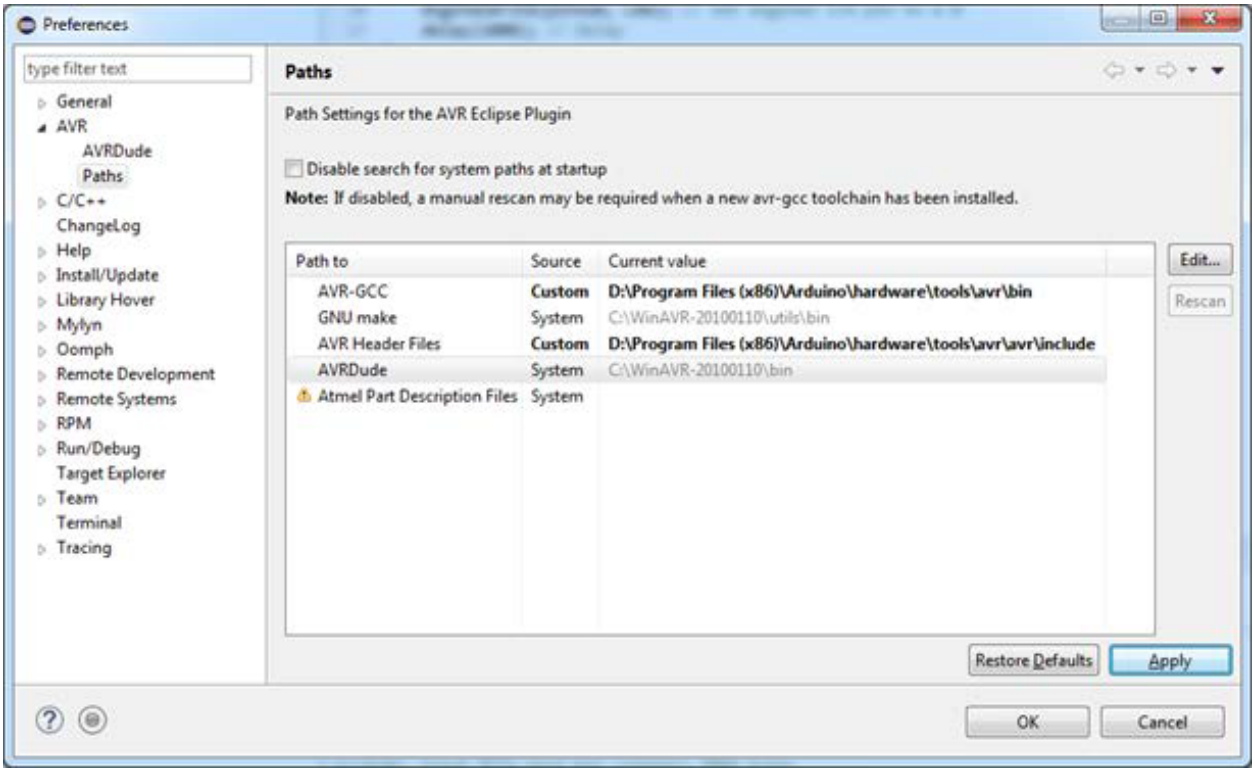
and click OK. Do the same for customizing "AVR Header Files" paths:

AVR Header Files:	**"<arduino_base_path>\Arduino\hardware\tools\avr\avr\include"**	
GNU Make:	**"<WinAVR_base_path>\WinAVR-201001110\utils\bin"**	(unchanged)
AVRDude:	**"<WinAVR_base_path>\WinAVR-201001110\bin"**	(unchanged)

For **Mac**, the Paths are:

AVR-GCC:	**"/Applications/Arduino.app/Contents/Java/hardware/tools/avr/bin"**
GNU make:	**unchanged**
AVR Header Files:	**"/Applications/Arduino.app/Contents/Java/hardware/tools/avr/avr/include"**
AVRDude:	**"/usr/local/CrossPack-AVR-20131216/bin"**

After you have completed all the steps, it will look like below. The Eclipse IDE is now ready for Arduino development.

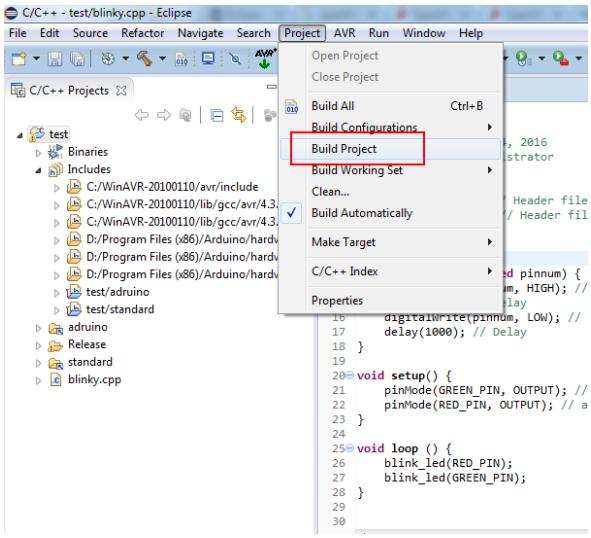


4.5 Keying in The Code and Compiling

Create a new file with name “blinky.cpp” and type the code below to the file.

```
/*
 * blinky.cpp
 *
 *  Created on: Jan 14, 2016
 *      Author: Administrator
 */

#include <avr/io.h> // Header file to access Atmega328 I/O registers
#include <Arduino.h> // Header file for the Arduino library
#define GREEN_PIN 12
#define RED_PIN 13

void blink_led(unsigned pinnum) {
	digitalWrite(pinnum, HIGH); // Set digital I/O pin to a 1
	delay(1000); // Delay
	digitalWrite(pinnum, LOW); // Set digital I/O pin to a 0
	delay(1000); // Delay
}

void setup() {
	pinMode(GREEN_PIN, OUTPUT); // Set digital I/O pins 12
	pinMode(RED_PIN, OUTPUT); // and 13 to OUTPUT.
}

void loop () {
	blink_led(RED_PIN);
	blink_led(GREEN_PIN);
}
```

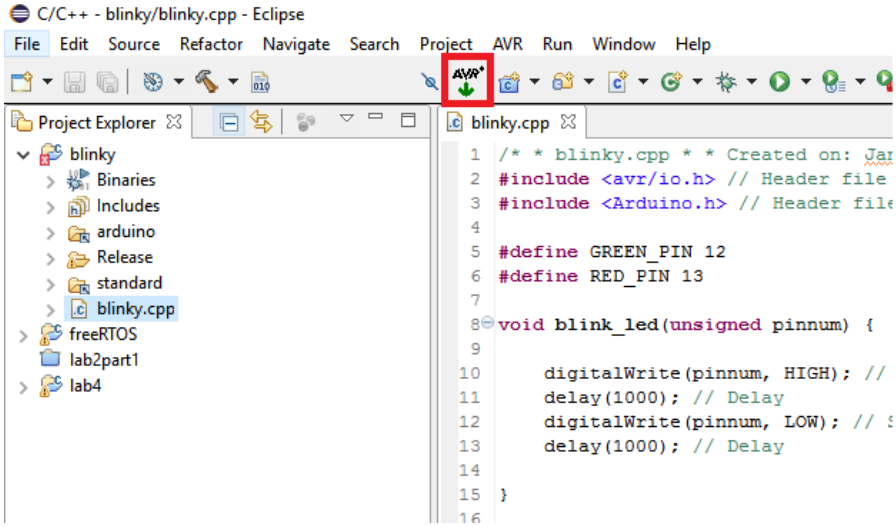
Click “Build Project (Ctrl + B)” in Project menu to compile this project



If all goes well, the “Problems” window will show “0 errors, x warning, 0 others”. If not you will have to check your typing, and ensure that all the directories are set up properly according to the instructions.

Assuming that the code has compiled properly, connect the Arduino board now to your PC/notebook using the USB cable provided. Ensure that you know which COM port the Arduino is connected to. The examples shown here will assume COM4, but it is DIFFERENT for different computers! In fact if you connect two different Arduino boards to the same computer, the COM ports will be different!

Click “AVR” button in your toolbar to upload your project to device.



Finally, you will see that the red and green lights on your breadboard are blinking.

5. Demo

You have to show a demo that your program is working correctly to your lab TA on Wednesday 13 Sept.

6. Summary

In this lab you have set up the Eclipse with AVR plugin environments, and have gotten a feel for assembling a circuit and programming the Sparkfun Inventor Kit.