# Getting Started

See: http://www.cs.ou.edu/~fagg/classes/general/atmel/

Summary:

- Install compiler
- Download your subversion tree
- Plug the programmer into your computer
- Plug the programmer into the Arduino board
- Create a program

# Subversion

Similar to “Dropbox”: allows you to easily share a folder across multiple computers

# Subversion

Key commands:

- **Checkout**: get initial copy of the shared folder
- **Add**: mark a file or a folder as shared
  - Only share necessary files: .c, .h, makefile, .ppt, .pptx, .avrsln, .avrsuo
- **Update**: copy changes to the folder down to your computer
- **Commit**: copy your changes to the folder up to the server

# Subversion

When you sit down to work:

- It is best if you are the only one editing a particular file (so coordinate with your group members)
- Perform an update
- Make your changes (until you are happy)
- Add any new files
- Commit your changes:
  - Always remember to do this when you are done

# Subversion

Conflicts occur when two people edit the same file & then try to check in their changes

- The second person to commit will end up with several versions of the file in their folder:
  - A file with the two sets of changes (with changes clearly marked)
  - A file each that corresponds to the changes made by one individual
- The second person must select one, copy it over to the original file name, make any necessary changes, and commit again

# Downloads from Atmel HOWTO

Already in your subversion tree:

- lib/libou_atmega2560.a
- include/oulib.h
- Include/oulib_serial_buffered.h

For Unix users (also in your tree):

- makefile

# Compiling and Downloading (the Unix way)

- Makefile:
  - Modify the “TARGET” and “OULIB_DIR” lines for your program
- Type “make”
  - You should see no errors
- Type “make program”
  - This will download your code to the processor
  - Again, you should see no errors

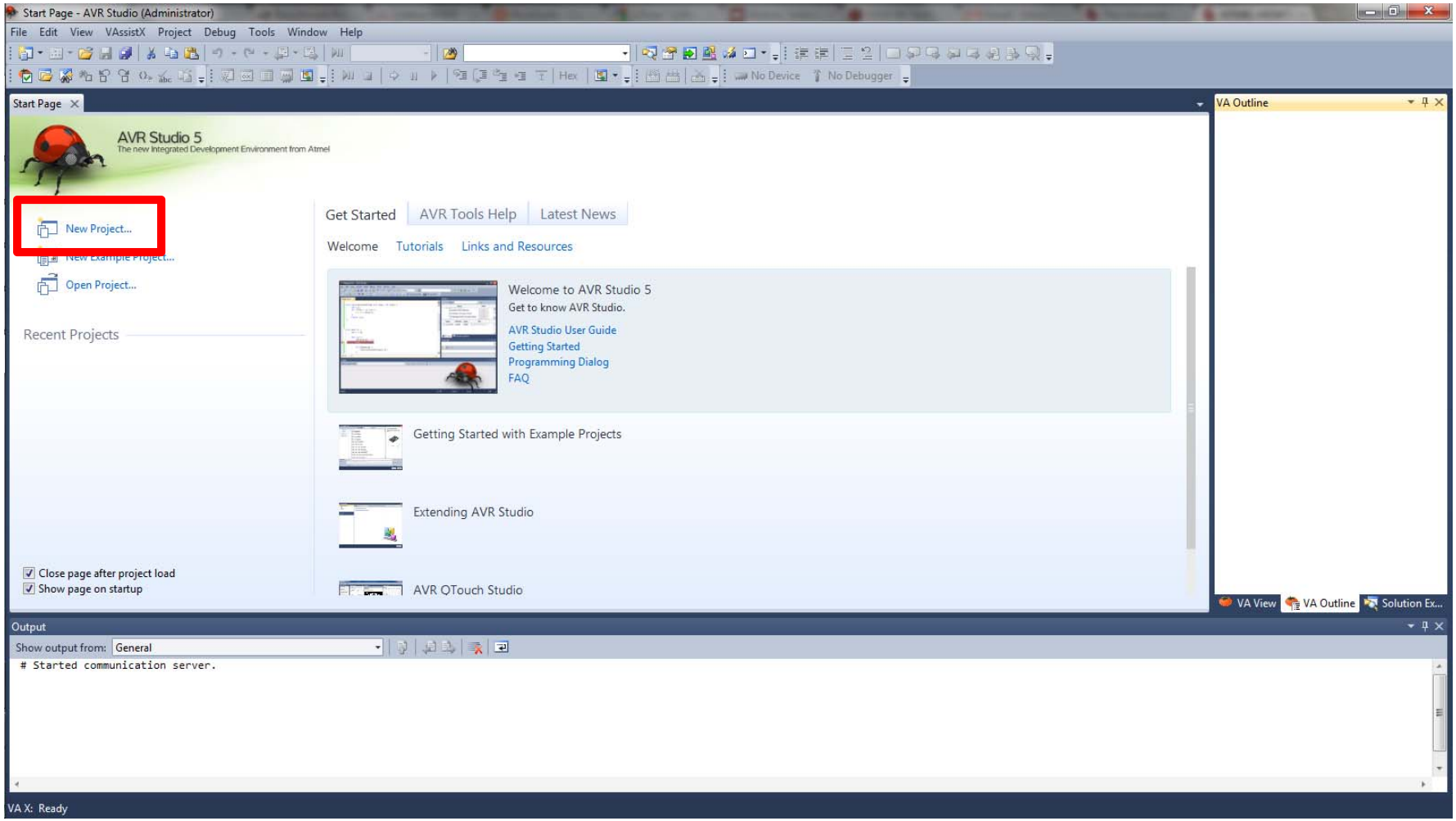
# Plan for Today

- Start working through exercise 1
  - All group members must show some form of LED control
  - Groups need to show some wiring of additional LEDs
- Project 1

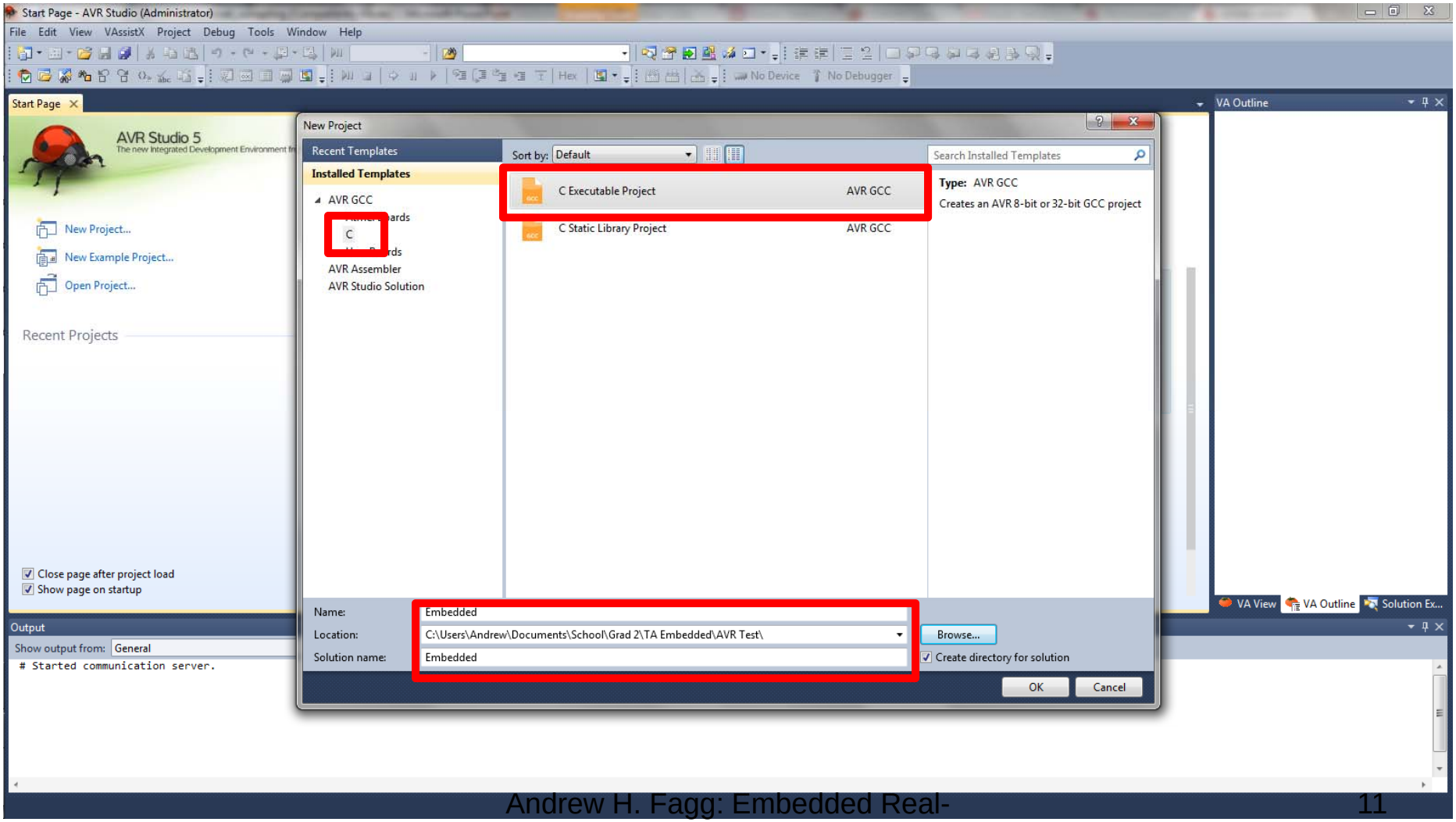
Everyone must demonstrate:

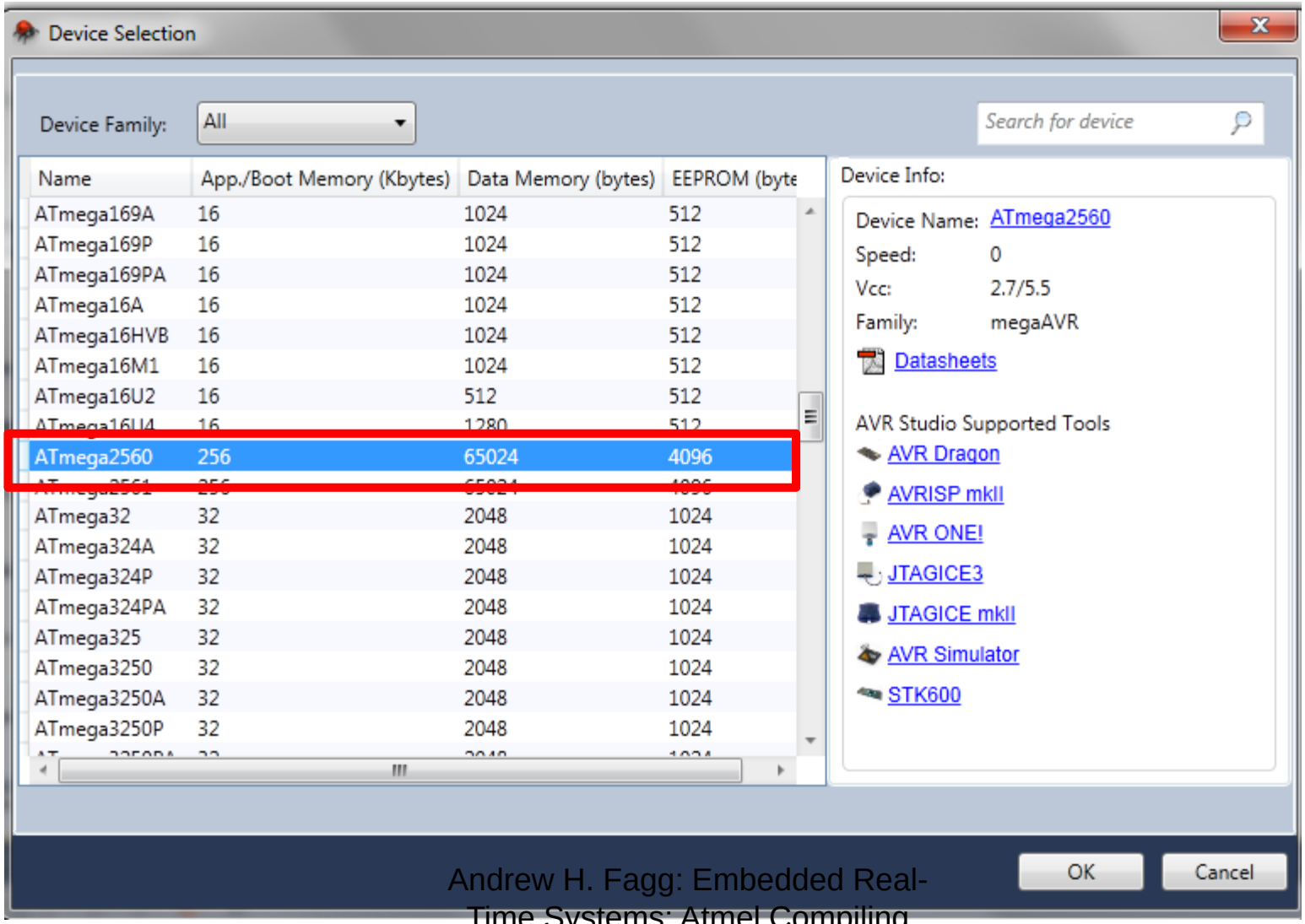
- Svn works
- Compiling/downloading to Atmel works

# Windows: Getting Started

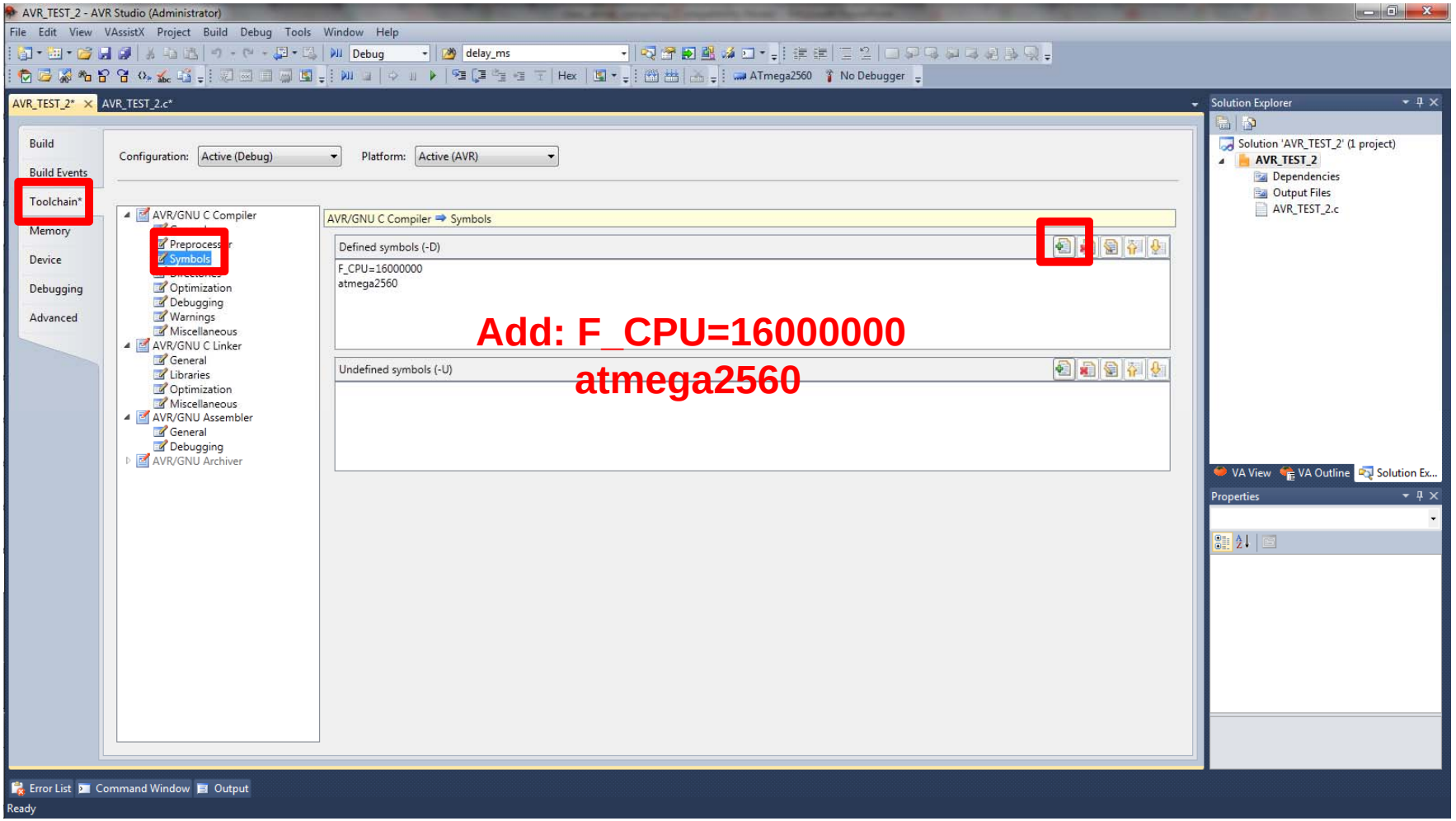


# New Project

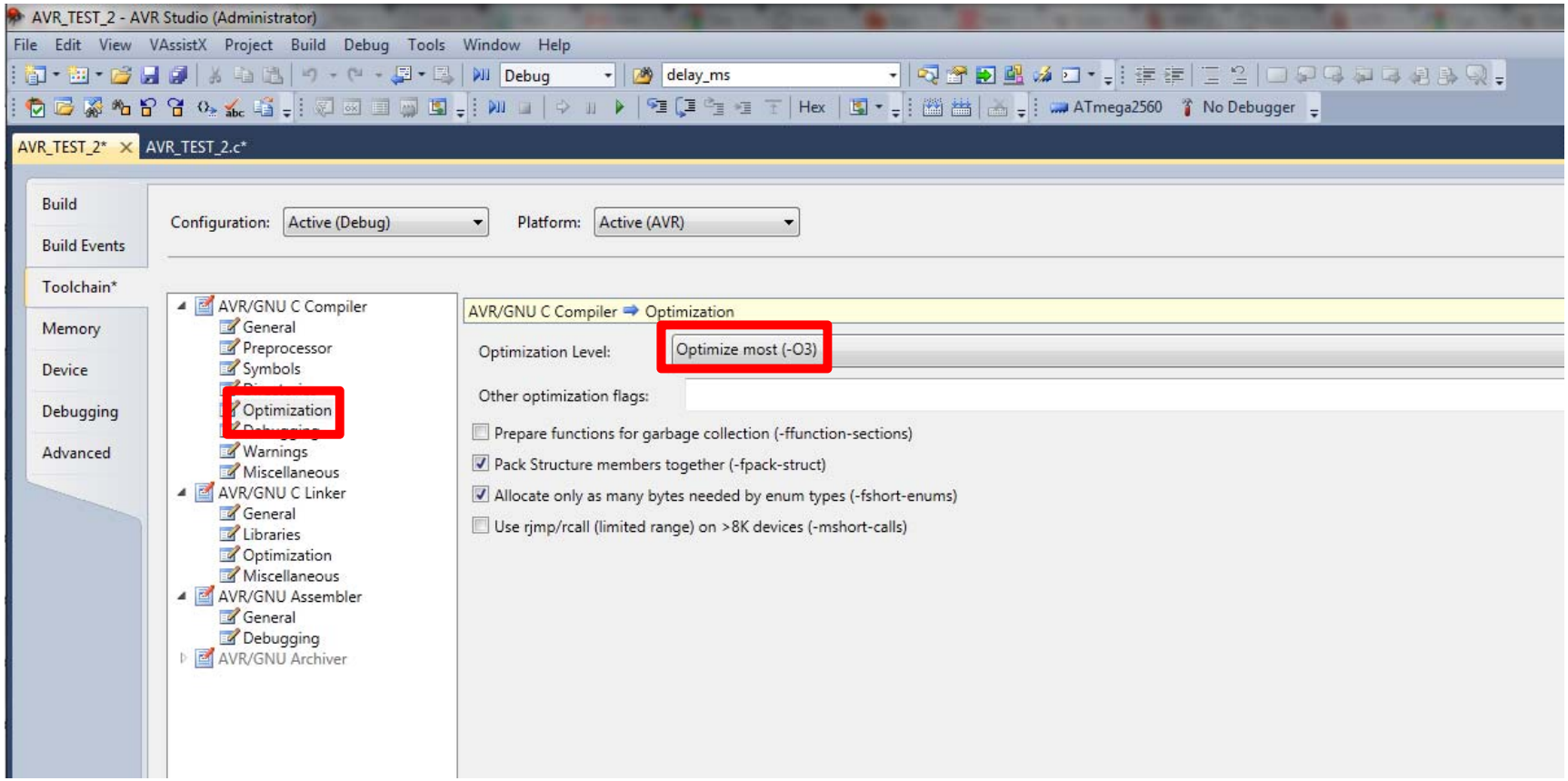

Start Page - AVR Studio (Administrator)
File Edit View VAssistX Project Debug Tools Window Help
Start Page
AVR Studio 5
New Project...
New Example Project...
Open Project...
Recent Projects
Close page after project load
Show page on startup
New Project
Recent Templates
Installed Templates
AVR GCC
C
AVR Assembler
AVR Studio Solution
Sort by: Default
C Executable Project AVR GCC
C Static Library Project AVR GCC
Search Installed Templates
Type: AVR GCC
Creates an AVR 8-bit or 32-bit GCC project
Name: Embedded
Location: C:\Users\Andrew\Documents\School\Grad 2\TA Embedded\AVR Test\
Solution name: Embedded
Browse...
Create directory for solution
OK
Cancel
VA Outline
VA View
VA Outline
Solution Ex...
Output
Show output from: General
# Started communication server.


Device Selection
Device Family:
All
Search for device
Name
App./Boot Memory (Kbytes)
Data Memory (bytes)
EEPROM (byte
ATmega169A 16 1024 512
ATmega169P 16 1024 512
ATmega169PA 16 1024 512
ATmega16A 16 1024 512
ATmega16HVB 16 1024 512
ATmega16M1 16 1024 512
ATmega16U2 16 512 512
ATmega2560 256 65024 4096
ATmega32 32 2048 1024
ATmega324A 32 2048 1024
ATmega324P 32 2048 1024
ATmega324PA 32 2048 1024
ATmega325 32 2048 1024
ATmega3250 32 2048 1024
ATmega3250A 32 2048 1024
ATmega3250P 32 2048 1024
Device Info:
Device Name: ATmega2560
Speed: 0
Vcc: 2.7/5.5
Family: megaAVR
Datasheets
AVR Studio Supported Tools
AVR Dragon
AVRISP mkII
AVR ONE!
JTAGICE3
JTAGICE mkII
AVR Simulator
STK600
OK
Cancel

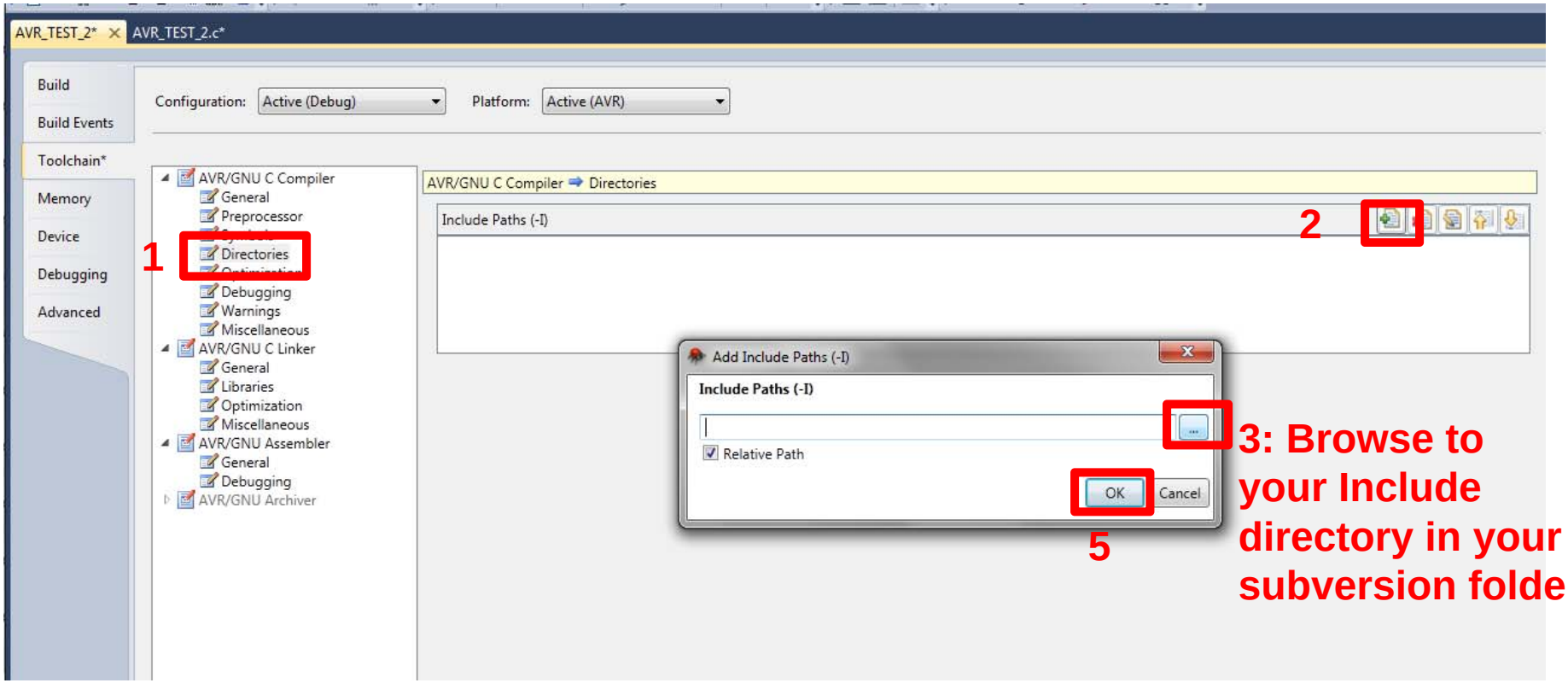
# Project➔
# <Project Name> Properties (Alt+F7)



# Compiler Optimization

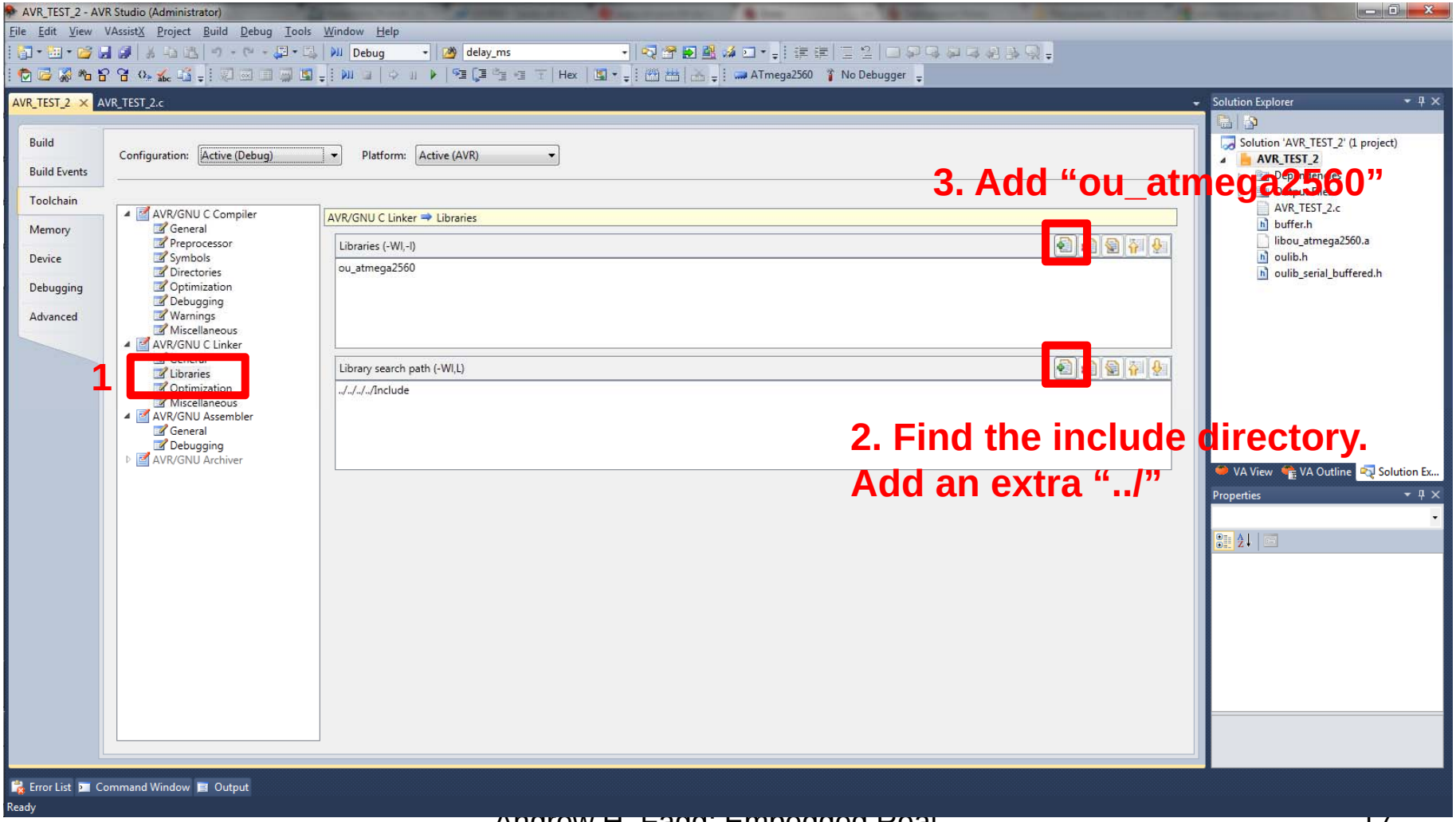


# Add Directories

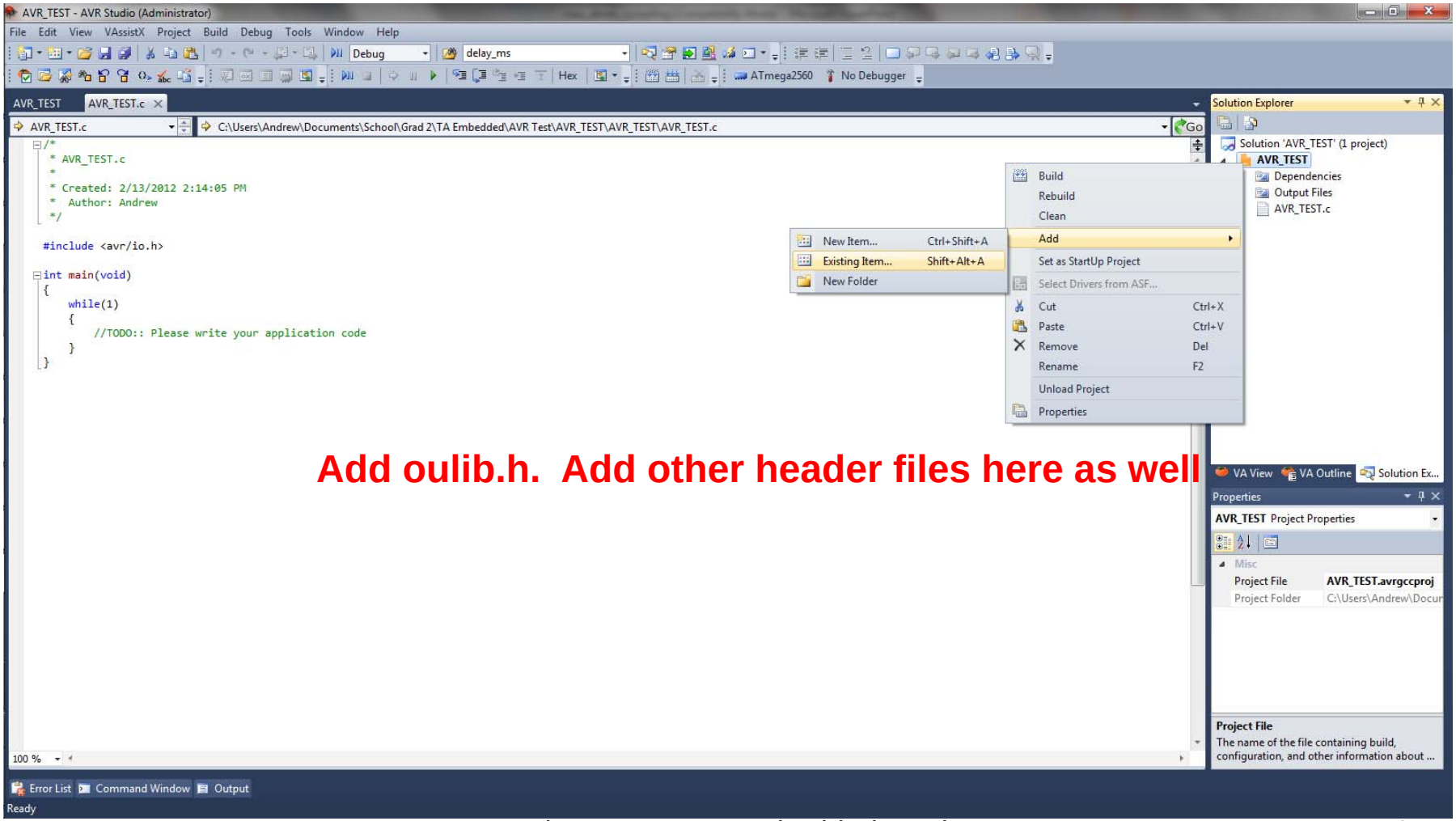



**4 Add "../" to the text it generates**

# Add Libraries



# Add Header Files

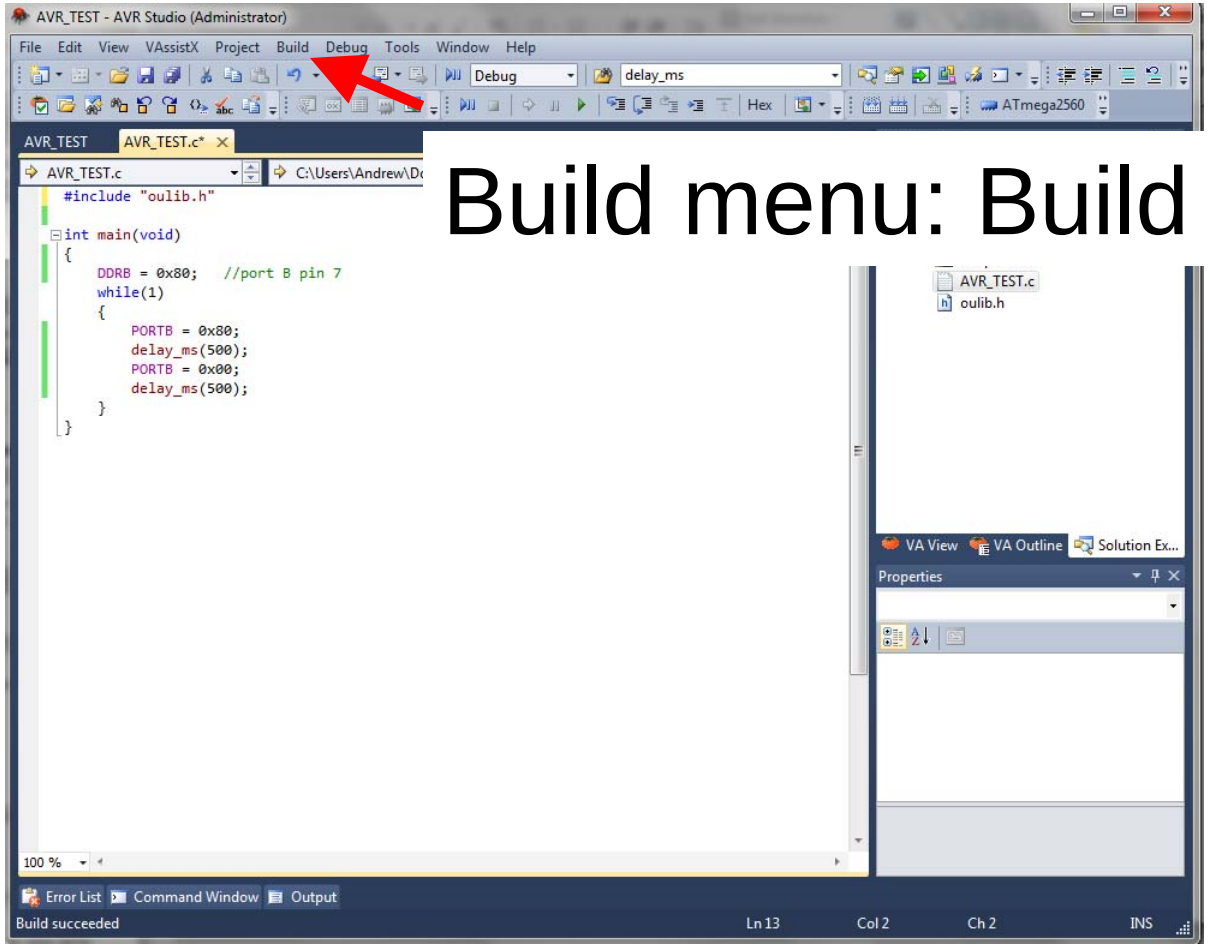


# Now for the code…

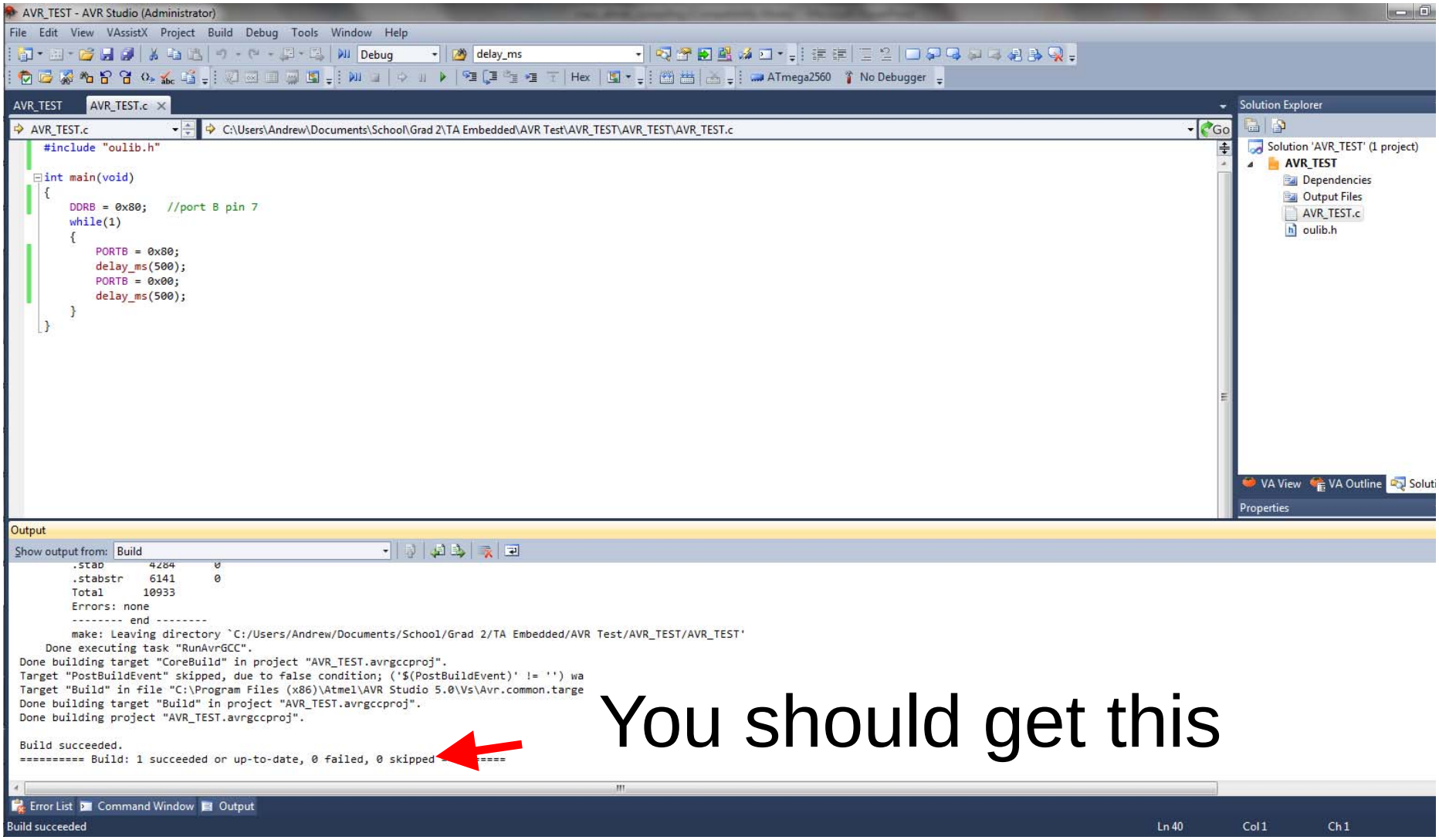
```
#include "oulib.h"

int main(void)
{
   DDRB = 0x80;        // port B, pin 7

   while(1) {
       PORTB = 0x80;
       delay_ms(500);
       PORTB = 0x00;
       delay_ms(500);
   }
}
```




# Build menu: Build

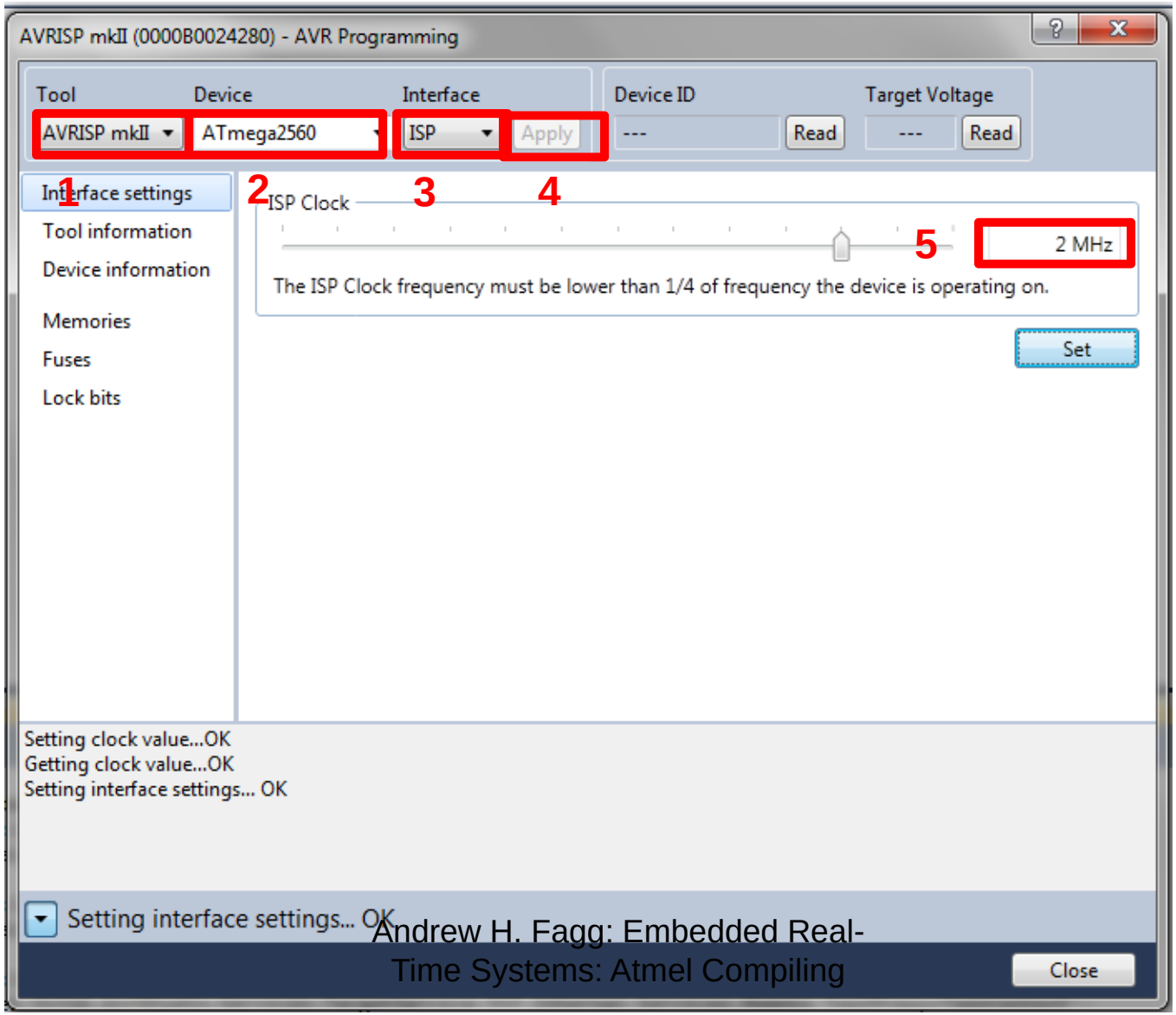


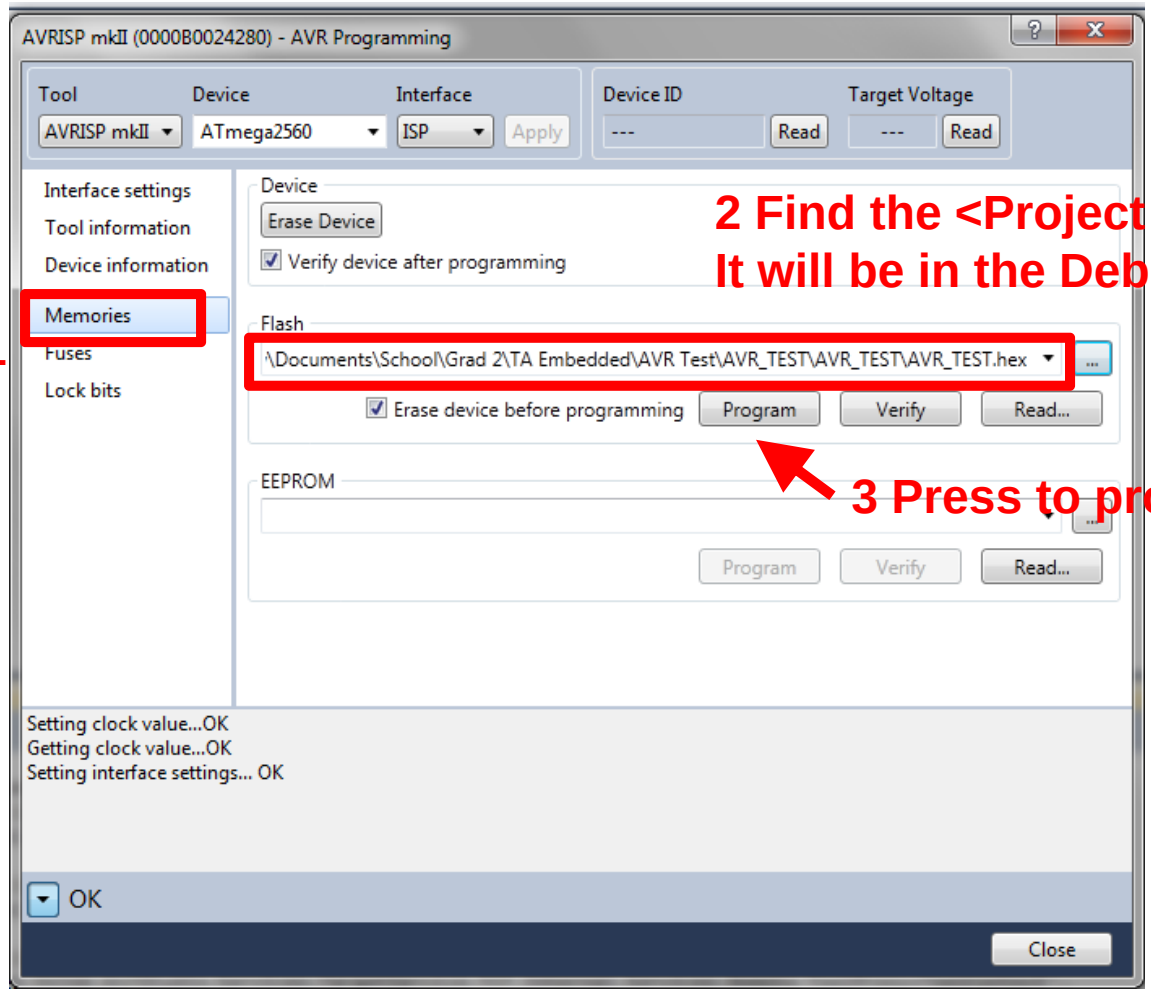

You should get this

# Now We Are Ready…

- Plug the programmer into your computer **and** into the Arduino board (If it is not already)
- Make sure your Arduino board has power
  - Either from USB or batteries
- And download the program…
  - Tools Menu: AVR Programming

# Select the AVR Mk II




AVRISP mkII (0000B0024280) - AVR Programming
Tool
AVRISP mkII
Device
ATmega2560
Interface
ISP
Apply
Device ID
---
Read
Target Voltage
---
Read
Interface settings
Tool information
Device information
Memories
Fuses
Lock bits
Device
Erase Device
Verify device after programming
Flash
\Documents\School\Grad 2\TA Embedded\AVR Test\AVR_TEST\AVR_TEST\AVR_TEST.hex
Erase device before programming
Program
Verify
Read...
EEPROM
Program
Verify
Read...
Setting clock value...OK
Getting clock value...OK
Setting interface settings... OK
OK
Close
1
2 Find the <Project Name>.hex file
It will be in the Debug folder
3 Press to program

# Flashing?

Your program will start executing as soon as the download is complete …

Your on-board Light Emitting Diode should be blinking at 1 Hertz (once per second)

# Next Task

- Add several more LEDs in a line
- Write a program that turns the LEDs on in sequence