Lab 9: C programming with AVR Studio 4

The goal of this lab is to introduce you to "C" programming language. We will repeat the concepts from previous labs and use "C" instead of assembly programming. There are two parts to this lab. The first one introduces you to a blink program and the second one is the LCD display program. **Submit lab9.c at the end of your lab.**

PART I: Lab Evaluations

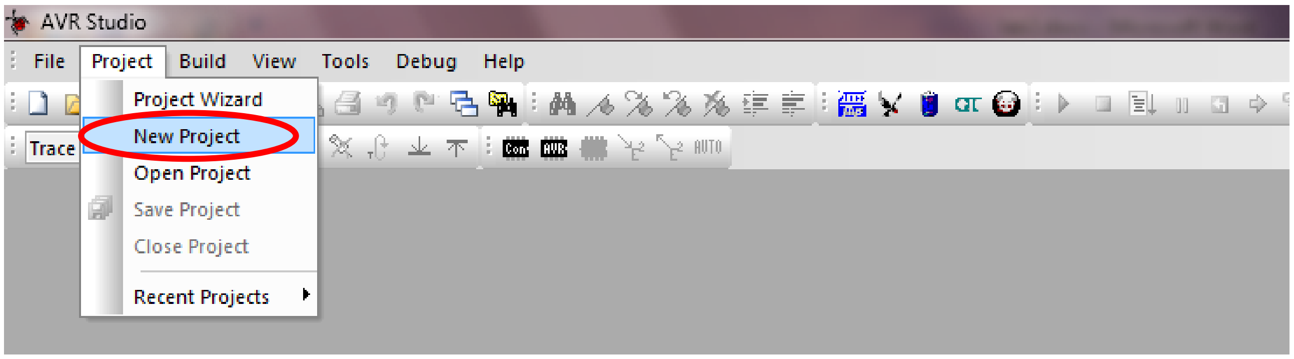
- Lab Evaluations URL: https://evals.csc.uvic.ca/
- Lab Evaluations instruction file is in the same directory as lab9.
- The full names of the three Lab instructors this semester are Tom Arjannikov, Victoria Li and Adithya Rathakrishnan. Please select the instructor for lab you have been attending for your evaluation.

PART II: LED blinking

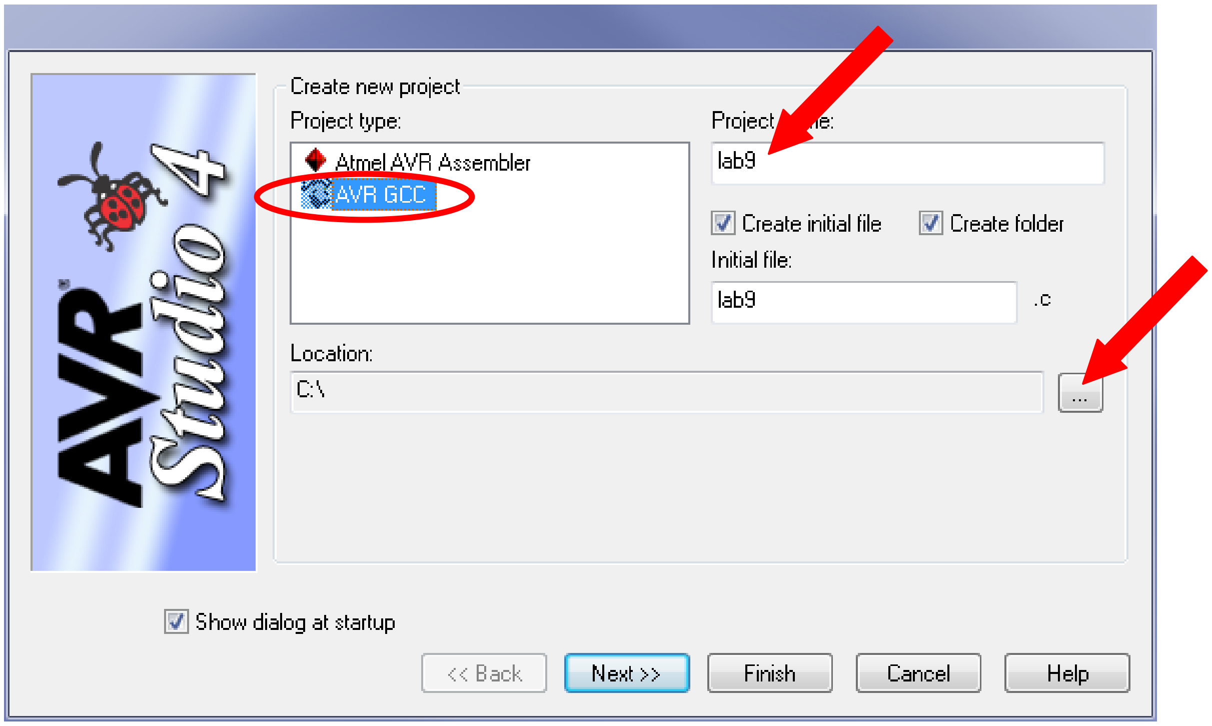
I. Create a C project in AVR Studio 4

Start the AVR Studio 4.

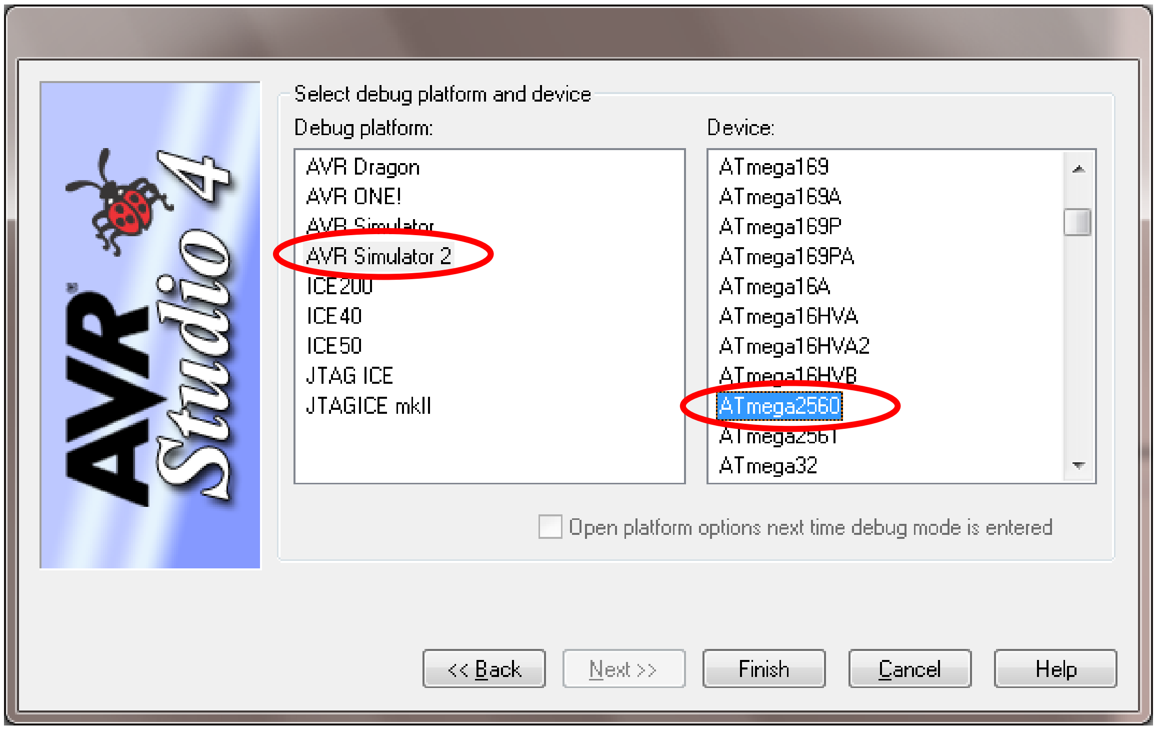
- Select "New Project".



- Select "Project Type" →AVR GCC, name the project "lab9", click "…" button to choose a location (H drive)



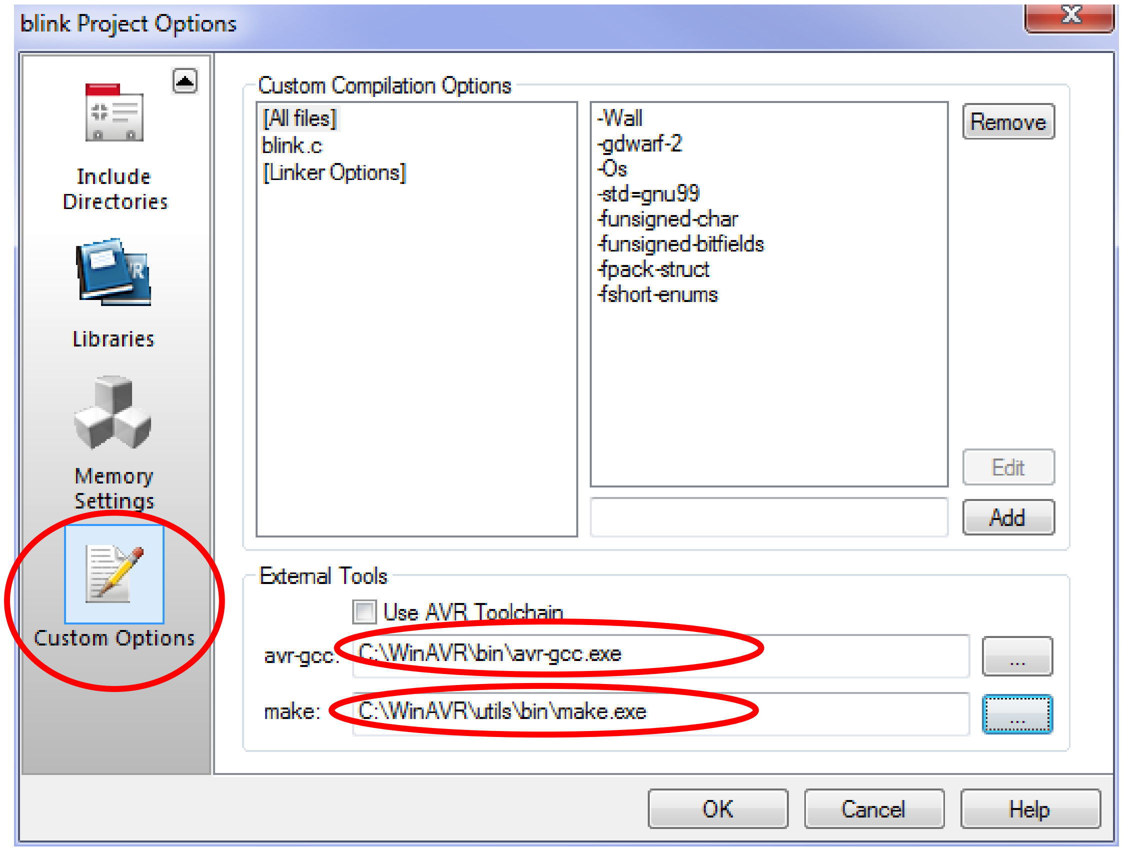
- Go to → Next.
- Select a platform: Debug Platform → AVR Simulator 2
- Select a Device: Device → ATmega2560. Click the "Finish" button.



II. Compile C project

After creating a C project, it opens up the project space. There are issues with the avr-gcc compiler within the AVR Studio. Hence we need to replace this. **This step needs to be done for every project you create**.

- Go to Project → Configuration Options
- Go to the last item "Custom Options" in the left plane.
- Uncheck "Use AVR Toolchain" under External Tools at the bottom
- Select the button for "avr-gcc" to point to avr-gcc.exe in the folder "C:\WinAVR\bin\avr-gcc.exe"
- Select the button for "make" to point to make.exe in the folder "C:\WinAVR\utils\bin\make.exe"



blink Project Options

Custom Compilation Options

[All files]
blink.c
[Linker Options]

-Wall
-gdwarf-2
-Os
-std=gnu99
-funsigned-char
-funsigned-bitfields
-fpack-struct
-fshort-enums

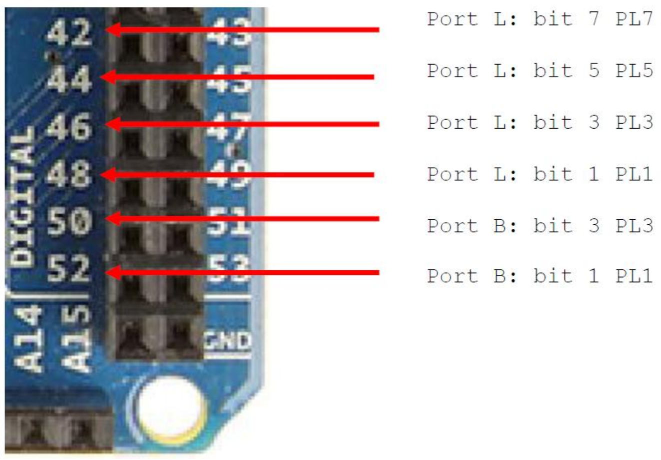
External Tools

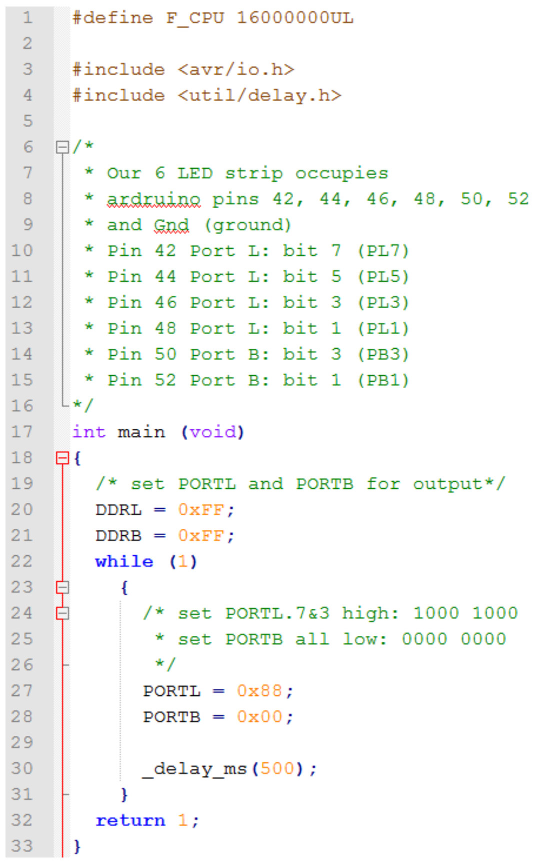
Use AVR Toolchain

avr-gcc: C:\WinAVR\bin\avr-gcc.exe

make: C:\WinAVR\utils\bin\make.exe

- Copy the contents of lab9.c provided to you in the Lab09 folder on connex
- Read and understand the code





```c
#define F_CPU 16000000UL

#include <avr/io.h>
#include <util/delay.h>

/*
 * Our 6 LED strip occupies
 * ardruino pins 42, 44, 46, 48, 50, 52
 * and Gnd (ground)
 * Pin 42 Port L: bit 7 (PL7)
 * Pin 44 Port L: bit 5 (PL5)
 * Pin 46 Port L: bit 3 (PL3)
 * Pin 48 Port L: bit 1 (PL1)
 * Pin 50 Port B: bit 3 (PB3)
 * Pin 52 Port B: bit 1 (PB1)
*/
int main (void)
{
  /* set PORTL and PORTB for output*/
  DDRL = 0xFF;
  DDRB = 0xFF;
  while (1)
    {
      /* set PORTL.7&3 high: 1000 1000
       * set PORTB all low: 0000 0000
       */
      PORTL = 0x88;
      PORTB = 0x00;

      _delay_ms(500);
    }
  return 1;
}
```

- Build the code (F7) or click on the Build
- See the build window below to see if there any possible errors
- If the program compiled with no errors it is time to load the binary hex file to the unit
    - Upload the lab9.hex file to the board by typing the following command at the command window:

      "C:\Program Files (x86)\Arduino\hardware\tools\avr\bin\avrdude" -C "C:\Program Files (x86)\Arduino\hardware\tools\avr\etc\avrdude.conf" -p atmega2560 -c wiring -P COM4 -b 115200 -D -F -U flash:w:filename.hex

      You may need to change the path and filename if the settings are different from this.

    - Or, download runbd.bat file to the debug folder in your lab9.c program folder. Open a command window get to the same directory where runbd.bat is stored, and type runbd.bat. "runbd.bat" is a batch file. It is another way to type a command. You need to change the file name if your project is not called lab9.hex.

III. Exercises:
Download lab9.c.

Implement the following Pseudo-code:

```
turn the first led on;
1 second delay;
while(1)
{
    turn the current led off, turn the next led on; //wrap around when appropriate
    1 second delay;
}
```

Submit lab9.c at the end of your lab.