CSC 230 Software Tools Installation Notes

Sudhakar Ganti, Department of Computer Science, University of Victoria.

Software Installation

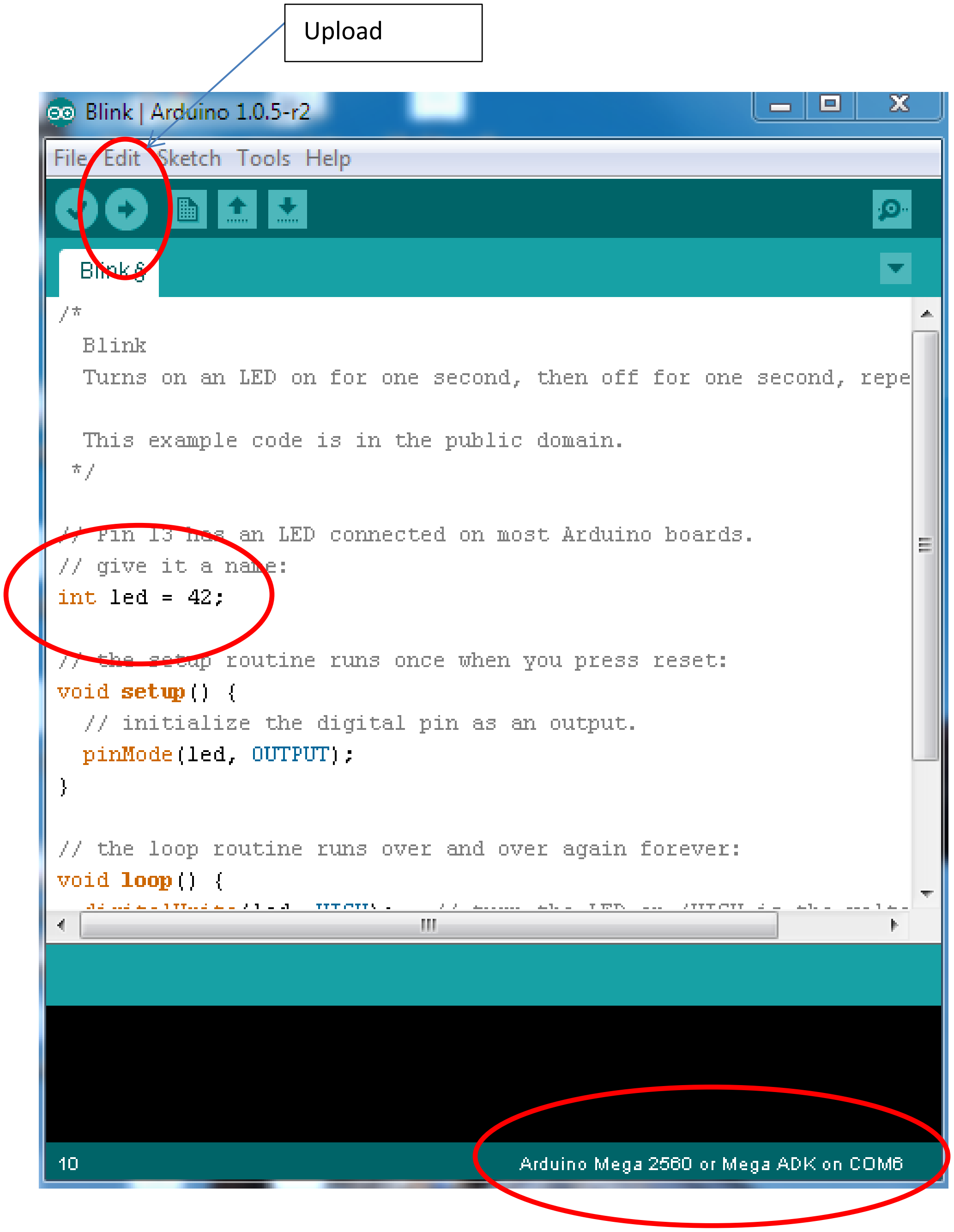
This note is for successful setup of tool chain on Window machines and to be used in conjunction with **Arduino Mega 2560** platform. We are going to use older **AVR studio** (Version 4) as it is lighter install when compared to the new version of studio. One issue with **AVR studio 4** is that the **avrdude** program that comes with the tool chain is also an older version that needs to be updated. We do this first by installing **Arduino IDE** and then copying the **avrdude.exe** and **avrdude.conf** files into respective directories. Also note that the AVR tool chain has issues with C program compilations and hence we will also install **winavr** tool set as a replacement. Please follow these steps:

1. Install **Arduino IDE** by executing **Arduino-1.0.5-r2-windows.exe**
2. Install **AVR Studio 4** by executing **AvrStudio4Setup.exe**
3. Install **AVR Toolchain** by executing **avr-toolchain-installer-3.4.1.1195-win32.win32.x86.exe**
4. Install **WinAVR Toolchain** by executing **WinAVR-install.exe** to the directory C:\WinAVR
5. Backup (rename) **avrdude.exe** and **avrdude.conf** in **C:\Program Files (x86)\Atmel\AVR Tools\AVR Toolchain\bin**
6. Copy **avrdude.exe** from **C:\Program Files (x86)\Arduino\hardware\tools\avr\bin** to above directory (**C:\Program Files (x86)\Atmel\AVR Tools\AVR Toolchain\bin**)
7. Copy **avrdude.conf** from **C:\Program Files (x86)\Arduino\hardware\tools\avr\etc** to **C:\Program Files (x86)\Atmel\AVR Tools\AVR Toolchain\bin**
8. Backup (rename) **avrdude.exe** and **avrdude.conf** in **C:\winAVR\bin**
9. Copy **avrdude.exe** from **C:\Program Files (x86)\Arduino\hardware\tools\avr\bin** to above directory (**C:\WinAVR\bin**)
10. Copy **avrdude.conf** from **C:\Program Files (x86)\Arduino\hardware\tools\avr\etc** to **C:\WinAVR\bin**
11. Make sure that C:\WinAVR\bin; C:\WinAVR\utils\bin; are the first entries in your Windows Path variable.

Software Testing: Arduino IDE

1. It is assumed that you have a setup similar to our lab setup (Arduino Mega 2560 R3), DFrobot LCD Shield and a 6 LED strip. Make sure the LCD shield and LED strip are connected appropriately. It is assumed that LED strip will seat in a row of pins starting from 42.
2. Start the Arduino IDE by double clicking the desktop shortcut
3. Make sure that your Arduino Mega is detected by the IDE as shown in picture below. The com port should be the right one to which the Arduino board is connected.
4. Open the basic blink sketch **File→Examples→01.Basics→Blink**
5. Change the led pin value to 42 (used by one of the led strip) as shown in the picture below.

6. Upload your sketch to the board. The LED on pin 42 should now blinks. You can change the LED pin value to 44, 46, 48, 50, 52 to check for the remaining LEDs. You can also play around with delay values for on and off durations to see how the blink rates change.



```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repe

  This example code is in the public domain.
 */

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 42;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the volta
```

```
10                                  Arduino Mega 2560 or Mega ADK on COM6
```

Software Testing: AVR Studio4

Now we will take the same blink project and work with actual ports. We will use AVR Studio 4 for this purpose. Start the AVR Studio 4.

- Select “New Project”.
- Select “Project Type” →AVR GCC .
- Give a project name (say blink).
- Give an initial file name (say blink) if it is not filled.
- Go to → Next.
- Select a platform: Debug Platform → AVR Simulator 2
- Select a Device: Device → ATmega2560

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”
- Copy the contents of blink.c provided to you in the Lab4 folder on connex
- Read the code and understand
    - Check PORTL is used for output by writing 1s to DDRL
    - Check a specific bit is toggled to make LED blink
- 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
- AVR dude is the program we use to upload the binary program to the board
- Go to the **default** folder from the project directory
- Open a command window from this directory
- avrdude takes specific parameter. You need to specify what com port your device is (COM4?) and what is the hex file.
- Copy the command provided to you the way it is and change hex file name and COM port if necessary
    - c:\WinAVR\bin\avrdude -CC:\WinAVR\bin\avrdude.conf -v -v -v -v -patmega2560 -cwiring -P\\.**COM4** -b 115200 -D -F -U flash:w:**filename.hex**
    - Don’t change anything else in this command. Otherwise the program will not work!!

Note that if you do not have access to Arduino board, you can still use the AVR Studio 4 for compilation and use the built in simulator to check the functionality of the code.