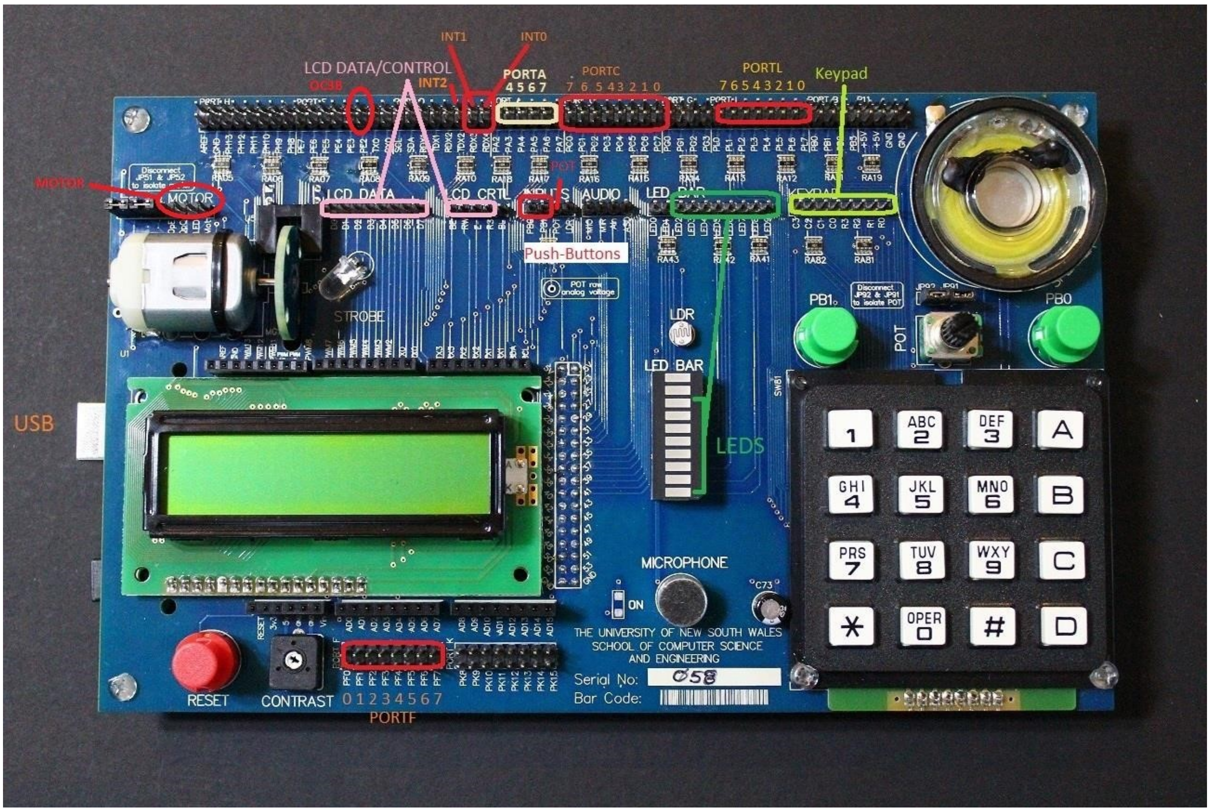
Getting Started with the COMP2121 board

Introduction



Top row and bottom rows of pins map to ATmega 2560 pins. The middle row of pins map to corresponding Input/Output devices on the board. In the top and bottom rows, 2 pins are present above each label and note that both pins are the same.

Note that some labels on the board are not correct. Strictly follow the provided wiring guidelines.

Installing the drivers and flashing tools

Skip this step if you are using a lab computer – they have the drivers and flashing tools already installed. On your personal computers, download and install the Arduino IDE (from the link under lab 3 on the course page). This requires a Windows operating system. You can set up a virtual machine if you do not have Windows (see Appendix A at the bottom).

Connecting to the Computer

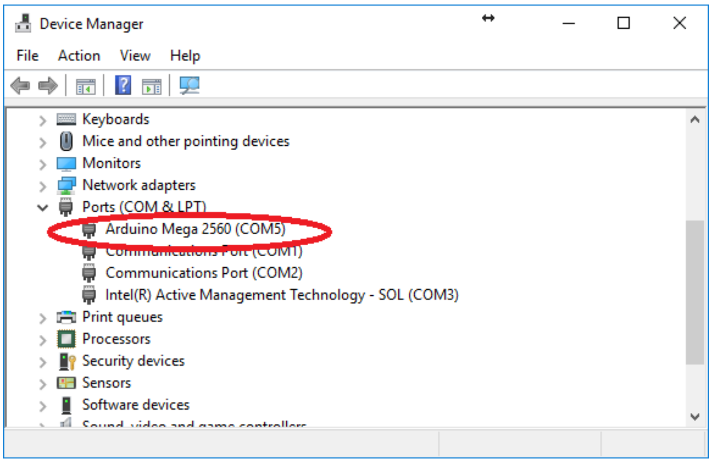
To connect the board to your computer, use the USB Type-B port on the Arduino (left). A green light should be visible on the Arduino board on the back of the development board.

Connecting the LEDs

Use the provided cables to make connections from Port C to the LEDs. To connect correctly, none of the wires needs to cross over (Connect LED2-LED9 to PC0-PC7 respectively).

Find COM port number

To work out which COM port the Arduino is connected to, open Device Manager. You may get a warning about administrator privileges that you can ignore. Scroll down to ‘Ports (COM & LPT)’. There should be one item called ‘Arduino Mega 2560 (COMX)’. See the example figure below.



Programming

- Download the file ‘Downloader’ from the Lab3 on the course website.
- Extract the files and open the folder.
- Run the batch file ‘console.bat’ to open up a command prompt.
- To load the test program, run use the download command:

  ```
  download COMX test.hex
  ```

  COMX is the name of the port you found earlier. Eg: COM3 or COM16

- If the download is successful and the LEDs are connected correctly, the LEDs should start flashing one by one.
- To download a different hex file, replace ‘test.hex’ with the path to the other file.

Note: There is a way to directly flash from the Atmel Studio if you want. See Appendix B below.

Board Test Procedure

Wiring

For the design to perform correctly, the following connections should be made. These connections are described in terms of the labelling on the board.

AVR Pins (top and bottom row)		Input/Output Device Pins (middle row)	
Port Group	**Pin**	**Port Group**	**Pin**
PORT F	PF0	LCD DATA	D0
PORT F	PF1	LCD DATA	D1
PORT F	PF2	LCD DATA	D2
PORT F	PF3	LCD DATA	D3
PORT F	PF4	LCD DATA	D4
PORT F	PF5	LCD DATA	D5
PORT F	PF6	LCD DATA	D6
PORT F	PF7	LCD DATA	D7
PORT K	PK8	INPUTS	POT
PORT K	PK9	INPUTS	LDR
PORT K	PK10	AUDIO	MiO
PORT E	PE5	LCD CTRL	BL
PORT E	PE3	AUDIO	Ain
PORT E	PE2	MOTOR	Mot
PORT D	TDX2	MOTOR	OpO
PORT D	RDX3	INPUTS	PB1
PORT D	RDX4	INPUTS	PB0
PORT A	PA2	-	-
PORT A	PA3	MOTOR	LED
PORT A	PA4	LCD CTRL	BE
PORT A	PA5	LCD CTRL	RW
PORT A	PA6	LCD CTRL	E
PORT A	PA7	LCD CTRL	RS
PORT C	PC0	LED BAR	LED2
PORT C	PC1	LED BAR	LED3
PORT C	PC2	LED BAR	LED4
PORT C	PC3	LED BAR	LED5
PORT C	PC4	LED BAR	LED6
PORT C	PC5	LED BAR	LED7
PORT C	PC6	LED BAR	LED8
PORT C	PC7	LED BAR	LED9
PORT G	PG0	-	-
PORT G	PG1	AUDIO	ASD
PORT G	PG2	LED BAR	LED0
PORT G	PG3	LED BAR	LED1
PORT L	PL0	KEYPAD	C3
PORT L	PL1	KEYPAD	C2
PORT L	PL2	KEYPAD	C1
PORT L	PL3	KEYPAD	C0
PORT L	PL4	KEYPAD	R3
PORT L	PL5	KEYPAD	R2
PORT L	PL6	KEYPAD	R1
PORT L	PL7	KEYPAD	R0
P11	+5V (any)	MOTOR	OpE

Test Procedure

You will be given a sheet with a list of behaviours to test. Mark the correct box to indicate success or failure. The behaviours to test are also listed below.

Component	**Test**	**Pass**	**Fail**
Program	Board can be programmed successfully		
LED	LED bar displays single moving LED pattern when '1' is pressed		
	LED bar displays alternating pattern when '2' is pressed		
	LED bar displays filling bar pattern when '3' is pressed		
Reset	Reset button works		
Motor	Motor turns on when '4' is pressed		
	Motor turns off when '5' is pressed again		
LCD	LCD backlight turns on when '7' is pressed		
	LCD backlight turns off when '8' is pressed		
	LCD backlight brightness fades on and off when '9' is pressed		
Strobe	Strobe LED turns on when PB0 is pressed (right push button)		
	Strobe LED turns off when PB1 is pressed (left push button)		
LCD	Bottom right corner displays last keypad button pressed		
	Display for all keypad buttons is correct		
	Top right corner shows '0' when motor is off		
	Top right corner shows motor speed (100-200) when motor is on		
	Top left corner shows potentiometer voltage (eg P0000 or P03FF)		
	Potentiometer voltage display changes when potentiometer is		
	Bottom left corner shows LDR voltage (eg L0260)		
	LDR voltage display increases when LDR is covered with finger		

DO NOT remove the wires once you are done. You will use this wiring throughout the semester.

Appendix A: Flashing the AVR development board on non-windows computers

Method 1

The easiest way is to set up a Windows virtual machine. We recommend Windows 10 on VMware as it has worked quite well for past students.

Obtain necessary software for setting up a Windows virtual machine from the following links.

- VMware Workstation (for Linux) or VMware Fusion (for Mac OS): https://taggi.cse.unsw.edu.au/FAQ/VMware_Academic_Program/
- Microsoft Windows: https://taggi.cse.unsw.edu.au/FAQ/Microsoft_Imagine_and_ELMS/

You can pass the AVR serial port into the virtual machine - in VMware a prompt usually pop up when you plug in a new device to the host. Then flash using the steps that we have given for Windows.

Method 2

Please note that this method is for Windows haters who really do not want to use a virtual machine. Nevertheless, here are some tips, but you should be willing to spend some time setting things up.

1. Install *avrdude* using the package manager or any other method

   In Ubuntu: *apt-get install avrdude*

   In MacOS: *brew install avrdude*

2. Locate the serial port (which is the COMport in Windows)

   Eg: */dev/ttyACMxx in Linux* and */dev/tty.usbmodemxx* in MacOSX where *xx* is a number.
   Note: You can list the files in */dev* before and after plugging the board to observe which device file newly appear.

3. Get the ownership of the port using

   sudo chown username /dev/ttyACMxx

   Note: Ownership of the port is revoked as soon as the board is unplugged. One fix is to call *usermod -a -G dialout <username>* and then restart the computer to get permanent access to the port.

4. Create a shell script *downloader.sh* and add the following to the script.

    For Linux:

    ```
    avrdude -C "/etc/avrdude.conf" -c wiring -p m2560 -P $1 -b 115200 -U flash:w:$2:i -D
    ```

    For MacOSX:

    ```
    avrdude -C "/usr/local/etc/avrdude.conf" -c wiring -p m2560 -P $1 -b 115200 -U flash:w:$2:i -D
    ```

    Note: *avrdude.conf* locations above might differ on your distribution and make sure you locate it and put the correct path. You can you the following command to find a file.

    ```
    find / -name avrdude.conf
    ```

5. Give executable permission to the script.

    ```
    chmod +x downloader.sh
    ```

6. Run the script with correct arguments where the first argument is the port that you found in step 2 and the second argument is the generated hex file.

    ```
    ./downloader.sh /dev/ttyACMxx text.hex
    ```

Note: However, still you should find a way to run the AVR Studio/AVR assembler. Wine might be an option to run the Window's version of AVR studio or the AVR assembler exe. The *avr-gcc* will not work - the assembler syntax is different.

Appendix B: Flashing the Atmel 2560 directly from the Atmel Studio

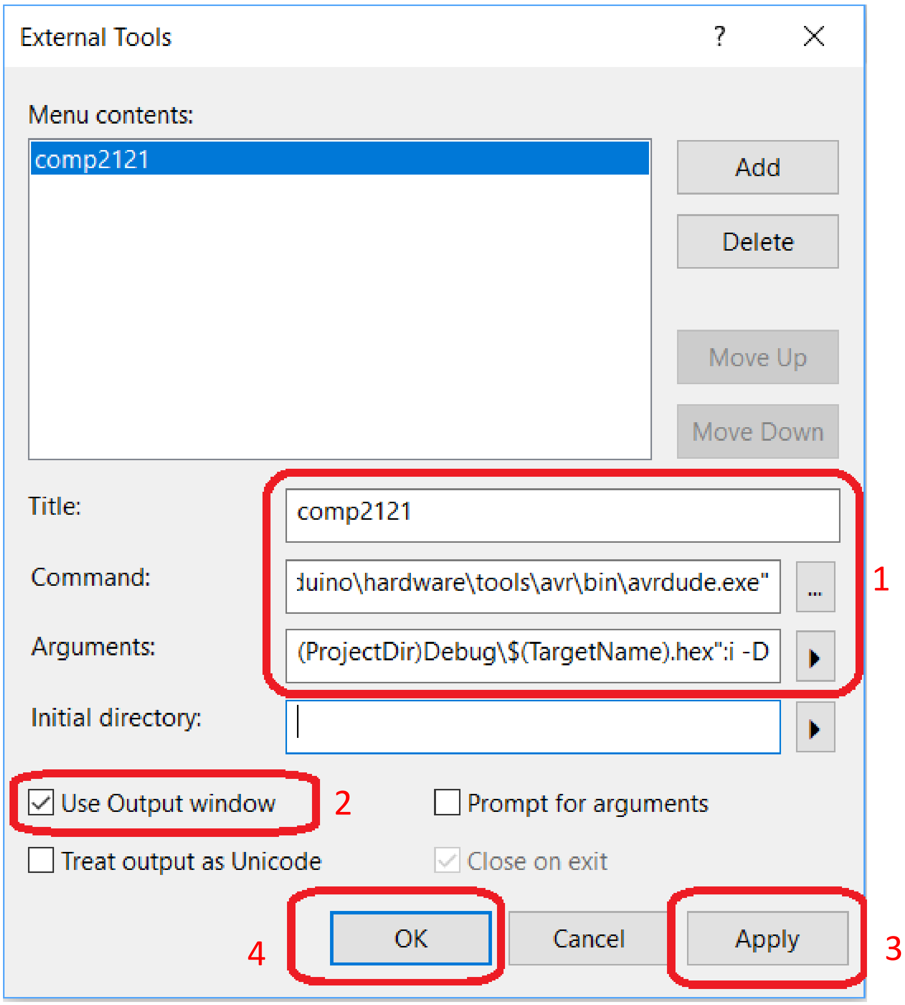
Source : https://www.instructables.com/id/How-to-Load-Programs-to-an-Arduino-UNO-From-Atmel-/

One way to flash the AVR development board (which contains an Arduino Mega board) is to use the provided scripts and calling the command **download COMX file.hex** where *COMX* is your com port and *file.hex* is the hex file generated by your source code (usually located under the project's Debug folder).

However, there is an easy way to flash directly from the Atmel studio which you can set up on your laptops. Steps are as follows:

1. Go to the menu item *Tools->external tools* in Atmel Studio

2. Fill the dialogue box as follows.



External Tools

Menu contents:

comp2121

Title: comp2121

Command: :luino\hardware\tools\avr\bin\avrdude.exe"

Arguments: (ProjectDir)Debug\$(TargetName).hex":i -D

Initial directory:

☑ Use Output window

☐ Prompt for arguments

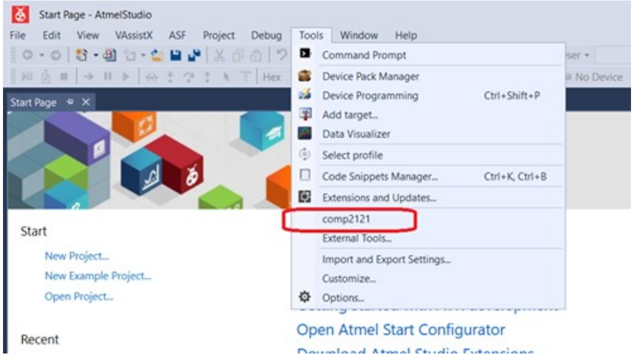
☐ Treat output as Unicode

☑ Close on exit

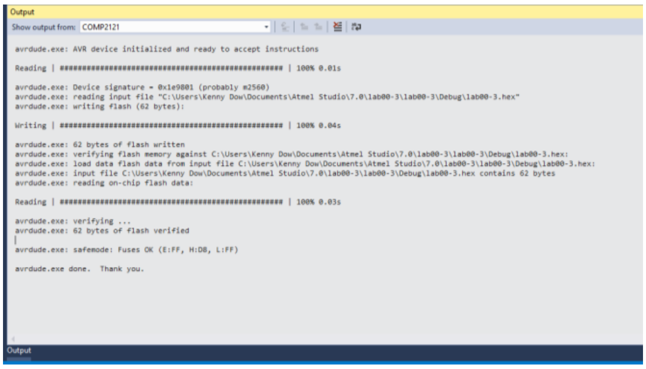
- Give any name you like for the *Title*.
- The *Command* should be the location of the *avrdude* program – the command line program that performs the flashing.
  "C:\Program Files (x86)\Arduino\hardware\tools\avr\bin\avrdude.exe"
- The *Arguments* should be

-C "C:\Program Files (x86)\Arduino\hardware\tools\avr\etc\avrdude.conf" -c wiring -p m2560 -P COMX -b 115200 -U flash:w:"$(ProjectDir)Debug\$(TargetName).hex":i -D
Make sure you change COMX to your COM port number. This COM port number would be consistent as long as you use the same board (probably on the same USB port).

3. Then build your program so that the hex file will be generated.
4. Click on the added tool name.



5. You should see the output of the programmer as below.



```
avrdude.exe: AVR device initialized and ready to accept instructions

Reading | ################################################## | 100% 0.01s

avrdude.exe: Device signature = 0x1e9801 (probably m2560)
avrdude.exe: reading input file "C:\Users\Kenny Dow\Documents\Atmel Studio\7.0\lab00-3\lab00-3\Debug\lab00-3.hex"
avrdude.exe: writing flash (62 bytes):

Writing | ################################################## | 100% 0.04s

avrdude.exe: 62 bytes of flash written
avrdude.exe: verifying flash memory against C:\Users\Kenny Dow\Documents\Atmel Studio\7.0\lab00-3\lab00-3\Debug\lab00-3.hex:
avrdude.exe: load data flash data from input file C:\Users\Kenny Dow\Documents\Atmel Studio\7.0\lab00-3\lab00-3\Debug\lab00-3.hex:
avrdude.exe: input file C:\Users\Kenny Dow\Documents\Atmel Studio\7.0\lab00-3\lab00-3\Debug\lab00-3.hex contains 62 bytes
avrdude.exe: reading on-chip flash data:

Reading | ################################################## | 100% 0.03s

avrdude.exe: verifying ...
avrdude.exe: 62 bytes of flash verified

avrdude.exe: safemode: Fuses OK (E:FF, H:D8, L:FF)

avrdude.exe done.  Thank you.
```