How To Use The Arduino Development Environment

MXEN2002

This guide is for students who are not able to use Atmel Studio at home. It will talk you through how to get the dependencies you will need to program your robot and controller in a similar manner to how you did it in class. Note that whilst the Arduino IDE will be used, we will refrain from using the Arduino programming language in favour of Embedded C.

Contents

1. Installing the Arduino IDE

The Arduino IDE is an open source integrated development environment used to make firmware deployment to an Arduino microcontroller easy. It is free to download, and is officially supported on Windows, OS X and Linux. It may ask you to donate, but this is not necessary and you can skip it. You can get the software from `https://www.arduino.cc/en/Main/Software`

1.1. Windows

The given link will allow you to either download a compressed Zip file, or an executable installer. The executable installer is easier to use as it will install everything that is needed to use the Arduino IDE. Using the Zip file, you will need to manually install the required drivers.

Execute the installer and ensure that all the boxes are ticked as seen in Figure 1. Ensure that the installer is given permission to install the drivers required, it is likely you will get a warning message popup.

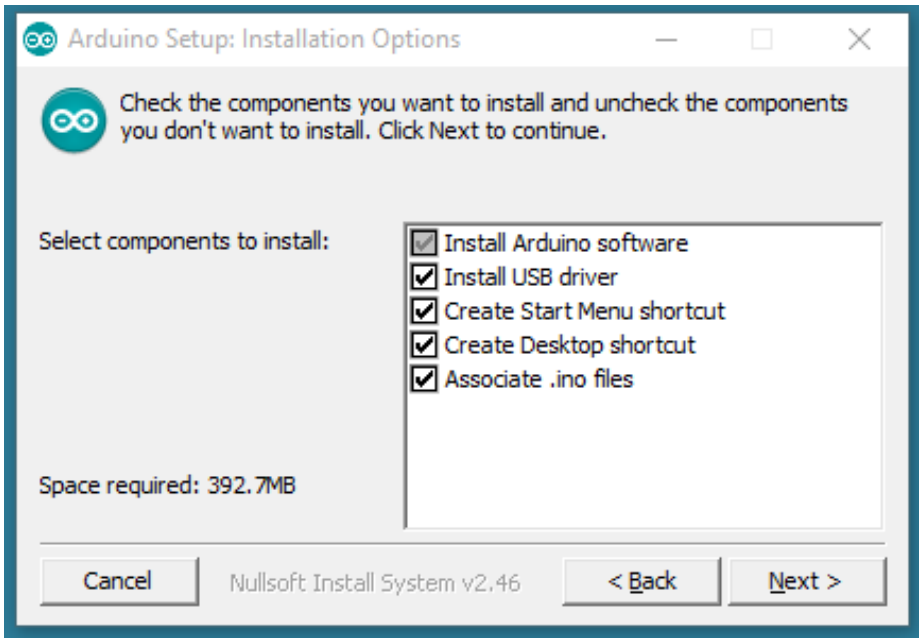

Figure 1: *Components to install*

Choose the installation directory, it is recommended you use the default directory.

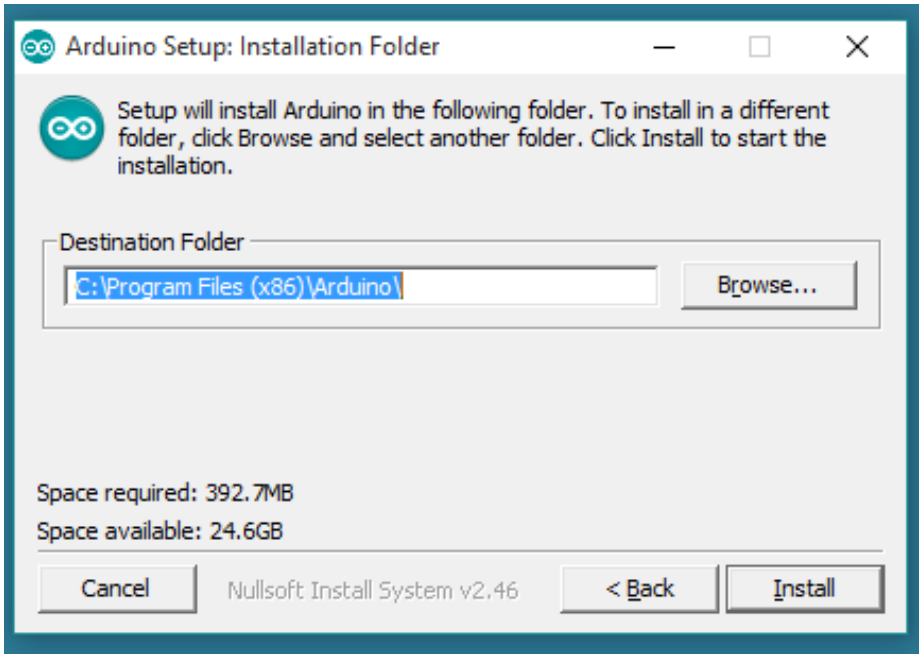

Figure 2: *Install directory*

The process will extract and install all the required files to execute the Arduino IDE.

1.2. OS X

The file downloaded from the given link will be in a compressed Zip format. If you use Safari it will be automatically expanded. If you use a different browser you may need to extract it manually.

Once extracted, you should see the Arduino application similar to Figure 3.

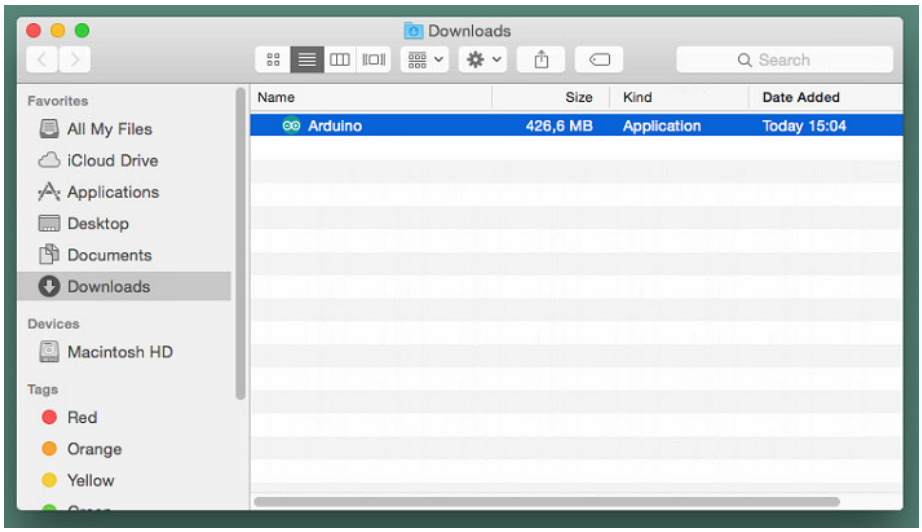

Figure 3: *The extracted Arduino application*

Copy the Arduino application into your Applications folder.

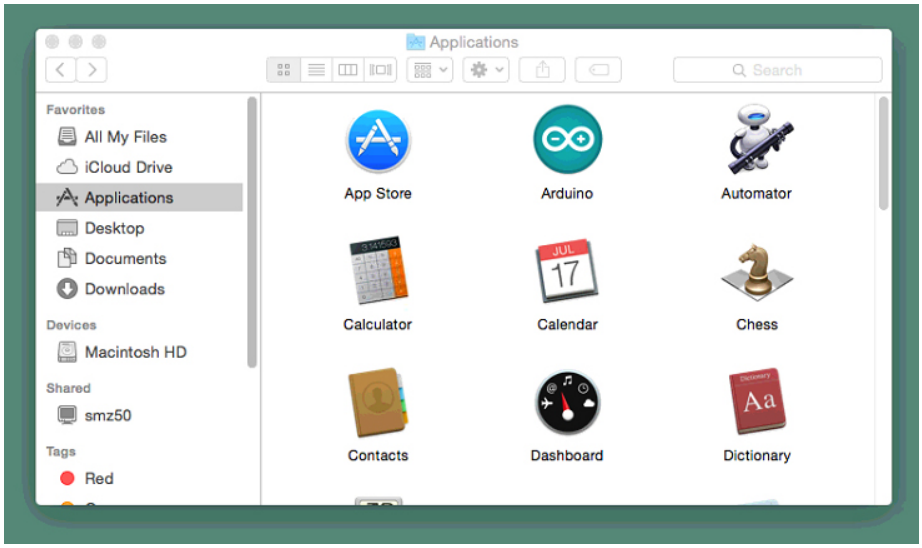

Figure 4: *Copying the Arduino app into Applications*

1.3. Linux

From the given link, ensure that you download the correct version for your Linux distro. A pop up will ask if you want to save the file, click yes. The downloaded file will be in a tarball, and will need to be extracted. Remember where it is extracted to, as this will be the location of the Arduino IDE on your computer.

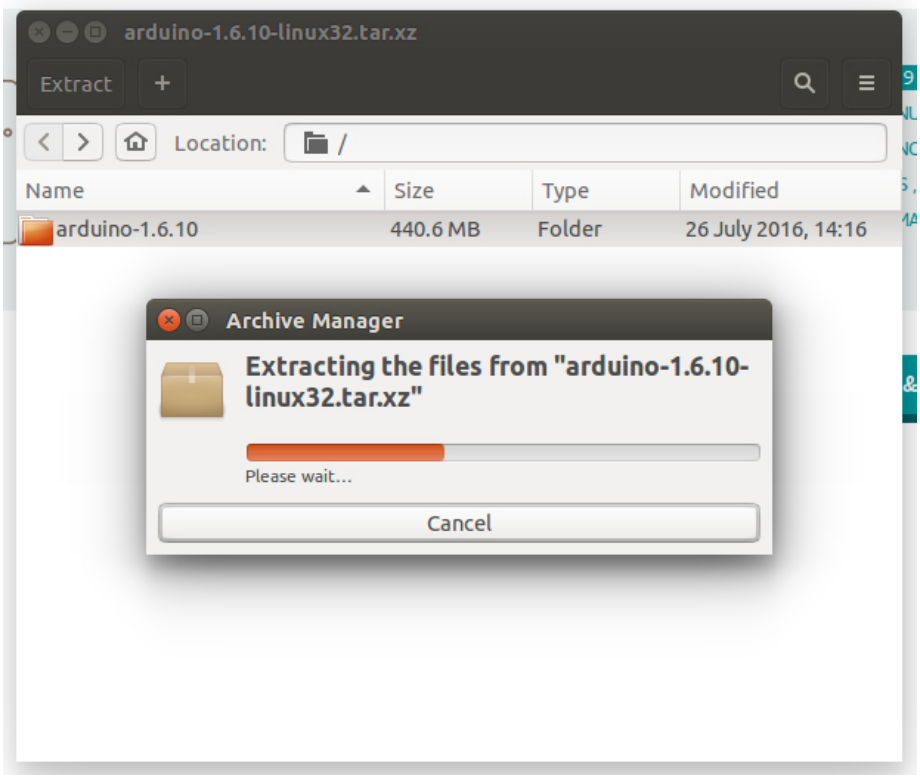

Figure 5: *Extracting the Arduino tarball*

Open a terminal and navigate to the extracted directory. Execute the install script by typing the command:

./install.sh

If an error tells you that the file is not executable, you will need to change the mode:

chmod 755 install.sh

You should see a similar output on your terminal as can be seen in Figure 6, and a shortcut on your desktop.

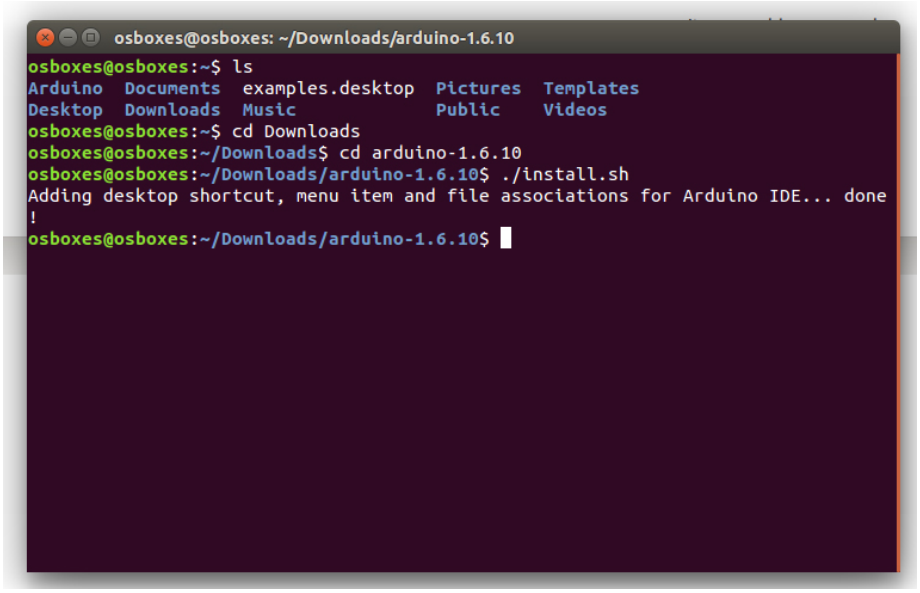

Figure 6: *Installing Arduino*

2. Getting The New Repository

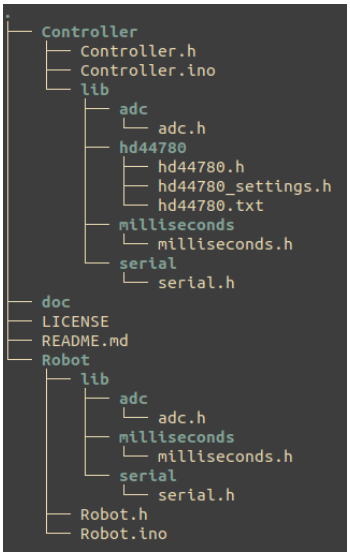

Figure 7: *The hierarchy of mcp-arduino*

Some modifications were made to the original mcp repository that was being used in class to have it work with the Arduino IDE. The new repository can be accessed at:

`https://github.com/mxeng/mcp-arduino`

As with before, on the Github page click on the green "Clone or download" button and download the Zip file. Extract the Zip into your preferred working directory, and you will see three main directory files; Controller, Robot and doc. Controller and Robot you are familiar with, doc contains a copy of this guide. The files inside the Controller and Robot directories are a bit different than the Atmel Studio version, they now have a .ino file in each of them. the .ino extension is the file format that Arduino uses to build its projects, so we will now be working in these files. Inside these directories is also the respective project's header file, as you will have seen previously. Inside the *lib* directory you will see the libraries that previously were in the root directory in the Atmel Studio project. These have been moved and duplicated into each of the project directories due to how Arduino requires libraries to be included. To use library files outside of your project directory (in this case Robot and Controller), you would need to place them into your Arduino install location, under the libraries directory. You will notice that each of the library directories now only consist of the header file rather than the header and c file. This is because the header file and c file have been combined into one, declaring function signitures and providing their implementation in the header file. This is bad programming practice, and I would

not recommended doing so in your own programming endevours - however, due to how Arduino builds projects, these .c and .h files would need to be placed into the Arduino install directory and hence have been consolidated for ease of use.

3. Using The New Repository

After installing the Arduino IDE on your computer, you should be able to double click on the Controller and Robot .ino files to open them in the IDE. In the case of the Robot.ino file, you will see a screen similar to that in Figure 8.

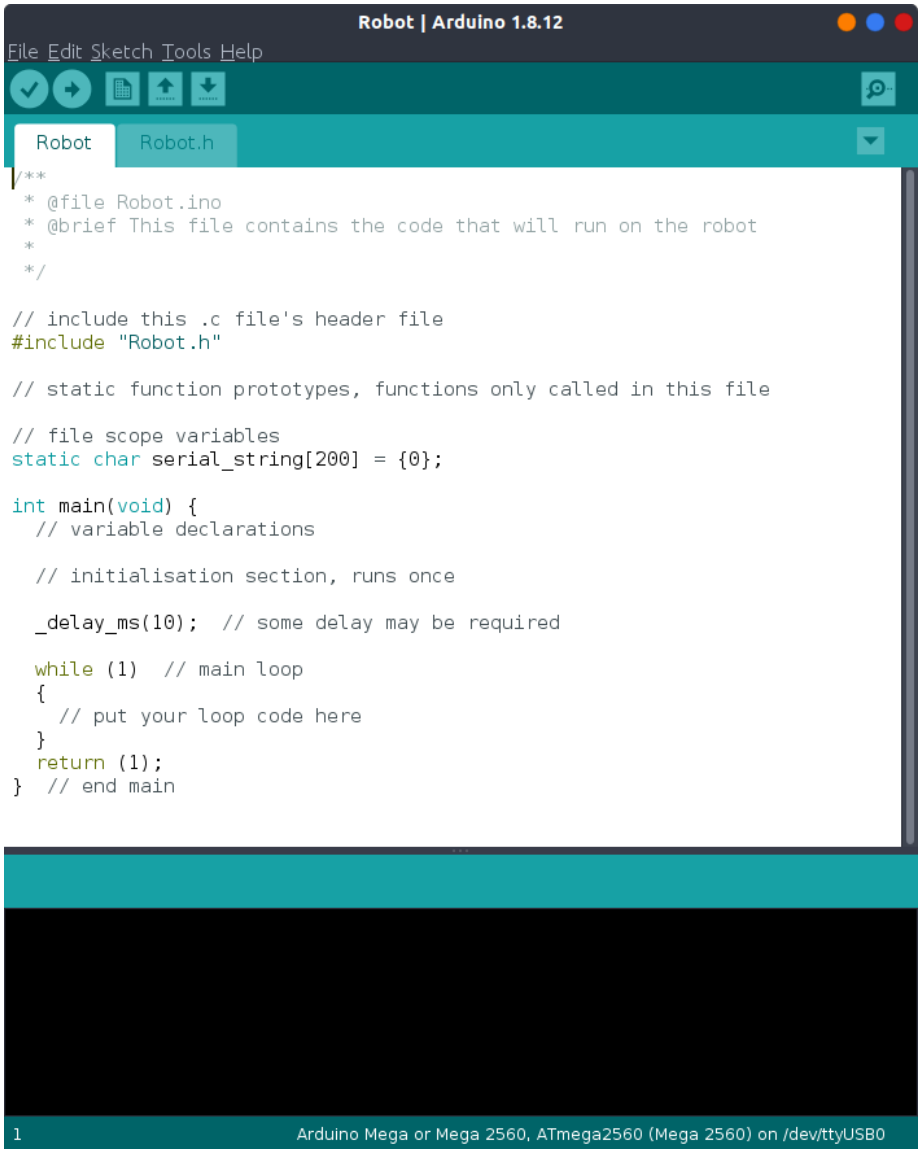

Figure 8: *Robot.ino project default screen*

The white box in the center is where you will do your programming. The black box at the

bottom will give you verbose building information. With the Arduino IDE open, click on the 'Tools' drop down menu in the toolbar. Hover over the 'Board' item, and a popout box will allow you to select 'Arduino Mega or Mega 2560' as seen in Figure 9. **This is an important step, as data direction registers and ports are different on other Arduino boards and can prevent your code from building correctly**. Also ensure that ATmega 2560 is selected for your processor.

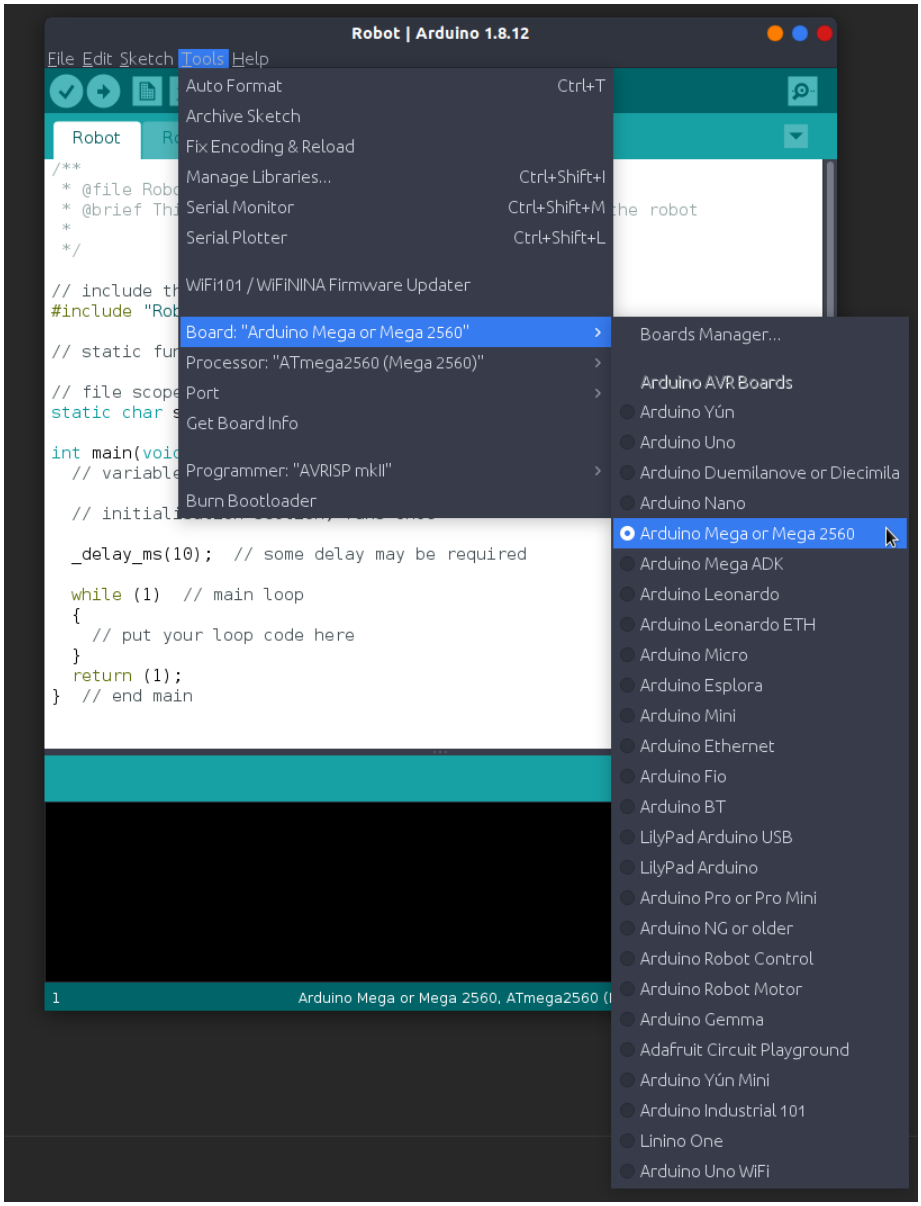

Figure 9: *Board selection*

Finally, once you plug your Arduino into your computer, you need to tell the Arduino IDE what port it has been assigned so it can program it. To do this, again click on the 'Tools' drop down menu in the toolbar. Hover over the 'Port' item, and a popout box will allow you select the port your Arduino is on, similar to Figure 10. On Linux and OS X your ports will start with

/dev/, on Windows they will be listed as COMn where n refers to the port number. Sometimes Arduino will tell you next to the port that a microcontroller is attached there. If it doesn't and you aren't sure, unplug the board and check the ports list. Plug it back in and you should see which port was allocated.

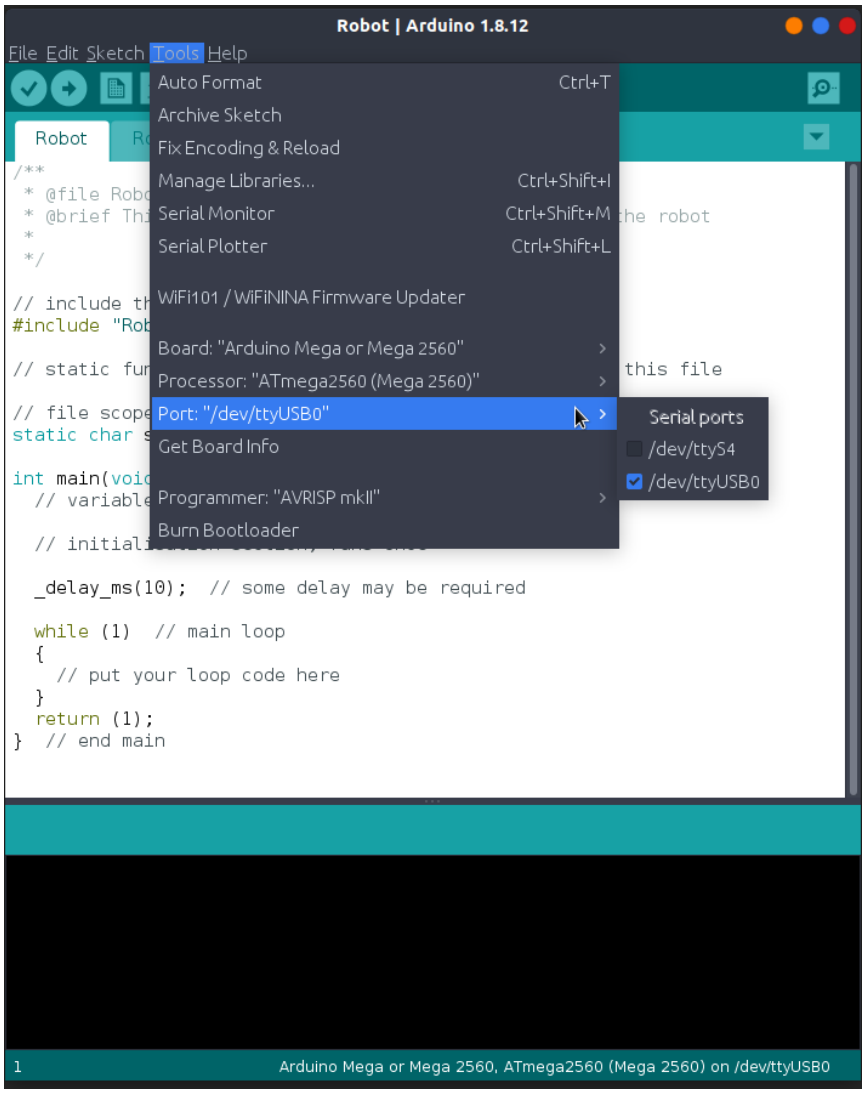

Figure 10: *Port selection*

To verify your code builds successfully, click on the blue tick button in the top left of the screen, seen in Figure 8. With the board and port set up successfuly, you will now be able to click on the blue right arrow button next to the check button so deploy your code to the board. The black box at the bottom will display some orange text, and the blue accent above it will also turn orange if you have some build errors. Use the information displayed here to debug your code.

The last thing to mention is that the magnifying glass in the top right of the screen seen in Figure 8 will open up the serial monitor. This can be used with the 'serial0_print_string' function in the serial library to print messages from your microcontroller to the screen.