



Carleton University
School of Computer Science

COMP 2801 Introduction to Robotics Course Outline (F2025 - Online)



Course Information

Instructor: Mark Lanthier (lanthier@scs.carleton.ca)

The course instructor will be on zoom during the class time. There will be no additional scheduled office hours ... however ... questions may be asked via email. You should ask all your questions during the lab/class time, as you work on the lab assignments.

TAs:

Rayan Choudhary	(RayanChoudhary@cmail.carleton.ca)
Jill Li	(JillLi@cmail.carleton.ca)
Rudy Jarvis	(RudyJarvis@cmail.carleton.ca)
Colin McCormack	(ColinMcCormack@cmail.carleton.ca)
Evan Raz	(EvanRaz@cmail.carleton.ca)
Yasin Waili	(YasinWaili@cmail.carleton.ca)

Class: via zoom (ID posted on Brightspace)

Calendar Course Description

A course on programming simulated mobile robots with various sensors such as wheel encoders, distance sensors, cameras, compasses, accelerometers, and laser range finders. Topics include: programming robot behavior; performing position estimation; implementing algorithms related to navigation, mapping, path planning, area coverage, and localization.

Prerequisite(s): (COMP 1006 or COMP 1406) with a minimum grade of **C-**.

Learning Outcomes

This course will introduce you to the basics of programming robots (in JAVA). Although we will be using 3D simulated robots, the concepts that you will learn will carry forward to real robots. You will learn about common issues that arise when dealing with noisy sensor data and learn to identify invalid readings. You will learn about various timing-related programming issues that are inherent with robot programming. The course will also touch upon some computational geometry algorithms related to path-planning. Here is an overview of the topics that the course will address:

- **Robot Movement and Sensing:** Position Sensor, Compass, Accelerometer, Camera, Laser Range Finder
- **Programming Behaviors:** Collision Avoidance, Wall-Following, Homing, Tracking, Navigation
- **Position Estimation:** Kinematics, Beacon Triangulation, Grid-Based Estimation & Odometry Correction
- **Mapping:** Raw Data, Sensor Models, Converting to Vector Maps
- **Path Planning:** Shortest Collision-Free Travel,
- **Localization & Map Coverage** Algorithms

The course will also give you **paired-programming** experience so that you learn what it is like to work with a partner to solve your programming problems.

Attendance Requirements

All class lectures will be “live” on zoom. The lectures will usually take less than a half hour and will be followed by a lab assignment that you must complete at that time. You **MUST** attend a class if you want to hand in the lab assignment that day. Each lab assignment must be worked on and completed in your zoom breakout room before you leave if you want full marks. You are allowed to miss two labs with no penalty ... but you will need to reserve these “missed labs” for days that you are sick or unable to attend due to some event. **You CANNOT hand in lab assignments for grading unless you attended (and stayed for) the lab that day.**

The instructor and some of the TAs will be available on zoom to help you during the lab session. You need to select the “Ask for Help” icon when in your breakout room to call for a TA to help.

You will be working in a breakout room with a different partner (randomly chosen by zoom) for each lab assignment (although, one student will work alone when the number of students that day is odd-numbered).

When on zoom, you do not need to have your camera on if you prefer not to. However, **your login name MUST match the name that you are registered for at Carleton** (not your nickname ... your official name). That will allow TAs to verify who you are working with each time. Only one of you will submit your working version for grading and both students will share the same grade. Make sure to include BOTH names in the submitted source files. You should have a working microphone. It will slow your work down if you cannot communicate easily with your zoom partner and the TAs. The TAs will have limited time in your breakout room when you need help, so time should not be wasted by being forced to use the chat system.

Evaluation (see footnotes * and ** below)

Component	Weight	Details
22 Labs	70%	20 at 3.5% each ... done during class (2 labs discarded (e.g., illness))
Team Project	15%	(on Dec 3 & Dec 5)
Final Exam	15%	(time and date yet to be determined)

* A [full list of important dates](https://calendar.carleton.ca/academicyear) is available on the Calendar website. Please note that the **academic withdrawal dates have changed recently**. Consult the Calendar website for the most updated information: <https://calendar.carleton.ca/academicyear>

** The compassionate grading policy from the previous two years (SAT/UNS) is no longer in effect.

Course Material Copyrighted

All materials created for this course (i.e., course notes, coding examples, lectures, labs assignments, assignment code bases, marking schemes and exams) remain the intellectual property of the instructor. They are intended for the personal and non-transferable use of students registered in the course. Reproducing, reposting, and/or redistributing any course materials, in part or in whole, without the written consent of the instructor, is a copyright violation and is strictly prohibited.

I will re-use some of the content from these lab assignments for future offerings of this course. So, **you MUST NOT share anything with anyone**. This is meant to be a fun course where students get to learn to program robots. If lab solutions are shared (illegally as mentioned in the above copyright), then you are ruining the course experience for other students and you are opening up the way for plagiarism. You **MAY NOT** post your code solutions on GitHub or any other on-line resource. You may **NOT** post any of the course material online.

Laboratory Software

The course will use the **Webots** open source robot simulator, which is quite impressive visually. Check out the <http://www.cyberbotics.com> website to make sure that your laptop or desktop can support the 3D simulator. You should try to install it and test out the code right away (follow steps below). In order to develop and run your JAVA code, it is also necessary to have the 64-bit version of the Java Development Kit (JDK) version 1.8 or later (which can be downloaded from the [Oracle Technology Network](#)). **Unfortunately, if you cannot run this simulator, you will need to withdraw from the course, as all labs are done within the simulator.** Alternatively, you “may” be able to have access at the University on the Gaming Lab computers (depending on availability) ... contact the instructor.

1. Follow the **Installation Instructions** mentioned on the next page.
2. Download the **Community Edition** (i.e., NOT the **Ultimate Edition**) of the IntelliJ IDE from <https://www.jetbrains.com/idea/download/other.html>

If you are having trouble with the installation of Webots, try the instructions in the User Guide on the cyberbotics homepage under the Documentation menu. There is a subsection on the left called “Programming Language Setup”. Unfortunately, you will need to do this on your own since we cannot meet together to debug. Do this before class starts. And “No” ... you CANNOT use any other programming language for this course.

Cheating and Plagiarism (University Policies)

Sadly, every term, students are caught cheating on assignments. Copying on an assignment or a test is considered plagiarism. Since you will be working with a partner on each lab, there will be one shared assignment solution between you and that student. However, your code MUST NOT be copied/shared/obtained from any other students in the class. Penalties for such offences can be found on the ODS webpage: <https://science.carleton.ca/academic-integrity/>. You may NOT use ChatGPT, Google Bard, Bing Chat, nor any such generative AI in this course for completing the labs or exams. Please note that content generated by an unauthorized A.I.-based tool **is** considered plagiarized material.

So, in your breakout room, you and your lab partner are allowed to openly share your code, share your screen and even write your solutions together. Only one of you will submit the lab assignment which will contain both of your names. Keep in mind that you will have a different lab partner for each assignment.

Undergraduate Academic Advisor

The Undergraduate Advisor for the School of Computer Science is available in **Room 5302C HP**; or by email at scs.ug.advisor@cunet.carleton.ca. The undergraduate advisors can assist with information about prerequisites and preclusions, course substitutions/equivalencies, understanding your academic audit and the remaining requirements for graduation. The undergraduate advisors will also refer students to appropriate resources such as the Science Student Success Centre, Learning Support Services and Writing Tutorial Services.

Requests for Academic Accommodation

Carleton is committed to providing academic accessibility for all individuals. You may need special arrangements to meet your academic obligations during the term. The accommodation request processes, including information about the Academic Consideration Policy for Students in Medical and Other Extenuating Circumstances, are outlined on the Academic Accommodations website (students.carleton.ca/course-outline). As mentioned earlier, you may miss two labs without penalty. If you require accommodations for this course that are for a long-term period, please email me (as soon as the problem arises) to discuss how/whether accommodation needs could be met for this course.

You may need special arrangements to meet your academic obligations during the term. Please go here to read about requests for special accommodations: <https://students.carleton.ca/course-outline>. Here are some of the reasons you may request special accommodations ... most of these have specific deadline for requesting accommodation. It is the student’s responsibility to make the need known to the instructor within the stated deadlines: **Religious obligation, Pregnancy obligation, Students with Disabilities, Survivors of Sexual Violence & Student Activities**

Installation Instructions

Here are the instructions for getting Webots up and running. **Section 1** is for a **Windows** installation and **Section 2** is for a **Mac OS** installation. I will not explain any other operating system installations, but you can check more out on the cyberbotics website. **Section 3** is to test to make sure that all works.

(1) Windows Installation

Go to cyberbotics.com and download the **Webots Open Source Robot Simulator**. From the pulldown menu, choose the **Windows Installer** as shown at the top of the menu. At the time of this writing, the latest windows version was **webots-R2025a_setup.exe**. Just follow the installation instructions.



Download the latest 64-bit version of JAVA (It MUST be the 64-bit version!). Go to <https://www.oracle.com/java/technologies/javase-downloads.html> and select your operating system ... then on the JDK Download link (e.g., **x64 Installer** for windows, **x64 DMG Installer** for MacOS, or **Arm 64 DMG Installer** for Mac M1).

At the time of this writing, the latest Windows version was **Java SE Development Kit 22**. The **Windows x64 Installer** file was named **jdk-22_windows-x64_bin.exe**. Go ahead and install it in the default directory. Once installed, there will still be a few steps that you must follow to get webots ready to use it.

- In the start menu, type **cmd** and open the command prompt window. Then type in **java -version**. It should show the latest “**22**” version and indicate that it is a **64-bit Server VM** as shown here. Typing **javac -version** should also show **javac 22**. If this is not correct, then you need to set the PATH variable (check here for more details: <https://cyberbotics.com/doc/guide/using-java#windows>).

```
Command Prompt
C:\Users\lanth>java -version
java version "22.0.1" 2024-04-16
Java(TM) SE Runtime Environment (build 22.0.1+8-16)
Java HotSpot(TM) 64-Bit Server VM (build 22.0.1+8-16, mixed mode, sharing)

C:\Users\lanth>javac -version
javac 22.0.1

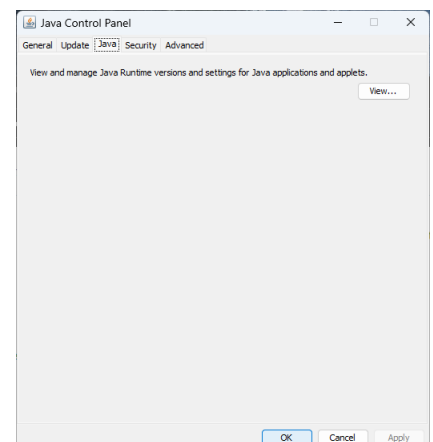
C:\Users\lanth>
```

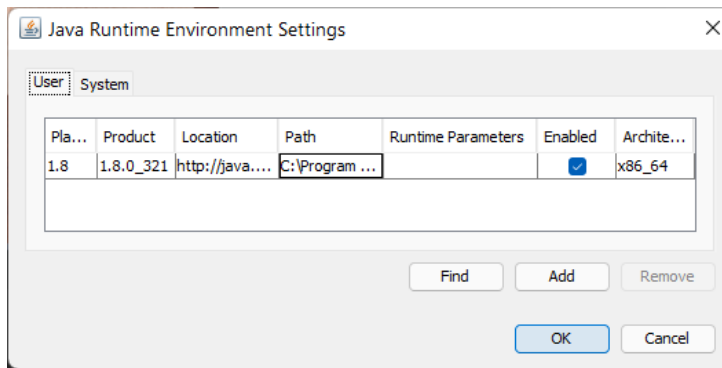
- At this point, you can try to see if you can compile and run the code in **Section 3** (below). If not, then come back here and continue. If you get an error that says something like this:

```
Native code library failed to load. See the chapter
on Dynamic Linking Problems in the SWIG Java
documentation for help.
java.lang.UnsatisfiedLinkError:
libJavaController.jnilib: no suitable image found.
```

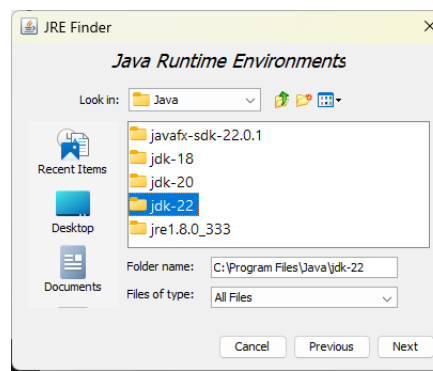
Then you will have to add the latest JAVA installation to your system in the control panel. To do this, type **control** in the start menu and start the **Control Panel** application. Then select **Java**. You should see something like this →

Click the **Java** tab. Then click the **View...** button. You should see something like this ... but it will vary according to what you had already installed:





Likely, your latest java version is not appearing. Click the **Find** button to load up the **JRE Finder**. Click **Next** and then navigate to the folder that contains your Java installation. Mine was under the **This PC** option on the left. Then select **OS (C:)** then **Program Files** then **Java ...** and you should see something like this with your latest java installation folder. Select the **jdk-22** folder and press **Next**.



Click **Finish** to add it and it should then appear in the view of installed java versions. Select it, then click **OK**. Then click **Apply** and **OK** from the previous dialog box. Hopefully, everything in **Section 3** will work now. If not ... Maybe something was from with the PATH files. Refer to the webots documentation again.

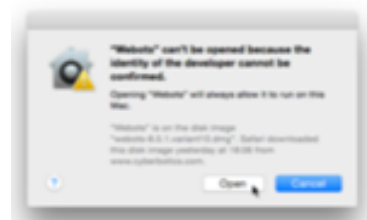
(2) Mac OS Installation

I have not verified this MacOS installation, so it could be off a little. Go to cyberbotics.com and download the **Webots Open Source Robot Simulator**. From the pulldown menu, you can choose the **macOS Bundle** as the second option in the menu. This will download a **webots.dmg** file.



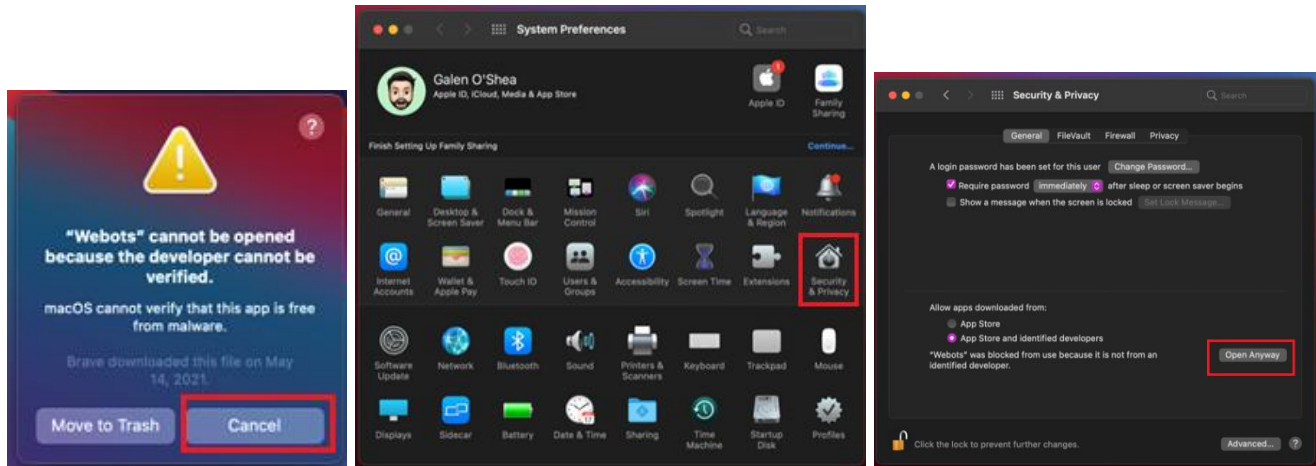
Drag the **webots.dmg** file to your **Applications** folder and then open the **Applications** folder.

Alternatively... open the **webots.dmg** file by double clicking it (similar to ISO files, opening it will mount and display a directory on Desktop which contains an APP file) and drag the **Webots.app** file into the Application folder.



Do a **Ctrl-click** on the **Webots** application and select **Open** from the popup menu. You may receive a popup warning saying **"Webots" can't be opened because the identity of the developer cannot be confirmed**". You can press **Open**.

You may receive an additional popup window that looks as shown below on the left. In that case, you can press **cancel** and go to **system preferences** (from the top left **Apple** menu on your computer). From there, go to the **Security & Privacy** icon (see below middle). Go to the **General** tab in the dialog box that appears and select **Open Anyway** for the “Webots” application (see right below).



Then select **Open** from the final warning box, even though it says “macOS cannot verify the developer of “Webots”. Are you sure you want to open it?”.

When finally getting the application to open, you may receive a warning saying that your computer/laptop does not have enough VRAM. Do not worry about this, I think all will be ok.

You will need to ensure that you have a proper JAVA version installed. You can do this by opening up the **Terminal** application (available in the **Applications** folder). Just type-in the command `java -version` and make sure that it is at least version 8 (it likely will be, since we are at version 16 at this time). Then define the environment variable by typing this into the terminal window as well:



```
export CLASSPATH=/usr/lib/jvm/java/bin:relative/mylib.jar
```

(3) TEST COMPILING AND RUNNING

Do the following to test compiling and running.

- From the **File** menu, select **Open Sample World...**
- Expand the **languages** option then the **java** option and double click on **example.wbt**
- You should see the blue robots moving around a round checkered environment (press the **Play**

button  if it is currently on pause ).

- Pause the simulation by pressing the pause button. 
- In the **Driver.java** code ... scroll to the bottom and enter the highlighted line below as the first line of the **main()** function:

```

public static void main(String[] args) {
    System.out.println("Hello World");
    Driver controller = new Driver();
    controller.run();
}

```



- Press the compile button:
- It will ask you if you want to copy the files to another location, since it doesn't want you modifying the installation directory. Click **OK**. Then click **Copy** and then **Close** in the next window that pops up.
- You may get a warning as follows ... but you can ignore:


```

"Driver.java:39: warning: [this-escape] possible 'this' escape before subclass is fully initialized
emitter = getEmitter("emitter");
            ^
1 warning

```
- Select **Reload** once the compilation is successful (ignore the warning).
- You may have to start the simulation with the play button  if it had stopped.
- You should see **Hello World** printed at least once in the bottom console window before the **Commands:** menu.
- If this all works ... then you should be able to compile and run the code for all the labs.