Webots with ROS2 Installation and Setup Documentation

V1.0

By Noah Grzywacz

May 19, 2021

Table of Contents:

1. Background Information, Usage, and Purpose:

Webots:

Webots is a platform program created by Cyberbotics that allows the creation, development, and test of various robotic systems. This open-source project is free to download, and easy to set up thanks to the extensive documentation Cyberbotics provides. Webots provides an easy-to-use interface for developing robot controllers and testing these controllers through simulation as well as providing lots of other additional tools and information for the purposes of debugging and documenting any project. For the purposes of the project outlined in this documentation, we will **not** be directly downloading the Webots program from cyberbotics.com because we need to set up Webots to work directly with ROS2. However, if you wish to explore all that Webots has to offer feel free to download and install the program using the link above and follow the documentation provided by Cyberbotics to start creating your own projects within Webots.

ROS2:

ROS2 is the second installment of the ROS toolkit. ROS, or Robot Operating System, is a set of packages and libraries that aid in the development of complex and high-end robotics systems. The packages included with ROS contain modules and logic for almost all the basic functions of a robot or robot system to allow users to jump into more sophisticated robotics projects without needing to spend time setting up the logic for the intrinsic elements of the robot. These elements that ROS provides the basic logic for are things like robot movement, odometry, infrared sensors, camera devices, and a multitude of other things. We will be using many of these libraries and modules directly or in auxiliary throughout the creation of the project outlined in this documentation.

Now that we know what both Webots and ROS2 are, we can describe the purpose of this documentation and how to use it. This documentation is one part of a series of documents that outlines how to create robotics projects with ROS2 and Webots. This document specifically, deals with the installation and building of the necessary software to get started with these robotics projects. Subsequent documents such as the “Webots with ROS2 Custom Controller Documentation” provide more information and guidance on how to implement your own robotics system using ROS2 and Webots platforms. Subsequent supplemental documents such as the “ROS2 Custom Messages” provide more information on optional features that the user can use to develop more complex robotics systems using these platforms.

Overall, this documentation and the other associated documents are meant to provide the reader with, at the very least, a basic understanding of how robotics projects can be developed using ROS2 and Webots, as well as provide beginner to complex concepts to acquaint themselves with ROS’s powerful platform and how it can be paired with Webots robotics simulator so that they can go on to develop their own robotics projects.

With all that said, let’s get started!

2. Downloading, Installing, and Building ROS2 with Webots:

This section is dedicated to installing and building ROS2 and Webots to work together. There are multiple methods that ROS provides documentation for to install ROS2 on Linux operating systems, but the method that will be directly employed in this documentation section can be found here. We will be directly building ROS2 from the ROS2 repository along with its dependencies and other useful tools.

This section also derives some of its content from the first video of Soft-Illusion's Webots ROS2 tutorial series. The Webots installation outlined in this documentation follows this video procedure, but while the video follows the standard install through downloading the ROS2 .tar Debian package and uncompressing the files it contains, we'll be using the method of building the ROS2 repos and all its dependencies. This is because, while doing this project originally, the standard installation method did not work, but the build method did.

2.1 Installing and Building ROS2:

ROS has made the process of installing and building ROS2 easy by providing easy-to-follow documentation where you essentially just copy and paste bash commands. Throughout the following procedure, if you encounter any errors, make sure to check out the official documentation provided by the link in the first paragraph above.

Procedure:

1.) First, you need to make sure that you have a locale that supports UTF-8 by running the following commands:

```
locale  # check for UTF-8

sudo apt update && sudo apt install locales
sudo locale-gen en_US en_US.UTF-8
sudo update-locale LC_ALL=en_US.UTF-8 LANG=en_US.UTF-8
export LANG=en_US.UTF-8

locale  # verify settings
```

2.) Next, you will need to add the ROS2 apt repository to your system using the following commands:

```
sudo apt update && sudo apt install curl gnupg2 lsb-release
curl -s https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc | \
       sudo apt-key add -

sudo sh -c 'echo "deb [arch=$(dpkg --print-architecture)] \
http://packages.ros.org/ros2/ubuntu $(lsb_release -cs) main" > \
/etc/apt/sources.list.d/ros2-latest.list'
```

3.) Now we will need to install install development tools and ROS tools using the following commands:

```
sudo apt update && sudo apt install -y \
      build-essential \
      cmake \
      git \
      libbullet-dev \
      python3-colcon-common-extensions \
      python3-flake8 \
      python3-pip \
      python3-pytest-cov \
      python3-rosdep \
      python3-setuptools \
      python3-vcstool \
      wget


# install some pip packages needed for testing
python3 -m pip install -U \
      argcomplete \
      flake8-blind-except \
      flake8-builtins \
      flake8-class-newline \
      flake8-comprehensions \
      flake8-deprecated \
      flake8-docstrings \
      flake8-import-order \
      flake8-quotes \
      pytest-repeat \
      pytest-rerunfailures \
      pytest

# install Fast-RTPS dependencies
sudo apt install --no-install-recommends -y \
      libasio-dev \
      libtinyxml2-dev

# install Cyclone DDS dependencies
sudo apt install --no-install-recommends -y \
      libcunit1-dev
```

4.) Next, you'll need to get the ROS2 code to build. Start by making a workspace and cloning all the repositories using the following commands:

```
mkdir -p ~/ros2_foxy/src
cd ~/ros2_foxy
wget https://raw.githubusercontent.com/ros2/ros2/foxy/ros2.repos
vcs import src < ros2.repos
```

5.) You also need to install the ROS2 dependencies using rosdep by using the commands:

```
sudo rosdep init
rosdep update
rosdep install --from-paths src --ignore-src --rosdistro foxy -y \
--skip-keys "console_bridge fastcdr fastrtps rti-connext-dds-5.3.1 \
urdfdom_headers"
```

6.) Now, you just need to build the code in the workspace by moving to the workspace directly and building it with a specific command. These two actions are performed by the commands:

```
cd ~/ros2_foxy/
colcon build --symlink-install
```

This will take some time to complete. For the project outlined in this documentation, this build process took about 4 hours to complete.

7.) The last thing you need to do is source the setup script to set up your environment by using the command:

```
. ~/ros2_foxy/install/setup.bash
```

It is useful to place this command in your local bash script so that it runs every time you open a new bash window. We will cover how to do this in the next section when installing Webots because there are additional commands that we need to add to the bash script.

8.) If you wish to test that your ROS2 package has been installed correctly, you can test it using a basic talker/listener program. This program follows the client/server model where the talker simply provides data to the listener which outputs the data. To run this program source the setup file and run the talker program in one terminal using the commands:

```
. ~/ros2_foxy/install/local_setup.bash
ros2 run demo_nodes_cpp talker
```

In another terminal window, source the same setup file and run the listener program using the commands:

```
. ~/ros2_foxy/install/local_setup.bash
ros2 run demo_nodes_py listener
```

You should see the talker publishing some information like the following:

```
[INFO] [1601514167.378817538] [talker]: Publishing: 'Hello World: 1'
[INFO] [1601514168.378839138] [talker]: Publishing: 'Hello World: 2'
[INFO] [1601514169.378881175] [talker]: Publishing: 'Hello World: 3'
[INFO] [1601514170.378740550] [talker]: Publishing: 'Hello World: 4'
[INFO] [1601514171.378843104] [talker]: Publishing: 'Hello World: 5'
[INFO] [1601514172.378830576] [talker]: Publishing: 'Hello World: 6'
[INFO] [1601514173.378906496] [talker]: Publishing: 'Hello World: 7'
[INFO] [1601514174.378857683] [talker]: Publishing: 'Hello World: 8'
[INFO] [1601514175.378928169] [talker]: Publishing: 'Hello World: 9'
[INFO] [1601514176.378941249] [talker]: Publishing: 'Hello World: 10'
[INFO] [1601514177.378942062] [talker]: Publishing: 'Hello World: 11'
[INFO] [1601514178.378853192] [talker]: Publishing: 'Hello World: 12'
[INFO] [1601514179.378947397] [talker]: Publishing: 'Hello World: 13'
[INFO] [1601514180.378798009] [talker]: Publishing: 'Hello World: 14'
[INFO] [1601514181.378765866] [talker]: Publishing: 'Hello World: 15'
[INFO] [1601514182.378880819] [talker]: Publishing: 'Hello World: 16'
[INFO] [1601514183.378953444] [talker]: Publishing: 'Hello World: 17'
[INFO] [1601514184.378955802] [talker]: Publishing: 'Hello World: 18'
[INFO] [1601514185.378896901] [talker]: Publishing: 'Hello World: 19'
[INFO] [1601514186.378953552] [talker]: Publishing: 'Hello World: 20'
[INFO] [1601514187.378885663] [talker]: Publishing: 'Hello World: 21'
```

And the listener publishing the message that the talker publishes to it like so:

```
[INFO] [1601514182.403821863] [listener]: I heard: [Hello World: 16]
[INFO] [1601514183.381654433] [listener]: I heard: [Hello World: 17]
[INFO] [1601514184.381445840] [listener]: I heard: [Hello World: 18]
[INFO] [1601514185.380091156] [listener]: I heard: [Hello World: 19]
[INFO] [1601514186.381901437] [listener]: I heard: [Hello World: 20]
[INFO] [1601514187.380469353] [listener]: I heard: [Hello World: 21]
[INFO] [1601514188.383152106] [listener]: I heard: [Hello World: 22]
[INFO] [1601514189.380726653] [listener]: I heard: [Hello World: 23]
[INFO] [1601514190.381730807] [listener]: I heard: [Hello World: 24]
[INFO] [1601514191.382057014] [listener]: I heard: [Hello World: 25]
```

If you have any errors running these programs, first make sure that you sourced the setup scripts correctly. Often, it's easy to forget to source the setup script but it must be done every time you rebuild your package/workspace. We elaborate more on this later in this documentation. If you correctly sourced the local setup and it still doesn't work, try sourcing the system ROS setup script using the command:

```
source /opt/ros/foxy/setup.bash
```

The ros directory created in the system `/opt` folder is a self-contained set of dependencies for your ROS distribution. If this directory doesn't exist in your system and you continue to have errors, it is recommended to refer to the ROS documentation.

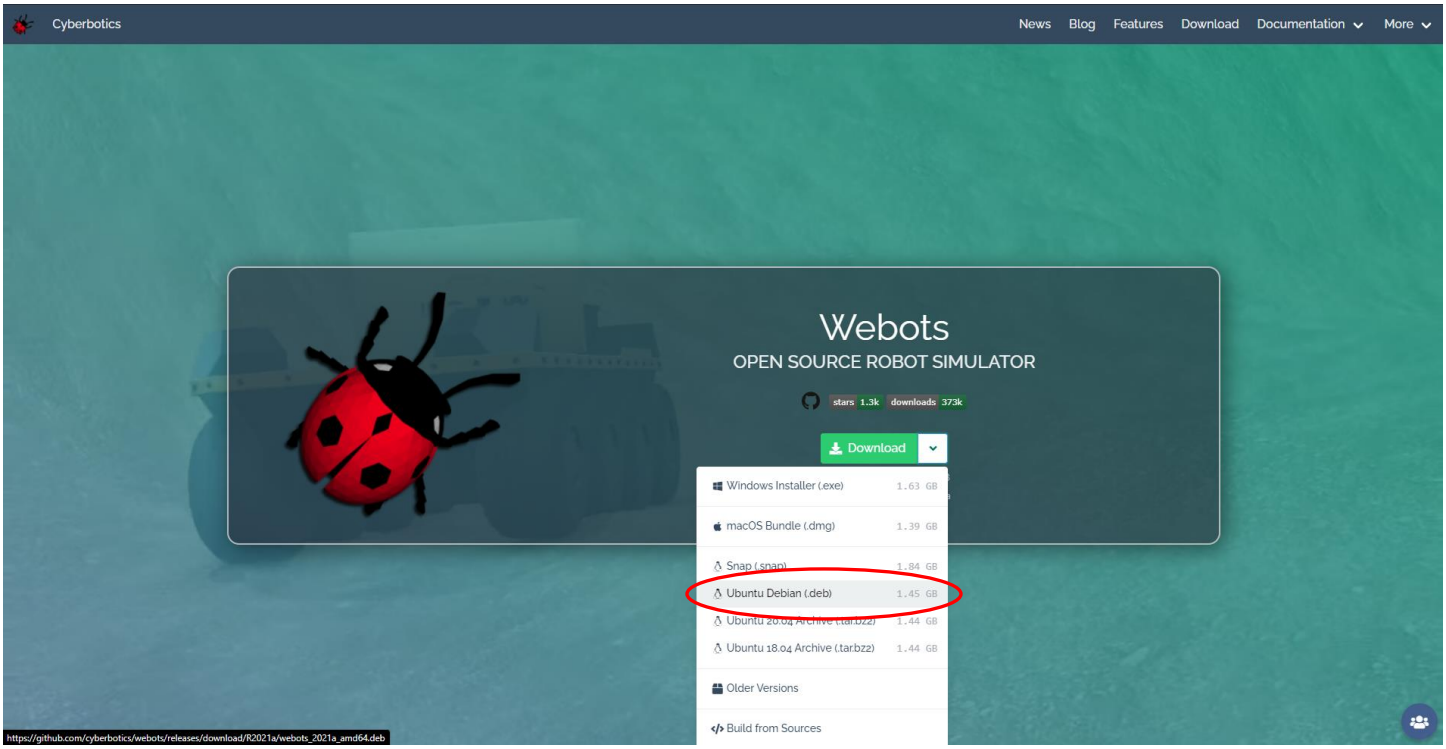
At this point, you should have the base installation of ROS2 installed and built on your Ubuntu virtual machine. As always, if you encountered any errors while performing this procedure, be sure to go back and make sure you completed each of the previous steps thoroughly and using the correct commands. If you still have issues, you should check out the ROS2 documentation that this installation is based off by using the link at the top. Again, if errors continue to persist, Google can be your best friend in resolving these.

2.2 Installing Webots and Webots ROS2 Packages:

While you already have the base packages for ROS2, to use Webots, you need to have both Webots and the ROS2 packages that work with it. This section is dedicated to guiding you through both installations. As mentioned at the beginning of the chapter, these installations are based on the first video in Soft-Illusion's Webots ROS2 tutorial series, so you need additional reference material, be sure to check that video out using the link provided.

Procedure:

1.) We will start by installing Webots. Go to https://cyberbotics.com/ and next to the "Download" button in the center of the screen, click the down arrow to show all the Webots downloadable applications and select the "Ubuntu Debian (.deb)" Debian package to download it.



Once the package is installed, simply open it by clicking it and follow the instructions to install Webots.

2.) Next, you need to install and build the ROS2 packages that work with Webots. If need additional reference resources, this section is ripped from github.com/cyberbotics/webots_ros2/wiki/Build-and-Install. You can start by creating a workspace directory in your `/home/<Your Username>` directory, so open the "Terminal" and enter:

```
cd /home/<Your Username>
```

Note: this should be default directory you enter when opening Terminal

Then create a new Webots workspace where you will be working on your projects and move into it:

```
mkdir -p ros2_ws/src

cd ~/ros2_ws
```

3.) Before you clone the repository for webots_ros2, you need to run the setup script for your ROS2 distro by running:

```
source /opt/ros/$ROS_DISTRO/local_setup.bash
```

4.) You now need to clone the webots_ros2 repository from github, simply enter the following command:

```
git clone –recurse-submodules -b $ROS_DISTRO \
https://github.com/cyberbotics/webots_ros2.git src/webots_ros2
```

5.) To check for dependencies and make sure we have all the necessary ones, run the following commands:

```
rosdep update
rosdep install --from-paths src --ignore-src --rosdistro $ROS_DISTRO
```

Note: This is a useful command to run whenever you run into an error dealing with dependencies. That is, if you get an error saying that some software package cannot be found or doesn't exist in your system, you can run this command to try and resolve the error.

6.) Now that you have all the necessary dependencies, you can build your ros2_ws workspace, using the following command in the `~/ros2_ws/` directory:

```
colcon build
```

7.) Lastly, you need to source the local setup of the workspace you just created and built by running:

```
source install/local_setup.bash
```

You now have both ROS2 and Webots set up on your Ubuntu virtual machine, however next you will need to install a text editor to work on your projects in. In this documentation, you will be instructed to download VS Code, but if you already have another editor you are comfortable in, that should work just as well.

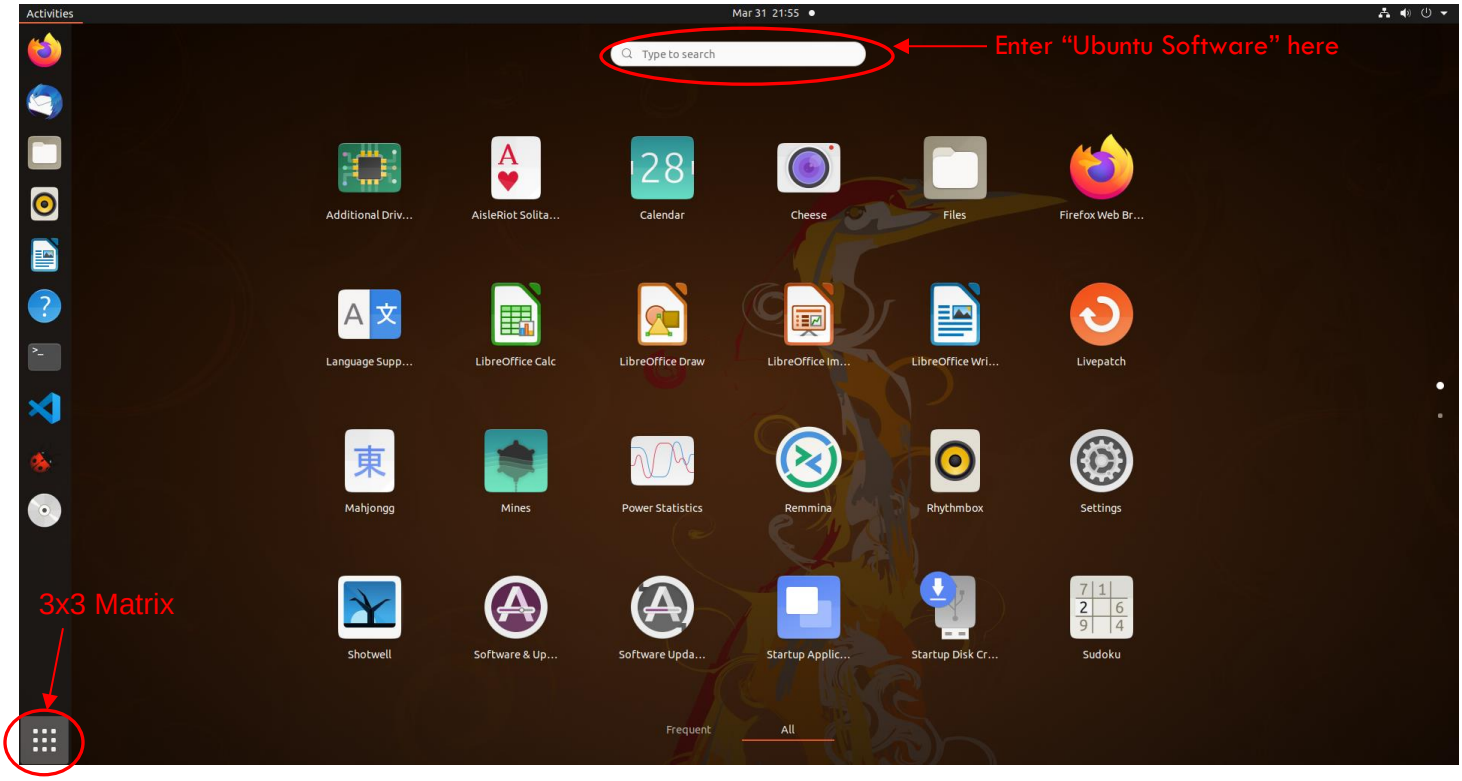
2.3 Download and Installing VS Code:

This quick section is dedicated to showing you how to download and install VS Code, the text editor that you will be working on your projects in. VS Code is a great text editor and one of the leading editors present in the computing industry.

The easiest way to download VS Code is through Ubuntu's "Ubuntu Software" application. Simply follow the quick procedure below to install VS Code.

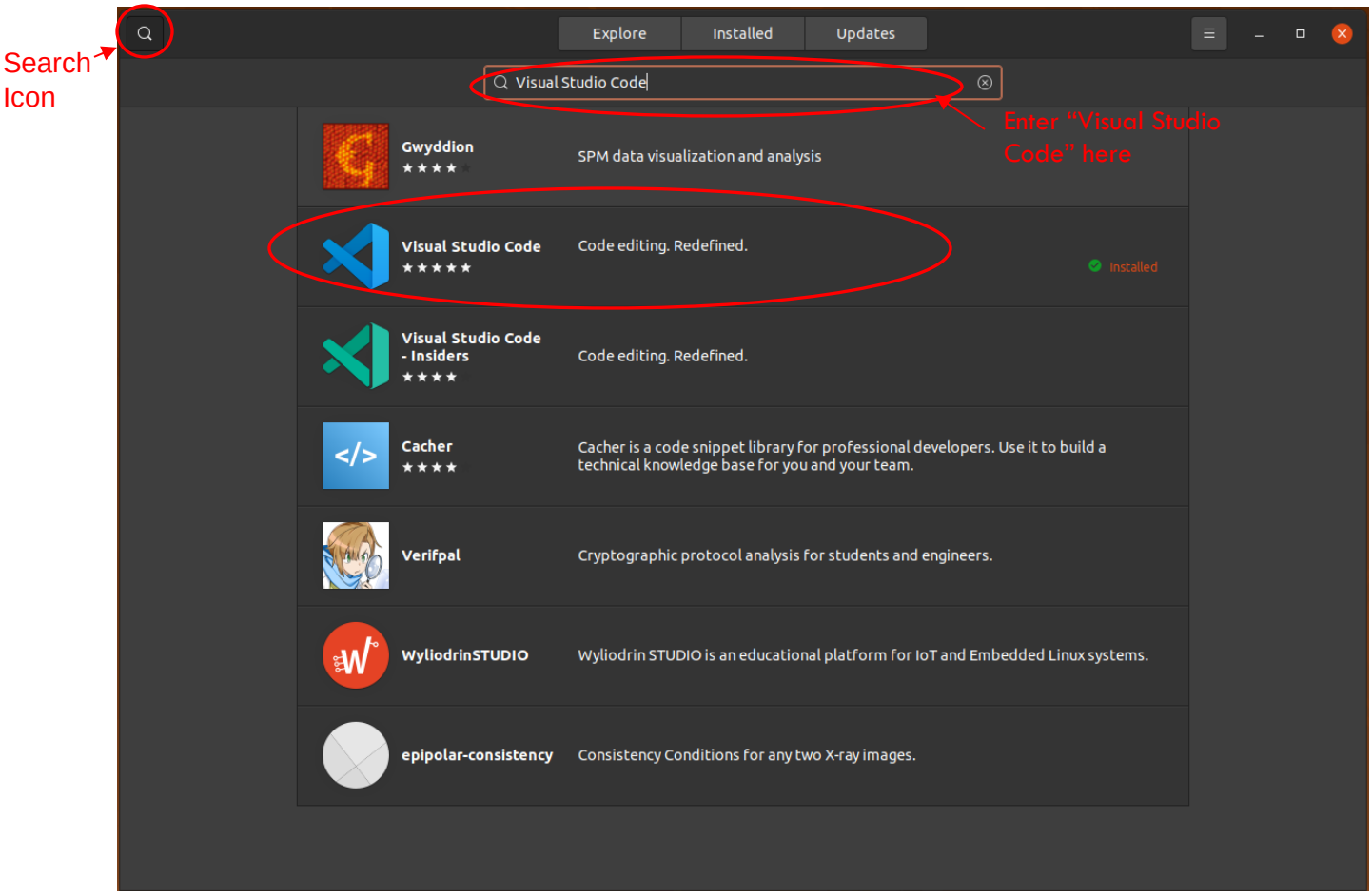
Procedure:

1.) In the Ubuntu desktop click the 3x3 matrix of dots at the bottom left to be able to see and search for all of your installed applications. In the search bar, type "Ubuntu Software" and select the Ubuntu Software icon.



The Ubuntu Software icon looks like this:

2.) Once you have opened that, select the search icon at the top left and enter “Visual Studio Code.” The application you will want to install should be the second option with the blue icon:



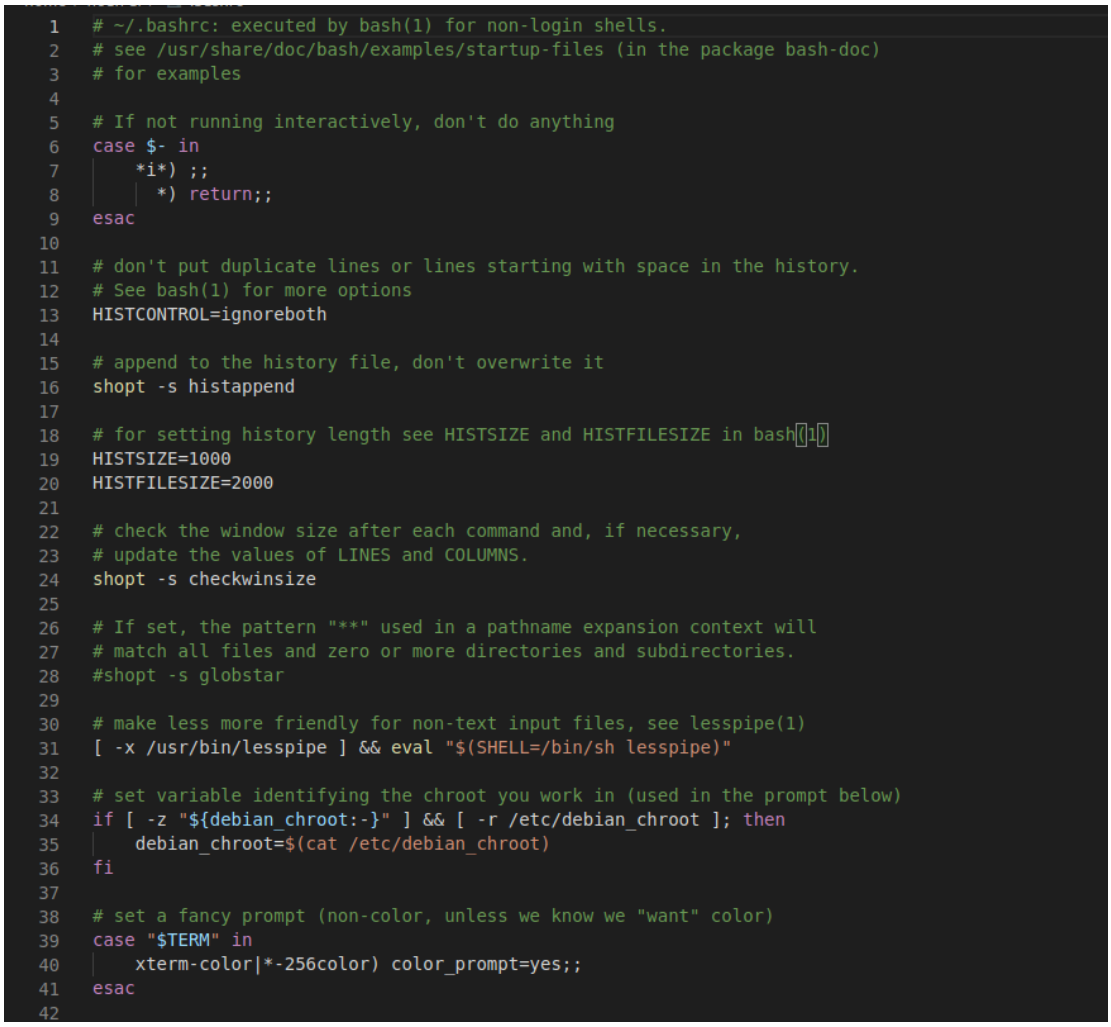
Note: Yours will not say it is already installed.

3.) Once you have selected the second option, click the green “Install” button and you’re done! However, before moving on to the next section, it would be helpful to add some commands and values to your native bash script. Once Visual Studio Code is finished installing, open up a new Terminal window. Tip: You can use the shortcut CTRL + ALT + T to open a new Terminal window. Once your window is open, enter the following command:

```
code ~/.bashrc
```

This simple command will open your native .bashrc script which is run every time you open a new shell window or tab. For this reason, it’s helpful to place some of the ROS2 and Webots sources and values so that you don’t have to run the same commands every time you open a new shell. They’ll automatically be run for you. The `code` part is a cmdlet that tells Ubuntu to open this bash script in Visual Studio Code.

4.) When you open this script with VS Code, it should look something like the following:

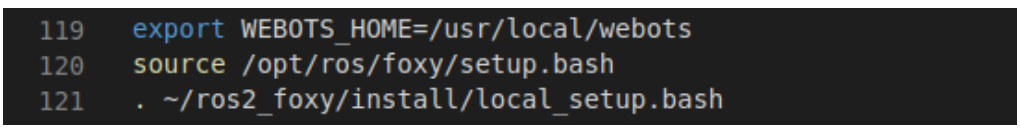


```
# ~/.bashrc: executed by bash(1) for non-login shells.
# see /usr/share/doc/bash/examples/startup-files (in the package bash-doc)
# for examples

# If not running interactively, don't do anything
case $- in
    *i*) ;;
      *) return;;
esac

# don't put duplicate lines or lines starting with space in the history.
# See bash(1) for more options
HISTCONTROL=ignoreboth

# append to the history file, don't overwrite it
shopt -s histappend

# for setting history length see HISTSIZE and HISTFILESIZE in bash(1)
HISTSIZE=1000
HISTFILESIZE=2000

# check the window size after each command and, if necessary,
# update the values of LINES and COLUMNS.
shopt -s checkwinsize

# If set, the pattern "**" used in a pathname expansion context will
# match all files and zero or more directories and subdirectories.
#shopt -s globstar

# make less more friendly for non-text input files, see lesspipe(1)
[ -x /usr/bin/lesspipe ] && eval "$(SHELL=/bin/sh lesspipe)"

# set variable identifying the chroot you work in (used in the prompt below)
if [ -z "${debian_chroot:-}" ] && [ -r /etc/debian_chroot ]; then
    debian_chroot=$(cat /etc/debian_chroot)
fi

# set a fancy prompt (non-color, unless we know we "want" color)
case "$TERM" in
    xterm-color|*-256color) color_prompt=yes;;
esac

```

Scroll down to the bottom of this file and you'll want to add these three lines to the bash script:

```
export WEBOTS_HOME=/usr/local/webots
source /opt/ros/foxy/setup.bash
. ~/ros2_foxy/install/local_setup.bash
```

These three lines assign provide some useful information to Ubuntu for building and running your ROS2 projects with Webots. The first line assigns the environment variable WEBOTS_HOME to the directory where webots was installed so that ROS2 knows where to look when trying to simulate your project packages. If this directory is not under `/usr/local/` it may be in your `/opt/` directory. The second line you have seen before earlier in this documentation. This line simply sources the setup script in your self-contained installation of your ROS distribution. The third line does something similar and sources the local setup script for the ROS2 directory you built. Your bash script should now have these lines in it and look like so:



```
119  export WEBOTS_HOME=/usr/local/webots
120  source /opt/ros/foxy/setup.bash
121  . ~/ros2_foxy/install/local_setup.bash
```

Once you have your bash script looking like this, you can save and close the script and VS Code.

As always, if you have any issues with this procedure or any of the previous ones in this chapter, first make sure that you followed the procedure correctly. If you still have errors, consult the ROS documentation and other references provided. While this documentation attempts to account for some errors that were faced when originally going through these steps, it does not include solutions to every issue. But with a little digging and researching, you should be able to find solutions to any of the errors you encounter. In the next chapter you'll be learning about ROS2's packages and how to create your own for your projects.

3. Final Notes:

You should now have ROS2 fully installed and set up to work with Webots. As mentioned before, if you ran into errors when installing and building any of the software included in this documentation, you should first check to make sure that you completed the previous steps correctly. Also feel free to check out the ROS2 documentation linked in the References section below for more information.

This is just the start of the journey though, because now it is time to do something with your newly installed software. If you do not already have an idea of how you want to use Webots with ROS2, feel free to check out the “Webots with ROS2 Custom Controller Documentation” to see how we can implement a custom controller to allow a robot to complete a specific task.

5. TODO:

- Fix up documentation by including internal references and links
- Continue to proofread and make more concise

6. References:

Soft Illusion Webots ROS2 Tutorial Videos:
https://youtube.com/playlist?list=PLt69C9MnPchkP0ZXZOqmIGRTOch8o9GiQ

ROS Documentation Homepage:
http://wiki.ros.org/Documentation

Building ROS2 Repository on Linux: (Non-standard Install)
https://docs.ros.org/en/foxy/Installation/Ubuntu-Development-Setup.html

Installing ROS2 Repository on Linux: (Standard Install)
https://docs.ros.org/en/foxy/Installation/Ubuntu-Install-Binary.html

Webots Install from ROS2 GitHub Repo:
https://github.com/cyberbotics/webots_ros2/wiki/Build-and-Install