Software setup

To match the examples in this book, and to have access to the same tools that are used in the code samples, you will have to set up three environments:

- A laptop or desktop computer: This will run our control panel, and also be used to train neural networks. I used a Windows 10 computer with Oracle VirtualBox supporting a virtual machine running Ubuntu 16.14. You may run a computer running Ubuntu or another Linux operating system by itself (without Windows) if you want. Several of the AI packages we will use in the tutorial sections of the bookwill require Ubuntu 16 or later to run. We will load ROS on this computer. I will alsobe using a PlayStation-type game controller on this computer for teleoperation (remote control) of the robot.
- Raspberry Pi 3: Also running Ubuntu Linux (you can also run other Linux versions, but you will have to make any adjustments between those OS versions yourself). The Pi 3 also runs ROS. We will cover the additional libraries we need in each section of the text.
- Arduino Mega 256: We need to be able to create code for the Arduino. I'm using the regular Arduino IDE from the Arduino website. It can be run in Windows or in Linux. Installation will be covered regardless.

Laptop preparation

I will just cover creating the Ubuntu Linux virtual machine under VirtualBox, since that is my setup. You can find instructions for installing Ubuntu 16.04 LTS without VirtualBox at https://tutorials.ubuntu.com/tutorial/tutorial-install-ubuntu-desktop#0:

1. Download and install VirtualBox from https://www.virtualbox.org/wiki/Downloads. Pick the version that matches your computer setup(Windows).
2. Download the Ubuntu system image from https://www.ubuntu.com/download/desktop. It will be an .iso file that is quite large (almost 2 GB). An .iso is a disk image file that is a byte-for-byte copy of another filesystem.
3. Open VirtualBox and select **New.**
4. Make up a descriptive name for this virtual machine and select **Linux** and **Ubuntu (64-bit)** in the **Type** and **Version** fields. Select **Next**.
5. Set a **Base Memory size**. I picked **3** GB.
6. Select a size for your virtual machine partition. I made a **40** GB virtual drive.
7. Click **Next.**
8. Select **Start** (green arrow) and pick your **Media Source** as the .iso we downloaded in Step 2.
9. Finish the installation by following the prompts.
10. Restart the virtual machine once you are finished.

Installing Python

The Linux Ubuntu system will come with a version of Python. I am going to assume that you are familiar with Python, as we will be using it throughout the book. If you

need help with Python, *Packt Publishing* has several fine books on the subject.

Once you log on to your Virtual Machine, check which version of Python you have by opening up a terminal window and typing python at the command prompt. You should see the Python version as follows:

```
Python 2.7.12 (default, Dec  4 2017, 14:50:18)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

You can see that I have version 2.7.12.

We are going to need several add-on libraries that augment Python and extend its capability. The first thing to check is to see whether you have pip installed. **Pip** is a **Python Installation Package** that helps to load other packages from the internet to extend Python. Check to see whether you have pip by typing the following:

```
pip
```

If you get No command 'pip' found, then you need to install pip. Enter the following:

```
Sudo apt-get install python-pip python-dev build-essential
Sudo pip install –upgrade pip
```

Now we can install the rest of the packages that we need. To begin with, we need the python math packages numpy, the scientific python libraries scipy, and the math plotting libraries matplotlib. I will direct you to install other packages as required in the relevant chapters.

Let’s install our other libraries. We need the numerical python library (numpy), the scientific python library (scipy), and matplotlib for making graphs:

```
>>sudo apt-get install python-numpy python-scipy python-matplotlib python-sympy
```

If you want to use the iPython (interactive Python) and Jupyter Notebook process to test your code (and document it at the same time), you can install those as well. I will not be using them in this book, but I do admit that they are quite useful.

```
>>sudo apt-get install ipython ipython-notebook
```

I'll cover the other Python libraries that we will use later (Open CV, Scikit-Learn, Keras, and so on) as we need them in the appropriate chapters.

Installing ROS on the laptop

We need a copy of ROS to talk to our robot and to execute development. You can find directions on how to do this at the ROS website repository at http://wiki.ros.org/kinetic/Installation/Ubuntu.

I'll give you the quick and dirty method here:

Get on your Linux laptop/desktop computer – virtual or otherwise – and get to a command prompt. We need to establish the source of the software. We do this as follows:

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main"
> /etc/apt/sources.list.d/ros-latest.list'
```

We need to set up our key to get access to the software with this command. Remember that you can cut and paste this from the ROS website as well:

```
sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net:80 --recv-key
421C365BD9FF1F717815A3895523BAEEB01FA116
```

At this point, it is recommended that you check that your Linux installation is up to date. We can do this by entering the following command:

```
sudo apt-get update
```

This is going to take a bit of time depending on your internet connection and computer.

Now, we are finally ready to install the main ROS distribution. Since we are on a desktop or laptop computer, we can get the whole thing, so that we have all of the tools and user interfaces. Note that we are using ROS Kinetic, the latest full release as at the time of publication of this book:

```
sudo apt-get install ros-kinetic-desktop-full
```

Expect that this step can take quite a while – it was about 30 minutes on my computer. Once it finally finishes, we can proceed to setting up the ROS on our computer. We need to set up the ROS dependency manager using these commands:

```
sudo rosdep init
rosdep update
```

rosdep keeps up with which other packages your program depends on, and will download them automatically if they are missing. This is a good feature to have, and we won't be needing it very often – just when we install a new package.

This next part is really important. ROS uses several environment variables to configure some important parts of the system, such as keeping up with which computer is in charge. The default configuration is set by doing the following:

```
echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc source
~/.bashrc
```

The first setup appends the /opt/ros/kinetic/setup.bash to our default .bashrc file so that those environment variables will get automatically created each time we start a new terminal window. The .bashrc script is executed when a new bash shell program is started.

We can check the default setup by looking at the environment variables. Let's get only the ones with the string ROS inside them by using a pipe (|) symbol and the grep command:

```
>>:~$ env | grep ROS
ROS_ROOT=/opt/ros/kinetic/share/ros
ROS_PACKAGE_PATH=/opt/ros/kinetic/shar
```

e ROS_MASTER_URI=http://localhost:11311

SESSION_MANAGER=local/BookROS:@/tmp/.ICE-unix/1847,unix/BookROS:/tmp/.ICE- unix/1847

ROS_DISTRO=kinetic
ROS_ETC_DIR=/opt/ros/kinetic/etc/ros

The most important variable in this set is the ROS_MASTER_URI. This points to the place where the ROSCORE program is running. Normally, this will be on the robot. Let's say the robot's IP address is 192.168.1.15. Then, we set the ROS_MASTER_URI to be http://192.168.1.15:11311. You can also set up an entry in the /etc/hosts file with the name of the robot and the IP address as follows:

>>:sudo nano /etc/hosts

192.168.1.15 tinma

Then you can use the hostname tinman in place of the IP address:

env ROS_MASTER_URI=http://tinman:11311.

Remember to change this in the. bashrc script as well (nano .bashrc).

Setup of Raspberry Pi 3

I ended up doing two different operating systems on the Raspberry Pi 3, depending on whether or not I wanted to use the Google Voice interface. In the chapter on Artificial Personality, we will be using the Google Voice Assistant as one option for providing the robot with the ability to listen and respond to commands, as well as adding some personality and additional conversation capability. The easiest way to perform this is to use the Raspbian operating system image that Google provides with the DIY Voice Kit. You can see https://aiyprojects.withgoogle.com/voice to look at the kit. This very inexpensive piece of hardware ($24.95) includes a very nice speaker, a pair of sensitive microphones, and even a cool LED light-up button. We will cover the use of this hardware add-on in the chapter on Artificial Personality. It is much easier to use the operating system image provided with that kit than try and transplant all that capability onto an already built Raspberry Pi operating system. So, if you want to do the voice part of the robot project, go to that chapter and we will cover the setup of the Raspberry Pi 3 imageand the Raspbian operating system there.

Otherwise, if you don't want to use the Google Voice Assistant, you can build up a Raspberry Pi 3 with another operating system image. I happen to prefer to run Ubuntu on my Pi 3 to match my virtual machine.

For this setup, we will use an image provided by Ubuntu. Go to the Ubuntu Wiki website concerning the Raspberry Pi 3 (https://wiki.ubuntu.com/ARM/RaspberryPi).

The basic step, which you can follow on the website, is to prepare an SD card with the operating system image on it. I used Win32DiskImager, but there are several programs available that will do the job. You need an SD card with at least 8 GB of space –and keep in mind you are erasing the SD card in doing this.

Download the disk image from (https://downloads.ubiquityrobotics.com/*)* and pick the version for the Raspberry Pi 3. It is 1.2 GB of data, so it may take a bit. This image already

has the ROS on it, so it will save a lot of time later.

Follow the directions with your SD card – the website advises using a Class 10 memory card of at least 8 GB, or preferably 16 GB. Put the SD card in your reader and start up your disk imager program. Double (and triple) check that you are picking the right drive letter – you are erasing the disk in that drive. Select the disk image you downloaded – I used the 16.04.2 version. Hit the **write** button and let the formatter create your Pi 3 disk image on the SD card.

You can follow the usual setup for setting your language and keyboard, as well as setting up the network. I like to use a static IP address for the robot, since we will be using this a lot. Use the same instructions from the preceding section on setting up ROS environment variables. Put your robot's name in the /etc/hosts file and set the ROS_MASTER_URI to the Pi 3's host name – tinman in my case.

The operating system comes with Python already installed, as before when we set up the laptop/desktop virtual machine, so we follow the same procedures as previously to load the python libraries NumPy, SciPy, and a new package, Pyserial. We need this for talking to the serial port:

```
>>: pip install pyserial
```

VNC

One tool that I have added to my Raspberry Pi 3 is Virtual Network Computing, or VNC. This utility, if you are not familiar with it, allows you to see and work with the Pi3 desktop as if you were connected to it using a keyboard, a mouse, and a montor. Sincethe Pi 3 is physically installed inside a robot that travels by itself, attaching a keyboard, mouse, and monitor is not often convenient (or possible). There are many different versions of VNC, which is a standard protocol used among many Unix-type – and non-Unix type - operating systems. The one I used is called RealVNC. You need two parts – the server and the client. The server side runs on the Pi 3 and basically copies all of the pixels appearing on the screen and sends them out the Ethernet port. The client catches all of this data and displays it to you on another computer. Let's install the VNC server on the Pi 3 using this command:

```
>>: sudo apt-get install realvnc-vnc-server
```

You can reference the RealVNC website at https://www.realvnc.com/en/raspberrypi/. This will cover configuration items and how to set up the software. The VNC software is generally included in most Pi 3 Raspbian releases.

Load the viewer on your Windows PC, Linux virtual machine, or do like I did, and load VNC on your Apple iPad. You will find the ability to log directly into the robot and use the desktop tools to be very helpful.

Setting up catkin workspaces

We will need a catkin workspace on your development machine –laptop or desktop – as well as on the Raspberry Pi. Follow the instructions at http://wiki.ros.org/catkin/Tutorials/create_a_workspace.

If you are already a user of ROS, then you know what a catkin workspace is, and how it is used to create packages that can be used and deployed as a unit. We are going to

keep all of our programs in a package we will call tinman. Let's go ahead and put this package together. It takes just a few steps.

Start in the home directory:

```
mkdir –p catkin_ws/src
cd catkin_ws/src
catkin_make

source devel/setup.bash
catkin_create_pkg tinman
catkin_make

cd src/tinman/src
mkdir script
mkdir launch
```

You'll be left with a directory structure that looks something like this: