Cloud-Powered Robotics with Raspberry Pi

Build, deploy, and manage intelligent robots effectively

Edgardo Peregrino

www.bpbonline.com

First published: 2024

Published by BPB Online
WeWork
119 Marylebone Road
London NW1 5PU

UK | UAE | INDIA | SINGAPORE

ISBN 978-93-55516-275

www.bpbonline.com

Dedicated to

My father who supported me

My dear mother

&

My dear Love

About the Author

Edgardo Peregrino is a freelance software developer, IT technician and the author of the book *Programming Raspberry Pi in 30 Days*. He has developed projects to help educate and inspire new programmers and aspiring developers. He has certifications from both CompTIA and Microsoft which include A+ and Microsoft Technology Associate. He is the author of the article *Using GUIs to Control Two Robots*, *Using GUIs to Control Two Robots- Update: Multi-Platform GUIs*, *Dual Robot Control Using Web Development + the Cloud* and *Cloud Native Computing + Robots = Magic* and the latest article *My Cloud Native Journey* which can be found on Servo Magazine. His work is also featured on the Grafana Blog under the blog post *Monitoring robots in real time with Grafana and other cloud native solutions*.

About the Reviewers

- **Dana Alsani** is a passionate robotics engineer with extensive professional experience in the domain of robotics software development. Her skill set includes Python, Javascript, Java, ROS framework, Linux OS, cloud computing, and electronics. She is a holder of multiple industry certifications such as the AWS Solutions Architect, ROS2 Humble, and more.

  She is currently working as part of a research and development team that is building a remotely controlled autonomous robot and has been using cloud technologies to enhance her robotics software development. In addition to that, she has taught many students of different ages and backgrounds various skills in computer programming, electronics, and robotics concepts.

  She believes that education is the key to empowering people and creating positive change in the world. She uses her expertise and enthusiasm to inspire and motivate her students to pursue their passions and goals in robotics and beyond.

- **George Okoroafor** is a dedicated robotics engineer with a strong background in mechanical engineering and a deep passion for robotics and artificial intelligence. He is known for his proficiency in various programming languages, including Python, C++, Java, HTML, PHP, and JavaScript. Additionally, he has expertise in using frameworks such as ROS 1 and ROS 2, OpenCV, and Qt5 for developing robotics applications.

  With experience in simulation and software tools like Gazebo, Rviz, and Moveit, George excels in creating and testing robotic systems before deploying them. His hardware skills encompass microcontrollers like Arduino and ESP32, as well as single-board computers like the Raspberry Pi and the Nvidia Jetson series. He is well-versed in sensor integration and has a strong foundation in computer-aided design and engineering, with proficiency in Solidworks. He works with Leon Legion as a motion planning engineer for the last-mile delivery robot.

Acknowledgement

I want to thank my family who supported me throughout this journey, including my mom, who has always been there for me. I thank my love for encouraging me in my journey and for her never ending love.

I also want to thank BPB publications for giving me the chance to write the book that I wanted to write for two years and their quick and responsive support.

I want to thank the Raspberry Pi Foundation for their hard work in bringing computer science education to the world and for inspiring me to get into robotics and computing. I want to thank Eben Upton himself for making computing accessible to everyone.

I want to acknowledge Udacity for giving me the platform to succeed and the tools to enhance my skills. I also want to thank them for the SUSE Scholarship challenge and the Cloud Native Application Architecture Nanodegree program. Thanks to them I was able to obtain the skills needed to build more complex projects and the knowledge to write this book. My thanks also extends to SUSE for partnering up with Udacity for both programs.

I also want to thank my friends who have been supportive in my journey and I thank everyone who has been inspired by my projects as I am forever grateful.

Preface

DevOps has become the fastest growing industry in the technology sector and continues to expand. Cloud Native Computing has been embraced by many companies like Amazon, Google and Facebook and allows companies to save time and money when setting up their services. However, for some it has been a challenge to learn how to use Cloud Native tools and apply them to their own projects.

This book seeks to remedy that by introducing you to the fundamentals of DevOps and Cloud Native Computing and showing you how to use Cloud Native tools with robotics. You will learn how each tool works as you build up your application from a simple web-based robot control application to a robot monitoring system that allows you to monitor your robots in real time.

Throughout this book, you will learn how to use tools like Docker, Kubernetes and Argo CD. While this book focuses solely on robotics, learning how to use these tools will help you with your own projects such as a website or a server. You will also learn about the future of Cloud Native Computing and robotics and concrete examples of small companies using these tools for their own products.

This book is intended for software developers who want to enter the field of DevOps for the first time, makers who want to take their projects to the cloud, robotics enthusiasts, robotics engineers, IoT enthusiasts and students.

With this book, you will be able to take your Raspberry Pi robots and control and monitor them with Cloud Native tools.

Chapter 1: Introduction to DevOps and Cloud Native Computing – This chapter introduces the basics of DevOps and Cloud Native Computing. This explains the steps for the reader which include building the robot and setting up the software for the project.

Chapter 2: Flask Robot Control – This chapter introduces the Flask web framework for Python and goes through a step by step process of building a Flask robot control application.

Chapter 3: Node.js/Express Robot Control – This chapter introduces the Node.js language and the Express web framework. The reader will learn the basics of Node.js, Express and how to build their first Express robot control application.

Chapter 4: Spring Boot Robot Control – This chapter introduces the Spring Boot web framework for Java and the basics. The reader will learn how to build a Spring Boot robot control application.

Chapter 5: Containerization with Docker – This chapter introduces Docker, the containerization application and the differences between containers and virtual machines. The reader will learn how to install Docker on their own system and some basic commands. Then the reader will package their application with Docker and push their newly created image to Dockerhub.

Chapter 6: Container Orchestration with Kubernetes – This chapter introduces Kubernetes, a container orchestration service that is used to deploy one or many containers. The reader will learn the different Kubernetes flavors like k3s and Kind and the reader will learn to install Kubernetes on their own machine, virtual machine or the cloud. The reader will learn the basics of Kubernetes, some basic commands to build a cluster and also deploy a cluster with manifests.

Chapter 7: Continuous Integration with GitHub Actions – This chapter introduces GitHub actions and the concept of Continuous Integration. The reader will learn the fundamentals of Actions, workflows and then they will setup their first workflows on GitHub.

Chapter 8: Continuous Delivery with Argo CD – This chapter introduces both the concept of Continuous Delivery and the service Argo CD. The reader will learn how to setup Argo CD on their cluster, how to access the interface and deploy their cluster to the interface.

Chapter 9: Monitoring with Prometheus – This chapter introduces the reader to metrics monitoring service Prometheus. The reader will update their application code so Prometheus can gather metrics and ensure that Prometheus can see the application in real time.

Chapter 10: Building a Dashboard with Grafana – This chapter will continue from the previous chapter and introduce Grafana which consumes the metrics from Prometheus and displays them as a table or graph. The reader will learn about the query language promQL which Grafana uses to query data and the reader will build their very first robot monitoring dashboard.

Chapter 11: Use Cases and the Future of Cloud Native for Robotics – This chapter explores the use cases for Cloud Native computing and discusses concrete examples of small companies using Cloud Native tools for their products.

Code Bundle and Coloured Images

Please follow the link to download the
Code Bundle and the ***Coloured Images*** of the book:

https://rebrand.ly/gvbhh53

The code bundle for the book is also hosted on GitHub at **https://github.com/bpbpublications/Cloud-Powered-Robotics-with-Raspberry-Pi**. In case there's an update to the code, it will be updated on the existing GitHub repository.

We have code bundles from our rich catalogue of books and videos available at **https://github.com/bpbpublications**. Check them out!

Errata

We take immense pride in our work at BPB Publications and follow best practices to ensure the accuracy of our content to provide with an indulging reading experience to our subscribers. Our readers are our mirrors, and we use their inputs to reflect and improve upon human errors, if any, that may have occurred during the publishing processes involved. To let us maintain the quality and help us reach out to any readers who might be having difficulties due to any unforeseen errors, please write to us at :

errata@bpbonline.com

Your support, suggestions and feedbacks are highly appreciated by the BPB Publications' Family.

Piracy

If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at **business@bpbonline.com** with a link to the material.

If you are interested in becoming an author

If there is a topic that you have expertise in, and you are interested in either writing or contributing to a book, please visit **www.bpbonline.com**. We have worked with thousands of developers and tech professionals, just like you, to help them share their insights with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions. We at BPB can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about BPB, please visit **www.bpbonline.com**.

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

https://discord.bpbonline.com

Table of Contents

Chapter 1
Introduction to DevOps and Cloud Native Computing

Introduction

When it comes to robotics, there have been many ways to run and control them, whether through a web interface, a GUI application, or using a remote control. However, there are no tutorials that show how to use cloud native tools to control and monitor robots. The purpose of this book is to show you how to use cloud native tools to build something different. This book will go through the steps to build a project that will monitor our robot. Through this book, we will also learn about these tools and how we can use them in our own projects outside of robotics.

In this chapter, we will cover the purpose of this book, the structure of the project, and introduce DevOps. We will learn about cloud native computing for the first time, learn how cloud native tools can be used with robotics, learn about the cloud native computing foundation, and we will go through getting set up for our project. This will include the compatible Raspberry Pi boards, setting up Python, NodeJS, or Java for the first time, installing the needed libraries for each language, and the environment for our initial tests.

Structure

In this chapter, we will cover the following topics:

- Introduction to cloud native

- Cloud native tools and robotics
- Cloud native and cloud native computing foundation
- Getting started

Objectives

After reading this chapter, you should be able to learn about the basics of DevOps. You will also gain knowledge about cloud native computing and learn how cloud native tools and robotics connect. You will also learn how to get started for the project.

To get a better view of our project structure, the following figure shows the progress we will be making from start to finish:

***Figure 1.1**: Project Flowchart*

To explain the flowchart above, first, we will require our code. Then we will package it in a container, then we will take that container and orchestrate the container. Then, we will use the CI/CD pipeline to make changes to our application and push them to production. This will be explained in *Chapter 7, Continuous Integration with GitHub Actions*, and *Chapter 8, Continuous Delivery with Argo CD*. We will then monitor our application and finally visualize our data.

Introduction to DevOps

DevOps is used to automate several processes. This includes code automation, application maintenance, and application delivery. DevOps engineers are responsible for making sure their applications are deployed correctly and can update any changes as needed. Thereafter, any changes made to the applications are deployed on time. Many companies like Facebook and Google rely on DevOps for their services and rely on DevOps engineers to automate their processes. From IT services to healthcare and other industries, DevOps plays an important role not just in deploying applications but also in ensuring that these companies continue to innovate.

Introduction to cloud native computing

Cloud native computing involves software development that uses cloud computing to scale applications in public, private, and hybrid cloud environments. Cloud native tools like Docker, Kubernetes, and Argo CD are important for companies like Amazon and Google to expand

and grow their services. Cloud native is different from cloud-based computing since cloud-based computing requires downtime for upgrades, and cloud native computing can upgrade without any interruption to services. This is important especially when dealing with services such as AWS or Google Cloud. Cloud native tools are often used for serving web pages, creating databases, setting up, and securing servers.

When it comes to cloud native, there are two different types of architectures: monolith and microservices. The following table shows the differences between a monolith and microservice architecture:

Monolith	Microservices
Part of the same application unit.	Managed in separate repositories.
Managed in a single repository (that is, GitHub).	Has its own allocated resources.
Shares the existing resources.	Well-defined **Application Programming Interface** (**API**) to connect to other interfaces.
Developed in one programming language.	Implemented with the programming language of choice.
Uses only one binary.	Uses its own binary to be released.

Table 1.1: *Monolith vs. Microservices*

To visualize the differences, *Figure 1.2* shows a monolith where there are three components for the application that are a part of the same unit:

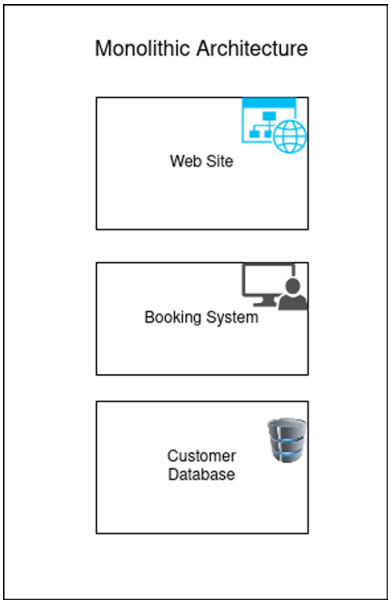

Figure 1.2*: Monolith architecture diagram*

Figure 1.3 shows the same booking application using the microservices architecture:

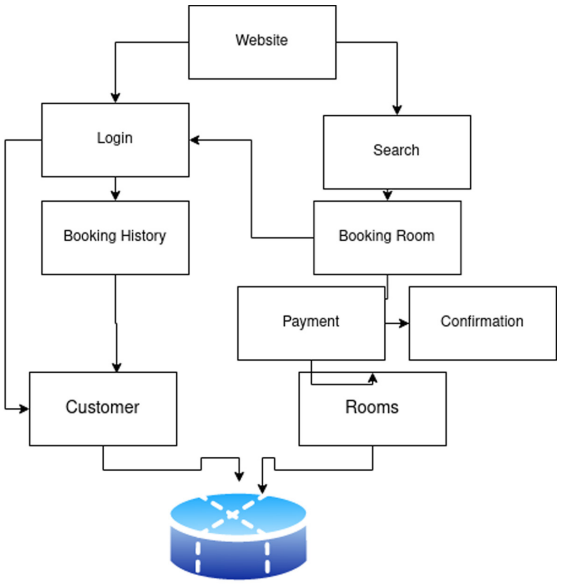

Figure 1.3: *Microservices architecture diagram*

For the purpose of this book, we will use the Monolith architecture since it will use the same repository on GitHub, will be developed in one programming language and will share existing resources.

Cloud native tools and robotics

How do cloud native tools and robotics relate to each other?

First, cloud native tools can be used for robots that utilize AI or robots that work in an assembly. This means we can use a monitoring service like Grafana to monitor the activity of our robots. As mentioned earlier, cloud native tools have often been used for serving web pages and databases but the same tools can be used for our robots too. Throughout the book, we will learn how these tools can be used for our robotics projects.

Cloud native and cloud native computing foundation

The cloud native computing foundation was founded in 2015 and has hosted many cloud native tools that we will be using throughout this book. Tools such as Kubernetes, Prometheus and Grafana are a part of this organization. The **Cloud Native Computing Foundation** (**CNCF**) also host events on-site and virtually, such as KubeCon which is the largest convention for cloud native computing, and team up with other organizations such as the Linux Foundation.

Getting started

Before we can begin with our monitoring application, it is important to set up our environment and get our robot assembled. The instructions for installation of our tools will be different for Windows, Mac, and Linux. First, let us discuss the compatible Raspberry Pi boards we can use for our robot.

Compatible boards

The first step is to build our robot, which means selecting a board that will be compatible for our project. Since our project requires a network connection, we will need a board that has built-in Wi-Fi. The boards that are acceptable for this project are as follows:

- Raspberry Pi 3B, as shown in *Figure 1.4*. For power recommendations, it is recommended to use at least a USB Power Bank with 5V and 3 Amps:

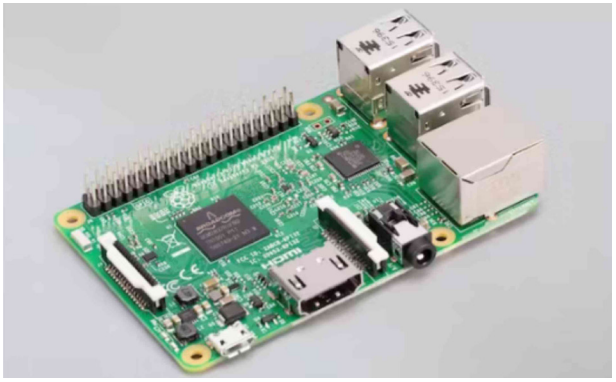

***Figure 1.4**: Raspberry Pi 3B (Source: raspberrypi.com)*

- Raspberry Pi 3B+, as shown in *Figure 1.5*. Just like with Pi 3, a 5V 3 Amp USB power bank is recommended:

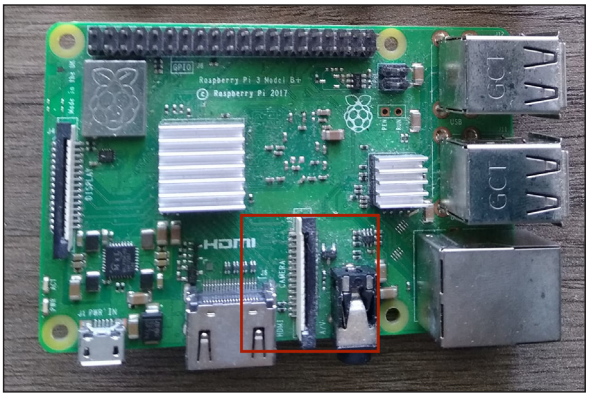

***Figure 1.5**: Raspberry Pi 3B+*

- Raspberry Pi 3A+, as shown in *Figure 1.6*. Just like with the 3B and 3B+, a 5V 3 Amp USB power bank is recommended:

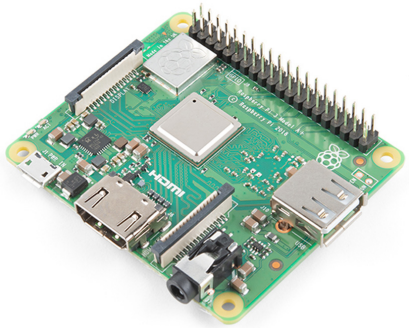

Figure 1.6: *Raspberry Pi 3A+ (Source: raspberrypi.com)*

- Raspberry Pi 4, as shown in *Figure 1.7*. A USB power bank of at least 5V and 3 Amps is recommended along with a USB-C cable. The 1, 2, 4 or 8GB model is acceptable:

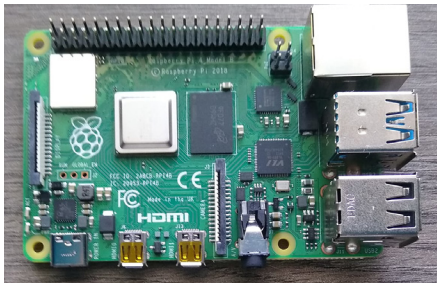

Figure 1.7: *Raspberry Pi 4*

- Raspberry Pi Zero W, as shown in *Figure 1.8*. A headless setup is preferred for the Zero W since a desktop is not required. A 5V 2 Amp USB power bank is recommended:

Figure 1.8*: Raspberry Pi Zero W*

- Raspberry Pi Zero W 2, as shown in *Figure 1.9*. A headless setup is preferred, just like the Zero W, and a 5V 3 Amp power bank is recommended due to the faster CPU:

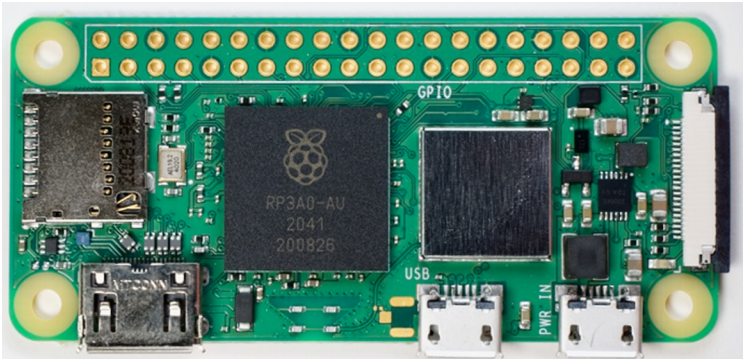

Figure 1.9: *Raspberry Pi Zero W 2*

- The Model B, B+, A+, Pi 2, and Pi Zero can be used but will require a USB Wi-Fi dongle.

However, in this case, we do not want to add extra hardware, so it is best to use the boards with built in Wi-Fi. The Pi 400 is not recommended as it is meant to be used as a Desktop. Also, the Pico W is a microcontroller, so, it is not compatible for this project.

Raspberry Pi OS

With the board chosen, the next step is to choose the OS that will run on our Raspberry Pi. For the purposes of this project, we will use Raspberry Pi OS Lite since we do not require a desktop. The following table shows which OS version is compatible for each board:

Board	Compatible OS
Pi 3B	Raspberry Pi OS Lite 32-bit and 64-bit
Pi 3B+	Raspberry Pi OS Lite 32-bit and 64-bit
Pi 4	Raspberry Pi OS Lite 32-bit and 64-bit
Pi 3A+	Raspberry Pi OS Lite 32-bit and 64-bit
Pi Zero W	Raspberry Pi OS Lite 32-bit
Pi Zero W 2	Raspberry Pi OS Lite 32-bit and 64-bit

Table 1.2: *Raspberry Pi compatibility table*

Wiring diagram

After choosing our board and OS version, next, we will need the parts for our robot. For Python and NodeJS, it is possible to use more than one robot for this project, which is entirely optional. We will need the following parts:

- Motor controller
- Jumper wires
- Two DC motors
- Batteries
- Battery holder
- Wheels
- Robot chassis

The most common motor controller used is the L298N. For this project, it is recommended to use this motor controller, but other controller boards can also be used. *Figure 1.10* shows a typical L298N:

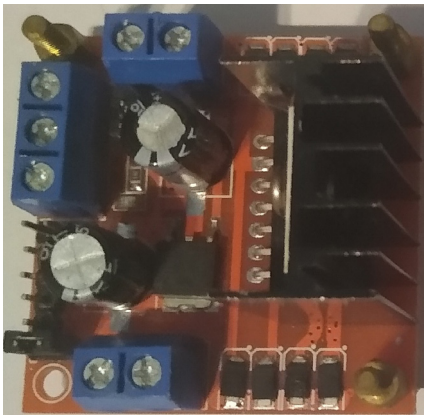

Figure 1.10: *L298N H-Bridge Motor controller*

After obtaining the parts, we then assemble the robot as shown in *Figure 1.11*. Other pins can be used for the motors and all that would be required is to update the code for those pins. In the diagram below we will be using GPIO pins 13, 21, 17 and 27 for our motors:

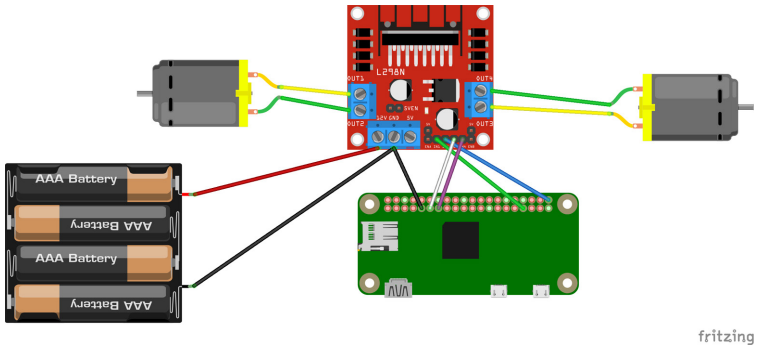

Figure 1.11: *Robot Fritzing diagram*

Speed control is optional for this project. *Figure 1.12* adds two extra jumper wires for speed control. We will be using GPIO pins 12 and 26, as shown below:

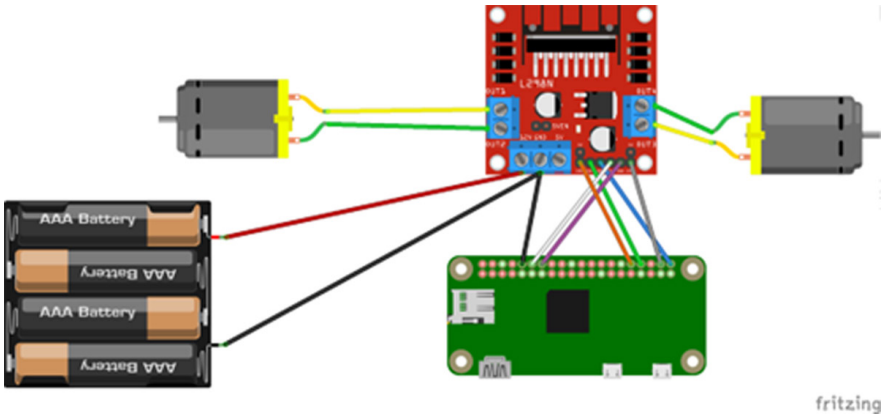

Figure 1.12: *Speed Control Robot Fritzing Diagram*

When it comes to building the robot, any chassis can be used for this project. For example, the robot below in *Figure 1.13* is built with cardboard for the face and uses PC style panel mount LEDs:

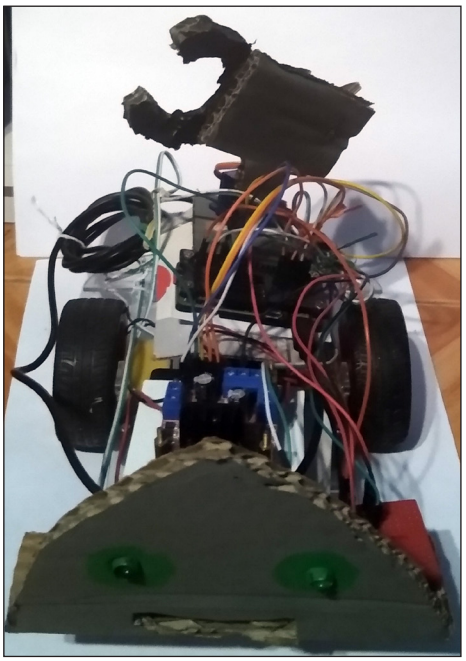

Figure 1.13*: Raspberry Pi robot with cardboard face*

Another example is to buy a chassis. *Figure 1.14* shows a robot built using the Devastator Tank Mobile Platform from DFRobot:

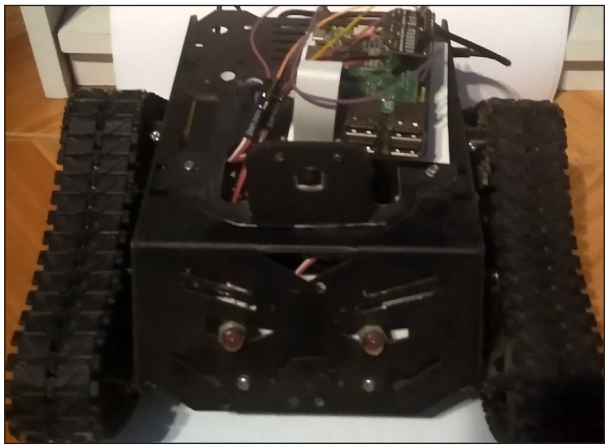

***Figure 1.14**: Raspberry Pi robot using devastator tank mobile platform*

Required tools and dependencies

After assembling our robot, we then need to install the required tools and dependencies for our project. As we proceed throughout the book, we will install the other required tools. However, we will need our base tools to get started.

Python

For Linux users Python is installed by default. However, if it is not installed, go to the official Python website to install Python. For Windows and Mac users, the best way to install Python is using the official Python website. After installing Python, we will need **`pip`** for installing Python libraries. For example, to install **`pip`** on Ubuntu, run the following command:

```
$ sudo apt install python3-pip
```

For Windows users, download the **`get-pip.py`** file and then run the following command:

```
$ py get-pip.py
```

For Mac users, first obtain the **`get-pip.py`** file by using this command:

```
$ wget http://bootstrap.pypa.io/get-pip.py
```

Then, run the following command:

```
$ python get-pip.py
```

After Python and `pip` are installed, we then need a text editor or **Interactive Developer Environment** (**IDE**) for our Python code. For this project, we will use Thonny (shown in *Figure 1.15*), but Mu, Visual Studio Code, Sublime, Nano, Vim, Geany, or a plain text editor can be used. The code will be run through the terminal after we write our application, as shown: