gribot

Specification for an open source agricultural robotics platform

gribot: Specification for an open source agricultural robotics platform

Table of Contents

List of Figures

List of Equations

Chapter 1. Introduction

In the requirement document, we described *what* the gribot platform should do and the goal to reach. The specification document describes how we will do it. This document is much more technical as it contains many subjects such as:

- Analysis
- Calculations and dimensioning
- Technology choices
- etc.

During the preparation of the specification document, it is often necessary to revisit the requirements definition document to clarify a point, and sometimes even to modify an unclear specification, or to add one to clarify the situation.

In this document, specification will be split into two main categories:

Platform specification	These specification concerns the whole *gribot* platform, such as general architecture, software to use, technology to use, etc. In other word, all elements common to the *gribot* robot family. As an example, the navigation system concerns the whole platform.
Robot specification	These specification concerns only one robot in the family, performing a specific function. It contains only those elements that are useful for this robot. As an example, the mower cutterbar does not concern the weeding robot.

The following chapters will describe these different specifications.

Chapter 2. Platform specification

The platform specification chapter contains element common to all robot that can be derived from the *gribot* platform. It goes from the propulsion system to navigation, including communication between the various elements.

System architecture

To be done

Electronic

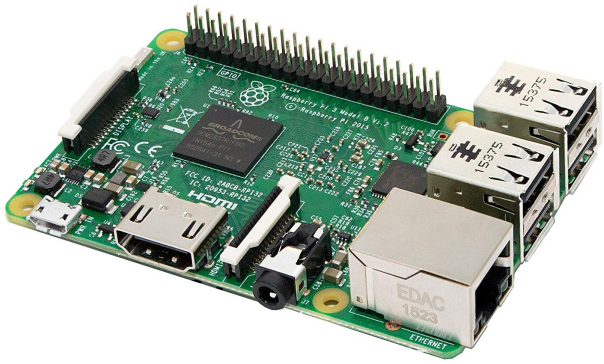
Embedded computer

The embedded computer is one of the critical elements of the robot. He is in charge of hosting the software that manages the robot. The embedded computer must satisfy several constraints:

- It must be affordable
- It must have enough memory to allow the operation of the various embedded software packages.
- Its disk capacity must be sufficient for the storage of the files necessary for the robot's operation.
- It should have a WiFi interface.
- It must have a family of interfaces, including CAN, I2C, USB, serial, etc. Either directly on the card or via plug-in cards.
- I must have interfaces or navigation charts such as GPS, accelerometers, electronic compasses, etc.
- It must support Ubuntu distribution.

Raspberry Pi is one of these small computer that fulfill almost all of the requirements above. At the time of writing, the available version is the Raspberry Pi 3.

Figure 2.1. Raspberry Pi



The Raspberry Pi 3 has enougth memory to run Ubuntu distribution together with ROS. It also supports high-capacity SD cards, and has a whole collection of interfaces and various cards.

Communication bus

Les différents composants électronique doivent pouvoir communiquer de manière fiable avec l'ordinateur embarqué. Il existe plusieurs solutions de communication:

Star communication
: Each component is connected individually to the central computer. This solution requires one connection cable per element as well as a dedicated interface on the central computer. The number of peripheral elements and the extension of the system is limited by the interfaces available on the computer.

Bus communication
: Data bus communication may be a little more complex to implement than star communication, but it offers much greater flexibility. It requires only a small number of interfaces, and the number of systems connected to the bus, and therefore its extension possibilities, are limited only by the technology used. The current buses make it easy to connect several tenths of peripheral equipment, which is more than enough for our needs.

We will retain the communication bus for our application. However, it is possible to connect some equipment directly to the central computer if necessary, provided that the interfaces are available.

The communication bus must comply with the following criteria:

- The bus must allow the prioritization of messages.
- The bus must allow real-time communication.
- The bus must be a standard on the market, in order to be able to connect various types of equipment.
- The bus must be reliable.
- The bus must be cheap.

Among the various buses available on the market, the CAN (Controller Area Network) bus is one that meets the above criteria. It comes from the automotive world and is widely distributed. The use of the CANOpen protocol allows the connection of a large number of equipment that support this standard.

Figure 2.2. CanOPEN logo

Development of the CAN bus started in 1983 at Robert Bosch GmbH. The protocol was officially released in 1986 at the Society of Automotive Engineers (SAE) conference in Detroit, Michigan. The CAN bus is not only limited to the automobile. Due to its characteristics, it is widely used in many sectors such as industrial automation, aviation, etc.

Software

Robot software

After some research, we decided to use Robot Operating System (ROS) as the management software for the gribot platform. ROS is an open source platform with a large community. This software is used by many robots, both in the research field and for production robots. On the other hand, the ROS platform has a very large number of modules and interfaces, which reduces development times.

Thanks to its modular design and communication bus, ROS makes it easy to develop missing modules: part of the software is installed on the robot, and the other part is under development on the developer's workstation. It is even theoretically possible that the developer may be located at a distance, in another city, another country or even another continent, from the development robot[1].

Operating system

At the time of writing, ROS is only supported on the Ubuntu operating system. However, this is not a major limitation given the very large installed base of this OS. There are Ubuntu distributions for a wide variety of platforms from servers to workstations, from embedded systems such as Raspberry Pi, Beaglebone, etc.

[1] We have not tested the feasibility of this way of developing at the time of writing.

Chapter 3. Mower specification

This chapter describes the specification related to the mower robot.

Propulsion

The propulsion might depend on the size of the robot and its geometry. that's why there's a propulsion paragraph in the mower chapter.

Maximum resistance to advancement

We can calculate the maximum resistance to advancement force with Equation A.6, “Total resistance”. With the following values:

C_{rr}: 0.3

m: 20kg

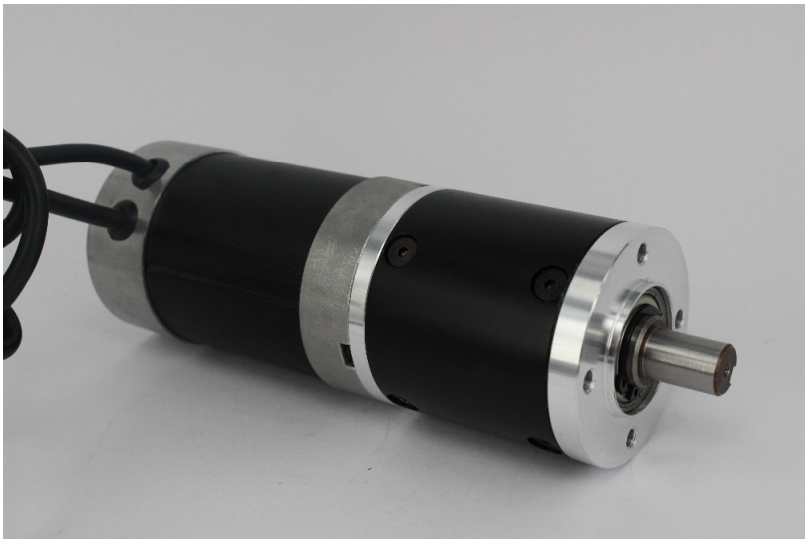
Slope: 45°

The maximum resistance is 180N.

Propulsion motor

To be done

Figure 3.1. Propulsion motor



Appendix A. Appendix

Resistance to advancement

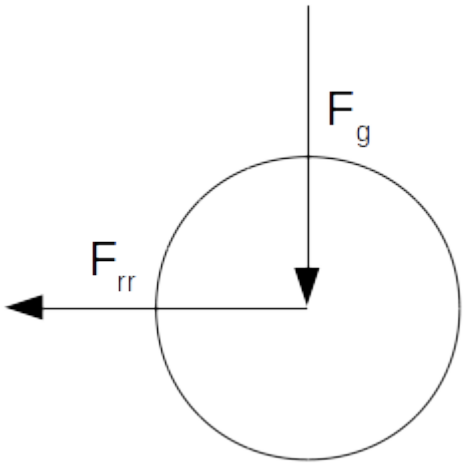
Wheel resistance to driving

The forward speed resistance is one of the parameters required to calculate the propulsion power of a wheeled vehicle. The two main parameters that influence the resistance to advancement are:

- The wheel forward resistance.
- The slope to be crossed.

According to [wiki01], rolling resistance calculations can be simplified if the vehicle does not move fast, which is our case. Not having found the rolling resistance coefficient for a tire on grass, we take the sand coefficient, 0.3, which is probably higher than that of grass. However, since the robot will move on uneven ground, which is not very similar to grass, this hypothesis leaves us a little margin.

Figure A.1. Forces on a wheel



The rolling resistance is given by:

Equation A.1. Rolling resistance

$F_{rr} = C_{rr} \cdot F_g$

Equation A.2. Gravitational force

$F_g = m \cdot g$

where

F_{rr}: rolling resistance force in N.

C_{rr}: rolling resistance coefficient.

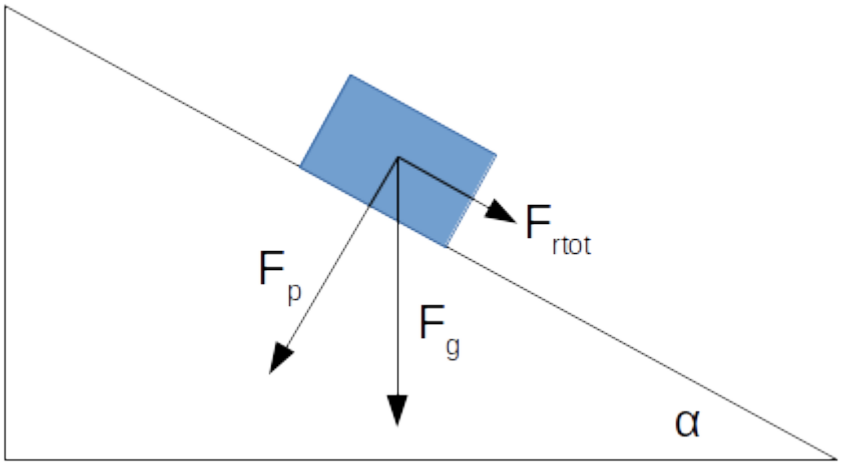
m: .mass of the vehicle (the robot for us) in Kg.

g: gravitational constant, in m/s^2.

Total resistance

Taking into account the slope, the calculation is a little more complex: it is necessary to add the components due to gravitation. In addition, the rolling resistance force decreases with increasing slope.

Figure A.2. Total resistance



The total resistance is due to the force of gravity and rolling resistance, which is:

$F_{rtot} = F_{rr} + F_{rg}$

where

F_{rr} is the rolling resistance force.

F_{rg} is the resistance force due to gravitation.

F_{rot} is the total resistance force: the one that the motors must overcome.

From the diagram and the triangle of forces, we calculate:

Equation A.3. Rolling resistance

$F_{rr} = C_{rr} \cdot F_p = C_{rr} \cdot F_g \cdot \cos(\alpha)$

Equation A.4. Gravity resistance

$F_{rg} = F_g \cdot \sin(\alpha)$

Equation A.5. Gravitational force

$F_g = m \cdot g$

by replacing the terms, you get:

Equation A.6. Total resistance

$F_{rtot} = C_{rr} \cdot F_g \cdot \cos(\alpha) + F_g \sin(\alpha) = F_g (C_{rr} \cdot \cos(\alpha) + \sin(\alpha)) = m \cdot g . (C_{rr} \cdot \cos(\alpha) + sin(\alpha))$

Chapter 4. Bibliography

Bibliography

[wiki01] *Rolling resistance*. Wikipedia.

[ROS01] *Robot Operating System*. ROS.

[Joseph2017] *ROS Robotics Projects*. Build a variety of awesome robots that can see, sense, move, and do a lot more using the powerful Robot Operating System. Packt Publishing Ltd.. Joseph Lentin. 978-1-78355-471-3. Copyright © 2017.

[Carol2016] *ROS Robotics By Example*. Bring life to your robot using ROS robotic application. Packt Publishing Ltd.. Carol Fairchild and Dr.. Thomas L. Harman. 978-1-78217-519-3. Copyright © 2016.

[ben2016] *VESC – Open Source ESC*.

Glossary

GPS RTK	Real Time Kinematic (RTK) is a satellite positioning technique based on the use of phase measurements of the carrier waves of signals emitted by GPS, GLONASS or Galileo systems. A reference station provides real-time corrections to achieve an accuracy in the centimeter range. In the particular case of GPS, the system is then called Carrier-Phase Enhancement or CPGPS.
SLAM	Simultaneous Localization And Mapping.
GPL	GNU General Public License