Step-by-step Development of an Omnidirectional Mobile Robot

Eddy Denegri, Emanuel Muñoz-Panduro, and Oscar E. Ramos
Department of Electrical Engineering, Universidad de Ingenieria y Tecnologia - UTEC, Lima, Peru

***Abstract*—Mobile robots are ubiquitous nowadays, and some companies build and sell them; but an inconvenience with these already built platforms is, for most of them, the limited access to their low level. Moreover, in spite of their ubiquity, there exists scarce information on how to build a complete mobile platform from scratch. This paper shows the detailed steps to build an omnidirectional robot based on mecanum wheels, for applications on autonomous exploration and navigation. The advantage of the holonomic configuration of the robot is the unconstrained instantaneous direction of motion. We provide a bottom-up guide for the construction of this robot, including the electronic devices, wiring, and the connection setup. Control and software frameworks are also detailed, and the result of the final robot together with some technical characteristics are presented.**

***Index Terms*—Omnidirectional robot, Swedish wheel, ROS (*Robot Operating System*).**

I. Introduction

Current trends in warehouse automation are related to the incorporation of mobile robot platforms for product transportation [1]. The development of these robots deals with issues related to energy efficiency, human-safe trajectory, multi-robot coordination, among others [2], [3]. Swedish omnidirectional robots stand out from this group for having only holonomic constraints, in contrast to the widely known differential robots. Their performance for warehouse applications presents advantages due to the narrow workspaces of these environments, where it is desirable that the robot easily move in any direction, such as sideways or diagonally [4].

Besides their omnidirectional motion, these robots are also required to perform optimal trajectories, avoiding obstacles, in their working environments. This problem is often referred to as the navigation problem, and it requires the robot to compute its motion based on the imprecise measurements of its surroundings. These measurements are acquired through sensors such as cameras, laser scans, RFID technology, among others. A good low-cost alternative is the Kinect sensor, which provides depth data, so that tasks like mapping or even SLAM algorithms are achievable.

The complete development of an omnidirectional mobile robot needs to take into account some aspects both related to hardware and software. There exist several approaches for controlling this type of robots, for improving their hardware performance, or for obtaining better accuracy for their pose estimation [5]. However, there does not exist a detailed guide for their full construction, which describes every part of the design and implementation process, from the hardware description to the kinematic equations. This paper presents a step-by-step construction of an omnidirectional robot based on swedish wheels and integrating a Kinect sensor, with a mechanical structure similar to the one shown in Fig. 1. A detailed description of the hardware and software setup, as well as the control and the kinematics, is provided.

978-1-7281-3818-3/19/$31.00 ©2019 IEEE

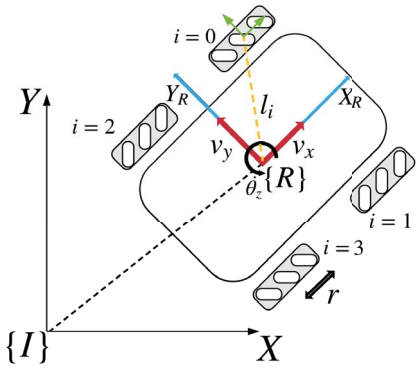

Fig. 1: Representation of the omnidirectional robot developed in this paper, with respect to a fixed inertial frame.

II. Robot Design and Motion Analysis

This section provides some specifications of the robot characteristics, and an analysis of its kinematics.

A. Omnidirectional Robot Design Considerations

Omnidirectional robots are one of the best approaches for dealing with two-dimensional environments due to their maneuverability [5], in contrast to other non-holonomic robots such as the differential robot. This remarkable performance is possible because of the widely known mecanum wheels, also called swedish wheels, which possess rollers with a certain fixed angle with respect to the main axis of rotation, allowing for omnidirectional motion. In our design we decided to use wheels with a standard angle of 45° due to their lower slippage when compared to wheels with 90°. Although only three contact points, represented by three wheels, are needed for motion and for the stability of a mobile robot, four contact points increase the support polygon and the maximum load it can withstand. Rectangular-shaped robots with wheels symmetrically located with respect to their center of mass,

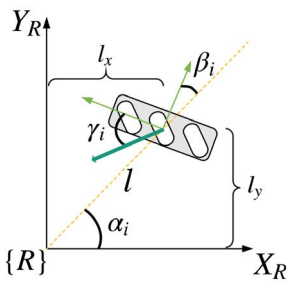

Fig. 2: Wheel representation from the robot frame

such as the one shown in Fig. 1, take advantage of the omnidirectionallity of each wheel leading to an omnidirectional platform with a smooth motion. In this work we decided to use this model, but the whole framework presented in this paper is applicable to robots with other shapes or configurations, being the kinematic anaylisis the only change.

B. Omnidirectional Robot Kinematics

The model of the robot used in this paper is shown in Fig. 1, where the wheel rollers depict the contact with the floor. Each wheel is tagged with a numerical index i, with $i = \{0, 1, 2, 3\}$, and it should be noted that their configuration is not interchangeable. The velocity of rotation of each wheel will be denoted as $\dot{\varphi}$. Let frame $\{R\}$ be attached to the center of mass of the robot, with its axes pointing as shown in Fig. 1. The instantaneous velocity of the robot, with respect to its own reference frame $\{R\}$, will be denoted as

$$ {}^{R}\dot{\zeta} = \begin{bmatrix} v_x & v_y & \omega \end{bmatrix}^T \tag{1} $$

where v_x and v_y represent the linear velocities in axes x and y, respectively, and ω represents the angular velocity about its z-axis. Let $\{I\}$ be an inertial frame, such that the robot pose is represented with respect to this frame as ${}^{I}\zeta = [x \quad y \quad \theta]^T$, and its velocity as ${}^{I}\dot{\zeta} = [\dot{x} \quad \dot{y} \quad \dot{\theta}]^T$. Note that $\omega = \dot{\theta}$ since the z axis of both frames $\{R\}$ and $\{I\}$ are always parallel (assuming a purely planar motion). The relation between frames $\{R\}$ and $\{I\}$ is given by:

$$ {}^{R}\dot{\zeta} = R^T(\theta)\ {}^{I}\dot{\zeta} \tag{2} $$

where $R_z(\theta) \in SO(3)$ is a rotation matrix about the z-axis and represents the orientation of the robot frame with respect to the inertial frame. The kinematic relation between the velocity of the wheels $\dot{\varphi}$ and the velocity of the robot in its reference frame $\dot{\zeta}$ can be represented as

$$ J_1\, {}^{R}\dot{\zeta} = -J_2\dot{\varphi} \tag{3} $$

where $\dot{\varphi} = [\dot{\varphi}_1 \quad \dot{\varphi}_2 \quad \dot{\varphi}_3]^T$, and the components of J_1, J_2 are

$$ J_1 = \begin{bmatrix} J_{1_0}^T & \cdots & J_{1_4}^T \end{bmatrix}^T \qquad J_2 = \text{diag}(J_{2_0}^T, \cdots, J_{2_4}) $$

with J_{1_i} and J_{2_i} representing the rolling constraint of each wheel as

$$ J_{1_i} = [\sin(\alpha_i + \beta_i + \gamma_i), -\cos(\alpha_i + \beta_i + \gamma_i), -l_i \cos(\beta_i + \gamma_i)] $$

$$ J_{2_i} = r\cos(\gamma_i) $$

The angles for each wheel (α, β, γ) are defined as specified in Fig. 2, where γ is an angle depending on the roller orientation within the wheel. From (3), the inverse kinematics is

$$ \dot{\varphi} = -J_2^{-1} J_1\, {}^{R}\dot{\zeta} = J\, {}^{R}\dot{\zeta} \tag{4} $$

where $J = -J_2^{-1}J_2$ represents the Jacobian transformation from ${}^{R}\zeta$ to $\dot{\varphi}$ Rearranging (3), it is also possible to find the forward kinematics as

$$ {}^{R}\dot{\zeta} = J^{+}\dot{\varphi} \tag{5} $$

Using (5) and (2), we can compute the pose of the robot with respect to the inertial frame at every instant of time through integration as:

$$ {}^{I}\zeta(t) = \int_0^t {}^{I}\dot{\zeta}(\tau)d\tau. \tag{6} $$

All these kinematic expressions are important to compute the control scheme in the following sections.

III. Hardware Setup

This section shows the details of the electrical connections of the robot system, as shown in Fig. 3.

A. Digital Connection

The connections of the embedded systems are done in the following way: a Jetson Tx1 is connected to a Raspberry Pi 3b+ through an Ethernet interface. The exchange of data between these two controllers is achieved through sockets, where the parameters that should be set up to obtain a successful connection, both in the master and in the slave, are the IP number, the protocol type, and the buffer size. In this case, the Jetson Tx1 behaves as the master controller since it is the board where most of the high-level processing of the whole system will be carried out, and the Raspberry Pi 3b+ is configured as a slave, sending all the odometric information to the master. The Raspberry also performs the function of controlling the desired speed of the motors through several Arduino boards using the serial port protocol. The robot also has a Kinect v2 sensor, which is connected to the master controller using a USB 3.0 port.

The number of counts of the encoders are read using an Arduino, which uses an ATmega 328p microcontroller, for each encoder. For reading these values, it is necessary to configure two pins for quadrature reading in an interrupt mode, which work both for signals A and B (which are in quadrature) of the encoder. With these counts, it is possible to determine the measured angular position and the measured angular velocity of each motor.

B. Power Connection

The power connection provided to the robot must be enough to cope with the four high-current motors that were used. For the control of these actuators, it is necessary to use a power circuit which, in this work, is composed of two H-bridges, two pins for the generation of the PWM (*Pulse Width Modulation*)

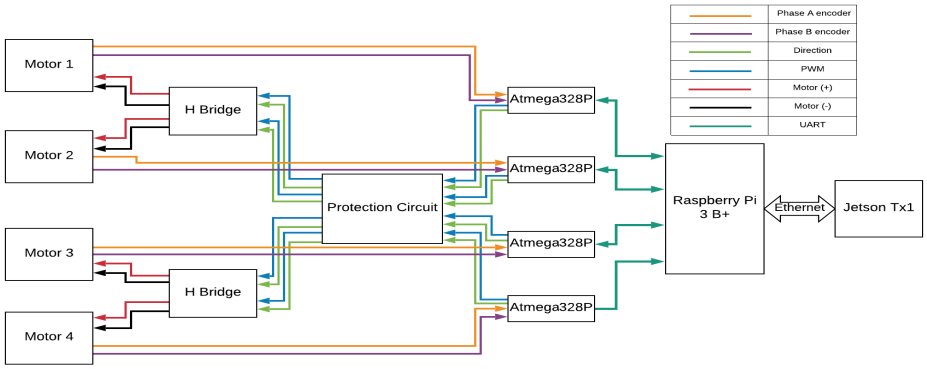

Fig. 3: General connection diagram of the system, showing the protocols that were used

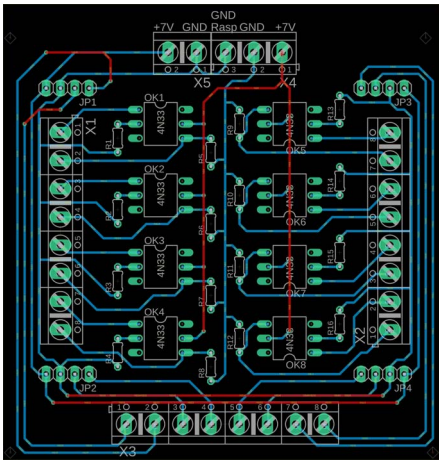

Fig. 4: Optocoupler board

signal, and the logical pin. The last two signals are required since the H-bridge uses these two inputs. The frequency set for the PWM signal is 32.984 KHz. The H-Bridge model used is a Dual MC33926 Motor Driver Carrier, which has an operating voltage ranging from 5V to 28 V, and it can provide a current of almost 3A. The battery used in this work has a maximum voltage of 12V and a capacity of 45 Watts/Hr.

C. Protection Circuit

We designed a board to protect the GPIOS of the microcontroller ATmega 328p. Fig. 4 shows the design of this board which uses eight optocouplers of model CNY17. Te use of these optocouplers is necessary to separate two circuits: the digital connection and the power connection. The signals that flow through this integrated signal are PWM and logical pins. The optocoupler operates with a 7.5V power supply and can operate at frequencies up to 1 MHz.

IV. Software Setup and Control Scheme

The implementation of the proposed system requires an adequate software setup for the management of the data. This section provides details of the software framework used for the proposed robotic system.

A. Software Setup

ROS (*Robot Operating System*) is a highly used framework for different robot models since it is multi-purpose, multi-lingual, and open-source. It is based on a node-message-topic paradigm that allows the communication between processes and between different computers, following the master-slave paradigm. In this work, we exploit its data management paradigm to exchange data from sensors, actuators, and controllers. ROS Kinetic was used in the Raspberry Pi+3 and in the Jetson, with the latter one in master form.

1) Raspberry Configuration: The Raspberry uses ROS for interchanging information about the current and desired angular velocities of the motors with other devices. Specifically, we use four nodes for publishing the measured angular velocity of each motor. Independent nodes, each publishing to its respective topic, is used instead of a unique one for all motor information, to avoid problems with delays in the signals. Recall that the read value is provided to the Raspberry by the Arduino, which works at high frequency to avoid high jerks in the motors. In addition, there are four other nodes receiving desired angular velocities sent to the Arduino through the aforementioned protocols. Another extra node performs forward and inverse kinematic operations, publishing the desired angular velocity and reading its measured value. Integration and derivation, used in the control scheme, are also a task performed in this node, which publishes the pose of the robot in the topic called `/odom`.

2) Jetson Configuration: The Jetson board receives information through the topic `/odom` for higher-level feedback in applications like mapping or exploration. Furthermore, for using the kinect with ROS we run the iai_kinect2 package [6], and all the dependencies that are needed are specified in this package configuration.

B. Control scheme

The complete control framework for controlling the velocity of the robot and determining its current position from odometry is shown in Fig. 5. We opted for a velocity controller since it is a standard for path planning algorithms. Odometric information from the wheels is also desirable for the prediction

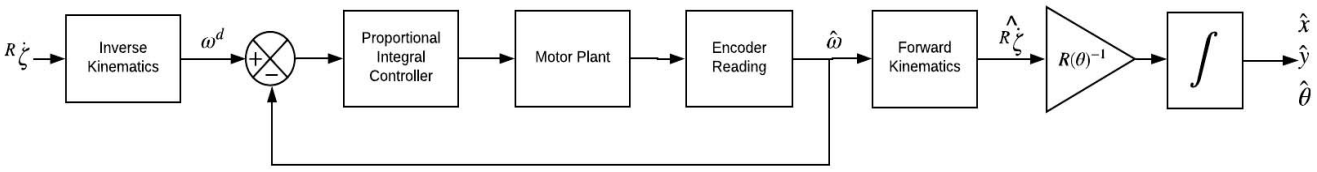

Fig. 5: Scheme of the low level control of the robot

TABLE I: Robot parameters

wheel i	α_i	β_i	γ_i	l_{ix}	l_{iy}
0	$\pi/4$	$\pi/2$	$-\pi/4$	l_x	l_y
1	$-\pi/4$	$-\pi/2$	$\pi/4$	l_x	l_y
2	$3\pi/4$	$\pi/2$	$\pi/4$	l_x	l_y
3	$-3\pi/4$	$-\pi/2$	$-\pi/4$	l_x	l_y

of the robot localization, but it can be omitted if exteroceptive sensors are used. However, the encoder information is important for this scheme because it is used for the low-level closed-loop feedback proportional-integral controller.

The first phase of the controller consists in converting the desired Cartesian velocity to a desired wheel velocity using (4). Then, a PI controller is applied to the error between the desired wheel rotational velocity $\dot{\varphi}^d$ and its estimated value $\hat{\varphi}$. The output of the controller is sent to each motor using PWM. Parameter tuning could be obtained from the model of each motor or could be empirically calculated by trial and error. Also, for negative angular velocities, a logical law for manipulating reverse or forward configuration is specified. The response of the motor is then measured by encoders that read the angular position and velocity of each wheel. After the closed-loop feedback, the estimated angular velocity is transformed into robot velocity using (5). As (6) states, an inverse rotation and integration needs to be applied to obtain the estimated robot pose with respect to an inertial frame.

V. Results

The implemented robot is shown in Fig. 6, which contains all the connections described above. For this, the parameters of the mechanical design are shown in Table I, where $l_x = 22.5$ cm and $l_y = 17.5$ cm. Note that the chosen swedish wheels have an angle of 45°. Using these parameters, the inverse kinematics in (7) becomes

$$\begin{bmatrix}\dot{\varphi}_1\\ \dot{\varphi}_2\\ \dot{\varphi}_3\\ \dot{\varphi}_4\end{bmatrix} = \frac{1}{r}\begin{bmatrix}1 & -1 & -(l_x+l_y)\\ 1 & 1 & (l_x+l_y)\\ 1 & 1 & -(l_x+l_y)\\ 1 & -1 & (l_x+l_y)\end{bmatrix}\begin{bmatrix}v_x\\ v_y\\ \omega\end{bmatrix} \tag{7}$$

and the forward kinematics in (8) becomes

$$\begin{bmatrix}v_x\\ v_y\\ \omega\end{bmatrix} = \frac{1}{r}\begin{bmatrix}1 & 1 & 1 & 1\\ -1 & 1 & 1 & -1\\ \frac{-1}{l_x+l_y} & \frac{1}{l_x+l_y} & \frac{-1}{l_x+l_y} & \frac{1}{l_x+l_y}\end{bmatrix}\begin{bmatrix}\dot{\varphi}_1\\ \dot{\varphi}_2\\ \dot{\varphi}_3\\ \dot{\varphi}_4\end{bmatrix} \tag{8}$$

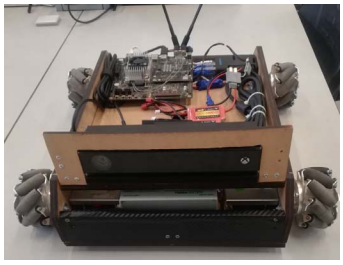

Fig. 6: Prototype of the Omnidirectional Robot

The chasis of the robot is based on wood MDF because of its easy manipulation, strength and low cost. On the top of the robot, a Kinect v2 sensor is located for the acquisition of environmental information. The robot has a payload of approximately 8 kg. In total, the dimensions of the robot are $60 \times 40cm$ with wheels of radius $5cm$.

VI. Conclusions

In this work, we presented a complete guideline for the construction of an omnidirectional robot based on four swedish wheels. We showed a bottom-up description of the development, providing details for the hardware and software implementation. The robot is composed by widely known devices and DC motors, together with a low-cost chasis. We also used open-source software and a straight control scheme for implementation. A Kinect camera was implemented on the top of the robot to provide data sensing commonly used for navigation. The developed framework can be applied to applications related to mapping, exploration, path planning implementation among others.

References

[1] Z. Shuangxin, "The present and future of warehouse robot," *Logistics Engineering and Management*, vol. 35, no. 6, pp. 171–172, 2013.

[2] M. Plooij, M. Wisse, and H. Vallery, "Reducing the energy consumption of robots using the bidirectional clutched parallel elastic actuator," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1512–1523, 2016.

[3] P. A. Lasota, T. Fong, J. A. Shah *et al.*, "A survey of methods for safe human-robot interaction," *Foundations and Trends® in Robotics*, vol. 5, no. 4, pp. 261–349, 2017.

[4] J. Efendi, M. Salih, Y. Sazali, M. Rizon, and M. Juhari, "Omni-directional mobile robot with mecanum wheel," 2005.

[5] R. Comasolivas, J. Quevedo, and Others, "Low level control of an omnidirectional mobile robot," in *Mediterranean Conference on Control and Automation*. IEEE, 2015, pp. 1160–1166.

[6] T. Wiedemeyer, "IAI Kinect2," https://github.com/code-iai/iai_kinect2, Institute for Artificial Intelligence, University Bremen, 2014 – 2015, accessed June 12, 2015.