M. S. RAMAIAH INSTITUTE OF TECHNOLOGY

(Autonomous Institute Affiliated to VTU)

MSR Nagar, M.S.R.I.T Post, Bangalore - 560054

Project entitled

Behavioral Control Design for Robot Navigation

submitted in partial fulfillment for the award of degree in

Bachelor of Engineering in
Electronics and Communication Engineering

By

Paritosh Kelkar	1MS10EC073
Prajwal S	1MS10EC079
Santhosh Shetty	1MS10EC107
Siddharth	1MS10EC120

under the guidance of
Suma K. V.
Assistant Professor

Department of Electronics and Communication
M. S. Ramaiah Institute of Technology

June 2014

Department of Electronics and Communication
M. S. Ramaiah Institute of Technology
Bangalore

CERTIFICATE

This is to certify that the project report entitled 'Behavioral Control Design for Robot Navigation' is a bonafide record of work done by Paritosh Kelkar [USN: 1MS10EC073], Prajwal S [USN: 1MS10EC079], Santhosh Shetty [USN: 1MS10EC107] and Siddharth [USN: 1MS10EC120] under my supervision during the year 2013-2014. The project report has been approved as it satisfies the academic requirements with respect to project work prescribed for the Bachelor of Engineering degree.

Signature of the guide	Signature of the HOD
Mrs. Suma K. V.,	Dr. S. Sethu Selvi,
Assistant Professor,	Head of the Department,
Dept. of E&C,	Dept. of E&C,
MSRIT	MSRIT

Name of examiner	Signature with date

Department of Electronics and Communication
M. S. Ramaiah Institute of Technology
Bangalore

DECLARATION

We hereby declare that the project entitled 'Behavioral Control Design for Robot Navigation' was carried out at M. S. Ramaiah Institute of Technology under the guidance of Asst. Prof. Suma K. V., Dept. of Electronics and Communication. No part of this project has been submitted elsewhere for any other degree or qualification. Various sources from which information was obtained for assistance in carrying out the project have been duly acknowledged and referenced.

Paritosh Kelkar	1MS10EC073
Prajwal S	1MS10EC079
Santhosh Shetty	1MS10EC107
Siddharth	1MS10EC120

Acknowledgement

We express our sincere thanks to Dr. S. Y. Kulkarni, Principal, MSRIT, and to Dr. S. Sethu Selvi, Head of the Department, Electronics and Communication for the technical and administrative support.
We express our sincere gratitude to Mrs. Suma K. V., for having whole heartedly agreed to guide our project. Her guidance, support and confidence in our ability has been instrumental in getting the project to where it is today.
We thank Mr. C. G. Raghavendra, Project Co-ordinator for conducting the Project Exhibition successfully on such a large scale and especially for his patience and consideration in addressing our concerns and providing us with an appropriate venue for exhibiting the project.
We finally express our gratitude to our beloved parents. To all those nights, days, weeks and months that we worked in each others' houses, we thank you for accommodating the group and for the overwhelming support.

Abstract

The project involves navigation of a wheeled robot in an unknown cluttered environment. The differentially driven mobile robot is modeled as a unicycle for control design. Wheel odometry is used to track the position and orientation (pose) of the robot and Sonar is used to sense obstacles in the environment. A behavior based approach with three controllers (behaviors), namely, go-to-goal, avoid obstacle and follow wall is employed. A supervisor that switches between these controllers using appropriate guards and reset conditions is designed for robust navigation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Robotics is the technology that deals with design, construction, operation, and application of robots. It is a multi-disciplinary field and borrows for mechanical and electrical engineering as well as computer science. Modern robotics involves elements such as Mechanical Design, Perception, Artificial Intelligence, Control Design and Programming. Robots can operate in either static or dynamic environments. A robotic arm in an assembly line, for instance, is required to perform exactly the same task repeatedly. Such robotics systems know what they are required to perform at every point in time and do not need to make real time decisions on what actions to execute. On the other hand, a robot operating in a dynamic unknown environment, such as a driverless car needs to able to quickly sense and react to the surroundings. Such systems have to satisfy requirements of stability (a driverless car should not crash), tracking (it should stick to its lane on the road) and robustness (a gale shouldn't throw the car off course). Such a system poses several challenges to the designer in terms of perception and control.

1.1 Behavior Based Robotics

Behavior based robotics is one approach to control robots operating in dynamic environments. This approach involves breaking down the problem into smaller portions and designing independent controllers for each portion. The robot thus exhibits a finite number of fairly simple behaviors. The solution to the overall problem then simplifies to that of choosing one of the behaviors depending on the environment. Such behavior based robots exhibit complex-appearing behaviors despite little internal representation to model the immediate environment, mostly gradually correcting actions via sensory-motor links. Behavior based robots usually show biological-appearing actions and are sometimes considered examples of weak artificial intelligence.

1.2 Autonomous Robot Navigation

A vital component of unmanned machines or rovers is the navigation system that enables these machines to autonomously navigate to the required destinations. The machines with autonomous navigation capability can be employed in various applications such as autonomous land navigation, unmanned extraterrestrial and underwater exploration, maintenance and repairs in nuclear power plants, operation in chemical and toxic industries, unmanned vigilance and security systems, etc.

1.3 Objective

To drive a differentially-driven wheeled mobile robot to a set of destination co-ordinates on the floor plane in an indoor environment while smoothly navigating around any obstacles in the path of the robot.

Chapter 2

Overview

The robotic system employs hardware and software elements to meet the stated objective. Hardware components include ultrasonic sensors for Sonar, infrared sensors for wheel odometry, an embedded processor that executes the control strategy, and a motor driver to run the motors and drive the robot. Software refers to the program that implements the control strategy.

An overview of electrical signal flow, power distribution and the layout of sensors employed on the robot along with relevant physical dimensions is provided in this chapter, the sensing and actuation mechanisms (and sensor details) are explained in chapter 3, the details of the embedded processor are provided in chapter 4 and the control design is described in chapter 5.

2.1 Signal Flow

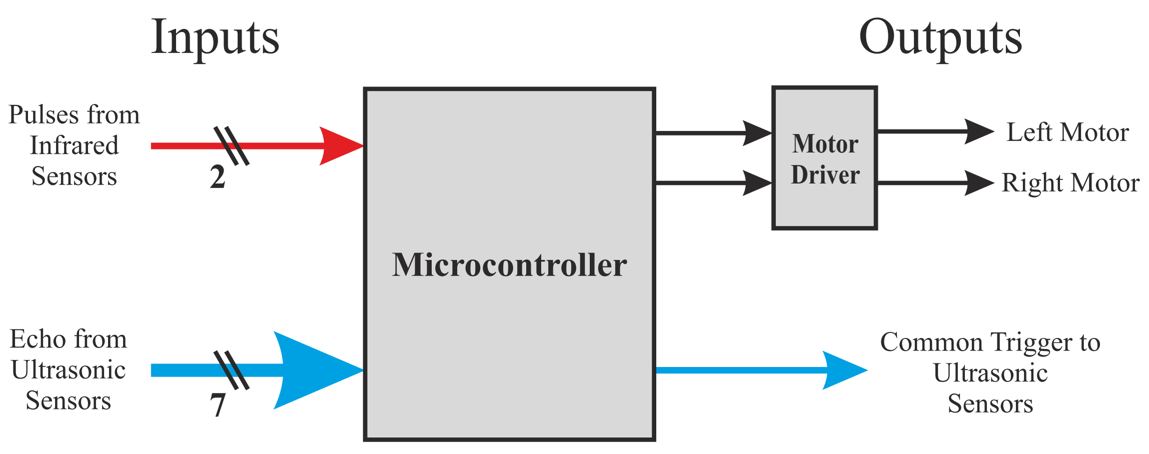
Inputs
Outputs
Pulses from Infrared Sensors
2
Echo from Ultrasonic Sensors
7
Microcontroller
Motor Driver
Left Motor
Right Motor
Common Trigger to Ultrasonic Sensors

Figure 2.1: Signal Flow

The signal flow diagram shown in Fig. 2.1 provides an understanding of the different inputs and outputs coming in and going out of the microcontroller.

The IR sensors and the ultrasonic sensors continuously send data in the form of pulses to the microcontroller. Based on the received signal, the microcontroller performs computations and provides an appropriate signal to the motor driver for running the motors. Each motor gets an independent signal. The microcontroller also sends a common trigger signal to the ultrasonic sensors to periodically initiate the sensing action.

2.2 Power Distribution

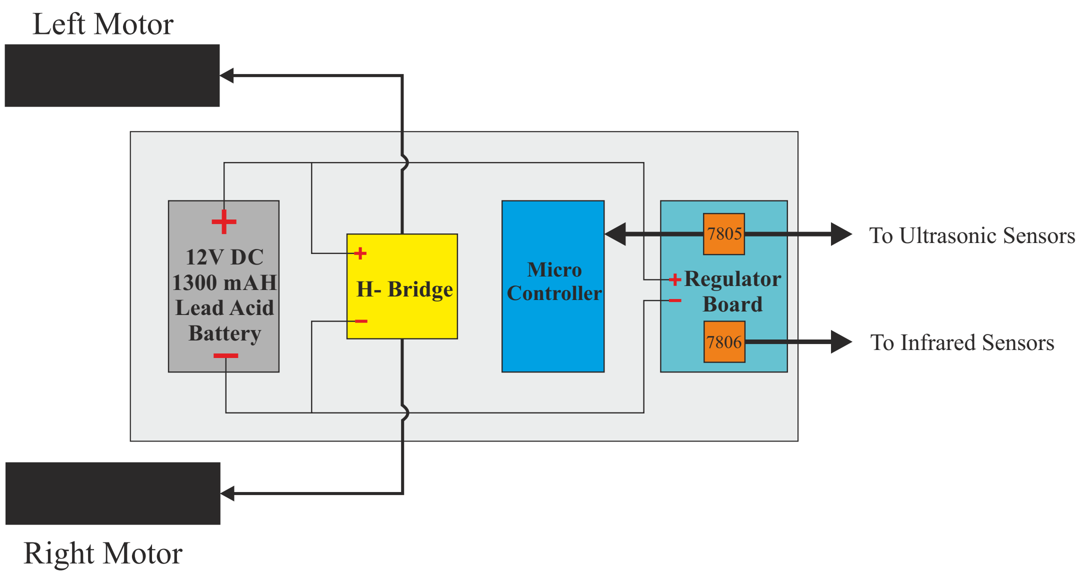

Figure 2.2: Power Distribution

The main power supply to the robot comes from a 12V lead acid battery which has an energy capacity of 1300mAh. The power to the motors is fed through the H-Bridge. Two regulators (7805 and 7806) are used to regulate the battery voltage from 12V supply to 5V (used as a power supply for the microcontroller board and ultrasonic sensors) and from 12V to 6V (used as a power supply for the IR sensors).

2.3 Layout of the Robot

The layout of the ultrasonic sensors was designed such that the perception ability of the robot was optimized. By placing the ultrasonic sensors at angular intervals of 30°, the blind spots between the sensors was reduced. The robot, being differentially driven has a caster wheel at the front and two independently driven wheels at the back. The radius of the back wheel is 3.5 cm, and the distance between the two wheels is 20 cm. The infrared sensors face the wheels to form the wheel encoder.

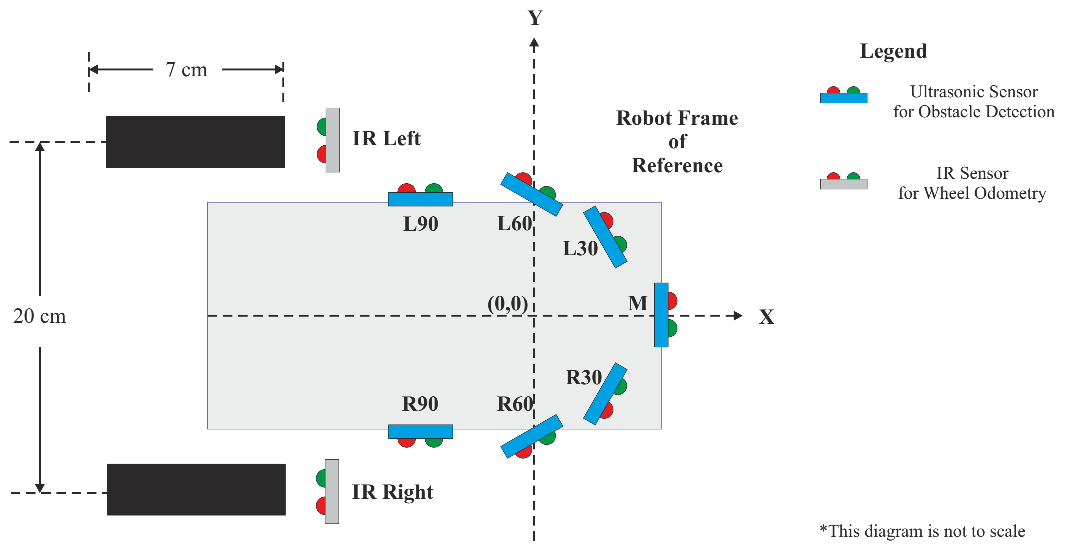
Y
7 cm
Legend
Ultrasonic Sensor for Obstacle Detection
IR Left
Robot Frame of Reference
IR Sensor for Wheel Odometry
L90
L60
L30
20 cm
(0,0)
M
X
R30
R90
R60
IR Right
*This diagram is not to scale

Figure 2.3: Layout of the Robot

Chapter 3

Sensing and Actuation

This chapter explains the working of the ultrasonic sensors for Sonar and infrared sensors for wheel odometry. It also gives a brief description of the motor driver used and its working.

3.1 Wheel Odometry

Odometry is the use of data from sensors to estimate change in position over time. Wheel odometry has been implemented to compute the pose (position and orientation) of the differentially driven robot using a wheel encoder.

A wheel encoder counts the number of revolutions of the wheels of the robot in a fixed interval of time. (In practice, the time interval is small and the fraction of one revolution is what is measured.) The wheels of the robot are stuck with alternate white and black strips. An infrared (IR) sensor is used to count the number of transitions between the white and black strips. The IR sensor (Fig. 3.1) consists of a transmitting LED, a receiving photodiode and an Op-amp in open-loop configuration. The LED beams out infrared waves while the photodiode collects the reflections. When the IR sensor is "looking at"(pointed at) a white strip, incident waves are reflected back and the Op-amp output is at positive saturation. When the IR sensor is pointed at a black strip, most of the radiation is absorbed by the strip and the Op-amp output is at negative saturation (ground level). The Op-amp output is continuously read by the microcontroller which counts the fraction of revolution completed in the fixed time interval.

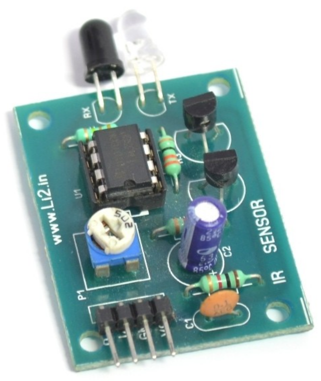

Figure 3.1: Infrared (IR) Sensor

There are 24 strips of white paper on the wheel interspersed with 24 strips of the natural black of the wheel. Hence, one revolution is 48 transitions between black and white. (The greater the number of strips, the greater is the resolution of the wheel encoder. However, divergence of the IR beam is a limiting factor in deciding the spacing between 2 white strips)

The number of revolutions, N made by the wheel in the fixed time interval is the given as

$$N = \frac{Transitions}{48} \tag{3.1.1}$$

If R is the radius of the wheel, the linear distance D traveled by the wheel is

$$D = 2\pi RN \tag{3.1.2}$$

The distance traveled by the robot (not to be confused with the distance traveled by each wheel) in the fixed time is found by calculating the mean of distance traveled by the left and right rear wheels.
If D_L is the distance traveled by the left wheel and D_R is the distance traveled by the right wheel, the distance traveled by the robot is

$$D_{bot} = \frac{D_L + D_R}{2} \tag{3.1.3}$$

The change in orientation (angular displacement) of the robot is proportional to the difference in speeds of the left and right wheels (Fig. 3.2). Knowing wheel radius R, and the separation between the wheels L, the angular displacement ($d\phi_{bot}$) can be found as

$$d\phi_{bot} = \frac{D_R - D_L}{L} \tag{3.1.4}$$

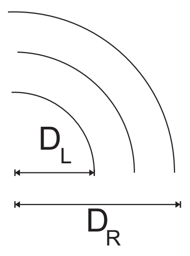

Figure 3.2: Difference in distance travelled by left wheel and right wheel

3.1.1 Issues with Infrared Sensors

1. Since the transmitting LED of the IR sensor emits a divergent beam and receiver photodiode of the IR sensor has a wide aperture, there is a high possibility of two adjacent white strips being interpreted as a single white strip with the black strip between them being skipped (Fig. 3.3). In such a scenario, there is a miscount of 2 transitions (failure to count 1 white to black and 1 black to white transition). To overcome this difficulty, careful tuning of the potentiometer which controls the Op-amp reference signal was employed. Also, by experimentation an appropriate strip width for the white paper strips was found.

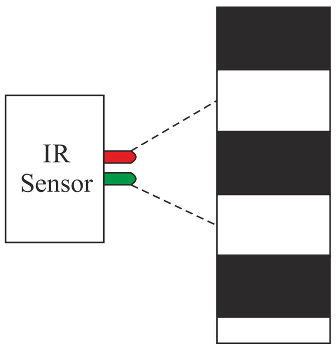

Figure 3.3: Two adjacent white strips interpreted as a single white strip

2. The output of the sensor's Op-amp is about 2V lesser than the Vcc of the Op-amp. Hence, when the regulated 5V available from the motor driver board was used as Vcc, the output of the sensor sometimes fell below 3V (which is required by the micro-controller to recognize the signal as a HIGH). As a result, the micro-controller would sometimes misinterpret the signal as logic LOW when it was actually logic HIGH. This problem was espe-

cially significant when the battery not only powered the sensor but the motors as well. To overcome this problem, a 6V regulator (7806) is made use of to supply power to the IR sensors.

3.2 Sonar

The HC-SR04 ultrasonic sensor shown in Fig. 3.4, uses Sonar to determine distance to an object like bats or dolphins do. It offers excellent non-contact range detection with high accuracy and stable readings. Its lobe shaped beam pattern has a beam width of around 30° at 17.5cm and reduces drastically as the distance to the object increases. Its operation is not affected by sunlight or black material (although acoustically soft materials like cloth can be difficult to detect).

3.2.1 Specification of the Ultrasonic Sensor

Table 3.1 lists the key parameters of the HC-SR04 Ultrasonic Sensor.

Figure 3.4: Ultrasonic Sensor (HC-SR04)

PARAMETER	VALUE
Working Voltage	5V DC
Working Current	15mA
Working Frequency	40KHz
Max Range	4m
Min Range	2cm
Measuring Angle	15°
Trigger Input Signal	10μs TTL Pulse
Echo Output Signal	I/p TTL lever Signal and the range in proportion
Dimension	45*20*15 mm

Table 3.1: Key parameters of Ultrasonic Sensor

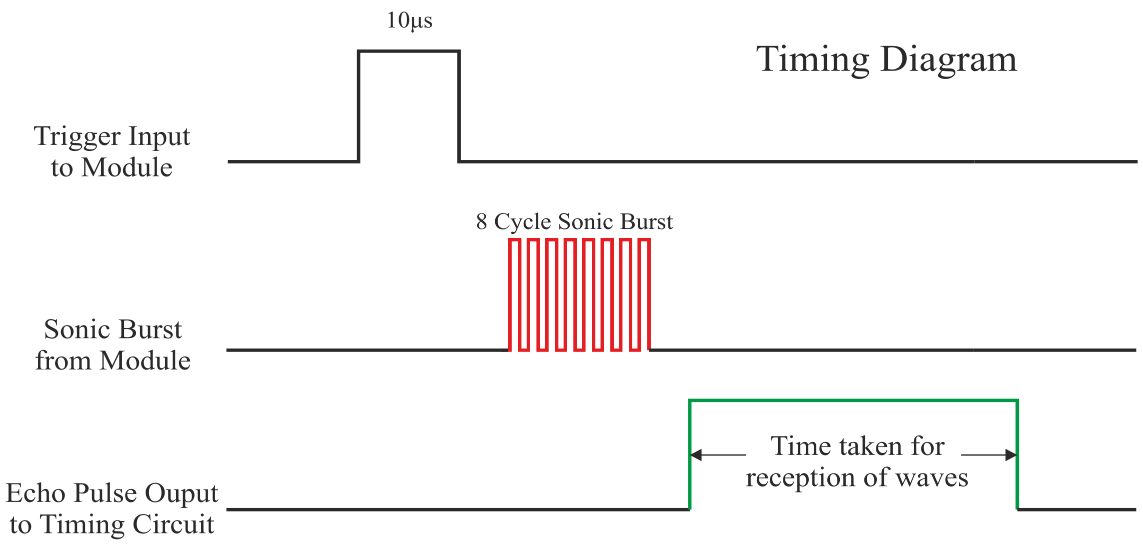

Figure 3.5: Timing Diagram for the Ultrasonic Sensor

3.2.2 Operation of Sonar

The timing diagram of HC-SR04 is shown in Fig. 3.5. To start measurement, initially the Trigger pin on SR04 must receive a HIGH pulse for at least 10μs. Upon triggering the sensor, it will transmit out 8 cycles of ultrasonic burst at 40kHz, after which it sets an Echo pin on the SR04 to a HIGH. When the sensor detects ultrasonic waves from receiver, the Echo pin is reset. The time interval between the HIGH and LOW of the Echo pin is measured. The distance of the obstacle is measured by,

$$D = S * T \tag{3.2.5}$$

where,

D	Distance of Obstacle
S	Speed of Sound
T	Time Interval between HIGH and LOW of Echo Pin

The software (on the microcontroller) polls for a reflection to be received by the sensor. However, there cannot be an indefinite wait for the reflections. Hence, a maximum wait interval is decided based on the maximum distance at which the obstacle is to be detected (35cm). If there is no reflection received by the sensor within this period, the obstacle is at a safe distance from that particular sensor(35cm or more). In such a case, a default value of 35cm is assumed for the sensor. (The reason for assigning a default such as this should be obvious on reading the obstacle avoidance strategy in chapter 5). It is important to note that polling for only a short time interval that corresponds to a distance such as 35 cm is warranted as the robot is only concerned with obstacles in its immediate vicinity.

3.3 Motor Actuation

As enough current cannot be sourced from the microcontroller for the operation of the DC motors, there is a need for a motor driver. The motor driver translates the actuation signals received from the microcontroller into drives for the motors with sufficient current. L293D, which uses the H-Bridge configuration, is the IC which performs this function. Fig. 3.6(a) shows the connection of the two DC motors to IC L293D.

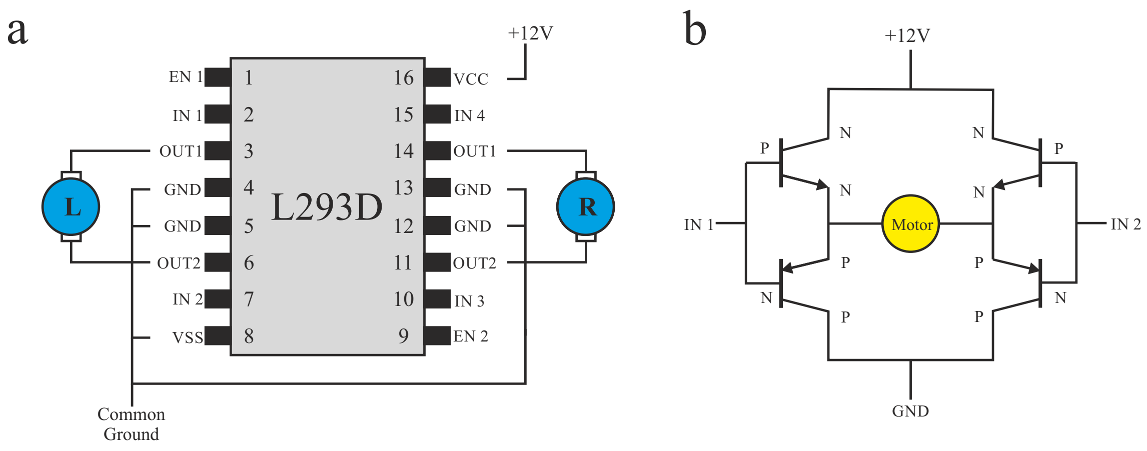

Figure 3.6: Motor Driver IC L293D (a) Connection to Motors (b) H-Bridge Configuration

3.3.1 Operation of H-Bridge (Motor Driver)

The H-Bridge is configured as shown in the Fig. 3.6(b). The NPN transistors switch on when a positive input is applied at their base terminal, while the PNP transistors switch on when zero or negative input is applied at the base.

When IN1 and IN2 inputs are supplied with +5V and 0V respectively from the microcontroller, the motor is supplied with +12V across it, and it moves in a particular direction. If IN1 and IN2 inputs are switched, the motor moves in the opposite direction. Assuming only rotation in one direction is required, if a pulse-width modulated

(PWM) signal is applied at IN1 and IN2 is maintained at 0V, the speed of the the DC motor varies proportional to the analog value of the PWM signal.

Chapter 4

Embedded System

The embedded system is responsible for reading the signals from various sensors, performing necessary computations, and providing the actuation signals to the motors. Hence, choosing an appropriate processor is crucial in system implementation.

4.1 Processor

The embedded processor chosen for the implementation of the project is the ATmega328 microcontroller from the megaAVR family by Atmel. It is an 8-bit RISC-based microcontroller with the following key features:

- Operating Voltage: 5V
- Input Voltage (recommended): 7-12V
- Input Voltage (limits): 6-20V
- Digital I/O Pins: 14 (of which 6 provide PWM output)
- Analog Input Pins: 6

- DC Current per I/O Pin: 40 mA
- DC Current for 3.3V Pin: 50 mA
- Flash Memory: 32 KB, of which 0.5 KB used by bootloader
- SRAM: 2 KB
- EEPROM: 1 KB
- Clock Speed: 16 MHz

4.2 Development board

A crucial factor in choosing the ATmega microcontroller is its compatibility with Arduino boards. Keeping in mind the time constraints of the project, the Arduino platform provided a great advantage of quick learnability and implementation. In addition, the platform provides the following advantages:

- A streamlined software-hardware platform for easy programming allowing the user to focus on algorithm implementation.
- Arduino is an open source platform. Even though the Arduino software environment is high level (Java-like), C code compatible with AVR can be added to Arduino programs.
- Arduino has in built libraries making interfacing with various sensors and actuators easy.
- The platform is well supported with a lot of good documentation.

- Arduino boards are quite inexpensive.

The Arduino Uno which is the basic Arduino board was chosen as it provided the required functionality. The Arduino Uno is a microcontroller board based on the ATmega328. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button.

4.2.1 Programming

Programs can be uploaded to the ATmega328 on the Arduino Uno using the Arduino IDE (Integrated Development Environment). The ATmega328 comes preloaded with a bootloader and there is no requirement for a external programmer. The computer running the IDE is connected to the board through a USB cable. It communicates using the original STK500 protocol. The bootloader can be bypassed and the microcontroller can be programmed through the ICSP (In-Circuit Serial Programming) header.

4.2.2 Power

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically.
The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended

range is 7 to 12 volts.

The power pins are as follows:

- VIN: The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). Voltage can be supplied through this pin, or, if supplying voltage via the power jack, accessed through this pin.
- 5V: This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7-12V), the USB connector (5V), or the VIN pin of the board (7-12V).
- 3V3: A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- GND: Ground pins.
- IOREF: This pin on the Arduino board provides the voltage reference with which the microcontroller operates.

4.2.3 Memory

The ATmega328 has 32 KB of Flash memory (with 0.5 KB used for the bootloader). It also has 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library).

4.2.4 I/O

Each of the 14 digital pins on the Uno can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kΩ. In addition, some pins have specialized functions:

- Serial pins 0 (RX) and 1 (TX): Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega16U2 USB-to-TTL Serial chip.
- External Interrupt pins 2 and 3: These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value.
- PWM pins 3, 5, 6, 9, 10, and 11: Provide 8-bit PWM output with the analogWrite() function.
- SPI pins 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK): These pins support SPI communication using the SPI library.
- LED pin 13: There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

The Uno has 6 analog inputs, labeled A0 through A5, each of which provide 10 bits of resolution (i.e. 1024 different values) and can be read using the analogRead() function. By default, they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analogReference() function. Additionally, some pins have specialized functionality:

- TWI: A4 or SDA pin and A5 or SCL pin. Support TWI communication using the Wire library.

There are a couple of other pins on the board:

- AREF: Reference voltage for the analog inputs.
- Reset: Bringing this line LOW resets the microcontroller.

4.2.5 Communication

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual COM port to software on the computer. The '16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, a .inf file is required. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1). A SoftwareSerial library allows for serial communication on any of the Uno's digital pins.

The ATmega328 also supports I2C and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus and an SPI library for SPI communication.

4.3 Pin mapping and pin allocation

Mapping of the ATmega328 microcontroller pins to the pins on the Arduino Uno board is shown in Table 4.1 and allocation of the board pins to various components of the robot is shown in Table 4.2.

Pin number	Arduino Pin	Pin number	Arduino Pin
1	Reset Pin	15	Digital Pin 9
2	Digital Pin 0	16	Digital Pin 10
3	Digital Pin 1	17	Digital Pin 11
4	Digital Pin 2	18	Digital Pin 12
5	Digital Pin 3	19	Digital Pin 13
6	Digital Pin 4	20	AVcc
7	Vcc	21	Aref
8	GND	22	GND
9	X1	23	Analog in 0
10	X2	24	Analog in 1
11	Digital Pin 5	25	Analog in 2
12	Digital Pin 6	26	Analog in 3
13	Digital Pin 7	27	Analog in 4
14	Digital Pin 8	28	Analog in 5

Table 4.1: Pin mapping of ATMega328 in Arduino UNO

Pin number	Function
D0	Unused
D1	Unused
D2	IR Sensor Right
D3	Ultrasonic Echo - Right 90
D4	Ultrasonic Echo - Right 60
D5	Ultrasonic Echo - Right 30
D6	Ultrasonic Echo - Middle
D7	Ultrasonic Echo - Left 30
D8	Ultrasonic Echo - Left 60
D9 + PWM	Right Motor
D10 + PWM	Left Motor
D11 + PWM	Ultrasonic Sensor - Left 90
D12	IR Sensor Left
D13	Unused
A0	Trigger for Ultrasonic sensor
A1	Unused
A2	Unused
A3	Unused
A4	Unused
A5	Unused

(D - Digital Pin, A - Analog Pin)

*A0 is used as a digital pin

Table 4.2: Pin Allocation in the Robotic System

Chapter 5

Control Design

The Control Design explains the behavior based approach and the different ‘behaviors’ that are involved in navigating the autonomous robot in the unknown environment. A ‘behavior’ is a control solution for a particular situation, that is executed depending on the circumstances. There are three behaviors, namely, ‘Go to Goal’, ‘Obstacle Avoidance’, and ‘Follow Wall’.

The switching between these behaviors plays a vital role in the navigation of the autonomous robot and is described under the section ‘Supervisor’.

5.1 Go to Goal

Go to Goal is fundamental behavior, which directs the robot to navigate to a specified co-ordinate under a ‘No Obstacle’ situation.

Let the floor be the co-ordinate plane, (x_1, y_1) the desired destination, (x,y) the current position of the robot and ϕ_{bot}, its orientation with respect to the world frame x-axis. Here, the initial position of the

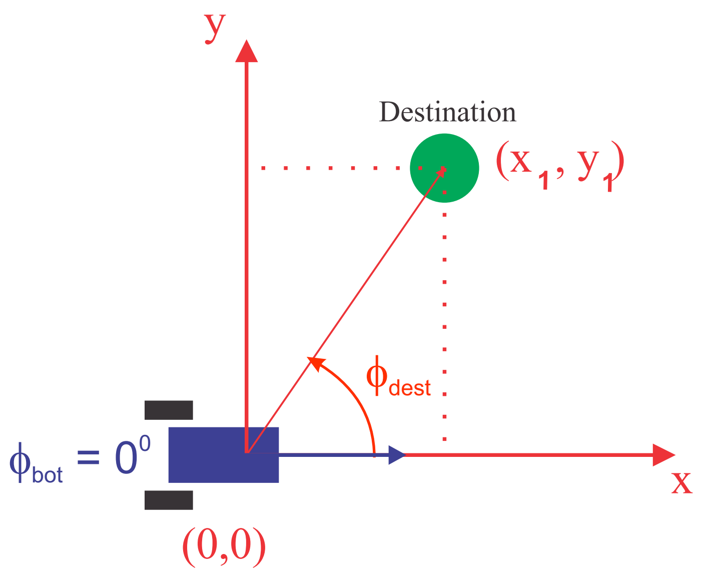

Figure 5.1: Go To Goal and the Co-Ordinate Frame

robot is considered to be the origin and the direction it faces is considered to be along the x axis as shown in the Fig. 5.1. Let the angle between the line joining the robot and destination and the x axis be ϕ_{dest}. The robot computes velocity to be given to each wheel as per the equations:

$$\mathrm{V}_R = \frac{2 \times \mathrm{V} + PID(\phi_{error}) \times \mathrm{L}}{2R} \tag{5.1.1}$$

$$\mathrm{V}_L = \frac{2 \times \mathrm{V} - PID(\phi_{error}) \times \mathrm{L}}{2R} \tag{5.1.2}$$

Where,

V	Average Velocity of the two wheels (kept a constant)
ϕ_{error}	Angle between the current pose of the robot and the destination
L	Distance between the wheels of the robot
R	Radius of the wheel

ϕ_{error}, which is the error is given by,

$$\phi_{error} = \phi_{dest} - \phi_{bot} \tag{5.1.3}$$

A PID (Proportional, Integral & Derivative) algorithm is used to drive the ϕ_{error} to zero.

As the robot navigates towards the destination, after a fixed small time interval, its new position and orientation are computed by:

$$x_{new} = x_{old} + \mathrm{D}_{bot} \times \cos(\phi_{bot}) \tag{5.1.4}$$

$$y_{new} = y_{old} + \mathrm{D}_{bot} \times \sin(\phi_{bot}) \tag{5.1.5}$$

$$\phi_{bot} = \phi_{bot} + \frac{D_R - D_L}{L} \tag{5.1.6}$$

Where,

D_{bot}	Distance travelled by the robot
ϕ_{bot}	Orientation of the robot
D_R	Distance travelled by the right wheel
D_L	Distance travelled by the left wheel

The robot continues to update its position and velocity dynamically, continuously, till it reaches the destination. However, owing to errors in wheel odometry, there is a small tolerance provided for the final position of the robot with respect to the destination.

5.2 Obstacle Avoidance

Obstacle Avoidance is a behavior that the robot executes to maneuver around obstacles in it's path to reach the goal location.

5.2.1 Sensing the environment

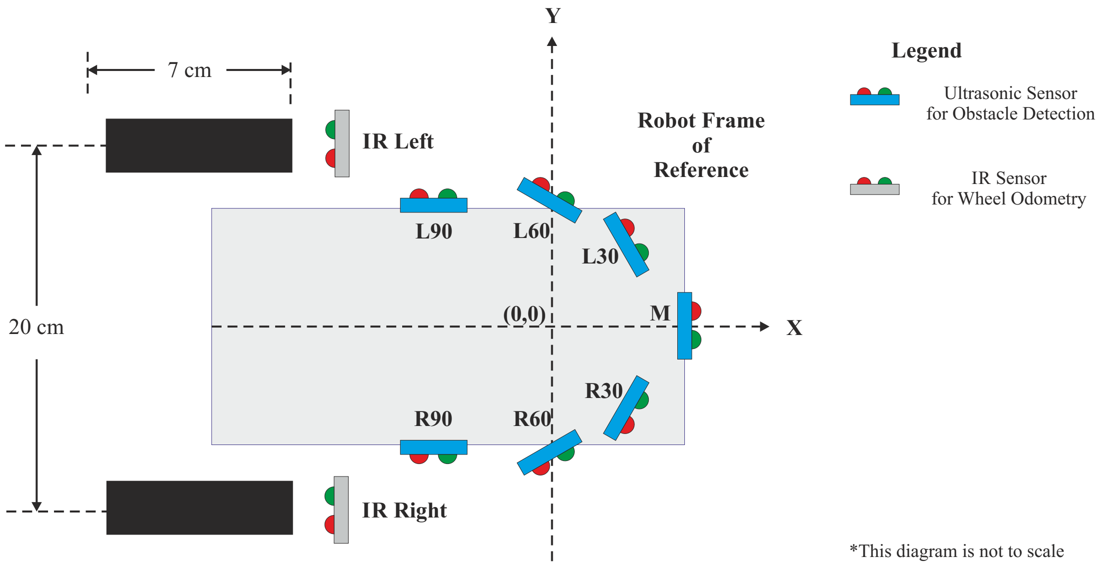
Y
7 cm
IR Left
Robot Frame of Reference
Legend
Ultrasonic Sensor for Obstacle Detection
IR Sensor for Wheel Odometry
L90
L60
L30
20 cm
(0,0)
M
X
R30
R90
R60
IR Right
*This diagram is not to scale

Figure 5.2: Illustration of the Robot Frame and the Sensors

Ultrasonic Sensor	Orientation (ϕ_s)	Co-ordinates (x_s, y_s)
L90	$+90°$	$(-7.5, +7.5)$
L60	$+60°$	$(+0.0, +8.5)$
L30	$+30°$	$(+3.5, +6.0)$
M	$0°$	$(+5.0, +0.0)$
R30	$-30°$	$(+3.5, -6.0)$
R60	$-60°$	$(+0.0, -8.5)$
R90	$-90°$	$(-7.5, -7.5)$

*All co-ordinates in centimeters

Table 5.1: Location of ultrasonic sensors on the robot

The Fig. 5.2 gives a clear illustration of the arrangement of various sensors on the robot. Seven ultrasonic sensors are placed on the robot as shown. These are arranged such that the beam widths of the sensors overlap in a manner that tries to avoid any blind spot.

The robot itself has a frame of reference. Ultrasonic sensors are placed at orientations and co-ordinates as dictated by Table 5.1.

Also, the center of the robot or the origin of the robot frame co-ordinate system is chosen carefully. The reason as to why the above arrangement was chosen will be clearer after the discussion of Obstacle Avoidance behavior.

The design of the robot is such that the center of the robot (origin of the robot frame co-ordinate system) is steered around the robot's environment. To obtain the co-ordinates of the obstacles detected as seen by the center of the robot, the values obtained by the ultrasonic sensors are transformed to the center of the robot.

5.2.2 Obstacle Avoidance Strategy

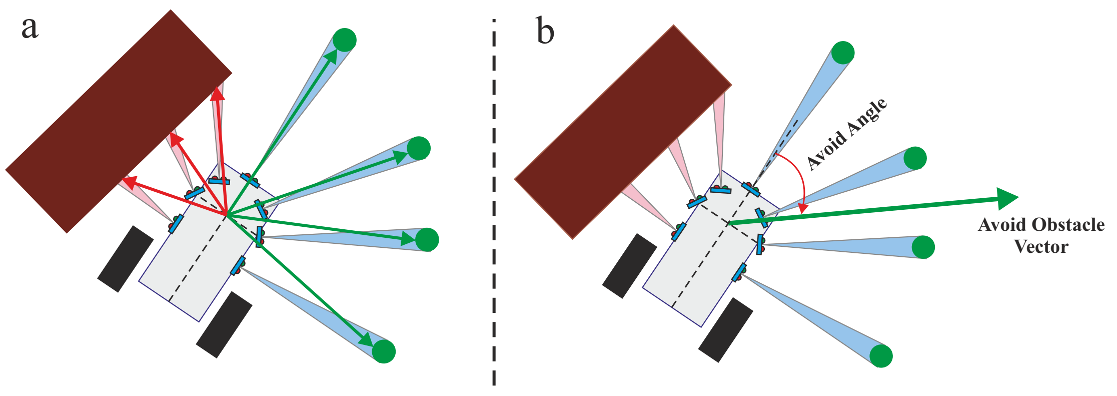

Figure 5.3: Obstacle Avoidance: (a) Computation of vector to each sensor point (b) Computation of Avoid Obstacle Vector

The steps to perform obstacle avoidance are:

1. Transform the ultrasonic distance measured by each sensor to a point in the reference frame of the robot.

   A point P_{s_i} that is measured to be d_{s_i} meters away by sensor i can be written as the vector (co-ordinate) $v_{s_i}= \begin{bmatrix} d_{s_i} \\ 0 \end{bmatrix}$ in the reference frame of sensor i.

We first need to transform this point to be in the reference frame of the robot. To do this transformation, we need to use the pose (location and orientation) of the sensor in the reference frame of the robot: $(x_{s_i}, y_{s_i}, \phi_{s_i})$. The transformation is defined as:

$$\begin{bmatrix} x_{r_i} \\ y_{r_i} \\ 1 \end{bmatrix} = R(x_{s_i}, y_{s_i}, \phi_{s_i}) * \begin{bmatrix} d_{s_i} \\ 0 \\ 1 \end{bmatrix} \tag{5.2.7}$$

where R is known as the transformation matrix that applies a translation by (x_{s_i}, y_{s_i}) and a rotation by ϕ_{s_i},

$$R(x_{s_i}, y_{s_i}, \phi_{s_i}) = \begin{bmatrix} \cos(\phi_{s_i}) & -\sin(\phi_{s_i}) & x_{s_i} \\ \sin(\phi_{s_i}) & \cos(\phi_{s_i}) & y_{s_i} \\ 0 & 0 & 1 \end{bmatrix} \tag{5.2.8}$$

2. Compute a vector to each sensor point from the robot:

   $u_1, u_2, u_3, u_4, u_5, u_6, u_7$.

   Use the point's co-ordinate for this computation. This is illustrated in the Fig. 5.3(a).

3. Weigh each vector according to their importance,

   $\alpha_1 u_1, \alpha_2 u_2, \alpha_3 u_3, \alpha_4 u_4, \alpha_5 u_5, \alpha_6 u_6, \alpha_7 u_7$.

   Here, the front and side sensors are typically more important for obstacle avoidance while moving forward.

4. Sum the weighted vectors to form a single vector:

$$u_{AO} = \alpha_1 u_1 + \alpha_2 u_2 + \alpha_3 u_3 + \alpha_4 u_4 + \alpha_5 u_5 + \alpha_6 u_6 + \alpha_7 u_7 \qquad (5.2.9)$$

This vector is called the Avoid Obstacle Vector, it makes an angle Avoid Angle with the robot's frame of reference as shown in Fig. 5.3(b). The robot is steered along this vector to move it away from obstacles. The center of the robot or the origin is chosen such that, the above equation yields a sufficient Avoid Angle.

5.3 Follow Wall

The Follow Wall behavior is the one that the robot executes to maneuver around concave obstacles. The robot follows the contour of the obstacle that is in its path as it moves towards the destination. Few examples of convex and concave obstacles are shown in Fig. 5.4.

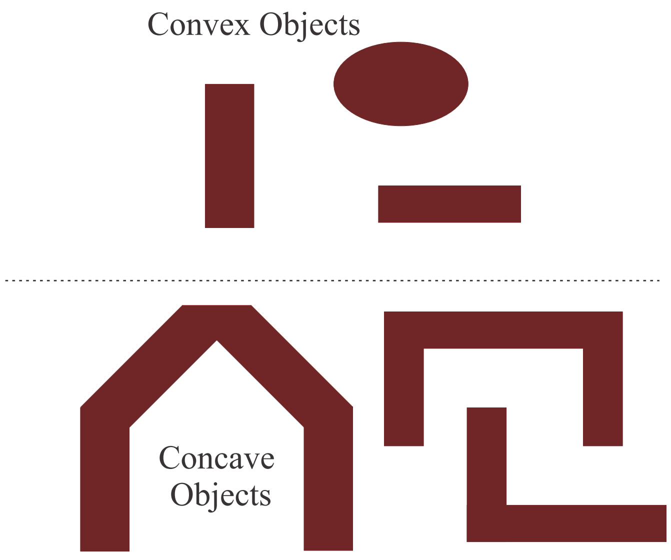

Figure 5.4: Types of Obstacles

5.3.1 Failure of Obstacle Avoidance behavior

The Obstacle Avoidance and Go To Goal behaviors alone are insufficient to make a robust autonomous navigation system. Fig. 5.5 illustrates a case where there is a necessity of a new behavior.

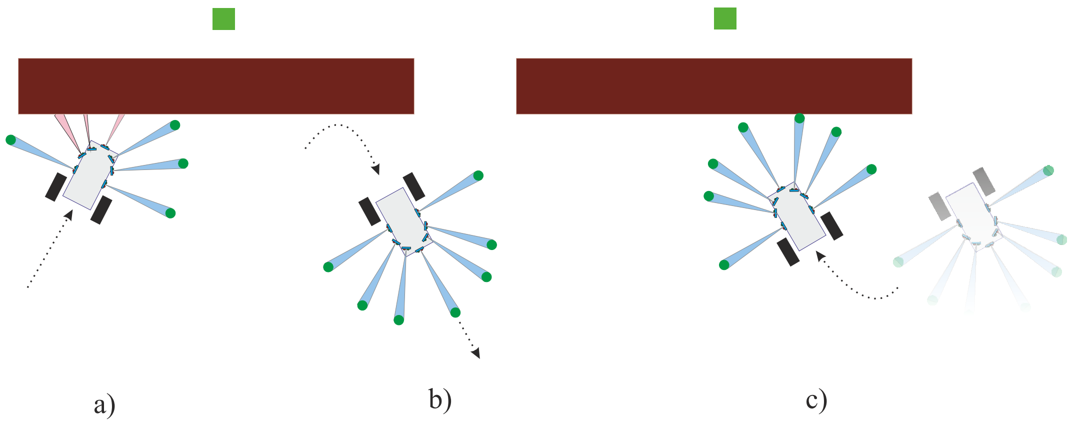

Figure 5.5: Failure of Obstacle Avoidance: (a) Robot detects and starts to avoid obstacle (b) Avoids obstacle (c) Tries to go back to goal, detects the obstacle again, the process repeats

5.3.2 Follow Wall Strategy

This concept primarily involves:

- Estimate of the wall which gives the Tangential vector
- Calculation of the perpendicular distance from the robot to the wall which results in the Perpendicular vector

The Tangential Vector

The Follow Wall concept uses two vectors to estimate the surface of the obstacle. These vectors are the sensor readings converted to robot frame. When at least two sensors are sensing the wall, we can obtain an approximation of the surface orientation as shown in Fig. 5.6(b). This is called the tangential component of the follow wall vector as this gives us the direction that the robot should move in to be parallel to the wall.

The direction of follow wall, i.e, whether the robot is going to follow wall to its left or right (Follow wall left or right) is determined by assessing the sign of the Avoid Angle. If the Avoid Angle is positive, follow wall left is the behavior used else it is the follow wall right behavior that is used. This initial step is important as we now deal with the sensors on the respective sides alone to determine the various angles and calculate the follow wall vector.

After deciding which side the robot is going to follow wall, the crucial step here is identifying which two vectors will be used to determine the tangential vector. There are two cases that arise when the robot is already in follow wall behavior,

- Only one sensor senses the wall
- At least two sensors are sensing the wall

Let $\overrightarrow{p1}$ and $\overrightarrow{p2}$ be the two vectors.

The tangential vector is calculated as:

$$\vec{T} = \vec{p}2 - \vec{p}1 \tag{5.3.10}$$

The unit vector along the tangential direction is,

$$\widehat{t} = \frac{\overrightarrow{T}}{\left|\overrightarrow{T}\right|} \tag{5.3.11}$$

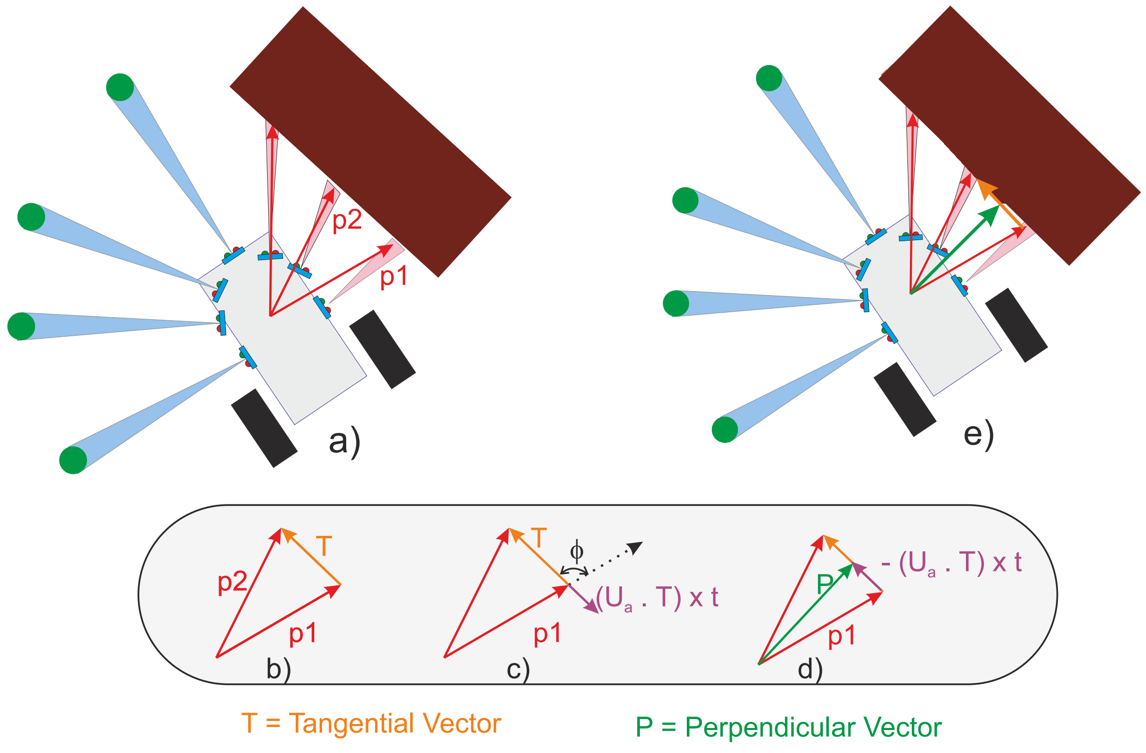

Figure 5.6: Calculating the Tangential and Perpendicular Vectors

$\overrightarrow{T}$ should always point in the direction the robot should move, to continue following the wall. As a result, care should be taken that $\overrightarrow{p2}$ should always be the vector that is closest to the middle of the robot. Only then, will $\overrightarrow{p2}$ - $\overrightarrow{p1}$ produce a resultant that is consistent with the direction the robot in which it is already moving.

If only one sensor is sensing, then that sensor is assigned $\overrightarrow{p2}$ and the sensor that is just before it, i.e, the sensor towards the 90° sensors is assigned $\overrightarrow{p1}$.

When at least two are sensing, the two sensors that are sensing the wall at the closest are obtained and then are assigned $\overrightarrow{p2}$ and $\overrightarrow{p1}$ in the same fashion as before wherein $\overrightarrow{p2}$ is the vector that is closest to the middle sensor.

The Perpendicular Vector

The other component of the Follow Wall vector is the perpendicular component. This component is responsible for maintaining the specified perpendicular distance from the wall. If the robot gets too close to the wall, this component is responsible for pushing the robot away from the wall and vice-versa.
Let $\overrightarrow{U_a} = \overrightarrow{p1}$, then the perpendicular vector $\overrightarrow{P}$ is,

$$\overrightarrow{P} = \overrightarrow{U}_a - (\overrightarrow{U}_a.\overrightarrow{T}) \times \widehat{t} \tag{5.3.12}$$

The unit vector in the perpendicular direction is,

$$\widehat{p} = \frac{\overrightarrow{P}}{\left|\overrightarrow{P}\right|} \tag{5.3.13}$$

In Fig. 5.6, those sensors that are reading the wall the closest are assigned vectors $\overrightarrow{p2}$ and $\overrightarrow{p1}$. The sensor that is closer to the middle is assigned $\overrightarrow{p2}$.

- First, the tangential vector, $\overrightarrow{T}$ is calculated as shown in Fig. 5.6(b).
- Then the dot product is calculated between $\overrightarrow{p1}$ and $\overrightarrow{T}$. The angle ϕ between the vectors is larger than 90°, cosine of the angle is negative. Resultant vector, $(\overrightarrow{U_a}.\overrightarrow{T}) \times \widehat{t}$, is shown in Fig. 5.6(c).
- The Perpendicular vector is,
  $\overrightarrow{P} = \overrightarrow{U_a} - (\overrightarrow{U_a}.\overrightarrow{T}) \times \widehat{t} = \overrightarrow{U_a} + [-(\overrightarrow{U_a}.\overrightarrow{T}) \times \widehat{t}]$
  as shown in Fig. 5.6(d).

Now let there be a vector Follow Wall Distance, $\overrightarrow{FWD}$, that is responsible for maintaining a user defined distance 'D' from the wall, then Follow Wall Distance vector, $\overrightarrow{FWD}$ is,

$$\overrightarrow{FWD} = \overrightarrow{P} - D \times \widehat{p} \tag{5.3.14}$$

If $|\overrightarrow{P}|$ >D then $\overrightarrow{FWD}$ points towards the wall, else it points away from the wall. As illustrated in Fig. 5.7(a), when the robot got too close to the wall, $\overrightarrow{FWD}$ points away from the wall.

The Follow Wall Vector

Once the Tangential and the Follow Wall Distance vectors are obtained, their vector sum produces the Follow Wall vector , $\overrightarrow{FW}$,

$$\overrightarrow{FW} = (W_T \times \overrightarrow{T}) + W_D \times (\overrightarrow{FWD}) \tag{5.3.15}$$

Here, W_D and W_T are the weights that can be assigned to the respective vectors to increase or decrease their contribution to the Follow Wall vector. The calculation of the Follow Wall Vector is illustrated in Fig. 5.7. Fig. 5.7(b) and Fig. 5.7(c) show the Follow Wall Vector being calculated for different weights of the Tangential and Perpendicular Vectors.

Once the Follow wall vector is calculated, the angle it is at with respect to the robot frame is calculated and this angle is the drive to the robot.

$$\overrightarrow{FW} = X\widehat{i} + Y\widehat{j} \tag{5.3.16}$$

$$\angle FW = \tan^{-1}\frac{Y}{X} \tag{5.3.17}$$

Here, $\hat{\imath}$ and $\hat{\jmath}$ are the unit vectors along the robots x and y axes.

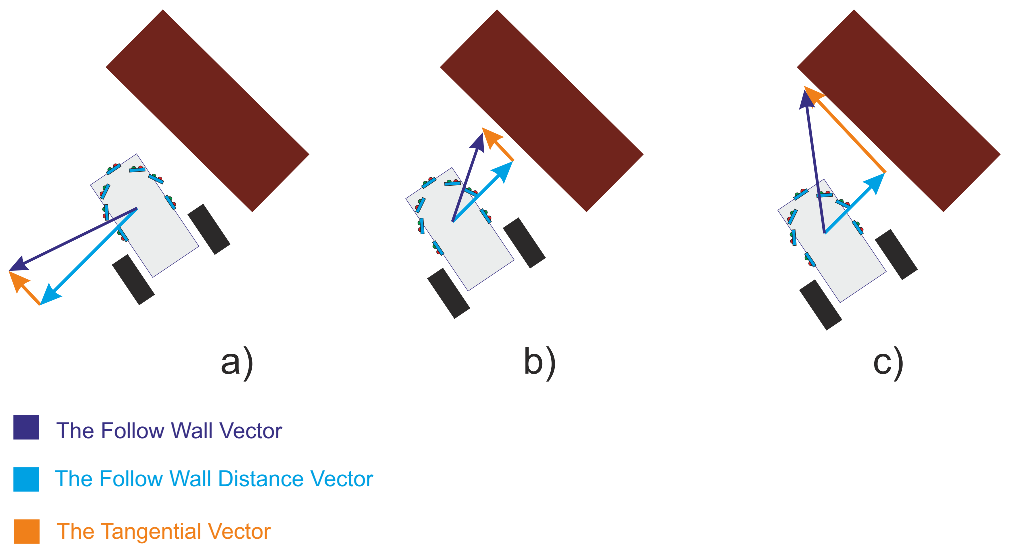

Figure 5.7: The Follow Wall Vector: (a) The robot is too close to the wall (b) & (c) Different weights assigned to the Tangential and Follow Wall Distance vectors

5.4 Supervisor

The robot is required to switch between different behaviors depending upon the situation. When there is no obstacle in the robot's vicinity, it executes the Go to Goal behavior. On sensing an obstacle, the angle between the Avoid Obstacle vector and the Go to Goal vector is used to decide between Follow Wall and simple Obstacle Avoidance. If the angle is greater than a defined threshold (experimentally found to be around 45°), it means that the obstacle is severely impeding the path of the robot to the destination. Hence, the robot employs the Follow Wall behavior to get around the obstacle. If the robot is sensing an obstacle but the angle between the Avoid Obstacle vector and the Go to Goal vector is small, this implies that there is an obstacle in the vicinity but it doesn't severely impede progress toward the destination. In such a case, the robot employs simple Obstacle Avoidance to ensure it doesn't crash into the obstacle.

Chapter 6

Cost Estimate

The components used and their prices are listed in Table 6.1

Component	Quantity	Price (₹)	Cost (₹)
Arduino Uno	1	1450	1450
Battery (12V, Lead Acid)	1	450	450
Battery charger	1	150	150
Chassis	1	120	120
Caster wheel	1	25	25
General purpose PCB	1	30	30
Infrared sensors	2	35	70
Jumpers	50	2	100
Misc. (screws, nuts, washers, screw driver, spanner)	1	500	500
Motors (200RPM)	2	200	400
Motor driver board	1	180	180
Multimeter	1	100	100
Votage regulators (6V)	1	20	20
Soldering iron with lead	1	180	180
Ultrasonic sensors	7	180	1260
Wires	5	4	20
Wheels	2	30	60
Total			5120

Table 6.1: Cost of components

Chapter 7

Conclusion

The differentially driven robot was modelled on the unicycle dynamics. Sonar and wheel odometry were respectively used for perception (obstacle estimation) and localization (estimation of robot pose, with pose referring to position and orientation). Three behaviors (controllers), namely, Go to Goal, Avoid Obstacle and Follow Wall were designed for the navigation problem. A Supervisor to appropriately switch between behaviors (based on robot pose and the perceived environment) was implemented. A PID regulator was used for driving the robot in the desired direction as it executed any one of the behaviors.

The robot was able to navigate around both convex and concave obstacles and reach the desired destination. However, wheel odometry alone is not sufficient for robust localization. Hence, due to errors in pose (orientation and position) estimation, the robot can consider itself to have reached the destination when infact there is an offset with respect to the desired destination. These errors can accumulate as the distance travelled by the robot and the number of turns executed increases. It is important to note that pose information is critical for the Supervisor to decide which behavior the robot should employ. If the error in pose estimation becomes too high, it can severely affect robot navigation.

7.1 Future scope

The wheel encoders used on the robot are inexpensive but quite crude. Using commercially available wheel encoders can improve the performance of the robot. However, wheel odometry is inherently not good enough for robust localization. Hence, future work can focus on improving the perception ability of the robot, for instance by using visual odometry. Mapping of the environment or perhaps simultaneous localization and mapping (SLAM) are other capabilities that can be added to the robot. The control principles learned in the implementation of this project apply to car-like robots as well as certain aerial and underwater robots. Robot navigation is also fundamental for implementation of robot swarms. Hence, this project has laid the foundation for working with such systems in future, which have wide ranging applications.

Bibliography

[1] Magnus Egerstedt, *Control of Mobile Robots*, Georgia Institute of Technology, Coursera.

[2] Nageswara S. V. Rao and S. S. Iyengar, *Autonomous Robot Navigation in Unknown Terrains: Incidental Learning and Environmental Exploration*, Transactions on Systems, Man, and Cybernetics, Vol. 20, No. 6, 1990.

[3] Alessandro De Luca, Giuseppe Oriolo, Marilena Vendittelli, *Control of Wheeled Mobile Robots: An Experimental Overview*, Dipartimento di Informatica e Sistemistica, Universita degli Studi di Roma“La Sapienza”, Italy.

[4] Arduino: http://arduino.cc/en/Main/arduinoBoardUno

Data Sheets