Sudan University of science and technology
Faculty of Engineering

School of Electronics Engineering

Implementation of an autonomous robot mapping and navigation system using SLAM and depth camera

A Research Submitted In Partial fulfillment for the Requirements of the Degree of B.Sc. (Honors) in Electronics Engineering

Prepared by:

Mohammed Ahmed Mohammed Alshambati

Waheeb Abubaker Ishaq

Supervised:

Dr. Khalifa Altayeb

March 2022

بسم الله الرحمن الرحيم

۞ وَلَوْ بَسَطَ ٱللَّهُ ٱلرِّزْقَ لِعِبَادِهِۦ لَبَغَوْا۟ فِى ٱلْأَرْضِ وَلَـٰكِن يُنَزِّلُ بِقَدَرٍ مَّا يَشَآءُ ۚ إِنَّهُۥ بِعِبَادِهِۦ خَبِيرٌۢ بَصِيرٌ ۞

(الشورى - 27)

﴿ هُوَ الَّذِي بَعَثَ فِي الْأُمِّيِّينَ رَسُولًا مِنْهُمْ يَتْلُو عَلَيْهِمْ آيَاتِهِ وَيُزَكِّيهِمْ وَيُعَلِّمُهُمُ الْكِتَابَ وَالْحِكْمَةَ وَإِنْ كَانُوا مِنْ قَبْلُ لَفِي ضَلَالٍ مُبِينٍ ﴾

(الجمعة - 2)

ACKNOWLEDGEMENT

First and foremost, I am grateful to God for the good health and wellbeing that were necessary to complete this research ,then I have to thank my research supervisor, Dr.khalifa Altayeb . Without his assistance and dedicated involvement in every step throughout the process, this paper would have never been accomplished. I would like to thank you very much for your support and understanding over the past years

I would also like to show gratitude to my teachers , including Dr.Hisham Ahmed, and Dr.Mayada Abdelgader. their teaching style and enthusiasm for the topic made a strong impression on me and I have always carried positive memories of their classes with me

Most importantly, none of this could have happened without my family. my father , who offered her encouragement through phone calls every week – despite my own limited devotion to correspondence., all my family has been kind and supportive to me over the last several years. To my parents and my sisters. Every time I was ready to quit, you did not let me and I am forever grateful. This dissertation stands as a testament to your unconditional love and encouragement.

ABSTRACT

This study will implement SLAM(Simultaneous localization and mapping) algorithms in small size autonomous vehicle using low cost sensors like depth camera , the study will use Microsoft Kinect sensor to map the environment and use that map to autonomously navigate in that environment . the results of mapping and navigation is then conducted against real world SLAM results using LIDAR sensors and simulations of the algorithms . the vehicle used in the study is two wheeled differential drive vehicle that uses raspberry pi as on board computer that run ROS (robot operating system)

ROS in our study is configured as a network between master robot (the vehicle) and slave robot (monitoring laptop) through LAN network to produce a map of the environment and navigate through that environment

An implementation and analysis of three open-source, Simultaneous Localization and Mapping (SLAM) techniques which are Fast-SLAM , Hector SLAM and RTABMAP using a Microsoft Kinect as a depth camera and a replacement the laser sensor is presented

السمتخلص

تطرقت هذه الدراسة لمفهوم رسم الخريطة و تحديد الموقع في الروبوتات ذاتية القيادة و شبه ذاتية القيادة اس ال أي ام, تم تحليل ثلاثة انواع من انوع الخوارزميات من هذا النوع و نفذت كل منها في بيئة داخلية لاختبار الاداء عند استخدام مستشعر كاميرا العمق كبديل لجهاز تحديد المسافة عن طريق الليزر لتقليل التكلفة

تم ربط اجزاء الروبوت و الخوارزميات عن طريق نظام تشغيل الروبوتات أر او اس في شبكة محلية بين خادم(الروبوت) و تابع (جهاز المراقبة) لتنفيذ الخوارزميات بإعدادات مخصصة لجسم الروبوت

في نهاية البحث تم مقارنة نتائج الخوارزميات المختلفة من ناحية الاداء و السرعة و تقديم توصيات لتحسيت الاداء زيادة دقة النتائج

TABLE OF CONTENTS

CHAPTER TITLE PAGE

الاستهلال I
ACKNOWLAGEMENT II
ABSTRACT III
المستخلص IV
LIST OF FIGURES VIII
LIST OF SYMBOLS XI
LIST OF APPENDICES XIII
CHAPTER ONE ..... **Error! Bookmark not defined.**
Introduction 1
1.1.Preface: ..... **Error! Bookmark not defined.**
1.2.Problem Statement: ..... **Error! Bookmark not defined.**
1.3.Research Objectives: ..... **Error! Bookmark not defined.**
1.4.Proposed solution: ..... **Error! Bookmark not defined.**
CHAPTER TWO ..... **Error! Bookmark not defined.**
Background and Literature Review ..... **Error! Bookmark not defined.**
2.1.Literature Review: ..... **Error! Bookmark not defined.**
2.2.Electrical sensors: ..... **Error! Bookmark not defined.**
2.2.1.Ultrasonic sensor: ..... **Error! Bookmark not defined.**
2.2.2.LIDAR sensor: ..... **Error! Bookmark not defined.**
2.2.3.IR sensor: ..... **Error! Bookmark not defined.**
2.2.4.Image sensor: ..... **Error! Bookmark not defined.**
2.2.5.Wheel encoder: ..... **Error! Bookmark not defined.**
2.2.6.Kinect sensor: ..... **Error! Bookmark not defined.**
2.2.7.Sensor’s sensitivity: ..... **Error! Bookmark not defined.**
2.3.Electric Motors: ..... **Error! Bookmark not defined.**
2.3.1.DC motors: ..... **Error! Bookmark not defined.**
2.3.2.The “Brushless” DC Motor: ..... **Error! Bookmark not defined.**
2.3.3.Servo motors: ..... **Error! Bookmark not defined.**
2.4.Darlington circuit: ..... **Error! Bookmark not defined.**
Base Activation Voltage: ..... **Error! Bookmark not defined.**
2.5. L298N H-bridge motor driver: ..... **Error! Bookmark not defined.**
2.5.the need for SLAM algorithms: ..... **Error! Bookmark not defined.**
2.5.1. methods for solving sensors limitations problem: ..... **Error! Bookmark not defined.**
CHAPTER THREE ..... **Error! Bookmark not defined.**
Methodology ..... **Error! Bookmark not defined.**
3.1.Methodology: ..... **Error! Bookmark not defined.**

3.1.1.Simultaneous localization and mapping.....**Error! Bookmark not defined.**
3.1.2.Introduction to SLAM.....**Error! Bookmark not defined.**
3.1.3 .EKF SLAM.....**Error! Bookmark not defined.**
3.1.4.FastSLAM.....**Error! Bookmark not defined.**
3.1.5. FastSLAM 2.0.....**Error! Bookmark not defined.**
3.1.6.GraphSLAM .....**Error! Bookmark not defined.**
3.1.7. Full SLAM.....**Error! Bookmark not defined.**
3.1.8.RTAB-Maping:.....**Error! Bookmark not defined.**
3.1.9. Our proposed method for SLAM algorithm...**Error! Bookmark not defined.**
3.2. Motion planning.....**Error! Bookmark not defined.**
3.2.1.Configuration space .....**Error! Bookmark not defined.**
3.2.2.Free space .....**Error! Bookmark not defined.**
3.2.3.Target space .....**Error! Bookmark not defined.**
3.2.4.Obstacle space.....**Error! Bookmark not defined.**
3.2.5.Algorithms .....**Error! Bookmark not defined.**
3.2.6.Grid-based search.....**Error! Bookmark not defined.**
3.2.7. A* Algorithm:.....**Error! Bookmark not defined.**
3.3. Robot Operating System .....**Error! Bookmark not defined.**
3.3.1. RVIZ .....**Error! Bookmark not defined.**
3.4. System block diagram.....**Error! Bookmark not defined.**
3.5. Robot model:.....**Error! Bookmark not defined.**
3.5.proposed system diagram.....**Error! Bookmark not defined.**
3.6.Robot module view:.....**Error! Bookmark not defined.**
3.6.1.Module excepected scenario: .....**Error! Bookmark not defined.**
CHAPTER FOUR.....**Error! Bookmark not defined.**5
Our Test Robot.....**Error! Bookmark not defined.**
4.1.ROS setup: .....**Error! Bookmark not defined.**
4.1.1 Transform Configuration (other transforms): .**Error! Bookmark not defined.**
4.1.2 Sensor Information (sensor sources) .....**Error! Bookmark not defined.**
4.1.3 Odometry Information (odometry source) .....**Error! Bookmark not defined.**
4.2 Base Controller (base controller) .....**Error! Bookmark not defined.**
4.2.1Mapping (map_server).....**Error! Bookmark not defined.**
4.3.Coding:.....**Error! Bookmark not defined.**
4.3.1.Creating a Package.....**Error! Bookmark not defined.**
4.3.2.Creating a Robot Configuration Launch File ..**Error! Bookmark not defined.**
4.3.3.motion planning: .....**Error! Bookmark not defined.**
4.4.FAST-SLAM package launch file: .....**Error! Bookmark not defined.**
4.4.2.FAST-SLAM flowchart:.....**Error! Bookmark not defined.**
4.4.3.FAST-SLAM rsults:.....**Error! Bookmark not defined.**
4.5.hector_SLAM launch file: .....**Error! Bookmark not defined.**
4.5.2:hector SLAM results:.....**Error! Bookmark not defined.**
4.5.3:hector SLAM results:.....**Error! Bookmark not defined.**
4.6.RTABMAP launch file: .....**Error! Bookmark not defined.**

4.6.2.RTABMAP flowchart: ..... **Error! Bookmark not defined.**
4.6.3.RTABMAP 3D map results: ..... **Error! Bookmark not defined.**
4.6.4.RTABMAP projection map results: ..... **Error! Bookmark not defined.**
CHAPTER FIVE ..... **Error! Bookmark not defined.**
Conclusion And Recommendation ..... **Error! Bookmark not defined.**
5.1. Conclusion: ..... **Error! Bookmark not defined.**
5.2.Recommendation: ..... **Error! Bookmark not defined.**
5.3.Reference: 80

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE
2.1	HC-SR04 ultrasonic sensor and its connecter pins	
2.2	LIDAR sensor working principle	
2.3	IR sensor	
2.4	example of optical wheel encoder	
2.5	Kinect sensor parts	
2.6	series and shunt connected moor	
2.7	2-pole permanent magnet motor , and 4- Pol wound field motor	
2.8	BLDC motor magnets and wounds	
2.9	The changing resultant flux continually pulls the rotor magnet, causing the rotor to turn.	
2.10	Sinusoidal Control	
2.11	PWM Output vs. Output Voltage	
2.12	servo motor control signals	
2.13	servo motor angle control signal	
2.14	Darlington Pair transistors	

2.15 L298N H-bridge motor driver pins

2.16 ultrasonic sensor detection range

2.17 Quantum efficiency range

3.1 landmarks locations and constrains

3.2 fastSLAM algorithm simulation

3.3 RTAB-MAP flowchart

3.4 A* algorithm

3.5 block diagram of proposed system (used components)

3.6 Visualization of Differential Drive Robot and Key Variables

3.7 proposed diagram of the software and hardware of the system

3.8 shows module representation of our test robot

3.9 circuit diagram proposed for master robot

4.1 ROS navigation stack components

4.2 FAST-SLAM flow chart

4.3 Fast-SLAM created map show the robot pose around its environment

4.4 hector SLAM flowchart

4.5 hector SLAM map

4.6 RTABMAP flowchart

4.7 RTAB-SLAM 3D map

4.8 RTAB-SLAM Projection map

LIST OF SYMBOLS

d	-	distance
T	-	time
C	-	speed of sound ~ 343 meters/second
Vcc	-	voltage terminal
I_T	-	electrical current
I_f	-	shunt field current
I_A	-	Armature
KG	-	kilogram
CM	-	centimeter
Vss	-	logic power supply
Vs	-	power supply
xt	-	position of vehicle
m	-	map
z1:t	-	vector of sensor's measurements
u1:t	-	vector of control signals of vehicle
ut	-	control input
δt	-	random Gaussian variable with zero mean
st	-	robot path
N	-	landmarks
μn	-	mean of particles
p	-	plane
SO	-	special orthogonal group of 2D rotations
c	-	the special Euclidean group
x	-	x plane
y	-	y plane

θ	-	rotation in z plane
α, β, γ	-	angles around x,y,z axis
C_{free}	-	obstecle free space
Vc	-	left wheel linear velocity
Vd	-	right wheel linear velocity
ω	-	angular velocity
R	-	distance between two wheel centers
θ	-	angle in x axis
ICC	-	instantaneous curvature center

LIST OF APPENDICES

APPENDICE	TITLE	PAGE
A	Arduino wheel controller code	
B	robot configuration ROS launch code	
C	FastSLAM ROS launch code	
D	RTABMAP ROS launch code	

CHAPTER ONE

Introduction

1.1.Preface:

Robotic devices are steadily becoming a significant part of our daily lives. Search-and-rescue robots, service robots, surgical robots, and autonomous cars are examples of robots most of us are familiar with.

Finding paths (motion planning) efficiently for such robots is critical for a number of real-world applications For example search-and-rescue settings, a small robot may need to find paths through rubble and semi-collapsed buildings to locate survivors. In domestic settings, it would be useful if a robot could, for example, put away kids' toys, fold the laundry, and load the dishwasher. Motion planning also plays an increasingly important role in robot-assisted surgery. For example, before a flexible needle is inserted or an incision is made, a path can be computed that minimizes the chances of harming vital organs. More generally, motion planning is the problem of finding a continuous path that connects a given start state of a robotic system to a given goal region for that system, such that the path satisfies a set of constraints (e.g., collision avoidance, bounded forces, bounded acceleration) . this study will not focus on motion planning process or any algorithm that implement motion planning , instead we will rely on OMPL (open source motion planning library) and MATLAB implementations to perform the motion planning process and we will discuss the criteria that an AI based robotic system uses to perform a certain task

using a group of robots .

Every autonomous system need to interact with the world around it and in order to be successful there are certain capabilities that the system needs to have, these capabilities can be divided into four main areas : sense, perceive, plan and act . sense refers to directly measuring the environment with sensors, it's the process of collecting information from the system and the external world , for a

self-driving car for example this sensor may include radar , lidar and visible

cameras , in this case simply gathering data with sensors isn't good enough because the system needs to be able to interpret the data and turn it into something that can be understood and acted on by the autonomous system this is the role of the perceive step to make use of the sensed data . in a typical situation the car has to ultimately interpret the blob of pixels as a road with lane lines and that there's something off to the side that could be any object or a person that standing at the road side, this level of understanding is critical in order for the system to determine what to do next , next comes the planning step where it figures out what it would like to do and finds a path to get there and lastly the system calculates the best actions that get the system to follow that path this last step is what the controller and the control system is doing

if we considered that a robot has to perform a certain task , moving around or grabbing an object from a table for instance, in order to perform this task, the robot has to get some information about its environment like the position of the table, other objects that would act like obstacles and also its current state during execution, it can perform this by using some kind of sensors a camera or an ultrasonic sensor for instance to detect positioning

and nearby objects . however in this case knowing only the position of the object isn't enough to perform this task . robot can still rely on the previously observed position, and update this position using odometry while it moves. Unfortunately, odometry will be especially unreliable during the movement because of the effect that environment make to the robot movement and the fact that mechanical parts don't work perfectly. There is, however, another solution: a map of the environment that contains the position of the robot , target object and the obstacles in the path to that object in order to compute the position of the robot relative to its environment and target objects and the movements required to accomplish that task .The above scenario illustrates a simple instance of the general approach that we suggest in this paper: to give a robot a enough vision of it's environment in order to perform tasks. In the above example, our robot needs a visual information to measure the relative position of objects in order to perform its task: it has the options to either compute this information using its own sensors, or to borrow this information from a second robot .More generally, we consider a group of autonomous robotic systems embedded in a common environment . Each robot in the group includes a number of functionalities: for instance, functionalities for image understanding, localization, planning, and so on; they may have different sensing, acting, and reasoning capacities, and some of them maybe as simple as a fixed camera monitoring the environment. The key point here is that each robot may also use functionalities from other robots in order to compensate for the ones that it is lacking, or to improve its own. this approach can be applied in mostly every field of application that can make use of robots to perform a task , in this research we focus on the how robots can use shared perception of the environment and positioning of objects and obstacles within that environment or simply use there own

detected and measured information to perform a task . also we view some approaches to use shared information in order to perform tasks.

1.2.Problem Statement:

For an autonomous robot to fully perform its required task , some kind of perception of it's environment and its location in that environment should be available to perform motion Planning and pre-calculate movements and their results , this required perception is the Localization and Mapping of the environment

In cases where robot used by remote control device , the localization and mapping is suitable for applications that require a map and locations of objects , example of these applications is Room service robot , mapping underground mines and hard-to-reach environments , robot vacuums and factories

1.3.Research Objectives:

To study the autonomous navigation problem in closed-door and small scale open-door environments for applications like search and rescue and home applications robotics along with when and how to change the approach that robot use to perform a task in response to changes in the environment or the task, and also a brief study for different SLAM algorithms that uses laser and cameras

1.4.Proposed solution:

The proposed solution include an implementation of SLAM in a small size vehicle that has depth camera , odometry sensors (wheel encoder) and ROS (Robot Operating System) that will act as a suitable environment to perform sensor fusion operation and SLAM algorithms

CHAPTER TWO

Background and Literature Review

CHAPTER TWO

Background and Literature Review

2.1.Literature Review:

Since the introduction of the first robot manipulators in 1960s, there has always been a demand for kinematic parameter identification and subsequent error correction operations in order to improve the ability of robot manipulators in reaching a specified pose consistently and accurately and therefore performing robot's requested task efficiently .The identification process provides some estimates of the parameters of the kinematic model ,sensors and actuators to be used for the analysis and motion control of a robot (Forward and Inverse Kinematics [1]) . Even if it is possible to dismantle a robot components to make a good model for motion planning of the robot , the resulting model will still contain some inaccuracies arising from joint and link compliances changing with the manipulator configurations, thermal effects, transducer errors and actuators inaccuracies **.** this is not always the case , in some configurations where the system is multi-functional and uses more than one robot to perform divergent tasks , the system can make use of different robots (or a robot part) to accomplish its required task . in case of using multiple sensors this process of composing is called sensor fusion[2],[3] if the sensors was from the same robot , in case of sensors from other robots and actuators this process is consider group of robots configured to perform the required task

Robert Lundh , Lars Karlsson and Alessandro Saffiotti in their paper "Plan-Based Configuration of a Group of Robots"[4] informally called configuration any way to allocate and connect the functionalities of a distributed multi-robot system. Often, the same task can be performed by using different configurations. For example, in the scenario proposed by the paper, Pippi can perform its door-crossing task by connecting its own door-crossing functionality to either (1)its own perception functionality, (2) a perception functionality borrowed from Emil, or (3) a perception functionality borrowed from a camera placed over the door. Having the possibility to use different configurations to perform the same task opens the way to improve the flexibility, reliability, and adaptivity of a society of robotic agents. Ideally, they would like to automatically select, at any given moment, the best available configuration, and to change it when the situation has changed. In this context, their

research created a method to automatically generate a configuration of a robot society for a given task

For the purpose of building a good idea of the mechanism of the robotic system that used to implement our approach we first need to explore working mechanism of various Transducer and motors (actuators) which will then be modeled when developing a prototype are shown in the next section.

2.2.Electrical sensors:

Sensor is a device, module, machine, or subsystem whose purpose is to detect events or changes in its environment and send the information to other electronics, A sensor is always used with other electronics. Sensors are used in everyday objects such as touch-sensitive elevator buttons (tactile sensor) and lamps which dim or brighten by touching the base, besides innumerable applications of which most people are never aware. With

advances in micro machinery and easy-to-use microcontroller platforms, the uses of sensors have expanded beyond the traditional fields of temperature, pressure or flow measurement, for example into MARG sensors[2]. Moreover, analog sensors such as potentiometers and force-sensing resistors are still widely used. Applications include manufacturing and machinery, airplanes and aerospace, cars, medicine, robotics and many other aspects of our day-to-day life. There are a wide range of other sensors, measuring chemical & physical properties of materials. A few examples include optical sensors for Refractive index measurement, vibrational sensors for fluid viscosity measurement and electro-chemical sensor for monitoring pH of fluids. here we will focus on ultrasonic,IR and Image sensors**.**

2.2.1.Ultrasonic sensor:

An ultrasonic sensor is an electronic device that measures the distance of a target object by emitting ultrasonic sound waves, and converts the reflected sound into an electrical signal. Ultrasonic waves travel faster than the speed of audible sound (i.e. the sound that humans can hear). Ultrasonic sensors are used primarily as proximity sensors. They can be found in automobile self-parking technology and anti-collision safety systems. Ultrasonic sensors are also used in robotic obstacle detection systems, as well as manufacturing technology. In comparison to infrared (IR) sensors in proximity sensing applications, ultrasonic sensors are not as susceptible to interference of smoke, gas, and other airborne particles (though the physical components are still affected by variables such as heat). Ultrasonic sensors are also used as level sensors to detect, monitor, and regulate liquid levels in closed containers (such as vats in chemical factories). Most notably, ultrasonic technology has enabled the medical industry to produce images of internal organs

Ultrasonic sensors have two main components: the transmitter (which emits the sound using piezoelectric crystals) and the receiver (which encounters the sound after it has travelled to and from the target).

In order to calculate the distance between the sensor and the object, the sensor measures the time it takes between the emission of the sound by the transmitter to its contact with the receiver. The formula for this calculation is D = ½ T x C (where D is the distance, T is the time, and C is the speed of sound ~ 343 meters/second).

A commonly used ultrasonic sensor is HC-SR04 which is a 4 pin module, whose pin names are Vcc, Trigger, Echo and Ground respectively. This sensor is a very popular sensor used in many applications where measuring distance or sensing objects are required. The module has two eyes like projects in the front which forms the Ultrasonic transmitter and Receiver. The sensor works with the simple high school formula that Distance = Speed × Time . The Ultrasonic transmitter transmits an ultrasonic wave, this wave travels in air and when it gets objected by any material it gets reflected back toward the sensor this reflected wave is observed by the Ultrasonic receiver module as shown in the picture below

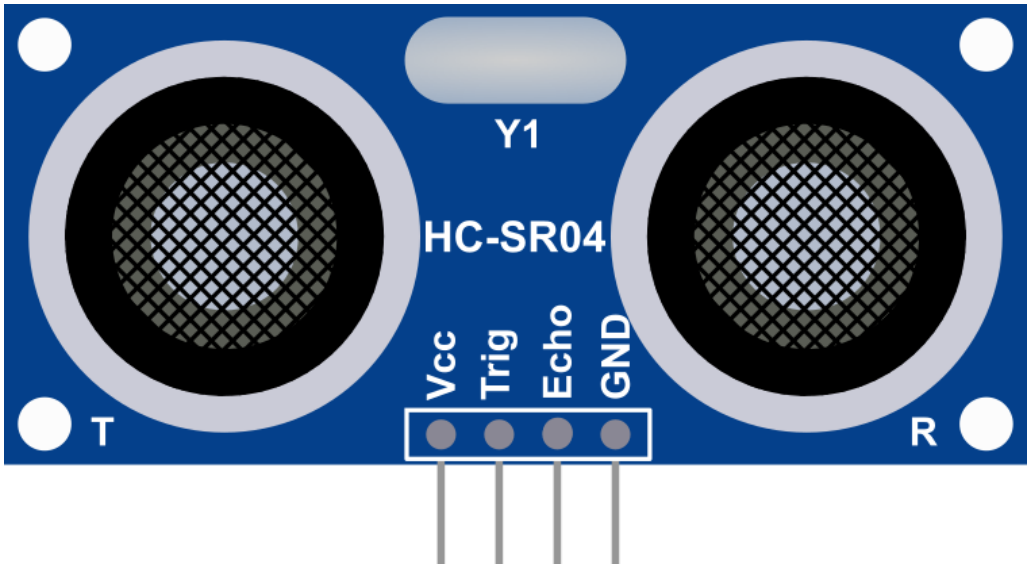


Fig 2.1:HC-SR04 ultrasonic sensor and its connecter pins

Now, to calculate the distance using the above formulae, we should know the Speed and time. Since we are using the Ultrasonic wave we know the universal speed of US wave at room conditions which is 330m/s. The circuitry inbuilt on the module will calculate the time taken for the US wave to come back and turns on the echo pin high for that same particular amount of time, this way we can also know the time taken. Now simply calculate the distance using a microcontroller or microprocessor.

2.2.1.1.How to use the HC-SR04 Ultrasonic Sensor:

HC-SR04 distance sensor is commonly used with both microcontroller and microprocessor platforms like Arduino, ARM, PIC, Raspberry Pie etc. The following guide is universally since it has to be followed irrespective of the type of computational device used. Power the Sensor using a regulated +5V through the Vcc ad Ground pins of the sensor. The current consumed by the sensor is less than 15mA and hence can be directly powered by the on

board 5V pins (If available). The Trigger and the Echo pins are both I/O pins and hence they can be connected to I/O pins of the microcontroller. To start the measurement, the trigger pin has to be made high for 10uS and then turned off. This action will trigger an ultrasonic wave at frequency of 40Hz from the transmitter and the receiver will wait for the wave to return. Once the wave is returned after it getting reflected by any object the Echo pin goes high for a particular amount of time which will be equal to the time taken for the wave to return back to the sensor. The amount of time during which the Echo pin stays high is measured by the MCU/MPU as it gives the information about the time taken for the wave to return back to the Sensor. Using this information the distance is measured as explained in the above heading. this sensors is used in many applications for example:

- Used to avoid and detect obstacles with robots like biped robot, obstacle avoider robot, path finding robot etc.
- Used to measure the distance within a wide range of 2cm to 400cm
- Can be used to map the objects surrounding the sensor by rotating it
- Depth of certain places like wells, pits etc. can be measured since the waves can penetrate through water

2.2.2.LIDAR sensor:

LIDAR is a method for determining ranges (variable distance) by targeting an object with a laser and measuring the time for the reflected light to return to the receiver. LIDAR can also be used to make digital 3D representations of areas on the earth's surface and ocean bottom, due to differences in laser return times, and by varying laser wavelengths. It has terrestrial, airborne, and mobile applications

Light detection and ranging (LIDAR) is a ranging method based on laser technology.

These units typically involve a laser with an oscillating mirror that

enables the unit to Conduct ranging in 2D space. 3D laser scanning can be achieved with a multi-axis unit. The operating principle is based on TOF: the unit emits a pulse of laser energy and then Measures the response using a photo detector. Once one position has been measured, the unit oscillates to the next position, and then performs another measurement. This process is repeated over the oscillating range of the unit allowing for a 2D measurement sweep to be recorded. LIDAR units can achieve up to 360 degree horizontal field of view with measurement rates up to 50Hz. Like radar, it is only useful in line of sight applications and cannot differentiate between objects and humans; however, it does have high accuracy and does not require an MWC . The basic operation principle is that a short pulse of light is emitted, reflected on an object, and then received. The time t_{wait} between the emitting and receiving can then be used to calculate the distance d to the object as

$$d = 1/2\ CT_{wait}$$

Where **c** is the speed of light

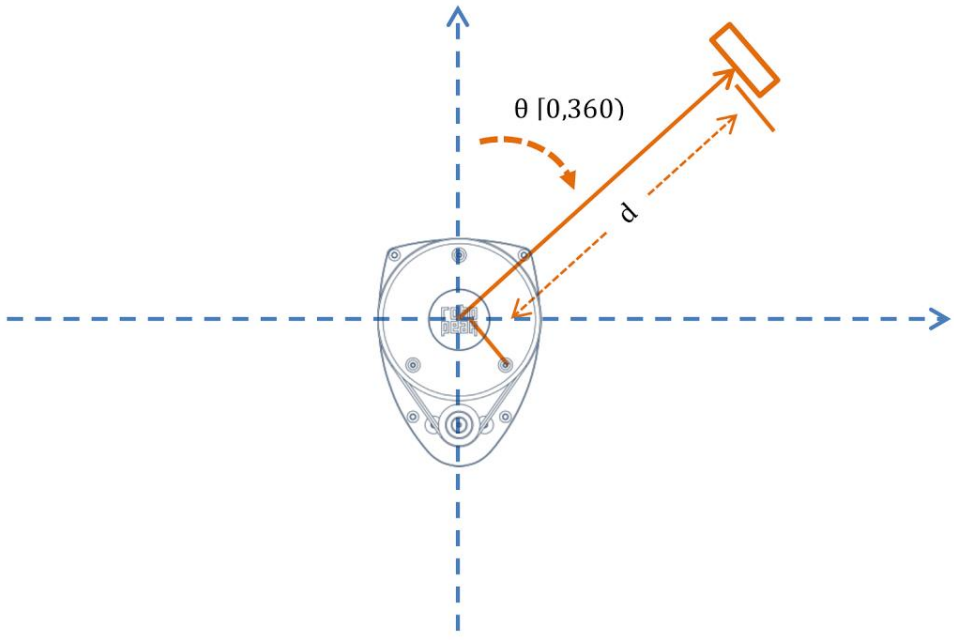


Fig 2.2: LIDAR sensor working principle

2.2.3.IR sensor:

An infrared (IR) sensor is an electronic device that measures and detects infrared radiation in its surrounding environment. Infrared radiation was accidentally discovered by an astronomer named William Hershel in 1800. While measuring the temperature of each color of light (separated by a prism), he noticed that the temperature just beyond the red light was highest. IR is invisible to the human eye, as its wavelength is longer than that of visible light (though it is still on the same electromagnetic spectrum). Anything that emits heat (everything that has a temperature above around five degrees Kelvin) gives off infrared radiation.

There are two types of infrared sensors: active and passive. Active infrared sensors both emit and detect infrared radiation. Active IR sensors have two parts: a light emitting diode (LED) and a receiver. When an object comes close to the sensor, the infrared light from the LED reflects off of the object and is detected by the receiver. Active IR sensors act as proximity sensors, and they are commonly used in obstacle detection systems robotic system in our case

Passive infrared (PIR) sensors only detect infrared radiation and do not emit it from an LED. Passive infrared sensors are comprised of:

- Two strips of pyro electric material (a pyro electric sensor)
- An infrared filter (that blocks out all other wavelengths of light)
- A Fresnel lens (which collects light from many angles into a single point)
- A housing unit (to protect the sensor from other environmental variables, such as humidity)

PIR sensors are most commonly used in motion-based detection, such as in-home security systems. When a moving object that generates infrared radiation enters the sensing range of the detector, the difference in IR levels between the two pyro electric elements is measured. The sensor

then sends an electronic signal to an embedded computer, which in turn triggers an alarm.

Infrared (IR) Sensor Module is a distance proximity sensor "switch". When there is an object or obstacles that are close enough to block the view in front of 2 LEDs, it triggers the infrared trans-receiver module. The clear LED is the IR emitter while the black LED is the IR receiver. It uses the electromagnetic reflection principle where when the reflective surface (object) is closer, the receiver will receive stronger signal from transmitter due to shorter distance traveled of reflected of wave.

When there is a object that is close enough, the IR electromagnetic detection received by the IR receiver is higher than the threshold level (user pre-set level), the sensor will change the output switch mode so that microprocessor board such as Arduino can execute what is going to do next. IR Sensor Module has only 1 main output signal which is Digital Output. Digital Output either go high (5V or 3.3V depends on the input voltage) or low (0V), thus this module cannot be used as a distance measurement but just as a trigger switch. When there is no obstacles or object within the detection distance, the output is at HIGH position (5V or 3.3V). When the distance shorter than or equal to the threshold set, the output signal will change to position LOW (0V). The distance threshold can be set by adjusting the potentiometer / trim-pot on the board. This sensor module only able to detect distance between 2cm and 30cm within the view of the IR LED and Photo-resistor. The trigger distance is somehow very subjective to object's surface material, color and shape. Practically I would recommend this sensor switch for application less than 10cm distance. Thus this module is suitable for very close range of detection such as obstacle avoidance and virtual touch switch application.

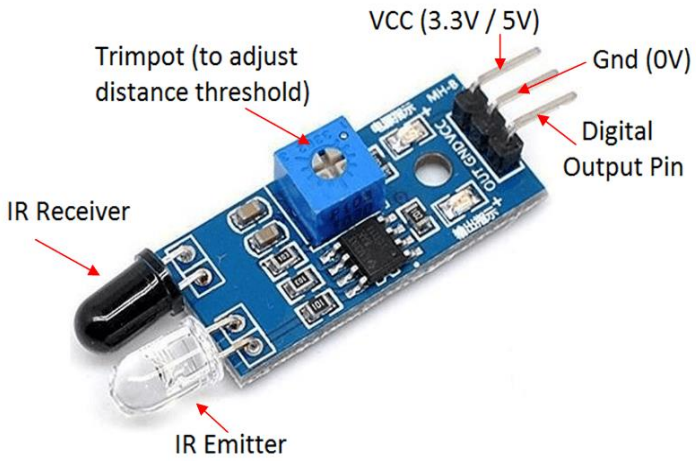


Fig 2.3:IR sensor

Arduino has the ability to detect voltage value as a ON/OFF switch by using Digital input pin. For Arduino UNO, there are 12 digital input / output pins (2-13) where you can use one of the pins to detect voltage switch. Arduino NANO has 11 pins while Arduino MEGA has 54 pins. The voltage input pins will map input voltages between 0 (LOW) and 3.3V / 5V (HIGH). Voltage more than 3.3V is consider high.

2.2.4.Image sensor:

An image sensor or imager is a sensor that detects and conveys information used to make an image. It does so by converting the variable attenuation of light waves (as they pass through or reflect off objects) into signals, small bursts of current that convey the information. The waves can be light or other electromagnetic radiation. Image sensors are used in electronic imaging devices of both analog and digital types, which include digital cameras, camera modules, camera phones, optical mouse devices, medical imaging equipment, night vision equipment such as thermal imaging

devices, radar, sonar, and others. As technology changes, electronic and digital imaging tends to replace chemical and analog imaging.

The two main types of electronic image sensors are the charge-coupled device (CCD) and the active-pixel sensor (CMOS sensor). Both CCD and CMOS sensors are based on metal–oxide–semiconductor (MOS) technology, with CCDs based on MOS capacitors and CMOS sensors based on MOSFET (MOS field-effect transistor) amplifiers. Analog sensors for invisible radiation tend to involve vacuum tubes of various kinds, while digital sensors include flat-panel detectors. Image sensors with built-in processing units for machine vision are known as smart image sensors or intelligent image sensors

The two main types of digital image sensors are charge coupled device (OCD) and the active-pixel sensor (CMOS sensor) fabricated in complementary MOS (CMOS) or N- type MOS (NMOS or Live MOS) technologies. Both CCD and CMOS sensors are based on MOS technology, with MOS capacitors being the building blocks of a CCD, and MOSFET amplifiers being the building blocks of a CMOS sensor.

Cameras integrated in small consumer products generally use CMOS sensors, which are usually cheaper and have lower power consumption in battery powered devices than CCDs. CCD sensors are used for high end broadcast quality video cameras, and CMOS sensors dominate in still photography and consumer goods where overall cost is a major concern. Both types of sensor accomplish the same task of capturing light and converting it into electrical signals.

Each cell of a CCD image sensor is an analog device. When light strikes the chip it is held as a small electrical charge in each photo sensor. The charges in the line of pixels nearest to the (one or more) output amplifiers

are amplified and output, then each line of pixels shifts its charges one line closer to the amplifiers, filling the empty line closest to the amplifiers. This process is then repeated until all the lines of pixels have had their charge amplified and output.

A CMOS image sensor has an amplifier for each pixel compared to the few amplifiers of a CCD. This results in less area for the capture of photons than a CCD, but this problem has been overcome by using microlenses in front of each photodiode, which focus light into the photodiode that would have otherwise hit the amplifier and not been detected. Some CMOS imaging sensors also use Back-side illumination to increase the number of photons that hit the photodiode.CMOS sensors can potentially be implemented with fewer components, use less power, and/or provide faster readout than CCD sensors.They are also less vulnerable to static electricity discharges.

Another design, a hybrid CCD/CMOS architecture (sold under the name "sCMOS") consists of CMOS readout integrated circuits (ROICs) that are bump bonded to a CCD imaging substrate – a technology that was developed for infrared staring arrays and has been adapted to silicon-based detector technology. Another approach is to utilize the very fine dimensions available in modern CMOS technology to implement a CCD like structure entirely in CMOS technology: such structures can be achieved by separating individual poly-silicon gates by a very small gap; though still a product of research hybrid sensors can potentially harness the benefits of both CCD and CMOS imagers.

2.2.5.Wheel encoder:

A rotary encoder is a type of position sensor that converts the angular position (rotation) of a knob into an output signal that is used to

determine what direction the knob is being rotated.

Due to their robustness and fine digital control; they are used in many applications including robotics, CNC machines and printers.

There are two types of rotary encoder – absolute and incremental. The absolute encoder gives us the exact position of the knob in degrees while the incremental encoder reports how many increments the shaft has moved.

How does an encoder work?

Encoders use different types of technologies to create a signal, including: mechanical, magnetic, resistive and optical – optical being the most common. In optical sensing, the encoder provides feedback based on the interruption of light.

The graphic below outlines the basic construction of an incremental rotary encoder using optical technology. A beam of light emitted from an LED passes through the Code Disk, which is patterned with opaque lines (much like the spokes on a bike wheel). As the encoder shaft rotates, the light beam from the LED is interrupted by the opaque lines on the Code Disk before being picked up by the Photodetector Assembly. This produces a pulse signal: light = on; no light = off. The signal is sent to the counter or controller, which will then send the signal to produce the desired function.

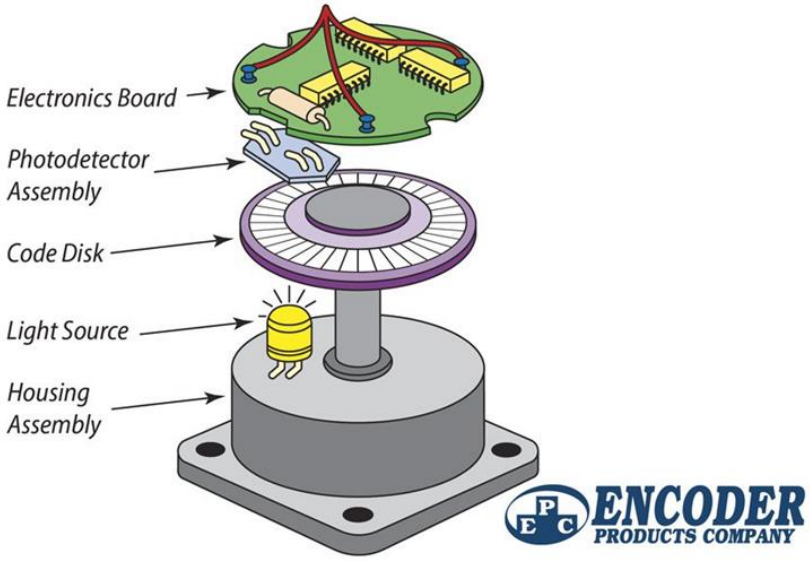


Fig 2.4:example of optical wheel encoder

2.2.6.Kinect sensor:

Kinect is a line of motion sensing input devices produced by Microsoft and first released in 2010. The devices generally contain RGB cameras, and infrared projectors and detectors that map depth through either structured light or time of flight calculations, which can in turn be used to perform real-time gesture recognition and body skeletal detection, among other capabilities. They also contain microphones that can be used for speech recognition and voice control

The Kinect contains three vital pieces that work together to detect motion and create 3D cloud image on the screen: an RGB color VGA video camera, a depth sensor, and a multi-array microphone.

The camera detects the red, green, and blue color components as well as body-type and facial features. It has a pixel resolution of 640x480 and a frame rate of 30 fps. This helps in object recognition.

The depth sensor contains a monochrome CMOS sensor and infrared

projector that help create the 3D imagery throughout the room. It also measures the distance of each point of the angle of view by transmitting invisible near-infrared light and measuring its "time of flight" after it reflects off the objects.

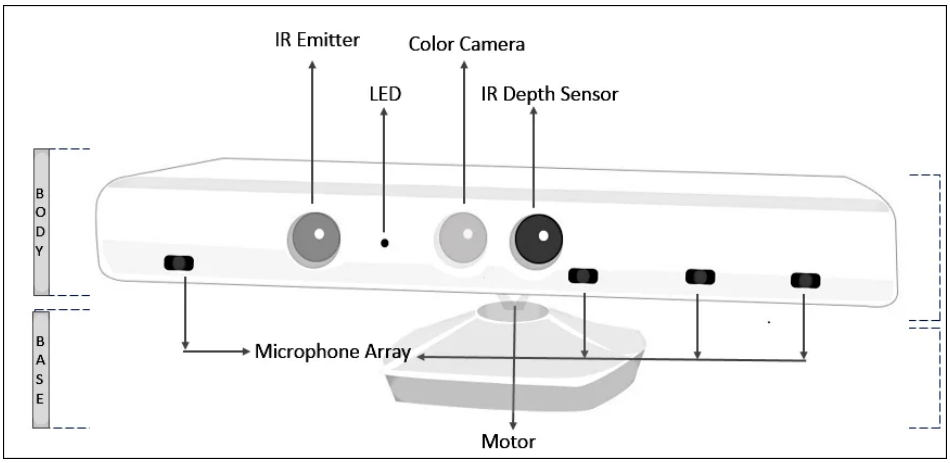


Fig 2.5:Kinect sensor parts

2.2.7.Sensor's sensitivity:

A sensor's sensitivity indicates how much the sensor's output changes when the input quantity being measured changes. For instance, if the mercury in a thermometer moves 1 cm when the temperature changes by 1 °C, the sensitivity is 1 cm/°C (it is basically the slope dy/dx assuming a linear characteristic). Some sensors can also affect what they measure; for instance, a room temperature thermometer inserted into a hot cup of liquid cools the liquid while the liquid heats the thermometer. Sensors are usually designed to have a small effect on what is measured; making the sensor smaller often improves this and may introduce other advantages.

Technological progress allows more and more sensors to be manufactured on a microscopic scale as micro-sensors using MEMS technology. In most cases, a micro-sensor reaches a significantly faster measurement time and higher sensitivity compared with macroscopic

approaches. Due to the increasing demand for rapid, affordable and reliable information in today's world, disposable sensors—low-cost and easy-to-use devices for short-term monitoring or single-shot measurements—have recently gained growing importance. Using this class of sensors, critical analytical information can be obtained by anyone, anywhere and at any time, without the need for recalibration and worrying about contamination.

2.3.Electric Motors:

An electric motor is an electrical machine that converts electrical energy into mechanical energy. Most electric motors operate through the interaction between the motor's magnetic field and electric current in a wire winding to generate force in the form of torque applied on the motor's shaft.

Electric motors can be powered by direct current (DC) sources, such as from batteries, or rectifiers, or by alternating current (AC) sources, such as a power grid, inverters or electrical generators. An electric generator is mechanically identical to an electric motor, but operates with a reversed flow of power, converting mechanical energy into electrical energy.

Electric motors may be classified by considerations such as power source type, internal construction, application and type of motion output. In addition to AC versus DC types, motors may be brushed or brushless, may be of various phase (see single-phase, two-phase, or three-phase), and may be either air-cooled or liquid-cooled. General-purpose motors with standard dimensions and characteristics provide convenient mechanical power for industrial use. The largest electric motors are used for ship propulsion, pipeline compression and pumped-storage applications with ratings reaching 100 megawatts. Electric motors are found in industrial fans, blowers and pumps, machine tools, household appliances, power tools and disk drives.

Small motors may be found in electric watches. In certain applications, such as in regenerative braking with traction motors, electric motors can be used in reverse as generators to recover energy that might otherwise be lost as heat and friction. Electric motors produce linear or rotary force (torque) intended to propel some external mechanism, such as a fan or an elevator. An electric motor is generally designed for continuous rotation, or for linear movement over a significant distance compared to its size. Magnetic solenoids are also transducers that convert electrical power to mechanical motion, but can produce motion over only a limited distance.

2.3.1.DC motors:

Normal DC motors have almost linear characteristics with their speed of rotation being determined by the applied DC voltage and their output torque being determined by the current flowing through the motor windings. The speed of rotation of any DC motor can be varied from a few revolutions per minute (rpm) to many thousands of revolutions per minute making them suitable for electronic, automotive or robotic applications. By connecting them to gearboxes or gear-trains their output speed can be decreased while at the same time increasing the torque output of the motor at a high speed. The “Brushed” DC Motor: A conventional brushed DC Motor consist basically of two parts, the stationary body of the motor called the Stator and the inner part which rotates producing the movement called the Rotor or “Armature” for DC machines. The motors wound stator is an electromagnet circuit which consists of electrical coils connected together in a circular configuration to produce the required North-pole then a South-pole then a North-pole etc, type stationary magnetic field system for rotation, unlike AC machines whose stator field continually rotates with the applied frequency. The current which flows within these field coils is known as the motor field current.

These electromagnetic coils which form the stator field can be electrically connected in series, parallel or both together (compound) with the motors armature. A series wound DC motor has its stator field windings connected in series with the armature. Likewise, a shunt wound DC motor has its stator field windings connected in parallel with the armature as shown.

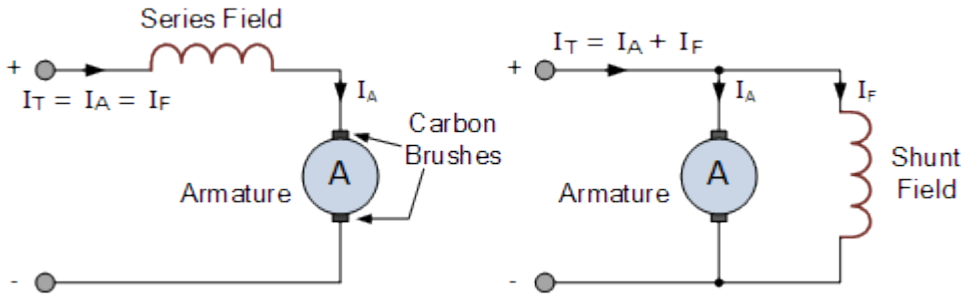


Fig 2.6 :series and shunt connected moor

The rotor or armature of a DC machine consists of current carrying conductors connected together at one end to electrically isolated copper segments called the commutator. The commutator allows an electrical connection to be made via carbon brushes (hence the name "Brushed" motor) to an external power supply as the armature rotates. The magnetic field setup by the rotor tries to align itself with the stationary stator field causing the rotor to rotate on its axis, but can not align itself due to commutation delays. The rotational speed of the motor is dependent on the strength of the rotors magnetic field and the more voltage that is applied to the motor the faster the rotor will rotate. By varying this applied DC voltage the rotational speed of the motor can also be varied.

2.3.1.1.Conventional (Brushed) DC Motor:

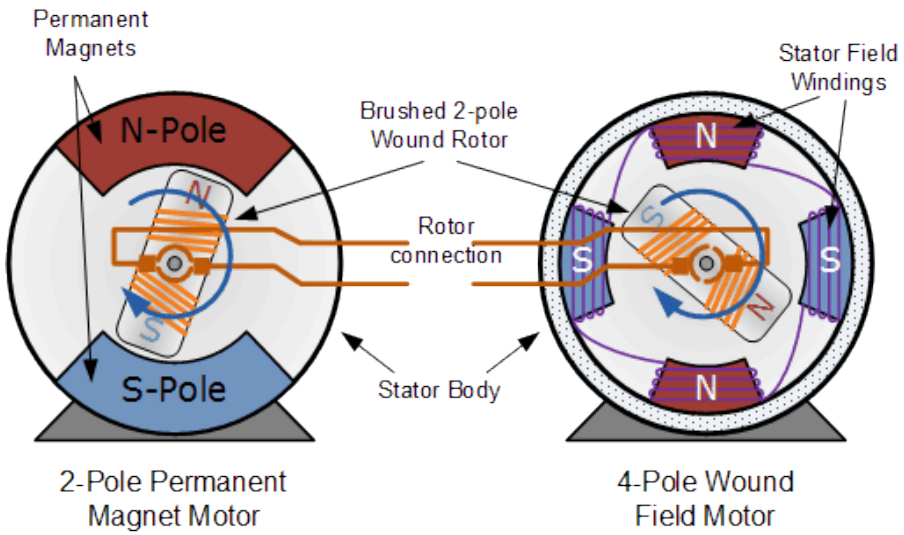


Fig 2.7: 2-pole permnanent magnet motor , and 4- Pol wound field motor

The Permanent magnet (PMDC) brushed DC motor is generally much smaller and cheaper than its equivalent wound stator type DC motor cousins as they have no field winding. In permanent magnet DC (PMDC) motors these field coils are replaced with strong rare earth (i.e. Samarium Cobolt, or Neodymium Iron Boron) type magnets which have very high magnetic energy fields.

The use of permanent magnets gives the DC motor a much better linear speed/torque characteristic than the equivalent wound motors because of the permanent and sometimes very strong magnetic field, making them more suitable for use in models, robotics and servos.

Although DC brushed motors are very efficient and cheap, problems associated with the brushed DC motor is that sparking occurs under heavy load conditions between the two surfaces of the commutator and carbon brushes resulting in self generating heat, short life span and electrical noise due to sparking, which can damage any semiconductor switching device such as a MOSFET or transistor. To overcome these disadvantages, Brushless DC

Motors were developed.

2.3.2.The “Brushless” DC Motor:

The brushless DC motor (BDCM) is very similar to a permanent magnet DC motor, but does not have any brushes to replace or wear out due to commutator sparking. Therefore, little heat is generated in the rotor increasing the motors life. The design of the brushless motor eliminates the need for brushes by using a more complex drive circuit were the rotor magnetic field is a permanent magnet which is always in synchronization with the stator field allows for a more precise speed and torque control. Then the construction of a brushless DC motor is very similar to the AC motor making it a true synchronous motor but one disadvantage is that it is more expensive than an equivalent “brushed” motor design.

The control of the brushless DC motors is very different from the normal brushed DC motor, in that it this type of motor incorporates some means to detect the rotors angular position (or magnetic poles) required to produce the feedback signals required to control the semiconductor switching devices. The most common position/pole sensor is the “Hall Effect Sensor”, but some motors also use optical sensors.

Using Hall effect sensors, the polarity of the electromagnets is switched by the motor control drive circuitry. Then the motor can be easily synchronized to a digital clock signal, providing precise speed control. Brushless DC motors can be constructed to have, an external permanent magnet rotor and an internal electromagnet stator or an internal permanent magnet rotor and an external electromagnet stator.

Advantages of the Brushless DC Motor compared to its “brushed” cousin is higher efficiencies, high reliability, low electrical noise, good speed control and more importantly, no brushes or commutator to wear out

producing a much higher speed. However their disadvantage is that they are more expensive and more complicated to control.

2.3.2.1.Controlling the Magnetic Field of BLDC

To rotate a BLDC motor, one needs to control the direction and timing of the current into the coils. Figure 1.7 illustrates a BLDC motor's stator (coils) and rotor (permanent magnets). Let's use this illustration to see how the rotor is made to turn. In this example we are using three coils, whereas in practice it is more common to use six or more. But here we look at just three coils, spaced at 120º. A motor, as we noted in our last session, performs the task of converting electrical energy into mechanical energy. So how does the motor in our illustration do this? Let's look at what goes on inside.

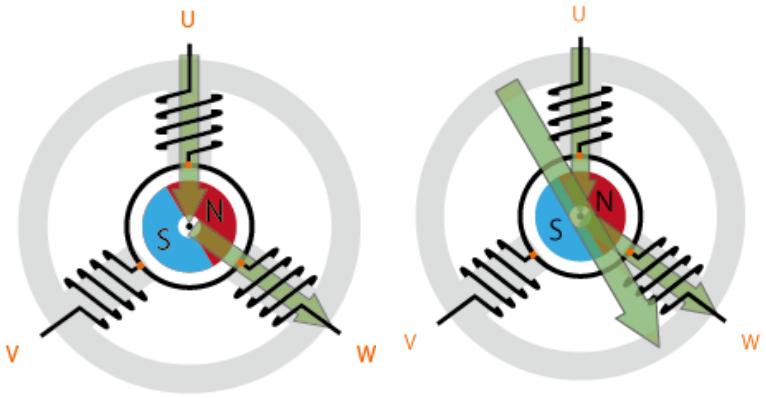


Fig 2.8:BLDC motor magnets and wounds

Typical configuration: three coils with 120º spacing. Driven by controlling the phases and coil currents.

For our example, we show a 3-coil motor. Let's label the coils U, V, and W. Remember that the passing of a current through a coil generates a magnetic field. Since there are three coils, there are three paths through which we can pass current; we can call these phase U (current into coil U), phase V

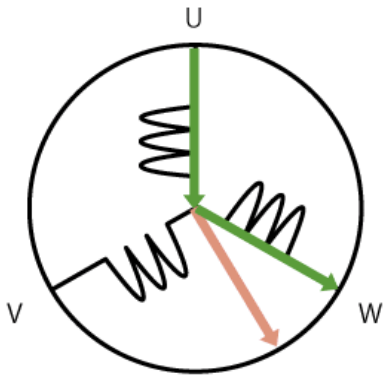
(into coil V), and phase W. Let's look first at phase U. If current moves through U only, magnetic flux is generated as shown by the arrow in Fig. 2(b). In actuality, all three coils are interconnected, through a single lead wire from each—and it is not possible to generate phase U in isolation. Figure 2(c) shows what happens when current moves through coils U and W (phase "U□W"), with the arrows again showing the flux generated at each coil. The wide arrow in Figure 2(d) is the resultant flux—the result of the combined magnetic fields from U and W. This large flux will cause the inner rotor to turn until the S and N poles of the rotor permanent magnet are aligned on this arrow (with the N pole closest to the arrow tip).

Current flows through U and W. The two arrows show flux generated by coils U and W, respectively.

The wide arrow shows the resultant flux—the sum of the flux produced by U and W.

Rotation is maintained by continually switching the flux so that the permanent magnet is constantly chasing the rotating magnetic field induced by the coils. In other words, energizing of U, V, and W must be continually switched so that the resultant flux keeps moving, producing a rotating field that continually pulls on the rotor magnet.

Figure 1.8 shows the relationship between energized phases and flux. As you can see, switching sequentially through modes 1 through 6 will cause the rotor to turn clockwise through one rotation. The speed of rotation can be controlled by controlling the rate at which the phases change. We use the name "120-degree conducting control" to the 6-mode control method described here.



Energizing Mode	Energized Phase	Resultant Flux
1	U → W	
2	U → V	
3	W → V	
4	W → U	
5	V → U	
6	V → W	

Figure 2.9: The changing resultant flux continually pulls the rotor magnet, causing the rotor to turn.

2.3.2.2.Sinusoidal Control Delivers Smooth Rotation

With 120-degree conducting control, there are only six resultant flux directions for driving the motor. For example, switching from Mode 1 to 2 (see Fig. 3) moves the resultant flux direction by 60º, pulling the rotor along accordingly. Switching from Mode 2 to 3 shifts the flux direction another 60º, again pulling the rotor. Repeating this process generates a continuous rotation, but it is a somewhat jerky rotation. In some cases, this jerkiness will create unwanted vibrations and mechanical noise.

As an alternative to 120-degree conducting control, we can use sinusoidal control to achieve smoother and quieter operation. With 120-degree conducting control, the motor is controlled by continually cycling through six fixed resultant fluxes. And as shown in Figure 1.7, U and V both generate fluxes of equal magnitude. By more carefully controlling the current into U, V, and W, however, we can generate differing flux magnitudes at each coil, allowing us to more precisely vary the resultant flux.

By carefully adjusting the current flow into each of the three phases, then, we can achieve a more continuous change in resultant flux, resulting in smoother motor rotation.

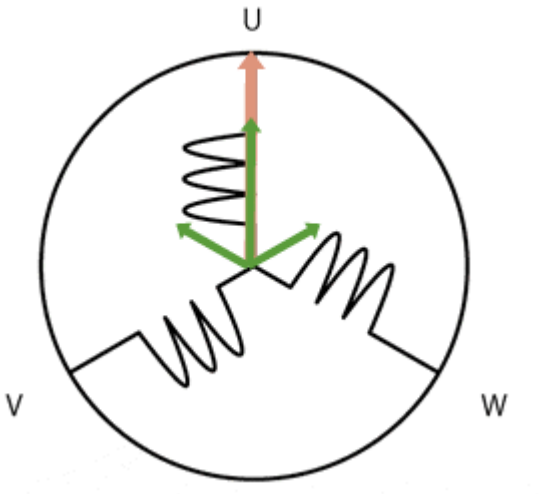


Figure 2.10: Sinusoidal Control.

By controlling the current into all three phases, resultant flux magnitude and direction can be controlled more precisely that with 120-degree conducting control, so as to achieve smoother rotation. Resultant flux is no longer limited to six discrete directions.

2.3.2.3.Control by Inverter

taking a look again at the nature of the current into U, V, and W. For simplicity, let's see how this works with 120-degree conducting control. Looking back to Figure 1.7, we see that in Mode 1 the current flows from U to W; in Mode 2, from U to V. As the arrows in the figure show, each change in the energized coil combination causes a corresponding change in the flux direction.

And by using the inverter circuitry to also adjust the voltage into each coil, we can in addition control the magnitude of the current. A typical way to adjust the voltage is with pulse width modulation (PWM). In this approach, we alter the voltage by lengthening or reducing the pulse ON time (also referred to as the "duty cycle": the ON time expressed as a ratio of the

ON+OFF switching interval). Increasing the duty cycle has the same effect as raising the voltage; reducing the duty cycle has the same effect as lowering the current.

PWM can be implemented using MPUs equipped with dedicated PWM hardware. While 120-degree conducting control only requires two-phase voltage control and be implemented relatively easily in software, sinusoidal control uses three-phase voltage control and is considerably more complicated. Appropriate inverter circuitry is therefore essential for driving BLDC motors. Note that inverters can also be used with AC motors. But when a term such as "inverter type" is used with reference to consumer electronics, it is usually referring to a BLDC motor.

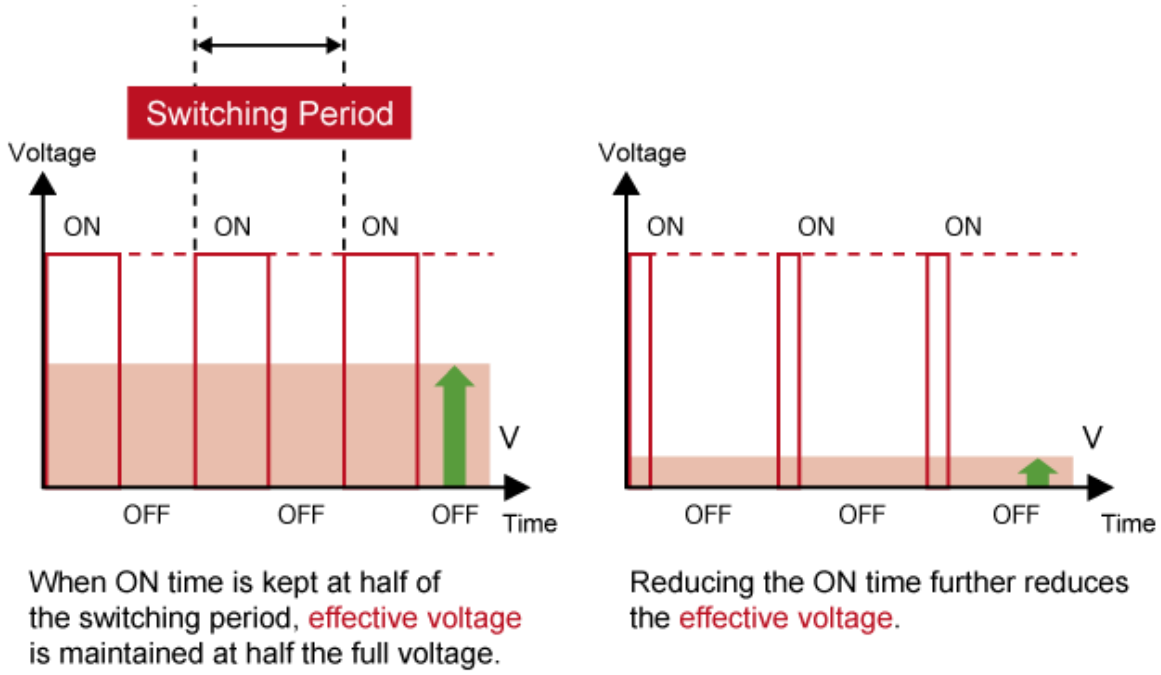


Fig 2.11 : PWM Output vs. Output Voltage.

Varying the duty cycle (the ON time within each switching period) changes the effective voltage.

2.3.3.Servo motors:

A servo motor is a type of motor that can rotate with great precision. Normally this type of motor consists of a control circuit that provides feedback on the current position of the motor shaft, this feedback allows the servo motors to rotate with great precision. If you want to rotate an object at some specific angles or distance, then you use a servo motor. It is just made up of a simple motor which runs through a servo mechanism. If motor is powered by a DC power supply then it is called DC servo motor, and if it is AC-powered motor then it is called AC servo motor.

Apart from these major classifications, there are many other types of servo motors based on the type of gear arrangement and operating characteristics. A servo motor usually comes with a gear arrangement that allows us to get a very high torque servo motor in small and lightweight packages. Due to these features, they are being used in many applications like toy car, RC helicopters and planes, Robotics, etc.

Servo motors are rated in kg/cm (kilogram per centimeter) most hobby servo motors are rated at 3kg/cm or 6kg/cm or 12kg/cm. This kg/cm tells you how much weight your servo motor can lift at a particular distance. For example: A 6kg/cm Servo motor should be able to lift 6kg if the load is suspended 1cm away from the motors shaft, the greater the distance the lesser the weight carrying capacity. The position of a servo motor is decided by electrical pulse and its circuitry is placed beside the motor.

2.3.3.1.Servo Motor Working Mechanism:

It consists of three parts: Controlled device ,Output sensor and Feedback system .It is a closed-loop system where it uses a positive feedback system to control motion and the final position of the shaft. Here the device

is controlled by a feedback signal generated by comparing output signal and reference input signal. Here reference input signal is compared to the reference output signal and the third signal is produced by the feedback system. And this third signal acts as an input signal to the control the device. This signal is present as long as the feedback signal is generated or there is a difference between the reference input signal and reference output signal. So the main task of servomechanism is to maintain the output of a system at the desired value at presence of noises.

2.3.3.2.Servo Motor Working Principle

A servo consists of a Motor (DC or AC), a potentiometer, gear assembly, and a controlling circuit. First of all, we use gear assembly to reduce RPM and to increase torque of the motor. Say at initial position of servo motor shaft, the position of the potentiometer knob is such that there is no electrical signal generated at the output port of the potentiometer. Now an electrical signal is given to another input terminal of the error detector amplifier. Now the difference between these two signals, one comes from the potentiometer and another comes from other sources, will be processed in a feedback mechanism and output will be provided in terms of error signal. This error signal acts as the input for motor and motor starts rotating. Now motor shaft is connected with the potentiometer and as the motor rotates so the potentiometer and it will generate a signal. So as the potentiometer's angular position changes, its output feedback signal changes. After sometime the position of potentiometer reaches at a position that the output of potentiometer is same as external signal provided. At this condition, there will be no output signal from the amplifier to the motor input as there is no difference between external applied signal and the signal generated at potentiometer, and in this situation motor stops rotating.

Interfacing Servo Motors with Microcontrollers:

Interfacing hobby Servo motors like s90 servo motor with MCU is very easy. Servos have three wires coming out of them. Out of which two will be used for Supply (positive and negative) and one will be used for the signal that is to be sent from the MCU. An MG995 Metal Gear Servo Motor which is most commonly used for RC cars humanoid bots etc. The picture of MG995 is shown below:

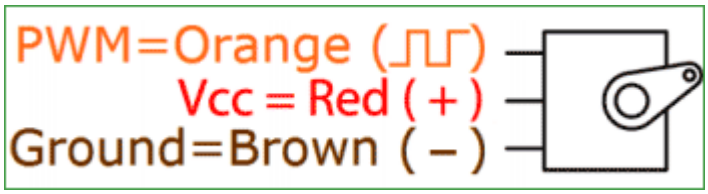


Fig 2.12 : servo motor control signals

The color coding of your servo motor might differ hence check for your respective datasheet. All servo motors work directly with your +5V supply rails but we have to be careful on the amount of current the motor would consume if you are planning to use more than two servo motors a proper servo shield should be designed.

2.3.3.3.Controlling Servo Motor:

All motors have three wires coming out of them. Out of which two will be used for Supply (positive and negative) and one will be used for the signal that is to be sent from the MCU.

Servo motor is controlled by PWM (Pulse with Modulation) which is provided by the control wires. There is a minimum pulse, a maximum pulse and a repetition rate. Servo motor can turn 90 degree from either direction form its neutral position. The servo motor expects to see a pulse every 20 milliseconds (ms) and the length of the pulse will determine how far the motor turns. For example, a 1.5ms pulse will make the motor turn to the 90°

position, such as if pulse is shorter than 1.5ms shaft moves to 0° and if it is longer than 1.5ms than it will turn the servo to 180°.

Servo motor works on PWM (Pulse width modulation) principle, means its angle of rotation is controlled by the duration of applied pulse to its Control PIN. Basically servo motor is made up of DC motor which is controlled by a variable resistor (potentiometer) and some gears. High speed force of DC motor is converted into torque by Gears. We know that WORK= FORCE X DISTANCE, in DC motor Force is less and distance (speed) is high and in Servo, force is High and distance is less. The potentiometer is connected to the output shaft of the Servo, to calculate the angle and stop the DC motor on the required angle.

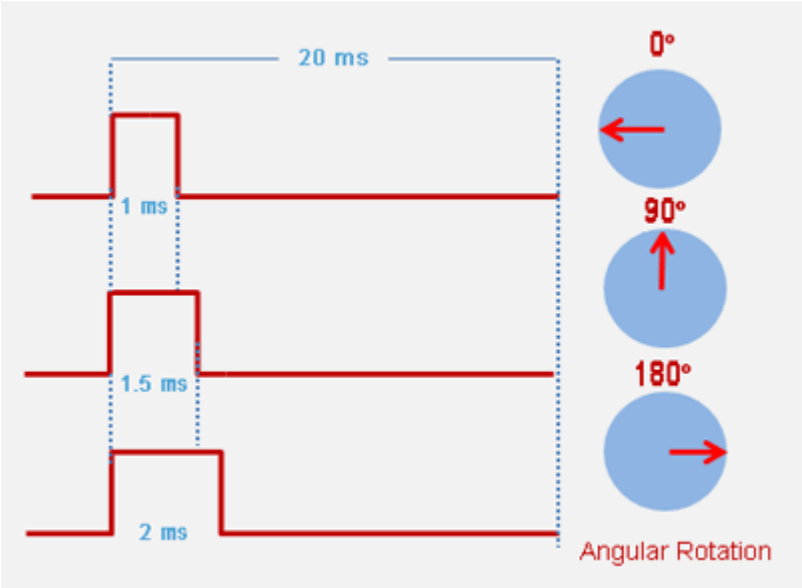


Fig 2.13:servo motor angle control signal

Servo motor can be rotated from 0 to 180 degrees, but it can go up to 210 degrees, depending on the manufacturing. This degree of rotation can be controlled by applying the Electrical Pulse of proper width, to its Control pin. Servo checks the pulse in every 20 milliseconds. The pulse of 1 ms (1

millisecond) width can rotate the servo to 0 degrees, 1.5ms can rotate to 90 degrees (neutral position) and 2 ms pulse can rotate it to 180 degree.

2.4.Darlington circuit:

A Darlington pair is two transistors that act as a single transistor but with a much higher current gain. This mean that a tiny amount of current from a sensor, micro-controller or similar can be used to drive a larger load. An example circuit is shown below:

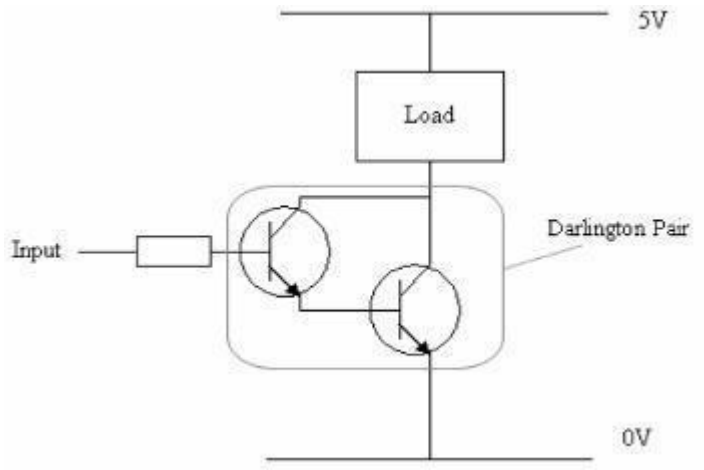


Fig 2.14: Darlington Pair transistors

The Darlington Pair can be made from two transistors as shown in the diagram or Darlington Pair transistors are available where the two transistors are contained within the same package.

What is current gain:

Transistors have a characteristic called current gain. This is referred to as its hFE. The amount of current that can pass through the load in the circuit above when the transistor is turned on is: *Load current = input current x transistor gain (hFE)* The current gain varies for different transistors and can be looked up in the data sheet for the device. For a normal transistor this would typically be about 100. This would mean that the current available to

drive the load would be 100 times larger than the input to the transistor.

Why use a Darlington Pair:

In some applications the amount of input current available to switch on a transistor is very low. This may mean that a single transistor may not be able to pass sufficient current required by the load. As stated earlier this equals the input current x the gain of the transistor (hFE). If it is not possible to increase the input current then the gain of the transistor will need to be increased. This can be achieved by using a Darlington Pair. A Darlington Pair acts as one transistor but with a current gain that equals: Total current gain (hFE total) = current gain of transistor 1 (hFE t1) x current gain of transistor 2 (hFE t2) So for example if you had two transistors with a current gain (hFE) = 100: (hFE total) = 100 x 100 (hFE total) = 10,000 You can see that this gives a vastly increased current gain when compared to a single transistor. Therefore this will allow a very low input current to switch a much bigger load current.

Base Activation Voltage:

Normally to turn on a transistor the base input voltage of the transistor will need to be greater than 0.7V. As two transistors are used in a Darlington Pair this value is doubled. Therefore the base voltage will need to be greater than 0.7V x 2 = 1.4V. It is also worth noting that the voltage drop across collector and emitter pins of the Darlington Pair when the turn on will be around 0.9V Therefore if the supply voltage is 5V (as above) the voltage across the load will be will be around 4.1V (5V – 0.9V)

2.5. L298N H-bridge motor driver:

The L298N is a dual-channel H-Bridge motor driver capable of driving a pair of DC motors. That means it can individually drive up to two

motors making it ideal for building two-wheel robot platforms.

The L298N motor driver module is powered through 3-pin 3.5mm-pitch screw terminals. It consists of pins for motor power supply(Vs), ground and 5V logic power supply(Vss).

The module has an on-board 78M05 5V regulator from STM microelectronics. It can be enabled or disabled through a jumper.When this jumper is in place, the 5V regulator is enabled, supplying logic power supply(Vss) from the motor power supply(Vs). In this case, 5V input terminal acts as an output pin and delivers 5V 0.5A. You can use it to power up the Arduino or other circuitry that requires 5V power supply. When the jumper is removed, the 5V regulator gets disabled and we have to supply 5 Volts separately through 5 Volt input terminal.

The L298N motor driver's output channels for the motor A and B are broken out to the edge of the module with two 3.5mm-pitch screw terminals. two DC motors having voltages between 5 to 35V can connected to these terminals.

Each channel on the module can deliver up to 2A to the DC motor. However, the amount of current supplied to the motor depends on system's power supply.[4]

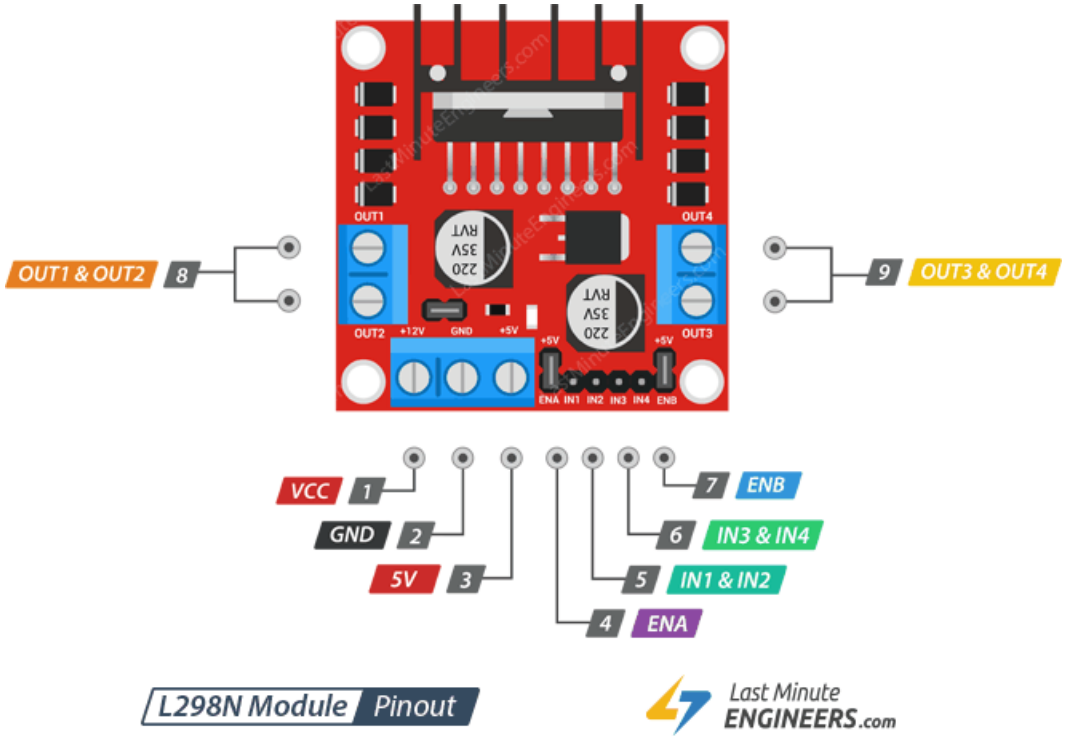


Fig 2.15: L298N H-bridge motor driver pins

2.5.the need for SLAM algorithms:

as previously stated a main reason for using SLAM to draw a map of the robot's environment is the lack of measured and collected sensors information to perform a certain task due to sensors limitations or inaccuracy . for example when using Ultrasonic sensors to detect obstacles that out of its range , as we know ultrasonic sensors are used to precisely detect the position of objects of any material and color, irrespective of external light levels even in harsh industrial environments . The sensors are characterized by high sound intensity that makes it possible to detect even the smallest of objects . In addition to their high precision ,outstanding repeatability and high degree of linearity , although this sensor has great performance under normal conditions , its performance drop in case of sensing small objects that are far relatively to the sensor (figure ... show characteristics of one of the ultrasonic

sensor)

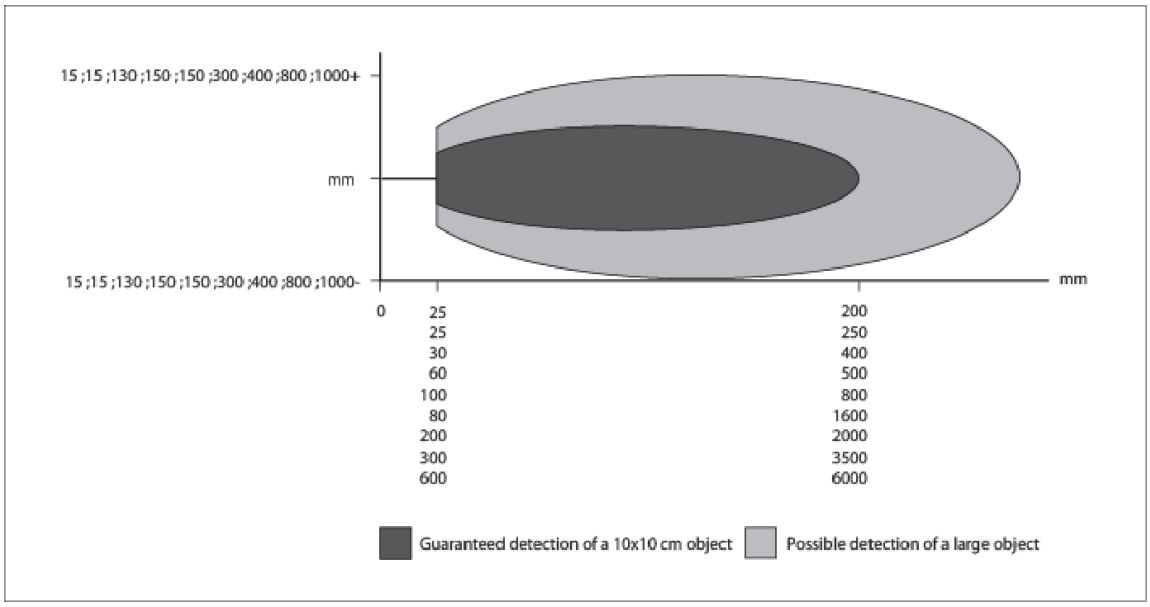


figure 2.16:ultrasonic sensor detection range

we see that the performance of the sensor gradually decreases while distance increases , this case is not acceptable in case of a self-driving car that is moving too fast

Also when using a camera sensor to create a machine vision about the external environment ,we experienced some difficulties due to camera narrow range to detect object , the complexity of of some objects textures and the efficiency limitation of camera modules . one efficiency measure is the Quantum efficiency (QE) which is the measure of the effectiveness of an imaging device to convert incident photons into electrons. For example, if a sensor had a QE of 100% and was exposed to 100 photons, it would produce 100 electrons of signal.

In practice, sensors are never 100% efficient, and different sensor technologies have different QE values. The highest-end scientific cameras can achieve up to 95% QE but this is dependent on the wavelength of light being detected, as seen in figure 2.2 which shows the efficiency of a camera

module against wavelength of the detected image

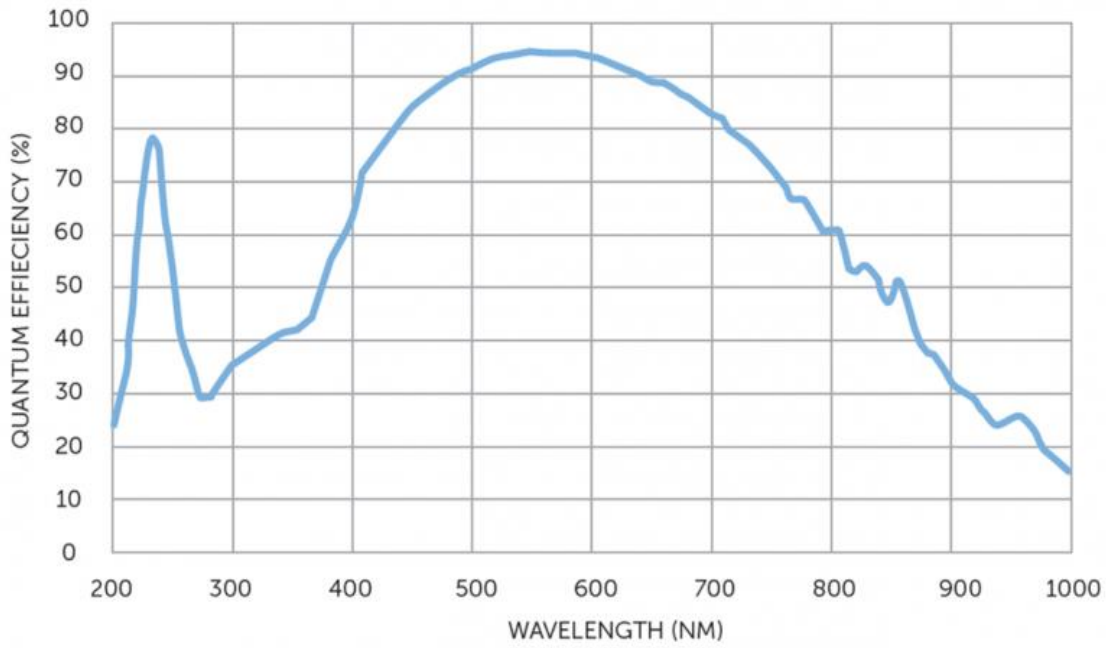


figure 2.17:Quantum efficiency range

2.5.1. methods for solving sensors limitations problem:

2.5.1.1.sensor fusion:

Sensor fusion is a method of integrating signals from multiple sources. It allows extracting information from several different sources to integrate them into single signal or information. In many cases sources of information are sensors or other devices that allow for perception or measurement of changing environment. Information received from multiple-sensors is processed using "sensor fusion" or "data fusion" algorithms. These

algorithms can be classified into three different groups. First, fusion based on probabilistic models, second, fusion based on least-squares techniques and third, intelligent fusion. The probabilistic model methods are Bayesian reasoning, evidence theory, robust statistics, recursive operators. The least-squares techniques are Kalman filtering, optimal theory, regularization and uncertainty ellipsoids. The intelligent fusion methods are fuzzy logic, neural networks and genetic algorithms . in chapter three in this study will use the intelligent fusion methods which are presented in [5]

2.5.1.2.SLAM configuration:

SLAM algorithms work by using robot's visual sensors to draw a map of its environment and navigating in that environment the steps of achieving a certain task in this situation goes through the same steps of controlling an autonomous robot using (sense , perceive , plan and act) steps . the next chapter will briefly discuss these steps from algorithmic point of view for single robot configuration and eventually develop a complete system to control the test robot and will give an insight of how this method can be expanded to multi-robot configuration**.**

CHAPTER THREE
Methodology

CHAPTER THREE
Methodology

3.1.Methodology:

In this chapter we will develop the perceive and plan steps hence sense step is done when the prototype of the robots are developed in the next chapter.

section one will discuss briefly the use of SLAM technology in order to draw a map for the environment and then we propose a method to draw the map when using multi-robot configuration.

the second section of this chapter will discuss the planning algorithm used to plan a path for our robotic system after drawing a map , we notice that this isn't always the case hence sometimes there is no need for the plan step if the SLAM is step is done correctly.

3.1.1.Simultaneous localization and mapping

This section describes the basic features of a generic SLAM algorithm, as well as introducing a few SLAM algorithms. The key features of these algorithms are then summarized in Section 3.5, where each algorithm is a also evaluated in terms of being appropriate for implementation in the group configuration .

3.1.2.Introduction to SLAM

Simultaneous localization and mapping is a problem where a moving object needs to build a map of an unknown environment, while simultaneously calculating its position within this map. [6]

There are several areas which could benefit from having autonomous vehicles with SLAM algorithms implemented. Examples would

be the mining industry, underwater exploration, and planetary exploration. The SLAM problem in general can be formulated using a probability density function denoted p(xt,m|z1:t,u1:t), where xt is the position of the vehicle, m is the map, and z1:t is a vector of all measurements. u1:t is a vector of the control signals of the vehicle, which is either the control commands themselves or odometry, depending on the
application.

3.1.2.2.Data association

Basically, the concept data association is to investigate the relationship between older data and new data gathered. In a SLAM context it is of necessity to relate older measurements to newer measurements. This enables the process of determining the locations of landmarks in the environment, and thus this also gives information regarding the robots position within the map.

3.1.2.3. Loop closure

The concept of loop closure in a SLAM context is the ability of a vehicle to recognize that a location has already been visited. By applying a loop closure algorithm, the accuracy of both the map and the vehicles position within the map can be increased. However, this is not an easy task to perform, due to the fact that the operating environment of the vehicle could contain similar structural circumstances as previously visited locations. In that case if the loop closure algorithm performs poorly, it could lead to a faulty loop closure, which could corrupt both the map as well as the pose of the vehicle within the map.

3.1.2.4.Full SLAM and online SLAM

There are two different types of SLAM problems. The online SLAM problem of which only the current pose xt and the map m are expressed, given the control input u1:t and measurements z1:t. As well as the full SLAM problem which expresses the entire trajectory of the robot. The PDF of the full SLAM problem is denoted as p(x0:t,m|z1:t,u1:t), where all the poses of the robot are considered, including its initial pose x0.

3.1.2.5 .General models for SLAM problem

A motion model for the SLAM problem can be described by

$$xt = g(ut,xt-1) + \delta t \quad (3.1)$$

where the current pose xt depends on the possible nonlinear function g(ut,xt−1), with ut being the control input, xt−1 the previous pose and δt being a random Gaussian variable with zero mean

A measurement model for the SLAM problem can be described by:

$$zt = h(xt,m) + t \quad (3.2)$$

where the current measurement zt, depends on the possible nonlinear function h(xt,m), where xt denotes the current pose, m represents the map, t a random Gaussian variable with zero mean.[7]

3.1.3 .EKF SLAM

The EKF SLAM (Extended Kalman Filter) approach of solving the SLAM problem is by using sensor data gathered from the movement and rotation of the robot. This can be done by for example using wheel encoders

and a gyroscope. Furthermore, it is also necessary to gather information of the environment by, for example, using a Laser Range Finder (LIDAR). With this data, the algorithm

keeps track of where the robot is likely positioned within a map, as well as keeping track of specific landmarks observed.[8]

3.1.3.2. Algorithm

When the robot is turned on, the sensors on the robot such as the wheel encoders and the gyroscope will gather information of the positioning of the robot. Moreover, landmarks from the environment are also extracted based on new observations from the LIDAR which is mounted on the robot. These new observations are associated with previous observations and updated in the EKF algorithm. However, if an observation of a landmark cannot be associated to a previous observation the observation itself is presented to the EKF algorithm as a new observation .

3.1.4.FastSLAM

Another way of solving the SLAM problem is using the FastSLAM algorithm,which makes use of the Rao-Blackwellized particle filter. FastSLAM takes advantage of something that most other SLAM algorithms do not, namely the fact that each observation only concerns a small amount of the state variables. The pose of the robot is probabilistically dependent on its previous pose, whereas the landmark locations are probabilistically dependent on the position of the robot. This is not taken into account by the EKF SLAM algorithm, for example, which with each update has to recalculate its covariance matrix, resulting in poor scaling in large maps. This is not a problem in FastSLAM, as it factors out the calculations to simpler subproblems, as shown by

$$p(s_t, \theta \mid z_t, u_t, n_t) = p(s_t \mid z_t, u_t, n_t) \prod_{n=1}^{N} p(\theta_n \mid s_t, z_t, u_t, n_t)$$

which factors out the robot path st and each of the N landmarks θn. One major difference here between FastSLAM and, for example EKF SLAM, is that the entire path st is calculated, as opposed to just the current pose xt. This means that FastSLAM is a full SLAM algorithm and EKF SLAM is an online SLAM algorithm,[9] as described in Section 3.1.1.4

3.1.4.1. Particle structure

The particle filter used in FastSLAM is comprised of M particles where each particle is defined by

$$S_t[m] = (s_t[m], \mu_{1,t}, \Sigma_{1,t}, \ldots, \mu_{N,t}, \Sigma_{N,t})$$

Each particle contains N + 1 variables, one robot path st[m], and N landmark posteriors, where each landmark is represented by an EKF with a mean $\mu_{n,t}$ and a variance $\Sigma_{n,t}$. During one iteration of the algorithm, the robot path for every particle st[m] is updated once, as well as each landmark filter $\mu_{n,t}, \Sigma_{n,t}$

3.1.4.2.Algorithm

Each iteration of the algorithm can be summarized in four steps. These are presented below

1.For each of the M particles, sample a new robot path st from

p(st |s[m] t−1,ut)

2.With the new measurement, update the landmark filters

3.Give each particle an importance factor

4.Resample the particles, with particles with a higher importance factor having a higher draw probability

In step 1, a new robot path is sampled for each particle. This path is drawn from the motion model, with the robot control as input. After this, the posteriors of the landmarks are updated due to the new measurement. After this, each particle is given an importance factor, which is a measure on how realistic the variables in the particle are. Later, the particles are resampled in order to get rid of the ones which do not have high probability.

3.1.5. FastSLAM 2.0

The FastSLAM algorithm (and particle filters in general) performs more accurately when there is more diversity among the particles. Particles will not be diverse if, for example, the odometry sensors are noisy and the laser range finder is accurate, as this will render many of the motion hypothesises unlikely and many particles will have a low importance factor. Because of this drawback of the FastSLAM algorithm, an updated one has been developed, named FastSLAM 2.0 . FastSLAM 2.0 is quite similar to its predecessor, with the exception that when estimating the motion, it also considers the range sensors. This creates better guesses in general, and will result in more particles having higher importance factors [10].

3.1.6.GraphSLAM

is another approach of solving the SLAM problem. The idea behind GraphSLAM is to address the SLAM problem by using a graph. The graph contains nodes which represent the poses of the robot x0,...,xt as well as landmarks in the map which are denoted as m0,...,mt. However, this is not enough to address the SLAM problem, as there are also constraints between the positions xt,xt−1,xt−2,...,xt−n of the robot and the landmarks m0,...,mt. These constraints represents the distance between adjacent locations such as xt−1,xt as well as the distance between the locations to the landmarks m0,...mt. These constraints can be constructed due to information from the odometry source ut. The graph mentioned earlier can be converted to a sparse matrix. Which in contrast to dense matrices can be more efficiently used. Fig. 3.1 illustrates how the landmarks m0,...,mt, the locations x0,...,xt and the constraints are linked together.

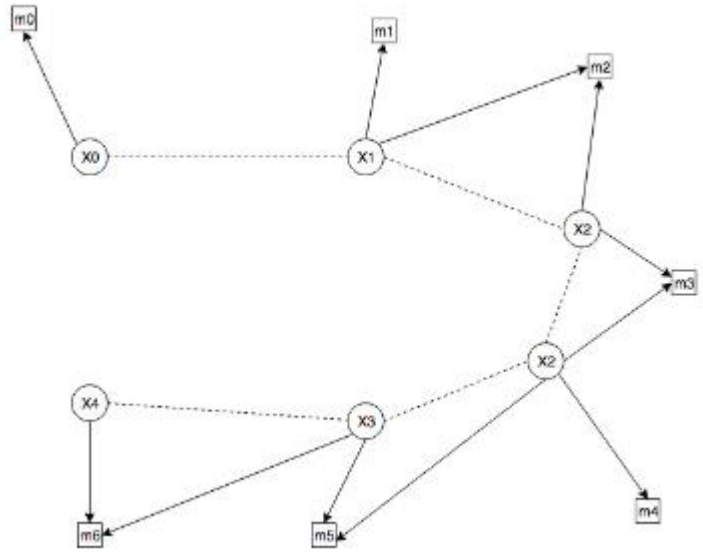


Fig 3.1:landmarks locations and constrains

in the figure above Robot poses seen as circles, landmarks of the map seen as squares, dashed lines representing the motion of the robot based

on control commands given to the robot as well as solid lines representing the distance to the landmarks given by measurements.[11]

3.1.7. Full SLAM

The PDF for the full SLAM problem is denoted p(x0:t,m|z1:t,u1:t). For simplicity, all the poses x0:t and the landmarks m can be described by a state space vector y

$$y = [x_0 \ \vdots_1 \ m\,]$$

Then in order to achieve the most likely pose of the robot within the map and the positions of the landmarks in the map, an optimization algorithm has to be computed. Moreover, the PDF p(y|z1:t,u1:t) contains the possibly nonlinear functions g(ut,xt−1) and h(xt,mi), whereas mi represents the i-th landmark observed at time t. The optimization algorithm expects linear functions. Thus, a necessary step is to linearize these. This could be done by for example executing a Taylor linearization algorithm.

As specified earlier, the GraphSLAM approach of solving the SLAM problem is by introducing a graph. This graph can be converted to a sparse matrix. Basically, this sparse matrix contains all the information acquired by the robot while moving in an environment. For example, the robots movement through the map and landmarks observed at different locations. Thus, as the robot moves through the environment the sparse matrix will increase in size and contain even more information.[12]

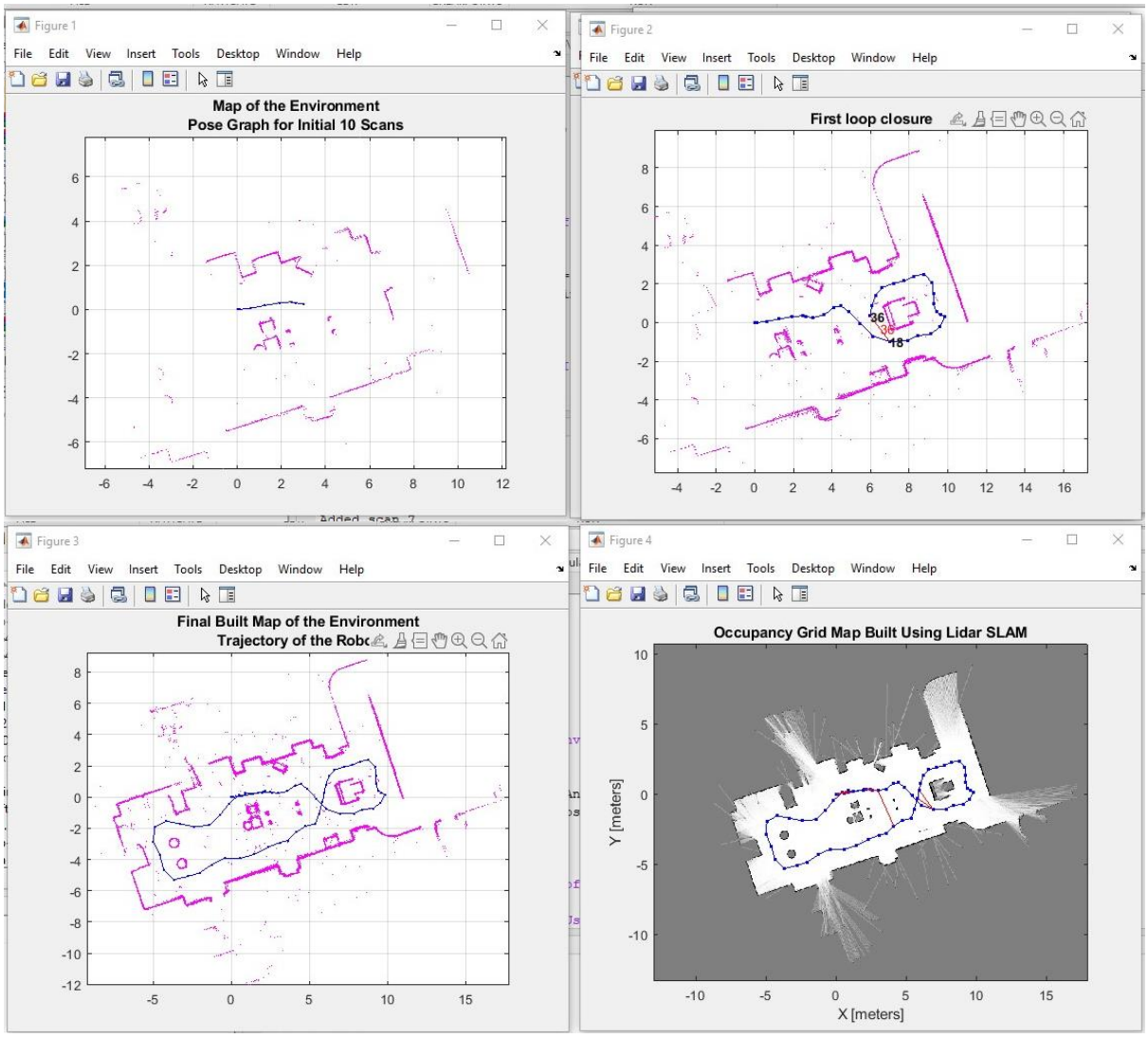


Fig 3.2 :fastSLAM algorithm simulation

the simulated algorithm uses dataset gathered earlier from indoor environment using LIDAR sensor to create a map of the environment and localize the robot's position , the final map is shown in the occupancy grid map in figure 4

3.1.8.RTAB-Maping:

(Real-Time Appearance-Based Mapping) is a RGB-D, Stereo and Lidar Graph-Based SLAM approach based on an incremental appearance-based loop closure detector. The loop closure detector uses a bag-of-words approach to determinate how likely a new image comes from a previous location or a new location. When a loop closure hypothesis is accepted, a

new constraint is added to the map's graph, then a graph optimizer minimizes the errors in the map. A memory management approach is used to limit the number of locations used for loop closure detection and graph optimization, so that real-time constraints on large-scale environnements are always respected. RTAB-Map can be used alone with a handheld Kinect, a stereo camera or a 3D lidar for 6DoF mapping, or on a robot equipped with a laser rangefinder for 3DoF mapping.[13]

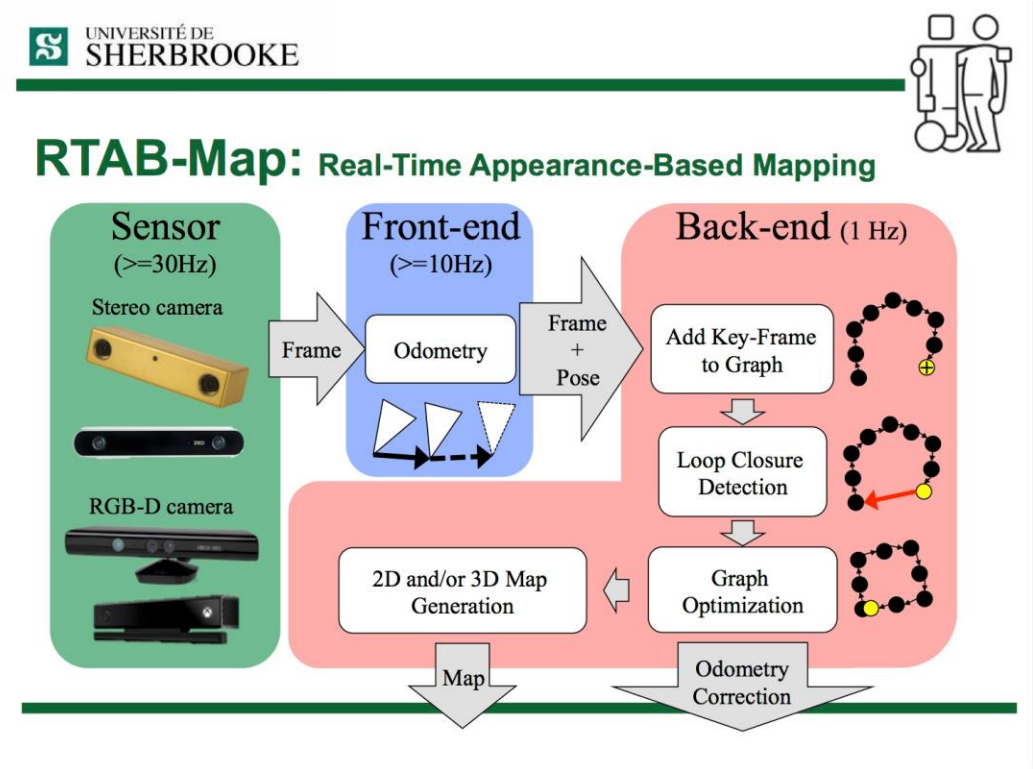


Fig 3.3:RTAB-Map flowchart[13]

3.1.9. Our proposed method for SLAM algorithm

Robots could either know initial poses of all the robots or determine their relative poses (distance and headings) when they meet. This algorithm works under the assumption that robots can recognize each member in the team and determine their pose and transfer their motions and observations reliably. The basic idea is that a robot has no prior information

of other robots, therefore robot performs single robot SLAM until it observes another one. When two robots (robot A and robot B) meet each other at time t, they determine their relative pose by using their own sensors. Then particle filter of robot A adds two additional parameters: a causal instance corresponding to forward motion of the robot, and an acausal instance corresponding to time-reversed motion of robot. Then the queued data of robot B (observation and odometry) is divided into causal instance and acausal instance as well. The causal instance is empty at the time when they are meeting and it is updated with the odometry and observation data of robot B received by wireless. The causal queue is used to localize the observation of robot B into the map of robot A after the time when they meet. The acausal queue is used to combine the observation of robot B to the map of robot A before the time they meet. Integration of data in the acausal queue is considered to be a virtual robot moving backward until the beginning. So in this manner, one robot can combine the observation of all the robots into a single map. Therefore, each robot will produce a map of whole area after they meet each other. No map merging is required since all robots presented a map of whole environment .

3.2. Motion planning

Motion planning, also path planning (also known as the navigation problem or the piano mover's problem) is a computational problem to find a sequence of valid configurations that moves the object from the source to destination. The term is used in computational geometry, computer animation, robotics and computer games

For example, consider navigating a mobile robot inside a building to a distant waypoint. It should execute this task while avoiding walls and not

falling down stairs. A motion planning algorithm would take a description of these tasks as input, and produce the speed and turning commands sent to the robot's wheels. Motion planning algorithms might address robots with a larger number of joints (e.g., industrial manipulators), more complex tasks (e.g. manipulation of objects), different constraints (e.g., a car that can only drive forward), and uncertainty (e.g. imperfect models of the environment or robot)

A basic motion planning problem is to compute a continuous path that connects a start configuration S and a goal configuration G, while avoiding collision with known obstacles. The robot and obstacle geometry is described in a 2D or 3D workspace, while the motion is represented as a path in (possibly higher-dimensional) configuration space.

3.2.1.Configuration space

A configuration describes the pose of the robot, and the configuration space C is the set of all possible configurations. For example:

> If the robot is a single point (zero-sized) translating in a 2-dimensional plane (the workspace), p is a plane, and a configuration can be represented using two parameters (x, y).
> If the robot is a 2D shape that can translate and rotate, the workspace is still 2-dimensional. However, c is the special Euclidean group $\mathbf{SE}(2) = \mathbf{R}^2 \times \mathbf{SO}(2)$ (where $\mathbf{SO}(2)$ is the special orthogonal group of 2D rotations), and a configuration can be represented using 3 parameters (x, y, θ).)
> If the robot is a solid 3D shape that can translate and rotate, the workspace is 3-dimensional, but c is the special Euclidean group $\mathbf{SE(3)} = \mathbf{R}^3 \times \mathbf{SO}(3)$, and a configuration requires 6 parameters: (x, y, z) for translation, and Euler angles (α, β, γ).
> If the robot is a fixed-base manipulator with N revolute joints (and no closed-loops), c is N-dimensional.

3.2.2.Free space

The set of configurations that avoids collision with obstacles is called the free space C_{free}. The complement of C_{free} in C is called the obstacle or forbidden region.

Often, it is prohibitively difficult to explicitly compute the shape of Cfree However, testing whether a given configuration is in Cfree is efficient. First, forward kinematics determine the position of the robot's geometry, and collision detection tests if the robot's geometry collides with the environment's geometry.

3.2.3.Target space

Target space is a subspace of free space which denotes where we want the robot to move to. In global motion planning, target space is observable by the robot's sensors. However, in local motion planning, the robot cannot observe the target space in some states. To solve this problem, the robot goes through several virtual target spaces, each of which is located within the observable area (around the robot). A virtual target space is called a sub-goal.

3.2.4.Obstacle space

Obstacle space is a space that the robot can not move to. Obstacle space is not opposite of free space.

3.2.5.Algorithms

Low-dimensional problems can be solved with grid-based algorithms that overlay a grid on top of configuration space, or geometric algorithms that compute the shape and connectivity of Cfree

Exact motion planning for high-dimensional systems under complex constraints is computationally intractable. Potential-field algorithms are efficient, but fall prey to local minima (an exception is the harmonic potential fields). Sampling-based algorithms avoid the problem of local minima, and

solve many problems quite quickly. They are unable to determine that no path exists, but they have a probability of failure that decreases to zero as more time is spent.

Sampling-based algorithms are currently considered state-of-the-art for motion planning in high-dimensional spaces, and have been applied to problems which have dozens or even hundreds of dimensions (robotic manipulators, biological molecules, animated digital characters, and legged robot)

There is the motion planning parallel algorithm (A1-A2) for objects manipulation (for catching the flying object) .

3.2.6.Grid-based search

Grid-based approaches overlay a grid on configuration space and assume each configuration is identified with a grid point. At each grid point, the robot is allowed to move to adjacent grid points as long as the line between them is completely contained within Cfree (this is tested with collision detection). This discretizes the set of actions, and search algorithms (like A*) are used to find a path from the start to the goal.

These approaches require setting a grid resolution. Search is faster with coarser grids, but the algorithm will fail to find paths through narrow portions of Cfree. Furthermore, the number of points on the grid grows exponentially in the configuration space dimension, which make them inappropriate for high-dimensional problems

Traditional grid-based approaches produce paths whose heading changes are constrained to multiples of a given base angle, often resulting in suboptimal paths. Any-angle path planning approaches find shorter paths by propagating information along grid edges (to search fast) without constraining their paths to grid edges (to find short paths)

Grid-based approaches often need to search repeatedly, for example, when the knowledge of the robot about the configuration space changes or the configuration space itself changes during path following. Incremental heuristic search algorithms replan fast by using experience with the previous similar path-planning problems to speed up their search for the current one.

3.2.7. A* Algorithm:

A* is a graph traversal and path search algorithm, which is often used in many fields of computer science due to its completeness, optimality, and optimal efficiency. One major practical drawback is its O (b d) space complexity, as it stores all generated nodes in memory. Thus, in practical travel-routing systems, it is generally outperformed by algorithms which can pre-process the graph to attain better performance, as well as memory-bounded approaches; however, A* is still the best solution in many cases.[14]

3.2.7.1.Explanation

Consider a square grid having many obstacles and we are given a starting cell and a target cell. We want to reach the target cell (if possible) from the starting cell as quickly as possible. Here A* Search Algorithm comes to the rescue. What A* Search Algorithm does is that at each step it picks the node according to a value-'f' which is a parameter equal to the sum of two other parameters – 'g' and 'h'. At each step it picks the node/cell having the lowest 'f', and process that node/cell.

We define 'g' and 'h' as simply as possible below

g = the movement cost to move from the starting point to a given square on the grid, following the path generated to get there.

h = the estimated movement cost to move from that given square on the grid to the final destination. This is often referred to as the heuristic, which is nothing but a kind of smart guess. We really don't know the actual

distance until we find the path, because all sorts of things can be in the way (walls, water, etc.). There can be many ways to calculate this ‘h’ which are discussed in the later sections.

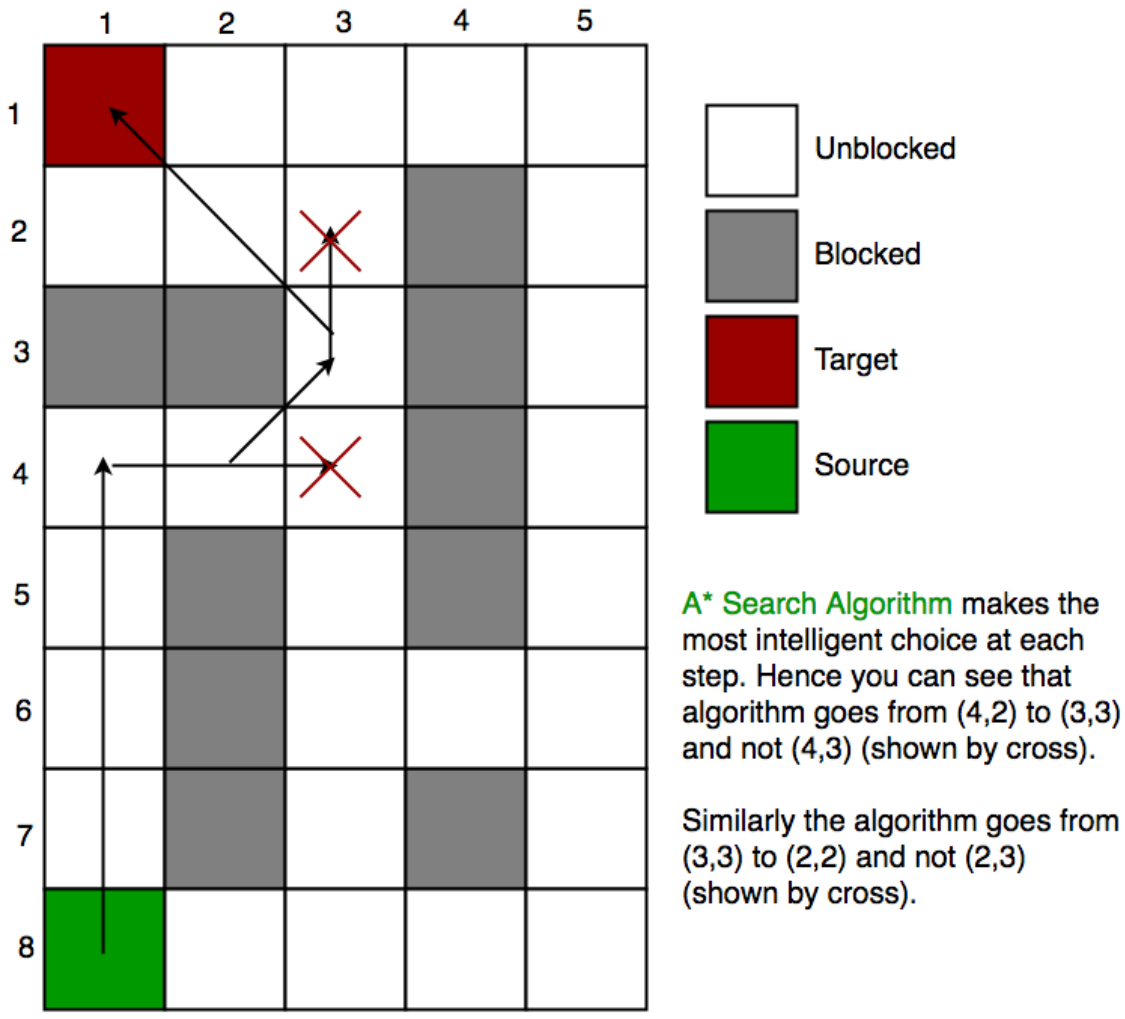


Fig 3.4 :show A* algorithm [15]

3.3. Robot Operating System

Robot Operating System (ROS) is used for all implementations in this thesis. ROS is a framework for writing robot software. The basics of ROS relies on a few core concepts:

nodes communicating with each other by sending or receiving messages on topics. Nodes are self-contained pieces of code, in our case written in linux terminal and rviz visualiztion tool. For example, the odometry model described in section 3.5 runs as a ROS node that subscribes to sensor data from the appropriate topics, processes the data, and publishes car coordinates on the /odom topic[16]. Another concept in ROS worth describing further is the tf transform . Tf is a ROS package that lets the user keep track of multiple coordinate frames over time.[17]

3.3.1. RVIZ

Rviz, abbreviation for ROS visualization, is a powerful 3D visualization tool for ROS. It allows the user to view the simulated robot model, log sensor information from the robot's sensors, and replay the logged sensor information. By visualizing what the robot is seeing, thinking, and doing, the user can debug a robot application from sensor inputs to planned (or unplanned) actions.

Rviz displays 3D sensor data from stereo cameras, lasers, Kinects, and other 3D devices in the form of point clouds or depth images. 2D sensor data from webcams, RGB cameras, and 2D laser rangefinders can be viewed in rviz as image data.

If an actual robot is communicating with a workstation that is running rviz, rviz will display the robot's current configuration on the virtual robot model. ROS topics will be displayed as live representations based on the sensor data published by any cameras, infrared sensors, and laser scanners

that are part of the robot's system. This can be useful to develop and debug robot systems and controllers. Rviz provides a configurable Graphical User Interface (GUI) to allow the user to display only information that is pertinent to the present task.

3.4. System block diagram

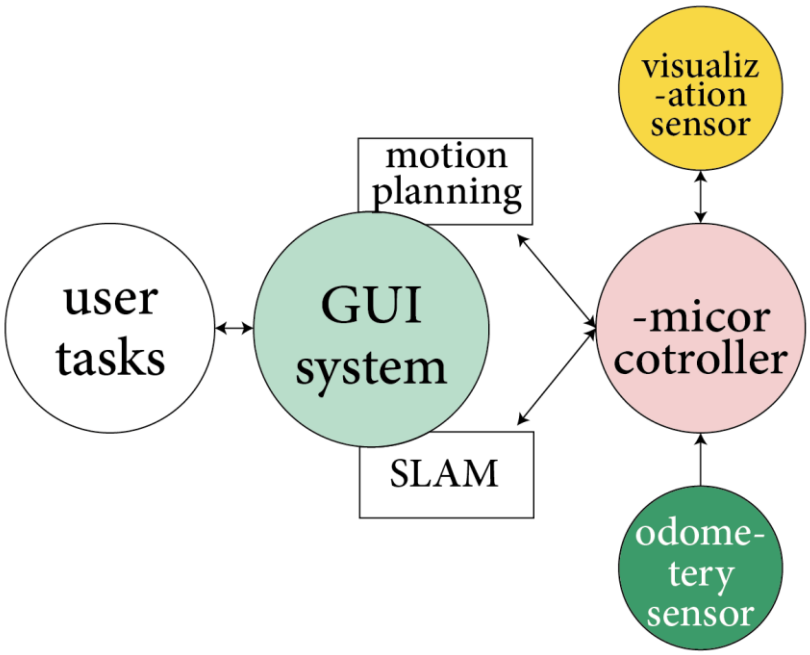


Fig 3.5:block diagram of proposed system (used components)

3.5. Robot model:

Within this section, we explain the kinematic and dynamic model of a non-holonomic robot that uses two differential wheels to move around the environment[18] . The kinematic model of a non-holonomic robot:

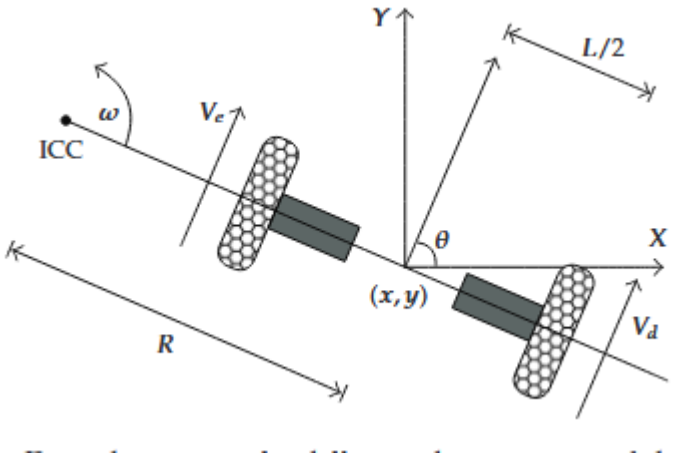


Fig 3.6 : Visualization of Differential Drive Robot and Key Variables

We assume that the robot is in one certain point (x, y) directed for a position throughout a line making an angle θ with x axis, as illustrated in Figure 3.6

$$ICC = [x - Rsin\,(\theta), y + Rcos\,(\theta)]$$

where ICC is the robot instantaneous curvature center. As ve and vd are time functions and if the robot is in the pose (x, y, θ)in the time t, and if the left and right wheel has ground contact speed ve and vd, respectively, then, in the time t → t δt the position of the robot is given by

$$[x'\ y'\ \theta'\ ] = [cos\,(\omega\delta t)\ -sin\,(\omega\delta t)\ 0\ sin\,(\omega\delta t)\ cos\,(\omega\delta t)\ 0\ 0\ 0\ 1\ ]\big[x - ICC_x\ y - ICC_y\ \theta\ \big] + \big[ICC_x\ ICC_y\ \omega\delta t\ \big]$$

Equation (3.2)describes the motion of a robot rotating a distance R about its ICC with an angular velocity given by ω

The forward kinematics problem is solved by integrating (3.2) from some initial condition (x0, y 0, θ 0,) it is possible to compute where the robot will be at any time t based on the control parameters vet and vdt . For the

special case of a differential drive vehicle, it is given by

$$x(t) = \frac{1}{2}\int_0^t [v_d(t) + v_e(t)]cos\,[\theta(t)]dt\; y(t)$$
$$= \frac{1}{2}\int_0^t [v_d(t) + v_e(t)]sin\,[\theta(t)]dt\; \theta$$
$$= \frac{1}{L}\int_0^t [v_d(t) - v_e(t)]dt$$

3.5.proposed system diagram

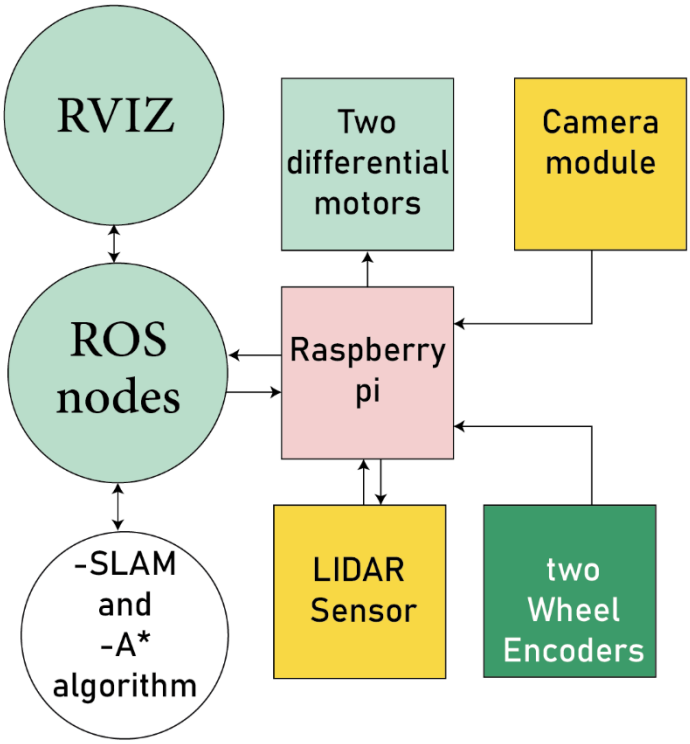


Fig 3.7 :proposed diagram of the software and hardware of the system

3.6.Robot module view:

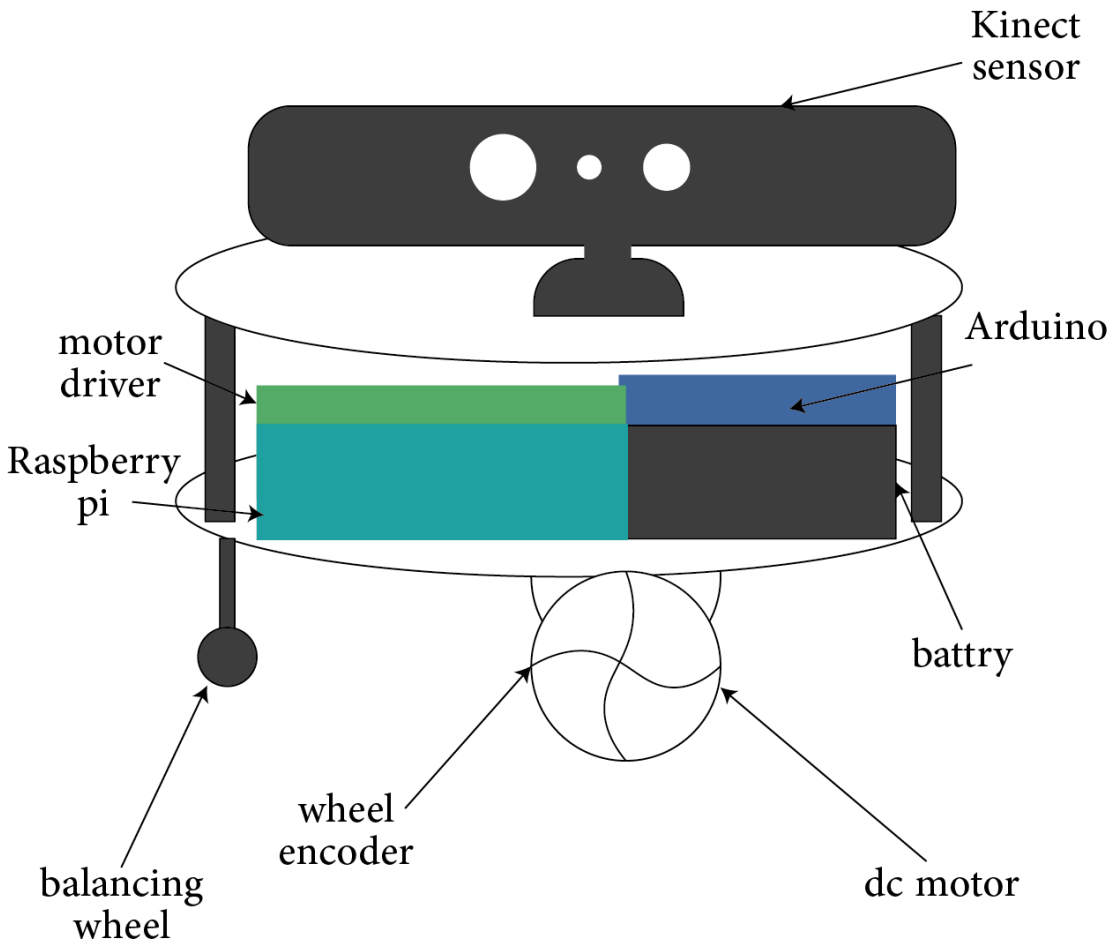


Fig 3.8 : shows module representation of our test robot

3.6.1.Module excepected scenario:

When the robot is turned on, the sensors on the robot such as the wheel encoders will gather information of the positioning of the robot. Moreover, landmarks from the environment are also extracted based on new observations from the LIDAR which is mounted on the robot. These new observations are associated with previous observations and updated in the SLAM algorithm However, if an observation of a landmark cannot be associated to a previous observation the observation itself is presented to the SLAM algorithm as a new observation , then the robot will decide its

behavior based on the task required , in case of search and rescue robots the robot will continue discovering the environment and recognize different objects and show those objects as points in the created map , if the robot couldn't recognize the objects the it will ignore them and continue considering our object recogniztion system is perfect.

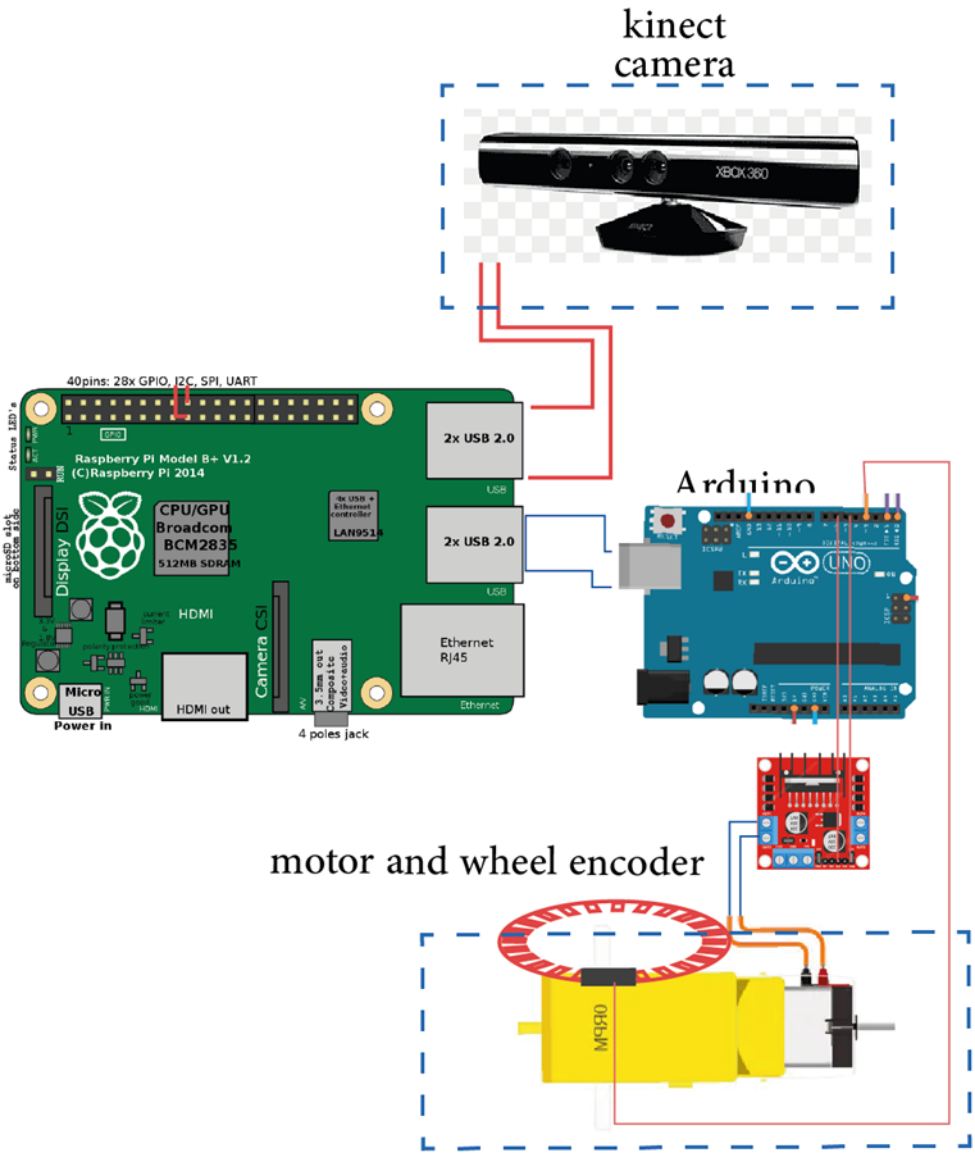


Fig 3.9:circuit digram proposed for master robot

the vehicle uses 5v stepper motor with uln2003 as Darlington circuit to provide power to motors
-the kinect sensor is used for 3D and 2D scan of environment
- number of ticks of stepper motor is used as wheel encoder for odometry measurement
- included rgb camera is the kinect sensor is used as visual feedback during mapping and navigation

CHAPTER FOUR
Our Test Robot

CHAPTER FOUR
Our Test Robot

this chapter will go through modelling and testing process , the robot model here is called “turtlebot “ which will simplify our test process due to its base differential movement model.

4.1.ROS setup:

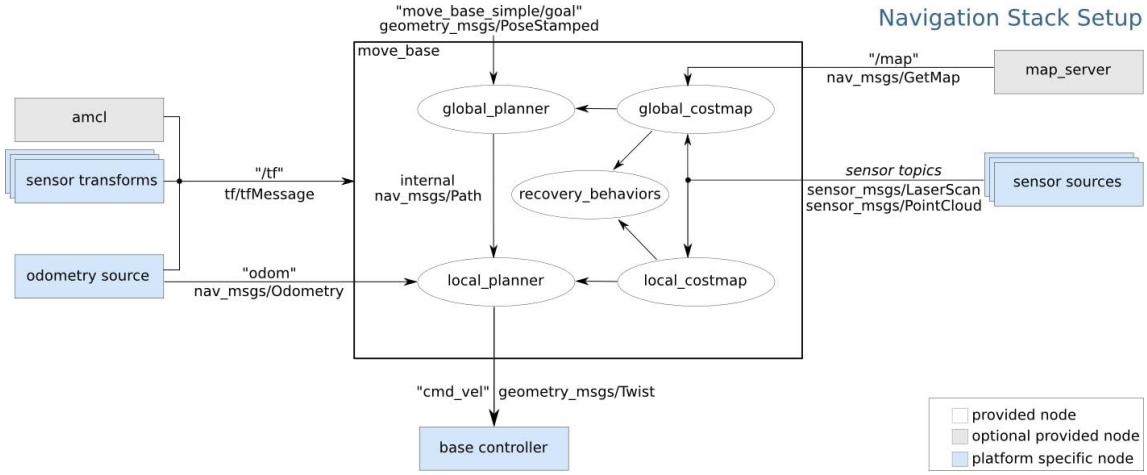


Fig 4.1:ROS navigation stack components

The navigation stack assumes that the robot is configured in a particular manner in order to run. The diagram above shows an overview of this configuration. The white components are required components that are already implemented, the gray components are optional components that are already implemented, and the blue components must be created for each robot platform. The pre-requisites of the navigation stack, are provided in the sections below.

4.1.1 Transform Configuration (other transforms):

The navigation stack requires that the robot be publishing information about the relationships between coordinate frames using tf which is basically the coordinate frame between the LIDAR and the base wheels.

4.1.2 Sensor Information (sensor sources)

The navigation stack uses information from sensors to avoid obstacles in the world, it assumes that these sensors are publishing either sensor_msgs/LaserScan or sensor_msgs/PointCloud messages over ROS , in our case the sensor information will be published as LaserScan from a LIDAR.

4.1.3 Odometry Information (odometry source)

The navigation stack requires that odometry information be published using tf and the nav_msgs/Odometry message which will gather information of speed of the wheel using wheel encoder.

4.2 Base Controller (base controller)

The navigation stack assumes that it can send velocity commands using a geometry_msgs/Twist message assumed to be in the base coordinate frame of the robot on the "cmd_vel" topic. This means there must be a node subscribing to the "cmd_vel" topic that is capable of taking (vx, vy, vtheta) <==> (cmd_vel.linear.x, cmd_vel.linear.y, cmd_vel.angular.z) velocities and converting them into motor commands to send to a mobile base for this top we will be using differential drive wheels using PID controller .

4.2.1Mapping (map_server)

The navigation stack does not necessary require a map to operate, but for the purposes of paper the robot will autonomously navigate through unknown environment in order to fulfil its task.

4.3.Coding:

4.3.1.Creating a Package

This first step for coding is to create a package where we'll store all the configuration and launch files for the navigation stack. This package will have dependencies on any packages used to fulfill the requirements in the Robot Setup section above as well as on the move_base package which contains the high-level interface to the navigation stack. So, pick the location of main directory ran the following command:

```
catkin_create_pkg my_ELEXBOT_2dnav move_base my_tf_configuration_dep my_odom_configuration_dep my_sensor_configuration_dep
```

4.3.2.Creating a Robot Configuration Launch File

Now that we have a workspace for all of our configuration and launch files, we'll create a roslaunch file that brings up all the hardware and transform publishes that the robot needs. So we created the the following launch file ELEXBOT_configuration.launch.

```
<launch>

  <node pkg="sensor_node_pkg" type="sensor_node_type" name="sensor_node_name" output="screen">
    <param name="sensor_param" value="param_value" />
  </node>
  <node pkg="odom_node_pkg" type="odom_node_type" name="odom_node" output="screen">
    <param name="odom_param" value="param_value" />
  </node>
```

```
        <node                    pkg="transform_configuration_pkg"
type="transform_configuration_type"
name="transform_configuration_name" output="screen">
          <param              name="transform_configuration_param"
value="param_value" />
        </node>

       </launch>
```

so now we have a template for a launch file, but we need to fill it in for our specific robot (ELEXBOT)

```
       <launch>

         <node       pkg="laserscan"       type="laser_scan_Node"
name="/scan" output="screen">
```

4.3.3.motion planning:

4.3.3.1.Costmap Configuration (local_costmap) & (global_costmap)

The navigation stack uses two costmaps to store information about obstacles in the world. One costmap is used for global planning, meaning creating long-term plans over the entire environment, and the other is used for local planning and obstacle avoidance. There are some configuration options that we'd like both costmaps to follow, and some that we'd like to set on each map individually. Therefore, there are three sections below for costmap configuration: common configuration options, global configuration options, and local configuration options.

4.3.3.2.Common Configuration (local_costmap) & (global_costmap)

The navigation stack uses costmaps to store information about obstacles in the world. In order to do this properly, we'll need to point the costmaps at the sensor topics they should listen to for updates. Let's create a file called costmap_common_params.yaml as shown below and fill it in:

4.4.Fast-SLAM package launch file:

This launch file will launch the gmapping package in configurations that fits our test robot

```
<node pkg="gmapping" type="slam_gmapping" name="slam_gmapping" output="screen">
        <param name="base_frame" value="/base_link"/>
        <param name="odom_frame" value="/odom"/>
        <param name="map_update_interval" value="5.0"/>
        <param name="maxUrange" value="6.0"/>
        <param name="maxRange" value="8.0"/>
        <param name="sigma" value="0.05"/>
        <param name="kernelSize" value="1"/>
        <param name="lstep" value="0.05"/>
        <param name="astep" value="0.05"/>
        <param name="iterations" value="5"/>
        <param name="lsigma" value="0.075"/>
        <param name="ogain" value="3.0"/>
        <param name="lskip" value="0"/>
        <param name="minimumScore" value="200"/>
        <param name="srr" value="0.01"/>
        <param name="srt" value="0.02"/>
        <param name="str" value="0.01"/>
        <param name="stt" value="0.02"/>
        <param name="linearUpdate" value="0.5"/>
        <param name="angularUpdate" value="0.436"/>
        <param name="temporalUpdate" value="-1.0"/>
        <param name="resampleThreshold" value="0.5"/>
```

```
<param name="particles" value="80"/>
```

4.4.2.Fast-SLAM flowchart:

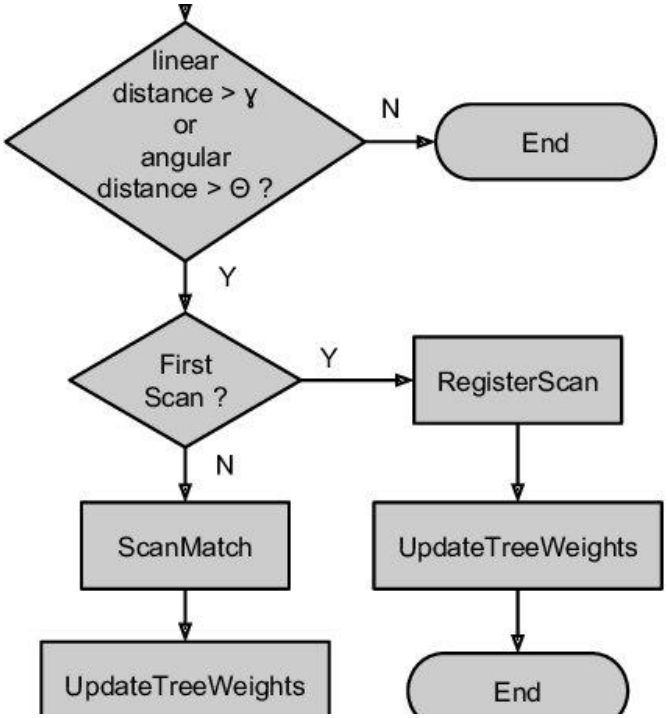


Fig4.2:Fast-SLAM flowchart

4.4.2.Fast-SLAM rsults:

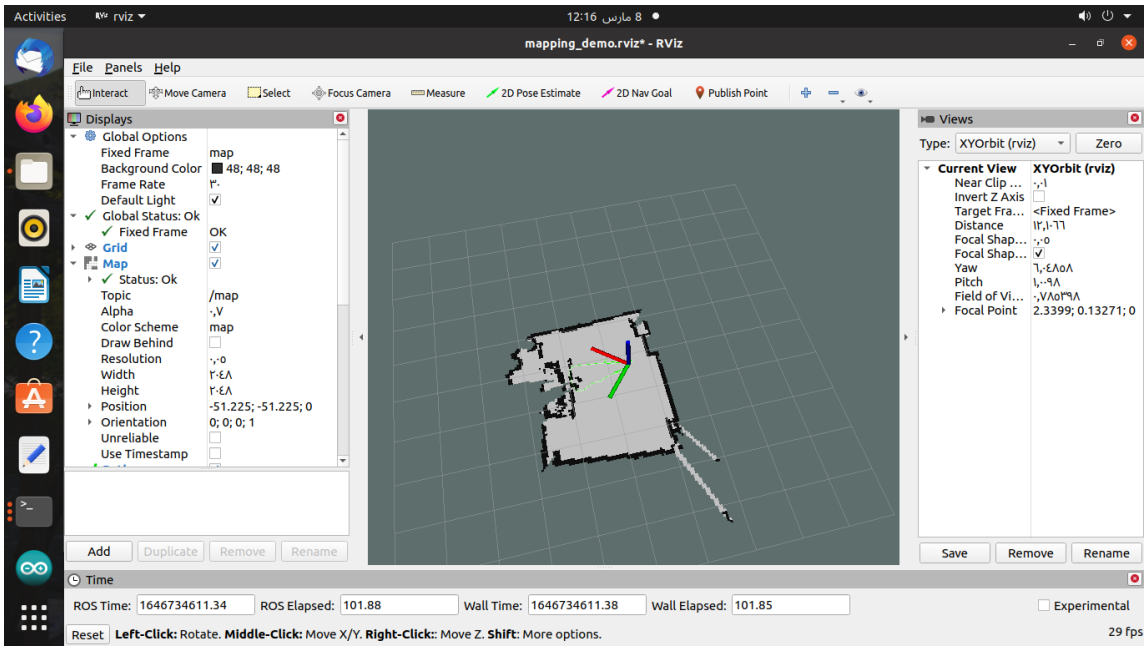


Fig4.3:Fast-SLAM created map show the robot pose around its environment

4.5.hector_SLAM launch file:

```
<arg name="geotiff_map_file_path" default="$(find hector_geotiff)/maps"/>

  <param name="/use_sim_time" value="false"/>

  <node pkg="rviz" type="rviz" name="rviz"
        args="-d $(find hector_slam_launch)/rviz_cfg/mapping_demo.rviz"/>

  <include file="$(find hector_mapping)/launch/mapping_default.launch"/>

  <include file="$(find hector_geotiff_launch)/launch/geotiff_mapper.launch">
        <arg name="trajectory_source_frame_name" value="scanmatcher_frame"/>
        <arg name="map_file_path" value="$(arg geotiff_map_file_path)"/>
```

4.5.2.Hector SLAM flowchart:

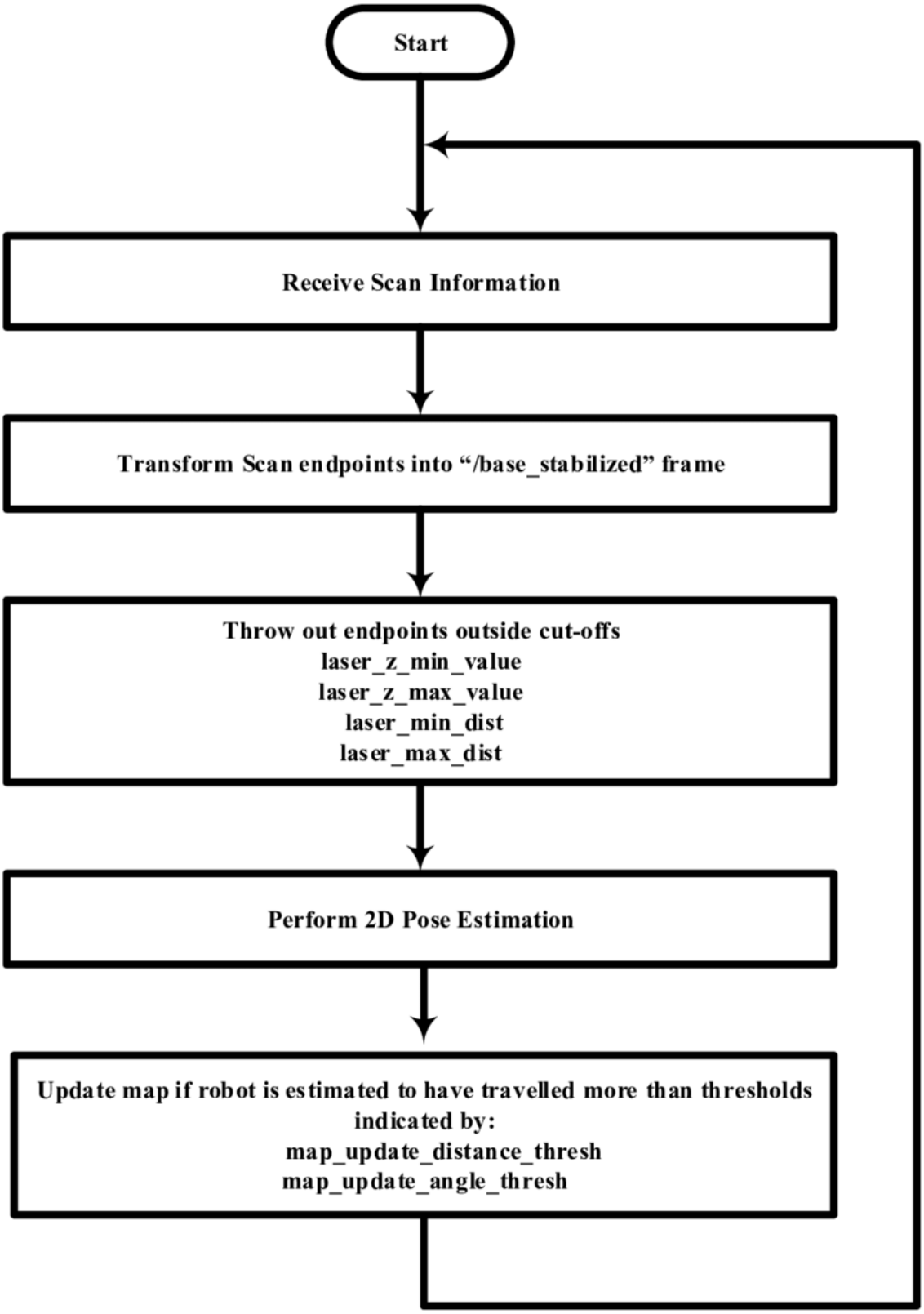


Fig.4.4:hector SLAM flowchart

4.5.2:hector SLAM results:

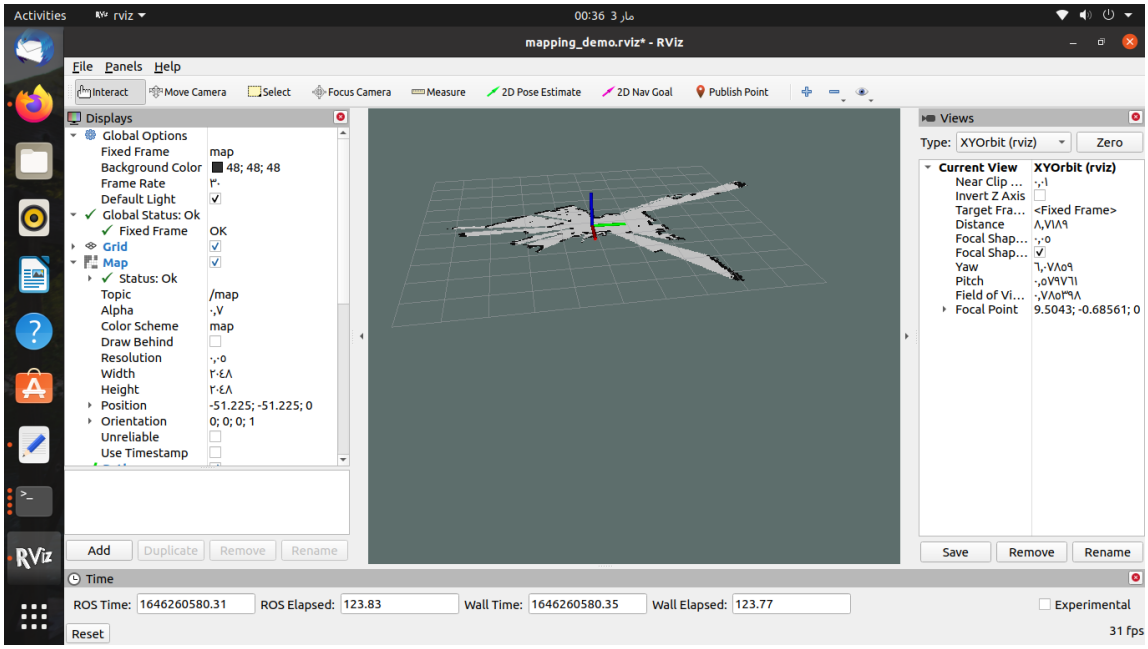


Fig 4.5: hector SLAM

4.6.RTABMAP launch file:

```
<node unless="$(arg stereo)" pkg="rtabmap_ros" type="rgbd_odometry"
name="rgbd_odometry" clear_params="$(arg clear_params)" output="$(arg output)"
args="$(arg rtabmap_args) $(arg odom_args)" launch-prefix="$(arg launch_prefix)">
        <remap from="rgb/image"     to="$(arg rgb_topic_relay)"/>
        <remap from="depth/image"  to="$(arg depth_topic_relay)"/>
        <remap from="rgb/camera_info" to="$(arg camera_info_topic)"/>
        <remap from="rgbd_image"  to="$(arg rgbd_topic_relay)"/>
        <remap from="odom"          to="$(arg odom_topic)"/>
        <remap from="imu"           to="$(arg imu_topic)"/>

        <param name="frame_id"              type="string" value="$(arg frame_id)"/>
        <param name="odom_frame_id"             type="string" value="$(arg
vo_frame_id)"/>
        <param name="publish_tf"            type="bool"   value="$(arg
publish_tf_odom)"/>
        <param name="ground_truth_frame_id"      type="string" value="$(arg
ground_truth_frame_id)"/>
        <param name="ground_truth_base_frame_id"  type="string" value="$(arg
```

```
ground_truth_base_frame_id)"/>
        <param name="wait_for_transform_duration" type="double" value="$(arg
wait_for_transform)"/>
        <param name="wait_imu_to_init"            type="bool"   value="$(arg
wait_imu_to_init)"/>
        <param name="approx_sync"                 type="bool"   value="$(arg
approx_sync)"/>
        <param name="config_path"           type="string" value="$(arg cfg)"/>
        <param name="queue_size"            type="int"       value="$(arg
queue_size)"/>
        <param name="subscribe_rgbd"              type="bool"   value="$(arg
subscribe_rgbd)"/>
        <param name="guess_frame_id"              type="string" value="$(arg
odom_guess_frame_id)"/>
        <param name="guess_min_translation"       type="double" value="$(arg
odom_guess_min_translation)"/>
        <param name="guess_min_rotation"          type="double" value="$(arg
odom_guess_min_rotation)"/>
        <param name="expected_update_rate"        type="double" value="$(arg
odom_expected_rate)"/>
        <param name="max_update_rate"             type="double" value="$(arg
odom_max_rate)"/>
        <param name="keep_color"            type="bool"   value="$(arg
use_odom_features)"/>
        </node>
```

4.6.2.RTABMAP flowchart:

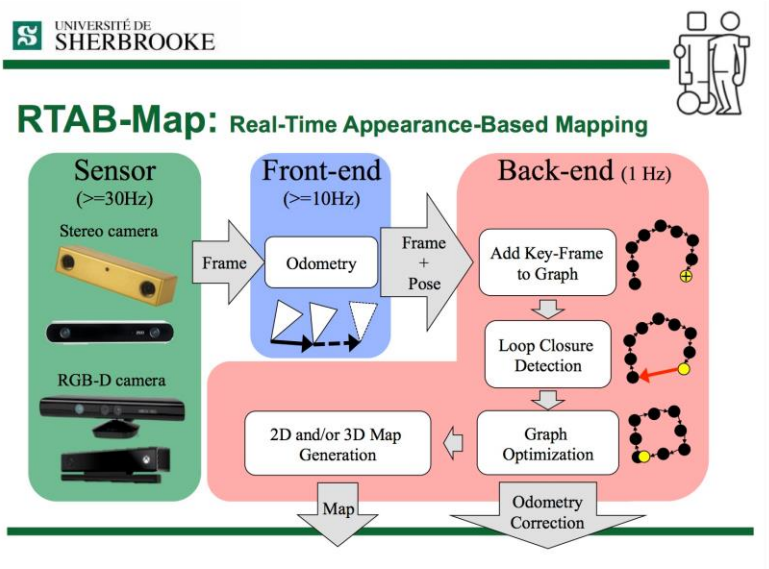


Fig.4.6:RTABMAP flow chart

4.6.2.RTABMAP 3D map results:

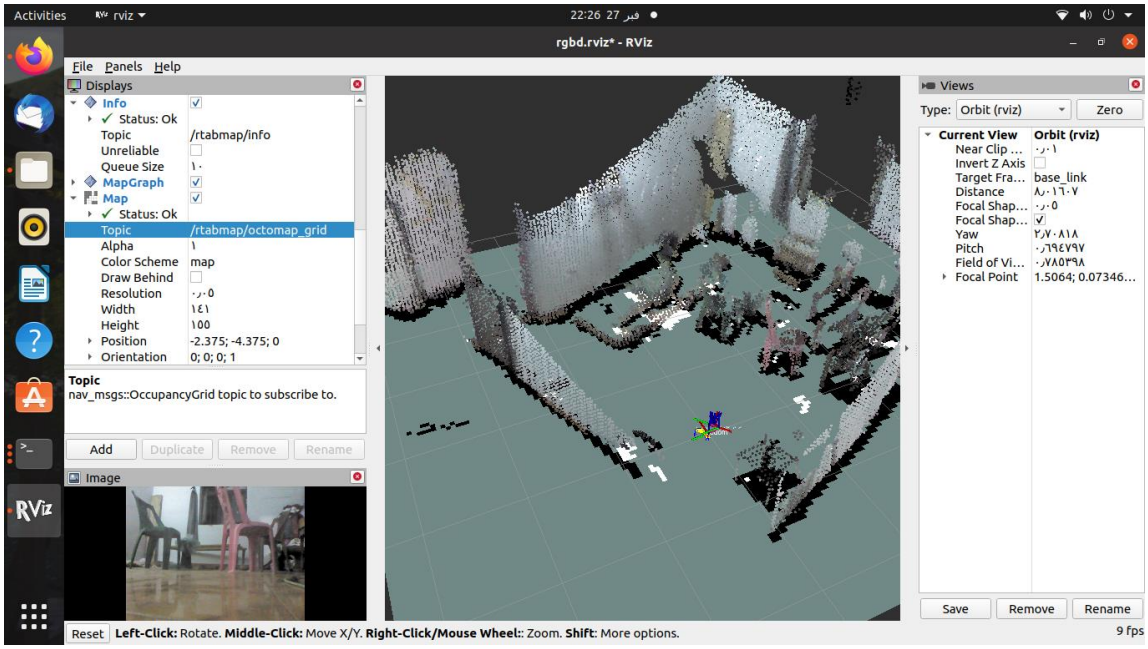


Fig 4.7: RTAB-SLAM 3D MAP

4.6.3.RTABMAP projection map results:

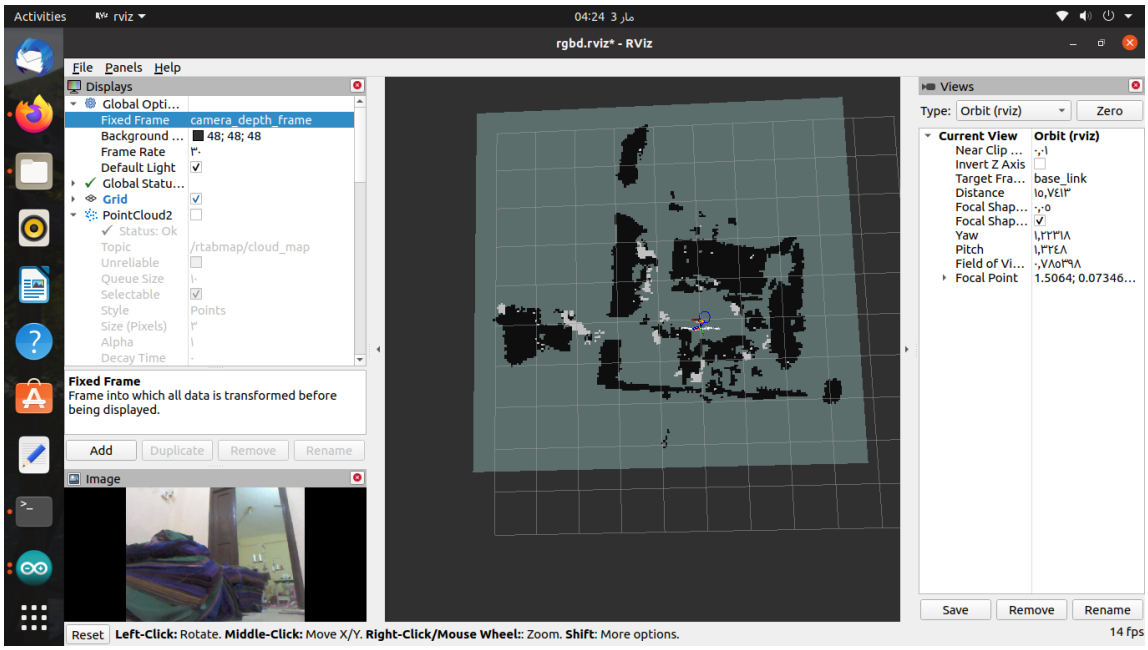


Fig 4.8: RTAB-SLAM Projection map

CHAPTER FIVE

Conclusion And Recommendation

CHAPTER FIVE

Conclusion And Recommendation

5.1. Conclusion:

In this paper, we have presented a small size vehicle localization and mapping-based comparison of three SLAM algorithms,
and within both of indoor and outdoor environment. fast-SLAM , hector SLAM and RTAB SLAM were successfully implemented and test with a reasonable map quality and control and communication latency , however RTAB SLAM suffered from some lagging and more latency due to the fact that it uses much CPU and GPU power that caused heating and overloud in our on-board computer (Raspberry PI) . The trajectory estimation by RTABMAP is accurate but the odometry estimation is not. In order to improve the quality of localization and mapping obtained by the RTAB-SLAM, we have to configure the RTAB-SLAM to our system by taking into account some parameters ; this is done by reducing the resolution of the map and reducing the update frequency of the sensor readings to match the processing power of the ARM Cortex CPU of the raspberry pi . We have tested the Fast-SLAM and hector-SLAM using fake laser scans obtained by converting depth image to laser scan by using depth_to_laserscan package from ROS , the Fast-SLAM also takes into account the odometer of the robot , Odometry is calculated using wheel encoder rotary movement sensed from each wheel of the differential drive model separately and then published in ROS topic list which will be used by different ROS packages and services , this odometry publishing operation is done in two different parts : controller layer (Arduino) (Appendix A) ,and ROS layer using C

code (Appendix E)

5.2.Recommendation:

The first observation of the vehicle's performance is the lack of sensed data from laser scanner due to narrow range of sight of Kinect sensor used , this issue can be overcame by using one of the high range 360 degree laser range finder (LIDARs) , this will significantly increase the performance of Fast-SLAM and hector-SLAM algorithms , however the problem of RTAB-SLAM can't be solved because it uses Depth image and RGB image (kwon as RGBD) , in the RTAB-SLAM context the simple short solution is to use more capable on-board computer that has much GPU processing speed (NVidia jetson nano for example)

5.3.Reference:

[1]Gustafsson, Fredrik Statistical sensor fusion Linköping University, Department of Electrical Engineering, Automatic Control. Linköping University, The Institute of Technology.

[2]Multi-Sensor Fusion: A Perspective Jay K . Hackett ‘and Mubarak Shah Computer Science Department University of Central Florida Orlando, FL 32816

[3] Robert Lundh and Lars Karlsson and Alessandro Saffiotti Plan-Based Configuration of a Group of Robots Proceedings of the 2006 conference on ECAI 2006: 17th European Conference on Artificial Intelligence August 29 -- September 1, 2006, Riva del Garda, ItalyMay 2006

[4] Robotics & Automation Taipei, Taiwan, September 14-19, 2003 Design and Implementation of MARG Sensors for 3-DOF Orientation Measurement of Rigid Bodies Eric R. Bachmannl, Xiaoping Yun2, Doug McKinney’, Robert B. McGhee, and Michael J. Zyda The MOVES Institute Naval Postgraduate School Monterey, CA 93943 yun@n,nas.navy.mil

[5] SENSOR FUSION J.Z. Sasiadek Department of Mechanical & Aerospace Engineering, Carleton University e-mail: jsas@mae.carleton.ca

[6] M. W. M. Gamini Dissanayake, Paul Newman, Steven Clark, Hugh F. Durrant-Whyte, and M. Csorba. A solution to the simultaneous

localization and map building (slam) problem. IEEE Transactions on Robotics and Automation, 17(3):229–241, 2001. Cited on pages 2 and 15.

[7] Zhang Xuex , 2019 SLAM Algorithm Analysis of Mobile Robot Based on Lidar . China . Guangdong University of Technology

[8] Axel Barrau, Silvere Bonnabel .2015 An EKF-SLAM algorithm with consistency properties . ParisTech, PSL Research University .

[9] Michael Montemerlo and Sebastian Thrun 2002 . FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem . Carnegie Mellon University

[10] Chanki Kim, R. Sakthivel and Wan Kyun Chung . 2007 . Unscented FastSLAM: A Robust Algorithm for the Simultaneous Localization and Mapping Problem . IEEE International Conference

[11] Sebastian Thrun . 2006 . The Graph SLAM Algorithm with Applications to Large-Scale Mapping of Urban Structures . The International Journal of Robotics Research

[12] Gamini Dissanayake, Hugh Durrant-Whyte and Tim Bailey . 2006 . A Computationally Efficient Solution to the Simultaneous Localisation and Map Building (SLAM) Problem . University of Sydney

[13] RTAB-MAP : http://introlab.github.io/rtabmap/ [visited 2022/3/8]

[14] A* algorithm : https://www.geeksforgeeks.org/a-search-algorithm/ [visited 8/9/2021]

[15] A* Figure https://www.geeksforgeeks.org/a-search-algorithm/ [visited 8/9/2021]

[16]odom_ros_wiki:http://wiki.ros.org/navigation/Tutorials/RobotSetup/Odom [online ; accessed 20-9-2021]

[17] tf - ROS Wiki: http://wiki.ros.org/tf. [Online; accessed 20-9-2021]

[18] Trajectory Planning for Nonholonomic Mobile Robot Using Extended Kalman Filter Leonimer Flavio de Melo and Jose Fernando Mangili Junior Department of Electrical Engineering, State University of Londrina, Londrina, PR, Brazil