TrackerBot: Object Tracking and Collision Avoidance using UGV

Major Project submitted to

Amity School of Engineering and Technology , Amity University

in partial fulfilment of the requirements for

The degree of

Bachelor of Technology

from

Department of Computer Science and Engineering,

Amity School of Engineering and Technology

by

Madhushree Sannigrahi

under

Dr. Pushan Kumar Dutta

Amity School of Engineering and Technology

AMITY UNIVERSITY

Kolkata-700135

West Bengal, India

May 2023

Certificate

This is to certify that the project work entitled "TrackerBot: Object Tracking and Collision Avoidance using UGV" has been prepared by "Madhushree Sannigrahi" under my supervision and guidance. The thesis is her original work completed after careful research and investigation. The work of the thesis is of the standard expected of a candidate for the Bachelor of Technology Programme in Artificial Intelligence and I recommend that it be sent for evaluation.

_____

Name and Signature of the Guide

_____

Date:

Name and Signature of the Head of the Department

Forwarded By:

_____

Name and Signature of the Head of Institution

Declaration

I, **Madhushree Sannigrahi**, registered as an undergraduate scholar, bearing Enrollment Numbers **A910119819007** for the Bachelor of TechnologyProgramme under the Amity School of Engineering and Technology of Amity University do hereby declare that I have completed the requirements as per mentioned by the university for project submission.

I do hereby declare that the project submitted is original and is the outcome of the independent investigations /research carried out by me and contains no plagiarism. The work is leading to the discovery of new techniques. This work has not been submitted by any other University or Body in quest of a degree, diploma or any other kind of academic award.

I do hereby further declare that the text, diagrams or any other material taken from other sources (including but not limited to books, journals and the web) have been acknowledged, referred and cited to the best of my knowledge and understanding.

Date:

_____

Signature of Student

Abstract

Name of the student: **Madhushree Sannigrahi**

Enroll No: **A910119819007**

Degree for which submitted: **B.Tech (AI)** Department: **CSE Department**

Project title: **TrackerBot: Object Tracking and Collision Avoidance using UGV**

Project Guide: **Dr. Pushan Kumar Dutta**

Month and year of project submission: **May 2023**

Autonomous rovers are mobile robots, designed to operate in various conditions without any human intervention. These rovers are equipped with sensors, actuators, and control systems that enable them to navigate and interact with their environment. They are widely employed in fields like exploration, search and rescue, environmental monitoring, and surveillance. This research presents the construction and development of an autonomous rover that integrates parallel object tracking, videography, and collision avoidance. The proposed system is designed to operate in real-world scenarios where the rover can navigate safely and capture high-quality visual data of its targeted moving object which can be a living or non-living thing. The rover's control system is based on a combination of classical controllers such as PID controller accompanied by computer vision techniques, enabling it to detect and track objects at a parallel level in its environment, avoid obstacles and collisions, and record video footage with minimal hardware requirements. The proposed system's performance will be evaluated through extensive testing, where the rover's ability to navigate complex environments and capture high-quality visual data will be assessed.

Acknowledgements

Words cannot express my gratitude to my professor Dr. Pushan Kumar Dutta for his invaluable patience and feedback. Without the assistance of Amity University Kolkata, which provided me with this research opportunity. Additionally, this endeavour would not have been possible without the generous support from the Indian Institute of Science Education and Research, Bhopal who kindly gave me the knowledge and experience I needed to conduct this project in the niche area of automated vehicles. Without them, I would not have been able to go on this adventure.

I am also grateful to the professors of Amity University Kolkata, especially Dr. Pushan Kumar Dutta and my classmates from my University, for their help, feedback sessions, and moral support. Lastly, I would be remiss in not mentioning my family, especially my parents. Their belief in me has kept my spirits and motivation high during this process.

Contents

Acknowledgements v

List of Figures viii

Abbreviations x

1 Introduction 1
1.0.1 Key Contributions: . . . 3

2 Related Works 5

3 Theoretical Background 8
3.1 The ROS System . . . 8
3.1.1 Node: . . . 8
3.1.2 Messages: . . . 9
3.1.3 Topics: . . . 9
3.1.4 roscore: . . . 10
3.2 Differential Drive . . . 11
3.3 Kalman Filter . . . 13
3.4 PID Controller . . . 16
3.5 YOLO . . . 18

4 Methodology 20
4.1 Mechanical Subsystem . . . 20
4.1.1 Chassis & Wheels . . . 20
4.1.2 Motors(200 rpm 4-6V DC): . . . 20
4.1.3 L298N Driver: . . . 23
4.1.4 Camera & Sensors . . . 24
4.1.4.1 WEBCAM (HP w100 480P 30 FPS Digital Webcam) . . . 24
4.1.4.2 LM393 SPEED SENSOR (Wheel encoder): . . . 25
4.1.4.3 Ultrasonic Sensors . . . 26

4.1.4.4 Wifi Adapter for Networking . . . 27
4.1.5 Power System . . . 28
4.1.6 Micro-controller & Onboard Computer . . . 28
4.1.6.1 Arduino . . . 28
4.1.6.2 Jetson Nano . . . 30
4.2 Motions and Navigations . . . 33
4.3 Vision, Tracking and Following . . . 35
4.3.1 Networking . . . 35
4.3.2 CV - Pipeline . . . 36
4.3.3 Navigation - Pipeline . . . 37
4.4 Obstacle Detection . . . 38

5 Result and Discussion **40**
5.1 Simulation . . . 40
5.2 Real World Implementation . . . 43

6 Conclusion **45**
6.1 Future Works . . . 46

Bibliography **48**

List of Figures

1.1 A few examples of Autonomous Ground Vehicles [1] 2

3.1 The ROS System 8
3.2 ROS Nodes with several packages and interactions between them [1] 9
3.3 ROS msg command example 10
3.4 ROS Architecture with topics and nodes [1] 10
3.5 roscore connection with other nodes in the system [1] 11
3.6 ROS Network Main (master node) system and Subsystems (slave nodes) [1] 11
3.7 (A)A differential drove robot with tri-wheel system (B)Course of wheels during the turn.[2] 12
3.8 Kinematics of Differential Driven Wheels [2] 14
3.9 A block diagram of a PID controller in a feedback loop. r(t) is the desired process value or setpoint (SP), and y(t) is the measured process value (PV)[3]. 17
3.10 The overall YOLOv3 architecture [4] 18

4.1 The bare chassis of Automated Ground Vehicle 21
4.2 The overall look of the self-built Automated Ground Vehicle (A)Front-view (B)Rear-view (C)Side-view 1 (D)Side-view 2 (E)Top-view 22
4.3 L298N Motor Driver[5] 23
4.4 HP w100 480P 30 FPS Digital Webcam 24
4.5 LM393 IR speed sensor module 26
4.6 (A) HC-SR04 Sensors (B) Working Principle of Ultrasonic Sensors [1] . . . 27
4.7 802.11n Wifi Adapter [5] 28
4.8 The overall power supply chain in the AGV 29
4.9 Arduino UNO PIN Configurationc[6] 30
4.10 Jetson Nano by Nvidia[7] 31
4.11 Labelled Schematic image of Jetson Nano by Nvidia[7] 32
4.12 Detailed diagram of the basic Navigation module 33
4.13 The overall schematic architecture of the Autonomous Ground Vehicle . . . 34
4.14 IP address Configuration of Jetson Nano 35
4.15 YOLO is running on the darknet-ros node and detecting everyday objects. . 36
4.16 The overall ros architecture for the UGV 37
4.17 The overall schematic architecture of the Obstacle Detection module in Autonomous Ground Vehicle 39

5.1 Simulated Environment in Webots using "Epuck" and "Pioneer" Robots . . 40

5.2 Covariance graph and Position-Velocity graph of Kalman filter from the simulated environment. 41
5.3 The initial prototypes of the AGV. 42
5.4 (A) Bounding box node publishing X axis position and velocity, (B) Kalman Filter Node which has been subscribed to bounding boxes publish refined output of X - position and velocity w.r.t. centre point of the camera. 43
5.5 ROS nodes and topics in RQT graph 44

Abbreviations

AGV	Autonomous Ground Vehicle
USV	Unmanned Surface Vehicle
AUK	Amity University Kolkata
YOLO	You Only Look Once
ROS	Robot Operating System
PID	Proportional-Integral-Derivative
NMS	Non-Maximum Suppression
RPM	Revolutions Per Minute
DC	Direct Current
PWM	Pulse Width Modulation
FPS	Frames per Second

Chapter 1

Introduction

Robots are programmable physical devices with sensors, actuators, and instructions for what they are to do in the environment. The sensor inputs are processed by perception algorithms, a control program determines how the robot should behave in light of its objectives and the situation at hand, and then orders are issued to the robot's motors to cause it to operate in the outside world [8]. While some robots can move around, others are stuck in one particular spot.

Autonomous mobile robots are now a key area of study and development in the science of robotics, which has made impressive strides in recent years[9]. An autonomous mobile robot is a self-contained machine that can sense its surroundings, make choices, and carry out activities without the direct involvement of a person. The design, development, and use of an autonomous mobile robot are explored in depth in this project report, along with the underlying technologies that make it possible 1.1.

A robot is a device that mimics the behaviour or outward appearance of an intelligent being—typically a human—and absorbs information about its surroundings through its senses before using that data to think and behave as directed. So a machine has to be capable of two things in order to be considered a robot:

1. Gather information about the area.
2. Engage in physical activity, such as moving or manipulating items.

The majority of robots have motorised wheels, a movable body and movable metal or plastic parts. Joints connect the portions together. In addition to hydraulic and pneumatic

FIGURE 1.1: A few examples of Autonomous Ground Vehicles [1]

systems, actuators for the robots include solenoids, electric motors, and hydraulic systems. All robots have the characteristics of a mechanical, moveable structure under some type of autonomous control [10], despite the fact that their appearances and functionalities differ.

Robots that can complete desired tasks in unstructured conditions without ongoing human supervision are called autonomous robots. High levels of autonomy are especially desired in industries like wastewater treatment, floor cleaning, mowing lawns, and space exploration. A completely autonomous robot is able to move all or part of itself across its operational environment without human aid, as well as:

- Learn about its surroundings.
- Work for a lengthy amount of time without human supervision.
- Gather information about its surroundings.
- Steer clear of circumstances that endanger individuals, property, or the situation itself.

The main goal of this project is to build an intelligent robot that can independently explore its environment, detect impediments, and carry out a variety of predetermined tasks. The robot can interact with its surroundings and make decisions that resemble

those of humans by combining cutting-edge technology like computer vision, artificial intelligence, and sensor integration.

This system runs on autopilot. This robot transmits a request to the controllers after receiving it from a computer. The controllers serve as the robot's brain and communicate with the motors to move [11]. The webcam receives a signal from the controllers to begin operating and analysing its surroundings. Unless there is a barrier in the way, the robot begins to advance. When the robot encounters an impediment, it pauses, and sends a signal to the controllers, and the controllers modify the robot's course by sending a signal to the motors. So, even though it can be physically operated from a great distance, as a remote-controlled robot, it can operate for a long period of time without any human supervision. Differential motors, which both revolve at various rates, are used to change the robot's direction.

An overview of the design process is given in the paper, along with details on the selection and integration of hardware parts, the development of a solid software architecture, and the implementation of algorithms for perception, decision-making, and control. It explores the problems that were faced during the project and the methods used to solve them, emphasising the value of iterative design and testing. The paper also goes over the many features and functionalities included in the transportable, autonomous robot. Map-making and localization, obstacle avoidance and detection, route planning and navigation, object manipulation, and human-robot interaction are a few examples of these. Each of these skills improves the robot's total autonomy and increases its utility in practical applications. The research also discusses the possible uses and advantages of autonomous mobile robots in a variety of fields. These robots have the potential to revolutionise sectors and enhance human skills in everything from industrial automation and logistics to healthcare and search and rescue missions. The paper looks at a few of these application areas and offers convincing case examples that illustrate how autonomous mobile robots may help with efficiency and difficult problem-solving.

1.0.1 Key Contributions:

1. A novel framework for combining videography, computer vision, navigation, and tracking is proposed in this research.
2. One-of-its-kind tracking system, using minimal sensors.
3. The rover's custom hardware was made up of both pre-built and recycled and re-purposed parts.

4. Special drivers have been developed for the hardware used in the rover which may be implemented in different forms as well other than AGVs.

5. The implemented algorithm parameters have all been fine-tuned specifically for this hardware component.

6. Low-cost and sustainable manufacturing of autonomous vehicles can be facilitated by the project. It provides insights into the future of robots and the vast opportunities that lie ahead by probing the design procedure, technological considerations, and prospective applications.

We hope that this initiative will progress autonomous robots and encourage more investigation and invention in this fast-paced area.

Chapter 2

Related Works

In recent years, with the deepening of the trend of Industry 4.0, the manufacturing industry has begun to develop the trend of customization, individuation and flexibility, which poses a severe challenge to the fixed production mode of traditional robots.

In the conventional approaches of obstacle avoidance, it is not easy for robots and humans to work safely in the common unstructured environment due to the lack of intelligence. In this system, one Kinetics is employed to monitor the workspace of the robot and detect anyone who enters the workspace of the robot. By using this active collision avoidance, the system can achieve the purpose that the robot is unable to touch the human[12]. This proposed system highlights the advantage that during the process, it can first detect the human, then analyze the motion of the human and finally safeguard the human. The human-machine collaboration feature will promote the wider use of robots and promote robots to play an indispensable partner role in human life [10]. The physical boundary between humans and robotic manipulators is no longer necessary in new industrial robotics paradigms [13]. Additionally, considerable cooperation between humans and robots is anticipated in order to maximise output. Common motion generation techniques may be insufficient for this goal in this scenario, which involves a shared environment between people and robots. In a study by Yu et al., an interactive control method for a gait rehabilitation robot is presented [14]. The new compact series elastic actuator that powers the robot offers intrinsic compliance and back-driving capability for secure human-robot interaction. However, according to the "Three Laws of Robotics," the robot must not harm human beings or sit and watch human beings be harmed. It can be seen how important the role of robot security in the industrial development process is. While the physical human-robot interface necessitates face-to-face connection with people displaying

flexible, obedient behaviour, cognitive human-robot interaction is based on perception and awareness. It is therefore inevitable to design the optimum safe robot manipulator with adjustable compliant actuation [15]. High inherent safety is enabled by fluid mechanics, and the interaction control strategy is easier than with other concurrent approaches. These investigations have outlined three crucial motion types needed for physical. [16]

Most industrial robots have been placed in a static state, which is to determine an "isolated" environment and accomplish a single repetitive task as portrayed by Lee et al. [17]. Space is often isolated from people by very strong fences. However, when a robot performs work in dynamic, unstructured environments or even environments with humans, it has been a more challenging issue that how the robot works safely and efficiently as described by Nozaki et al. [18]. Karami et. al. discussed the future robotics applications where humans and robots collaborate in carrying out tasks together, achieving safe and efficient human-robot interaction is indispensable [19]. There are also a few researches like Sadrfaridpour and Wang [20], which have been done to develop robot systems, including interaction and cooperation with humans in daily life. It is indicated that robots could be very useful to accomplish various tasks with humans jointly in this research. According to Flacco et al., there are three parts to active collision avoidance in real-time: (1) Environment perception; (2) Collision avoidance algorithm; (3) Robot control [21]. Collision avoidance is considered as one of the core technologies in robot research, which draws the widespread attention of scholars.

Instead of markers, a ToF (time-of-flight) camera was adopted by Schiavi et al.[22] for collision detection, and an approach using multiple 3D depth images were presented in Fischer and Henrich [11] for the same purpose. In Flacco et al.'s method, a new integrated approach was proposed to avoid real-time collision by using one Kinetic sensor and also raised depth space approach [23]. The key contribution is a quick approach based on the idea of depth space to measure distances between a robot and potentially moving objects (including people). Inverse kinematics of robot manipulators must account for the existence of joint velocity and acceleration limits in order to prevent improper task execution when these are breached. In order to address this issue, a brand-new algorithmic strategy for kinematic control is introduced in his study by Antonelli et al. [24], which ensures that the joint variables do not exceed their bounds. The suggested method is based on a fresh second-order inverse kinematics algorithm, which makes it possible to handle velocity and acceleration restrictions while following the required end-effector route.

As the robot performs a general motion job, the distances are used to create repulsive vectors that are used to drive the robot. The repulsive vectors can also benefit from the

knowledge of the velocity of the obstacle. In order to increase positioning accuracy in robot manufacturing and maintenance, robot calibration is a helpful diagnostic technique. Similarly, By concentrating on the objective of pushing an object, the foot placement is planned in real-time based on the outcome of object manipulation. The impedance control of the arms allows the humanoid robot to push an object steadily regardless of its bulk [25]. Furthermore, Harade et al. suggested a novel method for humanoid robot manipulation. By concentrating on the objective of pushing an object, the foot placement is planned in real-time based on the outcome of object manipulation [26]. The impedance control of the arms allows the humanoid robot to push an object steadily regardless of its bulk. The humanoid robot pushes heavy objects by walking slowly, and vice versa. However, Minguez et al. filled the gaps left by the original avoidance method by taking into account the collision avoidance layer's precise form, kinematics, and dynamics [27].

The novel work by Du et al. [28] presents a vision-based online robot calibration method that simply needs a few reference images, as opposed to conventional calibration methods that demand expensive hardware and intricate stages. The technique needs a camera that is firmly fastened to the robot's end effector (EE), and a calibration board that is placed so the camera can view it around the robot.

Chapter 3

Theoretical Background

3.1 The ROS System

ROS systems consist of numerous independent programs that communicate with each other through message exchanges 3.1. The entire system can be depicted as a graph, where the programs are the nodes and the edges are the messages linking them. The instance portrayed in Figure [] is the graph related to a simple teleop task, in which the robot is remotely controlled with the keyboard and can be moved around. The nodes in ROS are represented by oval-shaped blocks and correspond to the processes that carry out computations. On the other hand, the rectangular-shaped blocks represent topics, which serve as channels or buses for message exchange (represented by the edges in the graph).

3.1.1 Node:

A node in robot applications is a process in charge of performing computations that regulate every component of the robot. As an example, there are nodes that control the robot's wheel motors, localization, and other functions (3.2). The command "rosnode

FIGURE 3.1: The ROS System

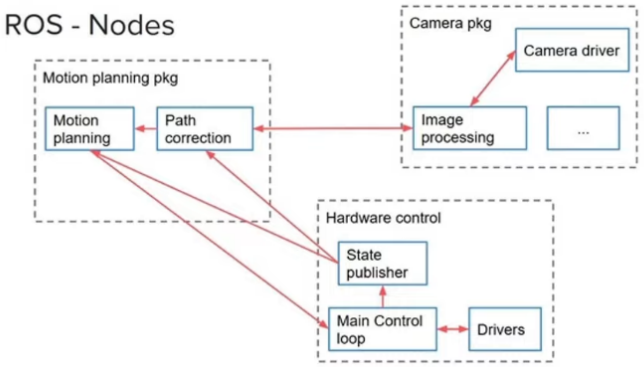

FIGURE 3.2: ROS Nodes with several packages and interactions between them [1]

info" followed by the name of the node is a useful tool for learning more about nodes. It displays which publishers and subscribers are linked to that node. Each node uses messages to communicate within the ROS network.

3.1.2 Messages:

ROS messages are data structures used for communication between nodes in the Robot Operating System (ROS). They define the structure and types of information being passed between nodes. Messages contain fields representing different data types such as integers, floats, booleans, strings, arrays, or nested message types (3.3). Messages are defined using a text-based language and allow nodes to publish and subscribe to topics, enabling flexible and modular communication in ROS systems. They facilitate information exchange and interoperability across different programming languages and platforms, simplifying the development of complex robotic applications.

Figure 3.3 shows an illustration of this command being executed on a Twist message. In this instance, the message's data structure is split into two sections, one of which is concerned with the robot's linear velocity and the other with its rotational velocity. Both structures make use of three float integers (x,y,z) that represent the robot's desired velocities along the x,y, and z axes. All of this message-based data and information transmission between nodes is made possible through the use of topics.

3.1.3 Topics:

In ROS, topics facilitate communication between nodes through a publish/subscribe mechanism. Nodes can publish messages to topics, and other nodes can subscribe to those topics

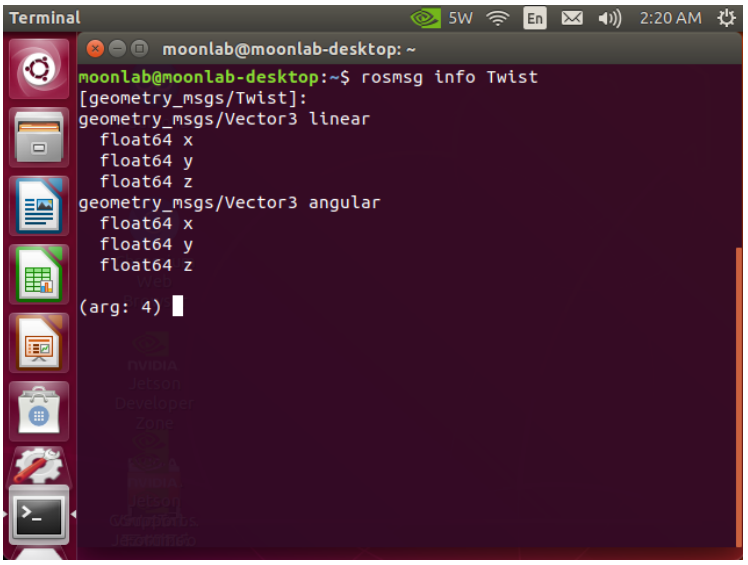

FIGURE 3.3: ROS msg command example

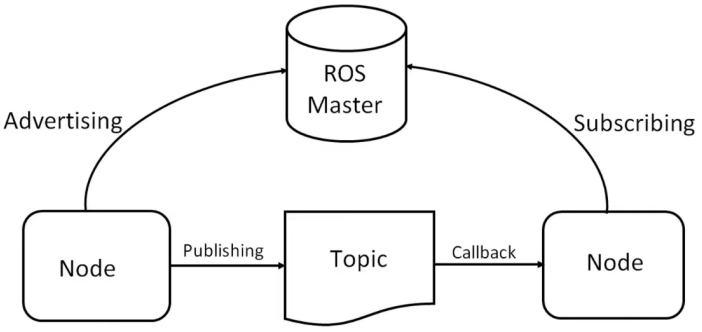

FIGURE 3.4: ROS Architecture with topics and nodes [1]

to receive the messages (3.4). Topics serve as channels for exchanging data and enable decoupled and asynchronous communication within the ROS system.

3.1.4 roscore:

Roscore is a key component that serves as the core of the ROS system. It provides essential infrastructure services for communication between nodes, including the Master, Parameter Server, and rosout logging. The roscore command initializes the ROS Master, which manages the registration and discovery of nodes, as well as the coordination of message passing via topics, services, and actions. It acts as a centralized hub for facilitating communication and enabling the seamless interaction of different nodes within the ROS network.

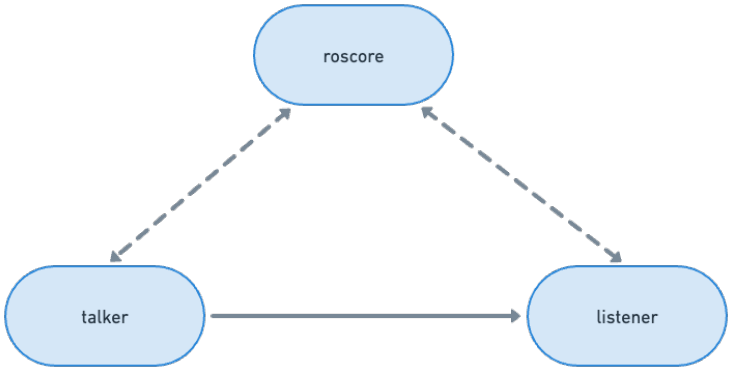

FIGURE 3.5: roscore connection with other nodes in the system [1]

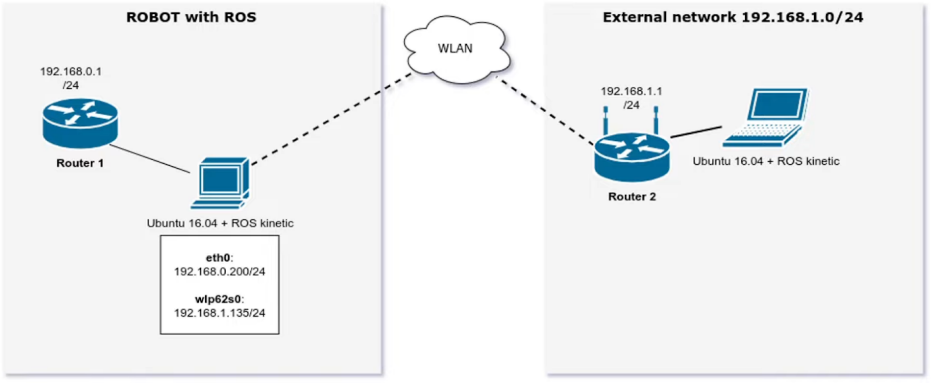

FIGURE 3.6: ROS Network Main (master node) system and Subsystems (slave nodes) [1]

In the illustrated scenario depicted in Figure 3.5, the listener node subscribes to the talker node, which implies that it receives and reads the messages generated by the talker. Both nodes periodically communicate with the roscore node. Additionally, when working with standalone robots or in multi-robot applications, managing coordinate frames becomes a significant concern. To address this, the tf package has been developed (3.6).

3.2 Differential Drive

A mobile robot with differential wheels moves by means of two independently powered wheels mounted on either side of the robot body. It doesn't need to steer at all because it can change its direction by altering the relative pace at which its wheels rotate [2]. In order to prevent the robot from tossing, such drives often contain one or more castor wheels.

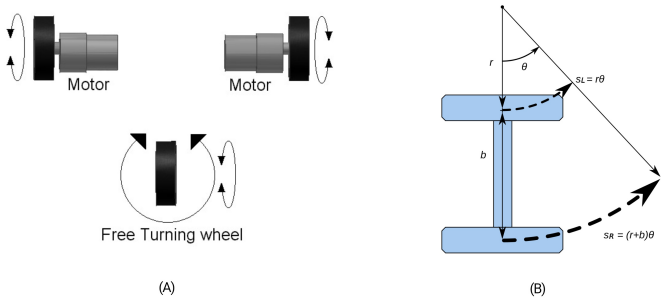

FIGURE 3.7: (A)A differential drove robot with tri-wheel system (B)Course of wheels during the turn.[2]

The robot will move in a straight path if both wheels are turned in the same direction and at the same speed. The robot will spin about the axis's centre if both wheels are spun at the same speed in opposite directions, as is evident from the diagram. Otherwise, the centre of rotation may lie anywhere along the line established by the two contact points of the tyres, depending on the speed and direction of the rotation [3.7]. The centre of rotation is infinitely far away from the robot when it is moving straight forward. The pace and direction of rotation of the two driven wheels determine the direction of the robot, hence it is important to sense and regulate these values. Figure 3.8 illustrates the differential drive kinematics of a wheeled mobile robot. The entire system is presented in an X-Y global coordinate system. Here, ICR is the instantaneous centre of rotation or the point around which the wheel rotates. From the diagram, one can obtain the angular velocity by taking:

$$\omega.(R + \frac{b}{2}) = v_R$$

$$\omega.(R - \frac{b}{2}) = v_L$$

where,

- ω = angular velocity
- b = width of the vehicle
- v_R = contact speed of the right wheel
- v_L = contact speed of the left wheel
- r = radius of the wheels

From solving the above Equations, the value of ω and R comes out to be:

$$\omega = \frac{(v_R - v_L)}{b}$$

$$R = \frac{b}{2}.\frac{(v_R + v_L)}{(v_R - v_L)}$$

The instantaneous velocity V of the point halfway between the wheels of the robot is determined by the angular velocity equation:

$$V = \omega.R = \frac{(v_R + v_L)}{2}$$

Using this relation,

$$R = \frac{V}{\omega}$$

$$\omega_R = \frac{v_R}{r}$$

$$\omega.(\frac{b}{2}) = v_R$$

Through this, one can obtain the right and left wheel's angular velocity as ω_R and ω_L respectively,

$$\omega_R = \frac{V + \omega.\frac{b}{2}}{r}$$

$$\omega_L = \frac{V - \omega.\frac{b}{2}}{r}$$

3.3 Kalman Filter

The Kalman filter is an algorithm used for estimation and control in systems that are subject to uncertainty, such as tracking objects or estimating the state of a dynamic system. It combines information from measurements and system dynamics to provide an optimal estimate of the system's true state. The filter is named after Rudolf E. Kálmán, who developed the algorithm in the 1960s.

The Kalman filter operates in a recursive manner, updating the estimate as new measurements become available. It consists of two main steps: the prediction step and the update step.

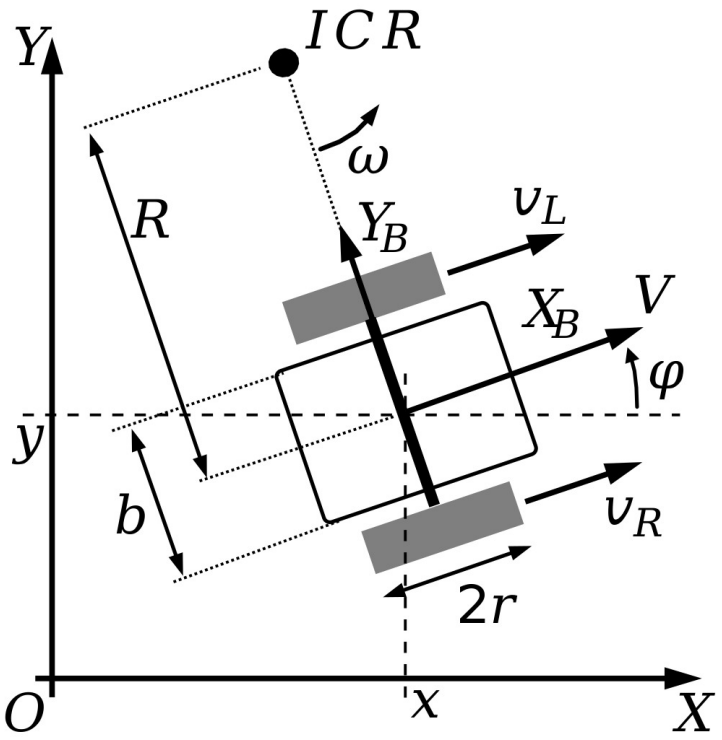

FIGURE 3.8: Kinematics of Differential Driven Wheels [2]

Prediction Step:

In this step, the Kalman filter predicts the current state of the system based on the previous state estimate and the system dynamics model. It represents beliefs by the moment's parameterization: At the time t, the belief is represented by the mean μ_t and the covariance Σ_t. Posteriors are Gaussian if the following three properties hold, in addition to the Markov assumptions of the Bayes filter. The state transition probability $p(\mu_t|\mu_t, x_{t-1})$ must be a linear function in its arguments with added Gaussian noise.

State Prediction : $\mu_t^{'} = A_t\mu_{t-1} + B_t u_t + \epsilon_t$
Covariance Prediction : $\Sigma_t^{'} = A_t\Sigma_{t-1}A_t^T + Q_t$ where,

- $\mu_t^{'}$is the predicted state vector at time step t.
- A_t is the state transition matrix that relates the previous state to the current state.
- ϵ_t is a Gaussian random vector that models the uncertainty introduced by the state transition.
- μ_{t-1} is the previous state estimate.
- B_t is the control input matrix (if applicable).

- u_t is the control input (if applicable).
- Q_t is the process noise covariance matrix, representing the uncertainty in the system dynamics.

Update Step:
In this step, the Kalman filter incorporates new measurements to update the state estimate and improve its accuracy.

Innovation: $I_t = (z_t - H_t\mu_t^{'})$
Innovation Covariance: $S_t = H_t\Sigma_t^{'}H_t^T + R_t$
Kalman Gain: $K_t = \Sigma_t^{'}H_t^T S_k^{-1}$
Updated State: $\mu_t = \mu_t^{'} + K_t I_t$
Updated Covariance: $\Sigma_t = (I - K_t H_t)\Sigma_t^{'}$ where,

- $z_t$is the measurement vector at time step t.
- H_t is the measurement matrix that maps the state space to the measurement space.
- R_t is the measurement noise covariance matrix, representing the uncertainty in the measurements.
- I_t is the innovation or measurement residual.
- S_t is the innovation covariance, measuring the consistency between the predicted and actual measurements.
- K_t is the Kalman gain, which determines the weight given to the measurement update.
- I is the identity matrix.

Algorithm 1: Kalman Filter

KalmanFilterAlgorithm$(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)$**:**
$\mu_t^{'} = A_t\mu_{t-1} + B_t u_t$;
$\Sigma_t^{'} = A_t\Sigma_{t-1}A_t^T + Q_t$;
$K_t = \Sigma_t^{'}H_t^T(H_t\Sigma_t^{'}H_t^T + R_t)^{-1}$;
$\mu_t = \mu_t^{'} + K_t(z_t - H_t\mu_t^{'})$;
$\Sigma_t = (I - K_t H_t)\Sigma_t^{'}$;
return μ_t, Σ_t

The Kalman filter provides an optimal estimate by minimizing the mean squared error between the true state and the estimated state. It combines information from the system dynamics model and measurements, taking into account their respective uncertainties. The filter adjusts its estimates based on the relative reliability of the predictions and measurements, giving more weight to the more accurate information.

The Kalman filter has widespread applications in various fields, including robotics, navigation, computer vision, and signal processing. It is a powerful tool for state estimation and provides an effective solution for systems affected by noise and uncertainty.

3.4 PID Controller

A PID (Proportional-Integral-Derivative) controller is a widely used feedback control algorithm that aims to regulate a system's output by adjusting an actuator based on the error between the desired setpoint and the measured process variable. The controller continuously computes an output signal that applies corrective actions to minimize the error and bring the system closer to the desired state.

The PID controller consists of three components: proportional, integral, and derivative, which act on the error signal in different ways:

1. Proportional (P) Term: The proportional term produces an output proportional to the current error. It provides an immediate response to the present error and helps reduce steady-state error.

$$P(t) = K_p * e(t)$$

- $P(t)$ is the proportional output at time t.
- K_p is the proportional gain, a tuning parameter that determines the proportional response.
- $e(t)$ is the error at time t, calculated as the difference between the setpoint and the measured value.

2. Integral (I) Term: The integral term integrates the cumulative error over time, aiming to eliminate any steady-state error and address systematic biases in the system. It increases the controller's output when the error persists for a longer duration.

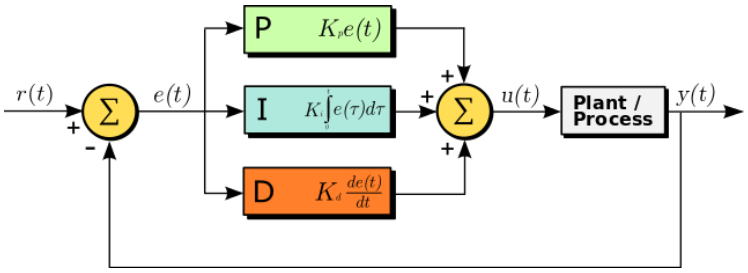

FIGURE 3.9: A block diagram of a PID controller in a feedback loop. r(t) is the desired process value or setpoint (SP), and y(t) is the measured process value (PV)[3].

$$I(t) = K_i * \int_0^t e(\tau)d\tau$$

- $I(t)$ is the integral output at time t.
- K_i is the integral gain, a tuning parameter that determines the integral response.
- $\int_0^t e(\tau)d\tau$ represents the integral of the error over time.

3. Derivative (D) Term: The derivative term considers the rate of change of the error and predicts its future behavior. It provides a dampening effect and helps prevent overshoot and oscillations.

$$D(t) = K_d * \frac{de(t)}{dt}$$

- $D(t)$ is the derivative output at time t.
- K_d is the derivative gain, a tuning parameter that determines the derivative response.
- $\frac{de(t)}{dt}$ represents the derivative of the error with respect to time.

The final control output of the PID controller is the sum of the three terms [3.9]:

$$Output = P(t) + I(t) + D(t)$$

The tuning of the PID controller involves adjusting the gains (K_p, K_i, and K_d) to achieve the desired response. The proportional gain determines the strength of the proportional

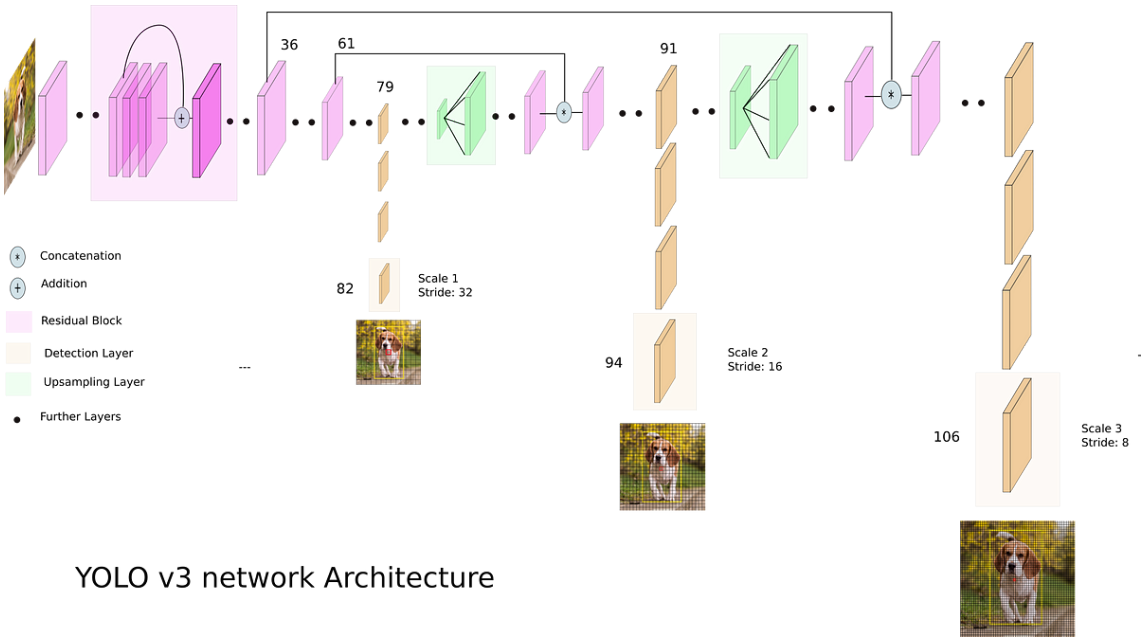

FIGURE 3.10: The overall YOLOv3 architecture [4]

response, the integral gain adjusts the impact of past errors, and the derivative gain controls the effect of the error rate of change.

The PID controller algorithm operates in a feedback loop, continuously computing the control output based on the current error and the accumulated error over time. It provides a balance between fast response, steady-state accuracy, and stability, making it widely applicable in control systems for various applications such as robotics, process control, and automation.

3.5 YOLO

YOLOv3 (You Only Look Once v3) is an advanced object detection algorithm that provides real-time and accurate detection and localization of objects in images and videos. It builds upon the success of its predecessors, YOLO and YOLOv2, by introducing several key improvements [29].

The main idea behind YOLOv3 is to divide the input image into a grid and perform object detection within each grid cell. This approach allows YOLOv3 to efficiently process the entire image in a single pass through a deep neural network, resulting in real-time inference speeds.

Here's a concise overview of the YOLO v3 architecture as shown is figure 3.10:

- **Input Image:** The input image is divided into a grid.
- **CNN Backbone:** YOLO v3 uses a convolutional neural network (CNN) as its backbone, typically based on the Darknet architecture. The CNN processes the image to extract features and generate a feature map.
- **Feature Extraction:** Multiple convolutional layers are applied to the input image to extract features at different scales and levels of abstraction.
- **Detection at Different Scales:** YOLO v3 predicts bounding boxes and class probabilities at three different scales. Each scale corresponds to a specific detection layer in the CNN.
- **Anchor Boxes:** Anchor boxes are predefined bounding boxes of different sizes and aspect ratios. YOLO v3 uses anchor boxes to generate predictions for objects of different shapes and sizes.
- **Predictions:** For each grid cell in the feature map, YOLO v3 predicts multiple bounding boxes and associated class probabilities. Each bounding box consists of coordinates (x, y, width, height) and a confidence score.
- **Non-Maximum Suppression (NMS):** To eliminate duplicate or overlapping detections, YOLO v3 applies NMS, which selects the most confident bounding box among overlapping ones based on a predefined threshold.
- **Output:** The final output of YOLO v3 is a list of bounding boxes, each assigned with a class label and a confidence score.

Chapter 4

Methodology

4.1 Mechanical Subsystem

4.1.1 Chassis & Wheels

A clear acrylic chassis was used for the base layer of the rover with dimensions of 22 x 14.7cm or 8.7' x 5.8' (L*W) as portrayed in Figure 4.1. The second tier was built by the authors by recycling random household objects, for example, the supporting pillars were made using used pens and the second storey floor is repurposed leftover waste PVC pipe, which has been cut, heated and straitened out as can be seen in the overall picture of the ground vehicle in Figure 4.2.

The chassis is rectangular in shape with an approximate weight of 292g. There are a total of 3 wheels attached to it, giving it a tricycle look. There is a single all-directional front wheel to support the chassis and distribute its load evenly. The main wheels have a dimension of 7 x 7 x 2.6cm or 2.8' x 2.8' x 1' (L*W*T) and are 5.8' apart. These wheels are attached to the DC motors.

4.1.2 Motors(200 rpm 4-6V DC):

An electric motor is an electromechanical tool that transforms electrical energy into mechanical energy. It has a rotor (the moving part) and a stator (the fixed element), among other parts. The magnetic field produced by the electric current in the motor interacts with the magnets in the rotor to drive rotation.

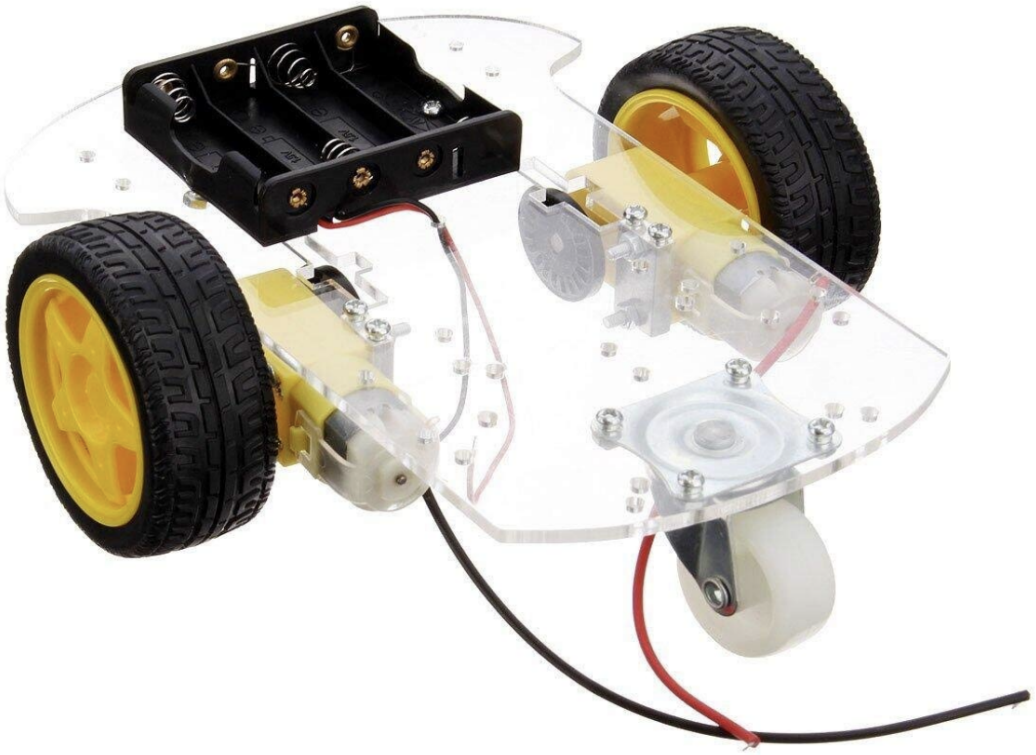

FIGURE 4.1: The bare chassis of Automated Ground Vehicle

The specifications we used are "200 rpm 4-6V DC," describing the operating parameters of the motor:

- **RPM (Revolutions Per Minute)**: The number of full rotations the motor makes in a minute is referred to as RPM. The motor in this instance rotates at a speed of 200 revolutions per minute. This number represents the output shaft rotational speed of the motor.
- **Voltage**: The motor is intended to function between 4-6 volts of direct current (DC). This indicates that for effective operation, it needs a direct current power source with a voltage of between 4 and 6 volts.

To sync and control these DC motors to absolute precision and to supply them with a steady voltage, motor drivers are introduced into the frame, which are in direct contact with the Arduino UNO board.

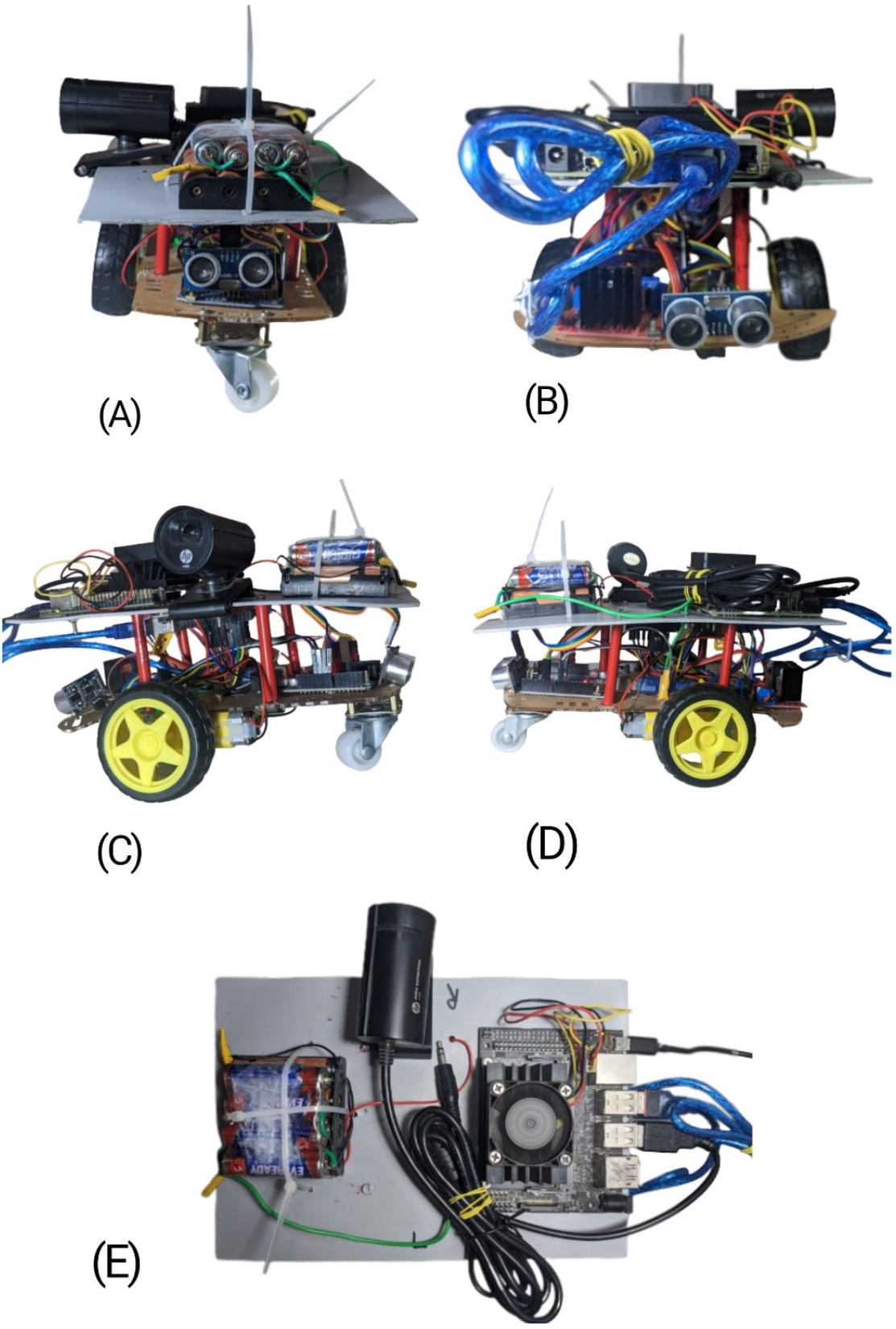

FIGURE 4.2: The overall look of the self-built Automated Ground Vehicle (A)Front-view (B)Rear-view (C)Side-view 1 (D)Side-view 2 (E)Top-view

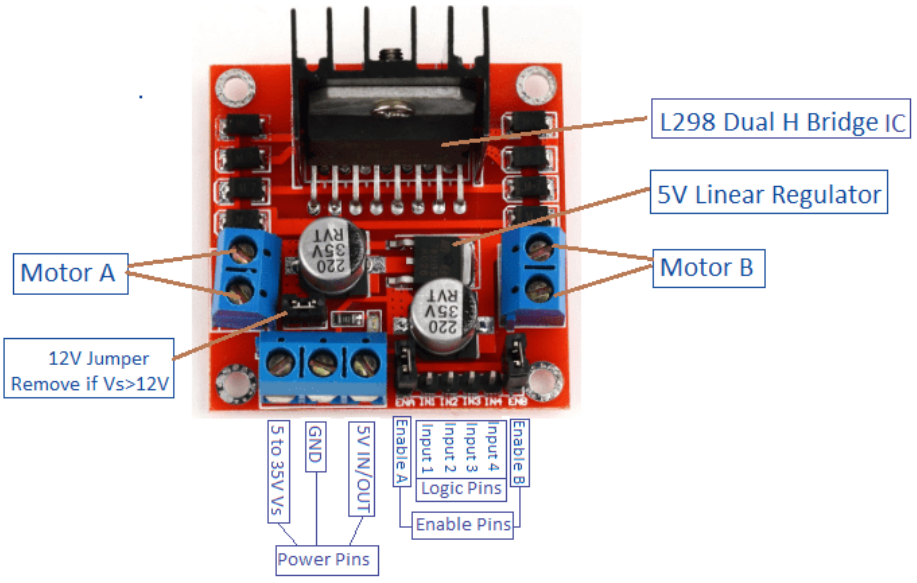

FIGURE 4.3: L298N Motor Driver[5]

4.1.3 L298N Driver:

Popular integrated circuit (IC) L298 is frequently used as an H-bridge driver for motors (Fig. 4.3). The direction and speed of DC motors or stepper motors must be controlled, and this is a common requirement in robotics and other applications.

Two DC motors can be driven by the L298 H-bridge in both forward and backward directions, making it possible to control the rotation of the motors in both directions. It can also use pulse width modulation (PWM) signals to regulate the motors' speed. It can also be utilised to power a single bipolar stepper motor.

The IC can drive motors that need more power because it is built to handle high currents and voltages. It comprises two H-bridges, each of which has an independent motor driving capability. Four transistors, often MOSFETs, make up each H-bridge and can be turned on or off to regulate the amount of current passing through the motor windings. For the L298 to power the motors, an external power source is necessary. The circuit is shielded from voltage spikes that happen when the motor is shut off by built-in diodes known as flyback diodes or freewheeling diodes. These diodes guard against harm to the IC and other parts.

Typically, input signals to the L298's control pins are required in order to control it. The inputs regulate the transistors' on/off states in the H-bridge, which affects how the motor rotates. You may adjust the motor speed by sending PWM signals to the control pins.

FIGURE 4.4: HP w100 480P 30 FPS Digital Webcam

All things considered, the L298 H-bridge motor driver is a flexible and popular IC that makes it easier to drive and control DC motors or bipolar stepper motors. It is a crucial component in

numerous robotics and motor control applications since it provides bidirectional control, PWM speed control, and protective features.

4.1.4 Camera & Sensors

4.1.4.1 WEBCAM (HP w100 480P 30 FPS Digital Webcam)

HP (Hewlett-Packard) is the manufacturer of the HP w100 480P 30 FPS Digital Webcam (Fig. 4.4). It's made to work with laptops or desktops and lets you record video and audio for a variety of uses, including video conferencing, live streaming, online meetings, and video recording.

The main characteristics and details of the HP w100 webcam are described below:

Resolution: The HP w100 can record video at a resolution of up to 640x480 pixels thanks to its 480P resolution. While not high-definition (HD), this quality is enough for basic video conversations and online communication.

Frame Rate:30 frames per second (FPS) is the frame rate of the webcam. This indicates that it can record and send 30 frames of video per second, producing relatively slick and fluid video playback.

This web camera is mounted on the rover for capturing video of the objects and detect them. Using the USB type A connector this camera is connected with Jetson Nano USB port.

4.1.4.2 LM393 SPEED SENSOR (Wheel encoder):

A typical sensor for determining the rotation or speed of a wheel or other rotating objects is the LM393 Speed Sensor, usually referred to as a wheel encoder. The LM393 comparator integrated circuit serves as its foundation (Figure 4.5).

An overview of the LM393 Speed Sensor is provided below:

- **Principle of Operation:**The LM393 Speed Sensor operates by sensing changes in the magnetic field brought on by the rotation of a ferrous target mounted on a rotating shaft, such as a toothed wheel or a magnet. It comprises a sensor module containing a Hall-effect sensor and a magnet.
- **Hall-Effect Sensor:** A Hall-effect sensor is used by the LM393 Speed Sensor to identify changes in the magnetic field. With respect to the strength of the magnetic field it is exposed to, the Hall-effect sensor produces a voltage output. The magnet on the target wheel generates a shifting magnetic field that is detected by the Hall-effect sensor as it revolves.
- **Comparator Circuit:** Two comparators are included in the LM393 integrated circuit and are used to transform the analogue voltage output from the Hall-effect sensor into a digital signal. The comparators determine a high or low output based on the comparison of the Hall-effect sensor output voltage with a predetermined threshold voltage.
- **Output Signal:** A digital output signal, often in the shape of a square wave, is provided by the LM393 Speed Sensor. The wheel's or the target object's spinning speed is directly inversely related to the square wave's frequency. The rotational speed can be determined by counting the number of pulses that occur during a particular time interval.

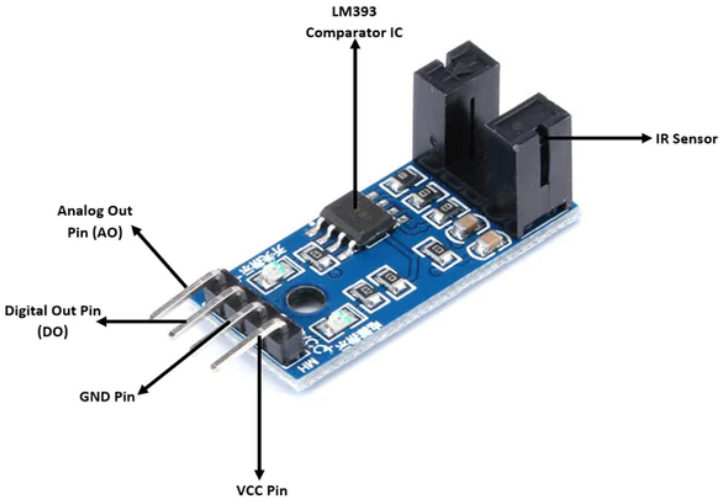

FIGURE 4.5: LM393 IR speed sensor module

4.1.4.3 Ultrasonic Sensors

In order to measure distance, identify objects, or navigate in a variety of applications, an ultrasonic sensor uses sound waves at frequencies higher than the upper audible limit of human hearing, often above 20 kilohertz. Similar to how bats or dolphins utilise sound waves to navigate and locate items in their surroundings, it works on the principle of echolocation.

Here is a quick explanation of how an ultrasonic sensor functions:

WORKING OF AN HC-SR04 SENSOR:

The frequency of ultrasonic sound vibrations is higher than the range of human hearing. Ultrasonic noises are sent and received via transducers, which are microphones. One transducer is used by the HC-SR04 and other ultrasonic sensor modules to both send and receive pulses. By monitoring the interval between sending and receiving ultrasonic pulses, the sensor calculates the distance to the target.

$$distance = \frac{sound\ of\ speed\ \ x\ \ time\ taken}{2}$$

Here is a quick explanation of how an ultrasonic sensor functions:

1. **Transducer:** The transducer in the sensor serves as both a speaker and a microphone. By turning electrical energy into mechanical vibrations, it creates ultrasonic sound waves.

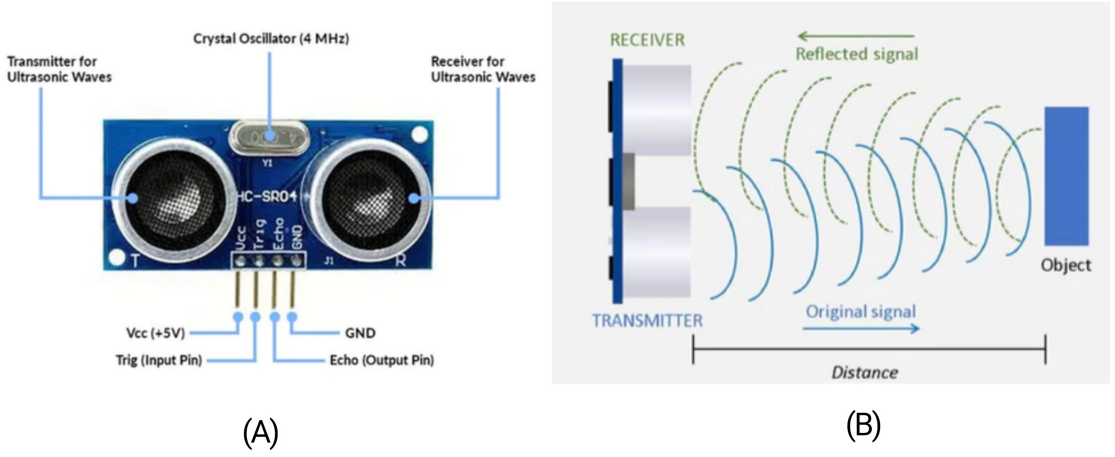

FIGURE 4.6: (A) HC-SR04 Sensors (B) Working Principle of Ultrasonic Sensors [1]

2. **Sound wave emission:** The transducer sends out a brief ultrasonic pulse into the environment, usually lasting a few microseconds. The pulse can move through solids or other substances like water or air.

3. **Reflection and reception:** The transmitted sound wave is reflected back towards the sensor when it comes into contact with an item in its path.

4. **Receiver:** The transducer that produced the pulse now serves as a microphone and picks up the reflections of sound waves.

5. **Time-of-flight calculation:** The sensor may calculate the distance between itself and an item by measuring the time it takes for the ultrasonic pulse to reach it and return. This computation is based on the sound speed of the medium (such as air or water) that the wave travelled through.

6. **Measurement of distance:** The sensor transforms time-of-flight information into distance data using the established sound speed. Frequently, this data is presented as a direct distance value, a digital signal, or an analogue voltage.

4.1.4.4 Wifi Adapter for Networking

A computer or other device can connect to a wireless network using a Wi-Fi adapter, also referred to as a wireless network adapter. The 802.11n wireless networking standard, which

FIGURE 4.7: 802.11n Wifi Adapter [5]

offers faster speeds and greater range compared to earlier Wi-Fi standards, is supported by an 802.11n Wi-Fi adaptor [(4.7)].

4.1.5 Power System

The system uses 2 types of power sources,i.e., direct AC current for the Jetson Nano since it needs a steady flow of currents to process. The Arduinos, Drivers and motors are all powered by 8 1.5V batteries, summing up to a massive 12V output. This has allowed a wider range of movement and velocity flexibility into the AGV. The entire power supply chain in the Autonomous Ground Vehicle is illustrated in Figure 4.8. According to the image a 220V AC supply is used to power Jetson Nano which is converted to the 5V, 2.5A DC supply. Jetson is been powered by a micro USB port. On the other hand motors and motor drivers are running on a 12 V DC power supply by batteries. Arduino and its peripheral devices such as ultrasonic sensors and IR encoders are fueled by Arduino itself.

4.1.6 Micro-controller & Onboard Computer

4.1.6.1 Arduino

A popular microcontroller board, known as the Arduino Uno, is built around the ATmega328P chip. Due to its simplicity, accessibility, and adaptability, it is frequently utilised in the electronics and prototype industries. The device contains a 16 MHz ceramic resonator, 6 analogue inputs, 14 digital input/output pins (of which 6 can be used

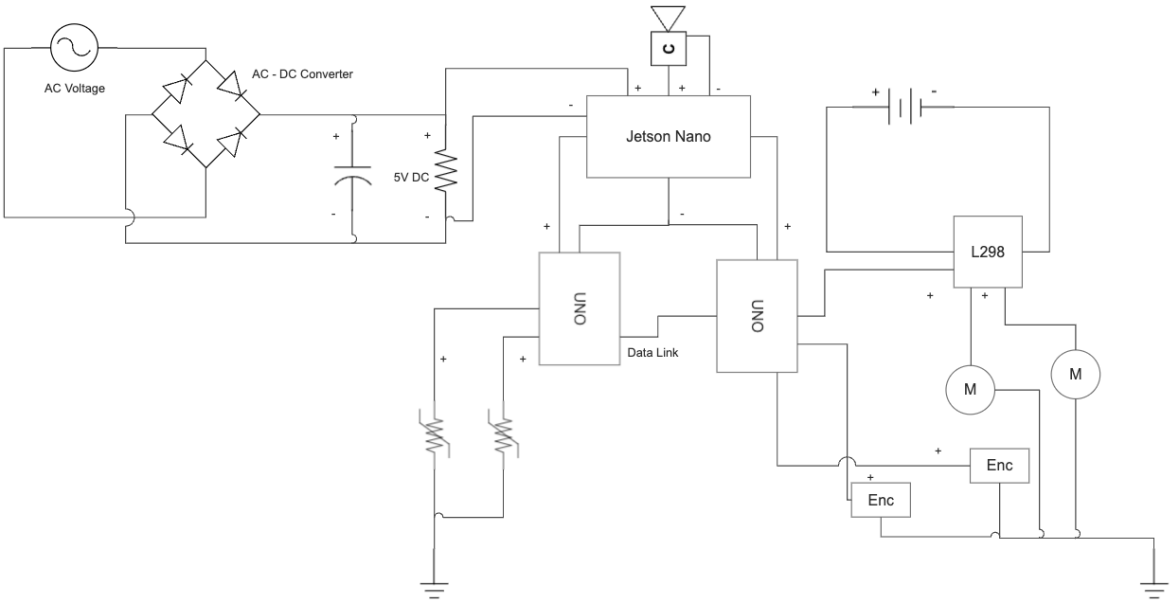

FIGURE 4.8: The overall power supply chain in the AGV

as PWM outputs), a USB port, a power jack, an ICSP header, and a reset button as showcased in Figure (4.9).

The Arduino Uno's salient features are as follows:

1. **Microcontroller:** The ATmega328P microprocessor, the device's brain, is the centre of the Arduino Uno's construction. It uses 5 volts and has 32KB of flash memory to store your programme, 2KB of SRAM for variables, and 1KB of EEPROM to store non-volatile data.

2. **Digital and Analog I/O:** Six of the 14 digital input/output (I/O) pins on the Uno can be utilised as pulse-width modulation (PWM) outputs, enabling you to control luminance-varying devices like LEDs. Additionally, it includes six analogue input pins that may read analogue sensors or other voltage levels.

3. **Power Supply:** Either an external power supply or a USB connection can be used to power the board. The Uno can be powered with 5 volts in addition to the suggested voltage range of 7 to 12 volts.

4. **Programming:** The Arduino Integrated Development Environment (IDE), which offers a user-friendly interface for creating and uploading code to the board, can be used to programme Arduino Uno. The programming language is based on C/C++

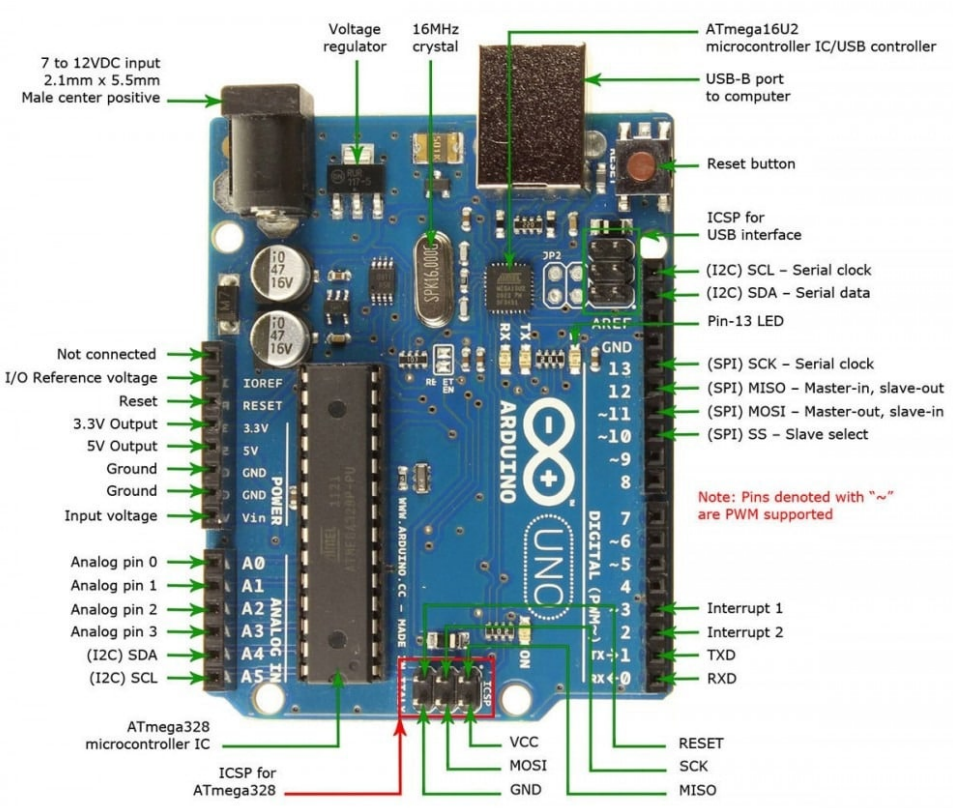

FIGURE 4.9: Arduino UNO PIN Configurationc[6]

and features a streamlined API and a large selection of libraries to enable different tasks.

5. **Shields and Expansion:** Due to the Arduino Uno's standardised physical factor, adding shields to it is simple. The Uno may be enhanced with shields, which are extra circuit boards that can be placed on top of it and provide features like Ethernet connectivity, wireless communication, motor control, and more.

6. **Community and Documentation:** Due to the enormous and vibrant user base of Arduino, a wealth of tutorials, examples, and online resources are readily available. There is a variety of information available to assist developers of all skill levels with their projects.

4.1.6.2 Jetson Nano

The Jetson Nano is a small, powerful single-board computer designed specifically for AI and robotics applications. It is developed by NVIDIA, a leading company in the field of graphics processing units (GPUs) and artificial intelligence [(4.10)].

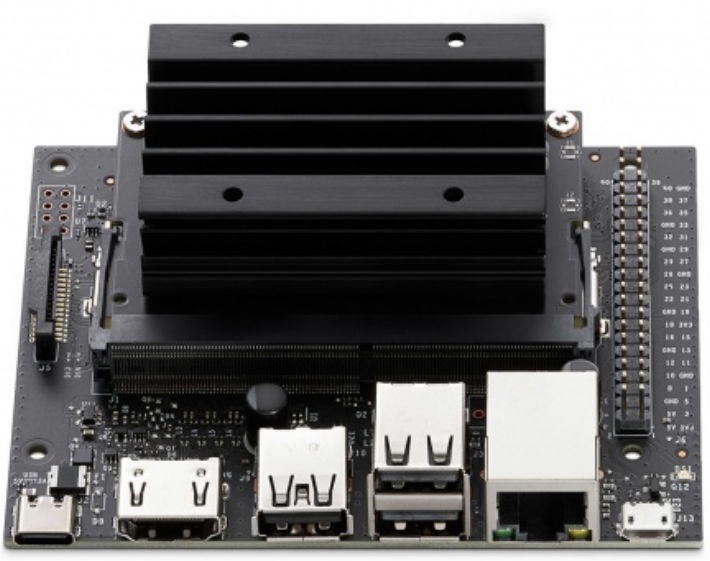

FIGURE 4.10: Jetson Nano by Nvidia[7]

Here are the key details about the Jetson Nano:

Architecture: The Jetson Nano is built around the NVIDIA Maxwell GPU architecture with 128 CUDA cores. It also features a quad-core ARM Cortex-A57 CPU. Performance: It offers impressive performance for its size, capable of delivering up to 472 GFLOPS (floating-point operations per second) of AI performance [(4.11)].

Memory: The Jetson Nano comes with 4 GB of LPDDR4 RAM, which provides ample memory for runni and store data. Additionally, it has support for external USB 3.0 storage devices.

Connectivity: The Jetson Nano includes various connectivity options, such as Gigabit Ethernet, HDMI 2.0, USB 3.0, and USB 2.0 ports. It also has an M.2 Key E connector for Wi-Fi and Bluetooth connectivity (module not included).

AI Capabilities: The Jetson Nano is specifically designed to accelerate AI workloads. It supports popular AI frameworks and libraries, including TensorFlow, PyTorch, Caffe, and MXNet. The built-in GPU provides hardware acceleration for neural networks, enabling faster inference and training.

Software Support: It runs on the NVIDIA JetPack SDK, which includes the CUDA parallel computing platform, cuDNN deep neural network library, and TensorRT for optimized deep learning inference. It also supports NVIDIA's deep learning software stack and the Jetson OS.

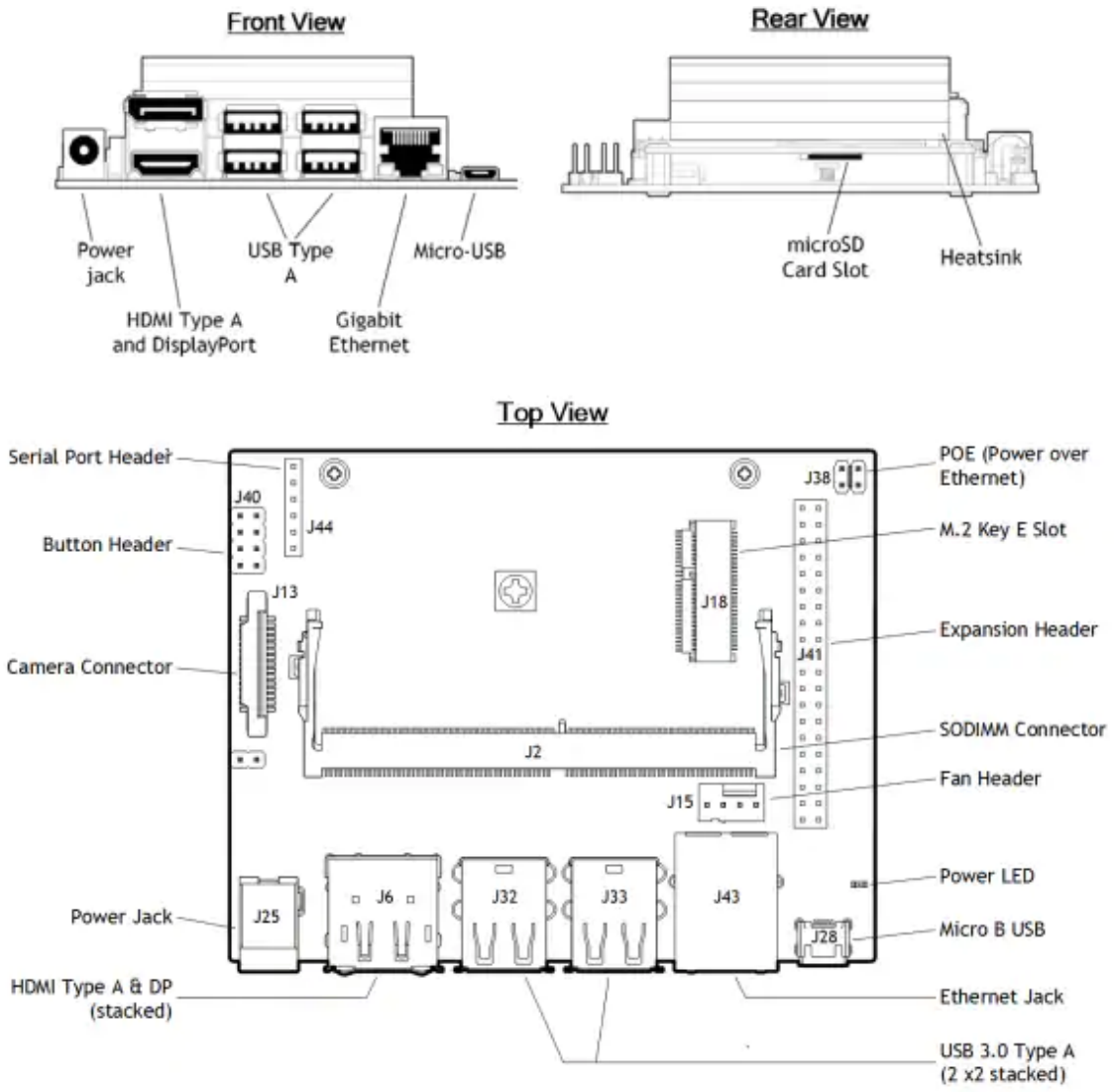

FIGURE 4.11: Labelled Schematic image of Jetson Nano by Nvidia[7]

Power Consumption: The Jetson Nano is energy-efficient and consumes around 5-10 watts of power, making it suitable for embedded systems and low-power applications. Micro USB power cable with 5 Volt, 2.5 A power supply capability has been used to power the Jetson Nano.

Expansion: It features a 40-pin GPIO header that allows for hardware interfacing with sensors, motors, and other peripherals. This enables you to connect and control a wide range of external devices. Only Fan Header Pins have been used to incorporate a cooling fan in the heat sink.

Storage: It has a microSD card slot for storage expansion, allowing you to install an

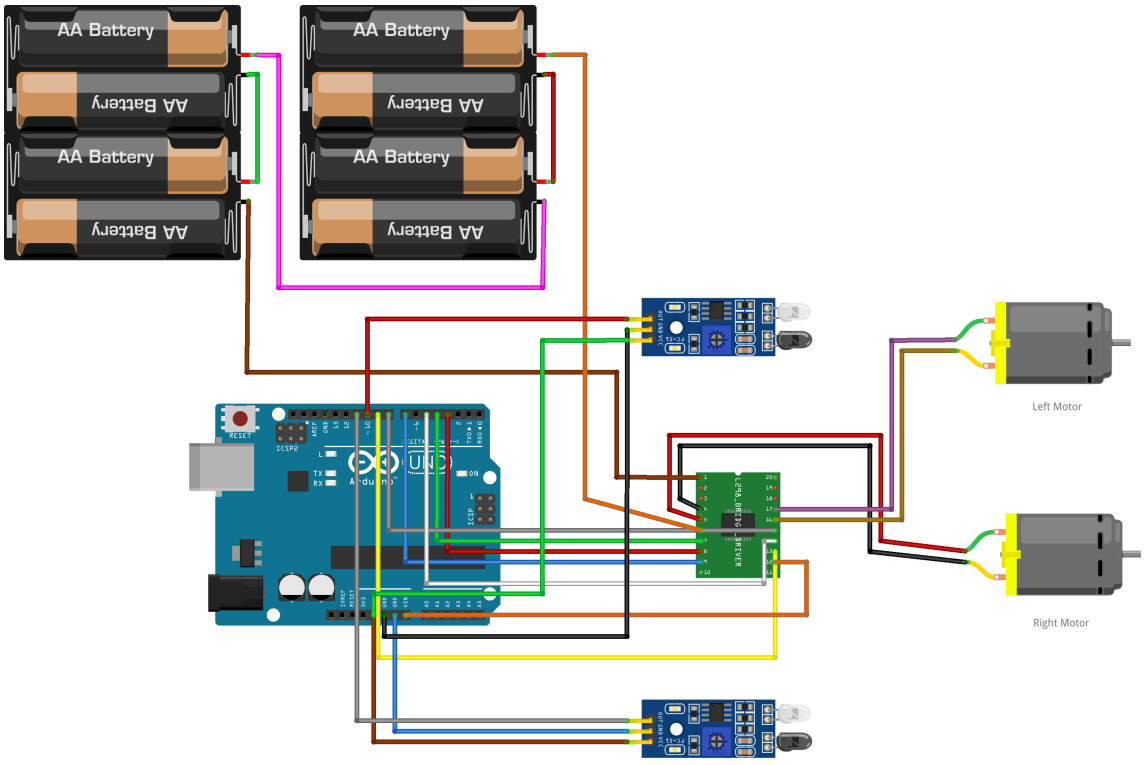

FIGURE 4.12: Detailed diagram of the basic Navigation module

operating system that comes in a compact form factor, measuring approximately 100 mm x 80 mm. Here 32 GB Sandisk memory card has been used to contain the operating system (Ubuntu 18.04), relevant packages and code.

Overall, the Jetson Nano provides a powerful and energy-efficient platform for developing and deploying AI applications in edge devices. Its combination of GPU acceleration, AI software support, and extensive connectivity options makes it a popular choice for AI enthusiasts, researchers, and developers working on edge computing and robotics projects.

4.2 Motions and Navigations

The basic movements of the AGV are programmed and controlled using an Arduino UNO board. This UNO board further receives signals from the Jetson Nano for movements and translates them to wheel commands. These commands are sent to the L298N motor driver module, which in turn drives the motors of the AGV. The entire architecture is given in the form of a schematic diagram 4.13 and as an overall model 4.12, showcasing the connections and making the basic architecture clearer. The driver module is also connected to LM393 speed encoders, which are responsible for monitoring the speed of the motors. The entire

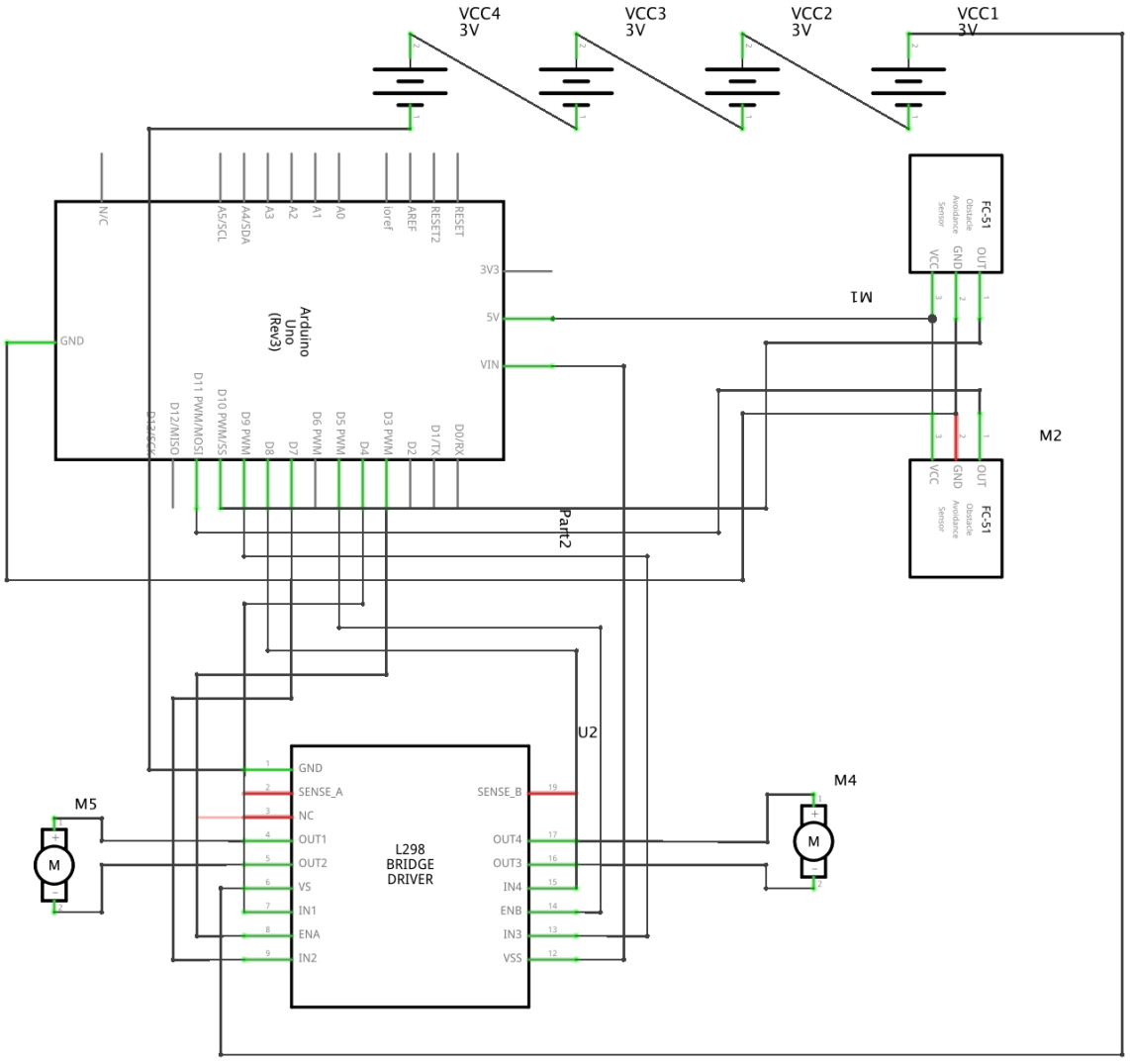

FIGURE 4.13: The overall schematic architecture of the Autonomous Ground Vehicle

system is powered by 8 AA Batteries of 1.2-1.5V, connected in a series. Additionally, a Wifi module is also integrated into the AGV, allowing for wireless communication with a remote device such as a smartphone or tablet. This enables remote control of the AGV's movements and allows for real-time monitoring of its status. Furthermore, the AGV is equipped with sensors that help it navigate through its environment and avoid obstacles. These sensors include ultrasonic sensors, infrared sensors, and a camera. Overall, the combination of the Arduino UNO board, Jetson Nano, L298N motor driver module, and various sensors makes for a powerful and versatile platform for developing autonomous mobile robots.

```
moonlab@moonlab-desktop:~$ ifconfig wlan0
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.111  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::307a:1d6c:e981:cbe7  prefixlen 64  scopeid 0x20<link>
        inet6 fe80::fe68:8cb5:9d2:329e  prefixlen 64  scopeid 0x20<link>
        inet6 fe80::ade5:df9d:eac8:574c  prefixlen 64  scopeid 0x20<link>
        ether 20:e9:17:05:93:5c  txqueuelen 1000  (Ethernet)
        RX packets 141539  bytes 25323545 (25.3 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 625017  bytes 886344294 (886.3 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

FIGURE 4.14: IP address Configuration of Jetson Nano

4.3 Vision, Tracking and Following

The main aim of the proposed AGV is to track and follow a desired object, while simultaneously recording the video of the object. This is done using several ros nodes for vision, locomotion and obstacle avoidance. All of the nodes are connected with a **ros-master** through the entire distributed ros network. Any client-side computer which is connected through the network can display the video of the tracked object.

4.3.1 Networking

To configure Jetson Nano through Local Area Network we need to setup remote VNC and configure SSH connection. To setup Virtual Network Computing, or VNC in Jetson Nano, several steps need to be followed, which are:

- Enabling the VNC server
- Configuration of the VNC server
- Setup a password access to the VNC server
- Reboot the system to access the changes

After setting up the VNC we need to find the IP address of the Jetson Nano. To find the IP address of the wireless connected network we need to use: **ifconfig wlan0**

Jetson Nano in our network has an IP **192.168.0.111** which is clearly displayed in figure 4.14. After knowing the IP address we can set up the SSH easily. First, we need to install the Open SSH client and server in our host computer and Jetson respectively, and then

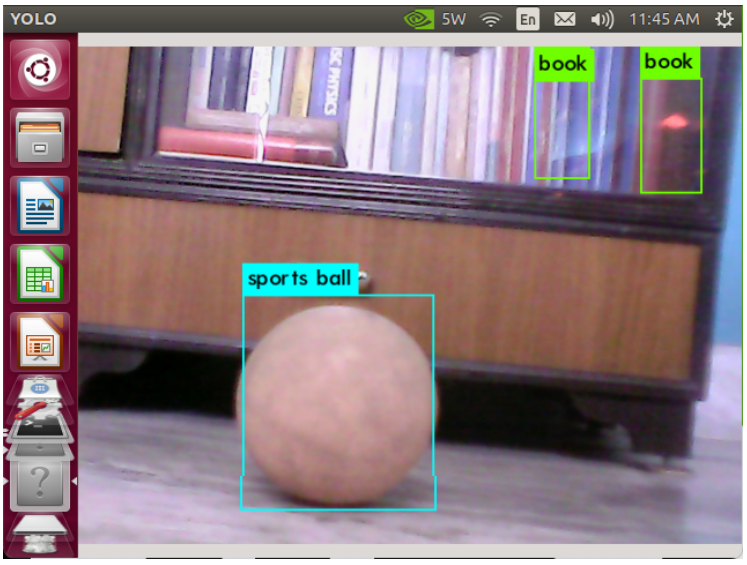

FIGURE 4.15: YOLO is running on the darknet-ros node and detecting everyday objects.

we enabled the SSH for remote devices in the server. Afterwards, the configuration of the password is done on the client and server end. Any registered client can log in to Jetson securely via SSH using: **ssh username@192.168.0.111**.

4.3.2 CV - Pipeline

In this section, the entire working pipeline is viewed in detail with reference to Figure 4.16. This entire pipeline is implemented using ROS (Robot Operating System), which provides a flexible and modular framework for developing robotic applications. Additionally, the Jetson Nano's processing power was crucial in enabling the AGV to make quick decisions and adjust its path in real time.

The HP Webcam fetches real-time images as input at about 7 fps and publishes them into a ros-topic called **/usb_cam/image_raw** as pictured in Figure 4.15. This topic is subscribed by **/darknet_ros** node which contains the YOLO architecture in it. YOLO (You Only Look Once) is a state-of-the-art object detection algorithm that can detect multiple objects in an image and classify them into different categories. Once the objects are detected, their bounding boxes and confidence scores are calculated by the YOLO algorithm. These bounding boxes and confidence scores are then published to another ros-node called **/bounding_boxes** which is responsible for tracking the objects in subsequent frames. The specific object class has to be selected by the user, which the UGV should follow. In this particular experiment, we have used the object class "Ball" for the same.

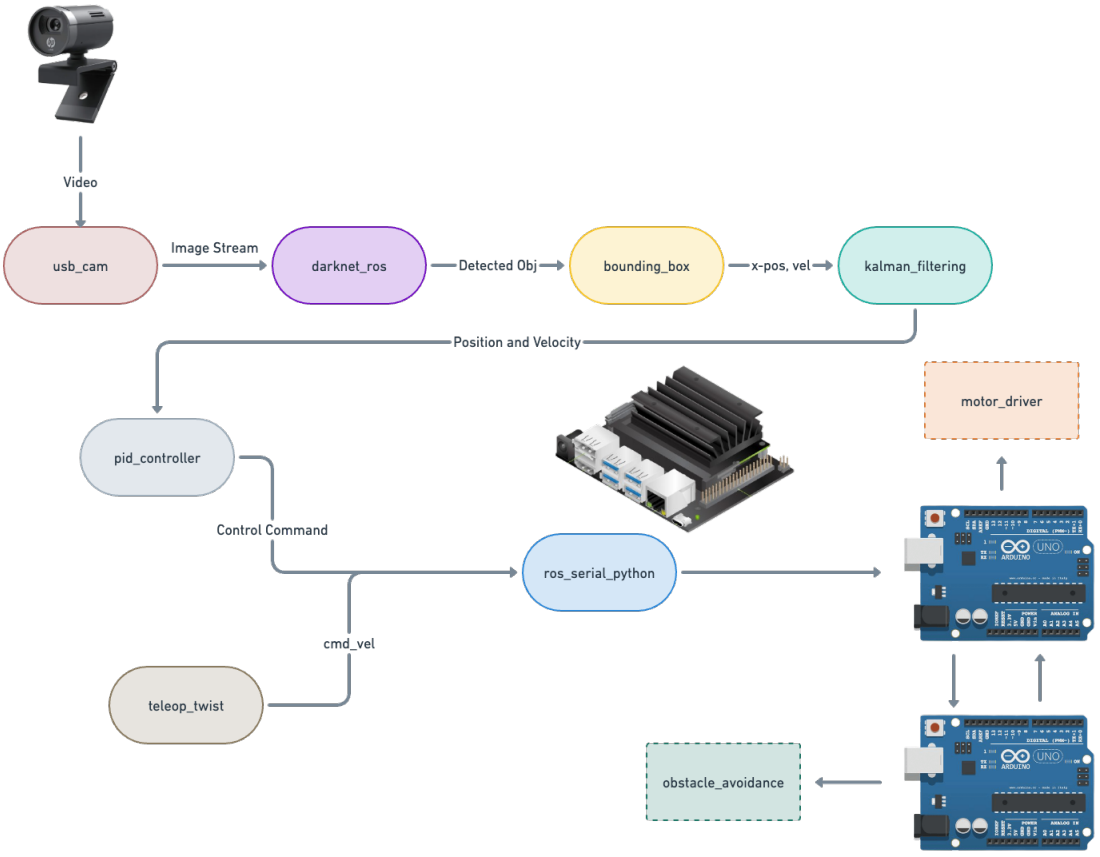

FIGURE 4.16: The overall ros architecture for the UGV

This node uses a Kalman filter to predict the location of the objects in the next frame based on their current location and velocity. The minimum and maximum length and width of these bounding boxes are also published as messages. From here, the midpoint of the bounding box is calculated as,

$$x_{pos} = \frac{(x_{min} + x_{max})}{2}$$

which is further required to calculate the error between the ideal and actual position of the detected object. This X positional error and the relative velocity of the object are fed as inputs to the Kalman Filter to remove discrepancies and predict their current values.

4.3.3 Navigation - Pipeline

After detecting the object, the UGV needs to follow it. For that purpose, the next part of the pipeline has been used. Here, the PID controller gets the input X-position error

and relative velocity. It calculates the velocity command proportional to the distance and the velocity of the detected object from the rover. The velocity commands are then published as twist messages in the **/cmd_vel** topic, which has been subscribed by the **/rosserial_python** node running on the Arduino board. The **/rosserial_python** node then converts the twist messages into PWM signals and sends the signals to the H-Bridge motor driver that controls the motors of the rover, allowing it to move towards or away from the detected object with the desired velocity. The signals from the IR wheel encoder sensors have also been fed into the Arduino microcontroller to calculate the distance travelled by the rover and ensure accurate movement control. Additionally, the rover's onboard ultrasonic sensors have been programmed to detect obstacles in its path and send signals to the microcontroller, which then adjusts the rover's movement accordingly. This combination of sensors and programming allows the rover to navigate autonomously and track the required object while detecting obstacles, making it a valuable tool for exploration and research in various environments.

4.4 Obstacle Detection

The AGV is also equipped with an ultrasonic sensor module that helps in obstacle detection and avoidance.

Obstacle detection is the process of identifying obstacles in the immediate environment in order to prevent collisions. This module sends out ultrasonic waves and measures the time taken for them to bounce back after striking an object as in figure 4.6, thus determining the distance of any obstacles in its path. If there is an obstruction within the sensor's range, it can be determined by calculating the distance between the sensor and the object. Ultrasonic sensors are connected to an Arduino UNO board as illustrated in Figure 4.17. The data from the sensor is processed by an Arduino microcontroller, which then sends signals to the L298N motor driver module to adjust the speed and direction of the motors accordingly.

The Arduino board plays a crucial role in this setup by providing the necessary interface and processing capabilities. The Arduino reads the sensor's output and interprets the distance measurements to determine if an obstacle is present. This information can be used to trigger various actions or responses, such as activating warning signals, controlling motors to avoid collisions, as well as generating alerts on a LED display.

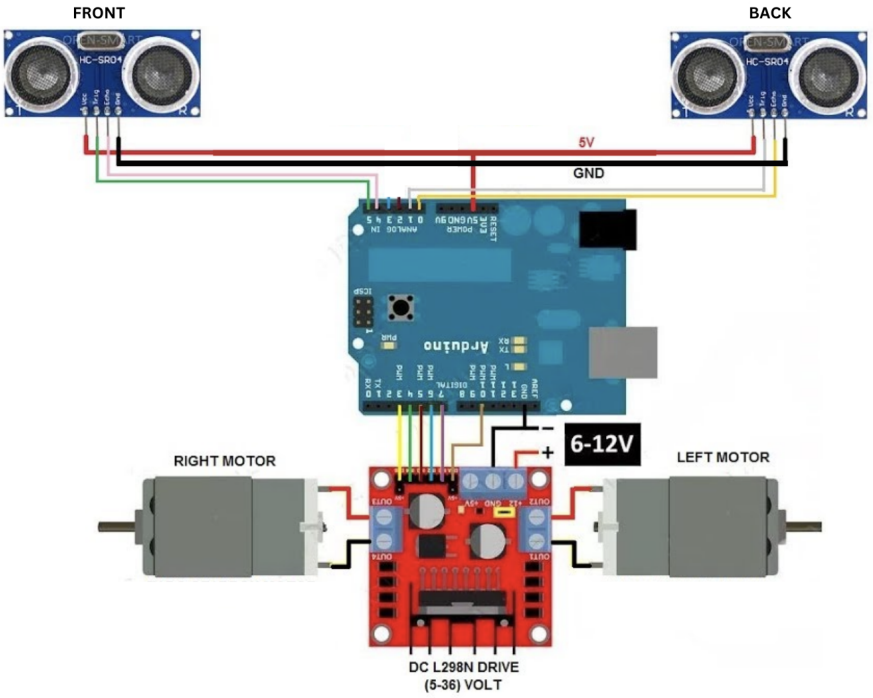

FIGURE 4.17: The overall schematic architecture of the Obstacle Detection module in Autonomous Ground Vehicle

Overall, obstacle detection using ultrasonic sensors and Arduino offers a cost-effective and straightforward solution for detecting objects and obstacles in real time, making it widely used in applications like robotics, automated systems, and home security.

Chapter 5

Result and Discussion

5.1 Simulation

Simulation played a huge role in the development of the AGV, as it allowed for testing and refining of the algorithms and software before deployment.

To begin with, the entire setup was initially implemented in a simulated environment using "WeBots" [30] before stepping into a real-world setting. For this, "Epuck" was taken as the subject robot and "Pioneer" as the tracking matter 5.1. The results were impressive, with the AGV successfully tracking and following the object through obstacles. It helped

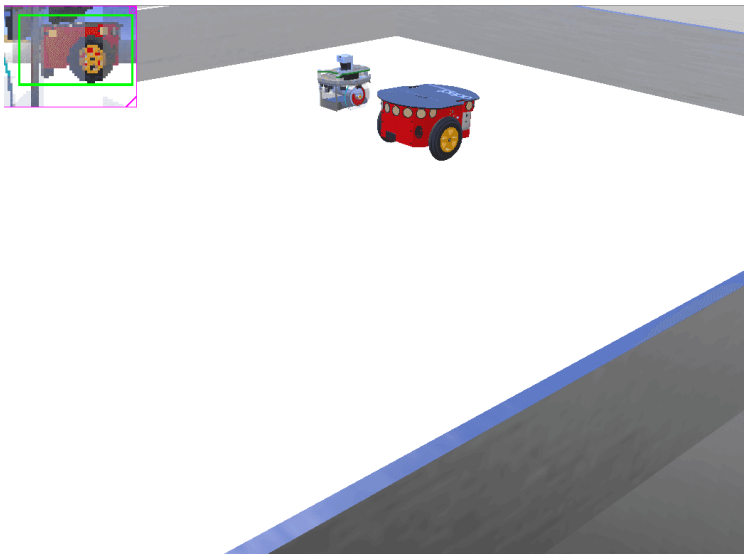

FIGURE 5.1: Simulated Environment in Webots using "Epuck" and "Pioneer" Robots

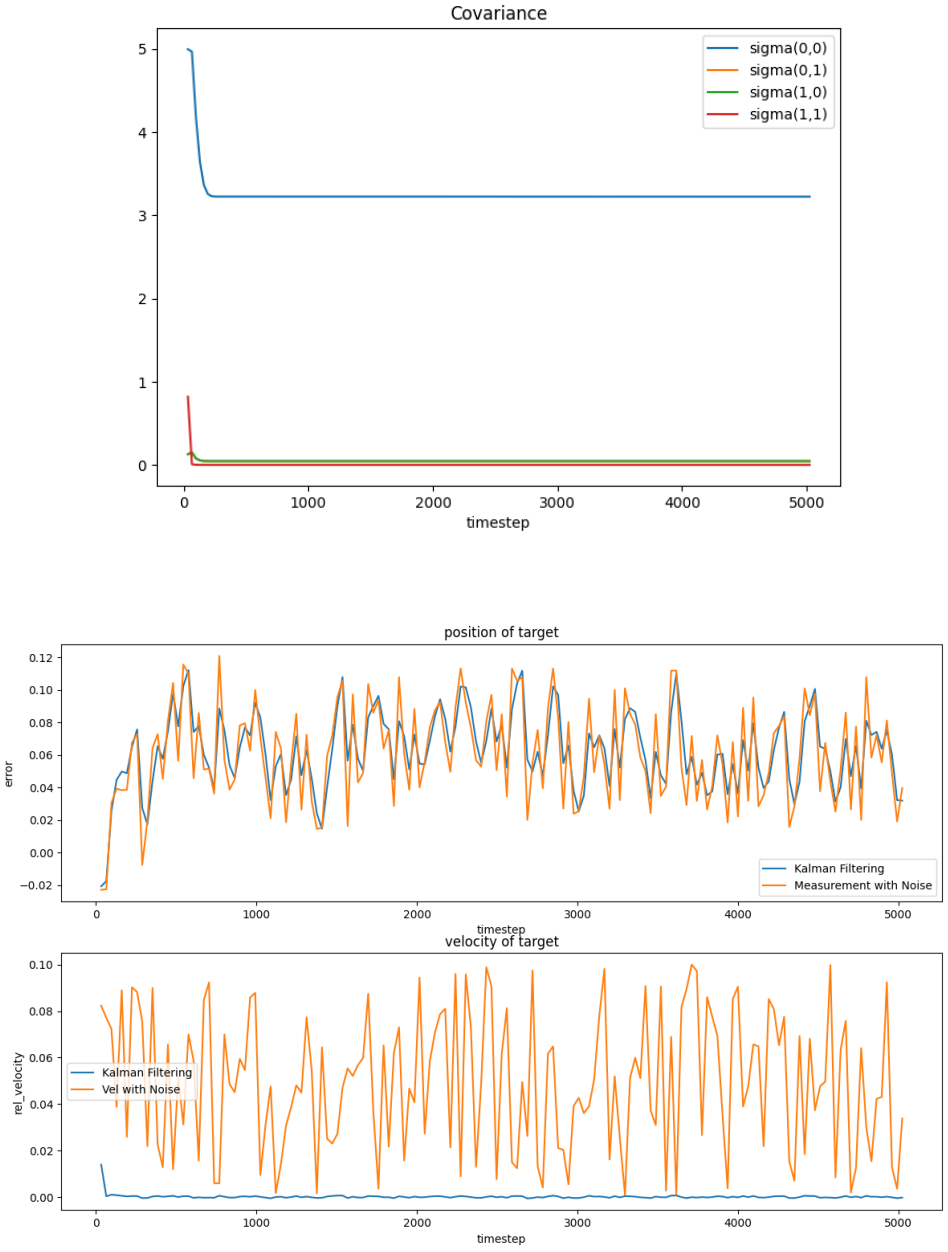

FIGURE 5.2: Covariance graph and Position-Velocity graph of Kalman filter from the simulated environment.

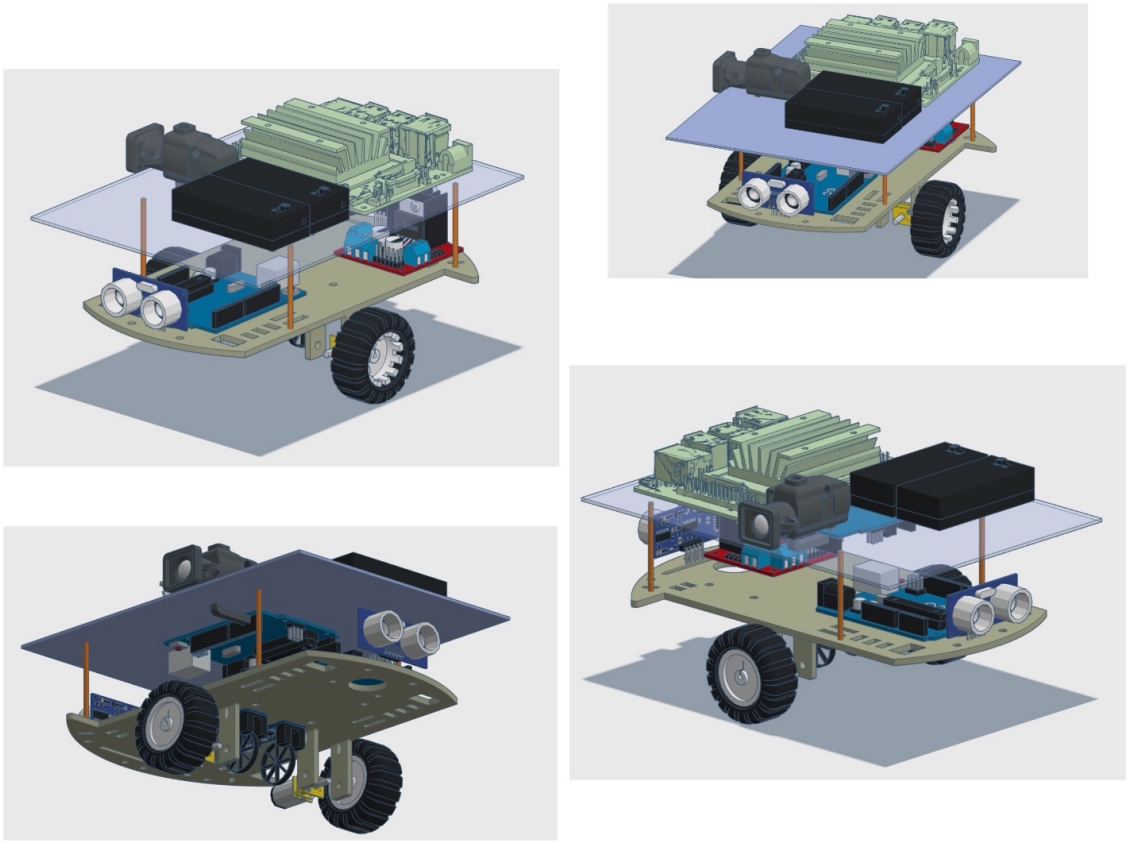

FIGURE 5.3: The initial prototypes of the AGV.

generate a real-world environment and allowed the authors to visualise the project. Crucial sections of the ASV were first enforced and tested thoroughly in the simulation to ideate the overall development procedure. Moreover, the literary and variable-dependent controllers like the PID controller and Kalman filter were also first calibrated and stabilised in the virtual setting before moving into the real hardware stuff. The resultant Covariance graph and Position-Velocity graph of the Kalman filter is illustrated in Figure 5.2. The object recognition and tracking capabilities also proved to be highly accurate, allowing the AGV to pick up and record items with ease. Furthermore, prototypes of the entire architecture of the AGV were initially designed in a simulation setting to make it easier to build and relieve frequent hardware changes. This made the hardware system much more stable. The prototypes of the AGV are described in Figure (5.3). The AGV is designed to be highly adaptable and customizable, with various attachment points for different payloads and accessories. Its compact size and manoeuvrability make it ideal for use in a variety of industries, from manufacturing to healthcare.

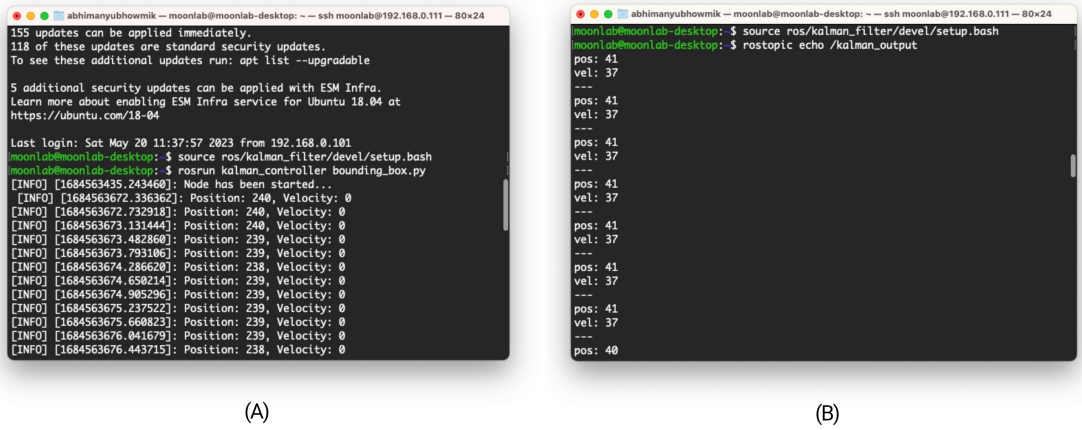

FIGURE 5.4: (A) Bounding box node publishing X axis position and velocity, (B) Kalman Filter Node which has been subscribed to bounding boxes publish refined output of X - position and velocity w.r.t. centre point of the camera.

5.2 Real World Implementation

In robotics, after testing all the algorithms in simulation and designing the hardware components, a project moves into its next phase of actual deployment on real hardware. This project's trajectory isn't an exception to the norm. As described in section 4.3.2 this project is based on **ROS Melodic** with **Ubuntu 18.04**. After setting up the network several ros node has been created which we have already discussed in section 4.3.3. All of the ROS packages are individually created, tested and optimised before being installed into a single Catkin workspace. Which include **darknet_ros**, **kalman_filter** and **pid_controller**. All of the dependencies have been added to respective CMake files. Next, the main Catkin workspace has been built, the necessary launch files have been made, and the **/devel/setup.bash** has been sourced before running the ros nodes. Permission to Jetson USB posts has been granted for its communication to Arduino using Serial Buses.

As a result, the UAV can detect the object and send the information for further processing and from the results it also can follow the detected object. The instances of **bounding_box** node and **kalman_filter** node results are shown in figure 5.4. It's clearly visible that the node can detect the position of the object and process the data to generate the error and relative velocity which is then used in the controller. The rqt graph 5.5 shows how the nodes and topics are interconnected in real time. The **teleop_twist_keybord** instance also can publish velocity commands and have a privilege upon the **pid_controller** node to

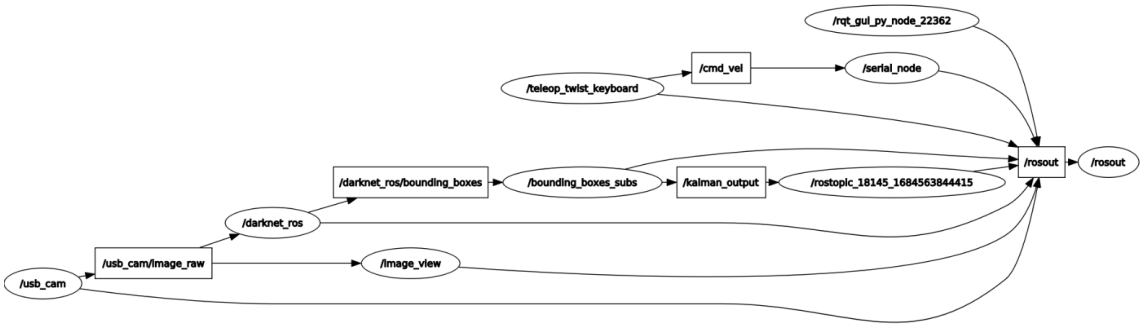

FIGURE 5.5: ROS nodes and topics in RQT graph

assert manual intervention whenever required. The **/cmd_vel** topic has been subscribed by the **rosserial_python** node running on Arduino which is responsible for controlling the motors of the robot. This allows for remote control of the robot's movements through the **teleop_twist_keyboard** instance, while also allowing for automatic control through the **pid_controller** node. The integration of these nodes and topics allows for a flexible and efficient control system for the robot. Furthermore, the **pid_controller** and **rosserial_python** node can also receive feedback from sensors such as encoders or Ultrasonic distance sensors to improve their control accuracy. This allows for a more robust and reliable control system in various applications such as robotics or autonomous vehicles. The successful installation and setup of these packages and dependencies enable the rover to perform various tasks such as object detection, tracking, and navigation. The system can also be further customized and expanded with additional packages for more advanced functionalities.

Chapter 6

Conclusion

A very effective and intelligent robotic system has been created as a consequence of the project on constructing an autonomous mobile robot. The robot demonstrated a surprising level of autonomy by integrating hardware parts, sophisticated software architecture, and cutting-edge technologies, exhibiting its capacity to travel, observe its environment, make decisions, and carry out tasks without direct human interaction.

The project served as a demonstration of the significance of integrating hardware and software when building a reliable and effective autonomous mobile robot. The robot was able to sense its surroundings properly, devise the best courses, and execute challenging tasks with accuracy because of the effective integration of sensors, actuators, and controllers with advanced software algorithms.

The robot's vision and sensing skills turned out to be crucial components of its autonomy. The robot was able to recognise things, find barriers, and adjust to changing situations by utilising computer vision algorithms and sensor fusion techniques. This enabled dependable object manipulation, safe and effective navigation, and obstacle avoidance.

The robot was equipped with a navigation and path-planning system that allowed it to move about its surroundings on its own. Modern algorithms were used by the robot to successfully map its surroundings, plan the best routes, and respond to changes in the environment in real time. The examination of navigational performance in numerous settings demonstrated the system's efficiency and dependability.

In order to facilitate smooth cooperation between the robot and human operators, the project also highlighted the significance of human-robot interaction. Users were able to

operate and get feedback in a natural and user-friendly way from the robot thanks to the deployment of intuitive interfaces and communication systems.

The robot's adaptability and potential were evaluated throughout the project in real-world application settings. The autonomous mobile robot demonstrated its worth by improving productivity, safety, and efficiency across a variety of industries, including industrial automation, logistics, healthcare, and search-and-rescue operations.

Although the project met key milestones, there were certain restrictions and difficulties along the way. These restrictions might have included hardware limits, algorithmic flaws, or performance-impacting environmental conditions. However, these difficulties offer priceless knowledge for upcoming enhancements and developments in autonomous mobile robotics.

The successful end of this project serves as a testament to the enormous potential and capabilities of autonomous mobile robots. The autonomy of the robot is influenced by the integration of hardware and software, sophisticated perception and sensing systems, effective navigation and path planning, dependable object handling, and seamless human-robot interaction. The project acts as a springboard for additional investigation and development in the area of autonomous robotics, fostering the creation of ever-smarter and more competent robotic systems with the potential to transform several sectors and enhance the quality of life.

6.1 Future Works

While the project has reached important milestones, there are still a number of opportunities for new research and advancements in the area of mobile autonomous robots.

1. **Enhanced Perception:** The capacity of the robot to comprehend and interact with its surroundings can be increased with further developments in perception skills, such as enhanced object identification algorithms and higher-resolution sensors.

2. **Intelligent Decision-Making:** By incorporating more sophisticated AI methods like deep learning and reinforcement learning, the robot's ability to make decisions may be improved, allowing it to handle complicated situations and adapt to changing settings more successfully.

3. **Multi-Robot Collaboration:** Researching methods for allowing numerous autonomous robots to cooperate and coordinate their activities may lead to more scalable and effective operations in a variety of applications, such as swarm robotics or warehouse automation.

4. **Long-Term Autonomy:** By creating energy-efficient tactics like adaptive power management or self-recharging capabilities, it will be possible to increase the robot's operating time and give it the ability to carry out activities for extended periods of time without the need for human assistance.

5. **Safety and Ethical Issues:** As autonomous robots become more integrated into human-centric environments, it will be essential to address safety issues and ethical issues related to them, such as ensuring human safety during human-robot interactions or creating ethical standards for robot behaviour.

6. **Real-World Scalability:** In order for autonomous mobile robots to be widely used across a range of sectors, it will be crucial to investigate methods for their smooth integration into current infrastructures, such as creating standardised communication protocols and interoperability frameworks.

Bibliography

[1] Vinay Lanka, Feb 2021.

[2] J. Minguez and L. Montano, Feb 2023.

[3] A. Urquizo, "Pid controller," Dec 2011.

[4] Y. Tian, G. Yang, Z. Wang, H. Wang, E. Li, and Z. Liang, "Apple detection during different growth stages in orchards using the improved yolo-v3 model," *Computers and electronics in agriculture*, vol. 157, pp. 417–426, 2019.

[5] G. Dandabathula, D. Agrawal, S. Lohiya, and S. S. Rao, "Design and implementation of lab grade interplanetary rover for educational purpose,"

[6] "Arduino uno," May 2023.

[7] L. Michal, "jetson," Oct 2021.

[8] A. Dietrich, T. Wimbock, A. Albu-Schaffer, and G. Hirzinger, "Integration of reactive, torque-based self-collision avoidance into a task hierarchy," *IEEE Transactions on Robotics*, vol. 28, no. 6, pp. 1278–1293, 2012.

[9] J. Diebel *et al.*, "Representing attitude: Euler angles, unit quaternions, and rotation vectors," *Matrix*, vol. 58, no. 15-16, pp. 1–35, 2006.

[10] A. Bhowmik, M. Sannigrahi, P. Guha, D. Chowdhury, and S. S. Gill, "Dynamite: Dynamic aggregation of mutually-connected points based clustering algorithm for time series data," *Internet Technology Letters*, p. e395, 2022.

[11] M. Fischer and D. Henrich, "3d collision detection for industrial robots and unknown obstacles using multiple depth images," *Advances in Robotics Research: Theory, Implementation, Application*, pp. 111–122, 2009.

[12] M. Li, R. Kang, D. T. Branson, and J. S. Dai, "Model-free control for continuum robots based on an adaptive kalman filter," *IEEE/ASME Transactions on Mechatronics*, vol. 23, no. 1, pp. 286–297, 2017.

[13] A. M. Zanchettin, N. M. Ceriani, P. Rocco, H. Ding, and B. Matthias, "Safety in human-robot collaborative manufacturing environments: Metrics and control," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 2, pp. 882–893, 2015.

[14] H. Yu, S. Huang, G. Chen, Y. Pan, and Z. Guo, "Human–robot interaction control of rehabilitation robots with series elastic actuators," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1089–1100, 2015.

[15] M. R. Ahmed and I. Kalaykov, "Static and dynamic collision safety for human robot interaction using magneto-rheological fluid based compliant robot manipulator," in *2010 IEEE International Conference on Robotics and Biomimetics*, pp. 370–375, IEEE, 2010.

[16] M. R. Ahmed and I. Kalaykov, "Semi-active compliant robot enabling collision safety for human robot interaction," in *2010 IEEE International Conference on Mechatronics and Automation*, pp. 1932–1937, IEEE, 2010.

[17] J. Lee, P. H. Chang, and M. Jin, "Adaptive integral sliding mode control with time-delay estimation for robot manipulators," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 8, pp. 6796–6804, 2017.

[18] T. Nozaki, T. Mizoguchi, and K. Ohnishi, "Decoupling strategy for position and force control based on modal space disturbance observer," *IEEE Transactions on Industrial Electronics*, vol. 61, no. 2, pp. 1022–1032, 2013.

[19] H. Karami, K. Darvish, and F. Mastrogiovanni, "A task allocation approach for human-robot collaboration in product defects inspection scenarios," in *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pp. 1127–1134, IEEE, 2020.

[20] B. Sadrfaridpour and Y. Wang, "Collaborative assembly in hybrid manufacturing cells: An integrated framework for human–robot interaction," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 3, pp. 1178–1192, 2017.

[21] F. Flacco, T. Kröger, A. De Luca, and O. Khatib, "A depth space approach to human-robot collision avoidance," in *2012 IEEE international conference on robotics and automation*, pp. 338–345, IEEE, 2012.

[22] R. Schiavi, A. Bicchi, and F. Flacco, "Integration of active and passive compliance control for safe human-robot coexistence," in *2009 IEEE International Conference on Robotics and Automation*, pp. 259–264, IEEE, 2009.

[23] A. De Luca and F. Flacco, "Integrated control for phri: Collision avoidance, detection, reaction and collaboration," in *2012 4th IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, pp. 288–295, IEEE, 2012.

[24] G. Antonelli, S. Chiaverini, and G. Fusco, "A new on-line algorithm for inverse kinematics of robot manipulators ensuring path tracking capability under joint limits," *IEEE Transactions on Robotics and Automation*, vol. 19, no. 1, pp. 162–167, 2003.

[25] J. Krüger, B. Nickolay, P. Heyer, and G. Seliger, "Image based 3d surveillance for flexible man-robot-cooperation," *CIRP annals*, vol. 54, no. 1, pp. 19–22, 2005.

[26] K. Harada, S. Kajita, F. Kanehiro, K. Fujiwara, K. Kaneko, K. Yokoi, and H. Hirukawa, "Real-time planning of humanoid robot's gait for force-controlled manipulation," *IEEE/ASME Transactions on Mechatronics*, vol. 12, no. 1, pp. 53–62, 2007.

[27] J. Minguez and L. Montano, "Extending collision avoidance methods to consider the vehicle shape, kinematics, and dynamics of a mobile robot," *IEEE Transactions on Robotics*, vol. 25, no. 2, pp. 367–381, 2009.

[28] G. Du and P. Zhang, "Online robot calibration based on vision measurement," *Robotics and Computer-Integrated Manufacturing*, vol. 29, no. 6, pp. 484–492, 2013.

[29] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[30] S. EPFL, "Webot."

[31] A. Bhowmik, M. Sannigrahi, D. Chowdhury, A. D. Dwivedi, and R. R. Mukkamala, "Dbnex: Deep belief network and explainable ai based financial fraud detection," in *2022 IEEE International Conference on Big Data (Big Data)*, pp. 3033–3042, IEEE, 2022.

[32] E. Ohashi, T. Aiko, T. Tsuji, H. Nishi, and K. Ohnishi, "Collision avoidance method of humanoid robot with arm force," *IEEE Transactions on Industrial Electronics*, vol. 54, no. 3, pp. 1632–1641, 2007.