C-Bot Beach Cleaning Robot

Project Phase - II Report

Submitted By

Aiswarya Jose (Reg. No. SJC20EC007)
Ann Mariya Shaji (Reg. No. SJC20EC027)
Isaac George (Reg. No. SJC20EC049)
Sharon Merin Sabu (Reg. No. SJC20EC084)

to

the APJ Abdul Kalam Technological University
in partial fulfillment of the requirements for the award of the degree

of

Bachelor of Technology

in

Electronics and Communication Engineering

Department of Electronics & Communication Engineering

St.Joseph's College of Engineering & Technology
Palai-686 579

May 2024

St.Joseph's College of Engineering and Technology, Palai

Department of Electronics and Communication Engineering

CERTIFICATE

This is to certify that the report entitled **C-Bot Beach Cleaning Robot** submitted by **Aiswarya Jose (Reg. No. SJC20EC007)**, **Ann Mariya Shaji (Reg. No. SJC20EC027)**, **Isaac George (Reg. No. SJC20EC049)** and **Sharon Merin Sabu (Reg. No. SJC20EC084)**, Eighth Semester B.Tech Electronics and Communication Engineering students, is a bonafide record of the Project Phase - II done by them, under our guidance and supervision, in partial fulfillment of the requirements for the award of the degree, **B.Tech Electronics and Communication Engineering** of **APJ Abdul Kalam Technological University, Kerala**.

Mr.Anto Manuel	**Mr. Sreesh P. R.**	**Dr. Arun. P**
Supervisor	Project Coordinator	Associate Professor & Head
Assistant Professor	Assistant Professor	Dept. of ECE
Dept. of ECE	Dept. of ECE	

Acknowledgment

I wish to record my indebtedness and thankfulness to all who helped me prepare this Project Phase - II Report titled **C-Bot Beach Cleaning Robot** and present it in a satisfactory way.

I would like to convey my special gratitude to **Dr. V.P. Devassia**, Principal, SJCET, Palai,for the moral support he provided. I express my sincere thankfulness to **Dr. Arun. P**, Head of the department,Department of Electronics and Communication Engineering for his co-operation and valuable suggestions. Also I express my sincere thanks to project co-ordinators **Mr. Sreesh P. R.** and **Mr. Anto Manuel** for their helpful feedback and timely assistance.

I am especially thankful to our guide **Mr.Anto Manuel**,Assistant Professor, Department of Electronics & Communication Engineering for giving me valuable suggestions and critical inputs in the preparation of this report.

I also extend my thanks to all our friends who helped us by giving motivation. Their smallest piece of advice was really valuable. Once again we convey our gratitude to all those persons who had directly or indirectly influenced our work as a whole.

Aiswarya Jose
Ann Mariya Shaji
Isaac George
Sharon Merin Sabu
St.Joseph's College of Engineering & Technology
Palai-686 579.

Abstract

Beaches are not only popular tourist destinations but are also an asset to the environment. In these days, the pollution on beaches are increasing day by day. The major wastes that are accumulated on the beach shore includes broken glass pieces, medical waste, jellyfish, plastic bottles and bags, rusted metal, etc. These waste ends up in the sea if they are not collected properly and manual collection of this waste will result in health problems for beach workers. The C-bot is a beach cleaning robot that collects waste with minimum human intervention. It uses a combination of sensors, such as ultrasonic sensors, IR sensors, cameras, and GPS to detect obstacles and navigate its path respectively. We implement the image detection and processing using the micro controller ESP 32 which has an inbuilt camera module which captures the images.The C-Bot operates on the sandy shores at all hours of the day. Our aim is to design and develop an affordable, easily portable and environmental friendly machine that solves the issue of beach pollution.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Oceans account for 70 percent of the surface of Earth and play a vital role in the health of human beings. According to the National Oceanic and Atmospheric Administration (NOAA), heaps of trash and other pollutants enter our oceans every year. This has became a serious issue these days. To tackle the situation of proper disposal of waste, maintaining hygiene and cleanliness of the beach we came up with the idea of C-bot, the beach cleaning robot specifically for cleaning beaches. It has the ability to operate on dry terrain, effectively collecting trash and other forms of waste. The robot uses image processing algorithms to detect various objects on the beach and locate them within the robot's field of view.

The system, finds the waste using a camera module which is integrated with the microcontroller ESP 32-CAM. The processing of image is done in such a manner that, ESP 32-CAM looks for the dataset for identifying the waste. The robots path is identified using path planning algorithm. Once the controller identifies the waste, the motor turns OFF and the motor of the arm turns ON and pick up the waste. This type of implementation will improve the systems usability and reduce power consumption. As per estimates, Kerala generates about 450 tonnes of plastic every day, 590km coastline needs an urgent cleansing and around 70 per cent of this waste ends up in the sea. The figure 1.1 and 1.2 shows the amount of waste accumulated on the beach shore.

The C-Bot is envisioned as an innovative robotic system that operates on the sandy shores, harnessing the power of the sun through integrated solar panels. Its primary mission is to efficiently and autonomously clean beaches while mitigating any adverse impact on the surrounding environment. In this paper we will explore the design, operation, and potential effects on beach laborers, highlighting its contribution to cleaner and healthier beaches while furthering the goals of

environmental conservation and sustainable growth.

Figure 1.1: Plastic accumulated at the Mamallapuram beach [1]

Figure 1.2: Plastic accumulated at the Kovalam beach [2]

Chapter 2

Literature Review

The given below are the research papers we referred for developing and expanding our project. Based on what we learnt from these papers, we were able to develop our block diagram, need metric matrix, target specifications etc.H. Ebrahim et al. [3] proposed a beach cleaning robot in 2022 titled ”Design and Analysis of an Environmentally Friendly Beach Cleaning Machine” which examines beach cleaning projects with the aim of addressing their limitations. The study provides an analysis of how these machines function, including motion and stress graphs. Our designed machine combines features derived from their research findings. The machine operates on a chain sprocket arrangement, where the sprocket turns the chain and the attached rakes lift up any trash in its path. However we identified some limitations in the design, such as difficulties in picking up debris and a tendency to collect any obstacles encountered along the way. To overcome these challenges our prototype includes a mesh that enables us to collect debris effectively. Additionally we integrated surveillance into the system to assist operators in avoiding picking up anything, than debris.

In the research conducted by N. Bolong et al. [4] in 2021, they focused on the development and analysis of a beach cleaning machine, at UMS prototype. The team successfully combined raking and sifting mechanisms to create the machine. It comprises components such as a collector, conveyor, motor and gears that need to be operated. The primary goal was to construct an affordable beach cleaning machine that can be easily handled by one person. The paper also provides an overview of how the prototype functions, on site testing results and feedback obtained from users surveyed during the study.

Prof. J. Shelke et al.[5] proposed a Beach cleaning system and surface cleaning system in 2020. In this study, a cleaning system with four base motors for

the robot's movement and a front servo motor to operate the arm for collecting garbage is proposed. Among the fundamental mechanisms are the control system's receiving requests or directions from the user through the HC-05 Bluetooth module, carrying out the cleaning process with the assistance from a claw-shaped arm that is fastened to the servo motor. Utilizing an Atmega32 microcontroller, the data is processed mandates. In this project, the beach and surface cleaning system's maximum communication range is 10 meters. Automatic operation of the system as well as control from remote distance are its limitations. V. Mepani et al. [6] proposed Design and Fabrication of Beach Sand Cleaning Machine in 2022. This paper compares and evaluates the several techniques used to clean beaches, as well as the construction of a low-cost manually powered sand cleaner. The conveyor moves in tandem with the machine's action when it is pushed ahead. The conveyor belt is indirectly rotated by the shaft's movement, and the debris is further locked in by the spokes on the belt, which moves it up the conveyor belt and deposits it in a container.[4] Although this machine's rake mechanism is quite effective at clearing waste, it is not suitable for use on beaches or in smaller spaces where the ecosystem is delicate. Our chosen method is made up of a sifter that gathers both larger and finer trash and sand on a mesh screen, allowing the sand to return to the beach bed via the mesh.

A. O. Qasim et al. [7] developed a Modular Type Beach Cleaning Robot 'Clean-B'. The 'Clean-B' robot is characterized by its modular design, which implies that it is composed of interchangeable and replaceable modules that facilitate ease of maintenance and customization according to specific requirements. This modular architecture allows for flexibility in adapting the robot to different beach environments and cleaning tasks. The machine is designed to autonomously navigate beach terrains and perform cleaning operations efficiently. It incorporates various sensors and actuators to detect and collect debris such as trash, seaweed, and other forms of pollution commonly found on beaches. Additionally, the paper likely discusses the integration of artificial intelligence and advanced algorithms to enhance the robot's capabilities in object detection, navigation, and obstacle avoidance. These features enable the 'Clean-B' robot to navigate complex beach environments while effectively identifying and removing debris.

D. Varghese et al. [8] developed a machine named Binman: An Autonomous Beach Cleaning Robot in 2022. Binman's design incorporates advanced robotics and sensing technologies to efficiently detect and remove debris from coastal areas. The robot's autonomy is achieved through the integration of sophisticated algorithms and sensors, enabling Binman to navigate complex beach terrains and identify various types of waste. The machine's functionality revolves around its

ability to detect and classify different objects commonly found on beaches, including plastic waste, organic materials, and other forms of litter. The robot incorporates a solar power system, reducing its reliance on non-renewable energy sources and minimizing its ecological footprint.

Chapter 3

Requirement Analysis

3.1 Mission Statement

Our mission is to design and develop a solar based beach cleaning robot that is easily portable, user-friendly by implementing motor drivers, motors and sensors to collect all types of waste particles and finally store the waste.

3.2 Objective

The primary purpose of our project is to maintain the cleanliness of beaches. This involves the removal of all types of waste, such as plastics, glass, seaweed, etc from the beach that may cause injuries to workers. The robot also reduces the labor required for manual beach cleaning, which can be less time-consuming. The main reason for choosing this project is the heaps of waste accumulated on the sea shore that causes severe health problems to humans and also causes death of aquatic life.

3.3 Motivation

The motivation for our project are due to the following reasons:

1. Increasing pollution in beaches.

   With growing urbanization, industrialization, and tourism, beaches face increasing pollution from plastic debris, microplastics, and other waste. This

pollution not only harms marine life but also poses health risks to humans and compromises the beauty of coastal areas. Traditional manual cleaning efforts are often insufficient to cope with the scale of the problem and can be labor-intensive and time-consuming. Beach cleaning robots offer a promising solution by automating the process, efficiently collecting and disposing of litter, and covering vast areas with greater speed and precision. By harnessing robotics and technology, these machines aim to mitigate the environmental impact of pollution, restore the ecological balance of coastal ecosystems, and preserve the natural beauty of beaches for future generations.

Beyond their practical utility, beach cleaning robots serve as powerful symbols of innovation and environmental stewardship. By showcasing the potential of technology to address pressing environmental challenges, they inspire public engagement, raise awareness about the importance of beach conservation, and foster a sense of collective responsibility towards safeguarding our coastal ecosystems for future generations.

Figure 3.1: Workers clean up waste in the beach shore[2]

2. Beach cleaning is often labor-intensive and time-consuming.

   Beach cleaning is typically demanding and time-consuming, relying heavily on manual labor. This process not only strains resources but also poses environmental challenges due to the vast amounts of waste that accumulate on beaches worldwide. Introducing beach cleaning robots addresses these issues by offering a more efficient, cost-effective, and environmentally friendly solution. These robots can autonomously navigate sandy terrains,

collecting debris efficiently while minimizing human effort. By automating the cleaning process, they significantly reduce the time and labor required, making beach maintenance more sustainable and accessible. Moreover, these robots contribute to preserving coastal ecosystems, safeguarding marine life, and enhancing the overall appeal of beaches for visitors. Thus, the motivation behind developing beach cleaning robots stems from the need to streamline beach maintenance operations, mitigate environmental impact, and ensure the long-term sustainability of coastal areas. Moreover, beach cleaning robots can operate continuously, day or night, regardless of weather conditions. Unlike human workers, they are not limited by factors such as fatigue or safety concerns, ensuring consistent and uninterrupted cleaning efforts. This round-the-clock operation is crucial for tackling the ever-growing problem of marine debris, which requires constant vigilance and action.

Furthermore, beach cleaning robots offer scalability and adaptability to different beach environments and conditions. They can be programmed and customized to target specific types of debris or pollution hotspots, allowing for tailored cleaning strategies based on local needs and priorities. This versatility ensures that beach cleaning efforts are optimized for maximum effectiveness and environmental benefit. In conclusion, the motivation for developing beach cleaning robots stems from the pressing need to address the challenges posed by marine pollution in a more efficient, cost-effective, and sustainable manner.

Figure 3.2: Workers clean up waste in the beach shore[2]

3. Causing injuries to beach workers.

   C-bot is motivated by the need to mitigate the risk of injury to beach workers caused by manual cleaning activities. These workers often face hazards such as sharp objects, toxic waste, and physical strain while cleaning beaches manually. By introducing robots, the aim is to reduce human exposure to such dangers and enhance safety protocols. Additionally, beach cleaning robots can work continuously without fatigue, covering large areas efficiently. This not only improves the safety of workers but also enhances the overall cleanliness and attractiveness of beaches for visitors. Ultimately, the motivation behind these robots is to create a safer and more sustainable environment for both beach workers and beachgoers while maintaining the natural beauty of coastal areas. Beyond physical strain, beach workers are also exposed to numerous health hazards, including sharp objects hidden in the sand, such as broken glass or metal, which can cause lacerations and puncture wounds.

   In light of these challenges, developing a beach cleaning robot offers a compelling solution to enhance worker safety and efficiency. By automating the task of debris collection and disposal, the robot can alleviate the physical strain and reduce the direct exposure of workers to hazardous materials. Equipped with sensors and cameras, the robot can navigate the beach terrain, detect obstacles, and identify and safely remove various types of debris. Ultimately, the deployment of beach cleaning robots can revolutionize coastal maintenance practices, transforming hazardous manual labor into a safer, more sustainable, and environmentally friendly endeavor. By prioritizing worker safety and well-being, these innovative technologies contribute to creating healthier and more resilient coastal communities for both humans and marine ecosystems alike.

Chapter 4

Survey

4.1 Offline Survey

For the field survey we have visited Shanghumugham beach and Vettucaud beach in Trivandrum. The fig 4.1 shows debris in the beach area.

Figure 4.1: Waste accumulated in Shanghumugham beach shore

The workers cleaned the beach around 2-3 times daily .Some of their concerns were: They usually encountered sharp or hazardous materials like broken

glass, medical waste, jellyfish or rusted metal, which require caution and proper handling. They find it difficult to deal with vast amounts of trash and debris that accumulate on the sea shore.

Figure 4.2: Discussing the difficulties faced by beach workers

Beach waste is a major environmental problem that has an impact on coastal communities all around the world. It harms marine life, pollutes the water, and makes beaches unsightly and unsafe for recreation. To learn more about the types and amounts of waste found on beaches, the following survey was conducted.

The survey was conducted on 10 September, 2023 at Shangumugham Beach, Trivandrum. Our team surveyed the beach and we noticed different types of waste like plastic, glass, cigarette butts, metal, clothing, footwear, fish waste(jellyfish), fishing gear, and personal hygiene products. We also had a good interaction with

Figure 4.3: Collecting information from beach cleaning authorities

the municipal staff who cleaned the beach 2-3 times daily.Some of their concerns are: They usually encounter sharp or hazardous materials like broken glass, medical waste, jellyfish or rusted metal, which require caution and proper handling. They find it difficult to deal with vast amounts of trash and debris that accumulate on the sea shore.

The beach waste survey conducted provided valuable insights into the urgent issue of pollution on the coastline.The results highlight the immediate need for measures to protect the beach environment and ensure a clean and pleasant experience for visitors. We also believe that our project will have a good impact on this growing problem as well. The figure 4.2 and 4.3 shows discussing the difficulties faced by beach workers.

4.2 Online Survey

From the survey we concluded that,

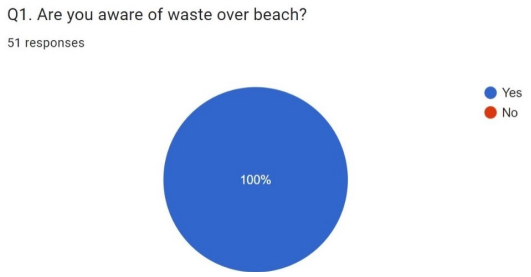


Figure 4.4: Survey Response of aware of waste over beach

- Plastics are the major wastes found in beaches.

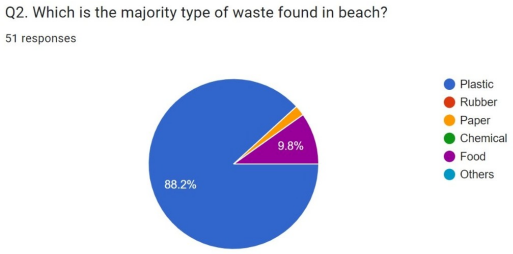


Figure 4.5: Survey response of the majority type of waste found in beach

- Rather than collecting waste manually, machine mode seems much better option.
- Waste segregation method is needed and segregration should be based on their type or size.
- Daily waste collection practice lead to hygiene and safety environment.

Figure 4.4 shows the response of are people aware of waste over beach. Even though all are aware of beach pollution still they neglect to manage the cleanliness of beach. The beach cleaning robot survey assesses waste presence on beaches

using sensors for detection. It identifies debris types, quantities, and distribution, aiding in robot navigation and waste collection optimization. This data-driven approach enhances efficiency, ensuring comprehensive cleaning while minimizing environmental impact.

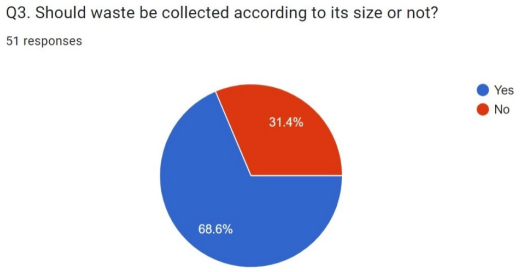


Figure 4.6: Survey response of should waste be collected according to its size or not

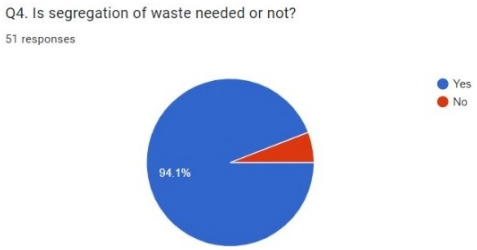


Figure 4.7: Survey response of segregation of waste whether needed or not

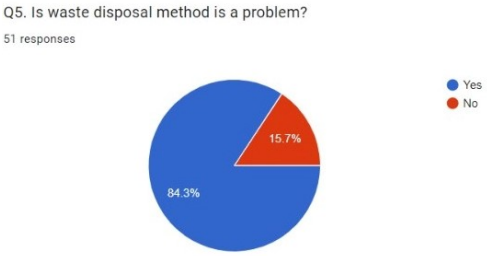


Figure 4.8: Survey response of is waste disposal method is a problem

Figure 4.5 shows the response of majority type of waste found in beach. The response shows that plastic is the majority type and is followed by food waste; the minority waste found is paper. The survey for beach cleaning robots aims to identify the predominant types of waste on beaches. It assesses various factors such as plastic, glass, organic matter, and metals. This data informs the design and functionality of robots to efficiently target and remove the most prevalent beach pollutants.

Figure 4.6 shows response of should waste be collected according to the size or not. The beach workers usually encounter sharp or hazardous materials like broken glass, medical waste, jellyfish or rusted metal, which require caution and proper handling. The survey seeks opinions on beach cleaning robots: whether waste should be collected based on size or not. Participants will provide feedback on the efficiency and practicality of sorting debris by size, aiding in the design and functionality of future beach-cleaning technologies.

Figure 4.7 shows the response of does the segregation of waste is needed or not. The survey for a beach cleaning robot seeks opinions on whether waste segregation is necessary. Participants assess the importance of sorting waste (plastic, organic, etc.) during beach cleaning operations. Results aim to inform the design and functionality of the robot to optimize waste management practices and environmental impact.

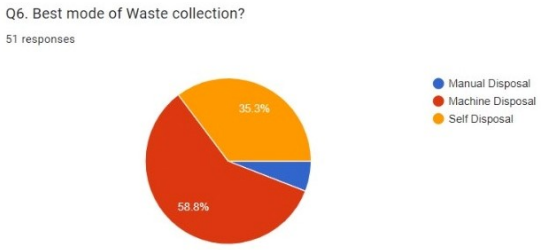


Figure 4.9: Survey response of best mode of waste collection

Figure 4.8 shows the response of waste disposal method. A survey on beach cleaning robots assesses their efficacy as waste disposal solutions. It examines factors like collection efficiency, environmental impact, and user acceptance. Insights gleaned inform improvements for efficient, sustainable beach cleaning, mitigating marine pollution, and preserving coastal ecosystems.

Figure 4.9 shows the response of best mode os waste collection. The survey

assesses preferences for beach cleaning methods, focusing on efficiency, cost, and environmental impact. Participants evaluate various modes, including manual, mechanical, and robotic. Results indicate robotic collection as optimal due to its ability to cover large areas, minimize human effort, and reduce ecological disturbance. From the response we concluded that machine disposal is the best mode of waste collection.

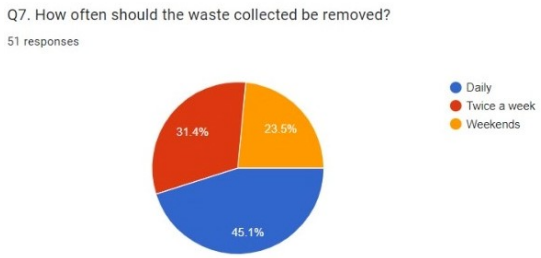


Figure 4.10: Survey response of how often the waste collected be removed

Figure 4.10 shows the response of how often should the waste collected be removed. The survey for beach cleaning robots seeks to determine optimal waste removal frequency. Participants will suggest intervals based on factors like beach traffic and waste accumulation rates. The goal is to balance efficiency with environmental preservation, ensuring timely waste removal without excessive disruption to beachgoers or ecosystems.

4.3 Target Users

The main users of our system are:

1. Municipalities and government agencies.

   The target users of beach cleaning robots are primarily municipalities and government agencies responsible for maintaining public beaches. These entities oversee large coastal areas that require regular cleaning to ensure environmental sustainability and public safety. Beach cleaning robots offer an efficient and cost-effective solution for removing debris, litter, and pollutants from shorelines. Municipalities and government agencies benefit from the robots' ability to navigate various terrains, including sandy beaches, efficiently covering vast areas that would be time-consuming for

manual cleanup crews. Parks and recreation authorities are another key user group. They oversee the management of public spaces, including beaches, and strive to provide enjoyable recreational experiences for visitors.

Additionally, these robots contribute to preserving marine ecosystems by reducing the impact of litter and pollutants on coastal wildlife. By investing in beach cleaning robots, municipalities and government agencies demonstrate their commitment to environmental stewardship and public welfare.

2. Coastal residents

   The target users of beach cleaning robots are coastal residents, including individuals, communities, and organizations residing near coastlines worldwide. These robots cater to those concerned with environmental conservation, marine ecosystem health, and the aesthetic appeal of their local beaches. Coastal residents often experience firsthand the detrimental effects of pollution on their shores, including plastic waste accumulation and debris from various sources.

   Local governments and municipal authorities responsible for beach maintenance and environmental protection are key stakeholders in the adoption of beach cleaning robots. These robots offer a cost-effective and efficient solution for managing beach cleanliness, complementing traditional cleanup efforts and reducing the workload of manual labor. By utilizing these robots, coastal residents can actively participate in preserving their natural surroundings, promoting tourism, and fostering a sense of environmental responsibility within their communities.

3. Beach-goers

   The target users of a beach cleaning robot are beach-goers, individuals who frequent beaches for leisure, recreation, or relaxation. These users include families, tourists, environmental enthusiasts, and local communities who value pristine shorelines. The beach cleaning robot aims to enhance their experience by ensuring cleaner, safer, and more enjoyable beaches. By automating the tedious task of litter collection, the robot allows beach-goers to focus on enjoying their time by the sea without the distraction or inconvenience of trash scattered along the shore.

   Overall, the target users seek a cleaner and more sustainable beach environment, and the beach cleaning robot caters to their needs by efficiently maintaining the coastal cleanliness.

4. Beach workers

   The target users of beach cleaning robots are primarily beach workers tasked with maintaining the cleanliness of shorelines. These workers often face challenges in manually collecting trash and debris scattered across vast stretches of beaches. Beach cleaning robots offer a technological solution to streamline and enhance their efforts. Firstly, beach workers benefit from the efficiency and effectiveness of a beach cleaning robot. These autonomous or semi-autonomous machines are equipped with sensors and advanced technology that enable them to navigate sandy terrain, detect and collect various types of waste, and operate in both wet and dry conditions. By automating the cleaning process, these robots can significantly reduce the time and effort required by beach workers to maintain cleanliness.

   Overall, beach cleaning robots empower beach workers by augmenting their capabilities and improving the effectiveness of their cleanup efforts.

4.4 Customer Needs

1. The device should be user-friendly.

   The customer needs for a beach cleaning robot center around usability and effectiveness. Primarily, the device must be user-friendly to enable easy operation for various users, including beach maintenance personnel and volunteers. This entails intuitive controls, simple maintenance procedures, and clear instructions for setup and operation. Additionally, the robot should efficiently remove debris from the beach, including plastics, seaweed, and other waste, to maintain cleanliness and safety for visitors and wildlife. Durability and reliability are also crucial to ensure uninterrupted operation in diverse beach environments. Ultimately, the customer seeks a solution that streamlines beach cleaning efforts while minimizing manual labor and environmental impact. A user-friendly beach cleaning robot addresses these needs by combining ease of use with efficient performance, contributing to the preservation and enjoyment of coastal environments.

2. It should effectively collect all types of waste.

   The primary customer need for a beach cleaning robot is efficient waste collection, encompassing all types of debris. This includes plastics, organic matter, glass, and other pollutants. The robot must possess adept sensors

and mechanisms to identify and gather various types of waste efficiently, ensuring thorough cleaning of the beach environment. Additionally, it should operate seamlessly across different terrains such as sand and shallow water, covering large areas within a reasonable timeframe. A key requirement is robustness, capable of withstanding the harsh beach environment and potential encounters with debris or obstacles. Ease of maintenance and operation are also crucial for user satisfaction, enabling beach authorities or operators to deploy and manage the robot effectively for sustainable beach maintenance.

3. It should be waterproof.

   A beach cleaning robot needs to be waterproof to effectively operate in the sandy, wet environment of beaches. Waterproofing ensures the robot can withstand exposure to water, waves, and moisture without damage, enabling it to clean efficiently and reliably. Beyond water resistance, the robot should possess robust cleaning capabilities, including the ability to navigate various terrains, collect debris effectively, and dispose of waste appropriately. It should also feature intelligent sensors for obstacle detection, efficient power management for extended operation, and user-friendly controls for ease of use. Ultimately, a waterproof beach cleaning robot fulfills the essential customer need for reliable, all-weather cleaning performance along coastal areas, contributing to the preservation and enjoyment of pristine beaches worldwide.

4. The device should be compactible.

   The primary customer need for a beach cleaning robot is efficiency in removing debris while being compact and easy to maneuver. Compatibility is essential for seamless integration into diverse beach environments, accommodating various terrains and obstacles without causing disruptions to natural habitats. A compact design ensures the robot can navigate tight spaces, including crowded beaches or narrow paths near sand dunes, effectively collecting litter without disturbing beachgoers or wildlife. Moreover, compatibility extends to maintenance and operational requirements, with user-friendly interfaces and interchangeable components for simplified upkeep. Overall, the beach cleaning robot must address these needs by balancing functionality with portability, enabling efficient and unobtrusive litter removal to preserve the beauty and ecological integrity of coastal areas.

5. The device should not collide with others.

   The primary customer need for a beach cleaning robot is efficient debris removal while ensuring safety and minimal disruption to beachgoers and wildlife. The device must navigate the beach terrain autonomously, avoiding obstacles such as people, animals, and beach equipment, to prevent collisions and ensure user safety. By employing advanced sensors and algorithms, the robot can detect and circumvent obstacles in its path, enhancing its efficiency and effectiveness in cleaning operations. Additionally, incorporating features such as audible warnings or visual indicators can alert beach visitors to the robot's presence, promoting awareness and cooperation. Ultimately, addressing the customer need for collision avoidance enables the robot to operate seamlessly within the beach environment, fulfilling its purpose of preserving the natural beauty of the coastline while prioritizing safety and convenience for all stakeholders.

Chapter 5

Need Metric Matrix

		Metric	Material	Power supply	Sensors	Dataset	Weight	Size	Path planning	Speed	Camera
Sl.No.	**N**eeds		1	2	3	4	5	6	7	8	9
1	Durability		x		x						x
2	Safety		x	x		x	x		x	x	x
3	User friendly		x	x		x	x		x	x	
1	Controlability		x	x	x		x		x	x	x
1	Load capacity		x	x			x			x	
1	Easy Implementation		x				x	x			
1	Energy efficient			x	3						x
1	Water proof		x					x			x
1	Battery life			x	x					x	x
1	Optimum cost	x		x				x			x

Table 5.1: Need Metric Matrix

The table 5.1 shows the need metric matrix, formulated using the requirements of target users during site visit.

Chapter 6

Target Specification

Metric No.	Need No.	Metric	Marginal Value	Ideal Value	Unit	IMP
1	2,3,4	Path Planning	Obstacle avoidance	Real time operation	-	1
2	1,2,4,7,8,9,10	Camera	180	180	Degree	2
3	2,3,4,5,9	Speed	1-2	2-4	km/h	3
4	6,8,10	Size	8x15x9	6x13x7	inch	4
5	2,3,4,5,6	Weight	2	20	kg	4
6	1,4,7,9,10	Sensors	-	Intruder detection	-	4
7	1,2,3,4,5,6,8,10	Material	Polymer	ABS	-	5
8	2,3,4,5,7,9	Power supply	-	15	V	5

Table 6.1: Target Specification

Target Specification is shown in the table 6.1, formulated by analyzing need metric matrix.

Chapter 7

Proposed System

7.1 Use case diagram

A use case diagram is a graphical representation of the interactions between a system (the subject under consideration, often a software application or a business process) and its external users or other systems (referred to as actors). It illustrates the various ways users or external systems interact with the system to achieve specific goals or tasks. The figure 7.1 shows the use case diagram of the proposed system, the system represents the overall beach cleaning robot system. When power is activated, the ESP32 Cam and Arduino turn on. The ESP32 Cam captures images, which are used for object detection. This use case describes the robot's functionality of detecting objects on the beach. Once an object is detected, the robot stops its movement. The diagram suggests that upon object detection, the wheel motor stops and the arm motor turns on to collect the waste. It also represents a cloud storage system where the collected data is presumably uploaded for further analysis.

The human users who can interact with the system in some way. They can monitor the system's status, send commands to the robot, or view collected data. This use case describes a functionality where the robot can be stopped in an emergency situation. It describes the system's ability to display the temperature, possibly for monitoring purposes. This actor represents the general public who can benefit from a clean beach. Overall, this use case diagram provides a high-level overview of the beach cleaning robot system and how it interacts with its environment.

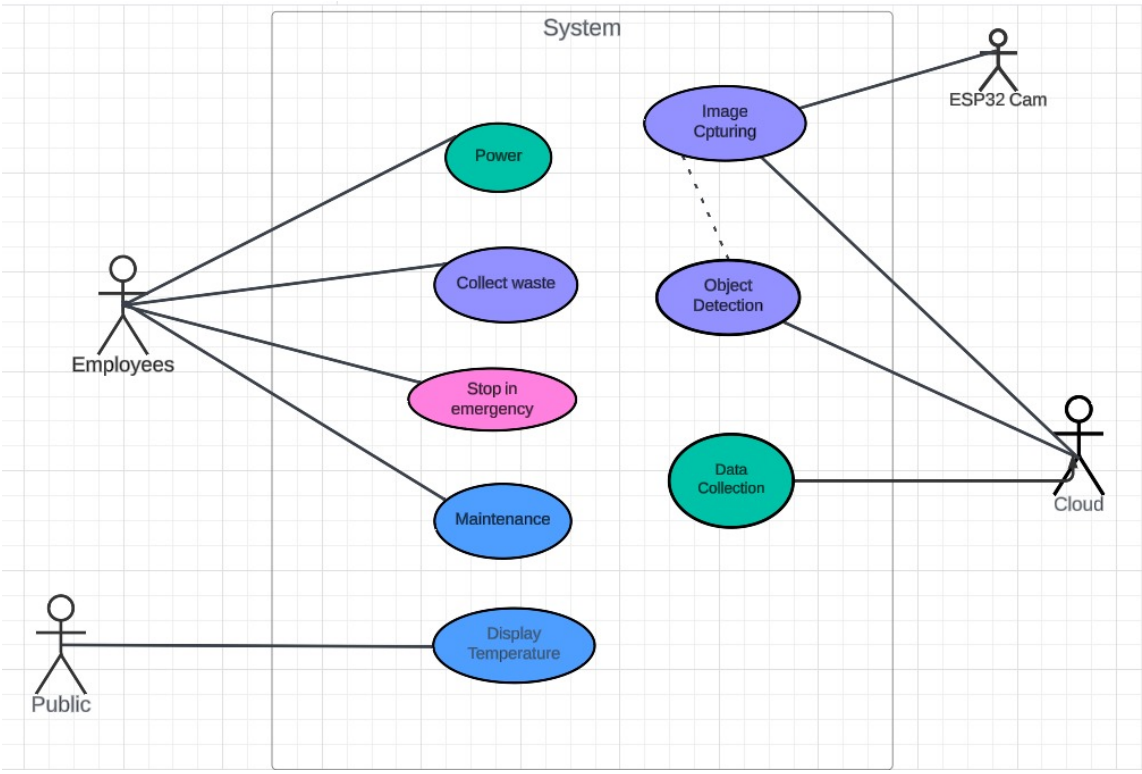


Figure 7.1: Use Case Diagram

7.2 Sequence Flow Diagram

The figure 7.2 shows the sequence flow diagram of the proposed system, the process begins when the system is powered on. The ESP32 Cam and Arduino likely boot up and initialize. The ESP32 Cam captures an image of the beach environment. The captured image is then uploaded to the cloud. An image detection service in the cloud analyzes the uploaded image. The image detection service determines if there is any waste in the image.

The results of the image analysis, likely consisting of a 'waste detected' signal or 'no waste detected' signal, are sent back to the robot. If waste is detected, a signal is sent to the robot instructing it to stop its movement. The robot's arm motor is activated, initiating the waste collection process. Once the waste collection is complete, the sequence restarts to capture the image again.

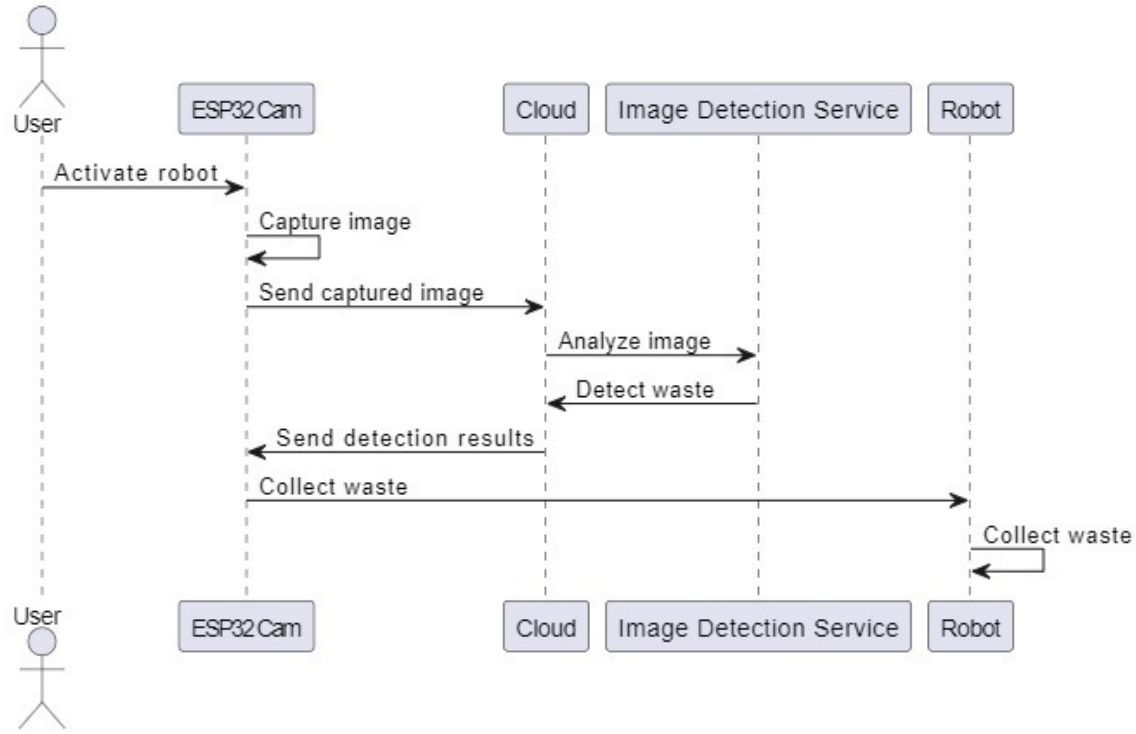


Figure 7.2: Sequence Flow Diagram

7.3 Block Diagram of Proposed System

Figure 7.3 shows the block diagram of the proposed system.It consists of 12 input blocks and 5 output blocks

7.4 Block Diagram Description

The block diagram is shown in the above figure 6.1 and each part is explained as below.

1. Power Supply

   Every electronic gadgets require power for its working. The total power provided by the battery is 15.8V. The output from L298N, ESP32 CAM and

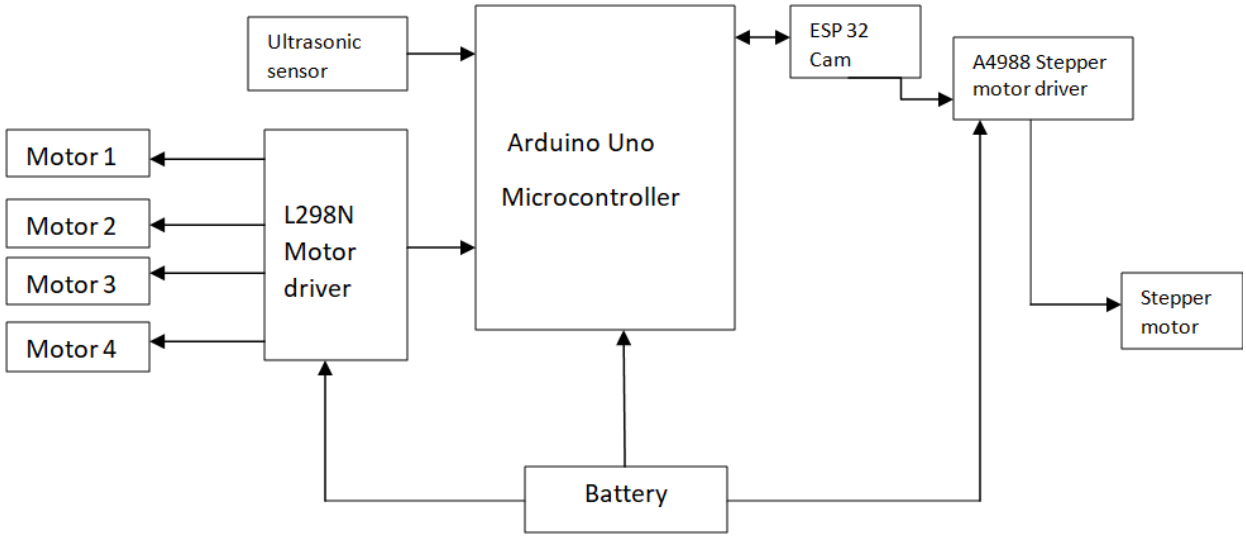


Figure 7.3: Block Diagram

ultrasonic sensor is 5V which is connected to the Arduino. The minimum output voltage for A4988 id 8V.

2. ATmega328P

   The heart of the Arduino Uno is ATmega328P microcontroller. It's an 8-bit microcontroller. It operates at a clock speed of 16 MHz, providing enough processing power to all devices.

3. ESP32 CAM

   The main application of ESP32-CAM is to capture the image of the waste. The capturing is done during the camera module OV2640. It then fed the image to the server for the identification process. The server then sends back a signal to the ESP32-CAM. When the image is detected the dc motor turns OFF and the stepper motor of the arm turns ON.

4. Ultrasonic Sensor

   It measures distances between the object and the robot. It helps the robot detect obstacles or objects in its path. This is crucial for obstacle avoidance and navigation. The robot can use ultrasonic sensors to map its surroundings and navigate through the beach without colliding with obstacles.

5. Database

   We need to create a specific database which in turn will be used in the micro controller for the image processing. So with the help of the data we can be able to feed information to the micro controller and the micro controller can analyze the data and arrive at a possible conclusion.

6. L298N Motor driver

   The L298N is an integrated circuit (IC) that is commonly used to control DC motors. It is used to control the speed and direction of DC gear motor. The H-bridge design allows the L298N to control the direction of the current through the motor, which allows the motor to be rotated in both directions.

7. DC Gear motor

   The motor converts electrical energy into mechanical energy. We use a total of 4 motors attached to the wheel which helps in the movement of the entire system.

8. A4988 Motor driver

   A4988 is specifically designed for driving bipolar stepper motors. It can control both the direction and the number of steps taken by the motor.

9. Stepper motor

   Stepper motors are DC brush-less and synchronous motors. They rotate in discrete steps of predefined values and are able to rotate both clockwise and anticlockwise. We use a stepper motor for operating the arm of the robot.

Chapter 8

Circuit Diagram

8.1 Detailed Circuit Diagram

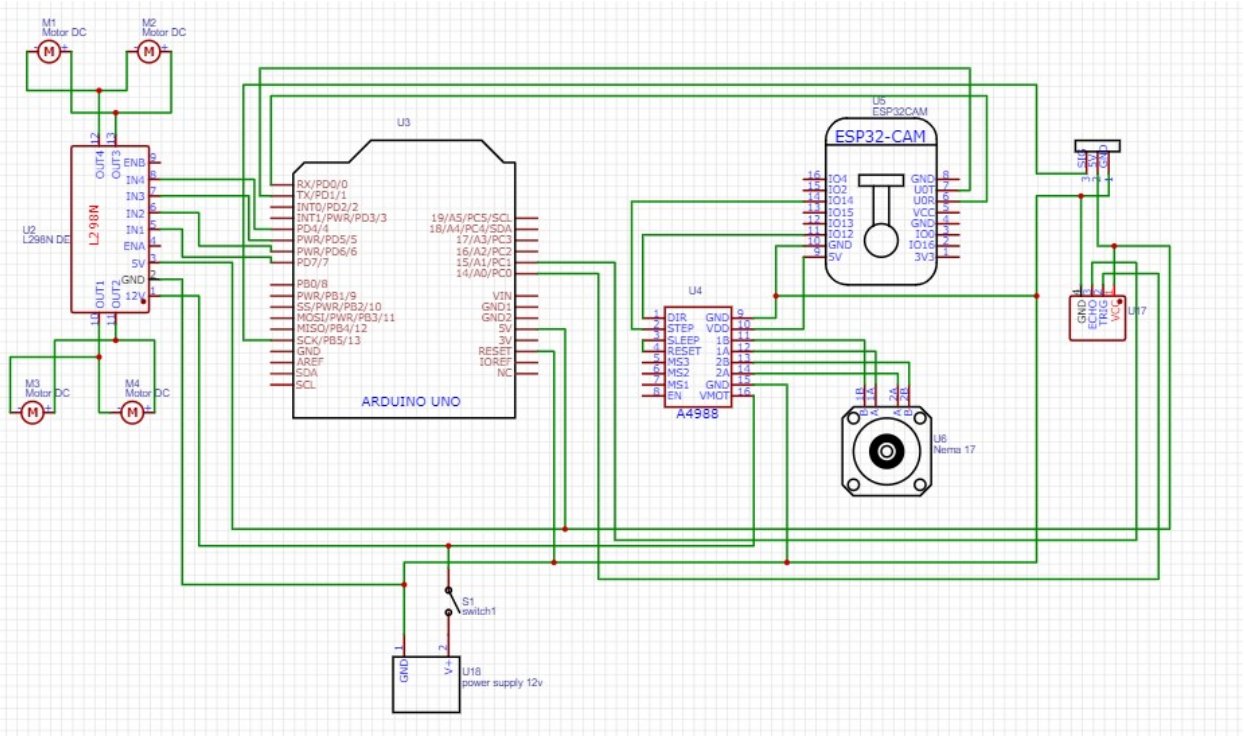


Figure 8.1: Circuit Diagram

The detailed circuit diagram is shown in the figure 8.1. When the power supply is turned on, the Arduino and ESP32 microcontrollers receive power and start executing their respective programs. The Arduino and ESP32 boot up and initialize their peripherals and pins according to the programmed instructions. Initially, upon power-up or as part of the startup routine, the Arduino sends signals to the DC gear motor driver to activate the motor. The driver receives the signals and starts rotating the DC gear motor, which is used for the entire system. The ultrasonic sensor attached to the ESP32 CAM which continuously monitor the environment for any types of waste. When an object is detected within a certain range by the ultrasonic sensor, it sends a signal to the Arduino or ESP32 indicating the presence of an object. Upon receiving the signal from the ultrasonic sensor, the ESP32 sends a command to deactivate the DC gear motor.

Simultaneously or shortly after deactivating the DC gear motor, the Arduino or ESP32 sends a command to activate the stepper motor attached to the arm responsible for waste collection. The stepper motor rotates in a controlled manner to position the arm at the desired location for waste collection.Additionally, a servo motor is used to change the direction of the ultrasonic sensor. Upon receiving instructions from the Arduino, the servo motor adjusts the orientation of the ultrasonic sensor to scan different areas effectively for waste detection.

With the help of Arduino UNO, the ESP32-CAM is programmed. First part of program is to initialize the camera module with the board. After the initialization we use the html code to fetch the data given by the ESP32-CAM board. Then we use the data and store it in the server for appropriate image classification process. The image classification process includes several processes which further converts the color image into binary bits.

8.2 Components Used

8.2.1 ESP32-CAM

The main application of ESP32-CAM is to capture the image of the object. The capturing is done by the camera module OV2640. It is then fed as an image to the server for the identification purpose. If the waste is detected, the server sends back a signal to the ESP32. The figure 8.2 shows ESP32-CAM.

Figure 8.2: ESP32-CAM[23]

8.2.2 Resistors

A resistor is an electrical component with two terminals that implements electrical resistance as a circuit element. Its main function is to control the flow of electric current in a circuit, limiting the amount of current that can pass through it. They are passive components with no moving parts, making them highly re-

Figure 8.3: Resistor[24]

liable and durable. The resistor incorporated in our circuit is 100k. The power rating is 0.5 watt for the resistor.The following figure 8.3 shows the diagrammatic representation of the resistor.

8.2.3 Capacitor

A capacitor is an electronic component used to store and release electrical energy in circuits. It consists of two conductive plates separated by an insulating material known as a dielectric.

Figure 8.4: Capacitor[25]

When voltage is applied across the plates, electric charge accumulates on them, creating an electric field between the plates. The capacitors incorporated in our circuit is 22uf and two 103pf. The figure 8.4 shows the capacitor.

8.2.4 Crystal oscillator

A crystal oscillator is an electronic device that generates electrical signals with precise frequencies. It utilizes the mechanical resonance of a vibrating crystal, typically made of quartz, to produce a stable frequency output. The natural resonance frequency of the quartz crystal is highly stable over time and temperature variations. It generally consumes low power, making them energy-efficient and suitable for battery-powered devices or applications where power consumption is a concern. For the project we use a 16MHz crystal oscillator. The figure 8.5 shows the crystal oscillator.

Figure 8.5: Crystal oscillator[26]

8.2.5 DC gear motor

A DC gear motor is a type of electric motor combined with a gearbox to provide torque amplification and speed reduction. Like any electric motor, a DC gear motor converts electrical energy into mechanical energy, generating rotational motion.

Figure 8.6: DC gear motor[27]

The gear reduction mechanism allows DC gear motors to produce high torque, making them suitable for applications requiring significant force or load handling. DC gear motors can easily change the direction of rotation by reversing the polarity of the applied voltage. The use of gears can also help reduce noise and vibration generated by the motor. We use a 200rpm dc motor for the project. The figure 8.6 shows the dc gear motor.

8.2.6 L298N

The L298N is a popular dual H-bridge motor driver integrated circuit. Its main function is to control the direction and speed of DC motors and stepper motors. It can drive motors in both directions. It can handle relatively high currents, typically up to 2A per channel, which makes it suitable for driving a wide range of motors.

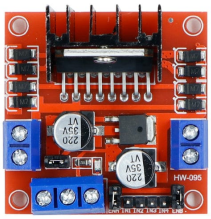


Figure 8.7: L298N motor driver[28]

The L298N can typically operate over a wide range of supply voltages, making it suitable for various applications. Common operating voltages include 5V, 12V, and 24V, making it compatible with a wide range of power sources. The L298N is a robust IC that can handle moderate to high currents, depending on the specific variant and configuration. The figure 8.7 shows l298N motor driver.

8.2.7 Stepper motor

Figure 8.8: Stepper motor [29]

A stepper motor is a type of electric motor that divides a full rotation into a number of equal steps. Each step corresponds to a discrete angular movement, allowing precise control over positioning without the need for feedback sensors. Stepper motors can provide high torque at low speeds, making them suitable for applications that require high holding torque. The figure 8.8 shows stepper motor.

8.2.8 A4988

A4988 is specifically designed to control bipolar stepper motors. The A4988 includes built-in current regulation, allowing users to set the motor current limit through a simple potentiometer adjustment. This feature helps prevent motor overheating and provides optimal torque for the application.

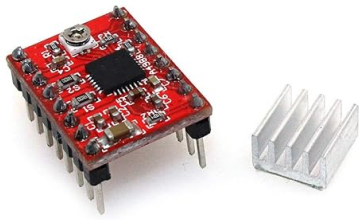


Figure 8.9: A4988[30]

The A4988 includes various protection features such as thermal shutdown, overcurrent protection, and under-voltage lockout, which help prevent damage to the chip and connected components in case of faults or abnormal conditions. The figure 8.9 shows A4988.

8.2.9 Arduino uno

The Uno is built around the ATmega328P microcontroller, which provides processing power and I/O capabilities for controlling electronic circuits and devices. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6

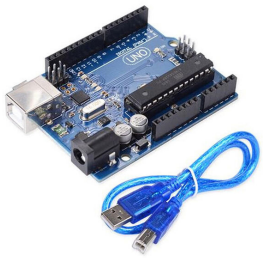


Figure 8.10: Arduino uno [31]

analog input pins, and a variety of communication interfaces (UART, SPI, and I2C), allowing it to interface with a wide range of sensors, actuators, displays, and other peripherals. The figure 8.10 shows arduino uno.

8.2.10 Servo motor

A servo motor(figure 8.11) is a rotary actuator that allows for precise control of angular position. It consists of a motor coupled to a sensor for position feedback. It also requires a servo drive to complete the system. The driver uses the feedback

Figure 8.11: Servo Motor[32]

sensor to precisely control the rotary position of the motor.

8.2.11 LM2596 buck converter

The LM2596 is a popular voltage regulator integrated circuit (IC) that is widely used as a buck (step-down) converter in various electronic applications. The primary function of the LM2596 is to step down higher input voltages to lower output voltages. This makes it useful for powering circuits that require a stable voltage lower than the input source.

Figure 8.12: Servo Motor[32]

8.3 Software used

8.3.1 Arduino IDE

Arduino Integrated Development Environment shown in figure 8.12 (IDE) is an open-source software used for programming Arduino micro controllers. It pro-

Figure 8.13: Arduino IDE[33]

vides a user-friendly interface for writing, compiling, and uploading code to Arduino boards, making it accessible for beginners and advanced users alike in the world of electronics and programming.

8.3.2 Fusion 360

Fusion 360(figure 8.13) is a cloud-based 3D modeling software developed by Autodesk. It offers a comprehensive set of tools for designing, simulating, and manufacturing 3D models and prototypes. Users can create assemblies of multiple

Figure 8.14: Fusion 360 [34]

components and simulate how they interact with each other, enabling the design of complex mechanisms and systems. Fusion 360 includes built-in simulation tools that allow users to analyze the structural, thermal, and modal behavior of their designs.

8.3.3 Google Slides

Google Slides(figure 8.14) is a cloud-based presentation tool by Google. It allows users to create, edit, and share presentations online. With collaborative features, real-time editing, and a variety of templates, it is widely used for creating and delivering presentations in a team or educational setting.

Figure 8.15: Google Slides [35]

8.3.4 LaTeX

LaTeX(figure 8.15) is a typesetting system commonly used for the production of scientific and mathematical documents due to its powerful handling of complex equations and references. It uses plain text commands to format and structure doc-

Figure 8.16: LaTeX [36]

uments, providing high-quality output for academic papers, theses, and technical reports.

8.3.5 EasyEDA

EasyEDA(figure 8.16) is an online platform that offers tools for designing electronic circuits and printed circuit boards (PCBs). It provides a user-friendly interface for schematic capture, PCB layout design, and circuit simulation. Users can create, edit, and share their designs collaboratively in real-time. EasyEDA also offers a vast library of electronic components, making it convenient for users to find and incorporate parts into their designs. Additionally, it supports exporting designs in various formats for fabrication and manufacturing.

Figure 8.17: LaTeX [37]

Chapter 9

Implementation

9.1 Breadboard implementation

The circuit was implemented in the breadboard level which was used for testing the motor.A specimen test was conducted and it is confirmed that the image detected was waste. Figure 9.1 shows the breadoard implementation.

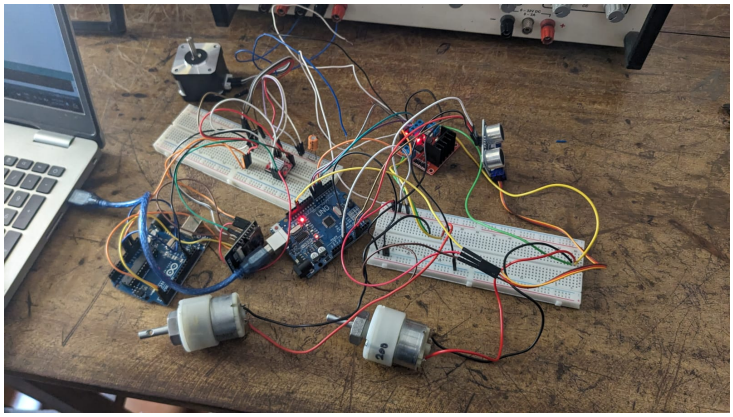


Figure 9.1: Breadboard implementation

9.2 PCB implementation

The PCB layout was drawn using the tool EasyEDA. Using layout PCB design was completed.

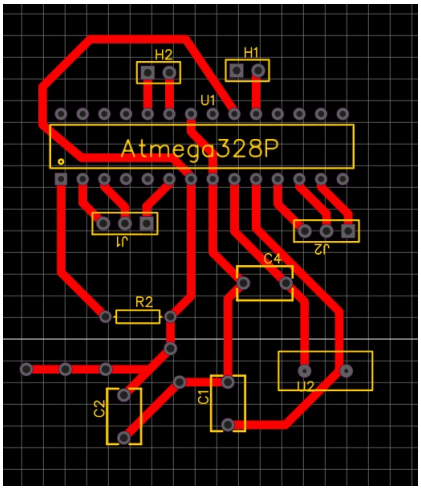


Figure 9.2: PCB Layout

Figure 9.3: PCB Design

The figure 9.2 shows the PCB Layout drawn using EasyEDA. The figure 9.3 shows PCB design.

Chapter 10

Bill of Materials

The below figure 10.1 shows the Bill of Materials.

SI.no	Item	Specification	Unit number	Unit cost	Total cost(Rs)
1	ATMega328P	AVR Microcontroller	1	650	650
2	Ultrasonic sensor	5V DC	1	150	150
3	DC Gear motor	200rpm	4	290	1160
4	ESP32-CAM	WiFi+Bluetooth	1	526	526
5	Resistors	10k	1	5	5
6	Capacitor	103pf 22uf	2 1	5 3	 13
7	Crystal oscillator	16MHZ	1	15	15
8	L298N	-	1	294	294
9	A4988	-	1	145	145
10	Stepper motor	3000rpm	1	650	650
11	Servo motor	-	1	189	189
12	BMS	15.8V	1	245	245
13	Form Board	48x27inch	-	-	340
14	Robotic wheel	9cm	4	127	508
15	Battery case	-	1	50	50
16	Nut,bolt,L bracket	-	20	-	80
				Grand Total	5020/-

Table 10.1: Bill of Materials

Chapter 11

Waste Detection and Identification

The waste detection is done using micro controller ESP32-CAM. The ESP32-CAM has a camera module which captures the image of the waste. After capturing the image of the object, the frames are transmitted to the cloud via ESP32-CAM. After comparing the captured image with the input database from the cloud, if the object is detected, then a signal is transmitted to one of the pin in ESP32-CAM, which turns OFF the DC motor and turns ON the stepper motor of the arm which collects the waste that was detected.

11.1 Algorithm for detection and identification

The algorithm used is CNN(Convolutional Neural Network).It is a type of deep learning algorithm commonly used for image recognition and classification tasks, although they can also be applied to other types of data such as audio and text.

11.1.1 Convolutional Neural Network

A Convolutional Neural Network(figure 11.1) is a type of deep learning model designed for processing structured grid data, such as images. It consists of multiple layers, including convolutional layers, pooling layers, and fully connected layers. Convolutional layers apply filters to input images, extracting features like edges or textures. Pooling layers downsample the feature maps, reducing computation. Fully connected layers combine extracted features for classification or regression tasks. CNNs excel in tasks like image recognition, object detection, and facial

recognition due to their ability to automatically learn hierarchical patterns from data, making them fundamental in modern computer vision applications.

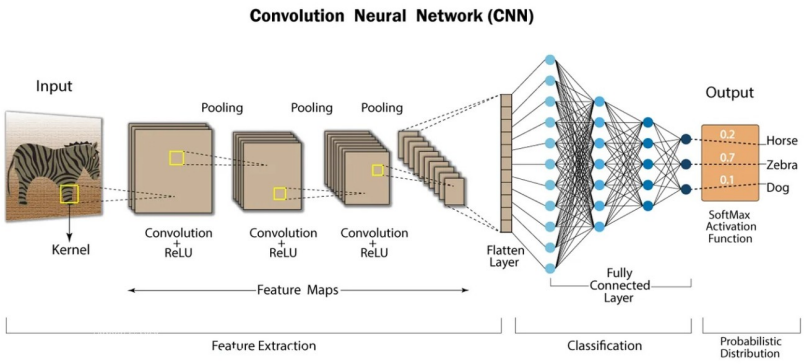


Figure 11.1: Convolutional Neural Network[33]

Convolutional layer

A Convolutional Layer in a Convolutional Neural Network (CNN) is the core building block for extracting features from input data, such as images. It applies filters to input data, sliding them across the input's spatial dimensions. Each filter detects specific patterns, like edges or textures, creating feature maps. Convolutional layers leverage parameter sharing to reduce the number of trainable parameters, aiding in capturing hierarchical features. Pooling layers often follow to downsample the feature maps, enhancing computational efficiency and promoting spatial invariance. The figure 11.2 shows the convolutional layer.

RLU Layer

The rectified linear unit is known as RLU. The next step is to transfer the feature layer maps to a RLU layer after they have been extracted. RLU performs an operation element-by-element, setting all the negative pixels to zero. The result is a corrected feature map, and it gives the network non-linearity. f(x) is defined in the neural network literature as the maximum (0, x) for a ReLU that accepts neuron input as x. As a result, the rectifier is defined as f(x) = max (0, x). A number of convolutions and ReLU layers are applied to the original image to find the features.

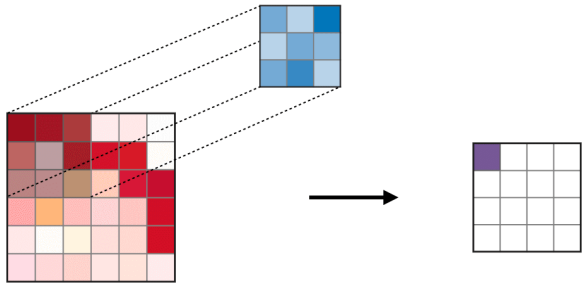


Figure 11.2: Convolutional Layer[34]

Pool Layer

The Pooling layer downsamples the spatial dimensions of its input feature maps, reducing computational complexity and controlling overfitting. Common pooling methods like Max Pooling retain the maximum value within a defined window, while Average Pooling computes the average. Pooling helps in capturing dominant features while reducing dimensionality, maintaining translation invariance, and enhancing computational efficiency. By abstracting spatial information, it aids in learning hierarchical representations, contributing to the CNN's ability to recognize patterns robustly across different regions of an image.

Flattering

Flattering typically involves presenting subjects, stories, or perspectives in a positive or appealing light to engage viewers and convey credibility. This can manifest through selective framing, highlighting achievements, or showcasing favorable traits. Flattery in CNN's reporting often aims to captivate audiences, build trust, and maintain viewer loyalty. Overall, flattering in CNN serves to inform, entertain, and connect with audiences while upholding journalistic standards of integrity and fairness. The figure 11.3 shows the flattering.

Fully connected layer

A fully connected layer connects every neuron from the previous layer to every neuron in the current layer, enabling the network to learn complex relationships in the data. Each neuron in the fully connected layer receives input from all the neu-

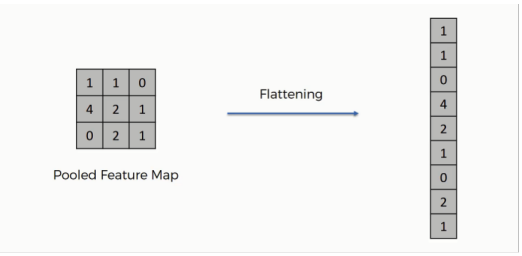


Figure 11.3: Flattering[35]

rons in the preceding layer, making it densely connected. This layer is typically used after the convolutional and pooling layers to flatten the output into a single vector, which is then fed into one or more fully connected layers. These layers are crucial for high-level feature extraction and classification tasks in CNNs. The figure 11.4 shows the fully connected layer.

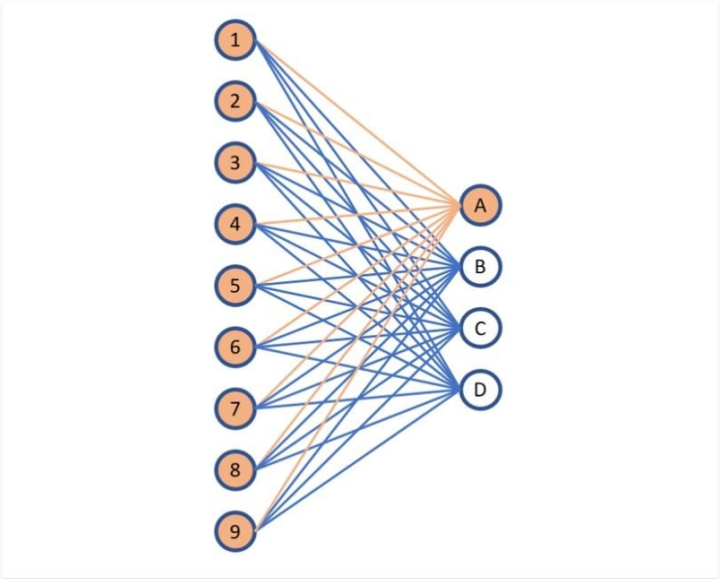


Figure 11.4: Fully connected layer[36]

11.1.2 Algorithm and Flowchart

- **Step 1:** Start.
- **Step 2:** Initialize ESP32 Camera module.
- **Step 3:** Check if ESP32 module is connected to network.
- **Step 4:** If not connected ,go to step 3,else go to step 5.
- **Step 5:** Get an IP address to broadcast.
- **Step 6:** Check if the camera has started capturing video.If not ,wait for a 500ms,else go to step 7.
- **Step 7:** Encrypt and send to webserver.
- **Step 8:** Access the live feed of the camera for classification.
- **Step 9:** Check for plastic bottle,if no,wait and remain in step 9,else move to step 10.
- **Step 10:** Turn ON the port in ESP32-CAM.
- **Step 11:** Stop the motor.
- **Step 12:** Initiate waste collection.
- **Step 13:** Stop.

The below diagram 11.5 shows the flowchart.

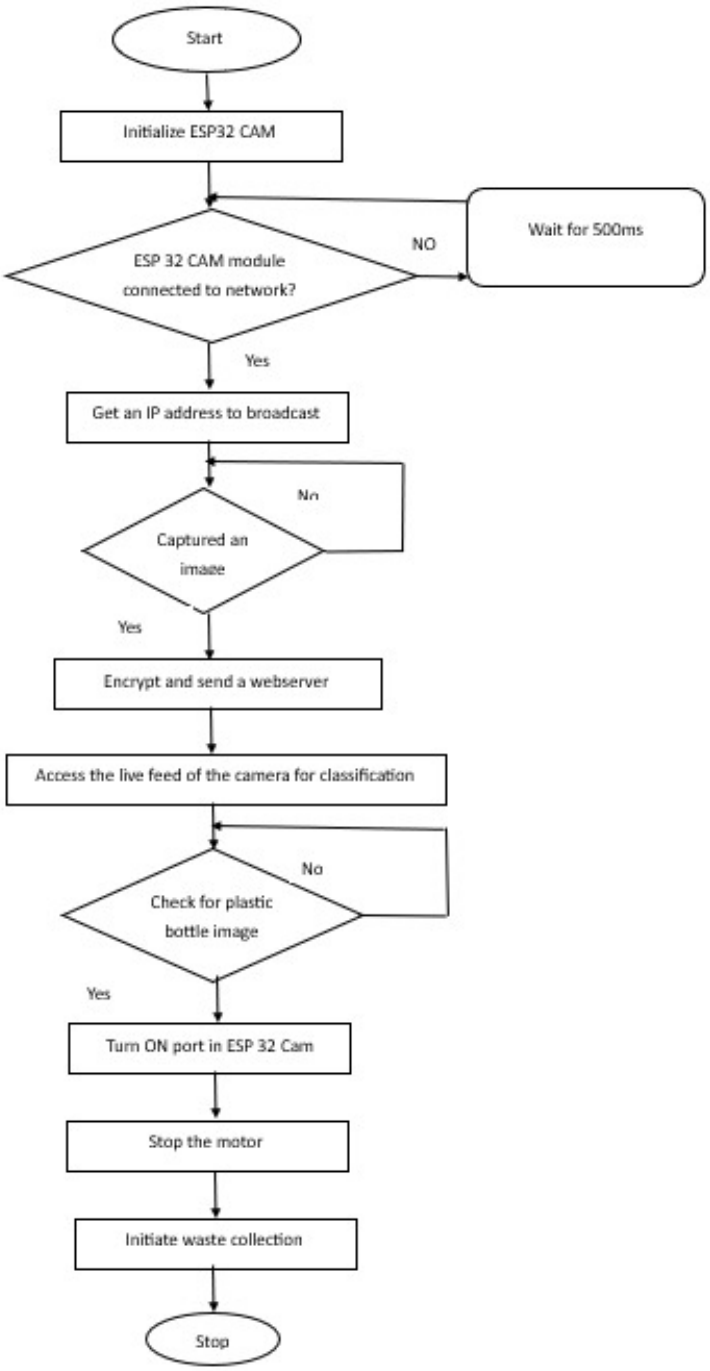


Figure 11.5: Flowchart

Chapter 12

Results and Conclusions

A detailed study was conducted regarding the beach pollution in kerala. Literature reviews of existing systems like BeBot, BeachBot, and Beach cleaning robot have been discussed briefly. These robots offer a sustainable and efficient way to address the issue of beach pollution, combining renewable energy technology. A field survey had been conducted for identifying the target requirements and customer needs, for this purpose, we have visited Shanghumugham beach and Vettucaud beach in Trivandrum. The main target users where the local government authorities and municipal workers. The power is driven by four motors on each wheel, with electrical power provided by a battery pack inside the vehicle. The implementation of circuit was done using breadboard. The PCB was also developed according to the circuit diagram. The layout for the PCB was designed using EasyEDA. The figure 12.1 and 12.2 shows the top view and front view of the project.

The C-Bot represents an innovative step toward a sustainable future for our coastlines.

However, the success of beach cleaning robots hinges not only on technological innovation but also on collaborative efforts among governments, NGOs, and private stakeholders. Sustainable waste management practices, public awareness campaigns, and policy interventions are essential components of a holistic approach to safeguarding our coastlines for future generations.

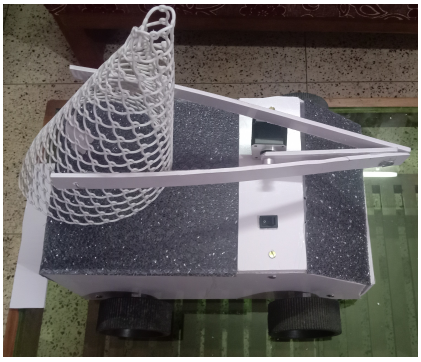


Figure 12.1: Top view

Figure 12.2: Front view

Chapter 13

Future Scope

The future scope of this project is both broad and multifaceted, encompassing various domains of research and technological innovation. By providing a solar panel to the sytem, the robot becomes a renewable energy and photovoltaic systems. This scope extends to investigating energy management algorithms, which balance the power requirements of the robot's cleaning mechanisms, further contributing to the overall sustainability and effectiveness of the technology.

The beach cleaning robot can be equipped with sophisticated sensors and algorithms to autonomously navigate the dynamic and irregular beach areas, recognize and classify waste, and apply efficient cleaning strategies. Additionally, the development of advanced materials for the robot's mechanical components is crucial to ensure durability and resistance to saltwater corrosion. These aspects of robotics research have broad applications beyond beach cleaning, including areas such as search and rescue, agriculture, and environmental monitoring.

Moreover, the environmental science dimension of the scientific scope is vital. This includes studies on the ecological consequences of various cleaning mechanisms and the evaluation of the robot's efficiency in reducing beach pollution. In summary, the scientific scope of a solar-based beach cleaning robot spans renewable energy technology, robotics, artificial intelligence, materials science, and environmental science, addressing critical challenges in beach maintenance and environmental management while offering insights and innovations applicable in various other domains.

Bibliography

[1] PRIYA CHAUHAN, "55,000 kg Plastic Waste Collected from South Kerala Ocean Within Two Years,"[Online] planetcustodian. https://www.planetcustodian.com/plastic-waste-collected-from-the-ocean-in-kerala/11316/

[2] Kumar Vishal, "Kerala Govt will collect plastic to construct roads,"[Online] thelogicalindian. https://thelogicalindian.com/environment/kerala-govt-will-construct-roads-using-waste-plastic/

[3] H. Ebrahim, W. Sheikh, A. Saeed, "Design and analysis of sustainable beach cleaner", 3C Technology. Glosses of innovation applied to SMEs, Special Edition, ISSN: 2254-4143, https://doi.org/10.17993/3ctecno.2022.specialissue9.167-179, 14 Feb 2022.

[4] N. Bolong, I. Saad, M. Amran Madlan, "Manufacturing of Beach Cleaning Machine at University Malaysia Sabah (UMS) Prototype Design and Analysis", Transactions on Science and Technology, Volume: 08, No. 3-2, 281 - 289, 02 Nov 2021.

[5] Prof. J. Shelke, B. Bhakare, K. Lute, A. Pateshwari, H. Khodiyar, "Beach cleaning system and surface cleaning system", International Research Journal of Modernization in Engineering Technology and Science, Volume:02, Issue:06, e-ISSN: 2582-5208, June 2020.

[6] V. Mepani, H. Patel, Vataliya Mohil, Prof. R. Sahu, "Design and Fabrication of Beach Sand Cleaning Machine", International Research Journal of Engineering and Technology, Volume: 07, Issue: 02, e-ISSN: 2395-0056, p-ISSN: 2395-0072, Feb 2020.

[7] Chandra, S. S., Kulshreshtha, M., Randhawa, P. (2021). A Review of Trash Collecting and Cleaning Robots. 2021 9th Interna-

tional Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions), ICRITO 2021, September 2021. https://doi.org/10.1109/ICRITO51393.2021.9596551

[8] D. Varghese and A. Mohan, "Binman: An Autonomous Beach Cleaning Robot," MysuruCon 2022 - 2022 IEEE 2nd Mysore Sub Sect. Int. Conf., pp. 1–5, 2022, doi: 10.1109/MysuruCon55714.2022.9972499.

[9] A. O. Qasim, A. A. Varghese, A. Sarkar, and S. Justus, "Modular Type Beach Cleaning Robot 'Clean-B,'" Proc. - Int. Conf. Augment. Intell. Sustain. Syst. ICAISS 2022, pp. 1231–1234, 2022, doi: 10.1109/ICAISS55157.2022.10010711.

[10] J. Shalini Priya, K. T. Balaji, S. Thangappan, and G. Yuva Sudhakaran, "Beach Cleaning Bot Based on Region Monitoring," 8th Int. Conf. Comput. Power, Energy, Inf. Commun. ICCPEIC 2019, pp. 1–4, 2019, doi: 10.1109/ICCPEIC45300.2019.9082368.

[11] A. VIshwakarma, P. Vishwakarma, and S. Waghunde, "Beach Cleaning Robot," vol. 6, no. 12, p. 71, 2015.

[12] P. Kaladharan, K. Vijayakumaran, V. V. Singh, D. Prema, P. S. Asha, Bindu Sulochanan, P. Hemasankari, L. Loveson Edward, Shelton Padua, S. Veena, A. Anasukoya H. M. Bhint, "Prevalence of marine litter along the Indian beaches: A preliminary account on its status and composition", Journal of Marine Biological Association of India, Volume: 59, No 1, June 30, 2017.

[13] C. Balasuthagar, D. Shanmugam, K. Vigneshwaran, "Design and fabrication of beach cleaning machine", IOP Conf. Series: Materials Science and Engineering 912 (2020) 022048 IOP Publishing doi:10.1088/1757-899X/912/2/022048, 2020.

[14]] Prathamesh Jangam, Sneha jangam, Rutuja kadu, Mubbashir Kazi, Prof. Sanobar Shaikh, "Beach Cleaning Robot", International Research Journal of Engineering and Technology (IRJET), ISSN: 2349-6002, Volume:6 Issue:12, May 2020.

[15] Ramamoorthi R, Ramachandran N, Nikiles PD, Jayasurya R, Natheesh MD, Nithin K Biju, "Design and fabrication of beach cleaning machine", International Journal of Innovative Technology and Exploring Engineering, ISSN:2278-3075, Volume: 08, Issue:12, Oct 2019.

[16] D. Vaishnavi, Sabeesh Kumar.S, "Swachh yantramanav: A multipurpose cleaning robot", International Journal of Innovative Science and Research Technology, ISSN:2456-2165, Volume:02, Issue:05, May 2017.

[17] M.Bhavani, S.Kalaiselvan, S.Jagan, S.Gopinath, "Semi-Automated Wireless Beach Cleaning Robot Vehicle", International Journal of Recent Technology and Engineering (IJRTE), ISSN: 2277- 3878, Volume:8 Issue:1S2, May 2019.

[18] Dr. F B Sayyad, Dr. Md. Imran Ansari, Dr. S F Sayyad, "Design and Development of Beach Cleaning Machine", International Journal for Research in Applied Science and Engineering Technology, ISSN: 2321-9653, Volume:07, Issue:06, June 2019.

[19] T. Subba Reddy, P. Satya Priyanka, L. Himaja, K. Sravani, N. Mounika, "Design and Fabrication of Beach Dust Collector", Research and Development in Machine Design, Volume:03,Issue:03,DOI:http://doi.org/10.5281/zenodo.4043052, Oct 2020.

[20] R Praveen, L Prabhu, P Premjith, Adarsh. K. Mohan, Ajayraj, "Design experimental of RF controlled beach cleaner robotic vehicle", IOP Conf. Series: Materials Science and Engineering 993 (2020) 012030 IOP Publishing doi:10.1088/1757-899X/993/1/012030, 2020.

[21] Zhai Yuyi, Zhou Yu, Luo Huanxin, Liu Yunjia, Liu Liang, "Control System Design for a Surface Cleaning Robot" International Journal of Advanced Robotic Systems, Volume:10, Feb 2013.

[22] Amit Kumar Yadav, Animesh Singh, M. A. Murtaza, Ajendra Kumar Singh, "Eco Beach Cleaner", International Journal of Engineering and Management Research, ISSN (ONLINE): 2250-0758, ISSN (PRINT): 2394-6962, Volume:08, Issue:03, June 2018.

[23] "ESP32-CAM, Camera Module Based On ESP32, OV2640 Camera and ESP32-CAM-MB adapter Included,"[Online] https://www.waveshare.com/, 2024.

[24] Michal, "Testing Resistor with Digital Multi Meter."[Online] https://electric-shocks.com/testing-resistor.

[25] "0.01uF - 10nF 50V Ceramic Disc Capacitor," Online Electronics.https://onlineelectronics.in/index.phprouteproduct/productproductid3879056024

[26] "16MHz Crystal,"[Online] researchdesignlab. https://researchdesignlab.com/16mhz-crystal.html

[27] "DC Gear motor," Probots. https://probots.co.in/200-rpm-high-torque-dc-geared-motor-side-shaft-for-robotics.html

[28] Dejan, "L298N Motor Driver – Arduino Interface, How It Works, Codes, Schematics,"[Online] How to mechatronics. https://howtomechatronics.com/tutorials/arduino/arduino-dc-motor-control-tutorial-l298n-pwm-h-bridge/textThe L298N is a dual,and explain how it works.

[29] "NEMA17 STEPPER MOTOR 48MM,"[Online] probots. https://probots.co.in/nema17-stepper-motor-48mm.html

[30] "Electrobot A4988 Compatible Stepper Motor Diver Module with Heat Sink for 3D Printer Controller,"[Online] amazon. https://www.amazon.in/ElectroBot-Compatible-Stepper-Printer-Controller/dp/B07P71S359

[31] "Arduino Uno R3 - Clone (USB Cable Included),"[Online] direnc. https://www.direnc.net/arduino-uno-r3-dip-model-usb-kablo-dahil-en

[32] "Servo motors,"[Online] kollmorgen. https://www.kollmorgen.com/en-us/products/motors/servo/servo-motorstext=A servo motor is a,rotary position of the motor.

[33] "Getting Started with Arduino IDE,"[Online] thestempedia. https://ai.c.com/docs/evive/evive-arduino-ide-tutorials/getting-started-with-arduino-ide/

[34] "Autodesk Fusion 360 Logo PNG Vector,"[Online] seeklogo. https://seeklogo.com/free-vector-logos/autodesk-fusion-360

[35] "Google Slides," [Online]wikidata. https://www.wikidata.org/wiki/Q46272775

[36] "LaTeX," [Online] github. https://github.com/topics/latex?oascsforks

[37] "EasyEDA: Getting Started with PCB Design,"[Online] sparkfun. https://www.sparkfun.com/news/6906

[38] "What is Convolutional Neural Network — CNN (Deep Learning)," linkedin. https://www.linkedin.com/pulse/what-convolutional-neural-network-cnn-deep-learning-nafiz-shahriar/

[39] Afshine Amidi and Shervine Amidi, "Convolutional Neural Networks cheatsheet," stanford. https://stanford.edu/ shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks

[40] "Convolutional Neural Networks (CNN): Step 3 - Flattening," superdatascience. https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-step-3-flattening

[41] Diego Unzueta, "Fully Connected Layer vs. Convolutional Layer: Explained," builtin. https://builtin.com/machine-learning/fully-connected-layer

Chapter 14

Achievements

1. Participated in the SmartIndia Hackathon conducted by SJCET, Palai.
2. Applied for YIP, K-DISC.
3. Paper was accepted in 1st International Conference on Security, Parallel Processing, Image Processing Networking (SPIN2K24).
4. Published a journal in International Journal of Scientific Research in Science and Technology

14.1 Proof

Figure 14.1: Smart India Hackathon

Figure 14.2: Smart India Hackathon

Figure 14.3: Smart India Hackathon

Figure 14.4: Smart India Hackathon

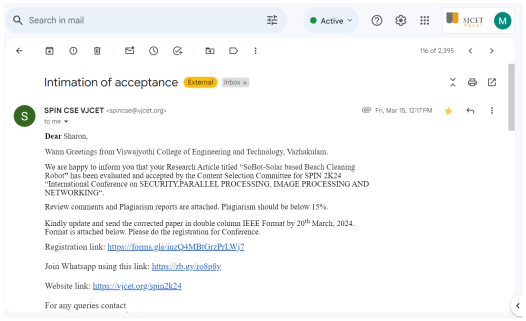


Figure 14.5: Paper acceptance(SPIN2K24)

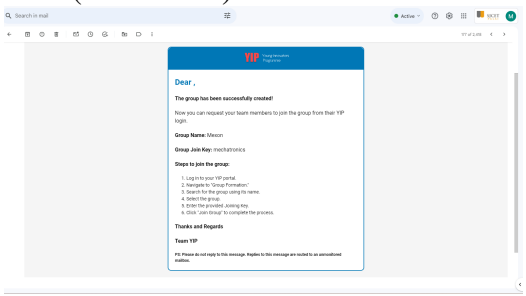
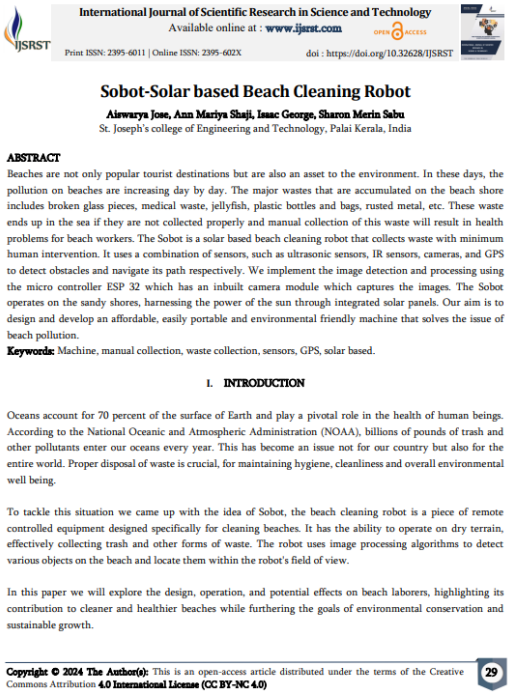


Figure 14.6: YIP



International Journal of Scientific Research in Science and Technology
Available online at : www.ijsrst.com
Print ISSN: 2395-6011 | Online ISSN: 2395-602X
doi : https://doi.org/10.32628/IJSRST

Sobot-Solar based Beach Cleaning Robot

Aiswarya Jose, Ann Mariya Shaji, Issac George, Sharon Merin Sabu
St. Joseph's college of Engineering and Technology, Palai Kerala, India

ABSTRACT
Beaches are not only popular tourist destinations but are also an asset to the environment. In these days, the pollution on beaches are increasing day by day. The major wastes that are accumulated on the beach shore includes broken glass pieces, medical waste, jellyfish, plastic bottles and bags, rusted metal, etc. These waste ends up in the sea if they are not collected properly and manual collection of this waste will result in health problems for beach workers. The Sobot is a solar based beach cleaning robot that collects waste with minimum human intervention. It uses a combination of sensors, such as ultrasonic sensors, IR sensors, cameras, and GPS to detect obstacles and navigate its path respectively. We implement the image detection and processing using the micro controller ESP 32 which has an inbuilt camera module which captures the images. The Sobot operates on the sandy shores, harnessing the power of the sun through integrated solar panels. Our aim is to design and develop an affordable, easily portable and environmental friendly machine that solves the issue of beach pollution.

Keywords: Machine, manual collection, waste collection, sensors, GPS, solar based.

I. INTRODUCTION

Oceans account for 70 percent of the surface of Earth and play a pivotal role in the health of human beings. According to the National Oceanic and Atmospheric Administration (NOAA), billions of pounds of trash and other pollutants enter our oceans every year. This has become an issue not for our country but also for the entire world. Proper disposal of waste is crucial, for maintaining hygiene, cleanliness and overall environmental well being.

To tackle this situation we came up with the idea of Sobot, the beach cleaning robot is a piece of remote controlled equipment designed specifically for cleaning beaches. It has the ability to operate on dry terrain, effectively collecting trash and other forms of waste. The robot uses image processing algorithms to detect various objects on the beach and locate them within the robot's field of view.

In this paper we will explore the design, operation, and potential effects on beach laborers, highlighting its contribution to cleaner and healthier beaches while furthering the goals of environmental conservation and sustainable growth.

Figure 14.7: Journal Publication(SPIN2K24)

Program 1. html code

```
#include <WiFi.h>
#include <WiFiClientSecure.h>
#include "esp_camera.h"
#include "soc/soc.h"
#include "soc/rtc_cntl_reg.h"
#include<WiFiManager.h>
//#include <ESP32Servo.h>
#include<AccelStepper.h>

const char* ssid = "vivo 1938";
const char* password = "12345678";
//Servo myservo;
const int DIR = 12;
const int STEP = 14;
const int  steps_per_rev = 200;

//Stepper stepper(200,14,12,13,15);


int pulse = 14;
String Feedback = "";
String Command = "", cmd = "", P1 = "", P2 = "", P3 = "", P4 = "", P5 = "", P6 = "", P7 = "", P8 = "", P9 = "";
byte ReceiveState = 0, cmdState = 1, strState = 1, questionstate = 0, equalstate = 0, semicolonstate = 0;

#define PWDN_GPIO_NUM 32
#define RESET_GPIO_NUM -1
#define XCLK_GPIO_NUM 0
#define SIOD_GPIO_NUM 26
#define SIOC_GPIO_NUM 27
#define Y9_GPIO_NUM 35
#define Y8_GPIO_NUM 34
#define Y7_GPIO_NUM 39
```

```
33   #define Y6_GPIO_NUM 36
34   #define Y5_GPIO_NUM 21
35   #define Y4_GPIO_NUM 19
36   #define Y3_GPIO_NUM 18
37   #define Y2_GPIO_NUM 5
38   #define VSYNC_GPIO_NUM 25
39   #define HREF_GPIO_NUM 23
40   #define PCLK_GPIO_NUM 22
41
42   WiFiServer server(80);
43
44   static const char PROGMEM INDEX_HTML[] = R"
        rawliteral(
45 ␣␣<!DOCTYPE␣html>
46 ␣␣<head>
47 ␣␣<meta␣charset="utf-8">
48 ␣␣<meta␣name="viewport"␣content="width=device-width,
     initial-scale=1">
49 ␣␣<script␣src="https:\/\/ajax.googleapis.com/ajax/libs
     /jquery/1.8.0/jquery.min.js"></script>
50 ␣␣<script␣src="https:\/\/cdn.jsdelivr.net/npm/
     @tensorflow/tfjs@1.3.1/dist/tf.min.js">␣</script>
51 ␣␣<script␣src="https:\/\/cdn.jsdelivr.net/npm/
     @tensorflow-models/coco-ssd@2.1.0">␣</script>
52 ␣␣</head><body>
53 ␣␣<img␣id="ShowImage"␣src=""␣style="display:none">
54 ␣␣<canvas␣id="canvas"␣width="0"␣height="0"></canvas>
55 ␣␣<table>
56 ␣␣<tr>
57 ␣␣<td><input␣type="button"␣id="restart"␣value="Restart
     "></td>
58 ␣␣<td␣colspan="2"><input␣type="button"␣id="getStill"␣
     value="Start Detect"␣style="display:none"></td>
59 ␣␣</tr>
60 ␣␣<tr␣style="color:red">
61 ␣␣<td>Object</td>
62 ␣␣<td␣colspan="2">
63 ␣␣<select␣id="object"␣onchange="count.innerHTML='';">
```

```
64 ␣␣<option␣value="bottle"␣selected="selected"␣>bottle</
     option>
65 ␣␣<option␣value="wine glass"␣selected="selected"␣>wine
     ␣glass</option>
66 ␣␣<option␣value="cup"␣selected="selected"␣>cup</option
     >
67 ␣␣</select>
68 ␣␣</td>
69 ␣␣<td><span␣id="count"␣style="color:red"><span></td>
70 ␣␣</tr>
71 ␣␣<tr␣style="display:none">
72 ␣␣<td>ScoreLimit</td>
73 ␣␣<td␣colspan="2">
74 ␣␣<select␣id="score">
75 ␣␣<option␣value="1.0">1</option>
76 ␣␣<option␣value="0.9">0.9</option>
77 ␣␣<option␣value="0.8">0.8</option>
78 ␣␣<option␣value="0.7">0.7</option>
79 ␣␣<option␣value="0.6">0.6</option>
80 ␣␣<option␣value="0.5">0.5</option>
81 ␣␣<option␣value="0.4">0.4</option>
82 ␣␣<option␣value="0.3">0.3</option>
83 ␣␣<option␣value="0.2">0.2</option>
84 ␣␣<option␣value="0.1">0.1</option>
85 ␣␣<option␣value="0"␣selected="selected">0</option>
86 ␣␣</select>
87 ␣␣</td>
88 ␣␣</tr>
89 ␣␣<tr␣style="display:none">
90 ␣␣<td>MirrorImage</td>
91 ␣␣<td␣colspan="2">
92 ␣␣<select␣id="mirrorimage">
93 ␣␣<option␣value="1">yes</option>
94 ␣␣<option␣value="0">no</option>
95 ␣␣</select>
96 ␣␣</td>
97 ␣␣</tr>
98 ␣␣<tr␣style="display:none">
```

```
99  <td>Resolution</td>
100 <td colspan="2">
101 <select id="framesize">
102 <option value="UXGA">UXGA(1600x1200)</option>
103 <option value="SXGA">SXGA(1280x1024)</option>
104 <option value="XGA">XGA(1024x768)</option>
105 <option value="SVGA">SVGA(800x600)</option>
106 <option value="VGA">VGA(640x480)</option>
107 <option value="CIF">CIF(400x296)</option>
108 <option value="QVGA" selected="selected">QVGA(320
    x240)</option>
109 <option value="HQVGA">HQVGA(240x176)</option>
110 <option value="QQVGA">QQVGA(160x120)</option>
111 </select>
112 </td>
113 </tr>
114 <tr style="display:none">
115 <td>Flash</td>
116 <td colspan="2"><input type="range" id="flash" min="
    0" max="255" value="0"></td>
117 </tr>
118 <tr style="display:none">
119 <td>Quality</td>
120 <td colspan="2"><input type="range" id="quality" min
    ="10" max="63" value="10"></td>
121 </tr>
122 <tr style="display:none">
123 <td>Brightness</td>
124 <td colspan="2"><input type="range" id="brightness"
    min="-2" max="2" value="0"></td>
125 </tr>
126 <tr style="display:none">
127 <td>Contrast</td>
128 <td colspan="2"><input type="range" id="contrast"
    min="-2" max="2" value="0"></td>
129 </tr>
130
131 </table>
```

```
  <div id="result" style="color:red" ><div>
  </body>
  </html>

  <script>
  var getStill = document.getElementById('getStill');
  var ShowImage = document.getElementById('ShowImage')
    ;
  var canvas = document.getElementById("canvas");
  var context = canvas.getContext("2d");
  var object = document.getElementById('object');
  var score = document.getElementById("score");
  var mirrorimage = document.getElementById("
    mirrorimage");
  var count = document.getElementById('count');
  var result = document.getElementById('result');
  var flash = document.getElementById('flash');
  var lastValue="";
  var myTimer;
  var restartCount=0;
  var Model;
  getStill.onclick = function (event) {
    clearInterval(myTimer);
    myTimer = setInterval(function(){error_handle()
    ;},5000);
    ShowImage.src=location.origin+'/?getstill='+Math.
    random();
  }
  function error_handle() {
    restartCount++;
    clearInterval(myTimer);
    if (restartCount<=2) {
      result.innerHTML = "Get still error. <br>Restart
     ESP32-CAM "+restartCount+" times.";
      myTimer = setInterval(function(){getStill.click
    ();},10000);
    }
    else
```

```
164 ␣␣␣␣result.innerHTML␣=␣"Get still error. <br>Please
      close the page and check ESP32-CAM.";
165 ␣␣}
166 ␣␣ShowImage.onload␣=␣function␣(event)␣{
167 ␣␣␣␣clearInterval(myTimer);
168 ␣␣␣␣restartCount=0;
169 ␣␣␣␣canvas.setAttribute("width",␣ShowImage.width);
170 ␣␣␣␣canvas.setAttribute("height",␣ShowImage.height);
171 ␣␣␣␣if␣(mirrorimage.value==1)␣{
172 ␣␣␣␣␣␣context.translate((canvas.width␣+␣ShowImage.
      width)␣/␣2,␣0);
173 ␣␣␣␣␣␣context.scale(-1,␣1);
174 ␣␣␣␣␣␣context.drawImage(ShowImage,␣0,␣0,␣ShowImage.
      width,␣ShowImage.height);
175 ␣␣␣␣␣␣context.setTransform(1,␣0,␣0,␣1,␣0,␣0);
176 ␣␣␣␣}
177 ␣␣␣␣else
178 ␣␣␣␣context.drawImage(ShowImage,0,0,ShowImage.width,
      ShowImage.height);
179 ␣␣␣␣if␣(Model)␣{
180 ␣␣␣␣␣␣DetectImage();
181 ␣␣␣␣}
182 ␣␣}
183 ␣␣
184 ␣␣restart.onclick␣=␣function␣(event)␣{
185 ␣␣␣␣fetch(location.origin+'/?restart=stop');
186 ␣␣}
187 ␣␣framesize.onclick␣=␣function␣(event)␣{
188 ␣␣␣␣fetch(document.location.origin+'/?framesize='+this
      .value+';stop');
189 ␣␣}
190 ␣␣flash.onchange␣=␣function␣(event)␣{
191 ␣␣␣␣fetch(location.origin+'/?flash='+this.value+';stop
      ');
192 ␣␣}
193 ␣␣quality.onclick␣=␣function␣(event)␣{
194 ␣␣␣␣fetch(document.location.origin+'/?quality='+this.
      value+';stop');
```

```
195 ␣␣}
196 ␣␣brightness.onclick␣=␣function␣(event)␣{
197 ␣␣␣␣fetch(document.location.origin+'/?brightness='+
      this.value+';stop');
198 ␣␣}
199 ␣␣contrast.onclick␣=␣function␣(event)␣{
200 ␣␣␣␣fetch(document.location.origin+'/?contrast='+this.
      value+';stop');
201 ␣␣}
202 ␣␣
203 ␣␣function␣ObjectDetect()␣{
204 ␣␣␣␣result.innerHTML␣=␣"Please wait for loading model.
      ";
205 ␣␣␣␣cocoSsd.load().then(cocoSsd_Model␣=>␣{
206 ␣␣␣␣␣␣Model␣=␣cocoSsd_Model;
207 ␣␣␣␣␣␣result.innerHTML␣=␣"";
208 ␣␣␣␣␣␣getStill.style.display␣=␣"block";
209 ␣␣␣␣});
210 ␣␣}
211 ␣␣
212 ␣␣function␣DetectImage()␣{
213 ␣␣␣␣Model.detect(canvas).then(Predictions␣=>␣{
214 ␣␣␣␣␣␣var␣s␣=␣(canvas.width>canvas.height)?canvas.
      width:canvas.height;
215 ␣␣␣␣␣␣var␣objectCount=0;
216 ␣␣␣␣␣␣
217 ␣␣␣␣␣␣//console.log('Predictions:␣',␣Predictions);
218 ␣␣␣␣␣␣if␣(Predictions.length>0)␣{
219 ␣␣␣␣␣␣␣␣result.innerHTML␣=␣"";
220 ␣␣␣␣␣␣␣␣for␣(var␣i=0;i<Predictions.length;i++)␣{
221 ␣␣␣␣␣␣␣␣␣␣const␣x␣=␣Predictions[i].bbox[0];
222 ␣␣␣␣␣␣␣␣␣␣const␣y␣=␣Predictions[i].bbox[1];
223 ␣␣␣␣␣␣␣␣␣␣const␣width␣=␣Predictions[i].bbox[2];
224 ␣␣␣␣␣␣␣␣␣␣const␣height␣=␣Predictions[i].bbox[3];
225 ␣␣␣␣␣␣␣␣␣␣context.lineWidth␣=␣Math.round(s/200);
226 ␣␣␣␣␣␣␣␣␣␣context.strokeStyle␣=␣"#00FFFF";
227 ␣␣␣␣␣␣␣␣␣␣context.beginPath();
228 ␣␣␣␣␣␣␣␣␣␣context.rect(x,␣y,␣width,␣height);
```

```
229 ␣␣␣␣␣␣␣␣␣␣context.stroke();
230 ␣␣␣␣␣␣␣␣␣␣context.lineWidth␣=␣"2";
231 ␣␣␣␣␣␣␣␣␣␣context.fillStyle␣=␣"red";
232 ␣␣␣␣␣␣␣␣␣␣context.font␣=␣Math.round(s/30)␣+␣"px Arial"
    ;
233 ␣␣␣␣␣␣␣␣␣␣context.fillText(Predictions[i].class,␣x,␣y)
    ;
234 ␣␣␣␣␣␣␣␣␣␣//context.fillText(i,␣x,␣y);
235 ␣␣␣␣␣␣␣␣␣␣result.innerHTML+=␣"[ "+i+" ] "+Predictions[
    i].class+", "+Math.round(Predictions[i].score*100)+
    "%, "+Math.round(x)+", "+Math.round(y)+", "+Math.
    round(width)+", "+Math.round(height)+"<br>";
236 ␣␣␣␣␣␣␣␣␣␣
237 ␣␣␣␣␣␣␣␣␣␣if␣(Predictions[i].class==object.value&&
    Predictions[i].score>=score.value)␣{
238 ␣␣␣␣␣␣␣␣␣␣␣␣objectCount++;
239 ␣␣␣␣␣␣␣␣␣␣}
240 ␣␣␣␣␣␣␣␣}
241 ␣␣␣␣␣␣␣␣count.innerHTML␣=␣objectCount;
242 ␣␣␣␣␣␣}
243 ␣␣␣␣␣␣else␣{
244 ␣␣␣␣␣␣␣␣result.innerHTML␣=␣"Unrecognizable";
245 ␣␣␣␣␣␣␣␣count.innerHTML␣=␣"0";
246 ␣␣␣␣␣␣}
247 ␣␣␣␣␣␣
248 ␣␣␣␣␣␣//if␣(count.innerHTML␣!=␣lastValue)␣{
249 ␣␣␣␣␣␣␣␣lastValue␣=␣count.innerHTML;
250 ␣␣␣␣␣␣␣␣if␣(objectCount>0)␣{
251 ␣␣␣␣␣␣␣␣␣␣//␣␣␣␣␣␣$.ajax({url:␣document.location.
    origin+'/?serial='+object.value+';stop',␣async:␣
    false});
252 ␣␣␣␣␣␣␣␣␣␣$.ajax({url:␣document.location.origin+'/?
    detectCount='+object.value+';'+String(objectCount)
    +';stop',␣async:␣false});
253 ␣␣␣␣␣␣␣␣}
254 ␣␣␣␣␣␣␣␣//}
255 ␣␣␣␣␣␣try␣{
256 ␣␣␣␣␣␣␣␣document.createEvent("TouchEvent");
```

```
␣␣␣␣␣␣␣␣setTimeout(function(){getStill.click();},250);
␣␣␣␣␣␣}
␣␣␣␣␣␣catch(e)␣{
␣␣␣␣␣␣␣␣setTimeout(function(){getStill.click();},150);
␣␣␣␣␣␣}
␣␣␣␣});
␣␣}
␣␣function␣getFeedback(target)␣{
␣␣␣␣var␣data␣=␣$.ajax({
␣␣␣␣␣␣type:␣"get",
␣␣␣␣␣␣dataType:␣"text",
␣␣␣␣␣␣url:␣target,
␣␣␣␣␣␣success:␣function(response)␣␣␣␣␣␣␣␣{
␣␣␣␣␣␣␣␣result.innerHTML␣=␣response;
␣␣␣␣␣␣},
␣␣␣␣␣␣error:␣function(exception)␣␣␣␣␣␣␣␣{
␣␣␣␣␣␣␣␣result.innerHTML␣=␣'fail';
␣␣␣␣␣␣}
␣␣␣␣});
␣␣}
␣␣window.onload␣=␣function␣()␣{
␣␣␣␣ObjectDetect();
␣␣␣␣
␣␣␣␣
␣␣}
␣␣</script>
␣␣)rawliteral";

  void ExecuteCommand() {
    if (cmd != "getstill") {
      Serial.println("cmd=␣" + cmd + "␣,P1=␣" + P1 + "
         ␣,P2=␣" + P2 + "␣,P3=␣" + P3 + "␣,P4=␣" + P4
         + "␣,P5=␣" + P5 + "␣,P6=␣" + P6 + "␣,P7=␣" +
         P7 + "␣,P8=␣" + P8 + "␣,P9=␣" + P9);
      Serial.println("Object:␣" + P1);
      Serial.println("Count:␣" + P2);
      if (P2.toInt() > 0) digitalWrite(pulse, 1);
      P1.trim();
```

```
292       if (P1 == "bottle" || P1 == "wine glass" || P1
             == "cup") {
293         digitalWrite(14, 1);
294         delay(1000);
295
296         steppermotor();
297
298         // myservo.attach(13); // Attach the servo to
               pin 13
299         //myservo.write(180); // Rotate the servo to
               180 degrees
300         // stepper.setSpeed(100); // Set the speed of
               the stepper motor
301         //stepper.step(100); // Rotate the stepper
               motor 100 steps
302         //delay(10000); // Wait for 10 seconds
303         // myservo.detach();
304       } else {
305         digitalWrite(14, 0);
306       }
307     }
308
309     if (cmd == "resetwifi") {
310       WiFi.begin(P1.c_str(), P2.c_str());
311       Serial.print("Connecting to ");
312       Serial.println(P1);
313
314       long int StartTime = millis();
315       while (WiFi.status() != WL_CONNECTED) {
316         delay(500);
317         if ((StartTime + 5000) < millis()) break;
318       }
319       Serial.println("");
320       Serial.println("STAIP: " + WiFi.localIP().
             toString());
321       Feedback = "STAIP: " + WiFi.localIP().toString()
             ;
322     } else if (cmd == "restart") {
```

```
323       ESP.restart();
324     } else if (cmd == "digitalwrite") {
325       ledcDetachPin(P1.toInt());
326       pinMode(P1.toInt(), OUTPUT);
327       digitalWrite(P1.toInt(), P2.toInt());
328     } else if (cmd == "analogwrite") {
329       if (P1 = "4") {
330         ledcAttachPin(4, 4);
331         ledcSetup(4, 5000, 8);
332         ledcWrite(4, P2.toInt());
333       } else {
334         ledcAttachPin(P1.toInt(), 5);
335         ledcSetup(5, 5000, 8);
336         ledcWrite(5, P2.toInt());
337       }
338     } else if (cmd == "flash") {
339       ledcAttachPin(4, 4);
340       ledcSetup(4, 5000, 8);
341
342       int val = P1.toInt();
343       ledcWrite(4, val);
344     } else if (cmd == "framesize") {
345       sensor_t* s = esp_camera_sensor_get();
346       if (P1 == "QQVGA")
347       s->set_framesize(s, FRAMESIZE_QQVGA);
348       else if (P1 == "HQVGA")
349       s->set_framesize(s, FRAMESIZE_HQVGA);
350       else if (P1 == "QVGA")
351       s->set_framesize(s, FRAMESIZE_QVGA);
352       else if (P1 == "CIF")
353       s->set_framesize(s, FRAMESIZE_CIF);
354       else if (P1 == "VGA")
355       s->set_framesize(s, FRAMESIZE_VGA);
356       else if (P1 == "SVGA")
357       s->set_framesize(s, FRAMESIZE_SVGA);
358       else if (P1 == "XGA")
359       s->set_framesize(s, FRAMESIZE_XGA);
360       else if (P1 == "SXGA")
```

```
361       s->set_framesize(s, FRAMESIZE_SXGA);
362       else if (P1 == "UXGA")
363       s->set_framesize(s, FRAMESIZE_UXGA);
364       else
365       s->set_framesize(s, FRAMESIZE_QVGA);
366     } else if (cmd == "quality") {
367       sensor_t* s = esp_camera_sensor_get();
368       int val = P1.toInt();
369       s->set_quality(s, val);
370     } else if (cmd == "contrast") {
371       sensor_t* s = esp_camera_sensor_get();
372       int val = P1.toInt();
373       s->set_contrast(s, val);
374     } else if (cmd == "brightness") {
375       sensor_t* s = esp_camera_sensor_get();
376       int val = P1.toInt();
377       s->set_brightness(s, val);
378     } else if (cmd == "serial") {
379       Serial.println(P1);
380     } else if (cmd == "detectCount") {
381       Serial.println(P1 + "␣=␣" + P2);
382
383     } else if (cmd == "tcp") {
384       String domain = P1;
385       int port = P2.toInt();
386       String request = P3;
387       int wait = P4.toInt();
388
389       Feedback = tcp_http(domain, request, port, wait)
             ;
390     } else if (cmd == "linenotify") {
391       String token = P1;
392       String request = P2;
393       Feedback = LineNotify(token, request, 1);
394       if (Feedback.indexOf("status") != -1) {
395         int s = Feedback.indexOf("{");
396           Feedback = Feedback.substring(s);
397           int e = Feedback.indexOf("}");
```

```
398         Feedback = Feedback.substring(0, e);
399         Feedback.replace("\"", "");
400         Feedback.replace("{", "");
401           Feedback.replace("}", "");
402       }
403     } else if (cmd == "sendCapturedImageToLineNotify")
          {
404       Feedback = sendCapturedImageToLineNotify(P1);
405       if (Feedback == "") Feedback = "The␣image␣failed
            ␣to␣send.␣<br>The␣framesize␣may␣be␣too␣large.
            ";
406     } else {
407       Feedback = "Command␣is␣not␣defined.";
408     }
409     if (Feedback == "") Feedback = Command;
410   }
411
412   void setup() {
413     WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0);
414
415     pinMode(pulse, OUTPUT);
416     digitalWrite(pulse, 0);
417
418     Serial.begin(115200);
419     Serial.setDebugOutput(true);
420     Serial.println();
421
422     Serial.begin(115200);
423     pinMode(STEP, OUTPUT);
424     pinMode(DIR, OUTPUT);
425
426     camera_config_t config;
427     config.ledc_channel = LEDC_CHANNEL_0;
428     config.ledc_timer = LEDC_TIMER_0;
429     config.pin_d0 = Y2_GPIO_NUM;
430     config.pin_d1 = Y3_GPIO_NUM;
431     config.pin_d2 = Y4_GPIO_NUM;
432     config.pin_d3 = Y5_GPIO_NUM;
```

```
433     config.pin_d4 = Y6_GPIO_NUM;
434     config.pin_d5 = Y7_GPIO_NUM;
435     config.pin_d6 = Y8_GPIO_NUM;
436     config.pin_d7 = Y9_GPIO_NUM;
437     config.pin_xclk = XCLK_GPIO_NUM;
438     config.pin_pclk = PCLK_GPIO_NUM;
439     config.pin_vsync = VSYNC_GPIO_NUM;
440     config.pin_href = HREF_GPIO_NUM;
441     config.pin_sscb_sda = SIOD_GPIO_NUM;
442     config.pin_sscb_scl = SIOC_GPIO_NUM;
443     config.pin_pwdn = PWDN_GPIO_NUM;
444     config.pin_reset = RESET_GPIO_NUM;
445     config.xclk_freq_hz = 20000000;
446     config.pixel_format = PIXFORMAT_JPEG;
447     //init with high specs to pre-allocate larger
          buffers
448     if (psramFound()) {
449       config.frame_size = FRAMESIZE_UXGA;
450       config.jpeg_quality = 10;  //0-63 lower number
            means higher quality
451       config.fb_count = 2;
452     } else {
453       config.frame_size = FRAMESIZE_SVGA;
454       config.jpeg_quality = 12;  //0-63 lower number
            means higher quality
455       config.fb_count = 1;
456     }
457
458     // camera init
459     esp_err_t err = esp_camera_init(&config);
460     if (err != ESP_OK) {
461       Serial.printf("Camera␣init␣failed␣with␣error␣0x%
            x", err);
462       delay(1000);
463       ESP.restart();
464     }
465
466     //drop down frame size for higher initial frame
```

```
      rate
467   sensor_t* s = esp_camera_sensor_get();
468   s->set_framesize(s, FRAMESIZE_QVGA);  //UXGA|SXGA|
      XGA|SVGA|VGA|CIF|QVGA|HQVGA|QQVGA
469   //added extra controls
470   s->set_vflip(s, 1);                   // flip it
      back
471   //  s->set_brightness(s, 1); // up the brightness
      just a bit
472   //  s->set_saturation(s, -2); // lower the
      saturation
473   // control ends
474
475   ledcAttachPin(4, 4);
476   ledcSetup(4, 5000, 8);
477
478   WiFi.mode(WIFI_AP_STA);
479   //  WiFi.config(IPAddress(192, 168, 43, 11),
      IPAddress(192, 168, 43, 2), IPAddress(255, 255,
       255, 0));
480   //WiFi.begin(ssid, password);
481   WiFiManager wm;
482   bool res;
483   res=wm.autoConnect("My Camera","password");
484   if(!res){
485     Serial.println("Failed to connect");
486   }
487   delay(1000);
488   Serial.println("");
489   Serial.print("Connecting to ");
490   Serial.println(ssid);
491
492   long int StartTime = millis();
493   while (WiFi.status() != WL_CONNECTED) {
494     delay(500);
495     if ((StartTime + 10000) < millis()) break;
496   }
497
```

```
498     if (WiFi.status() == WL_CONNECTED) {
499       Serial.println("");
500       Serial.println("STAIP address: ");
501       Serial.println(WiFi.localIP());
502
503       for (int i = 0; i < 5; i++) {
504         ledcWrite(4, 10);
505         delay(200);
506         ledcWrite(4, 0);
507         delay(200);
508       }
509     }
510
511     pinMode(4, OUTPUT);
512     digitalWrite(4, LOW);
513     server.begin();
514   }
515
516   void loop() {
517     Feedback = "";
518     Command = "";
519     cmd = "";
520     P1 = "";
521     P2 = "";
522     P3 = "";
523     P4 = "";
524     P5 = "";
525     P6 = "";
526     P7 = "";
527     P8 = "";
528     P9 = "";
529     ReceiveState = 0, cmdState = 1, strState = 1,
          questionstate = 0, equalstate = 0,
          semicolonstate = 0;
530
531     WiFiClient client = server.available();
532
533     if (client) {
```

```
534       String currentLine = "";
535       while (client.connected()) {
536         if (client.available()) {
537           char c = client.read();
538           getCommand(c);
539           if (c == '\n') {
540             if (currentLine.length() == 0) {
541
542               if (cmd == "getstill") {
543
544                 camera_fb_t* fb = NULL;
545                 fb = esp_camera_fb_get();
546                 if (!fb) {
547                   Serial.println("Camera capture
                        failed");
548                   delay(1000);
549                   ESP.restart();
550                 }
551
552                 client.println("HTTP/1.1 200 OK");
553                 client.println("Access-Control-Allow-
                      Origin: *");
554                 client.println("Access-Control-Allow-
                      Headers: Origin, X-Requested-With,
                      Content-Type, Accept");
555                 client.println("Access-Control-Allow-
                      Methods: GET,POST,PUT,DELETE,
                      OPTIONS");
556                 client.println("Content-Type: image/
                      jpeg");
557                 client.println("Content-Disposition:
                      form-data; name=\"imageFile\";
                      filename=\"picture.jpg\"");
558                 client.println("Content-Length: " +
                      String(fb->len));
559                 client.println("Connection: close");
560                 client.println();
561
```

```
562                 uint8_t* fbBuf = fb->buf;
563                 size_t fbLen = fb->len;
564                 for (size_t n = 0; n < fbLen; n = n +
                        1024) {
565                   if (n + 1024 < fbLen) {
566                     client.write(fbBuf, 1024);
567                     fbBuf += 1024;
568                   } else if (fbLen % 1024 > 0) {
569                     size_t remainder = fbLen % 1024;
570                     client.write(fbBuf, remainder);
571                   }
572                 }
573                 esp_camera_fb_return(fb);
574                 pinMode(4, OUTPUT);
575                 digitalWrite(4, LOW);
576               } else {
577
578                 client.println("HTTP/1.1 200 OK");
579                 client.println("Access-Control-Allow-
                        Headers: Origin, X-Requested-With,
                        Content-Type, Accept");
580                 client.println("Access-Control-Allow-
                        Methods: GET,POST,PUT,DELETE,
                        OPTIONS");
581                 client.println("Content-Type: text/
                        html; charset=utf-8");
582                 client.println("Access-Control-Allow-
                        Origin: *");
583                 client.println("Connection: close");
584                 client.println();
585
586                 String Data = "";
587                 if (cmd != "")
588                 Data = Feedback;
589                 else {
590                   Data = String((const char*)
                          INDEX_HTML);
591                 }
```

```
592                 int Index;
593                 for (Index = 0; Index < Data.length();
                        Index = Index + 1000) {
594                   client.print(Data.substring(Index,
                          Index + 1000));
595                 }
596                 client.println();
597               }
598               Feedback = "";
599               break;
600             } else {
601               currentLine = "";
602             }
603           } else if (c != '\r') {
604             currentLine += c;
605           }
606
607           if ((currentLine.indexOf("/?") != -1) && (
                  currentLine.indexOf("␣HTTP") != -1)) {
608             if (Command.indexOf("stop") != -1) {
609               http:  //192.168.xxx.xxx/?cmd=aaa;bbb;
                     ccc;stop
610               client.println();
611               client.println();
612               client.stop();
613             }
614             currentLine = "";
615             Feedback = "";
616             ExecuteCommand();
617           }
618         }
619       }
620       delay(1);
621       client.stop();
622     }
623   }
624
625   void getCommand(char c) {
```

```
626     if (c == '?') ReceiveState = 1;
627     if ((c == '␣') || (c == '\r') || (c == '\n'))
           ReceiveState = 0;
628
629     if (ReceiveState == 1) {
630       Command = Command + String(c);
631
632       if (c == '=') cmdState = 0;
633       if (c == ';') strState++;
634
635       if ((cmdState == 1) && ((c != '?') || (
             questionstate == 1))) cmd = cmd + String(c);
636       if ((cmdState == 0) && (strState == 1) && ((c !=
              '=') || (equalstate == 1))) P1 = P1 + String
             (c);
637       if ((cmdState == 0) && (strState == 2) && (c !=
             ';')) P2 = P2 + String(c);
638       if ((cmdState == 0) && (strState == 3) && (c !=
             ';')) P3 = P3 + String(c);
639       if ((cmdState == 0) && (strState == 4) && (c !=
             ';')) P4 = P4 + String(c);
640       if ((cmdState == 0) && (strState == 5) && (c !=
             ';')) P5 = P5 + String(c);
641       if ((cmdState == 0) && (strState == 6) && (c !=
             ';')) P6 = P6 + String(c);
642       if ((cmdState == 0) && (strState == 7) && (c !=
             ';')) P7 = P7 + String(c);
643       if ((cmdState == 0) && (strState == 8) && (c !=
             ';')) P8 = P8 + String(c);
644       if ((cmdState == 0) && (strState >= 9) && ((c !=
              ';') || (semicolonstate == 1))) P9 = P9 +
             String(c);
645
646       if (P2.toInt() < 1 or P2.toInt() == 0)
             digitalWrite(pulse, 0);
647
648       if (c == '?') questionstate = 1;
649       if (c == '=') equalstate = 1;
```

```
650       if ((strState >= 9) && (c == ';'))
             semicolonstate = 1;
651     }
652   }
653
654   String tcp_http(String domain, String request, int
         port, byte wait) {
655     WiFiClient client_tcp;
656
657     if (client_tcp.connect(domain.c_str(), port)) {
658       Serial.println("GET " + request);
659       client_tcp.println("GET " + request + " HTTP/1.1
             ");
660       client_tcp.println("Host: " + domain);
661       client_tcp.println("Connection: close");
662       client_tcp.println();
663
664       String getResponse = "", Feedback = "";
665       boolean state = false;
666       int waitTime = 3000;  // timeout 3 seconds
667       long startTime = millis();
668       while ((startTime + waitTime) > millis()) {
669         while (client_tcp.available()) {
670           char c = client_tcp.read();
671           if (state == true) Feedback += String(c);
672           if (c == '\n') {
673             if (getResponse.length() == 0) state =
                  true;
674             getResponse = "";
675           } else if (c != '\r')
676           getResponse += String(c);
677           if (wait == 1)
678           startTime = millis();
679         }
680         if (wait == 0)
681         if ((state == true) && (Feedback.length() !=
              0)) break;
682       }
```

```
683       client_tcp.stop();
684       return Feedback;
685     } else
686     return "Connection␣failed";
687   }
688
689   String LineNotify(String token, String request, byte
        wait) {
690     request.replace("%", "%25");
691     request.replace("␣", "%20");
692     request.replace("&", "%20");
693     request.replace("#", "%20");
694     //request.replace("\'","%27");
695     request.replace("\"", "%22");
696     request.replace("\n", "%0D%0A");
697     request.replace("%3Cbr%3E", "%0D%0A");
698     request.replace("%3Cbr/%3E", "%0D%0A");
699     request.replace("%3Cbr%20/%3E", "%0D%0A");
700     request.replace("%3CBR%3E", "%0D%0A");
701     request.replace("%3CBR/%3E", "%0D%0A");
702     request.replace("%3CBR%20/%3E", "%0D%0A");
703     request.replace("%20stickerPackageId", "&
          stickerPackageId");
704     request.replace("%20stickerId", "&stickerId");
705
706     WiFiClientSecure client tcp;
707     client_tcp.setInsecure();  //run version 1.0.5 or
          above
708   }
709
710   String sendCapturedImageToLineNotify(String token) {
711     String getAll = "", getBody = "";
712
713     camera fb t* fb = NULL;
714     fb = esp_camera_fb_get();
715     if (!fb) {
716       Serial.println("Camera␣capture␣failed");
717       delay(1000);
```

```
718       ESP.restart();
719       return "";
720     }
721
722     WiFiClientSecure client_tcp;
723     client_tcp.setInsecure();  //run version 1.0.5 or
          above
724     return getBody;
725   }
```

2. C programming code

```
1  void steppermotor()
2  {
3    digitalWrite(DIR, HIGH);
4    Serial.println("Spinning Clockwise...");
5
6    for(int i = 0; i<steps_per_rev; i++)
7    {
8      digitalWrite(STEP, HIGH);
9      delayMicroseconds(1000);
10     digitalWrite(STEP, LOW);
11     delayMicroseconds(1000);
12   }
13   delay(1000);
14
15   digitalWrite(DIR, LOW);
16   Serial.println("Spinning Anti-Clockwise...");
17
18   for(int i = 0; i<steps_per_rev; i++)
19   {
20     digitalWrite(STEP, HIGH);
21     delayMicroseconds(1000);
22     digitalWrite(STEP, LOW);
23     delayMicroseconds(1000);
24   }
25   delay(1000);
26 }
```

3. C programming code

```
#include <Servo.h>     //Servo motor library. This is standard library
#include <NewPing.h>    //Ultrasonic sensor function library. You must install this library
//our L298N control pins
const int LeftMotorForward = 7;
const int LeftMotorBackward = 6;
const int RightMotorForward = 4;
const int RightMotorBackward = 5;
int pos=0;
//sensor pins
#define trig_pin A1 //analog input 1
#define echo_pin A2 //analog input 2
#define maximum_distance 200
boolean goesForward = false;
int distance = 100;
NewPing sonar(trig_pin, echo_pin, maximum_distance); //sensor function
Servo servo_motor; //our servo name
Servo myservo;
void setup(){
  pinMode(RightMotorForward, OUTPUT);
  pinMode(LeftMotorForward, OUTPUT);
  pinMode(LeftMotorBackward, OUTPUT);
  pinMode(RightMotorBackward, OUTPUT);
  servo_motor.attach(10); //our servo pin
  myservo.attach(13);
  servo_motor.write(115);
  delay(2000);
  distance = readPing();
  delay(100);
  distance = readPing();
  delay(100);
  distance = readPing();
  delay(100);
  distance = readPing();
  delay(100);
}
```

```
36 void loop(){
37   int distanceRight = 0;
38   int distanceLeft = 0;
39   delay(50);
40   if (distance <= 20){
41     moveStop();
42     delay(300);
43     myservo.write(170);
44     delay(300);
45     myservo.write(50);
46     delay(300);
47     myservo.write(170);
48     delay(300);
49     myservo.write(50);
50     delay(300);
51     moveBackward();
52     delay(400);
53     moveStop();
54     delay(300);
55     distanceRight = lookRight();
56     delay(300);
57     distanceLeft = lookLeft();
58     delay(300);
59     if (distance >= distanceLeft){
60       turnRight();
61       moveStop();
62     }
63     else{
64       turnLeft();
65       moveStop();
66     }
67   }
68   else{
69     moveForward();
70   }
71   distance = readPing();
72 }
73 int lookRight(){
```

```
74    servo_motor.write(50);
75    delay(500);
76    int distance = readPing();
77    delay(100);
78    servo_motor.write(115);
79    return distance;
80  }
81  int lookLeft(){
82    servo_motor.write(170);
83    delay(500);
84    int distance = readPing();
85    delay(100);
86    servo_motor.write(115);
87    return distance;
88    delay(100);
89  }
90  int readPing(){
91    delay(70);
92    int cm = sonar.ping_cm();
93    if (cm==0){
94      cm=250;
95    }
96    return cm;
97  }
98  void moveStop(){
99    digitalWrite(RightMotorForward, LOW);
100   digitalWrite(LeftMotorForward, LOW);
101   digitalWrite(RightMotorBackward, LOW);
102   digitalWrite(LeftMotorBackward, LOW);
103 }
104 void moveForward(){
105   if(!goesForward){
106     goesForward=true;
107     digitalWrite(LeftMotorForward, HIGH);
108     digitalWrite(RightMotorForward, HIGH);
109     digitalWrite(LeftMotorBackward, LOW);
110     digitalWrite(RightMotorBackward, LOW);
111   }
```

```
112 }
113 void moveBackward(){
114   goesForward=false;
115   digitalWrite(LeftMotorBackward, HIGH);
116   digitalWrite(RightMotorBackward, HIGH);
117   digitalWrite(LeftMotorForward, LOW);
118   digitalWrite(RightMotorForward, LOW);
119 }
120 void turnRight(){
121   digitalWrite(LeftMotorForward, HIGH);
122   digitalWrite(RightMotorBackward, HIGH);
123   digitalWrite(LeftMotorBackward, LOW);
124   digitalWrite(RightMotorForward, LOW);
125   delay(500);
126   digitalWrite(LeftMotorForward, HIGH);
127   digitalWrite(RightMotorForward, HIGH);
128   digitalWrite(LeftMotorBackward, LOW);
129   digitalWrite(RightMotorBackward, LOW);
130 }
131 void turnLeft(){
132   digitalWrite(LeftMotorBackward, HIGH);
133   digitalWrite(RightMotorForward, HIGH);
134   digitalWrite(LeftMotorForward, LOW);
135   digitalWrite(RightMotorBackward, LOW);
136   delay(500);
137   digitalWrite(LeftMotorForward, HIGH);
138   digitalWrite(RightMotorForward, HIGH);
139   digitalWrite(LeftMotorBackward, LOW);
140   digitalWrite(RightMotorBackward, LOW);
141 }
```

Department of Electronics & Communication Engineering
St.Joseph's College of Engineering & Technology,Palai
Palai - 686 579