BSEE AUTONOMOUS LAWN MOWER Spring 2023

AUTONOMOUS LAWN MOWER

By

Ahmed Daniyal Ejaz	BEE193037
Ahmad Shahzaib	BEE193035
Mansoor Azhar	BEE193031

Faculty of Engineering
Capital University of Science & Technology
Islamabad

July, 2023

AUTONOMOUS LAWN MOWER

by

Ahmed Daniyal Ejaz
BEE193037

Ahmad Shahzaib
BEE193035

Mansoor Azhar
BEE193031

A Project Report submitted to the
DEPARTMENT OF ELECTRICAL ENGINEERING
in partial fulfillment of the requirements for the degree of
BACHELORS OF SCIENCE IN ELECTRICAL ENGINEERING

Faculty of Engineering
Capital University of Science & Technology,
Islamabad
July, 2023

From Daniyal; This endeavor is dedicated to my late father, Ahmed Ejaz Nadeem, who could not live long enough to see me achieve this milestone in life. He will always be my inspiration by the example of integrity, dedication and kind-heartedness that he set.

This project and its success are dedicated to our parents and families. The seemingly relentless work, meant that we could not participate in several important occasions and could not spend more time with them. We pray for their well-being and sincerely hope that this mountain of a task, finally put to words, makes up for the precious time lost and priceless moments missed. May ALLAH SWT keep us captains of our own ships, and soften the tides of time.

DECLARATION

It is declared that this is an original piece of our own work, except where otherwise acknowledged in text and references. This work has not been submitted in any form for another degree or diploma at any university or other institution for tertiary education and shall not be submitted by us in future for obtaining any degree from this or any other University or Institution.

Ahmed Daniyal Ejaz
BEE193037

Ahmad Shahzaib
BEE193035

Mansoor Azhar
BEE193031

July, 2023

CERTIFICATE OF APPROVAL

It is certified that the project titled “Autonomous Lawn Mower” carried out by Ahmed Daniyal Ejaz, Reg. No. BEE193037, Ahmad Shahzaib, Reg No. BEE193035, Mansoor Azhar, Reg No. BEE193031, under the supervision of Engr. Muhammad Moin Qasim, Capital University of Science & Technology, Islamabad, is fully adequate, in scope and in quality, as a final year project for the degree of BS Electrical Engineering.

Supervisor: -------------------------------------

Engr. Muhammad Moin Qasim
Senior Lecturer
Department of Electrical Engineering
Faculty of Engineering
Capital University of Science & Technology, Islamabad

HoD: -------------------------------------

Dr. Noor Mohammad Khan
Professor
Department of Electrical Engineering
Faculty of Engineering
Capital University of Science & Technology, Islamabad

ACKNOWLEDGMENT

First and foremost, all our thanks and praise is to ALLAH SWT who is the most merciful. Secondly, we would like to thank our supervisor “Engr. Muhammad Moin Qasim” whose guidance and support enlightened our vision and gave us the necessary thrust to complete this mammoth of an endeavor. We feel honored to have worked under his kind supervision. The room for comfort and improvement provided to us by him played a major part in the success of this project. We would also like to thank our instructors and faculty members who showed us support and also gave us constructive criticism which made it possible for us to see the realities of the task and to be ready to counter any setbacks that we may face along the way. We are thankful to our lab engineers for their sincere input throughout the project which helped us to mold the project according to the requirements of our degree. We would especially like to thank Mr. Ameer Hamza and Mr. Syed Abu Turab Naqvi for their crucial help, who were always available and willing to guide us in our queries and helped us to overcome the difficulties that we faced along the way. Dr. Aamir Iqbal Bhatti is worthy of mention for the interest he showed in our project early on, that gave us the hope that this project would be worthwhile. Special credit must be given to Dr. Umer Amir Khan whose support, guidance and wisdom became our compass whenever we ventured into the unknown and his encouragement our driving force. It was his never-ending faith in us throughout the duration of not only this project but also our degree that imbued in us the drive to devote and commit ourselves to our goals and to never give up. The fond memories of the time spent under his tutelage in his office and in the classroom would stay with the three of us for times to come. Finally, we would like to thank our batchmates and friends who supported us and gave us hope when there was little to come by. Especially Mr. Ahmad Zaman, who was a friend and a brother to us in our happiness and distress. He went out of his way to help us in any way he could. His efforts to keep us focused in the project will always be remembered and cherished. No one is more worthy of mention than Mr. Ahmed Jahanzeb Ejaz, had it not been for his selfless devotion to our cause and profound support given to us in our project, this project would never have seen the pleasant conclusion it sees today.

ABSTRACT

The project deals with the designing and implementation of a lawn mowing system that is completely autonomous and can map out its surroundings to a great degree of accuracy, plans its path based on the created proximity map, detects and avoids obstacles in its path as it moves along, by using fusion of proximity sensors and color sensors. The project was at first planned to develop an artificial intelligence-based design to control the movement and operation of lawn mower with images of lawn provided as input and perform movement and avoidance of obstacles accordingly. The aim of the project is to achieve autonomous lawn mowing operation. The idea was to gain and implement the knowledge of artificial intelligence using its different techniques to design a lawn mower prototype that would govern its own path within the area it detects and avoids obstacles in its path. For object detection, a CNN model has been implemented on software of Anaconda Navigator with multiple iterations to achieve and refine accuracy of model. Lawn boundary detection has been implemented using image processing techniques through OpenCV on python to achieve lawn area from fed images. Due to limitations of hardware procurements for the support of AI-based design, the overall design has been revised to achieve the end objective of prototype autonomy through the use of fusion of sensors which include ultrasonic and color sensors. An effective and efficient software algorithm has also been developed for the smooth processing of the sensor data and active working of the lawn mower prototype. The algorithm has been developed as a state machine. The state machine has 7 states in total. It has been used to determine the sequence of events, the relationship between the various states of sequential control, and the design flow of the lawn traversal based on the current state and the inputs received continuously. This allows for comprehensive understanding and implementation of the model in a step-by-step process. In order to implement and realize the movements of motors that are a part of the mower system, they are interfaced with the controller with ATmega2560 and speed is to be controlled by changing duty cycle of PWM as per the requirements. The prototype effectively detects and traverses along the boundary keeping check of mowing space while also monitoring incidence of obstacles where upon it is to correct its course. In order to avoid cutting in non-grass space, the color sensor detects grass within given range of color.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS/ABBREVIATIONS

AI	Artificial Intelligence
GB	Gigabyte
Hz	Hertz
CNN	Convolutional Neural Network
PWM	Pulse Width Modulation
I2C	Inter Integrated Circuit
MATLAB	Matrix Laboratory
SDG	Sustainable Development Goals
LiDAR	Light Detection And Ranging
IoT	Internet of Things
SNN	Shallow Neural Network
RF	Random Forest
GPU	Graphics Processing Unit
ResNet50	Residual Network 50
vSLAM	Visual Simultaneous Localization And Mapping

Chapter 1

INTRODUCTION

This project report is about the autonomous lawn mower. Using a lawn mower robot to achieve the purpose of cutting grass for lawns and playgrounds on its own. These mowers operate independently and navigate the lawn using cutting-edge technology, sensors, and navigation systems. Robotic lawn mowers come with a variety of sensors to help them avoid collisions with things like rocks, trees, and animals. These sensors, which aid in the mower's course adjustment, may be ultrasonic sensors, infrared sensors, or bump sensors, laser sensors.

Autonomous lawn mowers are designed to perform similar functions to humans but more quickly and accurately. Lawn mower utilizes state-of-the-art technology to maintain a properly mowed lawn with absolutely no work or next to nothing on your side. This thought has led to the idea of designing an autonomous lawn mower that can follow the defined area and avoid obstacles through real time image processing and sensors to assist in operation. Image processing adapts well to change and manipulates millions of photos in a short time frame, extracting valuable information.

Figure 1.1 Husqvarna Autonomous Lawn Mower [1]

1.1 Overview

The end objective of the project is to use the autonomous lawn mower in lawn space with path mapping, obstacle detection and avoidance. This is achieved using image processing techniques under the umbrella of artificial intelligence. Data acquisition using image acquisition tools, pre-processing the image, accurately extracting the region of interest, feature extraction and classification using classifier are the steps involved in image processing. A data set can be defined containing a vast distribution of relevant images labelled with different classes added for lawn mowing. Anaconda is the software that can be used for implementing these algorithms. A model is then trained and tweaked for optimum results pertaining to identification of area of operation and obstacles. The input lawn images are acquired using camera module placed above the lawn mower. The autonomous lawn mower depends upon two categories of motors, one for the mowing of grass and other for its motion and movement in lawn. The sensors used are calibrated according to meaningful threshold to achieve operational optimization for the achievement of autonomy.

1.2 Project Idea

The plan is to build a lawnmower that is completely autonomous and can map out its surroundings to a great degree of accuracy, plan its path based on the created map, detect and avoid obstacles in its path as it moves along, by using the image processing and artificial intelligence. The manually operated lawn mowers emit CO_2 and carbon monoxide, and some of them also produce noise pollution, making them neither user-friendly nor environmentally beneficial. AI has empowered numerous advancements and driven the multiplication of innovations like IoT, mechanical technology, investigation, and voice assistants. So, it is achievable and desirable to control the robotic lawn mower for lawns and play grounds through the use of available AI models trained to achieve such an objective and also employ related sensors to assist and achieve the same.

1.3 Purpose of the Project

The purpose of the project is to design a methodology that helps to minimize human efforts and maximize productivity. Artificial intelligence models can make wise and learned judgements since they are trained using enormous amounts of data. AI has become the newest advancement in technology and is influencing practically every industry's future. AI has widespread presence in daily routine of present-day life without realizing it, such as different applications based on AI. It is desired to utilize the concept of AI and implement it on a lawn mower prototype to achieve autonomy through self judgement of lawn space and existing obstacles encountered and avoided all on its own. In short, such an assembly of a lawn mower would reduce human efforts by using adequately trained models and effective sensors in the domain. Compared to conventional endothermic engine mowers, autonomous mowers are made to require less work to mow the lawn while also reducing local pollution and noise.

1.4 Project Specifications

This section provides the details and specifications of the project building blocks (components) which have been used for project completion. The bare minimum requirements for this project to succeed and operate as intended have been characterized and listed in Table 1.1 based on the purpose they are to serve and the performance they are to provide. These include the main controller, mowing and driving motors, proximity sensors and controller for image processing.

Table 1.1 Project Specifications

Building Blocks	**Specifications**
Controller	Clock speed 16MHz, Operating voltage of 5-12v Digital I/O Pins 14, PWM digital I/0 pins: 6 DC current per I/O pin: 20-30mA
Mowing/Driving Motor	Voltage: 12v, Current: 40mA Speed: 400-600 RPM
Proximity sensor	Input Voltage 5v Ranging distance 2cm to 400cm Trigger input pulse width 10us Working current 15mA
Controller for Image Processing	Input Voltage 5-12v Supports Linux distributions, Ubuntu, Raspbian Output HDMI, Composite Video, Frame rate 40-60 fps 40 GPIO PINS RAM 1-4 GBs, OS 32-64 Bits

1.5 Applications of the Project

For cutting, trimming, or mowing grass patches, fields, lawns, and gardens, lawn mowers are widely used around the world to keep the grass well-groomed and growing at an even, acceptable height. In short, the project design is applicable for all the operations that are repetitive and can be done by a human hand. Autonomous lawn mower will do the work more accurately without human assistance. Major applications of project are discussed in this section.

1.5.1 Sports Grounds

Sports fields, like football fields, are frequently utilized. The fertilizing, watering, and other lawn care requirements make it time-consuming and expensive to maintain the grass. The lawnmower can effectively identify area of operation and any obstructions it may encounter along the way. The blades automatically halt if the robot is lifted off the ground. as comparison to standard mowing. Robotic mowers offer significant operating cost savings for athletic clubs, fields, and pitches.

1.5.2 Lawns

Similar to how a robot vacuum picks up dust and crumbs inside your home, a robot lawn mower trims the grass in your yard. Robotic lawnmower operates by quietly moving about the wired perimeter on its own awkwardly pumping into objects occasionally, and providing you with grass cuttings along the way. Lawn mower in home lawns or in different field will autonomously map path and will cut the grass of your home lawn and detect the obstacle of some kind of pet, or stones etc.

1.5.3 Commercial Complexes

Maintaining the lawns in commercial buildings can benefit greatly from the use of autonomous lawn mowers. Large lawns found in commercial complexes frequently need to be maintained on a regular basis. Without the requirement for manual labor, autonomous lawn mowers are capable of handling the job well. They can be set up to work outside of regular business hours, minimizing disturbance of normal operations. Autonomous mowers' reliable and even mowing ensures that business complexes' lawns remain aesthetically pleasing. Commercial complexes can greatly cut the time and labor necessary for grass upkeep by adopting autonomous lawn mowers. Sensors in autonomous lawn mowers enable them to recognize impediments and change their course as necessary.

1.6 Project Plan

This section describes how work is distributed for both part I and II. The activities of the project include both software and hardware implementation tasks. The distribution of tasks among the project members has also been discussed.

1.6.1 Project Milestones

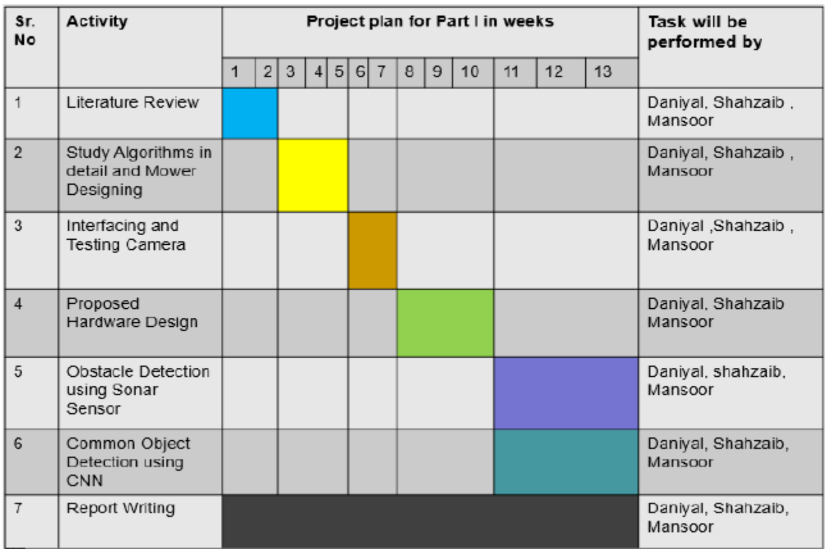
Given below are Tables regarding proposed distribution of tasks, task duration and resource person details for part I and II separately. The actually undertaken timelines throughout the period of the project in part I and part II are also listed separately.

Table 1.2 Proposed Project Plan

Tasks	Duration	Resource Persons
Literature Review	02 Weeks	Daniyal, Shahzaib, Mansoor
Study Algorithms in detail and Mower Designing	03 Weeks	Daniyal
Interfacing Motors with Microcontroller	02 Weeks	Shahzaib, Mansoor
Interfacing and Testing camera	02 Weeks	Daniyal
Interfacing Controller with Lawn Mower Mechanism	04 Weeks	Daniyal, Shahzaib, Mansoor
Testing and Improvement of Hardware-Software Interface and Design	06 Weeks	Daniyal, Shahzaib, Mansoor
Design Documentation	03 Weeks	Daniyal, Shahzaib, Mansoor
Obstacle Detection using Sonar sensor	02 Weeks	Shahzaib, Mansoor
Area Mapping, Path Mapping	04 Weeks	Daniyal

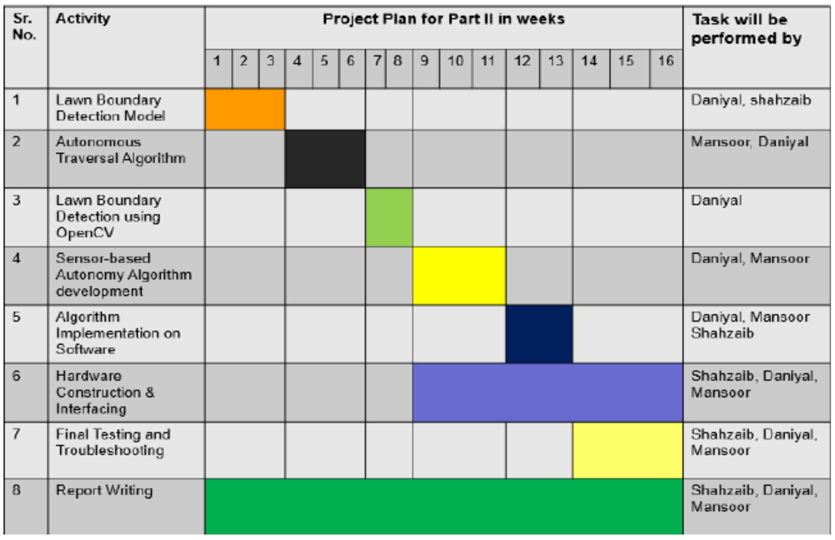
1.6.2 Project Timeline

The project plan followed for final year project Part-I and Part-II is listed in Figure 1.2 and Figure 1.3. The total project duration spanned across 29 weeks which included 13 weeks of Part-I and 16 weeks of Part-II. Activities specific to Part-I included literature review, camera interfacing, proposed hardware design, obstacle detection through sonar, object detection using CNN, each with their own duration. Activities specific to Part-II included lawn boundary detection model, autonomous traversal algorithm, sensor-based autonomy, hardware construction, final testing and troubleshooting. The activity of report writing was performed throughout the duration of the project.



Sr. No	Activity	Project plan for Part I in weeks													Task will be performed by
		1	2	3	4	5	6	7	8	9	10	11	12	13	
1	Literature Review	🟦	🟦												Daniyal, Shahzaib , Mansoor
2	Study Algorithms in detail and Mower Designing			🟨	🟨	🟨									Daniyal, Shahzaib , Mansoor
3	Interfacing and Testing Camera						🟧	🟧							Daniyal ,Shahzaib , Mansoor
4	Proposed Hardware Design								🟩	🟩	🟩				Daniyal, Shahzaib Mansoor
5	Obstacle Detection using Sonar Sensor											🟪	🟪	🟪	Daniyal, shahzaib, Mansoor
6	Common Object Detection using CNN											🟦	🟦	🟦	Daniyal, Shahzaib, Mansoor
7	Report Writing	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	Daniyal, Shahzaib, Mansoor

Figure 1.2 Project Timeline (PART-I)



Sr. No.	Activity	Project Plan for Part II in weeks																Task will be performed by
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
1	Lawn Boundary Detection Model	🟧	🟧	🟧														Daniyal, shahzaib
2	Autonomous Traversal Algorithm				⬛	⬛	⬛											Mansoor, Daniyal
3	Lawn Boundary Detection using OpenCV							🟩	🟩									Daniyal
4	Sensor-based Autonomy Algorithm development									🟨	🟨	🟨						Daniyal, Mansoor
5	Algorithm Implementation on Software												🟦	🟦				Daniyal, Mansoor Shahzaib
6	Hardware Construction & Interfacing									🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	Shahzaib, Daniyal, Mansoor
7	Final Testing and Troubleshooting														🟨	🟨	🟨	Shahzaib, Daniyal, Mansoor
8	Report Writing	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	Shahzaib, Daniyal, Mansoor

Figure 1.3 Project Timeline (PART-II)

1.6.3 Targeted Sustainable Development Goals

The Sustainable Development Goals (SDGs), sometimes referred to as the global goals, were enacted by the United Nations in 2015 as a global call to action to eradicate poverty, safeguard the environment, and guarantee that by the year 2030, peace and prosperity would be experienced by everyone [2]. As the world advances, it is imperative that all forms of development and research is conducted with the concept of sustainability in mind.

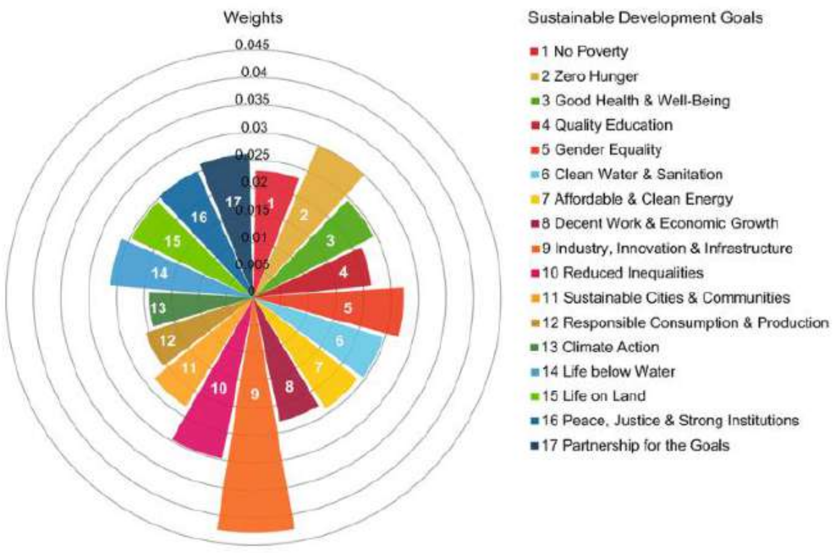


Figure 1.4 SDG Impact Chart [3]

Therefore, the final year project was directed accordingly to achieve and streamline the approaches along the SGDs. The SDGs most closely related to this project have been listed in Table 1.3.

Table 1.3 Targeted Sustainable Development Goals

S#	Targeted SDGs	SDG No.	Implementation Detail
1.	Industry, Innovation and Infrastructure	SDG #09	Reliable Lawn Upkeep through Innovation
2.	Responsible Consumption and Production	SDG #12	Reduction in Fuel & Waste
3.	Climate Action	SDG #13	Low Carbon Imprint

1.7 Report Organization

The report consists of six chapters. Chapter 1 gives an overview of the project which is about an AI based and interfaced sensors design for autonomous lawn mowing. The project plan, objectives and specifications are discussed. Chapter 2 discusses information about research and background study related to the project. Projects and technologies related to the project are also mentioned in the second chapter. Chapter 3 describes the project design and methodology including details of software and hardware designs of the proposed and revised implemented design. Chapter 4 gives detailed insight about software as well as hardware tools used for project work. Software's such as AutoCAD, SketchUp, Anaconda, its working and features are described along with hardware tools such as camera, sensors and controller. Chapter 5 discusses thoroughly the obtained results of the project software and hardware. The shortcomings and evaluations of those results and recommendations based on those evaluations. The tracking of attainment of SDG goals relevant to the project are also discussed in the fifth chapter. Chapter 6 concludes the project work and lays forth the recommendations and possible future work after the completion of the project and the achievement of closure on the project.

Chapter 2

LITERATURE REVIEW

This chapter includes the detailed overview of the study and the research done regarding the project work. The various concepts and algorithms related to the project scope are studied during the research work. The projects and technologies corresponding to the project work are also discussed in this chapter. At the end of this chapter, the limitations of the current and existing work are also listed and discussed.

2.1 Background Theory

Most establishments, including hotels, hospitals, schools, etc., rely on human labor to cut and trim the grass in their premises now as well as in the past. A worker may use a powered lawn mower, but he or she would still have to operate it manually. This is primarily because, in the past, most automatic lawn mowers were expensive and unreliable; however, recent technological advancements have changed this perception, and today's market offers a wide variety of self-governing lawn mower options, each with a different set of features and a wide range of prices.

Many lawn mowers on the market use cylinder mowing blades, primarily due to the cheaper costs to make these blades and ease of use. In its place, it has been decided that a rotating disc mower blade should be used. Firstly, it is able to withstand challenging circumstances while keeping clean cutting performance, maneuverability, and durability. Additionally, there are additional possibilities for trimming width. Thirdly, it may trim or cut lawns more quickly than other options because of its excellent cutting efficiency, which decreases overall cutting time.

Its objective was to assess advances in technology made to produce efficient and affordable grass cutting mechanisms. Modern technology is frequently employed in manually operated equipment to cut grass. It was discovered that there are many different kinds of grass cutters on the market, and they all use internal combustion engines, solar power, or electricity. It has been determined that new lawn cutting mechanisms must be developed for the achievement of efficiency and reducing human involvement while also maintaining than older ones utilizing superior engines and blade materials while utilizing less labor as well [4].

In computer vision, image processing encompasses a series of steps. It begins with acquiring digital images from cameras or other sources, followed by preprocessing to enhance quality and reduce noise. Next, relevant features are extracted to represent objects or patterns in the image. Image segmentation divides the image into meaningful regions for analysis. Object detection and recognition identify objects within the image, while object tracking monitors their movements over time. Machine learning techniques aid in image classification tasks. Post-processing refines results and may include filtering false positives or creating visualizations. Ultimately, computer vision systems make decisions or take actions based on the processed information. This dynamic process plays a crucial role in various applications, such as autonomous vehicles, surveillance, medical imaging, and facial recognition systems.

Path planning is a fundamental concept in robotics and artificial intelligence that involves finding an optimal path for a robot or agent to navigate from a starting point to a goal while avoiding obstacles and adhering to certain constraints. The field emerged from the need to efficiently plan routes for robots, autonomous vehicles, and unmanned aerial vehicles (UAVs). It draws upon various algorithms and techniques, such as graph search, potential fields, A* search, RRT (Rapidly-exploring Random Trees), and Dijkstra's algorithm. Path planning plays a critical role in enabling safe, efficient, and reliable autonomous systems to navigate complex environments and has applications in industries like logistics, exploration, and surveillance. The way path planning techniques work are through various steps which include firstly to represent the environment using grids, graphs, or potential fields to navigate. Then the initial position and the desired destination for the robot or agent is defined. After which strategies are employed to detect and avoid obstacles, ensuring safe navigation. In order to sketch out a path, algorithms like Dijkstra's, A*, or RRT are used to find the optimal path between the start and goal while also considering constraints. Further techniques are used to refine and simplify the generated path for efficiency. Handling dynamic obstacles and adapting paths in real-time for continuous navigation is also important for localization of the agent. In the end, the desirability of different paths based on factors like distance, time, or energy is evaluated.

2.2 Related Technologies

Technologies related to the project are listed below which include various techniques and employ different algorithms for the achievement of autonomy in lawn mowing.

2.2.1 IoT-Based Solar-Powered Smart Lawn Mower

High-end application and tool design and development opportunities have been made possible by the rapid advancement of technology. In reality, conventional mowers are often fuel-powered and need a person to operate them. In this work, a smart lawn mower is created that is driven by a solar photovoltaic (PV) panel and managed via an Internet of Things (IoT)-based method. The engineered lawn mower has a brushless DC (BLDC) motor, four gear motors, sensors, an Arduino-based charge controller, and a Raspberry Pi-powered renewable energy source, making it an environmentally friendly machine. Through an Android application, a lawn mower is operated and managed. When delivering data over the Internet and interacting with Android apps, Raspberry Pi is employed as an edge computing device as shown in Figure 2.1. The primary innovation in this research is an IOT-based motion control function that allows the user to control the lawnmower from a distance. Results of the developed model in Figure 2.2 show that, under a variety of weather circumstances, the system's average electrical efficiency is 89.5%. The developed concept is used in golf courses, playgrounds, and lawns to save operating costs, conserve energy, lower noise pollution, and achieve environmental sustainability goals [5].

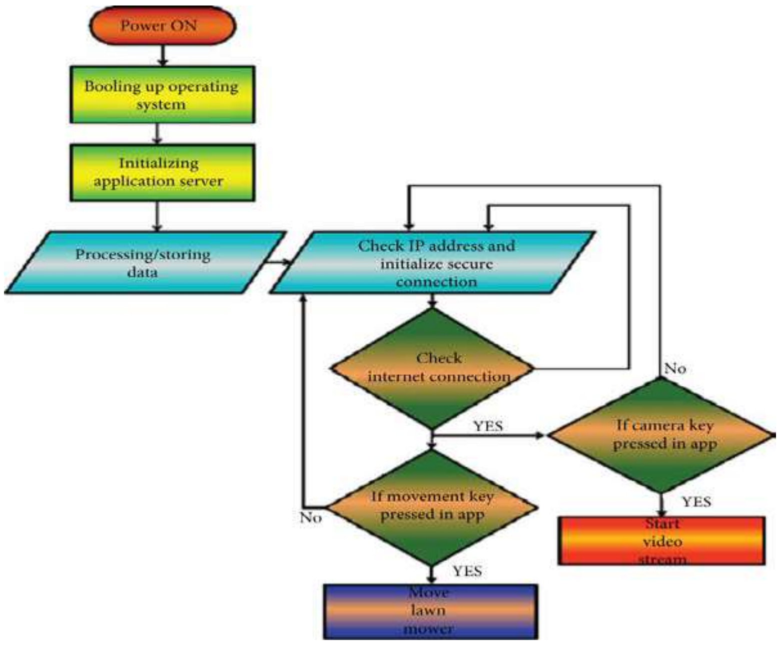

Power ON
Booting up operating system
Initializing application server
Processing/storing data
Check IP address and initialize secure connection
Check internet connection
No
YES
If camera key pressed in app
YES
Start video stream
If movement key pressed in app
No
YES
Move lawn mower

Figure 2.1 Smart Lawn Mower Flowchart [6]

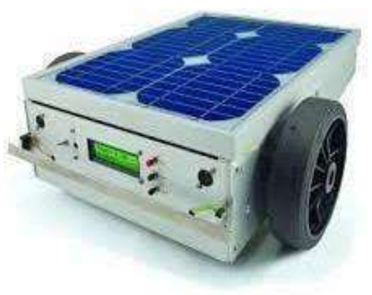


Figure 2.2 Smart Lawn Mower Design [5]

2.2.2 AI-Based Approach for Lawn Length Estimation

An aspect of autonomous driving is work sensing and mobility control. This technology focuses on the autonomous work sensing and mobility control of a commercial electric robotic lawn mower and proposes an AI-based solution for work vehicles like a robotic lawn mower. Two functions that are closely related to one another are work sensing and mobility control. In order to operate as efficiently as possible, a lawnmower, for example, should move at a slower speed when the workload is heavy and vice versa. At the same time, the battery must be preserved because it is necessary for both work performance and mobility. Focus is on developing an estimating system for a robotic lawn mower based on these criteria to assess ground conditions or grass lengths. To do this, observation data created by combining 10 distinct types of sensor data were used to build and test two AI algorithms, random forest (RF) and shallow neural network (SNN) as shown in Figure 2.3. The SNN obtained 95.0% in several tests on actual lawn grass areas, whereas the RF technique, which was evaluated using information from the fusion of sensors as shown in Figure 2.4, attained 92.3% correct estimation ratio. The SNN's accuracy is moreover 94.0% in experiments when sensor data are continually collected while the robotic lawn mower is operating.

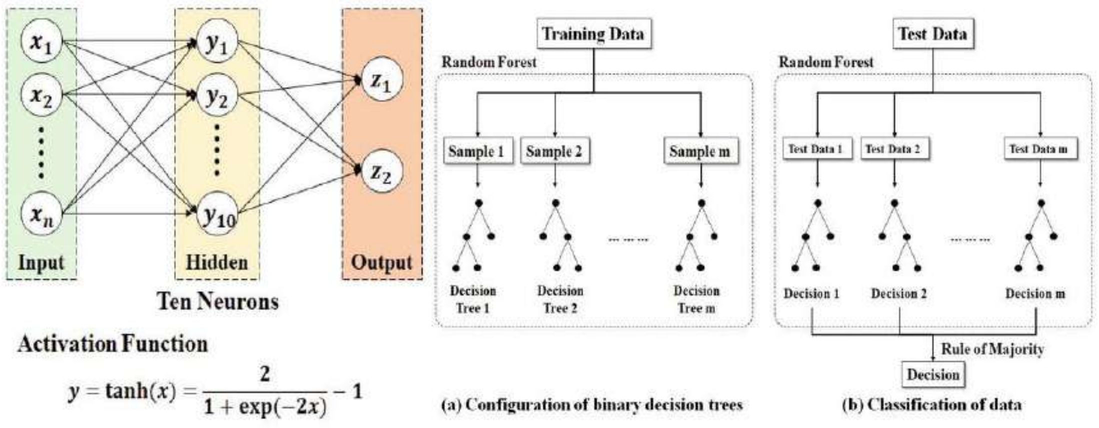


Figure 2.3 SNN and RF Architecture [7]

Currently, a robotic lawn mower that combines two motor control systems one for the robot's movement and another for cutting the grass is being created in order to implement the suggested estimation method [7].

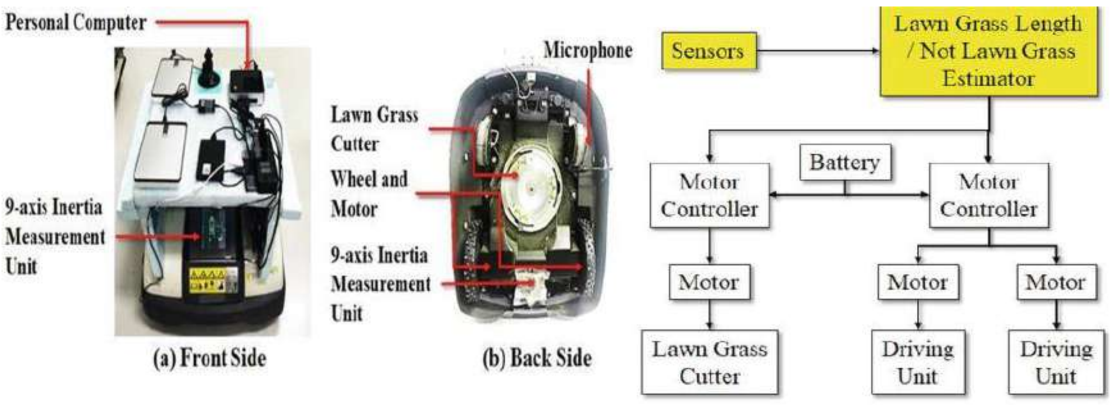


Figure 2.4 AI-Based Lawn Mower [7]

2.3 Related Projects

Different projects have been done involving robotic lawn mower concept using different concepts and algorithms. Few of them have been listed below.

2.3.1 Design and Manufacture of Automatic Robotic Lawn Mower

In this project, the creation of an image-recognition equipped autonomous robotic lawn mower is the subject of this study. The lawnmower, steering motor, slide rail, and camera are all combined on the platform above the crawler tracks in the system structure to achieve the personification goal. As a moving vehicle to represent human feet, crawler tracks with outstanding traction and terrain adaptability are chosen. To encourage innovation and efficiency, a lawn mower mechanism is also created that simulates the left and right swing of human mowing. Then, Webcams are used in lieu of human eyes to detect impediments. A human-machine interface that can be controlled remotely with a mobile phone has been added as shown in Figure 2.5, allowing users to choose among slow, inching, or obstacle avoidance modes. The autonomous robotic lawn mower will carry out the instruction in accordance with the predetermined path after the lengths of both sides of the rectangular area are entered into the program. This study topic may be used to the automatic mowing of farms, as well as to the upkeep of golf courses and playgrounds [8].

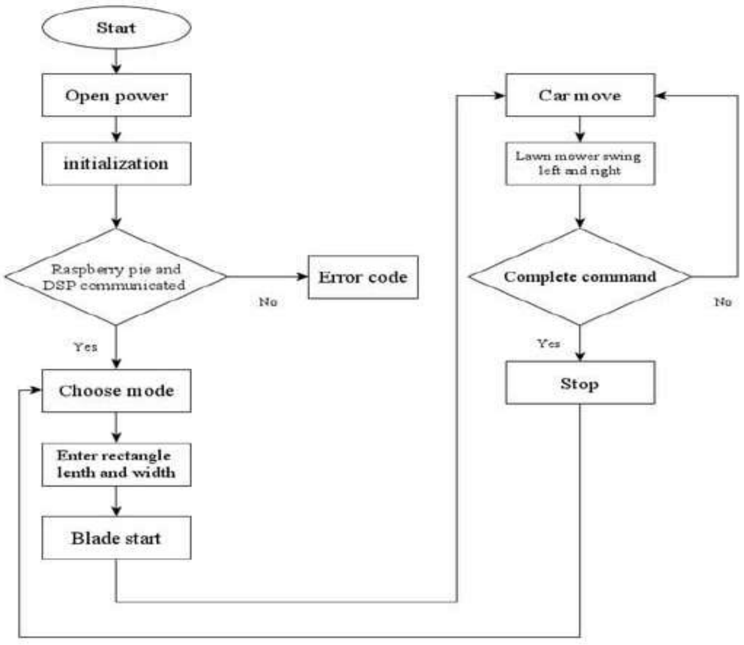


Figure 2.5 Automatic Lawn Mower Flowchart [8]

2.3.2 Embedded Robust Visual Obstacle Detection for Mowers

In this project, floor cleaners and lawn mowers are the only mass-market service robots available now. Despite being on the market for more than 20 years, they primarily lack intelligent features from current robot research. In instance, the obstacle detection and avoidance usually involve just detecting physical collisions. In this paper is discuss a camera-based non-contact obstacle avoidance system for a prototype autonomous lawn mower [9].There has been created a simple, low-cost module as shown in Figure 2.6, with color cameras and an ARM-based processor that can be easily fitted to an autonomous lawn mower as shown in Figure 2.7. 20 prototype devices that were dispersed throughout eight different European nations were used in a field test that lasted 3,494 hours to test the system.

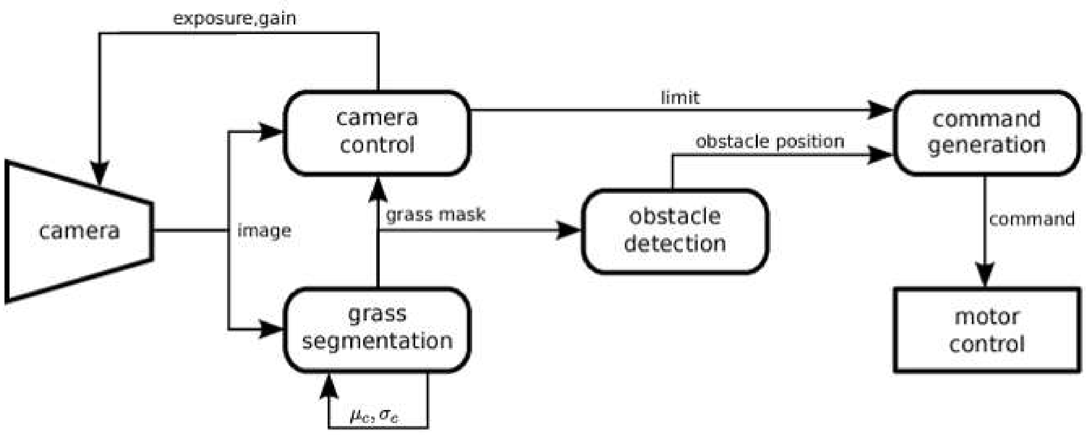


Figure 2.6 Robust Visual Obstacle Detection Scheme [9]

The findings demonstrate that the suggested approach is capable of operating for a whole season without the assistance of an expert and significantly lowers collision occurrences while maintaining high mowing performance.

Figure 2.7 Obstacle Detection on Autonomous Lawn Mowers [9]

2.3.3 Autonomous Boundary Detection Using Image Recognition

Most robotic lawnmowers use buried metal wires to designate their working area, which is time-consuming and labor-intensive. To autonomously recognise the border of the lawn and generate the navigation map, this study uses the integration of numerous sensors, including an inertial measurement unit (IMU), wheel encoders, light detection and ranging (LiDAR), and an RGB-D camera. Based on an RGB-D camera, visual simultaneous localization and mapping (vSLAM) is used. A support vector machine (SVM) is used to recognise boundaries and obstacles and create the augmented Octomap with virtual walls for the border pavement or barriers that are lower or comparable to grass height as shown in Figure 2.8. To reduce the computing burden of the embedded system and transform picture data into point cloud data to generate the navigation map, a set of image processing algorithms is presented.

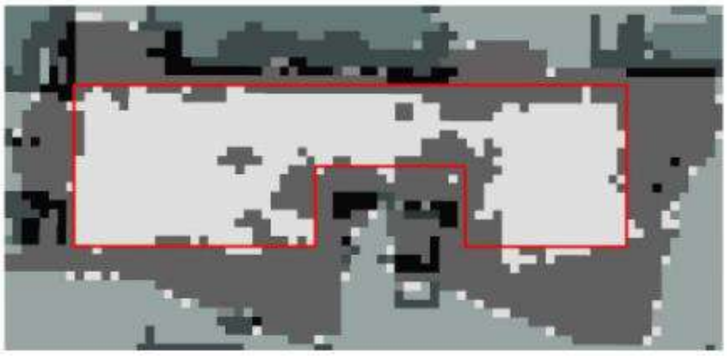


Figure 2.8 Augmented Octomap [10]

The prototype's robot operating system (ROS)-based control system was constructed with an embedded system (Jetson Nano from NVIDIA) and control boards (ARM STM32 from STMicroelectronics). The proposed technology might be applied to map development and navigation in the future [10].

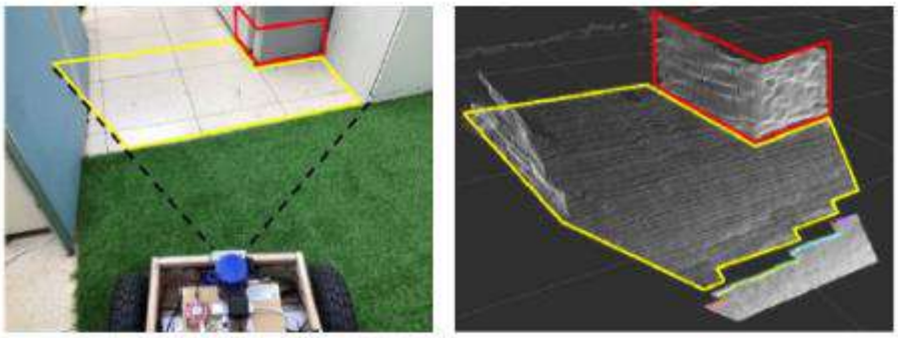


Figure 2.9 Autonomous Lawn Boundary Detection [10]

2.3.4 Fast Semantic Segmentation Model PULNet for Mowers

Data including visual information is expanding at an exponential rate due to the fast growth of artificial intelligence and big data. The goal of computer vision research is to extract targets with semantic information from enormous volumes of video and picture data so that computers may better comprehend and solve issues in the real world, providing users considerable ease. Although target detection can identify the position and category of a target in an image, it cannot detect the exact border of the object, nor can it detect large-area or irregular targets in an image, such as lawns, lakes, sky, and wall cracks. Medical treatment, intelligent robots, and drones, on the other hand, often work in a large-area particular region, needing a computer to detect the position of the target area and its border. This is a boundary detection task that combines area target identification with border location. This project creates and constructs a PULNet semantic segmentation model for deep learning to increase the speed and accuracy of semantic segmentation. It is ideal for grass segmentation. This includes Dilated_ResNet50, the Pooling pyramid (P), the Upsampling dimensionality reduction structure (U), and the image Local detail information structure (L), which can quickly analyse the range of the lawn in the image and then use the Eight-neighbor coding traversal method to record the changing trend of pixel values and locate the boundary as shown in Figure 2.10. PULNet has the merits of a few parameters and high accuracy, which is suitable for hardware environments such as portable computers, high-performance embedded, and mobile platforms [11].

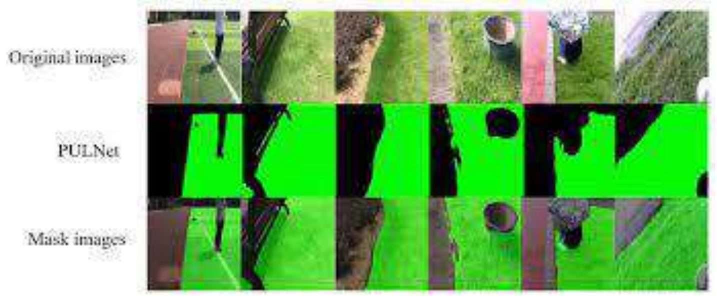


Figure 2.10 Grass Area Detection using PULNet [11]

2.4 Limitations and Bottlenecks of the Existing Work

The existing projects have a lot of advantages and applications but there are certain limitations too. Limitations of the projects that already exist in the industry include obstacle detection, their dependence on lighting conditions. In many cases the products fail to overcome the various terrains that exist in lawn conditions. Following are listed the limitations and bottlenecks in detail.

2.4.1 Obstacle Avoidance Inability

Most existing projects fall short in the area of avoiding obstacles that may arrive in their defined path. Ways to avoid these obstacles and course correction to achieve the desired path of the machine has not been achieved widely in this regard.

2.4.2 Dependence on Lighting Conditions

The projects and works that are widely available in the field are mostly dependent on the ambient lightning conditions of the working area. The accuracy of area mapping, robot positioning and object detection falls drastically in poor lightning conditions.

2.4.3 Financial Feasibility

The already existing projects and mowers have expensive initial purchase cost. For the efficient and continued working of the mowers they require high maintenance for both the outer body and the hardware components such as cameras, range sensors like LiDAR, which themselves are quite expensive.

2.4.4 Complexity of Terrain

Autonomous lawn mowers that exist in the market nowadays may struggle with regards to their movement due to difficult terrains. Slopes, uneven ground, and obstructions like as rocks or tree roots offer problems for their navigation systems.

2.4.5 Limited Battery Time

Autonomous lawn mowers are powered by batteries, and their operating time is determined by battery capacity. Large or intricate lawns may need many recharges during a single mowing session, resulting in longer mowing times.

2.5 Problem Statement

The project deals with curtailing manual labour involved in traditional lawn mowing. Manual lawn mowers involve the use of human-operated lawn mowing equipment. The key problem posed by manual mowing is the physical effort required to push or operate the mower, the time and labor involved in mowing large areas, the potential for uneven or missed spots due to human error, and the limitations on mowing during adverse weather conditions. Additionally, manual mowing may not be suitable for individuals with physical disabilities or those seeking more efficient and convenient lawn maintenance solutions. The solution to this problem is to create an autonomously controlled system for a robotic lawn mower that can assist in carrying out such a repetitive activity that is challenging for a human hand to do. There are many repetitive chores that take a long time and require a lot of labour. By using a robotic lawnmower to complete tasks with a similar effectiveness rate, this project will save time off the hands of the human. The available sensors and their operating platforms have also made it convenient to achieve concept of autonomy in robots. Therefore, the goal of this project is to achieve autonomy on lawn mowers through the learning and application of AI concepts and through the use of relevant imaging and ranging sensors while using a robotic lawn mower that can aid with time saving and perform repetitive activities such as lawn mowing in particular.

2.6 Summary

This chapter summarizes the literature study done for the project work. All the concepts and technologies related to the project work are also discussed in this chapter. The existing working projects related to current work are also described in this chapter and the limitations and bottle necks of the work are also discussed in it. At the end, problem statement of project is given which depicts the hypothesis of the project.

Chapter 3

PROJECT DESIGN AND IMPLEMENTATION

In this chapter, the project design and implementation methodology has been discussed. Different algorithms that have been implemented and tested are explained in detail, and their obtained results have been documented and evaluated in a future chapter. Firstly, CNN model has been implemented for the purpose of object detection, in which the model is trained through multiple iterations and changing values of test, train, validation to achieve the best accuracy. Secondly, OpenCV has been explored and implemented for the lawn boundary detection. The project mainly consists of two categories of design; the hardware design and the software design, which has been based on software modeling and algorithm development in detail. Both these designs are discussed in this chapter with their implementation procedures.

3.1 Proposed Design Methodology

The overall design methodology proposed is undertaken using an integrated approach for the detection of working area boundary, the path mapping and control of lawn mower using the required signals and instructions. The design is to recognize operational area using static image of the field using trained model of boundary detection. Secondly, to map the area and plan its track within the operation area, traversal and localization algorithm are to be used. To avoid obstacles encountered along the path, object detection using image processing has been used. Design proposed for project work is illustrated below in Figure 3.1.

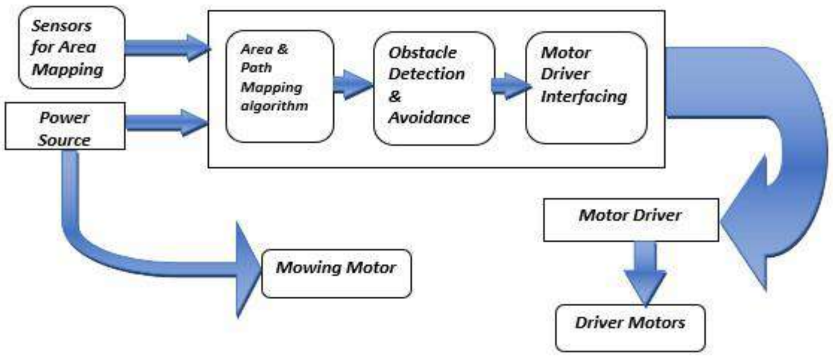


Figure 3.1 Proposed Design Methodology

The proposed design is divided into four main blocks. The first block is the acquisition and training of dataset for software modelling and testing of the object detection. Then the training of model for lawn boundary detection on relevant lawn image dataset. Afterwards the implementation of area traversal technique for the prototype to operate autonomously. Controller used for this purpose is intended to be Raspberry Pi. Required pins of Raspberry pi for hardware interfacing and providing input voltage are to be selected as per requirements. Then, the designed algorithm will be implemented which will lead to processing of input image. Afterwards, the path finding and determination will be undergone. After which control signals will be generated which will be sent by controller to the driving motors that are to be part of lawn mower prototype. The last one is the output block which will be obtained when the mower will receive signal for operation and execute movement accordingly and finally mow the grass.

3.2 Analysis Procedure

Since, the project is comprised of primarily two major parts i.e proposed design and implemented design, so they are discussed separately in the following sections.

For software as well as hardware implementation, different parameters such as algorithms and hardware components are analyzed through the pros and cons of respective models as listed in tables below.

Proposed algorithm has been researched upon for object detection which include CNN model and Lawn boundary detection method based on Canny edge detection using OpenCV.

Implemented algorithm has been developed making use of the concept of algorithmic state machine, distributing the crucial steps that must be achieved for the accurate traversal in lawn space using the inputs from integrated sensors and execution of commands to the motors.

3.2.1 Convolutional Neural Network (CNN) Model

Using its built-in functions, the Convolutional Neural Network is fully capable of processing and analyzing data. CNN's areas of expertise are image detection and classification. To identify the distinctive features of the image, each layer is trained. It takes CNN model significant time to view images and train the layers on the training data as shown in Figure 3.2.

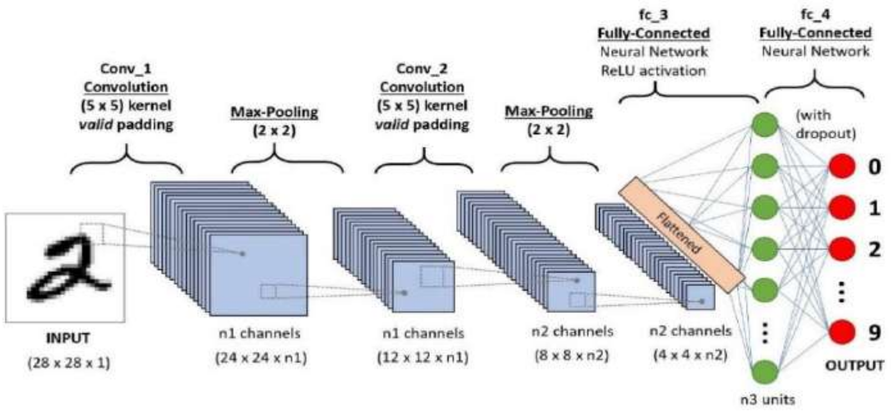


Figure 3.2 CNN Model Architecture [12]

3.2.2 Canny Edge Detector in OpenCV

The Canny edge detector is a popular image processing technique used to detect edges in digital images. It works by identifying areas of rapid intensity change, indicating the presence of an edge. Canny involves Gaussian blurring, gradient calculation, non-maximum suppression to thin edges, and hysteresis thresholding to determine strong and weak edges. It provides accurate and well-defined edge detection for various computer vision applications. In lawn boundary detection, the Canny edge detector is employed to identify the edges that outline the boundaries of a lawn area in an image. By applying the Canny algorithm, areas of significant intensity changes, which correspond to edges between the lawn and surrounding objects or backgrounds, are highlighted as shown in Figure 3.3. This enables applications such as autonomous lawnmowers, to accurately recognize and navigate within the defined lawn area while avoiding obstacles or unintended areas. The Canny edge detector plays a vital role in creating a reliable and precise map of the lawn's boundary for effective path planning and navigation.

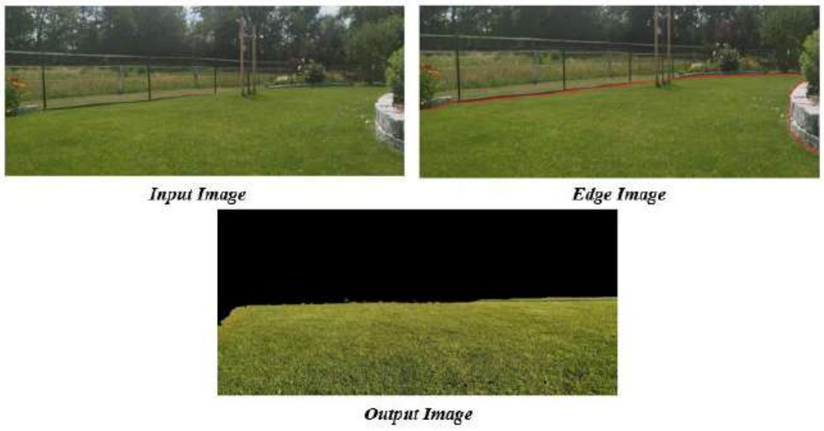


Figure 3.3 Canny Edge Detection [13]

3.3 Design of the Initial Project Software/Algorithm

The algorithm designing for CNN is done on Spyder notebook using Anaconda Navigator in which model is trained after multiple iterations then object segmentation is performed by training the CNN model through which object classes are identified and probability is measured. A classifier categorizes data into set of classes. One classifier is used for CNN model which is developed from scratch in CNN. The classifier has data set of 30 classes. Approximately 130 images are present in each class. Input image size for CNN from scratch is 224 x 224.

The canny edge detector using OpenCV has been implemented on Jupyter notebook. Input image size for lawn boundary detection is 255 x 255.

3.3.1 Dataset for CNN Object Detection Model

Following in Figure 3.4 are the classes of images present in dataset. Each image has size 224 x 224. 30 classes are a part of the dataset. Each image belongs to different class and each class has a unique name. The algorithm logic is built in such a way that with the recognition of each object, the name of class is also displayed which shows the corresponding class of image to which it belongs.

Figure 3.4 Dataset Sample Images

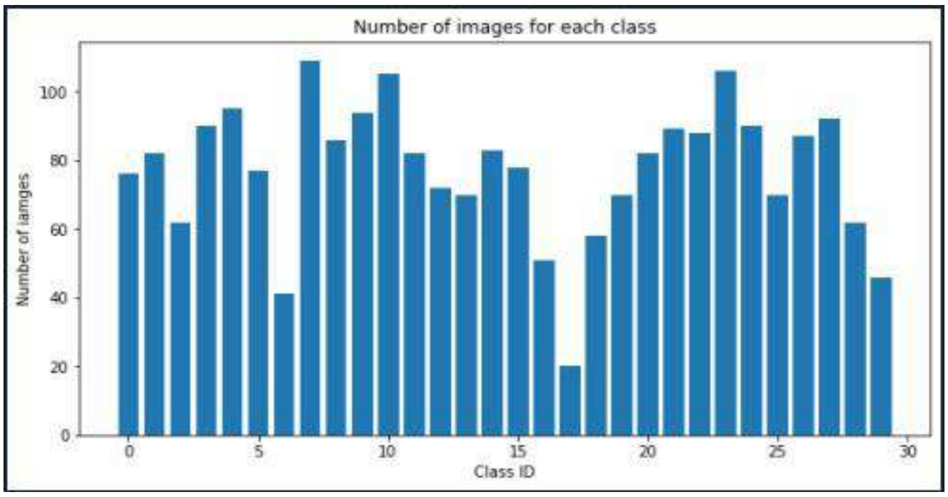


Figure 3.5 Dataset Classes

3.3.2 Working of Canny Edge Detection Algorithm

The technique has been implemented to identify abrupt changes in intensity or color between adjacent pixels, indicating the presence of lawn boundaries and significant features present in the image.

1. **Noise Reduction:** The image is smoothed using a Gaussian filter to reduce noise and remove small, insignificant details.
2. **Gradient Calculation:** The gradient magnitude and direction at each pixel are computed. This helps identify regions with rapid intensity changes.
3. **Non-Maximum Suppression:** The gradient magnitude is examined to suppress non-maximum values along the edges, thinning the edges to a single pixel width.
4. **Double Thresholding:** A pair of thresholds (high and low) are used to categorize edges as strong, weak, or non-edges. Strong edges are those with gradient values above the high threshold. Weak edges fall between the high and low thresholds. Non-edges have gradient values below the low threshold.
5. **Edge Tracking by Hysteresis:** Weak edges are traced along the strong edges using connectivity. If weak edges are connected to strong edges, they are considered part of the edge. Otherwise, they are suppressed.

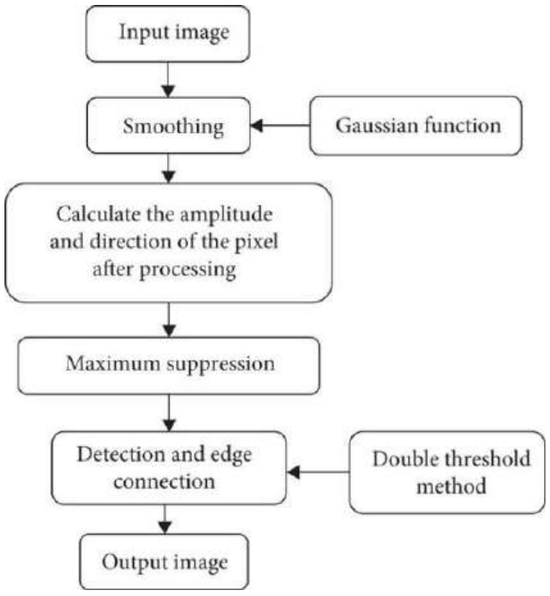

Input image
Smoothing
Gaussian function
Calculate the amplitude and direction of the pixel after processing
Maximum suppression
Detection and edge connection
Double threshold method
Output image

Figure 3.6 Canny Edge Detection Algorithm [14]

Canny edge detection has resulted in an extracted image as a result of the algorithm as shown in Figure 3.6 with pixel values indicating the presence of lawn edges. It is highly sensitive to detecting prominent edges while suppressing noise and weak edges.

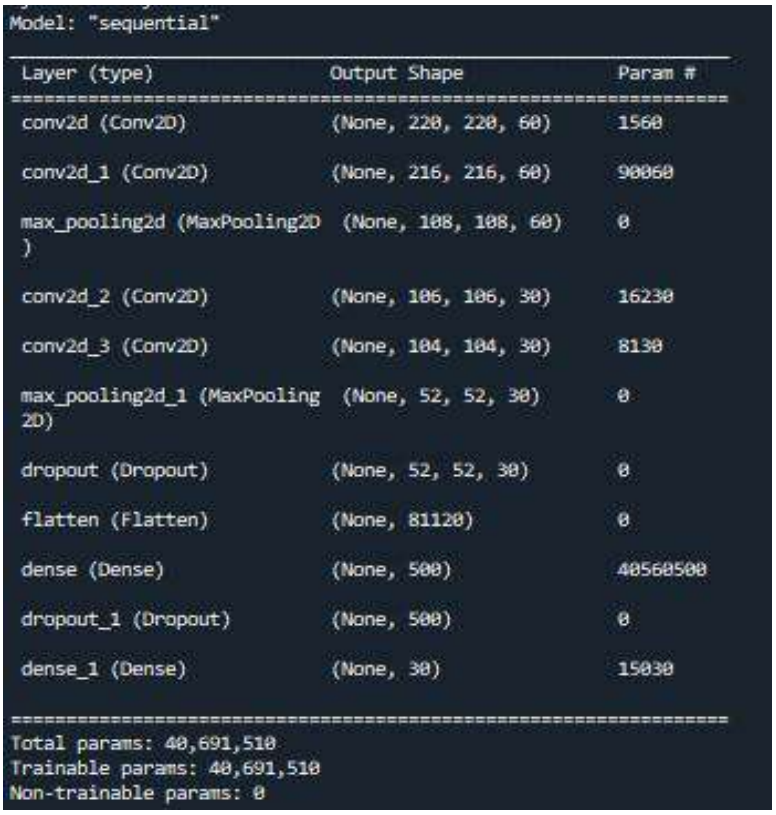
Canny's ability to produce accurate, well-defined edges has made it a cornerstone in various applications, such as object detection, image segmentation, robotic navigation, and more.

3.4 Details of Simulations / Modeling

The various simulations and models implemented have been explained in the following section. The step-by-step process involved in both CNN based object detection and Canny Edge based lawn boundary detection are discussed with regards to their overall architectures and working principles.

3.4.1 CNN Model Summary for Object Detection

CNN model summary is shown in Figure 3.7 which includes information of the parameters used in each layer and also shows the output shape of each layer.



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 220, 220, 60)      1560

 conv2d_1 (Conv2D)           (None, 216, 216, 60)      90060

 max_pooling2d (MaxPooling2D  (None, 108, 108, 60)     0
 )

 conv2d_2 (Conv2D)           (None, 106, 106, 30)      16230

 conv2d_3 (Conv2D)           (None, 104, 104, 30)      8130

 max_pooling2d_1 (MaxPooling  (None, 52, 52, 30)       0
 2D)

 dropout (Dropout)           (None, 52, 52, 30)        0

 flatten (Flatten)           (None, 81120)             0

 dense (Dense)               (None, 500)               40560500

 dropout_1 (Dropout)         (None, 500)               0

 dense_1 (Dense)             (None, 30)                15030

=================================================================
Total params: 40,691,510
Trainable params: 40,691,510
Non-trainable params: 0
```

Figure 3.7 CNN Model Summary

The output of the CNN is also a 4D array. Where batch size i.e number of samples processed would be the same as input batch size but the other 3 dimensions of the image might change depending upon the values of filter and kernel size.

From the above Figure 3.7, the output details obtained from CNN model summary are listed in Table 3.2.

Table 3.1 CNN Model Summary

Layer Type	Output Shape	Parameters
Conv2D	(None, 220,220,60)	1560
Conv2D_1	(None, 216,216,60)	90060
Max_Pooling 2D	(None, 108,108,60)	0
Conv2D_2	(None, 106,106,30)	16230
Conv2D_3	(None, 104,104,30)	8130
Max_Pooling 2D_1	(None, 52,52,30)	0
Dropout	(None, 52,52,30)	0
Flatten	(None,81120)	0
Dense	(None,500)	40560500
Dropout_1	(None,500)	0
Dense_1	(None,30)	15030

The first column in the Table shows the layer type which defines that first convolution has an output with shape (None, 220,220,60), where None is the batch size, 220 and 220 are the size of the resulting image, 60 are the number of filters of this convolution and also the number of channels in its output. Next is Max pooling layer that takes the output of the convolution as input. The output of the pooling has shape (None, 108,108,60), so it divided the size of image by two, leaving the rest as it was. Then there is another convolution, taking the output of the pooling as input, the output shape of this new convolution is (None, 106,106,30). 30 filters are here. After that, there is Max pooling layer. The output of the pooling has shape (None, 52,52,30). Then is the dropout layer. Then the flatten layer, which takes the images and transform them into a single vector, output shape is (None,81120), where None is still the batch size untouched, the 81120 are all elements present as the input, now in a single vector, one vector per sample in the batch. Then Dense layers, the first with 500 units, the second with 30 units. The final output shape of model is (None, 30). It outputs 30 values per sample in the batch.

3.4.2 Lawn Boundary Detection Using OpenCV

Lawn boundary detection using OpenCV has been accomplished by using computer vision techniques under the canny edge detector to identify and delineate the edges of the lawn area [15].

Firstly, we obtain an image or a sequence of images that capture the lawn area. This can be done using a camera or by loading pre-captured images. Then we apply preprocessing techniques to enhance the image quality and facilitate analysis. Common preprocessing steps include image resizing, noise removal, and color adjustment. Then we use OpenCV's edge detection algorithms to identify the edges within the image. The Canny edge detection algorithm is a popular choice in OpenCV for this task. It identifies areas with significant intensity changes, highlighting potential lawn boundaries. Afterwards thresholding and contour extraction is applied to convert the edge-detected image into a binary format using thresholding techniques. Thresholding converts the grayscale image into a binary image, where pixels are classified as either foreground (lawn boundary) or background (non-lawn area). Then, extract contours from the binary image using OpenCV's contour extraction functions. Lastly the detected lawn boundary is drawn on the original image using OpenCV's drawing functions. This helps visualize the identified lawn boundary for further analysis or display purposes.

By employing OpenCV's capabilities for edge detection, thresholding, contour extraction, and image manipulation, lawn boundary detection has been implemented. However, the effectiveness of this approach depends on factors such as image quality, lighting conditions, lawn complexity, and the presence of occlusions or variations in lawn texture.

After the performance of object detection using CNN and lawn edge detection using Canny edge detector the next step in this process is to implement these algorithms in real time using live images. This requires the availability of a high-quality camera and an equally capable GPU in order to process input and operate the system based on the output. The choice for camera in this case is narrowed down to stereo-vision cameras due to their fast and apt image localization capabilities. As for the GPU required for the task of processing real time images, an adequate option would be the AMD Radeon™ RX 6600 Graphics Card or other such choices having similar characteristics. Due to its high-performance parameters and economical demand as compared to other available options, this would prove to be a good choice.

But owing to the limited budget available for this project and the procurement limitations present, it has been decided, through careful consideration and more than adequate search for a way through in order to continue on this approach, that the proposed designs must be reshaped in order to achieve the ultimate aim of autonomous lawn mowing through the use of other available and effective methods.

3.5 Implemented Design Methodology

The final implemented design methodology has been undertaken using a cohesive approach for the detection of lawn boundary, the algorithm developed for charting out a map while also detecting and avoiding obstacles using ultrasonic sensors and the control of lawn mower using the required signals and instructions. The design is to work within relevant area using ultrasonic sensors and color sensors for grass detection working in unison. Firstly, to identify whether the prototype is on grass or not, using color sensor giving input. Secondly, to identify the length of grass to decide if grass has already been cut or not. Lastly, using the ultrasonic sensors, obstacles and lawn boundary are detected, and under the garb of state machine which is making decisions to avoid oncoming obstacles and following the boundary based on distances from sensor inputs. Design Implemented for project work is illustrated in Figure 3.8.

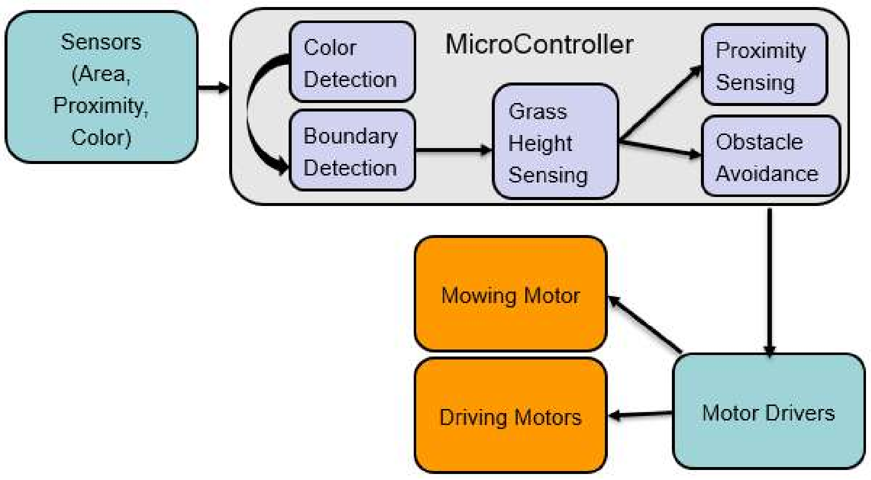
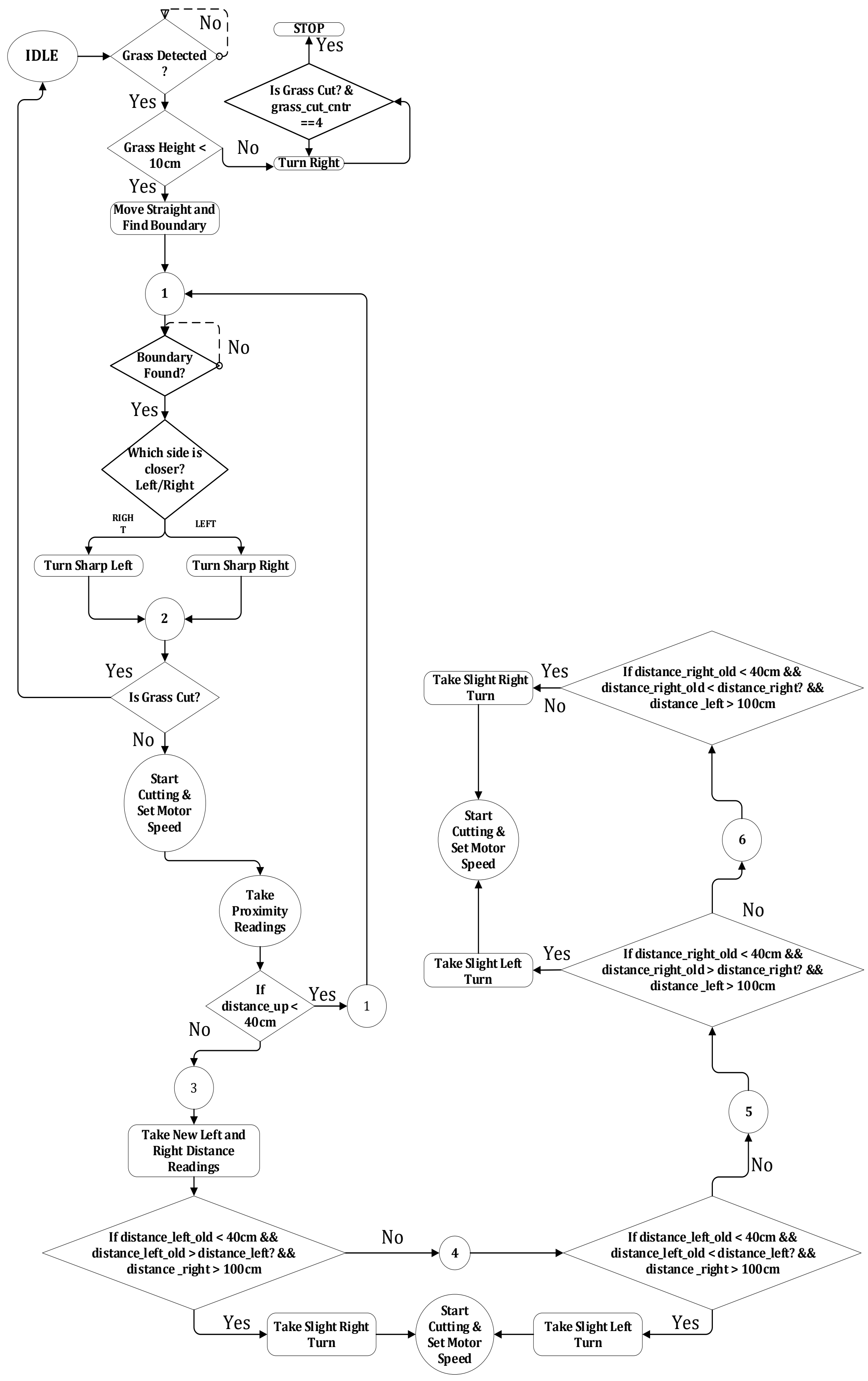


Figure 3.8 Implemented Design Methodology

3.6 Design of Implemented Software Algorithm

The implemented software algorithm as illustrated in detail in Figure 3.9 operates on the fusion of ultrasonic sensors for proximity and color sensor for grass detection.



IDLE
Grass Detected ?
No
Yes
Grass Height < 10cm
No
Yes
Move Straight and Find Boundary
STOP
Yes
Is Grass Cut? & grass_cut_cntr ==4
Turn Right
1
Boundary Found?
No
Yes
Which side is closer? Left/Right
RIGHT
LEFT
Turn Sharp Left
Turn Sharp Right
2
Yes
Is Grass Cut?
No
Start Cutting & Set Motor Speed
Take Proximity Readings
If distance_up < 40cm
Yes
1
No
3
Take New Left and Right Distance Readings
If distance_left_old < 40cm && distance_left_old > distance_left? && distance _right > 100cm
No
4
If distance_left_old < 40cm && distance_left_old < distance_left? && distance _right > 100cm
Yes
Take Slight Right Turn
Start Cutting & Set Motor Speed
Take Slight Left Turn
Yes
No
5
If distance_right_old < 40cm && distance_right_old > distance_right? && distance _left > 100cm
Yes
Take Slight Left Turn
No
6
If distance_right_old < 40cm && distance_right_old < distance_right? && distance _left > 100cm
Yes
No
Take Slight Right Turn
Start Cutting & Set Motor Speed

Figure 3.9 Implemented Algorithm State Machine

This algorithm has been developed on the Arduino IDE platform. The practical implementation of the algorithm in code is essentially the translation of the state machine developed for the autonomous function of the robot. This state machine involves the input and decision making with regards to proximity sensing, grass height measurement and color detection. The state machine has been expressed in terms of flowchart as illustrated in Figure 3.9. Following is the detailed description of the purpose and working of each state.

State IDLE: The idle state is the starting state of the algorithm depicted in Figure 3.10. As the system turns on, it first detects grass using the color sensor. If the color sensor senses no grass then it continues in this loop until grass is detected. Upon detection of grass, the height of the grass is detected using ultrasonic sensor. If the grass height is less than 10 cm, indicating that the grass is already cut, the agent is to turn right and traverse for a duration of 1 second. If the grass height is still not within range, it is to turn right again and follow on this process for a count of 4 times upon which the agent is instructed to stop. Otherwise, if the height is within range, the agent moves straight to find the boundary. Thus moving onto state 1.

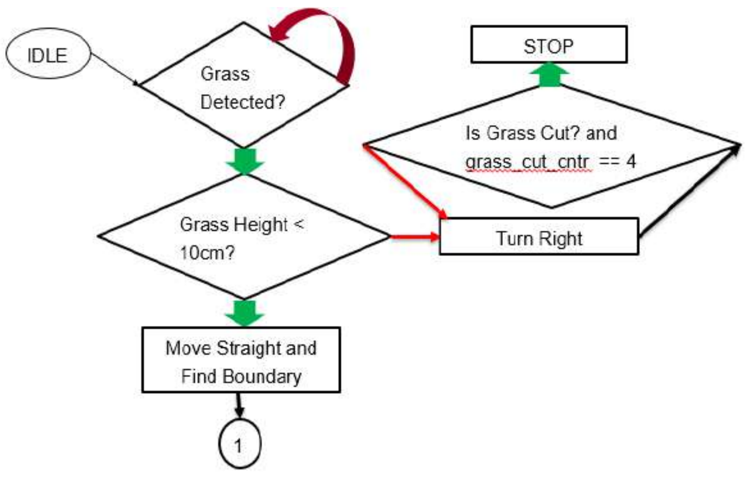

IDLE
Grass Detected?
STOP
Is Grass Cut? and grass_cut_cntr == 4
Grass Height < 10cm?
Turn Right
Move Straight and Find Boundary
1

Figure 3.10 State: IDLE

State 1: In the Figure 3.11, as the agent moves to find the boundary, it checks both the left and right-side inputs to seek out the boundary closest to it. In the case when the boundary is found, if on the right side then it takes a sharp left turn and moves on to the next state. If on the left side then it takes a sharp right turn and moves to the next state.

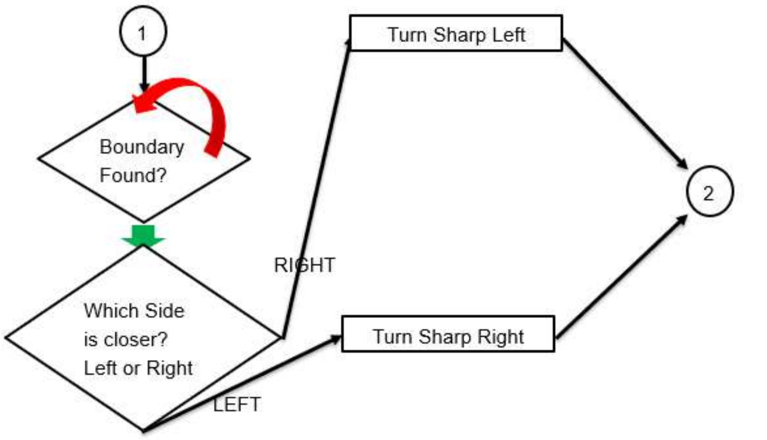


Figure 3.11 State 1: Boundary Check

State 2: In the Figure 3.12, after the agent takes a sharp turn to align itself parallel to the boundary, it moves on to check whether the grass is cut or not. Based on this, it goes to idle state or moves to start cutting and set motor speed to programmed value. From there it takes proximity readings to detect whether a boundary or obstacle is detected. If front distance is less than 40 cm then it goes to state 1, other it goes to state 3.

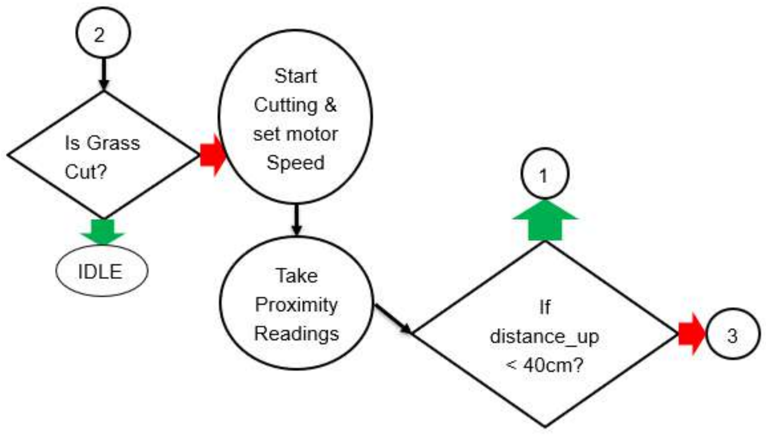


Figure 3.12 State 2: Obstacle Check

State 3: The 3rd state deals with the slight deviations off the driving path that the motors induce due to slip making the agent move off course as depicted in Figure 3.13. To course correct these deviations, new left and right proximity readings are taken to judge extent of deviations. If the old left side reading is less than 40 cm while also being greater than the new left side reading and the right side reading is greater than 100 cm (indicating that no boundary is approaching from that side) then the agent makes slight right turn adjustment and continues on to cut and move at intended motor speed. Otherwise, it moves on to state 4.

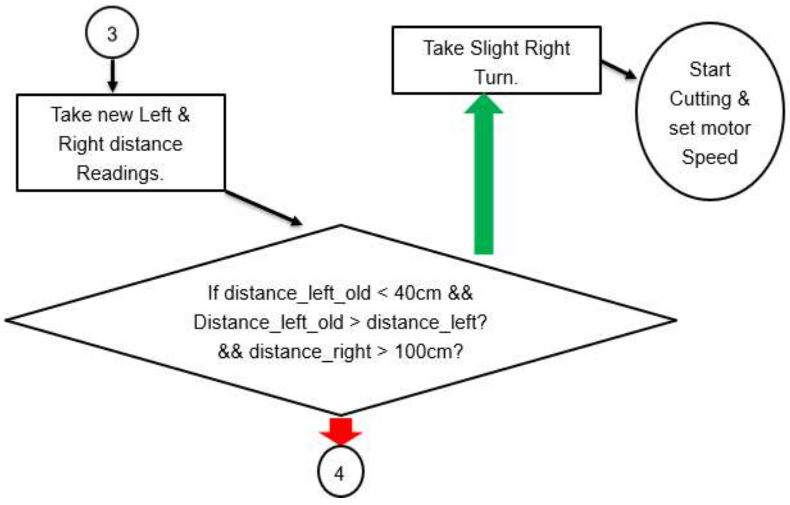

3
Take new Left & Right distance Readings.
Take Slight Right Turn.
Start Cutting & set motor Speed
If distance_left_old < 40cm && Distance_left_old > distance_left? && distance_right > 100cm?
4

Figure 3.13 State 3: Left-Left Deviation Check

State 4: If the deviations of the driving path are not towards the boundary, then to detect whether the agent has deviated away from the boundary it relies on comparison of old and new readings. To course correct these deviations, the new left and right proximity readings that have been taken are used to judge the extent of deviations. If the old left side reading is less than 40 cm while also being less than the new left side reading and the right side reading is greater than 100 cm (indicating that no boundary is approaching from that side) then the agent makes slight left turn adjustment and continues on to cut and move at intended motor speed, depicted in Figure 3.14. Otherwise, it moves on to state 5.

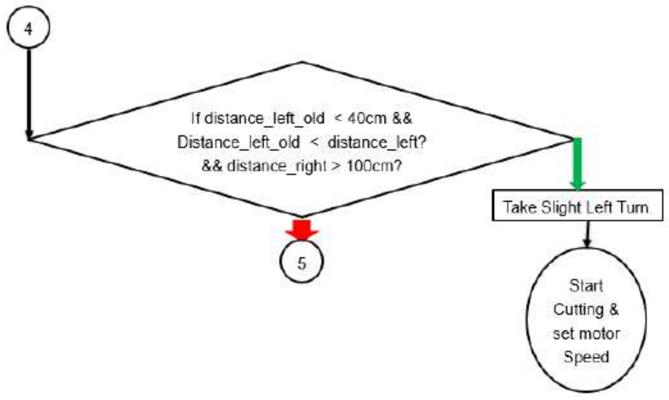

4
If distance_left_old < 40cm && Distance_left_old < distance_left? && distance_right > 100cm?
Take Slight Left Turn
Start Cutting & set motor Speed
5

Figure 3.14 State 4: Left-Right Deviation Check

State 5: In order to detect whether the deviations of the driving path are towards or away from the boundary on the right side, it relies on comparison of old and new readings. To course correct these deviations, the new left and right proximity readings that have been taken are used to judge the extent of deviations. If the old right side reading is less than 40 cm while being greater than the new right side reading and the left side reading is greater than 100 cm (indicating that no boundary is approaching from that side) then the agent makes slight left turn adjustment and continues to cut and move at intended motor speed, as in Figure 3.15. Otherwise, it moves on to state 6.

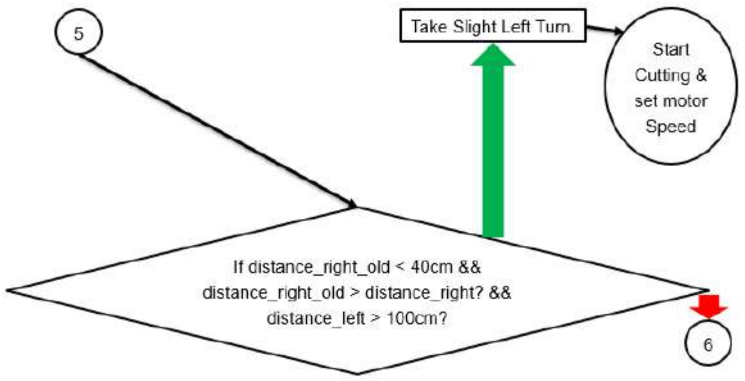

5
Take Slight Left Turn
Start Cutting & set motor Speed
If distance_right_old < 40cm && distance_right_old > distance_right? && distance_left > 100cm?
6

Figure 3.15 State 5: Right-Right Deviation Check

State 6: Finally, If the deviations of the driving path are not approaching the boundary then to detect whether the agent has deviated away from the boundary it relies on comparison of old and new readings. To course correct these deviations, the new left and right proximity readings that have already been taken are used to judge the extent of deviations. If the old right side reading is less than 40 cm while also being less than the new right side reading and the left side reading is greater than 100 cm (indicating that no boundary is approaching from that side) then the agent makes slight right turn adjustment and continues on to cut and move at intended motor speed as illustrated in Figure 3.16.

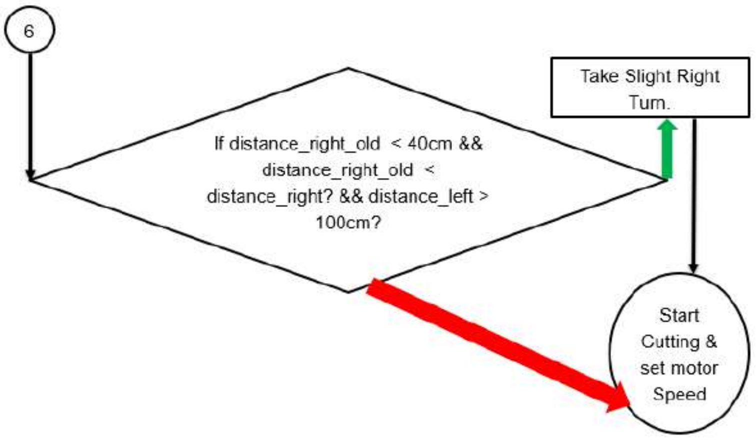

6
Take Slight Right Turn.
If distance_right_old < 40cm && distance_right_old < distance_right? && distance_left > 100cm?
Start Cutting & set motor Speed

Figure 3.16 State 6: Right-Left Deviation Check

3.7 Design of the Project Hardware

This design procedure consists of proposed design for the hardware prototype of the lawn mower model. Camera is to be used for image acquisition. Ultrasonic sensors are to be used for obstacle avoidance beyond the scope of image processing model.

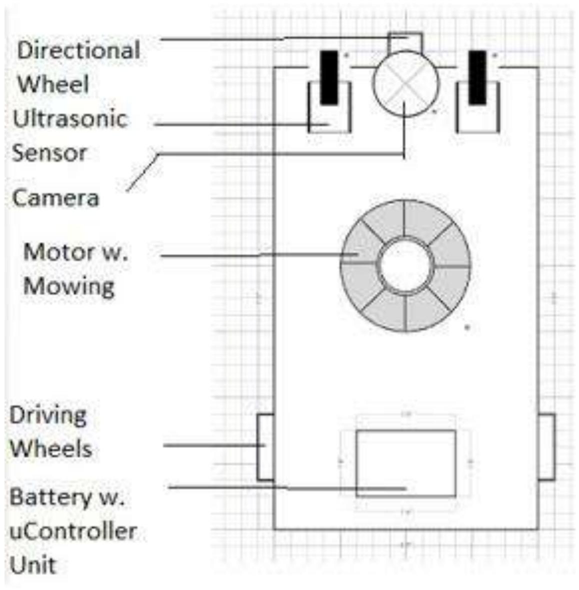


Figure 3.17 Hardware Diagram of Proposed Design

Raspberry pi is the proposed platform that may be used for image processing since it is fast and efficient. It needs 5V input to function. The motors will be supplied respective control signals for desired output. The prototype was initially intended to be designed as per the design diagram shown in Figure 3.17.

Due to the revision of approach from AI-based to a fusion of sensors, the hardware design has also been updated as per the requirements. The updated design includes two ultrasonic sensors placed on the left and right of the prototype base (perpendicularly). While there are two ultrasonic sensors placed in the front of the prototype.

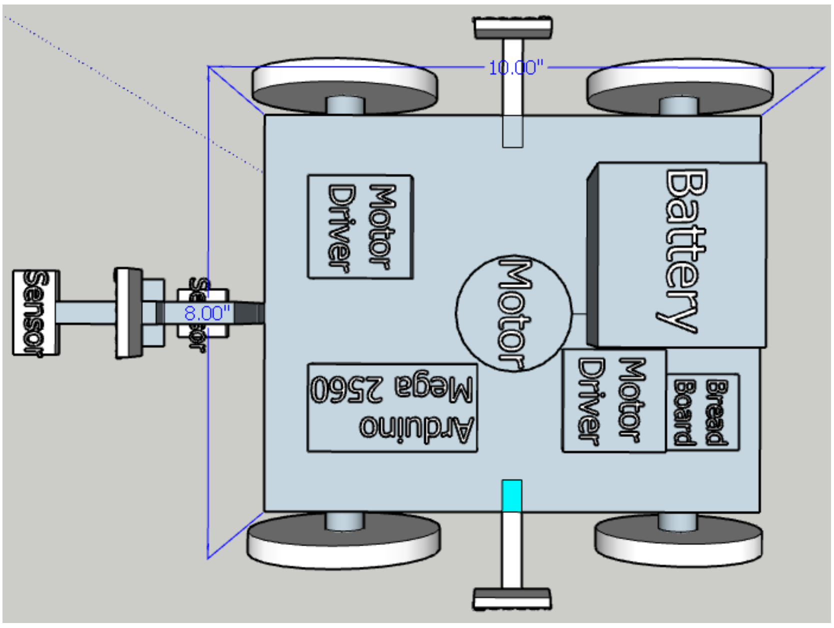


Figure 3.18 Revised Hardware Design - Top View

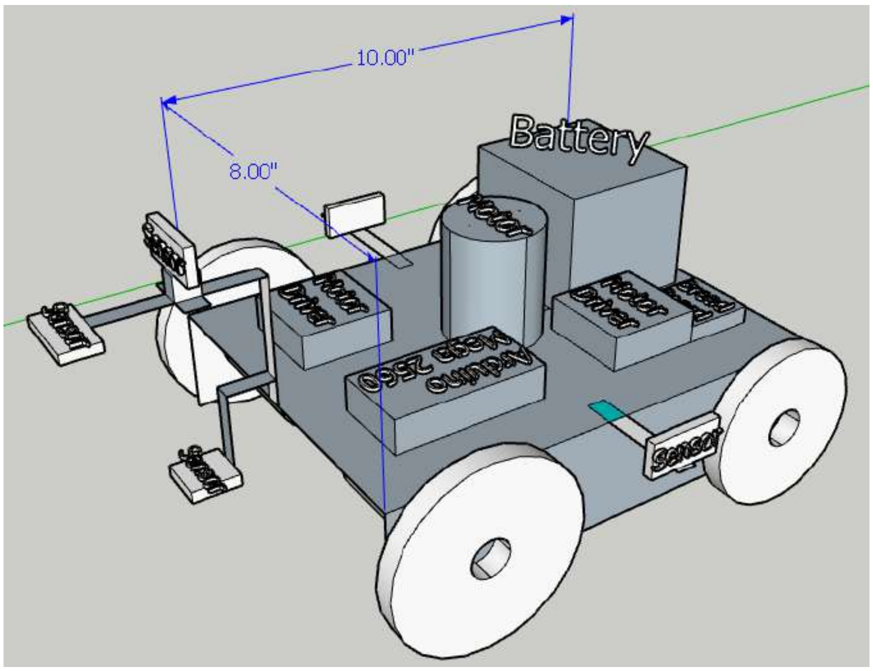


Figure 3.19 Revised Hardware Design - Diagonal Perspective

While there are two ultrasonic sensors placed in the front of the prototype which are the front facing sensor and the down facing grass height sensor. Alongside the down facing ultrasonic sensor is placed the color sensor for grass sensing. The hardware designs have been carefully and accurately drawn out on SketchUp software. The top view and diagonal perspective are illustrated in Figure 3.18 and Figure 3.19 respectively.

3.7.1 Obstacle Detection Using Ultrasonic Sensor

A system of obstacle detection has been made to identify barriers in a given space as shown in Figure 3.20. These systems may be applied in a wide range of contexts, including automobiles to assist with driving, robotics to aid with navigation, and industrial settings to safeguard personnel and machinery. Radar, lidar, cameras, and ultrasonic sensors are typical forms of obstacle detection systems. These systems employ a variety of methods, such as measuring distance, using infrared radiation, or analyzing pictures, to find impediments. Ultrasonic sensors may be used to construct an obstacle detection system using Arduino. These sensors send out ultrasonic waves, which they then time as they return to the sensor. The distance to the obstruction can be determined by timing the event and measuring the sound speed. The Arduino may then be programmed to perform tasks depending on the distance, such halting a robot or setting off an alert. In addition, an infrared sensor, which produces infrared light and detects the reflected beams, can be utilized for obstacle detection. With the help of the Arduino's digital or analogue pins and a programming language similar to C or C++, both of these sensors may be connected to the microcontroller.

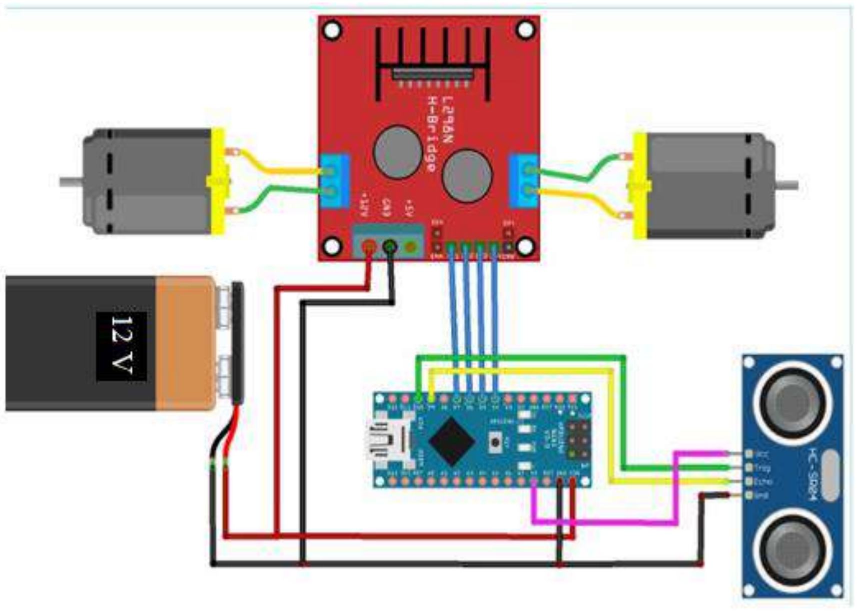


Figure 3.20 Obstacle Detection using Ultrasonic Sensor [16]

3.7.2 Proximity Sensing Using Ultrasonic Sensor

Proximity sensing using an ultrasonic sensor is a common method to detect the presence and distance of objects and boundaries in various applications. Ultrasonic sensors work based on the principle of sound waves. They emit high-frequency sound waves (ultrasonic waves) and measure the time it takes for the waves to bounce back after hitting an object. The time delay is then used to calculate the distance between the sensor and the object. The sensor emits a short burst of ultrasonic waves and starts a timer. The waves propagate through the air, and upon encountering an object or boundary, bounce back towards the sensor. The receiver detects the reflected waves. Using the known speed of sound, the sensor calculates the distance by dividing the total time by two. The result represents the approximate distance between the sensor and the boundary.

3.7.3 Grass Area sensing using Color Sensor

Grass area sensing using a color sensor can be utilized to detect and analyze the presence of grass in a given area. The TCS3200 color sensor, can be employed to measure the color components of the area. These sensors typically consist of an array of photodiodes that detect different wavelengths of light. The color sensor is positioned above the ground and directed towards the target area. By illuminating the area with a light source, the color sensor can capture the reflected light and analyze the RGB (Red, Green, Blue) values. The color sensor provides the intensity of red, green, and blue light components reflected by the area underneath. By comparing these values to reference values or predetermined thresholds, it is determined whether or not the area underneath is grass or not based on detection of green color. The RGB values obtained from the color sensor can be processed using algorithms or calibration curves to convert the color data into meaningful information about the area. This processing step may involve normalization, color space conversion, or comparison with predefined color thresholds [17].

3.8 Summary

This chapter summarizes the project software and hardware design and the model training of the object detection and lawn boundary detection using OpenCV. The implemented design using a fusion of sensors is also discussed. Block diagrams of software and hardware design have been shown in this chapter. The project implementation procedure is also discussed at the end which includes details of software as well as hardware implementation of both the initial design and the implemented design. The software algorithm in the form of a state machine developed for the purpose of achieving autonomous control has been explained in detail state by state.

Chapter 4

TOOLS AND TECHNIQUES

In this chapter, hardware and software tools that are used for the project work are discussed. The hardware tools include motors which are servo motors, DC Gear motors, various proximity sensors which include ultrasonic sensors, color sensors, a controller Arduino used for processing and controlling motor movements and sensor inputs, and camera for image acquisition. The software tools include Anaconda, Arduino IDE is used as programming language platform for model training, algorithm implementation and prototype implementation.

4.1 Hardware Tools

Hardware components that have been used for the practical implementation of designed work are illustrated below.

4.1.1 Servo Motor

A control circuit, a servo motor, a shaft, a potentiometer, driving gears, an amplifier, and either an encoder or a solver is all part of a closed loop system. A servo motor is a self-supported electric device that rotates machine parts precisely and efficiently as shown in Figure 4.1. The output shaft of a regular motor cannot be rotated to a specific angle, position, or speed, but the output shaft of that motor can. In the servo motor, a standard motor is attached to the sensor for location impressions. The controller is the most important part of the servo motor, which was specially designed for this purpose [18].

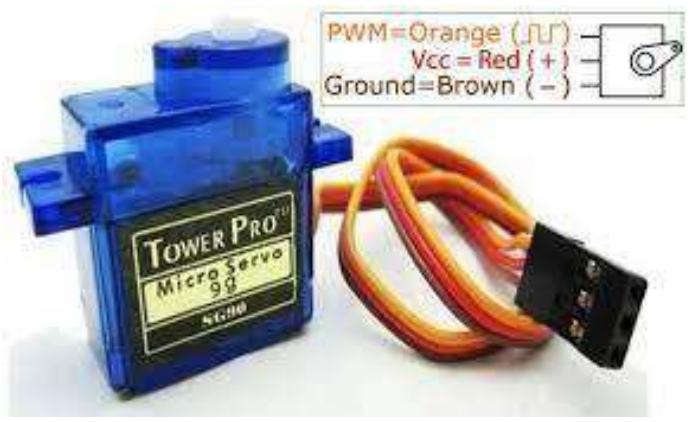


Figure 4.1 Servo Motor [19]

4.1.2 Camera

Digital cameras capture images using a sensor composed of millions of photosites that convert incoming light into electrical signals. These signals are then digitized and processed into a digital image. The camera's lens focuses light onto the sensor, while the aperture controls the amount of light reaching it. The sensor's photosites record the intensity of light, and the camera's image processor converts these values into a digital format. The resulting image can be stored in various formats, such as JPEG or RAW. Digital cameras offer adjustable settings for ISO sensitivity, shutter speed, and aperture, allowing users to control exposure and achieve desired effects. Illustrated in Figure 4.2.

Figure 4.2 Webcam as Camera [20]

4.1.3 DC Gear Motor

rotating at a slower speed as shown in Figure 4.3. The gear arrangement aids in matching the output characteristics of the motor to the needs of the individual application. The torque output and speed of a DC gear motor are determined by the motor's design and gear ratio. When compared to a regular DC motor, gear motors are intended to generate more torque at lower speeds. DC gear motors are used in a variety of sectors where precise control of speed and torque is required. They are widely utilized in robotics, electric cars, industrial machinery, and household products like as electronic locks, vending machines, and actuators [21].

Figure 4.3 DC Gear Motor [22]

4.1.4 Color Sensor

The color sensor used in this project is made up of a series of photodiodes and a white LED that lights the item being detected. By transforming light intensity into electrical impulses, the sensor can detect a wide spectrum of colors. It has customizable color sensing capabilities, allowing customers to tailor the sensitivity and integration time to their own requirements. The TCS3200 may be interfaced with microcontrollers or Arduino boards, making it useful for color sorting, detection, and analysis projects. The TCS3200 color sensor, as shown in Figure 4.4, with its small size and adaptability, provides an effective and dependable solution for color-related job [23].

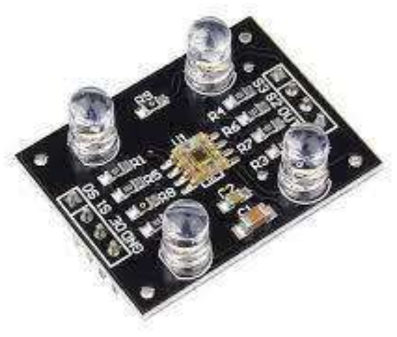


Figure 4.4 TCS3200 Color Sensor [24]

4.1.5 Micro-Controllers

Microcontrollers enable robotics by controlling the motors, servos, sensors, and other components that make up the prototype's body and behavior. Various controllers are available for the purpose of hardware implementation, such as Arduino, Raspberry Pi etc. Listed below in Table 4.1 is a comparison conducted between Arduino and Raspberry Pi controllers to seek out the best suited option for the project requirements.

Table 4.1 Analysis of Micro Controllers

Arduino	Raspberry Pi
1. The Arduino microcontroller has a Central Processing Unit (CPU), RAM (Random Access Memory), and ROM (Readable Only Memory).	1. The Raspberry Pi is a small computer that runs on a microprocessor and has all the essential components of a desktop computer, including a CPU, memory, storage,

2. No operating system is required for Arduino. It features a 16 MHz clock speed. 3. While Arduino is great for integrating sensors and controlling LEDs and motors, it does not facilitate the development of any Python-based software applications.	graphics driver, and connections. 2. A working operating system is required for the Raspberry Pi. Raspberry Pi has a 1.2 GHz clock speed. 3. The Raspberry Pi is useful for creating Python-based software applications, but it cannot connect to sensors and LEDs without a GPIO header.

Based on this comparison the Arduino has been selected as the controller for this project. The ATmega2560, as shown in Figure 4.5, is a microcontroller board known as Arduino Mega. In addition to a 16 MHz ceramic resonator (CSTCE16M0V53-R0), it contains a USB connector, a power jack, an ICSP header, six analogue inputs, fourteen digital input/output pins, six of which may be used as PWM outputs, and a reset button. To get started, just connect it to a computer, an AC-to-DC adapter, or a battery using a USB cable; it comes with everything needed to support the microcontroller [25].

Figure 4.5 Arduino Mega [26]

4.1.6 Ultrasonic Sensor

An ultrasonic sensor is a sensor that uses ultrasound, which travels through the air, to measure distances. The ultrasound will bounce back in the direction of the sensor if it encounters a wall or other obstruction on its way [27].

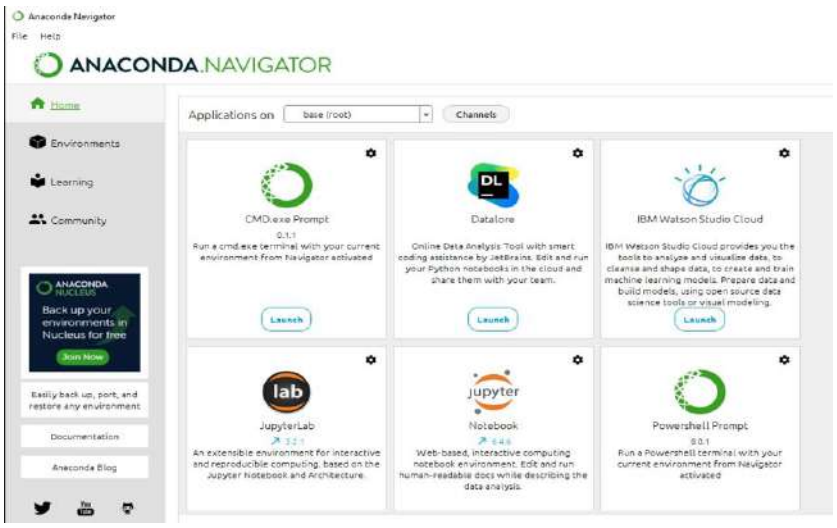
- The transmitter (trig pin) first transmits a sound wave.
- The wave is picked up by the object and returned to the sensor by reflection.
- It is picked up by the receiver (echo pin).

4.2 Software and Simulation Tools

The details of software's used for the software implementation of project are listed below. Different algorithms are implemented on these software's.

4.2.1 Anaconda Navigator

Anaconda is a Python distribution having built-in set of packages and is extensively used in data research. Graphical user interface (GUI) tool i.e., Anaconda Navigator is a part of Anaconda distribution, as shown in Figure 4.6, that helps in easy setup, install, and run products like Jupyter Notebook and Spyder IDE. To get the input and output of a Python or R script, programming is performed and then documents are created accordingly. Created files are saved and are used as per requirements. By default, Python and R are supported, although Notebook may be conFigured to run a number of alternative operating environments [28].



Run a cmd.exe terminal with your current environment from Navigator activated

Online Data Analysis Tool with smart coding assistance by JetBrains. Edit and run your Python notebooks in the cloud and share them with your team.

IBM Watson Studio Cloud provides you the tools to analyze and visualize data, to cleanse and shape data, to create and train machine learning models. Prepare data and build models, using open source data science tools or visual modeling.

Easily back up, port, and restore any environment

An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

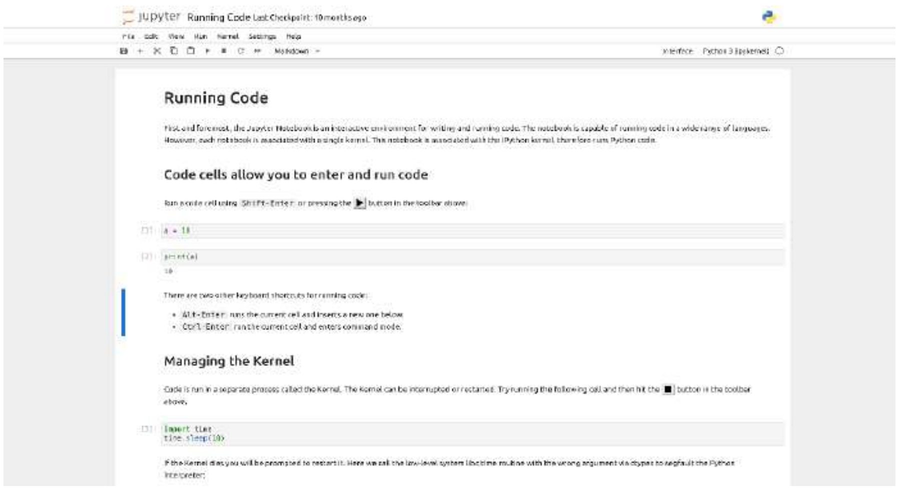
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Run a Powershell terminal with your current environment from Navigator activated

Figure 4.6 Anaconda Navigator

4.2.2 Jupyter Notebook

This project uses Jupyter notepad to inspire coding for the multiple models intended for image processing [29]. An open-source online application, as shown in Figure 4.7, the primary tasks carried out by Jupyter notebook are document creation and document sharing [30]. This also executes the live code, which enables functionality crucial for detecting grass space, obstacles, area mapping, and the task of image processing for boundary detection using OpenCV library can be effectively executed.


jupyter Running Code Last Checkpoint: 10 months ago
File Edit View Run Kernel Settings Help
Markdown
Interface Python 3 (ipykernel)

Running Code

First and foremost, the Jupyter Notebook is an interactive environment for writing and running code. The notebook is capable of running code in a wide range of languages. However, each notebook is associated with a single kernel. This notebook is associated with the IPython kernel, therefore runs Python code.

Code cells allow you to enter and run code

Run a code cell using `Shift-Enter` or pressing the ▶ button in the toolbar above:

```
[1]: a = 10
```

```
[2]: print(a)
10
```

There are two other keyboard shortcuts for running code:

- `Alt-Enter` runs the current cell and inserts a new one below.
- `Ctrl-Enter` run the current cell and enters command mode.

Managing the Kernel

Code is run in a separate process called the Kernel. The Kernel can be interrupted or restarted. Try running the following cell and then hit the ■ button in the toolbar above.

```
[3]: import time
time.sleep(10)
```

If the Kernel dies you will be prompted to restart it. Here we call the low-level system libc.time routine with the wrong argument via ctypes to segfault the Python interpreter:

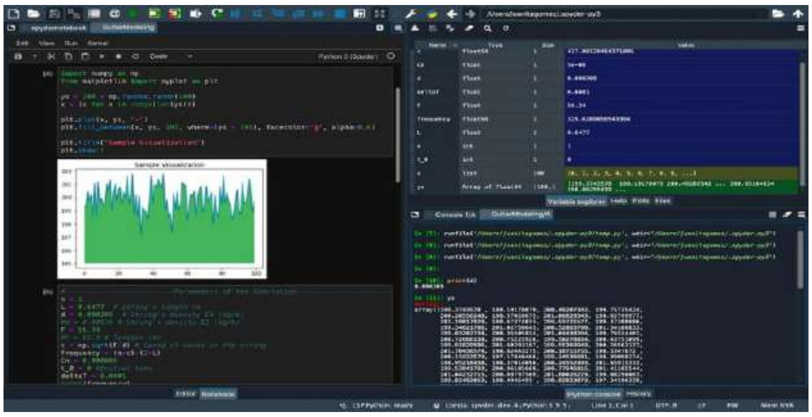
Figure 4.7 Jupyter Notebook [31]

It also performs other function such as:

- Machine learning
- Deep learning transformation
- Exploratory data analysis

4.2.3 Spyder IDE

Spyder is an integrated development environment (IDE) primarily used for Python programming and data analysis, as depicted in Figure 4.8. It offers a user-friendly interface with tools for code editing, debugging, visualization, and data exploration. Spyder provides an interactive environment with features like variable exploration, IPython console integration, and integration with scientific libraries. It's particularly popular among scientists, engineers, and data analysts for its focus on scientific computing and data manipulation tasks.



```
import numpy as np
from matplotlib import pyplot as plt
[illegible]
```

Figure 4.8 Spyder Notebook

4.2.4 Arduino IDE

The Arduino IDE (Integrated Development Environment) is a software platform that has been used extensively in this project for the implementation of algorithms and motor controls along with sensor interfacing based on the ATmega328P and ATmega2560 boards.

The Arduino IDE is an open-source software, making it freely available for download and modification. It is developed and maintained by the Arduino community. The Arduino IDE is compatible with multiple operating systems, including Windows, macOS, and Linux, making it accessible to a wide range of users. The IDE provides a code editor where users can write, edit, and organize their Arduino sketches (programs). It supports syntax highlighting and auto-completion, making it easier to write and navigate through the code. The Arduino IDE includes a library manager that allows users to easily add and manage libraries. Libraries provide pre-written code for specific functions or components, simplifying the development process. The IDE includes a serial monitor feature that allows users to communicate with the Arduino board via the serial port. It is useful for debugging, displaying sensor data, or receiving messages from the Arduino. The Arduino IDE is backed by a vibrant and supportive community. Users can find extensive documentation, tutorials, and examples on the Arduino website and other online resources. The community actively contributes libraries and code snippets that can be utilized in projects.

The Arduino IDE, whose logo is shown in Figure 4.9, offers a beginner-friendly environment for programming Arduino boards, enabling users to quickly start prototyping and building projects. Its simplicity, extensive documentation, and strong community support make it a popular choice for makers, hobbyists, and professionals alike [32].

Figure 4.9 Arduino IDE [32]

4.2.5 SketchUp

SketchUp is a user-friendly 3D modeling software, initially developed in August of 2000, that empowers individuals to create, visualize, and share their design ideas. Developed by Trimble, it offers a versatile platform for architects, engineers, artists, and hobbyists to bring their concepts to life. With its intuitive interface, users can swiftly generate intricate 3D models of buildings, interiors, landscapes, and objects. SketchUp provides an array of tools for precise geometry creation, texture application, and scene rendering. Its collaborative features enable real-time sharing and collaboration on projects. From professionals to beginners, SketchUp's accessibility and robust capabilities make it a popular choice for various design and visualization endeavors such as this project which requires detailed prototype designing for the efficient functioning and optimum space utilization by the various components being used. The sketchup user interactive interface in shown in Figure 4.10.

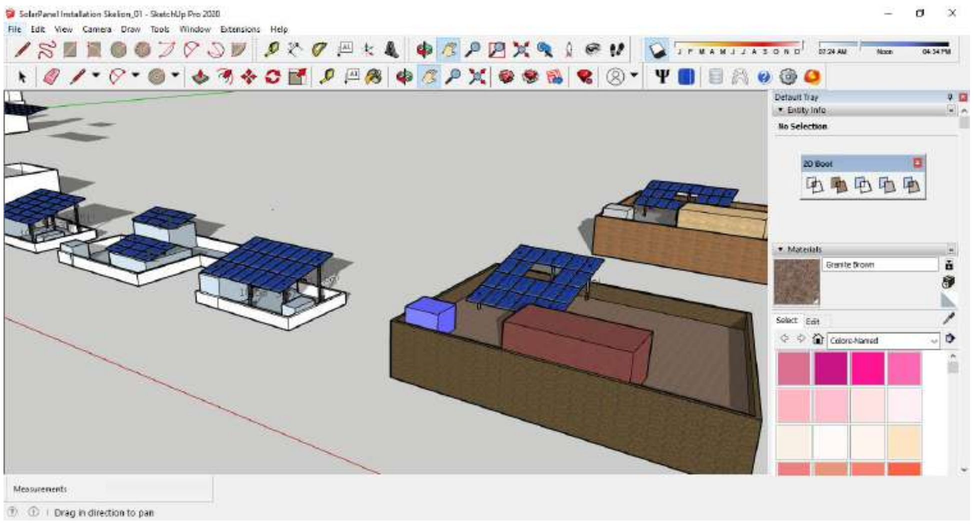

SolarPanel Installation Skelion_01 - SketchUp Pro 2020
File Edit View Camera Draw Tools Window Extensions Help
Default Tray
Entity Info
No Selection
Materials
Granite Brown
Colors-Named
Measurements
Drag in direction to pan

Figure 4.10 SketchUp User Interface

4.3 Summary

In this chapter, hardware and software tools used for the project work are discussed. Hardware tools consists of servo motors which are used due to high precision and efficiency, controller Arduino Mega and Uno that help in processing and moving motors, and webcam for image acquisition. In software tools Anaconda are used for programming.

Chapter 5

PROJECT RESULTS AND EVALUATION

This chapter discusses and explains the results obtained upon completion of software and hardware implementation of various models including common object detection, lawn boundary detection using OpenCV technique of canny edge detection, Arduino based outputs from sensors for boundary detection, obstacle detection and results from color sensor detecting working lawn area. The testing and evaluation of these results is only the next step in the natural flow of project activities.

5.1 Presentation of the Findings

Following are discussed and displayed the results acquired and achieved from the different models, techniques and prototype testing undergone for the achievement of the objectives of the project.

5.1.1 CNN Model Results

The results obtained from the CNN model trained for object detection include metrics such as precision, recall, and visualizations like bounding boxes around the detected objects. Successful CNN-based object detection is characterized by accurate localization, efficient identification in locating and classifying objects of interest in images provided, as shown in Figure 5.2.

1. **Accuracy:** Accuracy measures the proportion of correctly predicted instances (both true positives and true negatives) out of the total instances. It provides a general view of the model's overall performance, as shown below in Figure 5.1.

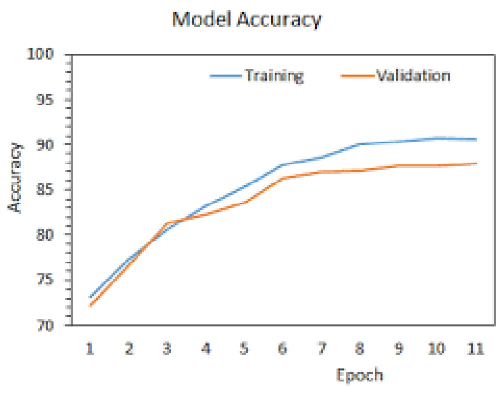


Figure 5.1 CNN Accuracy

2. **Precision:** Precision measures the ratio of correctly predicted positive instances (true positives) to all instances predicted as positive (true positives + false positives). It assesses the model's ability to avoid false positives. The results achieved are mentioned in Table 5.1.

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \qquad (Eq\ 5.1)$$

3. **Recall (Sensitivity or True Positive Rate):** Recall calculates the ratio of correctly predicted positive instances (true positives) to all actual positive instances (true positives + false negatives). It quantifies the model's ability to identify all relevant instances. The results achieved are mentioned in Table 5.1.

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \qquad (Eq\ 5.2)$$

4. **F1 Score:** The F1 score is the harmonic mean of precision and recall. It provides a balanced measure of a model's accuracy by considering both false positives and false negatives. F1 score is particularly useful when class distribution is imbalanced. The results achieved are mentioned in Table 5.1.

$$F1\ Score = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} \qquad (Eq\ 5.3)$$

$$\therefore\ F1\ Score = \frac{2\ \times Precision\ \times Recall}{Precision + Recall} \qquad (Eq\ 5.4)$$

Table 5.1 CNN Model Results

Metric (%)	**Model**	**Dataset for Object Detection**
Accuracy	**CNN**	88.71%
Precision		88.67%
Recall		88.67%
F1 Score		88.33%

In summary, accuracy gives an overall performance measure, precision focuses on minimizing false positives, recall emphasizes minimizing false negatives, and the F1 score balances both precision and recall. The choice of metric depends on the specific goals and requirements of the classification task.

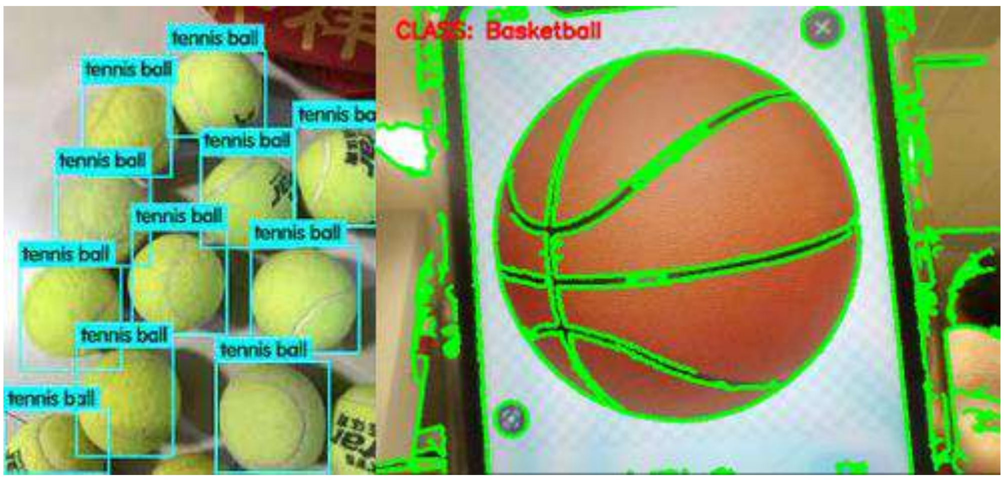


Figure 5.2 CNN Object Detection Results

5.1.2 Lawn Detection Results using Canny Edge

Canny edge detection results are the outcome of applying the Canny edge detection algorithm to the input image, as illustrated in Figure 5.3. These results consist of the image having edges highlighted, revealing the boundaries between the lawn area and other significant features. The quality of Canny edge detection results depends on parameters like threshold values and image characteristics, with well-defined and accurate edges indicating successful detection of intensity changes and potential object boundaries within the image.

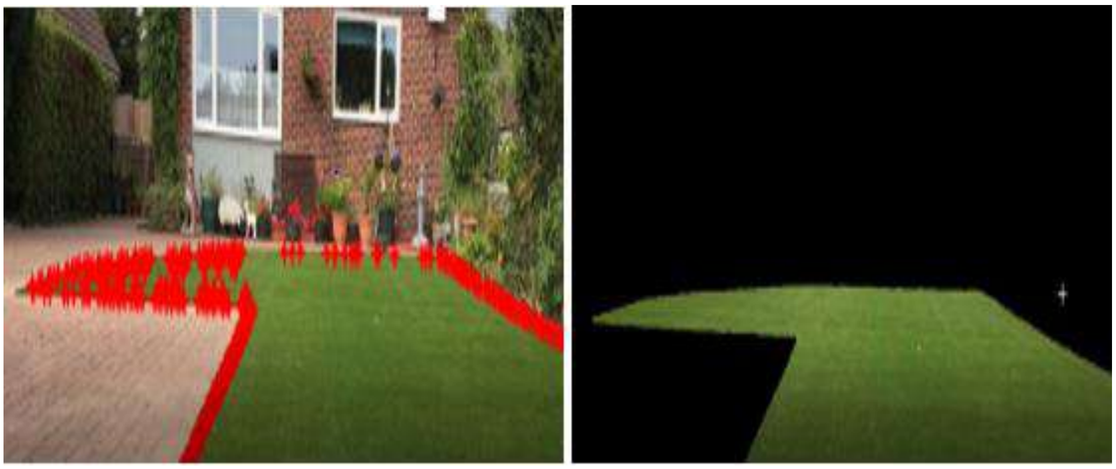


Figure 5.3 Canny Edge Lawn Boundary Detection Results

5.1.3 Ultrasonic Sensor Output

The ultrasonic sensor was tested for various conditions, one of which was proximity sensing and the other was for obstacle detection. The tests were conducted on sTable placements as well as on moving mower prototype. The different arrangements certainly affected the results. The output from static position of the sensor was accurate and consistent. Whereas the output from the moving position clearly showed slight uncertainty due to jitters initiated from movement. Under certain limit of speed of the mower the output was meaningful and reliable for the mower to make decisions based on them. The PWM testing performed has been evaluated in Table 5.2.

Table 5.2 PWM Setting Results

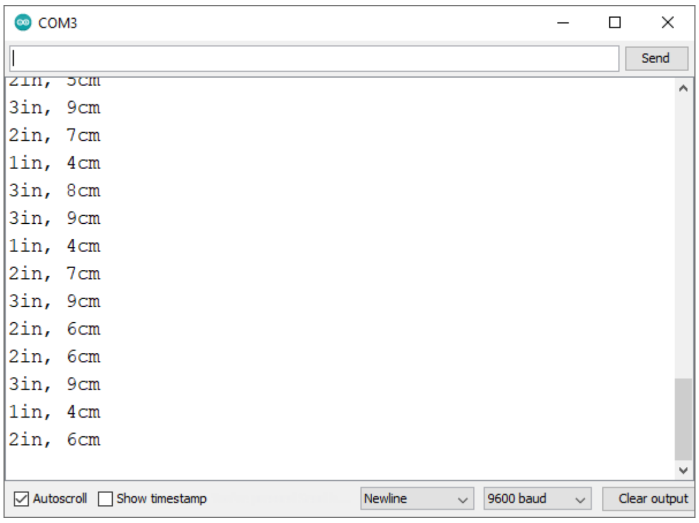
PWM (0-255) / %	**RPM (Max. 250 rpm)**	**Movement Speed ft/min**	**Sensor Error due to Jitters (+/-)**
64 / 25%	60 rpm	36 ft/min	+/- 1cm
90 / 35%	85 rpm	51 ft/min	+/- 3cm
102 / 40%	100 rpm	58 ft/min	+/- 6cm
127 / 50%	125 rpm	72 ft/min	+/- 10cm
153 / 60%	150 rpm	86 ft/min	+/- 14cm
204 / 80%	200 rpm	115 ft/min	+/- 20cm
229 / 90%	225 rpm	130 ft/min	+/- 28cm
255 / 100%	250 rpm	146 ft/min	+/- 35cm

This limit was observed to be 39% ≈ 40% PWM speed. The sensor output was observed to be accurate to +/- 6cm of actual value under these conditions. The movement speed of the prototype under this PWM limit was found to be adequate for efficient traversal and mowing with minimum sensor error rate while also covering lawn space in a suitable time period.

Another important observation for the ultrasonic sensor was that angles and edges greatly affected the sensed values due to uneven reflections of the sound waves, as shown in Table 5.3. Values obtained from serial monitor are shown in Figure 5.4.

Table 5.3 Ultrasonic Sensor Results

Sensor Position	Actual Distance (cm)	Measured Distance (cm)	Error
Front Sensor	20 cm	24 cm	+4 cm
	24 cm	21 cm	-3 cm
Left/Right Sensor	40 cm	46 cm	+6 cm
	37 cm	35 cm	-2 cm
Down Sensor	10 cm	12 cm	+2 cm
	6 cm	7 cm	+1 cm



```
2in, 5cm
3in, 9cm
2in, 7cm
1in, 4cm
3in, 8cm
3in, 9cm
1in, 4cm
2in, 7cm
3in, 9cm
2in, 6cm
2in, 6cm
3in, 9cm
1in, 4cm
2in, 6cm
```

[✓ Autoscroll] Show timestamp   Newline   9600 baud   Clear output

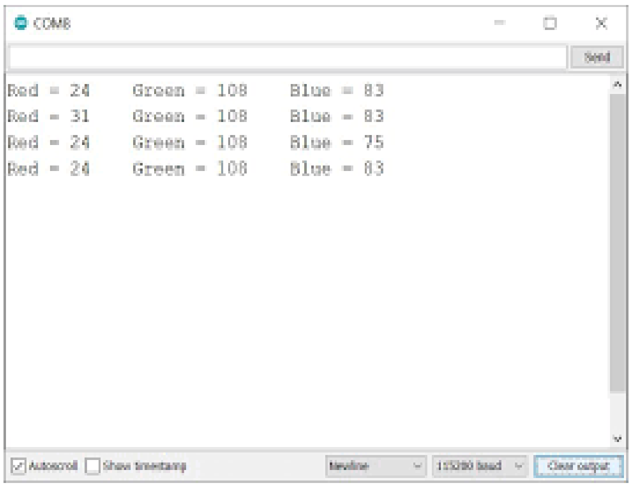
Figure 5.4 Ultrasonic Sensor Output on Serial Monitor

5.1.4 Color Sensor Results

The color sensor was tested for grass detection based on green color reception. The tests were conducted on stable placement as well as on moving mower prototype. The different arrangements slightly affected the results. The output from static position of the sensor was accurate and consistent. Whereas the output from the moving position clearly showed slight uncertainty due to jitters and constant change in lighting conditions and soil makeup initiated from movement, as discussed in Table 5.4. Values obtained from serial monitor are shown in Figure 5.5.

Table 5.4 Color Sensor Results

Sr no.	**Color Value (Red, Green, Blue)**	**Actual Color**	**Detected Color**
1.	144, 218, 175	Green	Green
2.	163, 227, 187	Green	Green
3.	209, 111, 130	Red	Red
4.	109, 149, 237	Blue	Blue



```
Red = 24     Green = 108     Blue = 83
Red = 31     Green = 108     Blue = 83
Red = 24     Green = 108     Blue = 75
Red = 24     Green = 108     Blue = 83
```

Figure 5.5 Color Sensor Output on Serial Monitor

5.1.5 Limitations and Bottlenecks of the Prototype

Upon completion of the project, a few observations have been sighted. These observations show the reasons for hinderance in performance of the final prototype, as depicted in Figure 5.6.

Listed below are the observations.

- Vehicle operates adequately only under certain speeds.
- Weight distribution of prototype is uneven primarily due to battery.
- Restriction in smooth movement due to tractionless tyres.
- Dew and moist conditions affect the operation.
- Battery life may be insufficient for long hours of operation.

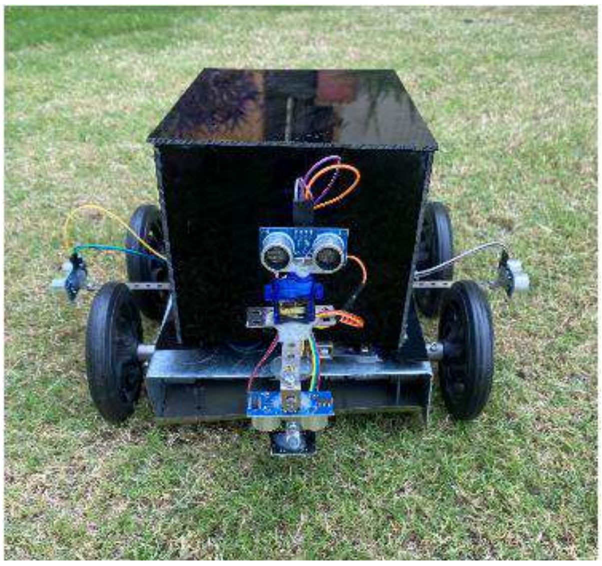


Figure 5.6 Final Working Prototype

5.2 Attainment of Sustainable Development Goals

It is important for us to track the progress regarding the attainment of sustainable development goals with regards to this project in order to serve society as responsible citizens and more importantly as responsible engineers. Following are the tracking details of SDG goals specific to this project.

5.2.1 SDG 9: Industry, Innovation and Infrastructure

The achievement of this project and related ventures of a similar nature draws funding for research and development initiatives to further the technology and effectiveness of autonomous lawn mowers.

It is ensured that the autonomous lawn mower designed in this project has been done so with sustainability in mind. This includes the use of eco-friendly materials such as steel as the main body, optimizing energy efficiency using rechargeable batteries instead of one-time charge and also replacing fuel-based engines and move to electric based motors, and implementing renewable energy sources such as solar power for charging.

5.2.2 SDG 12: Responsible Consumption and Production

This project has been integrated with sustainable design principles into its developments. Using eco-friendly materials, optimize energy efficiency, reduce noise emissions, and ensure ease of repair and recyclability by using electric motors, steel as base.

5.2.3 SDG 13: Climate Action

Throughout this project, it has been considered to implement carbon footprint reduction methods in the operation of autonomous lawn mower. Further step would be to collaborate with environmental organizations to calculate emissions and invest in components that greatly reduce the carbon emissions in order to move towards the goal of climate neutrality.

5.3 Summary

In this chapter, the results obtained from the hardware and software implementations have been discussed and evaluated thoroughly. Hardware results include the ultrasonic sensor results obtained from the various sensors positioned on the front, left and right, and the downward facing sensor for grass length measurement. The results from the grass color sensor. The prototype speed tests are also included, discussing the variation in measurement error due to jitters caused by various motor speeds. This has been conducted in order to choose the best PWM value for effective mowing operation. The results from the trained CNN model for object detection has also been discussed in terms of recall, F1 score and accuracy. The results for the lawn boundary detection using Canny edge detector have also been discussed. Finally, the attainment of sustainable development goals specified in the start of this report have been tracked and the steps taken throughout the duration of the project to achieve these goals have been explained.

Chapter 6

CONCLUSION AND FUTURE WORK

The development of autonomous lawn mowers marks a significant advancement in the field of technology used in lawn maintenance. These cutting-edge devices have completely changed the way we take care of our lawns, providing a wealth of advantages and improving the whole lawn care process.

The time-consuming process of pushing traditional lawn mowers has been successfully removed by autonomous lawn mowers. They can effectively and autonomously move around the lawn using their sophisticated navigation systems and intelligent algorithms, trimming the grass to the correct height and keeping a consistent look. Autonomous lawn mowers provide improved accuracy and precision when mowing grass. These machines can identify obstructions, alter their trajectories, and maneuver around any objects in their path thanks to sensors and artificial intelligence. This reduces the possibility of accidentally damaging trees, flowerbeds, or other yard features while ensuring that the entire grass is evenly trimmed. advantage of autonomous lawn mowers is their eco-friendliness. Many models are electrically powered, reducing the reliance on fossil fuels and minimizing carbon emissions.

It is important to consider their limitations. These machines may struggle with uneven terrains, steep slopes, or complex landscapes, requiring additional human intervention or alternative lawn care methods in certain cases. Additionally, initial setup and programming may require some technical knowledge.

In conclusion, the project offers a successful blend of ultrasonic sensors and color sensing which provides a cheap solution for the achievement of autonomy in lawn mowing. The prototype works best under the ceiling of 39% rated RPM of motors as higher speeds may prove to be hazardous to the onboard components and bring about damage due to collision. From the perspective of battery capacity, the selected battery which is most suiTable for the current design, may prove to be insufficient for lawns of substantial size. In the end, the achievement of autonomy using fusion of sensors provides a widespread low-cost solution especially for struggling economies.

6.1 The Way Forward

To improve the navigation and obstacle avoidance skills of these machines, one component involves the integration of cutting-edge sensors and artificial intelligence systems. Autonomous lawn mowers can identify and avoid obstacles more accurately and efficiently by using more advanced sensor technology, including LiDAR or 3D cameras.

The operating range and runtime of autonomous lawn mowers may be greatly increased thanks to advances in battery technology. These machines could cover bigger areas without stopping if the batteries lasted longer and the charge intervals were shorter, making lawn care more effective and efficient.

Artificial Intelligence may prove to be far more effective and efficient in tedious processes involved in lawn mowing. The use of high torque motors may greatly improve the performance of the prototype due to large weight requirements of onboard components. The use of tyres having better traction such as tank tracks may be inculcated into the design to remove and reduce tyre slipping and ineffective traversal. The inclusion of better computational platforms having greater performance and processing power would greatly help in the execution and swiftness of designed algorithmic logic and input output parameters. To manage the battery using an optimum battery management system would improve the reliability of the prototype and alert the user on the charging requirements for the continued performance of the prototype.

REFERENCES

[1] "Husqvarna Automowers," [Online]. Available: https://www.husqvarna.com/sg /robotic-lawn-mowers/automower-430x/. [Accessed 28 July 2023].

[2] "UNDP Sustainable Development Goals," United Nations Development Programme, [Online]. Available: https://www.undp.org/sustainable-development-goals. [Accessed 4 August 2023].

[3] C. Sciarra, G. Chiarotti, L. Ridolfi and F. Laio, "A Network Approach to Rank Countries Chasing Sustainable Development," Scientific Reports, vol. 11, p. 15441, 2021.

[4] "Hookii : Robotic Lawn Mowers," June 2023. [Online]. Available: https://hookii.com/blogs/robot-lawn-mowers/robotic-lawn-mower-why-they-are-better-for-the-environment. [Accessed 28 July 2023].

[5] B. Landoni, "A Robotic Lawn Mower Powered by Solar Energy," Open Electronics, 2014.

[6] T. Tahir, A. Khalid and J. Arshad, "Implementation of an IoT-Based Solar-Powered Smart Lawn Mower," Wireless Communications and Mobile Computing, vol. 2022, p. 12, 2022.

[7] Y. Shiraishi, H. Zhang and K. Motegi, "AI-Based Approach for Lawn Length Estimation in Robotic Lawn Mowers," in Robotics Software Design and Engineering, 2021.

[8] A. Devasia, "Cartesian Robot," Control Automation, 3 March 2021.

[9] M. Franzius, M. Dunn, et. al, "Embedded Robust Visual Obstacle Detection on Autonomous Lawn Mowers," 7 January 2017. [Online]. Available: https://ieeexplore.ieee.org/document/8014784. [Accessed 11 July 2023].

[10] M. H. Wu, C. W. Chang and J. C. Yu, "Autonomous Boundary Detection Using Image Recognition for Robotic Lawn Mower," in 2022 25th International Conference on Mechatronics Technology (ICMT), Kaohsiung, Taiwan, 2022.

[11] X. Li, J. Chen, Y. Ye, S. Wang and X. Wang, "Fast Semantic Segmentation Model PULNet and Lawn Boundary Detection Method," Journal of Physics: Conference Series, vol. 1828, p. 12036, February 2021.

[12] P. Ratan, "AnalyticsVidhya-What is the Convolutional Neural Network Architecture?," 28 October 2020. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/10/what-is-the-convolutional-neural-network-architecture/. [Accessed 06 August 2023].

[13] T. Werther, "StackOverFlow-Yard Inside Garden Detection-CV2," 18 February 2020. [Online]. Available: https://stackoverflow.com/questions/62886284/yard-inside-garden-detection-cv2. [Accessed 06 August 2023].

[14] W. Lu, B. Xei and Z. Ding, "Edge Detection Algorithm-Based Lung Ultrasound in Evaluation of Efficacy of High-Flow Oxygen Therapy on Critical Lung Injury," Computational and Mathematical Methods in Medicine, vol. 2022, pp. 1-10, 2022.

[15] "Stack Over Flow-Autonomous Lawn Edger OpenCV-Guidance," 07 May 2018. [Online].Available:https://stackoverflow.com/questions/71120238/autonomous-lawn-edger-opencv-guidance. [Accessed 07 December 2022].

[16] "Robotique-Obstacle Detection with Arduino," 24 February 2021. [Online]. Available: https://www.robotique.tech/robotics/obstacle-detection-system-with-arduino/. [Accessed 24 July 2023].

[17] J. Marin, "Autonomous WSN for Lawn Monitoring in Smart Cities," in IEEE Conference, Chicago, 2017.

[18] "Components 101," 18 September 2018. [Online]. Available: https://components101.com/motors/servo-motor-basics-pinout-datasheet. [Accessed 12 June 2023].

[19] A. Pandit, "Circuit Digest - Interfacing Servo Motor with AVR Microcontroller ATmega16," 09 February 2019. [Online]. Available: https://circuitdigest.com/microcontroller-projects/interfacing-servo-motor-with-atmega16-avr-microcontroller. [Accessed 04 August 2023].

[20] "Walmart-Webcams," January 2020. [Online]. Available: https://www.walmart.com/ip/Hi-FANCY-High-definition-1080P-Webcam-ISP-

Processing-Algorithms-3D-Noise-Removal-Smart-Web-Camera-with-Privacy-Cover-Video-Conference/1838611684. [Accessed 03 August 2023].

[21] "ISL Products - Motor Component Database," ISL Products International Ltd., 23 February 2014. [Online]. Available: https://islproducts.com/design-note/dc-motor-dc-gear-motor-basics/. [Accessed 12 June 2023].

[22] "Mootio Components - DC Gear Motors," March 2021. [Online]. Available: https://www.mootio-components.com/gear-motor-dc-12v-3rpm_refe_008073-12. [Accessed 12 June 2023].

[23] "ELPROCUS - Color Sensor," Electronics | Projects | Focus, 09 April 2013. [Online]. Available: https://www.elprocus.com/color-sensor-working-and-applications/. [Accessed 12 June 2023].

[24] "Electronics PRO - TCS3200 Detector Module Color Recognition Sensor," April 2021. [Online]. Available: epro.pk/product/sr078-tcs230-tcs3200-detector-module-color-recognition-sensor-in-pakistan/. [Accessed 12 June 2023].

[25] "OpenSource - What Arduino," 27 March 2021. [Online]. Available: https://opensource.com/resources/what-arduino. [Accessed 12 June 2023].

[26] "TEJAR - Arduino Mega 2560 Rev3 Board," January 2021. [Online]. Available: https://www.tejar.pk/arduino-mega-2560-rev3-board. [Accessed 12 June 2023].

[27] "Maxbotix - How Ultrasonic Sensors Work," 10 June 2005. [Online]. Available: https://maxbotix.com/blogs/blog/how-ultrasonic-sensors-work. [Accessed 20 June 2023].

[28] "Anaconda and Spyder for Windows," Anaconda Installer, 11 July 2023. [Online]. Available: https://docs.anaconda.com/free/anaconda/install/windows/. [Accessed 04 August 2023].

[29] C. Aggarwal, Neural Network and Deep Learning - A textbook, Springer, 2018.

[30] "Domino Data Lab," 12 May 2019. [Online]. Available: https://www.dominodatalab.com/data-science-dictionary/jupyter-notebook. [Accessed 22 June 2023].

[31] "Dataschool.io," 28 March 2019. [Online]. Available: https://www.dataschool.io/cloud-services-for-jupyter-notebook/. [Accessed 12 June 2023].

[32] "Java T Point - Arduino IDE," JavaTPoint, 02 November 2021. [Online]. Available: https://www.javatpoint.com/arduino-ide. [Accessed 22 June 2023].

APPENDICES

Appendix – A1

Python Code for CNN Model Training

```
import numpy as np

import cv2

import os

from sklearn.model_selection
import train_test_split

import matplotlib.pyplot as plt

from keras.preprocessing.image
import ImageDataGenerator

from keras.utils.np_utils import
to_categorical

from keras.models import Sequential

from keras.layers import Dense

from keras.optimizers import Adam

from keras.layers import Dropout,
Flatten

from keras.layers.convolutional import
Conv2D, MaxPooling2D

import pickle

###### Parameters ######

path = 'mydata'

testRatio = 0.2

valRatio = 0.2

imageDimensions = (224, 224, 3)

batchSizeVal = 2

epochVal = 15

stepsPerEpochVal = 2000

### Importing of the Images###

images = []

classNo = []

myList = os.listdir(path)

print("Total Number of classes
Detected", len(myList))

noOfClasses = len(myList)

print("Importing Classes......")

for x in range(0, noOfClasses):

        myPicList = os.listdir(path +
        "/" + str(x))

        for y in myPicList:

        curImg = cv2.imread(path +
        "/" + str(x) + "/" + y)

        curImg = cv2.resize(curImg,
        (imageDimensions[0],
        imageDimensions[1]))

        images.append(curImg)

        classNo.append(x)

        print(x, end=" ")

print(" ")

images = np.array(images)

classNo = np.array(classNo)

print(images.shape)


### Splitting the data###

X_train, X_test, y_train, y_test =
train_test_split(images, classNo,
test_size=testRatio)

X_train, X_validation, y_train,
y_validation =
train_test_split(X_train, y_train,
test_size=valRatio)
```

```
print(X_train.shape)
print(X_test.shape)
print(X_validation.shape)

### DISPLAY A BAR CHART

SHOWING NO OF SAMPLES FOR

EACH CATEGORY###

numOfSamples = []

for x in range(0, noOfClasses):

        #print(len(np.where(y_train==
        x)[0]))

        numOfSamples.append(len(np.
        where(y_train == x)[0]))

print(numOfSamples)

plt.figure(figsize=(10, 5))

plt.bar(range(0, noOfClasses),
numOfSamples)

plt.title("Number of images for each
class")

plt.xlabel('Class ID')

plt.ylabel("Number of images")

plt.show()

##PREPROCESSING THE IMAGES##

def preprocessing(img):

        img = cv2.cvtColor(img,
        cv2.COLOR_BGR2GRAY)

        img = cv2.equalizeHist(img)

        img = img / 255

        return img

X_train = np.array (list(map
(preprocessing, X_train)))

X_test = np.array (list(map
(preprocessing, X_test)))

X_validation = np.array (list(map
(preprocessing, X_validation)))

print(X_train.shape)

### add depth of 1###

X_train = X_train.reshape
(X_train.shape[0], X_train.shape[1],
X_train.shape[2], 1)

X_test = X_test.reshape
(X_test.shape[0], X_test.shape[1],
X_test.shape[2], 1)

X_validation = X_validation.reshape
(X_validation.shape[0],
X_validation.shape[1],
X_validation.shape[2], 1)

## AUGMENTATAION OF IMAGES:

TO MAKE IT MORE GENERIC##

dataGen = ImageDataGenerator
(width_shift_range=0.1,height_shift_r
ange=0.1,zoom_range=0.2,shear_ran
ge=0.1,rotation_range=10)

dataGen.fit(X_train)


y_train = to_categorical(y_train,
noOfClasses)

y_test = to_categorical(y_test,
noOfClasses)

y_validation = to_categorical
(y_validation, noOfClasses)


### CONVOLUTION NEURAL

NETWORK MODEL###

def myModel():

        noOfFilters = 60

        sizeOfFilter1 = (5, 5)

        sizeOfFilter2 = (3, 3)

        sizeOfPool = (2, 2)

        noOfNodes = 500
        model = Sequential()
        model.add((Conv2D(noOfFilters, sizeOfFilter1, input_shape=(imageDimensions[0],imageDimensions[1], 1), activation='relu')))
        model.add((Conv2D(noOfFilters, sizeOfFilter1, activation='relu')))
        model.add(MaxPooling2D(pool_size=sizeOfPool))
        model.add((Conv2D(noOfFilters // 2, sizeOfFilter2, activation='relu')))
        model.add((Conv2D(noOfFilters // 2, sizeOfFilter2, activation='relu')))
        model.add(MaxPooling2D(pool_size=sizeOfPool))
        model.add(Dropout(0.5))
         model.add(Flatten())
        model.add(Dense(noOfNodes, activation='relu'))
        model.add(Dropout(0.5))
        model.add(Dense(noOfClasses, activation='softmax'))
        model.compile(Adam(learning_rate=0.001), loss='categorical_crossentropy',metrics=['accuracy'])
        return model

###### TRAIN######
model = myModel()
print(model.summary())
history = model.fit(dataGen.flow(X_train, y_train,batch_size=batchSizeVal),steps_per_epoch=stepsPerEpochVal,epochs=epochVal, validation_data=(X_validation, y_validation),shuffle=1)

###### PLOT######
plt.figure(1)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.legend(['Training', 'Validation'])
plt.title('Loss')
plt.xlabel('epoch')
plt.figure(2)
plt.plot(history.history['accuracy'])
plt.plot(history.history['valaccuracy'])
plt.legend(['Training', 'Validation'])
plt.title('Accuracy')
plt.xlabel('epoch')
plt.show()
score = model.evaluate(X_test, y_test, verbose=0)
print('Test score =', score[0])
print('Test accuracy', score[1])

###STORE THE MODEL AS A
PICKLE OBJECT###
pickle_out = open("model_trained_16.p", "wb")
pickle.dump(model, pickle_out)
pickle_out.close()
```

Appendix – A2

Python Code for CNN Model Testing

```
import numpy as np
import cv2
import pickle
################################
width=640
height=480
threshold=0.85
font = cv2.FONT_ HERSHEY_
SIMPLEX
################################
cap=cv2.VideoCapture(0)
cap.set(3,width)
cap.set(4,height)
pickle_in=open("model_trained_15.p",
"rb")
model=pickle.load(pickle_in)

def preprocessing(img):
        img=cv2.cvtColor(img,cv2.
        COLOR_BGR2GRAY)
        img=cv2.equalizeHist(img)
        img=img/224
        return img

def getClassName(classNo):
if   classNo == 0: return 'baseball'
elif classNo == 1: return 'basketball'
elif classNo == 2: return 'beachballs'
elif classNo == 3: return 'billiard ball'
elif classNo ==4: return 'bowling ball'
elif classNo == 5: return 'brass'
elif classNo == 6: return 'buckeyballs'
elif classNo == 7: return 'cannon ball'
elif classNo ==8: return 'crochet ball'
elif classNo == 9: return 'cricket ball'
elif classNo ==10: return 'crystal ball'
elif classNo == 11: return 'eyeballs'
elif classNo == 12: return 'football'
elif classNo == 13: return 'golf ball'
elif classNo == 14: return 'marble'
elif classNo == 15: return 'meat ball'
elif classNo==16: return 'medicine
ball'
elif classNo == 17: return 'paint balls'
elif classNo == 18: return 'pokeman
balls'
elif classNo == 19: return 'puffballs'
elif classNo == 20: return 'rubber
band ball'
elif classNo == 21: return 'screwballs'
elif classNo == 22: return 'sepak
takraw ball'
elif classNo == 23: return 'soccer ball'
elif classNo == 24: return 'tennis ball'
elif classNo == 25: return 'tether ball'
elif classNo == 26: return 'volley ball'
elif classNo == 27: return 'water polo
ball'
elif classNo == 28: return 'wiffle ball'
elif classNo == 29: return 'wrecking
ball'
```

```
while True:
        success,imgOriginal =
        cap.read()
        gray = cv2.cvtColor
        (imgOriginal,
        cv2.COLOR_BGR2GRAY)
        edged = cv2.Canny(gray, 30,
        200)
        contours, hierarchy =
        cv2.findContours(edged,
        cv2.RETR_EXTERNAL,
        cv2.CHAIN_APPROX_NONE)
        cv2.drawContours(imgOriginal
        , contours, -1, (0, 255, 0), 3)


img=np.asarray(imgOriginal)
        img=img[200:300,200:300]
        #img=img[20:300,50:1000]
        cv2.imshow("cropped
        video",img)
        img=cv2.resize(img,(224,224))
        img=preprocessing(img)
        img=img.reshape(1,224,224,1)

        #cv2.putText(imgOriginal,
        "CLASS:", (20, 35), font, 0.75,
        (0, 0, 255), 2, cv2.LINE_AA)
        #cv2.putText(imgOriginal,
        "PROBABILITY: ", (20, 75),
        font, 0.75, (0, 0, 255), 2,
        cv2.LINE_AA)
        #predict
        #classIndex=model.predict_
        classes(img)
        classIndex=model.predict(img)
        classes_x=np.argmax(class
        Index,axis=1)
        print(classIndex)
        predictions=model.predict
        (img)
        probVal=np.amax(predictions)
        print(classIndex,probVal)


if probVal>threshold:
        cv2.putText(imgOriginal, str
        (classIndex) + " " + str
        (getClassName(classIndex)),
        (120, 35), font, 0.75,(0, 0, 255),
        2, cv2.LINE_AA)
        cv2.putText(imgOriginal,
        str(round(probVal * 100, 2)) +
        "%", (180, 75), font, 0.75, (0, 0,
        255), 2,cv2.LINE_AA)
cv2.imshow("Original Image",
imgOriginal)
 if cv2.waitKey(1) & 0xFF ==
ord('q'):
        break
```

Appendix – B

Python Code for Lawn Detection Using Canny Edge Detector

```
import cv2

import numpy as np

import time


def detect_grass(image, start_x,

start_y):

   hsv_image = cv2.cvtColor(image,
   cv2.COLOR_BGR2HSV)

   lower_green = np.array([35, 50,
   50])

   upper_green = np.array([85, 255,
   255])

   mask_grass = cv2.inRange
   (hsv_image, lower_green,
   upper_green)

   kernel = np.ones((5, 5), np.uint8)

   mask_grass = cv2.erode (mask_
   grass, kernel, iterations=1)

   mask_grass = cv2.dilate
   (mask_grass, kernel, iterations=1)

   contours, _ = cv2.findContours
   (mask_grass, cv2.RETR_
   EXTERNAL, cv2.CHAIN
   _APPROX_SIMPLE)

   largest_contour = max(contours,
   key=cv2.contourArea)

   mask_grass_patch = np.zeros_like
   (image)

   cv2.drawContours(mask_grass_
   patch, [largest_contour], -1, (255,
   255, 255), thickness=cv2.FILLED)

   result_grass = cv2.bitwise_
   and(image, mask_grass_patch)

#Get the coordinates of the grass patch

   grass_coordinates = largest_
   contour.reshape(-1, 2)

# Iterate through each coordinate

   for (x, y) in grass_coordinates:

# Create a copy of the original image

      image_with_circles =
      image.copy()

# Draw a cross on the image at the
current coordinate
      cross_size = 10

      cv2.line(image, (x-cross_size, y),
     (x+cross_size, y), (0, 0, 255), 2)

      cv2.line(image, (x, y-cross_size),
     (x, y+cross_size), (0, 0, 255), 2)

# Draw a circle on the image at the
current coordinate
      cv2.circle(image, (x, y), 5, (0, 0,

      255), -1)

# Display the image with the circle and

cross

      cv2.imshow('Grass Cutting',
      image_with_circles)

      cv2.waitKey(1)

      time.sleep(0.001)


   return result_grass, image

# Load the image

image = cv2.imread('lawn_image.jpg')


# Copy the original image
```

```
original_image = image.copy()

# Set video output parameters
output_file = 'grass_cutting
_simulation.mp4'
output_fps = 30.0
output_size = (image.shape[1],
image.shape[0])
fourcc = cv2.VideoWriter
_fourcc(*'mp4v')

# Create video writer object
video_writer = cv2.VideoWriter
(output_file, fourcc, output_fps,
output_size)

# Get the starting point with the
minimum x-coordinate

start_idx = np.argmin(image[:, :, 1])
start_x, start_y = np.unravel_index
(start_idx, image[:, :, 1].shape)

# Iterate until no grass is left in the
image

while cv2.countNonZero
(cv2.cvtColor(image,
cv2.COLOR_BGR2GRAY)) > 0:
# Detect grass in the image and get the
result
    grass_image, image = detect_grass
    (image, start_x, start_y)

# Write the grass image to video
    video_writer.write(grass_image)

# Display the grass patch image with
highlighted edges

    cv2.imshow('Grass Patch
    Detection', grass_image)
# Display the image with grass pixels
removed
    cv2.imshow('Grass Cutting', image)

# Wait for a while to slow down the
video playback

    time.sleep(0.1)

# Wait for key press to proceed to the
next iteration

    if cv2.waitKey(1) == ord('q'):
        break

# Release the video writer and close
windows

video_writer.release()
cv2.destroyAllWindows()
```

Appendix – C

C Code for Arduino Mega 2560 for State Machine

```
#define motor_lt_ina 14// These are the L298 pins for the Left Motor.
#define motor_lt_inb 15
#define motor_rt_ina 17//These are the L298 pins for the Right Motor.
#define motor_rt_inb 16

#define motor_lt_en  11// This is the PWM pin of the L298 for the Left Motor.
#define motor_rt_en  10// This is the PWM pin of the L298 for the Right Motor.

#define trig_up      18//These are the trig and echo pins for the up sensor.
#define echo_up      19
#define trig_lt      20//These are the trig and echo pins for the lt sensor.
#define echo_lt      21
#define trig_rt      22//These are the trig and echo pins for the rt sensor.
#define echo_rt      23
#define trig_dn      24//These are the trig and echo pins for the dn sensor.
#define echo_dn      25

#define s0_color     26//These are the trig and echo pins for the color sensor.
#define s1_color     27//s0 and s1 control the frequency of light emitted out of color sensor.
#define s2_color     28//s2 and s3 filter the frequency of light received by the color sensor.
#define s3_color     29
#define color_out    30//This pin sends the color output from sensor to Arduino.

long duration, distance_up, distance_lt, distance_rt, distance_dn; // distance variabled to store the distances measured using ultrasonic sensor.
long distance_lt_old, distance_rt_old;
int cutting_state, grass_cut_cntr;

int data,r,g,b;        //This is where we're going to stock our values
bool red,green,blue,white,black, yellow,cyan,magenta;

void setup()
{
  pinMode(motor_lt_ina, OUTPUT);
  pinMode(motor_lt_inb, OUTPUT);
  pinMode(motor_rt_ina, OUTPUT);
  pinMode(motor_rt_inb, OUTPUT);
  pinMode(motor_lt_en, OUTPUT);
  pinMode(motor_rt_en, OUTPUT);

  pinMode(trig_up, OUTPUT);       // Set Trig Pin As O/P To Transmit Waves
  pinMode(echo_up, INPUT);        //Set Echo Pin As I/P To Receive Reflected Waves
  pinMode(trig_lt, OUTPUT);       // Set Trig Pin As O/P To Transmit Waves
  pinMode(echo_lt, INPUT);        //Set
  Echo Pin As I/P To Receive Reflected
  Waves

  pinMode(trig_rt, OUTPUT);

  pinMode(echo_rt, INPUT);

  pinMode(trig_dn, OUTPUT);

  pinMode(echo_dn, INPUT);


  pinMode(s0_color, OUTPUT);

  pinMode(s1_color, OUTPUT);

  pinMode(s2_color, OUTPUT);

  pinMode(s3_color, OUTPUT);

  pinMode(color_out, INPUT);

    Serial.begin(9600);

    cutting_state = 0;

    grass_cut_cntr = 0;


  digitalWrite(s0_color,HIGH);
  //Putting S0/S1 on HIGH/HIGH
  levels means the output frequency
  scalling is at 100% (recommended)

  digitalWrite(s1_color,HIGH);
  //LOW/LOW is off HIGH/LOW is
  20% and LOW/HIGH is  2%
}

void loop()

{


  digitalWrite(trig_up, LOW);

  delayMicroseconds(2);

  digitalWrite(trig_up, HIGH);
//Transmit Waves For 10us

  delayMicroseconds(10);

  duration = pulseIn(echo_up, HIGH);
// Receive Reflected Waves

  distance_up = duration / 58.2;
  // Get Distance

  Serial.println("distance up");

  Serial.println(distance_up);


  digitalWrite(trig_lt, LOW);

  delayMicroseconds(2);

  digitalWrite(trig_lt, HIGH);
 // Transmit Waves For 10us
  delayMicroseconds(10);

  duration = pulseIn(echo_lt, HIGH);
// Receive Reflected Waves

  distance_lt = duration / 58.2;
// Get Distance

  distance_lt_old = distance_lt;

  Serial.println("distance lt");

  Serial.println(distance_lt);


  digitalWrite(trig_rt, LOW);

  delayMicroseconds(2);

  digitalWrite(trig_rt, HIGH);
 // Transmit Waves For 10us

  delayMicroseconds(10);

  duration = pulseIn(echo_rt, HIGH);
// Receive Reflected Waves

  distance_rt = duration / 58.2;
// Get Distance

  distance_rt_old = distance_rt;

  Serial.println("distance rt");

  Serial.println(distance_rt);


  digitalWrite(trig_dn, LOW);

  delayMicroseconds(2);

  digitalWrite(trig_dn, HIGH);
 // Transmit Waves For 10us

  delayMicroseconds(10);
  duration = pulseIn(echo_dn, HIGH);
// Receive Reflected Waves

  distance_dn = duration / 58.2;
// Get Distance

  Serial.println("distance dn");

  Serial.println(distance_dn);


   digitalWrite(s2_color,LOW);
   //S2/S3 levels define which set of
   photodiodes we are using LOW/LOW
   is for RED LOW/HIGH is for Blue
   and HIGH/HIGH is for green

   digitalWrite(s3_color,LOW);

   Serial.print("Red value= ");

   GetData();
  //Executing GetData function to get
the value

r=data;

Serial.print(r);

   digitalWrite(s2_color,LOW);

   digitalWrite(s3_color,HIGH);

   Serial.print("Blue value= ");

   GetData();

b=data;

Serial.print(b);

   digitalWrite(s2_color,HIGH);

   digitalWrite(s3_color,HIGH);

   Serial.print("Green value= ");

   GetData();

g=data*0.65;

   Serial.print(g);


//   if(r<b&r<g&r>5)   Serial.print("
RED       ");

//else if(r<b&g<b&b>8)
Serial.print("     YELLOW     ");

//else if(b<r&b<g&b>5)
Serial.print("      BLUE      ");
//

//else if(g<r&g<b&g&g>5)
Serial.print("      GREEN     ");
//

//else if(r<7&b<7&g<7)
Serial.print("      WHITE     ");


//else if(r>25&b>25&g>25)
Serial.print("      BLACK     ");

//else Serial.print(" NO color");


  if(distance_dn < 10 &&
  grass_cut_cntr < 4
  &&(g<r&g<b&g&g>5)) //Check
  GREEN
  {

    //GRASS IS CUT ALREADY.

            analogWrite(motor_lt_en,
           100);

            analogWrite(motor_rt_en,
           100);

            digitalWrite(motor_lt_ina,
           HIGH);

            digitalWrite(motor_lt_inb,
           LOW);   //GO STRAIGHT.

            digitalWrite(motor_rt_ina,
           HIGH);

            digitalWrite(motor_rt_inb,
           LOW);

            delay(4000);

            analogWrite(motor_lt_en, 0);

            analogWrite(motor_rt_en, 0);

            digitalWrite(motor_lt_ina,
           LOW);

            digitalWrite(motor_lt_inb,
           LOW);//STOP

            digitalWrite(motor_rt_ina,
           LOW);
```

```
            digitalWrite(motor_rt_inb, LOW);
            delay(1000);

            analogWrite(motor_lt_en, 200);
            analogWrite(motor_rt_en, 200);
            digitalWrite(motor_lt_ina, HIGH); //TURN LEFT
            digitalWrite(motor_lt_inb, LOW);
            digitalWrite(motor_rt_ina, LOW);
            digitalWrite(motor_rt_inb, HIGH);
            delay(800);
            grass_cut_cntr++;
}
else if((g<r&g<b&g&g>5)){
switch(cutting_state)
{
  case 0 : //IDLE
            analogWrite(motor_lt_en, 100);
            analogWrite(motor_rt_en, 100);
            digitalWrite(motor_lt_ina, HIGH);
            digitalWrite(motor_lt_inb, LOW);
            digitalWrite(motor_rt_ina, HIGH);
            digitalWrite(motor_rt_inb, LOW);
            delay(1000);
            if(distance_up < 40 && distance_up != 0)
{//STOP
            analogWrite(motor_lt_en, 0);
            analogWrite(motor_rt_en, 0);

            digitalWrite(motor_lt_ina, LOW);
            digitalWrite(motor_lt_inb, LOW);
            digitalWrite(motor_rt_ina, LOW);
            digitalWrite(motor_rt_inb, LOW);
            delay(1000);

            cutting_state = 1;
            }
            break;

    case 1 : //TAKE PROXIMITY READINGs & Re-Orient the LAWN MOWER.
            if(distance_lt < 10 && distance_lt != 0)
            {
            //TURN SHARP RIGHT (90 Degrees) .
            analogWrite(motor_lt_en, 200);
            analogWrite(motor_rt_en, 200);
            digitalWrite(motor_lt_ina, HIGH);
            digitalWrite(motor_lt_inb, LOW);
            digitalWrite(motor_rt_ina, LOW);
            digitalWrite(motor_rt_inb, HIGH);
```

```
 delay(500);
 //STOP
 analogWrite(motor_lt_en, 0);
 analogWrite(motor_rt_en, 0);

digitalWrite(motor_lt_ina,
LOW);
 digitalWrite(motor_lt_inb,
LOW);
 digitalWrite(motor_rt_ina,
LOW);
 digitalWrite(motor_rt_inb,
LOW);
 delay(800);
 cutting_state = 2;
 }
else if(distance_rt < 10 &&
distance_rt != 0)
 {
 //TURN SHARP LEFT (90
Degrees)
 analogWrite(motor_lt_en,
200);
 analogWrite(motor_rt_en,
200);
 digitalWrite(motor_lt_ina,
LOW);
 digitalWrite(motor_lt_inb,
HIGH);
 digitalWrite(motor_rt_ina,
HIGH);
 digitalWrite(motor_rt_inb,
LOW);
 delay(500);
 //STOP
 analogWrite(motor_lt_en, 0);
 analogWrite(motor_rt_en, 0);

digitalWrite(motor_lt_ina,
LOW);
 digitalWrite(motor_lt_inb,
LOW);
 digitalWrite(motor_rt_ina,
LOW);
 digitalWrite(motor_rt_inb,
LOW);
 delay(1000);
 cutting_state = 2;
 }
 else
 {
  if(distance_lt > distance_rt)
  {
 //TURN SLIGHTLY LEFT

analogWrite(motor_lt_en,
100);
 analogWrite(motor_rt_en,
200);
 digitalWrite(motor_lt_ina,
HIGH);
 digitalWrite(motor_lt_inb,
LOW);
 digitalWrite(motor_rt_ina,
HIGH);
 digitalWrite(motor_rt_inb,
LOW);
 delay(800);
 analogWrite(motor_lt_en, 0);
 analogWrite(motor_rt_en, 0);
digitalWrite(motor_lt_ina,
LOW);
digitalWrite(motor_lt_inb,
LOW);
digitalWrite(motor_rt_ina,
LOW);
 digitalWrite(motor_rt_inb,
LOW);
 delay(500);
  cutting_state  =  2;
  }
else if(distance_rt >
distance_lt)
  {
 //TURN SLIGHTLY RIGHT
analogWrite(motor_lt_en,
200);
 analogWrite(motor_rt_en,
100);
 digitalWrite(motor_lt_ina,
HIGH);
 digitalWrite(motor_lt_inb,
LOW);
 digitalWrite(motor_rt_ina,
HIGH);
 digitalWrite(motor_rt_inb,
LOW);
 delay(800);
 analogWrite(motor_lt_en, 0);
 analogWrite(motor_rt_en, 0);
 digitalWrite(motor_lt_ina,
LOW);
 digitalWrite(motor_lt_inb,
LOW);
 digitalWrite(motor_rt_ina,
LOW);
 digitalWrite(motor_rt_inb,
LOW);
 delay(500);
  cutting_state  =  2;
  }
 }
```

```
          break;
    case 2 : //START CUTTING
          analogWrite(motor_lt_en,
         100);
          analogWrite(motor_rt_en,
         100);
          digitalWrite(motor_lt_ina,
         HIGH);
          digitalWrite(motor_lt_inb,
         LOW);
          digitalWrite(motor_rt_ina,
         HIGH);
          digitalWrite(motor_rt_inb,
         LOW);
          delay(1000);
          cutting_state = 3;
          break;
    case 3 :
//CONDUCT PROXIMITY CHECKS
DURING CUTTING.
    if((distance_up < 40 &&
    distance_up != 0 ))
    {
          //STOP
          analogWrite(motor_lt_en, 0);
          analogWrite(motor_rt_en, 0);
          digitalWrite(motor_lt_ina,
         LOW);
          digitalWrite(motor_lt_inb,
         LOW);
          digitalWrite(motor_rt_ina,
         LOW);
          digitalWrite(motor_rt_inb,
         LOW);
          delay(1000);
          cutting_state = 1;
```

```
}
 if(distance_lt > 20 && distance_lt
 < 80  && distance_rt > 100)
 {
           //slight left
      analogWrite(motor_lt_en,
     100);
      analogWrite(motor_rt_en,
     200);
      digitalWrite(motor_lt_ina,
     HIGH);
      digitalWrite(motor_lt_inb,
     LOW);
      digitalWrite(motor_rt_ina,
     HIGH);
      digitalWrite(motor_rt_inb,
     LOW);
      delay(500);
 }
      if(distance_lt < 20 &&
     distance_lt != 0 &&
     distance_rt > 100)
 {
     analogWrite(motor_lt_en,
     200);
      analogWrite(motor_rt_en,
     100);
      digitalWrite(motor_lt_ina,
     HIGH);
      digitalWrite(motor_lt_inb,
     LOW);
      digitalWrite(motor_rt_ina,
     HIGH);
      digitalWrite(motor_rt_inb,
     LOW);
      delay(800);
     analogWrite(motor_lt_en,
     200);
      analogWrite(motor_rt_en,
     100);
      delay(500);
 }
     else if(distance_rt < 30 &&
     distance_rt > 10 &&
     distance_lt > 100)
 {
      analogWrite(motor_lt_en,
     100);
      analogWrite(motor_rt_en,
     200);
      digitalWrite(motor_lt_ina,
     HIGH);
      digitalWrite(motor_lt_inb,
     LOW);
      digitalWrite(motor_rt_ina,
     HIGH);
      digitalWrite(motor_rt_inb,
     LOW);
      delay(500);
 }
     if(distance_rt > 30 &&
     distance_rt < 50 &&
     distance_lt > 100)
 {
      analogWrite(motor_lt_en,
     200);
      analogWrite(motor_rt_en,
     100);
      digitalWrite(motor_lt_ina,
     HIGH);
      digitalWrite(motor_lt_inb,
     LOW);
      digitalWrite(motor_rt_ina,
     HIGH);
      digitalWrite(motor_rt_inb,
     LOW);
      delay(300);
```

```
    }
        else
         {
         digitalWrite(trig_lt , LOW);
         delayMicroseconds(2);
         digitalWrite(trig_lt, HIGH);
// Transmit Waves For 10us
         delayMicroseconds(10);
        duration = pulseIn(echo_lt,
        HIGH);        // Receive
        Reflected Waves
         distance_lt = duration / 58.2;
// Get Distance
           digitalWrite(trig_rt, LOW);
           delayMicroseconds(2);
           digitalWrite(trig_rt, HIGH);
// Transmit Waves For 10us
           delayMicroseconds(10);
           duration = pulseIn(echo_rt,
           HIGH);        // Receive
           Reflected Waves
           distance_rt = duration / 58.2;
           // Get Distance

           if(distance_lt_old< 70 &&
           (distance_lt_old > 40 &&
           distance_lt_old != 0 ) &&
           distance_lt_old > distance_lt
           && (distance_lt != 0))
          {
         //TURN SLIGHTLY RIGHT
        analogWrite(motor_lt_en,
        200);
         analogWrite(motor_rt_en,
        100);
         digitalWrite(motor_lt_ina,
        HIGH);
 digitalWrite(motor_lt_inb,
LOW);
 digitalWrite(motor_rt_ina,
HIGH);
 digitalWrite(motor_rt_inb,
LOW);
 delay(800);
  cutting_state  =  2;
 }
else if(distance_lt_old < 70
&& (distance_lt_old > 40 &&
distance_lt_old != 0) &&
distance_lt_old < distance_lt
&& (distance_lt < 80))
 {
//TURN SLIGHT LEFT
 analogWrite(motor_lt_en,
100);
 analogWrite(motor_rt_en,
200);
 digitalWrite(motor_lt_ina,
HIGH);
 digitalWrite(motor_lt_inb,
LOW);
 digitalWrite(motor_rt_ina,
HIGH);
 digitalWrite(motor_rt_inb,
LOW);
 delay(800);
 cutting_state = 2;
 }
else if(distance_rt_old < 70
&& (distance_rt_old > 40 &&
distance_rt_old != 0) &&
distance_rt_old > distance_rt
&&  (distance_rt != 0))
 {
 //TURN SLIGHTLY LEFT
analogWrite(motor_lt_en,
100);
 analogWrite(motor_rt_en,
200);
 digitalWrite(motor_lt_ina,
HIGH);
 digitalWrite(motor_lt_inb,
LOW);
 digitalWrite(motor_rt_ina,
HIGH);
 digitalWrite(motor_rt_inb,
LOW);
 delay(800);
 cutting_state = 2;


 }
else if(distance_rt_old < 70
&& (distance_rt_old > 40 &&
distance_rt_old != 0) &&
distance_rt_old < distance_rt
&& (distance_rt < 80))
 {
 //TURN SLIGHTLY RIGHT
 analogWrite(motor_lt_en,
200);
 analogWrite(motor_rt_en,
100);
 digitalWrite(motor_lt_ina,
HIGH);
 digitalWrite(motor_lt_inb,
LOW);
 digitalWrite(motor_rt_ina,
HIGH);
 digitalWrite(motor_rt_inb,
LOW);
 delay(800);
  cutting_state  =  2;

 }
          else {cutting_state = 3;}
          }
          break;
    default :
          break;
  }
  }
}
void GetData(){
   data=pulseIn(color_out,LOW);
   //here we wait until "out" go LOW,
   we start measuring the duration and
   stops when "out" is HIGH again

  //The higher the frequency the lower
  the duration

   delay(20);
}
```