School of Computer Science and Engineering

(Computer Science & Engineering)

Faculty of Engineering & Technology

Jain Global Campus, Kanakapura Taluk - 562112
Ramanagara District, Karnataka, India

2023-2024
(VIII Semester)

A Project Report on

"AUTONOMOUS NAVIGATION: OPTIMAL PATH FINDING AND OBSTACLE AVOIDANCE SURVAILANCE VEHICLE"

Submitted in partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

(ARTIFICIAL INTELLIGENCE)

Submitted by

JAKEY MOZAHID, ATIQUR RAHMAN
20BTRCA066,20BTRCA065

Under the guidance of

Dr Gowri Shankar J
Department of Computer Science and Engineering
School of Computer Science & Engineering
Faculty of Engineering & Technology
JAIN (Deemed to-be University)

CERTIFICATE

This is to certify that the project work titled **“AUTONOMOUS NAVIGATION: OPTIMAL PATH FINDING AND OBSTACLE AVOIDANCE SURVAILANCE VEHICLE”** is carried out by **JAKEY MOZAHID (20BTRCA066) & ATIQUIR RAHMAN (20BTRCA065)** a bonafide student(s) of Bachelor of Technology at the School of Engineering & Technology(Artificial Intelligence), Faculty of Engineering & Technology, JAIN (Deemed-to-be University), Bangalore in partial fulfillment for the award of degree in Bachelor / Master of Technology in Computer Science and Engineering, during the year **2023-2024**.

Dr Gowri Shankar J
Project Guide
Computer Science and Engineering(Artificial Intelligence)
School of Computer Science & Engineering
Faculty of Engineering & Technology
JAIN (Deemed to-be University)

Dr. Rajesh A
Program Head,
Computer Science and Engineering(Artificial Intelligence)
School of Computer Science & Engineering
Faculty of Engineering & Technology
JAIN (Deemed to-be University)

Dr. Geetha G
Director,
School of Computer Science & Engineering
Faculty of Engineering & Technology
JAIN (Deemed to-be University)

Date:

Name of the Examiner	Signature fo Examiner
1.	
2.	

DECLARATION

We , **JAKEY MOZAHID (20BTRCA066) & ATIQUR RAHMAN (20BTRCA065)** student of 8th semester B.Tech in **Computer Science and Engineering**, at School of Engineering & Technology, Faculty of Engineering & Technology, **JAIN (Deemed to-be University)**, hereby declare that the internship work titled **"AUTONOMOOUS NAVIGATION: OPTIMAL PATH FINDING AND OBSTACLE AVOIDANCE SURVAILANCE VEHICLE"** has been carried out by us and submitted in partial fulfilment for the award of degree in **Bachelor of Technology in Computer Science and Engineering(Artificial Intelligence)** during the academic year **2023-2024**. Further, the matter presented in the work has not been submitted previously by anybody for the award of any degree or any diploma to any other University, to the best of our knowledge and faith.

JAKEY MOZAHID	Signature
USN : **20BTRCA066**	
ATIQUR RAHMAN	Signature
USN : **20BTRCA065**	

Place : Bangalore

Date :

ACKNOWLEDGEMENT

It is a great pleasure for me to acknowledge the assistance and support of a large number of individuals who have been responsible for the successful completion of this project work.

First, I take this opportunity to express my sincere gratitude to Faculty of Engineering & Technology, JAIN (Deemed to-be University) for providing me with a great opportunity to pursue my Bachelors / Master’s Degree in this institution.

I am deeply thankful to several individuals whose invaluable contributions have made this project a reality. I wish to extend my heartfelt gratitude to **Dr. Chandraj Roy Chand, Chancellor**, for his tireless commitment to fostering excellence in teaching and research at Jain (Deemed-to-be-University). I am also profoundly grateful to the honorable **Vice Chancellor, Dr. Raj Singh, and Dr. Dinesh Nilkant, Pro Vice Chancellor**, for their unwavering support. Furthermore, I would like to express my sincere thanks to **Dr. Jitendra Kumar Mishra, Registrar**, whose guidance has imparted invaluable qualities and skills that will serve us well in our future endeavors

I extend my sincere gratitude to **Dr. Hariprasad S A, Director** of the Faculty of Engineering & Technology, **and Dr. Geetha G, Director** of the School of Computer Science & Engineering within the Faculty of Engineering & Technology, for their constant encouragement and expert advice. Additionally, I would like to express my appreciation to **Dr. Krishnan Batri, Deputy Director (Course and Delivery), and Dr. V. Vivek, Deputy Director (Students & Industry Relations),** for their invaluable contributions and support throughout this project.

It is a matter of immense pleasure to express my sincere thanks to **Dr. Program head, Program Head, Computer Science and Engineering**, School of Computer Science & Engineering Faculty of Engineering & Technology for providing right academic guidance that made my task possible.

I would like to thank our guide **Dr Gowri Shankar J**, Associate ,**Assistant Professor**, **Dept. of Computer Science and Engineering**, for sparing his/her valuable time to extend help in every step of my work, which paved the way for smooth progress and fruitful culmination of the project.

I would like to thank our Project Coordinator **Mrs. Sunena Rose M V, Assistant Professor,**
Dept. of Computer Science and Engineering (Artificial Intelligence) for their support.

I am also grateful to my family and friends who provided me with every requirement throughout the course.

Signature of Student(s

TABLE OF CONTENTS

ABSTRACT

This project describes the development of a low-cost, autonomous vehicle capable of obstacle avoidance and real-time facial recognition for surveillance applications. The vehicle utilizes ultrasonic sensors to detect and navigate around obstacles in its path, ensuring safe and efficient operation. Additionally, an ESP32-CAM module integrates a camera with Wi-Fi connectivity, providing a live video feed for remote monitoring. The facial recognition system, implemented on the ESP32 microcontroller, analyzes the video stream and identifies pre-programmed faces.

The project emphasizes cost-effectiveness by employing readily available components. Ultrasonic sensors offer a reliable and affordable solution for obstacle detection compared to more expensive LiDAR or radar systems. The ESP32-CAM combines camera functionality with Wi-Fi capabilities, simplifying video transmission and processing within the vehicle itself.

The core functionalities of the vehicle involve:

1. **Obstacle Detection:** Ultrasonic sensors are strategically positioned on the vehicle to emit sound waves and measure the reflected echo, determining the distance to nearby obstacles. The control system processes this information and implements path planning algorithms to avoid collisions.
2. **Live Video Streaming:** The ESP32-CAM captures a real-time video stream of the vehicle's surroundings. This video feed can be accessed remotely through a Wi-Fi connection, allowing for continuous monitoring of the vehicle's path and environment.
3. **Facial Recognition:** Facial recognition software is implemented on the ESP32 microcontroller. The software analyzes the live video stream, identifying pre-programmed faces within the frame. This feature enables the vehicle to potentially recognize authorized personnel or detect individuals of interest within the surveillance area.

The project culminates in the assembly and testing of the autonomous vehicle. The performance of the obstacle avoidance system is evaluated in various scenarios. Additionally, the effectiveness of the facial recognition software is assessed with different lighting conditions and face orientations. Finally, the project discusses potential applications for this autonomous surveillance vehicle in controlled environments, highlighting its advantages and limitations.

This project contributes to the development of cost-effective autonomous vehicles with integrated surveillance capabilities. By combining readily available components with open-source software, the project paves the way for further exploration of real-world applications in security and monitoring domains

LIST OF FIGURES

CHAPTER - 1
INTRODUCTION

Scientists and researchers are working hard to bring human life into a more comfortable zone as the world progresses. People all across the world are becoming excited about the arrival of self-driving automobiles. One of the most important applications of artificial intelligence is autonomous vehicles (AV) (AI). To put it another way, AI is the most crucial component of self-driving and connected automobiles. This is because the car generates a massive amount of environmental data. To perceive the environment, grasp its conditions, and make driving-related judgments, AVs rely on AI. Autonomous vehicles use a variety of cameras, sensors, radars, communication systems, and AI technologies to generate huge amounts of data that can be analyzed in fractions of a second, allowing them to see, hear, think, and make decisions in the same way that human drivers do. It navigates itself to the destination location specified by the user. Indeed, the robotics revolution is making a significant contribution to making the world a safer place. On a technological level, this car is built using several engineering disciplines such as electrical, mechanical, computer sciences, and control engineering, among others.

In addition, an electric car powered by a solar photovoltaic system offers a cost-effective and emission-free mode of transportation. EVs continue to gain market share in comparison to traditional internal combustion engines, and there is a growing demand for fast and efficient charging of electric car batteries. It promises to deliver a more efficient, less expensive, and environmentally friendly manner of powering homes and cars by allowing the EV charging process to be seamlessly integrated. A PV system ranging from 2 kW to 14 kW is required to charge an electric vehicle from a full day's solar production in an un-shaded location. One of the biggest concerns about EV adoption is the amount of time it takes to charge a car's battery. By integrating instantaneous solar generation with AC electricity from the grid at the same time, the Solar Edge organization has developed and invented an Inverter-Integrated EV Charger that enables faster charging than ever before. As a result of the boost mode operation, a full charge can be obtained at a rate 6x faster than the regular Level. When atypical Level A EV charger adds 5 miles per hour of charging, Solar Edge Corporation's solar boost technology adds between 25 and 30 miles in the same amount of time.

1.1 IDEA FORMATION

1.1.1 OVERVIEW

The solar power autonomous vehicle is a self-driving vehicle that is capable of session and navigating its environment without human intervention. Cyber security is to avoid the thefts of vehicles and help in securing the data in vehicles and help in securing the data in vehicles.

- This prototype will work following the line follower
- This prototype will also work through the navigation system
- Here we are using an obstacle detector sensor to see the surrounding obstacle to avoid various unwanted accidents.
- The power section plays a very important role in this Autonomous Vehicle. For this, we will use renewable energy as a charging system.
- As a renewable charging system here we will charge through PV Power and use the hybrid charging station
- We will also use the highest-density aluminum-air battery as backup power in our power system.

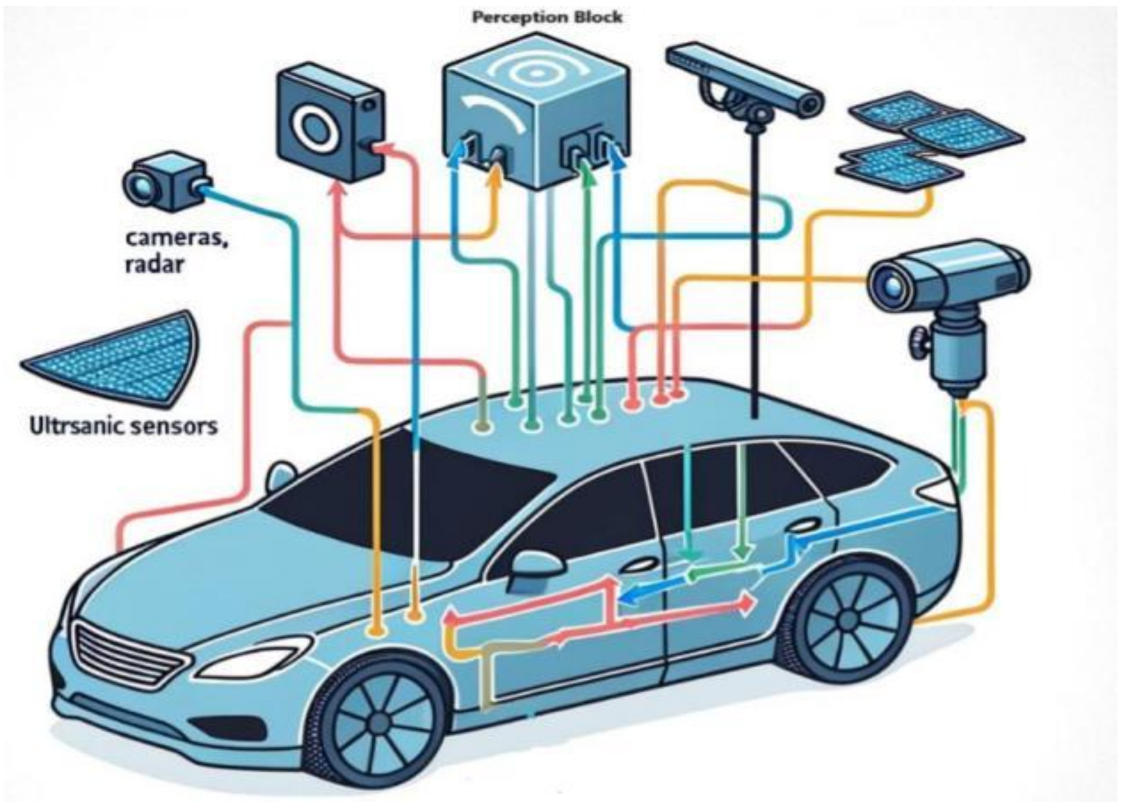


Fig. 1 Prototype of Autonomous system visual picture.

1.1.2 JUSTIFICATION

1.1.2.1 LINE FOLLOWING

Line Following is one of the most important aspects of robotics. A-Line Following Vehicle is an autonomous vehicle that is able to follow either a black or white line that is drawn on a surface consisting of a contrasting color. It is designed to move automatically and follow the made plotline. The Vehicle uses several sensors to identify the line thus assisting the Vehicle to stay on the track. The array of four sensors makes its movement precise and flexible. The Vehicle is driven by DC gear motors to control the movement of the wheels. The Arduino Uno interface is used to perform and implement algorithms to control the speed of the motors, steering the vehicle to travel along the line smoothly. This project aims to implement the algorithm and control the movement of their robot by proper tuning of the control parameters and thus achieve better performance. In addition, the LCD interface is added in order to display the distance traveled by the vehicle. It can be used for industrial automated equipment carriers, small household applications, tour guides in museums, and other similar applications, etc.

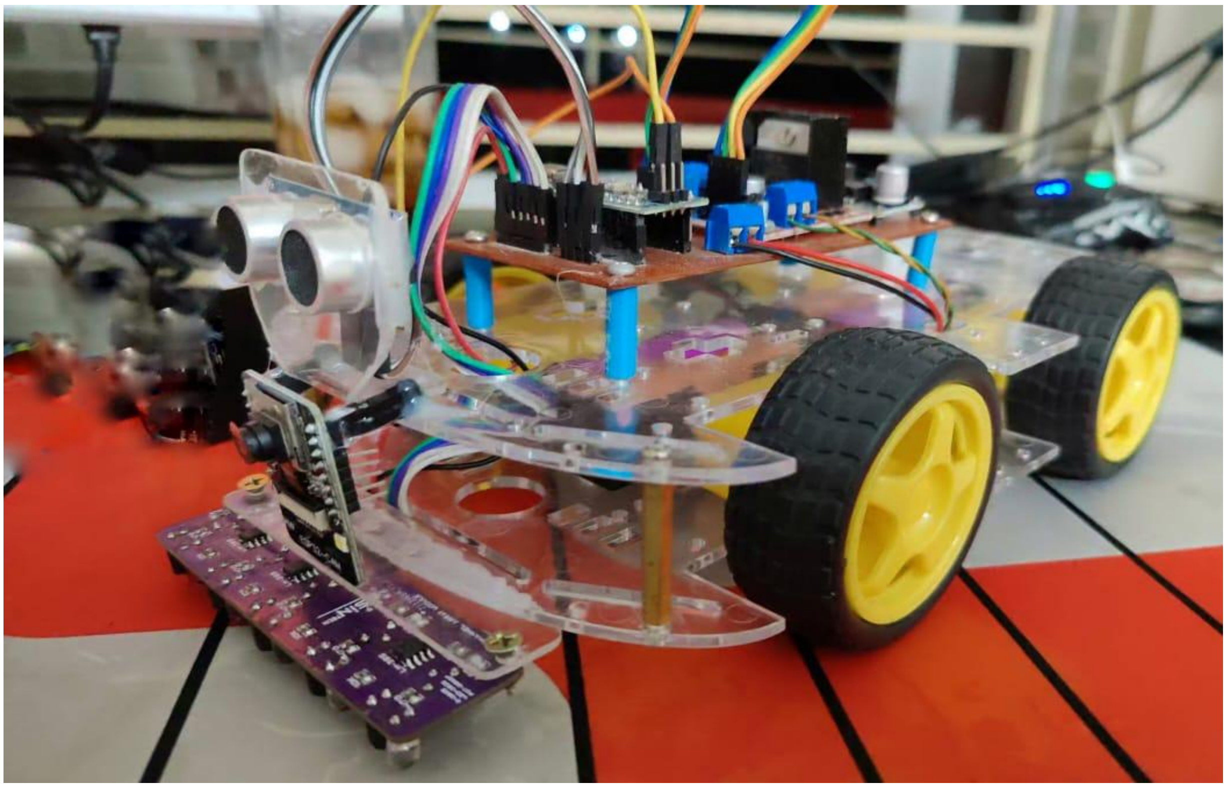


Fig. 2 Autonomous line-following vehicle.

1.1.2.2 OBSTACLE AVOIDING SYSTEM

Obstacle avoidance, in robotics, is a critical aspect of autonomous navigation and control systems. It is the capability of a robot or an autonomous system/machine to detect and circumvent obstacles in its path to reach a predefined destination. This technology plays a pivotal role in various fields, including industrial automation, self-driving cars, drones, and even space exploration. Obstacle avoidance enables robots to operate safely and efficiently in dynamic and complex environments, reducing the risk of collisions and damage. For a robot or autonomous system to successfully navigate through obstacles, it must be able to detect such obstacles. This is most commonly done through the use of sensors, which allow the robot to process its environment, make a decision on what it must do to avoid an obstacle, and carry out that decision with the use of its effectors, or tools that allow a robot to interact with its environment.

Here's a basic overview of how an obstacle-avoiding system typically works:

- Sensor Input
- Environmental Perception
- Obstacle Detection
- Path Planning
- Control and Actuation
- Feedback and Adjustment

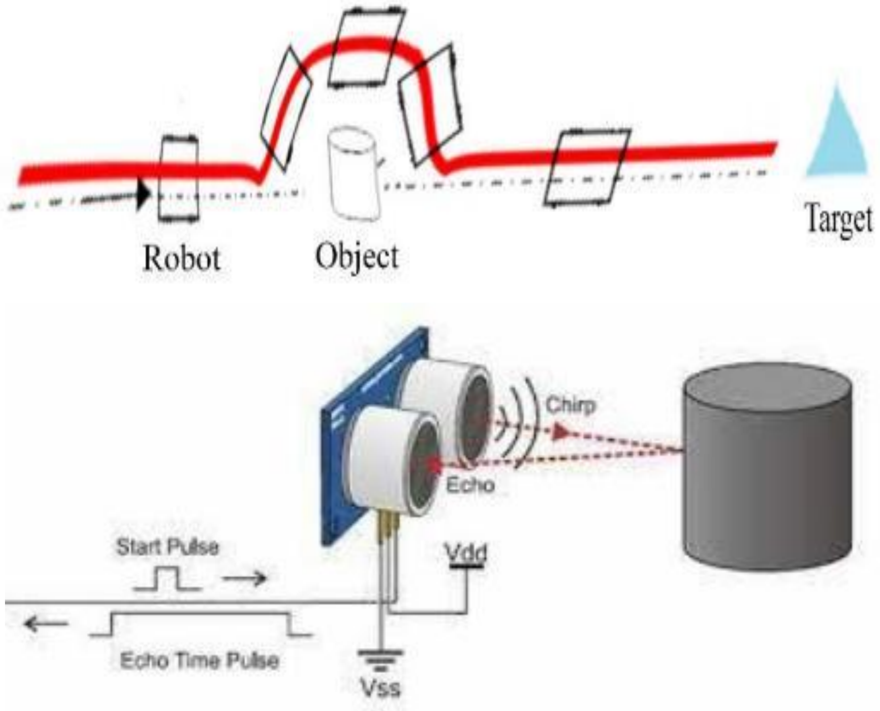
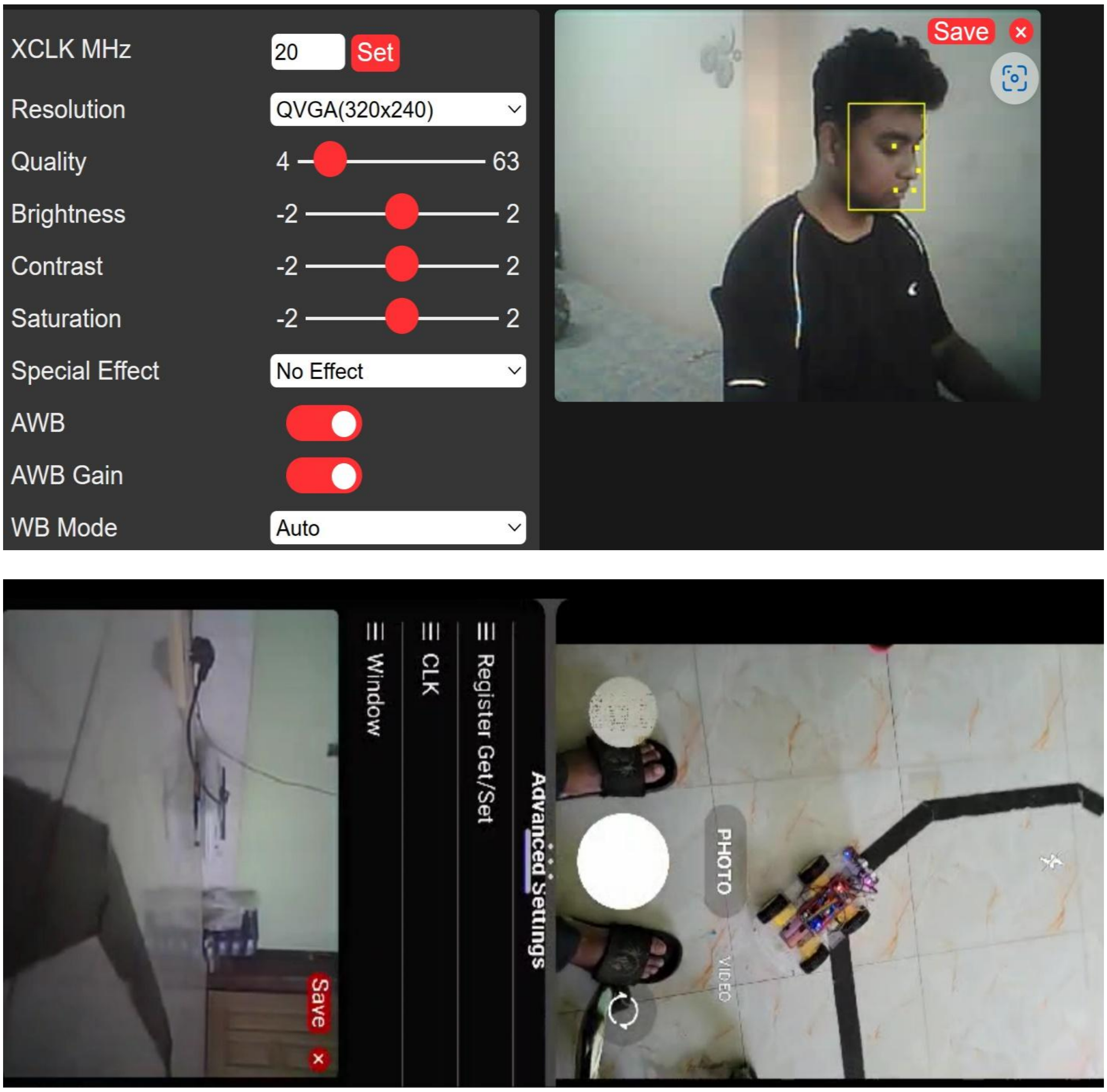


Fig. 3 Obstacle-avoiding system

1.1.2.3 SURVAILANCE AND FACE DETECTION

The ESP32-CAM combines Wi-Fi, a microcontroller, and a camera module into a single board, making it a cost-effective option for DIY surveillance projects. With its built-in camera, you can capture live video feeds for monitoring a location remotely. Furthermore, the ESP32 can be programmed to perform facial recognition on captured frames. This allows you to identify known faces within the live feed, potentially triggering alerts or other actions depending on your setup. However, it's important to note that facial recognition on the ESP32 might have limitations in accuracy and processing power compared to dedicated systems. The ESP32-CAM's small size and low power consumption make it ideal for discreet surveillance applications. You can set it up to stream live video to your phone or computer, allowing you to keep an eye on things from anywhere with an internet connection. The facial recognition capabilities add another layer of functionality.



XCLK MHz	20 Set
Resolution	QVGA(320x240)
Quality	4 — 63
Brightness	-2 — 2
Contrast	-2 — 2
Saturation	-2 — 2
Special Effect	No Effect
AWB	●
AWB Gain	●
WB Mode	Auto

Fig. 4 ESP32-CAM Feed

1.1.2.4 USE RENEWABLE ENERGY SOURCES

Electric vehicles can be charged from Renewable Energy sources and actually help in contributing to grid stability by storing wind power at night and solar power during the day and in turn help in the proliferation of Renewable Energy and contribute to reducing GHG emissions. In the sections provided below, we will understand how energy stored in electric vehicle batteries can provide value to its owner, and areas where you can evolve your business by utilizing power from electric vehicle batteries.

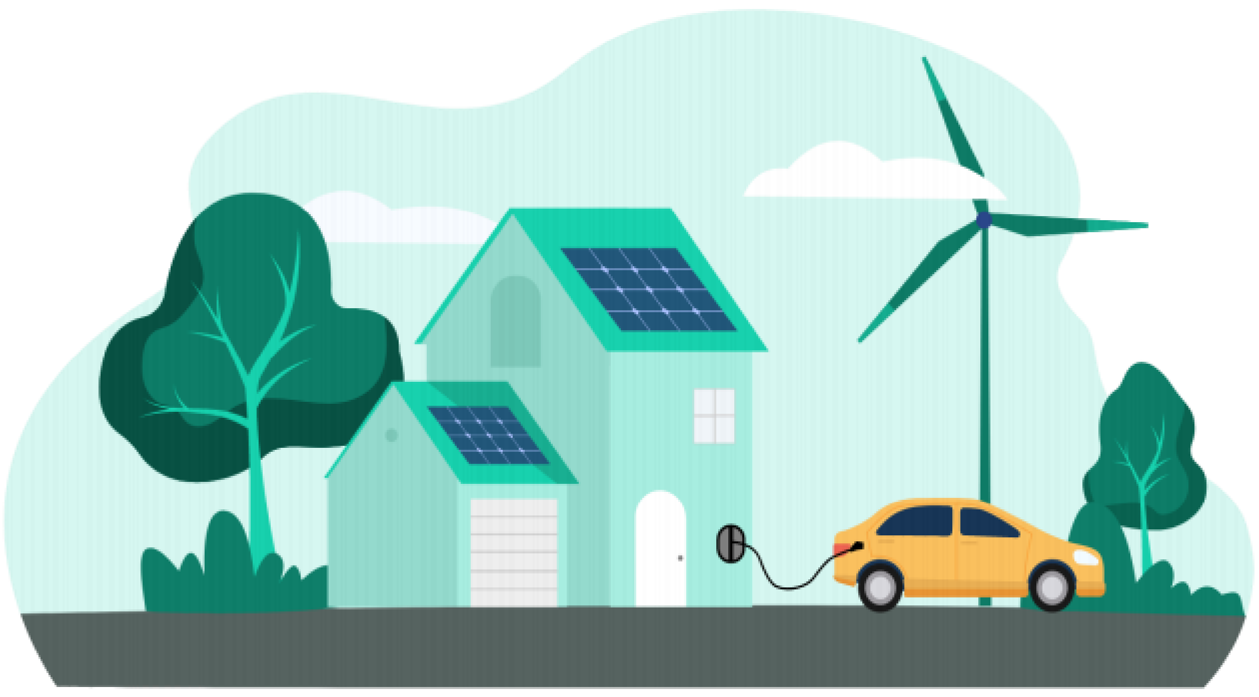


Fig. 5 Illustration of Solar PV-based Vehicle Charging Station.

CHAPTER - 2

LITERATURE REVIEW / MARKET SURVEY

2.1 General

A market survey is the survey research and analysis of the market for a particular product/service which includes the investigation into customer inclinations. A study of various customer capabilities such as investment attributes and buying potential. Market surveys are tools to directly collect feedback from the target audience to understand their characteristics, expectations, and requirements. Marketers develop new and exciting strategies for upcoming products/services but there can be no assurance about the success of these strategies. For these to be successful, marketers should determine the category and features of products/services that the target audiences will readily accept. By doing so, the success of a new avenue can be assured. Most marketing managers depend on market surveys to collect information that would catalyze the market research process. Also, the feedback received from these surveys can be contributory to product marketing and feature enhancement. Market surveys collect data about a target market such as pricing trends, customer requirements, competitor analysis, and other details.

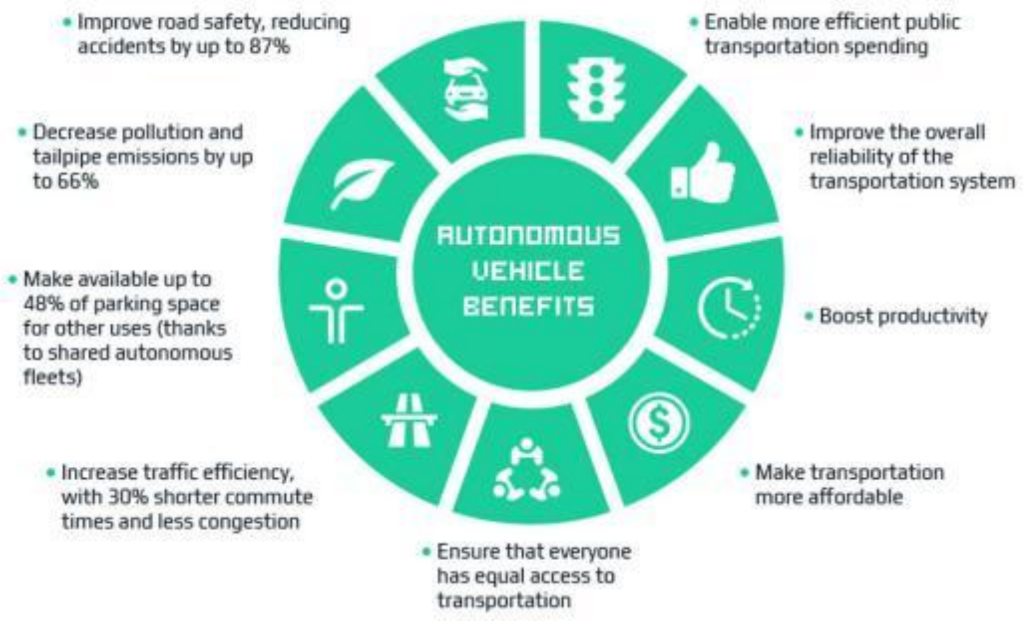

AUTONOMOUS VEHICLE BENEFITS
• Improve road safety, reducing accidents by up to 87%
• Enable more efficient public transportation spending
• Decrease pollution and tailpipe emissions by up to 66%
• Improve the overall reliability of the transportation system
• Make available up to 48% of parking space for other uses (thanks to shared autonomous fleets)
• Boost productivity
• Increase traffic efficiency, with 30% shorter commute times and less congestion
• Make transportation more affordable
• Ensure that everyone has equal access to transportation

Fig. 6 A market survey method.

2.2 Investigating impacts on photovoltaic cells and panels

This paper includes an experimental examination of increasing the thermal, electrical, and yield of a traditional solar still (CSS), Solar cells produce electricity using semiconductors and sunshine. They resemble memory chips for processing computers. Since silicon is the primary component of solar cells, sunlight is to blame. absorbs radiation photons. When exposed to direct sunlight, a non-mechanical device that converts electricity is known as a photovoltaic cell, also known as a solar cell. Photovoltaic electricity from the sun is anon-modern technology that is regarded as the most effective and the most widely utilized in solar energy technologies. Solar photovoltaic energy is based on the use of sunlight without the use of heat. In the creation of energy, heat is a negative factor. The lesser the manufacturing efficiency, the higher the temperature of the unit. The ideal temperature for solar PV panel electricity production is 25 °C, while in some circumstances; high-intensity radiation with a high degree affects the PV panel's efficiency. Photovoltaic solar panels consist of many solar cells; these cells are made of semiconductors such as silicon and are designed in two layers, a positive layer and a negative layer, which is what is known as the electric field.

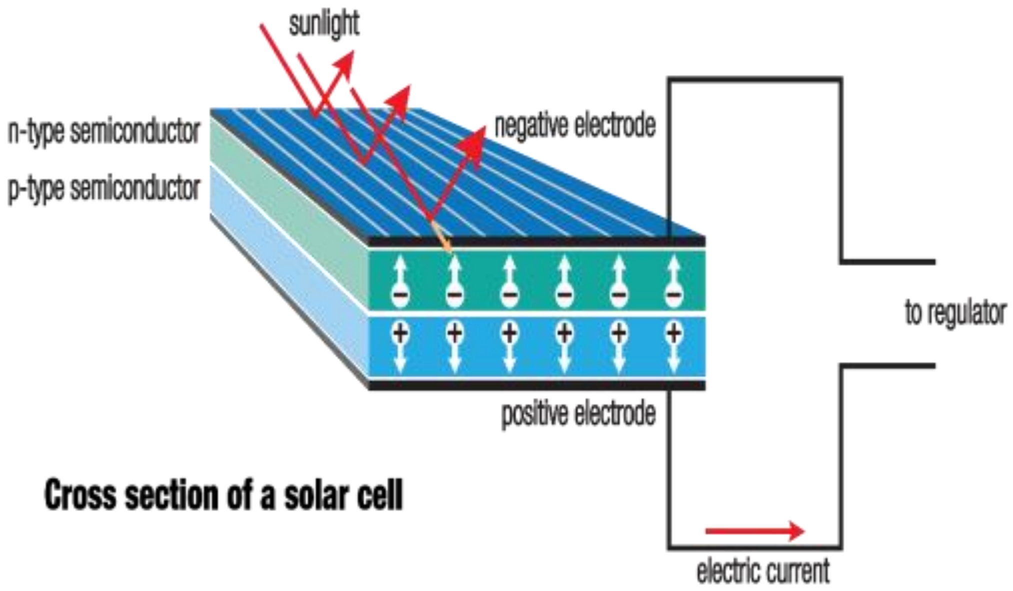


Fig. 7 Photovoltaic cell working principle.

CHAPTER - 3
DESIGN/PROTOTYPE MODEL

3.1 Circuit Board:

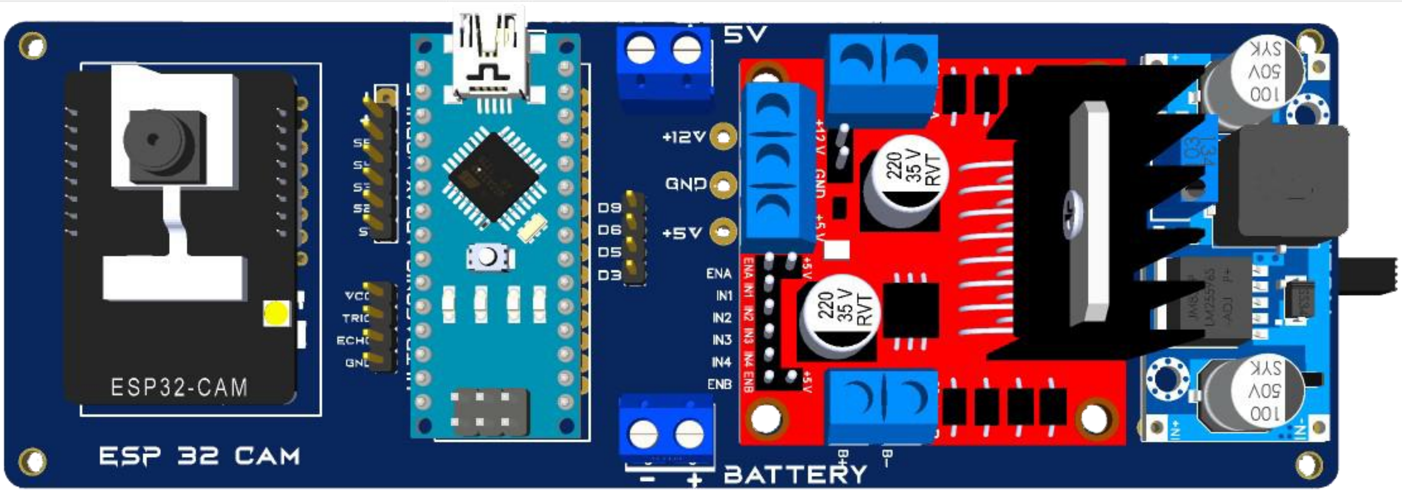


Fig. 8 Controlling Circuit Board

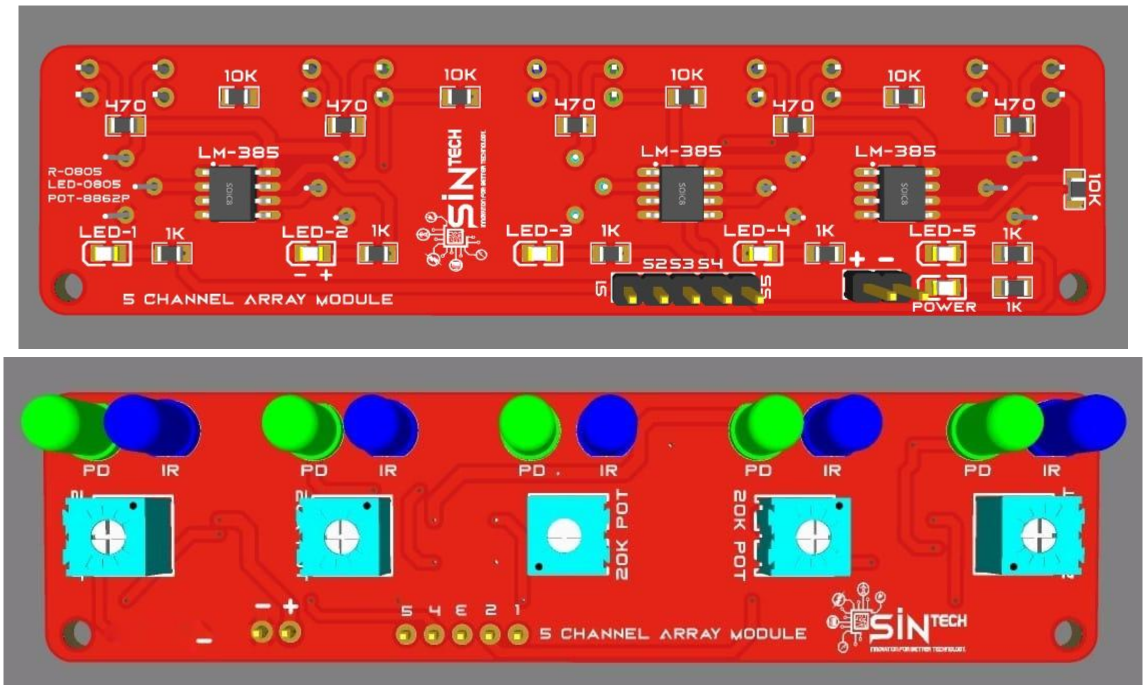


Fig. 9 IR Array Module

3.1 Body and Chassis

The frame and body is the main supporting structure of a motor vehicle. All other components are attached to this structure. The frame is also called the chassis. where all the machine parts are installed. This frame and body are made up of a tramsparent plastic sheets.These sheets are made of a thin layer of material. This material is not only lightweight but also durable.

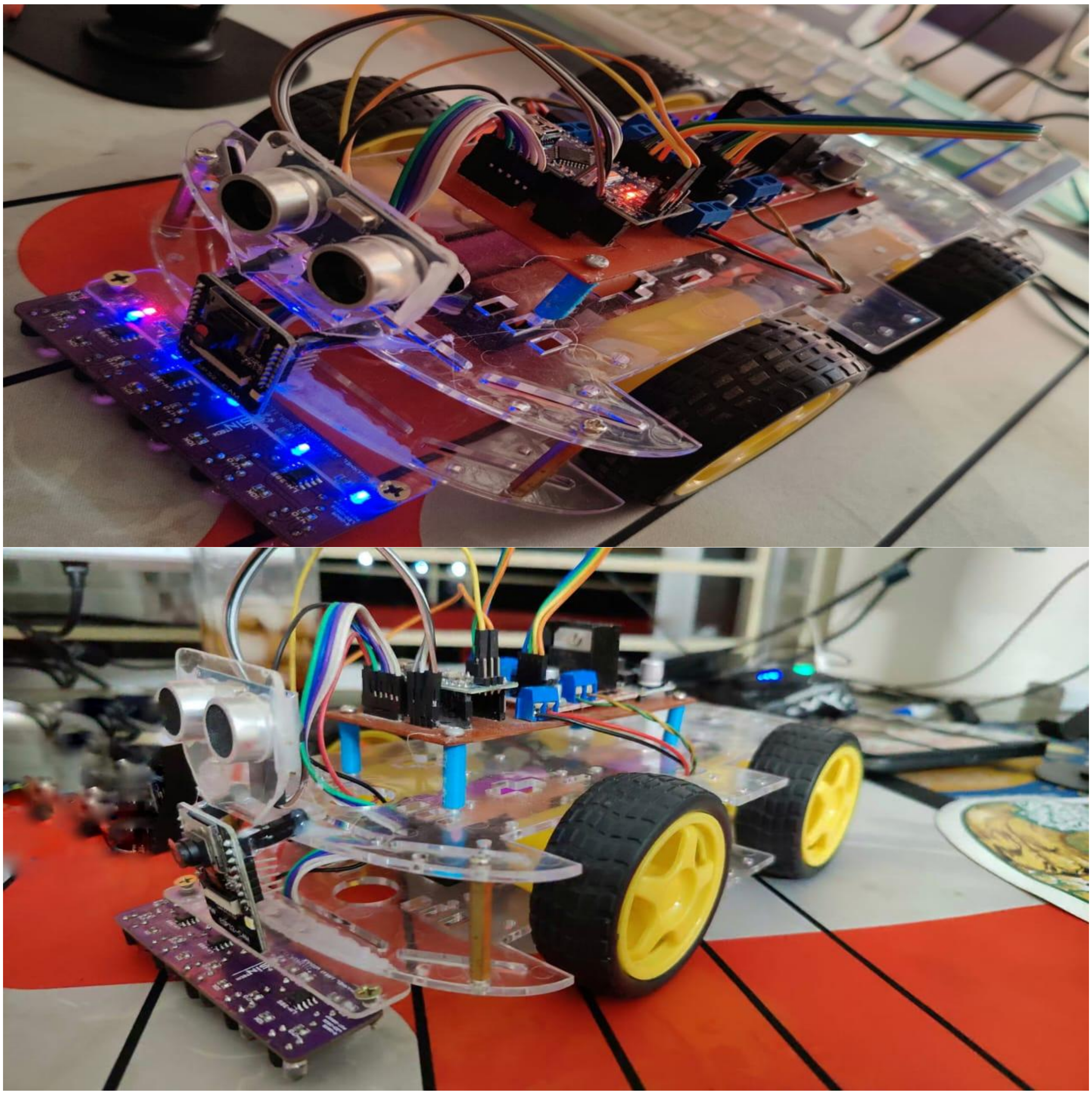


Fig.10 Body and Chassis.

3.2 Front view of our Vehicle

We are using several sensors in the vehicle, this sensor will control our vehicle via driving mode.

Here the sensor we are using is the ultrasonic sensor module, IR array module, IR sensor module.

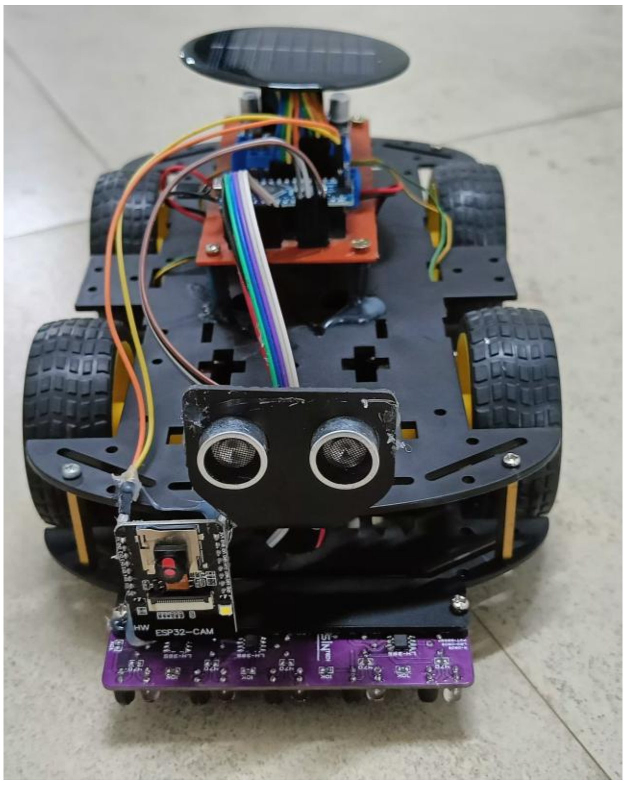


Fig.11 Front view of our Vehicle

3.3 Side view of our Vehicle

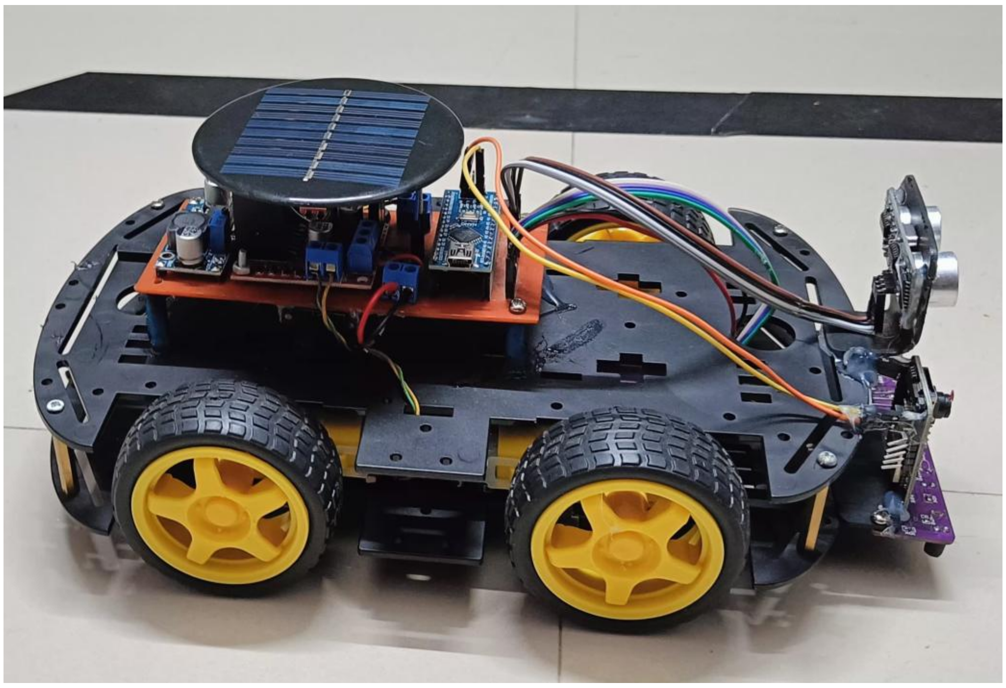


Fig.12 Side view of our vehicle.

3.4 Upper front view of our Vehicle

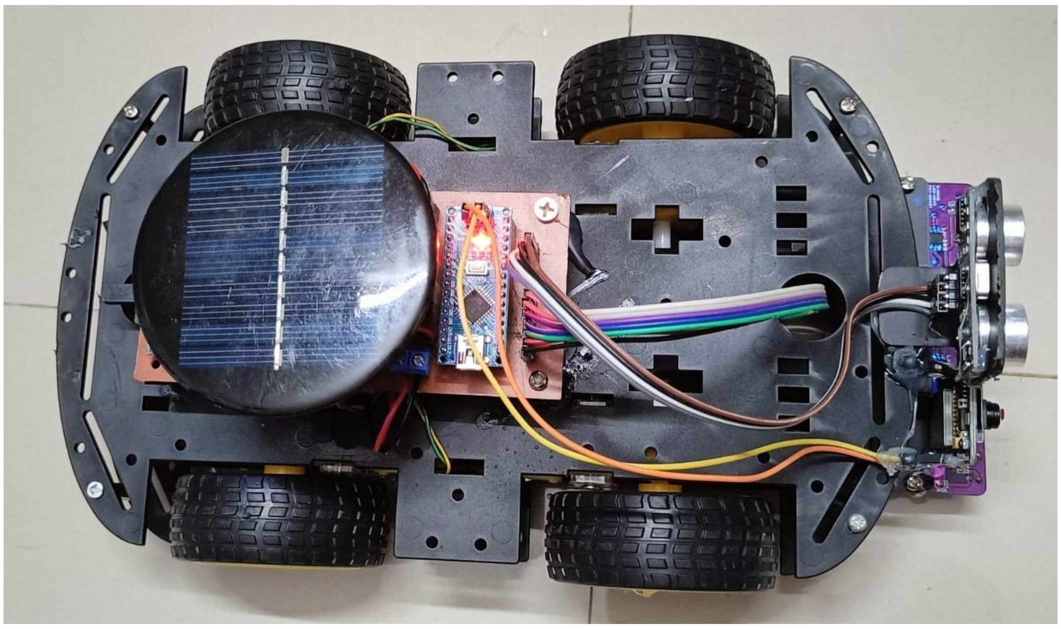


Fig.13 Upper front view of our Vehicle.

3.5 Code

```
#define trigPin 4
#define echoPin 2
long duration;
long distance;
int motor1 = 9;
int motor2 = 6;
int motor3 = 5;
int motor4 = 3;
intir1 = 7;
intir2 = 8;
intir3 = 10;
intir4 = 11;
intir5 = 12;
int Max_Left_Sensor ;
int Left_Sensor;
int Middle_Sensor;
int Right_Sensor;
int Max_Right_Sensor;
void setup() {
Serial.begin (9600);
pinMode(motor1,OUTPUT);
pinMode(motor2,OUTPUT);
pinMode(motor3,OUTPUT);
pinMode(motor4,OUTPUT);
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);
pinMode(ir1,INPUT);
pinMode(ir2,INPUT);
pinMode(ir3,INPUT);
pinMode(ir4,INPUT);
pinMode(ir5,INPUT);



}
void(* resetFunc) (void) = 0; //declare reset


function @ add
   void loop() {




    Max_Left_Sensor = digitalRead(ir1);
    Left_Sensor = digitalRead(ir2);
    Middle_Sensor = digitalRead(ir3);
    Right_Sensor = digitalRead(ir4);
    Max_Right_Sensor = digitalRead(ir5);
```

```
line_follow();
ultrasonic();
}

void ultrasonic(){
    digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);

    // Measure the response from the HC-SR04 Echo
Pin
duration = pulseIn(echoPin, HIGH);
distance= duration*0.034/2;
Serial.print("distance: ");
Serial.println(distance);
delay(20);


if(distance < 18){
//reverse();
//delay(300);
ultra_right(); //Move Right
delay(700);
forward();
delay(500);
ultra_left(); //FORWARD
delay(1000);
forward();

}
}
void line_follow(){
Max_Left_Sensor = digitalRead(ir1);
Left_Sensor = digitalRead(ir2);
Middle_Sensor = digitalRead(ir3);
Right_Sensor = digitalRead(ir4);
Max_Right_Sensor = digitalRead(ir5);
Serial.print(Max_Left_Sensor);
Serial.print(Left_Sensor);
Serial.print(Middle_Sensor);
Serial.print(Right_Sensor);
Serial.println(Max_Right_Sensor);

   if(Max_Left_Sensor==0 && Left_Sensor==0 &&
Middle_Sensor==0 && Right_Sensor==0 &&
Max_Right_Sensor==1) {
fastleft(); //fastleft move
}
   if(Max_Left_Sensor==0 && Left_Sensor==0 &&
Middle_Sensor==0 && Right_Sensor==1 &&
Max_Right_Sensor==1) {
fastleft(); //fastleft move
}
    if(Max_Left_Sensor==1 && Left_Sensor==0 &&
Middle_Sensor==0 && Right_Sensor==1 &&
```

```
    Max_Right_Sensor==1){
left(); //Move Left
}
```

```
        if(Max_Left_Sensor==1 &&
        Left_Sensor==1 &&
        Middle_Sensor==0 &&
        Right_Sensor==0 &&
        Max_Right_Sensor==1) {
           forward(); //FORWARD
           }
           if(Max_Left_Sensor==1 &&
        Left_Sensor==0 &&
        Middle_Sensor==0 &&
        Right_Sensor==0 &&
        Max_Right_Sensor==1) {
           forward(); //FORWARD
           }
           if(Max_Left_Sensor==1 &&
        Left_Sensor==0 &&
        Middle_Sensor==0 &&
        Right_Sensor==1 &&
        Max_Right_Sensor==1) {
           forward(); //FORWARD
           }


           if(Max_Left_Sensor==1 &&
        Left_Sensor==1 &&
        Middle_Sensor==0 &&
        Right_Sensor==0 &&
        Max_Right_Sensor==1) {
           right(); //Move Right
           }
           if(Max_Left_Sensor==1 &&
        Left_Sensor==1 &&
        Middle_Sensor==0 &&
        Right_Sensor==0 &&
        Max_Right_Sensor==0) {
           fastright(); //Move Right
           }
           if(Max_Left_Sensor==1 &&
        Left_Sensor==0 &&
        Middle_Sensor==0 &&
        Right_Sensor==0 &&

Max_Right_Sensor==0) {
fastright(); //Move Right
}
if(Max_Left_Sensor==0 && Left_Sensor==0
        && Middle_Sensor==0 &&
        Right_Sensor==0 &&
Max_Right_Sensor==0) {
Stop();  //STOP
}
```

```
}
void forward()

{
analogWrite(motor1, 120);
digitalWrite(motor2, LOW);
analogWrite(motor3, 120);
digitalWrite(motor4, LOW);
}
void reverse()
{
analogWrite(motor1, 0);
digitalWrite(motor2, 120);
analogWrite(motor3, 0);
digitalWrite(motor4, 120);
}
void right()
{
digitalWrite(motor1, LOW);
analogWrite(motor2, 0);
analogWrite(motor3, 100);
digitalWrite(motor4, LOW);
```

```
    }
    void fastright()
    {
            digitalWrite(motor1, LOW);
            analogWrite(motor2, 0);
            analogWrite(motor3, 130);
            digitalWrite(motor4, LOW);
    }
    void left()
    {
            analogWrite(motor1, 100);
            digitalWrite(motor2, LOW);
            digitalWrite(motor3, LOW);
            analogWrite(motor4, 0);
    }
    void fastleft()
    {
            analogWrite(motor1, 130);
            digitalWrite(motor2, LOW);
            digitalWrite(motor3, LOW);
            analogWrite(motor4, 0);
    }
    void Stop()
    {
            digitalWrite(motor1, LOW);
            digitalWrite(motor2, LOW);
            digitalWrite(motor3, LOW);
            digitalWrite(motor4, LOW);
    }


 void ultra_right()

{

digitalWrite(motor1, LOW);


analogWrite(motor2, 0);

analogWrite(motor3, 120);

digitalWrite(motor4, LOW);
```

```
}
void ultra_left()
{
analogWrite(motor1, 120);
digitalWrite(motor2, LOW);
digitalWrite(motor3, LOW);
analogWrite(motor4, 0);
}
```

```
#include <esp_camera.h>
#include <esp_int_wdt.h>
#include <esp_task_wdt.h>
#include <WiFi.h>
#include <DNSServer.h>
#include <WiFiUdp.h>
#include <ArduinoOTA.h>
#include "src/parsebytes.h"
#include "time.h"
#include <ESPmDNS.h>

#if __has_include("myconfig.h")
   struct station { const char ssid[65]; const char
password[65]; const bool dhcp;};  // do no edit
   #include "myconfig.h"
#else
   #warning "Using Defaults: Copy
myconfig.sample.h to myconfig.h and edit that to
use your own settings"
   #define WIFI_AP_ENABLE
   #define CAMERA_MODEL_AI_THINKER
   struct station { const char ssid[65]; const char
password[65]; const bool dhcp;}
   stationList[] = {{"ESP32-CAM-
CONNECT","InsecurePassword", true}};
#endif

// Upstream version string
#include "src/version.h"

// Pin Mappings
#include "camera_pins.h"

// Camera config structure

camera_config_t config;

// Internal filesystem (SPIFFS)
// used for non-volatile camera settings
#include "storage.h"

// Sketch Info
int sketchSize;
int sketchSpace;
String sketchMD5;

// Start with accesspoint mode
 disabled, wifi setup will activate it if
// no known networks are found, and
WIFI_AP_ENABLE has been defined
```

```
bool accesspoint = false;

// IP address, Netmask and Gateway,
populated when connected
IPAddress ip;
IPAddress net;
IPAddress gw;

// Declare external function from
app_httpd.cpp
extern void startCameraServer
int hPort, int sPort);
extern void serialDump();

// Names for the Camera. (set these in
myconfig.h)
#if defined(CAM_NAME

 char myName[] = CAM_NAME;

#else
   char myName[] = "ESP32 camera server";
#endif


#if defined(MDNS_NAME)
   char mdnsName[] = MDNS_NAME;
#else
   char mdnsName[] = "esp32-cam";
#endif

// Ports for http and stream (override in myconfig.h)
#if defined(HTTP_PORT)
   int httpPort = HTTP_PORT;
#else
   int httpPort = 80;
#endif

#if defined(STREAM_PORT)
   int streamPort = STREAM_PORT;
#else
   int streamPort = 81;
#endif

#if !defined(WIFI_WATCHDOG)
   #define WIFI_WATCHDOG 15000
#endif
```

```
// Number of known networks in stationList[]

int stationCount = s
izeof(stationList)/sizeof(stationList[0]);


  // If we have AP mode enabled, ignore
  first
  entry in the stationList[]
  #if defined(WIFI_AP_ENABLE)
     int firstStation = 1;

  #else
     int firstStation = 0;
  #endif

  // Select between full and simple
   index as the default.
  #if defined(DEFAULT_INDEX_FULL)
     char default_index[] = "full";
  #else
     char default_index[] = "simple";
  #endif

  // DNS server
  const byte DNS_PORT = 53;
  DNSServer dnsServer;
  bool captivePortal = false;
  char apName[64] = "Undefined";

  // The app and stream URLs
  char httpURL[64] = {"Undefined"};
  char streamURL[64] = {"Undefined"};

  // Counters for info screens and debug
  int8_t streamCount = 0;
  unsigned long streamsServed = 0;


  unsigned long imagesServed = 0;

  // This will be displayed to identify the firmware
  char myVer[] PROGMEM = _DATE_ " @ " _TIME_;

  // This will be set to the sensors PID (identifier) during
  initialisation
  //camera_pid_t sensorPID;
  int sensorPID;
```

```
.
#if !defined (XCLK_FREQ_MHZ)
   unsigned long xclk = 8;
#else
   unsigned long xclk = XCLK_FREQ_
MHZ;
#endif

// initial rotation
// can be set in myconfig.h
#if !defined(CAM_ROTATION)
   #define CAM_ROTATION 0
#endif
int myRotation = CAM_ROTATION;

// minimal frame duration in ms,
effectively 1/maxFPS
#if !defined(MIN_FRAME_TIME)
   #define MIN_FRAME_TIME 0
#endif

int minFrameTime = MIN_FRAME_TIME;

// Illumination LAMP and status LED
#if defined(LAMP_DISABLE)
   int lampVal = -1;
 // lamp is disabled in config
#elif defined(LAMP_PIN)
   #if defined(LAMP_DEFAULT)
      int lampVal =
constrain(LAMP_DEFAULT,0,100);
// initial lamp value, range 0-100
   #else
      int lampVal = 0; //default to off
   #endif
#else
   int lampVal = -1; // no lamp
 pin assigned
#endif

#if defined(LED_DISABLE)
   #undef LED_PIN
#endif

bool autoLamp = false;


int lampChannel = 7;
```

```
const int pwmfreq = 50000;
frequency
const int pwmresolution = 9;
range
const int pwmMax =
pow(2,pwmresolution)-1;

#if defined(NO_FS)
   bool filesystem = false;
#else
   bool filesystem = true;
#endif

#if defined(NO_OTA)
   bool otaEnabled = false;
#else
   bool otaEnabled = true;
#endif

#if defined(OTA_PASSWORD)
   char otaPassword[] = OTA_PASSWORD;
#else
   char otaPassword[] = "";
#endif

#if defined(NTPSERVER)
   bool haveTime = true;
   const char* ntpServer = NTPSERVER;
   const long  gmtOffset_sec = NTP_GMT_OFFSET;
   const int   daylightOffset_sec = NTP_DST_OFFSET;

#else
   bool haveTime = false;
   const char* ntpServer = "";
   const long  gmtOffset_sec = 0;
   const int   daylightOffset_sec = 0;
#endif



String critERR = "";

// Debug flag for stream and capture data
bool debugData;

void debugOn() {
   debugData = true;
   Serial.println("Camera debug
data is enabled (send 'd' for status
dump, or any other char
to disable debug)");
}

void debugOff() {
   debugData = false;
   Serial.println("Camera
 debug data is disabled
(send 'd' for status dump,
or any other char to enable debug)");
}

// Serial input (debugging controls)

void handleSerial() {
   if (Serial.available()) {
      char cmd = Serial.read();
      if (cmd == 'd' ) {
         serialDump();
      } else {
         if (debugData) debugOff();
         else debugOn();
      }
   }
   while (Serial.available()) Serial.read();  //
chomp the buffer
}

// Notification LED
void flashLED(int flashtime) {
#if defined(LED_PIN)
// If we have it; flash it.

   digitalWrite(LED_PIN, LED_ON);
 // On at full power.
   delay(flashtime);              // delay
   digitalWrite(LED_PIN, LED_OFF);
// turn Off
#else
   return;
 // No notifcation LED, do nothing,
 no delay
#endif
}

// Lamp Control
void setLamp(int newVal) {
```

```
#if defined(LAMP_PIN)
   if (newVal != -1) {
      // Apply a logarithmic function to the scale.

int brightness = round((pow(2,(1+(newVal*0.02)))-
2)/6*pwmMax);
      ledcWrite(lampChannel, brightness);
      Serial.print("Lamp: ");
      Serial.print(newVal);
      Serial.print("%, pwm = ");
      Serial.println(brightness);
   }
#endif
}

void printLocalTime(bool extraData=false) {
   struct tm timeinfo;
   if(!getLocalTime(&timeinfo)){
      Serial.println("Failed to obtain time");
   } else {
      Serial.println(&timeinfo, "%H:%M:%S, %A, %B
%d %Y");
   }
   if (extraData) {
      Serial.printf("NTP Server: %s, GMT Offset: %li(s),
DST Offset: %i(s)\r\n", ntpServer, gmtOffset_sec,
daylightOffset_sec);
   }
}

void calcURLs() {
   // Set the URL's
   #if defined(URL_HOSTNAME)
      if (httpPort != 80) {
         sprintf(httpURL, "http://%s:%d/",
URL_HOSTNAME, httpPort);

 } else {
         sprintf(httpURL, "http://%s/",
URL_HOSTNAME);
      }
      sprintf(streamURL, "http://%s:%d/",
URL_HOSTNAME, streamPort);
   #else
      Serial.println("Setting httpURL");
      if (httpPort != 80) {
         sprintf(httpURL, "http://%d.%d.%d.%d:%d/",
ip[0], ip[1], ip[2], ip[3], httpPort);
  } else {
         sprintf(httpURL,
"http://%d.%d.%d.%d/", ip[0],
 ip[1], ip[2], ip[3]);
      }

 sprintf(streamURL, "http://%d.%d.
%d.%d:%d/", ip[0], ip[1], ip[2], ip[3],
streamPort);
   #endif
}

void StartCamera() {
   // Populate camera config structure with
hardware and other defaults
   config.ledc_channel = LEDC_
CHANNEL_0;
   config.ledc_timer = LEDC_TIMER_0;
   config.pin_d0 = Y2_GPIO_NUM;
   config.pin_d1 = Y3_GPIO_NUM;
   config.pin_d2 = Y4_GPIO_NUM;
   config.pin_d3 = Y5_GPIO_NUM;
   config.pin_d4 = Y6_GPIO_NUM;
   config.pin_d5 = Y7_GPIO_NUM;
   config.pin_d6 = Y8_GPIO_NUM;
   config.pin_d7 = Y9_GPIO_NUM;

config.pin_xclk = XCLK_GPIO_NUM;
config.pin_pclk = PCLK_GPIO_NUM;
config.pin_vsync = VSYNC_GPIO_NUM;
config.pin_href = HREF_GPIO_NUM;
config.pin_sscb_sda = SIOD_GPIO_NUM;
config.pin_sscb_scl = SIOC_GPIO_NUM;
config.pin_pwdn = PWDN_GPIO_NUM;
 config.pin_reset = RESET_GPIO_NUM;
 config.xclk_freq_hz = xclk * 1000000;
config.pixel_format =
PIXFORMAT_JPEG;
 // Low(ish) default framesize and quality
  config.frame_size =
FRAMESIZE_SVGA;
   config.jpeg_quality = 12;
   config.fb_location =
CAMERA_FB_IN_PSRAM;
   config.fb_count = 2;
   config.grab_mode =
CAMERA_GRAB_LATEST;
```

```
  #if defined
(CAMERA_MODEL_ESP_EYE)
    pinMode(13, INPUT_PULLUP);
    pinMode(14, INPUT_PULLUP);
  #endif

  // camera init
  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    delay(100);


// need a delay here or the next serial o/p gets missed
    Serial.printf("\r\n\r\nCRITICAL FAILURE:
Camera sensor failed to initialise.\r\n\r\n");
    Serial.printf("A full (hard, power off/on) reboot will
probably be needed to recover from this.\r\n");
    Serial.printf("Meanwhile; this unit will reboot in 1
minute since these errors sometime clear
automatically\r\n");
    // Reset the I2C bus.. may help when rebooting.

periph_module_disable(PERIPH_I2C0_MODULE); //
try to shut I2C down properly in case that is the problem

periph_module_disable(PERIPH_I2C1_MODULE);
    periph_module_reset(PERIPH_I2C0_MODULE);
    periph_module_reset(PERIPH_I2C1_MODULE);
    // And set the error text for the UI
    critERR = "<h1>Error!</h1><hr><p>Camera
module failed to initialise!</p><p>Please reset (power
off/on) the camera.</p>";
    critERR += "<p>We will continue to reboot once
per minute since this error sometimes clears
automatically.</p>";
    // Start a 60 second watchdog timer
    esp_task_wdt_init(60,true);
    esp_task_wdt_add(NULL);
  } else {
    Serial.println("Camera init succeeded");

    // Get a reference to the sensor
    sensor_t * s = esp_camera_sensor_get();

    // Dump camera module, warn for unsupported
modules.
    sensorPID = s->id.PID;
 switch (sensorPID) {
      case OV9650_PID:
Serial.println("WARNING:
OV9650 camera module is not
 properly supported, will fallback
to OV2640 operation"); break;
      case OV7725_PID:
Serial.println("WARNING:
camera module is not properly
supported, will fallback to
OV2640 operation"); break;
      case OV2640_PID:
Serial.println("OV2640 camera
module detected"); break;
      case OV3660_PID:
Serial.println("OV3660 camera
module detected"); break;
      default: Serial.println
("WARNING: Camera module
 is unknown and not properly
 supported, will fallback
 to OV2640 operation");
    }

 // OV3660 initial sensors are flipped
are a bit saturated
    if (sensorPID == OV3660_PID) {
      s->set_vflip(s, 1);  //flip it back
      s->set_brightness(s, 1);
 //up the blightness just a bit
      s->set_saturation(s, -2);
 //lower the saturation

}

    // M5 Stack Wide has special needs
    #if defined
(CAMERA_MODEL_M5STACK_WIDE)
      s->set_vflip(s, 1);
      s->set_hmirror(s, 1);
    #endif

    // Config can override mirror and flip
    #if defined(H_MIRROR)
      s->set_hmirror(s, H_MIRROR);
    #endif
    #if defined(V_FLIP)
      s->set_vflip(s, V_FLIP);
    #endif

     // set initial frame rate
     #if defined(DEFAULT_RESOLUTION)
        s->set_framesize(s, DEFAULT_RESOLUTION);
     #endif

     /*
     * Add any other defaults you want to apply at
startup here:
     * uncomment the line and set the value as desired
(see the comments)
     *
     * these are defined in the esp headers here:
     * https://github.com/espressif/esp32-
camera/blob/master/driver/include/sensor.h#L149
     */

//s->set_framesize
(s, FRAMESIZE_SVGA);
     //s->set_quality(s, val);
// 10 to 63
     //s->set_brightness(s, 0);
 // -2 to 2
     //s->set_contrast(s, 0);
// -2 to 2
     //s->set_saturation(s, 0);
 // -2 to 2
     //s->set_special_effect(s, 0);
// 0 to 6 (0 - No Effect, 1 - Negative, 2 - Grayscale, 3 -
Red Tint,
4 - Green Tint, 5 - Blue Tint, 6 - Sepia)
     //s->set_whitebal(s, 1);
// aka 'awb' in the UI; 0 = disable ,
1 = enable
     //s->set_awb_gain(s, 1);     /
/ 0 = disable , 1 = enable
     //s->set_wb_mode(s, 0);
 // 0 to 4 - if awb_gain enabled
(0 - Auto, 1 - Sunny, 2 - Cloudy,
3 - Office, 4 - Home)
     //s->set_exposure_ctrl(s, 1);
// 0 = disable , 1 = enable
     //s->set_aec2(s, 0);
// 0 = disable , 1 = enable
     //s->set_ae_level(s, 0);
// -2 to 2
     //s->set_aec_value(s, 300);
 // 0 to 1200

//s->set_gain_ctrl(s, 1);      enable
     //s->set_agc_gain(s, 0);
     //s->set_gainceiling(s, (gainceiling_t)
0);  // 0 to 6
     //s->set_bpc(s, 0);

 // 0 = disable , 1 = enable
     //s->set_wpc(s, 1);
 // 0 = disable , 1 = enable
     //s->set_raw_gma(s, 1);

   // 0 = disable , 1 = enable
     //s->set_lenc(s, 1);
// 0 = disable , 1 = enable
     //s->set_hmirror(s, 0);
// 0 = disable , 1 = enable
     //s->set_vflip(s, 0);
 // 0 = disable , 1 = enable
     //s->set_dcw(s, 1);
 // 0 = disable , 1 = enable
     //s->set_colorbar(s, 0);
// 0 = disable , 1 = enable
   }
   // We now have camera with default init
}

void WifiSetup() {
   // Feedback that we are
 now attempting to connect
   flashLED(300);
   delay(100);
   flashLED(300);
   Serial.println("Starting WiFi");

   // Disable power saving on
WiFi to improve responsiveness
   // (https://github.com/espressif/arduino-
esp32/issues/1484)
   WiFi.setSleep(false);

   Serial.print("Known external SSIDs: ");
   if (stationCount > firstStation) {

 for (int i=firstStation;
i < stationCount; i++) Serial.printf(" '%s'",
stationList[i].ssid);
```

```
  } else {
      Serial.print("None");
    }
    Serial.println();

byte mac[6] = {0,0,0,0,0,0};
    WiFi.macAddress(mac);
    Serial.printf("MAC address:
%02X:%02X:%02X:%02X:
%02X:%02X\r\n", mac[0], mac[1],
mac[2], mac[3], mac[4], mac[5]);

    int bestStation = -1;
    long bestRSSI = -1024;
    char bestSSID[65] = "";
    uint8_t bestBSSID[6];
    if (stationCount > firstStation) {
      // We have a list to scan
      Serial.printf("Scanning local Wifi Networks\r\n");
      int stationsFound =
WiFi.scanNetworks();
      Serial.printf("%i
networks found\r\n", stationsFound);
      if (stationsFound > 0) {
        for (int i = 0;
i < stationsFound; ++i) {
            // Print SSID and
RSSI for each network found
            String thisSSID =

WiFi.SSID(i);
            int thisRSSI = WiFi.RSSI(i);
            String thisBSSID = WiFi.BSSIDstr(i);
            Serial.printf("%3i : [%s] %s (%i)", i + 1,
thisBSSID.c_str(), thisSSID.c_str(), thisRSSI);
            // Scan our list of known external stations
            for (int sta = firstStation; sta < stationCount;
sta++) {
              if ((strcmp(stationList[sta].ssid,
thisSSID.c_str()) == 0) ||
              (strcmp(stationList[sta].ssid,
thisBSSID.c_str()) == 0)) {
                Serial.print("  -  Known!");
                // Chose the strongest RSSI seen
                if (thisRSSI > bestRSSI) {
                  bestStation = sta;
                  strncpy(bestSSID, thisSSID.c_str(),
64);
// Convert char bssid[] to a byte array

  parseBytes(thisBSSID.c_str(), ':', b
estBSSID, 6, 16);
                  bestRSSI = thisRSSI;
                }


  }
            }
            Serial.println();
          }
        }
    } else {
        // No list to scan, therefore we are an
accesspoint
        accesspoint = true;
    }



  if (bestStation == -1) {
        if (!accesspoint) {
          #if defined(WIFI_AP_ENABLE)
Serial.println("No known networks found,
entering AccessPoint fallback mode");
            accesspoint = true;
          #else
            Serial.println
("No known networks found");
          #endif
        } else {
          Serial.println
("AccessPoint mode selected in config");
        }
    } else {
        Serial.printf
("Connecting to Wifi Network %d:
[%02X:%02X:%02X:
%02X:%02X:%02X] %s \r\n"
 bestStation, bestBSSID[0], bestBSSID[1],
bestBSSID[2], bestBSSID[3],
bestBSSID[4], bestBSSID[5], bestSSID);
 // Apply static settings if necesscary
  if (stationList[bestStation].dhcp == false)
{

 #if defined(ST_IP)
            Serial.println("Applying static IP
settings");
#if !defined (ST_GATEWAY)  || !defined
(ST_NETMASK)



 #error "You must supply both


Gateway and NetMask when specifying a static IP
address"

#endif
            IPAddress staticIP(ST_IP);
            IPAddress gateway(ST_GATEWAY);
            IPAddress subnet(ST_NETMASK);
            #if !defined(ST_DNS1)
               WiFi.config(staticIP, gateway, subnet);
            #else
               IPAddress dns1(ST_DNS1);
            #if !defined(ST_DNS2)
               WiFi.config(staticIP, gateway, subnet,
dns1);
            #else
               IPAddress dns2(ST_DNS2);
               WiFi.config(staticIP, gateway, subnet, dns1,
dns2);
            #endif
            #endif
         #else
            Serial.println("Static IP settings requested but
not defined in config, falling back to dhcp");
         #endif
      }

      WiFi.setHostname(mdnsName);

      // Initiate network connection request (3rd
argument, channel = 0 is 'auto')

WiFi.begin(bestSSID, stationList[bestStation].password,
0, bestBSSID);



 // Wait to connect, or timeout
      unsigned long start = millis();
      while ((millis() - start <= WIFI_WATCHDOG) &&
(WiFi.status() != WL_CONNECTED)) {
         delay(500);
Serial.print('.');
      }
      // If we have connected, inform user
      if (WiFi.status()
== WL_CONNECTED) {
         Serial.println("Client connection
succeeded");
         accesspoint = false;
         // Note IP details

ip = WiFi.localIP();
         net = WiFi.subnetMask();
         gw = WiFi.gatewayIP();

 Serial.printf("IP address:
%d.%d.%d.%d\r\n",ip[0],ip[1],i
p[2],ip[3]);
         Serial.printf("Netmask   :
%d.%d.%d.%d\r\n",net[0],net[1],
net[2],net[3]);
         Serial.printf("Gateway   :
%d.%d.%d.%d\r\n",gw[0],gw[1],
gw[2],gw[3]);
         calcURLs();
      for (int i = 0; i < 5; i++) {
            flashLED(50);
            delay(150);
         }
      } else {

 Serial.println("Client connection Failed");

  WiFi.disconnect();
 // (resets the WiFi scan)
      }
   }


   if (accesspoint && (WiFi.status() !=
WL_CONNECTED)) {
      // The accesspoint has
 been enabled, and we have not
connected to any existing networks
      #if defined(AP_CHAN)
         Serial.println("Setting up
 Fixed Channel AccessPoint");
         Serial.print("  SSID     : ");
         Serial.println(stationList[0].ssid);
         Serial.print("  Password : ");
Serial.println(stationList[0].
password);
            Serial.print("  Channel  : ");
            Serial.println(AP_CHAN);
            WiFi.softAP(stationList[0].ssid,
stationList[0].password, AP_CHAN);
        # else
            Serial.println("Setting up
 AccessPoint");
            Serial.print("  SSID     : ");

  Serial.println(stationList[0].ssid);
            Serial.print("  Password : ");
            Serial.println(stationList[0]
.password);
            WiFi.softAP(stationList[0].ssid,
stationList[0].password);
        #endif
        #if defined(AP_ADDRESS)
            // User has specified the AP
 details; apply them after a short delay

  delay(100);

 IPAddress local_IP(AP_ADDRESS);
        IPAddress gateway(AP_ADDRESS);
        IPAddress subnet(255,255,255,0);
      WiFi.softAPConfig(local_IP, gateway, subnet);
        #endif
        // Note AP details
        ip = WiFi.softAPIP();
        net = WiFi.subnetMask();
        gw = WiFi.gatewayIP();
        strcpy(apName, stationList[0].ssid);
        Serial.printf("IP address:
%d.%d.%d.%d\r\n",ip[0],ip[1],ip[2],ip[3]);
        calcURLs();
        // Flash the LED to
 show we are connected
        for (int i = 0; i < 5; i++) {
            flashLED(150);
            delay(50);
        }
        // Start the DNS captive
portal if requested
        if (stationList[0].dhcp == true) {
            Serial.println
("Starting Captive Portal");
dnsServer.start
(DNS_PORT, "*", ip);
            captivePortal = true;
        }
    }

}

void setup() {
    Serial.begin(115200);


 Serial.setDebugOutput(true);
    Serial.println();
    Serial.println("====");

Serial.print("esp32-cam-webserver: ");
Serial.println(myName);
 Serial.print("Code Built: ");
 Serial.println(myVer);
 Serial.print("Base Release: ");
  Serial.println(baseVersion);
  Serial.println();

    // Warn if no PSRAM is detected
(typically user error with board
selection in the IDE)
    if(!psramFound()){
        Serial.println(
"\r\nFatal Error; Halting");
        while (true) {
            Serial.println("No PSRAM found;
camera cannot be initialised:
Please check the board
config for your module.");
            delay(5000);
        }
    }

    if (stationCount == 0) {
        Serial.println
("\r\nFatal Error; Halting");
        while (true) {

Serial.println("No wifi details
have been configured; we
 cannot connect to existing
 WiFi or start our own AccessPoint,
there is no point in proceeding.");
            delay(5000);
        }
    }

    #if defined(LED_PIN)  // If we have a notification
LED, set it to output
        pinMode(LED_PIN, OUTPUT);

 digitalWrite(LED_PIN, LED_ON);
    #endif

    // Start the SPIFFS filesystem before we initialise the
camera
    if (filesystem) {
        filesystemStart();

 delay(200);    }

    // Start (init) the camera
    StartCamera();

        if (filesystem) {
        delay(200);
        loadPrefs(SPIFFS);
    } else {
        Serial.println
("No Internal Filesystem,
cannot load or save preferences");

  }

    /*
    * Camera setup complete; initialise the rest of the
hardware.
    */

    // Start Wifi and loop until we are connected or have
started an AccessPoint
    while ((WiFi.status() != WL_CONNECTED) &&
!accesspoint)  {
        WifiSetup();
        delay(1000);

    // Set up OTA
    if (otaEnabled) {
        // Start OTA once connected
        Serial.println("Setting up OTA");
// Port defaults to 3232
        // ArduinoOTA.setPort(3232);
        // Hostname defaults to esp3232
-[MAC]
        ArduinoOTA.setHostname
(mdnsName);
        // No authentication by default
        if (strlen(otaPassword) != 0) {
            ArduinoOTA.setPassword
(otaPassword);
            Serial.printf("OTA
Password: %s\n\r", otaPassword);
        } else {
            Serial.printf("\r\nNo OTA
password has been set! (insecure)\r\n\r\n");
        }
        ArduinoOTA
            .onStart([]() {

   String type;

if (ArduinoOTA.getCommand() ==
U_FLASH)
                    type = "sketch";
                else // U_SPIFFS
SPIFFS.end()
  type = "filesystem";
Serial.println("Start updating " + type);
   Serial.println("Stopping Camera");
esp_err_t err = esp_camera_deinit();
  critERR = "<h1>OTA Has been
started</h1><hr><p>
Camera has Halted!</p>";
 critERR += "<p>Wait for OTA
to finish and reboot, or
 <a href=\"control?var=reboot&val=0\"
title=\"Reboot Now (may interrupt OTA)
\">reboot manually</a> to recover</p>";
            })
            .onEnd([]() {
                Serial.println("\r\nEnd");
            })
            .onProgress([]
(unsigned int progress,
 unsigned int total) {
                Serial.printf
("Progress: %u%%\r", (progress
/ (total / 100)));
          })
          .onError([](ota_error_t error) {
         Serial.printf("Error[%u]: ", error);

if (error == OTA_AUTH_ERROR) Serial.println("Auth
Failed");
             else if
(error == OTA_BEGIN_ERROR) Serial.println("Begin
Failed");
             else if (error == OTA_CONNECT_ERROR)
Serial.println("Connect Failed");
             else if (error == OTA_RECEIVE_ERROR)
Serial.println("Receive Failed");
             else if

(error == OTA_END_ERROR) Serial.println("End
Failed");
          });

 ArduinoOTA.begin();
   } else {
      Serial.println("OTA is disabled");


  if (!MDNS.begin(mdnsName)) {
        Serial.println("Error setting up MDNS
responder!");
      }
      Serial.println("mDNS responder started");
   }

   //MDNS Config -- note that if OTA is NOT enabled
this needs prior steps!
   MDNS.addService("http", "tcp", 80);

 Serial.println("Added HTTP
service to MDNS server");



 // Set time via NTP server when
 enabled
   if (haveTime) {
      Serial.print("Time: ");
      configTime(gmtOffset_sec, daylightOffset_sec,
ntpServer);
      printLocalTime(true);
   } else {
Serial.println("Time functions disabled");
   }

   sketchSize = ESP.getSketchSize();


sketchSpace = ESP.getFreeSketchSpace();

 sketchMD5 = ESP.getSketchMD5();

   // Initialise and set the lamp
   if (lampVal != -1) {
      #if defined(LAMP_PIN)

 ledcSetup(lampChannel, pwmfreq,
pwmresolution);
         ledcAttachPin(LAMP_PIN,
 lampChannel
if (autoLamp) setLamp(0);
default value
         else setLamp(lampVal);

  #endif
   } else {

 Serial.println
("No lamp, or lamp disabled in config");
   }

   // Start the camera server
   startCameraServer(httpPort, streamPort);

   if (critERR.length() == 0) {
      Serial.printf
("\r\nCamera Ready!\r\nUse
 '%s' to connect\r\n", httpURL);
      Serial.printf
("Stream viewer available at
'%sview'\r\n", streamURL);
 Serial.printf("Raw stream URL is '%s'\r\n",
streamURL);
  #if defined(DEBUG_DEFAULT_ON)
   debugOn();
      #else
         debugOff();
      #endif
   } else {
      Serial.printf("\r\nCamera unavailable
due to initialisation errors.\r\n\r\n");
    }


    // Serial.print("\r\nThis is the 4.1
beta\r\n");


    while (Serial.available()) Serial.read();

}

void loop() {
    if (accesspoint) {
        unsigned long start = millis();
        while (millis() - start
< WIFI_WATCHDOG ) {
            delay(100);
            if (otaEnabled)
ArduinoOTA.handle();
            handleSerial();

if (captivePortal) dnsServer.processNextRequest();

   }
    } else {  static bool warned = false;
        if (WiFi.status() ==
WL_CONNECTED) {
           if (warned) {

   Serial.println("WiFi reconnected");
                warned = false;
            }

            unsigned long start = millis();
            while (millis() - start < WIFI_WATCHDOG ) {
                delay(100);
                if (otaEnabled)
ArduinoOTA.handle();
                handleSerial();



}
        } else {

            if (!warned) {
WiFi.disconnect();
Serial.println("WiFi disconnected,
retrying");
                warned = true;
            }
            WifiSetup();
        }
    }
}
```

CHAPTER - 4

TIMELINE OF THE PROPOSED WORK

Firstly, we are making a body frame using a carbon fibres sheet with the help of the CNC laser cutting method, Then we cut off the body frame with C.N.C laser cutting. For cutting this body frame we have to give an input signal to the CNC machine.

1st step :- We make this design with the help of Auto-cad software. Here time duration is **4 days**.

2nd step:-With the help of Auto-cad design we will cut the body frame. Here time duration is **1 day.**

3rd step:-In the third step, we will join all of the small cutting parts for making the body frame. Here time duration is **2 days**.

4th step :-After done the body-making part we will add all the necessary equipment to the body frame. This equipment is a circuit board, battery, motor and wheels, and sensors. Here time duration is **1 day**.

5th step :-After that we have to turn on the vehicles. For turning on the vehicles, We have to install a C-level program. Here time duration is **2 days.**

6th step :-Finally here comes the testing session whether the vehicle is working or not. For that time duration is **2 days**.

CHAPTER - 5
CONCLUSION

Modern technologies and autonomous navigation algorithms are employed in navigation systems technology to ensure accurate and safe vehicle navigation. It can be found in drones, robots, cars, boats, and other vehicles. The project begins by defining a performance problem associated with applications and ends with a prototype for an Autonomous based solution. The prototype was successfully able to navigate autonomously to a programmed destination and charge its battery using solar energy. The project begins by defining a performance problem associated with applications and ends with a prototype for an Autonomous based solution. The project begins by defining a performance problem associated with applications and ends with a prototype for an Autonomous based solution. This presentation describes the effects of an AI-based autonomous in-campus navigation system fueled by a hybrid renewable energy system. The technological revolution has reached all life activities starting from the day planning to reach communication, entertainment, industry, and transportation. Each of the previously mentioned categories gets improved in a way making human life easier and safer. In the use of automatic control, several types of research focused on automating vehicles' systems to make driving easier and safer. The availability of autonomous vehicles will avoid accidents caused by taking a late decision or lack of driving experience in such a situation.

Fig. 14 Oveerview of the autonomous driverless vehicle system.

REFERENCE

1. Baumgart, J. Whither the Driverless Car. Next City Magazine, 2013. Accessed March 3, 2014.
2. California Department of Motor Vehicles (California DMV). Autonomous Vehicles in California. Status update presentation at City of Carlsbad, June 2015. ZackaryMinshew; Adel El-Shahad," DC micro — Grid pricing and market model" IEEE Global Humanitarian Technology Conference (GHTC) Oct. 2017
3. Castro, D. The Road Ahead: The Emerging Policy Debates for IT in Vehicles. The Information Technology & Innovation Foundation, 2013. Accessed May 7, 2014.
4. Consumer Reports. The Road to Self-Driving Cars: Today's Crash-Avoidance Systems are the Mile Markers to Tomorrow's Autonomous Vehicles. Consumer Reports Magazine. 2014.Accessed April 18, 2014.
5. Government Accountability Office (GAO). Report to Congressional Requesters - Intelligent Transportation Systems: Vehicle-to-Vehicle Technologies Expected to Offer Safety Benefits, but Variety of Deployment Challenges Exist. United States Government Accountability Office, Nov.

   2013. Accessed May 7, 2014.
6. Gods mark, P. Autonomous Vehicles: Are We Ready? Focus on the Future, Self Help Counties Coalition, Santa Clara, CA, November 17, 2014. Guizot, E. How Google's Self-Driving Car Works. IEEE Spectrum. Accessed June 18, 2015.
7. Harris, M. The Unknown Start-up That Built Google's First Self-Driving Car. IEEE Spectrum.2014. Accessed May 26, 2015.
8. Heller,M. California Regulators Consider "Big Brother" Hazards of Driverless Cars: Cars that drive Themselves Could Be Pulling Out of Sci-Fi Fantasies and Tinseltown Dreams and Into Driveways and Roads Sooner than, Sending Privacy Advocates into Over-drive. Mint Press News. 2014 Accessed April 18, 2014.
9. Hennig, K. CUTR Launches Automated Vehicle Institute. University of South Florida News. 2013.Accessed April 2, 2014

INFORMATION REGARDING STUDENT(S)

STUDENT NAME	EMAIL ID	PERMANENT ADDRESS	PHONE NUMBER	LANDLINE NUMBER	PLACEMENT DETAILS	PHOTOGRAPH
JAKEY MOZAHD	20BTRCA066 @JAINUNIVERSITY.AC.IN	138/1/A MIRPUR 10 DHAKA BANGLADESH	+916360257445	N/A	N/A	
ATIQUR RAHMAN	20BTRCA0665 @JAINUNIVERSITY.AC.IN	149/2D MIRPUR 12 DHAKA BANGLADESH	+918147742393	N/A	N/A	

Photograph Along With Guide