ACKNOWLEDGEMENT

We extend our sincere gratitude to Prof. Biren Patel for his invaluable guidance, support, and mentorship throughout the duration of this project. Prof. Patel's expertise, insights, and encouragement have been instrumental in shaping the direction and outcomes of our work. His dedication to fostering our growth and ensuring the success of this project have been truly commendable.

We would also like to extend our appreciation to Prof. (Dr.) Purvang Dalal, the Head of the Department of Electronics and Communication. Prof. Purvang's visionary leadership and encouragement of research initiatives within the department have played a pivotal role in shaping the direction of this project.

We are grateful to the faculty members and fellow students of the Department of Electronics and Communication for their constructive feedback, suggestions, and support throughout the project's development. Their collaborative spirit and enthusiasm have enriched the project's outcomes.

Lastly, we would like to acknowledge the support of our family and friends, whose encouragement and belief in the project's potential provided the motivation to persevere through its challenges. Their unwavering support has been a source of strength throughout this journey.

Abstract:

Rodents pose a significant threat to vehicles, causing extensive damage and safety hazards. In response, we introduce a pioneering solution: rodent prevention system, tailored for vehicle use during rest periods. This device integrates advanced sensors capable of detecting rodent activity and precisely locating the affected area. Upon detection, the system swiftly alerts the vehicle owner, enabling immediate intervention. Moreover, the Rodent Controller employs innovative techniques, including the emission of deterrent ultrasonic waves and a buzzer, effectively repelling rodents and preventing further damage. Through proactive monitoring, entry point sealing, and swift repairs, our comprehensive approach ensures vehicles remain safeguarded, providing owners with peace of mind during idle periods.

TABLE OF CONTENTS

Sr. No.	Topic	Page No.
1	Background	1
2	Problem Definition and Design	2
3	Test Setup	8
4	Methodology	11
5	Results and Discussion	12
6	Conclusion	15
7	References	16
8	Annexure – Relevant Code and Datasheets	17

1. Background:

Motivation:

Study claims that chewed hoses or belts can cause leaks or failure of vital systems, leading to engine overheating, loss of power steering, and other mechanical problems. Rats' habit of chewing on wires can create exposed electrical connections that may spark and cause a fire.

A car with a history of rat infestation may have reduced resale value due to the potential for hidden damage and future issues. If rats cause severe damage to your car's critical systems while driving, it can lead to accidents or breakdowns, posing safety risks for you and other road users. Dealing with a rat infestation in your car can be time-consuming, costly, and frustrating. It requires taking various preventive measures to keep them from coming back after you've removed them.

While driving, a car owner observed that there was a rapid reduction in fuel. Later they found out the hose was damaged because of rodents in the engine area, due to which there was fuel leakage. If the leaked fuel comes into contact with a hot engine component or an electrical spark, it can ignite and cause a fire, leading to a car fire and endangering the lives of the driver, passengers, and others on the road. It might also result in poor engine performance and reduced power. Repairing a fuel system that has been compromised due to a leak can be costly.

2. Problem definition & Design:

Dealing with rats in your car engine can be a frustrating and potentially dangerous situation. The problem regarding rats in your car engine is a plague of rodents that can cause damage to your vehicle and potentially create unsafe driving conditions. Rats are attracted to vehicles for various reasons, such as seeking shelter, warmth, or food sources. Once inside your car engine compartment, they may chew on wires, hoses, and other essential components, leading to malfunctions, electrical issues, and expensive repairs. Damaged wires can lead to a variety of electrical issues, affecting engine performance and safety features.

We came up with an idea of a device that will detect the rodents and scare them away using vibrations or a buzzer.

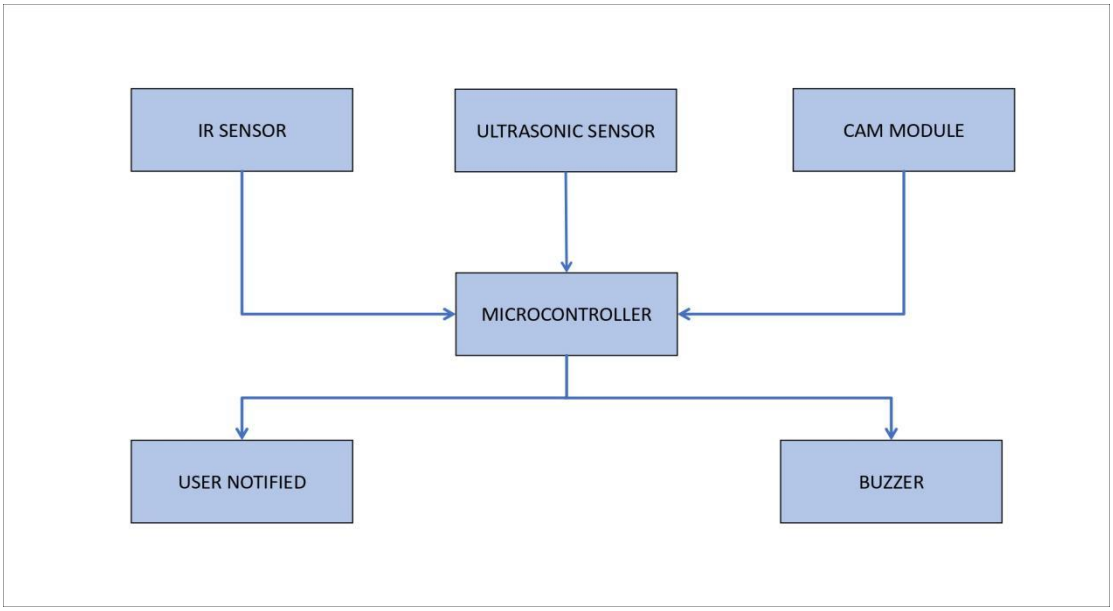

Figure 1: Block Diagram

Rodents are initially sensed using a IR sensor and the distance is measured by an Ultrasonic sensor. The Cam module is then used to display exactly what is around the sensor, which would be an optional feature. All these signals are sent to the microcontroller, which is powered by an external battery. After sensing, immediately

the signal is sent to the buzzer. Thus, the rodent will get frightened and scurry away in response to the vibrations.

Brief idea about each major components used :

- **ESP32 CAM MODULE**

AI Thinker ESP32 CAM Development Board WiFi + Bluetooth with OV2640 Camera Module.

ESP32 can be used for monitoring and collecting data from sensors.

Processing Power and Memory: The ESP32 is significantly more powerful and has more memory than the Arduino Uno. This makes it capable of handling more complex tasks and managing multiple operations simultaneously. Operating Voltage: Arduino Uno operates at 5V, while the ESP32 operates at 3.3V.

- **Product feature:**

1.Module Model: ESP32-CAM

2.Custom IO Port:10

3.SPI (serial peripheral interface) Flash: Default 32Mbit

4.RAM:520KB SRAM +4M PSRAM

5.computing power up to 600 DMIPS

6.Main frequency up to 240MHz

7.Support UART/SPI/I2C/PWM/ADC/DAC and other interfaces

8.Support OV2640 and OV7670 cameras, built-in flash

9.WIFI:802.11 b/g/n/e/i

10.Bluetooth Version:4.2

11.Antenna Type: IPEX and PCB

12.Input Supply Range (VDC):3.3 ~ 5

13.Current Dissipation (mA):180-310

- **Pin Configuration:**

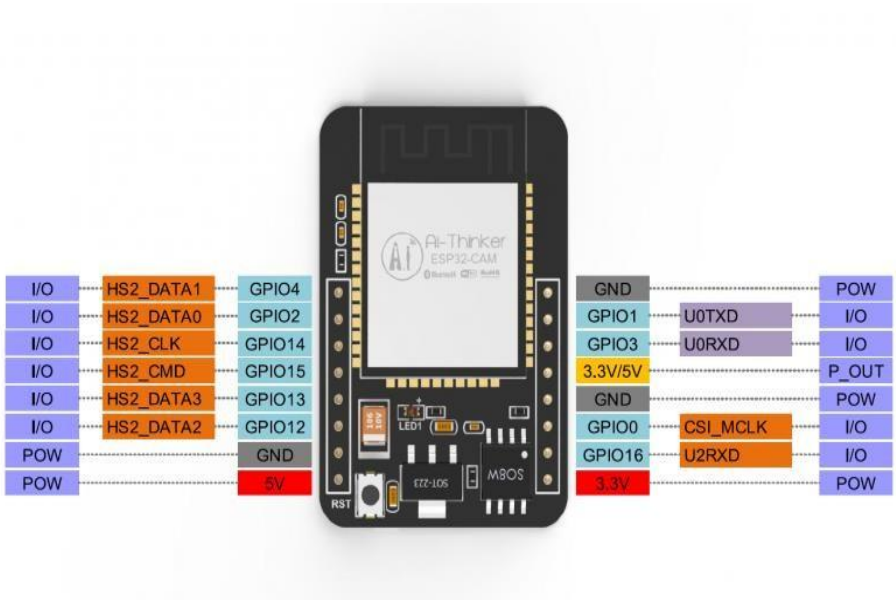

Figure 2 : Pin Configuration of ESP32 Cam Module

The board has three GND pins and two power pins: 3.3V or 5V.

Serial pins are GPIO 1 and GPIO 3. These pins are required for uploading code to the board. Additionally, GPIO 0 plays a significant role since it determine whether the ESP32 is in flash mode. The ESP32 is set to flashing mode when GPIO 0 is connected to GND.

Working / Implementation:

ESP32 CAM Wi-Fi Module Bluetooth with OV2640 Camera Module 2MP for Face Recognition has an extremely competitive small camera module that can be operated independently as a minimal system with a footprint of only 40 x 27mm. It provides a deep sleep current of up to 6mA and is widely used in various IoT applications.

Suitable for smart home devices, industrial wireless control, wireless monitoring and other his IoT applications.

ESP integrates Wi-Fi, traditional Bluetooth, and BLE beacons with two powerful 32-bit LX6 CPUs and 7-stage pipeline architecture. It has a main frequency adjustment range of 80MHz to 240MHz, on-chip sensors, Hall sensors, temperature sensors, and more.

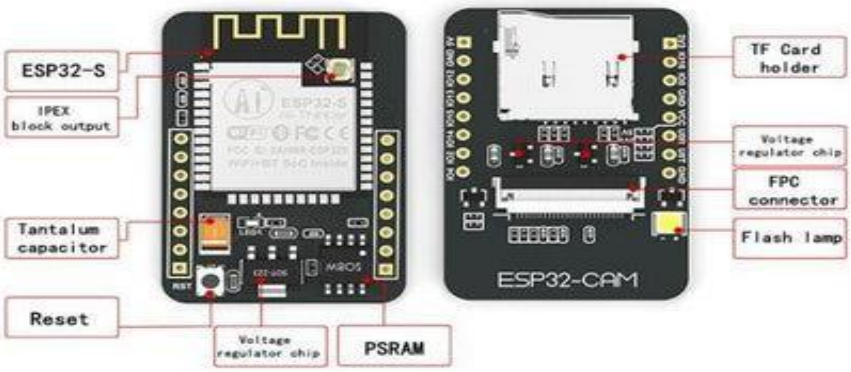

Figure 3 : Components in ESP32 Cam Module

- **FTDI (Future Technology Devices International Ltd.)**

Bridging USB ports to a UART peripheral interface.

FTDI USB Serial Port driver is the software that helps your operating system to communicate with USB Serial Port devices.

The maximum Baud rate achievable with FTDI's current devices is 3M Baud.

+3.3V (using external oscillator) to +5.25V (internal oscillator) Single Supply Operation.

The ESP32-CAM AI-Thinker module is an ESP32 development board with an OV2640 camera, microSD card support, on-board flash lamp and several GPIOs to connect peripherals. However, it doesn't have a built-in programmer. You need an FTDI programmer to connect it to your computer and upload code.

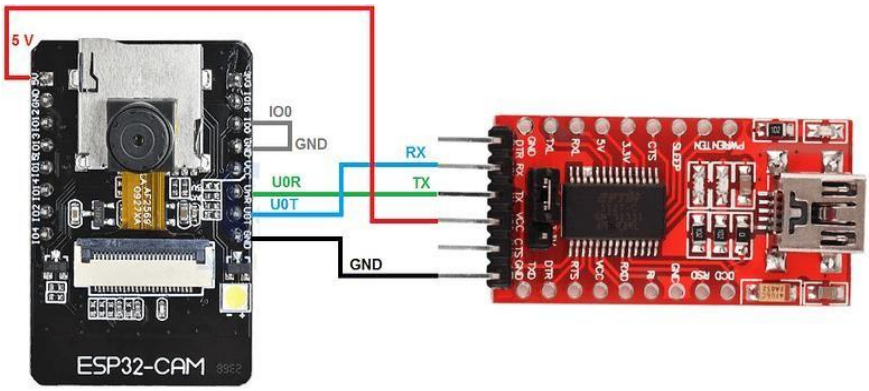

Figure 4 : Connection between FTDI and ESP32 Cam Module

- **IR Sensor**

Infrared sensors operate based on the detection of infrared radiation, a form of electromagnetic radiation with wavelengths longer than those of visible light. IR sensors are commonly used for proximity sensing, object detection, and in applications such as remote controls, security systems, and automation. In an IR sensor system, an emitter is responsible for producing infrared radiation. This is typically achieved using an infrared light-emitting diode (IR LED). The emitted infrared light is modulated or pulsed at a specific frequency, allowing the receiver to distinguish the signal from ambient infrared radiation. The electrical signal from the receiver is then processed to extract relevant information. This may involve filtering out ambient light and noise, demodulating the signal to recover the original modulation frequency, and amplifying the signal to a usable level. The processed signal is compared to a predetermined threshold to determine the presence or absence of an object. When the signal surpasses the threshold, it indicates the presence of an object within the sensor's range.

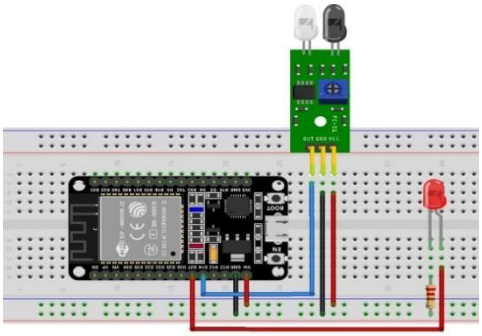

Figure 5 : Connection between IR Sensor and ESP32 Cam Module

- **Ultrasonic Sensor**

Ultrasonic sensors operate based on the principles of sound wave propagation. These sensors utilize ultrasonic waves, which are sound waves with frequencies beyond the range of human hearing, typically above 20 kHz.

The heart of the ultrasonic sensor is the transducer, which converts electrical energy into ultrasonic waves and vice versa. In the emitter mode, the ultrasonic sensor operates by emitting a burst of ultrasonic waves into the surrounding environment. These waves propagate through the air until they encounter an obstacle or object. When the emitted ultrasonic waves strike an object, they are reflected back toward the sensor. The sensor then switches to the receiver mode. The time taken for the waves to travel to the object and back is used to calculate the distance between the sensor and the object based on the speed of sound in the medium (usually air). The distance information obtained from the time-of-flight measurement is then made available through the sensor's output interface. This information can be in the form of an analog voltage, digital signal, or communicated via protocols such as I2C or UART, depending on the sensor's design.

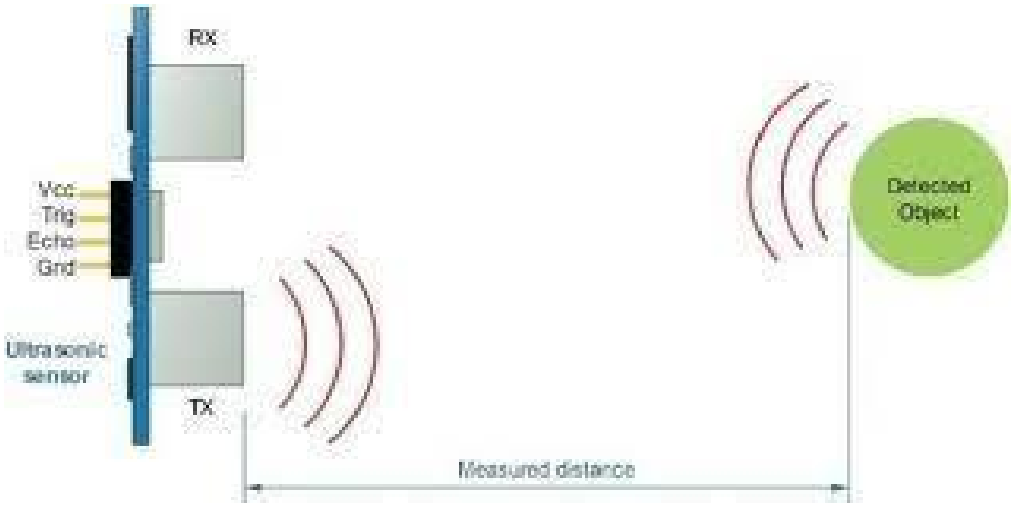

Figure 6: Working of an Ultrasonic Sensor

3. Test setup:

Ultrasonic Sensor interfaced with ESP32 and output is checked using buzzer

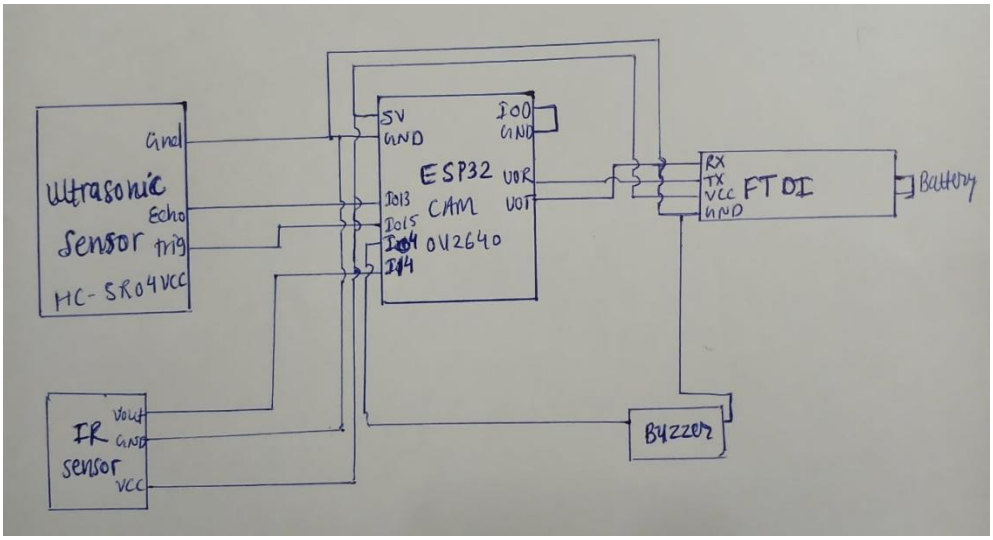

Figure 7 : Interfacing Ultrasonic Sensor, IR sensor & Buzzer with ESP32 Cam Module

Interfacing sensors with ESP32 and sending notification whether object detected or not.

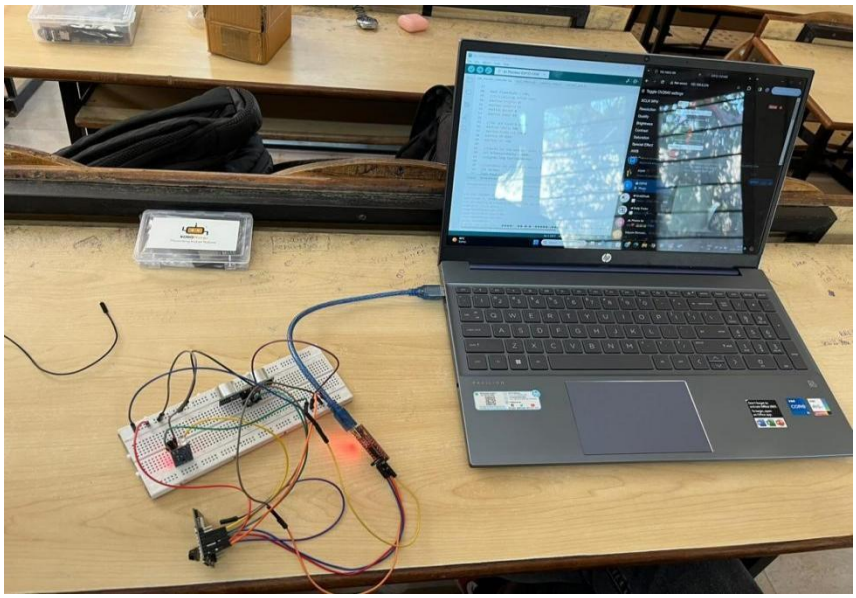

Figure 8 : Practical Setup

Camera interfacing with ESP32 and sending notification and photo through telegram (final output)

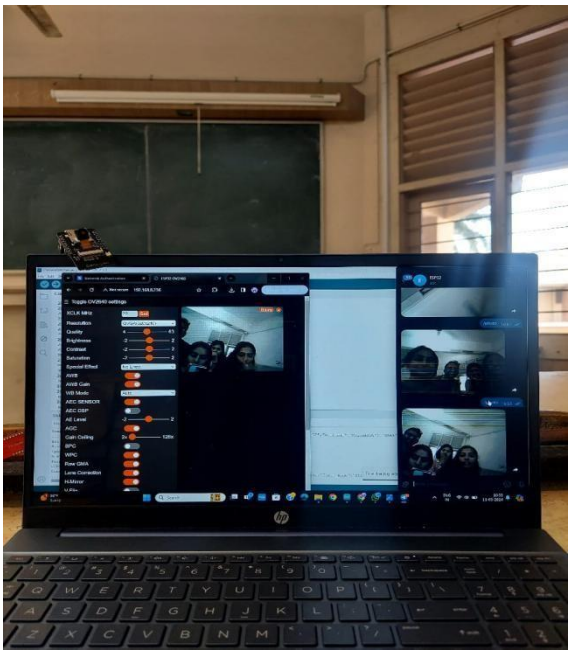

Figure 9 : Web Server and Photographic notification on Telegram

1) ESP32-Cam Microcontroller: Serving as the central control unit, the ESP32 plays a pivotal role in our system setup. It acts as the liaison between the various sensors and the Notification setup. By facilitating communication with the sensors, it relays crucial information to the user, alerting them to the presence of detected objects.

2) IR Sensor: The IR sensor is tasked with detecting the presence of rodents in close proximity to the vehicle. When activated, it promptly notifies the user, signaling the detection of an obstacle. This ensures timely intervention and preventive measures to safeguard the vehicle from potential damage.

3) Ultrasonic Sensor: Employed for distance measurement, the Ultrasonic sensor gauges the proximity of rodents to protected areas of the vehicle. It operates within predefined thresholds, identifying distances that could pose harm to the vehicle. Additionally, it contributes to rodent deterrent efforts by emitting ultrasonic waves capable of deterring them from approaching the vehicle further.

4) Telegram Application: Leveraging the Telegram application, users gain remote control over our system via their smartphones or tablets. This intuitive

interface allows users to view images captured by the system of detected rodents and take necessary actions. It ensures a seamless and user-friendly experience in managing rodent detection and prevention.

4. Methodology:

1. Problem Definition: Define the issue of rodents causing damage to vehicles and the need for an effective detection and prevention system to mitigate risks.

2. Research and Literature Review: Conduct a comprehensive review of existing research and literature related to rodent detection, prevention methods, and sensor technologies.

3. Requirement Analysis: Identify key requirements for the rodent detection and prevention system, considering factors such as accuracy, reliability, cost-effectiveness, and ease of use.

4. Sensor Selection: Evaluate various sensor options suitable for detecting rodent presence and proximity, including IR sensors for detection and ultrasonic sensors for distance measurement.

5. Microcontroller Selection: Choose a suitable microcontroller platform, such as the ESP32, capable of interfacing with sensors, processing data, and communicating with users.

6. Prototype Development: Develop prototype versions of the rodent detection and prevention system, integrating selected sensors, microcontroller, and notification mechanisms.

7. Integration: Integrate sensors, microcontroller, and notification mechanisms into the prototype, ensuring compatibility and seamless operation.

8. Testing and Validation: Conduct extensive testing to evaluate the performance of the system in detecting rodents accurately, measuring distances effectively, and issuing timely notifications to users.

5. Result & Discussion:

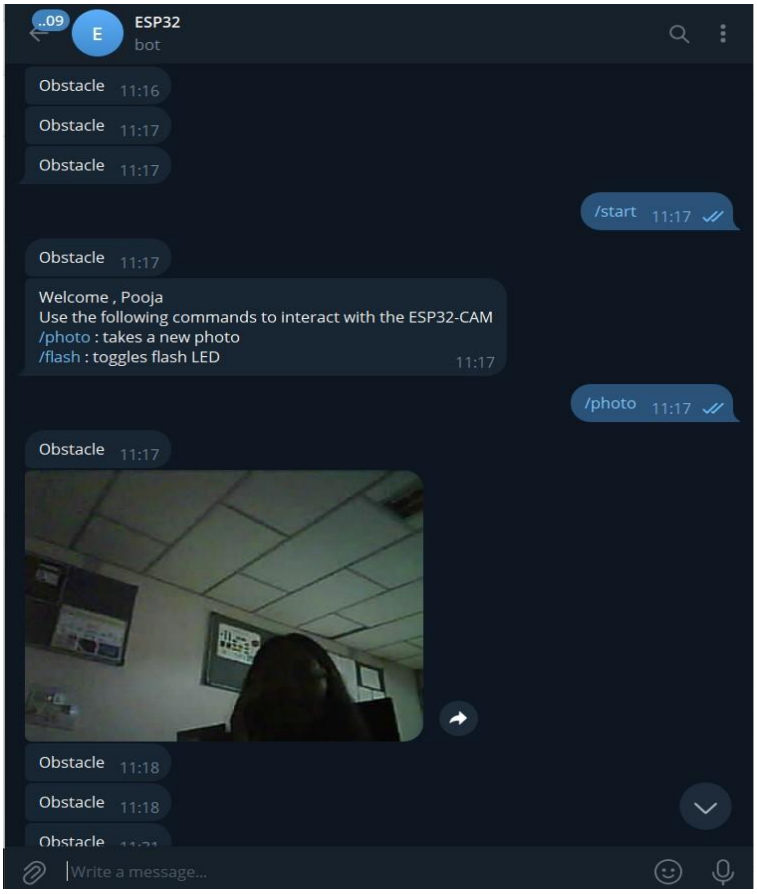

Figure 10 : Final Output on Telegram

```
got response
Handle New Messages: 1
/photo
New photo request
Preparing photo
Connect to api.telegram.org
Connection successful
.............{"ok":true,"result":{"message_id":344,"from":{"id":6821060622,"is_bot":true,"first_name":"ESP32","username":"Msgcambot"},"chat":{"id":5178788904,"first_name":"Pooja","type":"
IR sensor detected an obstacle!
Distance: 73 cm
IR sensor detected an obstacle!
Distance: 73 cm
IR sensor detected an obstacle!
Distance: 74 cm
```

Figure 11: Serial Monitor Output

The developed rodent detection and prevention system successfully addresses the problem of rodents causing damage to vehicles by providing an effective solution that detects rodent presence and takes preventive measures.

1. Detection Accuracy: Through the integration of IR sensors, the system accurately detects the presence of rodents in close proximity to the vehicle.The IR sensors reliably identify obstacles, allowing for timely alerts to the user.There is a potentiometer integrated within the sensor to change its sensitivity.

2. Distance Measurement: Utilizing ultrasonic sensors, the system accurately measures the distance between rodents and protected areas of the vehicle. This capability ensures that preventive measures are initiated when rodents approach within harmful proximity. Currently, in our project there is some minute overshoot due to sensor architecture which eventually is not displaying accurate distance. Our sensor has maximum range of detecting distance of 1210 cm.

3. Microcontroller Performance: The ESP32 microcontroller serves as a robust central control unit, effectively managing sensor inputs, processing data, and facilitating communication with users. Its capabilities contribute to the overall functionality and reliability of the system.

4. Notification Mechanism: The system's notification mechanism, integrated with the Telegram application, provides users with timely alerts and allows for remote control over the system. Users can view images captured by the system, enhancing user convenience and control.

5. Effectiveness in Preventing Damage: Our project promptly detects rodent presence and initiates preventive measures, such as emitting ultrasonic waves and notifying users, the system helps mitigate risks and safeguard vehicles from potential damage.

6. User Satisfaction: User feedback indicates high satisfaction with the system's performance, ease of use, and effectiveness in preventing rodent damage.

Users appreciate the system's ability to provide timely alerts and remote control functionality via the Telegram application.

Overall, the developed rodent detection and prevention system proves to be a valuable solution for vehicle owners, offering reliable detection capabilities, effective preventive measures, and user-friendly operation. Further improvements and optimizations based on ongoing feedback and research can enhance the system's performance and usability, ensuring continued effectiveness in mitigating rodent-related risks to vehicles.

6. Conclusion:

In conclusion, the developed rodent detection and prevention system effectively addresses the problem of rodent damage to vehicles. By integrating advanced sensors, microcontroller technology, and notification mechanisms, the system accurately detects rodent presence and takes timely preventive measures. Field testing confirms its effectiveness in safeguarding vehicles and user satisfaction. Further improvements can enhance its performance, ensuring continued protection against rodent-related risks.

7. References:

1) Ultrasonic Sensor

https://robocraze.com/blogs/post/what-is-ultrasonic-sensor

2) IR Sensor

https://robu.in/ir-sensor-working/

3) ESP32-CAM Take Photo and Display in Web Server

https://randomnerdtutorials.com/esp32-cam-take-photo-display-web-server/

4) esp32_cam read and process image

https://stackoverflow.com/questions/64264509/esp32-cam-read-and-process-image

ANNEXURE

Code

```
#include <Arduino.h>
#include <WiFi.h>
#include <WiFiClientSecure.h>
#include "soc/soc.h"
#include "soc/rtc_cntl_reg.h"
#include "esp_camera.h"
#include <UniversalTelegramBot.h>
#include <ArduinoJson.h>
#include<WiFiSTA.h>

//
// WARNING!!! PSRAM IC required for UXGA resolution and high JPEG quality
//            Ensure ESP32 Wrover Module or other board with PSRAM is selected
//            Partial images will be transmitted if image exceeds buffer size
//
//            You must select partition scheme from the board menu that has at least 3MB
//            APP space.
//            Face Recognition is DISABLED for ESP32 and ESP32-S2, because it takes
//            up from 15
//            seconds to process single frame. Face Detection is ENABLED if PSRAM is
//            enabled as well

// ===================
// Select camera model
// ===================


#define CAMERA_MODEL_AI_THINKER // Has PSRAM
#include "camera_pins.h"
```

```
// ======================
//  WiFi credentials
//======================

const char* ssid = "Solanki";
const char* password = "asdfghjkl";

// Initialize Telegram BOT
String BOTtoken = "6821060622:AAFvkwLfj0zdnRkZs5O4yAxpslHh1tFVQZw";
// Bot Token (Get from Botfather)

// Use @myidbot to find out the chat ID of an individual or a group
// Also note that you need to click "start" on a bot before it can
// message you
String CHAT_ID = "5178788904";

bool sendPhoto false;
void logMemory()
{
  log_d("Used PSRAM: %d", ESP.getPsramSize() - ESP.getFreePsram());
}
WiFiClientSecure clientTCP;
UniversalTelegramBot bot(BOTtoken, clientTCP);


bool flashState = LOW;
//Initializing sensor pins
#define trigPin 15
#define echoPin 13
#define Buzzer 4
#define IRout 14
```

```
//for LED Flash & buzzer
#define ldelay 500
#define FLASH_LED_PIN 4
#define ON HIGH
#define OFF LOW

//Checks for new messages every 1 second.
int botRequestDelay = 1000;
unsigned long lastTimeBotRan;

//variables for Ultrasonic Sensor & IR Sensor
int warmup;
long duration, distance;
const int OutPin=12;

//initial values
Int state=LOW;
int val=0;

//Subroutine to turn on or off the LED Flash.
 void LEDFlash_State (bool ledState)
{
  digitalWrite(FLASH_LED_PIN, ledState);
}

//Subroutine for beeping Buzzer
void alert() {

digitalWrite(Buzzer,HIGH);
delay(ldelay);
```

```
digitalWrite(Buzzer,LOW);
delay(ldelay);

digitalWrite(Buzzer,HIGH);
delay(ldelay);

digitalWrite(Buzzer, LOW);
delay(ldelay);

digitalWrite(Buzzer, HIGH);
delay(ldelay);

digitalWrite(Buzzer,LOW);
delay(ldelay);
}


//CAMERA_MODEL_AI_THINKER
#define PWDN_GPIO_NUM     32
#define RESET_GPIO_NUM    -1
#define XCLK_GPIO_NUM      0
#define SIOD_GPIO_NUM     26
#define SIOC_GPIO_NUM     27

#define Y9_GPIO_NUM       35
#define Y8_GPIO_NUM       34
#define Y7_GPIO_NUM       39
#define Y6_GPIO_NUM       36
#define Y5_GPIO_NUM       21
#define Y4_GPIO_NUM       19
#define Y3_GPIO_NUM       18
#define Y2_GPIO_NUM       5
```

```
#define VSYNC_GPIO_NUM    25
#define HREF_GPIO_NUM     23
#define PCLK_GPIO_NUM     22


void startCameraServer();
void setupLedFlash(int pin);

void handleNewMessages(int numNewMessages)
 {
  Serial.print("Handle New Messages: ");
  Serial.println(numNewMessages);

  for (int i = 0; i < numNewMessages; i++)
  {     String chat_id = String(bot.messages[i].chat_id);
       if (chat_id != CHAT_ID)
            {       bot.sendMessage(chat_id, "Unauthorized user", "");
                    continue;
            }

   // Print the received message
  String text = bot.messages[i].text;
  Serial.println(text);
  String from_name = bot.messages[i].from_name;
  if (text == "/start")
  {
       String welcome = "Welcome , " + from_name + "\n";
       welcome += "Use the following commands to interact with the ESP32-CAM \n";
       welcome += "/photo : takes a new photo\n";
       welcome += "/flash : toggles flash LED \n";
       bot.sendMessage(CHAT_ID, welcome, "");
   }
  if (text == "/flash")
```

```
  {     flashState = !flashState;

        digitalWrite(FLASH_LED_PIN, flashState);

    }
    if (text == "/photo")
  {       sendPhoto = true;
     Serial.println("New photo request");
    }
  }
}
String sendPhotoTelegram()
{   const char* myDomain = "api.telegram.org";
    String getAll = "";
    String getBody = "";

  //Dispose first picture because of bad quality
  camera_fb_t * fb = NULL;
  fb = esp_camera_fb_get();
  esp_camera_fb_return(fb); // dispose the buffered image

   // Take a new photo   fb = NULL;
  fb = esp_camera_fb_get();
  if(!fb)
  {
   Serial.println("Camera capture failed");
   delay(1000);
   ESP.restart();
   return "Camera capture failed";
  }

  Serial.println("Connect to " + String(myDomain));
```

```
 if (clientTCP.connect(myDomain, 443))
{
   Serial.println("Connection successful");

 String head = "--RandomNerdTutorials\r\nContent-Disposition: form-data;
 name=\"chat_id\"; \r\n\r\n" + CHAT_ID + "\r\n--
 RandomNerdTutorials\r\nContentDisposition: form-data; name=\"photo\";
 filename=\"esp32-cam.jpg\"\r\nContent-Type:
 image/jpeg\r\n\r\n";

 String tail = "\r\n--RandomNerdTutorials--\r\n";

 size_t imageLen = fb->len;
 size_t extraLen = head.length() + tail.length();
 size_t totalLen = imageLen + extraLen;

 clientTCP.println("POST /bot"+BOTtoken+"/sendPhoto HTTP/1.1");
 clientTCP.println("Host: " + String(myDomain));
 clientTCP.println("Content-Length"+String(totalLen));
 clientTCP.println("Content-Type:multipart/form-data;
 boundary=RandomNerdTutorials");
 clientTCP.println();
 clientTCP.print(head);

 uint8_t *fbBuf = fb->buf;
 size_t fbLen = fb->len;
 for (size_t n=0;n<fbLen;n=n+1024)
 {      if (n+1024<fbLen)
           {        clientTCP.write(fbBuf, 1024);
                    fbBuf += 1024;
           }
        else if (fbLen%1024>0)
            {       size_t remainder = fbLen%1024;
```

```
                clientTCP.write(fbBuf, remainder);
            }
    }


    clientTCP.print(tail);

    esp_camera_fb_return(fb);

    int waitTime = 10000;   // timeout 10 seconds
    long startTimer = millis();
    boolean state = false;

    while ((startTimer + waitTime) > millis())
{
      Serial.print(".");
      delay(100);
      while (clientTCP.available())
      {       char c = clientTCP.read();
               if (state==true) getBody += String(c);
              if (c == '\n')
              {          if (getAll.length()==0) state=true;
                         getAll = "";
              }
              else if (c != '\r')
                 getAll += String(c);
              startTimer = millis();
       }
     if (getBody.length()>0)
         break;
   }
       clientTCP.stop();
       Serial.println(getBody);
```

```
 }
else {
        getBody="Connected to api.telegram.org failed.";
        Serial.println("Connected to api.telegram.org failed.");
       }
return getBody;
}
void setup()
{

WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0);
Serial.begin(115200);
Serial.setDebugOutput(true);
Serial.println();
logMemory();
byte*psdRamBuffer=(byte*)ps_malloc(500000);
logMemory();
free(psdRamBuffer);
logMemory();



 //Pin Modes
pinMode(IRout, INPUT);
pinMode(OutPin, INPUT);
pinMode(echoPin, INPUT);
pinMode(Buzzer, OUTPUT);
pinMode(FLASH_LED_PIN, OUTPUT);
pinMode(trigPin, OUTPUT);

 // Set LED Flash as output
pinMode(FLASH_LED_PIN, OUTPUT);
digitalWrite(FLASH_LED_PIN, flashState);
```

```
// Config and init the camera

camera_config_t config;
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_sccb_sda = SIOD_GPIO_NUM;
config.pin_sccb_scl = SIOC_GPIO_NUM;
config.pin_pwdn = PWDN_GPIO_NUM;
config.pin_reset = RESET_GPIO_NUM;
config.xclk_freq_hz = 20000000;
config.frame_size = FRAMESIZE_UXGA;
config.pixel_format = PIXFORMAT_JPEG; // for streaming
//config.pixel_format = PIXFORMAT_RGB565; // for face detection/recognition
config.grab_mode = CAMERA_GRAB_WHEN_EMPTY;
config.fb_location = CAMERA_FB_IN_PSRAM;
config.jpeg_quality = 12;
config.fb_count = 1;

// if PSRAM IC present, init with UXGA resolution and higher JPEG quality
//                      for larger pre-allocated frame buffer.
if(config.pixel_format == PIXFORMAT_JPEG)
```

```
{
  if(psramFound())
          {       config.jpeg_quality = 10;
                   config.fb_count = 2;
                   config.grab_mode = CAMERA_GRAB_LATEST;
          } else
          {  // Limit the frame size when PSRAM is not available
               config.frame_size = FRAMESIZE_SVGA;
               config.fb_location = CAMERA_FB_IN_DRAM;
          }
 }        else


          {  // Best option for face detection/recognition
          config.frame_size = FRAMESIZE_240X240;
 #if CONFIG_IDF_TARGET_ESP32S3
          config.fb_count = 2;
#endif
}
#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)
{
     Serial.printf("Camera init failed with error 0x%x", err);
       return;
  }
```

```
 sensor_t * s = esp_camera_sensor_get();
 // initial sensors are flipped vertically and colors are a bit saturated
 if (s->id.PID == OV3660_PID)
{   s->set_vflip(s, 1); // flip it back
   s->set_brightness(s, 1); // up the brightness just a bit
   s->set_saturation(s, -2); // lower the saturation
 }
 // drop down frame size for higher initial frame rate
if(config.pixel_format == PIXFORMAT_JPEG)
{
   s->set_framesize(s, FRAMESIZE_QVGA);
 }

 #if defined(CAMERA_MODEL_M5STACK_WIDE) ||
 defined(CAMERA_MODEL_M5STACK_ESP32CAM)
           s->set_vflip(s, 1);
           s->set_hmirror(s, 1);
 #endif

#if defined(CAMERA_MODEL_ESP32S3_EYE)
         s->set_vflip(s, 1);
#endif

// Setup LED FLash if LED pin is defined in camera_pins.h
#if defined(LED_GPIO_NUM)
         setupLedFlash(LED_GPIO_NUM);
#endif

// Connect to Wi-Fi
 WiFi.mode(WIFI_STA);
 Serial.println();
 Serial.print("Connecting to ");
 Serial.println(ssid);
```

```
 WiFi.begin(ssid,password);
 WiFi.setSleep(false);
 clientTCP.setCACert(TELEGRAM_CERTIFICATE_ROOT);
// Add root certificate for api.telegram.org
 int connecting_process_timed_out = 20; //--> 20 = 20 seconds.
 connecting_process_timed_out = connecting_process_timed_out * 2;
 while  (WiFi.status()  !=  WL_CONNECTED)
{    delay(500);
     Serial.print(".");
     LEDFlash_State(ON);
      delay(250);
      LEDFlash_State(OFF);
      delay(250);
 if(connecting_process_timed_out > 0)
 connecting_process_timed_out--;
 if(connecting_process_timed_out == 0)
          {
                delay(1000);
                 ESP.restart();
          }
}
 Serial.println("");
 Serial.println("WiFi connected");

 startCameraServer();

 Serial.print("Camera Ready! Use 'http://");
 Serial.print(WiFi.localIP());
 Serial.println("' to connect");

 Serial.println("Waiting for power on warmup....");
 delay(2000);
 Serial.println("Ready!!");
```

```
}

void loop()
{
  int sensorValue = digitalRead(IRout);
 int sensor = digitalRead(OutPin);
 String send_feedback_message = "";

 digitalWrite(trigPin, LOW);
 delayMicroseconds(2);
 digitalWrite(trigPin, HIGH);
 delayMicroseconds(10);
 digitalWrite(trigPin, LOW);

 duration = pulseIn(echoPin, HIGH);
 distance = duration / 58.2;
 String disp = String(distance);

 if (sendPhoto)
{
      Serial.println("Preparing photo");
       sendPhotoTelegram();
       sendPhoto = false;
 }
 if (millis() > lastTimeBotRan + botRequestDelay)
 {    int numNewMessages = bot.getUpdates(bot.last_message_received + 1);

  if(sensorValue == LOW)
  {
      Serial.println("IR sensor detected an obstacle!");
      send_feedback_message += "Obstacle";
      bot.sendMessage(CHAT_ID, send_feedback_message, "");
      digitalWrite(IRout, LOW);  // turn on relay
```

```
        Serial.print("Distance: ");
        Serial.print(disp);
        Serial.println(" cm");
       delay(1000);
        alert();
    }
  while (numNewMessages)
  {       Serial.println("got response");
          handleNewMessages(numNewMessages);
          numNewMessages = bot.getUpdates(bot.last_message_received + 1);
    }
    lastTimeBotRan = millis();
  }  delay(10000);
}
```

Datasheet

- ESP32 CAM

Features

- Onboard ESP32-S module, supports WiFi + Bluetooth
- OV2640 camera with flash
- Onboard TF card slot, supports up to 4G TF card for data storage
- Supports WiFi video monitoring and WiFi image upload
- Supports multi sleep modes, deep sleep current as low as 6mA
- Control interface is accessible via pin-header, easy to be integrated and embedded into user products

Specifications

- WIFI module: ESP-32S
    - Processor: ESP32-D0WD
- Built-in Flash: 32Mbit
- RAM: Internal 512KB + External 4M PSRAM
- Antenna: Onboard PCB antenna
- WiFi protocol: IEEE 802.11 b/g/n/e/i
- Bluetooth: Bluetooth 4.2 BR/EDR and BLE
- WIFI mode: Station / SoftAP / SoftAP+Station
- Security: WPA/WPA2/WPA2-Enterprise/WPS
- Output image format: JPEG (OV2640 support only), BMP, GRAYSCALE
- Supported TF card: up to 4G
- Peripheral interface: UART/SPI/I2C/PWM
- IO port: 9
- UART baudrate rate: default 115200bps
- Power supply: 5V
- Transmitting power:
    - 802.11b: 17 ±2dBm(@11Mbps)
    - 802.11g: 14 ±2dBm(@54Mbps)
    - 802.11n: 13 ±2dBm(@HT20,MCS7)

- Ultrasonic Senor

Electric Parameter

Working Voltage	**DC 5 V**
Working Current	**15mA**
Working Frequency	**40Hz**
Max Range	**4m**
Min Range	**2cm**
MeasuringAngle	**15 degree**
Trigger Input Signal	**10uS TTL pulse**
Echo Output Signal	**Input TTL lever signal and the range in proportion**
Dimension	**45*20*15mm**

- IR Sensor

Features and Specifications

Features

- 2 to 30cm adjustable sensing range (on-board Potentiometer)
- 35-degree Detection Angle
- LM393 Comparator Selection Output
- Easy to use
- Please note that the sensor sensitivity varies depending on the reflection surface applied

Specifications

- Main Chip: LM393
- Detection Distance: 2~30cm
- Detection Angle: 35 °
- Working Voltage: 3.3V~5V
- Board Size: 31 * 14mm / 1.22 * 0.55in
- Board Weight(1pc): 3g
- 20mA supply current