Summer Vocational Training Report

on

Gesture Controlled Smart Surveillance Vehicle

Submitted By
Jagrit
B. Tech (III[rd] Year)
Electronics & Communication Engineering
Deenbandhu Chhotu Ram University
of Science &Technology

Training Period

(22 August 2022 to 18 September 2022)

Under the Guidance of

Ms. Preeti Verma, Scientist 'D'

Mr. Bhagwan Jee Mishra, Technical Officer 'C'

Centre for Fire, Explosive and Environment Safety (CFEES)
Defence Research & Development Organisation
Ministry of Defence, Government of India
Timarpur, Delhi, 110054

Assessment of guide

This is to certify that the project compiled by **Mr. Jagrit** which is entitled "**Gesture Controlled Smart Surveillance Vehicle**" is an organised work carried by him under our supervision and guidance during the period 22 August 2022 to 18 September 2022.This report of 42 pages does not contain any confidential information pertaining to DRDO. We wish him for a bright future.

Ms. Preeti Verma, Sc 'D'
Project GUIDE

Mr. Bhagwan Jee Mishra, ,Technical Officer 'C'
Project GUIDE

ACKNOWLEDGEMENT

Centre for Fire, Explosive and Environment Safety (CFEES) is one of the premier establishments of Defence Research & Development Organization (DRDO). This establishment is committed to provide the users state-of-the-art product & services to its customers in areas of explosive, fire and environmental safety through research and development, innovation, team work and following up the same by continual improvement based on user's perception.

I am highly obliged to **Shri Rajiv Narang,** Director CFEES, Delhi for allowing me to associate with this esteemed establishment as a summer trainee in 'Aircraft protection Lab' for a 4-week period from 22 August to 18 September 2022.

Most importantly, I express my sincere thanks to **Ms. Preeti Verma, Sc 'D'** for her unflagging guidance throughout the progress of this project as well as valuable contribution in the preparation and compilation of the text. I am thankful to all those who have helped me for the successful completion of my training at CFEES, Delhi

Further, I extend my heartfelt gratitude to **Mr. Bhagwan Jee Mishra, Technical Officer 'C'**, For entrusting me with a project on "Computer Vision". Working in this project has certainly been a good learning experience and has reinforced my knowledge of smart machines to a great deal.

Jagrit

B. Tech (IIIrd Year)

INDEX

Scope of Project

This Project (Gesture Controlled Smart surveillance vehicle) is an automated vehicle which can detect obstacles in front of it and can accordingly take a turn. It also has a mounted camera which detects the unrecognised faces and sends the signal to the administrator. Upon getting a specific response from the administrator. It also has the ability to aim on the unrecognised face.

Components Used

In Transmitter

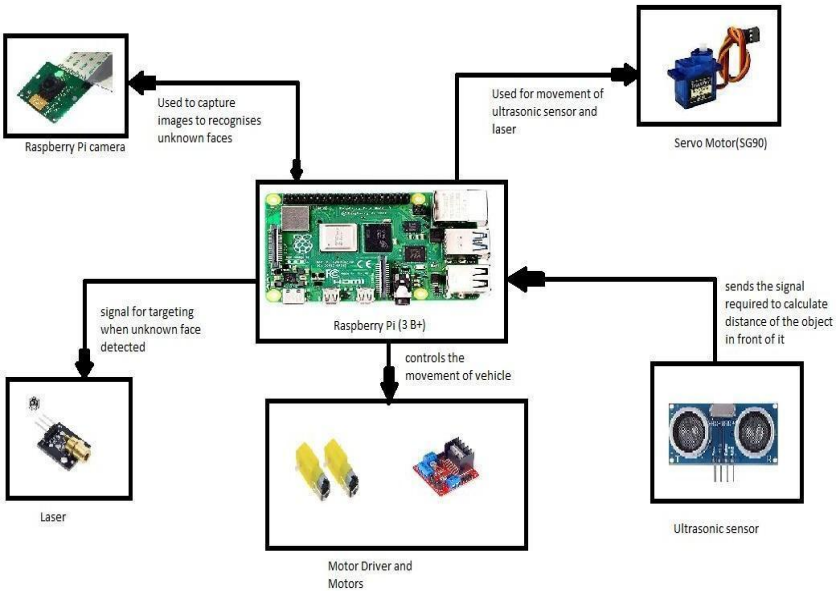
- Microcontroller (ESP32)
- Accelerometer (ADXL345)
- Push Button

In Receiver

- Raspberry Pi (3 B+)
- Raspberry Camera
- Servo Motor (SG90)
- DC Motor
- Motor Driver (L298N)
- Ultrasonic Sensor
- Laser

Hardware Description

Block Diagram of Transmitter



Block Diagram of Receiver

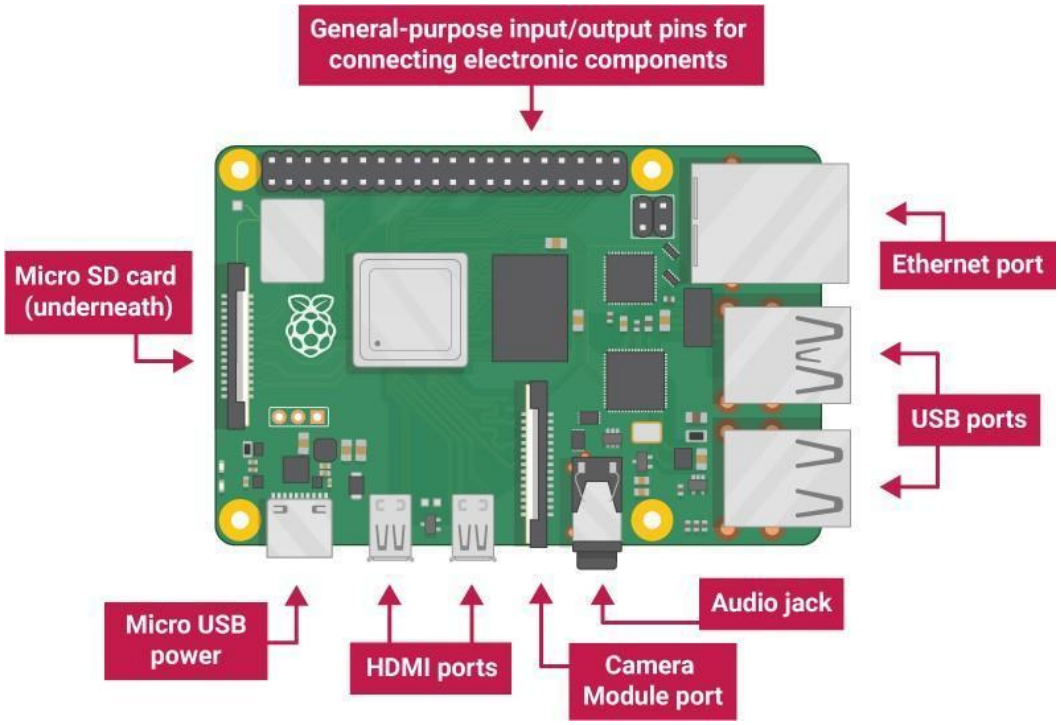
Receiver

Receiver uses ESP NOW protocol and works on 2.4 GHz frequency to receive Signal from Transmitter.

ESP NOW Protocol is generally used for communication between ESP devices. It is used for fast communication between ESP devices.

Specification of Component

Raspberry Pi:



Here we have used Raspberry pi 3 Model B+ and its Processor runs on 1.4 GHz frequency.

Raspberry Pi is a small computer generally used for IOT projects to control many components according to desired manner.

Specifications Of Raspberry Pi

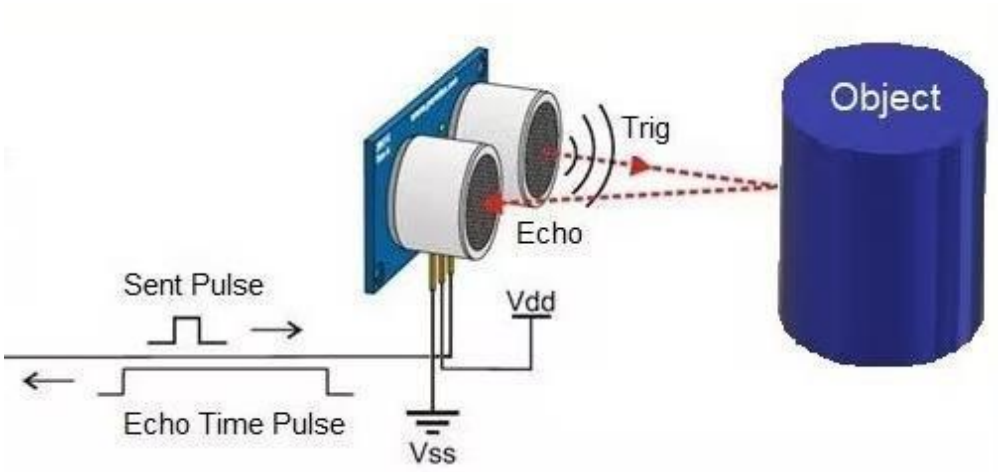
Processor	**:**	Broadcom BCM2837B0 ,Cortex-A53
Memory	**:**	1 GB
Access	**:**	40 Pin GPIO
Video & Sound	**:**	1x Full Size HDMI MIPI Display Port MIPI CSI Camera Port 4 pole Stereo output and composite video port
SD Card Support	**:**	Micro SD format for loading operating system and data storage
INPUT Power	**:**	5V/2.5A DC via micro USB Connector 5V DC via GPIO Power over Ethernet
Environment	**:**	Operating Temperature 0-50^0 C

Ultrasonic Sensor

Ultrasonic Sensor is a small sensor based on the reflection of sound from objects.

Working

For detecting a Signal we send a signal using Trig pin and then receive it back using Echo pin and then calculate the time taken between the sending and receiving.
Then We multiply it by speed of sound which is 343m/s in Air.
Then the result is divided by 2 to get the net Distance between object and sensor.



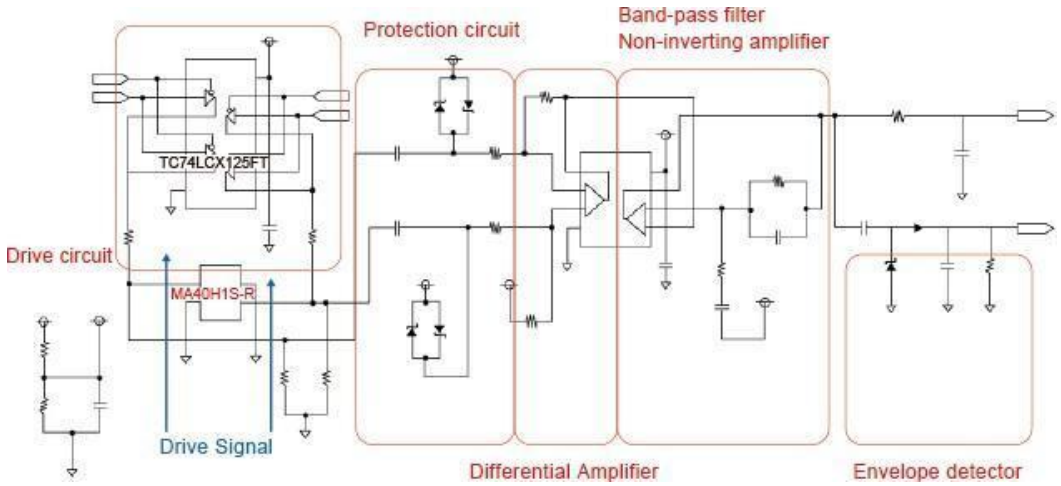
Formula Used

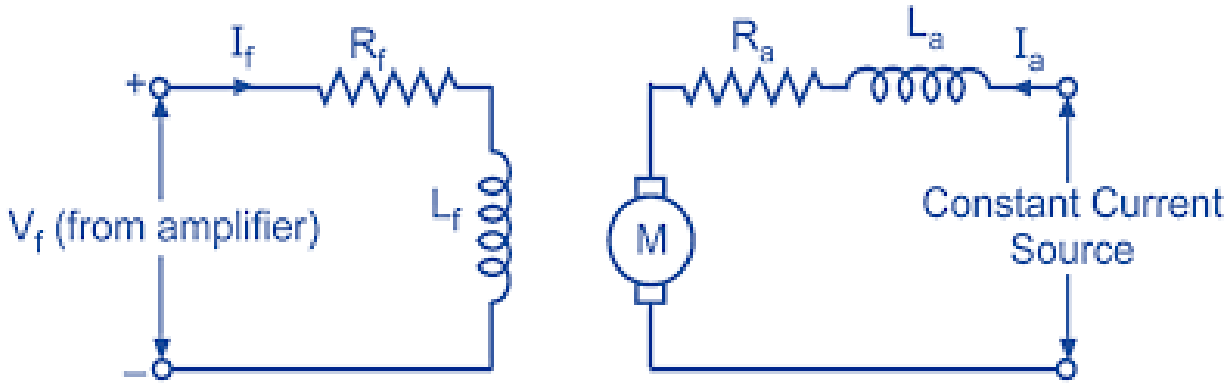
Distance between Sensor and Object = Speed of Sound in air * Time Taken between trig & echo /2

Specifications

Working Voltage	**DC 5 V**
Working Current	**15mA**
Working Frequency	**40 kHz**
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**

Internal diagram

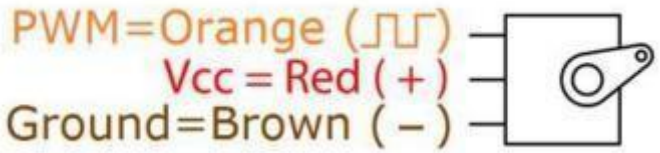




Servo Motor

A servo motor is an electromechanical device that **produces torque and velocity based on the supplied current and voltage**.
Here We have used SG90 Servo Motor. It is a tiny high output power. It can rotate approx. 180 degrees (90 degree in each direction)

Pin Configuration



Specifications

Time Period : 0.1 sec

Torque : 2.5 kg-cm

Weight : 14.7 g

Voltage : 4.8 - 6 volts

Raspberry pi Cam

Raspberry pi cam is a small camera which is connected to Raspberry pi using CSI Port.

Specifications

- Fully Compatible with Both the Model A and Model B Raspberry Pi
- 5MP Omnivision Camera Module
- Still Picture Resolution: 2592 x 1944
- Video: Supports 1080p @ 30fps, 720p @ 60fps and 640x480p 60/90 Recording
- 15-pin MIPI Camera Serial Interface - Plugs directly into the Raspberry Pi Board
- Size: 20 x 25 x 9mm
- Weight 3g

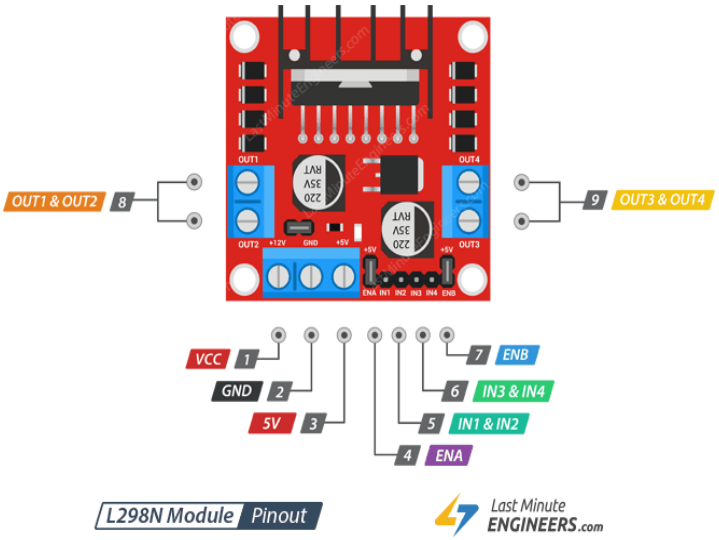
DC Motor

Here used to move the vehicle and connected through motor driver to get sufficient supply.

Specifications

Voltage	:	12V DC
Torque	:	1200 N.m
Speed	:	24RPM
Current	:	0.9A
Gear Ratio	:	264:1

Motor Driver

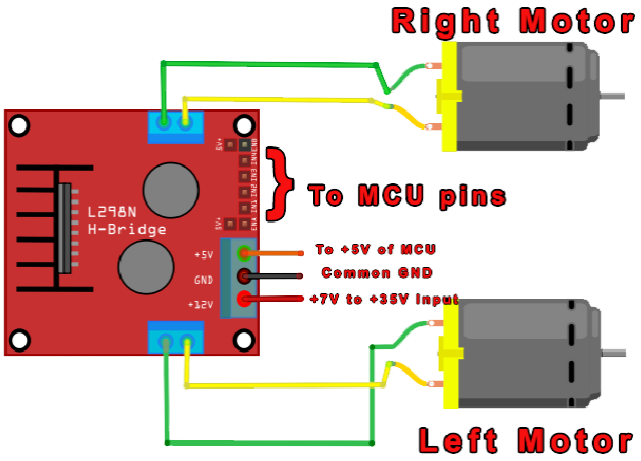


Here we have used L298N Motor driver. It is a high power motor driver module for drivind DC and stepper motors.
It consists of L298 IC and 7805 5V Voltage regulator.

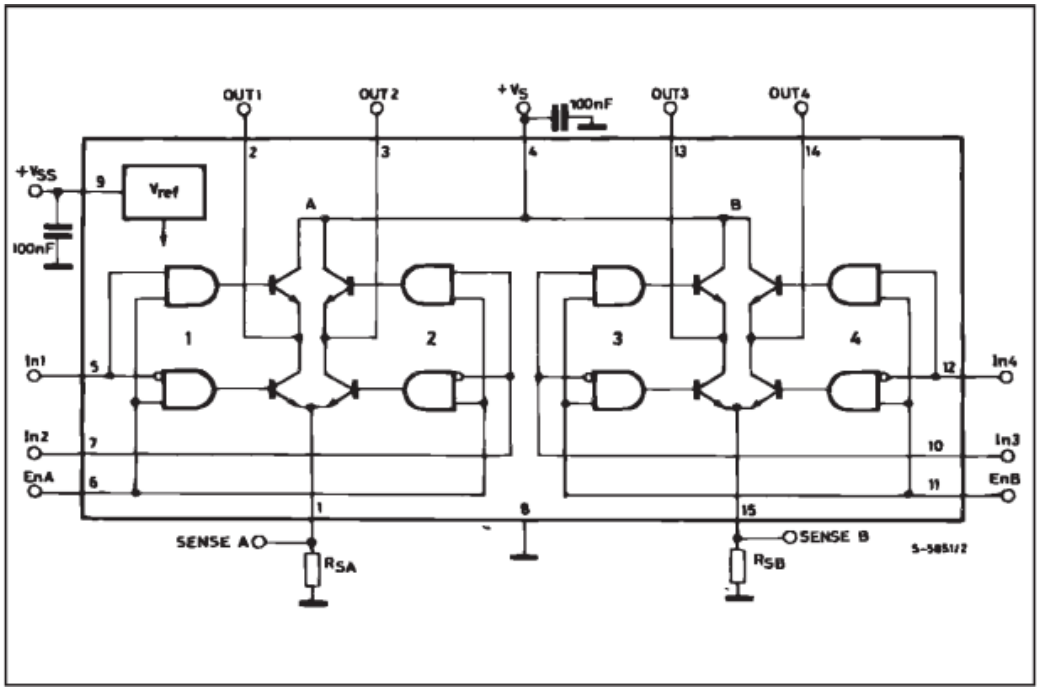
Specifications

- Driver Model: L298N 2A
- Driver Chip: Double H Bridge L298N (Reference no. 3)
- Motor Supply Voltage (Maximum): 46V
- Motor Supply Current (Maximum): 2A
- Logic Voltage: 5V
- Driver Voltage: 5-35V
- Driver Current:2A
- Logical Current:0-36mA
- Maximum Power (W): 25W
- Current Sense for each motor
- Heat sink for better performance
- Power-On LED indicator

L298 circuit with motors



L298 Internal Diagram

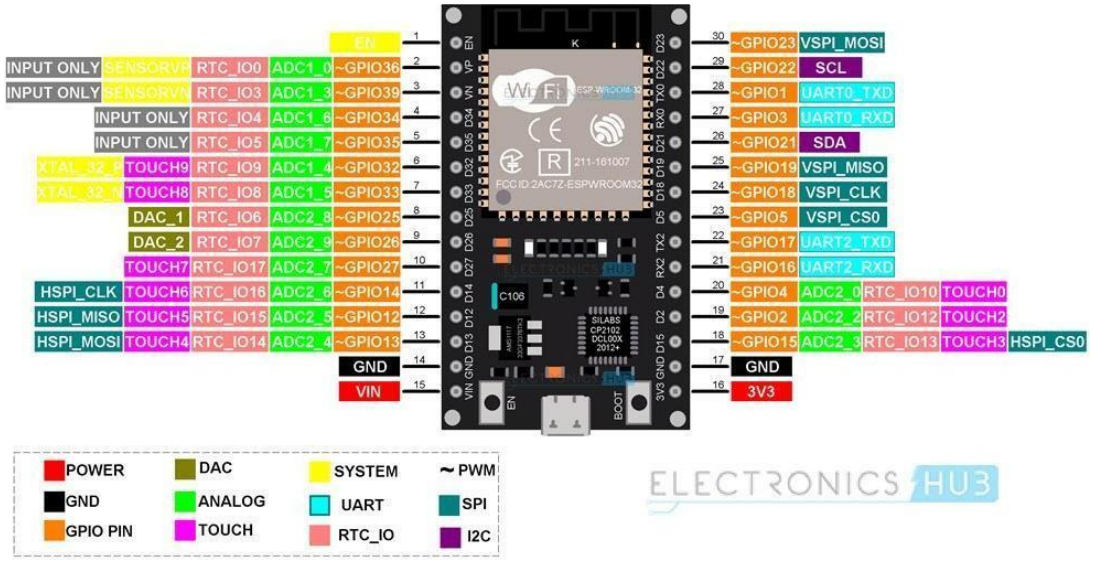


Laser

Specifications

Laser Wavelength :	650nm(red)
Supply Voltage :	5 V DC
Light Power :	<5mW
Operating Current :	<40mA
Operating temperature :	-36^0C~65^0C
Shell Material :	Brass
Weight :	1g

Microcontroller



Here We have used ESP32 module as MicroController
For transmission We are using Esp NOW protocol on 2.4GHz Frequency

Specifications

- Microcontroller: Tensilica 32-bit RISC CPU Xtensa LX106
- Operating Voltage: 3.3V
- Input Voltage: 7-12V
- Digital I/O Pins (DIO): 16
- Analog Input Pins (ADC): 1
- UARTs: 1
- SPIs: 1
- I2Cs: 1
- Flash Memory: 4 MB
- SRAM: 64 KB
- Clock Speed: 80 MHz
- USB-TTL based on CP2102 is included onboard, Enabling Plug n Play
- PCB Antenna
- Small Sized module to fit smartly inside your IoT projects

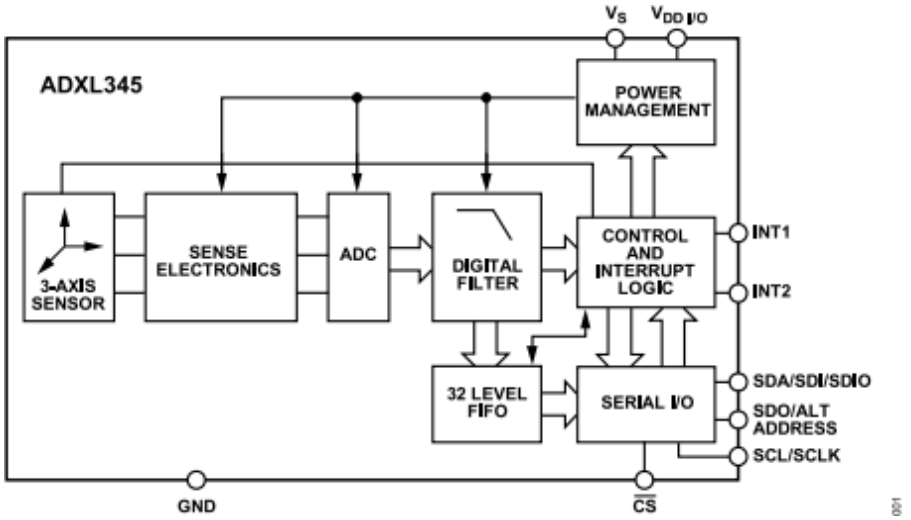
Accelerometer

Here we have used ADXL345 Accelerometer. We have used Accelerometer to detect hand gestures and control laser accordingly.

Specifications

- 3-Axis, ±2 g/±4 g/±8 g/±16 g Digital Accelerometer
- Ultralow power: as low as 23 µA in measurement mode and 0.1 µA in standby mode at VS = 2.5 V (typical)
- Power consumption scales automatically with bandwidth
- Embedded memory management system with FIFO technology
- Single tap/double tap detection
- Activity/inactivity monitoring
- Free-fall detection
- Supply voltage range: 2.0 V to 3.6 V
- I/O voltage range: 1.7 V to VS
- `Wide temperature range (−40°C to +85°C)`

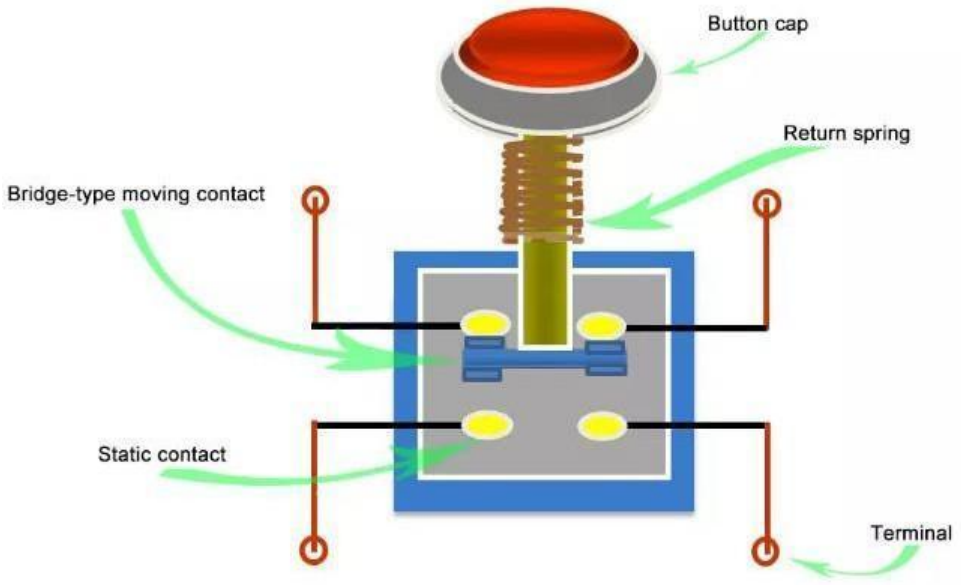
Functional Block Diagram



Push Button

Used for sending the signal for turning on laser

Internal Diagram

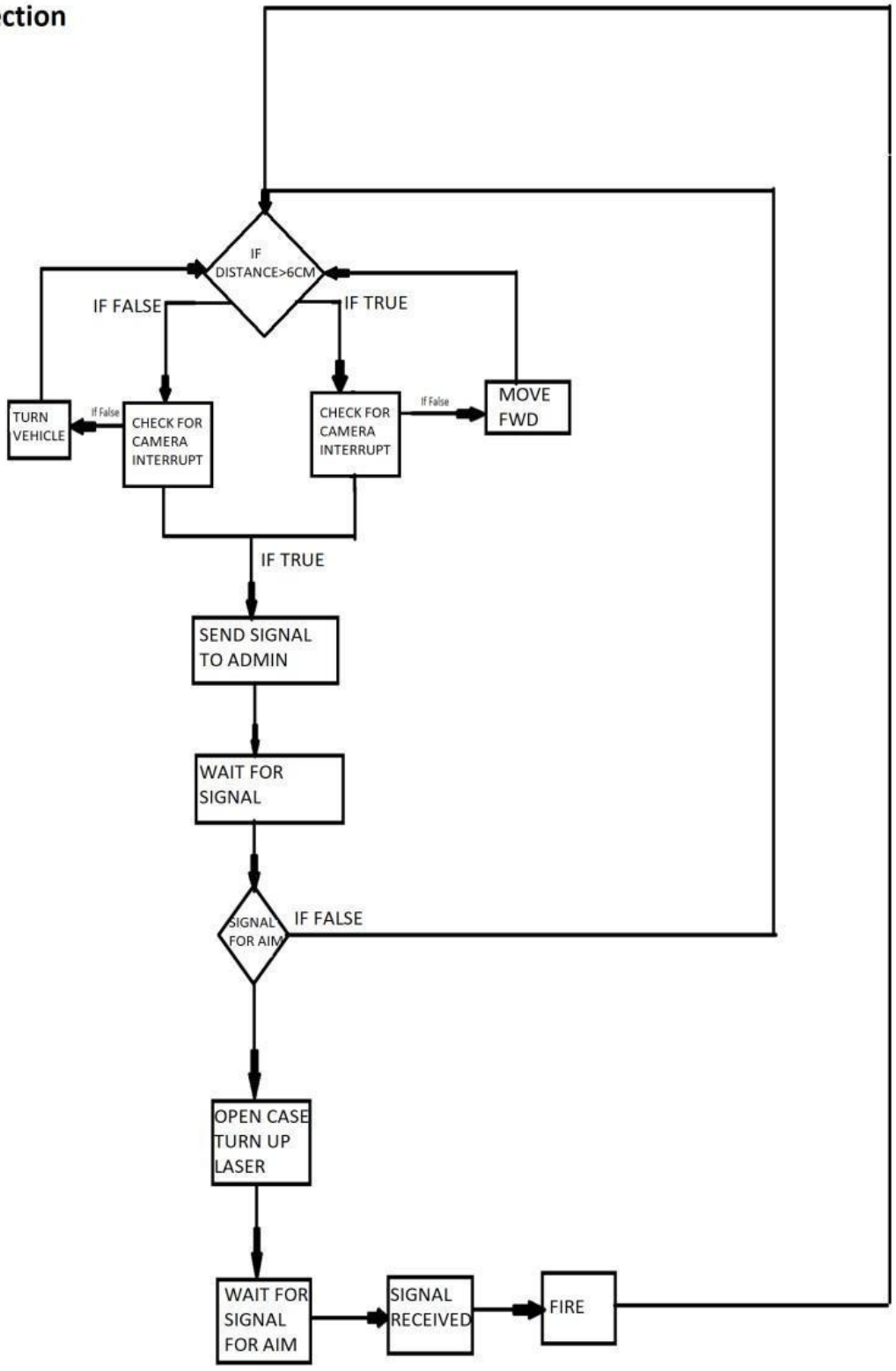


Specifications

- Power Rating: MAX 50mA 24V DC
- Insulation Resistance: 100 Mohm at 100v
- Operating Force: 2.55±0.69 N
- Operating Temperature Range: -20 to +70 ℃
- Storage Temperature Range: -20 to +70 ℃

Software Section

Flow Chart



Algorithm

Start

- Check for the distance of obstacle in front of vehicle
- If (distance <6cm)
  - Check for the interrupt from Camera
  - If Interrupt is not there
    - Turn Vehicle till Distance becomes >6cm
  - If interrupt is there
    - Go to Aim Section
- If (distance>6cm)
  - Check for the interrupt from Camera
  - If Interrupt is not there
    - Move Forward
  - If interrupt is there
    - Go to Aim Section

Aim Section:

- Sends Signal to Admin about the event
- Wait for Signal From Admin
- If Admin sends Signal for Aim
  - Open the Case
  - Wait for Signal from Admin for Aiming
  - Fire after Receivng signal for aiming and shooting
  - Go to Start
- Else
  - Go to Start

Software Code

Code for object Avoidance

```
import RPi.GPIO as GPIO                    #Import GPIO library
import time

#Import time library
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)                      # programming the GPIO by BCM
pin numbers
 TRIG = 17
ECHO = 27
TRIG1 = 18
ECHO1 = 28
led = 22
m11=16
m12=12
m21=21
m22=20

GPIO.setup(TRIG,GPIO.OUT)                   # initialize GPIO Pin as outputs
GPIO.setup(ECHO,GPIO.IN)                   # initialize GPIO Pin as input
GPIO.setup(TRIG1,GPIO.OUT)                    # initialize GPIO Pin as
outputs GPIO.setup(ECHO1,GPIO.IN)          # initialize GPIO Pin as input

GPIO.setup(led,GPIO.OUT)

GPIO.setup(m11,GPIO.OUT)
GPIO.setup(m12,GPIO.OUT)
GPIO.setup(m21,GPIO.OUT)
GPIO.setup(m22,GPIO.OUT)

GPIO.output(led, 1)

time.sleep(5)

def stop():
print "stop"
GPIO.output(m11, 0)
GPIO.output(m12, 0)
GPIO.output(m21, 0)
GPIO.output(m22, 0)
```

```
def forward():
GPIO.output(m11, 1)
GPIO.output(m12, 0)
GPIO.output(m21, 1)
GPIO.output(m22, 0)
print "Forward"

def back():
GPIO.output(m11, 0)
GPIO.output(m12, 1)
GPIO.output(m21, 0)
GPIO.output(m22, 1)
print "back"

def left():
GPIO.output(m11, 0)
GPIO.output(m12, 0)
GPIO.output(m21, 1)
GPIO.output(m22, 0) print
"left"

def right():
GPIO.output(m11, 1)
GPIO.output(m12, 0)
GPIO.output(m21, 0)
GPIO.output(m22, 0) print
"right"

stop() count=0
while True:
i=0 avgDistance=0
for i in range(5):
GPIO.output(TRIG, False)                  #Set TRIG as
LOW time.sleep(0.1)                       #Delay

GPIO.output(TRIG, True)                   #Set TRIG as HIGH
time.sleep(0.00001)                       #Delay of 0.00001
seconds GPIO.output(TRIG, False)          #Set TRIG as LOW

while GPIO.input(ECHO)==0:                #Check whether the ECHO is LOW
GPIO.output(led, False)
pulse_start = time.time()

while GPIO.input(ECHO)==1:                #Check whether the ECHO is
HIGH GPIO.output(led, False)
pulse_end = time.time()
pulse_duration = pulse_end - pulse_start #time to get back the pulse to sensor
distance = pulse_duration * 17150    #Multiply pulse duration by 17150 (34300/2) to get distance
distance = round(distance,2)                  #Round to two decimal
```

```
points avgDistance=avgDistance+distance

avgDistance=avgDistance/5
print avgDistance
flag=0
if avgDistance < 6:             #Check whether the distance is within 15 cm
range count=count+1
stop() time.sleep(1)
back()
time.sleep(1.5)
if (count%3 ==1) & (flag==0):
right()
flag=1 else:
left() flag=0
time.sleep(1.5) stop()
time.sleep(1)
else:
forward()
flag=0
```

Code for Face Recognition

Code for making dataset:

```
import cv2
import os
 # set video height
faceCascade =
cv2.CascadeClassifier('Haarcascades/haarcascade_frontalface_default.xml')
# For each person, enter one numeric face id
face_id = input('\n enter user id end press <return> ==> ')
print("\n [INFO] Initializing face capture. Look the camera and wait ...")
cap = cv2.VideoCapture(0)
# Initialize individual sampling face count
count = 0
while True:
    ret, img = cap.read()
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = faceCascade.detectMultiScale(img)
    print(faces)
    for (x,y,w,h) in faces:
        print("Jagrit")
        cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
        count+=1
        cv2.imwrite("Jagrit2/User." + str(face_id) + '.' +
                    str(count) + ".jpg", img[y:y+h,x:x+w])
        roi_gray = gray[y:y+h, x:x+w]
        roi_color = img[y:y+h, x:x+w]
    print("Jagrit")
    k = cv2.waitKey(30) & 0xff
    if k == 27: # press 'ESC' to quit
        break
    elif count >=10:
        break
cap.release()
cv2.destroyAllWindows()
```

Code for training Model

```
import cv2
import numpy as np
from PIL import Image
import os
# Path for face image database
path = 'trainit'
recognizer = cv2.face.LBPHFaceRecognizer_create()
detector =
cv2.CascadeClassifier("Haarcascades/haarcascade_frontalface_default.xml");
# function to get the images and label data
def getImagesAndLabels(path):
    imagePaths = [os.path.join(path,f) for f in os.listdir(path)]
    faceSamples=[]
    ids = []
    for imagePath in imagePaths:
        PIL_img = Image.open(imagePath).convert('L') # grayscale
        img_numpy = np.array(PIL_img,'uint8')
        id = int(os.path.split(imagePath)[-1].split(".")[1])
        faces = detector.detectMultiScale(img_numpy)
        for (x,y,w,h) in faces:
            faceSamples.append(img_numpy[y:y+h,x:x+w])
            ids.append(id)
    return faceSamples,ids
print ("\n [INFO] Training faces. It will take a few seconds. Wait ...")
faces,ids = getImagesAndLabels(path)
recognizer.train(faces, np.array(ids))
# Save the model into trainer/trainer.yml
recognizer.write('trainer/trainer.yml')
# Print the numer of faces trained and end program
print("\n [INFO] {0} faces trained. Exiting Program".format(len(np.unique(ids))))
```

Code for Recognising face

```
import cv2
import numpy as
np import os
import whatsapp
import io
import picamera
import logging
import socketserver
from threading import Condition
from http import server

PAGE="""\
<html>
<head>
<title>Raspberry Pi - Surveillance Camera</title>
</head>
<body>
<center><h1>Raspberry Pi - Surveillance Camera</h1></center>
<center><img src="stream.mjpg" width="640" height="480"></center>
</body>
</html>
"""

class StreamingOutput(object):
    def init (self):
        self.frame = None
        self.buffer = io.BytesIO()
        self.condition = Condition()

    def write(self, buf):
        if buf.startswith(b'\xff\xd8'):
            # New frame, copy the existing buffer's content and notify all
            # clients it's available
            self.buffer.truncate()
            with self.condition:
                self.frame = self.buffer.getvalue()
                self.condition.notify_all()
            self.buffer.seek(0)
        return self.buffer.write(buf)

class StreamingHandler(server.BaseHTTPRequestHandler):
    def do_GET(self):
        if self.path == '/':
            self.send_response(301)
```

```
            self.send_header('Location', '/index.html')
            self.end_headers()
        elif self.path == '/index.html':
            content = PAGE.encode('utf-8')
            self.send_response(200)
            self.send_header('Content-Type', 'text/html')
            self.send_header('Content-Length', len(content))
            self.end_headers()
            self.wfile.write(content)
        elif self.path == '/stream.mjpg':
            self.send_response(200)
            self.send_header('Age', 0)
            self.send_header('Cache-Control', 'no-cache, private')
            self.send_header('Pragma', 'no-cache')
            self.send_header('Content-Type', 'multipart/x-mixed-replace;
boundary=FRAME')
            self.end_headers()
            try:
                while True:
                    with output.condition:
                        output.condition.wait()
                        frame = output.frame
                    self.wfile.write(b'--FRAME\r\n')
                    self.send_header('Content-Type', 'image/jpeg')
                    self.send_header('Content-Length', len(frame))
                    self.end_headers()
                    self.wfile.write(frame)
                    self.wfile.write(b'\r\n')
            except Exception as e:
                logging.warning(
                    'Removed streaming client %s: %s',
                    self.client_address, str(e))
        else:
            self.send_error(404)
            self.end_headers()

output = StreamingOutput()
class StreamingServer(socketserver.ThreadingMixIn, server.HTTPServer):
    allow_reuse_address = True
    daemon_threads = True
def live():
    with picamera.PiCamera(resolution='640x480', framerate=24) as camera:

        #Uncomment the next line to change your Pi's Camera rotation (in degrees)
        camera.rotation = 180
        camera.start_recording(output, format='mjpeg')
        try:
            address = ('', 8000)
            server = StreamingServer(address, StreamingHandler)
            server.serve_forever()
```

```
        finally:
            camera.stop_recording()
recognizer = cv2.face.LBPHFaceRecognizer_create()
recognizer.read('trainer/trainer.yml')
cascadePath = "Haarcascades/haarcascade_frontalface_default.xml"
faceCascade = cv2.CascadeClassifier(cascadePath);
font = cv2.FONT_HERSHEY_SIMPLEX
#iniciate id counter
id = 0
# names related to ids: example ==> Marcelo: id=1, etc
names = ['None', 'Marcelo', 'Harsh', 'Elon Musk', 'Z', 'W']
# Initialize and start realtime video capture
cam = cv2.VideoCapture(0)
cam.set(3, 640) # set video widht
cam.set(4, 480) # set video height
# Define min window size to be recognized as a face
minW = 0.1*cam.get(3)
minH = 0.1*cam.get(4)
while True:
    ret, img =cam.read()
    img = cv2.flip(img, 0) # Flip vertically
    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

    faces = faceCascade.detectMultiScale(
        gray,
        scaleFactor = 1.2,
        minNeighbors = 5,
        minSize = (int(minW), int(minH)),
       )
    for(x,y,w,h) in faces:
        cv2.rectangle(img, (x,y), (x+w,y+h), (0,255,0), 2)
        id, confidence = recognizer.predict(gray[y:y+h,x:x+w])


        # If confidence is less them 100 ==> "0" : perfect match
        if (confidence < 100):
            id = names[id]
            confidence = " {0}%".format(round(100 - confidence))
        else:
            id = "unknown"
            confidence = " {0}%".format(round(100 - confidence))
            print("unknown face")
            whatsapp.whatsapp();
            cam.release()
            live()

        cv2.putText(
                img,
                str(id),
                (x+5,y-5,
```

```
                font,
                1,
                (255,255,255),
                2
               )
        cv2.putText(
                img,
                str(confidence),
                (x+5,y+h-5),
                font,
                1,
                (255,255,0),
                1
               )


    k = cv2.waitKey(10) & 0xff # Press 'ESC' for exiting video
    if k == 27:
        break
# Do a bit of cleanup
print("\n [INFO] Exiting Program and cleanup stuff")

cv2.destroyAllWindows()
```

Code for sending Whatsapp message

```
import os
from twilio.rest import Client


# Find your Account SID and Auth Token at twilio.com/console
# and set the environment variables. See http://twil.io/secure
def whatsapp():
        account_sid = "AC3d54c604bfd9b3671b24439b8933c013"
        auth_token = "f7e67cde3b94cf752c51a57628573752"
        client = Client(account_sid, auth_token)

        message = client.messages.create(
                        body='Enemy ahead For live stream check here
https://grabify.link/N2VICE',
                        from_='whatsapp:+14155238886',
                        to='whatsapp:+918168404341'
                    )

        print(message.sid)
def Ready():
        account_sid = "AC3d54c604bfd9b3671b24439b8933c013"
        auth_token = "f7e67cde3b94cf752c51a57628573752"
        client = Client(account_sid, auth_token)

        message = client.messages.create(
                        body='Ready for Surveillance',
                        from_='whatsapp:+14155238886',
                        to='whatsapp:+918168404341'
                    )

        print(message.sid)
Ready()
```

Code for Transmitter

```
#include <Arduino.h>
#include <WiFi.h>
#include <esp_now.h>
#include <Wire.h>
#include <Adafruit_Sensor.h>    // Adafruit sensor library
#include <Adafruit_ADXL345_U.h> // ADXL345 library

Adafruit_ADXL345_Unified accel = Adafruit_ADXL345_Unified(); // ADXL345
Object

// ESP8266 Mac address (first peer)
uint8_t mac_peer1[] = {0xC8, 0x2B, 0x96, 0x2E, 0xDD, 0x00};
esp_now_peer_info_t peer1;
//esp_now_peer_info_t peer2;
int i = 0;
int f=0;
typedef struct message {
  int laser;
  int cased;
};
struct message myMessage;
void setup() {
Serial.begin(115200);
pinMode(4,INPUT);
WiFi.mode(WIFI_STA);
 // Get Mac Add
 Serial.print("Mac Address: ");
 Serial.print(WiFi.macAddress());
 Serial.println("ESP32 ESP-Now Broadcast");
 // Initializing the ESP-NOW
 if (esp_now_init() != 0) {
   Serial.println("Problem during ESP-NOW init");
   return;
 }
 memcpy(peer1.peer_addr, mac_peer1, 6);
 peer1.channel = 1;
 peer1.encrypt = 0;
 // Register the peer
 Serial.println("Registering a peer 1");
 if ( esp_now_add_peer(&peer1) == ESP_OK) {
   Serial.println("Peer 1 added");
 }
   if(!accel.begin()) // if ASXL345 sensor not found
 {
```

```
   Serial.println("ADXL345 not detected");
   while(1);
  }
   myMessage.laser =0;

  /*memcpy(peer2.peer_addr, mac_peer2, 6);
  peer2.channel = 4;
  peer2.encrypt = 0;
  // Register the peer
  Serial.println("Registering a peer 2");
  if ( esp_now_add_peer(&peer2) == ESP_OK) {
    Serial.println("Peer 2 added");
  } */
}
void loop() {
  if(digitalRead(4)==1&&f==0){
     myMessage.cased =1;
     f=1;
  }
   sensors_event_t event;
 accel.getEvent(&event);
 if(event.acceleration.y>5&&f==1){
 Serial.print("Move left");
 myMessage.laser =2;
 }
  else if(event.acceleration.y<-5&&f==1){
  Serial.print("Move right");
  myMessage.laser =3;
 }
  Serial.println("Send a new message");
  esp_now_send(NULL, (uint8_t *) &myMessage, sizeof(myMessage));
  delay(500);
}
```

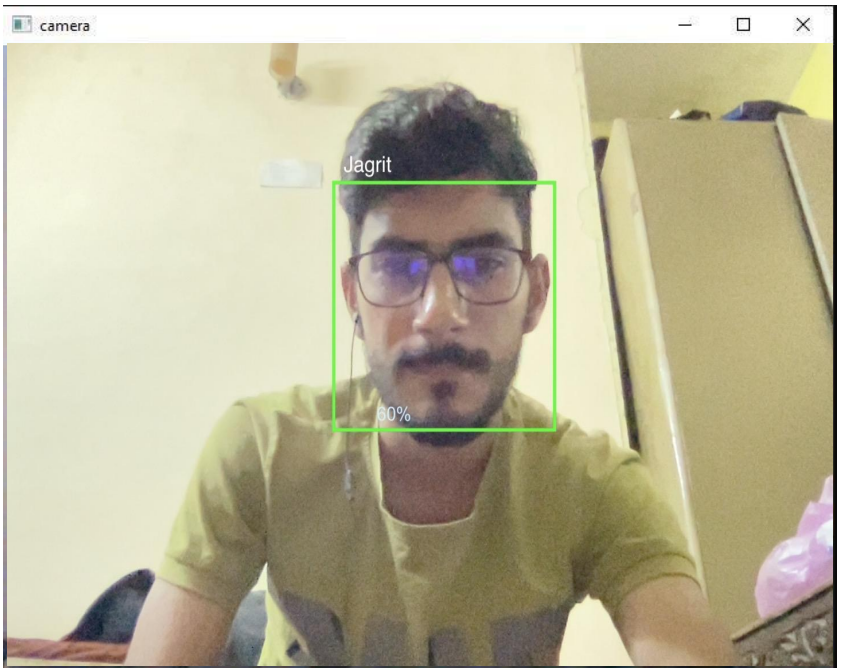
Code for receiving and moving servo Motors

```
#include <Arduino.h>
#include <ESP8266WiFi.h>
#include <espnow.h>
#include <Servo.h>
int pos =
0; int
pos1; int
f=0;
Servo servo_1;
Servo servo_4;
typedef struct message {
  int laser;
  int cased;
} message;
message myMessage;


void onDataReceiver(uint8_t * mac, uint8_t *incomingData, uint8_t len) {
  Serial.println("Message received.");
  // We don't use mac to verify the sender
  // Let us transform the incomingData into our message structure
  memcpy(&myMessage, incomingData, sizeof(myMessage));

  /*Serial.print("Red:");
  Serial.println(myMessage.red);
  Serial.print("Green:");
  Serial.println(myMessage.green);
  Serial.print("Blue:");
  Serial.println(myMessage.blue);*/
  if(myMessage.cased==1&&f==0){
  f=1;
    for (pos1=0; pos <= 80; pos +=1) {
    servo_4.write(pos1);
    // wait 15 ms for servo to reach the position
    delay(15); // Wait for 15 millisecond(s)

  }

  }
  if (myMessage.laser == 3&&f==1) {
    for (; pos <= 180; pos += 1) {
    servo_1.write(pos);
    if(myMessage.laser == 2){
      break;
    }
```

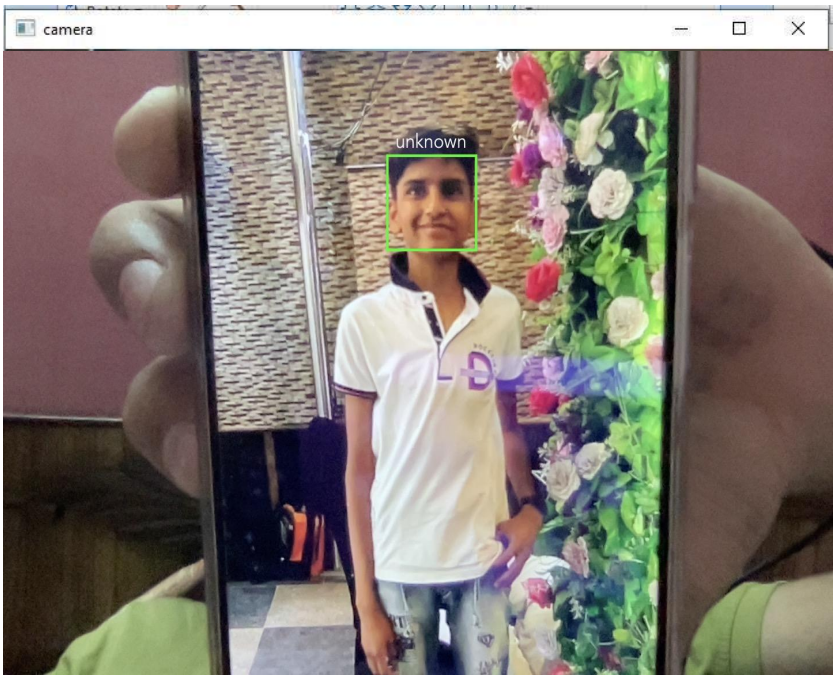
```
    // wait 15 ms for servo to reach the position
    delay(15); // Wait for 15 millisecond(s)
   }

   }
   else if (myMessage.laser == 2&&f==1) {
    for (; pos >= 0; pos -=1) {
    servo_1.write(pos);
    if(myMessage.laser == 3){
      break;
    }
    // wait 15 ms for servo to reach the position
    delay(15); // Wait for 15 millisecond(s)

   }
   }

}

void setup() {
  Serial.begin(115200);
  servo_1.attach(1, 500, 2500);
   servo_4.attach(4, 500, 2500);

  WiFi.disconnect();
  ESP.eraseConfig();
  // Wifi STA Mode
  WiFi.mode(WIFI_STA);
  // Get Mac Add
  Serial.print("Mac Address: ");
  Serial.print(WiFi.macAddress());
  Serial.println("\nESP-Now Receiver");

  // Initializing the ESP-NOW
  if (esp_now_init() != 0) {
    Serial.println("Problem during ESP-NOW init");
    return;
  }

  //esp_now_set_self_role(ESP_NOW_ROLE_SLAVE);
  // We can register the receiver callback function
  esp_now_register_recv_cb(onDataReceiver);
}
void loop() {
  // put your main code here, to run repeatedly:
}
```

Testing

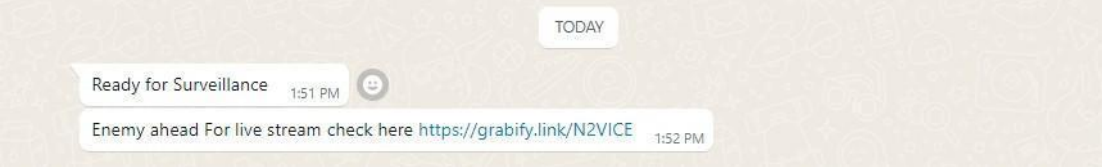
Ø Face Detection



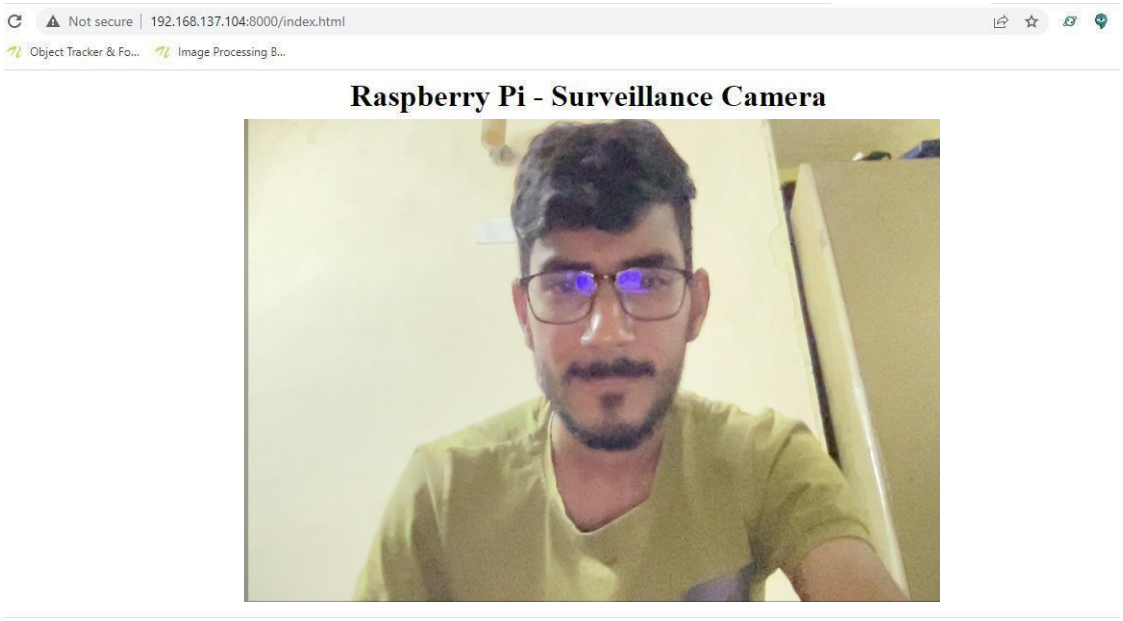
When an unrecognized face detected



Ø Whatsapp notification

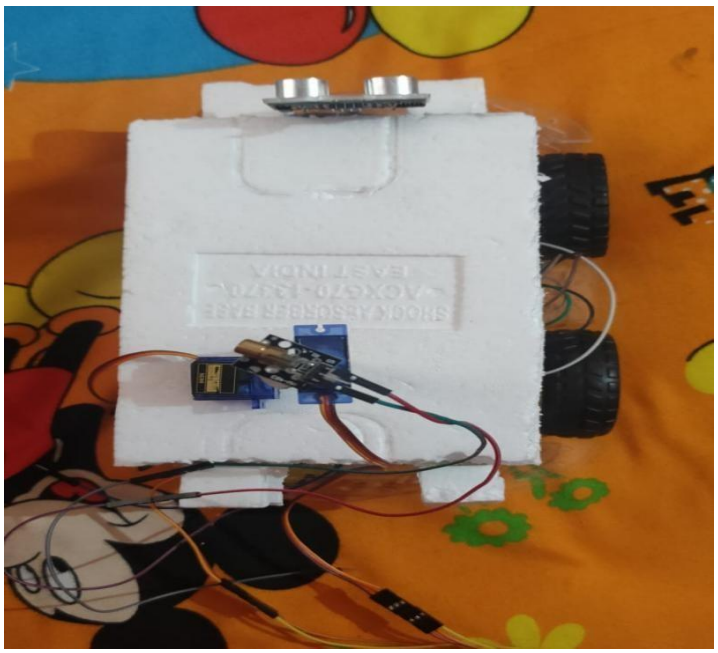


Testing Live Stream

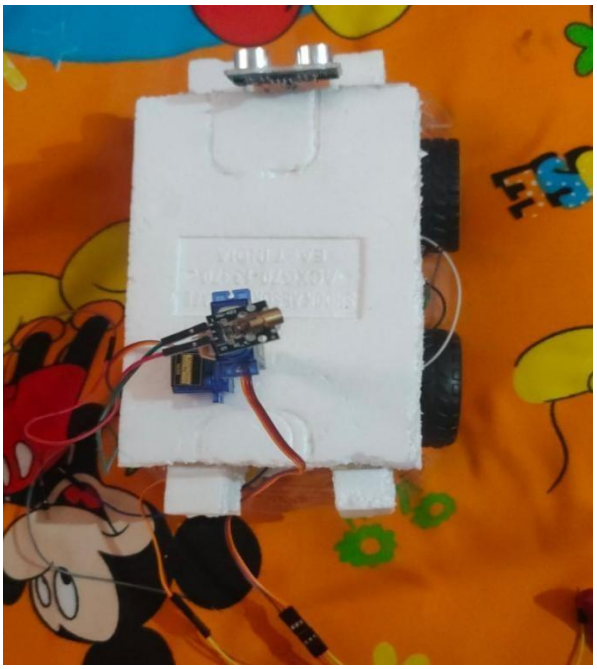


Turning laser up

Turning Laser Left’



Turning Laser Right



Result and Conclusion

Upon Testing the project we can conclude that our Gesture controlled smart surveillance vehicle can successfully recognise the unknown faces and It is able to send this information to the user through whatsapp application.This also enables the user to control the direction of the laser through hand gestures.

Future Scope of Project

This Project (Gesture Controlled Smart surveillance vehicle) aims to work in those areas where there is a higher risk of encountering an enemy. If It is used there It will protect our man power and benefit us a lot.Its object Avoidance capability makes it fully automated. It can also be used in wars.

References

- www.geeksforgeeks.com
- www.hackster.io.com
- www.github.com
- www.twilio.com
- www.cyberciti.biz
- www.randomnerdtutorials.com
- www.pyimagesearch.com
- www.projects.raspberrypi.org