# Project Report on
# Driver Assistant Bot

*Submission to The Robotics Club – SNIST as a part of Post Induction'23*
Team No – 5

THE ROBOTICS CLUB
*Integrating Knowledge...*


THE ROBOTICS CLUB – SNIST
SREENIDHI INSTITUTE OF SCIENCE AND TECHNOLOGY
(AUTONOMOUS)
(Affiliated to JNTU University, Hyderabad)
Yamnampet, Ghatkesar, Hyderabad – 501301

2023

# CERTIFICATE

This is the project work titled ‘Driver Assistant Bot’ by K PAVAN RAJU, K VENKATA SAI HARINI, ANANTH JEETH, SIDDHARTH, KARTHIK K, NISHANTH*`. This is a record of the project work carried out by them during the year 2023-24 as a part of POST INDUCTION under the guidance and supervision of

**Mr. G. Kovidh Addhish**

**&**

**Mr. Aarushraj Puduchery**

**Technical Heads**

**Mr. N V V S Narayana**

**The President of**

**The Robotics Club**

**Dr. A. PURUSHOTHAM**

**Faculty Advisor**

**Mechanical Department**

# DECLARATION

The project work reported in the present thesis titled "**DRIVER ASSSISTANT BOT**" is a record of work done by Team t5 in **THE ROBOTICS CLUB** as a part of **POST INDUCTION – 2023.**

**No part of the thesis is copied from books/journals/Internet and wherever the portion is taken, the same has been duly referred in the text. The report is based on the project work done entirely by Team team_number and not copied from any other source.**

# ACKNOWLEDGEMENT

This project report is the outcome of the efforts of many people who have driven our passion to explore into the implementation of **DRIVER ASSISTANT BOT**. We have received great guidance, encouragement and support from them and have learned a lot because of their willingness to share their knowledge and experience.

We thank our technical heads **Mr. G. Kovidh Addhish and Mr. Aarushraj Puduchery** for being with us till the end of the project completion.

We thank all members of the **Steering Body, Executive Body, Technical Advisory Board and Club's Incubation and Competence Committee** of **The Robotics Club** for helping us with crucial parts of the project. We are deeply indebted to **Mr. N V V S Narayana** – The President, **Ms. Mugala Shravani** – The Vice President, **Mr. N Abinav** – General Secretary and **Ms. Maliha** – SAB Chairman **THE ROBOTICS CLUB** respectively and also every other person who spared their valuable time without any hesitation whenever we wanted.

We also thank our faculty advisor **Dr. A. Purushotham**, Professor Mechanical Department, who encouraged us during this project by rendering his help when needed.

Contents Pg. No.

# ABSTRACT

# THE ROBOTICS CLUB – SNIST

# POST INDUCTION'23

# TEAM-NO – 5

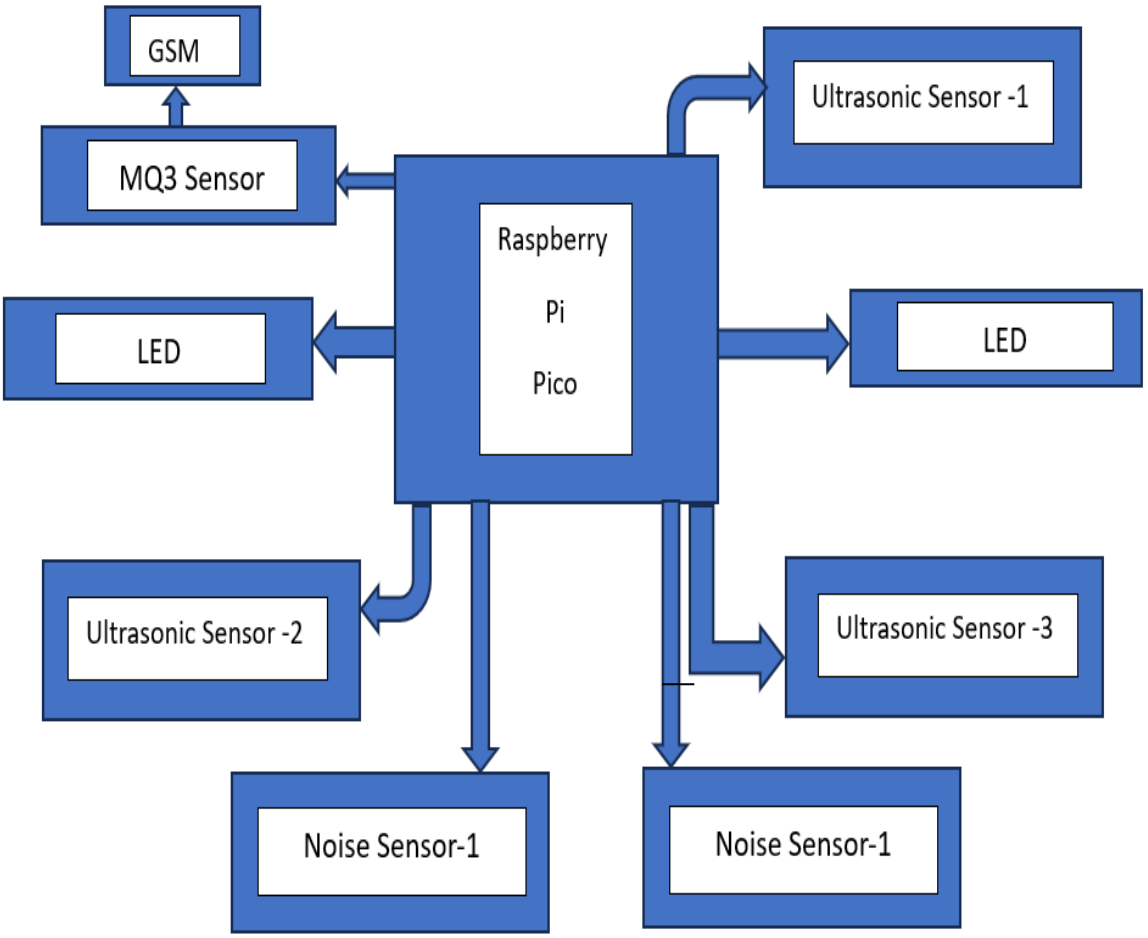
## THE PROBLEM:

The increasing occurrences of accidents during overtaking can be attributed to several factors. Firstly, individuals with hearing disabilities often struggle to perceive auditory signals, including horn sounds from vehicles approaching from behind, rendering them unaware of potential hazards. Furthermore, the trend of working professionals participating in virtual conferences while driving diverts their attention away from the road, increasing the likelihood of accidents. Lastly, the prevalence of listening to loud music while driving not only impairs drivers' ability to hear other vehicles but also contributes to distraction. Whenever any vehicle tries to overtake, the hearing-impaired drivers have no indication or alert for this, hence leading to accidents specially while overtaking. Also, during heavy rains and foggy times, the visibility of vehicles on a rear-view mirror is less as compared to a normal day, hence there must be a system to alert the target vehicle about the presence of surrounding vehicles.

## THE TEAMS APPROACH TO THE PROBLEM: -

Using Raspberry Pi Pico, we are developing a Driving Assistant Bot (DAB). DAB is equipped with Ultrasonic sensors to detect a close-by vehicle which is trying to overtake and gives the indication to the driver using led in front of the driver seat. Also, noise sensors are installed so as to detect horn sounds and indicate the driver. Additional feature of DAB is alcohol detection and communication to family members and local authorities.

BLOCK DIAGRAM:



# 1. Introduction

**Abstract:** A DAB is a system which is helpful to reduce the accidents due to neglectance of drivers these days. The increasing occurrences of accidents during overtaking can be attributed to several factors. Firstly, individuals with hearing disabilities often struggle to perceive auditory signals, including horn sounds from vehicles approaching from behind, rendering them unaware of potential hazards. Furthermore, the trend of working professionals participating in virtual conferences while driving diverts their attention away from the road, increasing the likelihood of accidents. Lastly, the prevalence of listening to loud music while driving not only impairs drivers' ability to hear other vehicles but also contributes to distraction. Whenever any vehicle tries to overtake, the hearing-impaired drivers have no indication or alert for this, hence leading to accidents specially while overtaking.

**1.1 Problem Statement:** Blind turns, characterized by limited visibility, pose significant risks to drivers, pedestrians, and cyclists. Accidents at blind turns often occur due to the inability to anticipate oncoming vehicles or pedestrians, leading to collisions and potential fatalities. The Smart Road System seeks to tackle this problem by employing advanced technologies and intelligent infrastructure.

**1.2 Approach:**

Using Raspberry Pi Pico, we are developing a Driving Assistant Bot (DAB). DAB is equipped with Ultrasonic sensors to detect a close-by vehicle which is trying to overtake and gives the indication to the driver using led in front of the driver seat. Also, noise sensors are installed so as to detect horn sounds and indicate the driver. Additional feature of DAB is alcohol detection and communication to family members and local authorities.

# 2. Architecture

## 2.1Hardware Used:

1. **Raspberry Pi Pico:**

The Raspberry Pi Pico is a microcontroller board developed by the Raspberry Pi Foundation. It features the RP2040 microcontroller chip, which offers dual ARM Cortex-M0+ cores, ample GPIO pins, and a range of peripherals. The Pico is ideal for embedded projects and IoT applications, supporting MicroPython and C/C++ programming. Its low cost, versatility, and extensive community support make it a popular choice for various DIY electronics projects.

The Raspberry Pi Pico microcontroller has 40 GPIO (General-Purpose Input/Output) pins that can serve various functions, including digital input and output, analog input, I2C, UART, SPI, PWM

2. **Noise sensor:**

The LM393 is a comparator IC often used in noise sensor circuits. It senses environmental noise by comparing an input signal with a reference voltage, and its digital output changes state when noise exceeds a certain threshold. It's commonly employed in sound-activated switches, noise level monitoring systems, or alarm triggers. The LM393 is known for its sensitivity and versatility in detecting and responding to variations in sound levels.

3. **Center Shaft Motor:**

A center shaft DC motor is a type of direct current motor with its rotating shaft positioned in the center of the motor housing. It typically has two terminals for electrical connections. When current flows through the motor, it generates rotational motion around the central axis, making it suitable for various applications such as robotics, toys, and automation. The direction and speed of rotation can be controlled by varying the voltage polarity or

using additional components like an H-bridge. These motors come in various sizes, torque ratings, and operational voltages to suit different applications.

**7. Jumper wires:** These are used to connect the components to the pins of Arduino boardwith the help of connecting pins.

**5. Ultrasonic Sensor:** The Ultrasonic sensor is one kind of electronic component, used to detect objects via tx and rx pins (transmitter and receiver of sound waves).

**6. LED:** A light-emitting diode (LED) is a semiconductor device that emits light when current flows through it. Electrons in the semiconductor recombine with electron holes, releasing energy in the form of photon**.**

**4. MQ3:**

The MQ-3 sensor is a gas sensor that detects the presence of various gases in the environment. It's commonly used to detect alcohol, ethanol, and other volatile organic compounds. The sensor operates on the principle of resistance change: when it comes into contact with a specific gas, its electrical resistance changes, which can be measured to determine gas concentration. MQ-3 sensors are used in applications like breathalyzer devices, gas leak detectors, and alcohol detection systems. They are typically equipped with an analog output that varies based on gas concentration and require calibration for accurate readings.

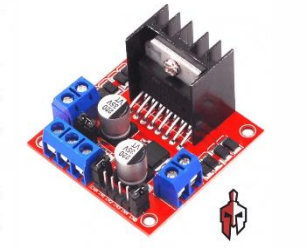

**5. GSM :**

The A7670c is a GSM (Global System for Mobile Communications) module used for cellular communication. It provides connectivity for sending and receiving calls, SMS, and data over cellular networks. The module typically uses AT commands for control and supports various communication protocols, making it suitable for applications like IoT, remote monitoring, and SMS notifications. It often interfaces with microcontrollers or single-board computers and requires a SIM card for network access.


**6. Motor Driver:**

The L298N is a dual H-bridge motor driver integrated circuit. It's commonly used to control the direction and speed of DC motors and stepper motors in robotics and automation projects. The L298N can handle high current and voltage

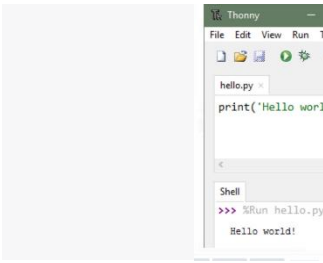
levels, making it suitable for driving larger motors. It provides bidirectional control, allowing motors to rotate in both directions, and can be controlled with logic-level signals from microcontrollers or other control systems. Additionally, it includes protection diodes for back-EMF (electromotive force) suppression.



## 2.2 Software Used:

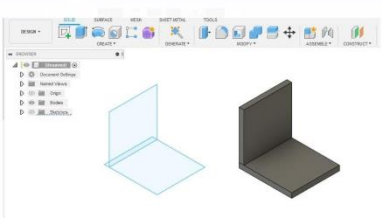
**1.Thonny IDE: -**

Thonny is an Integrated Development Environment (IDE) for Python programming. It's designed to be beginner-friendly, featuring a simple and clean interface. Thonny provides features like code highlighting, debugging, and package management, making it a great choice for learning and developing Python applications, particularly for newcomers to programming. We use MIcroPython for coding.

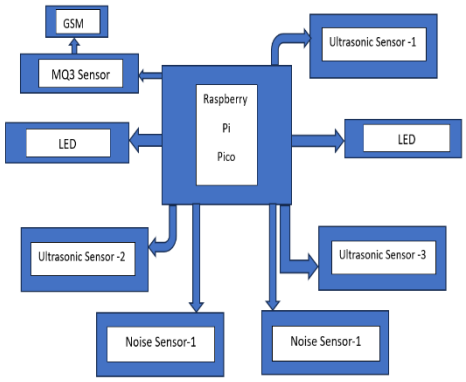



**2. Fusion 360:**

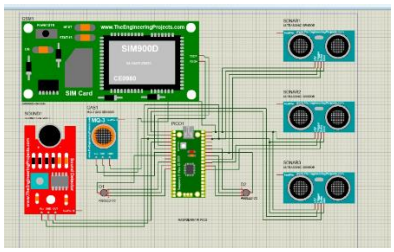
Fusion 360 is a commercial computer-aided design, computer-aided manufacturing, computer-aided engineering, and printed circuit board design software application, developed by Autodesk. It is available for Windows and macOS, with simplified applications available for Android and iOS. Fusion 360 is licensed as a paid subscription, with a free limited home-based, non-commercial personal edition available.



## 3.1 BLOCK DIAGRAM:




## 3.2 CIRCUIT DIAGRAM



## 3.3 WORKING:

The working of DAB (Alert system) is simple. When a vehicle is detected at a particular distance using Ultrasonic sensor either sides of the vehicle, then an alert signal is given using RGB led light which will blink. This is very useful while overtaking. Additionally, a noise sensor is installed to detect auditory sounds and give indication to hearing-impaired drivers. And MQ-3 sensor is used for alcohol-detection and furthermore sends sms and notification to family members.

## 1. Experimental Results and Conclusions:

### 4.1 Future enhancements:

The vehicle speed can be controlled via the amount of alcohol detected. This alert system can also be used for challan system.

### 4.2 Source Code:

```
{
   "BOT_TOKEN":
"6673412321:AAGh4Y7DTZfnkQs9JYLai81S4mlmkoM
E9J8",
   "CHAT_ID": "6312386599",
   "BAUDRATE": 115200,
   "UART_INTERFACE": 0,
   "TX_PIN": 12,
   "RX_PIN": 13,
   "APN": "JioNet"
}
from machine import UART, Pin
from time import sleep
import ujson
# import config
class A7600X:
   def __init__(self, interface, baudrate, txpin, rxpin):
      self.uart = UART(interface,
                       baudrate=baudrate,
                       bits=8,
                       parity=None,
                       stop=1,
                       tx=Pin(txpin),
                       rx=Pin(rxpin))
   def _exe_command(self, command):
      self.uart.write(command+'\r\n')
      sleep(2)
      response = self.uart.read()
      print(response.decode('utf-8'))
      return response

   def _http_post(self, url, type, data):
      json_data = ujson.dumps(data)
      data_length = len(json_data)
      uart_commands = [
         'AT+HTTPINIT',
         'AT+HTTPPARA="URL","' + url + '"',
         'AT+HTTPPARA="CONTENT","' + type + '"',
         'AT+HTTPDATA={},10000'.format(data_length),
         json_data,
         'AT+HTTPACTION=1',
         'AT+HTTPHEAD',
         'AT+HTTPTERM'
      ]

      for command in uart_commands:
         self._exe_command(command)
import A7600X
from time import sleep
import ujson
# import config
class Main:
   def __init__(self, config_file_path='config.json'):
      with open(config_file_path) as config_file:
         self.config_json = ujson.load(config_file)

      self.gsm =
A7600X.A7600X(self.config_json['UART_INTERFACE'
],
                    self.config_json['BAUDRATE'],
                    self.config_json['TX_PIN'],
                    self.config_json['RX_PIN'])
      self.telegram_url = 'https://api.telegram.org/bot' + \
         self.config_json['BOT_TOKEN'] +
'/sendMessage'
      self.http_type = 'application/json'
      sleep(10)
      response = self.gsm._exe_command('AT')
      response += self.gsm._exe_command('ATE')
      response += self.gsm._exe_command('ATI')
      response += self.gsm._exe_command('AT+CIMI')
      response += self.gsm._exe_command('AT+CNUM')
      response += self.gsm._exe_command('AT+COPS?')
      response += self.gsm._exe_command('AT+CPIN?')
      self.gsm._http_post(
         self.telegram_url,
         self.http_type,
         self._telegram_payload(response))

      while True:
         self.gsm._exe_command('AT+CMGF=1')
         sms_response =
self.gsm._exe_command('AT+CMGL="REC
UNREAD"')
         sms_messages = sms_response.decode('utf-
8').split('+CMGL: ')[1:]
#           text.decode("utf-8").encode("windows-
1252").decode("utf-8")
#           text.encode("windows-1252").decode("utf-8")
         print(sms_messages)
         for sms_message in sms_messages:
            message_lines = sms_message.split('\r\n')
            timestamp =
message_lines[0].split(',')[4].replace("", '')
```

```
            message = message_lines[1]
            formatted_message = "<b>Time:
</b>{}\r\n<b>Message: </b>{}".format(
                timestamp, message)
            self.gsm._http_post(
                self.telegram_url,
                self.http_type,
                self._telegram_payload(formatted_message))
            sms_index = message_lines[0].split(',')[0]

self.gsm._exe_command('AT+CMGD={}'.format(sms_in
dex))
            sleep(1)
        self.gsm._exe_command('AT+CMGD=1,4')
        sleep(10)
    def _telegram_payload(self, message):
        payload = {
            "chat_id": self.config_json['CHAT_ID'],
            "text": message,
            "parse_mode": "HTML"
        }
        return payload

if __name__ == "__main__":
    Main()
'''digital=Pin(0,Pin.IN)
#analog = Pin(17,Pin.IN)

while True:
    print("Digital Value:",digital.value())
    print("-----------------")
#     analog_value = gpio.input()
#     print("Analog Value:", analog_value)
    time.sleep(1)
from machine import Pin,PWM,ADC
from time import sleep,sleep_us,ticks_us,ticks_diff
#Led pinouts: R,Vcc(Common Anode=on is off, off is
on),G,B
'''
Ultrasonic:
u1_tx = 16,   u2_tx = 12,   u3_tx =4
u1_rx=17,   u2_rx=13,   u3_rx=5
Noise Sensors:
front, back ==AO==26,27
left,right==DO==11,18
GSM:
tx=8 , rx=9
Led:
red1=21, blue1=19, green1=20
red2=0, blue2=1
noise_blue=6 ,
noise_red=7
#(0,1),(4,5),(8,9),(12,13),(16,17)'''
#Ultra+LED
red1=Pin(21,Pin.OUT)
blue1=Pin(19,Pin.OUT)
green1=Pin(20,Pin.OUT)
red2=Pin(0,Pin.OUT)
blue2=Pin(1,Pin.OUT)
#Sound+LED
noise_red=Pin(7,Pin.OUT)
noise_blue=Pin(6,Pin.OUT)
noise_green=Pin(3,Pin.OUT)
#sound_front= Pin(26, Pin.IN, Pin.PULL_DOWN)
DOUBT
# sound_back= Pin(6, Pin.IN, Pin.PULL_DOWN)
sound_left= Pin(11, Pin.IN, Pin.PULL_DOWN)
sound_right= Pin(18, Pin.IN, Pin.PULL_DOWN)
def led_blink(l):
    for i in range(20):
        l.value(0)
        sleep(0.08)
        l.value(1)
        sleep(0.08)
def sound_led_blink(led):
    for i in range(10):
            led.off()
            sleep(0.1)
            led.on()
            sleep(0.1)

u1_tx=Pin(16,Pin.OUT)
u1_rx=Pin(17,Pin.IN)
u2_tx=Pin(12,Pin.OUT)
u2_rx=Pin(13,Pin.IN)
u3_tx=Pin(4,Pin.OUT)
u3_rx=Pin(5,Pin.IN)
def distance(tx,rx):
    tx.on()
    sleep_us(10)
    tx.off()
    while rx.value()==0:
        start_time=ticks_us()
    while rx.value()==1:
        end_time=ticks_us()
    t=ticks_diff(end_time,start_time)
    dist=(t*0.0343)/2
    return dist

while True:
    red1.on()
    red2.on()
    blue1.on()
    blue2.on()
    green1.on()
```

```
    noise_red.on()
    noise_blue.on()
    noise_green.on()
    adc1=ADC(Pin(26))
    adc2=ADC(Pin(27))
    f=adc1.read_u16()
    b=adc2.read_u16()

    dr=distance(u1_tx,u1_rx)
    print("Right Distance: {:.2f} cm".format(dr))
    dl=distance(u2_tx,u2_rx)
    print("Left Distance: {:.2f} cm".format(dl))
    df=distance(u3_tx,u3_rx)
    print("Front Distance: {:.2f} cm".format(df))
    print("-------")

    print("Front Sound::{:.2f} Hz  Back Sound::{:.2f}
Hz".format(f,b))
    print("Left  Sound::{:.2f} Hz   Right Sound::{:.2f}
Hz".format(sound_left,sound_right))
    print("------------")

    if(dr<=30 and dl>30):
        blue1.on()
        red1.off()
        led_blink(red1)
        red1.on()
    sleep(0.5)
    if(dl<=30 and dr>30):
        blue2.on()
        red2.off()
        led_blink(red2)
        red2.on()
    sleep(0.5)
    if (dr<=30 and dl<=30):
        red1.on()
        red2.on()
        blue1.off()
        blue2.off()
        if(dr<dl):
            led_blink(blue1)
        if(dl<dr):
            led_blink(blue2)
        blue1.on()
        blue2.off()
    sleep(0.5)
    if(20<=df<=40):
        green1.off()
        sleep(2)
#     Noise Sensor
    if f>=2500:
        sound_led_blink(noise_green)
    if b >= 3000:
        sound_led_blink(noise_red)
    if left==1 or right==1:
        sound_led_blink(noise_blue)

    noise_red.on()
    noise_blue.on()
    noise_green.on()
#     sleep(2)
    red1.on()
    red2.on()
    blue1.on()
    blue2.on()
    green1.on()
    sleep(1)
```

## 4.3 Conclusions:

This system will reduce the accident. Additionally, It will also help in reducing number of drunk and drive accidents. Furthermore, it helps in management and decision making.