University of Mumbai

Robotic system for monitoring environment and human body detection inside tunnel

Submitted in partial fulfillment of requirements

For the degree of

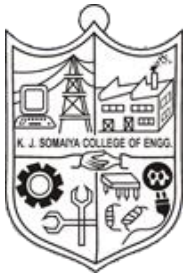
Bachelors of Technology

by

Charan Parasuraman
Roll No: 1613070
Aditi Gupta
Roll No: 1613078
Rahul Jadeja
Roll No: 1613080
Saud Ahmed Khan
Roll No: 1613082

Guide

Mrs. Savita Raut



Department of Electronics and Telecommunication Engineering
K. J. Somaiya College of Engineering, Mumbai-77
(Autonomous College Affiliated to University of Mumbai)

Batch 2016 -2020

K. J. Somaiya College of Engineering, Mumbai-77

(Autonomous College Affiliated to University of Mumbai)

Certificate

This is to certify that the dissertation report entitled **Robotic system for monitoring environment and human body detection inside tunnel** is bona fide record of the dissertation work done by

1. Charan Parasuraman
2. Aditi Gupta
3. Rahul Jadeja
4. Saud Ahmed Khan

in the year 2019-20 under the guidance of Mrs. Savita Raut of Department of Electronics and Telecommunication Engineering in partial fulfillment of requirement for the Bachelors of Technology degree in Electronics and Telecommunication Engineering of University of Mumbai.

_____ _____

Guide Head of the Department

_____

Principal

Date:5th June, 2020

Place: Mumbai-77

K. J. Somaiya College of Engineering, Mumbai-77

(Autonomous College Affiliated to University of Mumbai)

Certificate of Approval of Examiners

We certify that this dissertation report entitled **Robotic system for monitoring environment and human body detection inside tunnel** is bona fide record of project work done by

1. Charan Parasuraman
2. Aditi Gupta
3. Rahul Jadeja
4. Saud Ahmed Khan

This project is approved for the award of Bachelors of Technology Degree in Electronics and Telecommunication Engineering of University of Mumbai.

_____

Internal Examiner

_____

External Examiner

Date:12th June, 2020

Place: Mumbai-77

K. J. Somaiya College of Engineering, Mumbai-77

(Autonomous College Affiliated to University of Mumbai)

DECLARATION

We declare that this written thesis submission represents the work done based on our and / or others' ideas with adequately cited and referenced the original source. We also declare that we have adhered to all principles of intellectual property, academic honesty and integrity as we have not misinterpreted or fabricated or falsified any idea/data/fact/source/original work/ matter in my submission.

We understand that any violation of the above will be cause for disciplinary action by the college and may evoke the penal action from the sources which have not been properly cited or from whom proper permission is not sought.

_____ **Signature of the Student** _____**1613070**_____ **Roll No.**	_____ **Signature of the Student** _____**1613078**_____ **Roll No.**
_____ **Signature of the Student** _____**1613080**_____ **Roll No.**	_____ **Signature of the Student** _____ **1613082**_____ **Roll No.**

Date: 5th June, 2020

Place: Mumbai-77

Dedicated to

My family and friends

Abstract

The operation and maintenance of ageing civil infrastructure such as tunnels is one of the greatest challenges society is facing today. Due to ageing of the tunnels, they possess a risk of collapsing. When the tunnel collapses, huge damage is done to human lives in great number. Thereafter, it is also difficult to detect the bodies as tunnels have health and safety risk because the environment is frequently dusty, dark and hazardous.

In this project the aim is to automate the human detection and hazardous gas detection process using robotic system. A system for human detection in underground tunnels, using a fusion of three-dimensional (3D) information with Open computer vision (OpenCV) and python using algorithm HAAR cascade file is proposed in this project. Another method and more preferred method for human body detection is using deep learning algorithm and OpenCV.

The robotic structure consists of motor driver IC, geared motors, wheels, Raspberry pi, Arduino uno, camera module, sensors and fiber body. This is the proposed architecture for automating the process. The project report highlights the motivation behind the project, scope, usage, software and hardware used, future scope and conclusions and future scope of the project.

***Key words*:** R-CNN, Haar Cascade, Body Detection, Raspberry pi, Arduino, Sensors, Automation, Robotics, Tunnel Monitoring, Computer Vision, Image Processing etc.

Contents

List of Figures

List of Tables

Nomenclature

AC	Alternating current
ARM	Acorn RISC Machine
BLE	Bluetooth Low Energy
CO	Carbon Monoxide
CNN	Convolutional Neural Network
CSI	Camera Serial Interface
CH_4	Methane
DC	Direct current
EMF	Electromagnetic Field
GB	Gigabyte
GHz	Gigahertz
GPIO	General-purpose input/output
GPU	Graphics Processing Unit
H_2	Hydrogen gas
HDMI	High-Definition Multimedia Interface
HOG	Histograms of Oriented Gradients
IEEE	Institute of Electrical and Electronics Engineers
LAN	Local Area Network
LED	Light emitting Diode

LPDDR	Low-Power Double Data Rate
Mbps	Megabits per second
MLP	Multilayer perceptrons
OpenCV	Open Source Computer Vision Library
PCB	Printed Circuit Board
PoE	Power-over-Ethernet
PWM	Pulse Width Modulation
RH	Relative Humidity
ReLU	Rectified Linear Units
SD	Secure Digital
SDRAM	Synchronous Dynamic Random Access Memory
SoC	System On a Chip
TTL	Transistor-transistor Logic
USB	Universal Serial Bus

Chapter 1

Introduction

This chapter presents a brief introduction of the project. It gives a glimpse of the motivation behind the project and the significance of this topic in real world. Further this chapter highlights the future scope of the project and how it can be extended in other domains. This chapter also gives a brief introduction of complete project.

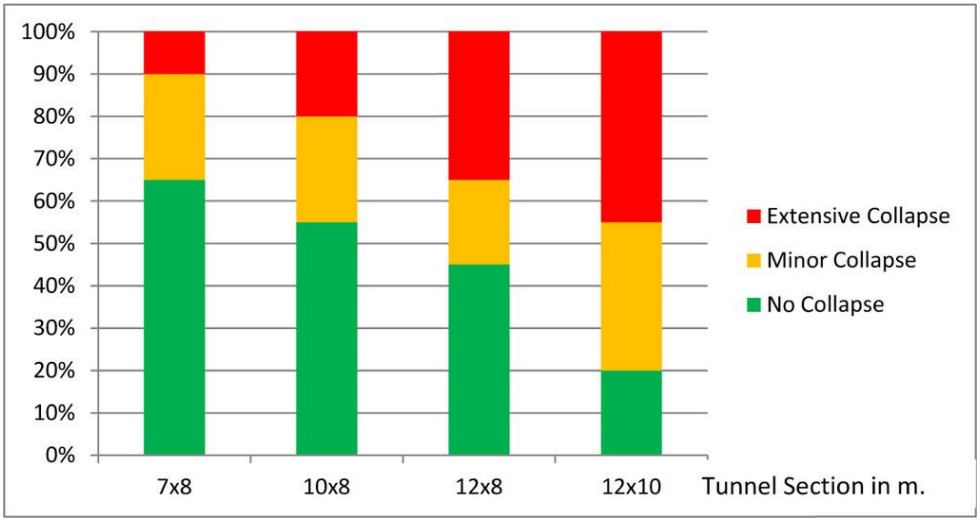
1.1 Background

The operation and maintenance of ageing civil infrastructure such at tunnels is one of the greatest challenges society is facing today. Thorough and timely inspection can result in the early identification and resolution of any problem. Traditionally, the visual inspection of large-scale assets has been an expensive, labor-intensive process. While inspection methods across the industry are gradually being modernized with the help of automation and machine vision, many asset owners still rely on manual inspection by teams of trained inspectors. For tunnels, this is carried out by time-taking manual walkthroughs, which can possess health and safety risk as the environment is frequently dusty, dark and hazardous. Inspection records often are subjective, incomplete and prone to human error. These erroneous records can lead o fatal accidents. In this work, a system is developed which allows to detect the hazardous gas levels and is useful for body detection in case of tunnel collapse.

1.2 Motivation

The following data shows the probability of tunnel collapsing based on cross section of the tunnel. From this graph it can be concluded that as the cross section of the tunnel increases, the probability of collapsing increasing. The life of the tunnel is also determined by how timely maintenance takes place and how labor intensive they are. But still it possesses risk to human health because of the presence of harmful gases.

Figure 1: Collapsing rate for tunnels of various cross sections.



1.3 Scope of the project

During regular inspection, there is a major health risk to workers due to accumulation of hazardous gases because of cracks. Also in fatal tunnel accidents, many workers are buried inside and hazardous gas accumulate thereafter. Due to accumulation of hazardous gases it becomes difficult to take out the bodies of buried workers or to detect where the bodies are. In this project various methods are used for body detection and to locate the bodies in such unfavorable condition. Furthermore, this project also covers measurement of the level of harmful and hazardous gases inside the tunnel by deploying various sensors on a robot. A robot can also be used to go through small crack to reach depth not accessible to humans and can be used to detect if there's someone alive. By automating this process, major shift can be done in the ways of inspection of tunnel and in case of some major accident this can save a lot of work force.

1.4 Brief description of project undertaken

In this project a prototype is prepared for human body and gas level detection. This report gives a brief summary and an evaluation of some of the open-source projects and libraries that have been used for human detection. Sensors of various specifications have been researched. The most appropriate and apt sensors have been installed on the robot. Raspberry pi for software application and some hardware and Arduino board for sensors are also deployed in this project.

Human Detection is a branch of Object Detection. Object Detection signifies identifying the presence of predefined types of objects in an image. This task involves both identification of the presence of the objects (like humans) and identification of the rectangular boundary surrounding each object (i.e. Object Localization). An object detection system which can detect the class "Human" can work as a Human Detection System [4].

An autonomous robotic system has been developed in this project so as to minimize humans' interaction during tunnel inspection and accident. Recent developments in robotics and associated fields of computer vision and sensors pave the floor for automated robotics and systems. This system consists of a camera module, a mini computer Raspberry pi, microcontroller Arduino Uno, motor driver module L298, various gas sensors such as MQ-2 and MQ-7 and various hardware components like geared motor, wheels, wires etc. Currently, tunnel inspections are performed visually by human operators. This can result in a slow, labor expensive and subjective process. In case of tunnel accidents as well laborers are employed for searching of bodies but this can risk their lives because of the presence of harmful gases. This robot can be helpful as it reduces the safety requirement and can be the future foundation for further research in this area.

Figure 2: Model Figure for reference.

Chapter 2

Literature survey

This chapter provides literature survey done for understanding the topic of the project. Literature survey presents all the possible options for completing a specific task apart from the one executed. In the chapter the major points discussed are possible methods for body detection, various sensors and their specifications. The objective of the thesis work is defined at the end of the literature survey.

2.1 Commonly used approaches for Human Detection

Commonly used approaches for Human Detection originated in the early 2000s. These approaches are still being used in the industry because of their utility and effectiveness.

2.1.1 Haar Cascade algorithms for Human Detection

In paper proposed by Paul Viola and Michael Jones, Haar feature based approach for Object Detection is used ["Rapid Object Detection using a Boosted Cascade of Simple Features" published in 2001]. This approach is widely used for Face Detection [4].

OpenCV has inbuilt functionality to provide Haar Cascade based Object Detection. OpenCV provides Pre-trained models provided for "Full Body Detection", "Upper Body Detection" and "Lower Body Detection" are available. Application of Haar Cascade for Human Detection using OpenCV shows a frame time of approximately 90 — 100 milliseconds per frame (equivalent to 11 frames-per-second) [3].

2.1.2 Histograms of Oriented Gradients for Human Detection

HOG pedestrian detection approach is proposed by B. Triggs and N. Dalal in their paper ["Histograms of oriented gradients for human detection" published in 2005].

OpenCV includes inbuilt functionality to provide Histogram of Oriented Gradient based detection. A pre-trained model for Human Detection is also included. Application of HOG Human Detection using Open CV, shows a frame time of approximately 150–170 milliseconds per frame (equivalent to 6.25 frames-per-second) [5].

2.2 Drawbacks of Commonly used Approaches

Listed below are some common drawbacks when using Haar Cascades and HOG for Human Detection. These observations are based on pre-trained models available with Open CV [6].

2.2.1 Missed Detections

The above mentioned two approaches are not very good in detecting humans in various poses unless multiple models are used to detect humans in each pose. Available pre-trained models with Open CV are trained to identify the standing pose of a person and not other poses. They perform well on detecting persons from front view and back view**.** However, detections from side views of persons are generally poor [6]**.**

2.2.2 False Detections and Duplicate Detections

These commonly used approaches are also susceptible for detecting non-human objects as humans. A trade-off between Missed Detections and False Detections can be achieved by adjusting the threshold parameters. Certain false detections can be avoided by defining thresholds on minimum detection box size.

Duplicate detections may also happen. A technique known as non max suppression can be used to avoid certain duplicate detentions [6].

2.2.3 Unreliable Detection Boundary

The detection boundary provided by HOG and Haar Cascade does not precisely fit the detected person. In fact, the margin of the boundary is not consistent between detections. This makes it difficult to derive positions of body parts of a person (say location of feet) using ratios calculated on the detection boundary [6].

2.2.4 Flickers in Detection

Quite often it is observed that a person detected in one frame is not detected in the subsequent frame and vice versa. Thereby, detentions are susceptible to flickering [6].

2.3 Strengths of commonly used approaches

Despite mentioned drawbacks, these approaches are still being used in the industry. They require comparatively less computing power compared to modern deep learning based approaches. (No need of GPUs to work in real-time.) These approaches are readily available in computer vision libraries such as OpenCV, making them attractive first choices. Most of the issues present in commonly used human detection approaches are fixed in newer deep learning based approaches.

2.4 Modern Approaches for Human Detection - Deep Convolution Neural Networks

Modern approaches for human detection are mostly based on Deep Convolution Neural Networks. This trend started with AlexNet which won the Imagenet Large Scale Visual Recognition Challenge (ILSVRC) in year 2012 for Image Classification using a Deep Convolution Network (CNN). Since then, CNNs have been widely adapted for various computer vision problems such as Object Detection (detecting different types of objects in an image), Object Localization (determining locations of detected objects) and Image Classification (identifying what type of an object an image contains). "Human Detection" is a special case of Object Localization and Object Detection [7].

They are Multi-class Object Detectors. Another key importance of modern CNN based Object Detection systems is that they are capable of detecting multiple classes of objects. Thus, modern state-of-the-art Human Detectors are not just Human Detectors, but accurate Object Detectors which can detect multiple types of objects including humans [7].

2.5 Strengths of CNN for Human detection

The advantages of CNN against traditional methods can be summarized as follows [7]:

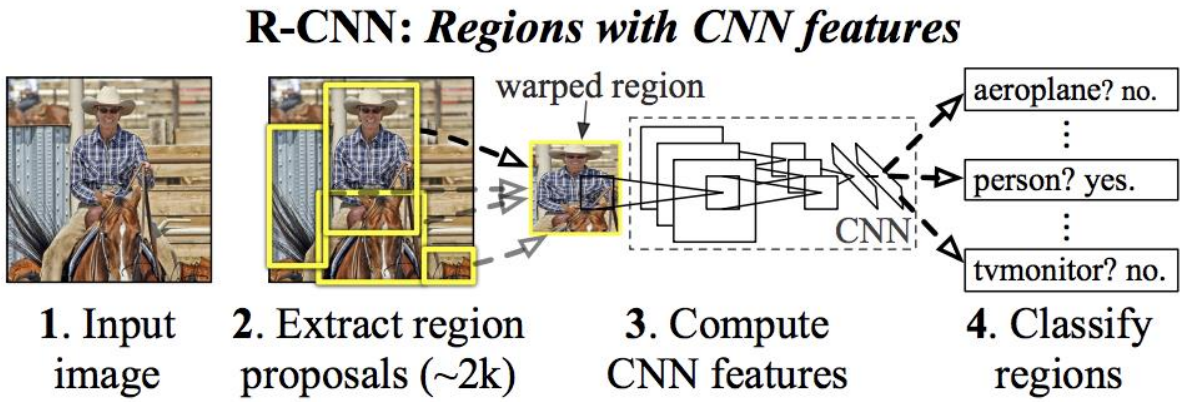
• Hierarchical feature representation is the multilevel representations from pixel to high-level semantic features learned by a hierarchical multi-stage structure can be learned from data automatically. Hidden factors of input data can be disentangled through multi-level nonlinear mappings.

• Compared with traditional shallow models, a deeper architecture such as CNN provides an exponentially increased expressive capability.

• The architecture of CNN provides a faster and efficient way to jointly optimize several related tasks together (e.g. Fast RCNN combines classification and bounding box regression into a multi-task leaning manner).

• Some classical computer vision challenges can be recast as high-dimensional data transform problems and solved from a different viewpoint by benefitting from the large learning capacity of deep CNNs [7].

Figure 3: Deep convolution neural network.



2.6 Introduction to Gas Sensor

A gas sensor is a device that detects the presence or concentration of gases in the atmosphere. Based on the concentration of the gas in the atmosphere the sensor produces a corresponding potential difference by changing the resistance of the material inside the sensor. This is measured as output voltage. Based on this voltage value the type and concentration of the gas can be found. The type of gas the sensor could detect depends entirely on the sensing material present inside the sensor. Normally these sensors are available as modules with comparators as s. These comparators are set for a particular threshold value of particular gas concentration. When the concentration of the gas exceeds this threshold value, the digital pin goes high. The analog pin is used mostly to measure the concentration of the gas [13].

Figure 4: A gas sensor.

2.6.1 Different Types of Gas sensors

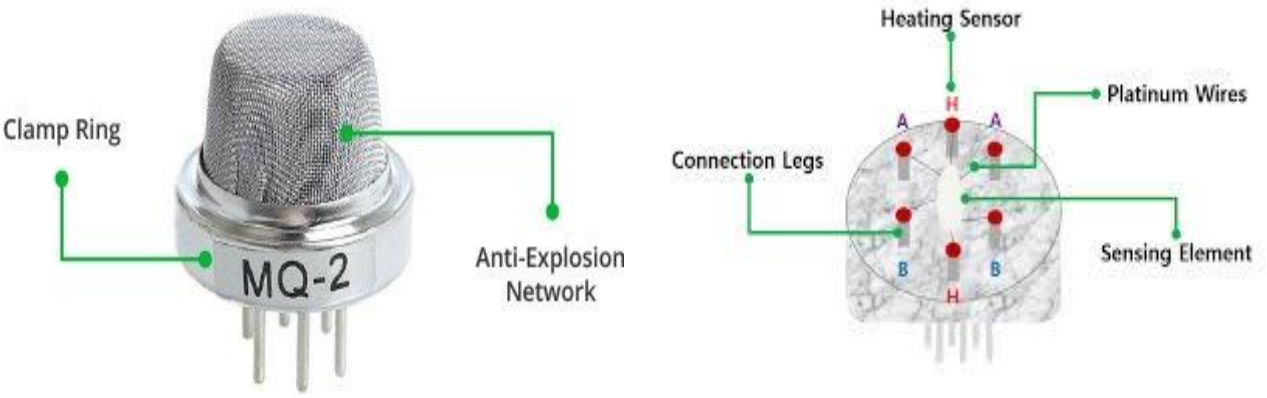
Gas sensors are typically classified based on the type of the sensing element it is built with into various types. Gas sensors are classified based on the sensing element that are generally used in various applications:

- Metal Oxide based gas Sensor.
- Optical gas Sensor.
- Electrochemical gas Sensor.
- Capacitance-based gas Sensor.
- Calorimetric gas Sensor.
- Acoustic based gas Sensor.

2.6.2 Internal of Gas Sensor

The sensor is generally enclosed in two layers of fine stainless steel mesh called Anti-explosion network. As we are sensing flammable gases, it ensures that heater element inside the sensor will not cause an explosion. It also provides protection for the sensor and filters out suspended particles so that only gaseous elements are able to pass through the chamber. The mesh is bound to rest of the body via a copper plated clamping ring [13].

Figure 5: Internal of gas sensor.

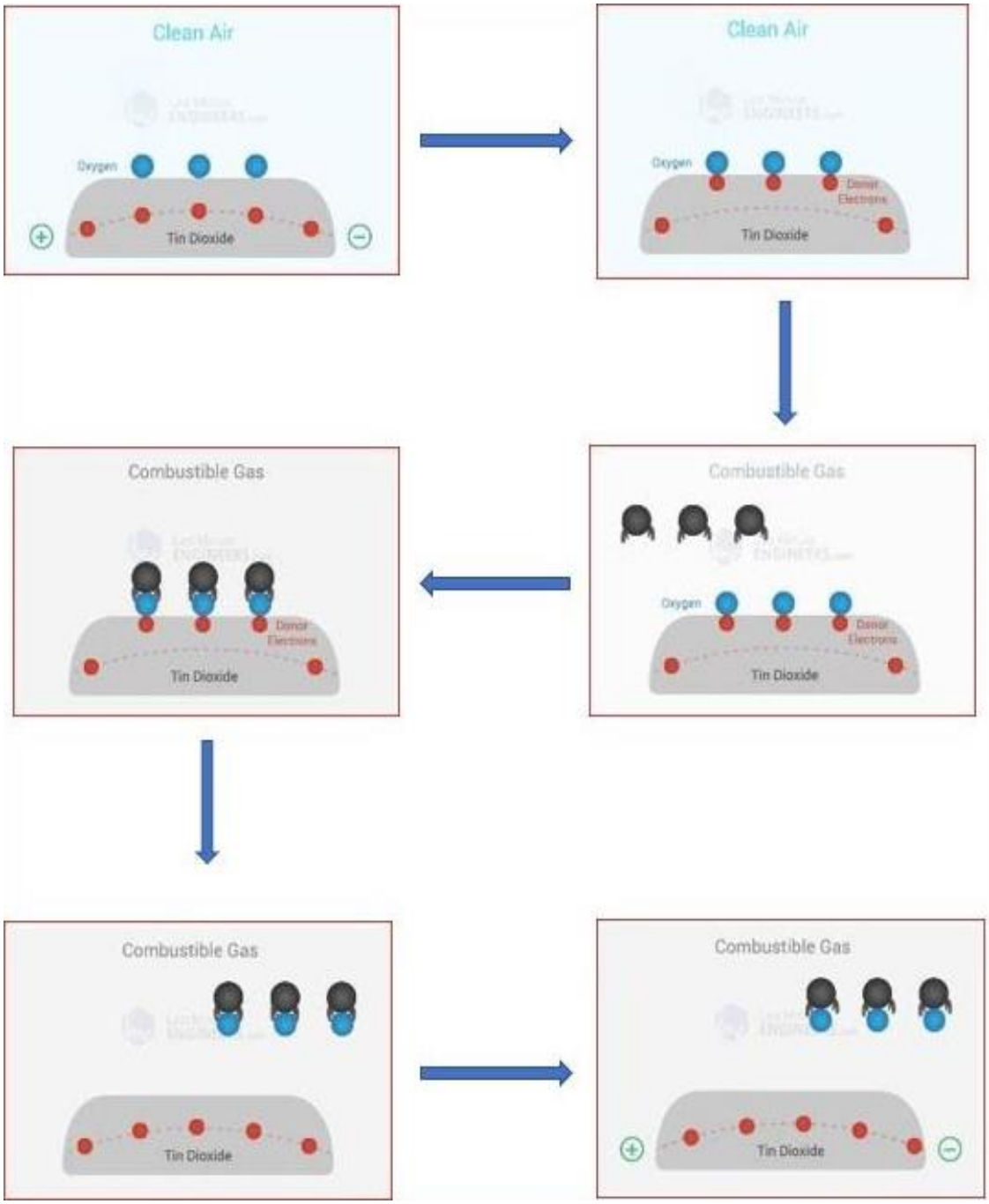


This is the look of the sensor when outer mesh is removed. The star-shaped structure is formed by the sensing element and six connecting legs (as shown above) that extend beyond the Bakelite base. Out of six, two leads (H) are responsible for heating the sensing element and are connected through Nickel-Chromium coil, well known conductive alloy. The remaining four leads (A & B) responsible for output signals are connected using Platinum Wires. These wires are connected to the body of the sensing element and convey small changes in the current that passes through the sensing element. The tubular sensing element is made up of Aluminum Oxide (AL2O3) based ceramic and has a coating of Tin Dioxide (SnO2). The Tin Dioxide is the most important material as it is sensitive towards combustible gases. However, the ceramic substrate merely increases heating efficiency and ensures the sensor area is heated to a working temperature constantly. So, the Nickel-Chromium coil and Aluminum Oxide based ceramic forms a Heating System; while Platinum wires and coating of Tin Dioxide forms a Sensing System [13].

2.6.3 Working of Gas Sensor

When tin dioxide which is semiconductor particles, is heated in air at high temperature, oxygen is adsorbed on the surface. In clean air, donor electrons in tin dioxide are attracted toward oxygen which is absorbed on the surface of the sensing material. This prevents electric current flow. In the presence of reducing gases, the surface density of adsorbed oxygen decreases as it reacts with the reducing gases so electric current flow increases. Electrons are then released into the tin dioxide, allowing current to flow freely through the sensor [13].

Figure 6: Pictorial representation of working of a gas sensor.



2.6.4 List of Gas Sensors and gases they sense

Sensor Name	Gas to measure
MQ-2	Methane, Butane, LPG, Smoke
MQ-3	Alcohol, Ethanol, Smoke
MQ-4	Methane, CNG Gas
MQ-5	Natural gas, LPG
MQ-6	LPG, butane
MQ-7	Carbon Monoxide
MQ-8	Hydrogen Gas
MQ-9	Carbon Monoxide, flammable gasses
MQ131	Ozone
MQ135	Air Quality
MQ136	Hydrogen Sulphide gas
MQ137	Ammonia
MQ138	Benzene, Toluene, Alcohol, Propane, Formaldehyde gas, Hydrogen
MQ214	Methane, Natural Gas
MQ216	Natural gas, Coal Gas

MQ303A	Alcohol, Ethanol, smoke
MQ306A	LPG, butane
MQ307A	Carbon Monoxide
MQ309A	Carbon Monoxide, flammable gas

2.6.5 Applications of Gas Sensors

- Used in industries to monitor the concentration of the toxic gases.
- Used in households to detect an emergency incident.
- Used at oil rig locations to monitor the concentration of the gases those are released.
- Used at hotels to avoid customers from smoking.
- Used in air quality check at offices.
- Used in air conditioners to monitor the CO2 levels.
- Used in detecting fire.
- Used to check concentration of gases in mines.
- Breath analyzer.

Chapter 3

Project design

This chapter presents the designing of the project. A tour from the scratch to the final product is taken in this chapter. This chapter specifies all the important components and other software and hardware used. As technology advancements are made in the field of electronics and electrical engineering, product design and development professionals must stay on top of the latest trends and advancements.

3.1 Introduction

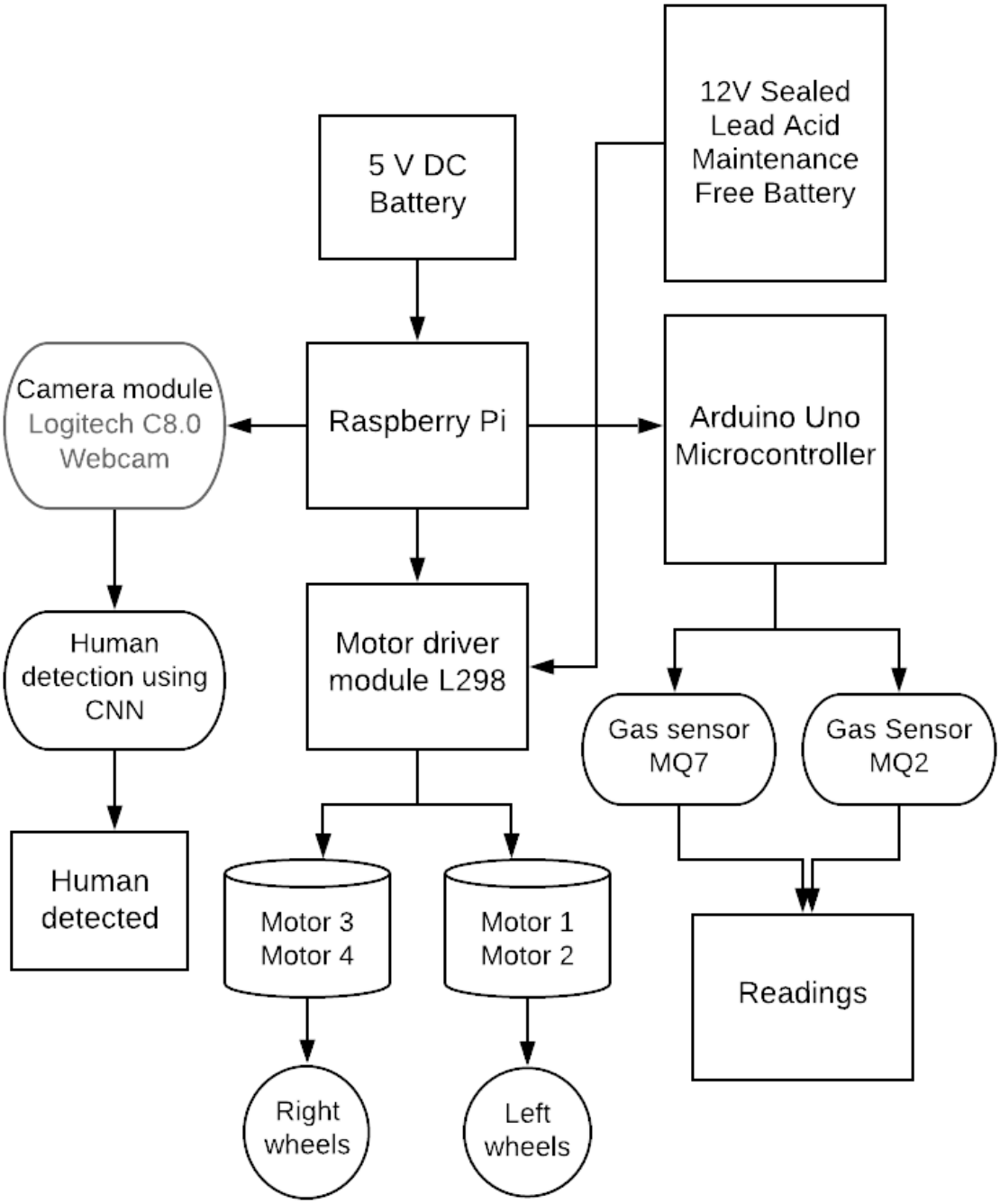
Recent research in the robotics sector and the relevant sectors such as computer vision and sensors, have significantly increased the competitiveness of components needed in automated systems that can perform during a tunnel collapse for inspection and assessment. Such an integrated and automated system is highly needed today. The work that will be presented in this publication, offers an automated system integrating all the required components

Human Detection is a branch of Object Detection. Object Detection is the task of identifying the presence of predefined types of objects in an image. This task involves both identification of the presence of the objects and identification of the rectangular boundary surrounding each object (i.e. Object Localization). An object detection system which can detect the class “Human” can work as a Human Detection System. This is done using a camera module and Raspberry pi. Various algorithms are used for this process. In this project Human detection is done using Haar Cascade algorithm and Deep Convolutional Neural Networks.

The mobility of the robot needs hardware like motor driver modules, motors, battery and minicomputer like Raspberry pi for functioning. Arduino board is used for interfacing sensors and other hardware components. It makes a hardware project simple. All these components are assembled together on the fiber body along with the wheels for making the robot move. In this chapter all the important components and other software and hardware used are discussed.

3.2 Block diagram of robotic system for monitoring environment and human body detection inside tunnel

Figure 7: Block diagram of working of the project



3.3 Problem statement

In this project, the aim is to:

- Start with Raspberry pi and interfacing the camera module with it. Installing the operating system and various packages of OpenCV on Raspberry pi for body detection.
- Installing OpenCV, which provides simple tools for video input and output.
- Configuring Raspberry pi for body detection using deep convolution neural networks for real-time object detector with OpenCV for which there is a need to:
    1. Access webcam/video streams in an efficient manner.
    2. Apply object detection to each frame.
- Assembling the body of the robot by attaching four wheels to four geared motors of 1000 RPM and using a motor driver module L298 to run it.
- Interfacing the motor driver module with the Raspberry pi to run the motors and there by wheels.
- Configuring the Arduino Uno with the Raspberry pi 3 Model B so as to make them work simultaneously.
- Connecting the sensors MQ-2 and MQ-7 with Arduino Uno for detecting the gas levels of various gases inside tunnel.
- Finally, checking for the mobility of the robot, working of the sensors and camera module for various tasks.

3.4 Hardware components

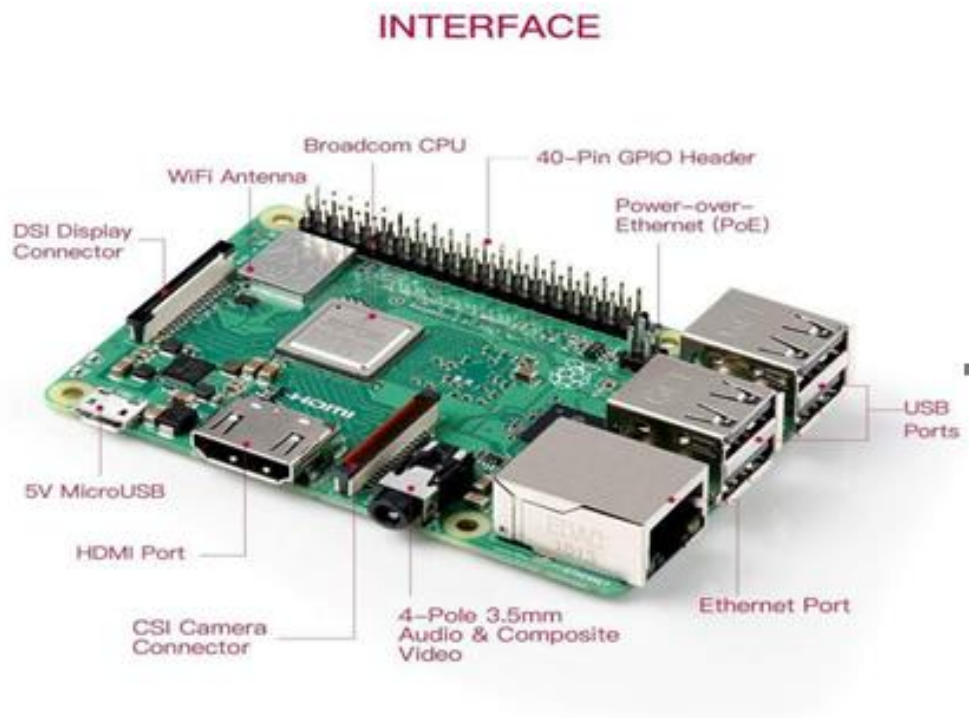
Hardware components employed in this project are:

1. Raspberry pi 3 Model B
2. Arduino Uno
3. Camera module
4. Motor driver Module L298
5. Sensors MQ-2 and MQ-7
6. Geared dc motors of 1000 RPM
7. A 12V battery to power up everything

3.4.1 Raspberry pi

The Raspberry Pi 3 Model B consists of a 64-bit quad core processor running at 1.4GHz, dual-band 2.4GHz and 5GHz wireless LAN, Bluetooth 4.2/BLE, faster Ethernet, and PoE capability via a separate PoE HAT. The dual-band wireless LAN comes with modular compliance certification, allowing the board to be designed into end products with significantly reduced wireless LAN compliance testing, improving both cost and time to market [1].

Figure 8: Raspberry pi 3 B model



The Raspberry Pi 3 Model B specifications are as follows: -

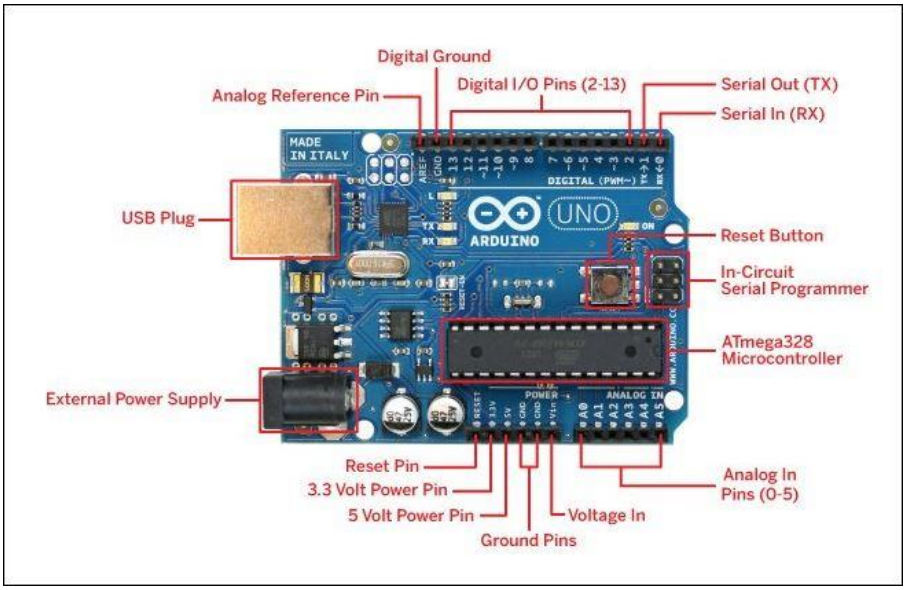
- Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz
- 1GB LPDDR2 SDRAM
- 2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE
- Gigabit Ethernet over USB 2.0 (maximum throughput 300 Mbps)
- Extended 40-pin GPIO header
- Full-size HDMI

- USB 2.0 ports
- CSI camera port for connecting a Raspberry Pi-camera
- Micro SD port for loading your operating system and storing data
- 5V/2.5A DC power input
- Power-over-Ethernet (PoE) support (requires separate PoE HAT)

3.4.2 Arduino Uno

Arduino Uno is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator (CSTCE16M0V53-R0), a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started [10].

Figure 9: Arduino Uno



Features of the Arduino Uno:

- Microcontroller: ATmega328
- Operating Voltage: 5V
- Input Voltage (recommended): 7-12V

- Input Voltage (limits): 6-20V
- Digital I/O Pins: 14 (of which 6 provide PWM output)
- Analog Input Pins: 6
- DC Current per I/O Pin: 40 mA and 50 mA (for 3.3 pin)
- Flash Memory: 32 KB of which 0.5 KB used by bootloader
- SRAM: 2 KB (ATmega328)
- EEPROM: 1 KB (ATmega328)
- Clock Speed: 16 MHz

3.4.3 Camera module

The camera module is used to take real time inputs for the onboard image processing operations carried out on the Raspberry pi. To ensure high resolution video input the Logitech C8.0 model is been used.

Figure 10: Camera module

Logitech C8.0 Webcam

Features: -

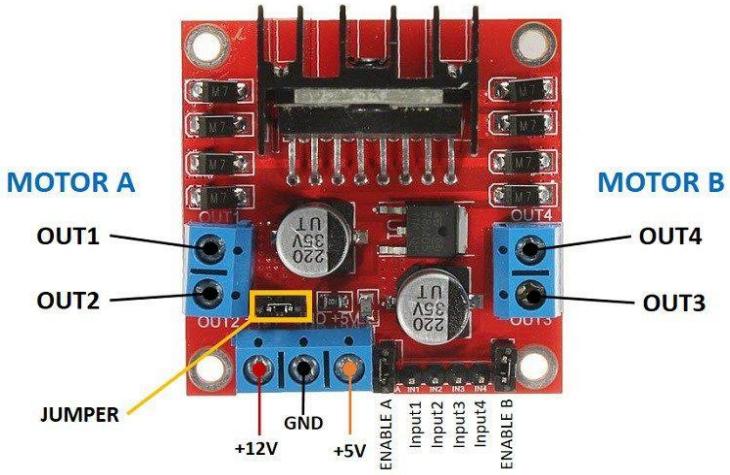
- It has interpolated 8.0 Mega Pixel (Image resolution)
- It has interpolated 4.0 Mega Pixel (Video Resolution)
- High quality 5G wide angle lenses
- 6 LEDs for night vision, with brightness controller
- Snapshot button for still image capture

- Built-in highly sensitive USB microphone
- 4x Digital zoom and auto face tracking system

3.4.4 Motor driver Module L298

The L298 Driver is a high voltage, high current dual full bridge driver designed to accept standard TTL logic levels and drive inductive loads such relays, solenoids, DC and stepping motors. Two enable inputs are provided to enable or disable the device independently of the input signals. The emitters of the lower transistors of each bridge are connected together the corresponding external terminal can be used for the connection of an external sensing resistor [14].

Figure 11: Motor driver module, L298



Features:

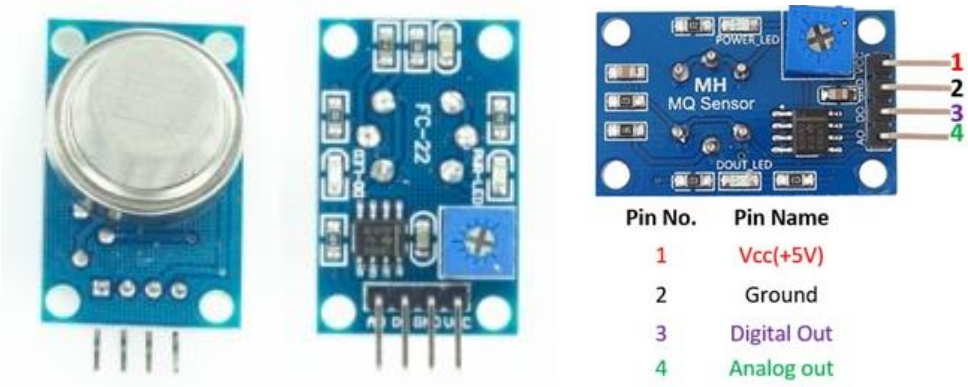
- Operating supply voltage up to 46 V
- Total DC current up to 4 A
- Low saturation voltage
- Over temperature protection.
- Logical "0" input voltage upto1.5 V (HIGH NOISE IMMUNITY)
- Two motor direction indicator LEDs
- An onboard user-accessible 5V low-dropout regulator
- Schottky EMF-protection diodes
- Screw-terminals for power and motor connections.

3.4.5 Sensors: MQ-2 and MQ-7

3.4.5.1 MQ-2 sensor

Gas Sensor(MQ-2) module is useful for gas leakage detection (home and industry). It is suitable for detecting H2, LPG, CH4, CO, Alcohol, Smoke or Propane. Due to its high sensitivity and fast response time, measurement can be taken as soon as possible. The sensitivity of the sensor can be adjusted by potentiometer [11].

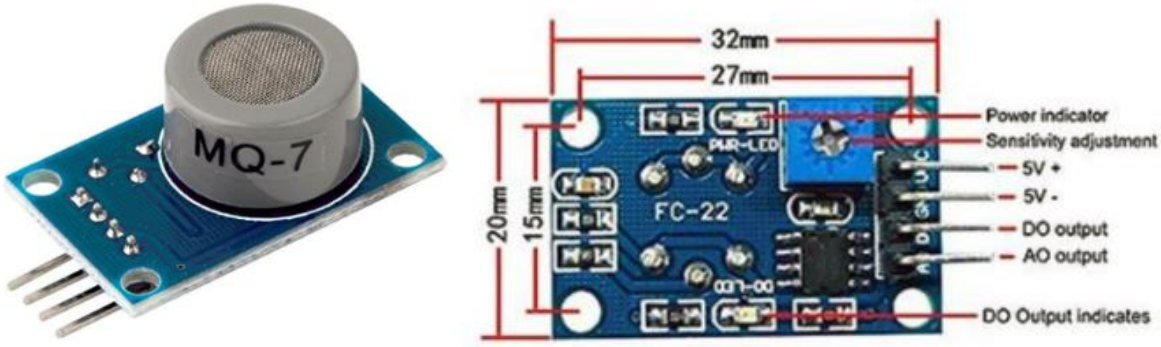
Figure 12: MQ-2 gas sensor



3.4.5.2 MQ-7 sensor

The MQ-7 is a simple-to-use Carbon Monoxide (CO) sensor suitable for sensing CO concentrations in the air. It can detect CO-gas concentrations anywhere from 20 to 2000ppm. MQ-7 is a high sensitivity to carbon monoxide and stable and long-life span [12].

Figure 13: MQ-7 gas sensor



3.4.6 Other Hardware components

Figure 14: Hardware components

The hardware components include a 5V battery to power up Raspberry pi and a 12V Sealed Lead Acid Maintenance Free Battery for motor driver module L298 and a 7*4 cm robotic wheel, Johnson Geared Motor of 1000 RPM with following specifications:

- Base Motor RPM: 18000
- Operating Voltage: 6-18 V
- Rated Voltage: 12 V
- Rated Torque: 0.8 kg-cm
- Stall Torque: 3.5 kg-cm
- Gearbox Dimensions: 25×37 mm

3.5 Software components

3.5.1 OpenCV

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code [2].

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in

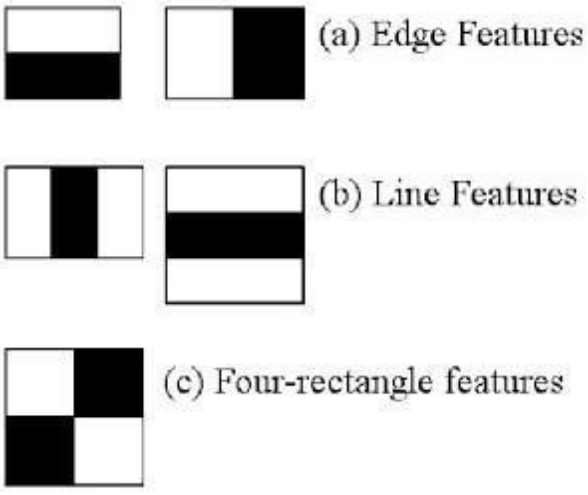
videos, track camera movements, track moving objects, extra, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. The library is used extensively in companies, research groups and by governmental bodies. It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV is written natively in C++ [2].

3.5.2 Haar Cascade for Human Detection

Haar Cascade is a machine learning object detection algorithm proposed by [Paul Viola and Michael Jones in their paper "Rapid Object Detection using a Boosted Cascade of Simple Features" in 2001]. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images (where positive images are those where the object to be detected is present, negative are those where it is not). It is then used to detect objects in other images. OpenCV offers pre-trained Haar cascade algorithms, organized into categories (faces, eyes and so forth), depending on the images they have been trained on.

The idea of Haar cascade is extracting features from images using a kind of 'filter', similar to the concept of the convolutional kernel. These filters are called Haar features and look like that:

Figure 15: Filters used to extract Haar features



The idea is passing these filters on the image, inspecting one portion (or window) at the time. Then, for each window, all the pixel intensities of, respectively, white and black portions are summed. Finally, the value obtained by subtracting those two summations is the value of the feature extracted. Ideally, a great value of a feature means it is relevant.

Figure 16: Real intensities of pixels

3.5.3 Deep Convolutional Neural Network for Human Detection

CNN architecture differs from the traditional multilayer perceptrons (MLP) to ensure some degree of shift and distortion invariance. They combine three architectural ideas to do the same:

- Local Receptive Fields
- Shared weights
- Spatial and Temporal Subsampling

The input and output of each stage are sets of arrays called as feature maps. In the case of a colored image, each feature map would be a 2D array containing a color channel of the input image, a 3D array for a video and a 1D array for an audio input. The output stage represents features extracted from all locations on the input. Each stage generally consists of a convolution layer, nonlinearity and a pooling layer. A single or multiple fully connected layers are present after several convolution and pooling layers [8].

A. Convolution layer

This layer is the core building block of a CNN. The layer's parameters consist of learnable kernels or filters which extend through the full depth of the input. Each unit of this layer receives

inputs from a set of units located in small neighborhood in the previous layer. Such a neighborhood is called as the neuron's receptive field in the previous layer. During the forward pass each filter is convolved with input which produces a map. When multiple such feature maps that are generated from a multiple filter are stacked, they form the output of the convolution layer. The weight vector that generates the feature map is shared which reduces the model complexity.

B. Non-linearity Layer

This is a layer of neurons which apply various activation functions. These functions introduce nonlinearities which are desirable for multi-layer networks. Functions like Rectified Linear Units (ReLU) are preferable because neural networks train several times faster.

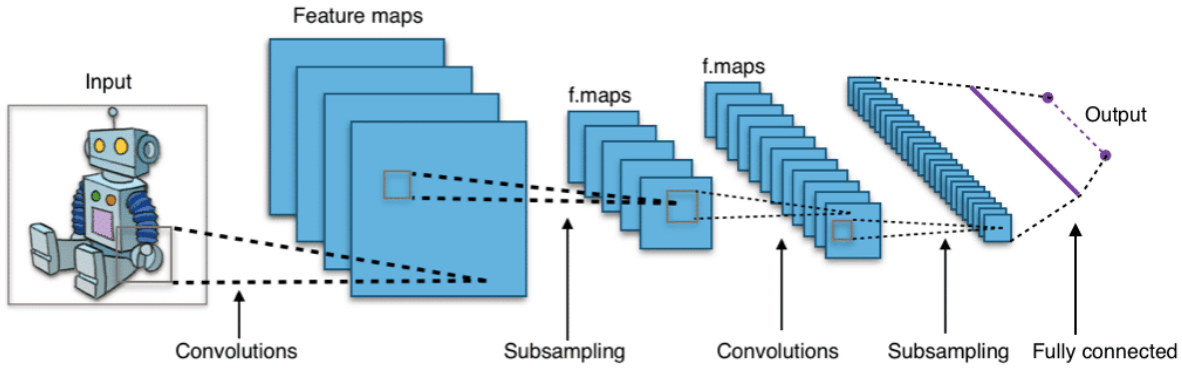
C. Pooling Layer

The Convolution layer may be followed by the pooling layer which takes small rectangular blocks from the convolution layer and subsamples it to produce a single maximum output from the block. Pooling layer progressively reduces the spatial size of the representation, thus reducing the parameters to be computed. It also controls overfitting.

D. Fully Connected Layer

There maybe one or more fully-connected layers that perform high level reasoning by taking all neurons in the previous layer and connecting them to every single neuron in the current layer to generate global semantic information.

Figure 17: Convolution neural network architecture



Chapter 4

Implementation and experimentation related to project work

This chapter presents the implementation of the project. The Implementation section is similar to the Specification and Design section. In latter the system is described, but it does so at a finer level of detail, down to the code level in former. This section is about the realization of the concepts and ideas developed earlier. It can also describe any problems that have arisen during implementation and how to dealt with them.

4.1 Steps to configure Raspberry pi

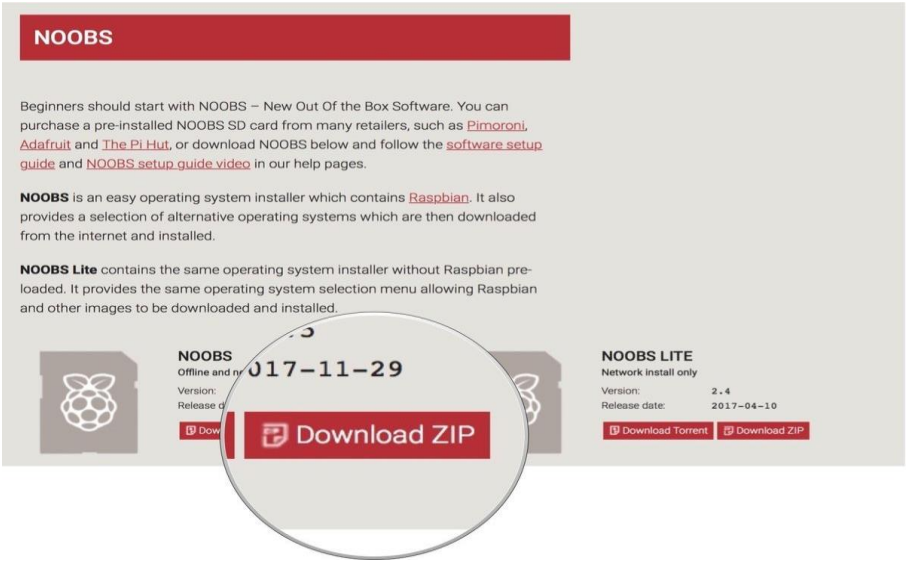
4.1.1 How to Download NOOBS onto the micro SD card

1. Download the ZIP file of NOOBS Version 3.0.0. It is a large file and will take a while to complete.
2. Extract the NOOBS file into the computer.
3. Drag and drop all selected NOOBS files into the SD card icon on the desktop.
4. Right-click on the SD card icon and select "Eject [SD Card Name]".
5. Remove the card reader from the computer.
6. Insert the SD card back to the rasp berry pi.

4.1.2 Configure your Raspberry Pi

1. Click Menu in the upper left corner of the screen.
2. Select Preferences in the dropdown menu.
3. Select Raspberry Pi Configuration under Preferences.
4. When the configuration window appears, click on the Localization tab.
5. Click on Set Locale… to set your location.
6. Click on Set time zone… to set your local time.
7. Click on Set Keyboard… to set your keyboard language [1].

Figure 18: NOOBS Zip file download.



4.2 Steps to implement body detection

4.2.1 Accessing the webcam

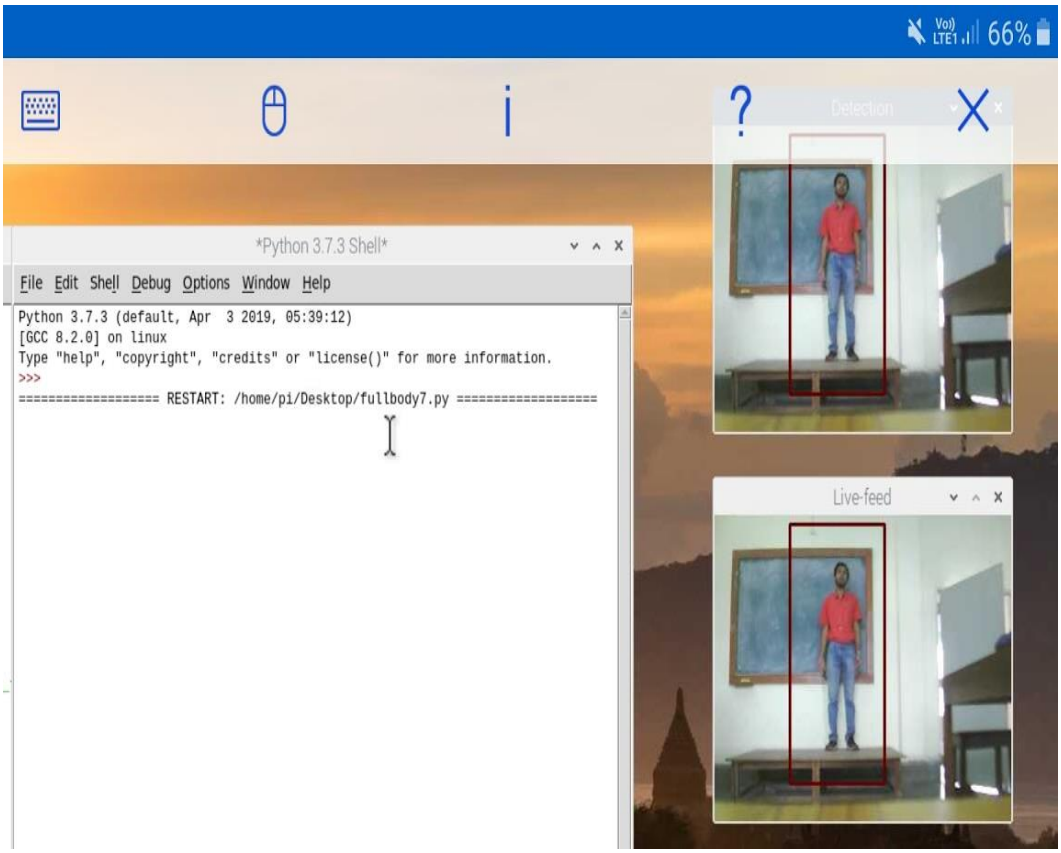
```
import numpy as np
import cv2
cap = cv2.VideoCapture(0)
def make_240p():
    cap.set(3, 352)
    cap.set(4, 240)
def change_res(width, height):
    cap.set(3, width)
    cap.set(4, height)
make_240p()
while 1:
    ret, img = cap.read()
    cv2.imshow('Live-feed',img)
        break
cap.release()
cv2.destroyAllWindows()
cv2.waitKey(1)
```

4.2.2 Human Body Detection using Haar Cascade file

```
import numpy as np
import cv2
cap = cv2.VideoCapture(0)
def make_240p():
    cap.set(3, 352)
    cap.set(4, 240)
def change_res(width, height):
    cap.set(3, width)
    cap.set(4, height)
make_240p()
face_detector = cv2.CascadeClassifier('/home/pi/Desktop/haarcascade_fullbody (1).xml')
while 1:
    ret, img = cap.read()
    faces = face_detector.detectMultiScale(img, 1.099,1)
    for (x,y,w,h) in faces:
        cv2.rectangle(img,(x,y),(x+w,y+h),(0,0,100),2)
        cv2.imshow('Detection',img)
    cv2.imshow('Live-feed',img)
    k = cv2.waitKey(3) & 0xff
    if k == 2:
        break
cap.release()
cv2.destroyAllWindows()
```

Output:

Figure 19: Haar Cascade Output in Raspberry pi.



4.2.3 Human Body Detection using CNN

USAGE

python real_time_object_detection.py --prototxt MobileNetSSD_deploy.prototxt.txt --model MobileNetSSD_deploy.caffemodel

import the necessary packages

from imutils.video import VideoStream

from imutils.video import FPS

import numpy as np

import argparse

import imutils

import time

import cv2

construct the argument parse and parse the arguments

```
ap = argparse.ArgumentParser()
ap.add_argument("-p", "--prototxt", required=True,
	help="path to Caffe 'deploy' prototxt file")
ap.add_argument("-m", "--model", required=True,
	help="path to Caffe pre-trained model")
ap.add_argument("-c", "--confidence", type=float, default=0.2,
	help="minimum probability to filter weak detections")
args = vars(ap.parse_args())
# initialize the list of class labels MobileNet SSD was trained to
# detect, then generate a set of bounding box colors for each class
CLASSES = ["background", "aeroplane", "bicycle", "bird", "boat",
	"bottle", "bus", "car", "cat", "chair", "cow", "diningtable",
	"dog", "horse", "motorbike", "person", "pottedplant", "sheep",
	"sofa", "train", "tvmonitor"]
COLORS = np.random.uniform(0, 255, size=(len(CLASSES), 3))
# load our serialized model from disk
print("[INFO] loading model...")
net = cv2.dnn.readNetFromCaffe(args["prototxt"], args["model"])
# initialize the video stream, allow the cammera sensor to warmup,
# and initialize the FPS counter
print("[INFO] starting video stream...")
vs = VideoStream(src=0).start()
time.sleep(2.0)
fps = FPS().start()
# loop over the frames from the video stream
while True:
	# grab the frame from the threaded video stream and resize it
	# to have a maximum width of 400 pixels
	frame = vs.read()
	frame = imutils.resize(frame, width=400)
      # grab the frame dimensions and convert it to a blob
```

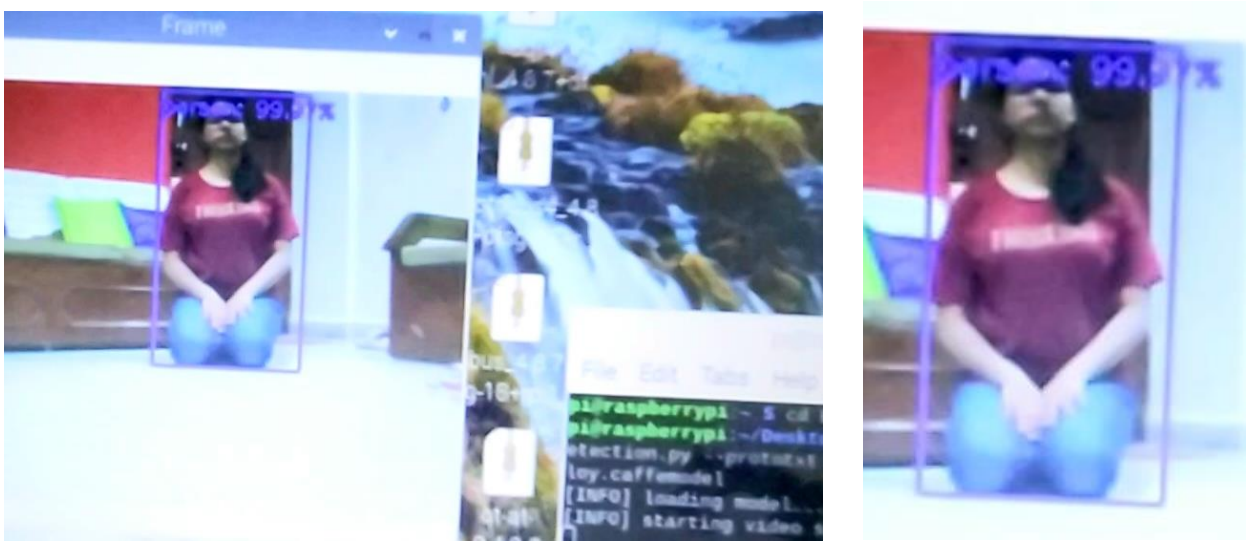
```
    (h, w) = frame.shape[:2]
    blob = cv2.dnn.blobFromImage(cv2.resize(frame, (300, 300)),
            0.007843, (300, 300), 127.5)
# pass the blob through the network and obtain the detections and
    # predictions
    net.setInput(blob)
    detections = net.forward()
# loop over the detections
    for i in np.arange(0, detections.shape[2]):
            # extract the confidence (i.e., probability) associated with
            # the prediction
            confidence = detections[0, 0, i, 2]
        # filter out weak detections by ensuring the `confidence` is
            # greater than the minimum confidence
            if confidence > args["confidence"]:
                    # extract the index of the class label from the
                    # `detections`, then compute the (x, y)-coordinates of
                    # the bounding box for the object
                    idx = int(detections[0, 0, i, 1])
                    box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
                    (startX, startY, endX, endY) = box.astype("int")

                    # draw the prediction on the frame
                    label = "{}: {:.2f}%".format(CLASSES[idx],
                            confidence * 100)
                    cv2.rectangle(frame, (startX, startY), (endX, endY),
                            COLORS[idx], 2)
                    y = startY - 15 if startY - 15 > 15 else startY + 15
                    cv2.putText(frame, label, (startX, y),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.5, COLORS[idx], 2)
# show the output frame
```

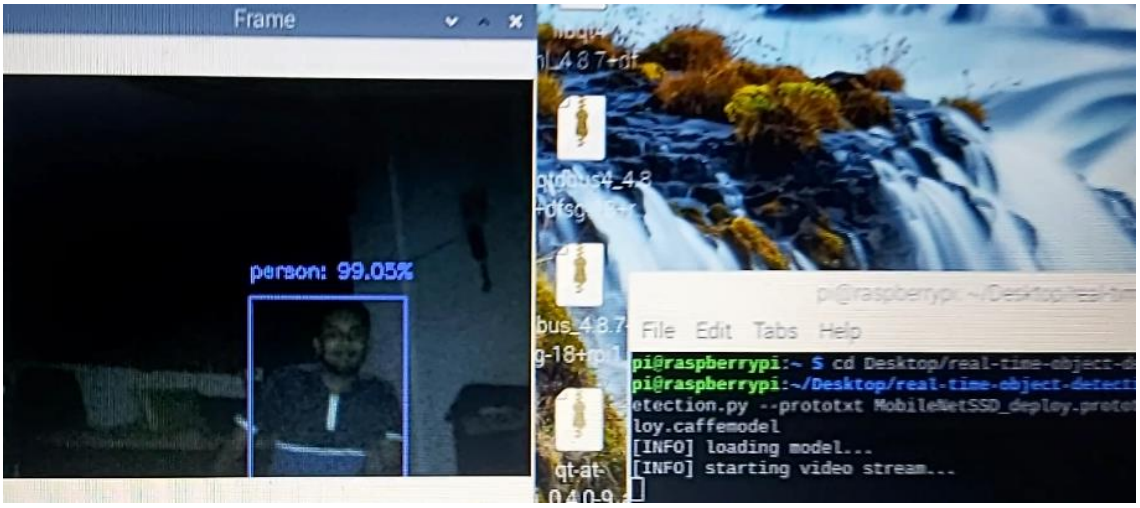
```
        cv2.imshow("Frame", frame)
        key = cv2.waitKey(1) & 0xFF
    # if the `q` key was pressed, break from the loop
        if key == ord("q"):
            break
    # update the FPS counter
        fps.update()
# stop the timer and display FPS information
fps.stop()
print("[INFO] elapsed time: {:.2f}".format(fps.elapsed()))
print("[INFO] approx. FPS: {:.2f}".format(fps.fps()))
# do a bit of cleanup
cv2.destroyAllWindows()
vs.stop()
```

Output [9]:

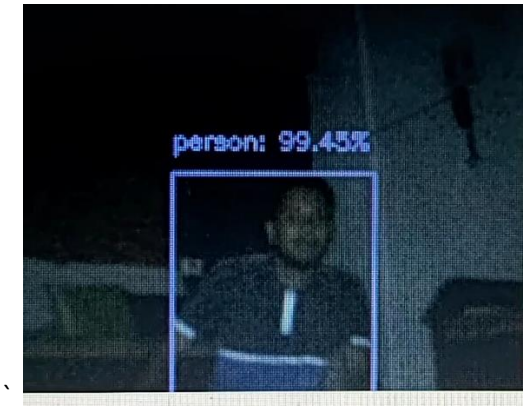
Figure 20: CNN Output in Raspberry pi.



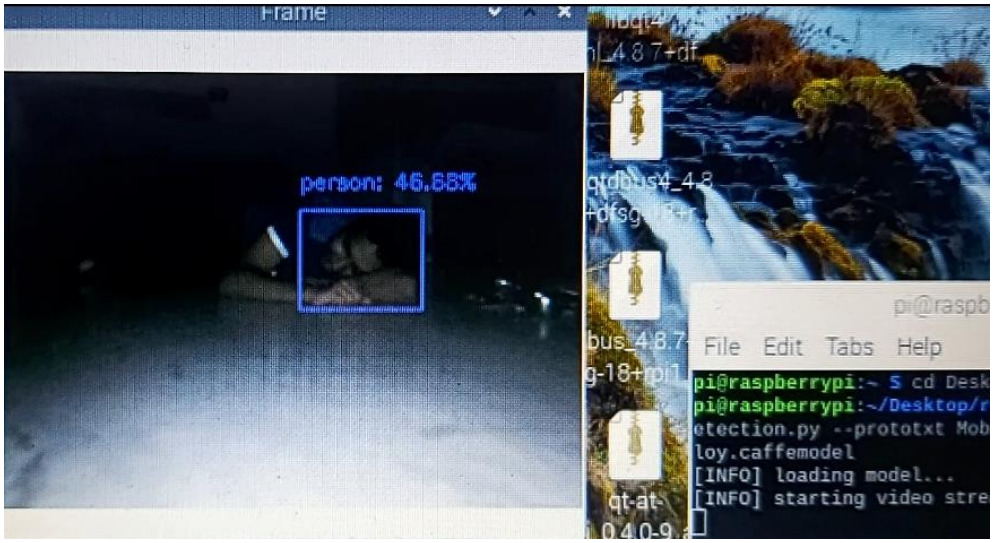
a. Detecting human body in a well-lighted place



b. Detecting human body in dark



c. The accuracy of detection



d. Detecting a human body lying on the ground

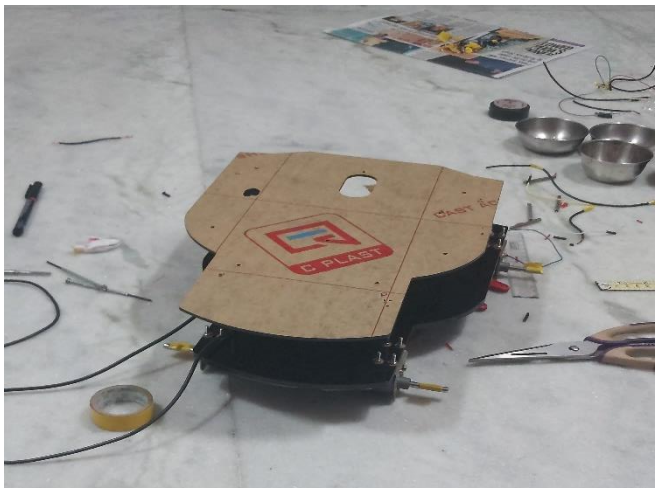
4.3 Steps to configure motors

4.3.1 Code for running the motors using motor driver Module L298

```
import curses
import RPi.GPIO as GPIO
import numpy as np
GPIO.setmode(GPIO.BOARD) #set GPIO numbering mode and define output pins
GPIO.setup(7,GPIO.OUT)
GPIO.setup(11,GPIO.OUT)
GPIO.setup(13,GPIO.OUT)
GPIO.setup(15,GPIO.OUT)
screen = curses.initscr()# Get the curses window, turn off echoing of keyboard to screen, turn on
curses.noecho()
curses.cbreak()
screen.keypad(True)
try:
        while True:
            char = screen.getch()
            if char == ord('m'):
                break
            elif char == curses.KEY_UP:
                GPIO.output(7,False)
                GPIO.output(11,True)
                GPIO.output(13,False)
                GPIO.output(15,True)
            elif char == curses.KEY_DOWN:
                GPIO.output(7,True)
                GPIO.output(11,False)
                GPIO.output(13,True)
                GPIO.output(15,False)
            elif char == curses.KEY_RIGHT:
```

```
            GPIO.output(7,True)
            GPIO.output(11,False)
            GPIO.output(13,False)
            GPIO.output(15,True)
        elif char == curses.KEY_LEFT:
            GPIO.output(7,False)
            GPIO.output(11,True)
            GPIO.output(13,True)
            GPIO.output(15,False)
        elif char == ord('/'):
            GPIO.output(7,False)
            GPIO.output(11,False)
            GPIO.output(13,False)
            GPIO.output(15,False)
finally:
    curses.nocbreak(); screen.keypad(0); curses.echo()
    curses.endwin();GPIO.cleanup()
```

Figure 21: Architecture of the Robot.



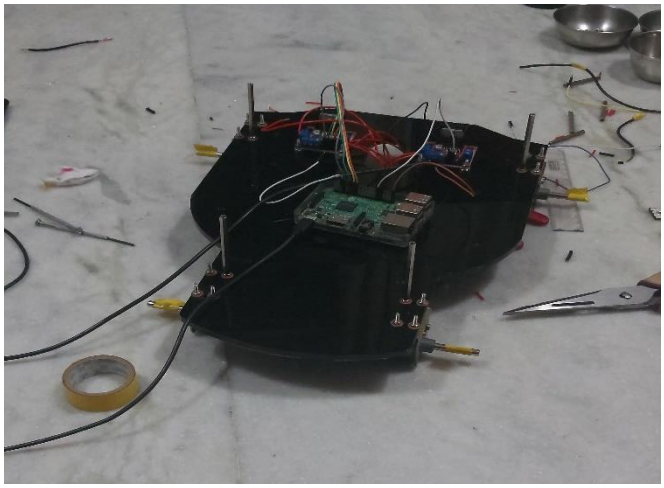


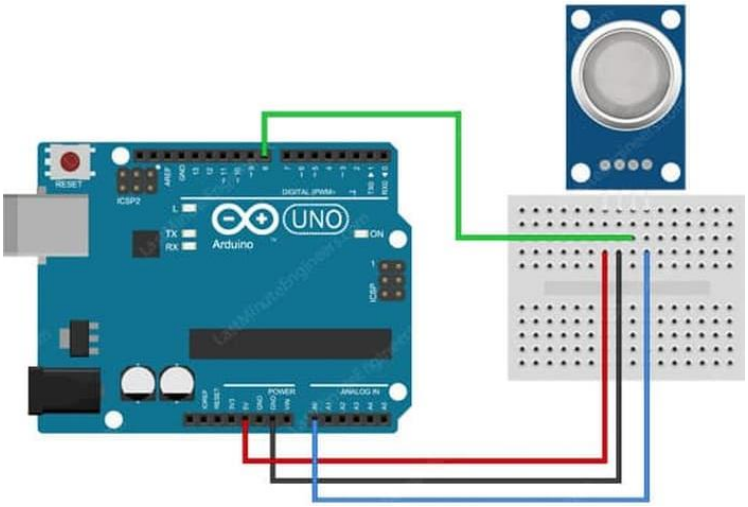
Figure 22: The final body of the Robot

4.4 Steps to configure gas sensors and interfacing with Arduino Uno

Following steps are to be followed for configuring the gas sensors and interfacing with Arduino Uno:

1. Connect VCC pin to the 5V pin on the Arduino and connect GND pin to the Ground pin on the Arduino.
2. Connect D0 output pin on the module to Digital pin#8 on the Arduino and A0 output pin on the module to Analog pin#0 on the Arduino.
3. Now, the gas sensor is ready to be used for taking the readings of the gas level.

Figure 23: Interfacing gas sensors with Arduino Uno



4.4.1 Code for MQ2 Gas Sensor

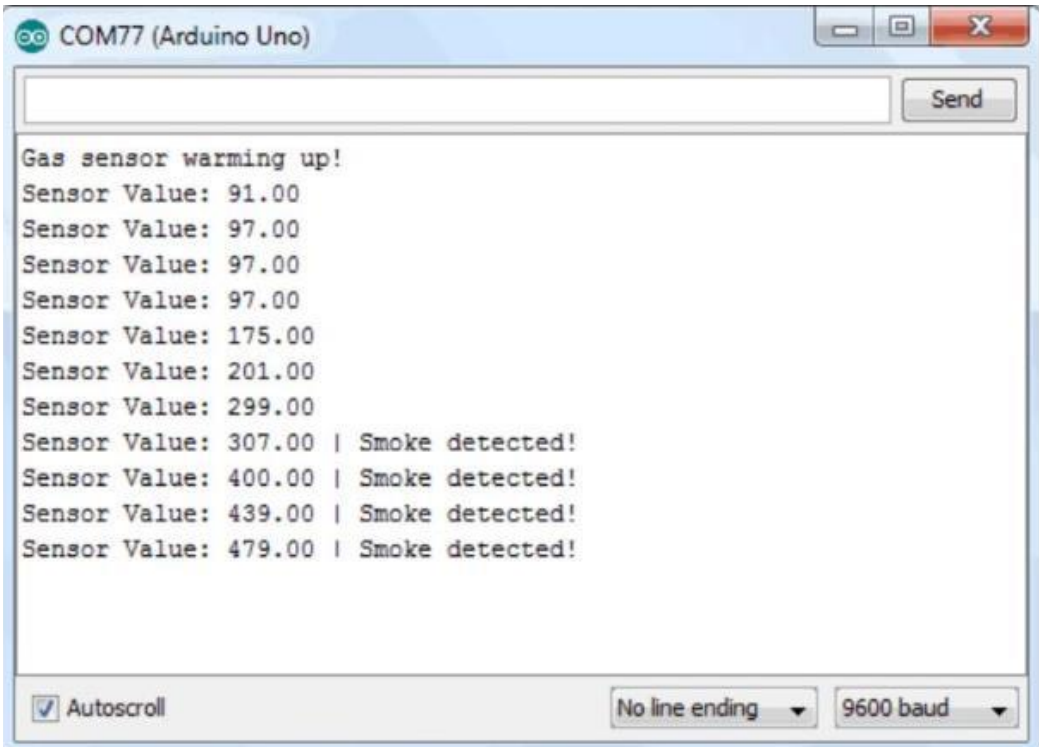
```
#define MQ2pin (0)
float sensorValue;  //variable to store sensor value
void setup()
{
  Serial.begin(115200); // sets the serial port to 115200
  Serial.println("Gas sensor warming up!");
  delay(20000); // allow the MQ-2 to warm up
}
void loop()
{
  sensorValue = analogRead(MQ2pin); // read analog input pin 0
  Serial.print("Sensor Value: ");
 Serial.print(sensorValue);
 if(sensorValue > 300)
  {
    Serial.print(" | Smoke detected!");
  }
  Serial.println("");
  delay(2000); // wait 2s for next reading
}
```

4.4.2 Code for MQ7 gas sensor

```
#define MQ7pin (10)
float sensorValue;  //variable to store sensor value
void setup()
{
  Serial.begin(9600); // sets the serial port to 9600
  Serial.println("Gas sensor warming up!");
  delay(20000); // allow the MQ-7 to warm up
```

```
}
void loop()
{
  sensorValue = analogRead(MQ7pin); // read analog input pin 10
  Serial.print("Sensor Value: ");
 Serial.print(sensorValue);
 if(sensorValue > 250)
  {
    Serial.print(" | Smoke detected!");
  }
  Serial.println("");
  delay(2000); // wait 2s for next reading
}
```

Figure 24: Reading on the gas sensors



4.5 Video links for the output: -

1. Mobility of the robot:

https://drive.google.com/drive/u/1/folders/1YbPLKTssIzUhXe95eYGNve5botEFvJBn

2. Body detection using CNN algorithm with Raspberry Pi and Pi-camera:

a) Day light:

https://drive.google.com/drive/u/1/folders/18mzI3GsPEKryOqt533XU0MxhmknK580a

b) Night vision:

https://drive.google.com/drive/u/1/folders/18qnUrICNXKV1U61OMghgVbaDJXciguMq

c) Night vision with person lying on the ground:

https://drive.google.com/drive/u/1/folders/18ts6slEtDeJ5JblxjaYj4-8l-UM4QQq5

3. Arduino output for gas sensor reading:

https://drive.google.com/drive/u/1/folders/19CwhsFyhOC1td5LR5d78z-cnBfcE1rL8

Chapter 5

Conclusions and further work

This chapter presents conclusion of all the work required for the completion of this project. It also highlights the future scope where this project can be extended. There are many areas where the given project finds its use. The project can be extended in terms of various other parameters and aspects. This chapter technically ends the scope of the project for this term.

5.1 Conclusions

This report presents an overview of the design, implementation, testing and performance of an innovative robotic system controlled wirelessly by a remote user. This can be used for surveillance of the tunnel in case of collapse and can be used to detect human body as well as hazardous gas levels inside the tunnel.

Results observed in the comparative study with other traditional methods suggest that CNN gives better accuracy and boosts the performance of the system due to unique features like shared weights and local connectivity. CNN is better than other deep learning methods in applications pertaining to computer vision and natural language processing because it mitigates most of the traditional problems. It is a preferred method for Human Detection as well [8].

The sensors used in this project are capable of detecting almost all the harmful gases found in the tunnel and give a warning whenever these gases cross a threshold value. The motor driver module L298 is used for running the motors which are then used for running the wheels for the mobility of the robot. This robotic system is a working prototype of how future technological advancement in this area can make tunnel inspection safer for workers by detecting gases and how in case of a collapse this system can be deployed.

5.2 Future scope

This robotic system can be very useful in war, terrorism and sensitive areas. It can also be deployed in forest and mines, the places which possess risk to human life. The robot can be controlled autonomously because of the presence of minicomputer Raspberry pi. Additional hardware can also be interfaced with this system as microcontroller Arduino Uno used in this project makes hardware implementation simple. Further, instead of using Raspberry Pi other minicomputer can be used for this project for faster and better computing. The main drawback of using Raspberry pi is over-loading and over-heating. When many programs run simultaneously on Raspberry pi, CPU usage increases and that causes crashing down of the system. To overcome this drawback, various other options can be considered for the robot in future works. The robotic system can also be extended to tunnels inspection and assessment on daily basis to decrease manpower and increase efficiency and effectiveness to avoid collapses. The technologies which are used in the proposed system are good enough to ensure practical and perfect implementation according to the scope of the project.

Bibliography

1. Tutorial on how-get-started-using-raspberry-pi by raspberrypi.org.
2. Installing OpenCV on your Raspberry pi https://www.pyimagesearch.com/2018/09/26/install-opencv-4-on-your-raspberry-pi/
3. Open CV 3.4 Documentation by opencv.org.
4. Paper on "Rapid Object Detection using a Boosted Cascade of Simple Features" by Paul Viola and Michael Jones published in 2001.
5. Paper on **"**Histograms of oriented gradients for human detection**"** by N. Dalal and B. Triggs published in 2005.
6. Real-time Human Detection in Computer Vision — Part 1; by Madhawa Vidanapathirana https://medium.com/@madhawavidanapathirana/https-medium-com-madhawavidanapathirana-real-time-human-detection-in-computer-vision-part-1-2acb851f4e55
7. Real-time Human Detection in Computer Vision — Part 2; by Madhawa Vidanapathirana https://medium.com/@madhawavidanapathirana/real-time-human-detection-in-computer-vision-part-2-c7eda27115c6
8. Applications of Convolutional Neural Networks by Department of Information Technology, Pune Institute of Computer Technology, Savitribai Phule Pune University, Pune
9. Real time object detection with deep learning and OpenCV https://www.pyimagesearch.com/2017/09/18/real-time-object-detection-with-deep-learning-and-opencv/
10. Documentation on getting started with Arduino Uno by arduino.cc.
11. Carbon Monoxide gas sensor MQ7 by Arduino Project Hub.
12. Smoke detection using MQ2 gas sensor by Arduino Project Hub.

13. Introduction to gas sensors: construction, type, internals and working https://components101.com/articles/introduction-to-gas-sensors-types-working-and-applications
14. Tutorial on how-to-control-a-dc-motor-with-an-l298-controller-and-raspberry-pi by raspberrypi.org.

Appendix A

Datasheet of sensor MQ-2

Model No.			MQ-2
Sensor Type			Semiconductor
Standard Encapsulation			Bakelite(Black Bakelite)
Detection Gas			Combustible gas and smoke
Concentration			300-10000ppm (Combustible gas)
Circuit	Loop Voltage	V_C	≤ 24V DC
	Heater Voltage	V_H	5.0V±0.2V AC or DC
	Load Resistance	R_L	Adjustable
Character	Heater Resistance	R_H	31Ω±3Ω（Room Tem.）
	Heater consumption	P_H	≤ 900mW
	Sensing Resistance	Rs	2KΩ-20KΩ (in 2000ppm C_3H_8)
	Sensitivity	S	Rs(in air)/Rs (1000ppm isobutane) ≥ 5
	Slope	α	≤ 0.6($R_{5000ppm}/R_{3000ppm}$ CH_4)
Condition	Tem. Humidity		20℃±2℃；65%±5%RH
	Standard test circuit		Vc:5.0V±0.1V； V_H: 5.0V±0.1V
	Preheat time		Over 48 hours

Datasheet of sensor MQ-7

Model No.			MQ-7
Sensor Type			Semiconductor
Standard Encapsulation			Plastic
Detection Gas			Carbon Monoxide
Concentration			10-10000ppm CO
Circuit	Loop Voltage	Vc	≤10V DC
	Heater Voltage	V_H	5.0V±0.2V AC or DC（High） 1.5V±0.1V AC or DC（Low）
	Heater Time	T_L	60±1S（High）90±1S（Low）
	Load Resistance	R_L	Adjustable
Character	Heater Resistance	R_H	31Ω±3Ω（Room Tem.）
	Heater consumption	P_H	≤ 350mW
	Sensing Resistance	Rs	2KΩ-20KΩ (in 100ppm CO)
	Sensitivity	S	Rs(in air)/Rs(100ppm CO) ≥ 5
	Slope	α	≤ 0.6(R_{300ppm}/R_{100ppm} CO)
Condition	Tem. Humidity		20℃±2℃；65%±5%RH
	Standard test circuit		Vc:5.0V±0.1V； VH（High）: 5.0V±0.1V； VH（Low）: 1.5V±0.1V
	Preheat time		Over 48 hours

Acknowledgement

We present our gratitude towards our Head of the Department, Mr. Amey Naik, Department of Electronics and Telecommunication, K. J. Somaiya College of Engineering, for providing all the facilities required for completion this project.

We would also like to thank our guide, Mrs. Savita Raut, Department of Electronics and Telecommunication, K. J. Somaiya College of Engineering, for all the support, guidance and encouragement provided by her to make this project possible.

At last special to thank the teammates and teaching and non-teaching staff for helping and working throughout the project.