Internet of Things (IoT) in Unmanned Ground Vehicle

A Project Report

Submitted by

VRUSHIT PATEL
DHRUV PATHAK
SHIVANSH SHARMA
SHREY THAPAR

Under the Guidance

PROFESSOR SANJAY DESHMUKH

In partial fulfilment for the award of the degree of

BACHELORS OF TECHNOLOGY
COMPUTER ENGINEERING

At

MUKESH PATEL SCHOOL OF TECHNOLOGY MANAGEMENT AND ENGINEERING

MARCH, 2022

DECLARATION

I, **Vrushit Patel, Dhruv Pathak, Shivansh Sharma, Shrey Thapar,** Roll No. **E006, E008, E027, E049** B. Tech (Computer Engineering), VIII semester understand that plagiarism is defined as anyone or combination of the following:

1. Un-credited verbatim copying of individual sentences, paragraphs or illustration (such as graphs, diagrams, etc.) from any source, published or unpublished, including the internet.

2. Un-credited improper paraphrasing of pages paragraphs (changing a few words phrases, or rearranging the original sentence order)

3. Credited verbatim copying of a major portion of a paper (or thesis chapter) without clear delineation of who did write what. (Source: IEEE, The institute, Dec. 2004)

4. I have made sure that all the ideas, expressions, graphs, diagrams, etc., that are not a result of my work, are properly credited. Long phrases or sentences that had to be used verbatim from published literature have been clearly identified using quotation marks.

5. I affirm that no portion of my work can be considered as plagiarism and I take full responsibility if such a complaint occurs. I understand fully well that the guide of the seminar/ project report may not be in a position to check for the possibility of such incidents of plagiarism in this body of work.

Signature of the Student: 

Name: Vrushit Patel, Dhruv Pathak, Shivansh Sharma, Shrey Thapar

Roll No.: E006, E008, E027, E049

Place: NMIMS, Mumbai

Date: 23/03/2022

CERTIFICATE

This is to certify that the project entitled “IoT in Unmanned Vehicle” is the bonafide work carried out by Vrushit Patel, Dhruv Pathak, Shivansh Sharma and Shrey Thapar of B.Tech (Computer Engineering), MPSTME (NMIMS), Mumbai, during the VIII semester of the academic year 2021-2022, in partial fulfillment of the requirements for the award of the Degree of Bachelors of Engineering, Integrated as per the norms prescribed by NMIMS. The project work has been assessed and found to be satisfactory.

Professor Sanjay Deshmukh

Internal Mentor

_____ _____

Examiner 1 Examiner 2

_____

Director

Table of Content

List of Figures

List of Tables

Abbreviations

ABBREVIATION	DESCRIPTION
UGV	Unmanned Ground Vehicle
TCP/IP	Transmission Control Protocol/Internet Protocol
Wi-Fi	Wireless Fidelity
IP	Internet Protocol
AWS	Amazon Web Services
IoT	Internet of Things
SDK	Software Development
OS	Operating System
MF, MM, FF	M: Male and F: Female
HDMI	High Definition Multimedia Interface
IDE	Integrated Development Environment
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
CLI	Command Line Interface
SSH	Secure Shell
HTTP	Hypertext Transfer Protocol
PHP	Hypertext Preprocessor
NDN	Named Data Networking
SQL	Structured Query Language
OMSS	Optimal Mobile Sync Selection
CET	Connection Execution Time
DSI	Display Serial Interface
CSI	Camera Serial Interface
GPIO	General Purpose Input Output
BCM	Broadcom
PWM	Pulse Width Modulation
IPSC	Circuit Serial Programming
AC	Alternate Current
DC	Direct Current
I/O	Input Output
GUI	Graphical User Interface
SUDO	Super User DO
RDP	Remote Desktop Protocol
MEMS	Micro Electro Mechanical System
PCB	Printed Circuit Board
FPS	Frames Per Second
RTSP	Real Time Streaming Protocol
URL	Uniform Resource Locator
GND	Ground

Chapter 1: Introduction

1.1 Project Overview

The Internet of Things (IoT) focuses on the physical devices that are equipped with sensors, processing power, software, and other technologies, and that communicate with other devices and systems over the Internet or other communication networks.

An unmanned ground vehicle is one which functions while in touch with the ground but without the presence of a human on board. Unmanned ground vehicles (UGV) can be employed in a variety of situations when having a human operator present is inconvenient, dangerous, or impossible.

The project focuses on the following objectives:

- Remotely control UGV over the internet
- Live stream from the UGV
- Attain serialization between Raspberry Pi and Ardiuno
- Store and analyse sensor data
- Semi automate the UGV
- Create a web application to control the UGV

1.2 Application Specification

1.2.1 Functional Requirements

1. **Motion Control:**

An Application is designed and linked to Raspberry Pi which uses TCP/IP to send and receive communication blocks that allow the devices to talk to each other over Wi-Fi. We establish the connection between different devices and a Raspberry Pi by providing the Internet Protocol (IP) address of Pi to the application. The Pi communicates through wires, the commands to the Arduino Uno, that are connected to the motors moving the wheels

2. **Cloud Data Storage:**

Data that is collected from different sensors such as gyroscope, ultrasonic sensor, air quality sensors etc. needs to be stored for further processing and training of algorithms. Firebase IoT core is used to store and analyse this data. Raspberry Pi is used to communicate with the Firebase cloud storage using Python programming language.

3. **Different Control Modes:**

The UGV has two different modes. The manual mode is executed using a controller interface (web application) integrated with a live stream. Manual mode is activated

when the vehicle is connected to a reliable network. In case of network failure, the vehicle stops to avoid damages. The semi-automatic mode is executed by clicking a button on the web application. The UGV, with object avoidance, continues to move forward without any human assistance.

4. **Live Stream**

The camera Module integrated with the Raspberry Pi feeds videos (or images as frames)to provide a live stream on the web application.

5. **Serialization**

Serial communication is a type of communication which sends and receives data via a one or two transmission line system, with data being transmitted and received one bit at a time. Serialization of the Pi and Uno is an important feature that allows you to get the most out of both boards.

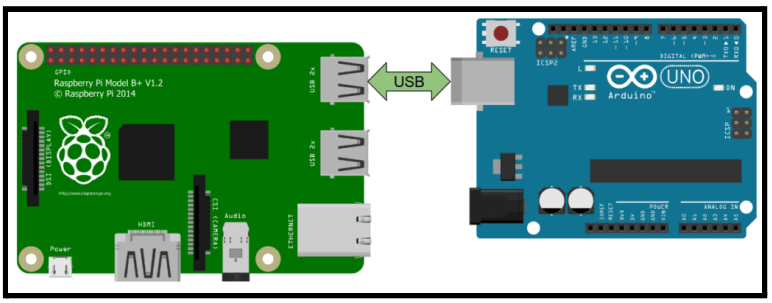


Figure 1.2.1 Serialization Demonstration

1.2.2 Non-Functional Requirements

1. **Performance and Scalability**

Under a given workload, performance refers to how quickly a software system or a specific portion of it responds to certain user inputs. In most cases, this metric explains how much a user must wait before the target operation happens (the results, motion sensor is activated, UGV movement etc.) given the sensitivity of the components (camera, gyroscope, hobby motors etc.) used. Scalability assesses the highest workloads under which the system will still meet the performance requirements. Raspberry Pi has a strong processing performance which ensures good results on all fronts. Moreover, using cloud services to store data enables us to scale the system. Moreover, Arduino Uno is highly compatible with motors improving the performance.

2. **Portability and Compatibility**

The term "portability" refers to how a system or its components may be launched in a different environment. Hardware, software, and other platform specifications are frequently included. Compatibility is an extra feature of portability. Compatibility refers to the ability of one system to coexist with another in the same environment. The vehicle's modest and compact design allows it to be operated in inaccessible areas on the same network and can be easily controlled over a simple web application.

3. **Security**

This non-functional criterion ensures that any data stored within the system or its components is safe from virus assaults and unauthorised access. Basic security is achieved by transferring data using common protocols such as TCP/IP and data is securely stored on the Google Firebase platform.

4. **Reliability**

This quality feature defines how probable it is that the system or its component will operate without failure for a certain amount of time under pre-set conditions. The Raspberry Pi is linked to the Firebase IoT core for data storage and processing, which ensures a stable connection. The Pi and Uno wired connections are 100% reliable.

5. **Usability**

Usability is yet another classical non-functional requirement that addresses a simple question: How hard is it to use the product?
The UGV has a simple design and can be deployed for a variety of applications like air quality monitoring, road diagnosis etc. The simple, clean web application makes it easy to control the rover and the graphs help analyse the sensor data through visualisations.

1.2.3 Supported Operating System

1. Raspberry Pi OS
2. Raspberry Pi OS Lite
3. RISC OS Pi etc.

1.2.4 Hardware Requirements

Raspberry Pi, Arduino Uno, Camera, Gyroscope, Breadboard, Jumper Wires (MF, MM and FF), Multi-meter, Display Screen, HDMI Cord, Micro USB, Keyboard, Mouse, DC Motor, GPS, Sensor, Ultrasonic Sensor, Servo Motor, Density, Humidity and Temperature sensor, Air Quality sensor etc.

1.2.5 Software Requirements

1. Arduino IDE
2. Python
3. Google Firebase
4. HTML
5. CSS
6. Javascript
7. C++

1.2.6 Compatibility Requirements

PC: Windows® operating system, versions 10 Build 10240 or higher
MAC®: Mac OS X® 10.5 or higher
Firebase Version 8.8.0
Python Version: 3.9
Pip Version: 21.1.2
C++: 14

1.3 Project Methodology

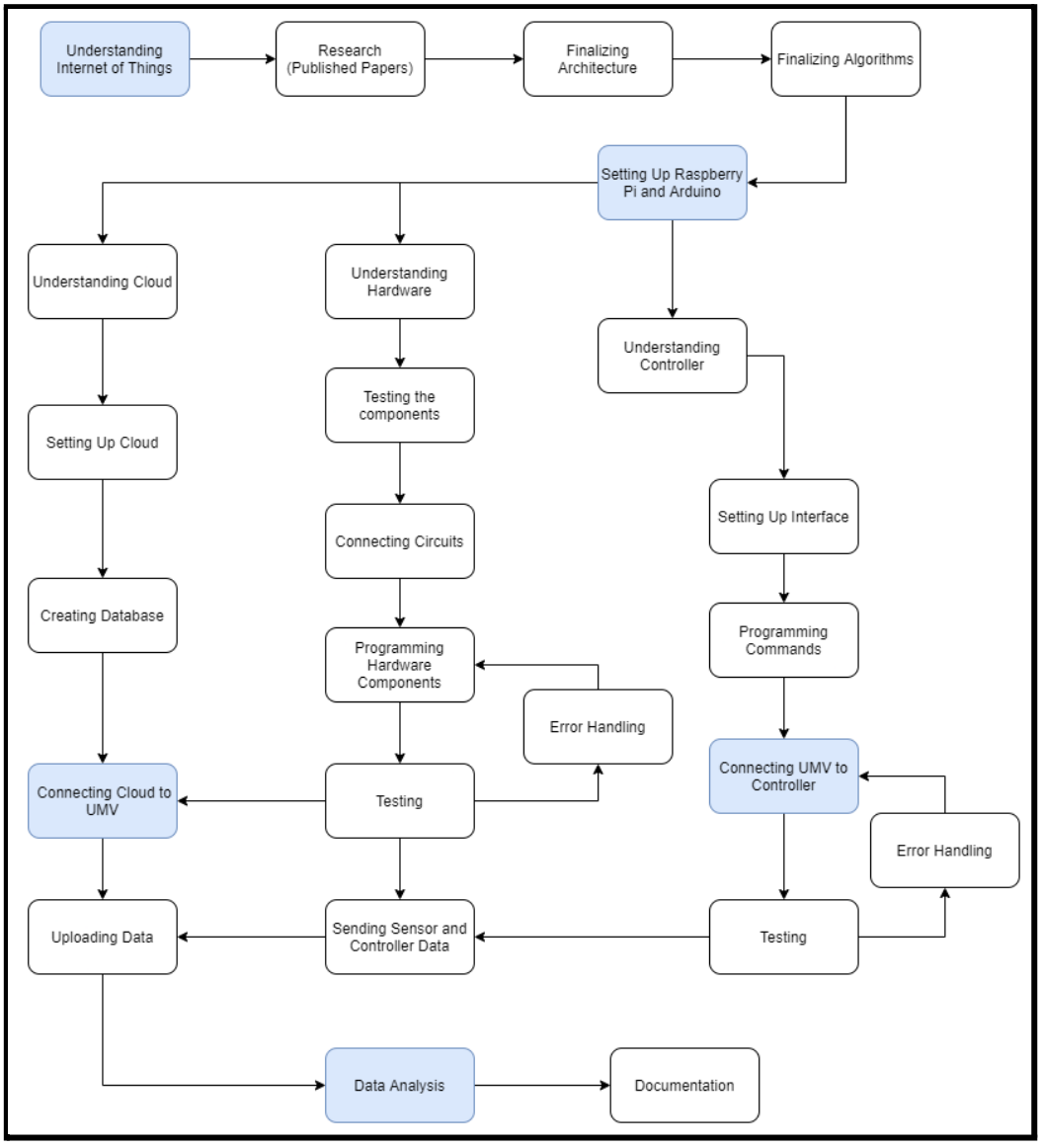


Figure 1.3 Project Methodology

Chapter 2: REVIEW OF LITERATURE

2.1 Literature Survey

After going through various research papers and building a strong foundation on the topic such as IoT, UGV, Pi, Uno etc, various explanations on the set-up, use case,design of unmanned vehicles were analysed.
There were various kinds of vehicles that were discussed at length in the research papers:

- Aerial Vehicles : Ability to stay ungrounded in the air.
- Autonomous Vehicles: Ability to move around without any external help.
- Semi-Autonomous Vehicles: Ability to perform only some operations without requiring any external help.
- Manual Vehicles: Require external assistance for performing any kind of functionality.
- Dummy Robots: Follow a given path without doing anything else.

An autonomous vehicle requires the successful implementation of various algorithms. Lane Detection Algorithm using the model based technique, real-time Object Detection Algorithm (example: You Only Look Once Algorithm) a using ultrasonic sensors and an Object Avoidance Algorithm (example: DistBug Algorithm) were learnt through the research papers.

The connectivity aspect of the vehicle was also discussed at length in all the research papers, ranging from internet connection to the server used for connecting the system. Common practice was a Wi-Fi 802.11n dongle used to connect to the RaspberryPi remotely and a Secure Shell (SSH) being used for establishing a cryptographic-network protocol. The Apache server was used for serving HTML files over HTTP and serving dynamic web pages using scripting languages such as PHP or Python.

The vehicle may be directed in a variety of ways, including command controlled mode, which is based on human decision making and offers navigation orders based on a live visual signal. Self-control mode is another mode that works on the vehicle's decision-making and navigation. Gesture Controlled, Raptor Controlled, Partial Self Controlled, and Complete Command Controlled Mode were among the other options.

Various autonomous vehicles' architectures were studied and analysed at length in each research paper. Each architecture comprises the integration of Raspberry Pi and Arduino with the system and represents the mode of functionality of the vehicle. The architecture also gave an in-depth view of the functionalities of each vehicle which included the vehicle having live stream video, UGV tracking and direction command abilities.

The research gaps that were identified consist of the inefficiency in Raspberry Pi and Arduino Uno communications, rover automations and user-friendly interface. Some other gaps that were found were the inability of the vehicle to deal with multiple obstacles and that the efficiency of the algorithms would be further enhanced if algorithms could start learning by themselves, thereby avoiding unnecessary calculations of familiar regions.

2.2 Existing System

- **System Design 1**

Hardware:

Raspberry Pi and Arduino are the main hardware devices to implement the prototype. The Raspberry Pi streams the video to the server and also provides sensor reading to the server. The Arduino controls the motor driver circuit. Serial communication is used to connect the Raspberry Pi to the Arduino. It instructs the Arduino to drive the automobile in the appropriate manner.

Cloud and Connectivity:

In this architecture apache is used as a web server with the help of HTTP. Apache is a popular web server application that was installed on the Raspberry Pi to allow it to serve web pages. Apache can serve HTML files over HTTP, and with additional modules, it can serve dynamic web pages using scripting languages such as PHP or Python.

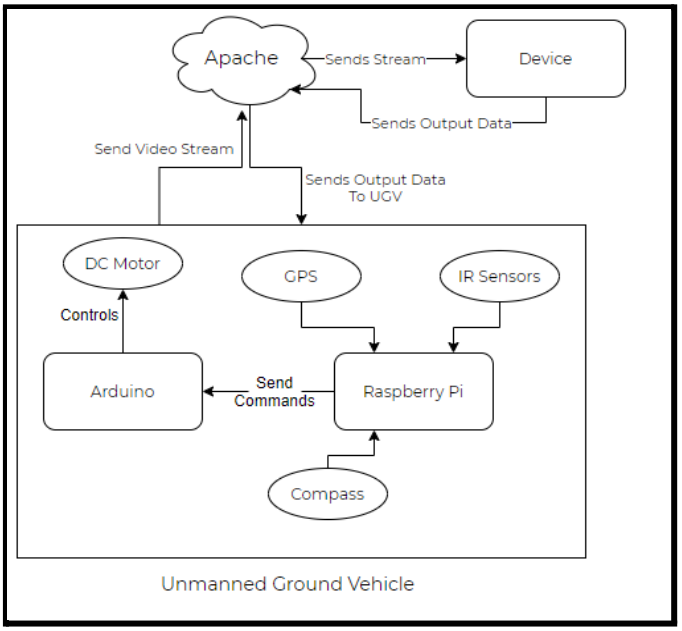


Figure 2.2.1 System Design 1 Architecture

Controlling:

Command Controlled Mode: They have taken into account human decision making in this mode and supply navigation orders based on the live video signal received from the camera installed on the UGV - Unmanned Ground Vehicle.

Self Controlled Mode: They evaluated self-decision making and self-navigation based on vision, artificial intelligence, path planning, and obstacle detection algorithms in this mode.

- ## System Design 2

Connectivity:

Named Data Networking (NDN) integrates pull-based communication to enable reliable and efficient message dissemination in the network, a data-oriented future Internet architecture (which retrieves data based on named content) offering several advantages over transmission control protocol (TCP)/IP networking including content delivery efficiency and data security. NDN architecture differs from TCP/IP at the middle or 'thin waist' layer. The NDN model, instead of pushing data to IP locations, does a demand-pull of data.

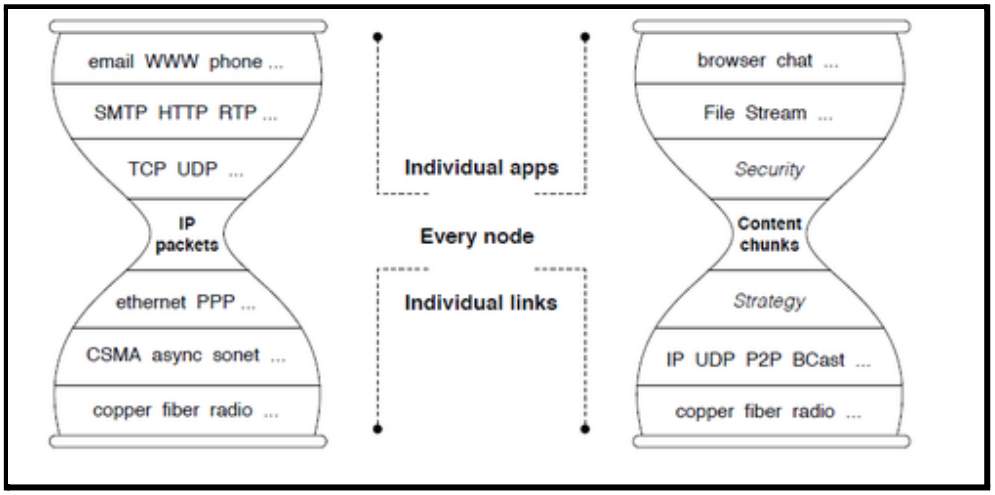


Figure 2.2.2 TCP/IP vs NDN

In NDN, security is incorporated into the data itself, rather than being determined by where or how the data is received. Each piece of data is signed along with its name, securing it. Data signatures are required – programmes cannot choose to "opt out" of security.

Cloud:

A Relational Database is made up of properly designed tables from which data may be handled and operated in a variety of ways without rearranging the complete database table set. SQL queries are used for interactive information retrieval as well as data collection for reporting and analysis. Therefore, Google Cloud Platform with MySQl Databases was proposed.

- **System Design 3**

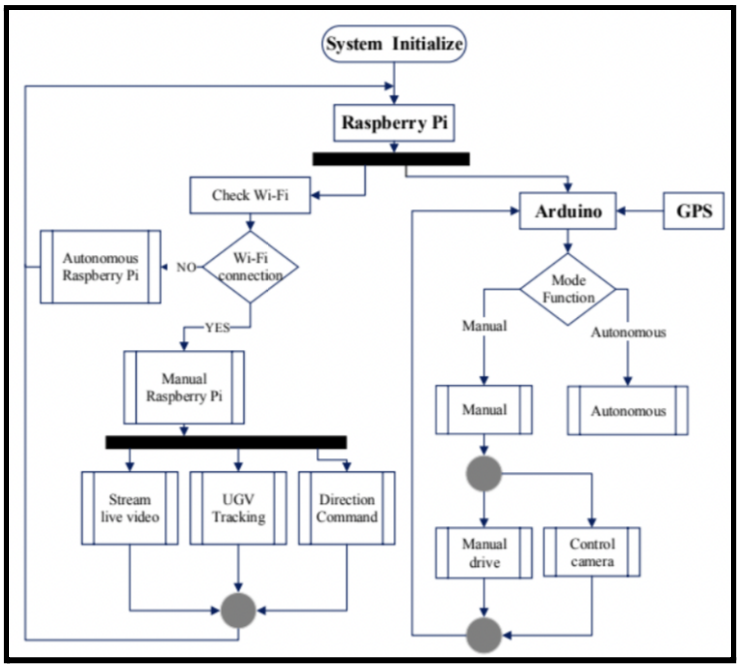


Figure 2.2.3 System Design 3 Architecture

Architecture:

In order to implement the system requirements, two microcontroller platforms were used: Raspberry Pi and Arduino.

The Raspberry main tasks are: Check for the internet connection; Receive data from Android application and send it to the Arduino to drive the UGV; Live streaming in the manual mode and recording the video in the autonomous mode.

The Arduino main tasks are: Receive the command data from the Raspberry Pi to make action as for control mobile robot direction or control camera direction in the manual mode; Check battery charge capacity; Save starting point coordinates; Mathematical calculation between two coordinate point; Obstacle avoidance in the autonomous mode.

	Model One	Model Two
Modes	Manual	Manual and Autonomous
Hardware Mechanical	Four wheels driven by four geared DC motors.	Four wheels driven by four geared DC motors
Hardware Electrical	Two Dual H-Bridge drivers.	Two Dual H-Bridge drivers.
Microcontrollers	Raspberry Pi 3 (Model B).	Raspberry Pi 3 (Model B). Arduino Mega. Arduino Mega
Camera	Raspberry Pi Camera. Movable camera mount with servo motor	Raspberry Pi Camera. Movable camera mount with servo motor

Sensors	Two ultrasonic sensors. Current sensor.	Three ultrasonic sensors. Current sensor. Compass, GPS module, Temperature and Humidity sensor Encoder
Battery	12v Batteries	12v Batteries
Software	Python	Python and C++
Android Application	Blynk user interface One	Blynk user interface One
Algorithm	Dist-Bug algorithm	Dist-Bug algorithm

Table 2.2.1 Rover Model Comparison

Connectivity:

The Optimal Mobile Sink Selection method (OMSS) attempts to increase network stability and minimise topology reconfiguration frequency. The OMSS is built on the idea that a sensor node will be able to estimate the duration of a connection with a mobile sink. The latter may determine the Connection Expiration Time using the UGV mobility data, such as speed and direction, link dependability, and the relative distance of the mobile sink from the sensor node (CET). CET is the amount of time a sensor node can stay reliably connected to a mobile sink before losing connectivity due to a communication failure (disconnection). The mobile sink with the lowest CET provides a more reliable network over time.

- ## System Design 4

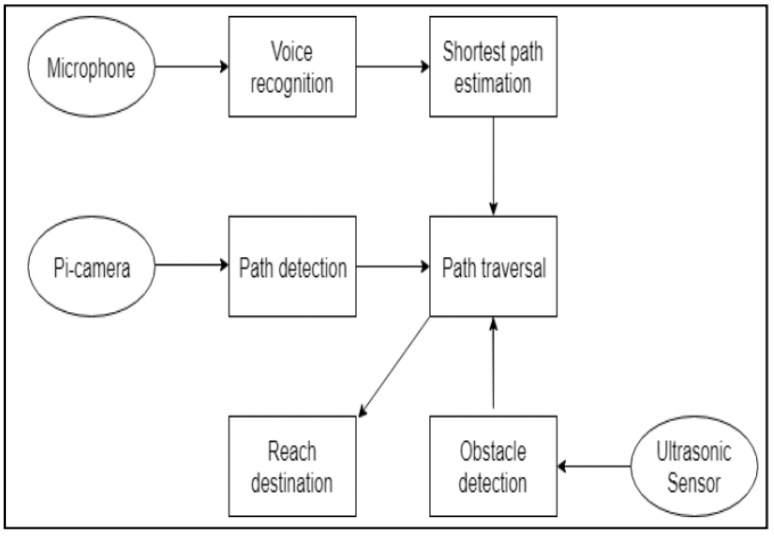


Figure 2.2.4 System Design 4 Architecture

Voice Recognition Module:

In our system, voice instructions via the microphone are used to give destination input. This input must be transformed from speech to text. The signal is then transformed to digital data. Several voice recognition services are used to convert this audio data to text.

Lane Detection Module:

The region of interest in the original image is cut out and used for further processing. The parallel lanes on the road between which the car is driving are then detected. The Pi Cam is then used to capture the road.

Edge Detection:

Canny Edge Detection Algorithm is used for detecting the areas where there is a mismatch between a pixel and its neighbours.

Obstacle Detection:

When the automobile comes to a halt, the pi-cam records a picture. This image is then sent to a pre-trained model, which recognises the objects in the image, especially those in front of the automobile. A Neural Network has previously been used to train this model using the ImageNet-1000 dataset. It has a 99.83 percent accuracy rate.

Web-based controls and video streaming:

For further functionality, the web app includes a live video stream created directly from the Pi camera, as well as basic controls like voice input and manual start/stop buttons.

Chapter 3: ANALYSIS & DESIGN

3.1 Understanding Raspberry Pi

The Raspberry Pi is a line of compact single-board computers developed by the Raspberry Pi Foundation in collaboration with Broadcom in the United Kingdom.

Raspberry Pi is a small computing device that aims to put the power of computing and digital making into the hands of people all over the world. Because of its low cost, versatility, and open design, it is widely utilised in a variety of applications, including weather monitoring, security cameras, and so on.

It's a low-cost, credit-card-sized computer with a normal keyboard and mouse that plugs into a computer monitor or TV. It's a capable small device that allows individuals of all ages to learn about computers and programming languages like Scratch and Python.

MODEL USED: RASPBERRY PI 3 B+

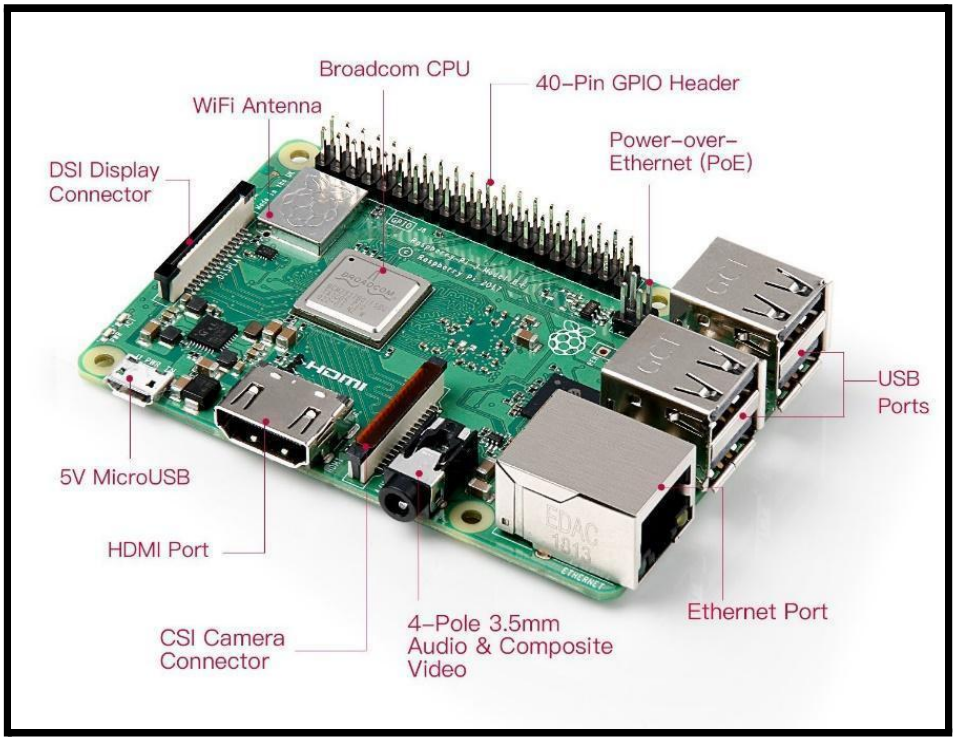


Figure 3.1.1 Raspberry Pi Components (Frontside)

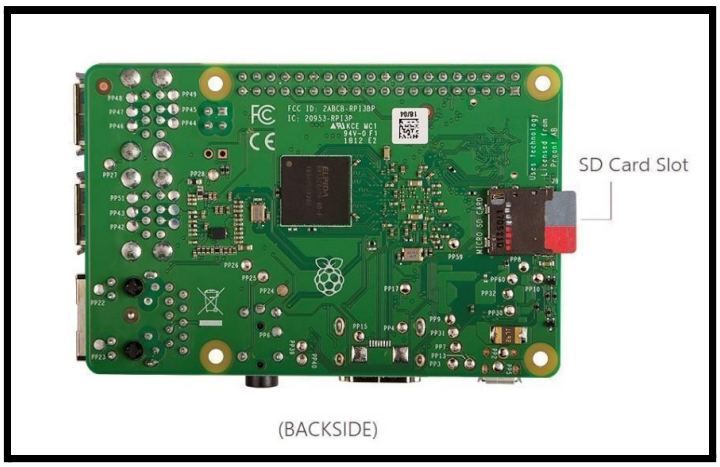


Figure 3.1.2 Raspberry Pi Components (Backside)

Components

1. **DSI Connector:** The Raspberry Pi can connect to a serial display similar to those seen in tablets via the DSI display connector. These touch-screen display modules come in a variety of sizes, including 7 inches.
2. **CSI Camera Connector:** The CSI camera port is a connection that connects a Raspberry Pi camera module to the Raspberry Pi. Generic web cams will not work because they typically only have a USB port.
3. **MicroSD Card Slot:** Power may be supplied to the Raspberry Pi through a micro USB cable connected to the micro USB connection (preferred) or by simply feeding 5V into the 5V GPIO port.
4. **Ports (HDMI / USB / Network):** These ports link the Pi to an HDMI display, USB peripherals like mice and keyboards, and an ethernet connection for internet access. However, because the Raspberry Pi 3 has built-in Wi-Fi, it is possible to connect wirelessly.
5. **GPIO Pins:** The Raspberry Pi's most significant feature are the 40 General Purpose Input Output Pins that is GPIO pins. These pins can be used to read electrical signals from circuits and give electrical signals for controlling circuits in programs. The Raspberry Pi's GPIO pins link to electronic circuits and allow it to control and monitor the outside world. The Raspberry Pi can control LEDs, switch them on and off, and operate motors, among other things. It can also tell if a switch has been pressed, the temperature, and the amount of light. This is referred to as physical computing. The GPIO Pins have 2 standard methods of numbering. First method is to number the pins from 1 to 40 starting from the top-left pin. This numbering is called the GPIO Board. The second method is called GPIO BCM (Broadcom SOC Channel), signified by the Broadcom SOC. Both these number systems can be seen in figures given below.

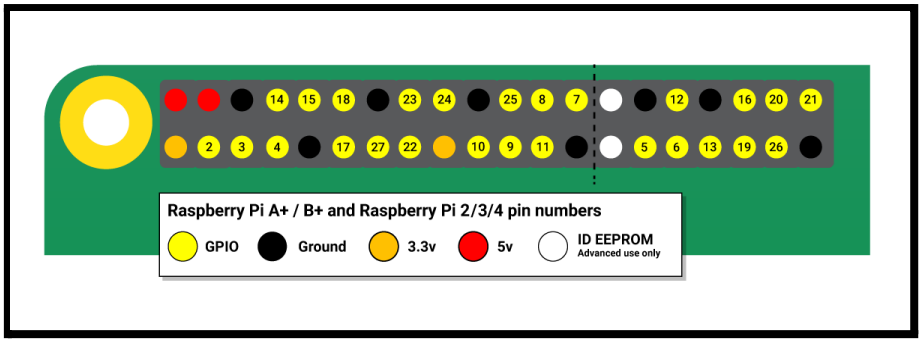


Figure 3.1.3 GPIO Pins

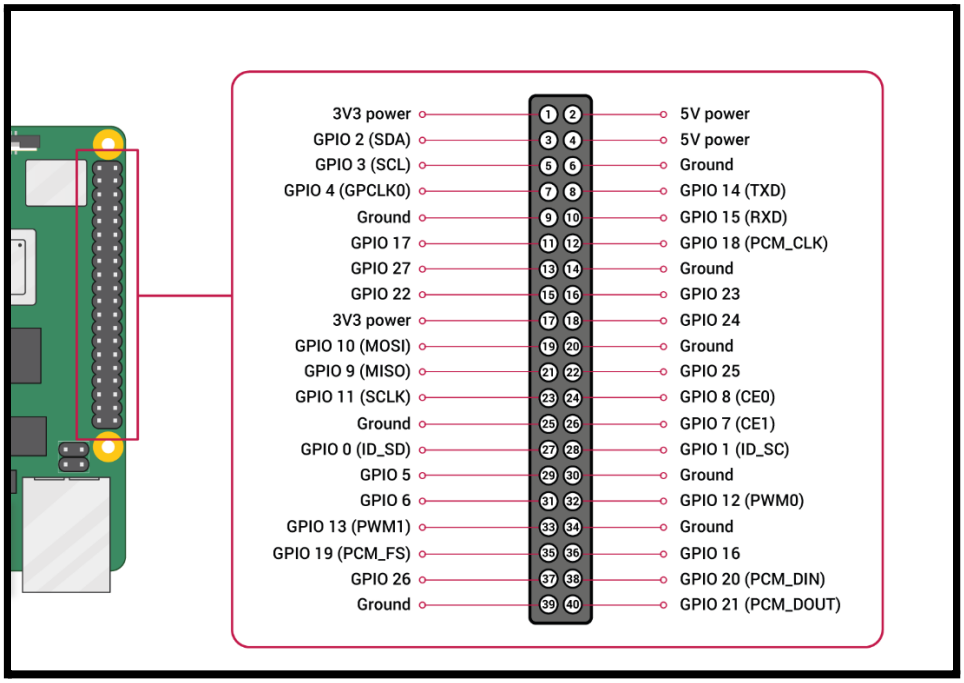


Figure 3.1.4. GPIO Pins Labelling

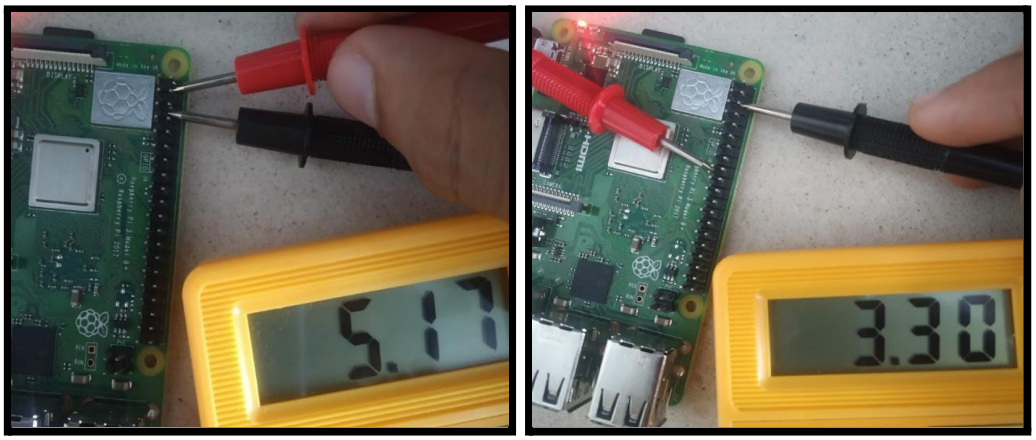


Figure 3.1.5 Voltage Pins

3.2 Understanding Arduino Uno

Arduino UNO is a low-cost, flexible, and easy-to-use open-source programmable microcontroller board that may be used in a wide range of electronic applications. This board can operate relays, LEDs, servos, and motors as an output and can be interfaced with other Arduino boards, Arduino shields, and Raspberry Pi boards.

The Arduino Uno is a microcontroller board that employs the ATmega328P microprocessor (datasheet). The board has a 16 MHz quartz crystal, 14 digital I/O pins (six of which may be used as PWM outputs), six analogue inputs, a USB connection, a power connector, an ICSP header, and a reset button. It includes everything you'll need to get started with the microcontroller, including a USB cable to connect it to a computer and an AC-to-DC adapter or battery to power it. You can tamper with your UNO without fear of doing something wrong; in the worst-case scenario, you can replace the chip for a few bucks and start over.

MODEL USED: Arduino UNO

Figure 3.2.1 Arduino Uno Components (Frontside)

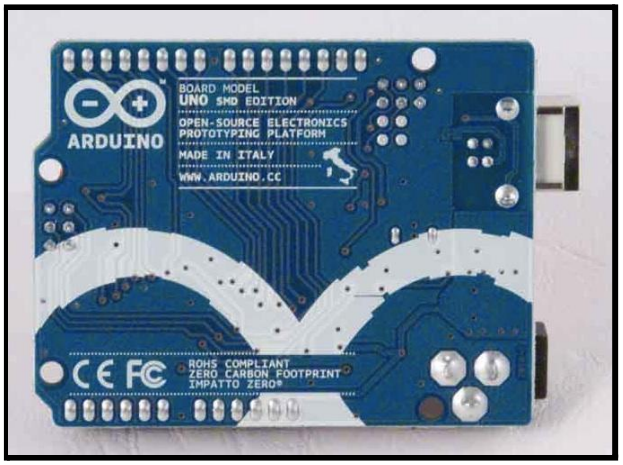


Figure 3.2.2 Arduino Uno Components (Backside)

Components

1. **ATmega328:** The programme is stored in the ATmega328, which is the brain of the board.

2. **Ground Pins:** There are a number of ground pins on the board.

3. **PWM:** There are six PWM pins on the PCB. PWM stands for Pulse Width Modulation, and it's a technique for controlling the servo motor, DC motor, and LED brightness.

4. **Digital I/O Pins:** The board has 14 digital (0-13) I/O pins that can be used to connect external electronic components.

5. **Analogue Pins:** The board is equipped with six analogue pins. These pins are capable of reading analogue sensors and converting them to digital signals.

6. Reset Button: This button will clear the code that has been loaded into the board. This button is important when the board hangs up; hitting it returns the board to its basic condition.

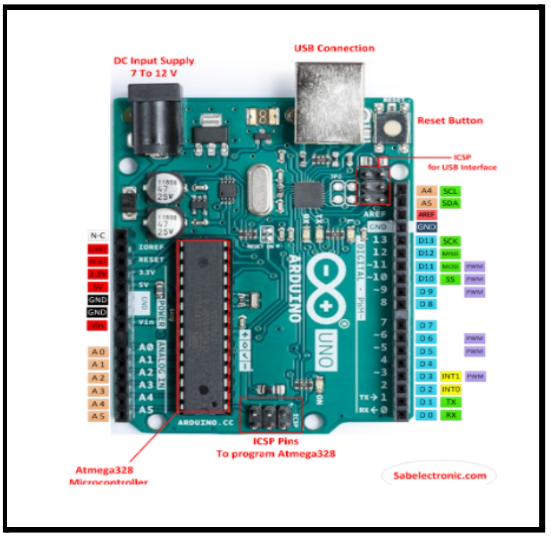
7. USB Interface: This interface is used to connect the board to the computer and upload the Arduino sketches (Arduino Program is called a Sketch)

8. DC Power Jack: This is used to power up the board using a power source.

9. Power LED: This is a power LED that illuminates when the board is linked to a power source.

Arduino UNO Pinout

The Arduino UNO is the most often used board in the electrical sector. For a better understanding, the following diagram depicts the Arduino UNO Pinout:



3.2.3 Arduino Uno Pinout Labelling

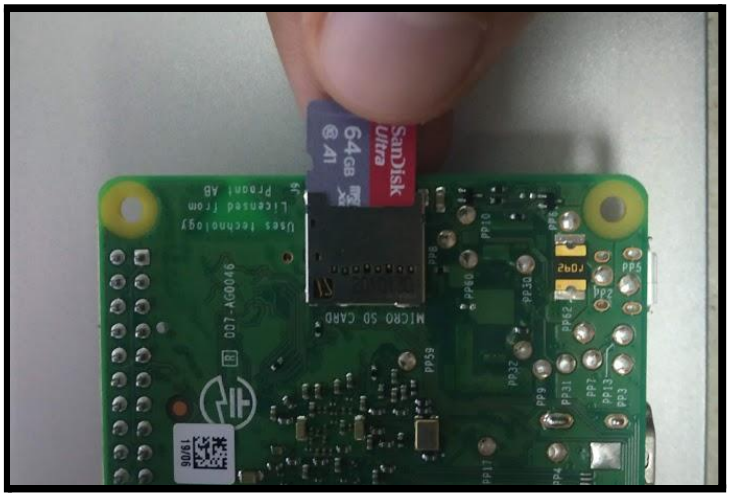
3.3 Setting up Raspberry Pi

Initial Setup:

To set up the Raspberry Pi for the first time ever, a few hardware requirements need to be met. The required hardware is:

1. TV (Display Monitor)
2. Mouse
3. Keyboard
4. SD Card (Minimum 4GB)
5. Power Cord (MicroUSB Type B)
6. HDMI Cable

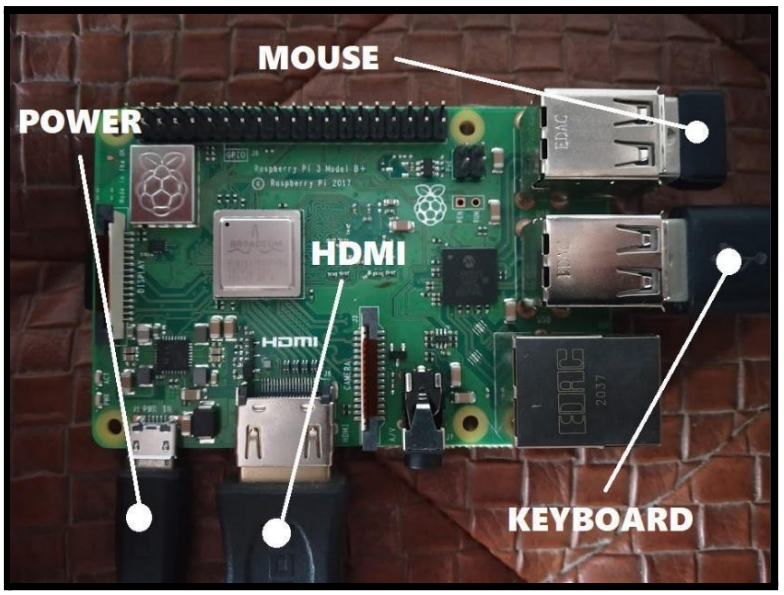
STEP 1: Use an SD Card and install an operating system on the SD Card. We will be working with the installation on Raspberry Pi OS. The Raspberry Pi OS can be found on the official RaspberryPi.org website. These OS files are written into the SD Card and then connected to the Raspberry Pi.



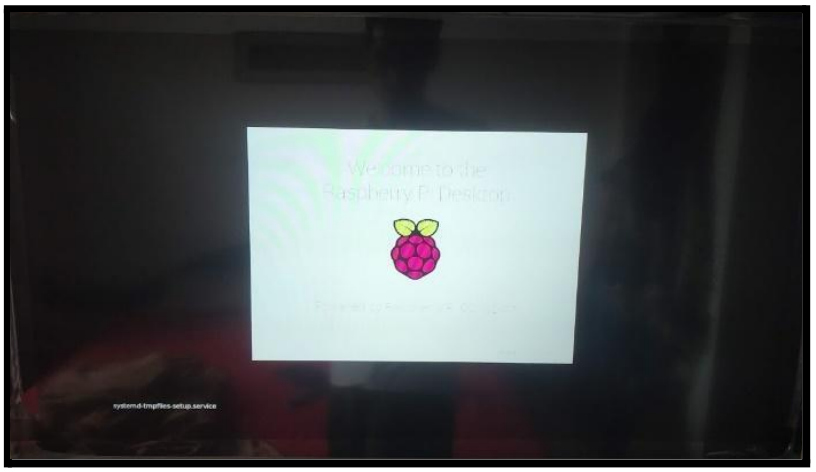
3.3.1 SD Card Slot

STEP 2: Connect the power cable to the Raspberry Pi and connect the HDMI cable to the TV. Also connect the mouse and keyboard to the USB ports.

STEP 3: Power is supplied to the Raspberry Pi and the OS boots up from the SD Card on the screen through the HDMI connection. Figure 3.3.3 shows the booting process on Raspberry Pi OS.



3.3.2 Initial Setup (Raspberry Pi)



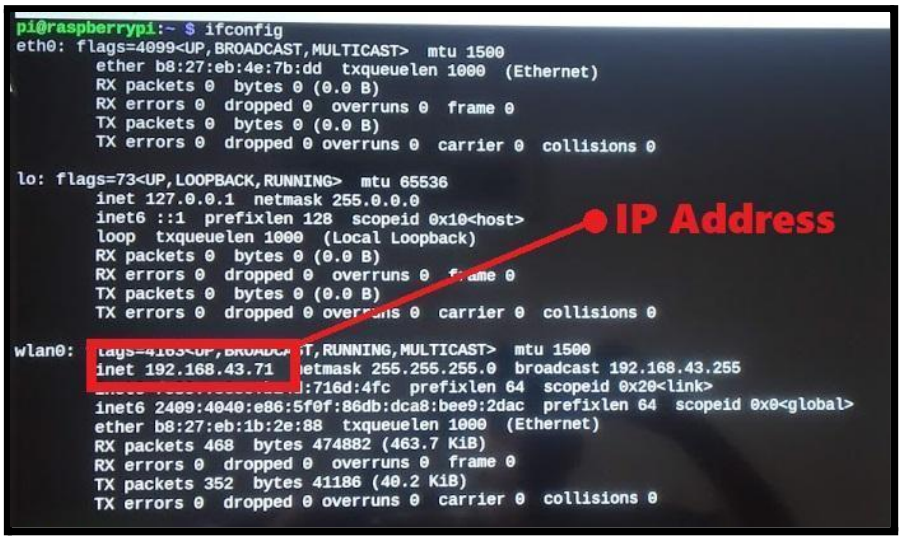
3.3.3 Raspian OS Boot-Up

3.3.4 Raspian OS Desktop

For Remotely Accessing Raspberry Pi on the Laptop

To remotely access the Raspberry Pi, one still requires to connect to a display monitor or TV once as shown above.

STEP 1: Finding the IP Address of the Raspberry Pi. To do this, the command *ifconfig* is executed on the Terminal (To Open Terminal => Ctrl + Alt + T).



3.3.5 Raspberry Pi IP Address

STEP 2: Execute the following command to open configuration settings :

sudo raspi-config.

STEP 3: Select Interface Options => SSH => Yes (Enable SSH) => OK.

STEP 4: Connect to the same WiFi as the Laptop for remote access. This can be done by clicking on the WiFi symbol in the top-right corner and connecting to the suitable network.

<u>For accessing the Terminal remotely :</u>

STEP 5:

MacOS - Open the terminal and execute *ssh username@ipaddress*

E.g.: ssh pi@192.168.43.71

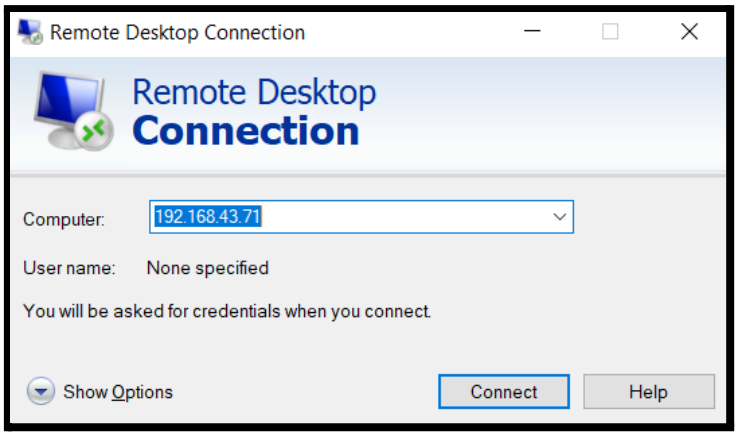
Windows: Install Putty Software, enter the IP Address to connect.

<u>For a GUI Desktop Remote Access :</u>

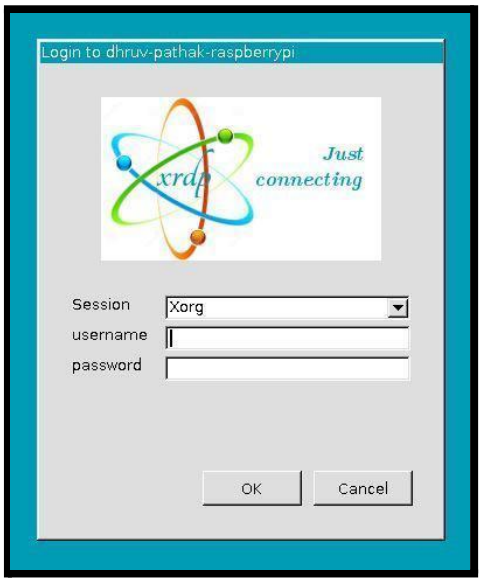
STEP 6: Installation of xrdp. To fulfil this requirement the following command is run on the terminal : *sudo apt-get install xrdp*

STEP 7: Open Remote Desktop Connection, an in-built application and enter the IP Address of your Raspberry Pi and connect.

STEP 8: Enter the username (default: pi) and password (default: raspberry) to successfully access the Raspberry Pi over WiFi.



3.3.6 Remote Desktop Software



3.3.7 Remote Desktop Login

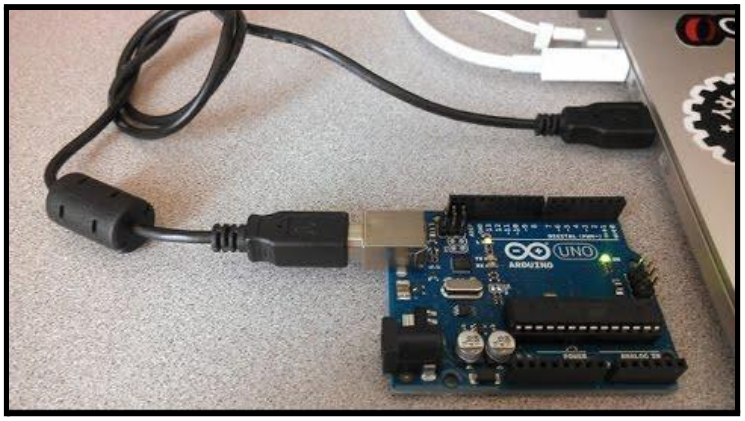
3.4 Setting up Arduino Uno

The Arduino UNO is simple to program, and even those with little or no technical skills may gain hands-on experience with it. The Arduino UNO board is programmed using the Arduino IDE software, which is an official software introduced by Arduino.cc.

If you wish to programme your Arduino Uno when you're not connected to the internet, you'll need to download and install the Arduino Desktop IDE. The Arduino Software (IDE), our

Integrated Development Environment that is shared by all of our boards, is used to programme the Uno. You must first install the Arduino Software (IDE) on your PC before proceeding.

Connect your Uno board to a computer via an A B USB cable, often known as a USB printer cable.



3.4.1 Arduino UNO Set-up

How to Program Arduino UNO:

The Arduino program is known as a sketch, and it must be loaded into the board. The sketch is nothing more than a collection of instructions that allows the board to perform certain operations based on your specifications.

Each Arduino sketch has two major components:

void setup() – this sets up the stuff that must be done just once and will not be repeated in the running application.

void loop() – this section contains instructions that are repeated until the board is shut off.

3.4.2 Arduino Uno IDE

Chapter 4: METHODS IMPLEMENTED

4.1 Proposed Algorithm

The project uses algorithms from various domains like Object Detection, Object Avoidance and Networking and Connectivity.

1. **Obstacle Avoidance**
   The problem of accomplishing some control target while adhering to non-intersection or non-collision position restrictions is known as object avoidance. It is distinguished from route planning in that the former is often implemented as a reactive control rule, whilst the latter requires the pre-computation of an obstacle-free path along which a controller would then lead a robot. To attain semi-automation, Python code is used. Object avoidance is attained using 3 ultrasonic sensors.

2. **Networking and Connectivity**
   Networking and connectivity forms a very important aspect of the project as the unmanned vehicle would require functional internet connectivity and a stabilised network at all times.

   Optimal Mobile Sink Selection Algorithm (OMSS):
   1. It is always striving to improve network stability and, as a result, reduce the frequency of topology reconfiguration in a regulated manner.
   2. It can compute a value known as the Connection Expiration Time (CET). CET is the amount of time that a sensor node may be reliably connected.

4.2 Architecture Diagrams

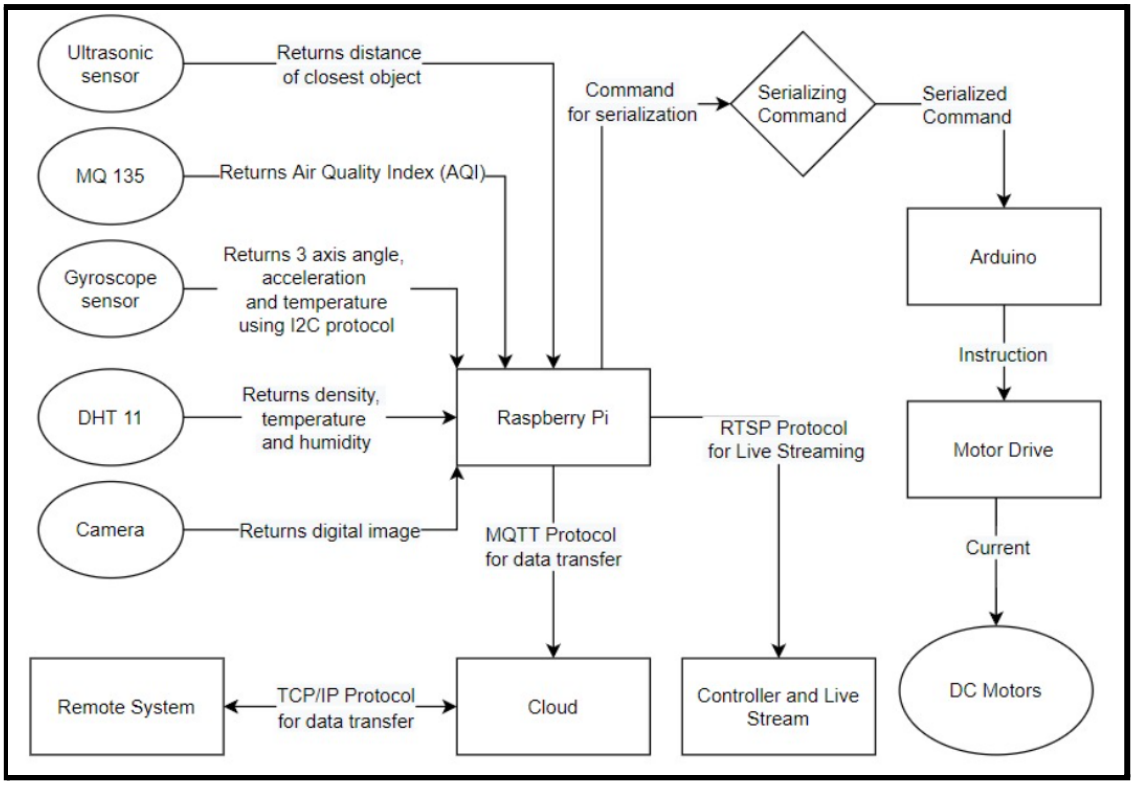


Figure 4.2 Architecture Diagram

4.3 Design Diagram

Fig. 4.3.1 tells us about the connections on the motor shield which is connected on the Arduino Uno. There are 4 motors which will help the Unmanned Ground Vehicle to move and an extra servo motor connection. This is also powered by a 5 to 9 volt battery with an on-off switch.

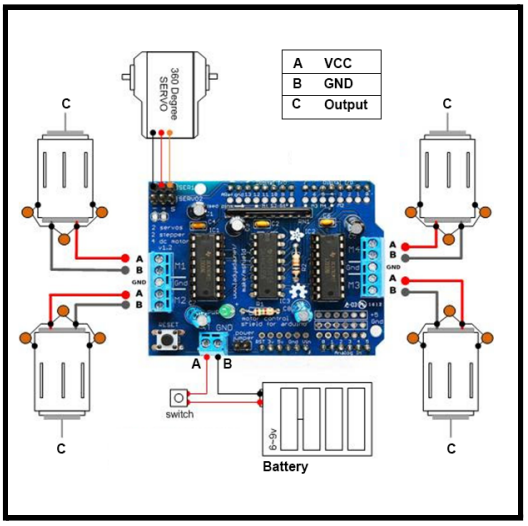


Figure 4.3.1 Arduino Design Diagram

The Fig. 4.3.2 tells us about the connections on the Raspberry Pi, there are 3 sensors connected on the board. A Gyroscope, ultrasonic sensor and a Pi Camera are connected on the board.

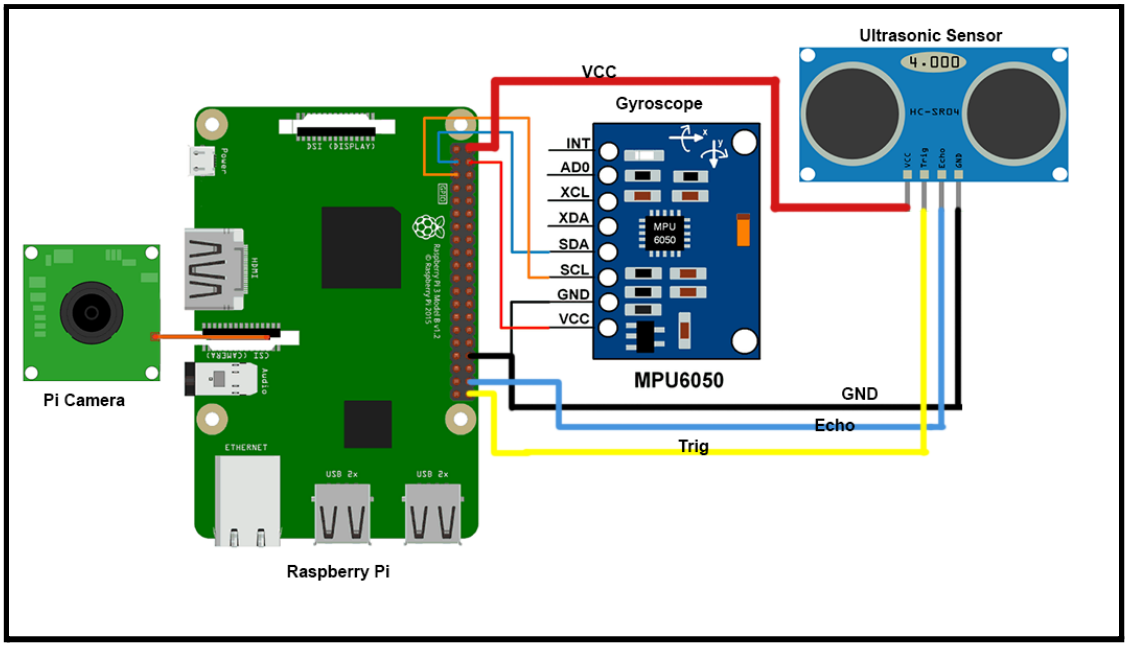


Figure 4.3.2 Rover Design Diagram

Chapter 5: DISCUSSION

5.1 Software Implementation

5.1.1 Web Application

The web application displays the following modules:

- **Data Module:** The data module displays the data collected from various sensors on the rover. Data from the sensor is gathered by the Raspberry Pi and sent to a cloud database. The cloud database sends the data back to the website. Sensors that display data include air quality sensors, ultrasonic sensors, and temperature and humidity sensors. The air quality sensor captures the current air quality around you, and the ultrasonic sensor displays the distance between the objects closest to the rover. Temperature and humidity sensors are used to detect local humidity and temperature.
- **Distance Module:** The display module helps users navigate the rover by displaying live feeds from the Pi camera attached to the rover. The live feed is captured by a Pi camera and sent to the Raspberry Pi. Raspberry Pi renders it into an HTML page. Live streams are displayed in this module with an i-frame tag that embeds a local HTML page in the Pi module responsible for the live feed. This live stream is only possible if the system is connected to a Pi module on your local network.
- **Utility Module:** The utility module has two buttons. One is for displaying charts and the other is for enabling or disabling the semi-automatic mode of the rover. The data collected by the sensors on the rover are visualized and displayed in chart format. The distance between the nearest objects to the rover is plotted as a function of time in the first plot. The second graph plots the data generated by the gyroscope to show the state of the road and shows the z-axis movement over time. The third graph shows the air quality of the area over time.
- **Semi-Automation:** One can click the semi-automation button to enter semi-automatic mode. In this mode, the data collected by the ultrasonic sensor in front of the rover can be used by the UGV to navigate in the immediate vicinity. The Raspberry Pi processes the data from the sensor and then guides the rover forward to avoid obstacles along the way.
- **Navigation Module:** The navigation module consists of four buttons. One is for driving forward, the second is for driving to the right, the third is for driving to the left, and the fourth is for driving backward. These buttons allow the user to navigate around the rover. On clicking the forward button, the Arduino commands all four DC motors to rotate in the forward direction. The reverse button works as well, with all four wheels rotating in the reverse direction. The button on the left rotates the motor on the right side of the rover forward when the motor on the left side is stationary. When the user clicks the right button, the motor on the left side of the rover rotates forward and the motor on the right side comes to rest.

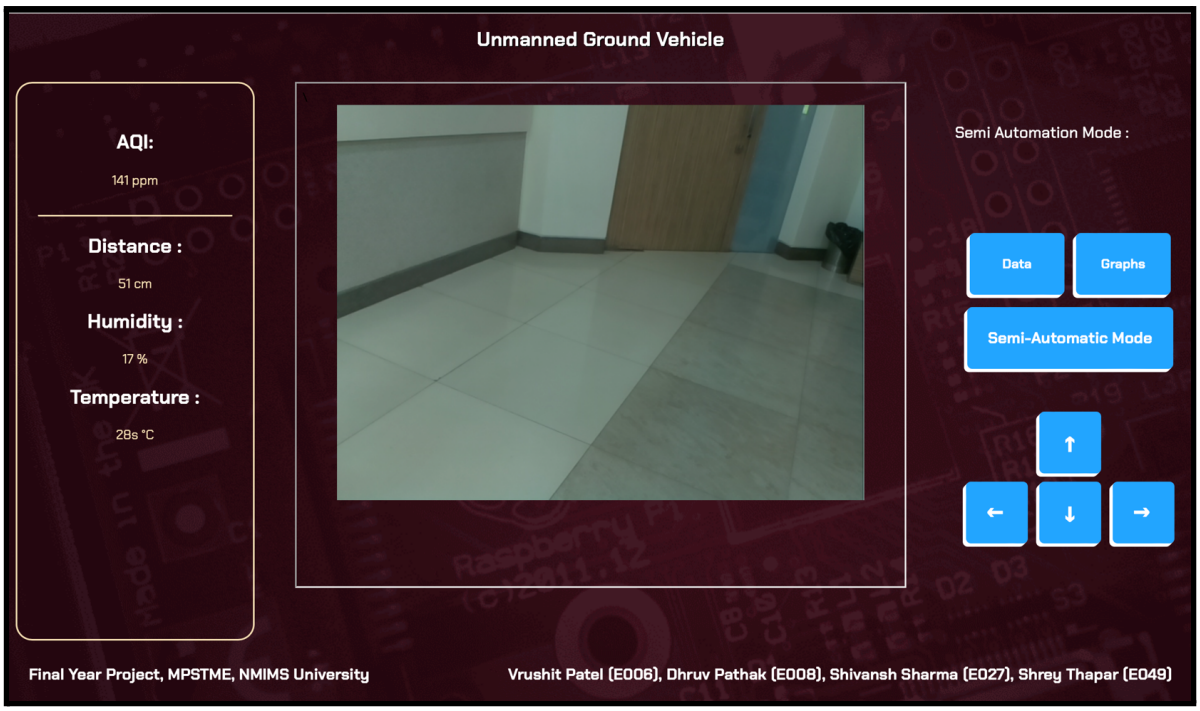


Figure 5.1.1.1 Web Application Snapshot

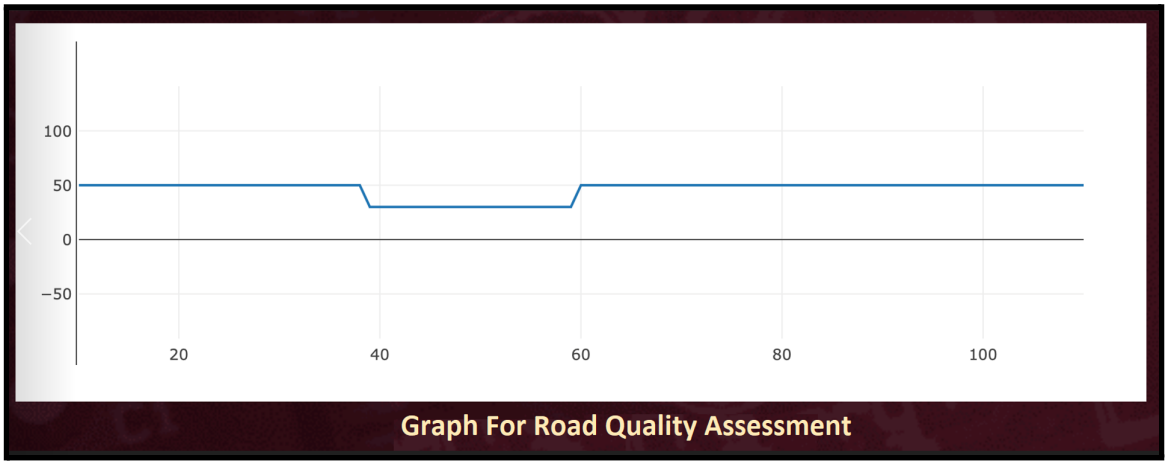


Figure 5.1.1.2 Road Quality Assessment

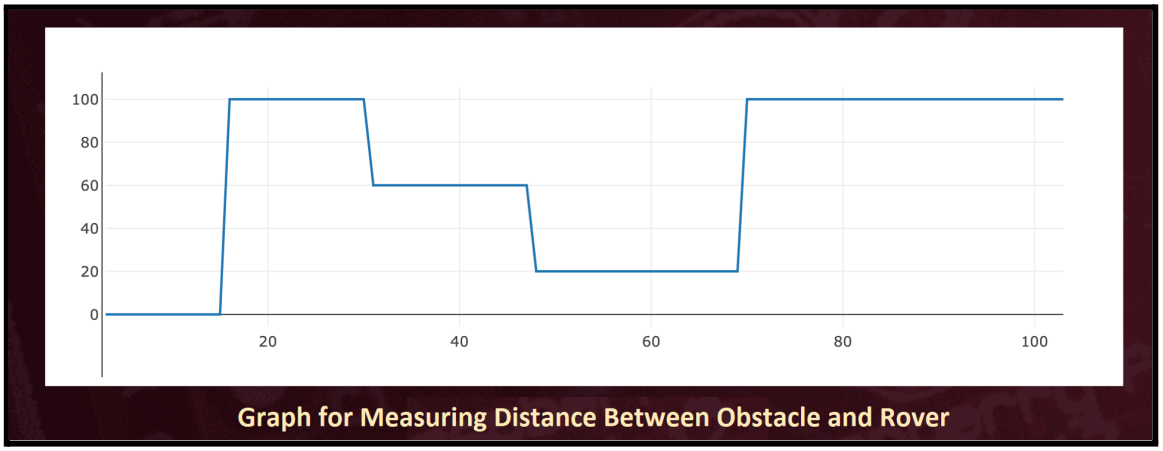


Figure 5.1.1.3 Distance between Obstacle and Rover

5.1.2 Serialization

Serialization on the Raspberry Pi.

Figure 5.1.2.1 Controller GUI

```
class RcApp(Tk):

    def __init__(self):
        Tk.__init__(self)

        self.left = Button(self, text='go left!')
        self.right = Button(self, text='go right!')
        self.forward = Button(self, text='go forward!')
        self.reverse = Button(self, text='reverse!')

        self.bind('<ButtonPress>', self.callback)
        self.bind('<ButtonRelease>', self.callback)

        self.left.grid(column=4,row=10, padx=10, pady=2)
        self.right.grid(column=10,row=10, padx=10, pady=2)
        self.forward.grid(column=7,row=8, padx=10, pady=2)
        self.reverse.grid(column=7,row=10, padx=10, pady=2)

        self.type = None
```

```
    def callback(self, event):
        print(event.type)
        self.type = event.type
        if self.type == '4':
            if event.widget == self.left:
                self.after(200, self.move_left)
            if event.widget == self.right:
                self.after(200, self.move_right)
            if event.widget == self.forward:
                self.after(200, self.move_forward)
            if event.widget == self.reverse:
                self.after(200, self.move_reverse)
        else:
            dir = "n"
            value = write_read(dir)
            print(value) # printing the value
```

```
    def move_right(self):
        if self.type == '4':
            dir = "3"
            value = write_read(dir)
            print(value)
            print('moving right!')

    def move_forward(self):
        if self.type == '4':
            dir = "1"
            value = write_read(dir)
            print(value)
            print('moving forward!')

    def move_reverse(self):
        if self.type == '4':
            dir = "2"
            value = write_read(dir)
            print(value)
            print('moving reverse!')
```

```
arduino = serial.Serial(port='/dev/ttyACM0', baudrate=115200, timeout=.1)
def write_read(x):
    arduino.write(bytes(x, 'utf-8'))
    time.sleep(0.05)
    data = arduino.readline()
    return data
```

5.1.2 Uploading Data to Firebase

Google Firebase is a Google-backed application development software that enables developers to store real-time data on the cloud. To send data from the Raspberry Pi to our Firebase cloud storage, the following code is executed.

```
while True:
 try:
        for i in range (0,5):
        # if GPIO.input(10) == GPIO.HIGH:
        print("pushed")
        now = datetime.now()
        dt = now.strftime("%d%m%Y%H:%M:%S")
        name = dt+".jpg"
        camera.capture(name)
        print(name+" saved")
        storage.child(name).put(name)
        print("Image sent")
        os.remove(name)
        print("File Removed")
        sleep(2)

 except:
        camera.close()
```

5.2 Hardware Implementation

1. Gyroscope

The gyroscope or MPU6050 is a Micro Electro-Mechanical Systems (MEMS) with a 3-axis accelerometer and 3-axis gyroscope. This allows us to measure a system's or object's acceleration, velocity, direction, displacement etc.

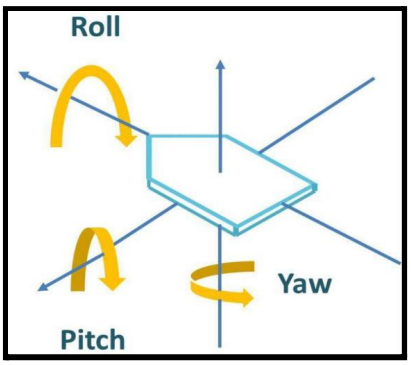


Figure 5.2.1.1 Gyroscope Working

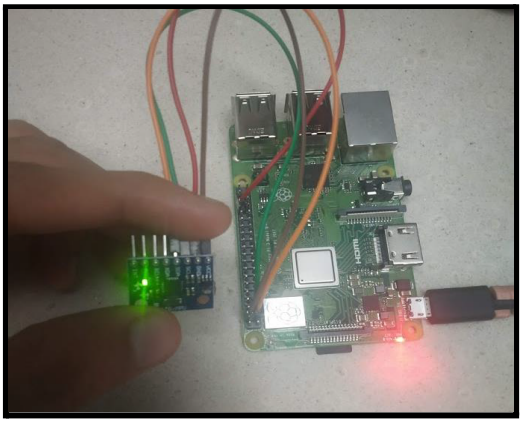


Figure 5.2.1.2 Gyroscope Connection

```
accel_data = mpu.get_accel_data()
print("Acc X : "+str(accel_data['x']))
print("Acc Y : "+str(accel_data['y']))
print("Acc Z : "+str(accel_data['z']))

gyro_data = mpu.get_gyro_data()
print("Gyro X : "+str(gyro_data['x']))
print("Gyro Y : "+str(gyro_data['y']))
print("Gyro Z : "+str(gyro_data['z']))
```

Output

```
Temp : 36.53

Acc X : 0.0
Acc Y : 0.0
Acc Z : 0.0

Gyro X : 0.0
Gyro Y : -5.030534351145038
Gyro Z : 6.297709923664122

--------------------------------
Temp : 35.353529411764704

Acc X : -6.433219860839843
Acc Y : 2.7246014892578123
Acc Z : 7.5153990112304685

Gyro X : 1.6641221374045803
Gyro Y : -3.8091603053435112
Gyro Z : 0.12213740458015267

--------------------------------
Temp : 35.49470588235294

Acc X : -6.399701037597656
Acc Y : 2.6336218261718747
Acc Z : 7.314286071777343

Gyro X : 1.549618320610687
Gyro Y : -2.5419847328244276
Gyro Z : -1.465648854961832
```

Figure 5.2.1.3 Gyroscope Output

2. Camera

The Raspberry Pi Camera Board is a custom-designed Raspberry Pi add-on module. In still capture mode, the sensor has a native resolution of 5 megapixels. It can take video at resolutions up to 1080p at 30 frames per second in video mode. A ribbon wire connects the camera board to the Raspberry Pi. The ribbon cable connections must be made correctly or the camera will not operate. The blue backing of the wire should be pointing away from the PCB on the camera PCB and towards the Ethernet connection on the Raspberry Pi hardware.

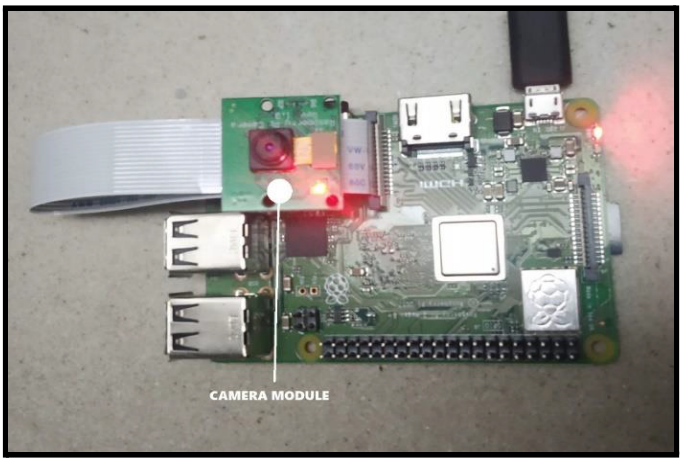


Figure 5.2.2.1 Camera Setup

2.1 Live Feed (VLC Media Player)

Firstly it is important to enable the camera in the RaspberryPi Configurations.

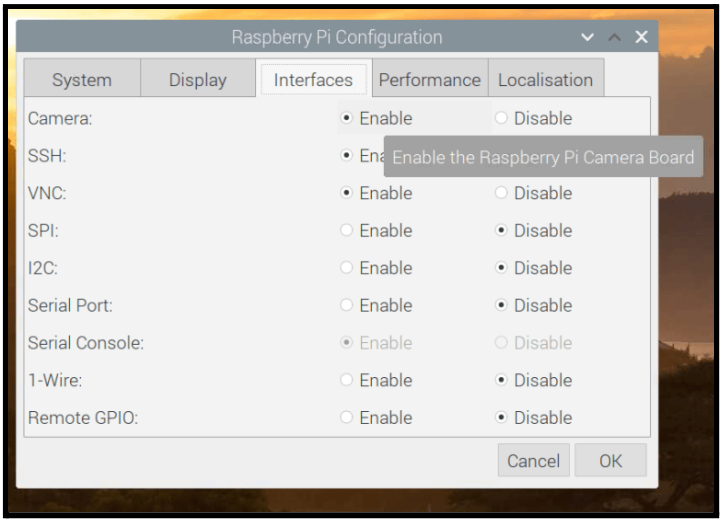


Figure 5.2.2.1.1 Enabling Camera Interface

To check the working of the camera, take a sample snapshot using the raspistill command as seen below.

Figure 5.2.2.1.2 Sample Image CLI Command

Figure 5.2.2.1.3 Sample Image CLI Output

First method of live feed implemented for the rover was through the VLC Media Player.

- It requires the use of the *raspivid* command to be run with the required attributes.

Figure 5.2.2.1.4 VLC Live Feed CLI Command

- Our output is set by **-o**.
- The duration of the video clip is specified by **-t**; a value of zero will result in an unlimited video clip.
- The video's width and height are specified by **-w** and **-h**.
- **fps** is the video stream's frames per second; a lower number should reduce dropouts.
- **cvlc** is a pipe that takes the output of the raspivid command, our video stream, and streams it using the h264 codec over our network using the real-time streaming protocol (rtsp). The port is set at 8080.

We use the network stream feature in VLC Media Player. Go to: Menu -> Open Network Stream -> Enter the network URL -> Play

URL Format:

A. rtsp:// (Protocol)
B. raspberrypi: (Hostname, default is this case)
C. 8080/ (Set port number is command run on terminal)

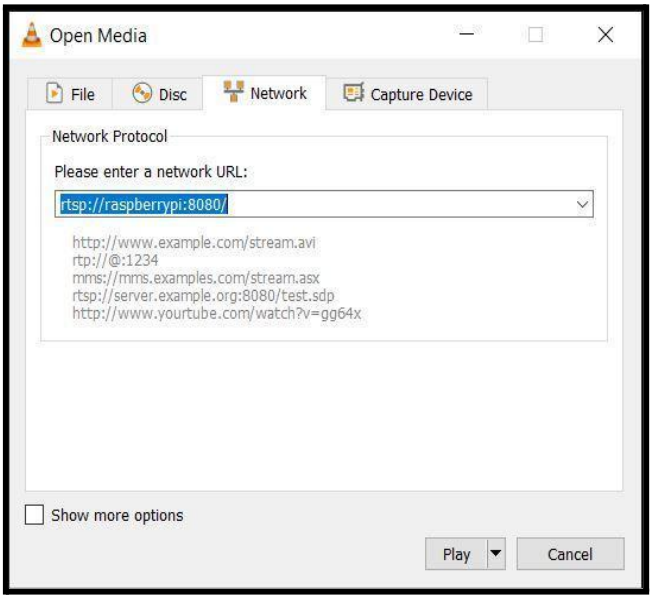


Figure 5.2.2.1.5 VLC Open Network Window

After a 3-4 sec blank screen, the live feed from the camera can be seen on the VLC Media Player. The executed output is shown below:

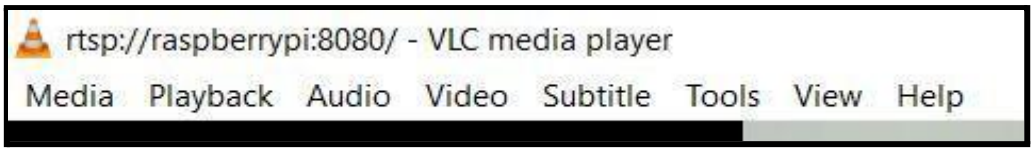


Figure 5.2.2.1.6 Network URL

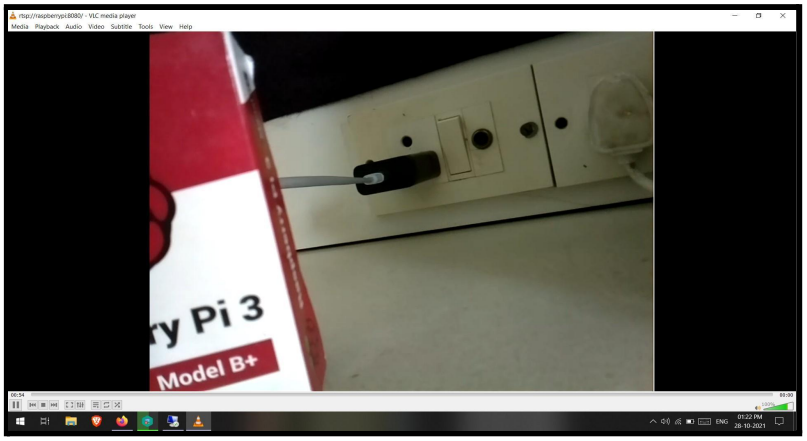


Figure 5.2.2.1.7 Live Feed Output

The terminal on the RaspberryPi shows the following information:

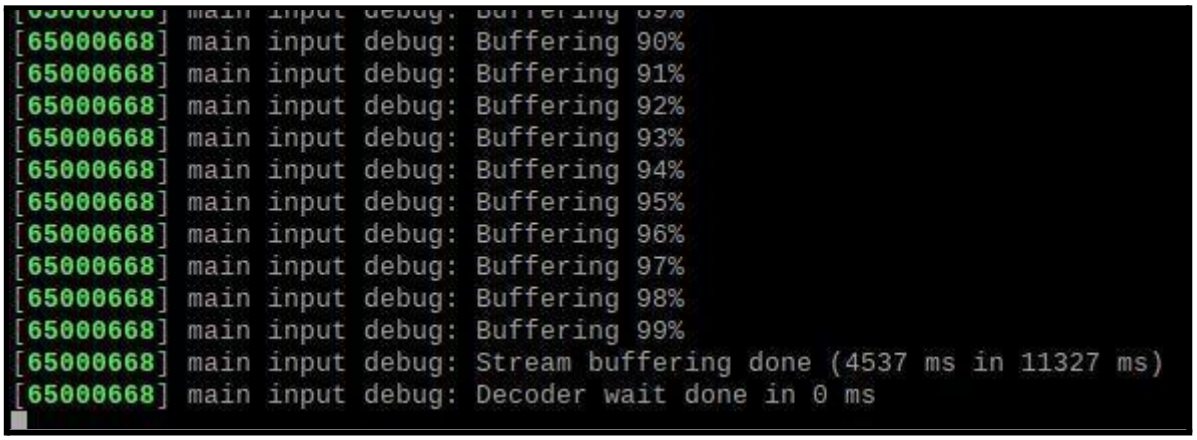


Figure 5.2.2.1.8 VLC Live Feed CLI Command Execution

Advantages:

- Easy to execute.
- Minimal Complexity and fast execution.

Disadvantages:

- 5-10 sec delay in the output stream.

- Increasing fps results in more dropout.

2.2 Live Feed on IP Address

After enabling the Camera interface on the Raspberry Pi and checking its working using *"raspistill"*, we execute the following code to live stream on port 8000.

```
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()
```

```
class StreamingServer(socketserver.ThreadingMixIn, server.HTTPServer):
  allow_reuse_address = True
  daemon_threads = True

with picamera.PiCamera(resolution='640x480', framerate=24) as camera:
  output = StreamingOutput()
  #Uncomment the next line to change your Pi's Camera rotation (in degrees)
  #camera.rotation = 90
  camera.start_recording(output, format='mjpeg')
  try:
    address = ('', 8000)
    server = StreamingServer(address, StreamingHandler)
    server.serve_forever()
  finally:
    camera.stop_recording()
```

The "StreamingOutput" Python class offers two methods for initialising and outputting the buffered frames. Aside from the aforesaid class, we define various others like "StreamingHandler" and "StreamingServer." The streaming handler keeps the live stream going by sending HTTP requests and picture data. Streaming Server keeps the server function running in the network and threading active.

Advantages:

- Easy to execute and connect.
- Less than 1 sec of delay.
- High FPS rate.
- No Dropouts
- Can be viewed on multiple devices on the same network.
- Minimal Complexity and fast execution.

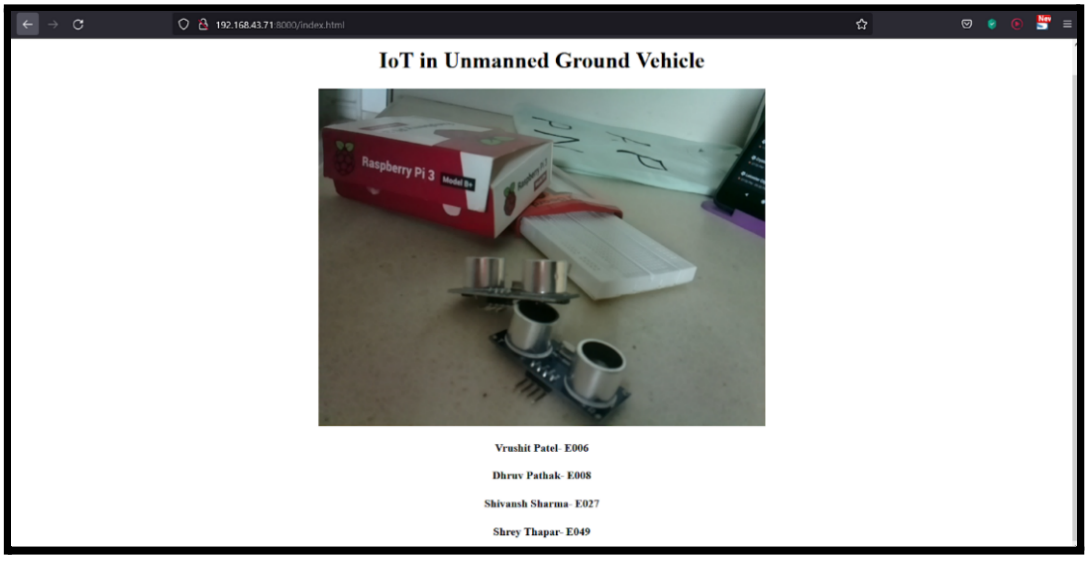


Figure 5.2.2.2.1 Live Feed Sample HTML page

Limitations:

- Complex to be uploaded on the internet.
- Increase in lag if the network becomes slower.

Figure 5.2.2.2.2 Sample Live Feed Mobile View

3. DC (Hobby) Motor with Driver Shield

Using an Arduino and the Arduino Motor Shield, you can simply control motor direction and speed. It's quite simple to incorporate a motor into your project because you can just address Arduino pins. It also enables you to run a motor from a separate power supply of up to 12V. A DC motor is the most common type of motor. There are only two leads in most DC motors: one positive and one negative. If these two lines are linked directly to a battery, the motor will spin. If the leads are switched, the motor will rotate in the opposite direction.

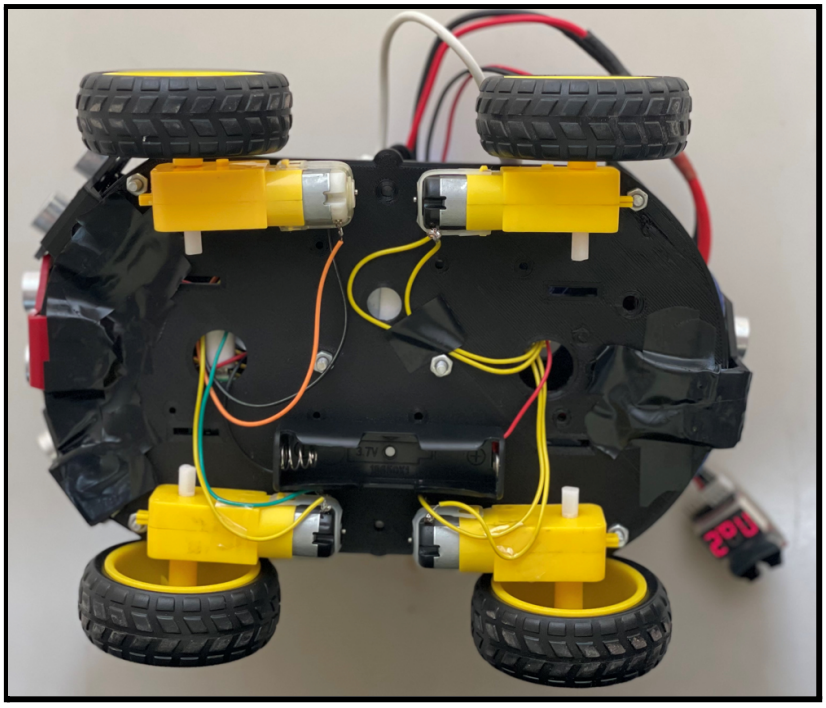


Figure 5.2.3.1 Hobby Motor Setup

The hobby motors are controlled as a 4-wheel drive for the rover. With different combinations of all 4 motors controlled, we have established the 4 basic movements to move forward, backward, left and right. The code for the execution is shown below.

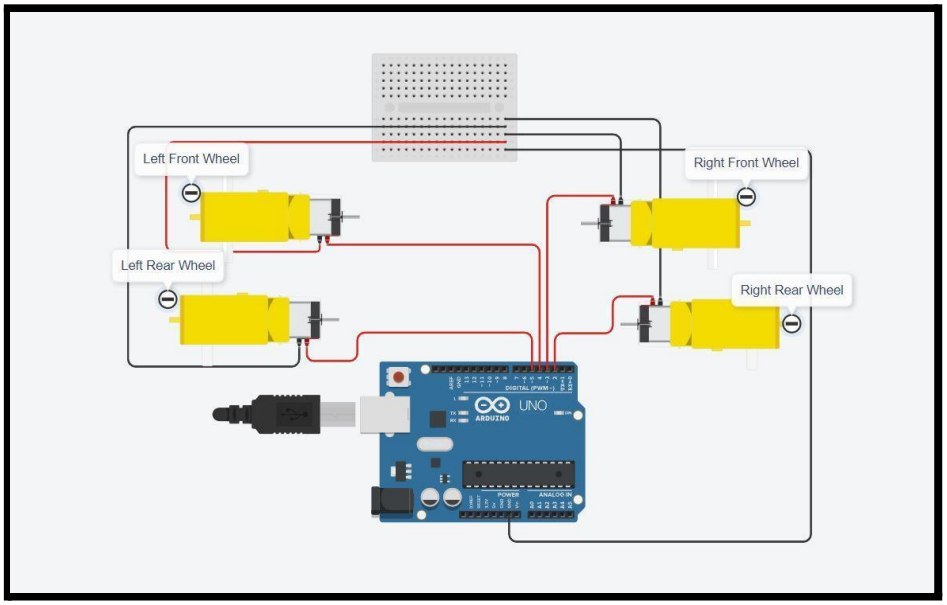


Figure 5.2.3.2 Hobby Motor Wheel Setup

```
void loop() {

  if (Serial.available()) {
    command = Serial.readStringUntil('\n');
    command.trim();
    if (command.equals("forward")) { //move forward(all motors rotate in forward direction)
      digitalWrite(2,LOW);
      digitalWrite(3,LOW);
      digitalWrite(4,LOW);
      digitalWrite(5,LOW);
      digitalWrite(2,HIGH);
      digitalWrite(5,HIGH);
    }

    else if (command.equals("backward")) { //move reverse (all motors rotate in reverse direction)
      digitalWrite(2,LOW);
      digitalWrite(3,LOW);
      digitalWrite(4,LOW);
      digitalWrite(5,LOW);
      digitalWrite(3,HIGH);
      digitalWrite(4,HIGH);
    }
    else if (command.equals("left")) { //turn left (right side motors rotate in forward direction,
      digitalWrite(4,LOW);
      digitalWrite(3,HIGH);
    }
    else if (command.equals("right")) { //turn right (left side motors rotate in forward direction
      digitalWrite(3,LOW);
      digitalWrite(4,HIGH);
    }
    else if (command.equals("stop")) { //turn right (left side motors rotate in forward direction,
      digitalWrite(2,LOW);
      digitalWrite(3,LOW);
      digitalWrite(4,LOW);
      digitalWrite(5,LOW);
    }
    Serial.print("Command: ");
    Serial.println(command);
  }
}
```

3.1 Serial Commands for DC Motor

Serial communication is nothing more than a means of transferring data. In contrast to parallel communication, which sends multiple bits at once, the data will be transferred sequentially, one bit at a time (1 byte = 8 bits). The serialisation code's implementation was previously shown in the Software Implementation section.

4. Li Po Battery

Lithium polymer (LiPo) batteries are used to power the motor shields connected to the Arduino UNO board. LiPo batteries are connected in series, and if you place regular batteries in series, the voltage of each cell will be different after use, and the charging time of each cell will be uneven, so you will get the desired voltage. To solve this problem, you can use the LiPo battery to charge each cell of the battery individually to prevent overcharging of the battery cell. Lithium-based batteries have advantageous characteristics such as high charge and discharge rates, durability, high energy density, and affordability, as well as a lightweight structure.

5. Buck Converter

The LiPo battery in the rover produces a higher amount of voltage than the required amount for powering the motors, thereby requiring a buck converter with a constant current, constant voltage (CC-CV) method. The buck converter then efficiently converts the high DC voltage generated by the LiPo battery to a low DC voltage, extending battery life and lowering the heat output.

6. Ultrasonic Sensor (HC-SR04)

An ultrasonic sensor is a device that uses ultrasonic sound waves to determine the distance between two objects. An ultrasonic sensor employs a transducer to emit and receive ultrasonic pulses that communicate information about the proximity of an item.
High-frequency sound waves bounce off boundaries, resulting in different echo patterns.

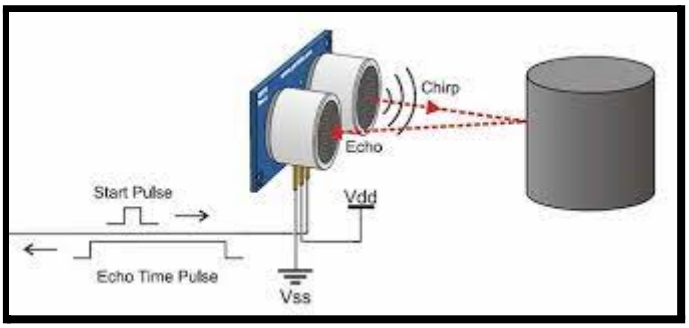


Figure 5.2.6.1 Ultrasonic Working

Ultrasonic sensors work by generating a sound wave at a frequency that is above the range of human hearing. The sensor's transducer serves as a microphone to receive and transmit ultrasonic sound. Our ultrasonic sensors, like many others, use a single transducer to generate a pulse and receive an echo. The sensor determines the distance to a target by measuring the time between transmitting and receiving an ultrasonic pulse.

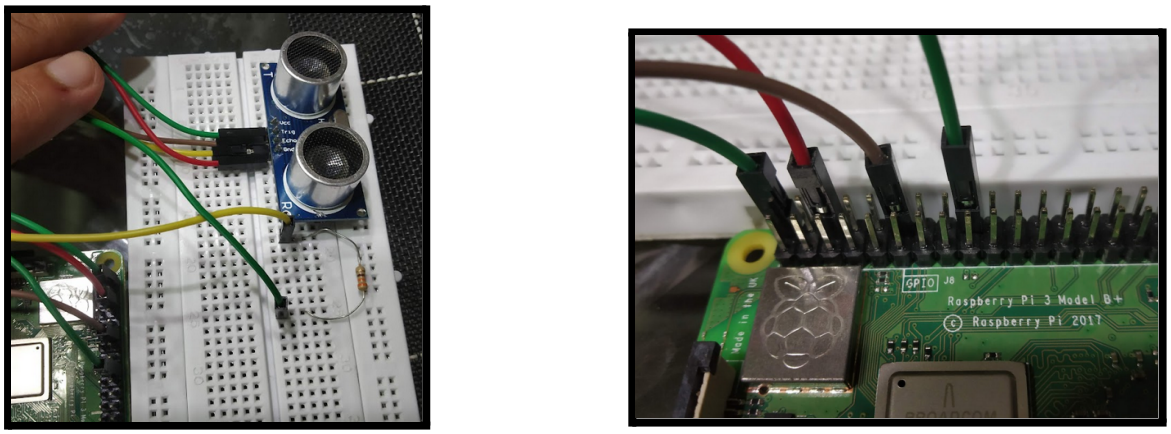


Figure 5.2.6.2 Ultrasonic Circuit

The code to find the distance based on the sonic waves is shown below. The output from this code is also attached below.

```
def distance():
    # set Trigger to HIGH
    GPIO.output(GPIO_TRIGGER, True)

    # set Trigger after 0.01ms to LOW
    time.sleep(0.00001)
    GPIO.output(GPIO_TRIGGER, False)

    StartTime = time.time()
    StopTime = time.time()

    # save StartTime
    while GPIO.input(GPIO_ECHO) == 0:
        StartTime = time.time()

    # save time of arrival
    while GPIO.input(GPIO_ECHO) == 1:
        StopTime = time.time()

    # time difference between start and arrival
    TimeElapsed = StopTime - StartTime
    # multiply with the sonic speed (34300 cm/s)
    # and divide by 2, because there and back
    distance = (TimeElapsed * 34300) / 2

    return distance
```

```
Measured Distance = 49.6 cm
Measured Distance = 51.2 cm
Measured Distance = 69.7 cm
Measured Distance = 50.9 cm
Measured Distance = 48.7 cm
Measured Distance = 21.7 cm
Measured Distance = 66.7 cm
Measured Distance = 55.6 cm
Measured Distance = 46.4 cm
```

Figure 5.2.6.3 Ultrasonic Output

7. Density, Humidity and Temperature Sensor (DHT11)

The DHT11 is a low-cost digital sensor that measures temperature and humidity. This sensor is simple to connect to any microcontroller. The code for it is provided below.

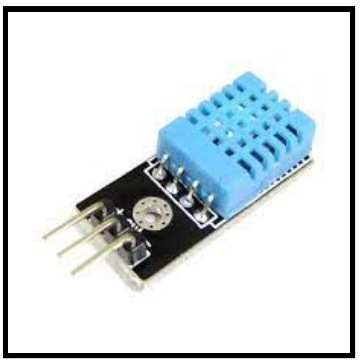


Figure 5.2.7.1 Humidity and Temperature Sensor

```
print("-----------------------------")
humidity, temperature = Adafruit_DHT.read_retry(11, 4)
print('Temp: {0:0.1f} C  Humidity: {1:0.1f} %'.format(temperature, humidity))
```

8. Air Quality Sensor (MQ135)

The MQ135 air quality sensor detects a wide variety of gases, including NH3, alcohol, benzene, smoke, and carbon dioxide. The MQ135 gas sensor is extremely sensitive to ammonia and sulphide, as well as smoke and other hazardous gases. It comes at a modest price.

Figure 5.2.8.1 Air Quality Sensor

```
int sensorValue;
void setup(){
Serial.begin(9600);                              // sets the serial port to 9600
 }
void loop(){sensorValue = analogRead(0);        // read analog input pin 0
Serial.print("AirQua=");
Serial.print(sensorValue, DEC);                 // prints the value read
Serial.println(" PPM");
delay(100);                                     // wait 100ms for next reading
}
```

9. Body Frame

The rover body has been 3D Printed. The first prototype is shown below.

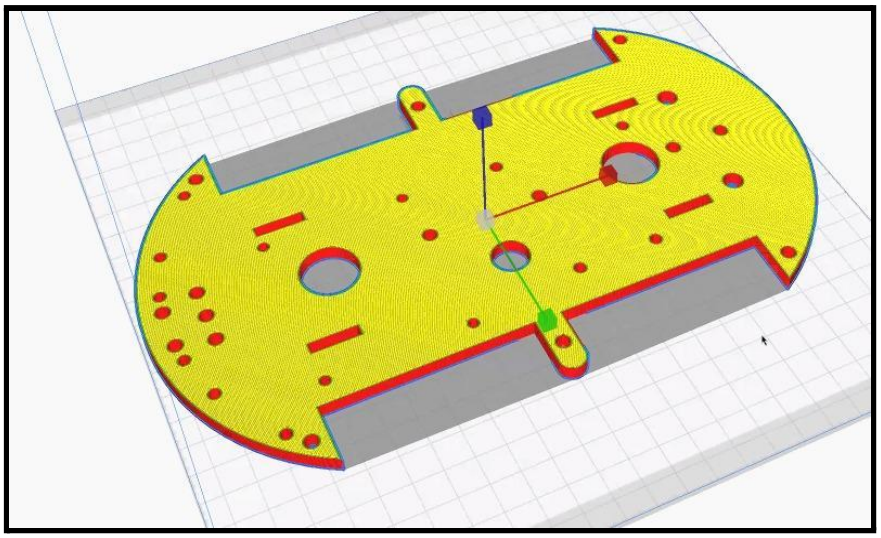


Figure 5.2.9.1 Body Design

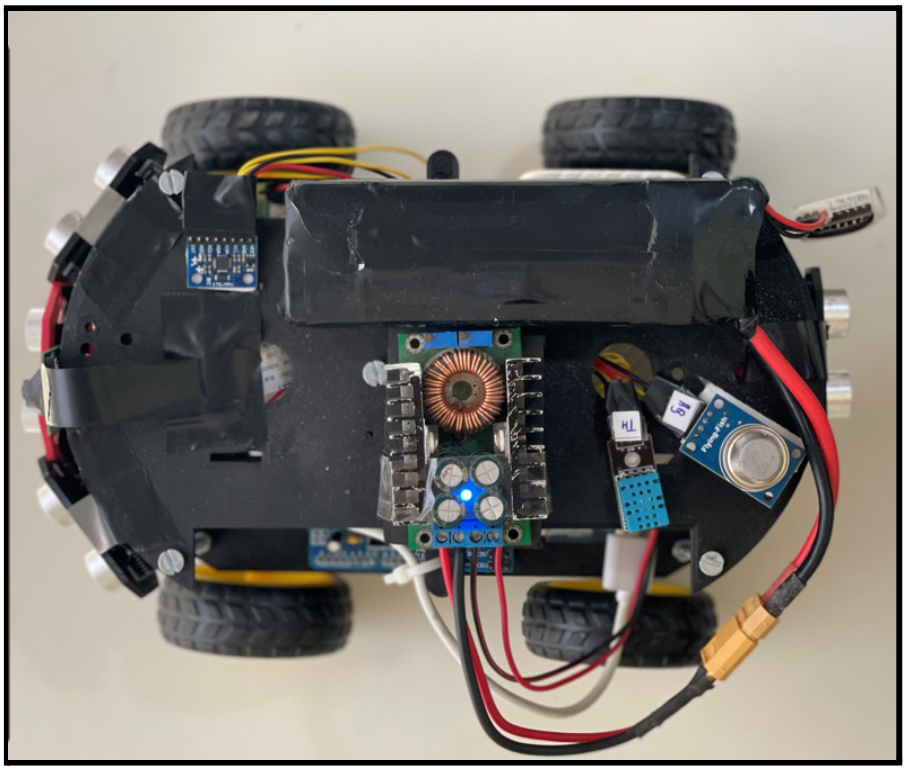


Figure 5.2.9.2 Rover Top View

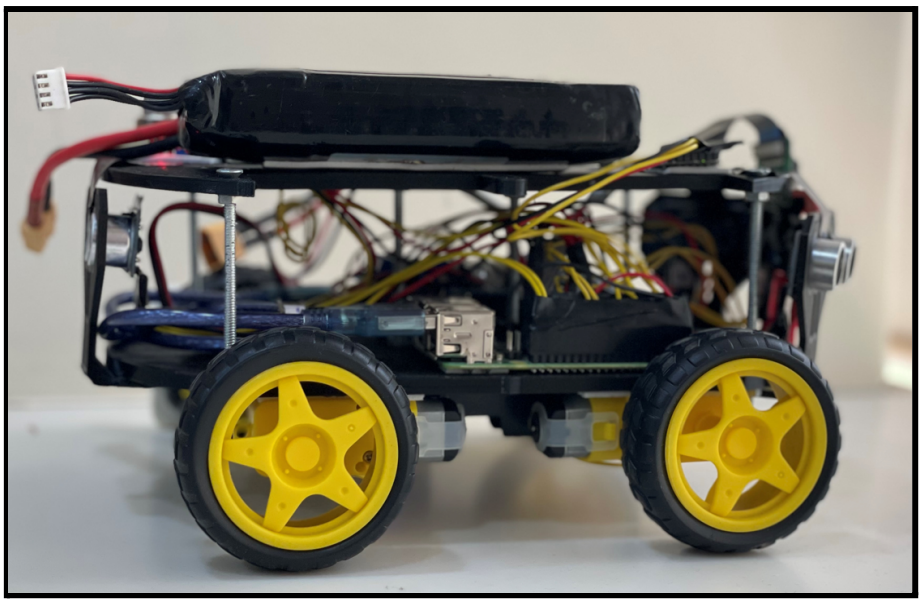


Figure 5.2.9.3 Rover Side View

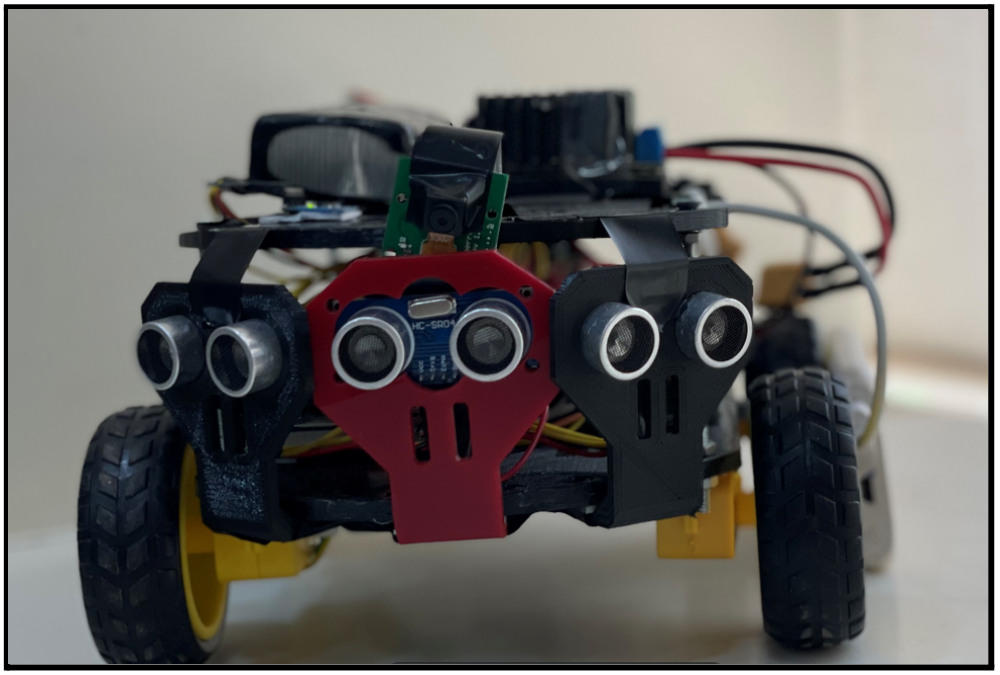


Figure 5.2.9.4 Rover Front View

Chapter 6: CONCLUSION & FUTURE SCOPE

6.1 Conclusion

On completion of the report, it can be safely concluded that after going through the various kinds of system architectures by reading research papers on “IoT in Unmanned Ground Vehicles”, the team has a much better understanding of system designs and architectures that are used in existing models of unmanned vehicles.

After obtaining some first hand experience on the working of Raspberry Pi and Arduino, we were able to set them up and tested their working using different sensors. For example, connecting Raspberry Pi with an ultrasonic sensor, gyroscope sensor and a camera module. We also connected the motor drive module to Arduino and the DC motors to the motor drive. We were successful in live streaming the video content from the Pi Camera attached to the Raspberry Pi using VLC Application on a remote device.

On collecting data from different sensors, we linked Raspberry Pi to AWS cloud using AWS IoT core to store the data. On gaining understanding on the topic by going through various algorithms in the research papers and after weighing the pros and cons of each algorithm, we selected the best suited algorithms for each domain of our project.

6.2 Future Plans

After getting a hand of Raspberry Pi and Arduino, we would be assembling all the components required for a working model of “an Unmanned Vehicle” and implement various algorithms using data from sensors for each domain like Object Detection using YOLO or Object Avoidance using DistBug.

On implementation of the above machine learning algorithms, we would head to Serialization, i.e., a mode of communication between Raspberry Pi and Arduino enabling Raspberry Pi to instruct Arduino for the movement of the vehicle.
For enabling the vehicle to have a manual mode, we aspire to design an application which grants us privileges of controlling the unmanned vehicle from a remote location using a device, like a smartphone.

6.3 TimeLine

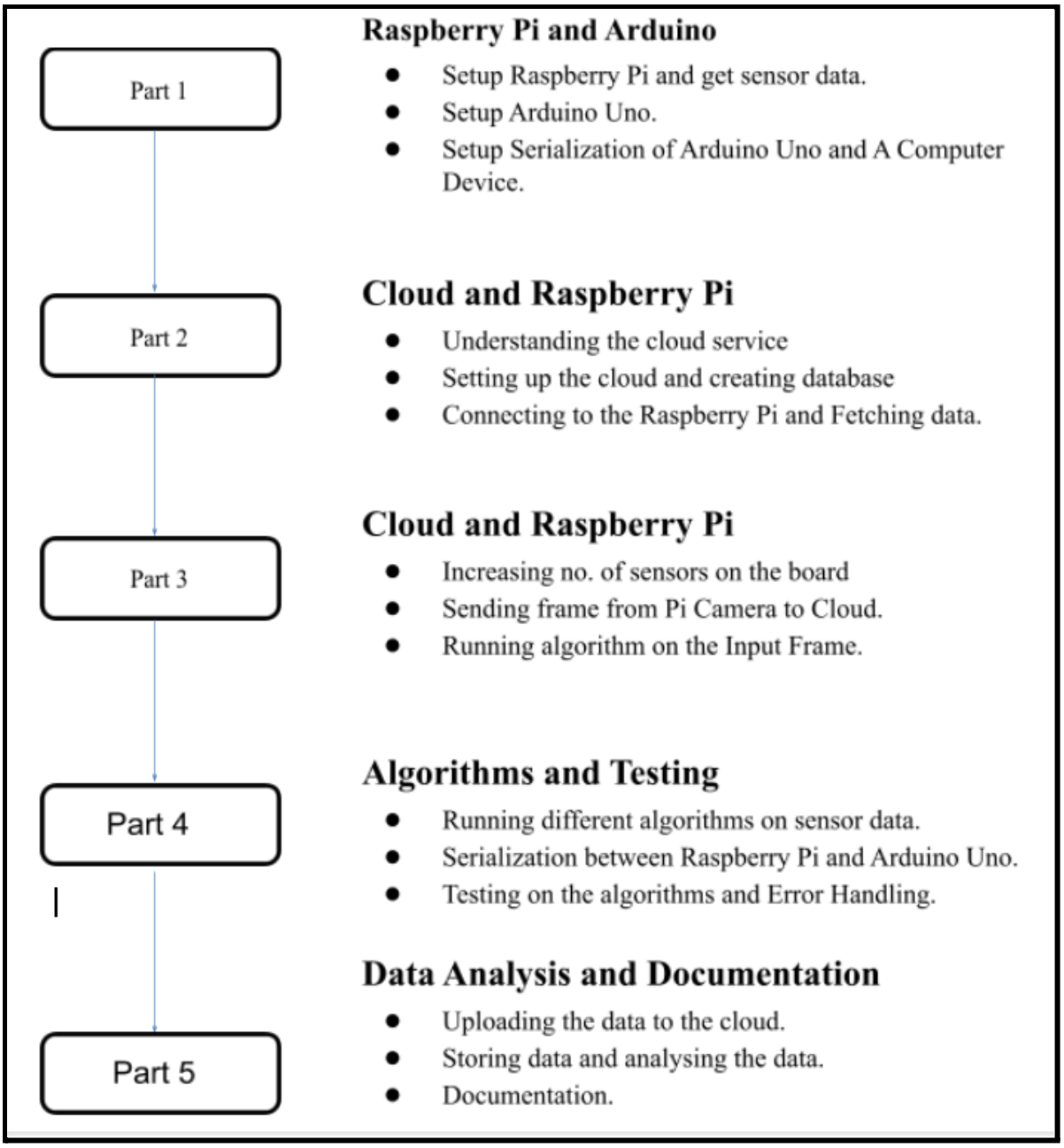


Figure 6.3 Timeline

REFERENCES

[1] Sonekar, S. V., Ditani, B. J., Pate, J. P., Shende, S. R., Shende, S. R., & Khadse, J. H. (2020, November). Developing an Unmanned Rover for Garbage Management System using Internet Protocol of Raspberry PI (RGMS). In *2020 IEEE International Conference for Innovation in Technology (INOCON)* (pp. 1-4). IEEE.

[2] Hossain, N., Kabir, M. T., Rahman, T. R., Hossen, M. S., & Salauddin, F. (2015, November). A real-time surveillance mini-rover based on OpenCV-Python-JAVA using Raspberry Pi 2. In *2015 IEEE International Conference on Control System, Computing and Engineering (ICCSCE)* (pp. 476-481). IEEE.

[3] Shirolkar, R., Dhongade, A., Datar, R., & Behere, G. (2019). Self-Driving Autonomous Car using Raspberry Pi. *International Journal of Engineering Research Technology (IJERT) Volume*, *8*.

[4] Pannu, G. S., Ansari, M. D., & Gupta, P. (2015). Design and implementation of autonomous cars using Raspberry Pi. *International Journal of Computer Applications*, *113*(9).

[5] Krasniqi, X., & Hajrizi, E. (2016). Use of IoT technology to drive the automotive industry from connected to full autonomous vehicles. *IFAC-PapersOnLine*, *49*(29), 269-274.

[6] Wardihani, E., Ramdhani, M., Suharjono, A., Setyawan, T. A., Hidayat, S. S., Helmy, S. W., ... & Saifullah, F. I. R. D. A. N. I. S. (2018). Real-time forest fire monitoring system using unmanned aerial vehicle. *Journal of Engineering Science and Technology*, *13*(6), 1587-1594.

[7] Hossai, M. R. T., Shahjalal, M. A., & Nuri, N. F. (2017, February). Design of an IoT based autonomous vehicle with the aid of computer vision. In *2017 International Conference on Electrical, Computer and Communication Engineering (ECCE)* (pp. 752-756). IEEE.

[8] Daryanavard, H., & Harifi, A. (2018, May). Implementing face detection system on uav using raspberry pi platform. In *Electrical Engineering (ICEE), Iranian Conference on* (pp. 1720-1723). IEEE.

[9] Gazis, A., Ioannou, E., & Katsiri, E. (2019). Examining the sensors that enable self-driving vehicles. *IEEE Potentials*, *39*(1), 46-51.

[10] Husni, E., Hertantyo, G. B., Wicaksono, D. W., Hasibuan, F. C., Rahayu, A. U., & Triawan, M. A. (2016, July). Applied Internet of Things (IoT): car monitoring system using IBM BlueMix. In *2016 International Seminar on Intelligent Technology and Its Applications (ISITIA)* (pp. 417-422). IEEE.

[11] Sathiyanarayanan, M., & Azharuddin, S. (2014). Four different modes to control unmanned ground vehicle for military purpose.

[12] Boursianis, A. D., Papadopoulou, M. S., Diamantoulakis, P., Liopa-Tsakalidi, A., Barouchas, P., Salahas, G., ... & Goudos, S. K. (2020). Internet of things (IoT) and agricultural unmanned aerial vehicles (UAVs) in smart farming: a comprehensive review. *Internet of Things*, 100187.

[13] Matos, J., & Postolache, O. (2016, October). IoT enabled aquatic drone for environmental monitoring. In *2016 International conference and exposition on electrical and power engineering (EPE)* (pp. 598-603). IEEE.

[14] Ejaz, W., Azam, M. A., Saadat, S., Iqbal, F., & Hanan, A. (2019). Unmanned aerial vehicles enabled IoT platform for disaster management. *Energies*, *12*(14), 2706.

[15] Ch, G. D. S., Venegas, C., & Marrone, L. (2019, April). MQTT-Based Prototype Rover with Vision-As-A-Service (VAAS) in an IoT Dual-Stack Scenario. In *2019 Sixth International Conference on eDemocracy & eGovernment (ICEDEG)* (pp. 344-349). IEEE.

[16] Srisuphab, A., Silapachote, P., Tantratorn, W., Krakornkul, P., & Darote, P. (2018, October). Insect detection on an unmanned ground rover. In *TENCON 2018-2018 IEEE Region 10 Conference* (pp. 0954-0959). IEEE.

[17] Kumar, V. S., Gogul, I., Raj, M. D., Pragadesh, S. K., & Sebastin, J. S. (2016). Smart autonomous gardening rover with plant recognition using neural networks. *Procedia Computer Science*, *93*, 975-981.

[18] Washington, R., Golden, K., Bresina, J., Smith, D. E., Anderson, C., & Smith, T. (1999, March). Autonomous rovers for Mars exploration. In *1999 IEEE Aerospace Conference. Proceedings (Cat. No. 99TH8403)* (Vol. 1, pp. 237-251). IEEE.

[19] Girma, A., Bahadori, N., Sarkar, M., Tadewos, T. G., Behnia, M. R., Mahmoud, M. N., ... & Homaifar, A. (2020). IoT-enabled autonomous system collaboration for disaster-area management. *IEEE/CAA Journal of Automatica Sinica*, *7*(5), 1249-1262.

[20] Tutunji, T. A., Salah-Eddin, M., & Abdalqader, H. (2020, December). Unmanned Ground Vehicle Control using IoT. In *2020 21st International Conference on Research and Education in Mechatronics (REM)* (pp. 1-5). IEEE.

[21] Aldabbas, O., Abuarqoub, A., Hammoudeh, M., Raza, U., & Bounceur, A. (2016). Unmanned ground vehicle for data collection in wireless sensor networks: mobility-aware sink selection. *The Open Automation and Control Systems Journal*, *8*(1).

[22] Deshpande, H., Singh, A., & Herunde, H. (2020). Comparative analysis on YOLO object detection with OpenCV. *International Journal of Research in Industrial Engineering*, *9*(1), 46-64.

[23] Li, Q., Zheng, N., & Cheng, H. (2004). Springrobot: A prototype autonomous vehicle and its algorithms for lane detection. *IEEE Transactions on Intelligent Transportation Systems*, *5*(4),

300-308.

[24] Zhu, W., Liu, F., Li, Z., Wang, X., & Zhang, S. (2008, December). A vision based lane detection and tracking algorithm in automatic drive. In *2008 IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application* (Vol. 1, pp. 799-803). IEEE.

[25] Gupta, A., & Merchant, P. S. (2016). Automated lane detection by K-means clustering: a machine learning approach. *Electronic Imaging*, *2016*(14), 1-6.

[26] Amin, P., Anushree, B. S., Shetty, B., Kavya, K., & Shetty, L. (2019). Object detection using machine learning technique. *Int. Res. J. Eng. Technol.(IRJET)*, 7948-7951.

[27] Hou, Y., Ma, Z., Liu, C., & Loy, C. C. (2019). Learning lightweight lane detection cnns by self attention distillation. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 1013-1021).

[28] Oltean, G., Florea, C., Orghidan, R., & Oltean, V. (2019, October). Towards real time vehicle counting using yolo-tiny and fast motion estimation. In *2019 IEEE 25th International Symposium for Design and Technology in Electronic Packaging (SIITME)* (pp. 240-243). IEEE.

[29] Krasniqi, X. and Hajrizi, E., 2016. Use of IoT technology to drive the automotive industry from connected to full autonomous vehicles. *IFAC-PapersOnLine*, *49*(29), pp.269-274.

[30] Xu, Z., Li, S. and Chen, Q., 2016. The Open Automation and Control Systems Journal. *Open Automation and Control Systems Journal*, *8*, pp.47-66.

ACKNOWLEDGEMENT

The success and final outcome of this project required a lot of guidance and assistance from many people and I am extremely privileged to have got this all along the completion of my project. All that I have done is only due to such supervision and assistance and I would not forget to thank them.

I respect and thank NMIMS's Mukesh Patel School of Technology Management and Engineering, for providing me an opportunity to do the project work in the domain of IoT and giving us all support and guidance which made me complete the project duly. I am extremely thankful for providing such nice support and guidance.

We owe deep gratitude to our project guide Professor Sanjay Deshmukh, who took keen interest in our project work and guided us all along, till the completion of our project work by providing all the necessary information for developing a good system.

I would not forget to remember my friends and family for their encouragement and more over for their timely support and guidance till the completion of our project work.

I heartily thank my teammates for their guidance, patiences and suggestions during this project work.

I am thankful to and fortunate enough to get constant encouragement, support and guidance from all Teaching staff of the department of Computer Science of NMIMS University which helped us in successfully completing our project work. Also, I would like to extend our sincere appreciation to all staff in the laboratory for their timely support.