GESTURE CONTROLLED AUTOMATED CAR PARKING USING NEURAL NETWORK

MT6811 PROJECT WORK

A PROJECT REPORT

Submitted by

AJITH KEVIN ANAND.K	**211616115002**
ILAMPARITHI.K	**211616115021**
VIGNESH CHANDRU	**211616115052**
VISHAL CHANDRU	**211616115057**

in partial fulfilment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

MECHATRONICS ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE, THANDALAM

ANNA UNIVERSITY::CHENNAI 600 025

MARCH 2020

ANNA UNIVERSITY : CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"GESTURE CONTROLLED AUTOMATED CAR PARKING USING NEURAL NETWORK"** is the bonafide work of "**AJITH KEVIN ANAND.K (2116115002), ILAMPARITHI.K (211616115021), VIGNESH CHANDRU (211616115052), VISHAL CHANDRU (211616115057)"** who carried out the project under my supervision.

SIGNATURE
Dr. V.Santhanam,
Professor & Head,
Dept. of Mechatronics Engineering,
Rajalakshmi Engineering College,
Chennai – 602105.

SIGNATURE
Ms. D. Keerthana,
SUPERVISOR,
Assistant Professor,
Dept. of Mechatronics Engineering,
Rajalakshmi Engineering College,

This project is Submitted for VIVA-VOCE examination held on _____ at Rajalakshmi Engineering College, Chennai – 602105.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGMENT

Initially we thank the almighty with us through every walk of life. It is our privilege to express our sincerest thanks to our respected chairman **Mr. S. Meganathan, B.E., F.I.E.** and beloved chairperson **Dr. Mrs. Thangam Meganathan, M.A., M.Phil., Ph.D.** for providing us with the requisite infrastructure and extending support in all endeavours.

Our heartfelt thanks to **Dr. S. N. Murugesan,** our Principal for his kind support and resources provided to complete our work in time. We also thank **Dr. G. Thanigaiyarasu, B.E., M.Sc.,** Dean Mechanical Sciences for his suggestion and guidance for completion of project.

We deeply express our sincere thanks to **Dr. V. Santhanam,** Head of our Department, for his encouragement and continuous support to complete the project in time.

We are glad to express our sincere indebtedness to our project coordinators **Dr. M. Balakarthikeyan,** Assistant Professor, Department of Mechatronics Engineering for their constructive criticism throughout the duration of our project.

We are glad to express our sincere thanks and regards to our supervisor **Ms.D.Keerthana,** Assistant Professor, Department of Mechatronics Engineering for his guidance and suggestion throughout the course of the project.

Finally we express our thanks for all teaching, non-teaching faculty of our Mechatronics Engineering Department for helping us with the necessary suggestions and guidance during the time of project.

ABSTRACT

The goal of this project is to train a Neural network algorithm capable of classifying images of different hand gestures, such as a fist, palm, showing the thumb, and others. This particular classification problem can be useful for Gesture Navigation.

By the use of gesture detection, hands free motion of the vehicle is controlled, and this technology is used to automate the vehicle parking system, which was prior achievable only by the access of remote and limited motions. It enables the vehicle to use the collision prevention system and guide the vehicle out of the garage. This simulation is done, considering the electrical vehicle.

Machine Learning is very useful for a variety of real-life problems. It is commonly used for tasks such as classification, recognition, detection and predictions. Moreover, it is very efficient to automate processes that use data. The basic idea is to use data to produce a model capable of returning an output. This output may give a right answer with a new input or produce predictions towards the known data.

TABLE OF CONTENT

TABLE OF FIGURES

CHAPTER-1

INTRODUCTION

1. ELECTRIC VEHICLE

An electric vehicle, also called an electric vehicle, uses one more electric motors or traction motors for propulsion. An electric vehicle may be powered through a collector system by electricity from off-vehicle sources, or may be self-contained with a battery, solar panels or an electric generator to convert fuel to electricity. EV include, but are not limited to, road and rail vehicles, surface and underwater vessels, electric aircraft and electric spacecraft.

Electric vehicle's first came into existence in the mid-19th century, when electricity was among the preferred methods for motor vehicle propulsion, providing a level of comfort and ease of operation that could not be achieved by the gasoline cars of the time. Modern internal combustion engines have been the dominant propulsion method for motor vehicles for almost 100 years, but electric power has remained commonplace in other vehicle types, such as trains and smaller vehicles of all types.

Commonly, the term electric vehicle is used to refer to an electric car. In the 21st century, electric vehicle's saw a resurgence due to technological developments, and an increased focus on renewable energy. A great deal of demand for electric vehicles developed and a small core of do-it-yourself (DIY) engineers began sharing technical details for doing electric vehicle conversions. Government incentives to increase adoptions were introduced, including in the United States and the European Union.

2. MACHINE LEARNING

In the automotive industry, machine learning (ML) is most often associated with product innovations, such as self-driving cars, parking and lane-change assists, and smart energy systems. But ML is also having a significant effect on the marketing function, from how marketers in the automotive sector establish goals and measure returns on their investments to how they connect with consumers. ML is poised to become as much an organizing principle as an analytic ingredient for sophisticated marketing campaigns across industries. This is especially true in the automotive industry, a capital-intensive, high-tech sector riven by disruption.

3. ADVANTAGES OF MACHINE LEARNING

1.3.1 Quality Control

Image recognition and anomaly detection are types of machine learning algorithms that can quickly detect and eliminate faulty parts before they get into the vehicle manufacturing workflow. Parts manufacturers can capture images of each component as it comes off the assembly line, and automatically run those images through a machine learning model to identify any flaws. Highly-accurate anomaly detection algorithms can detect issues down to a fraction of a millimetre. Predictive analytics can be used to evaluate whether a flawed part can be reworked or needs to be scrapped. Eliminating or re-working faulty parts at this point is far less costly than discovering and having to fix them later. It saves on more expensive issues down the line in manufacturing and reduces the risk of costly recalls. It also helps ensure customer safety, satisfaction and retention.

1.3.2 Root Cause Analysis

When an issue arises at any point in the product lifecycle — whether it's something found early in the manufacturing process or an issue affecting multiple vehicles in the field — organizations scramble to determine the exact cause and how to resolve it. The brand's reputation (and possibly consumer safety) are at stake.

During the manufacturing phase, identifying the root cause(s) of an issue is a lengthy and painstaking process. Root cause analysis uses massive amounts of testing data, sensor measurements, manufacturer parameters and more. Performed with traditional methods, it's also incredibly hard.

Root cause analysis for issues in the field isn't any easier. Today's vehicles are highly complex, and each driver has unique behaviour, maintenance actions and driving conditions. Some issues arise only under very unique circumstances that were unseen in the manufacturing process.

1.3.3 Predictive Maintenance

Machine learning can provide far more precise and — importantly — evolving maintenance recommendations to help drivers protect their vehicle investment as well as their safety. Rather than a static maintenance schedule that gets updated a few times a year, a predictive analytics model can continue to learn from thousands of performance data points collected from manufacturing plants, suppliers, service providers and actual vehicles on the road. The industry is well on its way to completely customized maintenance schedules that evolve over time to be increasingly more tailored to individual drivers and vehicles, and can even adapt to changing conditions and new performance information.

Predictive maintenance helps increase customer satisfaction and brand reputation, while also improving compliance with recommended maintenance. It can also be a source of additional revenue for car makers as an added-value service.

1.3.4 Supply Chain Optimization

Throughout the supply chain, analytical models are used to identify demand levels for different marketing strategies, sale prices, locations and many other data points. Ultimately, this predictive analysis dictates the inventory levels needed at different facilities. Data scientists constantly test different scenarios to ensure ideal inventory levels and improve brand reputation while minimizing unnecessary holding costs.

After analyzing the gap between current and predicted inventory levels, data scientists then create optimization models that help guide the exact flow of inventory from manufacturer to distribution centers and ultimately to customer-facing storefronts. Machine learning is helping parts and vehicle manufacturers — and their logistics partners — be more efficient and profitable, while enhancing customer service and brand reputation as shown in Fig 1.1.

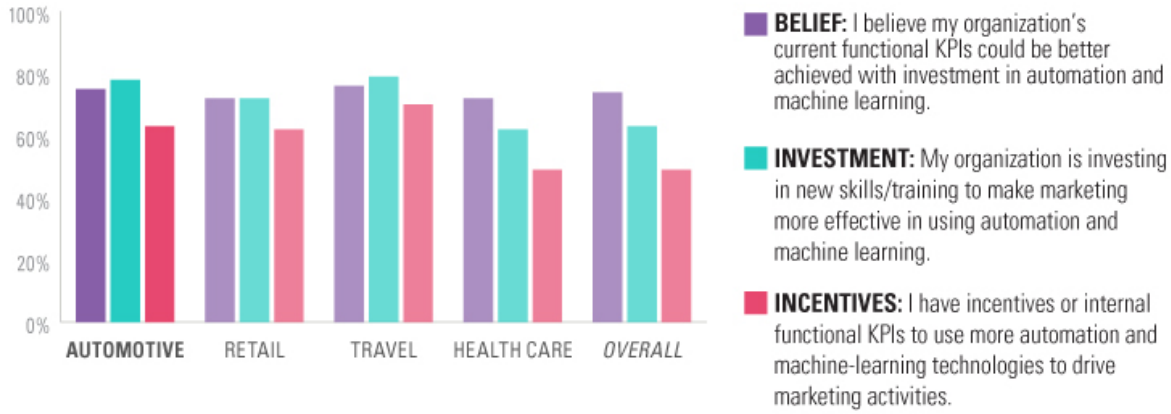

Fig 1.1 Machine Learning in Automotive Industry

1.4 IMAGE PROCESSING

Images have always played an important role in human life since vision is probably human beings' most important sense. As a consequence, the field of image processing has numerous applications (medical, military, etc.). Nowadays and more than ever, images are everywhere and it is very easy for everyone to generate a huge amount of images, thanks to the advances in digital technologies. With such a profusion of images, traditional image processing techniques have to cope with more complex problems and have to face their adaptability according to human vision. With vision being complex, machine learning has emerged as a key component of intelligent computer vision programs when adaptation is needed (e.g., face recognition). With the advent of image datasets and benchmarks, machine learning and image processing have recently received a lot of attention. An innovative integration of machine learning in image processing is very likely to have a great benefit to the field, which will contribute to a better understanding of complex images. The number of image processing algorithms that incorporate some learning components is expected to increase, as adaptation is needed. However, an increase in adaptation is often linked to an increase in complexity, and one has to efficiently control any machine learning technique to properly adapt it to image processing problems. Indeed, processing huge amounts of images means being able to process huge quantities of data often of high dimensions, which is problematic for most machine learning techniques. Therefore, an interaction with the image data and with image priors is necessary to drive model selection strategies.

1.5 PROCESS OF IMAGE PROCESSING

Most of the algorithm explained have been implemented in Python . The sk-learn library is utilized wherever appropriate. All the source codes of this research can be found in source code folder.

1.5.1 Standard Approach

- Structuring the initial dataset Our initial MNIST dataset consists of 6000 digits representing the numbers 0-9. These images are grayscale, 8x8 images with digits appearing as white on a black background. These digits have also been classified using image classification schemes.
- Splitting the dataset: We will be using three splits for our experiment. The first set is our training set, used to train our kNN classifier. We'll also use a validation set to find the best value for k. And we'll finally evaluate our classifier using the testing set.
- Extracting features: We will use the raw, grayscale pixel intensities of the image.
- Training our classification model: Our k-NN classifier will be trained on the raw pixel intensities of the images in the training set. We will then determine the best value of k using the validation set.
- Evaluating our classifier: Once we have found the best value of k, we can then evaluate our k-NN classifier on our testing set.

1.5.2 Preparation of Training data and Test data :

To achieve the classification by computer, the computer must be trained. Training is key to the success of classification MNIST data set has training and test data set (previous split of 60k/10k). After applying PCA, we obtained another set of data.

1.5.3 Set up for obtaining 2D data from original date

Feature selection algorithms attempt to select relevant features with respect to the performance task, or conversely remove redundant or irrelevant ones. In

contrast, the goal of dimensionality reduction techniques is to literally transform the raw input features while preserving the global information content. In essence, the dimensionality reduction algorithms attempt to extract features capable of reconstructing the original high dimensional data, irrespective to the classification label assigned to each data point. For example, principle components analysis (PCA) attempts to find a linear combination of principal components that preserves the variance of the data. The dimension of the original data has been reduced by applying PCA. This implementation is performed in Matlab platform .

1.5.4 Visualize the 2D data :

To interpret each component, we must estimate the correlations between the original data for each variable and each principal component. These correlations are obtained using the correlation procedure. In the variable statement we will include the first three principal components, x1, x2, and x3. The Scatter diagram of MNIST data, represents the relationship between the first three principle components in MNIST dataset.

1.6 GESTURE CONTROL

Gesture control is the ability to recognize and interpret movements of the human body in order to interact with and control a computer system without direct physical contact. The term “natural user interface” is becoming commonly used to describe these interface systems, reflecting the general lack of any intermediate devices between the user and the system.

Gesture recognition is a topic in computer science and language technology with the goal of interpreting human gestures via mathematical algorithms. Gestures can originate from any bodily motion or state but commonly originate from the face or hand as shown in Fig 1.2. Current focuses in the field include emotion recognition from face and hand gesture recognition. Users can use simple gestures to control or interact with devices without physically touching them. Many approaches have been made

using cameras and computer vision algorithms to interpret sign language. However, the identification and recognition of posture, gait, proxemics, and human behaviors is also the subject of gesture recognition techniques. Gesture recognition can be seen as a way for computers to begin to understand human body language, thus building a richer bridge between machines and humans than primitive text user interfaces or even GUIs (graphical user interfaces), which still limit the majority of input to keyboard and mouse and interact naturally without any mechanical devices. Using the concept of gesture recognition, it is possible to point a finger at this point will move accordingly.

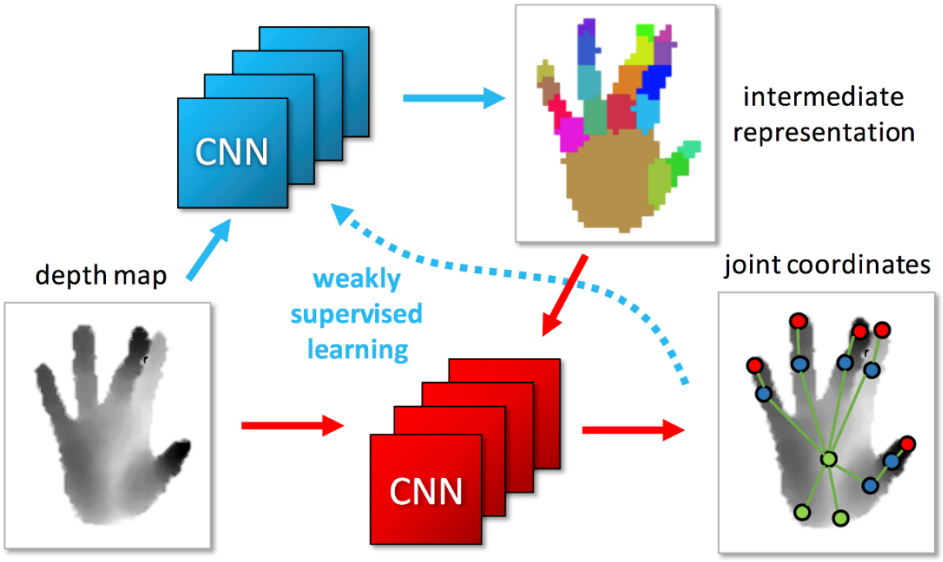

Fig 1.2 Gesture Recognition and Classification

CHAPTER - 2

LITERATURE REVIEW

2.1 GESTURE CONTROL MOTIONS

Pei Jia; Huosheng H. Hu et.al (2007) presents a novel hands-free control system for intelligent wheelchairs (IWs) based on visual recognition of head

gestures.A robust head gesture-based interface (HGI), is designed for head gesture recognition of the RoboChair user. The recognised gestures are used to generate motion control commands to the low-level DSP motion controller so that it can control the motion of the RoboChair according to the user's intention.

Masafumi Hashimoto et.al (2009) presented a nonverbal interface that used biopotential signals, such as electrooculargraphic (EOG) and electromyographic (EMG), captured by a simple brain-computer interface. The nonverbal interface to hands-free control of an electric wheelchair is achieved. Based on the biopotential signals, the interface recognizes the operator's gestures, such as closing the jaw, wrinkling the forehead, and looking towards left and right. By combining these gestures, the operator controls linear and turning motions, velocity, and the steering angle of the wheelchair. Experimental results for navigating the wheelchair in a hallway environment confirmed the feasibility of the proposed method.

2.2 GESTURE RECOGNITION

David Sachs et.al (2009) has studied on mobile devices using motion gesture recognition. In one aspect, processing motion to control a portable electronic device includes receiving, on the device, sensed motion data derived from motion sensors of the device and based on device movement in space. The motion sensors include at least three rotational motion sensors and at least three accelerometers. A particular operating mode is determined to be active while the

movement of the device occurs, the mode being one of multiple different operating modes of the device.

Guy Hoffman et.al (2010) presented Shimon, an interactive improvisational robotic marimba player, developed for research in Robotic Musicianship. The robot listens to a human musician and continuously adapts its improvisation and choreography, while playing simultaneously with the human. They discuss the robot's mechanism and motion-control, which uses physics simulation and animation principles to achieve both expressivity and safety and then present a novel interactive improvisation system based on the notion of gestures for both musical and visual expression. The system also uses anticipatory beat-matched action to enable real-time synchronization with the human player.

Jens Lambrecht et.al (2011) has developed a gesture-based control systems for industrial robotics benefit from a natural form of human machine interaction. Among the high costs of such systems, there are high requirements for safety, robustness of the gesture recognition and general applicability in the industrial environment. Novel consumer electronic sensors provide a robust recognition, reliability and adequate accuracies at low costs

2.3 INFERENCE FROM LITERATURE SURVEY

From the above literature survey we have inferred that gesture has been implemented for the purpose of handsfree motion control for various purposes such as wheel chair, operation of mobile devices, control of robots etc. Motion controllers are used for the purpose of gesture recognition. Image Processing has been used from early stages, and as the time passes, it has been improving at a faster pace. Usage of machine learning in such gesture recognition has come to

light in recent years with the development of improved computational power and increased storage accessibility.

CHAPTER - 3

METHODOLOGY

3.1 LOADING DATA

It contains around 100 images with different hands and hand gestures. There is a total of 10 hand gestures of 10 different people presented in the data set. There are 5 female subjects and 5 male subjects.

With that, we have to prepare the images to train the algorithm. We have to load all the images into a file and the label of the file is stored according to the gestures.

We then split our data into a training set and a test set. The training set will be used to build our model. Then, the test data will be used to check if our predictions are correct. A random_state seed is used so the randomness of our results can be reproduced. The function will shuffle the images it's using to minimize training loss.

3.2 TRAINING MODEL

To simplify the idea of the model being constructed here, we're going to use the concept of Linear Regression. By using linear regression, we can create a simple model and represent it using the equation,

$$y = ax + b \tag{3.1}$$

Where

- a and b (slope and intercept, respectively) are the parameters that we're trying to find. By finding the best parameters, for any given value of x, we can

predict y. This is the same idea here, but much more complex, with the use of Convolutional Neural Networks.

A Convolutional Neural Network (Fig 2.1) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases)

to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, CNNs have the ability to learn these filters/ characteristics.

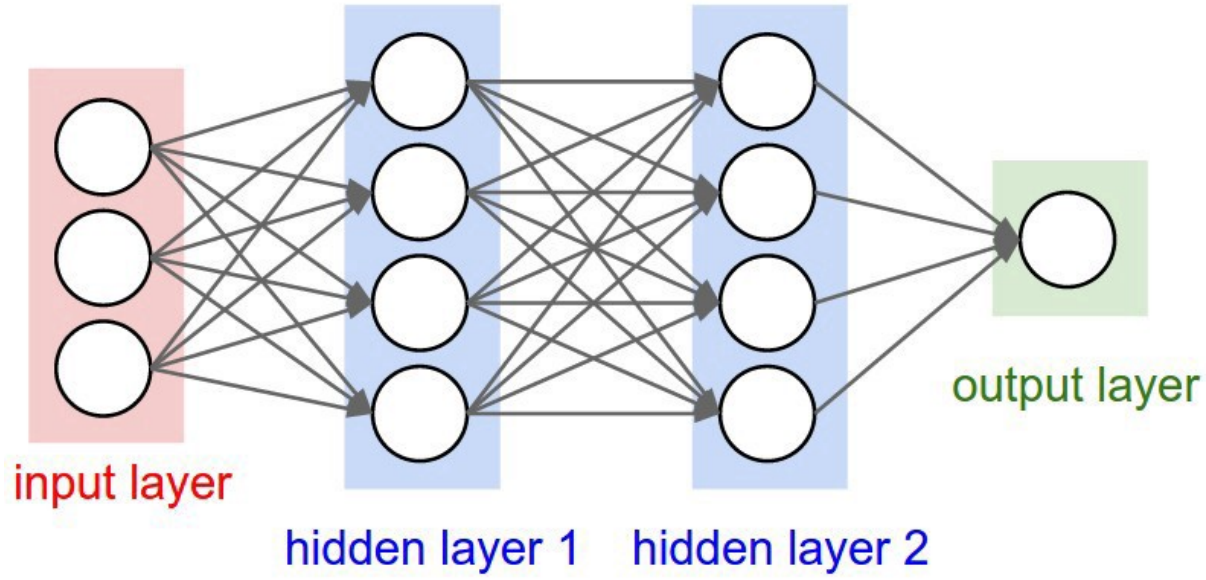

Fig 3.1 Layers of Neural Network

3.3 TESTING MODEL

Now that we have the model compiled and trained, we need to check if it's good. First, we run the model to test the accuracy. Then, we make predictions and plot the images as long with the predicted labels and true labels to check everything. With that, we can see how our algorithm is working.

3.4 AUTOMATED CAR PARKING

After the model is successfully built, then the camera detects for any gestures. Once a gesture is produced in front of the vehicle cameras, the vehicle begins to move, with the assistance of collision resistance system which, uses multiple sensors around the car to map the surroundings and prevent the car from colliding. According to the gestures provided the car moves in the desired directions.

The gesture provided in front of the camera is captured and this image is compared with the image which is used to test inside the model. This image is then classified, under the type of gesture produced. Accordingly, the commands for the provided gesture is sent to the microprocessor built inside the car, where in we use Arduino for simulation purposes. Thus, the car can be brought in and out of the garage.

Fig 3.2 Car Parking Sensor

3.4 FLOW CHART OF IMAGE CLASSIFIER PROCESS

TRAINING IMAGES

TESTING IMAGE

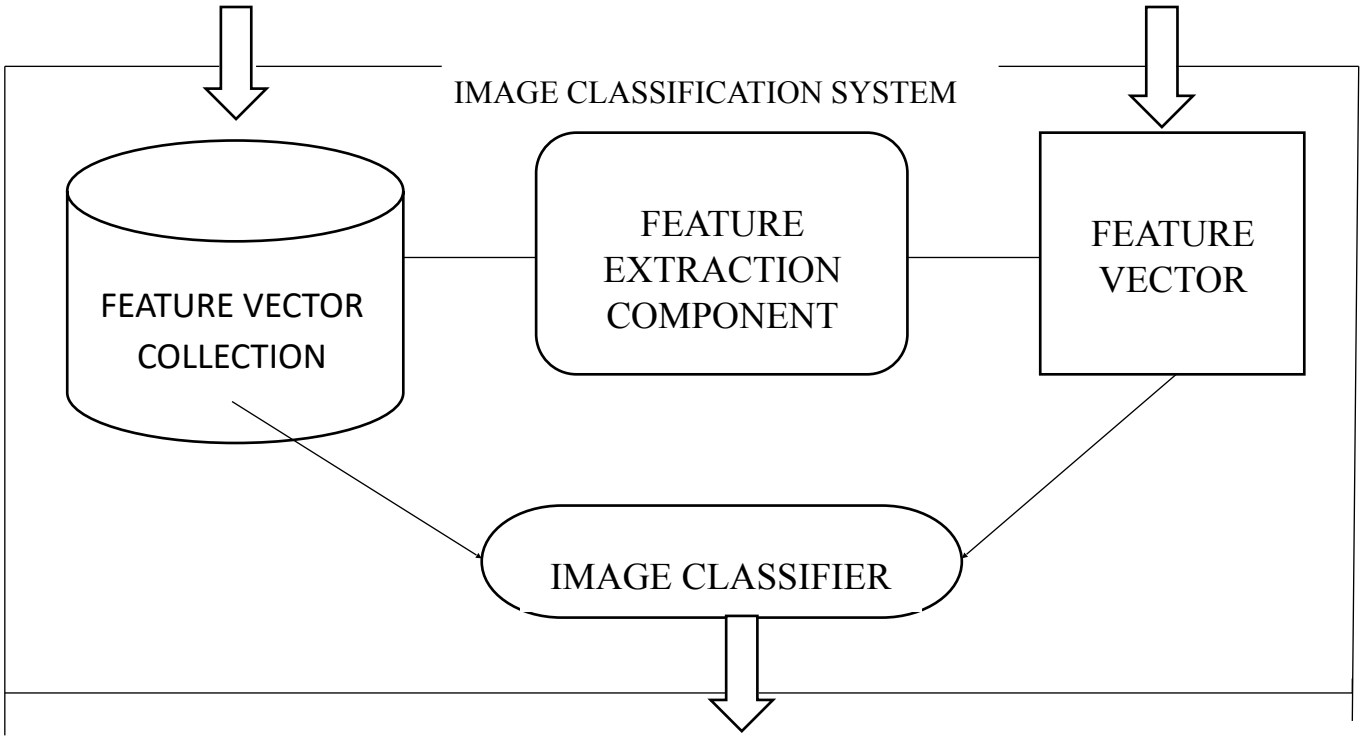

FIG.3.3 Flowchart for Image Classification System

CHAPTER – 4

FABRICATION PROCESS

4.1 MATERIALS USED

4.1.1 Fabrication

- Aluminium Sheet (29.5 x 22.3 cm)
- Wheel (Diameter: 7cm, width: 2cm ; Shaft hole: 6mm)
- Nut and Bolt

4.2 ELECTRICAL COMPONENTS

4.2.1 DC Motor

- Output RPM: 100 rpm
- Input Voltage: 6-12 V
- Stall Current: 500 - 600 mA
- Shaft length: 2.4 cm
- Shaft diameter: 6mm with internal hole
- Motor weight: ~100gms

4.2.2 Arduino UNO R3

- Microcontroller: ATmega328
- Operating Voltage: 5V
- Input Voltage (recommended): 7-12V
- Clock Speed: 16 MHz
- DC Current per I/O Pin: 40 mA

3. Power Distribution Board version 2

3. SOFTWARE

4.3.1 Design Software

- AutoCAD

4.3.2 Coding Software

- Python 3.7 IDE
- Arduino
- IP Webcam app

4.3.3 Operating Software

- Windows 10

- **ALUMINIUM SHEET**

Fig 4.1 Aluminium Sheet

Aluminium is one of the most used metals in modern industry, due to the optimal combination of several properties contained in it. Aluminium has a low specific weight; this characteristic offers huge advantages in many industrial applications.

Extruded semi-finished products can be offered in a wide range of profiles to drawing, extruded and drawn round and shaped bars. The variety of alloys makes possible the use of aluminium in several applications that can be very different, among which hot stamping.

For the purpose of building a prototype, aluminium sheet is used. Aluminium is preferred because of its high malleability, it can be easily folded. The body of the prototype electric car is built using aluminium sheet. The chassis of the car is built with holes in the aluminium sheet which embodies along the sides of the prototype.

- **DC MOTOR**

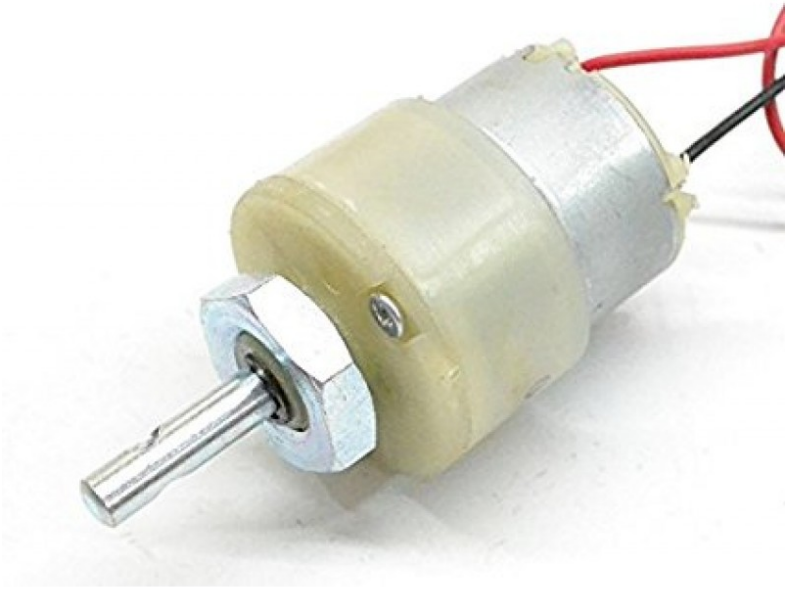

Fig 4.2 DC Motor

A DC motor is any of a class of rotary electrical motors that converts direct current electrical energy into mechanical energy. The most common types rely on the forces produced by magnetic fields. Nearly all types of DC motors have some internal mechanism, either electromechanical or electronic, to periodically change the direction of current in part of the motor.

DC motors were the first form of motor widely used, as they could be powered from existing direct-current lighting power distribution systems. A DC motor's speed can be controlled over a wide range, using either a variable supply voltage or by changing the strength of current in its field windings. Small DC motors are used in tools, toys, and appliances. The universal motor can operate on direct current but is a lightweight brushed motor used for portable power tools and appliances. Larger DC motors are currently used in propulsion of electric vehicles, elevator and hoists, and in drives for steel rolling mills. The advent of power

electronics has made replacement of DC motors with AC motors possible in many applications.

DC motors have the advantage of: higher starting torque, quick starting and stopping, reversing, variable speeds with voltage input and they are easier and cheaper to control than AC. As the prototype of Electric Vehicle involves the use of battery, DC motor is preferred

- **ARDUINO UNO R3**

Fig 4.3 Arduino UNO R3

Arduino Uno is a microcontroller board based on the ATmega328P (datasheet). 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. You can tinker with your UNO without wearing too much about doing something wrong, worst case scenario you can replace the chip for a few dollars and start over again

We have used Arduino UNO R3 for the following purposes

- It can be programmed with C++ language.
- Arduino Uno uses a different USB chip which makes installation of the Arduino software lot easier.
- Has higher speeds of communication with the computer.
- Comes equipped with the ATmega328 Microcontroller, which has more memory.
- The processor can be easily replaced if damaged.
- Can supply more current on its 3.3V supply.
- Arduino is an open-source electronics prototyping platform.
- It is flexible, easy-to-use hardware and software

- **POWER DISTRIBUTION BOARD VERSION 2**

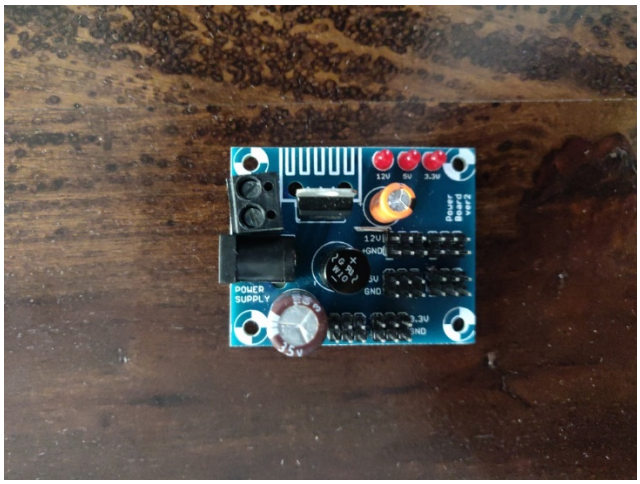

Fig 4.4 Power Distribution Board

Power Distribution Board (PDB) is a printed circuit board that is used to distribute the power from your flight battery to all different components of the multirotor. Prior to PDB's becoming common it was necessary to connect all the different components using wire and the result often resembled an octopus and weighed a considerable amount due to the amount of copper and solder joints in the wires. There are many different PDB's available from various manufacturers, the majority of them provide very similar features. Initially PDB's were very simple and were just a thick copper PCB with an input and multiple outputs. As the need for regulated voltages for various components has become more common, manufacturers have begun including voltage regulators on the PDB so that voltage sensitive components can be fed reliable, stable, and clean power.

A distribution board (also known as panel board, breaker panel, or electric panel) is a component of an electricity supply system that divides an electrical power feed into subsidiary circuits, while providing a protective fuse or circuit breaker for each circuit in a common enclosure. It protects the circuit from damaging due to high voltage, and also helps in providing equal distribution of voltage to all the 4 motors of the prototype built for this project.

- **JUMPER WIRES**

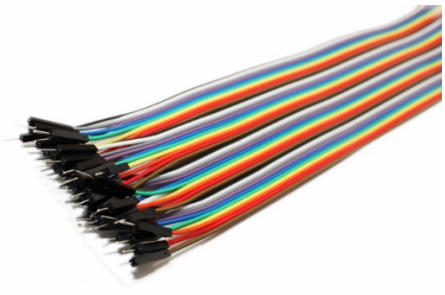

Fig 4.5 Jumper Wires

A jump wire (also known as jumper wire, or jumper) is an electrical wire, or group them in a cable, with a connector or pin at each end (or sometimes without them - simply "tinned"), which is normally used to interconnect the components of a breadboard or other prototype or test circuit, internally or with other equipment or components, without soldering.

- **WHEELS**

Fig 4.6 Wheel

Specifications of wheel :

- Wheels specifically designed for robotics
- Diameter: 7cm
- width: 2cm
- Shaft hole: 6mm
- Easily fits into a 6mm shaft dc gear motor

CHAPTER-5

PROTOTYPE PHOTOS

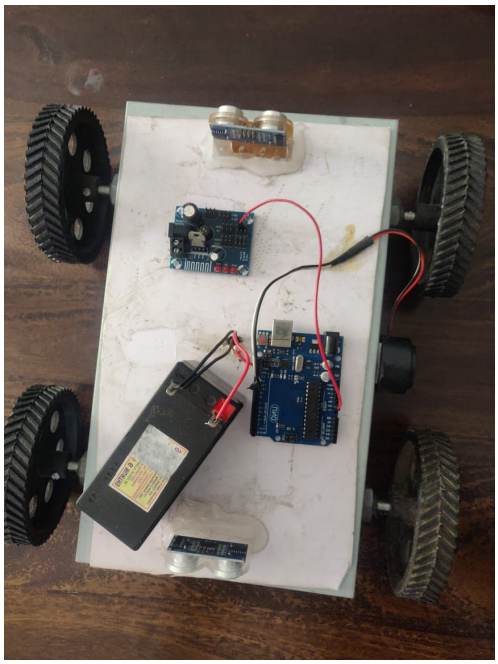

FIG 5.1 Prototype Top View

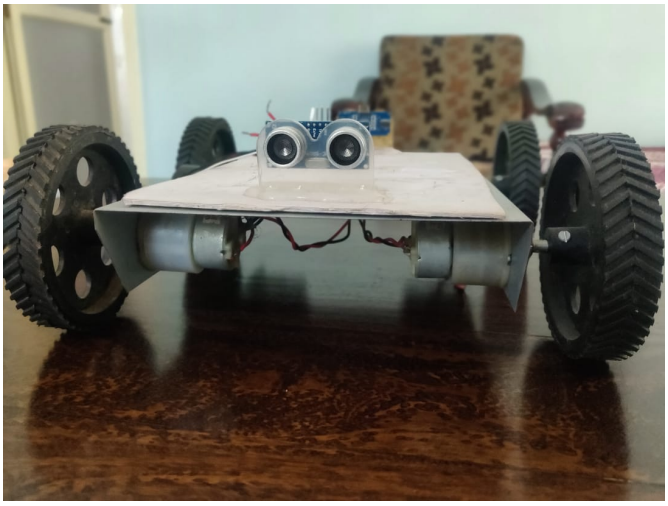

FIG 5.2 Prototype Front View

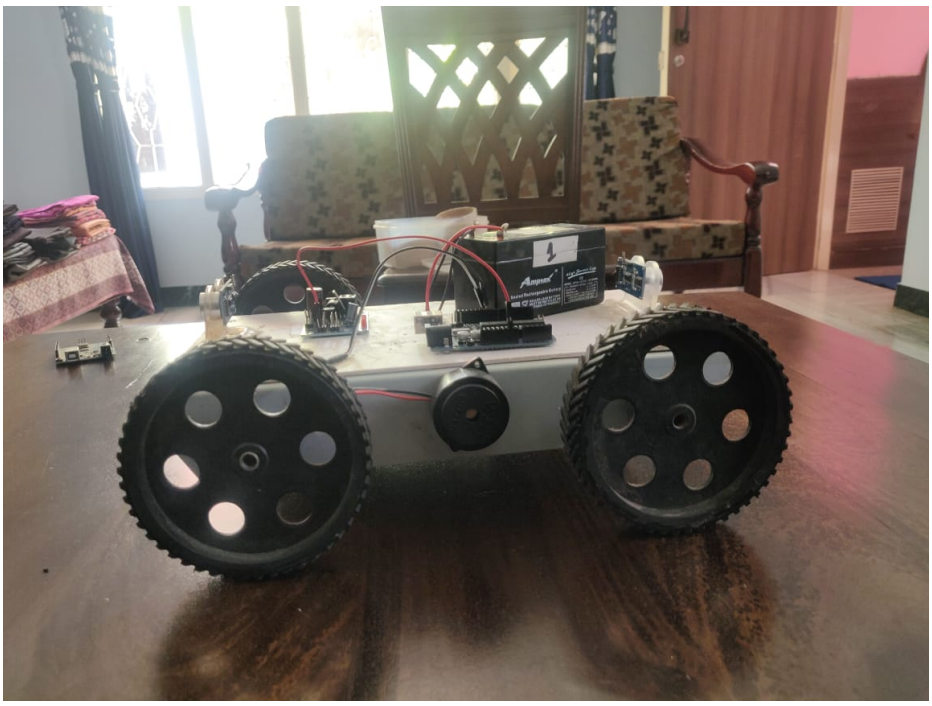

FIG 5.3 Prototype Side View

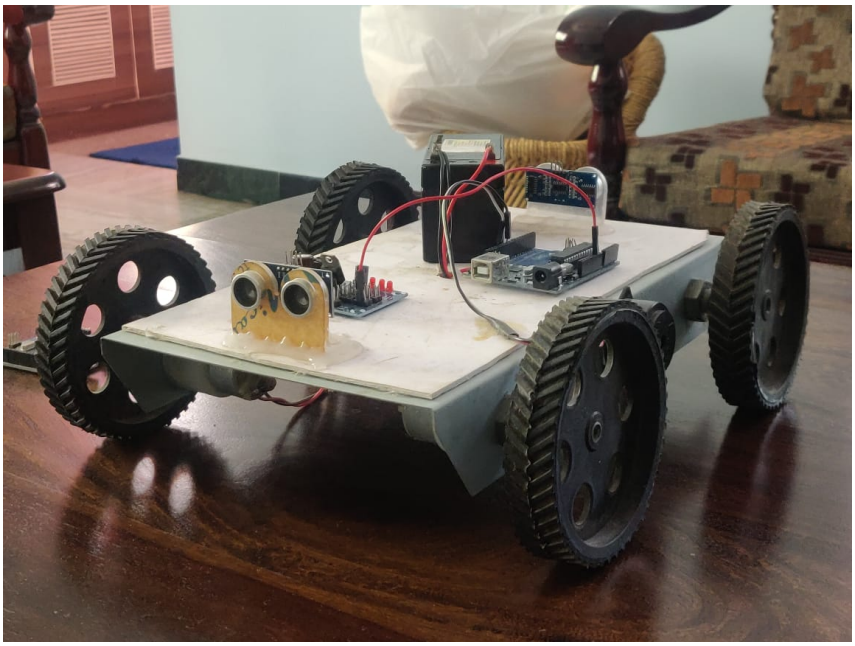

FIG 5.4 Prototype Isometric View

CHAPTER-6

DESIGN, SIMULATION AND HARDWARE

6.1. CONSTRUCTION

For the transmitter make the connections as follows-

1. Take a jumper wire and connect it at one of the end of the breadboard and the other end of the jumper wire to 5V pin of arduino board as shown in the figure.
2. Take another jumper wire and connect it at another end of the breadboard and the other end of the jumper wire to Gnd pin of the arduino board as shown in the figure.
3. Take another jumper wire and connect it's one end with the 5V hole of breadboard and other end to VCC pin of MPU6050.
4. Similarly, with another jumper wire to Gnd of breadboard and other end to Gnd of MPU6050.
5. Then connect SDA pin of MPU6050 to A4 pin of the Arduino and SCI pin of MPU6050 to A5 pin of the Arduino by the help of jumper wires.
6. Then take HC-05 Bluetooth module and connect it as follows-
7. Take a jumper wire and connect it's one end to VCC of breadboard and another end to VCC of the Bluetooth module.
8. Similarly take a jumper wire and connect it's one end to Gnd of Breadboard and another end to Gnd of Bluetooth module.
9. Now connect TX pin of Bluetooth module directly to pin D10 of Arduino.
10. Don't connect RX pin of Bluetooth module directly to any pin of arduino as Bluetooth module works on 3.3V level and arduino works on 5V level and hence

5V from arduino can burn the Bluetooth module. Hence to solve this problem we will make a voltage divider by the help of resistors. connect one end of 1000 ohm resistor to D11 pin of Arduino and other end to RX pin of Bluetooth module. Connect one end of 2000 ohm resistor to RX pin of Bluetooth module and other end to Gnd of breadboard.

Go to Thing speak website and make an account on this website.

Then follow these steps....

1. Go to my channels and create a new channel with any name and give any appropriate field name as you like....
2. Click on submit and save the channel.
3. Go to this channel and on API Keys field ,you can see write and read field keys. Copy the URL of the update channel feed on the right side of the screen.
4. Now click on the Apps option on the top of the screen and scroll down and click on the second last option i.e. Talk back option. This is the app we are going to use to feed data to this website.
5. Go to this app and click on New Talk Back for creating own app.
6. Edit the name of talk back and in the log to channel select your channel made in the previous steps.
7.Save your talk back app created.

Follow these steps-

1. Create a new project and name it.
2. On screen 1 , you will see an image of an android phone.
3. First click on Label on left hand side of the screen and drag it to the android screen.
4. Then click on List Picker from the left hand side and drag it on the screen and on the right hand side go on the text option and write connected there. This list will show all the devices waiting to be connected to the android phone.
5. Click on button on the left hand side and then drag on the screen in the text field write disconnected as when we will click on this button then device will be

disconnected from mobile.

6.Click on the Label on the left hand side and drag it on the screen. Then in the text field on the right side write Data.

7. Click on the Label on the left hand side and drag it on the screen. This is used to display the data received by the app.

8. Then in the connectivity option on the left hand side, click on the sub-option bluetooth client and drag on the screen.

9. Then from the same connectivity option ,click on the sub-option web and drag it on the screen.

10. Click on the sensor option in the left side of the screen and drag the suboption clock on the screen.

11. Click again on the sub-option clock and drag it on the screen.

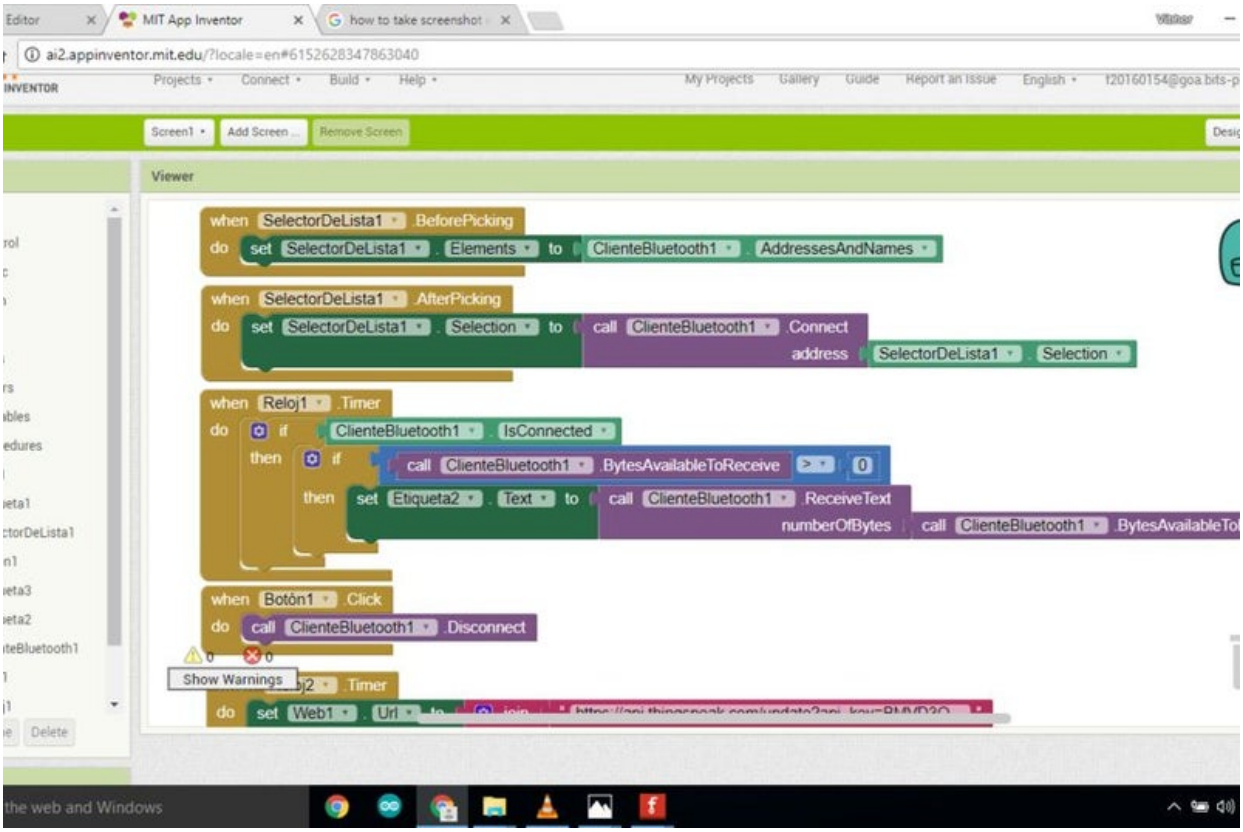

Fig 6.1 App Creation

Circuit for receiver part as shown in the figure is as follows-

First assemble your chassis and connect your motors in a appropriate way.

1. First connect battery to battery sniper and connect the red wire i.e. the VCC wire to one end of breadboard.

2. Similarly connect the other end of the wire to other end of breadboard.

3. Now take a jumper and connect it to VCC pin of NodeMCU and other end to VCC pin of breadboard.

4. Now take a jumper and connect it to Gnd pin of NodeMCU and other end to Gnd pin of breadboard.

5. Take your motor driving board and connect it on your chassis.

6. Take two jumper wires and connect their one end to VCC of breadboard and another to 9V pins of motor driving board.

7. Take two more jumper wires and connect their one end to Gnd of breadboard and another to Gnd of motor driving board.

8. Connect two wires of left motor to output pins of motor driving board.

9. Similarly connect two wires of right motor to output pins of motor driving board.

10. Connect four input pins on motor driving board to four digital pins of NodeMCU as shown in the figure.

11. Connect 5V pins on the motor driving board to Vout pin of NodeMCU.

For the transmitter part the code is in the file final_wire.h-

For the receiver part the code is in the file second_part_of_final_project-

For uploading code on NodeMCU through Arduino IDE, you have to follow these steps-

1. Firstly open the Arduino IDE.

2. Go to files at the top left hand corner of the screen and click on the preference in the drop down list in Arduino IDE.

3.Copy the below code in the Additional boards Manager http:// arduino.esp8266.com/stable/package_esp8266com_index.json

4. Click OK to close the preference Tab.

5. After completing the above steps , go to Tools and board, and then select board Manager.

6. Navigate to esp8266 by esp8266 community and install the software for Arduino. Once all the above process been completed we are ready to program our esp8266 with Arduino IDE.

6.2. WORKING MECHANISM

- This robot is a gesture controlled one which can be controlled from anywhere by the help of internet.
- In this project accelerometer MPU 6050 is used for collecting data regarding our hand movements.
- It measures the acceleration of our hand in three axes. This data is sent to Arduino which processes this data and decides where the robot should move.
- This data is sent to a Bluetooth module, which in turn sends this data to our android mobile which has an app made by us.
- The app has been made through MIT app inventor website. This app receives this data and sends this data to thing speak website. Thingspeak.com is a free IOT website which stores this data.
- On the receiver side, node mcu WI-FI module receives this data and then drives the motors through motor driving board connected to it.

6.3. CODE

realtime_gesturee_detection.py

```
import copy
import cv2
import numpy as np
from keras.models import load_model
from phue import Bridge
from soco import SoCo
```

```
import pygame
import time

# General Settings
prediction = ''
action = ''
score = 0
img_counter = 500



# pygame.event.wait()

class Volume(object):
    def __init__(self):
        self.level = .5

    def increase(self, amount):
        self.level += amount
        print(f'New level is: {self.level}')

    def decrease(self, amount):
        self.level -= amount
        print(f'New level is: {self.level}')



vol = Volume()

# Turn on/off the ability to save images, or control Philips Hue/Sonos
#save_images, selected_gesture = False, 'peace'
smart_home = True
```

```
# Philips Hue Settings
#bridge_ip = '192.168.0.103'
#b = Bridge(bridge_ip)
#on_command = {'transitiontime': 0, 'on': True, 'bri': 254}
#off_command = {'transitiontime': 0, 'on': False, 'bri': 254}

# Sonos Settings
#sonos_ip = '192.168.0.104'
#sonos = SoCo(sonos_ip)

gesture_names = {0: 'Fist',
                 1: 'L',
                 2: 'Okay',
                 3: 'Palm',
                 4: 'Peace'}

model = load_model('D:/gesture controlled iot project/project_kojak-master/
project_kojak-master/models/VGG_cross_validated.h5')


def predict_rgb_image(img):
    result = gesture_names[model.predict_classes(img)[0]]
    print(result)
    return (result)


def predict_rgb_image_vgg(image):
    image = np.array(image, dtype='float32')
    image /= 255
```

```
    pred_array = model.predict(image)
    print(f'pred_array: {pred_array}')
    result = gesture_names[np.argmax(pred_array)]
    print(f'Result: {result}')
    print(max(pred_array[0]))
    score = float("%0.2f" % (max(pred_array[0]) * 100))
    print(result)
    return result, score


# parameters
cap_region_x_begin = 0.5  # start point/total width
cap_region_y_end = 0.8  # start point/total width
threshold = 60  # binary threshold
blurValue = 41  # GaussianBlur parameter
bgSubThreshold = 50
learningRate = 0

# variableslt
isBgCaptured = 0  # bool, whether the background captured
triggerSwitch = False  # if true, keyboard simulator works


def remove_background(frame):
    fgmask = bgModel.apply(frame, learningRate=learningRate)
    kernel = np.ones((3, 3), np.uint8)
    fgmask = cv2.erode(fgmask, kernel, iterations=1)
    res = cv2.bitwise_and(frame, frame, mask=fgmask)
    return res
```

```
# Camera
camera = cv2.VideoCapture(0)
camera.set(10, 200)

while camera.isOpened():
    ret, frame = camera.read()
    frame = cv2.bilateralFilter(frame, 5, 50, 100)  # smoothing filter
    frame = cv2.flip(frame, 1)  # flip the frame horizontally
    cv2.rectangle(frame, (int(cap_region_x_begin * frame.shape[1]), 0),
                  (frame.shape[1], int(cap_region_y_end * frame.shape[0])), (255, 0, 0),
2)

    cv2.imshow('original', frame)

    # Run once background is captured
    if isBgCaptured == 1:
        print('hi')
        img = remove_background(frame)
        img = img[0:int(cap_region_y_end * frame.shape[0]),
              int(cap_region_x_begin * frame.shape[1]):frame.shape[1]]  # clip the
ROI
        #cv2.imshow('mask', img)

        # convert the image into binary image
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        blur = cv2.GaussianBlur(gray, (blurValue, blurValue), 0)
        # cv2.imshow('blur', blur)
        ret, thresh = cv2.threshold(blur, threshold, 255, cv2.THRESH_BINARY +
cv2.THRESH_OTSU)
```

```
        # Add prediction and action text to thresholded image
        # cv2.putText(thresh, f"Prediction: {prediction} ({score}%)", (50, 50),
cv2.FONT_HERSHEY_SIMPLEX, 1, (255,255,255))
        # cv2.putText(thresh, f"Action: {action}", (50, 100),
cv2.FONT_HERSHEY_SIMPLEX, 1, (255,255,255))  # Draw the text
        # Draw the text
        cv2.putText(thresh, f"Prediction: {prediction} ({score}%)", (50, 30),
cv2.FONT_HERSHEY_SIMPLEX, 1,
                    (255, 255, 255))
        cv2.putText(thresh, f"Action: {action}", (50, 80),
cv2.FONT_HERSHEY_SIMPLEX, 1,
                    (255, 255, 255))  # Draw the text
        cv2.imshow('ori', thresh)

        # get the contours
        thresh1 = copy.deepcopy(thresh)
        contours, hierarchy = cv2.findContours(thresh1, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
        length = len(contours)
        maxArea = -1
        if length > 0:
            for i in range(length):  # find the biggest contour (according to area)
                temp = contours[i]
                area = cv2.contourArea(temp)
                if area > maxArea:
                    maxArea = area
                    ci = i

            res = contours[ci]
            hull = cv2.convexHull(res)
```

```
            drawing = np.zeros(img.shape, np.uint8)
            cv2.drawContours(drawing, [res], 0, (0, 255, 0), 2)
            cv2.drawContours(drawing, [hull], 0, (0, 0, 255), 3)

        cv2.imshow('output', drawing)

    # Keyboard OP
    k = cv2.waitKey(10)
    if k == 27:  # press ESC to exit all windows at any time
        break
    elif k == ord('b'):  # press 'b' to capture the background
        bgModel = cv2.createBackgroundSubtractorMOG2(0, bgSubThreshold)
        #b.set_light(6, on_command)
        time.sleep(2)
        isBgCaptured = 1
        print('Background captured')
        #pygame.init()
        #pygame.mixer.init()
        #pygame.mixer.music.load('/Users/brenner/1-05 Virtual Insanity.mp3')
        #pygame.mixer.music.set_volume(vol.level)
        #pygame.mixer.music.play()
        #pygame.mixer.music.set_pos(50)
        #pygame.mixer.music.pause()

    elif k == ord('r'):  # press 'r' to reset the background
        time.sleep(1)
        bgModel = None
        triggerSwitch = False
        isBgCaptured = 0
        print('Reset background')
```

```
elif k == 32:
    # If space bar pressed
    cv2.imshow('original', frame)
    # copies 1 channel BW image to all 3 RGB channels
    target = np.stack((thresh,) * 3, axis=-1)
    target = cv2.resize(target, (224, 224))
    target = target.reshape(1, 224, 224, 3)
    prediction, score = predict_rgb_image_vgg(target)

    if smart_home:
        if prediction == 'Palm':
            try:
                action = "Lights on, music on"


                # sonos.play()
                #pygame.mixer.music.unpause()
            # Turn off smart home actions if devices are not responding
            except ConnectionError:
                smart_home = False
                pass

        elif prediction == 'Fist':
            try:
                action = 'Lights off, music off'
                #b.set_light(6, off_command)
                # sonos.pause()
                #pygame.mixer.music.pause()
            except ConnectionError:
                smart_home = False
                pass
```

```
elif prediction == 'L':
    try:
        action = 'Volume down'
        # sonos.volume -= 15
        #vol.decrease(0.2)
        #pygame.mixer.music.set_volume(vol.level)
    except ConnectionError:
        smart_home = False
        pass

elif prediction == 'Okay':
    try:
        action = 'Volume up'
        # sonos.volume += 15
        #vol.increase(0.2)
        #pygame.mixer.music.set_volume(vol.level)
    except ConnectionError:
        smart_home = False
        pass

elif prediction == 'Peace':
    try:
        action = 'peace'
    except ConnectionError:
        smart_home = False
        pass

else:
    pass
```

```
        if save_images:
            img_name = f"./frames/drawings/drawing_{selected_gesture}
_{img_counter}.jpg".format(
                img_counter)
            cv2.imwrite(img_name, drawing)
            print("{} written".format(img_name))

            img_name2 = f"./frames/silhouettes/{selected_gesture}
_{img_counter}.jpg".format(
                img_counter)
            cv2.imwrite(img_name2, thresh)
            print("{} written".format(img_name2))

            img_name3 = f"./frames/masks/mask_{selected_gesture}
_{img_counter}.jpg".format(
                img_counter)
            cv2.imwrite(img_name3, img)
            print("{} written".format(img_name3))

            img_counter += 1

    elif k == ord('t'):

        print('Tracker turned on.')

        cap = cv2.VideoCapture(0)
        ret, frame = cap.read()

        # Select Region of Interest (ROI)
```

```
        r = cv2.selectROI(frame)

        # Crop image
        imCrop = frame[int(r[1]):int(r[1] + r[3]), int(r[0]):int(r[0] + r[2])]

        # setup initial location of window
        r, h, c, w = 250, 400, 400, 400
        track_window = (c, r, w, h)
        # set up the ROI for tracking
        roi = imCrop
        hsv_roi = cv2.cvtColor(roi, cv2.COLOR_BGR2HSV)
        mask = cv2.inRange(hsv_roi, np.array((0., 60., 32.)), np.array((180., 255.,
255.)))
        roi_hist = cv2.calcHist([hsv_roi], [0], mask, [180], [0, 180])
        cv2.normalize(roi_hist, roi_hist, 0, 255, cv2.NORM_MINMAX)
        # Setup the termination criteria, either 10 iteration or move by at least 1 pt
        term_crit = (cv2.TERM_CRITERIA_EPS |
cv2.TERM_CRITERIA_COUNT, 10, 1)
        while (1):
            ret, frame = cap.read()
            if ret == True:
                hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
                dst = cv2.calcBackProject([hsv], [0], roi_hist, [0, 180], 1)
                # apply meanshift to get the new location
                ret, track_window = cv2.CamShift(dst, track_window, term_crit)
                # Draw it on image
                pts = cv2.boxPoints(ret)
                pts = np.int0(pts)
                img2 = cv2.polylines(frame, [pts], True, (0, 255, 0), 2)
                cv2.imshow('img2', img2)
```

```
        k = cv2.waitKey(60) & 0xff
        if k == 27:  # if ESC key
            break
        else:
            cv2.imwrite(chr(k) + ".jpg", img2)
    else:
        break
cv2.destroyAllWindows()
cap.release()
```

CHAPTER-7

RESULTS

The Prototype of automated car parking system and motion control using gestures was successfully built to achieve easier access to the car, to move in and out of the garage. The idea has been achieved by an obsolete method of using an accelerometer sensor to read the position of the hand and accordingly move the car to the desired position, over the years . We have considered the more efficient and accessible way of gesture recognition by methods of bleeding edge technology such as image processing and machine learning, and with the help of

IOT and app development we were able to receive the data to our phones and other electronic devices as well.

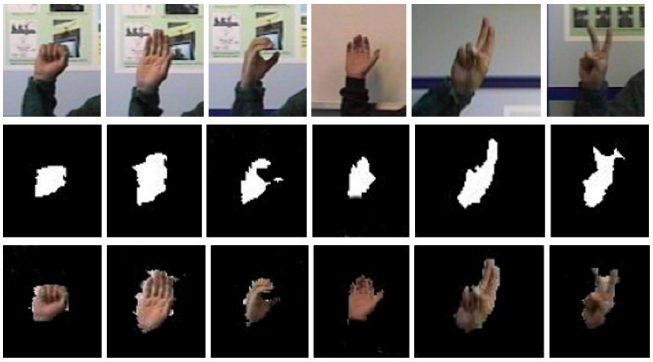

Fig 7.1 Hand Gesture Recognition

By implement such gesture detection methods has enabled us to have a hands free control of the car and thus, has become more reliable than the remote access as it requires less hardware and mostly revolves around the software built under the basis of machine learning and image recognition.

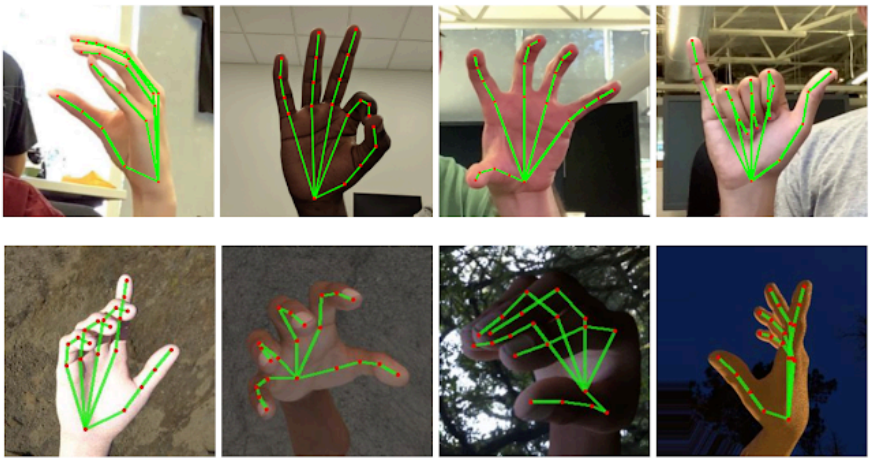

Fig 7.2 Gesture Mapping using machine learning

This use of technological advances is beneficiary to the consumer and the company to monitor the car. By implementing the obstruction detecting sensors

around the car, we can achieve more reliable gesture based control to the car, and prevent any accidents from occurring.

The advent of IoT in automotive industry has opened new avenues for carmakers and buyers all across the world. With usage at both industrial and commercial level, IoT in automotive sector has become a prominent hotspot for variegated multi-purpose applications. From connected cars to automated transport systems, the applications of Internet of Things have made a deep dent in the global automotive market.

Fig 7.3 Internet Of Things

Fig 7.4 IOT Cloud Transmission

CHAPTER-8

CONCLUSION

We have successfully built an gesture controlled automatic car parking which is more robust and efficient way to make a car enter and exit the garage. Initially, this way of automatic car parking was only possible by the use of remote or parking assist button in the car, which made it compulsory for the person to stay inside the car to access the automatic feature, which is now seems to be an obsolete method. Our approach is that the owner can be anywhere in front of the car or at the back or even within the car, and wave gestures, which is a more efficient way of controlling the cars movement.

This has enhanced the way to communicate to our car and helped to better access the cars automatic parking assist. Obstruction sensors around the car prevents the car from scratches or accidental damages, while moving into the garage. All the communications is achieved by Internet Of Things which has allowed us to have a seamless transmission of data across devices.

By avoiding the usage of accelerometer sensor, people are more free to move their hands and transmit information to the car. Remote access to the car have only limited movements, and hands free control can allow the car to move with wide degree of motions which is the future of electric cars. The car can be controlled within and outside the car, which has helped in more options to the owner to control the cars movements.

In future, usage of more efficient models of gesture recognition, can help in faster transmission of commands to the car and also enhances the reliability on the tech

CHAPTER-9

REFERENCE

1. S.Suman, S.Shrivatsa- Motion and system enabling control of different digital devices using gesture or motion control- 2018- Google Patents
2. J Pirker, M Pojer, A Holzinger, C Gütl- Gesture based interactions in video games using leap motion controller- 2017- International Conference on human

3. HY Lai- Hand gesture recognition system for controlling electronically controlled devices- 2019- Google Patents
4. S Song, D Yan, Y Xie – Design of control system based on hand gesture recognition- 2018- IEEE
5. A Jackowski, M Gebhard – Head motion and head gesture based robot control – 2017- IEEE
6. A Sarkar, KA Patel, RKG Ram- Gesture control of a drone using a motion controller- 2916- IEEE
7. G Strutt, J Bjørne, G Birkedal- Gesture control- 2019- Google Patents
8. W Luchner, J Pennock, E Veit- Gesture and motion based control of user interfaces- 2019- Google Patents
9. S Surve, A Nair, A Singh, V Krishna – Iot based gesture control gaming-2018- Academia.edu
10. W Zhi-heng, C Jiang-tao, L Jin-guo – Design of human computer interaction control system based on hand gesture recognition- 2017-IEEE