

Smart Wearable Gadget to Detect Stairs, Potholes, and Other Objects for Visually Impaired Users Using Computer Vision

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering degree in Mechatronics Engineering

by

Aman Dinodiya (38090002)

Raghavendra. R (38090017)



**DEPARTMENT OF MECHATRONICS ENGINEERING
SCHOOL OF MECHANICAL ENGINEERING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with Grade "A" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

MAY - 2022



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF MECHATRONICS ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of Aman Dinodiya and Raghavendra. R who carried out the project entitled "Smart Wearable Gadget to Detect Stairs and Objects for Visually Impaired Using Computer Vision" under my supervision from 01.11.2021 to 10.05.2022.

Internal Guide

Dr. S. PRAKASH, M.E., Ph.D.,

Head of the Department

Dr. S. PRAKASH, M.E., Ph.D.,

DEAN
SCHOOL OF MECHANICAL ENGINEERING
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Jeppiyaar Nagar, Rajiv Gandhi Salai,
Chennai - 600 119

Submitted for Viva-voce Examination held on 10.05.2022

Internal Examiner

External Examiner

DECLARATION

I Aman Dinodiya (38090002) and Raghavendra. R (38090017) hereby declare that the Project Report entitled "Smart Wearable Gadget to Detect Stairs and Objects for Visually Impaired Using Computer Vision" done by us under the guidance of Dr. S. PRAKASH, M.E., Ph.D., is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Mechatronics Engineering.

1)



2)



DATE: 10.05.2022

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to the Board of Management of SATHYABAMA for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my sincere thanks to Dr. S. Prakash, M.E., Ph.D., Head of the Department, Dept. of Mechatronics Engineering and Dean of School of Mechanical Engineering, for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide Dr. S. Prakash, M.E., Ph.D., for his valuable guidance, suggestions, and constant encouragement that paved the way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the Department of Mechatronics Engineering who were helpful in many ways for the completion of the project.

I would like to thank my senior Rahul for helping me to fabricate the project on time. I would like to thank my parents, well-wishers, and friends who were very supportive throughout the tenure of my project phase.

TABLES OF CONTENT

Chapter No.	Title	Page No.
	ABSTRACT	iii
	LIST OF FIGURES	iv
	LIST OF TABLES	v
	LIST OF ABBREVIATIONS	vi
1.	INTRODUCTION	1
1.1	INSPIRATION	1
1.2	PROPOSED MODEL	2
1.3	BLOCK DIAGRAM	2
1.4	FUTURE SCOPE	3
2.	LITERATURE SURVEY	4
2.1	INTRODUCTION	4
2.2	LITERATURE SURVEY	4
3.	AIM AND SCOPE OF THE PRESENT INVESTIGATIONS	9
3.1	AIM OF THE PRESENT INVESTIGATION	9
3.2	SCOPE OF THE PRESENT INVESTIGATION	9

4.	EXPERIMENTAL OR MATERIALS AND METHODS; ALGORITHMS USED	10
4.1	COMPONENTS	10
	4.1.1 Raspberry Pi	10
	4.1.2 Ultrasonic Sensor	13
	4.1.3 Buzzer	15
	4.1.4 Other Components	15
4.2	DESIGN THEORY	17
4.3	ALGORITHM	20
	4.3.1 YOLO Pretrained model	21
	4.3.2 YOLO Custom model	26
5.	RESULTS AND DISCUSSION, PERFORMANCE ANALYSIS	37
5.1	EXPERIMENT METHODOLOGY	37
5.2	CUSTOM MODEL ANALYSIS	42
5.2	PRE-TRAINED MODEL ANALYSIS	42
6.	CONCLUSION	46
6.1	SUMMARY	46
6.2	STATIC AND DYNAMIC CHARACTERISTICS	46
6.3	BUDGET REPORT	47
	REFERENCES	48

ABSTRACT

The sense of sight is one of the most vital aspects of life. People with partial and complete visual impairment encounter a lot of challenges in the day-to-day environment. The world is approaching development and stability expeditiously and it has become so tough to keep up with someone who lacks the basic senses. Visually impaired people are part of this pursuit of development as well, so the advancements must have a crucial effect on their lives to make what was impractical for them practical and accessible as well. The formal comfort provided earlier for visually impaired people was as a specific hardware device such as Optical Character Recognition, identifying color, barcode readers, which had a limited reach among people and were not at all economical. A wearable gadget to provide directional navigational information to visually impaired people and alert them about the environment whenever the sensors notice jeopardy in the surroundings. This gadget has a lot of prospects to guide and navigate smartly to the user. The implementation results in enhancement and understanding of the problems faced by a blind person, and it will help aware people to achieve more projects targeted to help blind people.

LIST OF FIGURES

Figure No.	Title	Page No.
1.1	Block Diagram representing the step-by-step process	2
4.1	Raspberry Pi 4 Model B	10
4.2	Pin Configuration of a Raspberry Pi 4 Model B	11
4.3	An Ultrasonic Sensor	14
4.4	A Buzzer	15
4.5	Jumper wires	16
4.6.	Breadboard and PiCam	16
4.7	Design of the Enclosure (4.7a, 4.7b, and 4.7c)	18
4.8	Hardware Design	19
4.9	Workflow of the Custom model: Block Diagram	35
5.0	Prototype: Circuit Assembly	36
5.1	Rendering Time vs, No of Observations	38
5.2	Detection Rate vs No of Observations	38
5.3	Du, Dm vs No of Observations	39
5.4	Model detecting (i)Motorbike, (ii)a Person, (iii)Staircase, (iv) Clock, (v)Cell Phone, (vi) computer mouse, and (vii) TV remote	44
5.5	Real-time detection- Experimenting with a subject in real-time	45

LIST OF TABLES

Table No.	Title	Page No.
5.1	Observation Table 1	40
5.2	Observation Table 2	41
6.1	Budget Report	47

LIST OF ABBREVIATIONS

IFTTT	- If This Then That
LIDAR	- Light Detection and Ranging
YOLO	- You Only Look Once
CNN	- Convolutional Neural Networks
COCO	- Common Objects and Context
NCBI	- National Center for Biotechnology Information
GSM	- Global System for Mobile Communication
USB	- Universal Serial Bus
GPIO	- General Purpose Input Output

CHAPTER – 1

INTRODUCTION

1.1 INSPIRATION:

As per WHO dataset reports, 7.7 billion people live on Earth, about 2.2 billion are examined to have near or distance vision impairment and more than 1 billion people have moderate Visual Impairments. The number of blind people raised by 43% over the previous 20 years. Since human vision plays a very significant part when it comes to interacting with the surroundings, In case it is not functioning well, then the person should be dependent on others forever. They encounter various struggles both indoors and outdoors, irrespective of whether the activity they perform is simple or complex, they are going through a lot of difficulties. Currently, largely visually impaired and blind people rely upon cane sticks to ensure hurdle-free movements. World Health Organization(WHO) states that the major factors of visual impairment are:

- Diabetic-retinopathy
- Trachoma
- Uncorrected refractive errors
- Glaucoma
- Corneal opacity
- Cataract
- Age-related macular degeneration

People can always use technologies to assist them to create new solutions for visually impaired people to make their lives easier. Many people have invented devices that primarily identify and warn the user of an obstacle in front of them using distance sensors, however systems that specify the actual object present are fairly limited and provide great accuracy and precision with object detection and alerting the user. With the use of numerous techniques such as You Only Look Once(YOLO), Tensorflow, Single Shot Detection(SSD), Convolutional Neural Networks(CNN), and others, the notion of Computer Vision helps to eliminate the issue of identifying objects. Though these frameworks are designed to interact with

the COCO Dataset's unique items, there are several other objects that humans encounter on a daily basis.

1.2 PROPOSED MODEL:

The prototype of the project serves the purpose of generally drawing the attention of the usage of such wearable devices throughout the globe with ease.

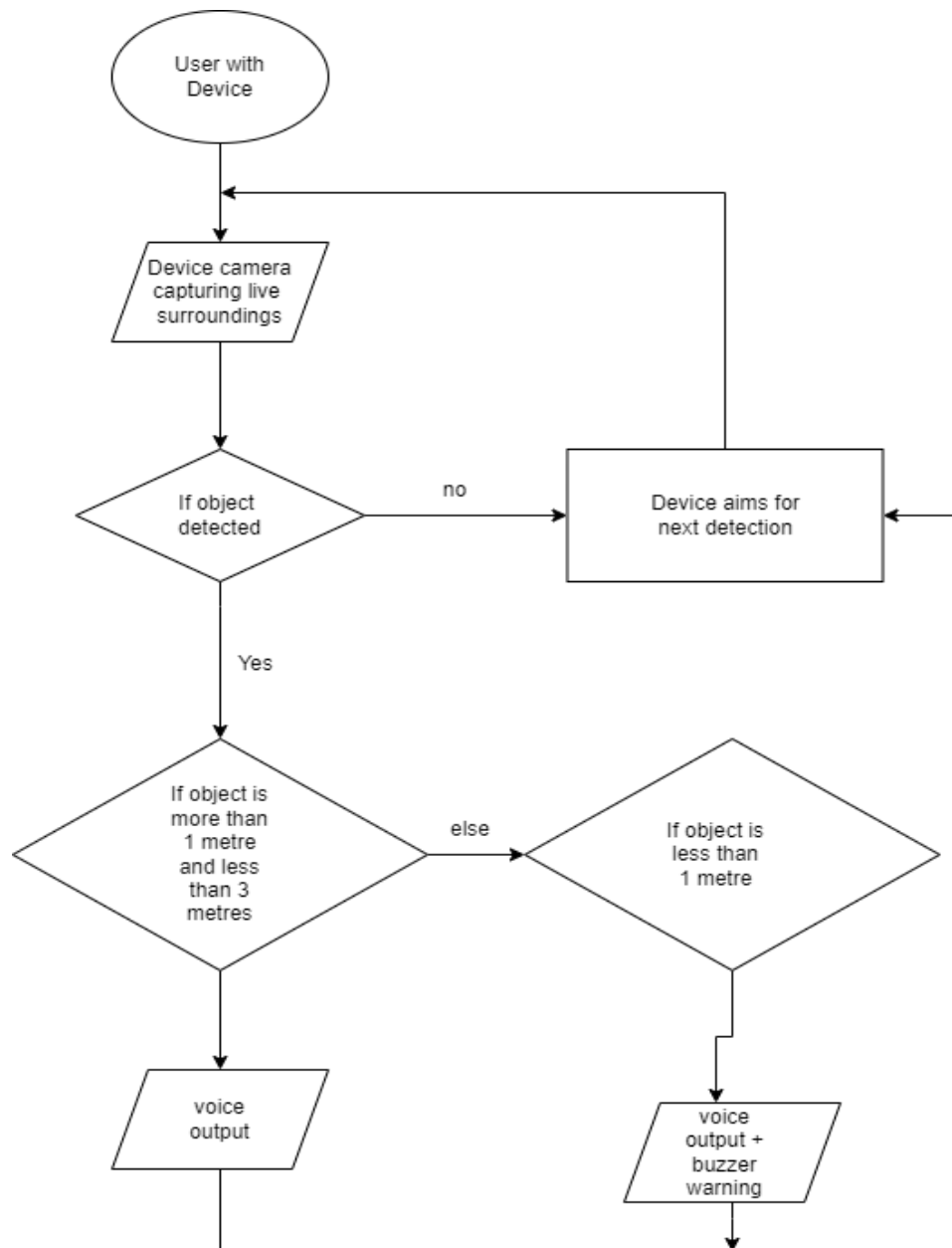


Fig.1.1: Block Diagram representing the step-by-step process of object detection

The block diagram shown below explains the whole research methodology being used in this prototype. The model represents a rough step-by-step demonstration of the process which was followed.

1.3 FUTURE SCOPE:

The conventional system is already being used by many users, most of the users are relying on the Cane Sticks in object detection. Further, some commercialized products are available that detect obstacles and alert the user. The further implementation of the project will be done in an overall cost-effective way and on a large scale to make the product reliable and effective for all the users. The future prototypes will be more accurate and economical, as every user can afford this assistant wearable gadget. The upcoming projects will be more compact and comfortable for the user to wear. We are planning to introduce some extra sensors along with the already existing ones in this model to provide more accurate and reliable results.

One aspect of future planning is to collaborate this device with an earbud or an earphone, to become more accurate in terms of alerting the user. This concept will enhance the output by providing a clean and clear sound directly to the ear of the user who is wearing the earphones/earbud, which is connected to our main controller. So, the feedback from the program on detecting an object, with a specific type of the object will be received by the control system and then, it will be processed and sent as output, directly to the user's earphones. Another future aspect of this project will be the implementation of LIDAR sensors along with the already existing ones. This will help us in doing 3D mapping to get the coordinates of the object. That object's coordinates will be compared to the existing data which we will gather from Computer Vision and Ultrasonic sensor will be compared and that final data will be processed by the control system and thus we will generate the final output, which will be comparatively far more accurate.

CHAPTER - 2

LITERATURE SURVEY

2.1 INTRODUCTION

A review of designing a wearable system that can detect objects was accomplished. In general, operations are carried out by constructing an algorithm, which is then integrated with various sensors and a sensory feedback system, which is either voice or vibration-based.

2.2 LITERATURE SURVEY

1. There were many previous works done before in the same area of research. **Usman Masud., Tareq Saeed et al., (2022)** proposed a system that will initiate a buzzer alarm whenever the user gets near an object. The implementation was on a cane and the framework used for object detection was Tensorflow. We came to know that Viola Jones algorithm and Haar filters, which are object detection frameworks and those gave significant results to their work. Upon testing the system, they concluded that their system gives an overall efficiency of 91%.
2. **Pradeep Kumar M, Inchara KM, Lekhashree S, Likhith C N, and Pavan U, presents their work on “Real-Time Assistive Shoes for Visually Impaired”(2021)**, led us to the first step of understanding the integration of google maps to the system as they integrate google maps using GPS sensor and Bluetooth module for communication. Their work also introduced us to a new microcontroller called Renesas, with which we made further research on using in our project.
3. **Gouthami V, Dixitha Kasturi, Ramalakshmi (2020)**, work on Object detection for the visually impaired, we got furthermore introduction to the COCO dataset and the YOLO detection model at the initial stage of our project. Their detection model was also involved with distance calculation using an ultrasonic sensor. They have stated that their model detects

objects at 0.6 frames per second in raspberry pi and that is comparatively slower than a system CPU.

4. **Roy ABI ZEID DAOU, Jeffrey CHEHADE, Ali HAYEK, Josef BOERSOEK, and Jose Javier Serrano OLMEDO (2020)** In their study designed a smart shoe system for secure movements of the user. They used a water detection sensor to monitor moisture on the floors, and a GSM module to notify the caretakers of the user in case of emergency. They used an Arduino mega microcontroller and its libraries to give voice outputs via a speaker. They have stated that their prototype achieves a success rate of around 95% and gives a false detection rate of 5%.
5. **Tahoun, N., Awad, A., & Bonny, T. (2019)** designed a smart system using Raspberry Pi and MobilenetSSD which is an object detection framework similar to YOLO. Their prototype detects 30 different objects successfully. The tests performed were fully focused to find the device's limitations. They have found that their algorithm couldn't recognize objects when subjected to a sudden change. Also, their prototype only detects the objects which are in front of the user(linear detection).
6. **Prasad Ramachandra Mavarkar, and Zarinabegam K Mundargi., (2019)** modeled a smart stick for blind people using YOLO. Their model seems to detect the objects which are present in the COCO dataset and intimates the user with voice output. They used an ultrasonic sensor that provides the distance from the user to the object.
7. **Aylwin Chai Bing Chun, Lau Bee Theng, Almon Chai WeiYen, Lil Deverell, Abdullah AL Mahmud, and Christopher McCarthy (2019)** designed a specific tool that helps the user detect ground-level hazards like stairs, ramps, etc. Their theory of staircase detection gave us a clear understanding and also we tried to modify that concept to the next level in our project. Their concept of placing ultrasonic sensors at some reference points and continuously monitoring the changes in the ground made us think of joining ultrasonic sensors along with the detection algorithm for

staircase, pits, and pothole detection. They experimented with their algorithm on staircases, ramps, potholes, and drainages. The average success rate was around 90% and the false rate was around 8%.

8. **Juan Du (2018)** review Understanding object detection based on CNN and YOLO gave us a clear understanding of Convolutional Neural Networks and their layers such as the Input layer, Convolutional(Conv.) layer, pooling layer, fully connected layer, and some other optional layers. We understood how the Conv layer works with the data provided by the input layer, as a result, it generates a shared weight with which the neural cells will develop a recognition algorithm and perform detections whenever it finds an object similar to the data provided.
9. **Dimitris E. Diamantis, Dimitris K. Iakovidis, Panagiotis Kalozoumis Dimitris E. Diamantis, and, George Dimas July 2019**, proposed a work that incorporates a stereoscopic camera that was mounted on an adjustable wearable frame-like structure. A raspberry PI zero W was used along with a Bluetooth module which was attached to the frame, The object detection module's creative approach employs fuzzy sets and GANs to simulate human eye fixation, causing the system to execute efficient real-time object detection with improved efficiency than current framework approaches.
10. **Huiliang Cao, Chong Shen, Shiwei Chen, and Dayue Yao** approached a smart wearable system in July 2019 which was based on a cloud server for image processing. The smart wearable consists of an IR sensor, an ultrasonic sensor, and a camera to scan. The recognition of faces is achieved along with the objects and texts using the model and algorithm they formed.
11. **Adrian Burlacu, Simona Caraiman, and Paul Herghelegiu in the year of 2017**, presented a unique algorithm to detect objects in negative format and along with tracking them. The algorithm was integrated along with a smart wearable device that uses stereo correspondence and ground place estimation to form the algorithm.

12. **B.K.T.P Wickramasinghe, K.K.V.S Anjana, G.D.S.T Ratnasekara, and D.U Vidanagama(2017)**, created a system using Arduino UNO microcontroller and Infrared sensors on a walking cane. This system is monitored via a mobile application created using MIT App Inventor. The IR sensors check if the user gets near any obstacle and if he/she does, give warnings accordingly. Their approach includes creating an IOT model as well using Arduino Mega and Bluetooth modules. Outputs are given via voice and they use google maps for navigation.
13. **Hotaka Takizawa and Mayumi Aoyagi 2016**, proposed a system to assist visually impaired users using image processing. They developed a product using a Kinect sensor on a cane stick to assist the user to detect obstacles like elevators, floors, walls, etc, and provide sensory feedback by vibrating the cane stick. Along with this, a Kinect goggle was designed as an alternative to the smart cane stick. The system was capable of detecting and recognizing obstacles and a smartphone-based light checking system informed the user about the On or OFF state of the room light or elevator's button lights.
14. **Ashish Patankar, Tooraj Nikoubin developed a Wearable System in 2016** which operates with three phases to detect obstacles, which include signal generation, transmitting and receiving the signal which was done using an ultrasonic sensor array, which was followed by a signal processing phase, and a device called myRio and a LabVIEW code, which detects the time of flight from the echo received by the LabVIEW platform. The time of flight was later used to detect the accurate distance between the sensor and the obstacle.
15. **Thi-Lan Le, Nicolas Vuillerme, and Huy-Hieu Pham** came up with the concept of detecting the obstacle using a Kinect device in 2015, ultimately by using RGB lights to detect the depth of the object. The acquisition data would involve the color and the depth, which after reconstruction along with filtration and segmentation, can be used as obstacle detection. They

developed the algorithm in C++ and worked with Point cloud Library(PCL), and along with that they externally worked with OpenCV, VTK, Qhull, etc.

16. **Kailun Yang, Ruiqi Cheng, Xiangdong Zhao, and Kaiwei Wang**, The method relies on data from a depth camera and data from the attitude angle. The basic component of the algorithm is a seeded region growing technique which is aided by photo edges and region boundaries.
17. **Paul Rohan, Garg Ankush, Singh Vaibhav, Mehra Dheeraj, Balakrishnan M. Paul Kolin, Manocha Dipendra 2007**, developed systems for people with visual impairments using an ultrasound-based cane-mounted knee-above obstacle detection and emergency management, and an RF-based user-triggered bus identification and localization system.
18. **Mauro Serpelloni, Alessandro Dionisi, and Emilio Sardini 2012**, coined a portable device intended to facilitate blind users in searching for objects over short distances using RFID technology. The device is portable, and can be easily inserted within a glove; the antenna, in particular, must be in the hand of the person searching. The device could provide data embedded in the device to the blind. The RSSI value and the scanned tags were linked to an indication of biofeedback.

CHAPTER - 3

AIM AND SCOPE OF THE PRESENT INVESTIGATION

3.1. AIM OF THE PRESENT INVESTIGATION

We aimed to build a solution that can assist a visually impaired person to do his/her routine tasks with more ease and comfort. Our aim includes forming an algorithm along with a designed assembly in a wearable form to provide precise results by detecting an object and alerting the user to make necessary decisions at that specific moment. Our motive was that the user need not be dependent on others in finding obstacles. Our scope is to define and achieve a final prototype of the assembly which is inculcated with our custom machine learning model to detect specifically, stairs and potholes, along with some random distinct objects within our expected and desired budget and resources. We believe that the system we developed will surely provide significant help to the users and satisfy their expected aspects.

3.2. SCOPE OF THE PRESENT INVESTIGATION

The project contributes to wearable technology for the visually impaired, the comfortability factor makes it an adaptable device for the users. The future investigational aspect of this project will make sure that the accuracy of the algorithm framed will remain above 90 %. Along with that, It will be more focused on comparatively quick response and accurate detection of objects, which will be achieved by considering the 3D Mapping technology using a LIDAR sensor, and thus comparing the output with the Machine learning's output to gather false or error detection, which in future can be eliminated. Another future of the project focuses on the up-gradation of the sensory feedback system, the future model will consist of a multilingual feedback system, to make the whole idea of making it economical and easy to use, would stand. This development will be beneficial for the local users who have difficulty speaking and understanding the English language.

CHAPTER - 4

EXPERIMENTAL OR MATERIALS AND METHODS; ALGORITHMS USED

The assembly we've created requires various components which are mentioned below:

4.1. COMPONENTS

4.1.1 Raspberry Pi 4 Model B:

In this project, we have a Raspberry Pi which is powered by a 5V battery, and all other sensors or components are powered by the Raspberry pi. The Raspberry Pi is a credit card-sized computer that can be programmed using various languages. It is not intended for use as a standalone computer, but rather as an inexpensive platform to experiment with electronics and programming. The Raspberry Pi Foundation states that the board can provide the same capabilities as more expensive microcontrollers, such as Arduino and BeagleBone Black, with more computational speed than its predecessors.



Fig. 4.1: Raspberry Pi 4 Model B

The Raspberry Pi is a compact-sized computer, which allows users to perform almost the same sort of tasks that can be performed on a usual personal computer. It runs Linux, but it also provides a set of GPIO (general purpose input/output) pins, which allows us to control electronic components for physical

computing. This board was designed to teach basic computer science in schools and developing countries. Nowadays, It is being widely used by students, especially in the Robotics domain, that it has become a great medium for students to perform the task of a personal computer in a more economic way. We have got GPIO pins which is the acronym for General Purpose Input-Output Pins, we have 40 GPIO pins i.e. 20 pins in each column. At the center, we have a Quad-Core 64-bit Broadcom 2711, Cortex A72 Processor with 4 GB Ram. In the corners, we have a micro- USB slot, which could be used to connect the power supply whereas the other got the HDMI port which is used to connect to the projector or screen. Contrarily. We have 4 USB ports and an Ethernet port at the other ends. An Ethernet port would be required to connect to internet connectivity however Raspberry Pi has built-in WiFi as well as Bluetooth connectivity, along with that on the backside we have a micro-SD card slot, which is used to load the operating system.

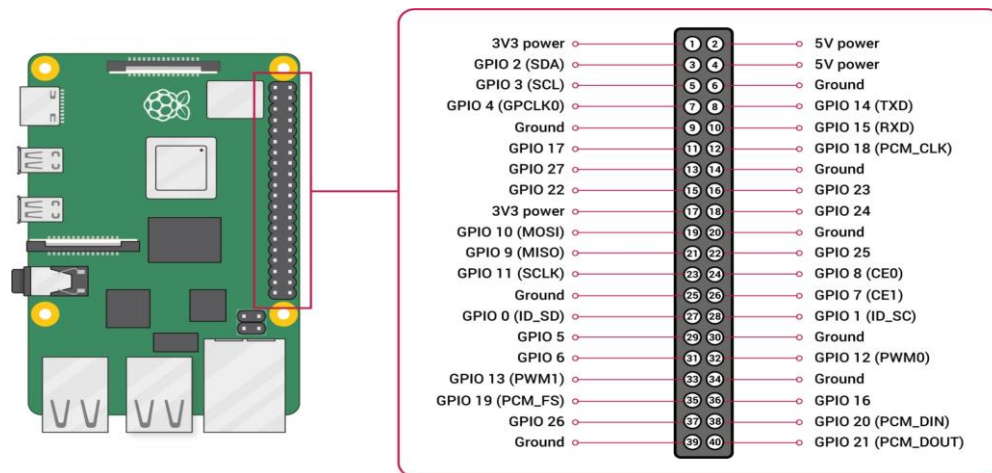


Fig.4.2: Pin configuration of a Raspberry Pi 4, Model B

Among the 40 GPIO pins, we have four power pins on the Raspberry Pi, two of which are 5v pins and another two are 3.3v pins. The 5v power pins are connected directly to the Raspberry Pi's power input and we can use these pins to run low power applications. The above-mentioned Raspberry Pi has an operating Power of 5V@3A via a USB Type-C port.

4.1.1.1 Power:

A Raspberry Pi can be powered via a USB connection or with an external power supply. Previous models like model 3 used the Type-B USB cable for the power supply but the latest raspberry pi 4 models use Type-C cables. The recommended input voltage is 5V, and the recommended input current is 2A. To make sure the device is turned on, there will be a red LED which is the power(PWR) LED that will glow if the device gets enough power supply. It also has another green LED which is the activity(ACT) which will glow if the SD card and the programs are properly functioning.

4.1.1.2 Memory:

Since the Raspberry pi has no internal storage, the memory is dependent upon the SD card we use. We have used a 16 GB SD card for this project so that after flashing and loading the operating system, we will get around 11 GB to work with.

4.1.1.3 Input And Output:

Raspberry Pi has 40 pins to perform various operations and those are done by General Purpose Input Output(GPIO). GPIO is a commonly used Input=Output system in embedded devices. In raspberry, there are 26 GPIO pins, 4 power pins(2 * 3.3V and 2 * 5V), 8 ground pins and the rest 2 are reserved pins used for advanced operations. It has 4 USB ports, of which two are USB 2.0 and the other two ports are USB 3.0 ports. Also, it has an Ethernet port, two micro HDMI ports, and a power port. To attach the SD card, there is an SD card slot on the backside of the pi. It also has a 3.5 mm audio jack which is used to connect audio output devices like headphones and speakers etc. Lastly, it has an LCD port where we can connect an LCD screen.

4.1.1.4 Programming:

In this project, we have used Python Programming language and by default, Raspberry Pi has Python language pre-installed. It has some default python editors like Thonny and Geany. We can also install commonly used IDEs like PyCharm using the command prompt. To install any tools or software we want, we can use the command prompt of the raspberry pi computer.

We can install the packages by using the sudo commands.

```
sudo apt-get install <package_name>
```

A common practice that is good to be followed is to update and upgrade the pi computer before installing any required package. This can be done by using the following commands.

```
sudo apt-get update
```

```
sudo apt-get upgrade
```

We can configure our raspberry pi by using the following line of code

```
sudo raspi-config
```

4.1.2 Ultrasonic Sensor:

Ultrasonic Sensors are used for distance calculation and are hence also known as Distance sensors. This HC-SR04 ultrasonic sensor has four pins. VCC pin accepts 5V or 3.3V power supply, Trigger pin which is a transmitter transmits ultrasonic sound waves, Echo pin also called receiver which receives the reflected signal and a Ground pin which connects to the ground. The distance is calculated by transmitting ultrasonic sound waves and measuring the time until we get the reflected waves from an object. An ultrasonic sensor is a sensor based on the principle of detecting an object's distance or bearing by sending out sound waves and measuring the time it takes for those waves to bounce off the object and return to the detector.

Modern ultrasonic sensors use semiconductor devices that can generate and detect a range of ultrasonic frequencies, from 20 kHz to 1 MHz. The output voltage of these sensors is proportional to both distance and speed. We can then calculate the distance with the help of the Distance Measurement formula

$$D = (S * T)/2,$$

Where,

D: is the distance to be measured

S: is the Speed of the sound wave which is equal to 343 m/s

T: is the Time measured

The distance measured is divided by 2 as the sound waves travel back and forth.

In this project, we have used two ultrasonic sensors. One sensor is positioned below the camera module which will indicate the distance between the user and the object whenever the camera detects an object. The other sensor is specifically positioned towards the floor or ground surface since the detection model detects staircases generally, irrespective of whether it is upstairs or downstairs, this sensor will hold a threshold distance between the user and floor and will indicate whenever a staircase is detected as upstairs or downstairs depending upon the change in distance. For example, if a staircase is detected and the distance is lesser than the threshold distance, it is said to be Upstairs. Similarly, if the distance is greater than the threshold distance, it is said to be Downstairs.

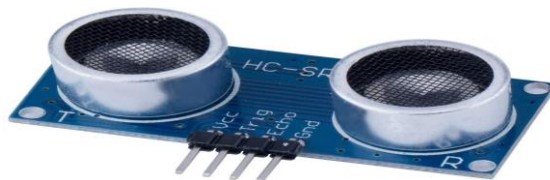


Fig.4.3: An ultrasonic sensor

4.1.3 Buzzer:

A buzzer is an output sensor that we can use to indicate sound warnings to the user. A buzzer has two pins, a longer pin that is positive and can be connected to any of the GPIO pins. The shorter pin is the negative pin which has to be connected to the ground.

In this project, we are using a buzzer that will give a sound warning whenever the distance of the object detected is 50 cm or less than 50 cm from the user.



Fig.4.4: A Buzzer

4.1.4 Other Components

Other components used in this project were a Breadboard and some jumper wires. Jumper wires are simple basic wires which help to pass the power around the circuit. A Breadboard has a base with electrical conductivity and sequential slots for joining jumper cables. On the sides of the breadboard, there are 2 parallel rails on each side, where one rail is the power rail which is indicated with a red line and another rail is the ground rail which is indicated with the blue line. In some breadboards, there may be a black line for ground rail. We have also considered using a PiCam which is a 5 MP compact camera model designed specifically for raspberry pi. This PiCam can be connected to the Raspberry Pi's Camera Serial Interface(CSI) port. The price of the camera module comes around 450 INR which is affordable and makes it likely to be used in any Raspberry Pi project.



Fig.4.5: Jumper wires

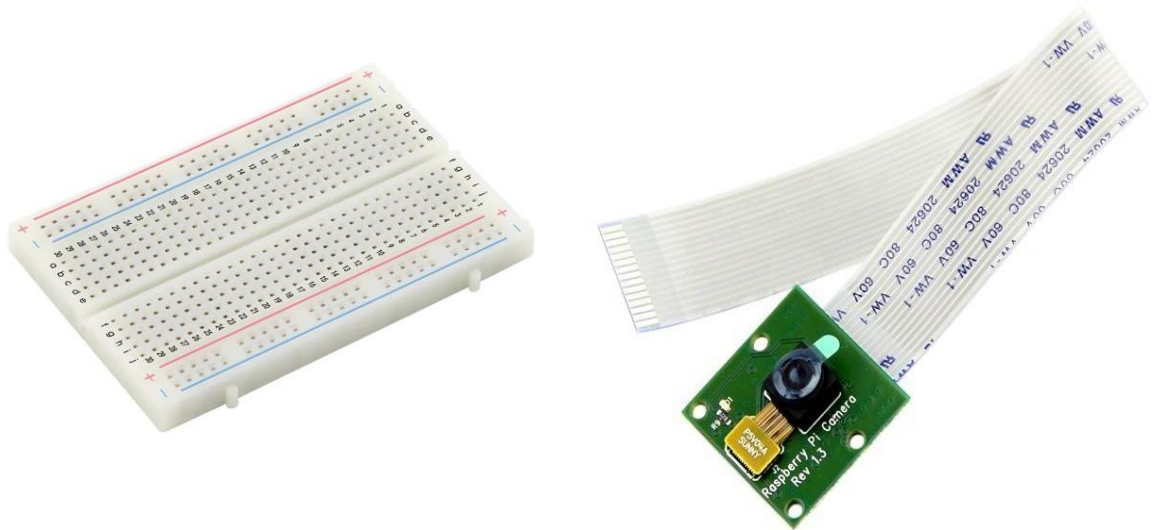
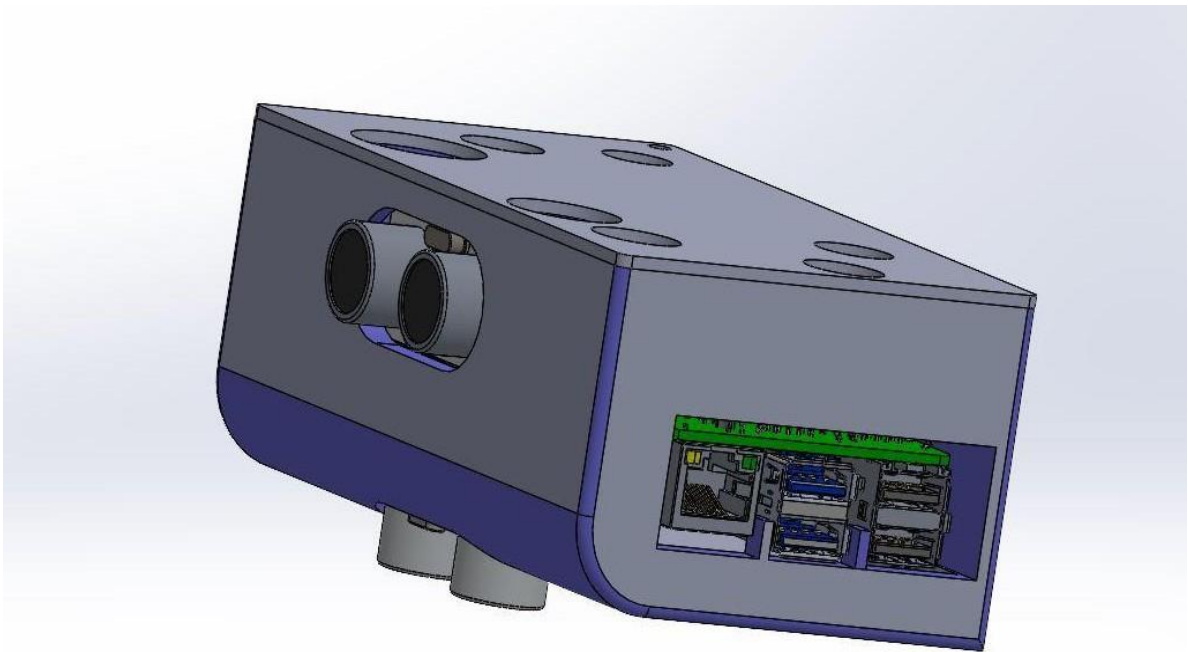


Fig. 4.6: A breadboard and PiCamera

4.2 DESIGN THEORY:

We came up with a design solution enclosure, which is comfortable to wear on the chest or can be worn as a waist belt too, which depends on the comfortability of the user.

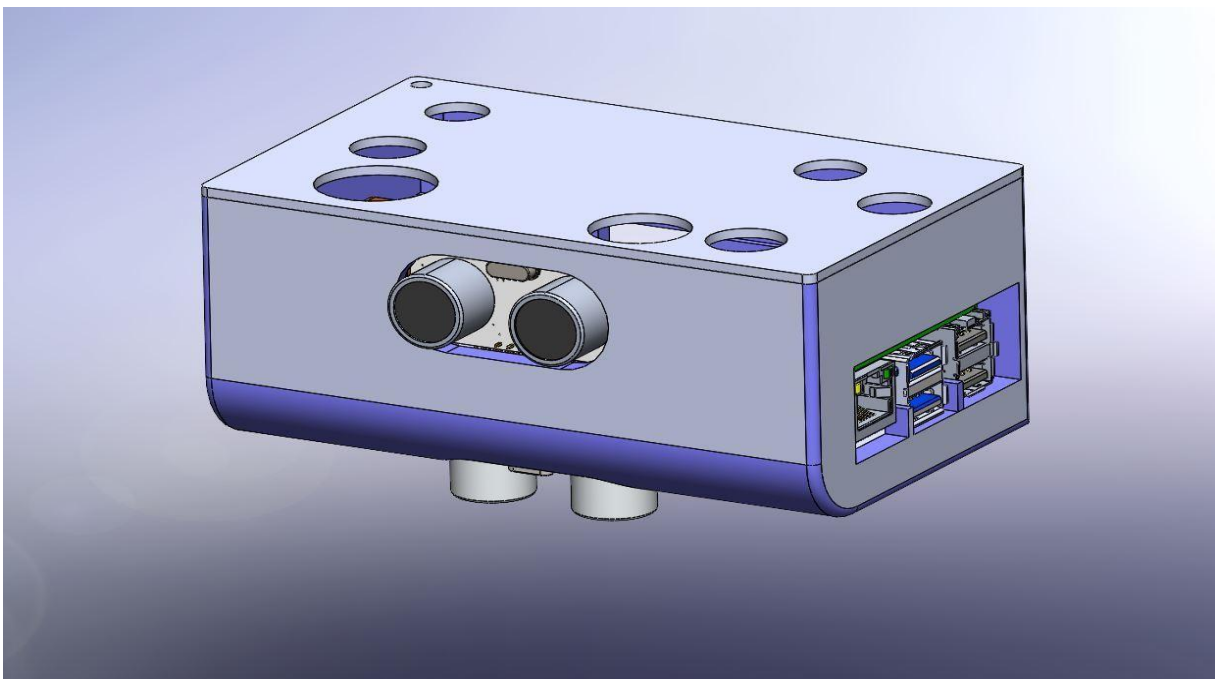
The enclosure assembly carries the Raspberry Pi, along with the whole circuit assembly which includes a breadboard, a pair of ultrasonic sensors, one spycam, and dozens of jumper wires. The design concept was to focus on an arrangement of all the components in such a way that they will be combined to form one single entity to turn into a wearable gadget. We considered the placement of the ultrasonic sensors in such a way that it will be properly oriented and aligned with the assembly to give precise output, we considered the volume or bulk of the enclosure to be as minimum as possible to make it cost-effective as well as comfortable for the user.



(a)



(b)



(c)

Fig.4.7: (a), (b), (c) CAD Design Of The Assembly Enclosure On Solidworks

The circuit assembly model is shown below in Fig 5.1, it includes a Raspberry Pi, a couple of ultrasonic sensors, a buzzer, and a breadboard with some jumper wires.

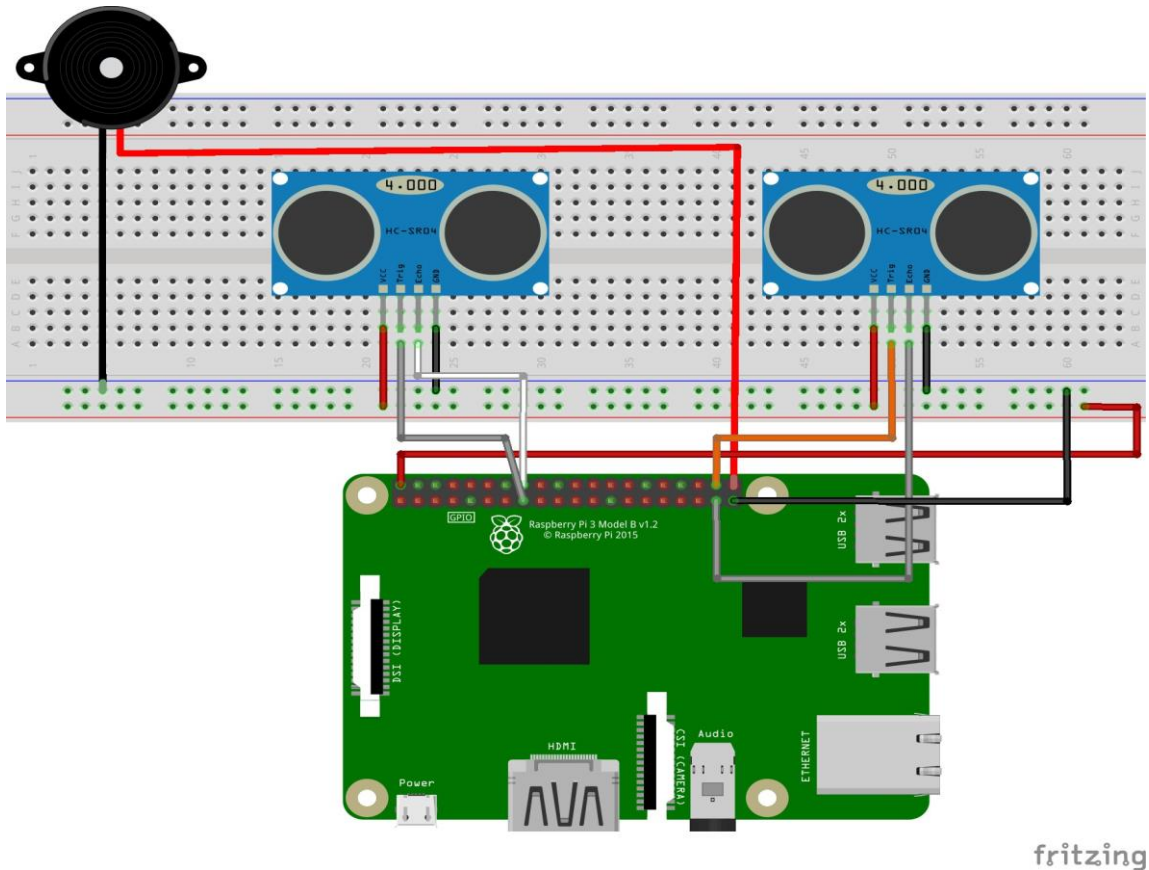


Fig.4.8: Hardware Design

Along with these components, we have also attached a camera that performs the main part of the detection, which is capturing the live surroundings in front of the user. Here we connected the PIN 2 which is a 5V to the power rail and PIN 39 which is a GND pin to the ground rail of the breadboard to achieve a common connection. There are various set modes to configure the GPIO pins of the pi. The two Commonly used modes are BCM and BOARD. BCM stands for Broadcom Mode. In this mode, the power and ground pins are named as such and the other pins are numbered according to the Broadcom SOC channel. In BOARD set mode, all the pins are named sequentially. For example, in BOARD mode, the third pin in raspberry pi is named 3 whereas, in BCM mode, it is named GPIO2. In this project, we have used the BOARD set mode. The VCC and GND pins of both the

ultrasonic sensors are connected to their respective rails in the breadboard. The trig and each pin of the first ultrasonic sensor which calculates the distance between object and device are connected to pins 15 and 16 of the pi. Likewise, the same two pins of the second sensor that calculates the distance between floors to the device are connected to pins 37 and 38 respectively. A buzzer is attached for the close object warnings to pin 40 of the pi. All these connected components are placed inside the enclosure.

4.3 ALGORITHM

Object detection algorithms and frameworks help to identify the type of the objects by the form of images(frames if to be found in a video source). Some of the detection algorithms include Single Shot Detection(SSD), Tensorflow APIs, You Only Look Once(YOLO), Region-Based Convolutional Neural Network(RCNN), Fast RCNN, Faster RCNN, etc. All these algorithms work under the concept of Deep Learning. Deep Learning is a part of Machine learning which consists of neural networks with three or more three layers. The human brain consists of neurons that form a network and transmit information to several regions of the brain. Similarly, neural networks in deep learning take large amounts of data(which is called a dataset) as input and try to learn from the dataset. From the learning, it should be able to make predictions when it is exposed to a particular object. These predictions can be determined by processing the data into several layers. Usually, three layers are used by the majority of frameworks and it is enough to determine appropriate predictions.

4.3.1 Yolo Pretrained Model:

You Only Look Once is an Object detection algorithm that is comparatively faster than other detection frameworks. It detects the object and represents the detection in the form of bounding boxes around the object. It works on the Darknet which is an open-source neural network. YOLO detects 80 different objects(a dataset called coco dataset) which are seen by a human on a day-to-day basis. Some of the objects include animals like horses, cats, dogs, vehicles like cars, vans, trucks, vegetables, fruits, and obviously, Humans. These objects have their configurations and weights which are pre-trained. We can use computer vision and utilize the pre-trained model. Along with this, there are certain common objects which are not included in the list, we can also train our custom model which will detect some specific objects that are defined. There are many versions of the YOLO algorithm and the latest is yolov5. To generate trained object weights, we can use the config files of the Yolo pre-trained model with the version of our choice and generate the weights. Weights are data of the trained objects that usually have values of the bounding box and other parameters like the width, and length of a particular image. It is usually shared as a file with the WEIGHTS extension. By processing these weights, the model will be able to identify the objects. In this project, we have utilized the pre-trained model for common object detection. Also, we have trained a custom model as we aim to detect staircases, pits, and potholes which are not included in the coco dataset.

Given below is the Source Code of the Pretrained Model:

```
# -*- coding: utf-8 -*-
```

```
''''
```

```
Created on Thu Jan 13 22:57:04 2022
```

```
@author: Raghavendra
```

```
''''
```

```
import cv2
```

```
import numpy as np
```

```

import os
import RPi.GPIO as GPIO
import time

#defining the ultrasonic sensor

GPIO.setmode(GPIO.BOARD)
trg=15
ech=16
buzzer=40

GPIO.setup(trg,GPIO.OUT)
GPIO.setup(ech,GPIO.IN)
GPIO.setup(buzzer,GPIO.OUT)

cap = cv2.VideoCapture(0)
targetWidthAndHeight = 320

classesFile = 'classes.names'
classes = open(classesFile).read().strip().split('\n')

COLORS = np.random.randint(0, 255, size=(len(classes), 3), dtype="uint8")

modelConfiguration = 'yolov3_custom.cfg'
modelWeights = "yolov3_custom_last.weights"

net = cv2.dnn.readNetFromDarknet(modelConfiguration, modelWeights)
net.setPreferableBackend(cv2.dnn.DNN_BACKEND_OPENCV)
net.setPreferableTarget(cv2.dnn.DNN_TARGET_CPU)

confidenceThreshold = 0.5
NMSThreshold = 0.3

```

```

if not cap.isOpened:
    print("error in loading the video")

while True:
    #calculating distance
    while True:
        GPIO.output(trg, False)
        time.sleep(2)
        GPIO.output(trg, True)
        time.sleep(0.00001)
        GPIO.output(trg, False)

        while GPIO.input(ech)==0:
            start_time=time.time()

        while GPIO.input(ech)==1:
            end_time=time.time()

        tof=end_time-start_time
        distance=(tof*34300.0)/2.0
        distance=round(distance,2)

        __, frame = cap.read()
        blob = cv2.dnn.blobFromImage(frame, 1/255, (targetWidthAndHeight,
targetWidthAndHeight), [0, 0, 0], 1, crop=False)
        net.setInput(blob)

        layerNames = net.getLayerNames()

        outputNames = [layerNames[i-1] for i in net.getUnconnectedOutLayers()]
        finalOutputs = net.forward(outputNames)

```

```

target_height, target_width, target_channels = frame.shape
bounding_box = []
# Contains x,y,w,h
class_ids = []
confidence_values = []
for output in finalOutputs:
    for det in output:
        target_arr = det[5:]
        class_id = np.argmax(target_arr)
        confidence = target_arr[class_id]
        if confidence > confidenceThreshold:
            width, height = int(det[2] * target_width), int(det[3] * target_height)
            x, y = int((det[0] * target_width) - width / 2), int(det[1] * target_height -
height / 2)
            bounding_box.append([x, y, width, height])
            class_ids.append(class_id)
            confidence_values.append(float(confidence))
    detectionNMS = cv2.dnn.NMSBoxes(bounding_box, confidence_values,
confidenceThreshold, NMSThreshold)
    if len(detectionNMS) > 0:
        for i in detectionNMS.flatten():
            (x, y) = (bounding_box[i][0], bounding_box[i][1])
            (w, h) = (bounding_box[i][2], bounding_box[i][3])

            color = [int(c) for c in COLORS[class_ids[i]]]
            cv2.rectangle(frame,(x, y),(x + w, y + h), color, 2)
            text = '{}: {:.4f}'.format(classes[class_ids[i]], int(confidence_values[i]))
            cv2.putText(frame, text, (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
color, 2)
            print(classes[class_ids[i]]+" detected at "+str(distance/100)+" meters")
            # VOICE OUTPUT
            if(distance>50 and distance<300):
                cmd=classes[class_ids[i]]+"detected at "+str(distance/100)+" meters"

```

```

        os.espeak(cmd)
    elif(distance<=50):
        GPIO.output(buzzer,HIGH)
        time.sleep(3)
        GPIO.output(buzzer,LOW)
        time.sleep(1)
        cmd="You are very close to a "+ classes[class_ids[i]
        os.espeak(cmd)

cv2.imshow('Frame', frame)

# Press Q on keyboard to  exit
if cv2.waitKey(25) & 0xFF == ord('q'):
    break

cap.release()

# Closes all the frames
cv2.destroyAllWindows()

```

4.3.2 Custom Model:

In this project, a custom-trained model has been made to detect staircases, pits, and potholes. We used google drive as storage where we load all the relevant files needed to make the custom detection. A GPU is needed and a Google Colab is an excellent tool for accessing and working with a GPU. Firstly, we need images of a particular object to train the dataset. We used a total of 2500 images for the training dataset. 1000 images for staircases, 750 images each for pits and potholes. The next step is to create a bounding box for all the images. It can be done using a tool called labeling. All the images in the dataset are manually labeled according to where the object is present in the image. After labeling each image, a text file will be generated with the same name as the image which will contain the details about the bounding box. The bounding box has four

parameters, the center of the X-axis(**x**), the center of the Y-axis(**y**), width(**w**), and height(**h**). We can use the text files to train the model. Upon labeling all the images, the dataset is uploaded to the drive. Then we import google drive to the colab.

```
from google.colab import drive
drive.mount('/content/drive')
```

As mentioned earlier, to use yolo, we need to use the darknet neural network. We can utilize darknet by cloning the darknet GitHub repository.

```
!git clone 'https://github.com/AlexeyAB/darknet.git' '/content/drive/My
Drive/FinalYearProject/darknet'
```

Next, we need to change the directory to the darknet folder and modify the MakeFile according to our requirements. Once it is modified, we can run the make command.

```
!make
```

After this command is executed we have to create train and test files for testing using the following python script.

```
"""
```

```
File: creating-train-and-test-txt-files.py
```

```
"""
```

```
# Creating files train.txt and test.txt for training in Darknet framework
```

```
# Algorithm:
```

```
# Setting up full paths --> List of paths -->
```

```
# --> Extracting 15% of paths to save into test.txt file -->
```

```
# --> Writing paths into train and test txt files
```

```
#
```

```
# Result:
```

```
# Files train.txt and test.txt with full paths to images
```

```
# Importing needed library
```

```

import os
"""

Start of:
Setting up the full path to the directory with labeled images
"""

full_path_to_images = 'Model_images' #dataset folder name

"""

End of:
Setting up the full path to the directory with labeled images
"""

"""

Start of:
Getting the list of full paths to labeled images
"""

# Check point
# Getting the current directory
# print(os.getcwd())

# Changing the current catalog
# to one with images
os.chdir(full_path_to_images)

# Check point
# Getting the current directory
# print(os.getcwd())

# Defining list to write paths in
p = []

# Using os.walk for going through all directories
# and files in them from the current directory
# Full Stop in os.walk('.') means the current directory

```

```

for current_dir, dirs, files in os.walk('.'):
    # Going through all files
    for f in files:
        # Checking if filename ends with '.jpg'
        if f.endswith('.jpg'):
            # Preparing path to save into train.txt file
            path_to_save_into_txt_files = full_path_to_images + '/' + f

            # Appending the line into the list
            # We use here '\n' to move to the next line
            # when writing lines into txt files
            p.append(path_to_save_into_txt_files + '\n')

# Slicing first 15% of elements from the list
# to write into the test.txt file
p_test = p[:int(len(p) * 0.15)]

# Deleting from initial list first 15% of elements
p = p[int(len(p) * 0.15):]

"""
End of:
Getting list of full paths to labeled images
"""

"""
Start of:
Creating train.txt and test.txt files
"""

# Creating file train.txt and writing 85% of lines in it
with open('train.txt', 'w') as train_txt:
    # Going through all elements of the list
    for e in p:

```

```

        # Writing current path at the end of the file
        train_txt.write(e)

# Creating file test.txt and writing 15% of lines in it
with open('test.txt', 'w') as test_txt:
    # Going through all elements of the list
    for e in p_test:
        # Writing current path at the end of the file
        test_txt.write(e)

```

End of:

Creating train.txt and test.txt files

Once the files are generated, there comes the actual training part. We change to the root directory and run the following command to start the training. Before that, we should download a file darknet53.conv.74 which contains some pre-trained weights.

```

!darknet/darknet detector train ModelImages/labelled_data.data
darknet/cfg/yolov3_custom.cfg custom_weight/darknet53.conv.74 -dont_show

```

The training took approximately 4 hours to complete. As a result, a custom trained weights file will be generated which will be used for the custom trained object detection. The flowchart representing the workflow of the custom model is shown in Fig. 6.1.

Given below is the Source Code of Custom Model:

4.3.2.1 Source Code - Custom Model

```

# -*- coding: utf-8 -*-

```

Created on Thu Jan 13 22:57:04 2022

@author: Raghavendra

.....

```
import cv2
import numpy as np
import os
import RPi.GPIO as GPIO
import time
#defining the ultrasonic sensor
```

```
GPIO.setmode(GPIO.BOARD)
trg=38
ech=37
buzzer=40
```

```
GPIO.setup(trg,GPIO.OUT)
GPIO.setup(ech,GPIO.IN)
GPIO.setup(buzzer,GPIO.OUT)
```

```
#calculation of threshold b/w floor and device
```

```
os.espeak("Calculating distance from floor to the device. Please do not move.")
```

```
i=0
```

```
while(i<5):
```

```
    GPIO.output(trg, False)
```

```
    time.sleep(2)
```

```
    GPIO.output(trg,True)
```

```
    time.sleep(0.00001)
```

```
    GPIO.output(trg,False)
```

```
    while GPIO.input(ech)==0:
```

```
        start_time=time.time()
```

```
    while GPIO.input(ech)==1:
```

```
        end_time=time.time()
```

```

    tof=end_time-start_time
    floor_threshold=(tof*34300.0)/2.0
    floor_threshold=floor_threshold/100 #converting to meters
    floor_threshold=round(distance,2)
    i++

os.espeak("floor distance is "+str(floor_threshold)+" meters. You can now proceed
with your journey.")

cap = cv2.VideoCapture(0)
targetWidthAndHeight = 320

classesFile = 'classes.names'
classes = open(classesFile).read().strip().split('\n')

COLORS = np.random.randint(0, 255, size=(len(classes), 3), dtype="uint8")

modelConfiguration = 'yolov3_custom.cfg'
modelWeights = "yolov3_custom_last.weights"

net = cv2.dnn.readNetFromDarknet(modelConfiguration, modelWeights)
net.setPreferableBackend(cv2.dnn.DNN_BACKEND_OPENCV)
net.setPreferableTarget(cv2.dnn.DNN_TARGET_CPU)

confidenceThreshold = 0.5
NMSThreshold = 0.3

if not cap.isOpened:
    print("error in loading the video")

while True:

```

```
#Floor distance calculation
```

```
GPIO.output(trg, False)
time.sleep(2)
GPIO.output(trg, True)
time.sleep(0.00001)
GPIO.output(trg, False)

while GPIO.input(ech)==0:
    start_time=time.time()

while GPIO.input(ech)==1:
    end_time=time.time()

tof=end_time-start_time
floor_distance=(tof*34300.0)/2.0
floor_distance=floor_distance/100 #converting to meters
floor_distance=round(distance,2)
_, frame = cap.read()
blob = cv2.dnn.blobFromImage(frame, 1/255, (targetWidthAndHeight,
targetWidthAndHeight), [0, 0, 0], 1, crop=False)
net.setInput(blob)
layerNames = net.getLayerNames()
outputNames = [layerNames[i-1] for me in net.getUnconnectedOutLayers()]
final_output = net.forward(outputNames)
# print(finalOutputs[0].shape)    # LENGTH SHOULD BE 3

target_height, target_width, target_channels = frame.shape
bounding_box = []
# Contains x,y,w,h
class_ids = []
confidence_values = []
for output in final_outputs:
    forest in the output:
```

```

target_arr = det[5:]
class_id = np.argmax(target_arr)
confidence = target_arr[class_id]
if confidence > confidence threshold:
    width, height = int(det[2] * target_width), int(det[3] * target_height)
    x, y = int((det[0] * target_width) - width / 2), int(det[1] * target_height -
height / 2)
    bounding_box.append([x, y, width, height])
    class_ids.append(class_id)
    confidence_values.append(float(confidence))
detectionNMS = cv2.dnn.NMSBoxes(bounding_box, confidence_values,
confidence threshold, NMSThreshold)
if len(detectionNMS) > 0:
    for l in detectionNMS.flatten():
        (x, y) = (bounding_box[i][0], bounding_box[i][1])
        (w, h) = (bounding_box[i][2], bounding_box[i][3])

        color = [int(c) for c in COLORS[class_ids[i]]]
        cv2.rectangle(frame,(x, y),(x + w, y + h), color, 2)
        text = '{}: {:.4f}'.format(classes[class_ids[i]], int(confidence_values[i]))
        cv2.putText(frame, text, (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
color, 2)
        print(classes[class_ids[i]]+" warning")
# VOICE OUTPUT
if(classes[class_ids[i]]==" Staircase" and distance<floor_threshold):
    os.espeak("Upstairs detected")
Elif(classes[class_ids[i]]==" Staircase" and distance>floor_threshold):
    os. speak("Downstairs detected")
else:
    cmd=classes[class_ids[1]]+" detected"
    os. speak(cmd)

# Press Q on the keyboard to exit
if cv2.waitKey(25) & 0xFF == ord('a):

```

```

break
cap. release()
# Closes all the frames
cv2.destroyAllWindows()

```

To describe in words, whenever the user turns on the device, it will give a voice output that it's going to calculate the distance between the floor and the device.

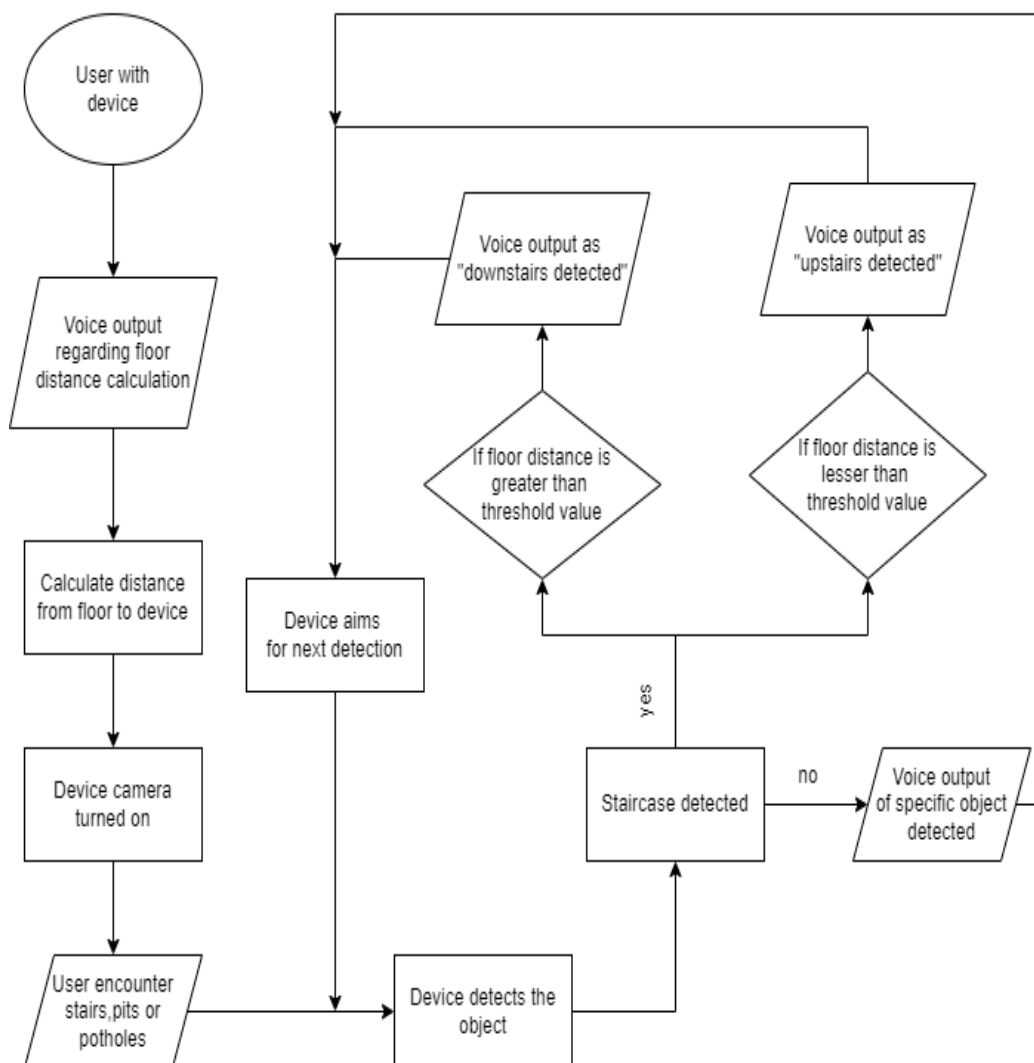


Fig 4.9 Workflow of the Custom model: Block Diagram

The user will be instructed not to move till the measurement is done and the time it takes to measure is 5 seconds. After that, the user can use the device as normal. Whenever the user gets near a staircase, pit, or pothole, the device's camera will detect the object and alert the user accordingly. One drawback is that there are chances that a user might get confused when encountering staircases whether it is upstairs or downstairs. To overcome this issue, an ultrasonic sensor constantly monitors the distance between the ground surface and the device provided with a threshold value. If the distance is greater than the threshold value, the user will get a voice output as "Downstairs detected" and "Upstairs detected", if the distance is lesser than the threshold. This process is undergone in a logical AND condition, that is the staircase will be detected only when the camera detects a staircase frame along with a change in floor distance.

With the combination of the codes and the hardware altogether, we have made a circuit prototype which is shown in the figure below:



Fig.4.9: Prototype: Circuit Assembly

CHAPTER 5

RESULTS, DISCUSSION, AND PERFORMANCE ANALYSIS

5.1. EXPERIMENT METHODOLOGY:

The prototype was tested with 24 subjects. The subjects were of different age groups, the majority were around 20 to 25 years. We have done several scenario-based experiments such as placing an object or person in the camera frame, a user walking directly to the object or person, a user sitting in a chair, holding a water bottle, putting several objects on a table and detecting them, capturing vehicles on the roads, detecting objects from a youtube video, etc. The tests were performed based on various parameters such as accuracy of object detection, voice audio, accuracy, buzzer warnings, etc. The prototype satisfied the users in most aspects as we have attached the graphs and explained below. There was one major problem we were encountering with the performance which is the delay in rendering time, which is the change of one camera frame to another, takes a significant amount of time to render. From the tests we conducted, the rendering delay time is approximately 17.304 seconds per frame on average. This is significantly a large amount of time when it comes to object detection. We tried to identify the solution for this issue but we found that all embedded devices face this kind of issue with object detection. The same code performs well without delay in a laptop or a PC but as portability is the key factor of this device, we went for an embedded device like the raspberry pi. We have also attached some screenshot images of the outputs from the device and their graphs of comparison.

Our experiment methodology included:

- Detection of objects (It could be a person, vehicle, stairs, pits, etc.)
- Distance between the person who's wearing the gear and the obstacle
- Distance from the ground level to the enclosure
- Buzzer sound, if the obstacle is < 50 cm (can be varied) to the user.

We have achieved some characteristics of our gadget according to the experimentation, and the plot of that is shown below:

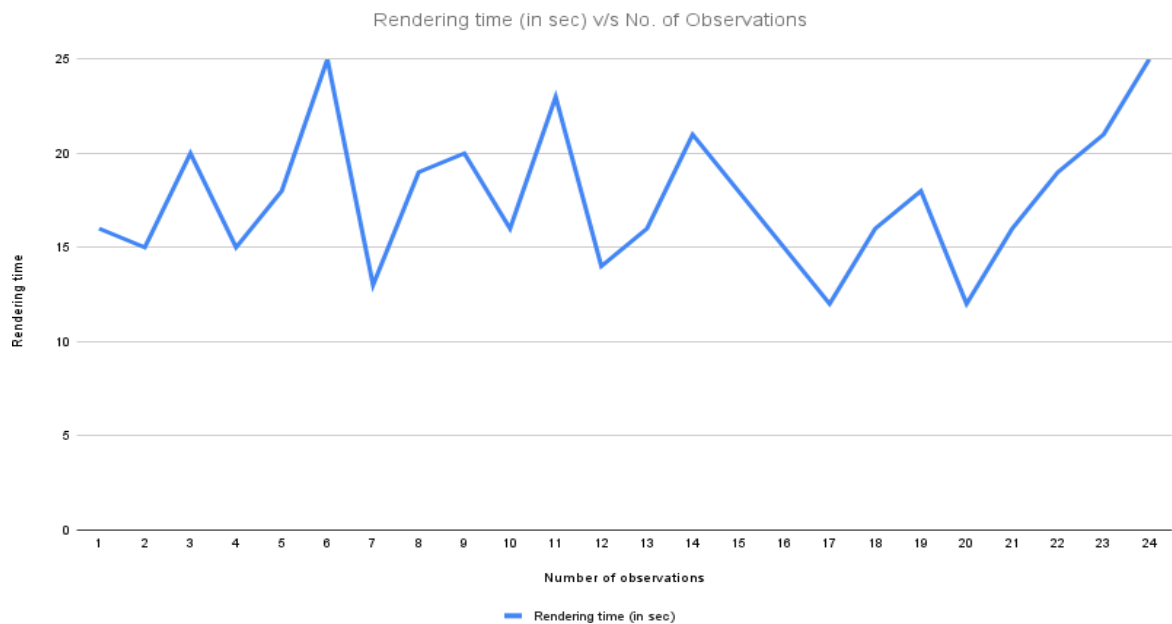


Fig.5.1: Rendering Time vs No of Observations

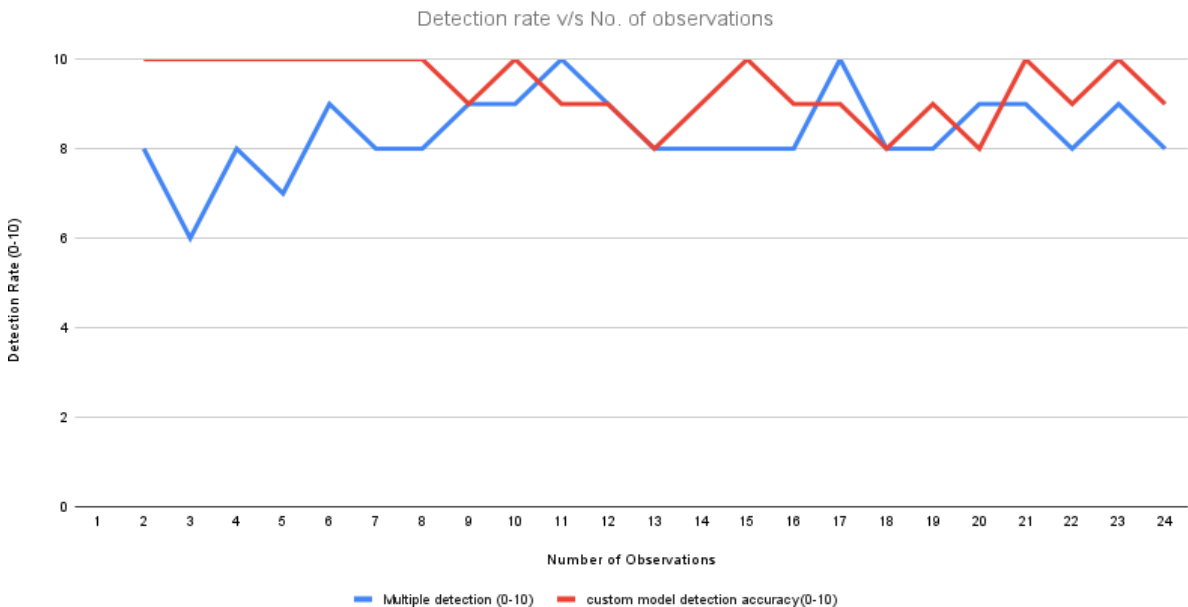


Fig.5.2: Detection Rate vs No of Observations

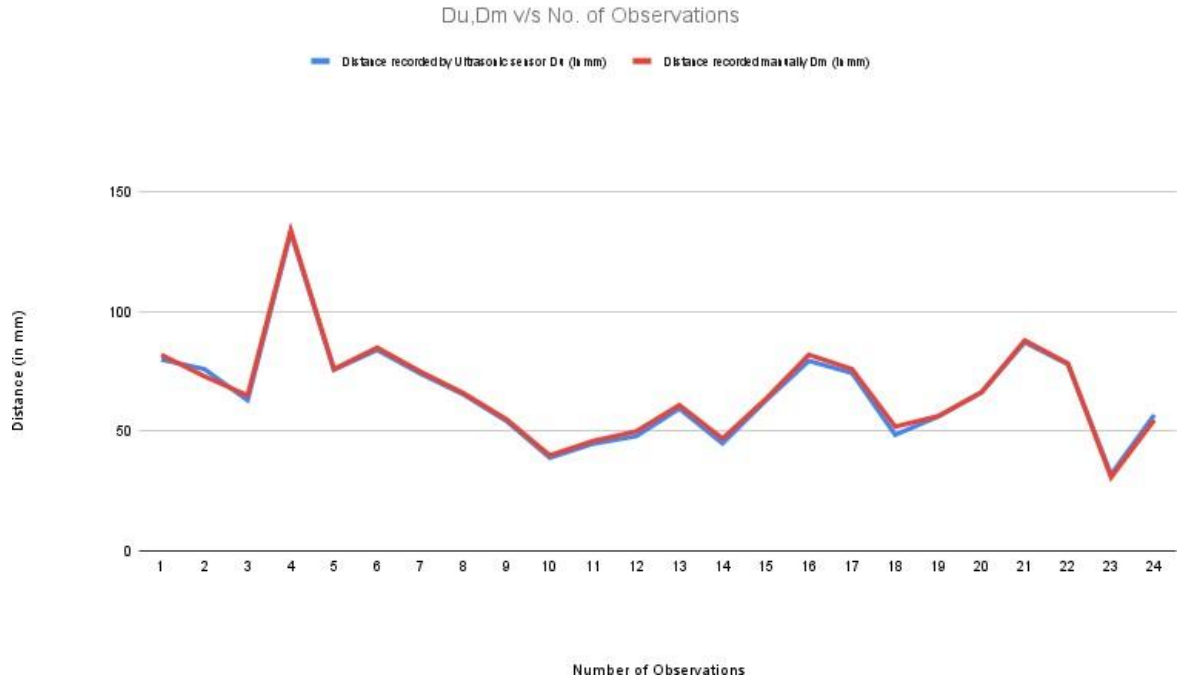


Fig.5.3: D_u , D_m vs No of observations

The above plots are a demonstration of the output of the Dataset, Fig. 5.2 presents the Rendering time the algorithm took for each frame detection for every subject. Here, we can observe a non-uniform trajectory. The average value for Rendering time we have obtained out of 25 subjects has been noticed. Fig 5.4 shows the detection rate vs no. of observation which was set from 0- 10. Fig. 5.3 we can observe the curve produced by the ultrasonic distance detection sensor vs manual readings. The curve trajectory is almost similar in both the cases, which concludes that the ultrasonic sensors were providing precise output.

Table 5.1. Observation 1 (The following tables show the observations taken from various subjects during the experimentation)

No of Observations	Name of Subject	Age	Object Detection Accuracy (0/1)	Time is taken for Object Detection (in sec)	Rendering time (in sec)	Distance recorded Ultrasonic sensor Du (in mm)	Distance recorded manually Dm (in mm)	Distance difference (Dm-Du) (in mm)
1	Aman	21	1	1	16	80	82	200
2	Raghav	21	1	1	15	76	73	300
3	Imran Khan	21	1	1	20	63	65	200
4	Reyman	21	1	1	15	133	134	100
5	Saravana	21	1	1	18	75.7	76	30
6	Deepak I	37	1	1	25	84	85	100
7	Deepak II	37	1	1	13	74.15	75	85
8	Niraj	28	1	1	19	65.4	66	60
9	Somanshu	23	1	1	20	54.28	55	72
10	Vara Prasad	23	1	1	16	39	40	100
11	Mohit	24	1	1	23	44.824	46	117
12	Rahul	28	1	1	14	48	50	200
13	Anantha	27	1	1	16	59.5	61	150
14	Saravanan	29	1	1	21	45	47	200
15	Prabhu	21	1	1	18	62.8	63.5	70
16	Kishore	22	1	1	15	79.45	82	275
17	Aravind	26	1	1	12	74.36	76	164
18	Shiva	32	1	1	16	48.56	52	344
19	Anas	29	1	1	18	56.25	56.4	15
20	Kovindhan	21	1	1	12	66.25	66.4	15
21	Shibin	21	1	1	16	87.2	88	80
22	Venkatesh	21	1	1	19	78	78.4	40
23	Sakthi Vel	21	1	1	21	32	30.8	120
24	Gaurav	21	1	1	25	56.89	54.76	213

Table 5.2. Observation Phase 2

No of Observations	Name of Subject	Age	Voice accuracy and audibility (in 0-10)	Comfort ability Level (0 to 10)	Comfort ability Level (0 to 10)	custom model detection accuracy (0-10)	upstairs detection (0-10)	downstairs detection (0-10)
1	Aman	21	8	6	8	9	9	8
2	Raghav	21	8	5	8	10	8	7
3	Imran Khan	21	7	3	6	10	8	6
4	Reyman	21	8	6	8	10	8.5	9
5	Saravana	21	8	7	7	10	7	9
6	Deepak I	37	8	7	9	10	8	8
7	Deepak II	37	8	7	8	10	8	8
8	Niraj	28	9	7	8	10	9	9
9	Somanshu	23	8	8	9	9	10	8
10	Vara Prasad	23	10	8	9	10	9	10
11	Mohit	24	10	8	10	9	8	8
12	Rahul	28	9	9	9	9	10	9
13	Anantha	27	8	9	8	8	8	7
14	Saravanan	29	7	8	8	9	10	9
15	Prabhu	21	8	7	8	10	9	9
16	Kishore	22	9	8	8	9	10	10
17	Aravind	26	10	9	10	9	9	10
18	Shiva	32	9	9	8	8	10	7
19	Anas	29	8	8	8	9	9	8
20	Kovindhan	21	10	7	9	8	8	10
21	Shibin	21	10	6	9	10	8	8
22	Venkatesh	21	8	7	8	9	9	9
23	Sakthi Vel	21	9	8	9	10	10	10
24	Gaurav	21	8	7	8	9	10	10

The above-mentioned Dataset experimentation was done in a controlled environment. According to the dataset, we have collected, the following characteristics can be concluded :

5.2 CUSTOM MODEL ANALYSIS :

- Error Rate = $| \text{Measured Value} - \text{Given Value} | / \text{Given Value} \times 100$

Measured Average Value = 203; Given Value or Total value = 230

$$= (|203-230|) / 230 * 100$$

$$\text{Error Percentage} = 10.434 \%$$

- Accuracy = $100\% - \text{Error Rate}$

$$= 100 - 10.434$$

$$\text{Accuracy percentage} = 89.566\%$$

5.3 PRE-DEFINED MODEL ANALYSIS:

- Error Rate = $| \text{Measured Value} - \text{Given Value} | / \text{Given Value} \times 100$

Measured Average Value = 184; Given Value or Total value = 230

$$= (|184-230|) / 230 * 100$$

$$\text{Error Percentage} = 20\%$$

- Accuracy = $100\% - \text{Error Rate}$

$$= 100 - 20$$

$$\text{Accuracy percentage} = 80\%$$

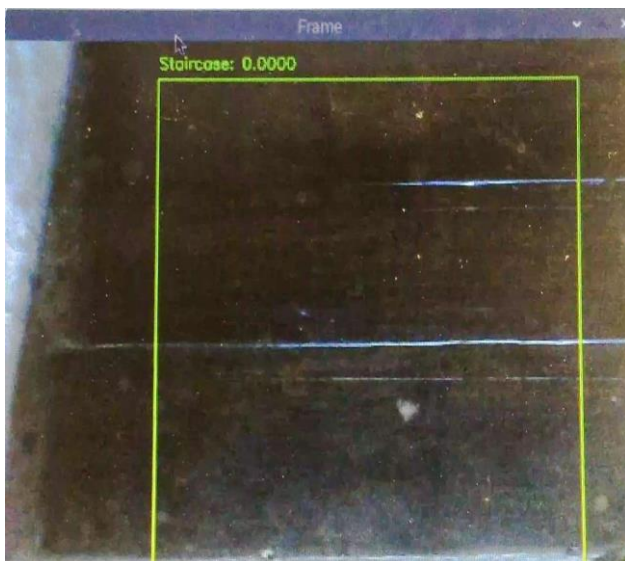
Below are some output screenshots as per the object detection algorithm and along with that, we have attached some real time detection and experimentation images.



(i)



(ii)



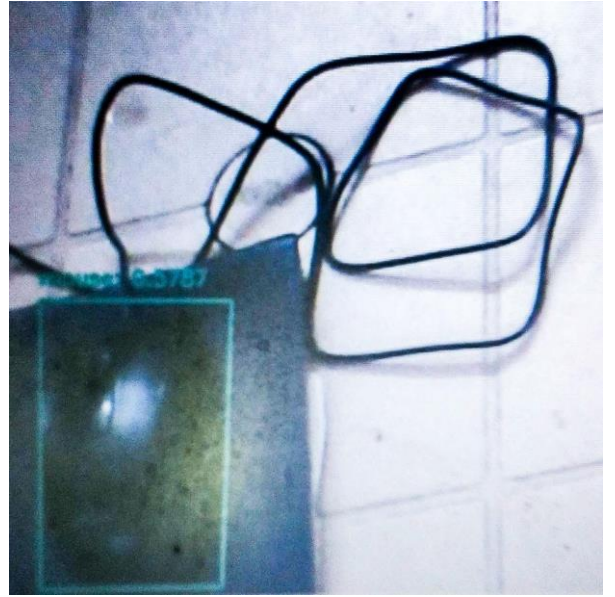
(iii)



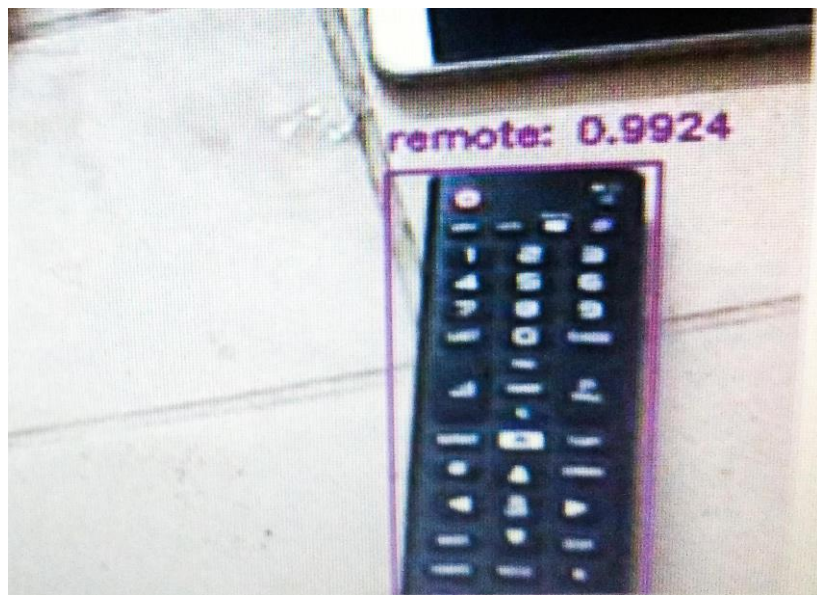
(iv)



(v)



(vi)



(vii)

Fig 5.4: Model detecting (i)Motorbike, (ii)a Person, (iii)Staircase, (iv) Clock, (v)Cell Phone, (vi) computer mouse, and (vii) TV remote.



Fig.5.5: Real-time object detection - Experimenting with a subject in real-time.

CHAPTER - 6

CONCLUSION

6.1 SUMMARY

Improving the lifestyle of visually impaired people is important and thus we have developed a **“SMART WEARABLE GADGET TO DETECT STAIRS, POTHOLE, AND OTHER OBJECTS FOR VISUALLY IMPAIRED USERS USING COMPUTER VISION”** which helps visually impaired users to guide themselves without depending on others. Our model detects obstacles, stairs, pits, and potholes at a very quicker rate when they are being captured by the camera. We also found our model faces some issues concerning the frame rendering. We will try to sort this out in our future works. Also, we have planned to integrate google maps using google's API to advance and improvise navigation for the user. This will also be followed in our future works.

This project work has provided us with an excellent opportunity and experience. We gained a lot of practical knowledge regarding, planning, purchasing, assembling, and programming while doing this project work. We have gained a high level of knowledge of computer vision and its various algorithms. By using more techniques, the device can be furthermore modified and developed according to the applications.

6.2 STATIC AND DYNAMIC CHARACTERISTICS:

6.2.1 Accuracy Percentage:

For the custom model, the accuracy percentage was recorded as, (data considered from table 5.1 and table 5.2)

Accuracy percentage = 89.566%

Whereas, for the Pretrained model, it was recorded as,

Accuracy percentage = 80.000%

6.2.2 Error Percentage:

For the custom model, the error percentage was recorded as, (data considered from table 5.1 and table 5.2)

Error percentage = 10.434%

Whereas, for the Pretrained model, it was recorded as,

Error percentage = 20.000%

6.3 BUDGET REPORT

Table 6.1: Budget Report

ITEMS	NUMBER OF ITEMS	PRICE
Raspberry Pi	01	Rs. 5850
Camera Module	01	Rs. 450
ULTRASONIC SENSOR	02-03	Rs. 300
VIBRATION SENSOR	02	Rs. 200
GPS SENSOR	01	Rs. 650
BATTERY	01	Rs. 40
	TOTAL COST:	Rs.7490 Approx.:Rs. 7500

The idea was to keep the cost of the project to as minimum as possible, we consider the current price is still comparatively higher, for that the future scope of the project will be specifically focused on the cost reduction of the product, so as every user can afford and make use of it.

REFERENCES

- [1] Tahoun, N., Awad, A., Bonny, T. (2019, October). Smart Assistant for Blind and Visually Impaired People. In Proceedings of the 2019 3rd International Conference on Advances in Artificial Intelligence (pp. 227- 231).
- [2] Masud, U., Saeed, T., Malaikah, H. M., Islam, F. U., Abbas, G. (2022). Smart Assistive System for Visually Impaired People Obstruction Avoidance Through Object Detection and Classification. IEEE ACCESS, 10, 13428-13441.
- [3] Alam, M. N., Islam, M. M., Habib, M. A., Mredul, M. B. (2018). Staircase detection systems for the visually impaired people: a review. International Journal of Computer Science and Information Security (IJCSIS), 16(12), 13-18.
- [4] Adrian Rosebrock, "Real-time object detection with deep learning and opencv", <https://pyimagesearch.com/2017/09/18/real-timeobjectdetection-with-deep-learning-and-opencv/> , September 2017.
- [5] CHEHADE, J., ABOU HAYDAR, G., HAYEK, A., BOERCSOEK, J., OLMEDO, J. J. S. (2020, December). Design and Implementation of Smart Shoes for Blind and Visually Impaired People for More Secure Movements. In 2020 32nd International Conference on Microelectronics (ICM) (pp. 1-6). IEEE.
- [6] Chun, A. C. B., Theng, L. B., WeiYen, A. C., Deverell, L., Mahmud, A., McCarthy, C. (2019). A ground plane hazards detection tool for the visually impaired. International Journal of Mechanical Engineering and Robotics Research, 8(1), 146-156.
- [7] Anisha, M., et al. "Low-Cost Smart Shoe for Visually Impaired." 2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV). IEEE, 2021.

- [8] Xu, Qianli, et al. "Design and evaluation of vibrating footwear for navigation assistance to visually impaired people." 2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData). IEEE, 2016.
- [9] Koli, Pooja S., Nikita Sunil Chavan, and Shraddha Gajanan Gaikar. "Construction of smart shoe system for visually impaired people." *Int J Res Analyt Rev* 6.1 (2019): 2349-5138.
- [10] Kumar, Pradeep, et al. "Real Time Assistive Shoe for Visually Impaired People." 2021 6th International Conference for Convergence in Technology (I2CT). IEEE, 2021.
- [11] Chandekar, T., Chouhan, R., Gaikwad, R., Gosavi, H., Darade, S. (2017). Implementation of obstacle detection and navigation system for visually impaired using smart shoes. *International Research Journal of Engineering and Technology (IRJET)*, 4(4).
- [12] Du, Juan. "Understanding of object detection based on CNN family and YOLO." *Journal of Physics: Conference Series*. Vol. 1004. No. 1. IOP Publishing, 2018.
- [13] Singh, Varsha, Savita Sindhu, and Romisha Arora. "BUZZFEET: Blind Man Shoes." 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon). IEEE, 2019.
- [14] Huy-Hieu Pham, Thi-Lan Le and Nicolas Vuillerme, "Real-time obstacle detection system in indoor environment for the visually impaired using microsoft kinect sensor", *Journal of Sensors*, vol. 2016,Article ID 3754918, 2015.
- [15] Ashish Patankar and Tooraj Nikoubin, "Wearable system for obstacle detection and human assistance using ultrasonic sensor array ",Dallas, TX, USA, *Proceedings of the 7th International Conference on Computing Communications and Networking Technologies, ICCCNT*, 2016.

- [16] Chen, Shiwei , Yao, Dayue , Cao, Huiliang , Shen, Chong. (2019). A Novel Approach to Wearable Image Recognition Systems to Aid Visually Impaired People. *Applied Sciences*. 9. 3350. 10.3390/app9163350. , 2019.
- [17] P. Herghelegiu, A. Burlacu and S. Caraiman, "Negative obstacle detection for wearable assistive devices for visually impaired," 2017 21st International Conference on System Theory, Control and Computing (ICSTCC), 2017, pp. 564-570, doi: 10.1109/ICSTCC.2017.8107095 , 2017.
- [18] Dimas, G., Diamantis, D. E., Kalozoumis, P., Iakovidis, D. K. (2020). Uncertainty-Aware Visual Perception System for Outdoor Navigation of the Visually Challenged. *Sensors* (Basel, Switzerland), 20(8), 2385. <https://doi.org/10.3390/s20082385>.
- [19] H. Takizawa, and M. Aoyagi, "Assistive Systems for the Visually Impaired Based on Image Processing", in *Causes and Coping with Visual Impairment and Blindness*. London, United Kingdom: IntechOpen, 2017 [Online]. Available: <https://www.intechopen.com/chapters/56916> doi: 10.5772/intechopen.70679.
- [20] Wickramasinghe, B. K. T. P., Anjana, K. K. V. S., Ratnasekara, G. D. S. T., Vidanagama, D. U. (2017). Smart Cane for Staircase and Water Detection.

OUTCOME OF THE PROJECT

- This has been submitted as Journal Paper to ICCET 2022

SMART WEARABLE GADGET
TO DETECT STAIRS,
POTHOLE, AND OTHER
OBJECTS FOR VISUALLY
IMPAIRED USERS USING
COMPUTER VISION. ➡



Inbox



prof prince Mar 15
to me ▾



Dear Author/s,

We are happy to inform you that your paper, submitted for the ICCET 2022 **conference** has been **Accepted** based on the recommendations provided by the Technical Review Committee. By this mail you are requested to proceed with Registration for the **Conference**. Most notable is that the **Conference** must be registered on or before **MARCH 20, 2022** from the date of acceptance.

www.iccet.in

Kindly fill the registration form, declaration form which is attached with the mail and it should reach us on above mentioned days.

- Conference Paper First page:

Smart Wearable Gadget to Detect Stairs, Potholes, and Other Objects for Visually Impaired Users Using Computer Vision

1st Aman Dinodiya
Department of Mechatronics
Sathyabama University
Chennai, India
amandinodiya0011@gmail.com

2nd Raghavendra R
Department of Mechatronics
Sathyabama University
Chennai, India
raghavnik22@gmail.com

3rd Dr. S.Prakash
Dean, School of Mechanical
Sathyabama University
Chennai, India
dean.mechanical@sathyabama.ac.in

Abstract—The sense of sight is one of the most vital aspects of life. People with partial and complete visual impairment encounter a lot of challenges in the day-to-day environment. The world is approaching development and stability expeditiously and it became so tough to keep up for someone who lacks the basic senses. Visually impaired people are part of this pursuit of development as well, so the advancements must flee a crucial effect on their lives to make what was impractical for them practical and accessible as well. The formal comfort provided earlier for visually impaired people was as a specific hardware device such as Optical Character Recognition, identifying color, barcode readers, which had a limited reach among people and were not at all economical. A wearable gadget to provide directional navigational information to visually impaired people and alert them about the environment whenever the sensors notice jeopardy in the surroundings. This gadget has a lot of prospects to guide and navigate smartly to the user. The implementation results in enhancement and understanding of the problems faced by a blind person, and it will help aware people to achieve more projects targeted to help blind people.

Keywords: Visually Impaired, Raspberry Pi, LIDAR sensors, Ultrasonic sensor, IFTTT, GPS Technology, Staircase detection, Image Processing, Computer vision.

I. INTRODUCTION

As per NCBI records, 7.7 billion people live on Earth, around 36 million are considered to be blind and more than 250 million people have moderate/severe Visual Impairments. The number of blind people raised by 43% over the last 20 years. Since human vision plays a very important role when it comes to interacting with the environment, In case it is not functioning well, then the person should be dependent on others forever. They face many challenges both indoors and outdoors, irrespective of whether the activity they perform is simple or complex, they are going through a lot of hurdles. Currently, most visually impaired and blind people rely upon cane sticks to ensure hurdle-free movements. The World Health Organization(WHO) states that the main causes for visual impairment are uncorrected refractive errors, Cataracts, Age-related macular degeneration, Glaucoma, Diabetic retinopathy, Corneal opacity Trachoma, etc. Technology helps people to create new devices and tools for blind people to make their lives easier. Many people have created devices to

generally detect the obstacle in front of the user and notify it using some ultrasonic distance sensors, whereas devices which specify the actual object present are quite minimal which provides high accuracy and precision with detection of the object and alert the user. The concept of Computer Vision helps to eliminate the challenge of specifying the objects with the help of several algorithms such as You Only Look Once(YOLO), Tensorflow, Single Shot Detection(SSD), Convolutional Neural Networks(CNN), etc. Though these frameworks work on some specific objects which are contained in the COCO(Common Objects and Context) Dataset, there are some other objects which humans see in their day-to-day lives. To ensure those objects are also included for detection, we can train our model specifically to detect a particular object or couple of objects. Taking various ideas about the implementation of such devices, we have managed to model a gadget with accurate results using the YOLO framework. The prototype of the project serves the purpose of generally drawing the attention of usage of such wearable devices throughout the globe with ease. The block diagram shown below explains the whole research methodology being used in this prototype. The model represents a rough step-by-step demonstration of the process which was followed. The conventional system is already being used by many users, most of the users are relying on the Cane Sticks in object detection. Further, some commercialized products are available that detect obstacles and alert the user. The further implementation of the project will be done in an overall cost-effective way and on a large scale to make the product reliable and effective for all the users. The future prototypes will be more accurate and economical, as every user can afford this assistant wearable gadget. The upcoming projects will be more compact and comfortable for the user to wear. We are planning to introduce some extra sensors along with the already existing ones in this model to provide more accurate and reliable results. One aspect of future planning is in collaboration of this device with an earbud or an earphone, so as to become more accurate in terms of alerting the user. This concept will enhance the output by providing a clean and clear sound directly to the ear of the

• **Plagiarism Report First Page of this Project Report Document:**

