VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnanasangama, Macche, Santibastwada Road
Belagavi-590018, Karnataka

B.E MINI-PROJECT (19EC6DCMPR) REPORT
on

THIRD EYE FOR NAVIGATION ASSISTANCE USING TINYML

Submitted in partial fulfillment of the requirement for the degree of

Bachelor of Engineering
in
Electronics & Communications Engineering - ECE
by

1DS19EC717:	**Hrushikesh H D**
1DS19EC719:	**Kishan R B**
1DS19EC724:	**N K Vinay Gowda**
1DS19EC741:	**Sujith Kumar S**

Under the guidance
of
Dr. S. THENMOZHI
Associate Professor
ECE, DSCE, Bengaluru

Department of Electronics & Communication Engineering
(An Autonomous College affiliated to VTU Belgaum, accredited by NBA & NAAC, Ranked by NIRF)
Shavige Malleshwara Hills, Kumaraswamy Layout,
Bengaluru-560078, Karnataka, India

2021-22

Certificate

Certified that the mini-project work **(19EC6DCMPR)** entitled "**THIRD EYE FOR NAVIGATION ASSISTANCE USING TINYML**" carried out by **Hrushikesh H D** (**1DS19EC717**), **Kishan R B** (**1DS19EC719**), **N K Vinay Gowda** (**1DS19EC724**), **Sujith Kumar S** (**1DS19EC741**) are bonafide students of the ECE Dept. of Dayananda Sagar College of Engineering, Bangalore, Karnataka, India in partial fulfillment for the award of Bachelor of Engineering in Electronics & Communication Engineering of the Visvesvaraya Technological University, Belagavi, Karnataka for the VI Semester course during the academic year 2021-22. It is certified that all corrections/suggestions indicated for the mini-project work have been incorporated in the mini-report submitted to the ECE department. This Mini-Project report has been approved as it satisfies the academic requirement in respect of mini-project work prescribed for the said degree.

Mini-Project Guide Sign : _____

Name: **Dr. S. Thenmozhi**

Mini-Project Section Coordinator Sign : _____

Name: **Prof. Shobha A S**

Mini-Project Convener & Chief Coordinator Sign : _____

Name: **Dr. Shashi Raj K.**

Dr. T.C. Manjunath : _____

HOD, ECE, DSCE

Dr. C.P.S. Prakash : _____

Principal, DSCE

External Mini-Project Viva-Voce (SEE)

Name of the mini-project examiners (int & ext) with date :

1 : Signature : _____

2 : Signature : _____

Declaration

Certified that the mini-project work entitled, "**THIRD EYE FOR NAVIGATION ASSISTANCE USING TINYML**" with the course code **19EC6DCMPR** is a bonafide work that was carried out by ourselves in partial fulfillment for the award of degree of Bachelor of Engineering in Electronics & Communication Engg. of the Visvesvaraya Technological University, Belagavi, Karnataka during the academic year 2021-22 for the VI Semester Autonomous Course. We, the students of the mini-project group/batch no. D1 do hereby declare that the entire mini-project has been done on our own & we have not copied or duplicated any other's work. The results embedded in this mini-project report has not been submitted elsewhere for the award of any type of degree.

Student Name-1 : Mr. Hrushikesh H D.
USN : 1DS19EC717

Sign : _____

Student Name-2 : Mr. Kishan R B.
USN : 1DS19EC719

Sign : _____

Student Name-3 : Mr. N K Vinay Gowda.
USN : 1DS19EC724

Sign : _____

Student Name-4 : Mr. Sujith Kumar S.
USN : 1DS19EC741

Sign : _____

Date : 10/08/2022

Place : Bengaluru -78

Acknowledgment

Presentation, inspiration and motivation have always played a key role in the success of any task. The successful completion of any task would be incomplete without mention of the people who made it possible, whose constant guidance and encouragement crowned the effort with success.

We are grateful to our institution, **DAYANANDA SAGAR COLLEGE OF ENGINEERING** for providing us all the required facilities, which has made the project a success. We are thankful to Dr. Hemachandra Sagar, Chairman, DSCE for providing very good infrastructure and we also extend our gratitude to Dr. Premachandra Sagar, Vice Chairman and Galiswamy, Secretary. We would like to express our gratitude to Dr. C.P.S. Prakash, Principal, DSCE, for providing us amiable environment and surroundings to work.

We would also like to thank our HOD of Electronics and Communication Engineering, Dr. T.C. Manjunath for his support and advice. We would like to extend our gratitude and thanks to our Mini-Project Guide Dr. S. Thenmozhi who helped us in each and every stage of Mini-project with good ideas for the successful completion of the Mini-project. We would like to thank Mini-Project Convener & Chief Coordinator Dr. Shashi Raj K the support and advice throughout. We would like to thank Mini-Project D-Section Coordinator Prof. Shobha A S for managing the schedules and helping out to complete the Mini-project in time without any delay. We are grateful to our teaching and non-teaching faculty for their support, cooperation and help throughout the endeavor.

We would also like to thank our family and friends for their unconditional support and patience throughout the months of the Mini-project.

Abstract

In edge technology devices, TinyML is quickly becoming as the new norm. As it is supplying the edge device with computational power, it aids in eliminating or minimizing the requirement for cloud computing, which aids in lowering power consumption, boosting data security, and lowering privacy hazards.

The most common and commonly used navigation system right now is GPS. It is inaccurate owing to geographical considerations and climatic circumstances taking more energy and requiring powerful computing. It is challenging to integrate GPS in edge devices like AR Glasses for navigation applications because of the aforementioned shortcomings.

We present a methodology and create a prototype in this mini-project, which is then effectively implemented in the edge device known as the Arduino Nano ble 33 Sense. Our major goal is to locate the image using an image recognition model based on TinyML. By training the photos of the path via which navigation must be performed, we first constructed the TinyML model, which is a few kilobytes in size, and then deploy it to the Edge device. Then employ voice commands to help the navigator.

Last but not least, the obtained outcomes demonstrated to us that this TinyML-approach can be used for useful navigation applications. However, there are restrictions and numerous difficulties that are discussed in this mini-project report along with potential solutions.

Keywords: TinyML, Data security, GPS, AR Glasses, Navigator, Edge device.

Table of Contents

List of Figures

List of Tables

Nomenclature and Acronyms

Abbreviations (Alphabetical Order) :

IEEE	Institute of Electrical & Electronics Engineers
DSCE	Dayananda Sagar College of Engineering
ECE	Electronics & Communication Engineering
TinyML	Tiny Machine Learning
GPS	Global Positioning System
AR	Augmented reality
SD	Secure Digital
MP	Mega Pixel
DAC	Digital To Analog Converter
IDE	Integrated Development Environment
I/O	Input/Output
ML	Machine Learning

Chapter-1

Introduction

1.1 Overview

In this mini-project work, we are putting forth an approach to address the issue of adding a navigation support system to edge devices that have limited power and memory. In this work, we will provide a real-world demo using the suggested methods.

GPS is currently the most popular and widely used navigation system. Geographical factors and environmental conditions require more energy and sophisticated computing, which is why it is wrong. Due to the aforementioned drawbacks, integrating GPS in edge devices like AR Glasses for navigation applications is difficult.

In this mini-project, we demonstrate an approach and build a prototype that is subsequently successfully integrated into the edge device known as the Arduino Nano ble 33 Sense. Our main objective is to locate the image using a TinyML-based image recognition model, then to assist him based on the present location.

1.2 Literature survey

[1]. BENCHMARKING TINYML SYSTEMS: CHALLENGES AND DIRECTION:

- This paper is basically stating the different ways of implementing the TinyML model for different applications.
- From this paper we took the help of a section that particularly explains about the which model must be used for image classification which is the main part of our model.
- This paper helped us in choosing the model which is more optimized and even gave the idea of which convolution neural network to use for your application.

[2]. TENSORFLOW LITE MICRO: EMBEDDED MACHINE LEARNING ON TINYML SYSTEMS:

- This paper is the main paper that gave us the way to practically built the machine learning models which can be deployed on the memory constraint embedded hardware devices using TensorFlow lite.
- Here we learned how memory mapping, memory allocation, and multi-threading are done in the edge devices.

- With help of this paper, we referred to how to optimize, quantize, and minimize the feature scope for portability.

[3]. QUANTIZATIONTION-GUIDED TRAINING FOR COMPACT TINYML MODELS:

- In this paper they proposed a methodology to use Mobilenetv1.0 to minimize the need for neurons which helped us in reducing the size of the model in a significant manner.
- The authors in this paper considered the concept of wakeup systems which are the core concepts for TinyML applications and made an attempt to build a model which is more compatible with this concept

1.3 Objectives / Scope / Aim of the mini-project work

To develop a new practical approach to creating a navigation assistance system that is much more effective than the currently popular system (GPS) by addressing the severe issues with increased power consumption and decreased real-time navigation accuracy in the GPS navigation systems [3]. Our goal is to solve the aforementioned issues in order to enable navigation assistance systems in cutting-edge gadgets like AR glasses. And by integrating it into an Arduino nano ble 33 sensing board that is TinyML enabled and testing for validity of the suggested technique, we hope to provide a workable, practical model on a small scale.

1.4 Motivation & Problem Statement

The main reason we undertook this study was to address the issues we were having with the limitations of GPS navigation systems, including their power consumption and lack of accuracy for short-range navigation. So, we were always looking for a solution to this issue. The TinyML paradigm [1] now enables the computational power for edge technology devices. In order to use TinyML to address the aforementioned issues, we proposed the following methodology.

1.5 Existing & Proposed (Developed) Mini-Project module

A system is GPS. Satellites, ground stations, and receivers make up its three components.

Satellites behave like the stars in constellations; we always know where they should be.

Radar is used by the ground stations to confirm that they are where we think they are.

A receiver, similar to the one in your phone or your parent's car, is always looking for a signal from these satellites. The receiver calculates their distance from some of them.

The receiver determines your location after calculating your separation from four or more satellites.

As it needs constant interactions from the satellites this system demands high power and if the geographical terrain is having tall structure this system fails to provide accurate results.

1.6 Proposed Methodology

The process used involves initially gathering image data from the user's current navigation path as part of training. After which adding appropriate labels to train the image classification model. The model should then be converted to TensorFlow lite [4] so that it may be used on an edge device. To further minimize the size of the model and also to ensure compatibility, optimization and quantization must be done prior to deployment.

Once the model has been built, it can then be inferred using the deployed device through actual testing. once satisfactory results have been attained. The next step is to map the visuals that have been trained so that the device may use them to determine where the navigator is right now and what to do next to get to the destination.

The location-mapping device, camera, speakers, and edge device with TinyML support are all necessary for this methodology proposed.

1.7 Organization of the mini-project report

The mini-project work undertaken is organized in the following sequence as follows.

A brief introduction to the mini-project work was presented in the introductory chapter in chapter-1. The Block diagram and working principle of mini-project work undertaken by us is presented in chapter 2. Hardware/ Software tools /Description/Interfacing

employed in our mini-project work is depicted in chapter – 3. Results and discussions are shown and explained in detail in chapter 4. Application, advantages, limitations and outcomes are listed in chapter 5. Finally, the mini-project report concludes with conclusion & future work in chapter –6.

Appendices, Data Sheets, Program codes, photos are presented at the end of the mini-project report one after the other in succession.

Chapter 2

Block diagram, Circuit Diagrams, Working principles, Algorithms and Flow-Charts

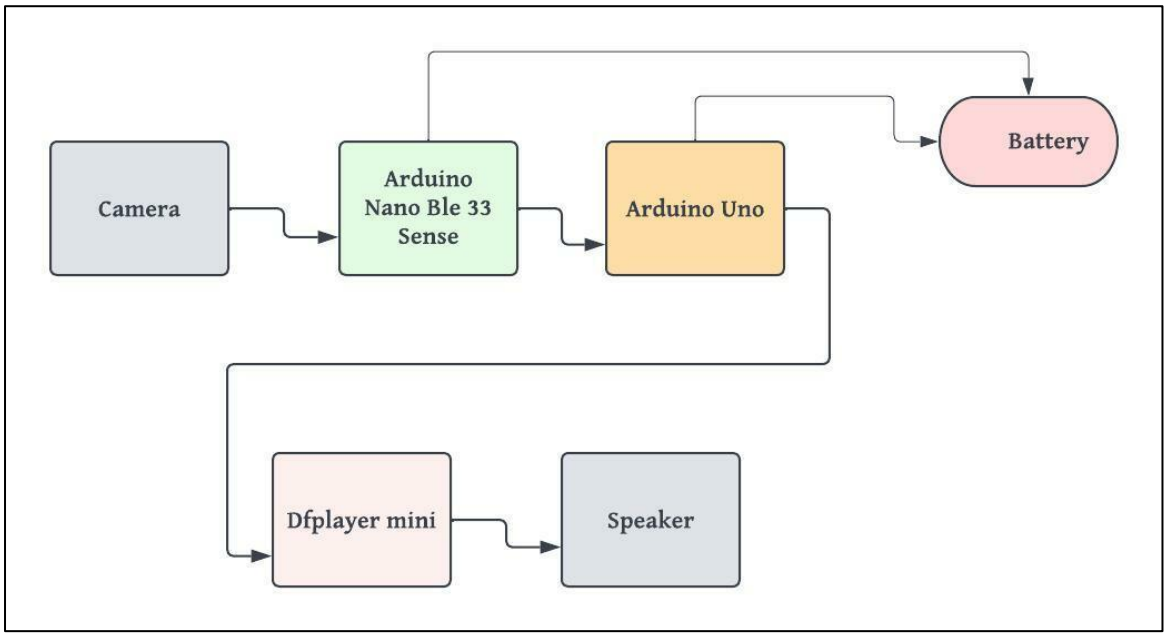

Figure 1:Block diagram of a Proposed methodology

Block-1 in Figure.1 shows the components in an abstract manner. This shows the connection and the arrangement of components. First, the OV7675 camera is connected to the Arduino nano ble 33 sense then it is connected to Arduino UNO for playing required voice assistance. Then DFplayer mini is used to control the operation of the speaker. The Arduino UNO and Arduino nano ble 33 sense need power that is connected to the 9V Battery.

The overall circuit diagram of the proposed undergraduate mini-project work is shown in Figure.2. As you can see it contains the camera OV7675 first which is having 20 pins. In 20 pins the 18 pins are connected to the Arduino nano ble 33 sense as per the pin connection shown in Figure.3. Then the 3 digital pins are connected to the Arduino UNO from Arduino nano ble 33 sense for communication to know the mapping. Then TX and RX pin of Arduino UNO is connected to the DFplayer mini. And DFplayer mini is connected to the speaker through two cables. And an SD card is used to store the audio

output which will be inserted in the DFplayer mini. The power of this Arduino boards will be provided by the USB cable while demonstrating.

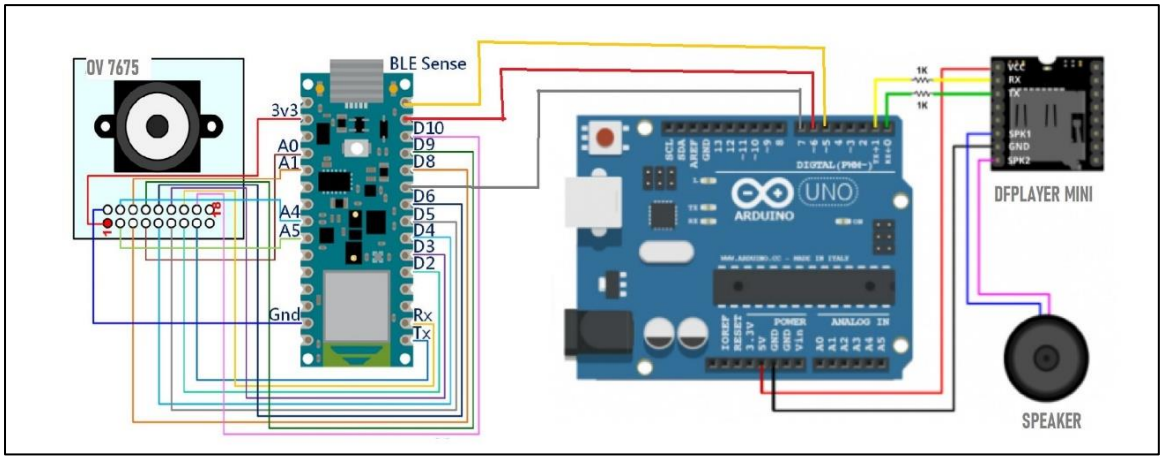

Figure 2:Circuit Diagram

Description	Camera Module Pin	Microcontroller Board Pin
VCC / 3.3V	1	3.3V
GND	2	GND
SIOC / SCL	3	SCL / A5
SIOD / SDA	4	SDA / A4
VSYNC / VS	5	D8
HREF / HS	6	A1
PCLK	7	A0
XCLK	8	D9
D7	9	D4
D6	10	D6
D5	11	D5
D4	12	D3
D3	13	D2
D2	14	D0/RX
D1 (may be labeled D0)	15	D1/TX
D0 (may be labeled D1)[1]	16	D10
NC	17	--
NC	18	--
PEN / RST	19	A2
PWDN / PDN	20	A3

Figure 3: Pin connection of OV7675

Algorithm:

Input

The images of the location where the navigator is present now from the camera OV7675.

Expected Outputs

Assistance audio outputs for navigator through speaker for the location identified by ML model.

For training the Model:

1. Start by collecting the images of the path that needs to be assisted.
2. Preparing the dataset by labeling the images and splitting the images [8] for training, testing, and validation.
3. Train the model by using MobileNet V1.0 convolution neural networks.
4. Then convert the TensorFlow model to the TensorFlow lite model.
5. Then optimize the model to 8-bit integers [14] to run on an embedded system.
6. Deploy the model to the edge device.

Working after deploying the model:

1. Start when the button is pressed initialization of the camera begins.
2. Capturing the image and converting it into a serial buffer for comparing with the model trained.
3. When the inferencing of the image is completed, the result is obtained.
4. Then the label of the detected image is taken.
5. Else the image of this location is unknown.
6. Then Arduino Nano ble 33 sense calls Arduino UNO to perform the voice assistance task.
7. The Arduino UNO for different labels [8] knows which audio file to output it performs that task with help of DFplayer mini.
8. End of program.

Working Principle: The overall working principle of the mini-project work takes place in two steps

1. TRAINING THE MODEL:

The first stage of this process consists of identifying the path in which the navigator needs to travel. Then consider the key location in the path where the general navigators need help like crossroads, junctions, and circles. The above-mentioned places are where the navigator is confused about choosing which path to take to reach his destination. As it is mentioned in the above stage after choosing the above key location then collecting the images of that location to train the model to accurately detect that location and then assist precisely. Then the next step consists of preparing the dataset, which is correctly segmenting the images by labeling them, resizing, rescaling, and then dividing the data into training and testing sets for validation. After the dataset is ready, we need to choose the convolution neural networks for training our dataset which will provide us more accurate model.

Here to train our model we are using MobileNet v1.0 [5] convolution neural networks. The image dataset size which is used to train the model is 96*96 and the color of the images is grayscale. This is specially developed for developing models for mobile and edge microcontroller devices which are provided with less computation power capability. Then the next step involves converting the model trained which is built using the TensorFlow framework to the TensorFlow lite [12] version model which will make the model compatible with the edge computing devices. After converting the model to .tflite format next step is to quantize the model to int8 format which is an important process because it reduces the model size to 114 Kbytes which is necessary to deploy it on the microcontroller.

Then the next steps involve testing the model prepared with the testing dataset considered initially and after obtaining the results, checking whether the model is satisfying our needs, and [10] deciding whether to deploy this model or not.

When the model satisfies our needs, the next step is to deploy the model to the microcontroller i.e., Arduino nano ble 33 sense.

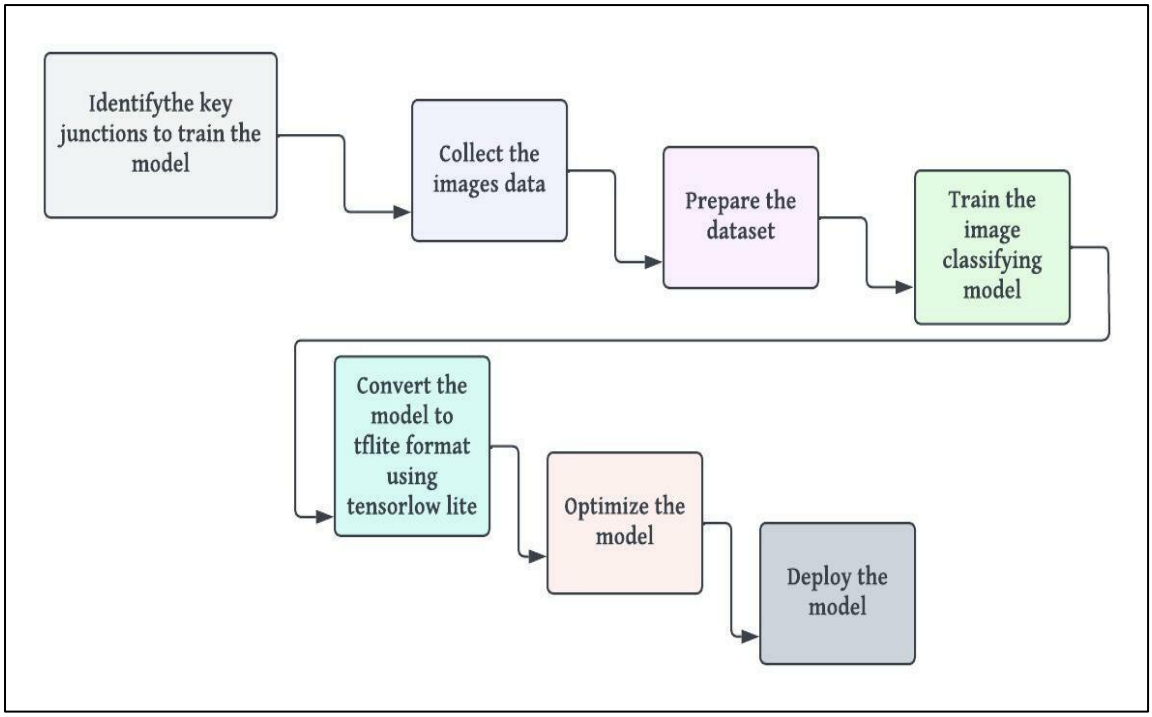

Figure 4: Flowchart of training the model

2. WORKING AFTER DEPLOYING THE MODEL:

At first when the user presses the button for help to assist the user on which step to take further to reach his destination. As soon as the button is pressed the camera gets initialized and starts capturing the image. The image captured by the camera will be of the dimension 96*96 and the color of the image will be grayscale. Then the image captured by the camera is converted into the static buffer of int8 quantization and then it is compared with the model that is deployed into the microcontroller.

After the inferencing of the image captured is done then the results are obtained on a scale of 1. And if the inference result of the label is more than 0.9 then the image is classified as that label. After the recognition of the image is done the next task is to assist the user for help. When the image is recognized, the data is sent to Arduino UNO which will be having the program recognize which audio file to play for different labels.

Then the audio file will be stored in the SD card that is placed in the DFplayer mini.

The Arduino UNO commands the DFplayer mini to play the respective audio file then the DFplayer will play the audio file using the speaker connected to the DFplayer mini. If the image is not recognized the command that the location is not recognized will be played. And after playing this command the microcontroller will wait for the button to be pressed. End of working.

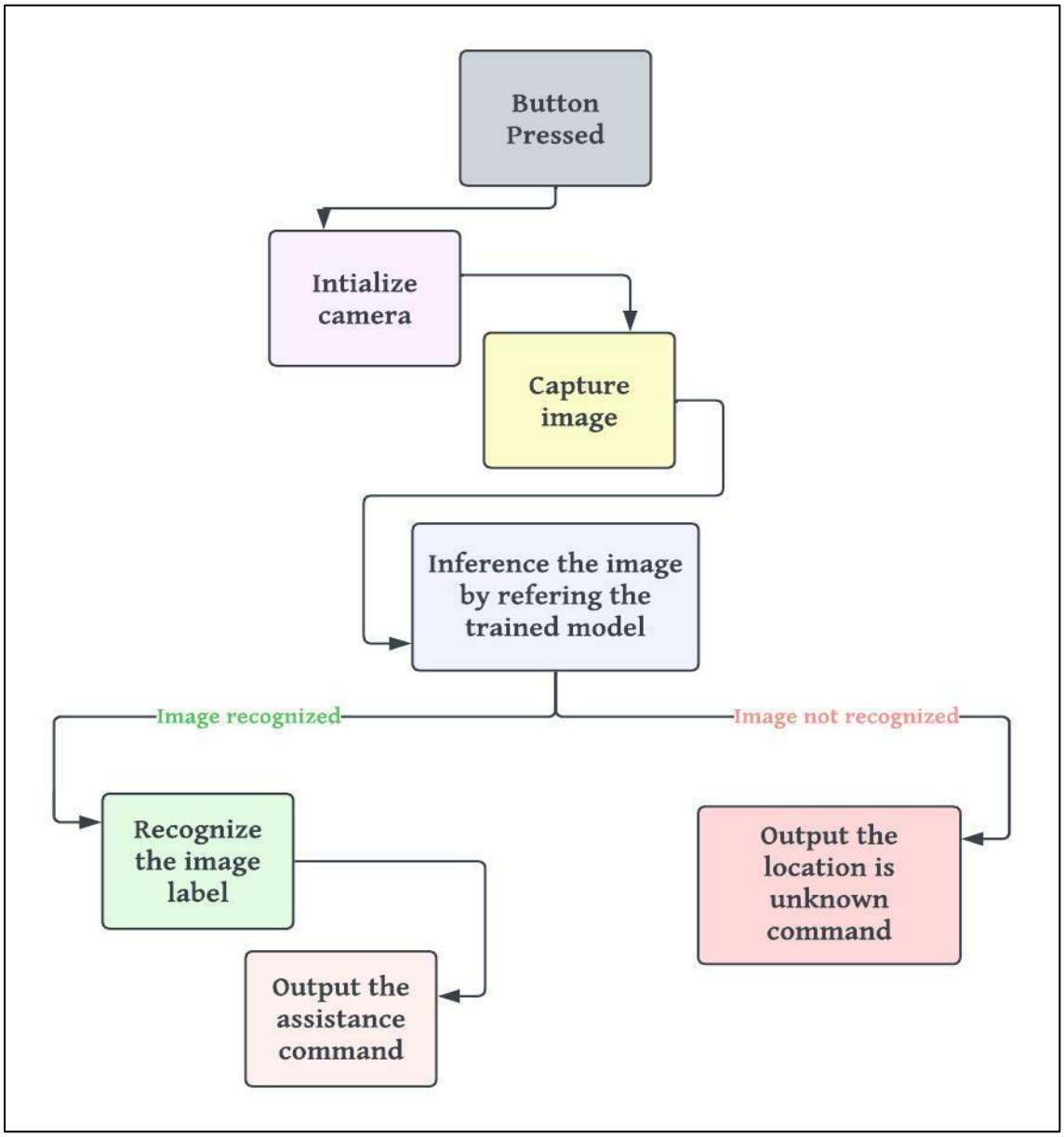

Figure 5:Flowchart of working

Chapter-3

Hardware / Software tools / Description / Interfacing / Working of the complete mini-project module

Hardware:

1. Microcontroller: ARDUINO NANO BLE 33 SENSE:

Table 1: Arduino nano ble 33 sense description [16]

Board	**Name**	Arduino® Nano 33 BLE Sense
	SKU	ABX00031
Microcontroller	nRF52840	

Clock speed	**Processor**	nRF52840 64MHz
Memory	**nRF52840**	256 KB SRAM, 1MB flash
Power	**I/O voltage**	3.3V

This microcontroller is used for image classification this is an edge computing device that is used to build our prototype.

2. Camera: Arducam OV7675 (0.3MP):

- Active array size: 640×480
- Power supply: Analog: 2.6 ~ 3.0v, core: 1.5v dc + 5% (internal regulator), I/O: 1.71 ~ 3.0v.
- Power requirements: active: 98 mW, standby: 60 µW.

This camera is used to provide the image to the Arduino nano ble 33 sense for image classification.

3. **Microcontroller: Arduino UNO R3:**

This microcontroller is used to control the audio commands after the image is recognized since the Arduino nano ble 33 sense cannot simultaneously do both the task this board is used to do that task.

Table 2: Arduino UNO description [17]

Board	**Name**	**Arduino UNO R3**
	SKU	**A000066**
Microcontroller	**ATmega328P**	
Memory	**ATmega328P**	2KB SRAM, 32KB FLASH, 1KB EEPROM

4. **DFplayer Mini :**

- Supported sampling rates (kHz): 8/11.025/12/16/22.05/24/32/44.1/48
- 24 -bit DAC output, support for dynamic range 90dB, SNR support 85db
- Fully supports FAT16, FAT32 file system, maximum support 32G of the TF card, support 32G of U disk, 64M bytes NOR FLASH. [18]

A compact and reasonably priced MP3 module with a streamlined output that goes directly to the speaker is the DFPlayer Mini MP3 Player for Arduino. This is used to play the audio commands which will be stored on an SD card.

5. **SD card:**

The 2GB SD card is used in this prototype for storing audio commands on the SD card.

6. **Speakers:**

Here in this prototype, a 0.5-Watt 8-ohms speaker is used for playing the audio command.

7. Jumper wires:

These jumper wires are used to connect different components in this prototype.

Software:

1. Arduino IDE:

It is simple to write code and upload it to the board using the free and open-source Arduino Software (IDE). Any Arduino board can be used with this software. We used this software for writing the code.

2. Jupiter Notebook:

The most recent web-based interactive development environment for code, data, and notebooks is Jupiter Lab. We used this editor build of Machine learning model to train and test the dataset.

3. TensorFlow:

Machine learning and creating neural networks are made quicker and simpler with TensorFlow, an open-source toolkit for numerical computation that is compatible with Python.

We Used This Machine Learning framework for building our Machine Learning model and we converted it into a TensorFlow lite file using this framework.

4. MobileNet V 1.0 [5]:

Convolutional neural networks, such as MobileNet, are specialized for use in embedded and mobile vision applications. They are built using depth-wise separable convolutions, which are lightweight deep neural networks that can have minimal latency for embedded and mobile devices.

We used this convolution network to build a model that is more compatible with edge devices like microcontrollers.

5. Impulse edge:

Edge Impulse is the top machine learning development platform for edge devices. It helps us to build a more efficient model by providing a variety of choices to build the

same model with different convolution neural networks. And by the help of this website, we can test your model using mobile phones to test for validation.

6. **Programming Languages:**

   a. **Python:**

      Python is a general-purpose, high-level, interpreted programming language. With the usage of extensive indentation, its design philosophy places an emphasis on code readability. Python uses garbage collection and dynamic typing. Python language is used to build our Machine Learning models.

   b. **Embedded C:**

      The C Standards Committee developed a set of language extensions called Embedded C to address compatibility problems between C extensions for various embedded devices. Supporting advanced microprocessor features like fixed-point arithmetic, several different memory banks, and fundamental I/O operations often necessitate nonstandard additions to the C language. Embedded C is used to program the Arduino for obtaining desired work.

Working :

1. **Training the Machine learning Model:**

   Identifying the route that the navigator needs to take is the first step in this process. Then take into account the critical points along the route where general navigators require assistance, such as intersections, junctions, and circles. The spots indicated above are where the navigator is unsure of the route to follow in order to get to his objective. As was said in the stage above, after selecting the aforementioned key site, photos of that location were collected as shown in Figure.5 in order to train the model to precisely detect that position and provide assistance.

   After accurately segmenting the photos by labeling them, resizing, and scaling them, the dataset must then be prepared. The data must then be split into training and testing sets for validation. Once the dataset is ready, we should use convolution neural networks to train on it because they will provide models that are more accurate. In this case, we are using MobileNet v1.0 [5] convolution neural networks to train our model. The 96*96 picture dataset is utilized to train the model, and the

images are grayscale in color. This was created specifically for the development of models for mobile and edge microcontroller devices, which have little computing capacity.

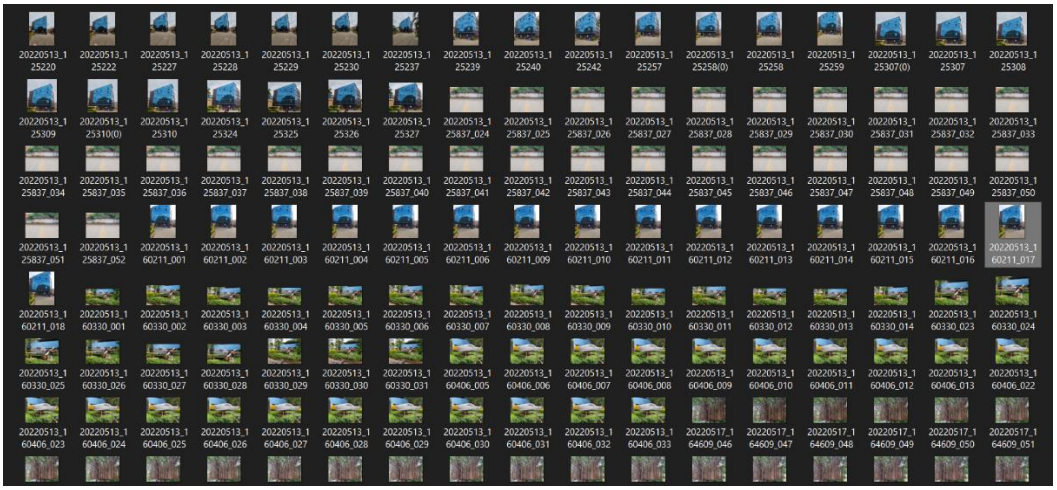

Figure 6:Images dataset collected

The model that was trained and constructed using the TensorFlow framework will then be converted to the TensorFlow light version model, making it compatible with edge computing devices. The next step after converting the model to.tflite format is quantization, which is crucial since it decreases the model's size to 114 Kbytes, which is required for deployment on the microcontroller.

The following phases comprise testing the prepared model with the testing dataset taken into consideration earlier, evaluating the results to see if the model is meeting our needs, and determining whether or not to deploy this model. When the model meets our requirements, the Arduino nano ble 33 sense microcontroller is used as the model's deployment platform.

2. **Working after deploying the model:**

When the user first presses the assistance button, the user will receive guidance on the next steps to follow in order to go to his destination. The moment the button is pressed, the camera is initialized and begins taking pictures. The camera will record an image with a resolution of 96*96 pixels and grayscale color. Next, the camera-captured image is transformed into an int8 static buffer for comparison with a model that has been programmed into the microcontroller. The outcomes are measured on a scale of 1

following the inference of the collected image. And if the label's inference result is greater than 0.9, the image will appear.

After the image has been correctly identified, the user will need assistance. When an image is detected, the information is transferred to an Arduino UNO, where the application will determine which audio track to play for each label. Following that, the audio file will be saved to the SD card that has been inserted into the DFplayer mini. The DFplayer mini will play the specified audio file via the DFplayer mini's speaker after receiving an order to do so from the Arduino UNO. The command that the location is not identified will be played if the image is not recognized. Additionally, the microcontroller will wait for the button to be pressed after playing this command.

Chapter-4

Results and Discussions

SIMULATION RESULTS OF THE TINYML MODEL BUILT:

In Figure.7 is the features vector graph and it shows the plot of the different labeled datasets, and the nearby clusters inform us of the similar type of images. This reference is used while classifying the image. The captured image is converted into a serial buffer and plotted in the graph below. Then it classifies the image based on which cluster is near the image captured.

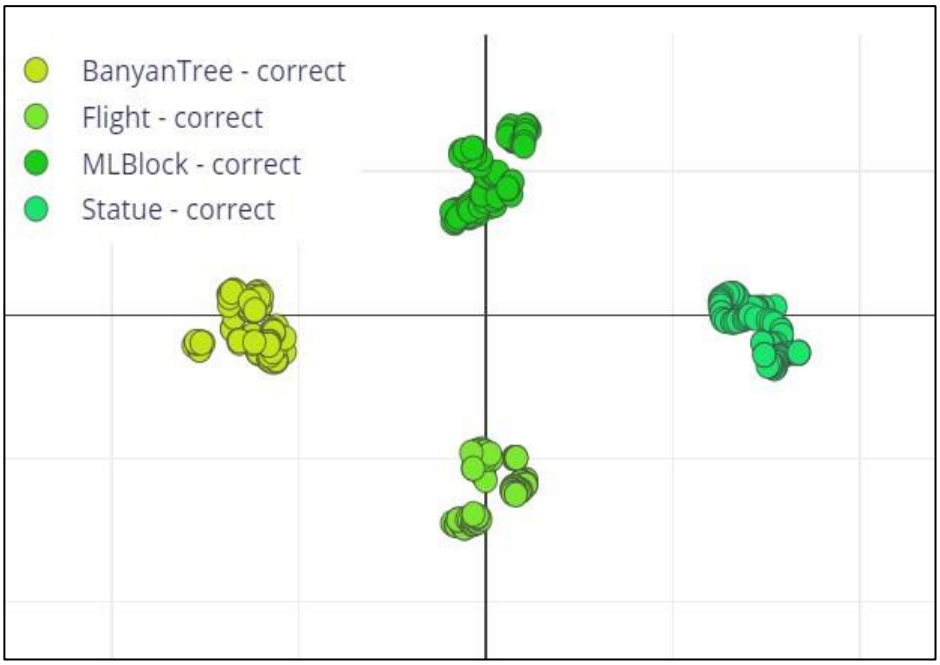

Figure 7:Feature vector graph after training the dataset [9]

This Figure.7 shows the graph (X-axis represents visualization layer 1 and Y-axis represents visualization layer 2) of the feature vector which is extracted when training the model using the training dataset. This is the clearly classified feature that is more efficient than all the combinations we have trained.

The vector graph shows the nodes and their distance between different labels in the dataset which help us to clearly verify the model for its testability. Intern it tells us to check the model

Figure.8 shows the accuracy of the Machine Learning model built against the testing dataset. Here in this stage the TF model is converted into TFlite model and int8 quantization is done.

This is just the accuracy against the testing dataset but the actual accuracy of the model changes when that is deployed on the edge device for performing real-time image classification.

Figure 8:The results obtained by verification using the testing dataset [9]

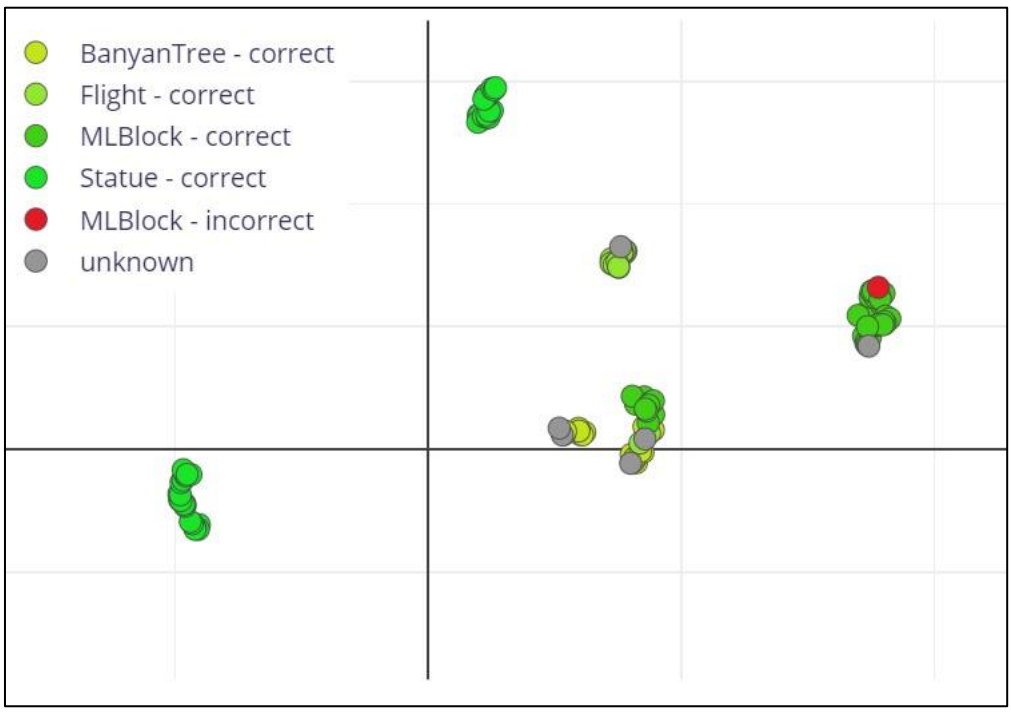

Figure 9:Feature vector graph after testing against a testing dataset [9]

Figure.9 shows the feature vector graph after the model was tested against the testing dataset. This shows where the model is not able to recognize the image and shows the false output.

Figure.10 shows the result and the size of the model after the model is quantized which is the important process needed before deploying the model into the microcontroller. Here for Arduino nano ble 33 sense, we are using int8 quantization since the memory capacity is only 1Mb. Figure.8 also shows the accuracy of the model after the quantized model is tested against the testing dataset.

Figure.10 gives us the picture of how much memory is needed while deploying the model and the accuracy that may be obtained after uploading the images. There are two ways of quantizing i.e. int8 and float32. The float32 is more accurate but takes up lot of memory. Hence it is best to use int8 quantization.

Figure 10:The size and results of the model after it is quantized(int8) [9]

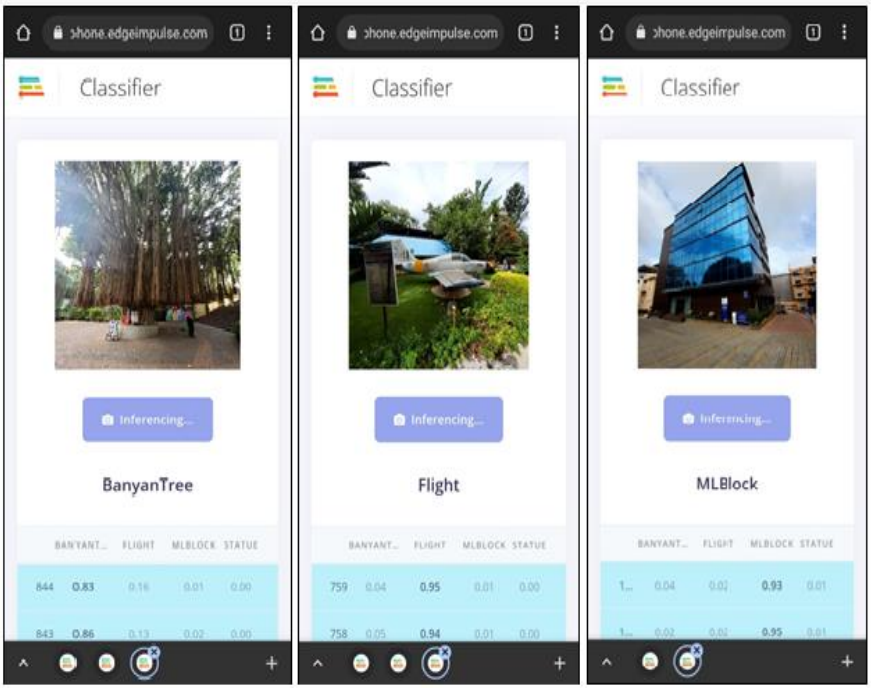

Figure 11:Inference screenshots from edge impulse

Figure.11 shows the output obtained after training the model and inferencing using the smart mobile. By the above inferencing, we obtained the desired output.

OUTPUT OBTAINED AFTER DEPLOYING THE MODEL:

Figure 12: Testing the model at banyan tree

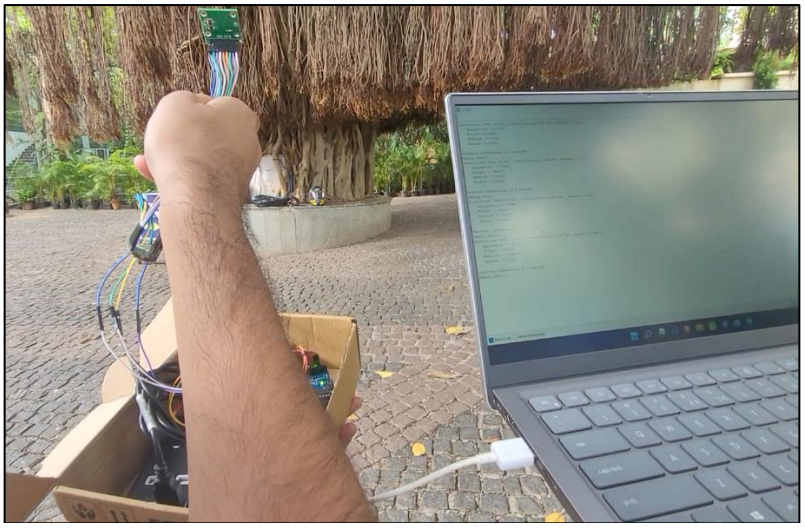

Figure 13: testing the model at banyan tree 2

COM7

```
Predictions (DSP: 10 ms., Classification: 304 ms., Anomaly: 0 ms.):
    BanyanTree: 0.90234
    Flight: 0.03125
    MLBlock: 0.04297
    Statue: 0.02344

Starting inferencing in 2 seconds...
Taking photo...
Predictions (DSP: 11 ms., Classification: 304 ms., Anomaly: 0 ms.):
    BanyanTree: 0.72656
    Flight: 0.05078
    MLBlock: 0.16016
    Statue: 0.06250

Starting inferencing in 2 seconds...
Taking photo...
Predictions (DSP: 11 ms., Classification: 304 ms., Anomaly: 0 ms.):
    BanyanTree: 0.77734
    Flight: 0.04688
    MLBlock: 0.11719
    Statue: 0.05469

Starting inferencing in 2 seconds...
Taking photo...
Predictions (DSP: 11 ms., Classification: 304 ms., Anomaly: 0 ms.):
    BanyanTree: 0.83594
    Flight: 0.03906
    MLBlock: 0.08203
    Statue: 0.04297

Starting inferencing in 2 seconds...
Taking photo...
Predictions (DSP: 11 ms., Classification: 304 ms., Anomaly: 0 ms.):
    BanyanTree: 0.72266
    Flight: 0.05078
    MLBlock: 0.16016
    Statue: 0.06641
 maxlabel =  0
```

Autoscroll Show timestamp

Figure 14: Output in the serial monitor

Figure.14 shows the output obtained when practically testing the prototype developed in front of the banyan tree which we trained to the model and obtained the results of accurately recognizing the image is around 85-95% and it accurately detected the image and played the audio file that will be stored on the SD card.

Since there was an error in the vector graph while testing the testing set. Then it was tested using the edge device and obtained the results as in Figure.15.

Figure 15:ML Building output

This shows the validity of the TinyML model in real-time image classification.

POWER CONSUMPTION COMPARISON:

Table 3: Power consumption comparison

Modules	Voltage(V)	Current(mA)	Power(mW)
GPS-9540	**3.3**	**170**	**561.2**
The prototype developed by us	**3.3**	**10-15**	**33-46.5**

Table 3. shows the power consumption of the GPS-9450 module and the prototype we built. It proves that we can save almost 12 times the power consumed by the prototype built.

This is obtained as the stated objective.

Chapter-5

Applications, Advantages, Outcome, and Limitations

APPLICATION:

There is the innumerous application of this methodology proposed by us

1. This technology can be employed for power-constrained VR and AR glasses that have navigation aid systems.
2. By making a few adjustments in accordance with the needs for autonomous driving and monitoring, this technology can be applied to drones.
3. These prototypes can also be utilized to detect anomalies in nuclear power plants, mining operations, and other settings where human monitoring is challenging.
4. Farmers can use the application from this prototype to photograph a plant and determine whether it has any illnesses. No internet connection is necessary for it to function on any device. It is essential for rural farmers and enables the preservation of agricultural interests.
5. This prototype can be modified and adjusted before being used in self-driving automobiles to increase their accuracy in anticipating the terrains even if the GPS fails.
6. In hospitals, this technology can be used to monitor patients affordably and without the usage of the internet.
7. This prototype can be utilized for object recognition in edge devices like smartwatches and augmented reality goggles, among others.
8. By categorizing the various products into distinct groups and tracking the flaws in the manufactured goods, these methodologies can be applied in smart manufacturing.

ADVANTAGES:

1. Low Power Consumption: Microcontrollers use relatively little power, as we've already discussed in RESULTS. They can run for a very long period without being charged because of this.
2. Low Bandwidth: Less internet bandwidth is required because the data is not constantly transferred to the server.
3. The data is not kept on any servers because the model is edge-based. So privacy is guaranteed.
4. Low Latency: Because the model is edge-based, inference can be performed without sending data to a server. This lowers the output's latency.

LIMITATIONS:

1. **Only has kilobytes or megabytes of memory:**

   TinyML devices. The size and duration of the machine learning models used on these devices are thus constrained. There are now just a few machine learning frameworks that can fulfill the needs of TinyML devices. One such framework is TensorFlow Lite.

2. **Hardware limitation:**

   The hardware also needs to be adjusted if we want to develop the prototype for the application. Because it is complicated and requires more money to design hardware for a specific purpose, it is not simple to perform.

3. **Troubleshooting:**

   Unlike in a cloud environment, where troubleshooting can be done remotely, since the Machine Learning model trains on the data that the device collects and runs on the device itself, it is more difficult to identify and resolve performance issues.

4. **Training limitations:**

   The path in which the navigator has to travel must be trained which takes a lot of memory and time and is a constraint in the model we built.

OUTCOMES:

We created a prototype for the mini-project that serves as a clear example of how to apply the fresh options to address the navigation system's current issue. It can be further expanded to several fields to address a wide range of issues. We showed how TinyML technology can be used in the navigation system to solve issues with power consumption and accuracy that we are now seeing with GPS systems.

We can design a practical working model for immediate implementation in the AR glasses and in edge technology devices by building application-specific hardware for this application utilizing the aforesaid methods.

The hardware was the biggest obstacle to creating a reliable prototype; however, by addressing this issue, it is now possible to create a product that will benefit society.

Chapter-6

Conclusions and Future Work

In this work, we attempted to develop a novel method for addressing the issues with the current navigation assistance system (GPS).

We provide a technique that makes use of TinyML [10] technology to address the issues of high-power consumption and accuracy in various geographic terrains. And in practice by applying the aforementioned methods to the prototype created using an Arduino nano ble 33 sense and obtaining the findings displayed in results in chapter 4. The developed device used three times less energy than the current GPS variant. And which can be utilized in remote locations where connectivity issues are common, and which do not require cloud computing.

The outcomes we got are in line with the objective we set, but there are several restrictions that make it difficult to use this methodology in practice. Although it is a laborious task, there are a few techniques to manually train the photos of the path we must walk through in order to practically accomplish this. And developing application-specific hardware for this application with the appropriate specifications to turn the aforementioned prototype—which could be the subject of future work—into a usable, practical model.

References

[1]. Colby R. Banbury, Vijay Janapa Reddi,Max Lam, William Fu, Amin Fazel, Jeremy Holleman, Xinyuan Huang, Robert Hurtado, David Kanter, Anton Lokhmotov, David Patterson,Danilo Pau, Jae-sun Seo, Jeff Sieracki , Urmish Thakker, Marian Verhelst , Poonam Yadav, "BENCHMARKING TINYML SYSTEMS: CHALLENGES AND DIRECTION", arXiv:2003.04821v4 [cs.PF], 29 Jan 2021

[2]. Robert David, Jared Duke, Advait Jain, Vijay Janapa Reddi, Nat Jeffries, Jian Li, Nick Kreeger, Ian Nappier, Meghna Natraj, Shlomi Regev, Rocky Rhodes, Tiezhen Wang, Pete Warden, "TENSORFLOW LITE MICRO: EMBEDDED MACHINE LEARNING ON TINYML SYSTEMS", *arXiv:2010.08678v2*, 20 Oct 2020.

[3]. Sedigh Ghamari, Koray Ozcan, Thu Dinh, Andrey Melnikov, Juan Carvajal, Jan Ernst, Sek Chai Latent AI, "Quantization-Guided Training for Compact TinyML Models", *arXiv:2103.06231v1*, 10 Mar 2021.

[4]. Altun, K., Barshan, B., and Tunc¸el, O. Comparative study on classifying human activities with miniature inertial and magnetic sensors. Pattern Recognition, 43:3605–3620, 10 2010. doi: 10.1016/j.patcog.2010.04.019.

[5]. Amir, A., Taba, B., Berg, D., Melano, T., McKinstry, J., Nolfo, C. D., Nayak, T., Andreopoulos, A., Garreau, G., Mendoza, M., Kusnitz, J., Debole, M., Esser, S., Delbruck, T., Flickner, M., and Modha, D. A low power, fully event-based gesture recognition system. In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 7388–7397, July 2017. doi: 10.1109/CVPR.2017.781.

[6]. O. D. Lara and M. A. Labrador, "A survey on human activity recognition using wearable sensors," IEEE Communications Surveys Tutorials, vol. 15, no. 3, pp. 1192–1209, Third 2013.

[7]. Banbury, C. R., Reddi, V. J., Lam, M., Fu, W., Fazel, A., Holleman, J., Huang, X., Hurtado, R., Kanter, D., Lokhmotov, A., et al. Benchmarking tinyml systems: Challenges and direction. arXiv preprint arXiv:2003.04821, 2020.

[8]. Garey, M. R., Graham, R. L., and Ullman, J. D. Worst-case analysis of memory allocation algorithms. In Proceedings of the fourth annual ACM symposium on Theory of computing, pp. 143–150, 1972.

[9]. Krishnamoorthi, R. Quantizing deep convolutional networks for efficient inference: A whitepaper. arXiv preprint arXiv:1806.08342, 2018

[10]. Lin, J., Chen, W.-M., Lin, Y., Cohn, J., Gan, C., and Han, S. Mcunet: Tiny deep learning on IoT devices. arXiv preprint arXiv:2007.10319, 2020.

[11]. Susto, G. A., Schirru, A., Pampuri, S., McLoone, S., and Beghi, A. Machine learning for predictive maintenance: A multiple classifier approach. IEEE Transactions on Industrial Informatics, 11(3):812–820, 2014.

[12]. Zhang, Y., Suda, N., Lai, L., and Chandra, V. Hello edge: Keyword spotting on microcontrollers. arXiv preprint arXiv:1711.07128, 2017.

[13]. https://axleaddict.com/safety/Disadvantages-of-GPS

[14]. https://www.tensorflow.org/lite

[15]. https://towardsdatascience.com/review-mobilenetv1-depthwise-separable-convolution-light-weight-model-a382df364b69

[16]. https://store.arduino.cc/products/arduino-nano-33-ble

[17]. https://store.arduino.cc/products/arduino-uno-rev3

[18]. https://wiki.dfrobot.com/DFPlayer_Mini_SKU_DFR0299

[19]. https://studio.edgeimpulse.com/studio/114525/learning/keras-transfer-image/57

[20]. TensorFlow. TensorFlow Lite FlatBuffer Model, 2020a. URL https://www.tensorflow.org/lite/ api docs/cc/class/tflite/flat-buffermodel.

[21]. Pete Warden, Daniel Situnayake, "TinyML", O'Reilly Media, Inc., ISBN: 9781492052043, December/2019.

Appendix

1. [Arduino Nano ble 33 sense datasheet.]

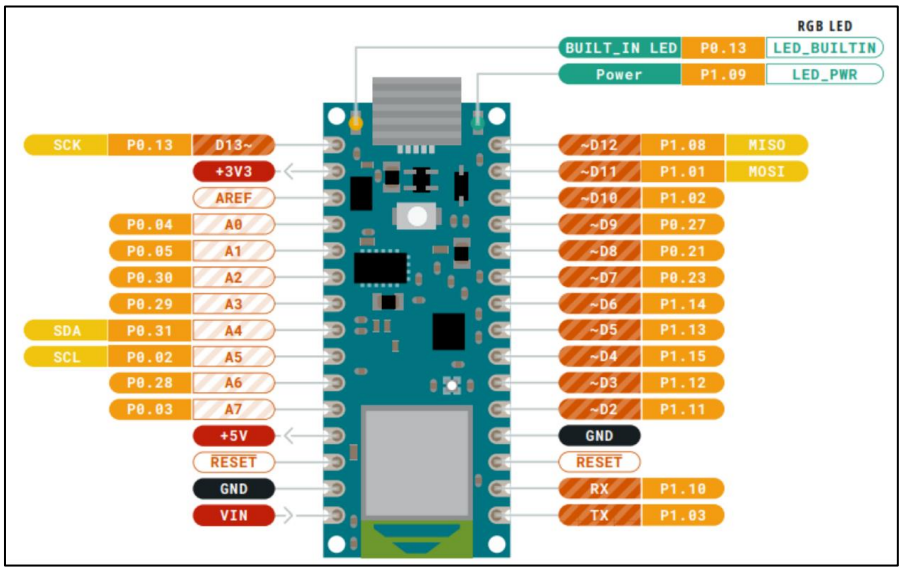

Figure 16:Arduino nano ble 33 sense pinout

2. [Arduino UNO R3 datasheet]

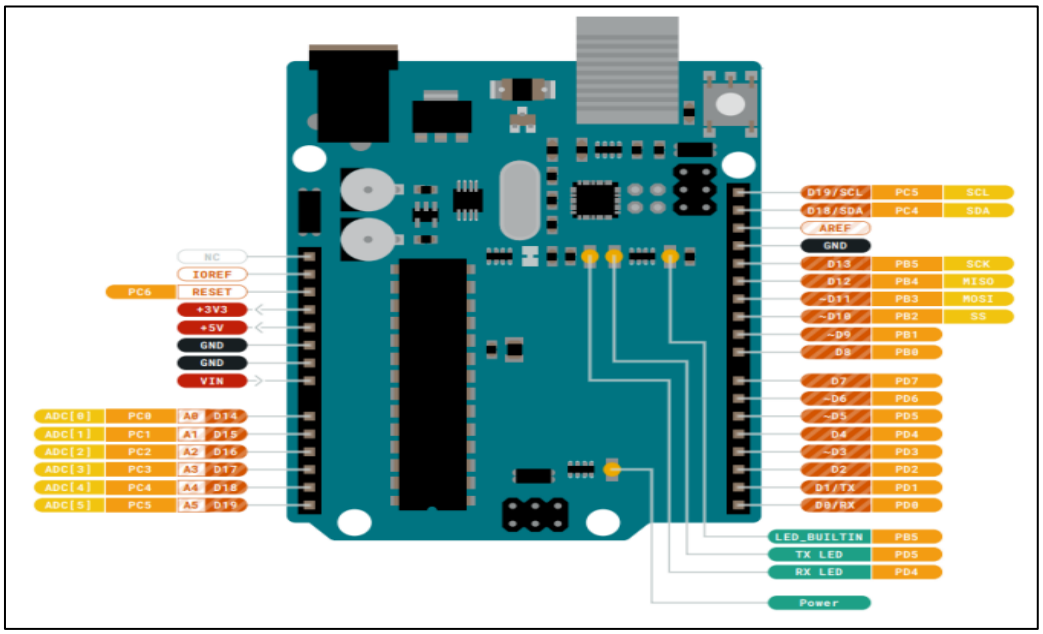

Figure 17:Arduino UNO R3 pinout

3. [DFplayer mini datasheet]

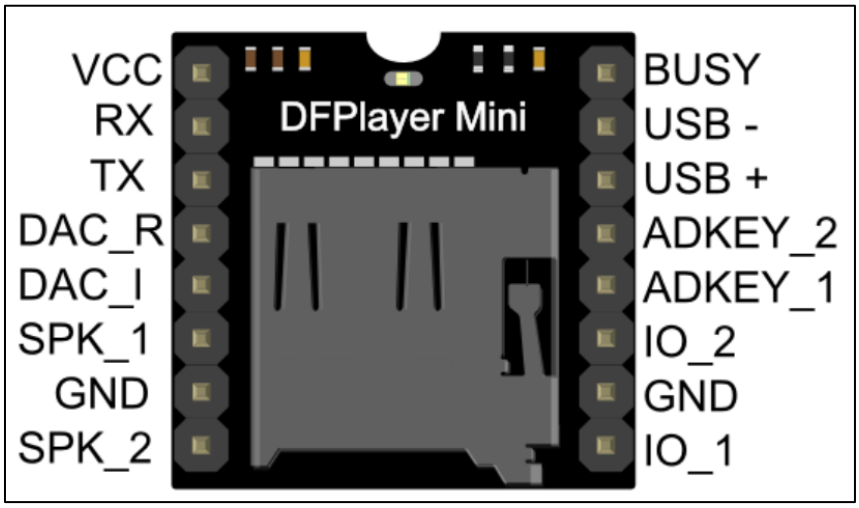

Figure 18:DFplayer mini pinout

4. [OV7675 CAMERA Datasheet]

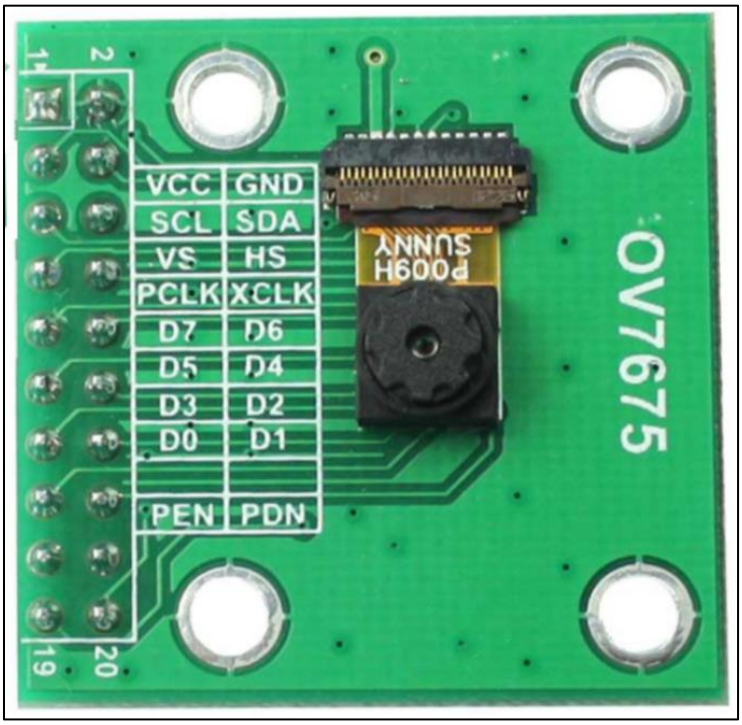

Figure 19:OV7675 camera

Photographs

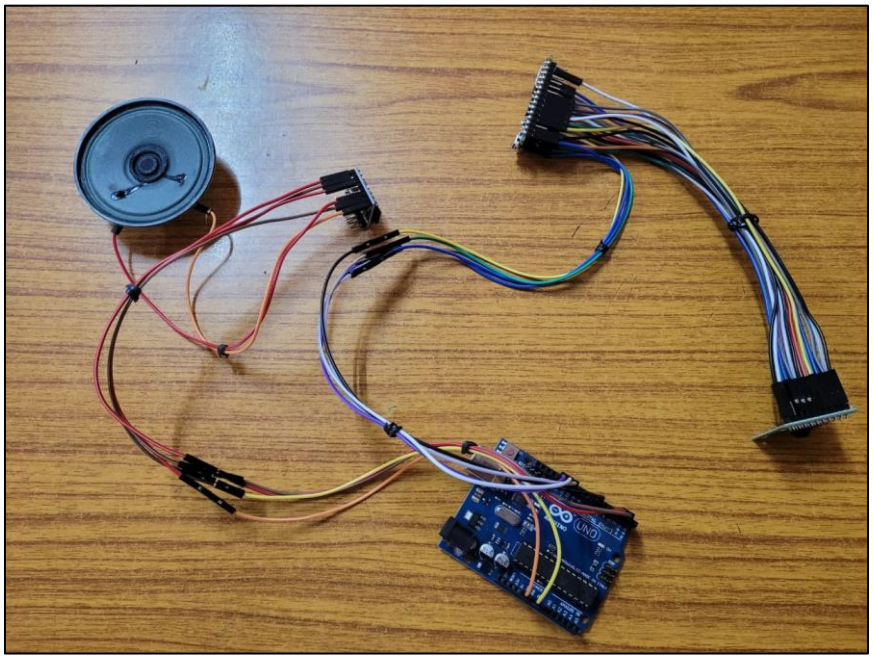

Figure 20:The circuit built 1

Figure 21: Testing the prototype built