

**THE TECHNICAL UNIVERSITY OF KENYA**

**FACULTY OF ENGINEERING AND THE BUILT ENVIRONMENT**

**SCHOOL OF ELECTRICAL & ELECTRONIC ENGINEERING**

**Project Report**

**(Artificial Intelligence Assisted Navigation Device for The Blind)**

**KELVIN KARANI NJOROGE**

**EEEQ/01703/2019**

**EEEQ591 TECHNICAL PROJECT REPORT**

**Course Project Coordinator**

*DR MARY AHUNA*

**Course Project Supervisor**

*MR THOMAS NYAJOWI*

**Bachelor of Engineering**

This project proposal is submitted to the School of Electrical and Electronics Engineering in partial fulfillment of the requirements for the award of the degree of Bachelor of Engineering

**@2024**

# Declaration

This project report is my original work and has not been presented in any other university for a degree of otherwise.

Kelvin Karani Njoroge

Student Name:………………………………Signature…...……… Date:………………..

This project report has been submitted for examination with my approval as the Program Project Supervisor.

Name ………………………… Signature ……………………. Date………………….

Program Project Supervisor

This project report has been submitted for examination with my approval as the Program Project Coordinator.

Name ………………………… Signature ……………………. Date………………….

Program Project Coordinator

# Acknowledgement

I would like to extend my heartfelt gratitude to my supervisor, Mr Thomas Nyajowi and project coordinator Dr Mary Ahuna, for their unwavering support, insightful guidance, and encouragement throughout the development of this project and project documentation. I am also deeply thankful to my friends, Stella Wambui and Jeff Kobia, for their invaluable contributions, encouragement, and feedback, which significantly enriched the project.

I am indebted to my family for their unwavering financial support and encouragement, which made this project possible. Their belief in me and their willingness to invest in my education and pursuits have been instrumental in my journey.

Lastly, I extend my appreciation to all the participants who generously volunteered their time and provided valuable feedback during the development and testing phase. This project is a testament to the collective effort and support of many, and I am truly grateful for each contribution.

Table of Contents

[Declaration 1](#_Toc151594524)

[Acknowledgement 2](#_Toc151594525)

[Table of Figures 5](#_Toc151594526)

[Table of Tables 6](#_Toc151594527)

[List of Acronym 7](#_Toc151594528)

[Abstract 8](#_Toc151594529)

[Chapter 1: Introduction 1](#_Toc151594530)

[1.1 Overview 1](#_Toc151594531)

[1.2 Background Information 1](#_Toc151594532)

[1.2.1 ETAs 1](#_Toc151594533)

[1.2.2 Artificial intelligence 2](#_Toc151594534)

[1.3 Problem Statement 4](#_Toc151594535)

[1.4 Proposed Solution 4](#_Toc151594536)

[1.5 Objectives 4](#_Toc151594537)

[1.5.1 Main Objective 4](#_Toc151594538)

[1.5.2 Specific objectives 4](#_Toc151594539)

[1.6 Block Diagram 5](#_Toc151594540)

[1.6.1 Algorithm development server 5](#_Toc151594541)

[1.6.2 Power supply subsystem 5](#_Toc151594542)

[1.6.3 Audio subsystem 5](#_Toc151594543)

[1.6.4 Programming computer 5](#_Toc151594544)

[1.6.5 Controller 5](#_Toc151594545)

[1.6.6 Tactile feedback 5](#_Toc151594546)

[1.6.7 Sensor subsystem 5](#_Toc151594547)

[Chapter 2: Literature Review 6](#_Toc151594548)

[2.1 Overview 6](#_Toc151594549)

[2.2 Related work 6](#_Toc151594550)

[2.3 Machine learning 6](#_Toc151594551)

[2.4 Object detection algorithm 8](#_Toc151594552)

[2.4.1 Training dataset 9](#_Toc151594553)

[2.4.2 Testing dataset 9](#_Toc151594554)

[2.4.3 Training and testing model 9](#_Toc151594555)

[2.4.4 Deployment 9](#_Toc151594556)

[2.5 Sensors 9](#_Toc151594557)

[2.5.1 Proximity sensor 10](#_Toc151594558)

[2.5.2 Camera 10](#_Toc151594559)

[2.6 Controller 11](#_Toc151594560)

[2.7 Audio subsystem 12](#_Toc151594561)

[2.7.1 Audio amplifier 12](#_Toc151594562)

[2.7.2 Speaker 14](#_Toc151594563)

[2.8 Tactile feedback subsystem 14](#_Toc151594564)

[2.9 Power supply subsystem 15](#_Toc151594565)

[2.9.1 Voltage source 15](#_Toc151594566)

[2.9.2 Voltage regulator 15](#_Toc151594567)

# Table of Figures

# Table of Tables

# List of Acronym

AI - Artificial intelligence

ANN - Artificial neural network

CNN - Convolution neural network

CPU - Central processing unit

DC - Direct current

ETA - Electronic travel aid

GPIO - General purpose input/output

GPU - Graphics processing unit

IDE - Integrated development environment

IR - Infrared

I2C - Inter integrated circuit

Lidar - Light detection and ranging

ML - Machine learning

Mosfet - Metal oxide semiconductor field effect transistor

Opamp. - Operational amplifier

PWM - Pulse width modulation

RAM - Random access memory

ROM - Read only memory

SNR - Signal to noise ratio

SPI - Serial peripheral interface

TCP - Transmission control protocol

THD - Total harmonic distortion

Tflite - TensorFlow lite

TinyML - Tiny machine learning

USB - Universal serial bus

YOLO - You only look

# Abstract

This project aimed to develop an assistive device for the visually impaired by utilizing image classification, proximity detection, and haptic feedback mechanisms. The device aimed to enhance the independence and safety of visually impaired individuals by providing real-time audio feedback of recognized objects and proximity alerts to detect nearby obstacles, specifically focusing on recognizing people and cars. The developed device successfully recognizes people and cars with a 0.9 accuracy, when powered with a large current source, due to current constraints, initial testing with a 9v battery proved challenging but later testing will be conducted. Additionally, the proximity alert system effectively detects obstacles and alerts the user through vibration feedback. One recommendation to improve the device would be to have an optimal current supply source and should be portable and light.

# Chapter 1: Introduction

## Overview

This chapter will cover the background information on how technology has been applied to help come up with devices that can be used by the blind, the problem facing the blind community, solution that has been proposed, objectives of the project and the block diagram of the project to be designed and constructed.

## Background Information

In a recent conducted by world health organization, it is estimated that a total of 49 million people is blind worldwide (World Health Organization, 2020). The affected have their autonomy jeopardized in terms of many everyday tasks, with the emphasis being placed on those that involve moving through an unknown environment.

Generally, people rely on sight to know their location in an environment and what is around them. This has shown that sight is an important part of daily navigation of unfamiliar and familiar environment. As for the blind community, lack of visual data has resulted to them relying on other means such as echolocation to navigate in an unfamiliar environment. Recent research conducted on blind people brain scans has found that when they use echolocation in lieu of sight, they’re actually using the visual cortex, the region of the brain that processes sensory information from our eyes (Thaler, Arnott, & Goodale, 2011) This has resulted to various research conducted to find ways to assist blind people to navigate in their environment. One example is the white cane which has become a symbol for the blind community. It helps them navigate in unfamiliar environment without having to worry about relying on means such as echolocation which might be a challenge in the open air. Though the white cane has been popular over the decades, it has still it’s limitations, such as unable to differentiate different obstacles and also limited range.

## Electronic travel aids (ETAs)

Recent advances in technology have led to development of assistive devices for the blind. The use of assistive devices has been increasing, and several electronic aid devices have been introduced over the past few years, called electronic travel aids (ETAs). ETAs have been combining various aspects of technology to come up with devices that are user friendly and easy to use. This has resulted to using sensors such as Lidar and ultrasonic sensor to provide distance information to the person.

These technologies have been relying on the various types of sensors to pick the required signals thus have some limitations. An example is when the white cane is fitted with an ultrasonic sensor, it will be limited to only objects that can reflect back the ultrasonic sound wave and has limited range. Research has been conducted by various institutions to see how computer vision technology used in self-driving cars and in robotics such as Boston Dynamic’s robot dog called Spot (Boston Dynamics, 2023) can be used as a guiding tool for the blind. They rely on vision from cameras, distance data from Lidar and an AI to be able to interpret what’s being fed by the camera to useful data that can be fed to the cars control system to perform the necessary action such as obstacle avoidance.

By applying the same concept, ETAs are being developed that are more capable compared to the one used earlier on.

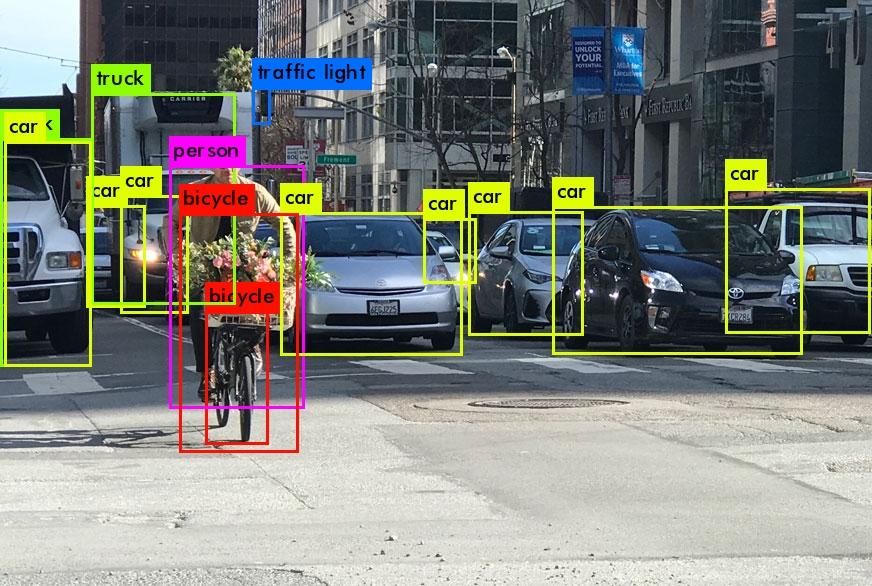
## Artificial intelligence

In 1956, John McCarthy coined the term artificial intelligence (Mintz & Brodie, 2019) He defined AI as the ability of a machine to emulate human behavior and perform tasks without human intervention. This involves using simple models created by writing simple programs to complex programs that are able to perform different tasks with minimal programming from a person which commonly use machine learning algorithms.

Advancement in the field of AI has led to researchers coming up with advanced models that are able to classify objects to various categories. These recent developments have led to innovative ideas such robots that can be able to identify various objects and detect anomalies such as a self-driving car will be able to classify a pedestrian crossing the road and be able to stop without relying on traffic lights or any other road sign.

The exponential growth of this field has led to development of algorithms such as YOLO which are able to classify multiple objects at the same time and can be used in many areas that requires object detection, such as creating a navigation system for the blind.

YOLO algorithm has become popular due to its large broad data classification and easy to use in a project. The below image shows one project created by Juan Cruz Martinez that shows how such an algorithm will classify various objects in a given frame (Martinez, 2021)

Figure 1.1 : YOLO algorithm

The above image shows clearly the different objects detected in the picture and labelled as per what they are. It can be seen that the algorithm is able to pick up various cars, bicycle, traffic light and person in the captured frame. This information can later be processed to work in various projects such as in self-driving cars which, as mentioned above, requires constant update of what’s around them. By this approach, the same algorithm can be used in development of ETAs. But this poses a challenge in which YOLO algorithm requires a machine with enough processing capabilities and ETAs are supposed to be light and portable to be comfortable, thus a light algorithm for object recognition will be required which will be discussed in the next chapter.

## Problem Statement

Human and vehicle traffic is a major factor found in every busy area, be it part of a city or inside a school premise. This can be a challenge for a blind person to navigate through the crowd as he or she will depend on the person ahead seeing him and letting him pass, otherwise, collision might be imminent.

In a pedestrian busy pathway, it’s quite challenging, even for a person who can see, to walk properly when there is a person in front of them walking at a slower pace than they are. This can be a big challenge to a blind person as he or she will not be able to tell if the distance between him and the person in in front is reducing thus can lead to him or her tripping over.

## Proposed Solution

In order to curb the above stated problems, an AI will be used to be able to detect and recognize cars and people present right at the front of the blind person by using a trained algorithm. The algorithm will be able to detect a person or/and a car captured by the camera and recognize accordingly. This will be combined with a proximity system that will provide an alert system by calculating the distance and determining whether collision is imminent by comparing distance measurements taken at a given time interval. The alert system will be informed if audio feedback informing the wearer action to take and a vibration motor that will act as a supplement Incase collision is too close.

## Objectives

The following were the objectives that guided the implementation of the proposed device.

## Main Objective

To design, construct, program and test of an AI Assisted navigation device.

## Specific objectives

1. To train, test and deploy a person and car recognition model.
2. To design, construct, test and interface the audio amplifier.
3. To design, construct and test a constant 5V power supply system.

## Block Diagram

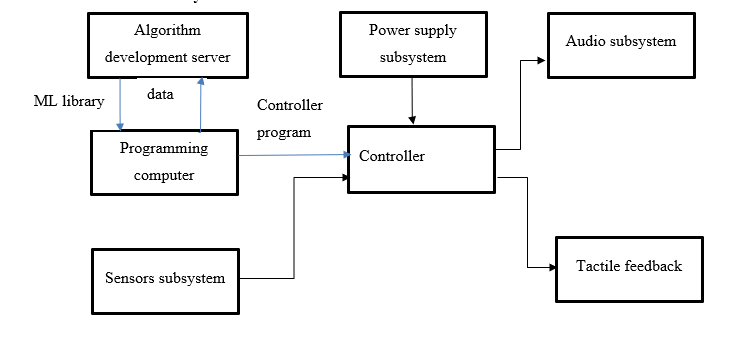


Figure 1.2 : Block Diagram of the system

## Algorithm development server

This will the cloud-based server to develop the machine learning algorithm.

## Power supply subsystem

This system comprises of the voltage regulator and the voltage source.

## Audio subsystem

Comprises of the speaker and the amplifier.

## Programming computer

This is used to communicate with the server and the controller during program development.

## Controller

This is the central control system of the device.

## Tactile feedback

This system comprises of the vibration device.

## Sensor subsystem

This comprises of the camera and the proximity sensor and are used to provide the image and the distance measurements.

The blue lines indicate communication off device by either using USB communication protocol for computer to microcontroller and TCP for computer to algorithm development server.

* 1. **Specifications**

The device that was designed in this project report included an esp32cam controller, lm736 amplifier, 0v2640 camera module and an lm7805 voltage regulator. The device operated by running an AI model to provide appropriate detection thus for optimal operation, it should be used in a well lit environment as the camera is only 2MP thus limited performance in low light.

Due to the limited memory, running a large model is a challenge thus this device was only trained on two classes. The limited memory also limits the audio size that can be played to provide useful information about the environment the device is exposed to.

The device has the following specifications;

Table 1.1 Device Specifications

|  |  |
| --- | --- |
| Parameter | Optimal value |
| Voltage supply | 9V DC |
| Supply current | 1A |
| Output resistance | 8Ω |
| SPI flash | 32Mbit |
| RAM | Internal -520kB  External -4MB PSRAM |
| Communication interface | UART, SPI and I2C |
| Serial baud rate | 115200bps |
| Camera type | OV2640, 2MP |
| Image output | JPEG, BMP and GRAYSCALE |
| Operating temperature | -20°C ~ 85°C |

# Chapter 2: Literature Review

## Overview

This chapter will cover the theory behind the project. This will include any of the related work, in depth look at machine learning and the various sub blocks and covering the theory for every component that will make up the complete system.

## Background Research

In the realm of assistive technologies for the visually impaired, various navigation systems have been developed to enhance mobility. Mounir Bousbia-Salah pioneered a system utilizing three ultrasonic sensors – two strategically positioned on the user’s shoulders and one integrated into the white cane (Bousbia-Salah, Maamar, & Larbi, 2011). This innovative design enabled real-time object detection within a range of 6 meters, covering both overhanging obstacles and those at ground level. The system conveyed information to users through tactile feedback via two vibration motors and auditory cues (Bousbia-Salah, Maamar, & Larbi, 2011).

In a parallel effort, Koharwal and colleagues devised a navigation system featuring a Raspberry Pi running an object recognition algorithm based on OpenCV. This system utilized IR sensors to map object shapes and sizes, while ultrasonic sensors provided distance information. The culmination of these inputs was converted into audible feedback for the user through headphones (Koharwal, Awwad, & Vyakaranam, 2019).

Building upon these foundations, the system in this project document introduces notable improvements. Firstly, a recognition model based on Convolutional Neural Networks (CNN) is employed, quantized to below 500KB. This reduction in computational requirements facilitates implementation on various microcontrollers, expanding the accessibility and deployment potential of the navigation system. Additionally, by using a cheap controller, the cost is reduced drastically while improving performance by using a neural network.

## Machine learning

According to (Mahesh, 2020) Machine learning is defined as the scientific study of algorithms and statistical models that computer systems use to perform a specific task without being explicitly programmed. These algorithms are capable of analyzing a large set of data and provide inferences as per the training dataset. This field has evolved over the years leading to development of algorithms that are capable of performing several tasks more effective than humans as was already covered in chapter one. Techniques based on machine learning have been applied successfully in diverse fields ranging from pattern recognition, computer vision, spacecraft engineering, finance, entertainment, and computational biology to biomedical and medical applications such as diagnosis of cancer and other medical conditions (El Naqa & Murphy, 2015)

Machine learning is a subfield of artificial intelligence thus any model developed using a machine learning algorithm will be considered an AI. Under machine learning, we have another subset called artificial neural networks (ANN) that makes up the core part of how an algorithm will be able to lean and generating unique features for different labels.

Features are defined as the inputs to a neural network while labels are the output. Deep learning is a subset of ANN and is defined as how deep, referring to the number of hidden layers in a given ANN, a particular model is going to learn

Machine learning algorithms can be classified to three main categories

* Supervised learning

This is where an algorithm is trained on a labeled dataset, meaning each input has a corresponding output. The goal is for the model to learn a mapping from inputs to outputs, allowing it to make predictions or decisions on new, unseen data.

* Unsupervised learning

In this type of learning, the algorithm is given unlabeled data and must find patterns, relationships, or structures within it without explicit guidance.

* Reinforcement learning

This involves an agent learning to make decisions by interacting with an environment. The agent receives feedback in the form of rewards or punishments, which guides its learning process. The goal is for the agent to learn a strategy that maximizes the cumulative reward over time.

Object detection falls under supervised learning as it will require labeled during training. Large sets of labeled data will be required to have a model with acceptable accuracy. In order to train a model, it is computationally intensive requiring CPUs and GPUs with enough processing power. This has posed a challenge over the years this limiting the areas of applications.

Recently TinyML, a new field of machine learning developed specifically for devices with low memory and extremely low power consumption has become popular. This allows machine learning to be conducted on edge devices such as microcontrollers with sufficient ram and rom to run inferencing (Warden & Situnayake, 2019) This new field has enabled development of low power devices to become possible.

* + 1. **Convolution Neural Network**

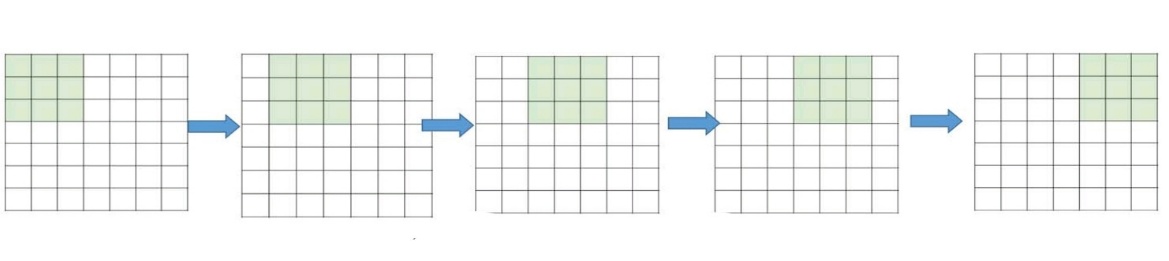
A Convolutional Neural Network, also known as CNN, is a class of neural networks that specializes in processing data that has a grid-like topology, such as an image (Ketkar & Moolayil, 2021)). These networks are able to extract features from images by applying different neurons to process data from different parts of the image such as the outline.

A convolution neural network is made up of different layers that work together to extract features at any given location of an input thus forming a deep neural network. These layers are:

* **Convolutional layer**

In the convolutional layer, multiple filters slide over the layer for the given input data. A summation of an element-by-element multiplication of the filters and receptive field of the input is then calculated as the output of this layer. The weighted summation is placed as an element of the next layer (Albawi & Mohammed, 2017).

Assuming that an input image has a dimension 7×7×3, when the image is passed through the first convolution layer, a filter window size can be set at (3 x 3) that will extract features of the image at that given point thus the next convolution layer will have features of corresponding parts of the image. This is similar to sliding the window through out the image to map specific features to the second convolution layer neurons. Moving the filter involves introducing stride that will shift the window at the given number of pixels, in the above, by using a stride of 1, the window will shift one pixel at a time. This will result to an output of 5x5 from initial 7x7. If we were to use a stride of 2, the output would be reduced to a 3x3 window (Albawi & Mohammed, 2017).

Figure 2.1 stride 1 on a 3 × 3 filter window

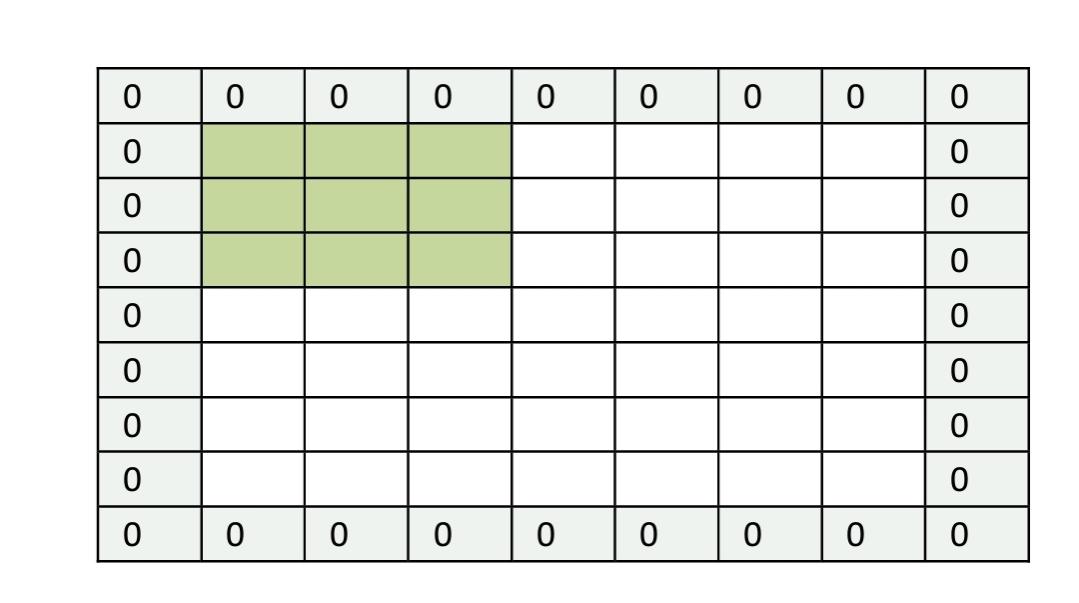
For stride mathematical model, given the image N×N dimension and the filter size of the F×F, the output size O and a stride of S, the equation will be:

*Equation 2.1 stride mathematical equation with no padding*

One of the drawbacks of the convolution step is the loss of information that might exist on the border of the image. Because they are only captured when the filter slides, they never have the chance to be seen. A very simple, yet efficient method to resolve the issue, is to use zero-padding. The other benefit of zero padding is to manage the output size. For example, in Figure 2.2 6 with N=7 and F=3 and stride 1, the output will be 5×5 (which shrinks from a 7×7 input). By adding zero padding to the output, the 7x7 shape is retained but now the actual N becomes 9. Considering zero padding, the equation is modified to:

*Where P is the number of the layers of the zero-padding*

*Equation 2.2 stride mathematical equation with zero padding*

Figure 2.2 Zero padding

This padding idea helps us to prevent network output size from shrinking with depth. Therefore, it is possible to have any number of deep convolutional networks.

* **Non-linearity layer**

Non-linearity layers are essential for introducing non-linearities to the network, enabling it to learn complex patterns and relationships in the data. The most common non-linearity layer used in CNNs is the Rectified Linear Unit (ReLU) activation function, softmax functions and sigmoid functions.

Relu has a definition of

*Equation 2.3 Relu activation function*

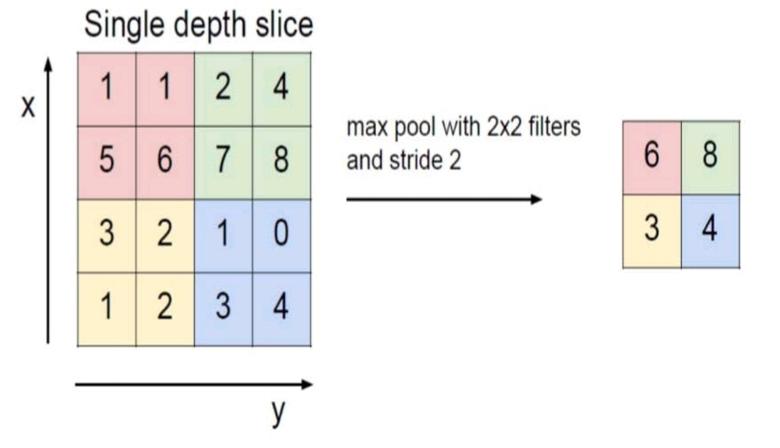
thus introducing non linearities by cutting off the negative values. It is commonly used in the convolution and fully connected layers in a cnn architecture.

Softmax and sigmoid are applied mostly in the output layer of the network as they always have a non zero result which would not be favourable for training.

* **Pooling layer**

Pooling main function is down-sampling in order to reduce the complexity for further layers. Pooling does not affect the number of filters.

Max-pooling is one of the most common types of pooling methods. It partitions the image to sub-region rectangles, and it only returns the maximum value of the inside of that sub-region as shown in the diagram below;

Figure 2.3 Max pooling

There are other common methods of pooling such as average pooling, which returns the average value of the sub region. Global pooling involves summarizing the entire feature map into a single value for each channel.

Max pooling is preferred where actual location of features in an input does not matter as it tends to preserve more prominent features. By selecting the maximum value within a local region, it emphasizes the most active feature, making it more effective in capturing and preserving distinctive patterns.

* **Fully connected layer**

In this layer, each node in a fully-connected layer is directly connected to every node in both the previous and in the next layer as in typical neural networks. The output from the pooling layer, which is usually 2D is flattened to 1D and applied as inputs to the fully connected layer. In this layer, the neuron applies a linear transformation to the input vector through a weights matrix. A non-linear transformation is then applied to the product through an activation function.

*Where y is the output, f the linear activation function, w the weight, x the neuron input and b the bias*

*Equation 2.4 features transformation*

The fully connected layer utilizes the output from the convolution process and predicts the class of the image based on the features extracted.

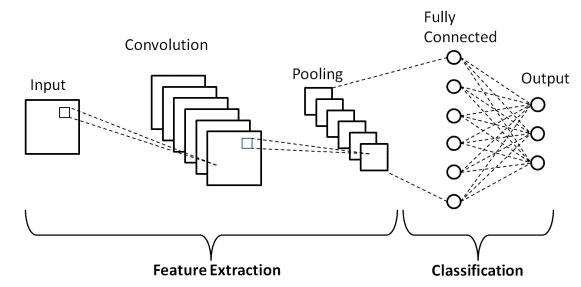


Figure 2.4 Architecture of a convolution neural network

* + - 1. **Training of a convolution neural network**

Training a Convolutional Neural Network (CNN) involves the process of optimizing its parameters, such as weights and biases, using a dataset. This involves the following steps:

**Data preprocessing**

Data preprocessing involves preparing and cleaning raw data to make it suitable for training a model. It aims to enhance the quality of the data and improve the performance of the machine learning model. Common data preprocessing tasks include:

* Data Cleaning: Handle missing values, outliers, and errors in the dataset to ensure accuracy and reliability.
* Normalization/Scaling: Scale numerical features to a standard range, often between 0 and 1, to prevent certain features from dominating others.
* Handling Categorical Data: Convert categorical variables into a numerical format, often through techniques like one-hot encoding.
* Data Splitting: Divide the dataset into training, validation, and testing sets to assess the model’s performance on unseen data
* Data Augmentation: Generate additional training examples by applying random transformations to the existing data.
* Handling Imbalanced Data: Address class imbalances by oversampling the minority class, under sampling the majority class.

**Passing the preprocessed data as inputs to the CNN**

This involves calling the model.fit () function in a cnn, providing the training, testing and validation data. Hyper parameters such as batch size, number of epochs are passed in the function.

**Adjusting the hyper parameters.**

This is performed to help the model learn from the training data without generalizing in a common feature by changing the epochs, batch size, learning rate and the model architecture.

If a model has limited dataset, the model will tend to memorize the training features thus only able to perform only on those features. This is commonly known as overfitting. Also, if the model is too simple, it may fail to capture the underlying patterns in the training data thus perform poorly in an unseen data. This is referred to under fitting.

Overfitting and under fitting provide undesirable outcome and can be mitigated by the following:

**Mitigating underfitting**

* Increase Model Complexity: If the model is too simple, adding more layers or neurons to increase its capacity to capture complex patterns in the data.
* Feature Engineering: Introduce additional relevant features or modify existing ones to provide more information to the model.
* Train Longer: Allow the model to train for more epochs, giving it more opportunities to learn from the data.

**Mitigating overfitting**

* Regularization: Apply regularization techniques, such as L1 or L2 regularization, to penalize large weights and prevent the model from becoming too complex.
* Dropout: Introduce dropout layers during training, randomly removing a fraction of neurons in each iteration to prevent co-adaptation of neurons.
* Data Augmentation: Generate additional training examples by applying random transformations to the existing data.
* Cross-Validation: Use techniques like k-fold cross-validation to assess the model’s performance on different subsets of the data, helping to identify overfitting.
* Early Stopping: Monitor the model’s performance on a validation set during training and stop when performance starts degrading, preventing overfitting due to excessive training.
* Reduce Model Complexity: Simplify the model architecture by reducing the number of layers or neurons, especially if the dataset is not large enough to support a more complex model.
  + - 1. **Dataset**

A dataset in machine learning (ML) is a structured collection of data points used to train, validate, or test a model. It typically consists of input-output pairs, where the inputs are features or attributes, and the outputs are corresponding labels or target values. The quality and diversity of the dataset play a crucial role in the performance of the ML model. A well-curated dataset should encompass a representative sample of the problem domain, cover various scenarios, and include enough instances for effective learning. Cleaning, preprocessing, and splitting the dataset into training and testing subsets are essential steps to ensure the model generalizes well to unseen data. Additionally, addressing issues like class imbalance or missing values contributes to a robust and reliable ML system. The dataset used in this project was obtained from roboflow.

**Assembly of the dataset**

The “Persona” dataset is a compilation of images portraying individuals from diverse backgrounds and contexts, primarily sourced from platforms like Unsplash and Google Images. The majority of these images are not owned by the dataset creator; they were obtained freely for academic purposes. The intentional diversity in image selection aims to provide a comprehensive representation of images falling under the label of “Person” and “car”

**Target population of the dataset**

This dataset seeks to encapsulate a broad representation of the human population labeled as “Person.” It deliberately includes images captured from various sources, including street photography, offering a glimpse into the lives of individuals in real-world scenarios. Moreover, the dataset encompasses images featuring children, aiming to be inclusive and representative of diverse age groups within the target population.

**Ethical issues of the dataset**

The inclusion of street images and images of children raises ethical considerations that require careful attention. Privacy concerns emerge, particularly in the context of individuals captured in public spaces without explicit consent. Addressing these concerns involves evaluating the potential impact of dataset use on privacy rights, especially for individuals in public settings. The inclusion of images of children adds an extra layer of ethical responsibility, necessitating measures to protect their privacy and well-being.

**Threats Posed by the dataset**

Privacy and consent are primary concerns posed by the inclusion of street images, as individuals in public spaces might not have anticipated being part of a dataset. Furthermore, images of children introduce potential risks related to child protection and ethical usage. These threats underscore the importance of implementing safeguards to ensure responsible handling and minimize any unintended consequences.

**Remedies for threats**

To mitigate privacy concerns, employing robust anonymization techniques, such as blurring faces and removing identifiable features, becomes crucial. When dealing with images of children, adhering to strict ethical guidelines, obtaining proper consent, and implementing measures to prevent any harm or exploitation are essential. Transparent communication regarding the ethical use and academic purposes of the dataset is vital to build trust and ensure responsible handling.

**Suitability of the dataset**

Despite the ethical challenges, the dataset’s suitability for academic purposes lies in its ability to offer a diverse and realistic portrayal of individuals labeled as “Person.” The intentional inclusion of street images and images of children enhances its representativeness, making it a valuable resource for studying real-world scenarios and societal dynamics.

**Analysis Tools**

To analyze the dataset effectively, a combination of image processing tools, computer vision algorithms, and statistical methods may be employed. Tools such opencv and python’s matplot lib can be used for dataset analysis, to plot and verify the various images in the dataset thus enabling easy identification of outliers.

## Training dataset

This is a collection of input images along with their corresponding labels. These images are used to train the network by adjusting its parameters (weights and biases) based on the error between the predicted output and the actual labels. The goal is for the CNN to learn and generalize patterns in the data, making accurate predictions on new, unseen images.

## Validation and testing dataset

During training, a portion of the dataset is set aside as the validation dataset. The model is not trained on this data, but it’s used to evaluate its performance on unseen examples. It helps in tuning hyperparameters and preventing overfitting by providing an independent measure of the model’s accuracy.

After the model has been trained, testing dataset is used to evaluate the model on a complete and unseen dataset that will asses model’s performance in real world data

* + - 1. **Object detection algorithm**

Development and deployment of the algorithm can be done on Edge impulse platform. According to (Hymel, Situnayake, Elium, Kelcey, & Reddi, 2022). Edge Impulse is a cloud-based machine learning operations (MLOps) platform for developing embedded and edge ML (TinyML) systems that can be deployed to a wide range of hardware targets. The platform allows training and testing models for edge devices using TensorFlow lite and deployed to the required format for a given microcontroller. It supports deployment of models to various microcontrollers such as those programmed in Arduino IDE or deploying the model directly to a C++ library that can be compiled to work with a custom microcontroller programmed in C++.

Developing a supervised machine learning model requires a large set of data. This data is fed at the input node of the neural networks during training. This allows the neural network to develop unique features for each label and provide the raw binary outputs that the model will understand as a label corresponding to the training data.

## Training and testing model

Transfer learning is applied during the training phase of the model. Transfer learning is method of machine learning where a model trained to solve a particular task such image recognition is reused by shedding of the final layer of the CNN, retaining the core weights and features of image recognition and later retraining the model with the new dataset, freezing all other layers except the last layer to create new features and weights to solve the new task (Torrey & Shavlik, 2010) This saves on time and computational resources as the model had already been pre-trained on a large dataset.

## Deployment

This refers to converting the output model from the tensorflow lite format to a format that will be compatible with the microcontroller to be used. Edge impulse provides a deployment functionality in which the supported library can be obtained. The library, example is Arduino library, is later uploaded to the IDE and later used as per the application area.

* + - 1. **Displaying results**

The results for the above trained model can be tested by uploading an image to folder named test data, and running the model.predict keras function in Google colab. This will display the class and the confidence level of the given image.

## Sensors

A sensor is a device that detects and responds to some type of input from the physical environment. There are various types of sensors for various applications. In this project, proximity sensors and camera sensor was be used.

## Proximity sensor

In robotics, proximity sensors have been applied in a wide range such as in obstacle avoidance. Sensors used in obstacle avoidance operate by providing a beam of either sound or light and measures time of reflection in reflective sensors or in transmissive sensors operate by having a transmitter and a receiver on the opposite side. When an object is in the path if light, the receiver will not receive a signal. Reflective proximity sensors are used to provide accurate distance readings by timing the total time taken for the sound or light to travel to and from the object. Distance is then calculated by using the below equation

*Equation 2.1*

There are three types of distance sensors that can be used in construction of a navigation system for the blind. These sensors are ultrasonic sensor and infrared distance sensor. Ultrasonic sensor was used for this project..

* + 1. **Ultrasonic sensor**

An ultrasonic sensor is a device that utilizes ultrasonic waves for distance measurement and obstacle detection. It typically consists of a transceiver module with a piezoelectric sensor. The sensor emits ultrasonic waves, usually in the range of 20 kHz to 200 kHz, and measures the time it takes for the waves to bounce back after hitting an object.

The key parameters of an ultrasonic sensor include the operating frequency, detection range, and resolution. The operating frequency determines the wavelength and affects the sensor’s precision. Detection range refers to the maximum distance over which the sensor can accurately measure objects, while resolution is the smallest distance change the sensor can detect. Ultrasonic sensors find widespread applications in robotics, automation, and security systems due to their non-contact nature and reliable performance in various environmental conditions.

## Camera

This is a type of sensor that converts light to image either through digital processing or by analog processing. The camera module forms the core of the project as it will be used as a ‘pair of eyes’ to provide vision information to the microcontroller for object recognition. The camera to be used in this project should be able to take at least 96 by 96 photos and have enough resolution to be applicable in the real world.

## Controller

This is the main processing unit of the navigation system. It is will be used to perform all the computational tasks such as object , distance calculation and text to speech conversion. The controller used will have to have specific requirements to handle the processing of the tasks. The requirements to be considered are:

Table 2.1 Controller Specifications

|  |  |
| --- | --- |
| Processing power | Enough processing power to handle image recognition |
| Memory | Enough ram to run software |
| Compatibility | Image recognition model and camera module |
| GPIO pins | Enough pins for all peripherals |
| Power efficiency | Consume less power |
| Size and form factor | Light and comfortable |

Considering the above, a microcontroller meeting the requirements such as an esp32cam was used for the project.

* + 1. **Esp32cam**

The ESP32-CAM is a versatile and compact development board based on the ESP32 microprocessor and an OV2640 camera module.

The ESP32-CAM features a powerful dual-core processor, built-in WiFi and Bluetooth connectivity, making it suitable for various IoT applications, especially those involving image capture and processing.

Programming the ESP32-CAM is typically done using the Arduino IDE with the ESP32 board support package. Developers can utilize libraries like ESP32 Camera, allowing for easy configuration and control of the camera module.

One notable feature of the ESP32-CAM is its GPIO pins, which provide flexibility for additional sensor integration or custom functionalities. However,esp32 has a higher power requirements, as the camera module can consume a significant amount of power during operation.

According to the datasheet, esp32cam has the following specifications:

Table 2.3 Esp32cam specifications

|  |  |
| --- | --- |
| Product name | ESP32-CAM |
| WiFi and Bluetooth modules | ESP-32S |
| Camera module | OV2640 2MP |
| Flash light | LED Built-in on Board |
| Operating Voltage | 3.3/5 Vdc. |
| Storage | Onboard TF card slot |
| RAM | Internal 512KB + External 4MB PSRAM |
| Power consumption | Flash off: 180mA@5V  Flash on and brightness max: 310mA@5V  Deep-Sleep: 6mA@5V  Modern-Sleep: 20mA@5V  Light-Sleep: [6.7mA@5V](mailto:6.7mA@5V) |
| Dimensions | 40.5mm x 27mm x 4.5mm |

* 1. **Audio subsystem**

This subsystem forms the output audio peripheral. It receives an analogue signal from the controller and amplifies it to the audible range.

The subsystem has an amplifier, a low pass filter and a speaker as the components.

## Audio amplifier

An amplifier is a device that increases the amplitude of a signal thus increasing its power. Audio amplifiers are used to amplify weak audio signals to audible signals. Some factors that affect choice of design of an audio amplifier are:

* Gain

This refers to a measure of a system’s capability to increase the amplitude of the output signal. It is expressed in the ratio of output voltage to the input voltage.

* Bandwidth

This refers to the frequency range in which the amplifier can operate.

* Clipping effect

This refers to voltage range amplifiers are designed to operate. Audio signals with voltage beyond the given range will have part of the audio removed from the signal.

* Efficiency

This is expressed in terms of output power of the amplifier to the power consumed by the amplifier itself.

* Noise and SNR

The semiconductor used to fabricate the amplifier results to some noise present in the output. The higher the power output of the amplifier, the more the noise.

An amplifier should be designed to have a constant noise output irrespective of the signal. SNR must be constant over its operating range.

There are different types of audio amplifiers that can be used for audio amplification. These amplifiers are divided into classes.

* + - 1. **Class A**

Class A amplifiers operate with a constant current, providing low distortion in the amplified signal. These amplifiers have a design where the output transistors are always conducting, making them suitable for high-fidelity audio applications. However, they are less power-efficient compared to other classes, as a significant portion of power is dissipated as heat, even when there is no input signal.

* + - 1. **Class B**

Class B amplifiers are more power-efficient than Class A. They use a push-pull configuration with two transistors, each handling one half of the signal waveform. While they offer improved efficiency, there’s a potential for crossover distortion at the point where the signal transitions from one transistor to the other. This distortion can affect audio quality.

* + - 1. **Class AB**

Class AB amplifiers aim to strike a balance between the low distortion of Class A and the efficiency of Class B. They use a combination of both classes, employing two transistors where one conducts during the positive half of the signal and the other during the negative half. This configuration reduces crossover distortion compared to Class B, making Class AB amplifiers a popular choice for various audio applications, including audio amplifiers in consumer electronics.

* + - 1. **Class D**

Class D amplifiers use digital switching techniques to rapidly switch the output transistors between fully on and fully off states. This results in high power efficiency, making them suitable for applications where energy conservation is crucial, such as in portable devices. Class D amplifiers are commonly used for subwoofers and portable audio systems, delivering good power efficiency with relatively lower heat dissipation.

Lm386 was used for this project as it offers a controllable gain between 20 to 200. This amplifier is versatile and offers low power consumption. According to the datasheet, the following are specifications.

Table 2.4 Lm386 Specifications

|  |  |  |  |
| --- | --- | --- | --- |
|  | | Min | Max |
| Voltage supply | 4V | 12V |
| Quiescent current (Vs-6V) |  | 8mA |
| Power dissipation |  | 1.25W |
| Input voltage | -0.4V | 0.4V |
| Speaker impedance | 4Ω |  |
| Output power (Vs- 9V, RL-8Ω) | 500mW |  |
| Input resistance |  | 50KΩ |

According to the datasheet, the schematic diagram is:

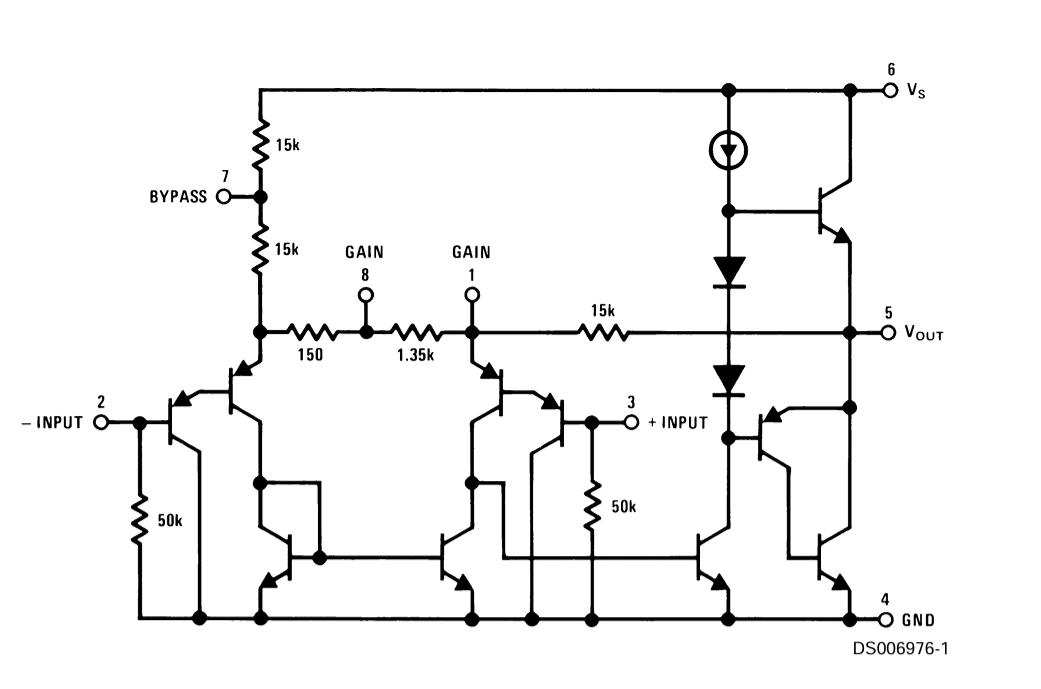


Figure 2.5 Lm386 schematic

## RC filter

RC filters, short for Resistor-Capacitor filters, are essential components in electronic circuits used to manipulate the frequency response of signals. They are widely employed in various applications, from audio processing to power supply stabilization. At the heart of RC filters lies the passive components: resistors and capacitors. These components interact to either attenuate or pass certain frequencies in a signal. The behavior of an RC filter is governed by the time constants determined by the resistor and capacitor values. In this project, we used low pass filter to attenuate the high frequency components from the audio signal. Low pass RC filters allow low-frequency signals to pass through while attenuating high-frequency components. The cutoff frequency (fc) determines the point where the filter begins to attenuate signals. The cutoff frequency can be calculated using the formula:

*Equation 2.5 cut-off frequency formula*

where R is the resistance and C is the capacitance. Frequencies below the cutoff pass through largely unaltered, while frequencies above are progressively attenuated.

## Speaker

A speaker converts the electrical signal from amplifier to sound. When selecting a speaker, several critical considerations come into play. The size and form factor of the speaker should align with the device’s design, ensuring it fits comfortably and doesn’t compromise portability. The power handling capacity and impedance of the speaker must be compatible with the output capabilities of the microcontroller, preventing issues like distortion. The frequency response range is crucial for clear speech synthesis, and sensitivity determines the speaker’s efficiency in converting electrical signals into sound. Additionally, factors such as connection interface, durability, weight, and power efficiency are pivotal for creating a reliable and effective audio feedback system within the assistive device.

## Tactile feedback subsystem

This system provides vibration feedback by using a vibration motor. A vibration motor is a compact and specialized type of electric motor designed to generate vibrations or oscillations. Also known as a pager motor or vibrating motor, it is commonly used in electronic devices to provide haptic feedback or alert the user through vibrations. The motor typically consists of an off-centered weight attached to the motor's rotor. When the motor rotates, the uneven distribution of weight causes the device to vibrate.

## Power supply subsystem

This system provides electric power to the microcontroller and the sensors and output devices used. It is made of two components:

## Voltage source

A voltage source is a device that provides a voltage and current to the system. It should be capable of providing 5V to the microcontroller and the sensors used.

The voltage source should be light weight and compact in size to fit in the assistive device. A common example is using dry cells which can provide the required voltage and current.

## Voltage regulator

A voltage regulator is an electronic circuit capable of providing a constant voltage supply no matter the input voltage. This allows flexibility in type of cells that can be used as the devices will receive a constant 5V supply.

Linear regulators are a type of voltage regulator that work by limiting voltage to provide one matching the desired voltage. An example of a linear regulator is 7805 chip that provides a regulated 5V output voltage.

# Chapter 3: Project Design

## Overview

This chapter introduced the design for the various parts of the device. This covered design of the convolution neural network, the architecture and the features extraction. System of design was also covered and the various components used as the amplifier and the voltage regulator.

## Design of the Audio Amplifier circuit

Audio amplifier circuit design is a crucial aspect of audio engineering. It involves designing and building circuits that amplify audio signals while maintaining their quality. The design of an audio amplifier circuit is essential to ensure that the audio signal is amplified to the desired level without distortion or noise.

There are several factors that must be taken into consideration to ensure optimal performance. This factors include:

* Gain
* Load impedance
* Noise and distortion

Lm386 was used due to it’s form factor and low power consumption. The amplifier has a range of gain that can be selected from 20 to 200. Higher gain means higher amplification of the output voltage. This is shown in the equation:

*Equation 3.1 Gain formula*

The gain is expressed in decibel by using the following formula:

*Equation 3.2 Gain on dB*

Thus the amplifier has gain range of:

According to the datasheet , to use gain of 20, all the resistors, from the schematic, that are used to find the gain(G) are:

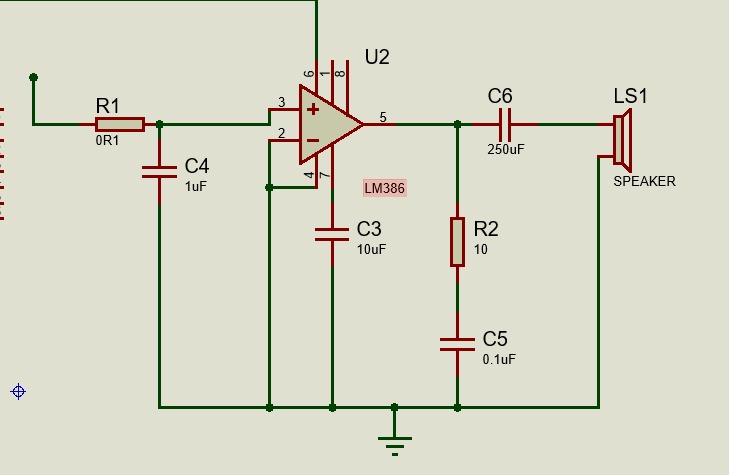
*Equation 3.3 Calculating minimum gain*

From the datasheet, to use the gain of 200, a bypass capacitor was introduced to the circuit to short out the 1.35K. The datasheet gives a 10uF capacitor.

*Equation 3.4 Calculating maximum gain*

Accounting for 5% tolerance, the gain will be 20 and 200 respectively as per the datasheet.

A gain of 20 was chosen for this project, this involved leaving pin 1 and floating.

****Figure 3.1 Amplifier Circuit

10KΩ potentiometer is used as a volume control by introducing a voltage divider circuit with the internal 50KΩ resistor indicated in the schematic. By increasing or decreasing the input voltage, the volume can be changed.

*Equation 3.5 Volume control*

Pin 7 being the bypass pin is connected to ground through capacitor C2. The datasheet suggested a value to be small for the given capacitor.

At the output pin 5, C3 is required between that and the speaker. This is because the output pin will have a DC offset which will be about half of your supply voltage – i.e., with no audio signal (or DC) at the input and a 9V power supply, you would measure 4.5V at pin 5.

C3 blocks this DC offset from shorting to ground through the speaker voice coil, allowing only AC to pass. Again, C3 forms a high filter with the impedance of the speaker itself. The calculation for the cut-off is 1/(2πRC). For an 8 ohm speaker and C3 of 250µF it’s:

*Equation 3.6 High frequency blocking*

R4 and C4 form a zobel network that is used to prevent oscillations at high frequencies.

An RC low pass filter was used to remove the high frequency components present in the pwm signal. Design for this filter was based upon the frequency range of audio signal. The cut-off frequency was selected to be 1600Hz. Accounting for impedance balancing of the microcontroller and the amplifier, a resistor of 1KΩ is selected and capacitor value is calculated as per the formula:

*Equation 3.7 Cutoff frequency*

As cut off frequency is selected to be 1000Hz and resistor value is 1K, capacitor value was calculated from the above formula:

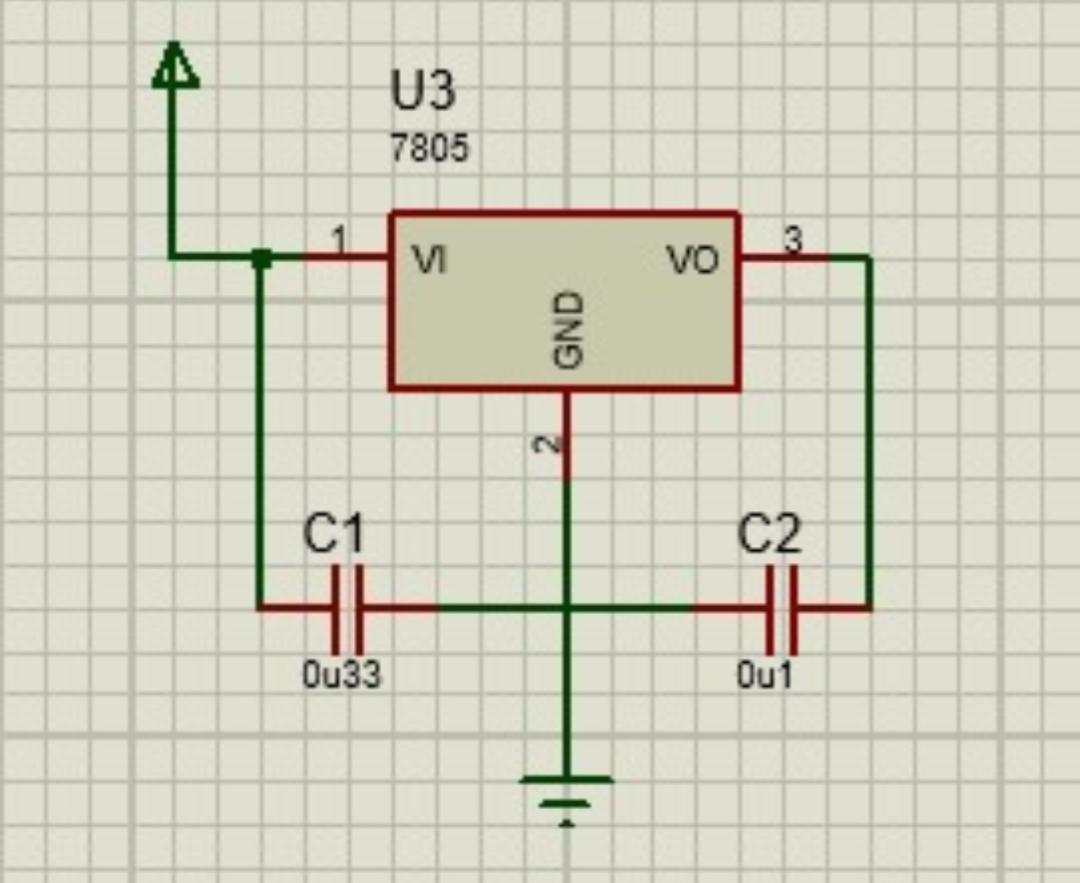
*Equation 3.8 Capacitance*

100nF capacitor was used after doing the calculations.

By using this RC filter, high frequency signals present in the simulated analog using pwm are filtered out leaving the low frequency components which contain the audio data.

## Design of 5V constant power supply circuit

A 5V power supply was designed by using lm7805 voltage regulator. From the datasheet, to obtain fixed output regulator, a 0.33uF and a 0.1uF capacitors were used to tie the input and output pins of the regulator to ground. This provides a smoothened 5V DC voltage.

Figure 3.2 Voltage regulator

## Software Design

This involved designing the software that will run the entire system, from the neural network to the device software that will assist in coordinating the various sensors and output devices. The convolution neural network was developed using Google colab and the system software was developed using Arduino ide.

## Design of the convolution neural network.

Designing an effective CNN requires the following to be observed:

* Type of dataset to be used
* Architecture of the model
* Deployment

## Data preprocessing

In this project, the underlying problem is to recognize two classes of images to their respective classes. This will involve preprocessing the data to the required format that can be fed through an input layer of the neural network. The dataset is prepared in a folder named ‘dataset’ with subfolders for various classes, for the case of this project the subfolders are ‘person’ and ‘car’.

Data preprocessing will be done using opencv (a python library) to provide the required tensors that can be fed to the input of the model. As pixel value ranges from 0 – 255, dividing the tensors with 255 provided a normalized training data of range 0 to 1.

Data augmentation is performed on the preprocessed data to virtually increase the size of the data in order to mitigate overfitting. The following steps will be carried out during data augmentation:

* Rotation range: This determines degree for random rotations. Random range = 20
* width shift range and height shift range: Determines random horizontal and vertical shifts, respectively, as a fraction of the total width or height. width shift range = 0.15 and height shift range = 0.2
* shear range: Shear intensity, for giving a diagonal shape to the images. It’s a specified angle in degrees. Shear range = 0.2
* zoom range: Range for random zoom. Zoom range = 0.3
* horizontal flip: Boolean, indicating whether to randomly flip images horizontally. Set to True
* Fill mode: Filling in newly created pixels after a rotation or a shift. Set to ‘nearest’

## Convolution neural network architecture design

A Convolutional Neural Network architecture was designed for image classification. It begins with a convolutional layer having 128 filters with a size of (3, 3), implementing L1 regularization. This was followed by a rectified linear unit (ReLU) activation function to introduce non-linearity. A max-pooling layer with a pool size of (2, 2) reduces spatial dimensions. A dropout layer helps prevent overfitting by randomly deactivating 20% of neurons. The architecture continued with a second convolutional layer featuring 64 filters and a similar structure of activation, pooling, and dropout. Subsequently, a flatten layer transformed the 2D feature map into a 1D vector, and a dense layer with 64 neurons follows, applying ReLU activation.

The final dense layer with 2 neurons and a softmax activation function is utilized for binary classification. .

Feature extraction

*Image size = 96 × 96*

*First Convolutional Layer:*

*Number of filters = 128*

*Filter size = (3, 3)*

*Stride is 1 and padding is ‘same’, the output size would remain 96x96.*

*First MaxPooling Layer:*

*Pool size = (2, 2)*

*The pooling layer halves the spatial dimensions.*

*Output size = 48x48 (features remain the same;1179648)*

*First Dropout Layer:*

*No change in the number of features.*

*Second Convolutional Layer:*

*Number of filters = 64*

*Filter size = (3, 3)*

*Using the same stride and padding, the output size would be 48x48.*

*Second MaxPooling Layer:*

*Pool size = (2, 2)*

*Halving the spatial dimensions again.*

*Output size = 24x24 (features remain the same;147456 )*

*Second Dropout Layer:*

*No change in the number of features.*

*Flatten Layer:*

*Flatten converts the 8x8 feature map to a 1D vector.*

*First Dense Layer:*

*Number of neurons = 64*

*Second Dense Layer (output layer for binary classification):*

*Number of neurons = 2*

## Design of the device software

The system software involves running inferencing, sending and receiving the ultrasonic pulse and calculating the distance, generating voice by use of text to speech library and activation of the vibration when a particular conditioning is met.

This software is developed under subsections:

* Sensors subsection – The program on this part was used to set up, activate and be ready to receive sensor readings such as image data from the camera and distance measurement from the ultrasonic sensor.
* Inferencing subsection – This involves running model. Predict function on the Arduino ide to classify the objects fed by the camera. This classification output was used in the for voice control.
* Output subsection – This code was used to take the various parts and activate output actuators accordingly. If an object was detected, audio would be played accordingly and if the distance to an object was to go below 50cm, vibration would be triggered.

## Final circuit Design

For the final circuit, the controller pins were connected to various sensors and outputs as shown in the table below:

Table 3.1 pin connections

|  |  |  |
| --- | --- | --- |
| Controller Pin | Actuator | Actuator pin |
| 12 | Ultrasonic sensor | Trig pin |
| 13 | Ultrasonic sensor | Echo pin |
| 15 | Vibrator | Vcc pin |
| 14 | Amplifier | Pin 3 |

The circuit diagram for the device was as:

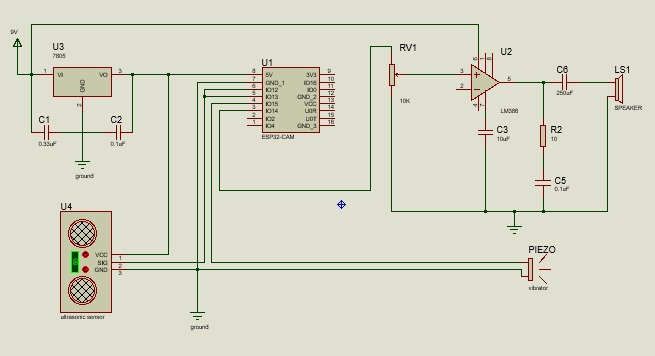
****

Figure 3.3 Final circuit diagram

## System flowchart

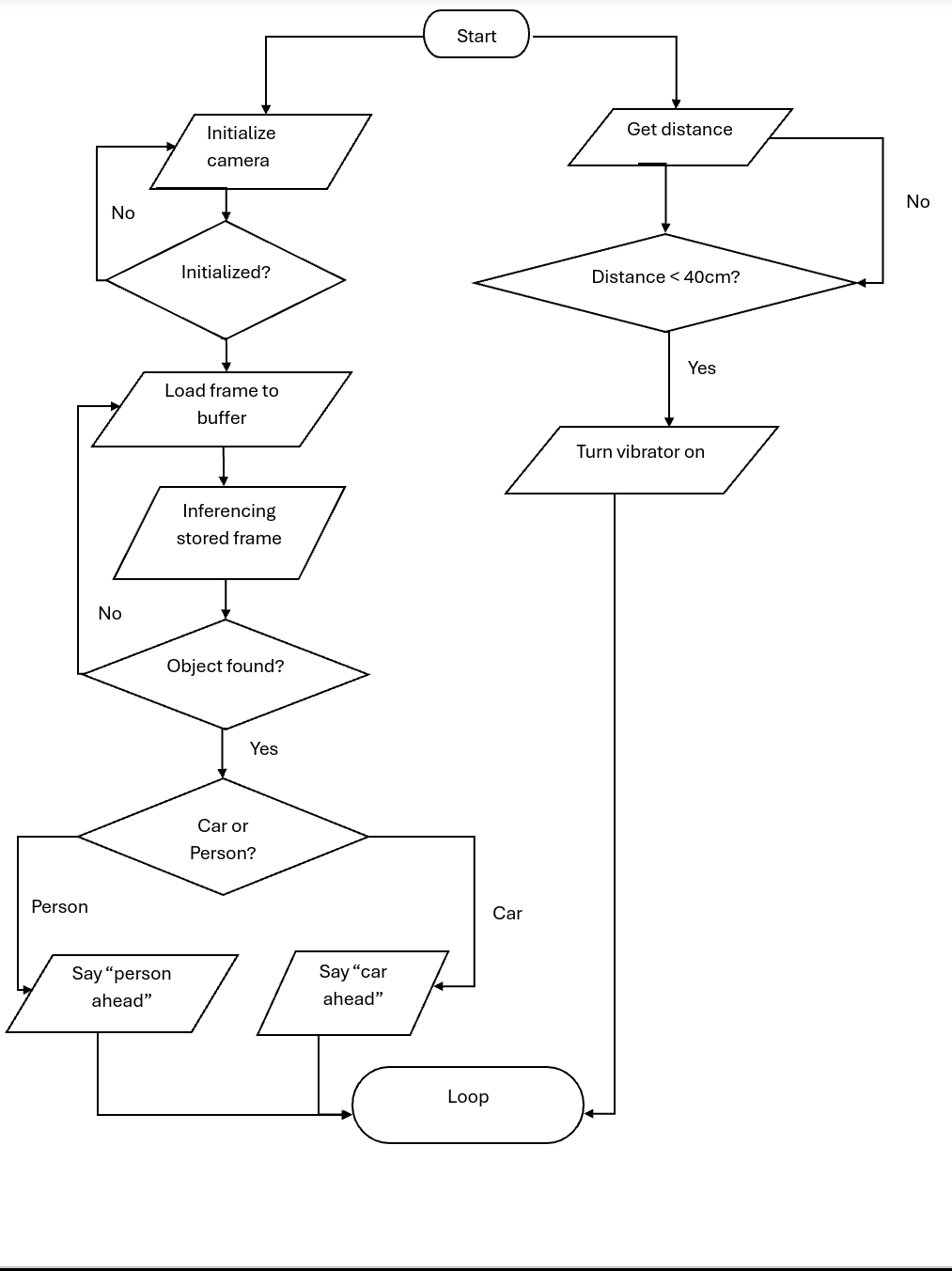
****

Figure 3.4 System Flowchart

# Chapter Four: Implementation And Testing

## 4.1 Overview

This chapter details how the device was implemented, constructed tested and validated from the circuit diagram designed in the previous chapter. It provides a detailed step by step of implementing of the power supply, audio amplifier and interfacing with the microcontroller and other sensors and actuators. It contains also provides detailed guidelines on how the program was implemented and the various environments that were used. This chapter also provides the test results from the objectives and details the challenges faced when implementing certain parts of the device.

## 4.2 Construction process

The components used for the construction of the AI Assisted navigation device for the blind were, Esp32Cam microcontroller, Ov2650 camera, lm386 class AB amplifier, resistors and capacitors, ultrasonic distance sensor, strip board, piezoelectric vibrator, 5V power supply and a speaker.

Detailed in this chapter is the implementation and construction of the various components listed to realize the device designed in the previous chapter.

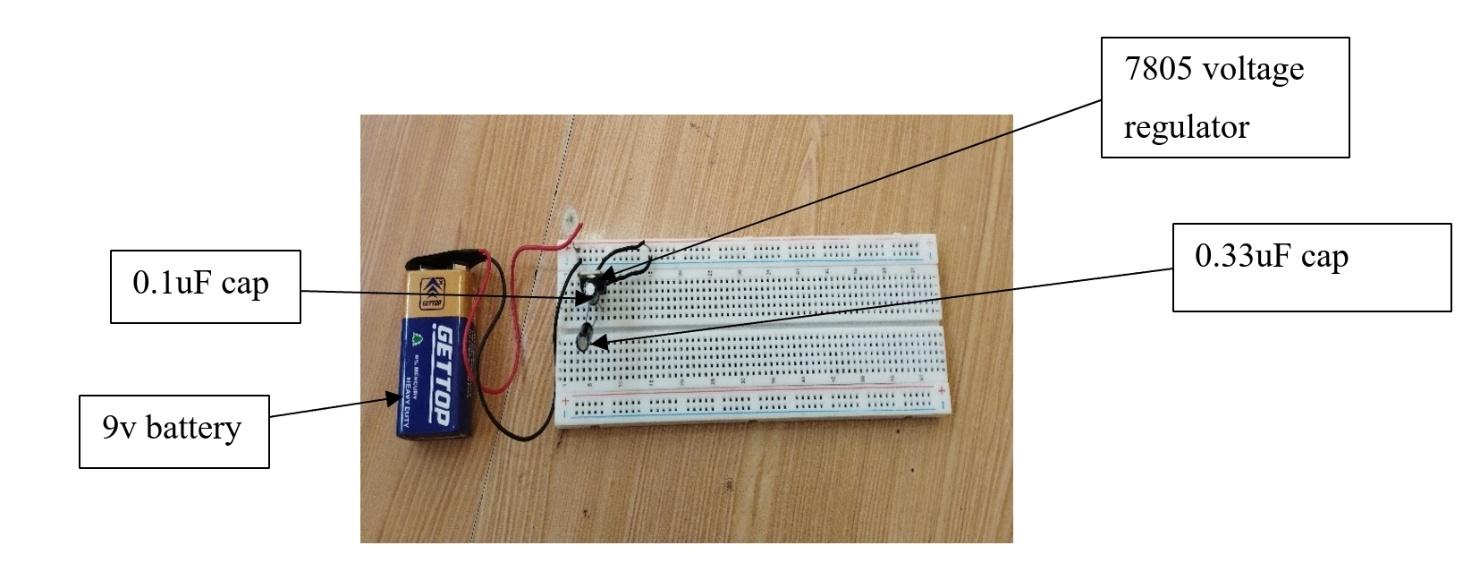
## 4.2.1 Tools and Equipment Used

The project utilized various tools and programming environments throughout the construction and testing process. The tools used are breadboard, usb to serial programmer, soldering iron, wires and a multimeter. The breadboard and wires were used to build a prototype for testing and validating the device. The usb to serial programmer was used to upload code to the esp32cam.The soldering gun was used to make permanent joints of the various components to a strip board. Multimeter was used during testing of voltage and current.

The programming environments that were utilized during the implementation of the software are Google Colab, Microsoft’s visual studio (VS) code ide, Arduino ide, Linux terminal’s xxd tool and Audacity. Google Colab was used during training, testing and deployment of the machine learning model. VS code, Audacity and xxd were used during processing of the audio and Arduino ide was used to develop the system software.

## 4.2.2 Implementing the voltage supply

Construction of the voltage supply was based on the circuit diagram designed in the previous chapter. IC 7805 was used and 0.33uF and 0.1uF capacitors were used as were implemented in the circuit diagram. Figure 4.1 shows how the voltage supply was implemented on the breadboard.

Figure 4.1 voltage supply

The 5v regulated output was connected to the positive rail of the breadboard and the ground to the negative rail.

Testing was done to verify that the voltage was regulated to 5v.

Table 4.1 Voltage supply

|  |  |  |  |
| --- | --- | --- | --- |
|  | Expected value | Measured value | Comments |
| Batt. Voltage | 9V | 8.23V | Within acceptable range |
| Regulated voltage | 5V | 4.99V | Within acceptable range |

The measured value was a bit lower than the expected value but this was in the normal range as the components have impedances that results to the small voltage drop.

## 4.2.3 Implementing the microcontroller with the camera

The camera was inserted in the camera slot of the microcontroller and the board was mounted on the breadboard, connecting 5v pin to positive power rail implemented in the previous step and the ground pin to the ground rail. This ensured that the microcontroller was powered.

One key challenge faced by mounting the microcontroller on a breadboard was inaccessibility of the reset button which was on the lower side of the microcontroller. Resetting the board required careful manipulation of the board.

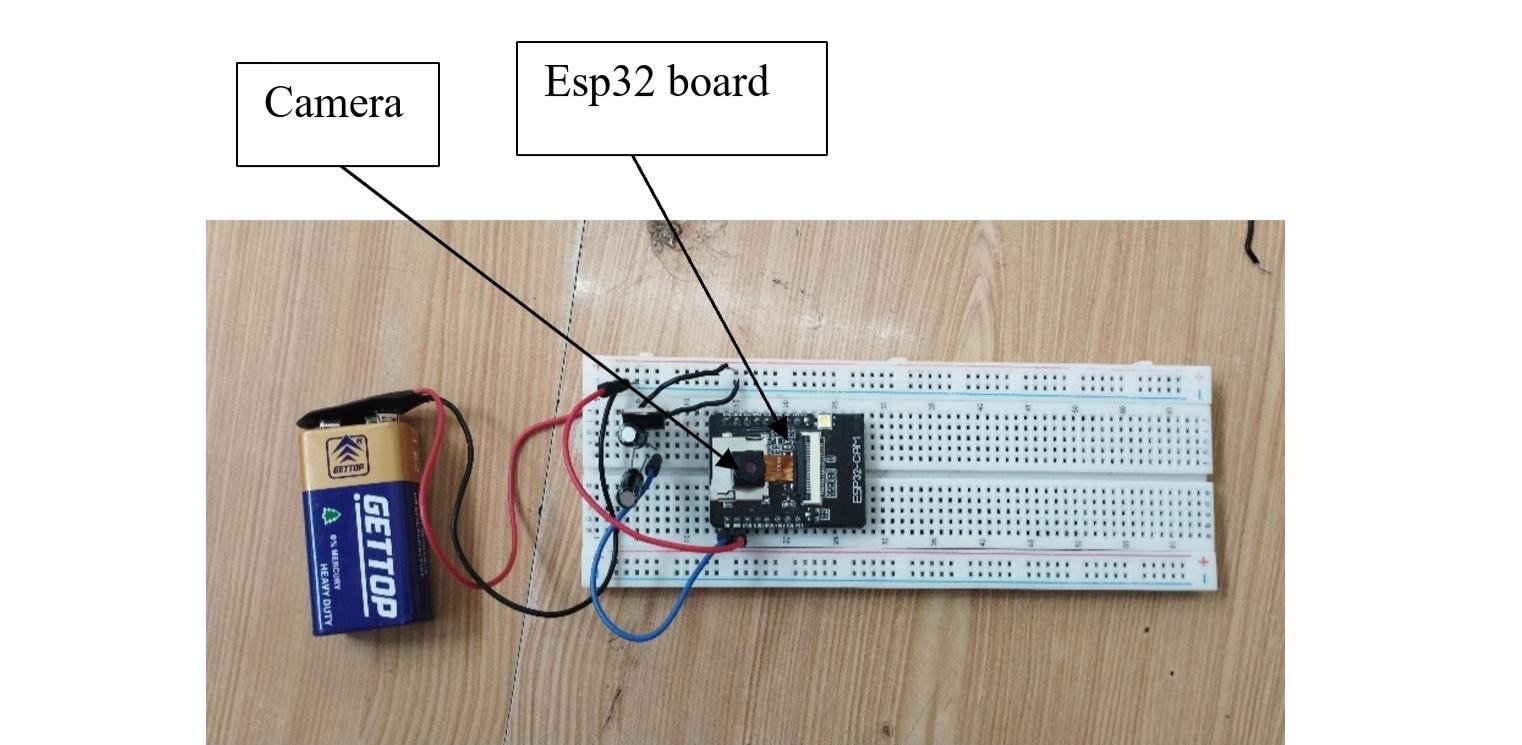
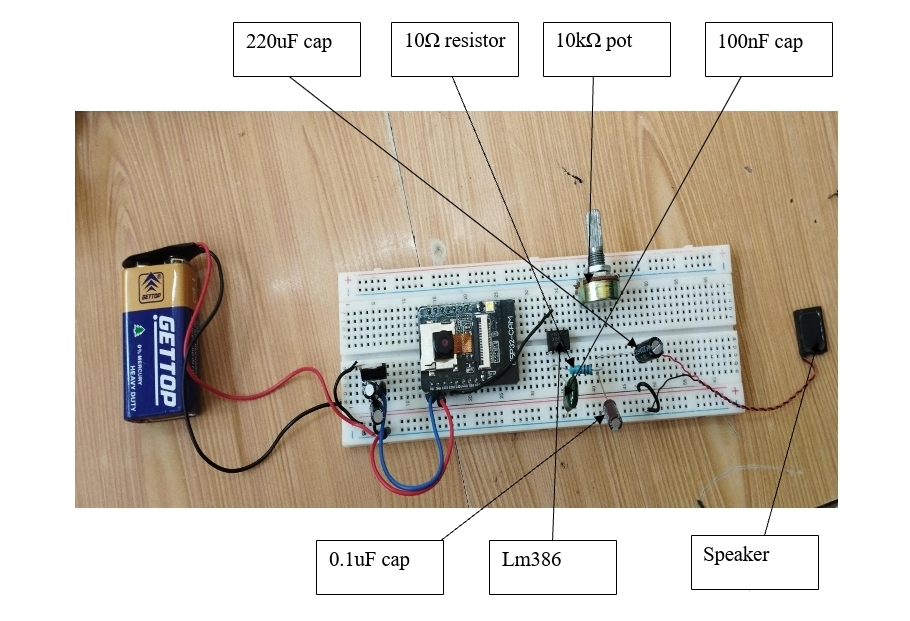
****

Figure 4.2 Esp32cam

## 4.2.4 Implementing the Audio amplifier

The audio amplifier was implemented as per the circuit diagram designed in the previous chapter. A gain of 20 was chosen thus pin 1 and pin 8 of the lm386 were left floating to utilize the internal 1.35K resistor. All the connections were made as per the circuit diagram, observing the polarity of the electrolytic capacitors. The input pin 3 is connected to pin 14 of the microcontroller and pin 5 is connected to the speaker through the zobel network.

Figure 4.3 Audio Amplifier

## 4.2.5 Implementing ultrasonic sensor

Ultrasonic sensor has four pins, two for power and two for signal. The vcc pin was connected to the positive rail on the breadboard and the ground was connected to the ground rail. This connection was used to power the sensor. The trig pin was connected to pin 12 of the esp32cam and echo pin was connected to pin 13 if the microcontroller. This enabled the sensor to be able to communicate with the microcontroller once the program was loaded.

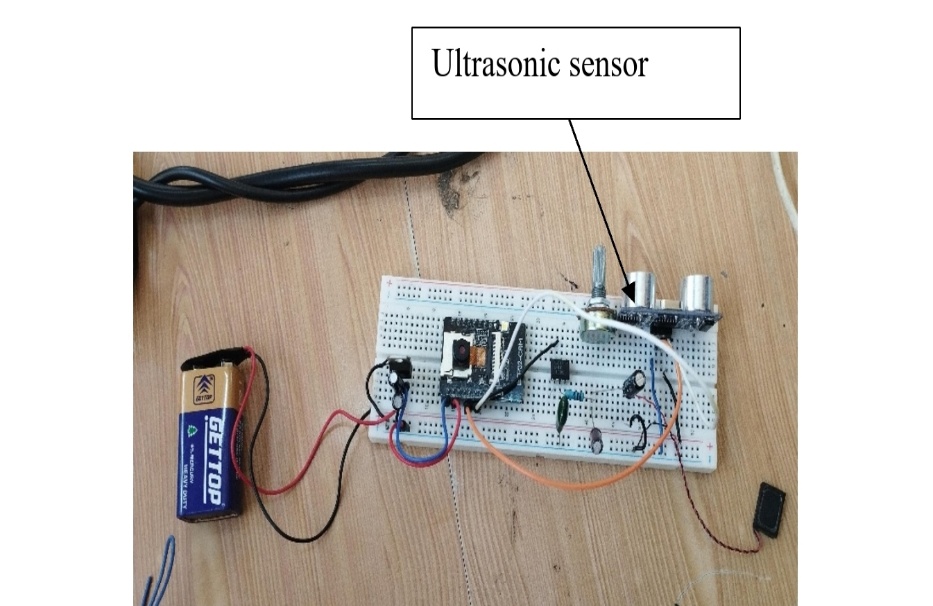
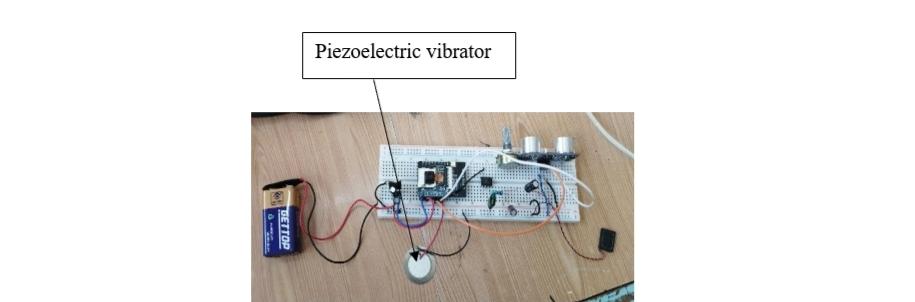


Figure 4.4 Ultrasonic sensor

## 4.2.6 Implementing the piezoelectric vibrator

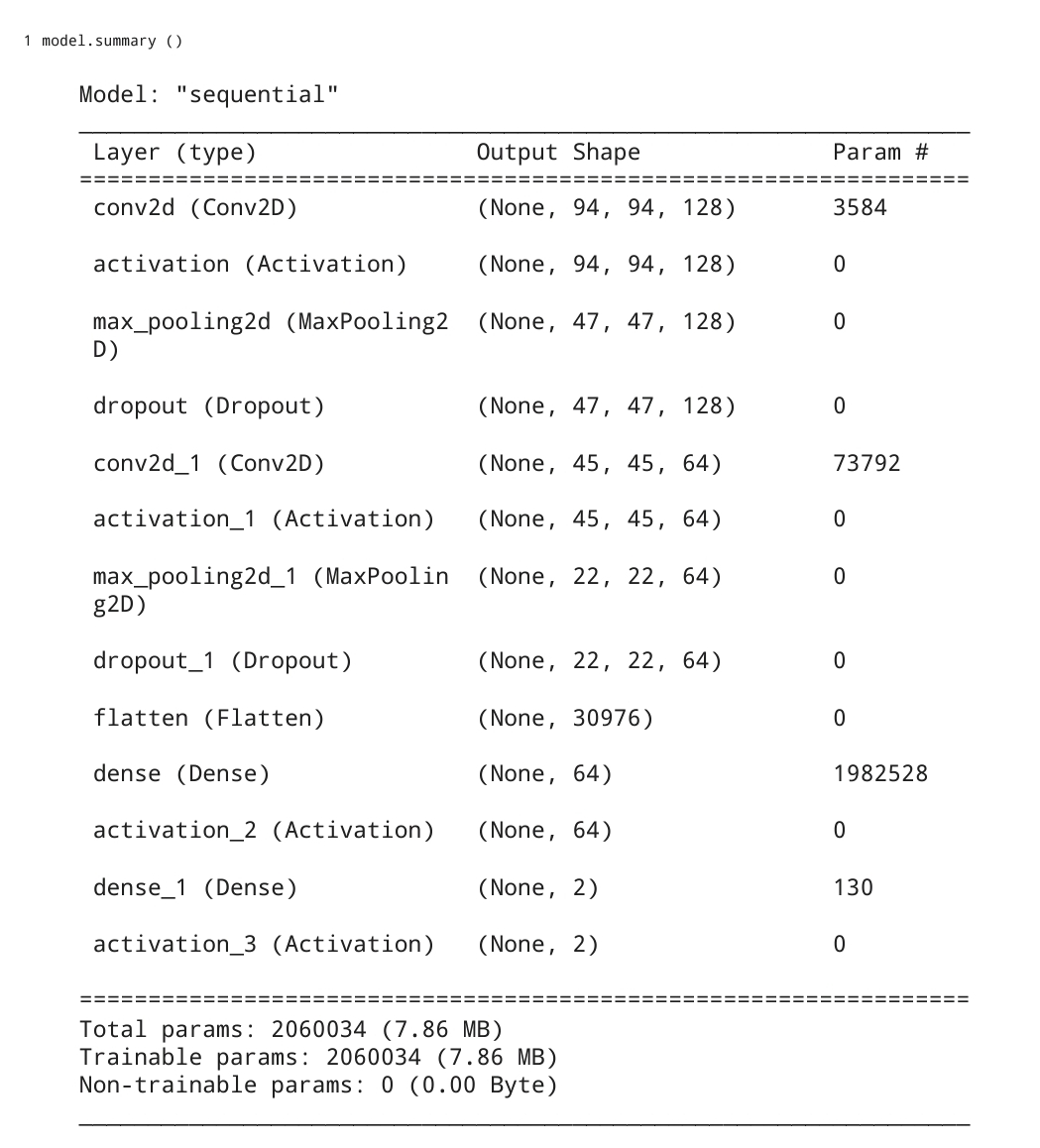
The piezoelectric vibrator has two pins, positive and ground. The two pins were used to provide power to the vibrator. The positive pin was connected to pin 15 of the microcontroller and the ground was connected to ground rail. This ensured that the vibrator received a signal at the appropriate time.

Figure 4.5 Piezoelectric Vibrator

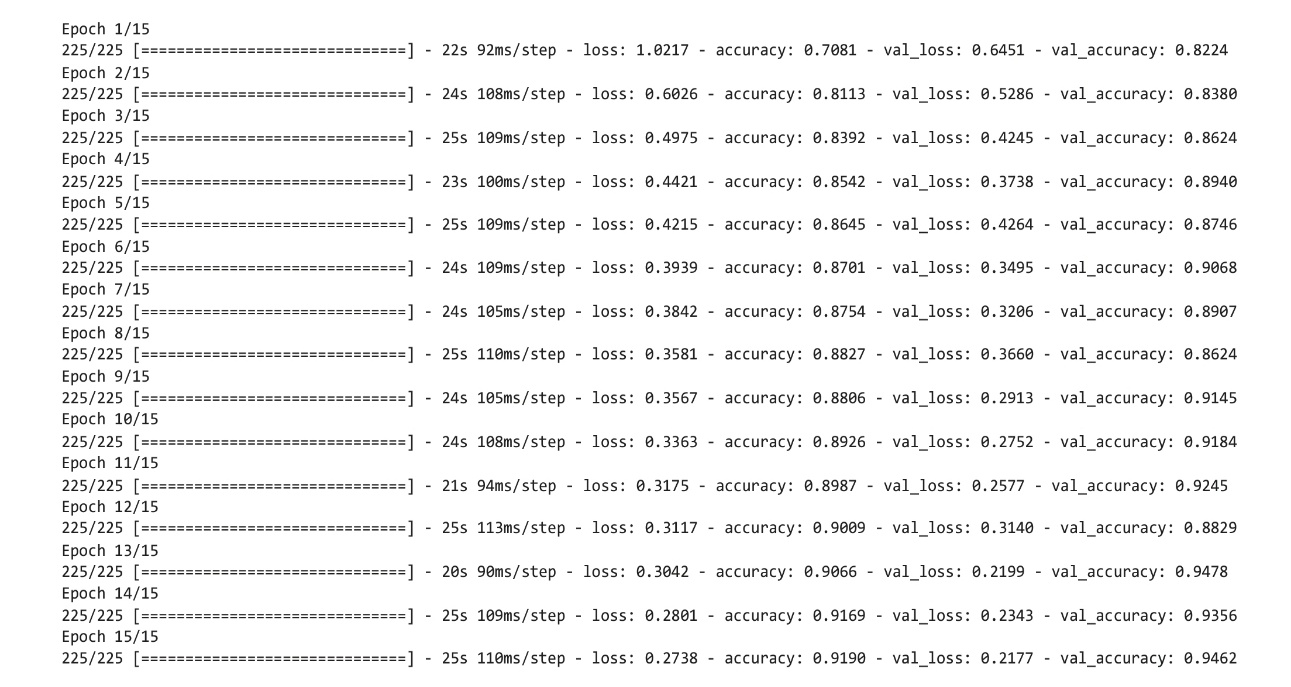
## 4.2.7 Implementing machine learning model

Google Colab was used to develop and train the model. The best CNN architecture was chosen as per the dataset size and number of features. The model was hosted on Google’s cloud and training required authentication through Google’s account.

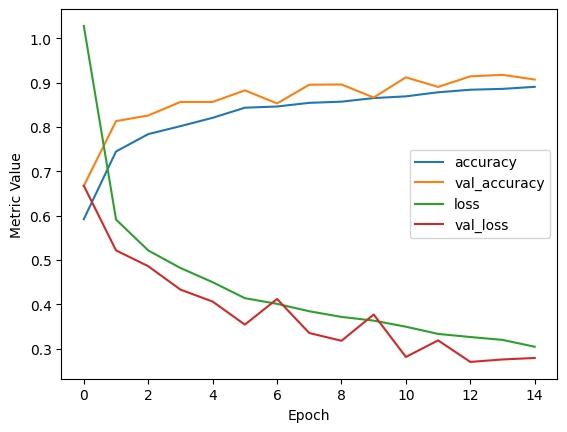
After tweaking the hyperparameters, this was the final architecture that was used to train the model with a validation accuracy of 0.9, which was within the acceptable range. Figure 4.6 shows the final model architecture that was used to develop the model.

Figure 4.6 Model Summary

The model was trained for 15 epochs to obtain the validation accuracy of 0.9 with a validation loss of 0.2

Figure 4.7 Model Training Progress

The training graph over number of epochs and metric value was plotted and was used to display the results from figure 4.7 in graphical format for the various parameters.

Figure 4.8 Graphical representation of model training

This model was tested by developing a test environment which consisted of a camera to take picture, a storage bucket in Google drive and a python script that would read the image and pass it through the model for detection. By using this configuration, the model was validated to be doing well with real world images. Deployment was later done by using edge impulse API to obtain the cpp library.

## 4.2.7 Implementing the audio

Implementing audio proved to be challenge due to the limited DAC(Digital to Analog) pins on esp32 boards. Esp32 boards come with two DAC pins which can be used to convert digital signal to analog signal, which is what works best in audio applications. In this project, the use of esp32cam came up with challenges as both DAC pins were already used by the camera.

This resulted to either using external DAC or using pwm(Pulse Width Modulation) to simulate an analog signal. Using external DAC was not possible due to the limited number of pins on the board as this process required three extra pins and one of the pins was to be clock pin. Thus using pwm was the only available method, which would impact the quality of the audio signal.

Using pwm required sampling the audio in digital form and varying the duty cycle depending on the frequency at any given moment. This method also had a challenge, convectional audio libraries were hard coded to use the DAC pins on the esp32 and modifying the source code caused conflict in esp32’s timers as different channels have different timers.

In order to overcome the above challenge, we wrote a program that would read the digital audio bytes and pass them through the pwm pins of the microcontroller, through a filter so as to remove the high frequencies.

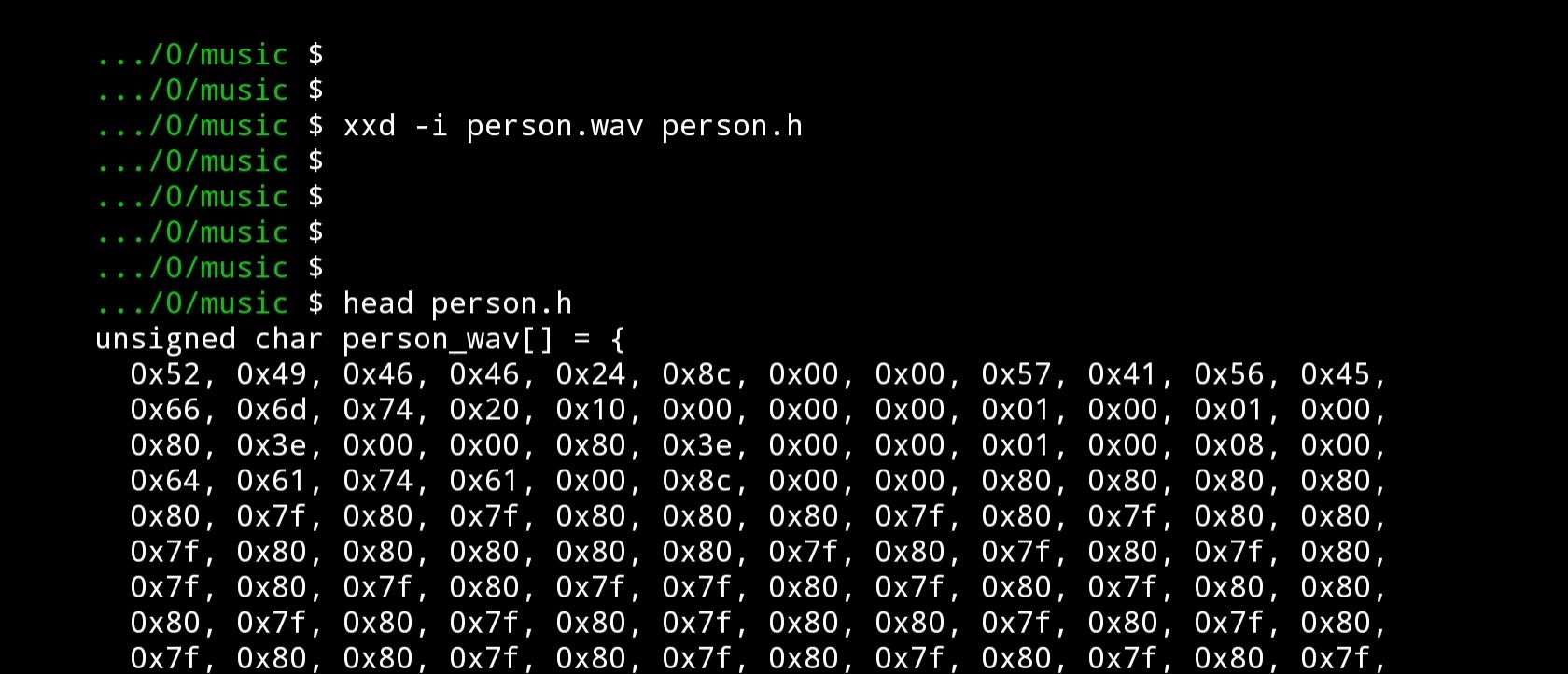
Converting the audio file to it’s hexadecimal equivalent required use of audacity to sample the audio to a lower resolution and damping the bytes using xxd to obtain a C type array as shown in figure 4.9

Figure 4.9 Audio Hexdump

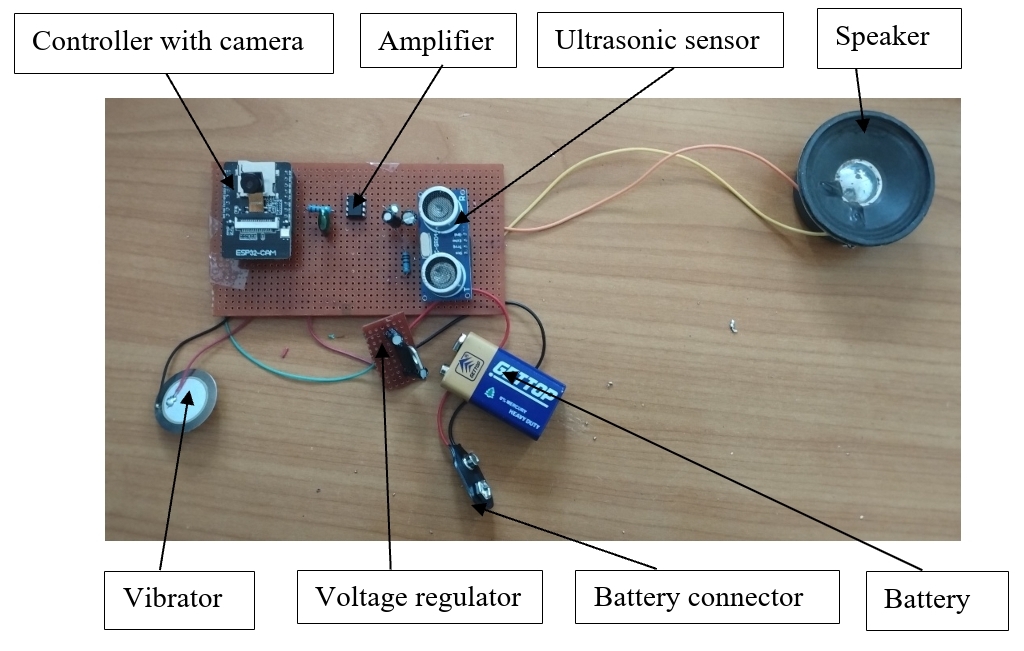
The obtained header file containing the arrays can then be imported in Arduino ide and used to simulate the audio. On testing, the speaker didn’t not provide the espected audio but there was a noticeable sound.

## 4.3 Fabrication stage

After testing the circuit on the breadboard and validating it’s working, the various components were transferred to a strip board as per the circuit diagram in chapter 3. The metal leads of resistors and capacitors were trimmed to a shorter length and then soldering was done to permanently mount the components.

Before powering the circuit, a multimeter was used to test continuity between the various solder joints to ensure that no short circuit would occur on powering the system.

This stage was crucial as the strip board would be transferred to a casing and the device be complete.

Figure 4.10 AI Assisted Navigation Device

## 4.4 System Testing

After fabricating the device on the strip board, testing was conducted on the prototype to verify it was working as intended and correct any errors that may have come up.

Testing results were recorded in the table below and providing relevant comments.

Table 4.2 Results

|  |  |  |  |
| --- | --- | --- | --- |
| Parameters | Expected | Measured | Comments |
| Voltage supply | 9V | 7.8V (on no load)  5.6 (on full load) | Voltage drop on full load was not expected, adding another battery to increase the overall current will be tested. |
| Voltage regulated | 5V | 5V (on no load)  2.3 (on full load) | On full load the device could not run the controller, testing was done using power supplied by computer’s usb port |
| Object detection | Detect person and car with a 0.9 accuracy | Detected person with more than 0.85 accuracy on well lit environment, and cars with 0.8 accuracy. | Detection worked as expected, within the expected range |
| Audio | Expected to say “car ahead” on detecting a car and say “person ahead” on detecting a person | Audio playback was not possible at the time of testing. | Audio playback was not possible due to the high frequency components on the signal, expected to use a better filter. |
| Vibration | Expected to vibrate when an object was less than 40cm. | Vibrated when object got too close. | Worked as expected, within the expected range. |

# CHAPTER FIVE: CONCLUSION AND RECOMMENDATION

## 5.1 Overview

This chapters covers the summary of the entire project, what was achieved, challenges encountered during design, implementation and testing and recommendations.

## 5.2 Statement of initial objectives

The main objective was to design a machine learning model that would be able to classify images of people and car, and run locally on the microcontroller. This was achieved and the model had a 0.9 validation accuracy, which was within the expected range. Converting the model to Arduino library dropped the accuracy to 0.85, which was within the expected range. On running the model locally, inferencing time was 500ms to 700ms which was near real time.

The secondary objectives were to power the device with a constant 5v portable power supply and to design an audio system. The voltage regulation worked and provided a constant 5v. The audio system proved challenging as there were no compatible libraries to work with it as discussed in the previous chapter. Testing on the new code had elements of audible high frequency tones that cancelled out the expected voice.

## 5.3 Summary of achievements

Upon implementing the circuit and the program, image classification was successful but displaying the results was only possible on a serial monitor as audio playing was not possible as per the time of this report.

## 5.4Applications of the system

With improvement to the audio, the system can be used to help the blind navigate areas with high traffic of both people and cars. This can limit injuries that can occur from collision.

Further improvement of the model can be made to have a wide range of objects detected thus useful in day to day applications.

## 5.5 Benefits of the study

This project covered a wide range of study, thus enabled us to learn. Some of the concepts we learnt were on machine learning and how neural networks are designed. We designed a convolution neural network, this helping us understand concepts involved in deep learning. Working with audio also helped us understand how sampling affects different audio formats and how to dump any audio to it’s respective digital bits. By conducting this study, we were able to learn how concepts we learnt in previous school units are used to develop circuits in the real world.

## 5.6 Challenges faced during design and implementation

This project was conducted mostly using Google colab, which at times the GPUs were not a available thus training speed was limited by the CPU.

Another challenge faced was lack of compatible audio library this resulting to using pwm to play the audio. This resulted in loss of quality of the expected audio.

## 5.7 Recommendations

In order to improve the device, some of the recommendations that can be done are:

* Finding a way to have Audio without relying on pwm.
* Improving the voltage supply circuit to have sufficient voltage and avoid the stated voltage drops.
* Using a stronger tactile feedback mechanism rather than relying on Piezoelectric Vibrator.

## 5.8 Conclusion

In conclusion, the device that was proposed in this project was implemented and constructed. The main objective being to be able to detect people and cars was achieved by use of CNN model. This model was ported to a lite format that was converted to arduino compatible library, which was later uploaded to the device. Inferencing worked as expected with a 0.9 accuracy. Audio playback was not possible as per the time of this report but fixes are being worked on.

# References

1. Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017, August). Understanding of a convolutional neural network. In 2017 international conference on engineering and technology (ICET) (pp. 1-6). Ieee.
2. Boston Dynamics. (2023, November 16). *Boston dynamics*. Retrieved from Spot Boston dynamics: https://bostondynamics.com/products/spot/ [Accessed 12th November, 2023]
3. Bousbia-Salah, M., Maamar, B., & Larbi, A. (2011). A navigation aid for blind people. *Journal of Intelligent & Robotics Systems*, 64, 387 - 400.
4. El Naqa, I., & Murphy, M. J. (2015). What is machine learning? . *Springler*, 3-11.
5. Hymel, S. B., Situnayake, D., Elium, A. W., Kelcey, M., & Reddi, V. J. (2022). Edge Impulse: An MLOps Platform for Tiny Machine Learning. *arXiv preprint arXiv:2212.03332.*
6. Ketkar, N., & Moolayil, J. (2021). Convolutional neural networks. Deep Learning with Python. *Learn Best Practices of Deep Learning Models with PyTorch*, 197-242.
7. Koharwal, S., Awwad, S. B., & Vyakaranam, A. (2019). Navigation system for blind - Third eye. *International Journal of Innovative Technology And Exploring Engineering*, 1086-1092.
8. Lin, J., Zhu, W. M., Wang, W.-C., Gan, C., & Han, S. (n.d). *On-Device Training Under 256KB Memory.* MIT-IBM Watson AI Lab.
9. Mahesh, B. (2020). Machine learning algorithms-a review. . *International Journal of Science and Research (IJSR)*, 381-386.
10. Martinez, J. C. (2021, march 15). *Infotech*. Retrieved from infotech.report: https://infotech.report/guest-articles/real-time-object-detection-in-video-with-intro-to-yolo-v3[Accessed 12th , November 2023]
11. Mintz, Y., & Brodie, R. (2019). *Mintz, Y., Introduction to artificial intelligence in medicine. Minimally Invasive Therapy & Allied Technologies.*
12. Thaler, L., Arnott, S. R., & Goodale, M. A. (2011). *Thaler, L., ArnoNeural correlates of natural human echolocation in early and late blind echolocation experts.*
13. Torrey, L., & Shavlik, J. (2010). Transfer learning. In Handbook of research on machine learning applications and trends: algorithms, methods, and techniques. *IGI global.*, 242-264.
14. Warden, P., & Situnayake, D. (2019). Tinyml: Machine learning with tensorflow lite on arduino and ultra-low-power microcontrollers. *O'Reilly Media*.
15. World Health Organization. (2020, october 9). *World health organization*. Retrieved from who: https://www.who.int/news/item/09-10-2003-up-to-45-million-blind-people-globally---and-growing#:~:text=Globally%2C%20an%20estimated%2040%20to,2020%3A%20The%20Right%20to%20Sight. [Accessed 8th November, 2023]

**APPENDICES**

**Appendix A : Machine learning model training and deployment code**

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

“””RGB custom data.ipynb

Automatically generated by Colaboratory.

Original file is located at

https://colab.research.google.com/drive/1Qkb3cdRrH7VlftB0y\_k5X1qJ68yRYOH6

Data preprocessing

“””

Import numpy as np

Import os

Import matplotlib.pyplot as plt

Import cv2

From google.colab import drive

Model\_name = ‘RGB’

# Mount Google Drive

Drive.mount(‘/content/drive’)

# Create a folder in Google Drive

Data\_path = ‘/content/drive/MyDrive/dataset’

Os.makedirs(data\_path, exist\_ok=True)

Print(f”Folder ‘{data\_path}’ created in Google Drive.”)

Save\_directory = ‘/content/drive/MyDrive/models’

Os.makedirs(save\_directory, exist\_ok=True)

Save\_path = os.path.join (save\_directory, f”{model\_name}.keras”)

Save\_directory = ‘/content/drive/MyDrive/models’

Os.makedirs(save\_directory, exist\_ok=True)

Save\_path\_lite = os.path.join (save\_directory, f”{model\_name}.tflite”)

Test\_dir = ‘/content/drive/MyDrive/Test\_img’

Os.makedirs(Test\_dir, exist\_ok=True)

Test\_folder= ‘/content/drive/MyDrive/Test\_img/test’

Os.makedirs(Test\_folder, exist\_ok=True)

Ard\_dir = ‘/content/drive/MyDrive/Arduino’

Os.makedirs(Ard\_dir, exist\_ok=True)

Lib\_folder= ‘/content/drive/MyDrive/Arduino/Library’

Os.makedirs(Lib\_folder, exist\_ok=True)

DATADIR = data\_path

CATEGORIES = [‘person’,’car’]

IMG\_SIZE = 96

Training\_data = []

Def create\_training\_data ():

For category in CATEGORIES:

Path = os.path.join (DATADIR, category)

Class\_num = CATEGORIES.index (category)

For img in os.listdir (path):

Try:

Img\_array = cv2.imread (os.path.join(path, img))

Img\_array = cv2.cvtColor(img\_array,cv2.COLOR\_BGR2RGB)

New\_array = cv2.resize (img\_array, (IMG\_SIZE, IMG\_SIZE))

New\_array = new\_array/255

Training\_data.append ([new\_array, class\_num])

Except Exception as e:

Pass

Return class\_num

Create\_training\_data ()

Save\_directory =’/content/drive/MyDrive/Training\_data’

Os.makedirs(save\_directory, exist\_ok=True)

Class\_path = os.path.join (save\_directory, “class\_num.txt”)

With open(class\_path, ‘w’) as file:

File.write(str(class\_num))

Import random

For i in range (3):

Random.shuffle (training\_data)

Print (len(training\_data))

For sample in training\_data[:20]:

Print (sample[1] )

From tensorflow.keras.utils import to\_categorical

X = []

Y = []

For features, label in training\_data:

X.append (features)

append (label)

X = np.array (X).reshape (-1,IMG\_SIZE,IMG\_SIZE,3)

Y = np.array (y)

Count\_zeros = np.count\_nonzero(y == 0)

Print (count\_zeros)

Y = to\_categorical(y, num\_classes=2)

“””Data saving”””

#saving training data

Import pickle

Save\_directory =’/content/drive/MyDrive/Training\_data’

Os.makedirs(save\_directory, exist\_ok=True)

Save\_path\_X = os.path.join (save\_directory, f”X-{model\_name}.pickle”)

Save\_path\_y = os.path.join (save\_directory, f”y-{model\_name}.pickle”)

Pickle\_out = open (save\_path\_X, “wb”)

Pickle.dump (X, pickle\_out)

Pickle\_out.close ()

Pickle\_out = open (save\_path\_y, “wb”)

Pickle.dump (y, pickle\_out)

Pickle\_out.close ()

#loading training data

Import pickle

Save\_directory =’/content/drive/MyDrive/Training\_data’

Os.makedirs(save\_directory, exist\_ok=True)

Save\_path\_X = os.path.join (save\_directory, f”X-{model\_name}.pickle”)

Save\_path\_y = os.path.join (save\_directory, f”y-{model\_name}.pickle”)

Pickle\_in = open (save\_path\_X, “rb”)

X = pickle.load (pickle\_in)

Pickle\_in = open (save\_path\_y, “rb”)

Y = pickle.load (pickle\_in)

Print (X.shape)

Print (y.shape)

Print (y[10:])

“””Data Augmentation”””

From tensorflow.keras.preprocessing.image import ImageDataGenerator

Datagen = ImageDataGenerator(

Rotation\_range=20,

Width\_shift\_range=0.15,

Height\_shift\_range=0.2,

Shear\_range=0.2,

Zoom\_range=0.3,

Horizontal\_flip=True,

Fill\_mode=’nearest’

)

“””CNN model”””

Import tensorflow as tf

From tensorflow.keras.models import Sequential

From tensorflow.keras.optimizers import Adam

From tensorflow.keras.callbacks import ModelCheckpoint

From tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D, Dropout, Activation

From tensorflow.keras import regularizers

From sklearn.model\_selection import train\_test\_split

L\_r = 0.0005

Epochs = 15

Opt = Adam(learning\_rate = l\_r)

Batch\_size = 32

X\_train, X\_val, y\_train, y\_val = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

Model = Sequential()

Model.add(Conv2D(128, (3, 3), input\_shape=X.shape[1:],kernel\_regularizer=regularizers.l1(0.01)))

Model.add(Activation(“relu”))

Model.add(MaxPooling2D(pool\_size=(2, 2)))

Model.add(Dropout(0.2))

Model.add(Conv2D(64, (3,3)))

Model.add(Activation(“relu”))

Model.add(MaxPooling2D(pool\_size = (2,2)))

Model.add(Dropout(0.1))

Model.add(Flatten())

Model.add(Dense(64))

Model.add(Activation(“relu”))

Model.add(Dense(2))

Model.add(Activation (“softmax”))

Model.compile(loss = “binary\_crossentropy”,

Optimizer = opt,

Metrics = [“accuracy”])

Datagen.fit(X\_train)

Train\_generator = datagen.flow(X\_train, y\_train, batch\_size = batch\_size)

#print(type(train\_generator))

History = model.fit(train\_generator, steps\_per\_epoch=len(X\_train) // batch\_size, epochs=Epochs, validation\_data=(X\_val, y\_val))

#checkpoint = ModelCheckpoint(save\_path,save\_freq = ‘epoch’, monitor=’val\_loss’, save\_best\_only=True, save\_weights\_only=False, mode=’auto’)

#history = model.fit(X,y, batch\_size = 32, epochs = Epochs, verbose = 1, callbacks = [checkpoint])

Model.summary ()

Model.evaluate(X\_val,y\_val)

Model.save(save\_path)

Import matplotlib.pyplot as plt

Plt.plot(history.history[‘accuracy’], label=’accuracy’)

Plt.plot(history.history[‘val\_accuracy’], label=’val\_accuracy’)

Plt.plot(history.history[‘loss’], label=’loss’)

Plt.plot(history.history[‘val\_loss’], label=’val\_loss’)

Plt.xlabel(‘Epoch’)

Plt.ylabel(‘Metric Value’)

Plt.legend()

Plt.show()

“””Random images test”””

Import tensorflow as tf

Model = tf.keras.models.load\_model(save\_path)

Print(f”Model loaded successfully: {model}”)

Directory\_path = Test\_dir

Folder\_name = Test\_folder

Test\_folder\_path = os.path.join(directory\_path, folder\_name)

IMG\_SIZE = 96

DATADIR = data\_path

CATEGORIES = [‘person’,’car’]

If os.path.exists(test\_folder\_path):

Image\_files = [f for f in os.listdir(test\_folder\_path) if os.path.isfile(os.path.join(test\_folder\_path, f))]

For image\_file in image\_files:

Image\_path = os.path.join(test\_folder\_path, image\_file)

Img = cv2.imread(image\_path)

Img = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)

Plt.imshow (img)

Plt.show ()

Img = cv2.resize(img,(IMG\_SIZE,IMG\_SIZE))

Img = img/255

Input\_img = np.expand\_dims(img, axis=0)

Predictions = model.predict([input\_img])

Predicted\_class = CATEGORIES[np.argmax(Predictions)]

Confidence = np.max(Predictions)

If confidence >= 0.8:

Print (predicted\_class, confidence)

Else:

Print (“unrecognisable class”)

Else:

Print(f”The folder ‘{folder\_name}’ doesn’t exist”)

“””Converting to tflite”””

Import tensorflow as tf

IMG\_SIZE = 96

DATADIR = data\_path

CATEGORIES = [‘person’, ‘car’]

Def representative\_data\_gen():

Data\_dir = DATADIR

Batch\_size = 32

Img\_height = IMG\_SIZE

Img\_width = IMG\_SIZE

Class\_names = CATEGORIES

Train\_images = tf.keras.preprocessing.image\_dataset\_from\_directory(

Data\_dir,

Class\_names=class\_names,

Validation\_split=0.2,

Subset=”training”,

Seed=123,

Image\_size=(img\_height, img\_width),

Batch\_size=batch\_size)

#standardize the images

Normalization\_layer = tf.keras.layers.experimental.preprocessing.Rescaling(1/255)

Normalized\_ds = train\_images.map(lambda x, y: (normalization\_layer(x), y))

Image\_batch, labels\_batch = next((iter(normalized\_ds)))

First\_image = image\_batch[0]

#print(image\_batch)

# Notice the pixels values are now in `[0,1]`.

Print(np.min(first\_image), np.max(first\_image))

For input\_value in tf.data.Dataset.from\_tensor\_slices(image\_batch).batch(1).take(100):

# Model has only one input so each data point has one element.

Yield [input\_value]

Converter = tf.lite.TFLiteConverter.from\_keras\_model(model)

Converter.optimizations = [tf.lite.Optimize.DEFAULT]

Converter.representative\_dataset = representative\_data\_gen

Converter.target\_spec.supported\_ops = [tf.lite.OpsSet.TFLITE\_BUILTINS\_INT8]

Converter.inference\_input\_type = tf.uint8

Converter.inference\_output\_type = tf.uint8

Tflite\_quant\_model = converter.convert()

With open(save\_path\_lite, “wb”) as f:

f.write(tflite\_quant\_model)

interpreter = tf.lite.Interpreter(save\_path\_lite)

interpreter.allocate\_tensors()

input\_details = interpreter.get\_input\_details()

output\_details = interpreter.get\_output\_details()

print(“Input details:”, input\_details)

print(“Output details:”, output\_details)

summary = interpreter.get\_signature\_list()

print(“Model Summary:”, summary)

input\_tensor\_shape = input\_details[0][‘shape’]

output\_tensor\_shape = output\_details[0][‘shape’]

input\_tensor\_type = input\_details[0][‘dtype’]

output\_tensor\_type = output\_details[0][‘dtype’]

print(“Input Tensor Shape:”, input\_tensor\_shape)

print(“Output Tensor Shape:”, output\_tensor\_shape)

print(“Input Tensor Type:”, input\_tensor\_type)

print(“Output Tensor Type:”, output\_tensor\_type)

“””Model deployment”””

!pip install edgeimpulse

Import edgeimpulse as ei

Ei.API\_KEY = API\_KEY

Ei.model.list\_profile\_devices()

Profile = ei.model.profile(model=tflite\_quant\_model, device=’espressif-esp32’)

Print(profile.summary())

#print(f”Estimated RAM usage: {profile.model.profile\_info.float32.memory.tflite.ram}”)

#print(f”Estimated ROM usage: {profile.model.profile\_info.float32.memory.tflite.rom}”)

#print(f”Estimated inference time (ms): {profile.model.profile\_info.float32.time\_per\_inference\_ms}”)

Ei.model.list\_deployment\_targets()

Ei.model.deploy(model=tflite\_quant\_model,

Model\_input\_type=ei.model.input\_type.OtherInput(),

Model\_output\_type=ei.model.output\_type.Classification(),

Output\_directory=Lib\_folder)

**Appendix B : Arduino code**

#include <person\_and\_car\_perfect\_model\_inferencing.h>

#include “edge-impulse-sdk/dsp/image/image.hpp”

#include “esp\_camera.h”

#include “sound.h”

//sensors pins

#define trigPin 12

#define echoPin 13

#define vibratorPin 15

#define SpeakerPin 14

//constants

Const int sampleRate = 16000;

//camera pins

#define PWDN\_GPIO\_NUM 32

#define RESET\_GPIO\_NUM -1

#define XCLK\_GPIO\_NUM 0

#define SIOD\_GPIO\_NUM 26

#define SIOC\_GPIO\_NUM 27

#define Y9\_GPIO\_NUM 35

#define Y8\_GPIO\_NUM 34

#define Y7\_GPIO\_NUM 39

#define Y6\_GPIO\_NUM 36

#define Y5\_GPIO\_NUM 21

#define Y4\_GPIO\_NUM 19

#define Y3\_GPIO\_NUM 18

#define Y2\_GPIO\_NUM 5

#define VSYNC\_GPIO\_NUM 25

#define HREF\_GPIO\_NUM 23

#define PCLK\_GPIO\_NUM 22

//Constants

#define EI\_CAMERA\_RAW\_FRAME\_BUFFER\_COLS 320

#define EI\_CAMERA\_RAW\_FRAME\_BUFFER\_ROWS 240

#define EI\_CAMERA\_FRAME\_BYTE\_SIZE 3

//Private variables

Static bool debug\_nn = false; // Set this to true to see e.g. features generated from the raw signal

Static bool is\_initialised = false;

Uint8\_t \*snapshot\_buf; //points to the output of the capture

Static camera\_config\_t camera\_config = {

.pin\_pwdn = PWDN\_GPIO\_NUM,

.pin\_reset = RESET\_GPIO\_NUM,

.pin\_xclk = XCLK\_GPIO\_NUM,

.pin\_sscb\_sda = SIOD\_GPIO\_NUM,

.pin\_sscb\_scl = SIOC\_GPIO\_NUM,

.pin\_d7 = Y9\_GPIO\_NUM,

.pin\_d6 = Y8\_GPIO\_NUM,

.pin\_d5 = Y7\_GPIO\_NUM,

.pin\_d4 = Y6\_GPIO\_NUM,

.pin\_d3 = Y5\_GPIO\_NUM,

.pin\_d2 = Y4\_GPIO\_NUM,

.pin\_d1 = Y3\_GPIO\_NUM,

.pin\_d0 = Y2\_GPIO\_NUM,

.pin\_vsync = VSYNC\_GPIO\_NUM,

.pin\_href = HREF\_GPIO\_NUM,

.pin\_pclk = PCLK\_GPIO\_NUM,

.xclk\_freq\_hz = 20000000,

.ledc\_timer = LEDC\_TIMER\_0,

.ledc\_channel = LEDC\_CHANNEL\_0,

.pixel\_format = PIXFORMAT\_JPEG, //YUV422,GRAYSCALE,RGB565,JPEG

.frame\_size = FRAMESIZE\_QVGA, //QQVGA-UXGA Do not use sizes above QVGA when not JPEG

.jpeg\_quality = 12, //0-63 lower number means higher quality

.fb\_count = 1, //if more than one, i2s runs in continuous mode. Use only with JPEG

.fb\_location = CAMERA\_FB\_IN\_PSRAM,

.grab\_mode = CAMERA\_GRAB\_WHEN\_EMPTY,

};

//Function definitions

Bool ei\_camera\_init(void);

Void ei\_camera\_deinit(void);

Bool ei\_camera\_capture(uint32\_t img\_width, uint32\_t img\_height, uint8\_t \*out\_buf) ;

Void setup()

{

pinMode(trigPin, OUTPUT);

pinMode(echoPin, INPUT);

pinMode(SpeakerPin, OUTPUT);

pinMode(vibratorPin, OUTPUT);

pinMode(LedPin, OUTPUT);

Serial.begin(115200);

While (!Serial);

Serial.println(“Edge Impulse Inferencing”);

If (ei\_camera\_init() == false) {

Ei\_printf(“Failed to initialize Camera!\r\n”);

}

Else {

Ei\_printf(“Camera initialized\r\n”);

}

Ei\_printf(“\nStarting continious inference in 2 seconds...\n”);

}

//function to play audio

Void playAudio(const unsigned char\* audioArray) {

Unsigned int audioLength = 35884;

For (unsigned int i = 44; i < audioLength; i++) {

Uint8\_t sample = pgm\_read\_byte(&audioArray[i]);

analogWrite(SpeakerPin,sample);

delayMicroseconds(1000000 / sampleRate);

}

}

Void loop()

{

// Trigger pulse

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

// Read the echo pulse duration

Long duration = pulseIn(echoPin, HIGH);

// Calculate distance in centimeters

Int distance = duration \* 0.034 / 2;

// Print distance to Serial Monitor

Serial.print(“Distance: “);

Serial.print(distance);

Serial.println(“ cm”);

// Check if distance is 50cm or below to switch on the vibrator

If (distance <= 40) {

analogWrite(vibratorPin, 255); // Turn on the vibrator

} else {

analogWrite(vibratorPin, 0); // Turn off the vibrator

}

Delay (200);

//EDGE IMPULSE INFERENCING

If (ei\_sleep(5) != EI\_IMPULSE\_OK) {

Return;

}

Snapshot\_buf = (uint8\_t\*)malloc(EI\_CAMERA\_RAW\_FRAME\_BUFFER\_COLS \* EI\_CAMERA\_RAW\_FRAME\_BUFFER\_ROWS \* EI\_CAMERA\_FRAME\_BYTE\_SIZE);

// check if allocation was successful

If(snapshot\_buf == nullptr) {

Ei\_printf(“ERR: Failed to allocate snapshot buffer!\n”);

Return;

}

Ei::signal\_t signal;

Signal.total\_length = EI\_CLASSIFIER\_INPUT\_WIDTH \* EI\_CLASSIFIER\_INPUT\_HEIGHT;

Signal.get\_data = &ei\_camera\_get\_data;

If (ei\_camera\_capture((size\_t)EI\_CLASSIFIER\_INPUT\_WIDTH, (size\_t)EI\_CLASSIFIER\_INPUT\_HEIGHT, snapshot\_buf) == false) {

Ei\_printf(“Failed to capture image\r\n”);

Free(snapshot\_buf);

Return;

}

// Run the classifier

Ei\_impulse\_result\_t result = { 0 };

EI\_IMPULSE\_ERROR err = run\_classifier(&signal, &result, debug\_nn);

If (err != EI\_IMPULSE\_OK) {

Ei\_printf(“ERR: Failed to run classifier (%d)\n”, err);

Return;

}

#if EI\_CLASSIFIER\_OBJECT\_DETECTION == 1

Bool bb\_found = result.bounding\_boxes[0].value > 0;

For (size\_t ix = 0; ix < result.bounding\_boxes\_count; ix++) {

Auto bb = result.bounding\_boxes[ix];

If (bb.value == 0) {

Continue;

}

}

If (!bb\_found) {

Ei\_printf(“ No objects found\n”);

}

#else

For (size\_t ix = 0; ix < EI\_CLASSIFIER\_LABEL\_COUNT; ix++) {

String prediction;

float confidence;

Prediction = result.classification[ix].label;

Confidence = result.classification[ix].value;

If (prediction == “person” && confidence >= 0.8) {

playAudio(person\_wav);

}

Else if (prediction == “car” && confidence >= 0.8)

{playAudio(car\_wav);

}

}

#endif

#if EI\_CLASSIFIER\_HAS\_ANOMALY == 1

Ei\_printf(“ anomaly score: %.3f\n”, result.anomaly);

#endif

Free(snapshot\_buf);

........