**THE UNIVERSITY OF ZAMBIA**

SCHOOL OF NATURAL SCIENCES

DEPARTMENT OF COMPUTER SCIENCE

****

**SMART CAR PARKING SYSTEM BASED ON AUTOMATIC LICENCE PLATE RECOGNITION SYSTEMS AND CLOUD TECHNOLOGIES**

ANDYSON MUPETA 2018258460

CSC4004 FINAL YEAR PROJECT

A report submitted in partial fulfilment of the requirements for a Bachelor’s Degree in Computer Science.

DATE: 15/10/2022

**DECLARATION**

I hearby declare that the ‘AUTOMATIC NUMBER PLATE RECOGNITION SYSTEM FOR BOOM GATES ON UNZA ENTRANCE AND PARKING SPACE GATES’ is the work of ANDYSON MUPETA and not any other individual or group I may know under the UNIVERSITY OF ZAMBIA. All sources of information and inspiration have been indicated in the references section at the end of this document.

**CREATOR:**

**Name:**

**Signature:**

**Date:**

**SUPERVISOR:**

**Name:**

**Signatures:**

**Date:**

**ACKNOWLEDGEMENT**

I would like to express my gratitute and appreciation to all those who gave me the possibility to complete this project and report. Special thanks is due to my supervisor Dr. Jackson Phiri whose help, consistency, stimulating suggestions and encouragement helped me greatly in completing this project, not forgetting all the great members of staff at the Computer Science department for their support.

I would also like to acknowledge with much appreciation the crucial role my parents, family and friends played in helping me complete this project and my university education. Most important of all I want to thank God for making all this a possibilty and been with me through every step of the journey.

**ABSTRACT**

Automatic license plate recognition (ALPR) is the process of retrieving license plate information from a captured image or video frames from a sequence of videos. ALPR can assist law enforcement officers to identify stolen vehicles or to capture vehicle information from those that violate traffic laws instantly. It is also commonly used as an electronic payment system for toll payment or parking fee payment and in the case of this project it is used for parking lot access control management. Traditionally, ALPR is installed in a PC-based platform to take advantage of its processing power to process high-quality images captured by high-resolution cameras. The main purpose of this project is to provide a means of smart parking management and control based on weather an individual is registered in the system or not. This is achieved by using a camera attached to a microcontroller on the gate which then uses Optical Character Recognition to extract characters from a number plate and Cloud Technologies to check weather an individual has been registered or not.

**TABLE OF CONTENTS**

DECLERATION**i**

ACKNOWLEDGEMENT**ii**

ABSTRACT**iii**

INTRODUCTION**1**

Background1.1

Problem Statement1.2

Aim1.3

Objectives1.4

Research Questions1.5

Scope of the Project1.6

Justification1.7

Outline of Project1.8

Chapter Summary1.9

LITERATURE REVIEW**2**

Introduction2.1

Steps2.2

Image Acquisition and Preprocessing2..2.1

License Plate Localization2..2.2

Segmentation2..2.3

Character Recognition2..2.4

Operations2.3

Chapter Summary2.4

METHODOLOGY**3**

Software Tools3.1

Programming Languages3.1.1

Frameworks and Libraries3.1.2

IDEs and Text Editors3..1.3

Hardware Tools3.2

Microcontrollers3.2.1

Pi Camera3.2.2

Relays3.2.3

HC-SR04 UltraSonic Sensor3.2.4

HC-05 Bluetooth Module3.2.5

Car Wiper Motor3.2.6

Other Hardware Components3.2.7

Sytem Implementation Diagrams3.3

Number Plate Character Recognition Flow Diagram3.3.1

Microcontroller Circuit Diagram3.3.2

Software Flow Diagram3.3.3

System Flow Diagram3.2.4

Other Hardware Components3.2.7

Chapter Summary3.4

RESULTS**4**

Admin Web App4.1

User Mobile Application4.2

Application Programming Interface4.3

Arduino Uno Program4.4

Raspberry Pi Program4.5

Arduino Uno Setup4.6

Raspberry Pi Setup4.7

Charge Controller and Solar Panel4.8

12V DC Motor4.9

Housing Unit4.10

Chapter Summary4.11

DISCUSSION AND CONCLUSION**5**

REFERENCES**6**

APPENDIX**7**

Arduino Uno Code7.1

Raspberry Pi Code7.2

Number Plate Recognition Software Code7.3

**TABLE OF FIGURES**

DECLERATION**i**

ACKNOWLEDGEMENT**ii**

ABSTRACT**iii**

INTRODUCTION**1**

1. **INTRODUCTION**

This chapter introduces the project. It gives a general overview of the project that is the problems this project addresses and its aims and objectives

* 1. **BACKGROUND**

Automatic number-plate recognition (ANPR) is a technology that uses optical character recognition (the electronic or mechanical conversion of images of typed, handwritten or printed text into machine-encoded text) on images to read vehicle registration plates. It can use existing closed-circuit television, road-rule enforcement cameras, or cameras specifically designed for the task. ANPR is used by police forces around the world for law enforcement purposes, including to check if a vehicle is registered or licensed. It is also used for electronic toll collection on pay-per-use roads and as a method of cataloguing the movements of traffic, for example by highways agencies. Automatic number-plate recognition can be used to store the images captured by the cameras as well as the text from the license plate, with some configurable to store a photograph of the driver. Systems commonly use infrared lighting to allow the camera to take the picture at any time of day or night.ANPR technology must take into account plate variations from place to place.

Privacy issues have caused concerns about ANPR, such as government tracking citizens' movements, misidentification, high error rates, and increased government spending. Critics have described it as a form of mass surveillance.

ANPR was invented in 1976 at the Police Scientific Development Branch in Britain. Prototype systems were working by 1979, and contracts were awarded to produce industrial systems, first at EMI Electronics, and then at Computer Recognition Systems (CRS, now part of Jenoptik) in Wokingham, UK. Early trial systems were deployed on the A1 road and at the Dartford Tunnel. The first arrest through detection of a stolen car was made in 1981. However, ANPR did not become widely used until new developments in cheaper and easier to use software were pioneered during the 1990s. The collection of ANPR data for future use (i.e., in solving then-unidentified crimes) was documented in the early 2000s. The first documented case of ANPR being used to help solve a murder occurred in November 2005, in Bradford, UK, where ANPR played a vital role in locating and subsequently convicting killers of Sharon Beshenivsky.

The software aspect of the system runs on standard home computer hardware and can be linked to other applications or databases. It first uses a series of image manipulation techniques to detect, normalize and enhance the image of the number plate, and then optical character recognition (OCR) to extract the alphanumerics of the license plate. ANPR systems are generally deployed in one of two basic approaches: one allows for the entire process to be performed at the lane location in real-time, and the other transmits all the images from many lanes to a remote computer location and performs the OCR process there at some later point in time. When done at the lane site, the information captured of the plate alphanumeric, date-time, lane identification, and any other information required is completed in approximately 250 milliseconds.[citation needed] This information can easily be transmitted to a remote computer for further processing if necessary, or stored at the lane for later retrievalIn the other arrangement, there are typically large numbers of PCs used in a server farm to handle high workloads, such as those found in the London congestion charge project. Often in such systems, there is a requirement to forward images to the remote server, and this can require larger bandwidth transmission media.

ANPR uses optical character recognition (OCR) on images taken by camera. Software then processes these images. This software is comprised of certain algrithms and the complexity of these algorithms determine the accuracy of the system. Some of these algorithms include:

I. Plate localization – responsible for finding and isolating the plate on the picture

II. Plate orientation and sizing – compensates for the skew of the plate and adjusts the dimensions to the required size

III. Normalization – adjusts the brightness and contrast of the image

IV. Character segmentation – finds the individual characters on the plates

Optical character recognition

V. Syntactical/Geometrical analysis – check characters and positions against country-specific rules

ANCR systems work hand in hand with a gate inroder to provide some form of access control. In the case of this project we use a boom gate.

* 1. **PROBLEM STATEMENT**

ANCR systems have uses cases ranging criminal cases to access control. This particular project focuses more on access control. So what exactly is access control? Access control is a security process that enables organisations or individuals to manage who is authorized to access something, could be a data, a parking lot or a bulding, the list goes on and on. The ANCR system in this project tackles this issue.

Looking at the staff car park near the School of Natural Sciences it doesn’t take a genuis to see that there is no clear distiction between staff member and regular people. Anybody can enter and use it and that is not supposed to happen. Not only the staff car park but also the main university access points (gates). The lack of a proper and efficient access control systems means anyone can gain entry into the university which has several detrimental consequences to the university. That been said implementing an ANCR system we’ll grately improve not only the issures metioned above but will also;

i. Reduce the cost spent on salaries and food for men and woman who man the existing boom gates the points mentioned above.

ii. Reduce the amount of errors associated with humans manning these gates at entry points.

Besides at public instutions such as the University of Zambia in this case, ANCR systems can also be used at domestic level when implemented on an existing gate at a home inorder to provide quick access into a home not requiring an individual to manually open a gate when one of the household members arrives. No matter what type of place or establishment you have and wish to protect against unauthorized access an ANCR paired with a gate is the way to go

At its core this ANCR system solves the access control issue. Given a specified area weather an office parking lot, airport entrance or city entrace a boom gate is meant to control the which individuals are allowed in and out of an area. This project not only touches on the aspect of access control but also aims to significantly improves said process of access control via automating several process using technology such as motion sensory and optical character recognition (OCR).

* 1. **AIM**

The project is aimed at improving access control in specified areas by developing a model and a prototype of a solar powered automatic number plate detection boom gate by using motion sensors and optical character recognition (OCR).

* 1. **OBJECTIVES**

• Research and investigate how to develop an automatic number plate recognition system for a boom gate using motion sensors and optical character recognition.

• To develop a model for the automatic number plate recognition system for a boom gate.

• Build a prototye for the said model.

* 1. **RESEARCH QUESTIONS**

The questions below serve as a guideline on which this study is based.

I. How can we develop a model for an automatic number plate recognition system for boom gates using motion sensors and optical character recognition technology?

II. How can we build a prototype based on the model above for the number plate recongnition access control system?

* 1. **SCOPE OF THE PROJECT**

Developing an Automatic number plate recognition system for area access control will involve number plate recogniton and motion sensory technology. This will involves indetifying characters from a number a plate using a camera and detecting weather a car is close enough to a gate using a motion sensor.

In order to develop this system besides the camera and motion sensor metioned above we also need a Raspiberry Pi microcontroller which will serve as the brain for the whole system, a car wiper motor to control the gate, the actual gate insteal which could be some sort of foam material, cardboard or wood, A solar panel, Charge control and a battery.

* 1. **JUSTIFICATION**

Various institutions from banks, universities, shopping malls, government buildings, etc have parking lots. Many of them have some form of parking management which in most cases is manual and archaic involving someone having to physically open parking gates, use of exercise book registers to record time stamps and signatures of guests, labelling parking slots for certain individuals just to metion a few. With an automatic number plate recognition system all these issues are tackled at once and bundled into a single easy to use package. This system can also be used for private and not only public use. In private use it can completely remove the need for physically opening a house gate.

* 1. **OUTLINE OF PROJECT**

The outline of this project is as follows:

• Chapter 1 gives a full introduction to this report, highlighting its background, the problem statement and the purpose of the study.

• Chapter 2 give a literature review and shows some works that are related to this study.

• Chapter 3 elaborates the different methods that are used in order to archive the aim and objectives of this study.

• Chapter 4 shows the results that were archived by the methods implemented in this study, highlighting what the outcomes of each task were.

• Chapter 5 gives a summary of how the project went and gives a conclusion that states the challenges faced, possible areas of improvement. It also gives a general overall discussion of the project.

• Chapter 6 shows a list of references that were made in order to archive the objectives of this study.

• Chapter 7 gives the appendix of this report.

* 1. **CHAPTER SUMMARY**

The basis of this project is to implement a system that will improve parking lot access control management using automatic number plate recognition and cloud technologies.

1. **LITERATURE REVIEW**
   1. **INTRODUCTION**

Automatic number plate recognition systems (ANPR) provide a means to overcome the drawbacks and deficiency of successful surveillance of the cctv cameras. The ANPR system is well developed in certain countries such as USA and Dubai, and existed from a long time, but only in the late 90s it became an important application because of the large increase in the number of vehicles. The information extracted from the license plates is mainly used for traffic monitoring, access control, parking, motorway road tolling, and border control, making car logs for parking systems, journey time measurement for toll booth etc. by the law enforcement agencies. The recognition problem is generally sub-divided into 5 parts: (1) image acquisition i.e. capturing the image of the license plate (2) pre-processing the image i.e. normalization, adjusting the brightness, skewness and contrast of the image (3) localizing the license plate (4) character segmentation i.e. locating and identifying the individual symbol images on the plate, (5) optical character recognition. There may be further refinements over these (like matching the vehicle license number with a particular database to track suspected vehicles etc.) but the basic structure remains the same. A guiding parameter in this regard is country-specific traffic norms and standards. This helps to fine tune the system i.e. number of characters in the license plate, text luminance level (relative index i.e. dark text on light background or light text on dark background) etc. So the problem can then be narrowed down for application in a particular country. For example the norm in Zambia is printing the liscense plate numbers in black color on a white background. The general format for liscense plate is three letters followed by a four digit number.

The main focus in this research project is to experiment deeply with, and find alternative solutions to the image segmentation and character recognition problems within the License Plate Recognition framework. Three main stages are identified in such applications. First, it is necessary to locate and extract the license plate region from a larger scene image. Second, having a license plate region to work with, the alphanumeric characters in the plate need to be extracted from the background. Third, deliver them to an OCR system for recognition. In order to identify a vehicle by reading its license plate successfully, it is obviously necessary to locate the plate in the scene image provided by some acquisition system (e.g. video or still camera). Locating the region of interest helps in dramatically reducing both the computational expense and algorithm complexity. For example, a currently common 1024x768 resolution image contains a total of 786,432 pixels, while the region of interest (in this case a license plate) may account for only 10% of the image area. Also, the input to the following segmentation and recognition stages is simplified, resulting in easier algorithm design and shorter computation times. The paper mainly work with the standard Egyptian license plates but the techniques, algorithms and parameters that is be used can be adjusted easily for any similar number plates even with other alpha-numeric set.

* 1. **STEPS**

Number plate is a pattern with very high variations of contrast. If the number plate is very similar to background it’s difficult to identify the location. Brightness and contrast is changes as light fall changes to it. The morphological operations are used to extract the contrast feature within the plate. The work is divided into several parts. The basic four stage algorithm for ALPR system is:

* + 1. **Image acquisition and preprocessing:**

The system of automatic number plate recognition faces many challenges. So, this step is essential to enhance the input image and making it more suitable for the next processing steps. The first step done in the preprocessing is to apply minimum filter to the image in order to enhance the dark values in the image by increasing their area. This is mainly done to make the characters and the plate edges bold, and to remove the effect of the light diagonal strips that appear in the characters and edges of the Zambian license plates . This process is followed by increasing saturation of the image to increase the separation between colors. Then the image is converted to grayscale (taking the luminance component of NTSC). Then increasing the image contrast to separate the background from highlights

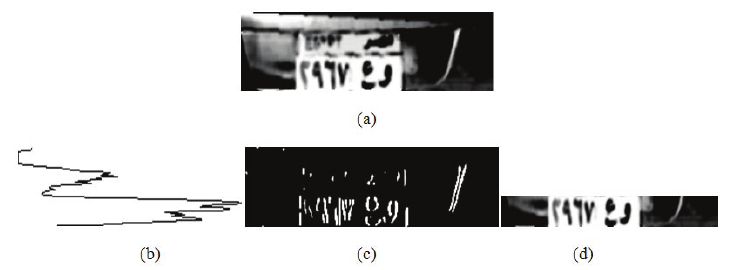


*Figure 1: Image aquisition and preprocessing*

* + 1. **License plate localization:**

In this stage, the location of the license plate is identified and the output of this stage will be a sub-image that contains only the license plate. This is done in two main steps: Locating a large bounding rectangle over the license plate. In this step a rectangle that contains the license plate is located (this rectangle may also has some extra parts from the four sides), and this rectangle is the input to the next step for further processing (removing the extra parts, character segmentation then recognition). First, Sobel vertical edge detection is applied to the image. Then a threshold of 36 (This value is determined using trial and error) is applied, Such that every edge with magnitude less than 36 is considered false edge and is set to 0. Then a vertical projection (projecting on the Y-axis) of the edge detected image is taken and smoothed using an average filter with width equals 9. It’s obvious that the characters of the plate along with the plate’s vertical edges will have very strong vertical edges. Moreover, these edges will sum up horizontally in the vertical projection and a strong peak will appear in the rows of the plate (These row will be called band). So, the approach is to take some number of peaks in the vertical projection and processing each of them individually in the next steps and when a successful band is found, the processing of the following bands is canceled. The reason behind taking more than one peak is that the image may contain objects (logos, road advertisement, etc..) that produce many vertical edges also these ”false” edges may be centered in the same area so they will form a peak that may be stronger than the peak of the plate itself. For each band, we take a sub-image referenced by this band and all subsequent processing will be applied on this sub-image. Now the problem is to cut the band image from the left and right to get a bounding rectangle over the license plate (Again, this rectangle doesn’t have to be tight on the plate). For this sake, a vertical Sobel edge detection is applied again, but the height is larger than the width of the filter, this is to decrease the effect of false edges and noise, experimentally, the best size is 6x3 filter. Again a threshold of 30 is applied for the same reason as before. Now, a horizontal projection of the edge detected band image is taken (projection on the X-axis) and smoothed using an average filter of large size this time, since There are gaps between the letters and the projection will have many peaks at the x coordinates where letters exist but it will drop down in the x coordinates of the gaps. So, smoothing it with average filter of large width will resolve this problem and many number of peaks will be converted to one wide peak that represents the range of the X-axis where the plate is located in that specific band we are working with. The width of the average filter is taken to be the height of the band. Relating the height of the band with the width of the average filter is very important since over-smoothing of the projection will merge the plate peak with the other main peaks in the band like the peak got from vehicle lamps for example (and it already explained why the width shouldn’t be very small). Now, a predefined number of peaks (It’s already explained why we take more than one candidate peak not just the strongest one) will be selected from the smoothed projection. For each peak, a sub-image is taken according to the range of current peak. So,the bounding rectangle of the license plate is located. This is will be the input to the next step.

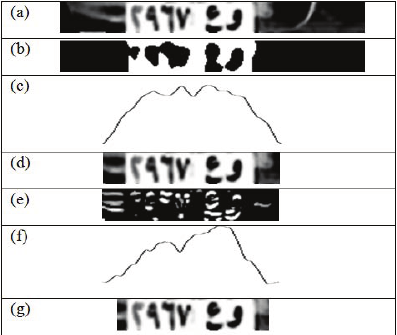
Determining the exact location of the license plate. Using the sub-image from the last step which contains the license plate with some extra parts (if any), the following processing is applied to this sub-image. The license plate may be skewed because of the angle of the camera while image acquisition process. And it is very important to de-skew the plate to its original orientation, thus making the plate Aligned with the X and Y axes (The reason behind its importance will be clear below). So a Hough transform is applied to the horizontally edge detected image in order to find the shear parameters by which the image can be de-skewed to retrieve the standard orientation. After this operation we have a plate with its axes aligned with the X and Y axes. Then a Gaussian smoothing filter is applied to smooth the image and remove noise. Then a morphological operation consists of subtracting the bottom-hat of the image from the image itself is done using a structuring element of a horizontal line of length 150. This operation makes the characters of the plate bold and increases the characters area along with the effect of increasing contrast, and subsequently this will ease the process of segmentation and recognition afterward. All the above is considered a preprocessing for this step. Next, we aim at finding the exact band of the plate. In other words, the goal of this step is to cut the top and bottom extra parts of the previously cut rectangle (but this time the cut will be accurate because we have limited the area we are working with and moreover we de-skewed the plate). This is done using the same idea we used previously to get the plate band. It consists of applying Sobel vertical edge detection, then applying a threshold, then doing a vertical projection (projecting on the Y-axis), Then getting the strongest peak in this projection and cut the image accordingly using the range of this peak, thus cutting the exact plate band from the image and leaving the top and bottom extra parts. This time just the strongest peak is taken since we already limited the possibility that false edges appear when we cut a rectangle around the plate and we are sure that the vertical edges produced by the plate’s characters are summed up correctly in a limited number of rows due to the de-skew operation. We got rid of the top and bottom extra parts. But we still have extra parts from left and right that have to be cut to end up with an exact rectangle around the plate. So, a stamp filter is applied to the sub-image we got from the previous step. This filter is just a blurring followed by a soft threshold operation. Now the white color will dominate the plate area. After this a horizontal projection is done then smoothed using average filter with width equals 40. Then we get the strongest peak from this projection. This peak corresponds to the plate range on the X-axis. So, a sub-image is cut using the peak range. In many cases when the color of the vehicle is bright, the previous operation is not sufficient to cut all the extra pieces from left and right. So, this is followed by getting Sobel horizontal edge detection, applying a threshold, then getting the horizontal projection, then smoothing this projection with average filter of size 40. Then we will get two points that will define range of the peak. The first point is the point with least x coordinate that has a value (from the smoothed projection) greater than or equal the average value. The second point is the point with maximum x coordinate that has a value greater than or equal to the average. We will cut the image again using these two points we got. And this is the final plate that the next processing stages willwork on. At the current moment we have a”candidate” final plate. The next processing stages are computationally expensive. Also using the fact that all the plates have a very similar (if not exact) values for some measures like aspect ratio, contrast, average brightness, average saturation in both the colored and grayscale plate images. We can begin to reject the plates based on the previous measures, such that, If we found that the current candidate plate for any measure has a very far value from the ranges of values for the true plates, It’s simply rejected and the processing continues on the next candidate plate. But a false plate may pass these tests, and it will be rejected in subsequent stages. The next stage is to segment characters from the plate that passed all the measures tests.



*Figure 2: Liscense plate localization*

* + 1. **Segmentation:**

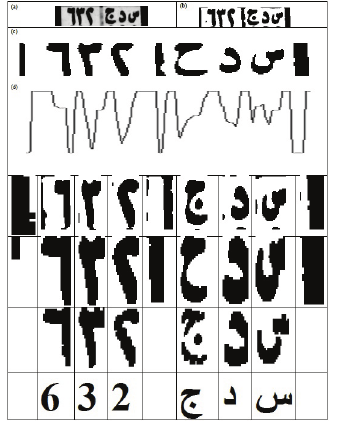
This stage is meant for segmentation of the characters from the plate. The output of this stage is a set of monochrome images for Each candidate character in plate. The first step in this stage is to convert the plate image to a binary image. This is done using adaptive threshold with a window of size 11 (This is selected using trial and error). Then a process of noise removal is applied. This is done by getting the connected components from the binary image based on the 8-neighbourhood using flood fill. For every component, we decide if it’s a noise or not based on the aspect ratio of the component and based on the number of pixels in that component. This is based on the fact that the characters of the plate have a certain range of aspect ratio and a certain range of number of pixels. After removing the noise components a maximum filter is applied to make the effect of thinning the characters to make sure that no two components are merged. This is followed by a horizontal projection, to detect the boundaries between the characters to be able to cut them individually. The peaks in this projection correspond to the gaps between the characters. So, we get all of these peaks and a rejection process is applied also, since a true plate has a fixed range of gaps between characters. So, any plate that has number of peaks that do not fit in that range, will be rejected. Also, there is a powerful rejection measure; it is the variance of the characters width (the variance of the spaces between peaks). After this the characters are cut according to the peaks of the previous projection. Then another set of measures are computed to reject the false characters that may still exist after the noise removal operation. These measures are aspect ratio, deviation from average height test, deviation from average contrast, deviation from average brightness, deviation from hue, deviation from average saturation. After rejecting false characters, if the number of characters is not located in a predefined range, then the plate is rejected. Otherwise, the processing is continued And for every character a copy of its corresponding location in the grayscale is got. The gray level histogram is computed for the sub-image of each character, This gray level histogram will have a standard shape which is one peak at the dark values (this corresponds to the character’s pixels) and another peak at the bright values (this corresponds to the background) and some small values between them. So, this gray level image is converted to binary using the following procedure. First, we find two peaks in the histogram then we find the minimum value in between, this will be the value of the threshold (thus, every pixel that has a gray level value less than the mentioned value, will be converted to black, every other value will be converted to white). This way for converting the grayscale image that contains only a character to binary one proved to be effective. At this point we have a set of binary images each contains one character and this is the output of this stage and the input to the next.



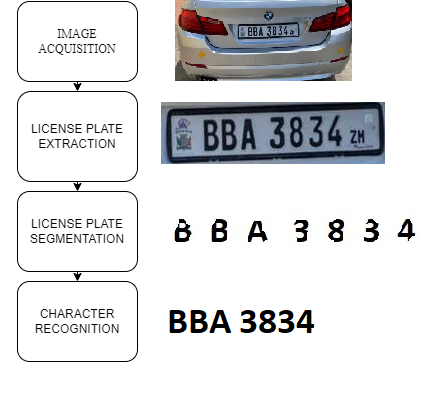
*Figure 3: Character Segmentation*

* + 1. **Character recognition:**

The goal of this stage is to recognize and classify the binary images that contain characters received from the previous one. After this stage every character must have a label and an error factor, and this error factor if greater than a predefined value will be used to reject false characters accidently passed from the previous steps. For the sake of classification, some features must be collected from the characters. The feature we work with in this system is the chain code of the contour of the image after dividing it into four tracks then into four sectors. Also we used a feed forward artificial neural network trained with back propagation with sigmoid activation function and the ANN is trained on the chain code feature of the optimal characters images. The neural network has 4X4X8=128 input neuron, it also has 37 output neurons corresponds to the Arabic alpha-numeric set of characters except zero, it also ceil (37+128)/2)=83 hidden neurons. So, for every character we get the chain code feature and do a feed forward on the trained FFNN (Feed Forward Neural Network) then the class the corresponds to the neuron with the maximum value will the predicted class of that character. If the error exceeds a predefined value then the character is considered a false one and rejected. The plate is known to have a fixed range of characters that may appear in it, so if the total number of passed characters does not match this range, then the plate is rejected. Otherwise, the license plate number is found.



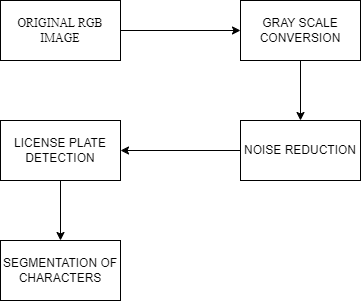
*Figure 4: Character recognition*



*Figure 5: ALPR 4 step algorithm*

In the basic four stage algorithm, first of all the image containing the license plate is acquired, then the license plate is localized followed by license plate segmentation and character recognition. The aforesaid four stage algorithm could be elaborated to improve the efficiency of the system as per the requirement, to elaborate this basic algorithm here we present the proposed system in detail which is as follows:

* Input raw image
* Grey scale conversion
* Median filtering or noise reduction
* License plate localization
* Segmentation of the characters

\

*Figure 6: ALPR 4 step algorithm flow diagram*

* 1. **OPERATIONS**

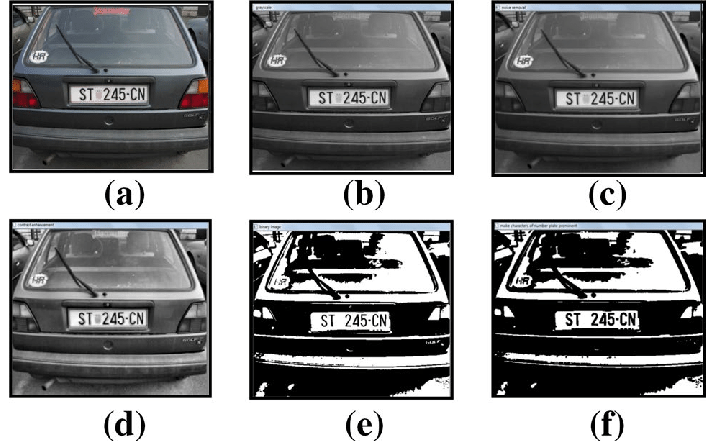
A. **Input Raw Image**

B. **Gray Scale Conversion:**

From the input RGB image it has to be convert to gray scale and the 8it gray value is calculated. After grey scale conversion the morphological operations begin which starts with dilation of an image, in the diagram we show the process of dilation. After the dilation of image we erode the image and apply median filtering technique which is the basis of noise reduction

C. **Noise Reduction:**

We used median filtering technique to reduce the paper and salt noise. We have used 3x 3 masks to get eight neighbors of a pixel and their corresponding gray value. After applying the median filtering technique we erode the image by a structural element in the form of any shape. We go on to find the morphological gradient for edge enhancement, then we convert the class to double for brightening the edges and perform the convolution of the double image. The figure below shows the convoluted image.



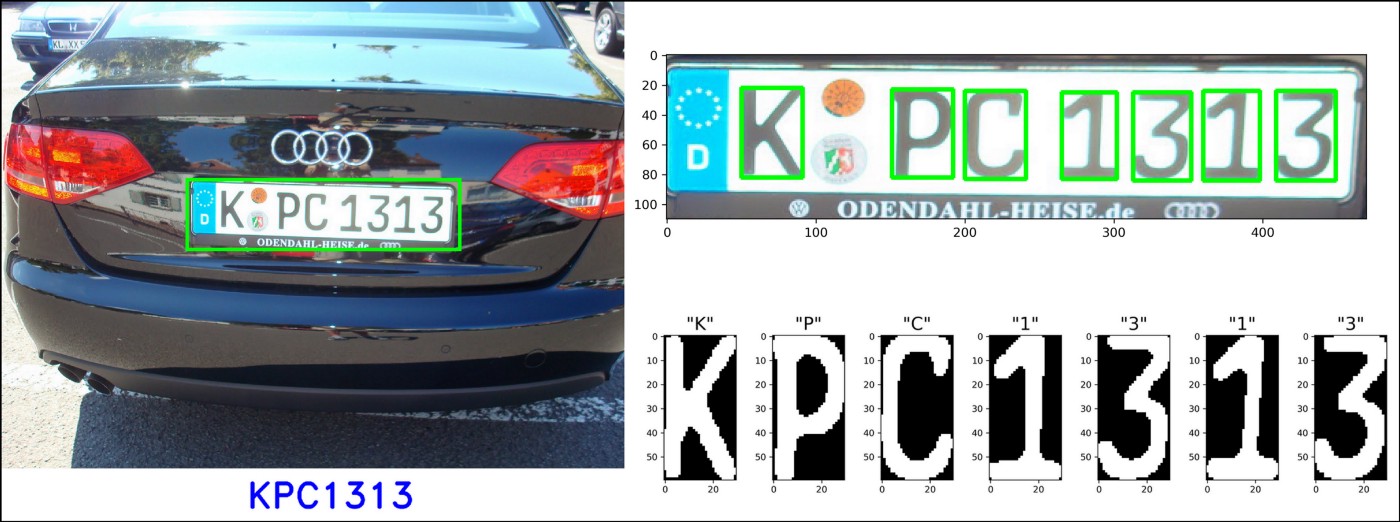
*Figure 7: Noise reduction*

D. **Contrast enhancement using histogram equalization:**

Using histogram equalization technique the contrast of each image is being enhanced.

E. **Plate Localization:**

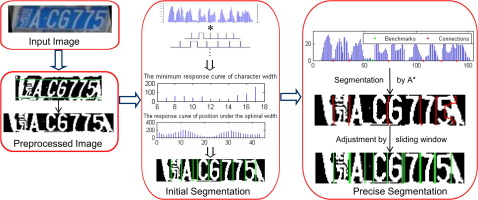
The basic step in recognition of vehicle number plate is to detect the plate size. In general number plates are rectangular in shape. Hence we have to detect the edges of the rectangular plate.Mathematical morphology will be used to detect that region. Using Sobel edgedetector we used to high lightregions with a high edge magnitude andhigh edgevariance are identified. Depending upon the threshold value edge will be detected from the input image.



*Figure 8: Plate Localisation*

F. **Character Segmentation:**

Matlab toolbox function provides a function called regionprops(). It measures a set of properties for each labeled region in the label matrix. We use boundingbox to measure the properties of the image region. After labeling the connecting components, the region will be extracting from the input image.



*Figure 9: Character Segmentation 2*

* 1. **CHAPTER SUMMARY**

In general, an ALPR system consists of four processing stages. In the image acquisition stage, some points have to be considered when choosing the ALPR system camera, such as the camera resolution and the shutter speed. In the license plate extraction stage, the license plate is extracted based onsome features such as the color, the boundary, or the existence of the characters. In the license plate segmentation stage, the characters are extracted by projecting their color information, by labeling them, or by matching their positions with template.

Finally, the characters are recognized in the character recognition stage by template matching, or by classifiers such as neural networks and fuzzy classifiers. Automatic license plate recognition is quite challenging due to the different license plate formats and the varying environmental conditions. There are numerous ALPR techniques that have been proposed in recent years. Issues, such as main processing procedure, experimental database, processing time, and recognition rate, are provided.

1. **METHODOLOGY**

This chapter introduces the methods used to implement this project with the use of research papers, views and work related to the study of necessary technology to be used in this project.

* 1. **SOFTWARE TOOLS**
     1. **Programming languages**
* **Python:**

Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation.Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.Guido van Rossum began working on Python in the late 1980s as a successor to the ABC programming language and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000 and introduced new features such as list comprehensions, cycle-detecting garbage collection, reference counting, and Unicode support. Python 3.0, released in 2008, was a major revision that is not completely backward-compatible with earlier versions. Python 2 was discontinued with version 2.7.18 in 2020.Python consistently ranks as one of the most popular programming languages. Python's large standard library provides tools suited to many tasks, and is commonly cited as one of its greatest strengths. For Internet-facing applications, many standard formats and protocols such as MIME and HTTP are supported. It includes modules for creating graphical user interfaces, connecting to relational databases, generating pseudorandom numbers, arithmetic with arbitrary-precision decimals, manipulating regular expressions, and unit testing.

In this project Python is used in for creation of the backend of the system that is the server side code and the API (Application Programming Interface) and programming of the **Raspberry Pi microcontroller**.

* **C++:**

C++ (pronounced "C plus plus") is a general-purpose programming language created by Danish computer scientist Bjarne Stroustrup as an extension of the C programming language, or "C with Classes". The language has expanded significantly over time, and modern C++ now has object-oriented, generic, and functional features in addition to facilities for low-level memory manipulation. It is almost always implemented as a compiled language, and many vendors provide C++ compilers, including the Free Software Foundation, LLVM, Microsoft, Intel, Embarcadero, Oracle, and IBM, so it is available on many platforms.

C++ was designed with systems programming and embedded, resource-constrained software and large systems in mind, with performance, efficiency, and flexibility of use as its design highlights. C++ has also been found useful in many other contexts, with key strengths being software infrastructure and resource-constrained applications, including desktop applications, video games, servers (e.g. e-commerce, web search, or databases), and performance-critical applications (e.g. telephone switches or space probes).

C++ is standardized by the International Organization for Standardization (ISO), with the latest standard version ratified and published by ISO in December 2020 as ISO/IEC 14882:2020 (informally known as C++20). The C++ programming language was initially standardized in 1998 as ISO/IEC 14882:1998, which was then amended by the C++03, C++11, C++14, and C++17 standards. The current C++20 standard supersedes these with new features and an enlarged standard library. Before the initial standardization in 1998, C++ was developed by Stroustrup at Bell Labs since 1979 as an extension of the C language; he wanted an efficient and flexible language similar to C that also provided high-level features for program organization. Since 2012, C++ has been on a three-year release schedule with C++23 as the next planned standard.

In this project c++ is used for programming the **Arduino Uno Micro-controller** and the devices associated with it which are the **HC-05 Bluetooth Module** and the **HC-SR04 Ultrasonic Sensor.**

* **JavaScript:**

JavaScript often abbreviated as JS, is a programming language that is one of the core technologies of the World Wide Web, alongside HTML and CSS. As of 2022, 98% of websites use JavaScript on the client side for webpage behaviour, often incorporating third-party libraries. All major web browsers have a dedicated JavaScript engine to execute the code on users' devices. JavaScript is a high-level, often just-in-time compiled language that conforms to the ECMAScript standard. It has dynamic typing, prototype-based object-orientation, and first-class functions. It is multi-paradigm, supporting event-driven, functional, and imperative programming styles. It has application programming interfaces (APIs) for working with text, dates, regular expressions, standard data structures, and the Document Object Model (DOM).The ECMAScript standard does not include any input/output (I/O), such as networking, storage, or graphics facilities. In practice, the web browser or other runtime system provides JavaScript APIs for I/O.

JavaScript engines were originally used only in web browsers, but are now core components of some servers and a variety of applications. The most popular runtime system for this usage is Node.js. Although Java and JavaScript are similar in name, syntax, and respective standard libraries, the two languages are distinct and differ greatly in design.

JavaScript in this project is used for development of the frontend of the system. That is everything that the end users will see which includes the admin dashboard,

the users dashboard and mobile app.

* + 1. **Frameworks and Libraries**
* **Django:**

Django is a free and open-source, Python-based web framework that follows the model–template–views (MTV) architectural pattern. It is maintained by the Django Software Foundation (DSF), an independent organization established in the US as a 501(c)(3) non-profit.

Django's primary goal is to ease the creation of complex, database-driven websites. The framework emphasizes reusability and "pluggability" of components, less code, low coupling, rapid development, and the principle of don't repeat yourself. Python is used throughout, even for settings, files, and data models. Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models.

Some well-known sites that use Django include Instagram, Mozilla, Disqus, Bitbucket, Nextdoor and Clubhouse

Django web-framework in this project is used to create the backend that is the server side code and together with a package used in Django called Django Rest Framework is used to create an API(Application Programming Interface). Django comes shipped with several database packages such as MySQL and Posgresql build-in, that been said it is only required for you to define database models and Django will do all the heavy lifting of creating the database. A MySQL database is used in this project.

* **React:**

React is a free and open-source front-end JavaScript library for building user interfaces based on UI components. It is maintained by Meta and a community of individual developers and companies. React can be used as a base in the development of single-page, mobile, or server-rendered applications with frameworks like Next.js. However, React is only concerned with state management and rendering that state to the DOM, so creating React applications usually requires the use of additional libraries for routing, as well as certain client-side functionality.

In this project React is used for the development of the frontend of the system which includes the admin dashboard and user dashboard. React Native which is a library close to react will be used for the development of the mobile app.

* **OpenCV:**

OpenCV (Open Source Computer Vision Library) is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itseez (which was later acquired by Intel. The library is cross-platform and free for use under the open-source Apache 2 License. Starting with 2011, OpenCV features GPU acceleration for real-time operations.

Used frequently in Facial recognition in this project OpenCV will be used to mainly detect and recognize where and what exactly the number is.

* **Tesseract:**

Tesseract is an optical character recognition engine for various operating systems. It is free software, released under the Apache License. Originally developed by Hewlett-Packard as proprietary software in the 1980s, it was released as open source in 2005 and development has been sponsored by Google since 2006.In 2006, Tesseract was considered one of the most accurate open-source OCR engines available

With regards to this project Tesseract is used to read the characters obtained from the identified number plate.

* + 1. **IDE’s and Text Editors**
* **Visual Studio Code:**

Visual Studio Code, also commonly referred to as VS Code, is a source-code editor made by Microsoft with the Electron Framework, for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. Users can change the theme, keyboard shortcuts, preferences, and install extensions that add additional functionality.

In the Stack Overflow 2021 Developer Survey, Visual Studio Code was ranked the most popular developer environment tool among 82,000 respondents, with 70% reporting that they use it

Visual studio in this project is as a text editor in creation of the backend(Server side, database and API) and frontend(admin dashboard, user dashboard and mobile app).

* **Arduino IDE:**

The Arduino integrated development environment (IDE) is a cross-platform application (for Microsoft Windows, macOS, and Linux) that is written in the Java programming language. It originated from the IDE for the languages Processing and Wiring. It includes a code editor with features such as text cutting and pasting, searching and replacing text, automatic indenting, brace matching, and syntax highlighting, and provides simple one-click mechanisms to compile and upload programs to an Arduino board. It also contains a message area, a text console, a toolbar with buttons for common functions and a hierarchy of operation menus. The source code for the IDE is released under the GNU General Public License, version 2.

The Arduino IDE supports the languages C and C++ using special rules of code structuring. The Arduino IDE supplies a software library from the Wiring project, which provides many common input and output procedures. User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub main() into an executable cyclic executive program with the GNU toolchain, also included with the IDE distribution. The Arduino IDE employs the program avrdude to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware.

From version 1.8.12, Arduino IDE windows compiler supports only Windows 7 or newer OS. On Windows Vista or older one gets "Unrecognized Win32 application" error when trying to verify/upload program. To run IDE on older machines, users can either use version 1.8.11, or copy "arduino-builder" executable from version 11 to their current install folder as it's independent from IDE

In this project the Arduino IDE is used for the programming and the parts that work with it.

* **Thonny IDE:**

Thonny is an integrated development environment for Python that is designed for beginners. It was created by Aivar Annamaa, an Estonian programmer. It supports different ways of stepping through the code, step-by-step expression evaluation, detailed visualization of the call stack and a mode for explaining the concepts of references and heap.

In this project Thonny IDE is used in programming the raspberry pi microcontroller and its associated parts such as the Pi Camera.

* 1. **HARDWARE TOOLS**
     1. **MICROCONTROLLERS**

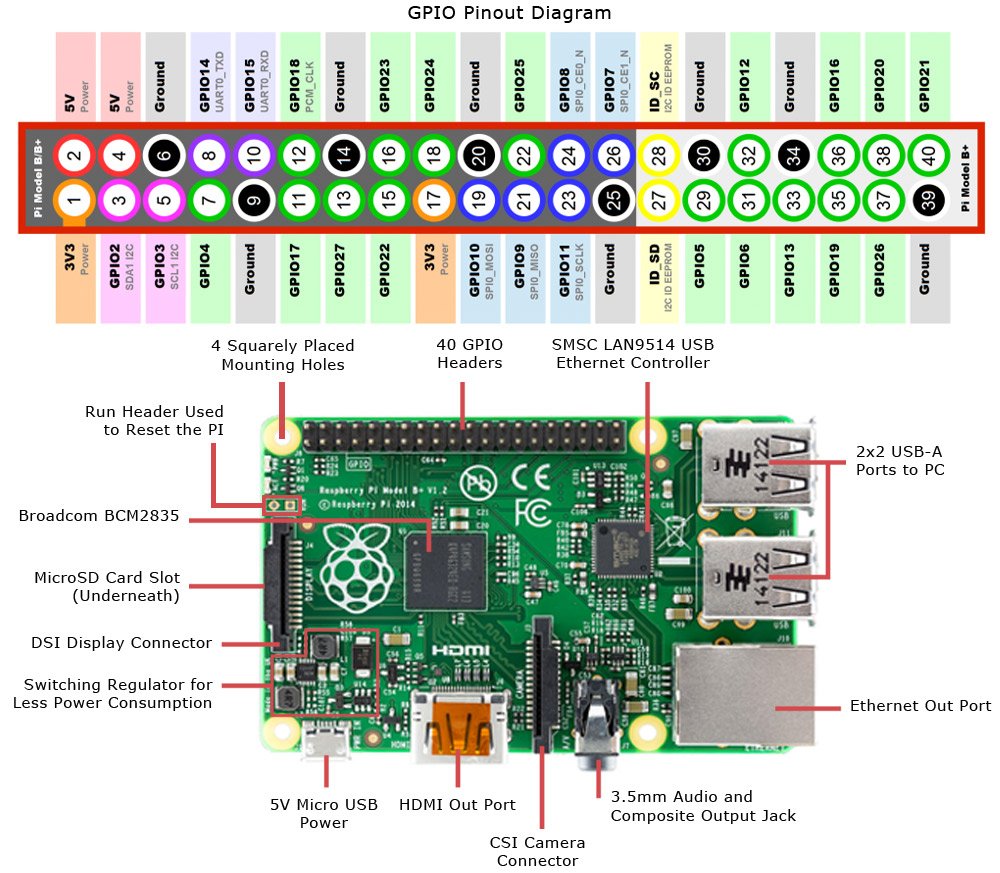
A microcontroller (MCU for microcontroller unit) is a small computer on a single VLSI integrated circuit (IC) chip. A microcontroller contains one or more CPUs (processor cores) along with memory and programmable input/output peripherals. Program memory in the form of ferroelectric RAM, NOR flash or OTP ROM is also often included on chip, as well as a small amount of RAM. Microcontrollers are designed for embedded applications, in contrast to the microprocessors used in personal computers or other general purpose applications consisting of various discrete chips.

The microcontrollers used in this project are the Arduino Uno and the Raspberry Pi 3.

* **Raspberry Pi:**

Raspberry Pi is a series of small single-board computers (SBCs) developed in the United Kingdom by the Raspberry Pi Foundation in association with Broadcom. The Raspberry Pi project originally leaned towards the promotion of teaching basic computer science in schools and in developing countries. The original model became more popular than anticipated, selling outside its target market for uses such as robotics. It is widely used in many areas, such as for weather monitoring,because of its low cost, modularity, and open design. It is typically used by computer and electronic hobbyists, due to its adoption of the HDMI and USB standards.

In this project the Raspberry works as the brain of the system it brings together all the components of the system inorder to perform the intended purpose of the system. It receives signals from the Arduino Uno, Controls the operations of the motor and sends and recieves data to and from the cloud thus informing the web apps and mobile apps.

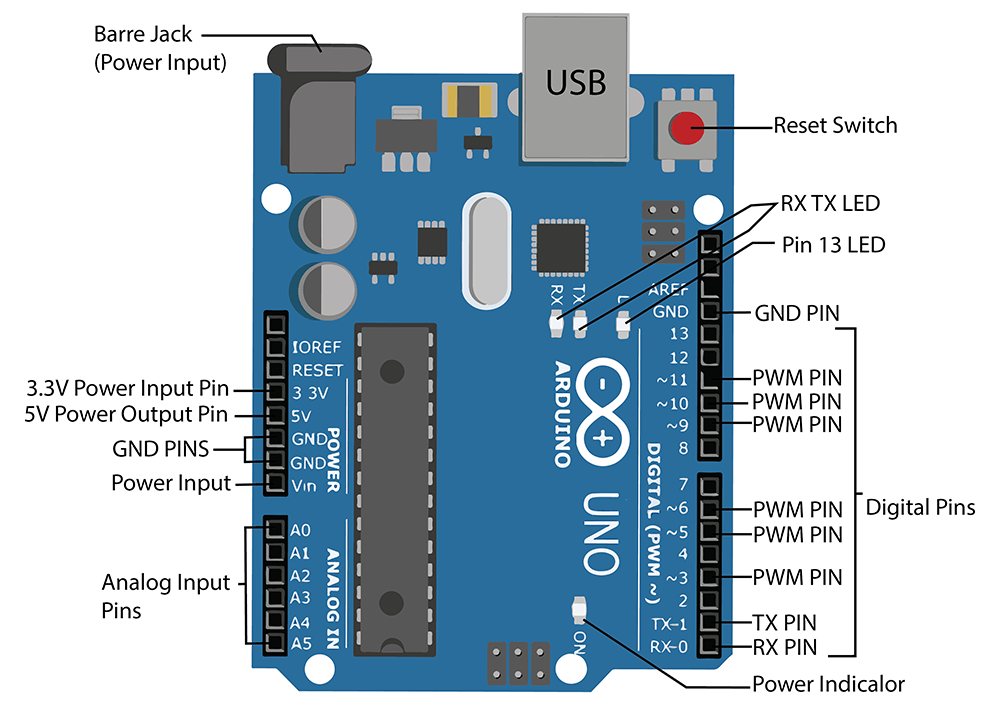


*Figure 10: Raspberry Pi Model 3*

* **Arduino Uno:**

The Arduino Uno is an open-source microcontroller board based on the Microchip ATmega328P microcontroller and developed by Arduino.cc and initially released in 2010. The board is equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (shields) and other circuits. The board has 14 digital I/O pins (six capable of PWM output), 6 analog I/O pins, and is programmable with the Arduino IDE (Integrated Development Environment), via a type B USB cable. It can be powered by the USB cable or by an external 9-volt battery, though it accepts voltages between 7 and 20 volts. It is similar to the Arduino Nano and Leonardo. The hardware reference design is distributed under a Creative Commons Attribution Share-Alike 2.5 license and is available on the Arduino website. Layout and production files for some versions of the hardware are also available

In this project the Arduino Uno is used for mainly for determing whether a vehicle is at the gate or not and then notifying the raspberry pi of this. It determines the presence of a vehicle via the use of an HC-SR04 UltraSonic Sensor and then notifies the Raspberry Pi of this by using the HC-05 Bluetooth Module.



*Figure 11: Arduino Uno Microcontroller*

* + 1. **PI CAMERA**

This 5 megapixels sensor with OV5647 camera module is capable of 1080p video and still images that connect directly to your Raspberry Pi. This is the plug-and-play-compatible latest version of the Raspbian operating system, making it perfect fortime-lapse photography, recording video, motion detection and security applications. Connect the included ribbon cable to the CSI (Camera Serial Interface) port on your Raspberry Pi, and you are good to go!

The board itself is tiny, at around 25mm x 23mm x 9mm and weighing in at just over 3g, making it perfect for mobile or other applications where size and weight are important. The sensor has a native resolution of 5 megapixel, and has a fixed focus lens on board. In terms of still images, the camera is capable of 2592 x 1944 pixel static images, and also supports 1080p30, 720p60 and 640x480p90 video.

In the project the Pi Camera is used to take a picture of the vehicle once its within close range, it receives a signal from the Raspberry Pi which inturn receives a signal from the Arduino Uno.



*Figure 12: Pi Camera*

* + 1. **RELAYS**

The relay module is an electrically operated switch that can be turned on or off deciding to let current flow through or not. They are designed to be controlled with low voltages like 3.3V like the ESP32, ESP8266, etc, or 5V like your Arduino**.**

In this project the Raspberry Pi uses the relays to control when the motor receives power from the battery. This project has 2 relays in order to create an H-bridge which is needed to rotate the motor in the anti-clockwise direction.

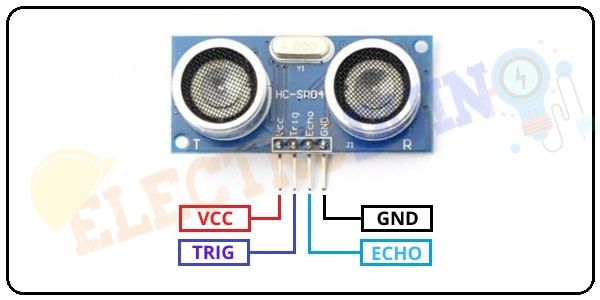


*Figure 13: Relay Moodule*

* + 1. **HC-SR04 ULTRASONIC SENSOR**

An HC-SR04 ultrasonic distance sensor actually consists of two ultrasonic transducers. One acts as a transmitter that converts the electrical signal into 40 KHz ultrasonic sound pulses. The other acts as a receiver and listens for the transmitted pulses. When the receiver receives these pulses, it produces an output pulse whose width is proportional to the distance of the object in front. This sensor provides excellent non-contact range detection between 2 cm to 400 cm (~13 feet) with an accuracy of 3 mm. Since it operates on 5 volts, it can be connected directly to an Arduino or any other 5V logic microcontroller.

In this project the ultrasonic detects if a vehicle is within range.

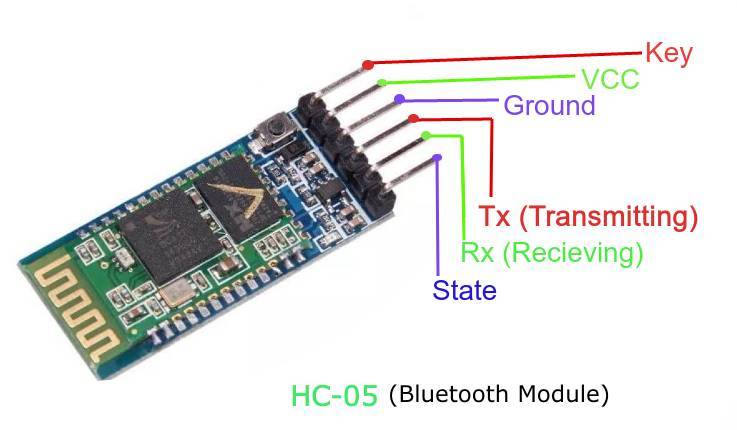


*Figure 14: HC-SR04 UltraSonic Sensor*

* + 1. **HC-05 BLUETOOTH MODULE**

HC-05 Bluetooth Module is an easy to use Bluetooth SPP (Serial Port Protocol) module, designed for transparent wireless serial connection setup. Its communication is via serial communication which makes an easy way to interface with controller or PC. HC-05 Bluetooth module provides switching mode between master and slave mode which means it able to use neither receiving nor transmitting data.

When the UltraSonic sensor detects a car in range the Bluetooth Module sends a signal to the Raspberry Pi which the triggers the process that determines whether the car is registered or not.



*Figure 15: HC-05 Bluetooth module*

* + 1. **CAR WIPER MOTOR**

Simply a motor used in cars to control wiper movement. This particular motor was selected because it contains gears which can be adjustable to control the speed at which the motor moves. The motor used is the project is 12V motor.



*Figure 16: Car wiper motor*

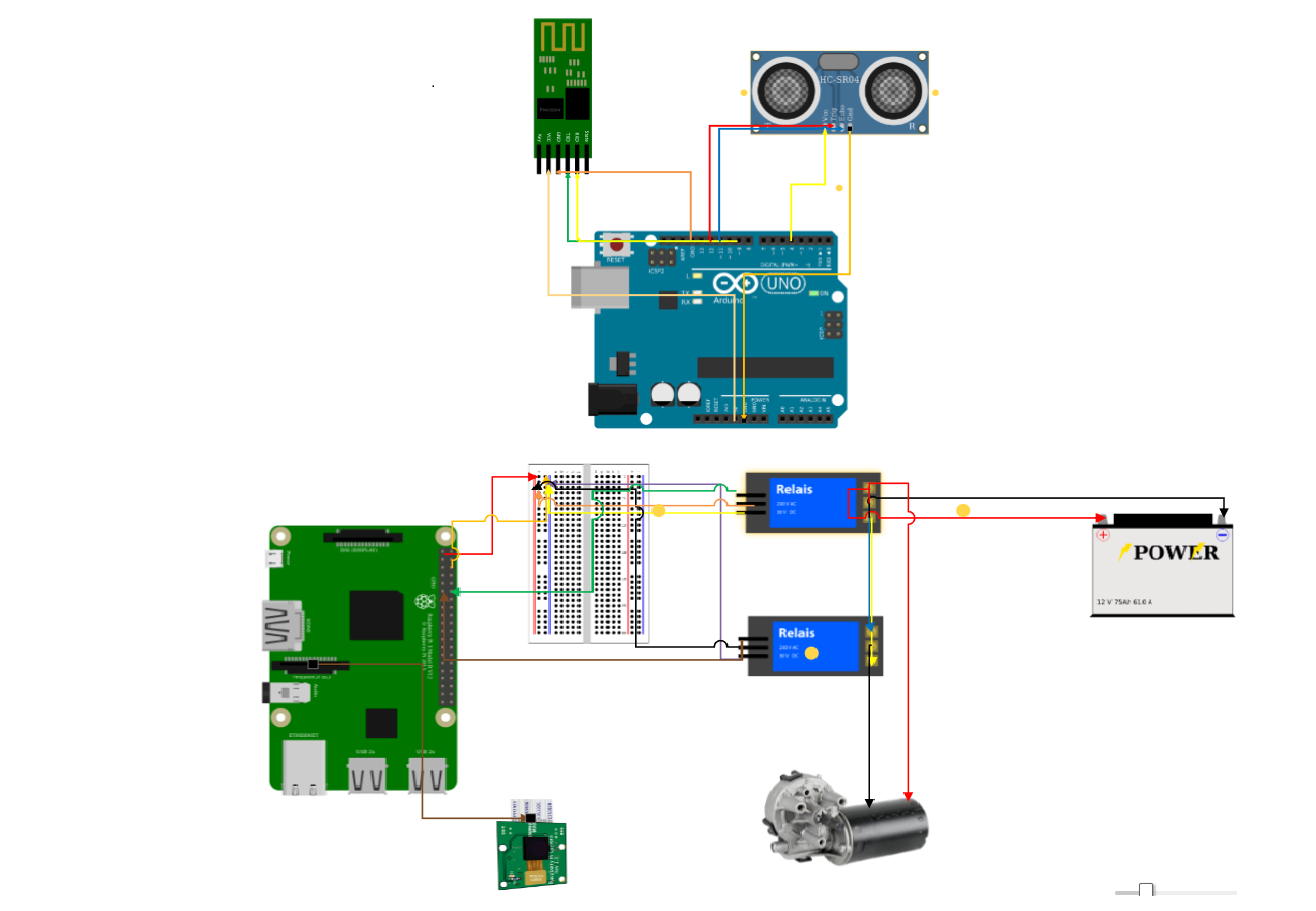
* + 1. **OTHERS HARDWARE COMPONENTS USED**
* Solar panel
* Solar charge controller
* 12V battery
* Personal computer
* Connecting wires
* Power bank
  1. **SYSTEM IMPLEMENTATION DIGRAMS**
     1. **NUMBER PLATE CHARACTER RECONGNITION FLOW DIGRAM**

****

*Figure 17: Number plate characcter recognition flow diagram*

The diagram above shows the necessary steps taken in order to identify and extract characters from a number plater given only an image of the vehicle. Each step is extensively expalined in the literature review chapter of this document.

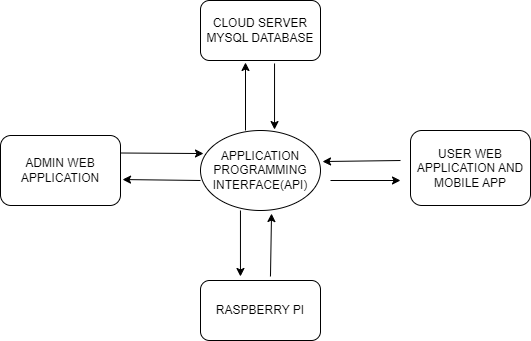
* + 1. **MICROCONTROLLER CIRCUIT DIAGRAM**



*Figure 18: Microcontroller circuit diagram*

A circuit diagram like the one above is crucial in order to understand the operations of the system. It shows how each component is connected and how they work together inorder to reach the goal of powering the motor at the appropriate time inorder to open the gate. In the diagram above the **Arduino Uno** is connected to the **HC-05 Bluetooth Module** and the **HC-SR04 Ultra-Sonic Sensor**. The **Raspberry Pi** is connected to the **Pi Camera** and is connected to the **Relay Modules** via the **Breadboard** inorder to draw power from the **Battery** when needed to power the **Wiper Motor** thus turning the gate.

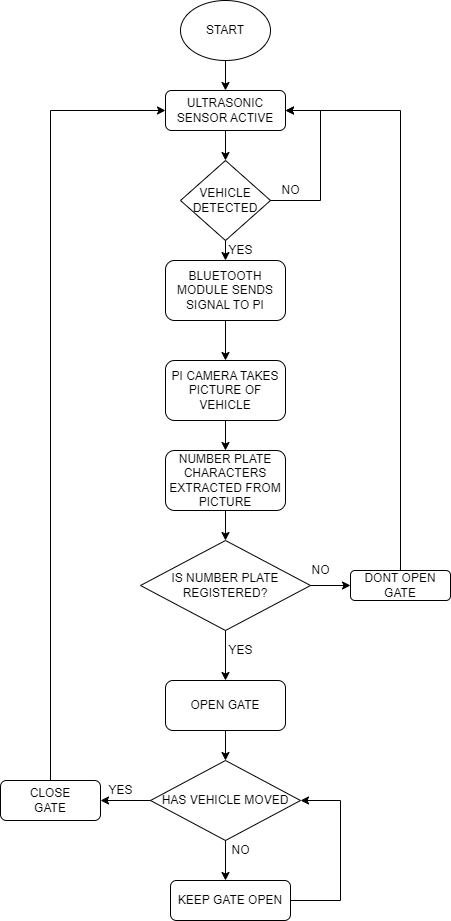
* + 1. **SOFTWARE FLOW DIAGRAM**



*Figure 19: System software flow diagram*

Just like the circuit diagram, a software flow diagram like the one above is crucial is in understanding how the system works. It displays how the different parts of the software section of the system work together. In the system the **Application Programming Interface (API)** serves as a middle sending data to and from each component.

* + 1. **SYSTEM FLOW DIAGRAM**



*Figure 20: System flow diagram*

The system flow diagram above demonstrates how the entire system and its components both hardware and software work and interact to achieve the goal of opening and closing the gate when necessary. When the vehicle approaches the gate the Ultrasonic sensor detects it and a signal is sent to the Raspberry pi via the Bluetooth module. When the Raspberry pi receives this signal it informs the camera to take a picture of the vehicle. Once the image is taken it is then processed and using Machine learning and OCR(Optical Character Recognition) the number plate in the picture is identified and the characters are extracted. The Raspberry pi then checks the database server via the API(Application Programming Interface) to see if the extracted characters are registered or not. If they are then the gate is opened and if they arent it remains closed. Throught this process the Ultrasonic sensor keeps detecting the presence of the vehicle, once the vehicle moves it sends a signal to the Raspberry pi via the Bluetooth module which inturn directs the motor to close the gate.

* 1. **CHAPTER SUMMARY**

This chapter introduced all the different steps and methods used to archive the desired results of the project. It lists all the necessary requirements such as the hardware and the software as well as software packages. With the use of diagrams, it shows how the system operates and also how it is setup and connected.

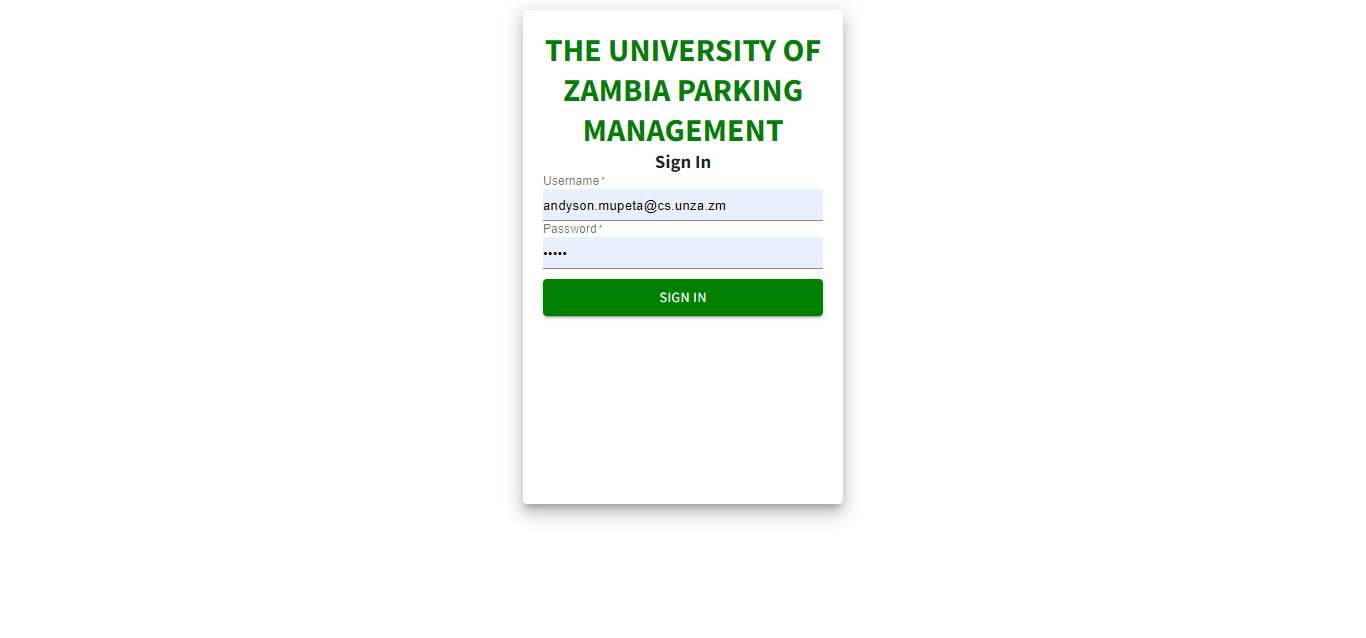
1. **RESULTS**

This chapter examines the different end results of the project ranging from hardware to software. The software includes a web app and a mobile app, the artificial intelligent software running in the raspberry pi and the code giving the arduino functionality. The hardware part includes the various electrical components involved and how they connect to each other and also includes the physical structure of the boom gate which houses all the components.

**SOFTWARE RESULTS**

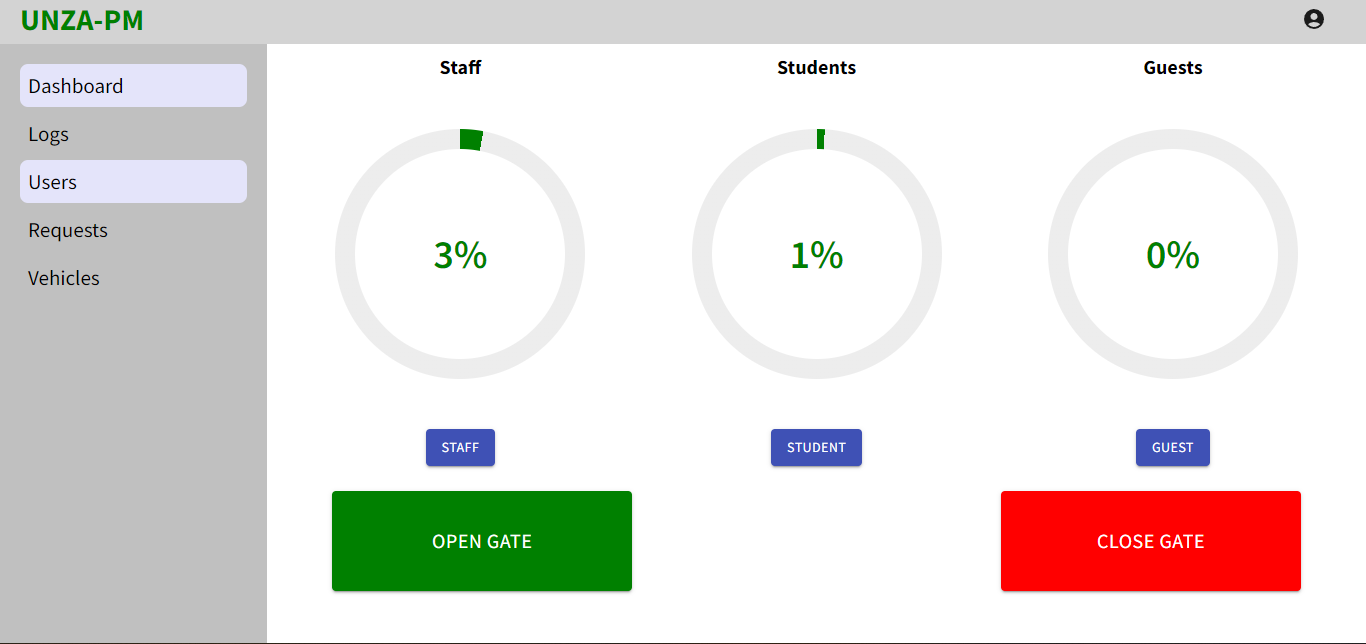
* 1. **ADMIN WEB APP**

The admin web app is comprised of a dashboard and other components that enable the system administrator to effectively manage the system. It features parking lot capacity indicators to show how much space is left in a specified parking lot, it has system controls such as choosing which parking lot the gate should be used and also has controls that can open and close the gate instantly.It features a list of users and their vehicle details, the logs of when and who used the gate, requests of users who want to access the system and the ability to accept and reject these requests. This admin portal can only be accessed by an administrator.



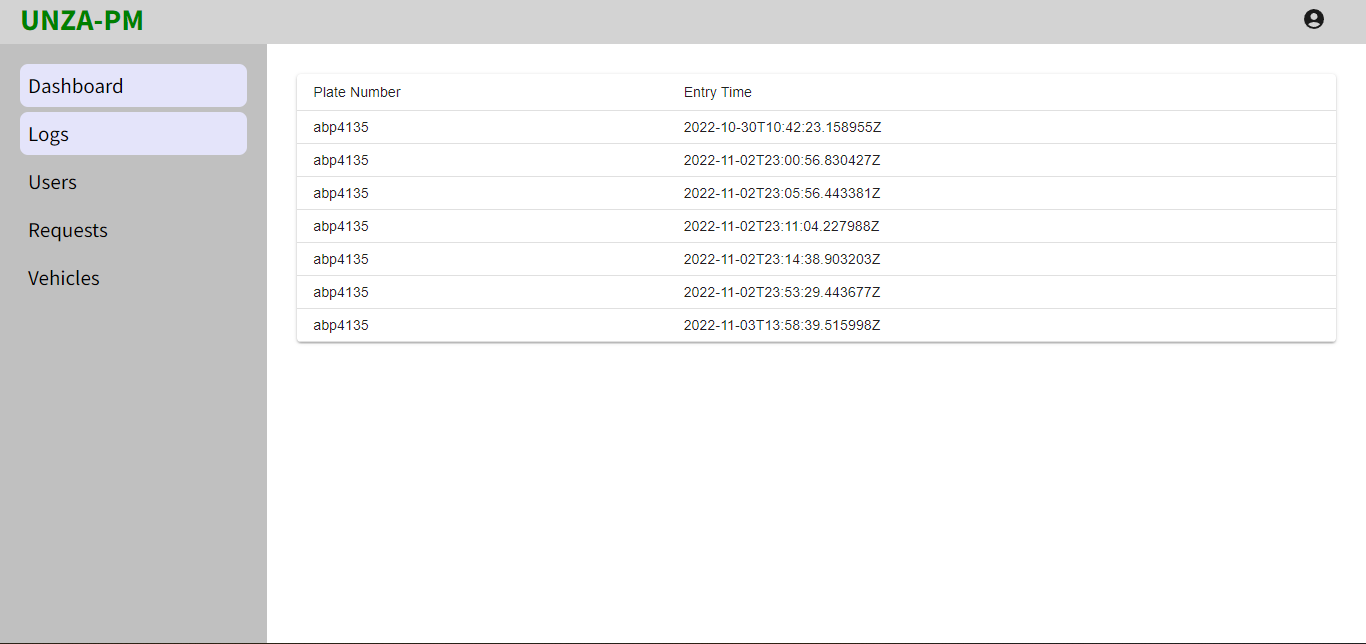
*Figure 21: Admin web app login page*

The login page below provides access into the web app.



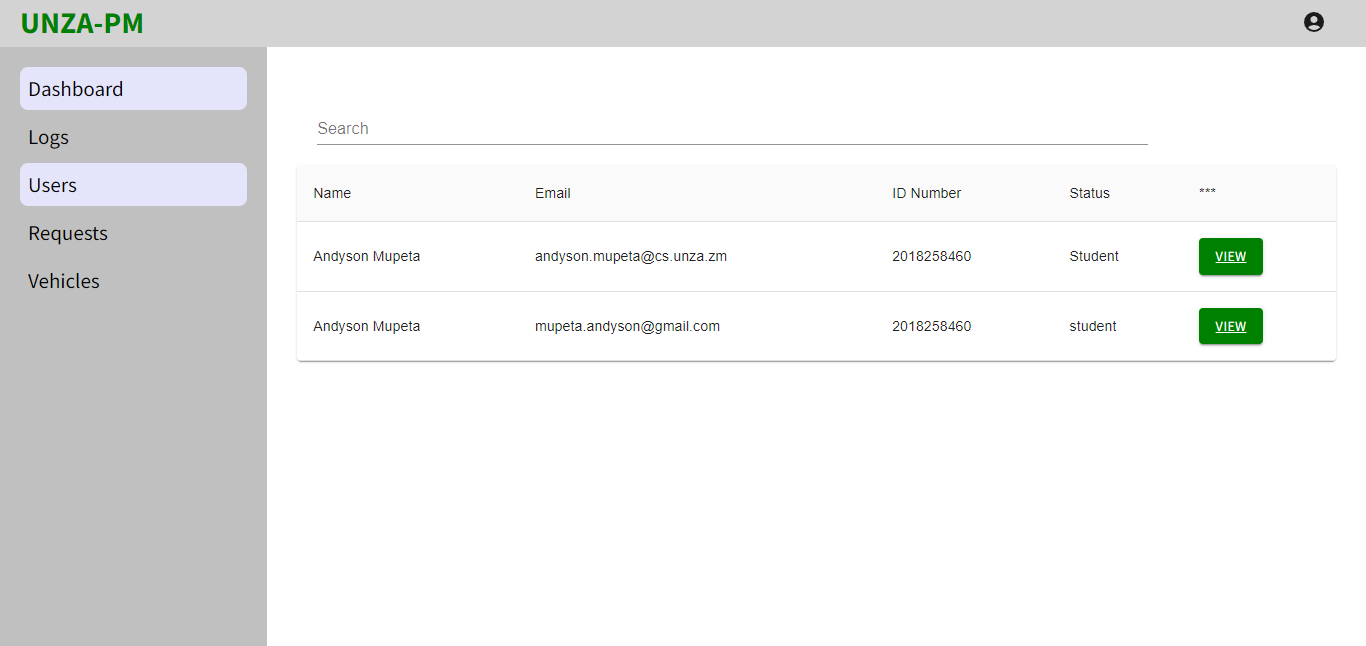
*Figure 22: Admin web app dashboard*

The dashboard is the main display shown when the platform is accessed. Features the parking lot capacity indicators and the open and close gate buttons



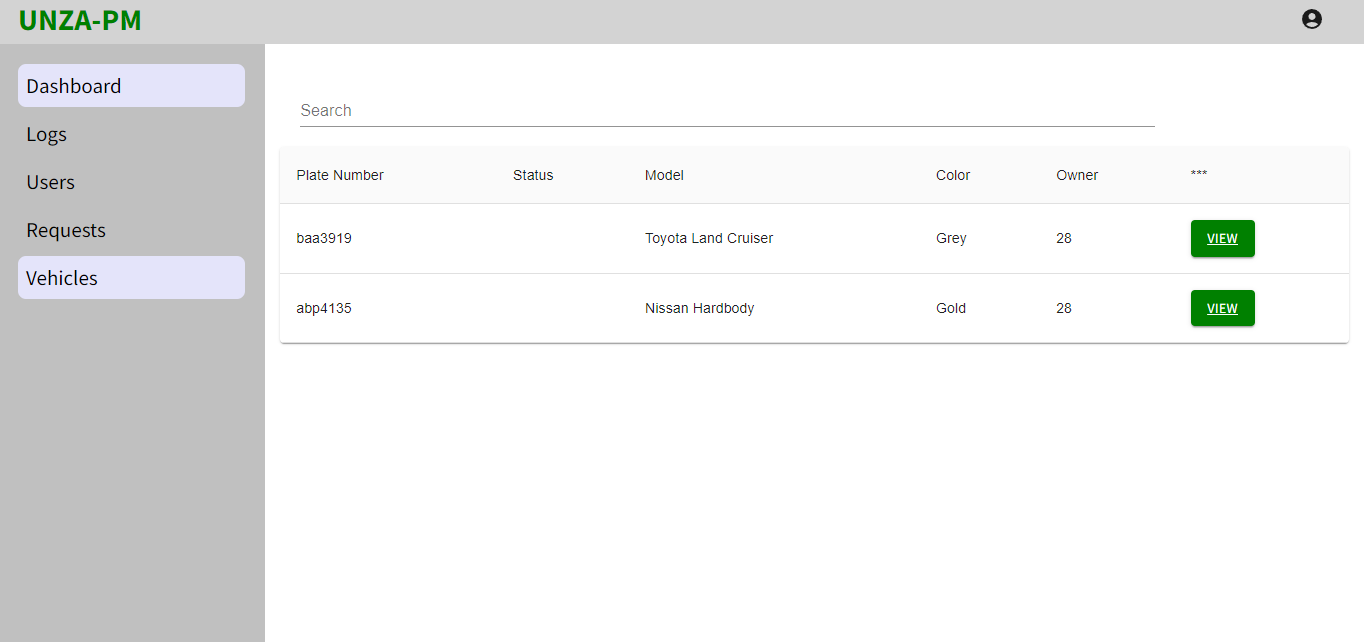
*Figure 23: Admin web app logs page*

The logs are a record of when and who accessed the gate. The logs page display this.



*Figure 24: Admin web app users page*

The users page lists the approved uses and their details. Clicking on view goes to the user detail view which gives further details on the user.



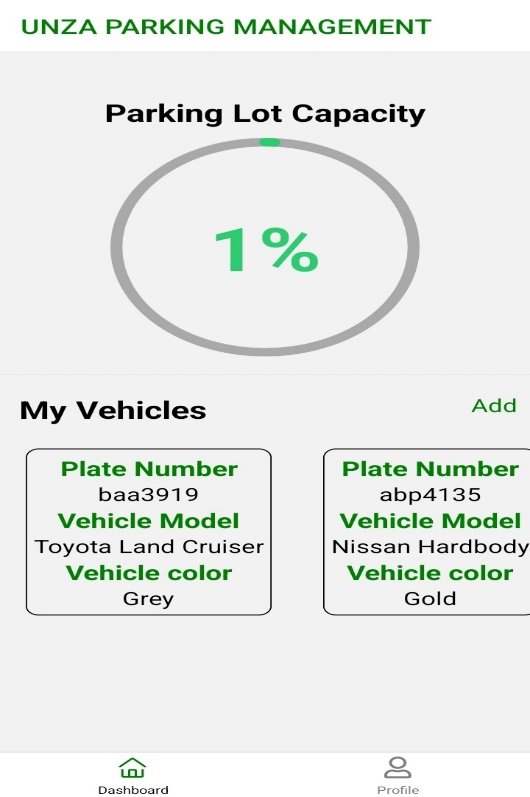
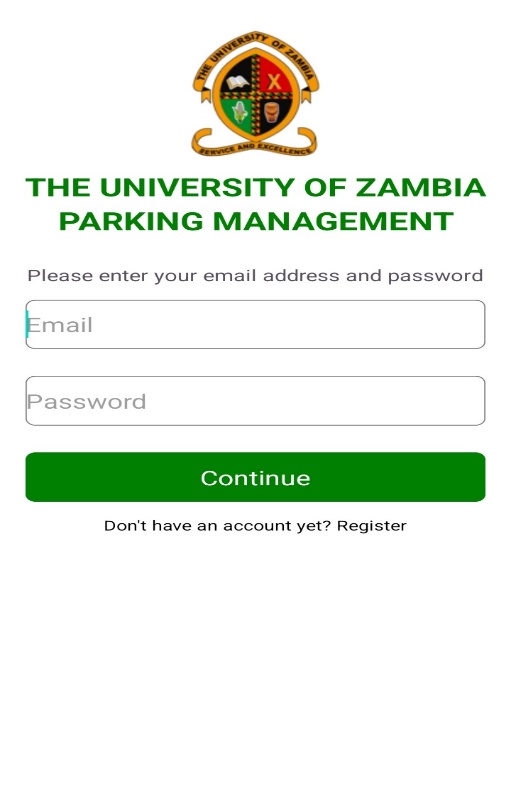
*Figure 25: Admin web app vehicles page*

The vehicles lists all the vehicles that are currently registered. A user can register multiple vehicles.

The admin web portal was build with Reactjs and is hosted on Netlify.

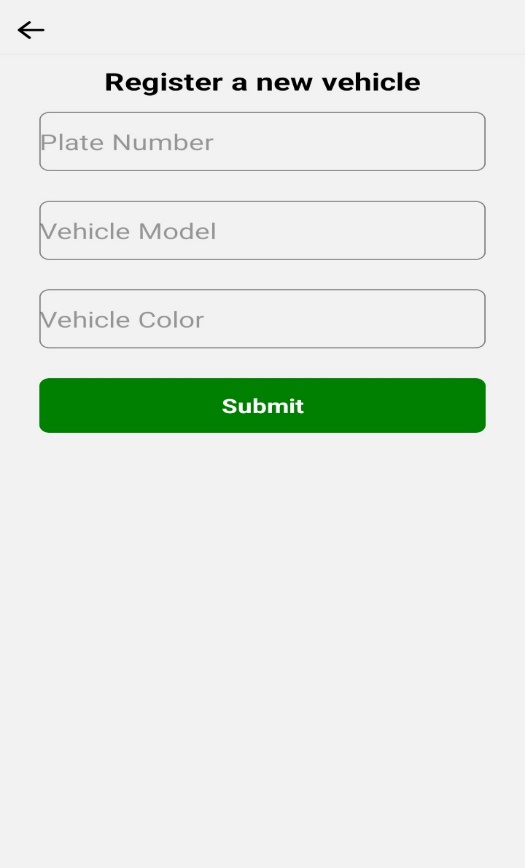
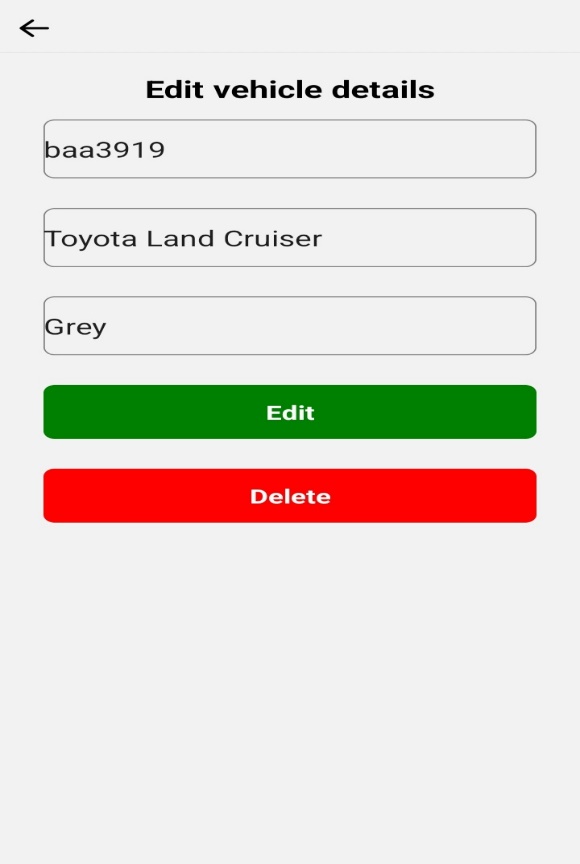
* 1. **USER MOBILE APPLICATION**

The user mobile app is compirsed of a dashboard and other components that enable a user to easily access the system. Users are able to create accounts view how much space is available in the parking lot, add ,edit and delete their vehicles and also edit their personal information. Only students and staff with authorized ID’s can access the app and only after the administrator approves a request will the user be able to access the gate.



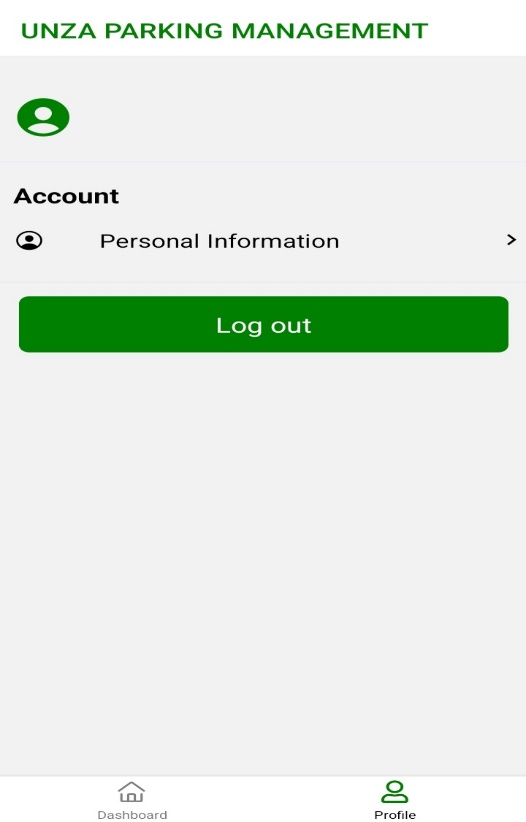
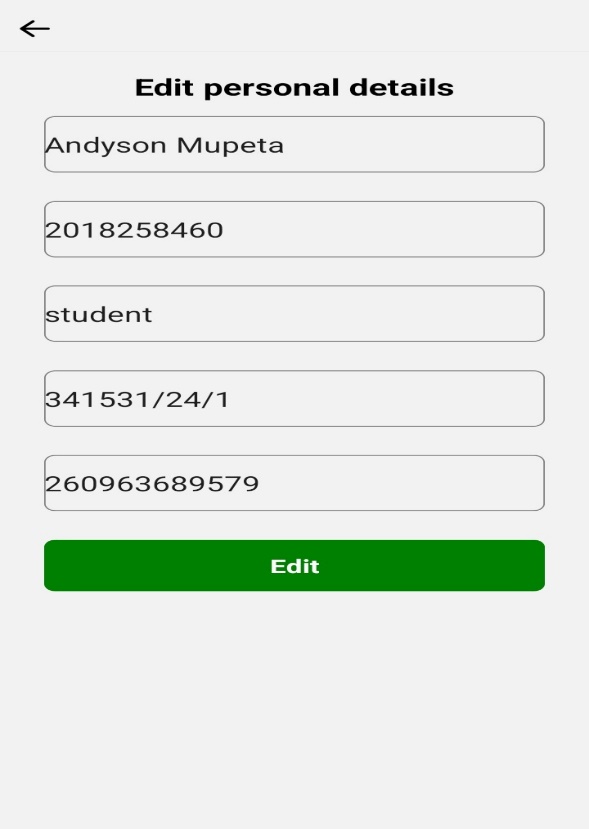
*Figure 26: mobile app login and dashboard*

The login page provides access into the mobile app and the dashboard is the initial page on the app it displays the parking lot capacity details and information about a users cars and a link to add and edit their cars.



*Figure 27: mobile app edit and register vehicle page*

The pages above are the edit and add vehicles pages. The edit page enables you to edit the details of an existing car or delete it. The add page enables you to register a new vehicle to the system.

****

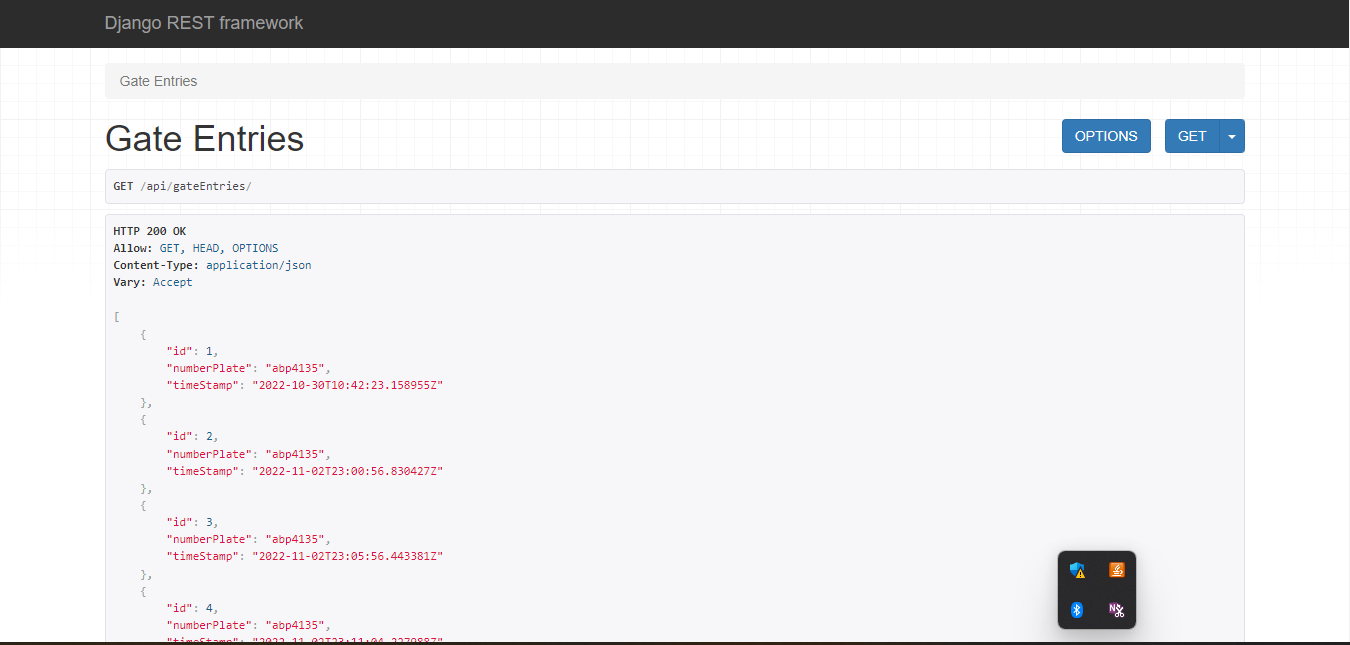
*Figure 27: mobile app edit presonal details and profile page*

The profile page has the logout button which enables the user to exit the app. The edit user detail page enables the user to change his details.

The app was built with React Native and Expo and can be accessed on the expo store.

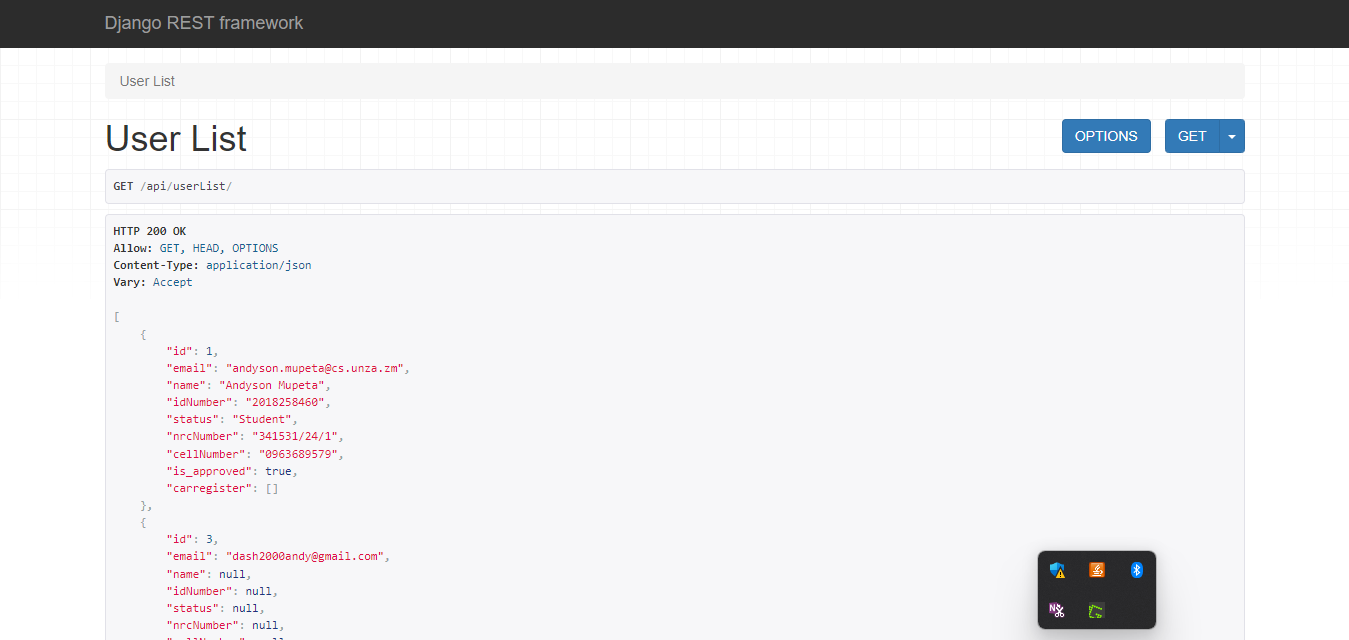
* 1. **APPLICATION PROGRAMMING INTERFACE(API)**

The Application Programming Interface(API) serves as a middle man between the frontend(the admin web portal and mobile app) and the database(SQL). It consists of several endpoints for different POST,PUT AND DELETE functions.



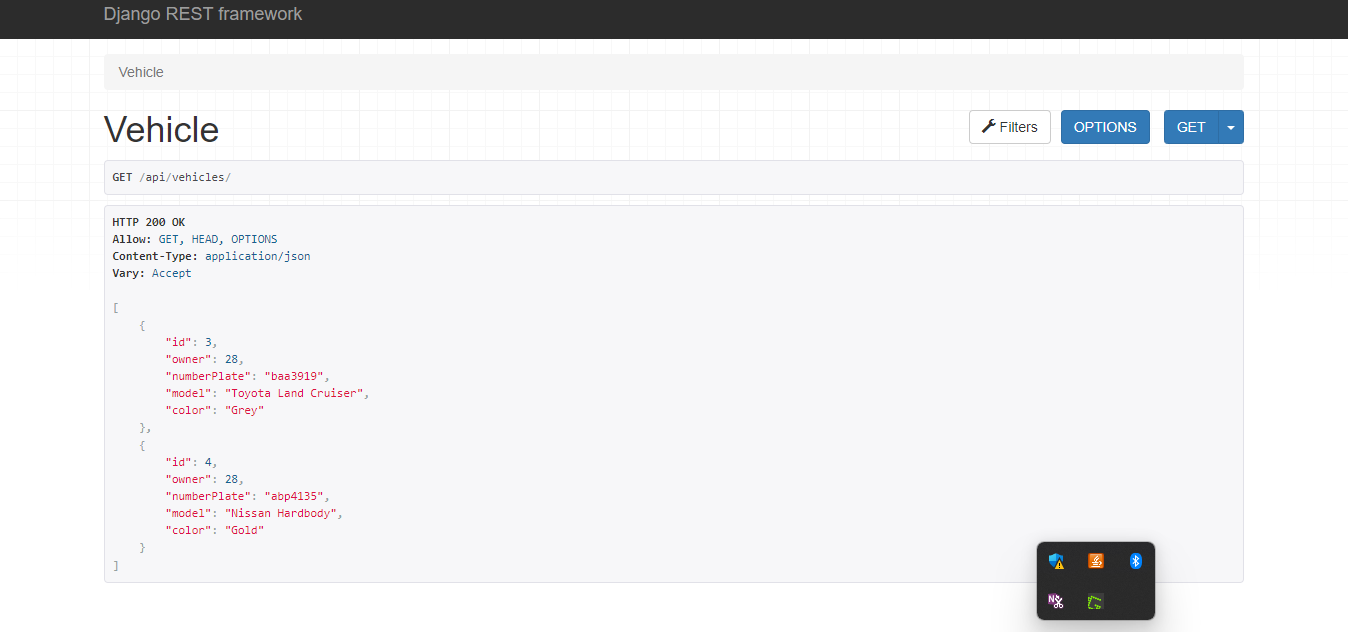
*Figure 28: Application programming interface gate entries endpoint*

Gate entries endpoint displays the list the type each vehicle accessed the gate.



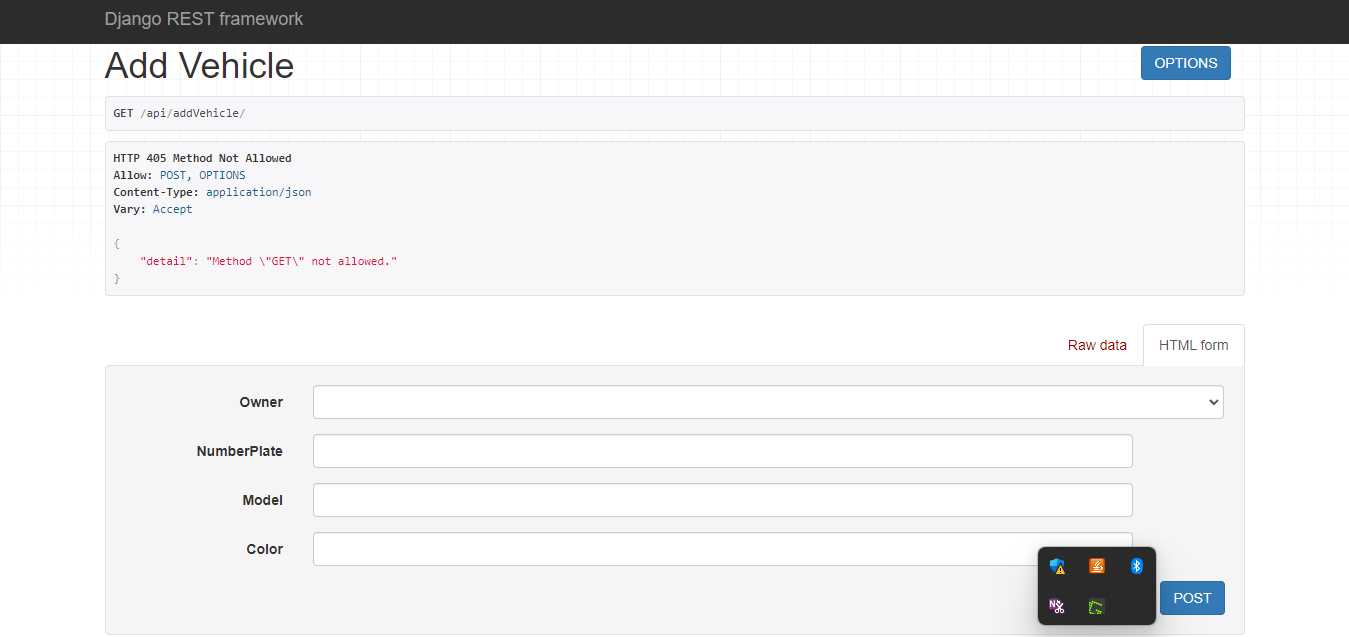
*Figure 29: Application programming interface user list endpoint*

User list endpoint displays the number of users and their details which includes their vehicles



*Figure 30: Application programming interface vehicles endpoint*

The vehicles list endpoint lists all the vehicles currently registered in the system.

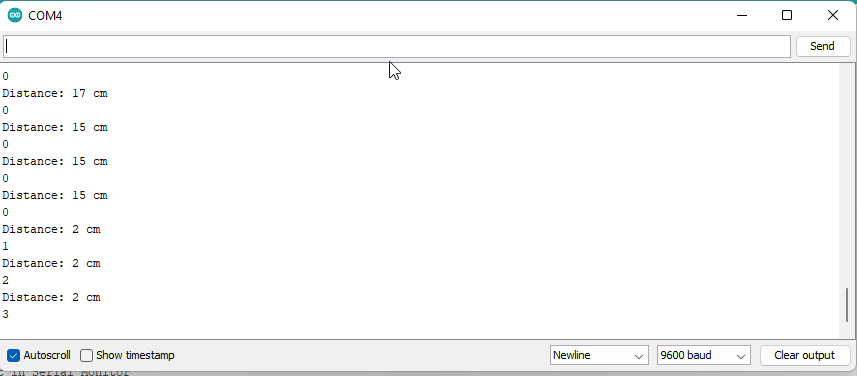


*Figure 31: Application programming interface add vehicles endpoint*

The add vehicle endpoint enables the adding of vehicles to the list of vehicles. When a user adds a vehicle it is done through the add vehicle endpoint.

* 1. **ARDUINO UNO PROGRAM**

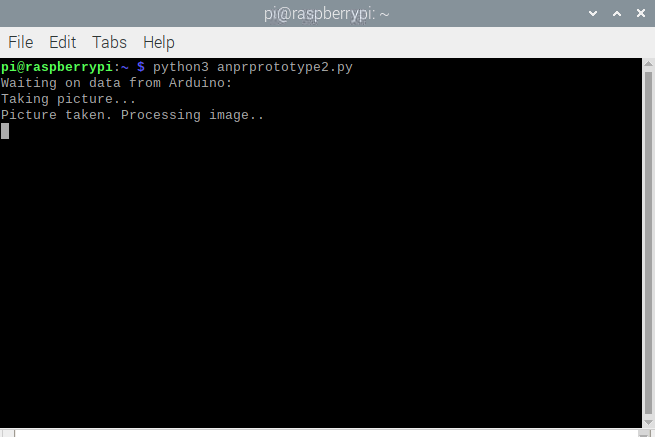
The Arduno Uno software setup connects the all the components together. The picture below shows the serial output console window. This show the different distances picked up by the ultrasonic sensor.



*Figure 32: Arduino Uno serial output console results*

* 1. **RASPBERRY PI PROGRAM**

If the Raspberry Pi is the brain of the whole system the Raspberry Pi program is the directions that control the brain. It connect everything from the API to the Arduino Uno. The result of the Raspberry Pi program looks something like this when running.

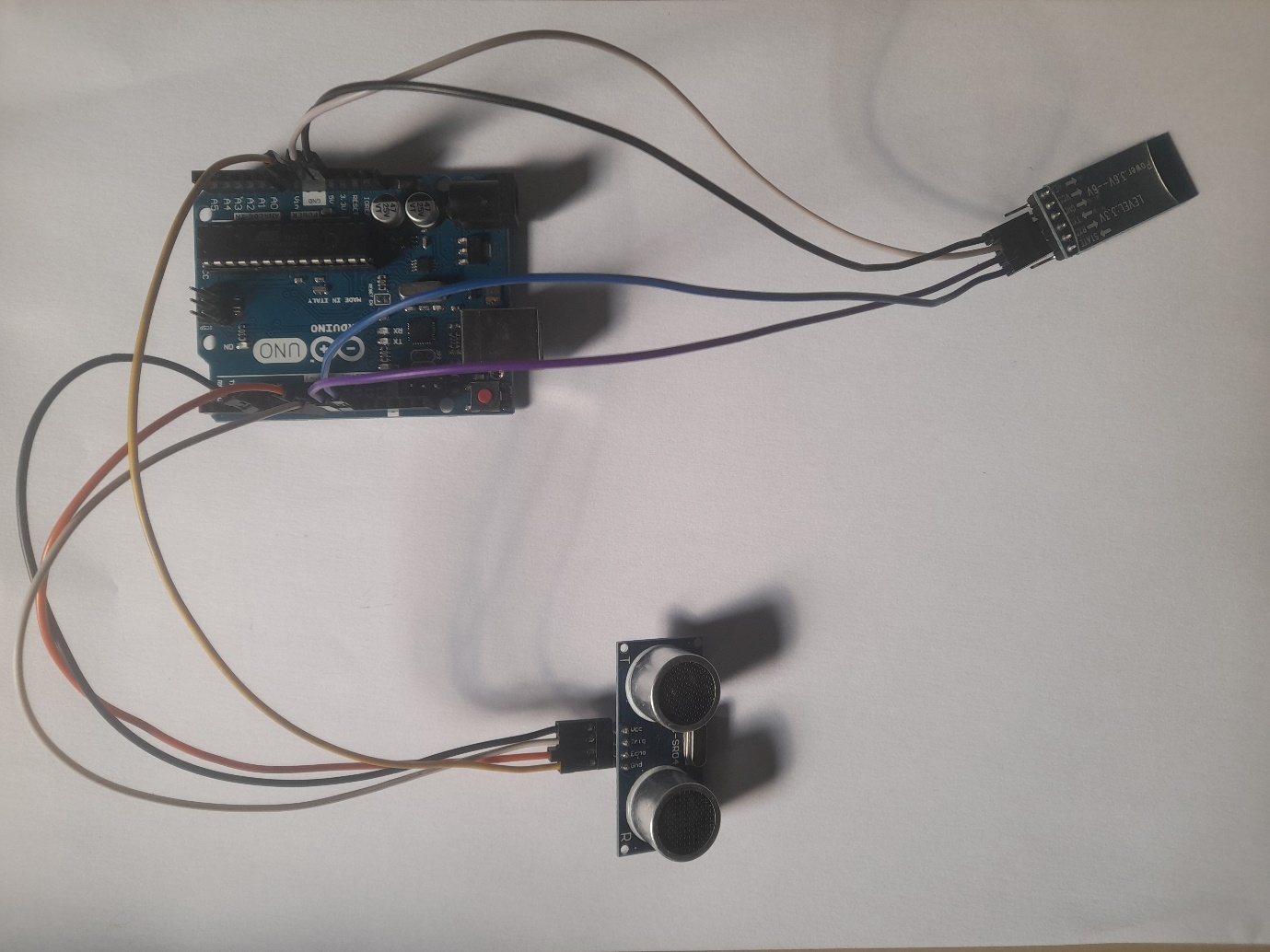


*Figure 33: Raspberry Pi console running system program*

**HARDWARE AND ELECTRONICS RESULTS**

This project has the following hardware connected and setup successfully:

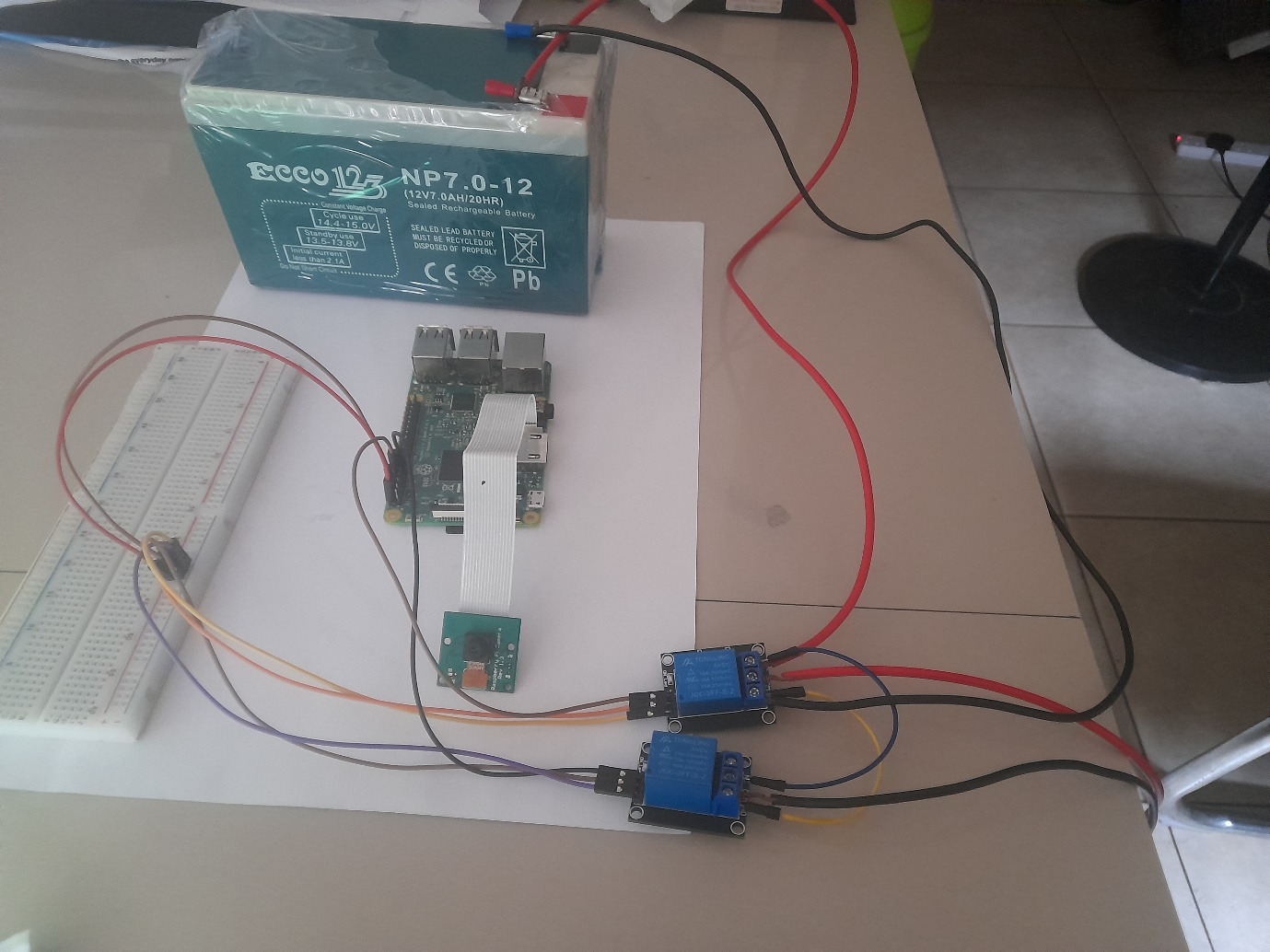
* Raspberry Pi
* 2 Relays
* Pi Camera
* Arduino Uno
* Bluetooth Module
* UltraSonic Sensor
* BreadBoard
* Battery
* 12V Car Wiper Motor
* Connecting Wires
* Charge Controller
* Solar Panel
  1. **ARDUINO UNO SETUP**

****

*Figure 34: Arduino Uno setup(bluetooth module and ultrasonic sensor)*

The Arduino Uno setup consists of an Arduino microcontroller, a bluetooth module and an ultrasonic sensor connected all together with connecting wires.

* 1. **RASPBERRY PI SETUP**



*Figure 35: Raspberry Pi setup*

Consists of the Raspberry Pi microcontroller connected to 2 relays, the breadboard,the Pi camera and the battery.

* 1. **CHARGE CONTROLLER AND SOLAR PANEL**



*Figure 36: Solar charge controller and solar panel*

* 1. **12V DC MOTOR**

****

*Figure 37: 12V DC motor*

The image shows the motor in the housing unit. It has five pins, 1 negative and 4 positives. Rotates to enable the opening and closing of the gate

* 1. **HOUSING UNIT**



*Figure 38: Housing structure(front view)*



*Figure 39: Housing structure(back view)*

The images above are the back and front view of the housing unit. The housing unit is used to house and organise all the required components and also keeps the actual gate barrier attached to it. The gate barrier sits on the rotating pin of the motor and the motor is physicall attached to the gate on the inside.

* 1. **CHAPTER SUMMARY**

This chapter showed the results that were attained by following the methods highlighted in the methodology. It uses pictures to give a better view, showing how the hardware is connected and how it works together and also showing how the admin web app, the mobile app and application programming interface interact with the microcontroller and database to produce the desired functionality.

1. **DISCUSSION AND CONCLUSION**

This chapter brings the report to a conclusion by disscusing the results and challenges faced in trying to achieve the projects aims and objectives.

The project was a success because we managed to build and deploy a mobile app, application programming interface (api) and admin web app, programmed and connected 2 microcontroller and their components and built a housing unit in order to create a working model of automatic number plate detection boom gate using optical character recognition used to achieve the project goal of improving access control in specified areas. The main challenge faced in this project was the cost of development, purchases ranged from components to fabrication costs to mobile bundles used for different purposed throught development.

1. **REFERENCES**

[1] B.D. Acosta, Experiments in image segmentation for automatic US license plate recognition, M.Sc. thesis, Department of Computer Science, Faculty of the Virginia Polytechnic Institute and State University (2004).

[2] C.N.E. Anagnostopoulos, I.E. Anagnostopoulos, I.D. Psoroulas, V. Loumos and E. Kayafas, License Plate Recognition From Still Images and Video Sequences: A Survey, IEEE Transactions on Intelligent Transportation Systems 9 (2008), no. 3, 377–391.

[3] C.N.E. Anagnostopoulos, I.E. Anagnostopoulos, V. Loumos and E. Kayafas, A License PlateRecognition Algorithm for Intelligent Transportation System Applications, IEEE Transactions on Intelligent Transportation Systems 7 (2006), no. 3, 377–392.

[4] S.-L. Chang; L.-S. Chen; Y.-C. Chung; S.-W. Chen, Automatic License Plate Recognition, IEEE Transactions on Intelligent Transportation Systems 5 (2004), no. 1, 42–53.

[5] S. Draghici, A neural network based artificial vision system for licence plate recognition, International Journal of Neural Systems 8 (1997), no. 1, 113–126.

[6] B. Enyedi, L. Konyha, C. Szombathy and K. Fazekas, Strategies for fast license plate number localization, Proceedings of the 46th International Symposium Electronics in Marine, Elmar 2004, Zadar, Croatia, June 16-18, 2004, IEEE Press (2004), 579–584 .

[7] J.-W. Hsieh, S.-H. Yu and S.-H. Yu, Morphology-based License Plate Detection from Complex Scenes, IEEE Proceedings of the 16th International Conference on Pattern Recognition, Qubec City, Canada, August 11-15, 2002, Vol. 3 (2002), 176-179.

[8] O. Martinsky, Algorithmic And Mathematical Principles Of Automatic Number Plate Recognition Systems, B.Sc. thesis, Department of Intelligent Systems, Faculty of Information Technology, Brno University of Technology (2007).

[9] J.R. Parker and P. Federl, An Approach To Licence Plate Recognition, University of Calgary 1996.

[10] H. Sarukhanyan, S. Alaverdyan, and G. Petrosyan, Automatic Number Plate Recognition System, Proceedings of the 7th International Conference on Computer Science and Information Technologies, Yerevan, Armenia, September 28-2 October, 2009, Electronic Copy of the CSIT 2009 Proceedings, 347-350.

[11] V. Shapiro, D. Dimov, S. Bonchev, V. Velichkov and G. Gluhchev, Adaptive License Plate Image Extraction, Proceedings of the 5th international conference on Computer systems and technologies, Rousse, Bulgaria, June 17-18, 2004 (K. Boyanov), ACM New York, NY, US (2004), 1-7

[12] S. Nicu, M. S. Lew, Y. Sun, I. Cohen, T. Gevers and T. S. Huang, "Image and Vision Computing," Authentic facial expression analysis, pp. 1856-1863, 2007.

[13] K. Ioannis, "Recognition of complex human activities in multimedia streams using machine learning and computer vision," Cardiff University, 2013.

[14] A.-B. Ali, A. Tarhini and A. I. Khan, "Exploring Big Data Governance Frameworks," in 9th International Conference on Emerging Ubiquitous Systems and Pervasive Networks

(EUSPN 2018), Leuven, 2018.

[15] Z. Fan, W. Li, Y. Zhang and Z. Feng, "Data Driven Feature Selection for Machine Learning Algorithms in Computer Vision," IEEE Internet of Things Journal, pp. 4262-4272, 2018.

[16] B. Raymond, A. Koene, A. Dix, J. Boger, M. D. Mulvenna, M. Galushka, B. W. Bradley, H. W. Fiona Browne and A. Wong, "Democratisation of Usable Machine Learning in Computer Vision.," arXiv preprint arXiv:1902.06804, 2019.

[17] F. Alireza, "Recent Hot Machine Learning Hammers used in Computer Vision," 2012.

[18] E. Floriana and D. Malerba, "Machine learning in computer vision," *Applied Artificial Intelligence,* vol. 705, p. 693, 2001.

[19] A. Kulkarni Kiran, "Classification of Faults in Railway Ties using Computer Vision and Machine Learning," *Virginia Tech,* 2017.

[20] D. Ami, "Human Detection, Pose and Shape Estimation and," Computer Vision and Machine Learning for Biomechanics Applications Tracking in Unconstrained Environment From Uncalibrated Images, Videos and Depth, 2017.

[21] P. Constantine and T. Poggio, " A trainable system for object detection," International journal of computer vision, pp. 15-33, 2000.

[22] K. Patel Kumar, A. Kar and M. Khan, "Development and an Application of Computer Vision System for Nondestructive Physical Characterization of Mangoes.," Agricultural Research.

[23] E. Floriana, D. Malerba and F. A. Lisi, "Machine learning for intelligent processing of printed documents," Journal of Intelligent Information Systems, pp. 175-198, 2000.

[24] W. David, J. D. Dunn, A. C. Schmid and R. I. Kemp, "Error rates in users of automatic face recognition software.," PloS one, 2015.

[25] S. Carsten, M. Ulrich and C. Wiedemann, "Machine vision algorithms and applications," John Wiley & Sons, 2018.

[26] D. Tirtharaj and T. Nayak, "English Character Recognition using Artificial Neural Network," arXiv preprint arXiv:1306.4621, 2013.

[27] H. Brody, T. Wang, S. Tandon and J. Kiske, "An empirical evaluation of deep learning on highway driving," arXiv preprint arXiv:1504.01716, 2015.

[28] A. K. S. Ankush K Singh, "Behaviour analysis of machine learning algorithms for detecting P2P botnets," in International Conference on Advanced Computing Technologies.

[29] H. Sunao, S. Kobayashi and M. Abe, "Sound collection systems using a crowdsourcing approach to construct sound map based on subjective evaluation," in IEEE International Conference on Multimedia & Expo Workshops, 2016.

[30] P. Álava and J. Antonio, " A brief survey of application areas. in Argentine Symposium on Artificial Intelligence (ASAI 2015)-JAIIO 44 (Rosario, 2015)," Computer Vision and Medical Image Processing, 2015.

[31] D. R. Stephanie, "Mapping Sub-Saharan African Agriculture in High-Resolution Satellite

Imagery with Computer Vision & Machine Learning," Princeton University, 2017.

[32] N. Mubin Abd, E. Nadarajoo, H. Z. M. Shafri and A. Hamedianfar, "Young and mature oil

palm tree detection and counting using convolutional neural network deep learning method,"

International Journal of Remote Sensing, p. 14, 2019.

[33] K. S. Simen, "Automated Front Detection-Using computer vision and machine learning to explore a new direction in automated weather forecasting," The University of Bergen, 2017.

[34] M. Shounak, "Applications of Machine Learning and Computer Vision for Smart

Infrastructure Management in Civil Engineering," 2017.

[35] A. Ian Gamrat, "Estimating Traffic Levels in Montreal using Computer Vision and Machine Learning Technique," Forbes, 2015. [Online]. Available:

http://rl.cs.mcgill.ca/comp598/fall2014/comp598\_submission\_95.pdf. [Accessed 13 June

2021].

[36] W. Haiyan, H. Owens, J. Smith, W. Chernicoff, M. Mazari and M. Pourhomayoun, "The

Challenges in Real-Time Processing," in 3rd International Conference on Advances in

Signal, Image and Video Processing, Nice, 2018.

[37] Y. Gu and J. Yang, "Application of Computer Vision and Deep Learning in Breast Cancer

Assisted Diagnosis," in 3rd International Conference on Machine Learning and Soft

Computing, 2019.

[38] V. S. Anant, "Survey of Computer Vision and Machine Learning in Gastrointestinal

Endoscopy," arXiv preprint arXiv:1904.13307, 2019.

[39] S. Rui, i. Teixeira, B. Barbosa, A. J. R. Neves, S. C. Soares and I. D. Dimas, "Human Body Posture Detection in Context: The Case of Teaching and Learning Environments.," in Third International Conference on Advances in Signal, Image and Video Processing, Nice, 2018.

[40] C. Trigueiros and P. J. d. Albuquerque, "Hand Gesture Recognition System based in

Computer Vision and Machine Learning Applications on Human-Machine Interaction, in

Tese de Doutoramento Electronic and Computer Engineering," Universidade do Minho

Escola de Engenharia, 2013.

[41] J. Zhang, H. S. Naik, T. Assefa and S. Sarkar, "Computer vision and machine learning for robust phenotyping in genome-wide studies," Scientific reports, 2017.

[42] N. Pedro, F. Pérez, J. Weiss and M. Egea-Cortines, "Machine learning and computer vision system for phenotype data," Sensors, 2016.

[43] C. Roshni, "Accelerating neuronal genetic research in C. elegans with computer vision &

machine learning.," CS231A Final Project Report, 2012.

[44] S. Asif, Z. Ahmad, R. Rayan and L. Alam, "An approach to automate the scorecard in cricket with computer vision and machine learning," in 3rd International Conference on Electrical Information and Communication Technology, 2017.

[45] V. Mora and Silvia, "Computer vision and machine learning for in-play tennis analysis:

framework, algorithms and implementation," in Imperial College, London, 2018.

[46] J. Brownlee, "A Gentle Introduction to Transfer Learning for Deep Learning," Deep Learning for Computer Vision, 16 December 2017.

[47] A. Hussain, M. Hammad, K. Hafeez and T. Zaibnab, "Programming a microcontroller,"

International Journal of Computer Applications, December 2016.

[48] J. M. Perkel, "Why Jupyter is data scientists’ computational notebook of choice,"

TOOLBOX, 30 October 2018.

[49] T. Ojala, M. Pietikainen and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 971-987, 2002.

[50] T. H. Kim, D. C. Park, D. M. Woo, T. Jeong and S. Y. Min, "Multi-class classifier-based

adaboost algorithm," in International conference on intelligent science and intelligent data

engineering.

[51] S. Yang, P. Luo, C. C. Loy and X. Tang, "Wider face: A face detection benchmark," in

Proceedings of the IEEE conference on computer vision and pattern recognition, 2016.

[52] B. F. Klare, B. Klein, E. Taborsky and A. Blanton, "Pushing the frontiers of unconstrained face detection and recognition: Iarpa Janus Benchmark A," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2015.

[53] B. Yang, J. Yan, Z. Lei and S. Z. Li, "Fine-grained evaluation on face detection in the wild," IEEE, pp. 1-7, May 2015.

[54] P. Jones, P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," University of Rochester; Charles Rich, 2001.

[55] K. Zhang, Z. Zhang, Z. Li and Y. Qiao, "Joint face detection and alignment using multi-task cascaded convolutional networks.," IEEE Signal Processing Letters, pp. 1499-1503, 2016.

[56] D. Chen, S. Ren, Y. Wei, X. Cao and J. Sun, "Joint cascade face detection and alignment,"

in European conference on computer vision, 2014.

[57] S. Ren, K. He, R. Girshick and J. Sun, "Advances in neural information processing systems," Faster r-cnn: Towards real-time object detection with region proposal networks, pp. 91-99, 2015.

[58] H. Li, Z. Lin, X. Shen, J. Brandt and G. Hua, "A convolutional neural network cascade for face detection," in Proceedings of the IEEE conference on computer vision and pattern

recognition, 2015.

[59] M. Opitz, G. Waltner, G. Poier, H. Possegger and H. Bischof, " Grid loss: Detecting occluded vehicles," in European conference on computer vision, 2016.

[60] G. J. Chowdary, N. S. Punn, S. K. Sonbhadra and S. Agarwal, "Liscense Mask Detection Using Transfer Learning of InceptionV3," 3 January 2012.

[61] M. Fang, C. Liang, and X. Zhao, "A method based on rough set andSOFM neural network for the car's plate character recognition, " inIntelligent Control and Automation, WCICA 2004, Fifth World Congress on, 5, 2004, 4037-4040. IEEE.

[62] N. A. Jusoh, J. M. Zain, and T. A. A. Kadir, "Enhancing Thinning Method for Malaysian Car Plates Recognition, "in Innovative Computing, Information and Control, ICICIC'07, Second International Conference on IEEE, 2007, 378-378. IEEE.

[63] E. R. Lee, P. K. Kim, and H. J. Kim, "Automatic recognition of a car license plate using color image Processing, " in Image Processing, ICIP-94., IEEE International Conference, 2, 1994, 301-305.

[64] S. Du, M. Ibrahim, M. Shehata, and W. Badawy, "Automatic License Plate Recognition (ALPR): A State-of-the-Art Review," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 23, no. 2, pp. 311-325, Feb. 2013.

[65] A. Mutholib, T. S. Gunawan, and M. Kartiwi, "Design and implementation of automatic number plate recognition on android platform," 2012 International Conference on Computer and Communication Engineering (ICCCE), Kuala Lumpur, 2012, pp. 540-543.

[66] Chuin-Mu Wang, and Ching-Yuan Su, "Fast license plate location and recognition using wavelet transform in android," 2012 7th IEEE Conference on Industrial Electronics and Applications (ICIEA), Singapore, 2012, pp. 1035-1038.

[67] H. F. Chen, C. Y. Chiang, S. J. Yang, and C. C. Ho, "Android-based patrol robot featuring automatic license plate recognition," 2012 Computing, Communications and Applications Conference, Hong Kong, 2012, pp. 117-122.

[68] R.K. Romadhon, M. Ilham, N.I. Munawar, S. Tan, and R. Hedwig," Android-based license plate recognition using pre-trained neural network," in Internet Working Indonesia Journal, 2012, 15-18.

[69] H. N. Do, M. T. Vo, B. Q. Vuong, H. T. Pham, A. H. Nguyen, and H.Q. Luong, "Automatic license plate recognition using mobile device," 2016 International Conference on Advanced Technologies for Communications (ATC), Hanoi, 2016, pp. 268-271.

[70] J. Chen, "Chinese license plate identification based on Android platform," 3rd International Conference on Computational Intelligence & Communication Technology (CICT), Ghaziabad, 2017, pp. 1-5.

[71] T. Kohonen, "Self-Organizing Maps, Springer-Verlag, " Berlin, Heidelberg, New York, 1995.

[72] C.K. On, T.K. Yao, R. Alfred, A.A.A. Ibrahim, W. Cheng, and T.T. Guan," A Comparison of BPNN, RBF, and ENN in Number Plate Recognition" in Berry M., Hj Mohamed A., Yap B. (eds) Soft Computing in Data Science. SCDS 2016. Communications in Computer and Information Science, vol 652. Springer, Singapore

[73] R. Smith, "An Overview of the Tesseract OCR Engine," Ninth International Conference on Document Analysis and Recognition (ICDAR 2007), Parana, 2007, pp. 629-633.

[74] G. Bradski, "The OpenCV Library", Dr. Dobb's Journal of Software Tools, 2000.

[75] C. Busch, R. Domer, C. Freytag, and H. Ziegler, "Feature based recognition of traffic video streams for online route tracing," Vehicular Technology Conference, 1998. VTC 98. 48th IEEE, Ottawa, Ont., 1998, pp. 1790-1794 vol.3.

[76] P. Hurtik, and M. Vajgl, "Automatic license plate recognition in difficult conditions — Technical report," 2017 Joint 17th World Congress of International Fuzzy Systems Association and 9th International Conference on Soft Computing and Intelligent Systems (IFSA-SCIS), Otsu, 2017, pp. 1-6.

[77] S. Li, and Y. Li, "A Recognition Algorithm for Similar Characters on License Plates Based on Improved CNN," 2015 11th International Conference on Computational Intelligence and Security (CIS), Shenzhen, 2015, pp. 1-4.

[78] D. N. T. How, and K. S. M. Sahari, "Character recognition of Malaysian vehicle license plate with deep convolutional neural networks," 2016 IEEE International Symposium on Robotics and Intelligent Sensors (IRIS), Tokyo, 2016, pp. 1-5.

[79] D. Yao, W. Zhu, Y. Chen, and L. Zhang, "Chinese license plate character recognition based on convolution neural network," 2017 Chinese Automation Congress (CAC), Jinan, 2017, pp. 1547-1552.

[80] M. Z. Abedin, A. C. Nath, P. Dhar, K. Deb, and M. S. Hossain, "License plate recognition system based on contour properties and deep learning model," 2017 IEEE Region 10 Humanitarian Technology Conference (R10-HTC), Dhaka, Bangladesh, 2017, pp. 590-593.

[81] R. R. Palekar, S. U. Parab, D. P. Parikh, and V. N. Kamble, "Real time license plate detection using openCV and tesseract," 2017 International Conference on Communication and Signal Processing (ICCSP), CHENNAI, India, 2017, pp. 2111-2115.

1. **APPENDIX**

The appendix provides additional content that is useful to the reader but would have disrupted the flow of the chapter where it was referenced.

* 1. **ARDUINO UNO CODE**

#include<SoftwareSerial.h>

long duration; // variable for the duration of sound wave travel

int distance;

const int trigPin = 12;

const int echoPin = 11;

int RX\_pin=10;

int TX\_pin=9;

SoftwareSerial BTserial(RX\_pin,TX\_pin);

int ledPin = 13;

int ultraPin = 4;

int figure = 1;

void setup() {

// put your setup code here, to run once:

Serial.begin(9600); // // Serial Communication is starting with 9600 of baudrate speed

BTserial.begin(9600);

pinMode(trigPin, OUTPUT); // Sets the trigPin as an OUTPUT

pinMode(echoPin, INPUT); // Sets the echoPin as an INPUT

pinMode(ledPin, OUTPUT);

pinMode(ultraPin, OUTPUT);

pinMode(RX\_pin, INPUT);

pinMode(TX\_pin, OUTPUT);

Serial.println("Ultrasonic Sensor HC-SR04 Test"); // print some text in Serial Monitor

Serial.println("with Arduino UNO R3");

digitalWrite(ultraPin, HIGH);

}

void loop() {

// put your main code here, to run repeatedly:

// Clears the trigPin condition

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

// Sets the trigPin HIGH (ACTIVE) for 10 microseconds

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

// Reads the echoPin, returns the sound wave travel time in microseconds

duration = pulseIn(echoPin, HIGH);

// Calculating the distance

distance = duration \* 0.034 / 2; // Speed of sound wave divided by 2 (go and back)

// Displays the distance on the Serial Monitor

Serial.print("Distance: ");

Serial.print(distance);

Serial.println(" cm");

static bool runOnce = false;

if(distance<10){

digitalWrite(ledPin,HIGH);

// Serial.println(figure);

// BTserial.println(figure);

Serial.println('2');

BTserial.println('2');

// delay(1000);

figure++;

}else{

Serial.println('0');

BTserial.println('0');

digitalWrite(ledPin,LOW);

figure = 1;

// delay(2000);

}

delay(650);

}.

* 1. **RASPBERRY PI CODE**

import keyboard

import requests

from pprint import pprint

from picamera import PiCamera

from time import sleep

import os

import serial

import time

import RPi.GPIO as GPIO

import json

# camera = PiCamera()

# camera.rotation = 180

# camera.resolution=(1920,1080)

# camera.start\_preview()

# sleep(5)

# camera.capture('/home/pi/Downloads/image.jpg')

# camera.stop\_preview()

GPIO.setmode(GPIO.BCM)

GPIO.setwarnings(False)

GPIO.setup(18,GPIO.OUT)

GPIO.setup(17,GPIO.OUT)

registeredPlates = [{'id':'1','plate':'abp456'},

{'id':'2','plate':'abp578'},

{'id':'3','plate':'abp4135'},

{'id':'4','plate':'baa3919'},

{'id':'5','plate':'abc124'}]

val=''

if os.path.exists('/dev/rfcomm0') == False:

#print('hello')

path = 'sudo rfcomm bind 00:20:10:08:40:65'

os.system(path)

time.sleep(1)

#bluetoothSerial = serial.Serial('/dev/rfcomm0', baudrate=9600)

#RXData = bluetoothSerial.readline()

print("Waiting on data from Arduino: ")

isGateOpen = False;

n = True

try:

while n:

bluetoothSerial = serial.Serial('/dev/rfcomm0', baudrate=9600)

RXData = (bluetoothSerial.readline()).strip().decode("utf-8")

val = RXData

gateGate = requests.get('https://andyson2.pythonanywhere.com/api/carPark/1/')

data1 = gateGate.text

parse\_json1 = json.loads(data1)

info2 = parse\_json1['carpark']

gateState = requests.get('https://andyson2.pythonanywhere.com/api/gateControl/1/')

data = gateState.text

parse\_json = json.loads(data)

info = parse\_json['state']

print(val)

# print(info)

# print(info2)

if val=='2':

print('Taking picture...')

camera = PiCamera()

camera.rotation = 180

camera.resolution=(1920,1080)

camera.start\_preview()

sleep(5)

camera.capture('/home/pi/Downloads/image.jpg')

camera.stop\_preview()

camera.close()

print('Picture taken. Processing image..')

regions = ['zm'] # Change to your country

with open('/home/pi/Downloads/nissan.jpg', 'rb') as fp:

response = requests.post(

'https://api.platerecognizer.com/v1/plate-reader/',

#'http://localhost:8080/v1/plate-reader/',

data=dict(regions=regions), # Optional

files=dict(upload=fp),

headers={'Authorization': 'Token 9f6fdfaa8a1ce4542f52e7429fc2cdfdb4de7c8b'})

result = response.json()

newPlate = result['results'][0]['plate']

thislist = []

for plate in registeredPlates:

thislist.append(plate['plate'])

if newPlate in thislist:

print('Open Gate')

GPIO.output(18,GPIO.HIGH)

GPIO.output(17,GPIO.LOW)

time.sleep(2)

GPIO.output(18,GPIO.LOW)

GPIO.output(17,GPIO.LOW)

parkLink = requests.get('https://andyson2.pythonanywhere.com/api/setPark/'+ info2 + '/')

print('step1')

data2 = parkLink.text

print('step2')

parse\_json2 = json.loads(data2)

print('step3')

info3 = parse\_json2['slots']

print('step3')

num = info3 + 1

print(num)

myrtn = {'id':info2,'slots':num}

print('step5')

newLink = 'https://andyson2.pythonanywhere.com/api/setPark/'+ str(info2) + '/'

print('step6')

y = requests.put(newLink,json=myrtn)

print('step7')

url = 'https://andyson2.pythonanywhere.com/api/createGateEntries/'

myobj = {'numberPlate':newPlate}

x = requests.post(url,json=myobj)

print(x.text)

isGateOpen = True

while isGateOpen == True:

RupData = (bluetoothSerial.readline()).strip().decode("utf-8")

print(RupData)

if RupData == '0':

GPIO.output(18,GPIO.LOW)

GPIO.output(17,GPIO.HIGH)

time.sleep(2)

GPIO.output(18,GPIO.LOW)

GPIO.output(17,GPIO.LOW)

isGateOpen = False

print('closed')

break

else:

print('Car not registered!')

elif info == '1':

print('open gate')

GPIO.output(18,GPIO.HIGH)

GPIO.output(17,GPIO.LOW)

time.sleep(1)

GPIO.output(18,GPIO.LOW)

GPIO.output(17,GPIO.LOW)

link = 'https://andyson2.pythonanywhere.com/api/gateControl/1/'

myobj = {'id':1,'state': '0'}

x = requests.put(link,json=myobj)

elif info == '2':

print('close gate')

GPIO.output(18,GPIO.LOW)

GPIO.output(17,GPIO.HIGH)

time.sleep(0.6)

GPIO.output(18,GPIO.LOW)

GPIO.output(17,GPIO.LOW)

link = 'https://andyson2.pythonanywhere.com/api/gateControl/1/'

myobj = {'id':1,'state': '0'}

x = requests.put(link,json=myobj)

* 1. **NUMBER PLATE RECOGNITION SOFTWARE CODE**

import cv2

import imutils

import numpy as np

import pytesseract

from PIL import Image

from picamera.array import PiRGBArray

from picamera import PiCamera

import smtplib

server=smtplib.SMTP('smtp.gmail.com',587)

server.starttls()

server.login("Your Email ID", "Email ID Password")

camera = PiCamera()

camera.resolution = (640, 480)

camera.framerate = 30

rawCapture = PiRGBArray(camera, size=(640, 480))

for frame in camera.capture\_continuous(rawCapture, format="bgr", use\_video\_port=True):

image = frame.array

cv2.imshow("Frame", image)

key = cv2.waitKey(1) & 0xFF

rawCapture.truncate(0)

if key == ord("s"):

gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY) #convert to grey scale

gray = cv2.bilateralFilter(gray, 11, 17, 17) #Blur to reduce noise

edged = cv2.Canny(gray, 30, 200) #Perform Edge detection

cnts = cv2.findContours(edged.copy(), cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)

cnts = imutils.grab\_contours(cnts)

cnts = sorted(cnts, key = cv2.contourArea, reverse = True)[:10]

screenCnt = None

for c in cnts:

peri = cv2.arcLength(c, True)

approx = cv2.approxPolyDP(c, 0.018 \* peri, True)

if len(approx) == 4:

screenCnt = approx

break

if screenCnt is None:

detected = 0

print ("No contour detected")

else:

detected = 1

if detected == 1:

cv2.drawContours(image, [screenCnt], -1, (0, 255, 0), 3)

mask = np.zeros(gray.shape,np.uint8)

new\_image = cv2.drawContours(mask,[screenCnt],0,255,-1,)

new\_image = cv2.bitwise\_and(image,image,mask=mask)

(x, y) = np.where(mask == 255)

(topx, topy) = (np.min(x), np.min(y))

(bottomx, bottomy) = (np.max(x), np.max(y))

Cropped = gray[topx:bottomx+1, topy:bottomy+1]

text = pytesseract.image\_to\_string(Cropped, config='--psm 11')

print("Detected Number is:",text)

server.sendmail("Sender's Email ID@gmail.com","Sender's Email ID@gmail.com",text)

cv2.imshow("Frame", image)

cv2.imshow('Cropped',Cropped)

cv2.waitKey(0)

break

cv2.destroyAllWindows()