Automated Car Parking System with Increased Security by Digital Image Processing

By

Md. Tanzir Chowdhury
12201010
Sheikh Ashik Rahman
13101004
Md. Fahad Islam
19141034
Md. Rezwan Mallick
12201031

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
August 2019

Declaration

It is hereby declared that

1. The thesis submitted is our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Authors:

Md. Tanzir Chowdhury	Sheikh Ashik Rahman
12201010	13101004
Md. Fahad Islam	Md. Rezwan Mallick
19141034	12201031

Approval

The thesis/project titled "Automated Car Parking System with Increased Security by Digital Image Processing" submitted by

1. Md. Tanzir Chowdhury (12201010)
2. Sheikh Ashik Rahman (13101004)
3. Md. Fahad Islam (19141034)
4. Md. Rezwan Mallick (12201031)

Of Summer, 2019 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on August 28, 2019.

Examining Committee:

Supervisor:

Dr. Jia Uddin

Associate Professor
Department of Computer Science and Engineering
Brac University

Program Coordinator:

Dr. Jia Uddin

Associate Professor
Department of Computer Science and Engineering
Brac University

Head of Department:

Mahbubul Alam Majumdar

PhD Professor and Chairperson
Department of Computer Science and Engineering
Brac University

Abstract

In Bangladesh car stealing is very common. Our cars are not secure even in the parking garages. Security guards can not monitor large garages properly all the time. If someone can unlock the car he can easily pass through the security guards by providing a fake receipt. So we thought of introducing a digital parking system for the very first time in our country. This reduces the manpower required in the larger garages of shopping mall or convention halls or offices and also keeps the cars secured as the whole system is digitized with the help of image processing. When a car enters a garage it will automatically save the car's and driver's information in the database. The car's information is taken and stored through image processing. The car's owner will be given access to the system through a mobile application where he will be notified if someone else tries to steal his/her car. He/she needs to confirm while exiting the garage also. Thus we can provide a very unique and efficient way to keep our cars safe.

Acknowledgement

We would like to thank Almighty Allah for providing us the opportunity to complete our thesis without any sort of difficulties. A major thanks goes to our dear Advisor Dr. Jia Uddin sir for his constant support, feedback and advice during the process of our work. He helped us in every way possible and and shaped our thesis in what it is today. We would also like to thank our whole judging panel for their reviews and feedback. All the feedback and recommendations they gave us helped us a lot. And last but not the least, our parents, friends and mentors who helped us throughout the journey. Without their support it may not have been possible at all.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the modern world, object detection has become a major part of many important applications and industries to shape up the user experience and saving up enough time for more work. Object detection is an old topic on which several research has been done. In order to achieve more efficiency and accuracy we can work on the remaining scopes for improvement.

Object detection is basically a method of identifying objects based on their shape, color, size, characteristics and much more. The primary approach is to construct a computer vision that emulates the detection process of normal vision. The color-based approach is the mostly used object detection method used because of its efficiency. However, the pace of the detection needs improvement.

In order to extract information from images, scholars from all over the world applied many methods considering geometry and texture. Segmentation is one of the most important steps for information extraction from images. At present, optimal scale calculation method mainly uses calculation models, expert experience, objective functions, and much more.

This paper proposes a new considerable option for object detection in Bangladesh through OpenCV library.

OpenCV (Open Source Computer Vision) is one of the most widely used machine learning software library [7] in order to solve computer vision problems. OpenCV-Python is a fast Python API for OpenCV. The background has C/C++ coding included, that is very easy to code.

Chapter 2

Literature Review

In the past few years, object detection using image processing has been a major area of research and many proposals of approach were suggested. Many researches were conducted on this topic.[6] Such as

1. The IEEE paper titled "Face Detection and Tracking using OpenCV" by Mukund Katti, Aditya Khatawkar, S.V. Viraktamath, & Pavan Kulkarni has conducted a study on openCV that converts 2nd dimentional webcam Images into 3rd dimensional Images of human faces by constructing 3rd dimensional geometrical data outputs.[5] The successful implementation of automatic face detection and tracking was done in the Prototype system. The test results showed accurate face detection results. They used openCV and arduino real time for the intersection of image processing and embedded systems.

2. The paper "Sophisticated Image Encryption Using OpenCV" written by Sunnet Kumar, Ashish Pant, Arjun Arora, and Prof. R.P. Arora form DIT Dehradun focused on image Processing and image encryption and safe transfer over the networks.[3] Data structure of the OpenCV has also been analyzed in detail.

3. Kevin Hughes, wrote a lot of blogs and project tutorials and instructions to use softwares on openCV.

4. IEEE members S. Belongie and J. Malik have conducted a study on their publication "Contour and Texture Analysis for Image Segmentation" that focused on Shape Matching and Object Matching and have developed a general algorithm that partitions all the gray scale images into disjointed regions of coherent brightness and texture.[1]

5. Orlando J. Tobias and Rui Seara, Member of IEEE have written a paper called "Image Segmentation by Histogram Thresholding Using Fuzzy Sets" that focused on the ways and techniques for Image Segmentation and histogram Thresholding. They have introduced a procedure for histogram thresholding [2] which is not based on the minimization of a criterion function, rather it focuses on the similarity between gray levels.

Object detection research might be a common thing nowadays but there was hardly any implementation on the security services in Bangladesh. Most of the services are not automated service, and not all the residential and commercial properties are provided with adequate protection against unusual conditions. Security is important for every man owned property in order to prevent thefts and to ensure safety for personal belongings. If stealth is unprevented, it can cost someone a fortune. Security systems can help prevent robberies and thefts and ensure safety. They all require integrated security solutions. In order to bring change in that situation, we decided to come up with a solution that has not yet been implemented, but can be considered a milestone in the near future.

Chapter 3

Proposed Model

3.1 Initial data collection

Initially we have taken pictures of 100 different cars . We have tested our application with these pictures. We have created an in-house database of the pictures we have collected.

We have taken only pictures of those cars which have government provided digital number plates. These number plates are of similar pattern white colored plates where the numbers are written in black font. The cars are from different BRTA. We have Dhaka Metro, Khulna Metro, Chattogram Metro etc. We have only taken the picture of the front side of each car. Every pictures were taken from the same angle and the framing was also similar.

3.2 Workflow of the proposed model

In our workflow we have various steps to follow. But the app really initiates with a car entering through the sensors and follow these major steps :

1. Sensor detects the car.
2. Raspberry Pi receives signal from sensor.
3. Raspberry Pi takes a picture by camera.
4. Raspberry Pi sends the picture to server.
5. We run image processing and get the license plate number.
6. Server generates field for new entry.
7. Drivers mobile app gives data to server.
8. Driver receives parking spot from server.
9. Exit.

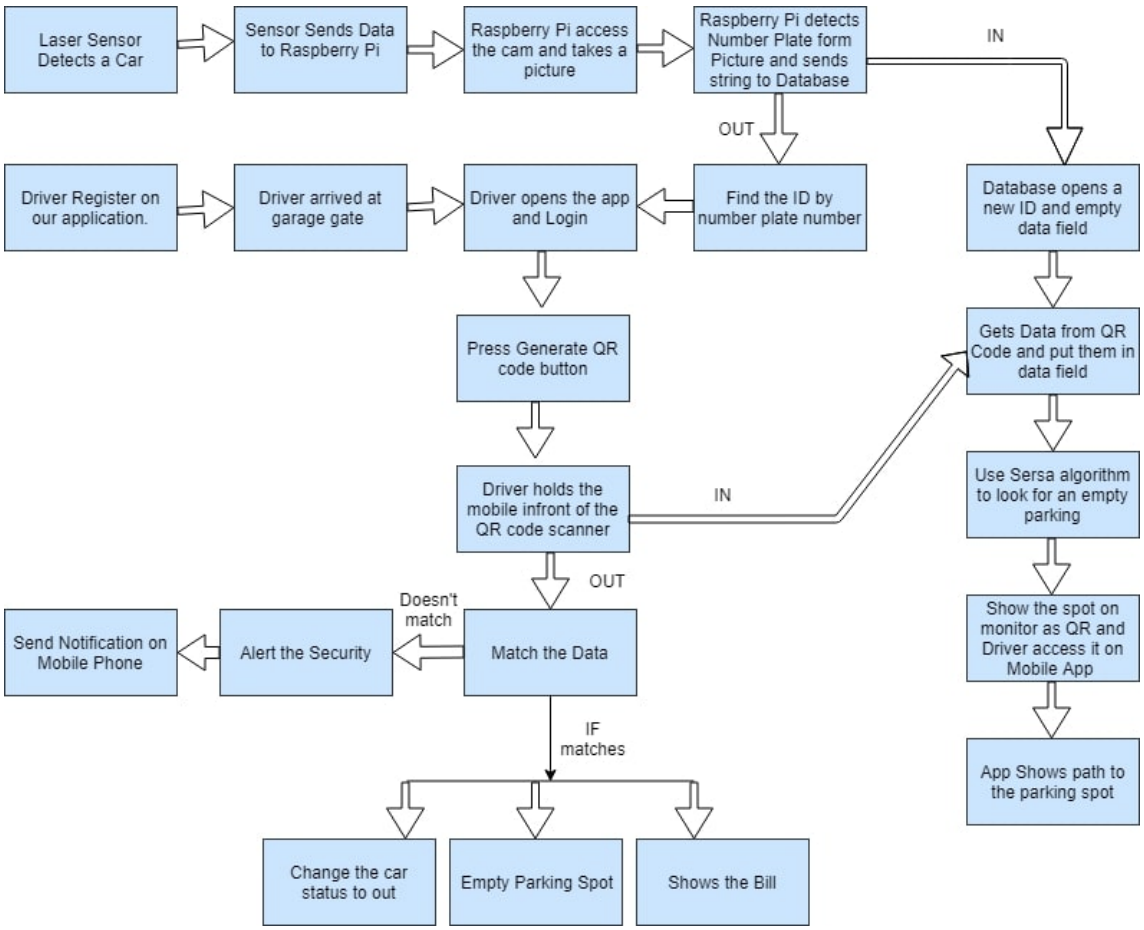


Figure 3.1: Block diagram of the workflow

3.2.1 Sensor detects the car

The first step is when our laser sensor detects a car. Our sensor is built and placed in a way so that only when a car comes our sensor's logic turn to true. Fig 3.2 shows how the sensor works.

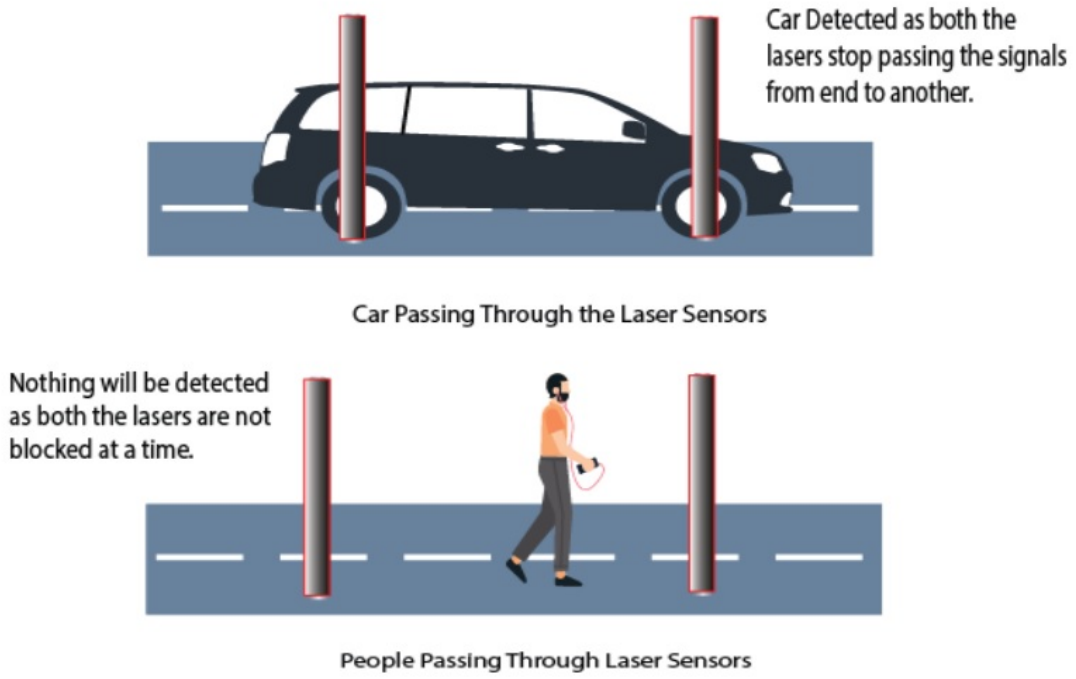


Figure 3.2: Laser sensor detecting car

3.2.2 Raspberry Pi receives signal from sensor

When our sensor's logic gets true, the sensor sends a signal to our Raspberry Pi by a Bluetooth module which is connected to our sensor.

3.2.3 Raspberry Pi takes a picture by camera

The When Raspberry Pi gets the signal from the Bluetooth it accesses the camera which is attested to it and takes a picture. And our camera is placed in a certain way so it takes the full front side picture of the car.

3.2.4 Raspberry Pi sends the string to the server

After taking the picture Raspberry Pi detects the car's number from the number plate and send the string of letters and numbers to the main server. Server saves

the string by a unique id. Fig 3.3 is a flowchart of Raspberry Pi sending string to the server.

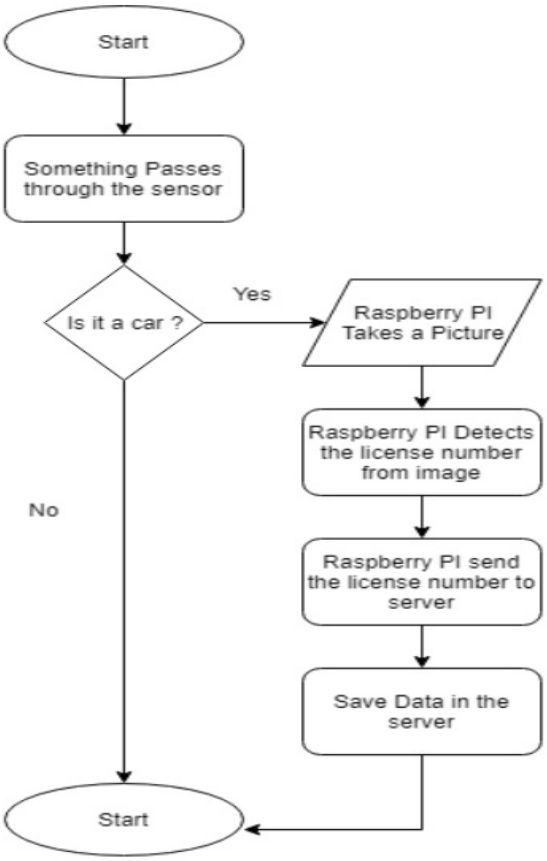


Figure 3.3: Flowchart of the sensor

3.2.5 Image processing run and obtaining the license plate number

Now the picture goes through a process built with Python and in the end of the process we get a string which is the license number. Fig 3.4 is the flowchart of image processing.

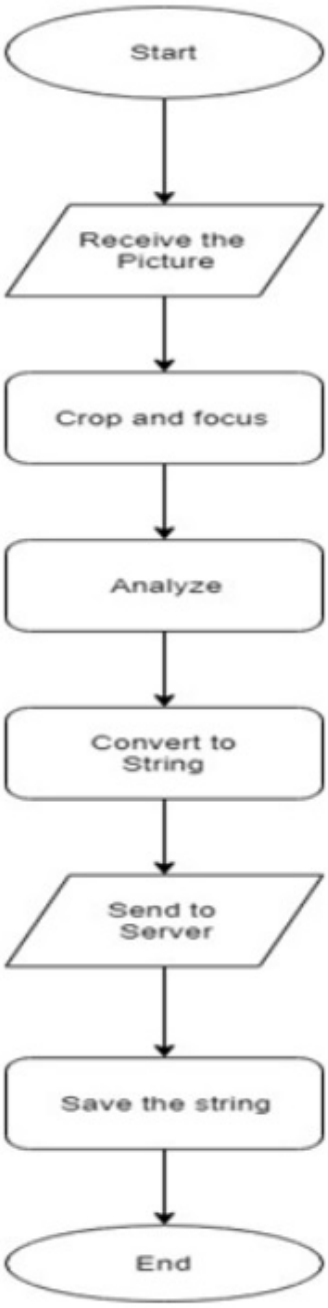


Figure 3.4: Flowchart of image processing

3.2.6 Server generates field for new entry

While the image processing is running the server generates number of fields with a unique id and when the image process ends it saves the string to its field.

3.2.7 Drivers mobile app gives data to server

In this part, Drivers mobile app gives data to server. As we are using offline service, so now driver opens our app and press the generate QR code button and an QR gets generated by the app which contain driver's dataset what we took while registering for the app. And the QR code scanner reads the QR code and takes the data from the driver and saves it to database fields. Fig 3.5 is the flowchart of our mobile application.

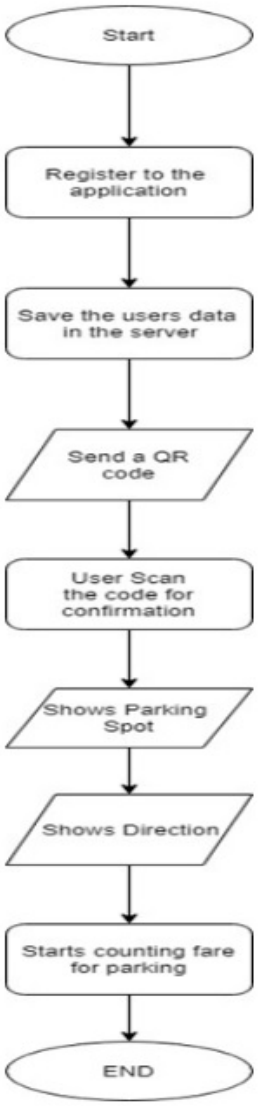


Figure 3.5: Flowchart of the mobile application

3.2.8 Driver receives parking spot from server

Vertical The moment database gets and saves the license number it starts a search algorithm and finds an empty parking spot. And when it gets and saves all the data from driver it shows an QR code to its monitor. Our app scans the QR code and a path and the parking spot number comes to his screen.

3.2.9 Exit

Now when the car is going to exit the garage, he will go to garage exit door and he will go through the same procedure from 3.1 to 3.5 but after 3.5 he will do 3.7 and this time after getting the data from the driver we will cross match the data with existing data, if it matches he can exit or if not match guards will be alerted. To enter our garage first, the driver needs to register to our app. In time of the registration our app will take data about the driver and the car he is driving. A single driver can have multiple car registered. The server will be run by our software; it will have many functions. Guards can login and see the information of all the cars present at that moment, but if the admin logs in he will get access to the information of all the cars ever entered the garage.

Chapter 4

Implementation

4.1 A brief walkthrough

The system that we are trying to develop is in itself a considerable task at hand. There is a series of tasks involved in the whole system, each of which needs to be in perfect motion for the others to follow and work properly. There are fail gaps here and there and contingency plans to deal with them as well. We have gone through many earlier research papers related, but not limited to, this concept and came up with the methodology to build upon. Our work can be divided into 4 major divisions, each being a prerequisite for the following operations to function properly.

1. We have the sensors that start of the system
2. Image processing to analyse and operate on the licence plates
3. The software in the server ends that deal with a variety of information
4. And finally the app on then user ends for his interaction with the whole system.

Initially the laser sensor detects whether an approaching object is essentially a vehicle for which the system is designed. If it is indeed a vehicle, the camera takes a picture of the front of the vehicle mainly prioritising the license number. This picture is sent to the server database, where simultaneously when the camera was taking a picture, the database starts creating a data table with the required fields. It creates a new entry for the previously taken license plate number with a unique id generated for it along with other necessary fields which are currently empty. The user, at the time of his registration in the mobile app, will be prompted for necessary valid information like his username along with his license plate number, tax token, etc. These information will be stored in the mobile app database and will be filled in the empty spaces in the data table when the user generates a QR code in his app and scans his code in the QR reader. A new query will run in the system software and an appropriate parking spot for the user vehicle will be selected. At the time of departure, when certain conditions are met, the user will be given a receipt of his fare according to the time he used the parking spot. The user will pay the bill manually and the parking spot will be labelled empty again.

4.2 Sensors

To begin we placed two laser lights at an appropriate distance, keeping the average length of a typical automobile such as the toyota corolla in mind. The purpose of this sensor is to identify whether the object that passes through these lights are indeed vehicles or not. When the first light is hindered by an approaching vehicle, it waits for the next light to be restrained by the vehicle simultaneously in a few moments as well. If both these lights are hindered then the approaching object is indeed a vehicle and the system passes to its next stage as the bluetooth module signals the Raspberry Pi of the imminent vehicle. The Raspberry Pi corresponds by signalling the digital camera to take a picture of the front of the vehicle with the license plate as the primary focus. The picture is sent to the data servers in the server end for further processing.

4.2.1 Hardware components

Laser Diode

It is a component used to illuminate the LDR in our system. We used the Laser Diode 650nm 3v 5mw, whose dimension is: 6.5mm (diameter) x 10.5 mm, and as the name suggests, mean wavelength is 650nm and power rating is 5mw. It can work continuously according to our need, and continuously lights up the LDR. The light falls as a red point on the LDR and works between temperature of 10-40c. We use 2 of these in our system and these 2 work alongside our 2 LDRS and 2 signal lights. Fig 4.1 shows a laser diode.

Figure 4.1: Laser diode

In the circuit board, we initially connect the Arduino and the bluetooth module. When we are delivering power to the board we use a typical 12V adapter to provide current in the circuit. But since both the arduino and the module operates properly with a voltage around 5V, we use a MOSFET, to convert the incoming 12V to the required 5V. 2 lasers and 2 LDRs are then connected with arduino via the board. 2

extra lights are also connected which acts as a signal control, to let us know whether the lasers and the LDRs are functioning properly and according to our need, each light for a corresponding laser and LDR. The LDRs and laser are connected in such a way that the light directly falls on the resistor. When this is the scenario no data is being generated in the microcontroller. When one of the lights is hindered by an incoming object, temporary data A is generated in the microcontroller, and data B is generated once the other light is hindered by the incoming object. If data A is generated and then the data is lost due to the object not hindering the light anymore, and then data B is generated after a few moments, then the object is not a vehicle and no data is forwarded for further execution. If and only if data A is generated first and then, without this data being lost, data B is generated as well, then we can confirm the incoming object is indeed a vehicle and further execution of data is necessary so it is forwarded to the raspberry Pi through the bluetooth module. If on the other hand, the similar happens but in a reverse order, that is, data B is generated before data A, then the object is leaving the parking system.

Light Dependent Resistor (LDR)

An LDR is a component that has a (variable) resistance that changes depending on the amount of light falling on its surface. Fig 4.2 and 4.3 shows a typical LDR and the LDR circuit symbol respectively.

Figure 4.2: LDR

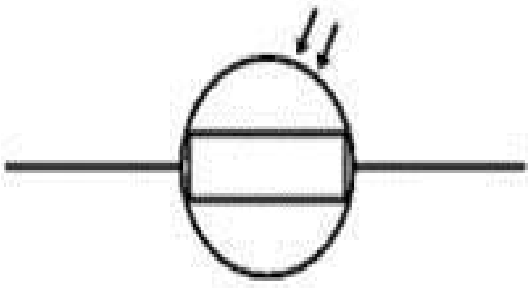


Figure 4.3: LDR circuit symbol

These resistors are often used in many circuits where it is required to sense the presence of light. LDR generally has a resistance that falls with an increase in the light intensity falling upon the device and rises with a decrease in light sensitivity [10] as can be seen in figure 4.4.

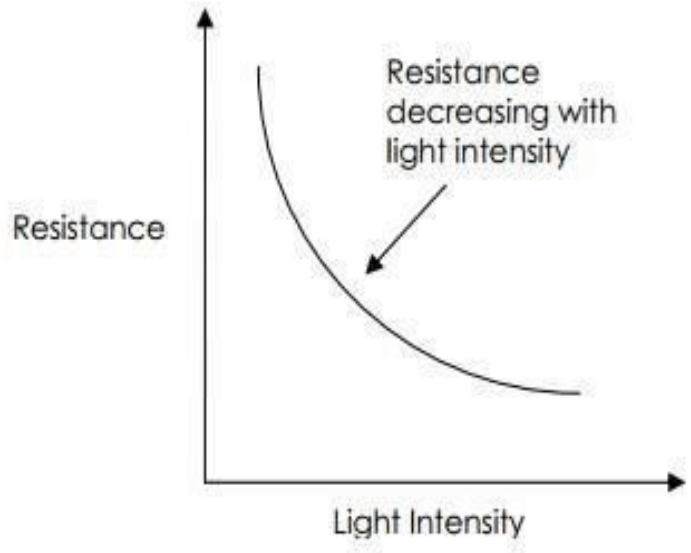


Figure 4.4: LDR resistance vs light intensity graph

we use 2 LDR's in our system, correspondent with 2 lasers and 2 lights which act as a signal control. The LDR and the laser work hand in hand. The LDR sends a specific value to the microcontroller when it stops receiving light from its corresponding laser, where the microcontroller generates data A and B respectively for the 2 LDRs and lasers.

Arduino pro mini

Arduino.cc developed the microcontroller board Arduino Pro Mini that is fundamentally based on the ATmega328 microcontroller incorporated inside the board. It comes with 14 digital input/output pins, of which 6 can be used as PWM outputs, 6 analog inputs, an on-board resonator, a reset button, and holes for mounting pin headers. In short it has everything we need to support the microcontroller. Since there is no USB port available on the board we can connect a six pin header to an FTDI cable or a USB to serial adapter to provide USB power and communication to the board. It comes with a flash memory of 32KB out of which 2KB is used by bootloader. The flash memory is used for storing the code of the board. The Arduino Pro Mini is intended for semi-permanent installation in objects or exhibitions. The Arduino Pro Mini is an open-source hardware and comes without pre-mounted headers, allowing the use of various types of connectors or direct soldering of wires. The dimensions of the Pro Mini PCB are approximately 0.7" x 1.3". We are using the version that runs at 5V and 16MHz. The arduino that we used in shown in fig 4.5.

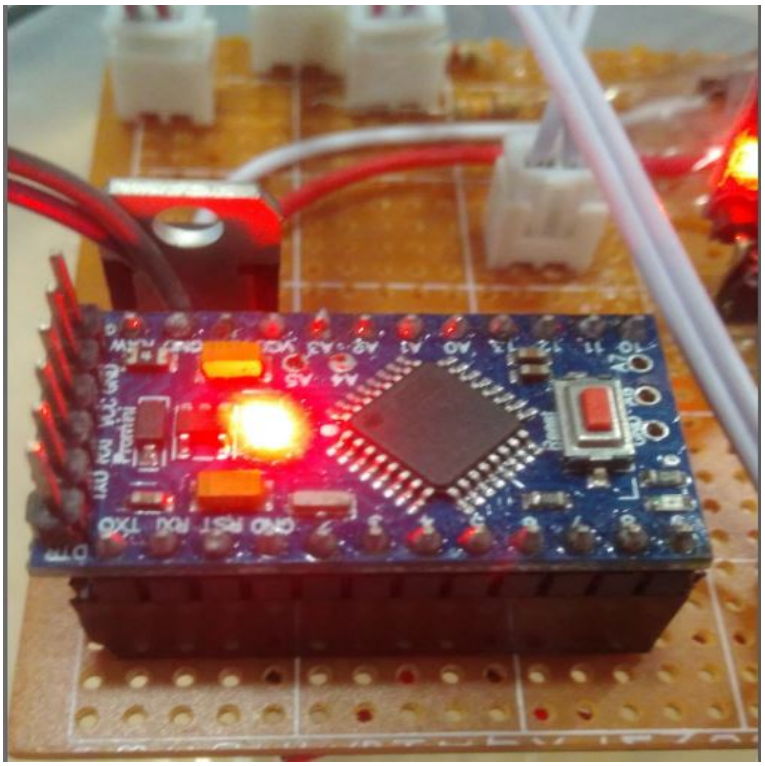


Figure 4.5: Arduino pro mini

In our system Arduino Pro mini plays a vital role. When the light falling on the LDRs are hindered by an incoming object, it is the microcontroller in the Arduino that runs its native code to generate data A and B. If light is continuously non-interrupted by anything the arduino stays dormant. If indeed a vehicle is recognised to be entering, the microcontroller sends these data to the raspberry Pi, where further execution takes place.

Bluetooth HC-06

HC-06 is a Bluetooth module which is specifically designed to establish short range wireless data transmission between two microcontrollers or systems. The Bluetooth module HC-06 has 4 pins, 2 for power and 2 to establish connection. It adopts the Bluetooth 2.0+EDR standard and works on Bluetooth 2.0 communication protocol.[12] In Bluetooth 2.0, the workload of bluetooth chip is decreased significantly since the transmit time of signals from different devices stands at a 0.5 seconds interval. The data transfer speed of this module can reach approximately upto 2Mb/s. HC-06 uses frequency hopping spread spectrum technique (FHSS) to avoid interference with other devices. The frequency range of the device is around 2.4 GHz. This is cheaper than other methods of wireless data transfer such as wifi or radio transmitters and provides much more flexibility. The module consumes very low power to operate and is very user friendly. Fig 4.6 shows a 2D model of a general bluetooth model. Fig 4.7 shows our bluetooth module HC-06.

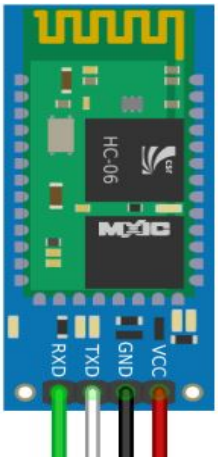


Figure 4.6: 2D model of bluetooth module HC-06

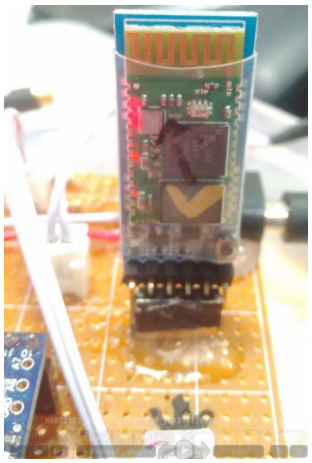


Figure 4.7: Bluetooth module HC-06

The bluetooth module works as the communication medium between our circuit board, particularly the arduino pro and the raspberry Pi. When a vehicle is indeed recognised by the sensor to be entering with the help of the lasers and LDRs, arduino generates data A and B and these data are forwarded to the raspberry Pi via the bluetooth module. We could have used wifi or radio communication device here for the data transfer, but we opted for the cheaper, feasible and the optimum method of bluetooth transmission.

Mosfet

The Metal Oxide Semiconductor Field Effect Transistor or more commonly known as MOSFET is a voltage controlled field effect transistor. It works by the controlled oxidation of a semiconductor, mainly silicon [8]. It has an insulated gate, whose voltage determines the conductivity of the device. The main advantage is that it requires almost no input current to control the load current, when compared with bipolar transistors. We used the Mosfet to convert the incoming 12V to 5V power which is essential for the arduino and the bluetooth module to function properly. Fig 4.8 shows the Mosfet we used.

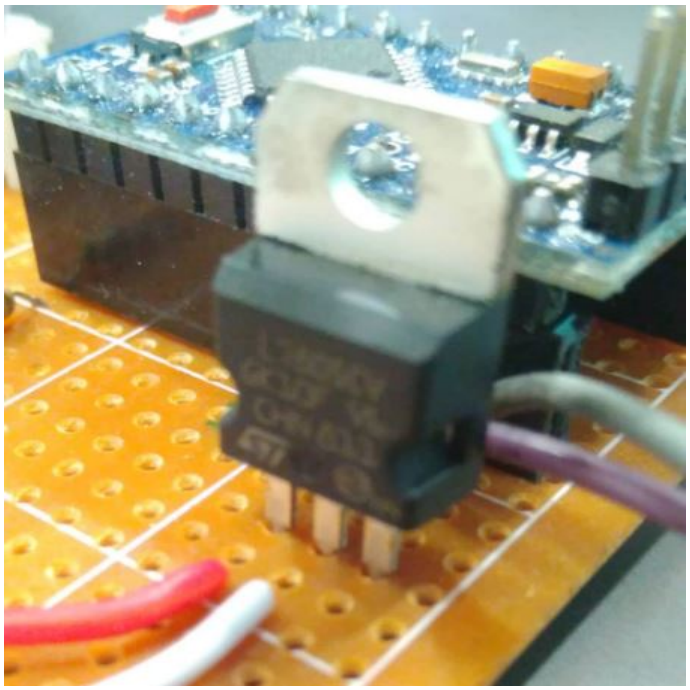


Figure 4.8: Mosfet

4.3 Image Processing

Image processing deals with the captured picture of the front of the vehicle, mainly the license plate. The picture is detected, taken, analysed and cropped to focus on the license plate number, which is accepted as a string value and sent to the server database. There the string value is further operated on. This whole procedure is done using python language in its OpenCV library. The whole process takes place at the time of departure as well, when the license plate is read again to ensure that the data matches with the existing information in the databases.

4.3.1 Raspberry Pi 3

The Raspberry Pi is a small computing system that is used in electronics projects, spreadsheets, word-processing and several games. There are currently four Raspberry Pi models. But for our project we have used Raspberry Pi 3B. Our Raspberry PI 3 - Model B packs the powerful 1.2GHz Quad-Core ARM Cortex-A53, 64 bit CPU and contains 1GB ram. The GPU is 400MHz VideoCore IV. Its wireless connectivity comprises of 802.11n wireless LAN (WiFi) and Bluetooth 4.1. It has 4 x USB 2.0 ports. We used the Pi 3 Model B instead of the 2 Model B, since the latter has a weaker 900MHz Quad core ARM Cortex-A7, 32Bit CPU and most importantly has no builtin wireless connectivity option.

Firstly the sensors will detect a vehicle and it will automatically send a signal to the raspberry Pi, the Pi will then access the camera to acquire the image of the oncoming vehicle. After the image is acquired by digital camera, the image processing will start. The captured input image will be on RGB format. The first step of pre-processing is to convert the RGB image into gray-scale image in order to reduce the

number of colors. In the figure 4.9 we can see the input image captured in the RGB format.

Figure 4.9: Input image on RGB format

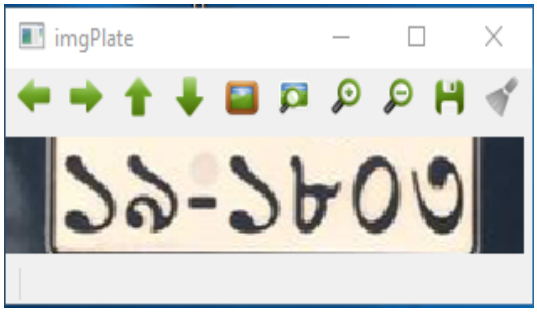


Figure 4.10: Detection of the number plate

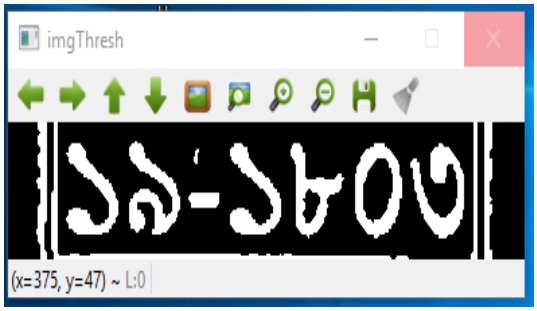


Figure 4.11: Conversion from RGB to grey-scale

The R, G and B components are separated from 24-bit color value of each pixel (i, j) and 8-bit gray value is calculated. After that image thresholding is done. It is a non-linear operation that converts a gray-scale image into a binary image where the two levels are assigned to pixels that are below or above the specified threshold value [14]. In order to automatically detect the number plate of the vehicle, we will be usings contours to do that. A contour is a closed curve of points or line segments, representing the boundaries of an object in an image. We find contours with the findContours() function, and then examine the results. We chose the contours based on height, width, aspect ratio, pixel area and many other criteria.

These selected contours are then put on a specific set in order to make vectors. Then a rectangular bounding box will be created around every individual vector. We will re-run the whole image processing procedures to eliminate few of the vectors that do not match certain criteria.

After the elimination we will be selecting the largest vector box that will be the number plate of the vehicle. In the figure 4.10 and 4.11 we can see the detection of the number plate and the conversion from RGB into grey-scale. After our python code finds out the number plater we take that vector and find all the contours in that vector. Then we draw a boundary around each contour and run them through our own Bangla character recognition algorithm which is based on KNN algorithm. From there we can get the string of the number plate.

4.4 Software and Server

A number of software operates in our proposed system. Primarily the back end of the software system is developed using Javafx. We used the language to navigate how the system software would function. The GUI of this software is designed using Scene Builder which is compatible with JavaFx ecosystem. The output was a FXML file which was integrated with the java project to build the UI. This allowed us to build a convenient and simple interface for the user. Fig shows the homepage of the software built using the scene builder. We created the server database and tables using mySQL based on SQL and we established the relation between the tables using mySQL workbench. We used Enhanced Entity Relationship (EER) model to build the relationship between the tables. Finally after the database was created, we chose BFS as the algorithm to find the shortest and accurate parking spot for the vehicle. Although BFS consumes a bit more memory compared to other searching algorithms, it still gives the optimum location and the shortest path in less time.
To understand how the system works we must follow the use of the software in a sequence. There will be 2 user categories of the software - admin and guard. The login page for the software can be seen in fig 4.12.

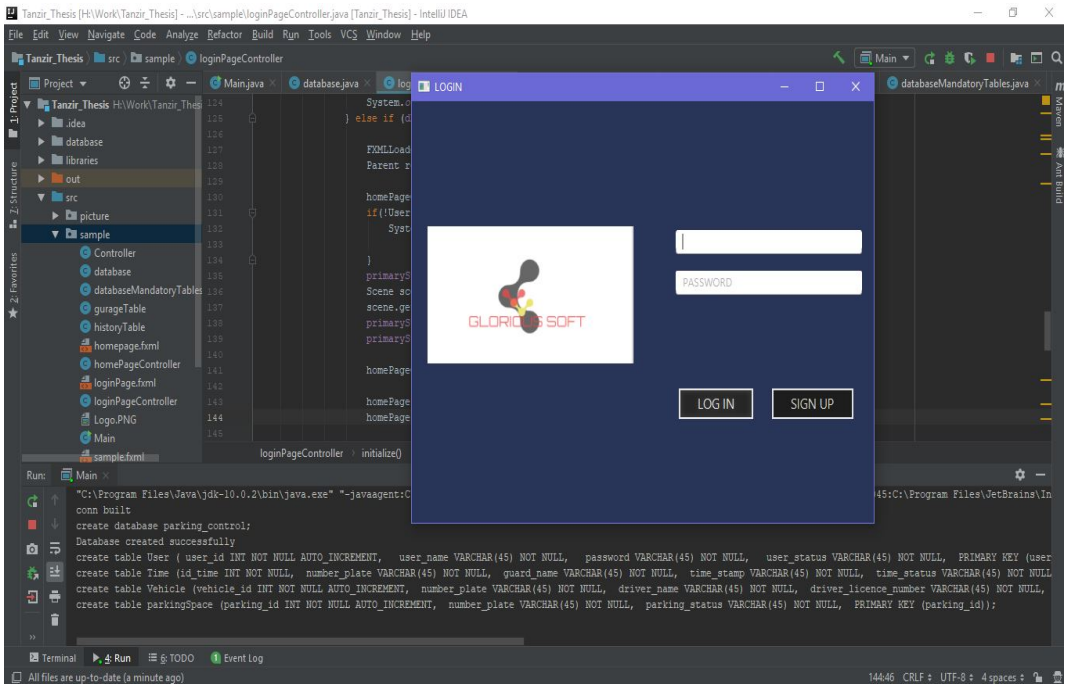


Figure 4.12: Login page for software

The admin will have access to the whole system software and will be able to access all the pages to see information like all car history, the driver's information, etc. The guard however will have limited access. They will only be able to have liveview, i.e, a real time view of the parking gate, and will be able to see in which spots are which vehicles parked, and where are the free spots available. The guards will be notified with any kind of alerts from the parking system. When a vehicle wants to enter the parking spot, the server end creates the data tables parking space, time and vehicle. These tables will have fields consisting of information like number plate which is acquired using image processing and other information like name, username, etc acquired from the user app database, in which the user has already stored his valuable information like his name, username, tax token, etc.There are some additional fields generated which are filled with information when the car is given a parking spot, like time status and number of entries as can be seen in the time and vehicle table. The server database gets the driver information when the driver generates a QR code in his app and scans his code in the QR reader. The server end is also provided with the string value obtained from the captured image of the license plate number. The server database creates an auto-incremented unique vehicle id for the user vehicle. This id is sorted with the driver information in a single entry in the vehicle table along with car status and number of entry. The system then finds the nearest free parking spot for the user using BFS algorithm. This parking id along with the number plate is stored in the parking space table. The parking status is switched from empty to full. The table time has a new entry now where the time stamp records the time the vehicle entered along with its parking id. The system will generate a new QR code and show it in the monitor which the user will scan to see the direction to his parking spot in his app. Figures 4.13 and 4.14 show the server databases at 2 particular point in time.

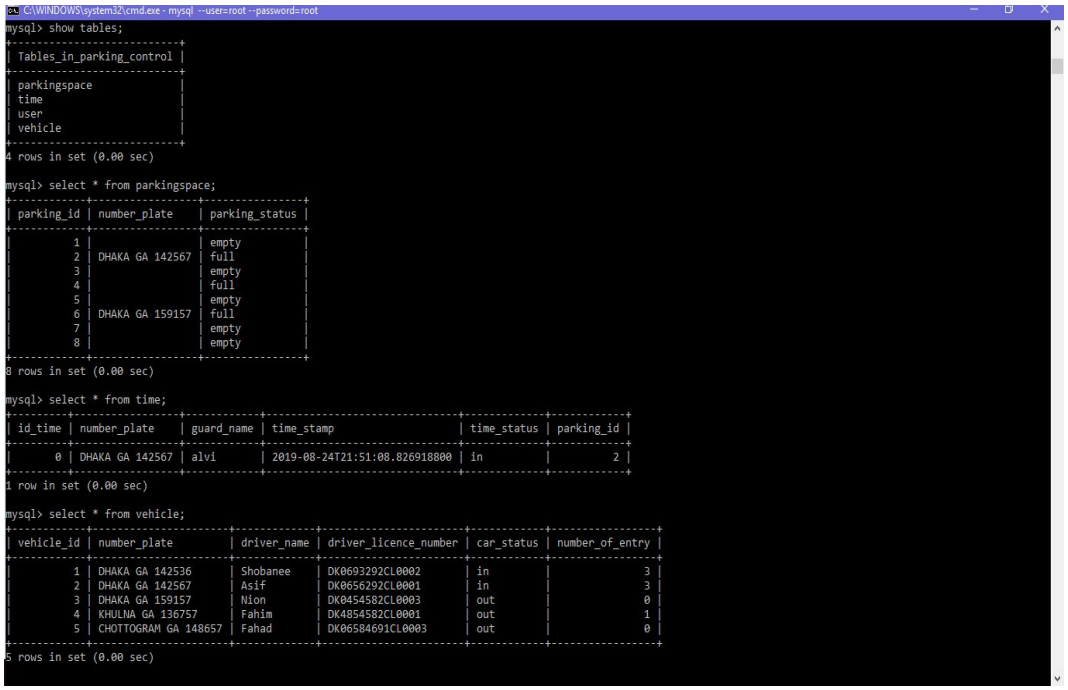


Figure 4.13: Database sample 1

When leaving the camera again will take a picture of the front of the vehicle and send it to the server where the license plate number will be cross matched with the existing entries in the database. The user will be given a given a new QR code and upon scanning, his information from the app database will be cross matched with the information in the server database. If everything matches, the user will be charged for the duration of time he occupied the parking spot. The parking spot will be emptied. If the data does not match, the guards will be alerted and notification will be sent to the user of the vehicle.

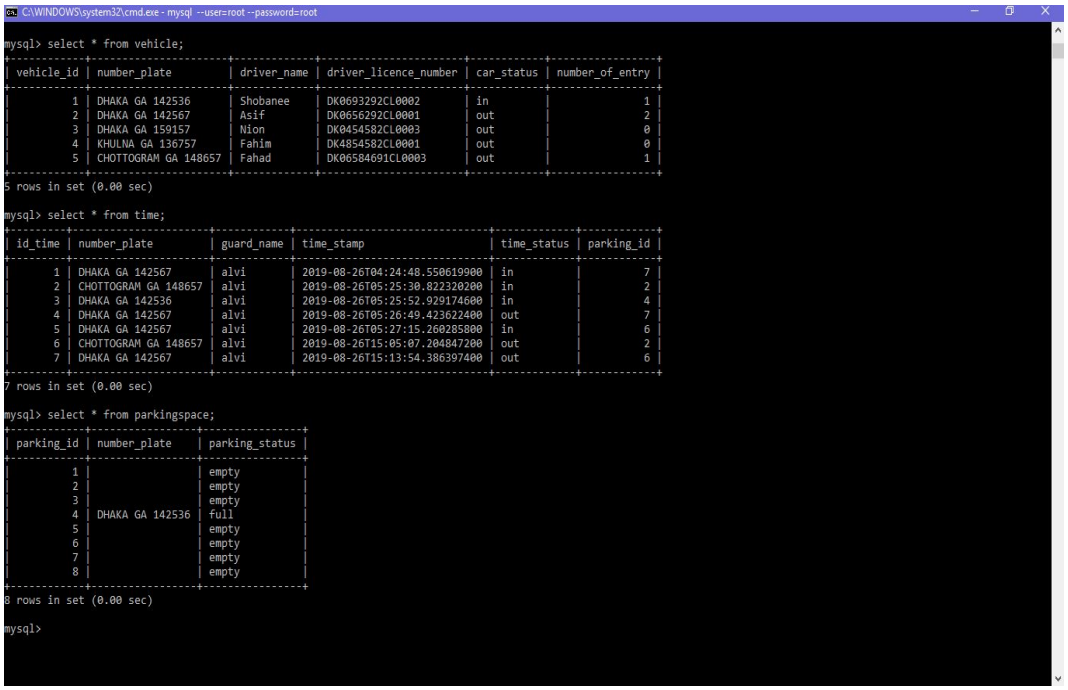


Figure 4.14: Database sample 2

In our system, we had to opt for manual reservation of parking spots, due to the limitation of not being able to link the data in real time. Fig 4.15 shows the interface of the software being used along with the pop up "reserve a spot" tab, which displays all the possible options available to us currently.

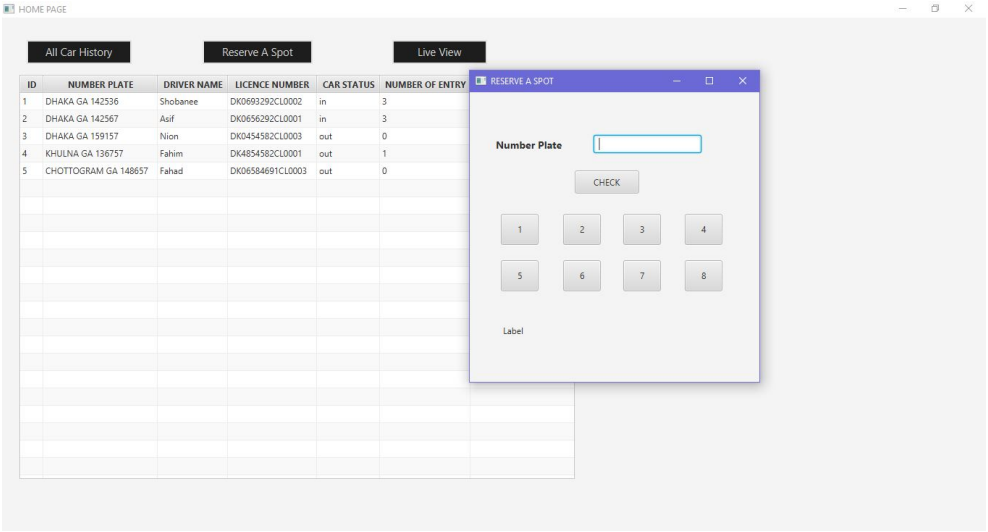


Figure 4.15: Interface of software homepage

Clicking on reserve a spot option opens up the tab, where we can check whether a car is already inside the parking system, by clicking on the specific number plate in the car history page and clicking check as shown in fig 4.16.

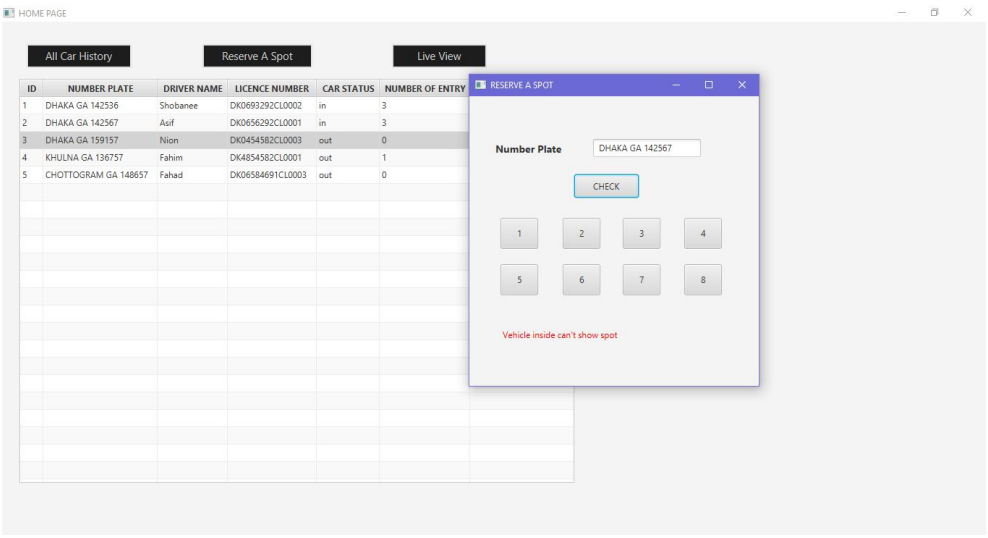


Figure 4.16: Reserve a spot checking parking spot

This tab also shows us which parking spots are currently in use and which ones are empty, and we can allot a specific parking spot manually for a vehicle that is not inside the parking system already like we did in the fig 4.17.

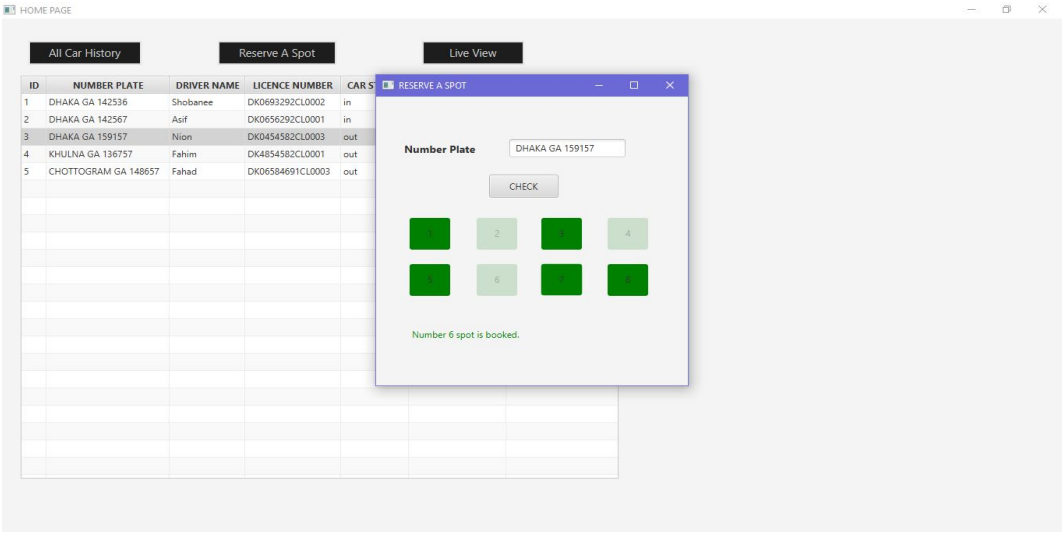


Figure 4.17: Parking spot booked for a vehicle

We can then prompt the live view option to see the current condition of the parking system. In addition, when we want to free a particular spot when a car empties it, we added a new option release a spot to do that. This tab initially shows which spots are empty and which are filled. We can search a specific car to find its parking spot by inputting the car's number plate number. When the car leaves, we can click on the release option to empty the parking spot.

4.4.1 Software Components/elements

Javafx

Javafx is a digital library used to develop Desktop applications and rich Internet applications (RIA). One can develop applications using Javafx which can run on multiple diverse platforms like desktops and mobile. Javafx application code, written as a Java API, can reference APIs from any Java library. For example, Javafx applications can use Java API libraries to access native system capabilities and connect to server-based middleware applications [4].

Some Key Features of JavaFX

Java APIs: JavaFX is a Java library that consists of classes and interfaces that are written in native Java language [13].

Built-in UI controls and CSS: JavaFX provides all the major UI controls required to develop a full-featured application independent of the operating system. Javafx code can be embedded with CSS to provide a custom interface of the application.

Canvas API: It enables us to draw right in a JavaFX scene area. Integrated Graphics Library: An integrated set of classes are provided to deal with 2D and 3D graphics [13].

Hardware-accelerated graphics pipeline: JavaFX graphics are based on Graphics rendered pipeline (prism) [13]. It offers smoother graphics if, for example, used with a supported graphics processing unit.

Self-contained application deployment model: The application packages contained within JavaFX have all of the application resources and also include a private copy of Java and JavaFX Runtime.

FXML file by Scene Builder

FXML is an XML-based user interface markup language created to define the user interface of a JavaFX application. It provides a convenient alternative to constructing such graphs in standard code. Moreover scene builder, which is a visual layout tool for JavaFX language, lets us swiftly design JavaFX application user interfaces, without actually writing the code for the interface. When using scene builder, users can drag and drop UI components to a work space, modify their features, apply style sheets on the components, and all these generate the FXML code for the layout automatically in the background. This results in an FXML file which can then be combined with a Java project by binding the UI to the application's logic [15].

MySQL Database

MySQL is an open source Relational Database Management System (RDBMS) that uses Structured Query Language (SQL). SQL is the most popular language for adding, accessing and managing content in a database. MySQL is sought-after for its swift processing, dependability, feasibility and ease of use. It is most commonly used to develop a web database. It can be used to store anything from a single piece of information to a vast array of information in the database. MySQL is internationally famous for being the most safe and dependable database management system and it also has been an integral part in giant web applications like Facebook, WordPress and Twitter.

MySQL Workbench

MySQL Workbench is a unified visual database design tool that integrates SQL development, administration, database design, creation and maintenance into a single integrated development environment for the MySQL database system. It delivers visual tools for creating, executing, and optimizing SQL queries. It provides a visual console to easily administer MySQL environments and gain better visibility into databases. We used Enhanced entity–relationship (EER) model to establish the relationship between various entries in the mySQL data tables. EER model is an advanced model of diagrams which represents requirements and complexities of complex databases and is used in the design of these databases.[9] We designed EER diagram to illustrate the relationship between the entries and mySQL workbench uses this diagram and backtracks these relationships and generates the code automatically to illuminate the relationship in the database.

BFS for empty parking spot search

Breadth-first search (BFS) is an algorithm for traversing or searching tree or graph data structures. In BFS, one starts traversing from a selected node or point (source node) and traverses the graph in the current depth and explores the adjacent nodes directly linked to the source node and then further traverse towards the next depth level adjacent nodes. It keeps this traversal up until it explores all the unexplored nodes and reaches its goal [11] which for us is the shortest and the optimum path.

4.5 Mobile Application

We introduced the mobile application for a more user friendly and convenient experience. At the time of registration, the app will initially prompt the user for information like name, username, password, license number, vehicle registration number, tax tokens, etc. The app database will store all these information, which when needed will be used to fill the empty fields in the server database. The user can use QR codes in his phone to interact with the system. The app was developed on Android Studio and SQLite was used to create the app database. Fig 4.18 and 4.19 shows samples for registration and login page in the android app.

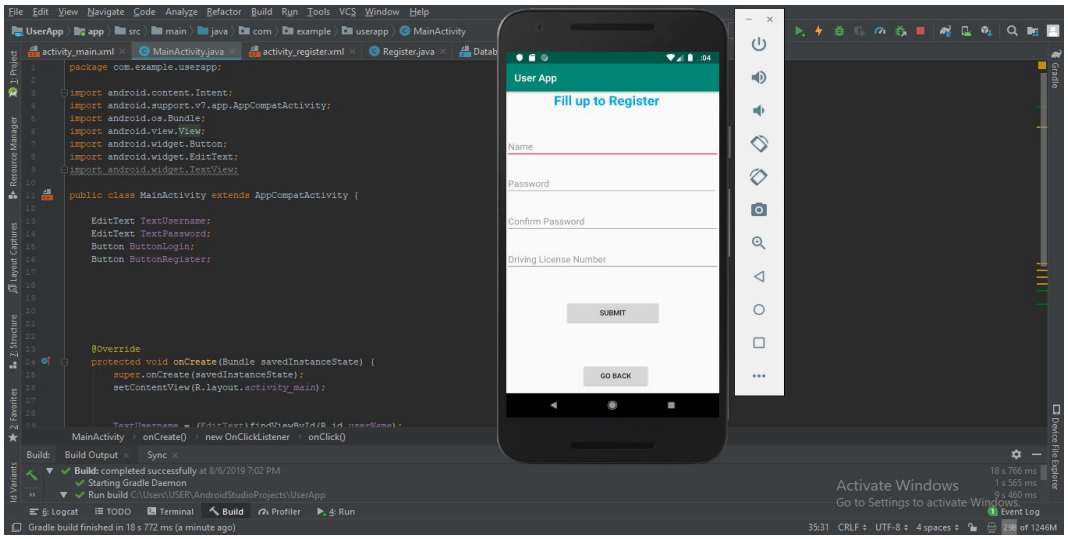


Figure 4.18: UI of app registration page

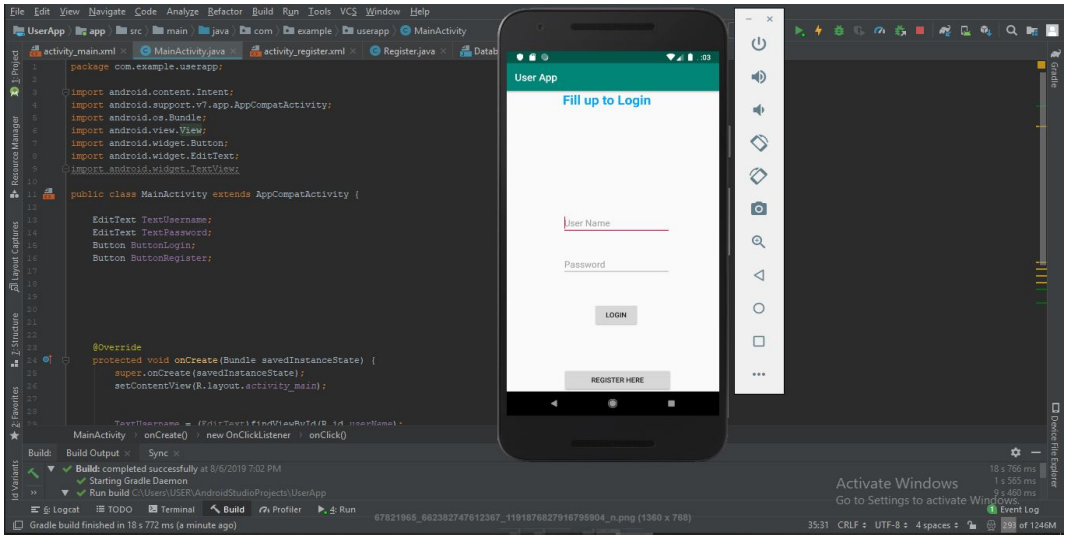


Figure 4.19: App login page UI

We used android studio to build our android application using the Java language. We could also use Kotlin but we preferred java over kotlin is because kotlin is comparatively a new language and there are less sources for kotlin compared to java. SQLite was used to create the database of our system instead of Firebase. We did not use firebase because firebase is an online based database system but since our plan is to build an offline application so we prefer not to use firebase.

Chapter 5

Experimental Tests and Results

5.1 Experimental test outputs

The initial step of the test result occurs when our laser sensor detects a car. We have used two LDRs and lasers as part of the vehicle sensor in our system, which are connected in such a way that the light directly falls on the resistor. The LDRs work by sending a high resistance value to the micro-controller when it stops receiving light from its corresponding laser for a certain period of time. We declared integer variables LDR1 and LDR2 and assigned them initial values of 0 when they are constantly receiving light from the sensor. We also declared the integer variables vehicleDetected, DataA and DataB and the string variable Vehicle. When one of the lights is hindered by an incoming object, its corresponding LDR value becomes 1, and temporary DataA is generated in the micro-controller, and DataB is generated once the other light is hindered by the incoming object and switches the value of the other LDR to 1 confirming the presence of a vehicle. vehicleDetected variable now shows 1 as well as the variables DataA and DataB and the string vehicle displays Yes.

Obstacle	LDR1	LDR2	vehicleDetected	DataA	DataB	Vehicle
1	0	0	0	0	0	No
2	0	1	0	0	1	No
3	1	0	0	1	0	No
4	1	1	1	1	1	Yes

Table 5.1: Sensor detection of the first vehicle

From Table 5.1, we see that when there is no obstacle on either of the LDR's, no data is generated. When there is an obstacle detected on LDR 2 only, no data has been made and hence no vehicle is detected. Again when there is an obstacle detected on LDR1 only, no data has been made and no vehicle is detected. But during the fourth attempt the obstacle block both LDR1 and LDR2, turning their

values to 1 sequentially causing DataA and DataB to be 1 as well the existence of a vehicle is confirmed. We check the arrival of 12 vehicles this way among which 10 were detected by the LDR system as seen in Table 5.2 .

Vehicle No	DataA	DataB	Vehicle
1	1	1	Yes
2	1	1	Yes
3	0	0	No
4	1	1	Yes
5	0	1	No
6	1	1	Yes
7	1	1	Yes
8	1	1	Yes
9	1	1	Yes
10	1	1	Yes
11	1	1	Yes
12	1	1	Yes

Table 5.2: Arrival of the 12 vehicles

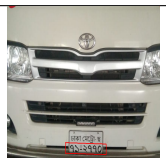
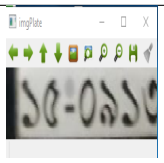
When the detection of the sensors are done and the vehicle is confirmed, a signal will automatically go to the raspberry pi. The raspberry pi will access the camera to acquire the image of the oncoming vehicle.

Attempt	DataA	DataB	raspberryPiSignal	cameraAccess	pictureCaptured
1	0	0	0	0	0
2	0	1	0	0	0
3	1	0	0	0	0
4	1	1	1	1	1

Table 5.3: Image capture of the first vehicle

Similar to the table of sensor detection, we see 4 possible attempts in Table 5.3 . We make use of the earlier DataA and DataB variables here along with some newly declared integer variables raspberryPiSignal, cameraAccess and pictureCaptured all default assigned to 0. During the first 3 attempts, either or both of the DataA and DataB values is found to be 0, so no signal is sent from raspberry PI and the camera is not be accessed and no picture is captured. But in the final attempt, DataA and DataB both are received by raspberry pi, making their values 1 and the raspberry pi

send a signal to the camera to take a picture, turning all the other variables values to 1 as well.

No	LDR1	LDR2	Original	Segmented	Detected	Complement
1	1	1				
2	1	1				
3	1	1			Could Not Detect	Could Not Detect
4	1	1				
5	0	1			Could Not Detect	Could Not Detect
6	1	1				
7	1	1				
8	1	1				
9	1	1				
10	1	1				

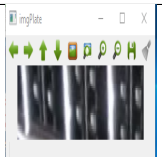
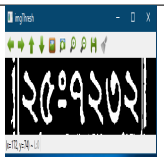
No	LDR1	LDR2	Original	Segmented	Detected	Complement
11	1	1				
12	1	1				

Table 5.4: License plate numbers processed into complement of binary images

Here we again make use of the earlier variables DataA and DataB variables to ensure whether a vehicle is incoming or not. We then input our test original images, which are segmented to focus on the license plate. We then have the detected number plates and finally these detected number plate images are converted to complement of the binary black and white images. The whole illustration is visible in Table 5.4 . In test numbers 3 and 5, we were unable to detect any number plate image and complements. And in case of test 11, we were able to detect the picture but this was not of the license plate since the segmented image focused on a different part of the vehicle. As a result we had an absurd complement in this case.

After the picture of the vehicle is taken by the camera and the vehicle wants to enter the parking spot, the server creates the data tables of the parking space, time and vehicle. The server database also creates a unique ID for the user vehicle. This ID is sorted according to the number of the entry. The parking ID along with the number plate is stored in the parking space table and the status of the parking space changes. This is how an entire parking cycle works.

Parking Space	In Time	Car Number	Name	UniqueID	Status	Number
1	-	-	-	-	Empty	-
2	-	-	-	-	Empty	-
3	-	-	-	-	Empty	-
4	-	-	-	-	Empty	-
5	-	-	-	-	Empty	-
6	7:10	15-5831	Ripon	0001	Full	0001
7	-	-	-	-	Empty	-
8	-	-	-	-	Empty	-
9	-	-	-	-	Empty	-

Table 5.5: Instance showing the first vehicle information in database

After the picture of the first vehicle is taken by the camera the server created the data on the parking space number 6 at time 7:10 a.m. and stored the vehicle number 15-5831 and parking number 0001 as shown in table 5.5 . The entire information is updated on the database chronologically after each of the vehicles enter the parking system and the parking status changes from empty to full and time table is updated. We see in table 5.6 , out of the arrival of 10 cars, 9 of them enter the parking system and are checked with a unique number and a parking number.

Parking Space	In Time	Car Number	Name	UniqueID	Status	Number
1	7:36	20-4176	Monjur	0003	Full	0003
2	8:55	71-2770	Afzal	0006	Full	0006
3	9:15	15-0913	Sharif	0007	Full	0007
4	8:15	37-6433	Rifat	0004	Full	0004
5	7:30	18-2158	Shams	0002	Full	0002
6	7:10	15-5831	Ripon	0001	Full	0001
7	9:23	19-2710	Noyon	0008	Full	0008
8	8:30	32-7945	Rahim	0005	Full	0005
9	9:20	25-7232	Jobbar	0009	Full	0009

Table 5.6: Instance showing all the vehicles' information in database

Vehicle List			
1	2	3	4
20-4176 Monjur 87549658623 7:36	71-2770 Afzal 84596357872 8:55	15-0913 Sharif 87889855463 9:15	37-6433 Rifat 8652324598 8:15
5	6	7	8
18-2158 Shams 87965478455 7:30	15-5831 Ripon 80978654322 7:10	19-2710 Noyon 87485968575 9:23	32-7945 Rahim 86427952462 8:30
9			
25-7232 Jobbar 82469696741 9:20			

Table 5.7: A tabulated illustration of the parking system

5.2 Accuracy and limitation

After experimenting with 12 different vehicles and trying to extract the license plate number from them, we only succeeded in 9 occasions. Since this was a short scale test result, we wanted to achieve 100% accuracy but that was not the case. The 3 times that we failed led us to dig further into the matter to see where we went wrong. With thorough observation we came to the conclusion that light intensity was the major reason behind the failure for 2 of the test samples.
Illuminance is an evaluation of how much luminous flux spans over a given area. It can be thought of as a measure of how much intensity of illumination is falling on a given surface, and luminous flux being an assessment of the total amount of visible light present. We decided on a working formula for illuminance to illustrate the success and failure rates of our test vehicles.

$$Ev(lx) = \phi v(lm)/A(m2) \tag{5.1}$$

where Ev is the symbol of illuminance measured in lux, phi(v) is the luminous flux measured in lumen, and A is the surface area of the object measured in m2. Illuminance is equal to the luminous flux divided by the surface area. Lux(lx) is the SI derived unit of illuminance and lumen(lm) is the SI derived unit of luminous flux.

We need to calculate the value of illuminance to ensure a fixed range of lux which is suitable for a picture to be taken that is easily recognised by our server to process the binary negative on the image. Fortunately most license plate numbers have a similar constant surface area, so the only changing variable that we need to measure is the luminous flux. Our future plan includes the use of a photometer to measure the luminous flux of the light source and calculate the illuminance of the source. We can use this value to determine a sustainable range of value of lux to illuminate the license plate number properly, thereby minimizing the failures in our system. We also found one other failure and the reason behind that as well. We did not take motorbikes into account when we build the sensors for our system, and thus when a motorbike tried to enter the parking system, our sensors failed to detect that as a vehicle and no further processing of the license plate number took place. We intend to remedy this as well by expanding our parking system for all sorts of vehicles.

Chapter 6

Conclusion

6.1 Concluding notion

Our Thesis basically aims to discuss the methods and problems related to the task of Multiple Object Tracking using image processing while including, but not limited to, the application of our methodology towards the remedy of an existing chaotic scenario in our own country and in turn reach a global audience. This is an extensive concept, with a believe that a lot of research scope is ready to be implored here that can be implemented in a vast array of technological fields, of which we are only touching the surface. Our goal is to present a unified problem formulation and several ways of categorization of existing methods, comparison between the existing methods with our own, and describing the key components within state-of-the-art tracking approaches, discussing the evaluations of appropriate algorithms including evaluation metrics, public datasets, open source implementations, and benchmark results. We are targeting to build a system that works subtly towards but to a substantial significance in the people's daily life, prominently increasing their productivity indirectly as well as work as a solution for a constant tension in the general people's life. Our idea will notably benefit fuel and time consumption but most importantly give rise to a secure yet convenient parking system compared to the existing manual units.

Our proposed plan will no doubt cost a higher initial pitching expense, but will cover the starting cost in the long run, since the maintenance cost along with other running expenses being minimal, making it a more future proof prospect with automation slowly but surely becoming a necessity in daily life. Automation engineering will be more encouraged and will open doors to further technological development in cohesion with digital image processing. Great progress has been made in the past decades, but there still remain several issues in the current researches and many open problems that need to be studied.

6.2 Future work

The whole idea has been a solid work in progress. This is an extensive concept and it was hard to realistically say how much of our whole idea we could have implemented in the long run in the short period of time that we could work on it. We have successfully reached a satisfactory point in our run with our effort. We built a running software that works hand in hand with our respective algorithms and databases to deploy a functioning automated parking system. In future we have plans to make the whole concept online, making it more user convenient and giving it a whole new reach to a vast population. Compared to now when the user uses a QR code scanner to input his informations which are cross checked with his already existing information in the local servers which are operated offline, we plan to introduce national servers which handle and update peoples' required information regularly, and build local servers in the parking slots which cross check the user information in real time, thus removing the hassle of the use of QR codes from user point of view. The user will also not need the QR scanners to receive his cost receipts digitally, which needs him to pay his tolls directly, rather he can simply use his app to digitally complete his payment through several online payment methods that we plan to introduce and merge with our current app. There will be no need of human operators in the parking spots which will reduce human errors to minimum. Another major prospect that we plan to work on in future is detecting the driver's face in real time through a digital camera along with the person in the front passenger seat, if there is any. The server database would store the images and at the time of exit, the vehicle will only be permitted to leave when the aforementioned driver or the front passenger were in the driving seat, whose faces would be cross matched again at the time of leaving. The app user will obviously have an override option in scenarios where he entrusts someone else with his vehicle. We wanted to utilize this plan in our current prototype, but taking into consideration the overall cost rise in our already stable plan, we thought of keeping it as a solid milestone down the road. There is no previous model for us to build our research upon, so we believe our implemented work can be considered a suitable model that can be worked upon in the future to provide even additional advancements to this system.

Bibliography

[1] J. Malik, S. Belongie, T. Leung, and J. Shi, "Contour and texture analysis for image segmentation", *International journal of computer vision*, vol. 43, no. 1, pp. 7–27, 2001.

[2] O. J. Tobias and R. Seara, "Image segmentation by histogram thresholding using fuzzy sets", *IEEE transactions on Image Processing*, vol. 11, no. 12, pp. 1457–1465, 2002.

[3] A. Pant, A. Arora, S. Kumar, and R. Arora, "Sophisticated image encryption using opencv", *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 2, no. 1, 2012.

[4] oracle.com, *Javafx 2.2.21*, https://docs.oracle.com/javafx/2/overview/jfxpub-overview.htm#A1131418, 2013.

[5] S. Viraktamath, M. Katti, A. Khatawkar, and P. Kulkarni, "Face detection and tracking using opencv", *The SIJ Transactions on Computer Networks & Communication Engineering (CNCE)*, vol. 1, no. 3, pp. 45–50, 2013.

[6] M. Nidhi, "Image processing and object detection", *Dept. of Computer Applications in National Institutes of Technology*, vol. 1, no. 9, pp. 396–399, 2015.

[7] opencv.org, *About opencv*, https://www.opencv.org/about/, 2017.

[8] (n.d.), *Introduction to mosfet*, https://www.theengineeringprojects.com/2018/02/introduction-to-mosfet.html, Feb. 2018.

[9] ——, *Enhanced er model*, https://www.geeksforgeeks.org/enhanced-er-model/, 2019.

[10] watelectronics.com, *Light dependent resistor (ldr) - working principle and its applications*, https://www.watelectronics.com/light-dependent-resistor-ldr-with-applications/, Feb. 2019.

[11] (n.d.), *Bfs algorithm - javatpoint*, https://www.javatpoint.com/breadth-first-search-algorithm/.

[12] ——, *Bluetooth transceiver module hc-06*, http://wiki.sunfounder.cc/index.php?title=Bluetooth_Transceiver_Module_HC-06.

[13] ——, *Javafx tutorial - javatpoint*, https://www.javatpoint.com/javafx-tutorial.

[14] ——, *Wavemetrics*, https://www.wavemetrics.com/products/igorpro/imageprocessing/thresholding/.

[15] G. (n.d.), *Gluonhq/scenebuilder*, https://github.com/gluonhq/scenebuilder/.