COLLEGE OF ENGINEERING, KIDANGOOR

(Under Co-operative Academy of Professional Education (CAPE), Estd. by the Govt. of Kerala)
Kidangoor South P.O, Kottayam - 686583

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

SECURE PARKING SYSTEM USING COMPUTER VISION

Project report submitted for the partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Electronics and Communication engineering by the Cochin University of Science and Technology

By:

ANANDHU BALAKRISHNAN
EMIL ZACHARIA GEORGE
JOICEMON JOSEPH
LEKSHMI VIJAYAKUMAR

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

MARCH 2015

COLLEGE OF ENGINEERING KIDANGOOR

(Under Co-operative Academy of Professional Education (CAPE), Estd. by the Govt. of Kerala)

KIDANGOOR SOUTH P.O, KOTTAYAM - 686583

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

MARCH 2015

CERTIFICATE

Date: .....

*Certified that this project work titled "**SECURE PARKING SYSTEM USING COMPUTER VISION**" is the bonafide record of the work done by **EMIL ZACHARIA GEORGE, ANANDHU BALAKRISHNAN, JOICEMON JOSEPH & LEKSHMI VIJAYAKUMAR** of **B.Tech. Eighth Semester, Electronics And Communication Engineering**, towards the partial fulfilment of the requirement for the award of the Degree of Bachelor of Technology, by the Cochin University of Science and Technology **(CUSAT)** , in the year 2015.*

Project Guide:

Asst.Prof. APARNA PS

Project Co-ordinator:

Asst. Prof. MUHAMED AMEEN

Head of the Department:

Asst. Prof. DEEPTHY MATHEW

ACKNOWLEDGEMENT

"Enthusiasm is the feet of all progresses, with it there is accomplishment and without it there are only slits alibis."

An attempt at any level cannot be satisfactorily completed without the support and guidance of learned people. We owe to many great people whose constant support and motivation that encouraged me to come up with this project.

First and foremost, we are grateful to ***God almighty***, for his divine grace bestowed on me, to select this topic and giving us the hunger and interest to presume this interesting topic. Without his blessing we would not be able to complete this project.

We are extremely thankful to ***Dr. Roobin D Varghese***, the respected principal of College of Engineering, Kidangoor. We are also thankful to ***Mrs. Deepthi Mathew***, Head of the department of Electronics and Communication Engineering, College of Engg. Kidangoor, for her cooperation and guidance for preparing and this project.

We thank wholeheartedly our project guides ***Mrs. Jismi Babu & Mrs. Shilpa Rajan,*** Asst. Professors in ECE dept. , College of Engg. Kidangoor, who acted as mentors, providing us their guidance and support rendered in time for the completion of our project.

We would also like to express our sincere gratitude to ***Mr. Muhamed Ameen, Ms. Aparna PS*** for their support and valuable suggestion throughout this venture.

We also extend our sincere thanks to ***all other faculty members*** of the department for their help and support.

Last but not least, we express our gratitude to ***our parents*** and ***our friends*** for their co-operation and advices for the wonderful completion of this project.

.

ABSTRACT

Parking lots have become quite common nowadays. Implementing security personnel in car lots require large labour costs & complicated manual tasks. In this paper, we propose a solution applying computer vision based automation. Cameras acts as digital eyes, the key input tools. Four cameras are altogether fitted in both at the inward & outward paths near to the gates. Two of them are for face recognition & the other two for number plate recognition. Four IR sensors are used to identify the presence of vehicle at different positions. Two PIR motion sensors are used to ensure that input images are captured in steady state. LED indicators & display screens provide the driver with the status of recognition procedures. Motors are used to operate the gates. Each incoming vehicle is assigned a serial number. The face image of the driver is taken & the Eigenface database is updated using this. The number plate recognized using template matching is stored in the number plate register. Vehicles parked are allowed to exit only after successful verification of face-numberplate match.

CONTENTS

LIST OF FIGURES

CHAPTER 1:
INTRODUCTION

Security is an important concern in this 21st century. Growing crime rate calls for developments in security systems. Parking lots are common in business centres, large office buildings, towns etc. Implementing security personnel require large labour costs and complicated manual tasks.

We propose a solution to this issue through our project titled "Secure Parking System". Cameras acts as digital eyes, the key input tools in here. The parking area consists of an inward path and an outward path. Two cameras each are fitted in the inward path as well as outward path near to the gates. Two of them are for face recognition and the other two are for number plate recognition. Four IR sensors are used to identify the presence of vehicle at different positions. Two PIR motion sensors are used to ensure that input images are captured in steady state. LED indicators are used to provide the driver with the status of recognition procedures. Motors are used to operate the gates.

Webcams are use in the project to capture images. The captured images are send to the MATLAB. The MATLAB Image Processing Toolbox is utilised to process the images and extract the required information. The MATLAB communicates with the microcontroller through UART serial communication. The microcontroller controls all the embedded devices. The working of the system is as follows:

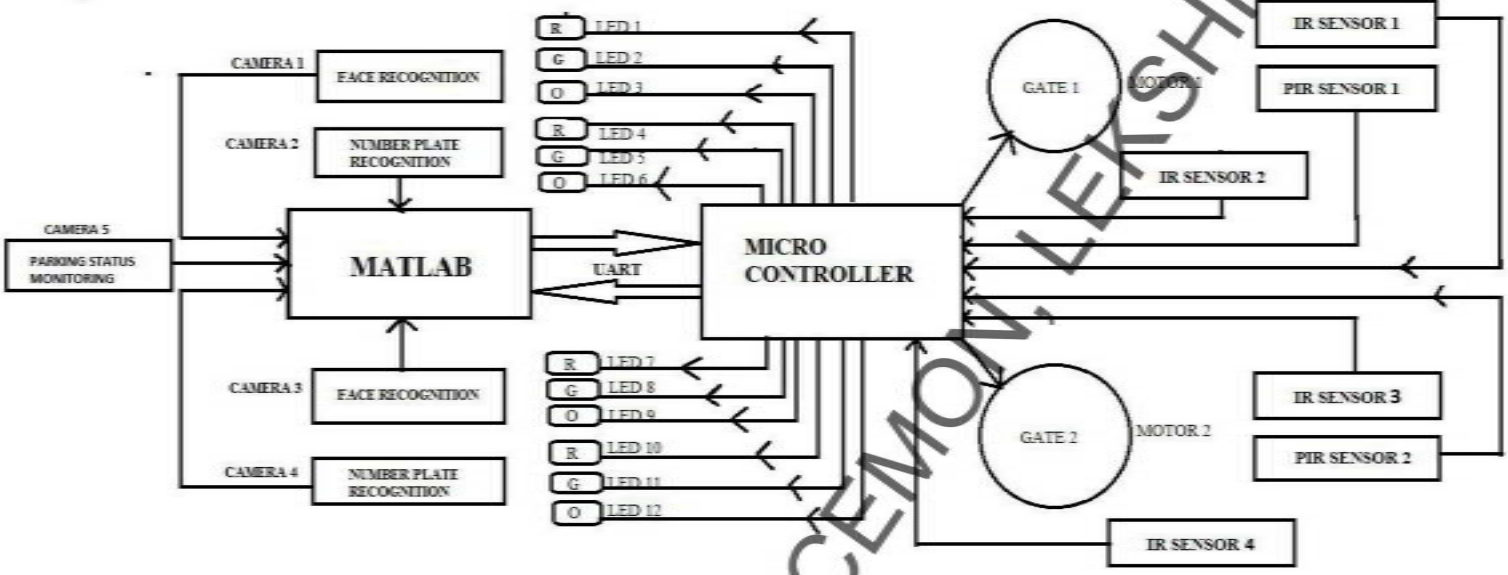
Each incoming vehicle is assigned a serial number. The face image of the driver is taken and the Eigenface database is updated. The weights for recognition are stored corresponding to the serial number in a face register. The number plate is recognised and the value is stored in the number plate register corresponding to the same serial number. Following the safe completion of both these, the gate is opened and the vehicle is allowed to move in.

The system also provides a display message stating the open park spaces available. The driver can park in any of those spaces.

As a departing vehicle approaches the outward gate, the face image of the driver is taken. The processor searches the face register for a match. If obtained, the serial number of that particular face is stored temporarily. Following this, the number plate value of the vehicle is identified. It is compared with the number stored corresponding to the serial number. If it matches, the gate opens and the vehicle can exit.

CHAPTER 2:

BLOCK DIAGRAM



BLOCK DIAGRAM DESCRIPTION

2.1) Microcontroller Development Board

The Rhino control board L293 is versatile and expandable platform for embedded circuits. Due to its expansion capabilities the board can be used to control all robots starting from beginner robot to advanced robots with multiple functionality.

The board is compatible to 6-25VDC input compared to all robot control boards available which accepts inputs just up to 12VDC. It has on board four 1Amp motor drivers (2X L293NE). Other features include:

- 2 Switches including reset
- Power on/off toggle switch 10A (Use better capacity switch or just short the terminals, if your power requirement is more than 10A)

- 16MHz crystal for maximum speed
- Power Indicator LED
- 4 DC/2 Stepper motor 5A driving capability (Normal configuration offers capability of 2 DC/1 Stepper motors)
- 8 ADC/Standard servo compatible connectors
- All Pins accessible through male header pins
- Can be expanded for I/O through expansion connector via SPI/I2C/UART.
- Reverse polarity protection using a diode (Short the diode to SMD pad provided on board to reduce voltage drop by 0.7V or increase current requirement if more than 6A. In this case reverse polarity protection will also be removed.)
- Can be programmed via USB through bootloader or through ISP programming.

Parts Identification

1. MICROCONTROLLER

Figure 1: Atmega 16A

ATmega16 is an 8-bit high performance microcontroller of Atmel's Mega AVR family with low power consumption. Atmega16 is based on enhanced RISC (Reduced Instruction Set Computing, Know more about RISC and CISC Architecture) architecture with 131 powerful instructions. Most of the instructions execute in one machine cycle. Atmega16 can work on a maximum frequency of 16MHz.

ATmega16 has 16 KB programmable flash memory, static RAM of 1 KB and EEPROM of 512 Bytes. The endurance cycle of flash memory and EEPROM is 10,000 and 100,000, respectively. It is a 40 pin microcontroller. There are 32 I/O (input/output) lines which are divided into four 8-bit ports designated as PORTA, PORTB, PORTC and PORTD. It also has various in-built peripherals like

USART, ADC, Analog Comparator, SPI, JTAG etc. Each I/O pin has an alternative task related to in-built peripherals.

2. LM317 3-Terminal Positive Adjustable Regulator

It is a three terminal 5V voltage regulator IC used to provide a constant voltage supply of 5V to the micro controller and other peripherals (for example, motor driver) attached in the main board.

3. L293DNE MOTOR DRIVER

This is a motor driver IC which takes the input from the microcontroller. It drives the DC and stepper motors using a separate power supply.

4. RST(Reset Switch)

The Reset switch is used to reset a program in execution, to the beginning. It is similar to the reset switch of a PC.

5. POWER(Power On Switch)

This is a toggle switch used to provide power supply to the main board. The supply can either be from a battery (through LS), or usb powered. The power switch can be toggled between MP (Main Power) and UP.

6. DS(Driver Supply)

DS consists of two pins. It is used to provide a separate high current power supply to the Motors. A power supply of 5V-40V can be provided to operate DC motors.

7. MOTOR DRIVER CONTENTS

The motor drivers are used to run the DC motor/ stepper motors that are connected to the board according to the data from the microcontroller.

The link between motor and micro controller is given below.

MOTOR 1 - TERMINAL 1 – Port C-2

MOTOR 1 - TERMINAL 2 – Port C-3

MOTOR 2 - TERMINAL 1 – Port C-4

MOTOR 2 - TERMINAL 2 – Port C-5

Programming

WinAVR is a suite of executable, open source software development tools for the Atmel's AVR series of RISC microcontrollers hosted on the Windows platform. It includes the GNU_GCC compiler for C and C++.

Steps for writing a code using WinAVR:

1. Open the Programmer's Notepad and write your code.
2. Create a new folder and save your code in that folder with extension name "main.c".
3. Now open the make file and edit it as mentioned bellow:
    a. Make file→ main filename (give your file name here without extension)
    b. Make file→ MCU type→ ATmega→ (chose your UC)
    c. Make file→ Debug format→ VR-ext-COFF
    d. Make file→ Programmer→ select your programmer (if your programmer is not in the list, then follow the step3.d)
    e. Make file→ port→ (select the port where you have connected your programmer, here its USB)
    f. Make file→ enable editing make file→ then in your make file edit the following things
    g. F_CPU = 16000000 (change it as for your crystal frequency)
    h. AVRDUDE_PROGRAMMER =avrisp (here write down you programmers name).
4. Save the make file in your folder without changing its name.
5. Now open the programmer's notepad.
6. To compile your code and to generate hex file (Tools→ make all).
7. To upload your code into your UC, open AVR USB Programmer, choose the device (here Atmega 16), select the hex file & click on WRITE.

2.2) Motor System

DC motors are used to drive the robot. Microcontroller controls the motion of the motors. The basic principle behind the motion of the motors is H-bridge concept.

H Bridge

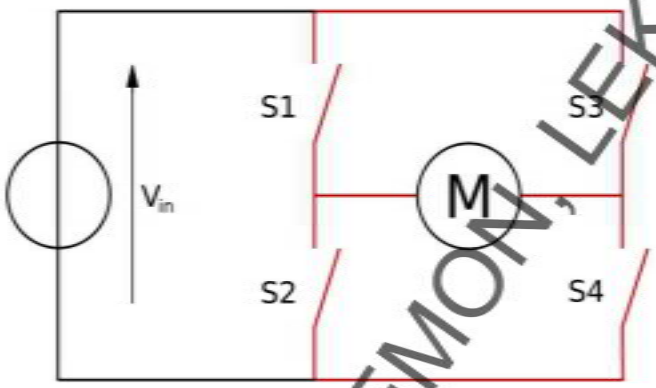


Figure 2: H Bridge

An H bridge is an electronic circuit that enables a voltage to be applied across a load in either direction. These circuits are often used to allow DC motors to run forwards and backwards. H bridges are available as integrated circuits, or can be built from discrete components.

The term H Bridge is derived from the typical graphical representation of such a circuit. An H bridge is built with four switches (solid-state or mechanical). When the switches S1 and S4 (according to the first figure) are closed (and S2 and S3 are open) a positive voltage will be applied across the motor. By opening S1 and S4 switches and closing S2 and S3 switches, this voltage is reversed, allowing reverse operation of the motor.

Using the nomenclature above, the switches S1 and S2 should never be closed at the same time, as this would cause a short circuit on the input voltage source. The same applies to the switches S3 and S4. This condition is known as shoot-through.

The two basic states of an H bridge

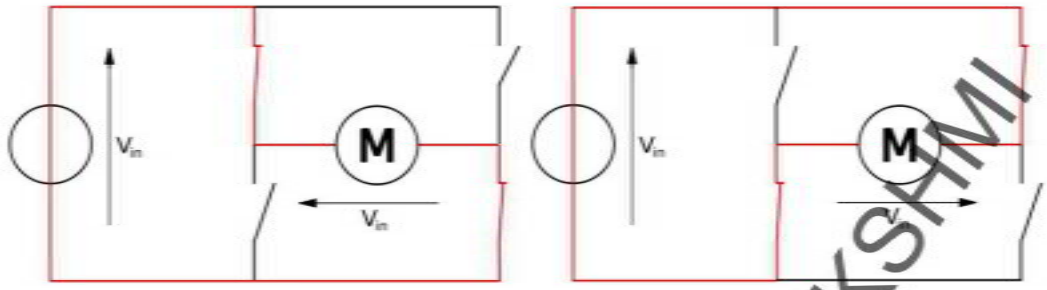


Figure 3: H Bridge States

The H-bridge arrangement is generally used to reverse the polarity of the motor, but can also be used to 'brake' the motor, where the motor comes to a sudden stop, as the motor's terminals are shorted, or to let the motor 'free run' to a stop, as the motor is effectively disconnected from the circuit.

2.3) Cameras

The cameras used in the system are

- iball Face2Face Webcam C8.0(3264 x 2448 resolution): Face Camera
- INTEX ZC0301H webcam.(800 X 600 resolution): Park space Camera
- HS-WC-139 Scarlet Flame webcam.(640 x 480 resolution): Numberplate Camera

They are connected to the PC using USB port. The face camera is used to capture the face images, the numberplate camera for the numberplate images and the park space camera for the parking status.

2.4) MATLAB

The MATLAB version used here is MATLAB v 8.1.0.604(R2013a). It is run on an Intel core i5 processor laptop.

- **Image Acquisition Toolbox:** Three different cameras are used as primary input tools. Images are taken using these cameras when required and acquired as variables in MATLAB with the help of Image Acquisition Toolbox.

- **Image Processing Toolbox:** The images acquired are operated upon using this toolbox to make a decision on the commands to be sent to the microcontroller.
- **Computer Vision System Toolbox:** Exclusively used for face detection and tracking.

2.5) USB to serial converter

The commands from MATLAB are to send via a serial COM port. Since a laptop does not possess a serial port, a USB to serial converter is necessary to convert the data from a USB port to a serial form. The converter consists of a CP2102 chip.

Figure 4: PC-MCU Serial Link

2.6) Power supply

Motor system uses 9V dc power from a battery or some other variable dc power supply. Micro controller uses 5V dc power obtained using a voltage regulator. A laptop is used as base station which can also provide power to the microcontroller through usb.

2.7) PIR Sensor

The PIR (Passive Infra-Red) Sensor is a pyro-electric device that detects motion by sensing changes in the infrared (radiant heat) levels emitted by surrounding objects. This motion can be detected by checking for a sudden change in the surrounding IR pattern. When motion is detected the PIR sensor outputs a high signal (3.3V) on its output pin and otherwise 0V. It is used here to detect vehicle motion.

Features

- Detect a person up to approximately 30 ft.
- Source current up to 12 mA @ 3 V, 23 mA @ 5 V

- Onboard LEDs light up the lens for fast visual feedback when movement is detected.

Key Specifications

- Power Requirements: 3 to 6 VDC; 130 µA idle, 3 mA active (no load)
- Communication: Single bit high/low output
- Operating temperature: 32 to 122 °F (0 to 50 °C)
- Dimensions: 1.41 x 1.0 x 0.8 in (35.8 x 25.4 x 20.3 cm)

Figure 5: PIR Motion Sensor

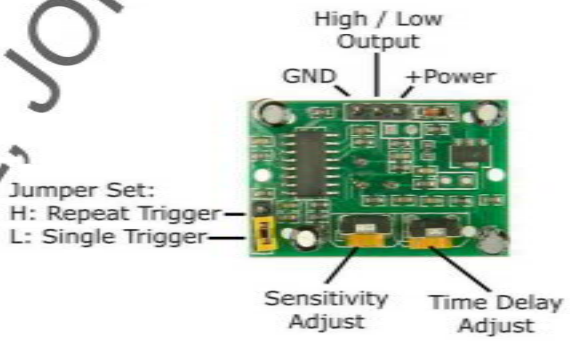


Figure 6: PIR Motion Sensor Circuit

2.8) IR Object Sensor

IR object sensor detect the presence of objects in its signal path. It consists mainly of an IR led and a photodiode and an LM358 comparator. It gives 3.5V output when an object is sensed and 0V otherwise. The range of detection can be altered using a potentiometer provided in it. It has 10-12cm range and potentiometers are used for varying the range.

Figure 7: IR Object Sensor

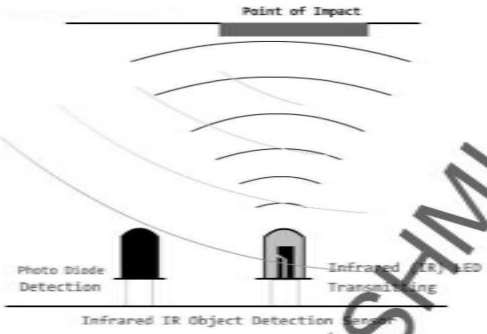


Figure 8: IR Sensor Working

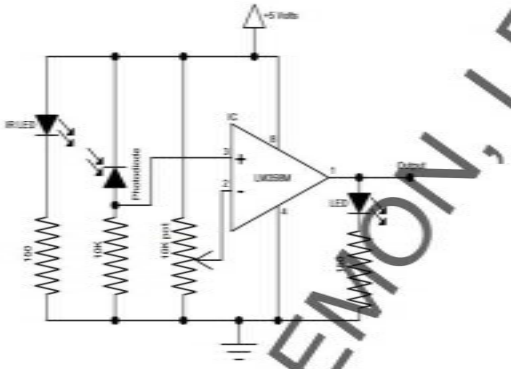


Figure 9: IR Sensor Circuit

2.9) LED Indicators

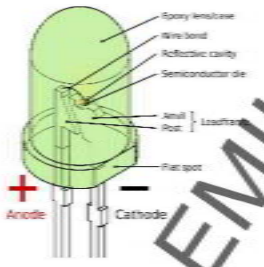


Figure 10: Basic LED Design

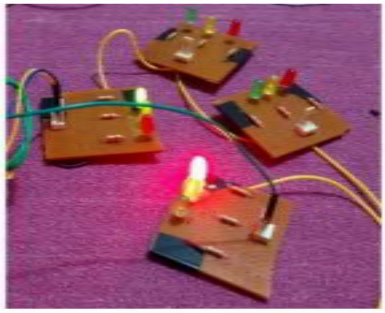


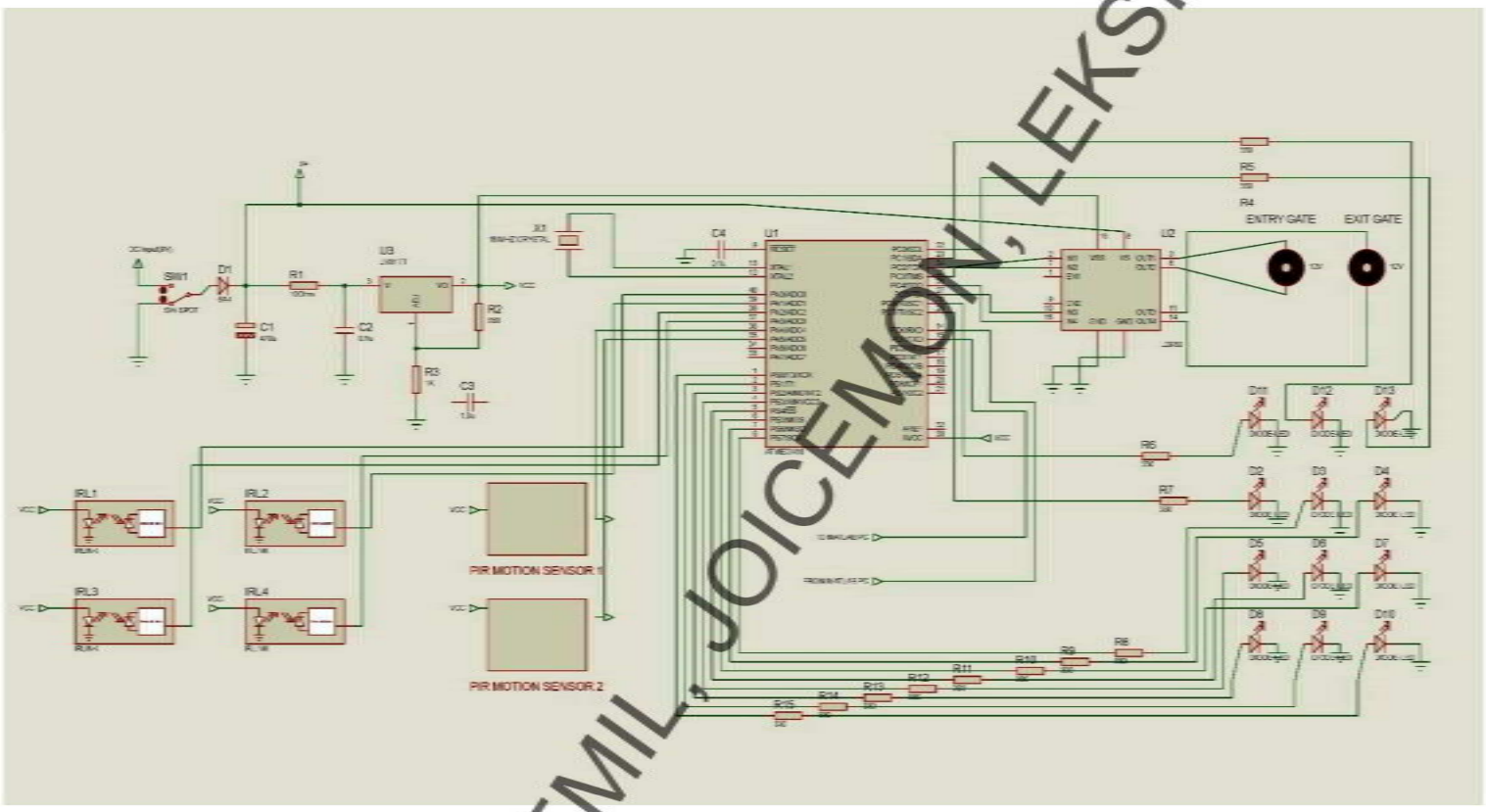
Figure 11: LED Modules

LED modules consisting of a red, yellow and green led in each is used as indicators to the driver. They indicate the status of recognition procedures. LEDs are connected to the microcontroller port pins through a 330 ohm resistors.

Figure 12: LED Connection Diagram

CHAPTER 3:

CIRCUIT DIAGRAM



CIRCUIT DIAGRAM DESCRIPTION & WORKING

3.1) CIRCUIT DESCRIPTION

- MATLAB installed in the PC works as the brain of the system. The connection between the MATLAB and real world is established with the help of an Atmega16A microcontroller. This connection is made possible with the help of UART (Universal Asynchronous Serial Receiver & Transmitter). A usb to serial converter is made use of as the laptop pc doesn't contain a serial port. The microcontroller reads the inputs from the

sensors and send to PC whereas the PC sends the commands for output operations to microcontroller. The serial data configuration is as follows:

- 8 Data bits.
- 2 Stop bits.
- Baud Rate: 9600 bps.
- No Parity.
- No Stop bits.

- Microcontroller used is ATMEGA 16A. It is a 40 pin IC. It has four 8-bit ports. We use 4 pins(PA0, PA1, PA2, PA3) for IR sensor inputs, 2 pins(PA4, PA5) for PIR sensor inputs, 12 pins(PA6, PA7, PBO, PB1 PB2, PB3, PB4, PB5, PB6, PB7, PC0, PC1)
- A 9V dc power from a battery or some other variable dc power supply is applied to the microcontroller board. This is directly applied to the dc motors controlling the entry and exit gates through motor driving IC L293D.
- An LM317 voltage regulator gives 5V supply used as VCC in the entire board which is applied to microcontroller, sensors, leds and L293D IC control inputs.
- Presence of the vehicle at different positions determined by four IR object sensors.
- Motion of the vehicle is sensed using two PIR motion sensors place infront of the gates.
- Status of the procedures indicated to driver with the help of four LED modules. A Red, Yellow and a Green led is soldered into each led indicator module. These led lights are connected to port pins of microcontroller through 330 ohm resistors. Yellow LED indicates 'under processing' green LED indicates 'successful processing' and red LED indicates 'failure of processing'.
- USB to serial converter is used to establish a serial link between the pc and UART of the microcontroller.
- Two DC geared motors which run at approximately 150 rpm when applied with 9V control the gates.

3.2) CIRCUIT WORKING

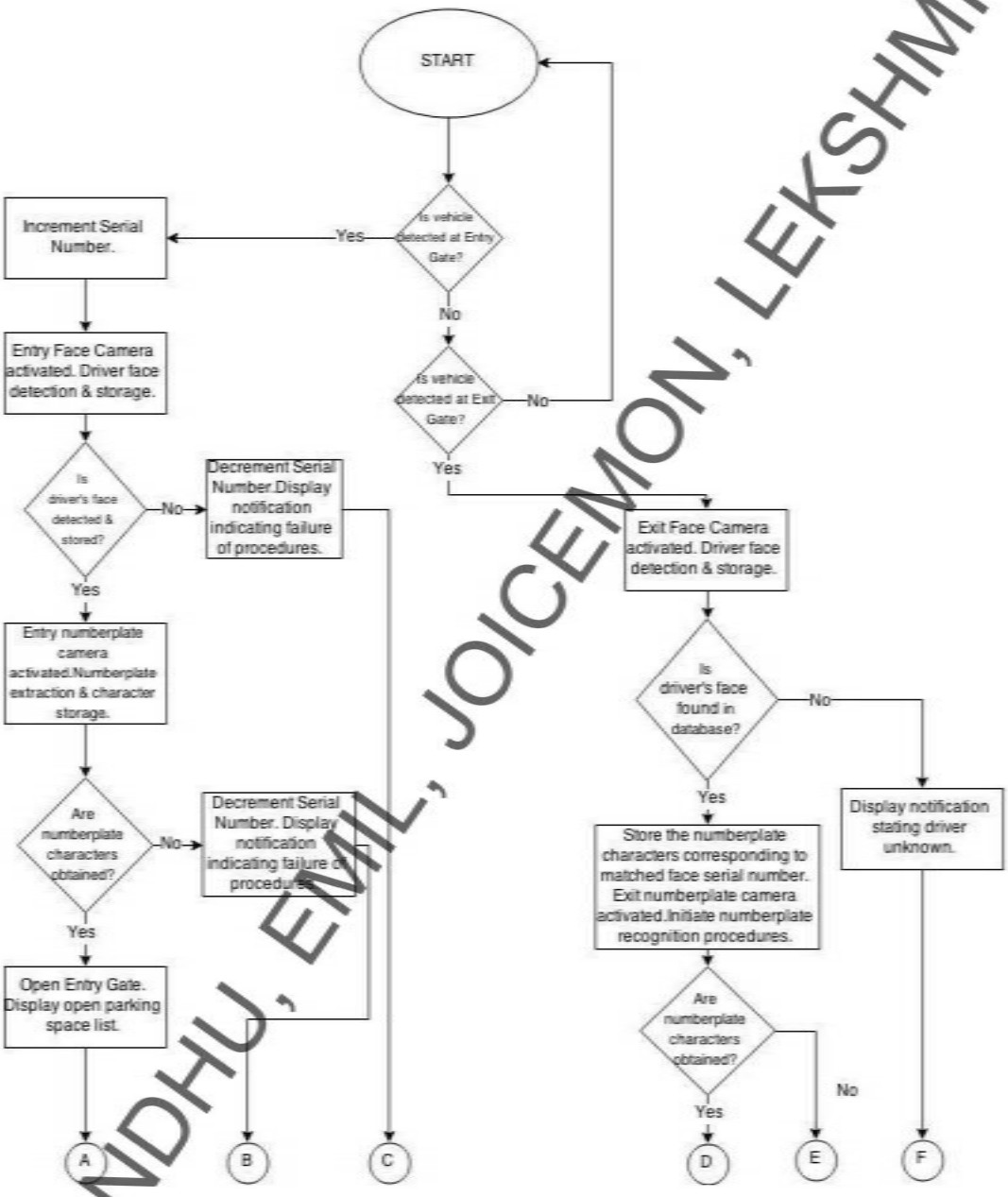
Initially, in the absence of any vehicles at the gates, the sensor outputs are: IR1=0, PIR1=0, IR2=0, IR3=0, PIR2=0, IR4=0. Then the program enters a continuous loop checking for presence of vehicle infront of any of the gates.

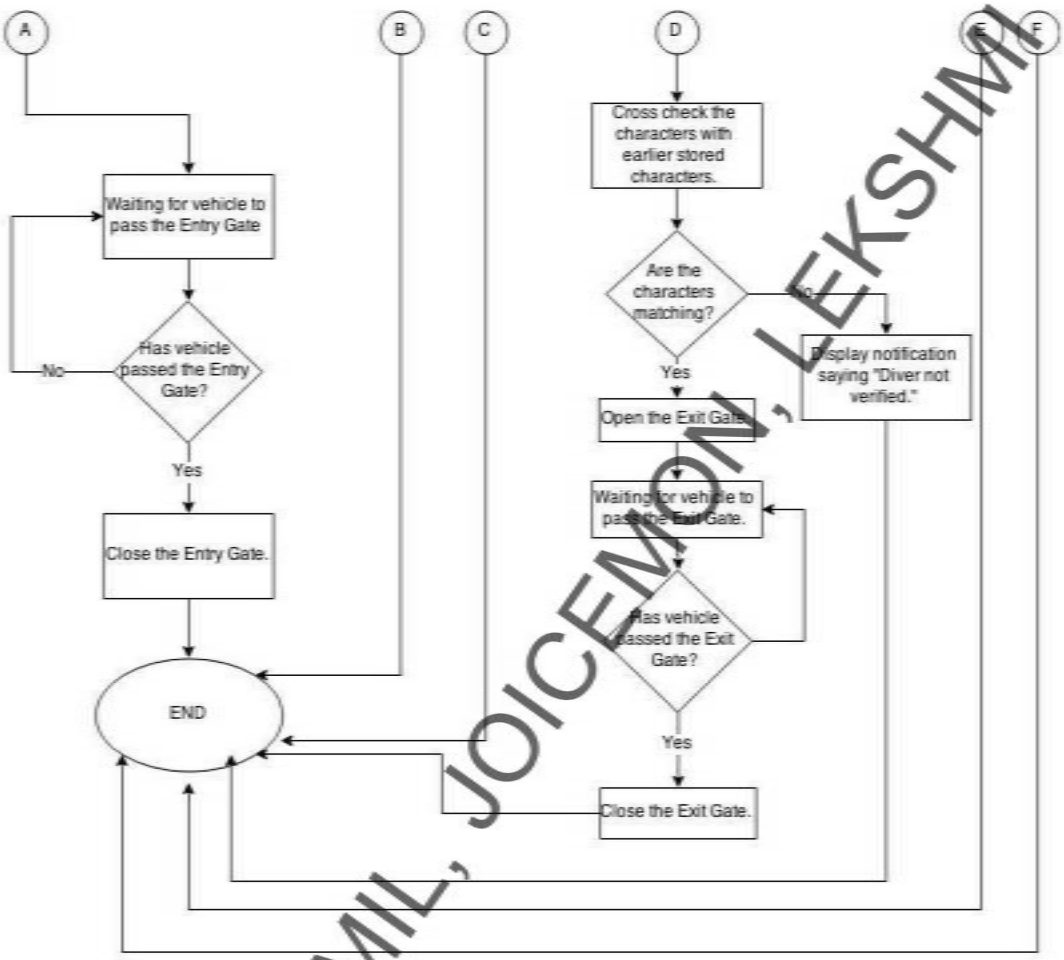
When a vehicle reaches in front of entry gate, IR1 output goes high (3.5V) and PIR1 also goes high (3.5V) indicating its motion. Once the vehicle comes to rest, PIR1 also goes low (0V) following a settling time of 5 seconds. As the condition IR1=1 and PIR1=0 is satisfied, the microcontroller sends character 'a' serially to PC. MATLAB detects it and initiates face detection and recognition program. MATLAB sends character 'A' to microcontroller resulting in lighting Yellow led of entry face detection status module. The successful completion of face detection and Eigen database updation processes is indicated by lighting the Green led of entry face detection status module and turning off the yellow led. A check for available vacant spaces in parking are is done at this point. If the area is full, notification will be given to user at the computer screen. In case the process have failed, the Red led would light up by sending of character 'G'. Following the successful completion of face recognition, the numberplate recognition process initiates. This is indicated by lighting up Yellow led of entry numberplate recognition status module. If the number plate is successfully extracted, then the Green led of entry numberplate recognition status module lights up upon receiving the character 'C' from MATLAB. Failure of the same would cause the Red led of same module to glow upon receiving character 'D'. The text notification messages of all the above mentioned events are shown in computer screen too. Following the successful completion of entry procedures, the entry gate would open upon receiving serial character 'X'. This is done by rotating the dc motor connected to entry gate. The vacant spaces in the parking lot are notified at the computer screen along with this. Once the vehicle has crossed the gate and moved to parking area, IR2 turns high (3.5V) for a small time. This status is sent by microcontroller to pc through character 'b'. The gate closes upon receiving character 'Y' from pc. If any of the procedures have failed, the user will be asked to leave the area through text message. The return is detected by reception of character 'c'. Upon receiving this, the entry program exits and initial status check continues.

When a vehicle reaches in front of exit gate, IR3 output goes high (3.5V) and PIR2 also goes high (3.5V) indicating its motion. Once the vehicle comes to rest, PIR2 also goes low (0V) following a settling time of 5 seconds. As the condition IR3=1 and PIR2=0 is satisfied, the microcontroller sends character 'd' serially to PC. MATLAB detects it and initiates face detection and recognition program. MATLAB sends character 'F' to microcontroller resulting in lighting Yellow led of exit face detection status module. The successful completion of face detection and Eigen face recognition processes is indicated by lighting the Green led of exit face detection status module and turning off the yellow led. Following the successful completion of face recognition, the numberplate recognition process initiates. This is indicated by lighting up Yellow led of exit numberplate recognition status module. If the number plate is successfully extracted and it matches with the stored number, the Green led of exit numberplate recognition

status module lights up upon receiving the character 'L' from MATLAB. Failure of the same would cause the Red led of same module to glow upon receiving character 'J'. The text notification messages of all the above mentioned events are shown in computer screen too. Following the successful completion of exit procedures, the exit gate would open upon receiving serial character 'U'. This is done by rotating the dc motor connected to exit gate. Once the vehicle has crossed the gate, IR4 turns high (3.5V) for a small time. This status is sent by microcontroller to pc through character 'e'. The gate closes upon receiving character 'V' from pc. If any of the procedures have failed, the user will be asked to leave the vehicle back at parking area through text message. The return is detected by reception of character 'c'. Upon receiving this, the entry program exits and initial status check continues.

FLOWCHART





CHAPTER 4:

PROGRAMMING SECTION

4.1) MATLAB PROGRAM

```
% Serial communication Index

% Serial Inputs
%--------------%
% a => IR1=1,PIR1=0,IR2=0,IR3=0,PIR2=0,IR4=0(Incoming vehicle
approaching entry gate)
% b => IR1=0,PIR1=0,IR2=1,IR3=0,PIR2=0,IR4=0(Vehicle crossed the
entry gate)
% c => IR1=0,PIR1=0,IR2=0,IR3=0,PIR2=0,IR4=0(Vehicle moved away
from gate)
% d => IR1=0,PIR1=0,IR2=0,IR3=1,PIR2=0,IR4=0(Outgoing vehicle
approaching exit gate)
% e => IR1=0,PIR1=0,IR2=0,IR3=0,PIR2=0,IR4=1(Vehicle crossed the
exit gate)

% Serial Outputs
%---------------%
% A => Face1YELLOWLED=1, all others=0
% B => Face1YELLOWLED=0, Face1GREENLED=1, NP1YELLOWLED=1, all
others=0
% C => NP1GREENLED=1, Face1GREENLED=1, NP1YELLOWLED=0, all
others=0
% D => NP1REDLED=1, Face1GREENLED=1, NP1YELLOWLED=0, all others=0
% E => All LEDs & motors off
% F => Face2YELLOWLED=1, all others=0
% G => Face1YELLOWLED=0, Face1REDLED=1, all others=0
% H => Face2YELLOWLED=0, Face2GREENLED=1, all others=0
% I => Face2GREENLED=1, NP2YELLOWLED=1, all others=0
% J => NP2REDLED=1, Face2GREENLED=1, NP2YELLOWLED=0 all others=0
% K => Face2REDLED=1, all others=0
% L => NP2GREENLED=1, Face2GREENLED=1, NP2YELLOWLED=0 all others=0
% X => Open Entry Gate
% Y => Close Entry Gate
% U => Open Exit Gate
% V => Close Exit Gate

fwrite(ser,'E'); % Reset all
sl=0; % Initializing serial number
clear np_reg
clear face_reg
clear facedir
clear poslstface
clear meandist
clear similarity
clear EIGEN
clear eigen
clear faceinwt
clear absdist
faceDetector = vision.CascadeObjectDetector;
```

```
stat_init=getsnapshot(parkcam); %Taking initial parkspace image
stat_initg=rgb2gray(stat_init); %Converting to grayscale
stat_initb=stat_initg>230; %Converting to binary
statinit_fin=bwareaopen(stat_initb,200);
boundel=regionprops(statinit_fin,'BoundingBox','Image'); %
Obtaining individual boundaries
boundel1=cat(1,boundel.BoundingBox);
nobound=size(boundel1,1);
for carsl=1:nobound
    bnd(carsl)=boundel1(carsl,1);
end
while(1)
    % Inward path program.
    fwrite(ser,'Q'); % Receive serial input.
    if (fscanf(ser,'%c',1)=='a') % If IR1=1 & PIR1=1(Incoming
vehicle approaching gate)
        fwrite(ser,'A'); % Face1YELLOWLED=1, all others=0
        for i=1:3 % Loop until face is detected.
            pause(1); %
            frame=step(fcam);
            bboxes=step(faceDetector,frame);
            if(isempty(bboxes)==0 && size(bboxes,1)==1)
                facefind=1;
                break;
            end
            facefind=0;
        end
        if (facefind==1) % If a face image is obtained?
            sl=sl+1; % Increment serial No.
            display(sl);
            sum=zeros([11025,1]);
            crpface = imcrop(frame, 'rgb', bboxes);
            facedir{sl,1}=imresize(crpface,[105,105]);
            if(sl>1)
                for fi=1:sl
                    gray=rgb2gray(facedir{fi,1});
                    column{fi,1}=gray(:);
                    sum=sum+column{fi,1};
                end
                average=sum/sl;
                for fi=1:sl
                    normal{fi,1}=column{fi,1}-average;
                    if(fi==1)
                        A=horzcat(normal{fi,1});
                    else
                        A=horzcat(A,normal{fi,1});
                    end
                end
                ATRANS=transpose(A);
                C=ATRANS*A;
                for fi=1:sl
                    eigen{fi,1}=C(:,fi);
                    EIGEN{fi,1}=A*eigen{fi,1};
                end
                for fi=1:sl % Finding & storing weights.
                    for fj=1:sl

face_reg{fi,fj}=dot(normal{fi,1},EIGEN{fj,1});
```

```
                    end
                end
            end
            fwrite(ser,'B'); % Face1YELLOWLED=0, Face1GREENLED=1,
NP1YELLOWLED=1 all others=0

            % Parking Status Analysis
            statnw=getsnapshot(parkcam); %Taking parkspace image
            statnwg=rgb2gray(statnw); %Converting to grayscale
            statdiff=imsubtract(statnwg,stat_initg); %Finding
difference from initial image.
            statnwb=statdiff>30; % Converting to binary.
            statnw_fin=bwareaopen(statnwb,500); % Separating
vehicles occupying >500 pixels.
            carel=regionprops(statnw_fin,'BoundingBox','Image'); %
Obtaining individual vehicle boundaries
            carel1=cat(1,carel.BoundingBox);
            nocar=size(carel1,1); % Finding total no. of vehicles.
            for carsl=1:nocar
                car(carsl)=carel1(carsl,1); % Finding position of
each vehicle.
            end
            space=[0 0 0];
            % Evaluating the pstatus of parking slots. Variable
space gives
            % the output.
            for carsl=1:nocar
                for bndsl=1:nobound
                    if bndsl==1
                        if car(carsl)<bnd(bndsl)
                            space(bndsl)=1;
                            break;
                        end
                    else
                        if (car(carsl)<bnd(bndsl) &&
car(carsl)>bnd(bndsl-1))
                            space(bndsl)=1;
                            break;
                        end
                    end
                end
            end


            % Numberplate Detection
            frame1=getsnapshot(npcam); % Reading the number plate
image.
            in400=imresize(frame1,[400 NaN]); % Resizing the
image keeping aspect ratio same.
            g=rgb2gray(in400); % Converting the RGB (color) image
to gray (intensity).
            gmed=medfilt2(g,[3 3]); % Median filtering to remove
noise.
            se=strel('disk',1); % Structural element (disk of
radius 1) for morphological processing.
            gdil=imdilate(gmed,se); % Dilating the gray image
with the structural element.
            gero=imerode(gmed,se); % Eroding the gray image with
structural element.
```

```
            gdiff=imsubtract(gdil,gero); % Morphological Gradient
for edges enhancement.
            gdiffg=mat2gray(gdiff,[20 255]); % Converting the
class to double.
            gdiffcon=conv2(gdiffg,[1 1;1 1]); % Convolution of
the double image for brightening the edges.
            gdiffad=imadjust(gdiffcon,[0.5 0.7],[0 1],0.1); %
Intensity scaling between the range 0 to 1.
            B=logical(gdiffad); % Conversion of the class from
double to binary.
            % Eliminating the possible horizontal lines from the
output image of regiongrow that could be edges of license plate.
            er=imerode(B,strel('line',50,0));
            out1=imsubtract(B,er);
            % Filling all the regions of the image.
            F=imfill(out1,'holes');
            % Thinning the image to ensure character isolation.
            H=bwmorph(F,'thin',1);
            H=imerode(H,strel('line',3,90));
            % Selecting all the regions that are of pixel area
more than 100.
            final=bwareaopen(H,100);
            % Two properties 'BoundingBox' and binary 'Image'
corresponding to these are acquired.
            Iprops=regionprops(final,'BoundingBox','Image');
            % Selecting all the bounding boxes in matrix of order
numberofboxesX4;
            NR=cat(1,Iprops.BoundingBox);
            if size(NR,1)==10
                  r=1:10;
            else
            % Calling of findchars function.
            r=findchars(NR); % Function 'findchars' outputs the
array of indices of boxes required for extraction of characters.
            end
            vacant=0;
                  for slnom=1:3
                  vacant=vacant+space(slnom); % Finding
number of vacant spaces available for parking.
                  end
            if (vacant ~= 3)
                  if ~isempty(r) % If succesfully indices of
desired boxes are achieved.
                  fwrite(ser,'C'); % NP1GREENLED=1,
Face1GREENLED=1, NP1YELLOWLED=0, all others=0
                  I={Iprops.Image}; % Cell array of 'Image'
(one of the properties of regionprops)
                  for v=1:length(r)
                      N=I{1,r(v)}; % Extracting the binary
image corresponding to the indices in 'r'.
                      letter=readLetter(N); % Reading the
letter corresponding the binary image 'N'.
                      while letter=='O' || letter=='0' % If
difficult to distinguish between '0' and 'O'.
                            if v<=2 || v==5 || v==6      %
1st,2nd,5th & 6th characters can only be alphabets.
                                letter='O';
                            else
```

```
                                letter='0';
                            end
                            break;
                        end
                        while letter=='2' || letter=='Z' % If
difficult to distinguish between '0' and 'O'.
                            if v<=2 || v==5 || v==6      %
1st,2nd,5th & 6th characters can only be alphabets.
                                letter='Z';
                            else
                                letter='2';
                            end
                            break;
                        end
                        np_reg{sl,v}=letter; % Storing the number
elementwise.
                    end


                    % Open space notifications.
                    if space(1)==0
                        display('A4 Open');
                    end
                    if space(2)==0
                        display('A3 Open');
                    end
                    if space(3)==0
                        display('A2 Open');
                    end

                    fwrite(ser,'X'); % Open inward gate.
                    pause(3); % Wait for 3 seconds for entry gate
to open.
                    while(1)
                        fwrite(ser,'Q'); % Receive serial input.
                        if (fscanf(ser,'%c',1)=='b') % If
IR2=1(Vehicle crossed the entry gate)
                            fwrite(ser,'Y'); % Close entry gate.
                            pause(3); % Wait for 3 seconds for
entry gate to open.
                            fwrite(ser,'E'); % All LEDs & motors
off
                            break;
                        end
                    end
                else % If fail to extract the indexes in 'r' this
line an error will be displayed.
                    sl=sl-1; % Deleting alloted serial number.
                    fwrite(ser,'D'); % NP1REDLED=1,
Face1GREENLED=1, NP1YELLOWLED=0 all others=0
                    display('Unable to extract the characters
from the number plate. Please take reverse & exit.');
                    while(1)
                        fwrite(ser,'Q'); % Receive serial input.
                        if (fscanf(ser,'%c',1)=='c') % If IR1=0 &
PIR1=0(Vehicle has left without entering)
```

```
                            fwrite(ser,'E'); % All LEDs & motors
off
                            break;
                        end
                    end
                end
            else
                sl=sl-1; % Deleting alloted serial number.
                display('Parking full. Sorry for the
inconvenience.');
                while(1)
                    fwrite(ser,'Q'); % Receive serial input.
                    if (fscanf(ser,'%c',1)=='c') % If IR1=0 &
PIR1=0(Vehicle has left without entering)
                        fwrite(ser,'E'); % All LEDs & motors off
                        break;
                    end
                end
            end

        else
            % If no face is detected.
            fwrite(ser,'G'); % Face1YELLOWLED=0, Face1REDLED=1,
all others=0
            display('No face is detected. Please take reverse &
exit.');
            while(1)
                fwrite(ser,'Q'); % Receive serial input.
                if (fscanf(ser,'%c',1)=='c') % If IR1=0 &
PIR1=0(Vehicle has left without entering)
                    fwrite(ser,'E'); % All LEDs & motors off
                    break;
                end
            end
        end
    end


    % Outward Path Program.
    fwrite(ser,'Q'); % Receive serial input.
    if (fscanf(ser,'%c',1)=='d') % If IR3=1 & PIR2=0(Outgoing
vehicle approaching exit gate)
        fwrite(ser,'F'); % Face2YELLOWLED=1 all others=0
        clear frame
        for i=1:3
            pause(1);
            frame=step(fcam);
            bboxes=step(faceDetector,frame);
            if(isempty(bboxes)==0 && size(bboxes,1)==1)
                facefind=1;
                break;
            end
            facefind=0;
        end

        if (facefind==1) % If a face image is obtained?
            crpface1 = imcrop(frame, 'rgb', bboxes);
```

```
            faceinp=imresize(crpface1,[105,105]);
            faceingr=rgb2gray(faceinp);
            faceincol=faceingr(:);
            faceinnor=faceincol-average;
            for fj=1:sl
                faceinwt{1,fj}=dot(faceinnor,EIGEN{fj,1});
%Finding the weights of input image.
            end
            for fi=1:sl
                for fj=1:sl
                    diff=faceinwt{1,fj}-face_reg{fi,fj};
                    absdist{fi,fj}=abs(diff);
                end
            end
            meandist=mean(cell2mat(absdist),2);
            minface=min(meandist);
            poslstface=find(meandist==minface); % Identify serial
no. of most similar face
            similarity=abs(cell2mat(faceinwt)-
cell2mat(face_reg(poslstface,:)));
            if(max(similarity)<1e+14)
                display('Face identified. Vehicle identification
in progress. Please wait.');
                fwrite(ser,'H'); % Face1YELLOWLED=0,
Face2GREENLED=1, all others=0

                % Numberplate matching
                frame1=getsnapshot(npcam); % Reading the number
plate image.
                in400=imresize(frame1,[400 NaN]); % Resizing the
image keeping aspect ratio same.
                g=rgb2gray(in400); % Converting the RGB (color)
image to gray (intensity).
                gmed=medfilt2(g,[3 3]); % Median filtering to
remove noise.
                se=strel('disk',1); % Structural element (disk of
radius 1) for morphological processing.
                gdil=imdilate(gmed,se); % Dilating the gray image
with the structural element.
                gero=imerode(gmed,se); % Eroding the gray image
with structural element.
                gdiff=imsubtract(gdil,gero); % Morphological
Gradient for edges enhancement.
                gdiffg=mat2gray(gdiff,[20 255]); % Converting the
class to double.
                gdiffcon=conv2(gdiffg,[1 1;1 1]); % Convolution of
the double image for brightening the edges.
                gdiffad=imadjust(gdiffcon,[0.5 0.7],[0 1],0.1); %
Intensity scaling between the range 0 to 1.
                B=logical(gdiffad); % Conversion of the class from
double to binary.
                % Eliminating the possible horizontal lines from
the output image of regiongrow that could be edges of license
plate.
                er=imerode(B,strel('line',50,0));
                out1=imsubtract(B,er);
                % Filling all the regions of the image.
                F=imfill(out1,'holes');
```

```
                % Thinning the image to ensure character
isolation.
                H=bwmorph(F,'thin',1);
                H=imerode(H,strel('line',3,90));
                % Selecting all the regions that are of pixel area
more than 100.
                final=bwareaopen(H,100);
                % Two properties 'BoundingBox' and binary 'Image'
corresponding to these are acquired.
                Iprops=regionprops(final,'BoundingBox','Image');
                % Selecting all the bounding boxes in the form of
order numberofboxesX4;
                NR=cat(1,Iprops.BoundingBox);
                if size(NR,1)==10
                    r=1:10;
                else
                    % Calling of findchars function.
                    r=findchars(NR); % Function 'findchars'
outputs the array of indices of boxes required for extraction of
characters.
                end
                if ~isempty(r) % If successfully indices of desired
boxes are achieved.
                    fwrite(ser,'I'); % Place2GREENLED=1,
NP2YELLOWLED=1, all others=0
                    I={Iprops.Image}; % Cell array of 'Image' (one
of the properties of regionprops)
                    for v=1:length(r)
                        N=I{1,r(v)}; % Extracting the binary image
corresponding to the indices in 'r'.
                        letter=readLetter(N); % Reading the letter
corresponding the binary image 'N'.
                        while letter=='O' || letter=='0' % If
difficult to distinguish between '0' and 'O'.
                            if v<=2 || v==5 || v==6      %
1st,2nd,5th & 6th characters can only be alphabets.
                                letter='O';
                            else
                                letter='0';
                            end
                            break;
                        end
                        while letter=='2' || letter=='Z' % If
difficult to distinguish between '0' and 'O'.
                            if v<=2 || v==5 || v==6      %
1st,2nd,5th & 6th characters can only be alphabets.
                                letter='Z';
                            else
                                letter='2';
                            end
                            break;
                        end
                        test_np{1,v}=letter; % Storing the number
elementwise.
                    end
                    exitcheck=1;
                    for checks1=1:size(test_np,2) % Comparing
character by character.
```

```
                        if test_np{1,checksl} ~=
np_reg{poslstface,checksl}
                            exitcheck=0; % Driver not verified.
                            break;
                        end
                    end

                    if(exitcheck==1)
                        display('Driver Verified. Please exit.
Hope you enjoyed the facility.');
                        fwrite(ser,'L');% NP2GREENLED=
Face2GREENLED=1, NP2YELLOWLED=0 all others=0
                        fwrite(ser,'U'); % Open exit gate.
                        pause(3); % Wait for 3 seconds for exit
gate to open.
                        while(1)
                            fwrite(ser,'Q'); % Receive serial
input.
                            if (fscanf(ser,'%c',1)=='e') % If
IR4=1(Vehicle crossed the exit gate)
                                fwrite(ser,'T'); % Close exit
gate.
                                pause(3); % Wait for 3 seconds for
exit gate to close.
                                fwrite(ser,'E'); % All LEDs &
motors off
                                break;
                            end
                        end
                    else
                        fwrite(ser,'J'); % NP2REDLED=1,
Face2GREENLED=1, NP2YELLOWLED=0 all others=0
                        display('Driver not verified. Please park
the vehicle back.');
                        while(1)
                            fwrite(ser,'Q'); % Receive serial
input.
                            if (fscanf(ser,'%c',1)=='c') % If
IR3=0 & PIR2=0(Vehicle has returned back to parking site)
                                fwrite(ser,'E'); % All LEDs &
motors off
                                break;
                            end
                        end
                    end
                else % If fail to extract the indexes in 'r' this
line of error will be displayed.
                    fwrite(ser,'J'); % NP2REDLED=1,
Face2GREENLED=1, NP2YELLOWLED=0 all others=0
                    display('Unable to extract the characters from
the number plate. Please park the vehicle back.');
                    while(1)
                        fwrite(ser,'Q'); % Receive serial input.
                        if (fscanf(ser,'%c',1)=='c') % If IR3=0 &
PIR2=0(Vehicle has returned back to parking site)
                            fwrite(ser,'E'); % All LEDs & motors
off
                            break;
```

```
                    end
                end
            end
        else
            display('Face not found in database. Please park
the vehicle back.');
            fwrite(ser,'K'); % Face2YELLOWLED=0,
Face2REDLED=1, all others=0.
            while(1)
                fwrite(ser,'Q'); % Receive serial input.
                if (fscanf(ser,'%c',1)=='c') % If IR3=0 &
PIR2=0(Vehicle has returned back to parking site)
                    fwrite(ser,'E'); % All LEDs & motors off
                    break;
                end
            end
        end
    else % If no face is detected.
        fwrite(ser,'K'); % Face2YELLOWLED=0, Face2REDLED=1,
all others=0
        display('No face is detected. Please park the vehicle
back.');
        while(1)
            fwrite(ser,'Q'); % Receive serial input.
            if (fscanf(ser,'%c',1)=='c') % If IR3=0(Vehicle
has returned back to the parking site)
                fwrite(ser,'E'); % All LEDs & motors off
                break;
            end
        end

    end
end
end
```

4.2) MICROCONTROLLER PROGRAM (EMBEDDED C)

```
#include <avr/io.h>      // Including AVR input output header file.
#include <delay.h>       // Including delay header file.
#define FOSC 16000000 // Setting Clock Speed as 16 MHz
#define BAUD 9600        // Setting Baud Rate as 9600 bps
#define MYUBRR FOSC/16/BAUD-1
void USART_Init( unsigned int ubrr);
unsigned char USART_Receive( void );
void USART_Transmit( unsigned char data );


void main( void )          // Main program
```

```
{
        unsigned char input;
        int st1=0; // Status of Entry Gate.
        int st2=0; // Status of Exit Gate.
        int cnt; // Serial Output loop time

        /* Clearing the ports initially. */
        PORTA=0x00;
        PORTB=0x00;
        PORTC=0x00;

        /* Setting the port functions. */
        DDRA=0xC0;
        DDRB=0xFF;
        DDRC=0xFF;

        USART_Init ( MYUBRR ); // Initializing UART communication.

        while(1)  // INFINITE WHILE LOOP
        {
                input=USART_Receive();

                /* Serial Input Section */
                if (input == 'A')
                {
                        PORTA=0x80;
                        PORTB=0x00;
                        PORTC=0x80;
                        _delay_ms(200);
                }
                else if (input == 'B')
                {
                        PORTA=0x00;
```

```
            PORTB=0x05;
            PORTC=0x00;
            _delay_ms(200);
        }
        else if (input == 'C')
        {
            PORTA=0x00;
            PORTB=0x09;
            PORTC=0x00;
            _delay_ms(200);
        }
        else if (input == 'D')
        {
            PORTA=0x00;
            PORTB=0x03;
            PORTC=0x00;
            _delay_ms(200);
        }
        else if (input == 'E')
        {
            PORTA=0x00;
            PORTB=0x00;
            PORTC=0x00;
            _delay_ms(200);
        }
        else if (input == 'F')
        {
            PORTA=0x00;
            PORTB=0x20;
            PORTC=0x00;
            _delay_ms(200);
        }
        else if (input == 'G')
```

```
{
        PORTA=0x40;
        PORTB=0x00;
        PORTC=0x40;
        _delay_ms(200);
}
else if (input == 'H')
{
        PORTA=0x00;
        PORTB=0x40;
        PORTC=0x00;
        _delay_ms(200);
}
else if (input == 'I')
{
        PORTA=0x00;
        PORTB=0x40;
        PORTC=0x01;
        _delay_ms(200);
}
else if (input == 'J')
{
        PORTA=0x00;
        PORTB=0xC0;
        PORTC=0x00;
        _delay_ms(200);
}
else if (input == 'K')
{
        PORTA=0x00;
        PORTB=0x10;
        PORTC=0x00;
        _delay_ms(200);
```

```
}
else if (input == 'L')
{
        PORTA=0x00;
        PORTB=0x40;
        PORTC=0x02;
        _delay_ms(200);
}
else if (input == 'X')
{
        if (st1==0) // If Entry gate is closed
        {
                PORTC=0X08; // OPEN GATE
                _delay_ms(300);
                PORTC=0X00;
                st1=1; // Entry gate is open.
        }
        _delay_ms(200);
}
else if (input == 'Y')
{
        if (st1==1) // If Entry gate is open
        {
                PORTC=0X04; // CLOSE GATE
                PORTB=0X00;
                _delay_ms(300);
                PORTC=0X00;
                st1=0; // Entry gate is closed.
        }
        _delay_ms(200);
}
else if (input == 'U')
{
```

```
                if (st2==0) // If Exit gate is closed
                {
                        PORTC=0X20; // OPEN GATE
                        PORTB=0X00;
                        _delay_ms(300);
                        PORTC=0X00;
                        st2=1; // Exit gate is open.
                }
                _delay_ms(200);
        }
        else if (input == 'V')
        {
                if (st2==1) // If Exit gate is opened
                {
                        PORTC=0X10; // CLOSE GATE
                        PORTB=0X00;
                        _delay_ms(300);
                        PORTC=0X00;
                        st2=0; // Exit gate is closed.
                }
                _delay_ms(200);
        }
        else if (input == 'Q')
        {
                /* Serial Output Section */
                if(PINA == 0x01 || PINA == 0x41 || PINA == 0x81 || PINA ==
0xC1)
                {
                        USART_Transmit('a');
                }
                else if (PINA == 0x02 || PINA == 0x42 || PINA == 0x82 || PINA
== 0xC2)
                {
```

```
                        USART_Transmit('b');
                    }
                    else if (PINA == 0x00 || PINA == 0x40 || PINA == 0x80 || PINA
== 0xC0)
                    {
                        USART_Transmit('c');
                    }
                    else if (PINA == 0x04 || PINA == 0x44 || PINA == 0x84 || PINA
== 0xC4)
                    {
                        USART_Transmit('d');
                    }
                    else if (PINA == 0x08 || PINA == 0x48 || PINA == 0x88 || PINA
== 0xC8)
                    {
                        USART_Transmit('e');
                    }
                    else
                    {
                        USART_Transmit('x');
                    }
                }
            }

}

/*=================================================*/

/* USART INITIALIZATION */

void USART_Init( unsigned int ubrr)
{
/* Set baud rate */
```

```
UBRRH = (unsigned char)(ubrr>>8);
UBRRL = (unsigned char)ubrr;
/* Enable receiver and transmitter */
UCSRB = (1<<RXEN)|(1<<TXEN);
/* Set frame format: 8data, 2stop bit */
UCSRC = (1<<URSEL)|(1<<USBS)|(3<<UCSZ0);
}
/*=====================================================*/

/* CONFIGURING USART RECEIVER */

unsigned char USART_Receive( void )
{
/* Wait for data to be received */
while ( !(UCSRA & (1<<RXC)) )
;
/* Get and return received data from buffer */
return UDR;
}

/*=====================================================*/

/* CONFIGURING USART TRANSMITTER */
void USART_Transmit( unsigned char data )
{
/* Wait for empty transmit buffer */
while ( !( UCSRA & (1<<UDRE)) )
;
/* Put data into buffer, sends the data */
UDR = data;
}
```

CHAPTER 5:

IMAGE PROCESSING TECHNIQUES

5.1) EIGENFACE DATABASE CREATION PROCEDURE

1) Read face image taken by camera using Viola Jones Algorithm.
2) Convert face image into face vector.
3) Normalize the face vector.
4) Concatenate face vectors into a matrix.
5) Calculate covariance matrix.
6) Calculate Eigen vectors using covariance matrix.
7) Represent each face image as a linear combination of all Eigen vectors.
8) Find weights of all Eigen faces & store in database.

5.2) EIGENFACE RECOGNITION PROCEDURE

1) Read face image taken by camera using Viola Jones Algorithm.
2) Convert face image into face vector.
3) Normalize the face vector.
4) Project normalized face vector onto the Eigenspace.
5) Find weight vector of input image.
6) Calculate distance between input weight vector & all weight vectors stored in database.
7) If distance is less than threshold set, then the face is recognized.
8) Otherwise, it is rejected.
9)

5.3) PARKSPACE EVALUATION PROCEDURE

1) Initial park space image is taken, converted to grayscale & stored.
2) Boundaries extracted by thresholding & selection.
3) Boundary coordinates evaluated & stored.
4) Present park space image is taken & converted into grayscale.

5) Initial image is subtracted from present image.
6) Thresholding & selection applied to extract the vehicle images.
7) Vehicle coordinates evaluated & stored.
8) Comparing the vehicle coordinates with the boundary coordinates, the status of each parking slot identified.
9) Driver is notified about the status.

5.4) NUMBERPLATE RECOGNITION PROCEDURE

1) Read image taken by camera.
2) Filtering and morphological processing.
3) Extracting the numberplate area.
4) Selecting all bounding boxes.
5) Extract only valid characters of numberplate.
6) Load template character data base.
7) Apply 2D-correlation to obtain the best match.
8) Giving the character outputs.

CHAPTER 6:

ADVANTAGES & DISADVANTAGES

6.1) ADVANTAGES

- Avoids security guards.
- Saves arrival time and departure time.
- Provides improved security.
- Vehicle is under the control of parked driver.
- Child safety parking system.
- Car thefts can be avoided.
- Traffic congestion is reduced at small parking areas. Mainly at underground parking.
- The time for searching of parking space greatly reduced.
- Computer vision based system has high reliability.
- Surveillance equipments in the parking area can be reduced.
- Avoids the waste of time for parking process.
- Vehicle driver given direction to park at parking space.
- Parking space equally available to all.
- Reduced environment pollution.
- Reduce the accidents in the parking area.
- Can be used in large area parking system.
- Unnecessary quarrel between drivers can be avoided.

6.2) DISADVANTAGES

- Difficult to encounter emergency situations.
- Variations in lighting can cause errors.
- The parking space cannot be reserved for anyone.
- Driver should wait at the gate till all procedures are completed.
- Lack of opportunity for paid parking.
- Vehicle queue occurs in parking entrance or exit gate.
- In a closed or underground parking system, dangerous situations for driver's health.
- Lack of night vision cameras.
- The environmental condition may cause problems in detection of number plate.

CHAPTER 7:
APPLICATIONS

The Secure Parking System has numerous applications with regard to parking of vehicles in places where a large number of people gather. It can be implemented permanently in shopping malls, cinema theatres, IT parks, etc. where large people arrive every day. Mobile unit of the system can be implemented in marriage receptions, exhibitions, cultural fests, etc.

CHAPTER 8:

FUTURE SCOPE

The Secure Parking System is going to revolutionize the present concepts of parking facilities. The security, order & easiness provided by the system will greatly reduce parking time. Eventhough the system is effective at its present state, there is still opportunity for improvement. The pay & park facility common nowadays can be added with the help of a currency recognizer & counter system. The cash credit facility can also be added to it setting a cash balance register for each vehicle.

CHAPTER 9:
CONCLUSION

In this project we designed and implemented a parking system which provides security and order in parking related activities. The system makes use of computer vision with the help of cameras. The implementation is done by programming in MATLAB. Some digital signal processing techniques, morphological processing concepts and common logic were made use of. Experience was also gained in Image acquisition and Image processing toolboxes of MATLAB.

From the work done in this project, it could be understood that computer vision and image processing can play an important role in improving most of the everyday activities. By using image processing, the simple system that was developed here could perform the tasks which would have been very difficult to execute by manual control.

REFERENCES

- Face recognition using Eigenfaces: Matthew Turk, Alex Pentland, Vision and Modelling Group, The Media Laboratory Massachusetts, Institute of Technology; Conference on Computer Vision and Pattern Recognition, 1991. Proceedings CVPR '91., IEEE Computer Society.
- TRACKING NUMBER PLATE FROM VEHICLE USING MATLAB Manisha Rathore and Saroj Kumari Department of Information Technology, Banasthali University, Jaipur, India
- Websites:
  - www.pages.drexel.edu/~sis26/Eigenface%20Tutorial.htm
  - openbio.sourceforge.net/.../eigenfaces/eigenfaces-html/facesOptions.html
  - http://in.mathworks.com/matlabcentral/fileexchange/
  - www.google.com.
  - www.youtube.com.