MINI-PROJECT - II
(2020-2021)

Smart Parking System
(Internet of Things)

PROJECT FINAL REPORT

Department of Computer Engineering & Applications
Institute of Engineering & Technology

Team Members:

Prateek Dubey
(181500494)

Shivam Samadhiya
(181500670)

Harshit Verma
(181500260)

Harshita Katara
(181500263)

Supervised By:-

Mr. Amir Khan Sir
(Senior Technical Trainer)

Certificate

This is to certify that Prateek Dubey, Shivam Samadhiya Harshita Katara, and Harshit Verma students of B.Tech (CSE) 3rd year has successfully Completed the **MINI PROJECT - II** named **Smart Parking System** on Internet of Things (IOT) under the Guidance of Mohd**. Amir Khan** during 2020-21.

Signature:

Mohd.Amir Khan
(Senior Technical Trainer)

ACKNOWLEDGEMENT

First and foremost, we would like to thank our mentor of this project Mohd.Amir Khan Sir for his valuable guidance and advice. He guided us greatly to work on this project. His willingness to motivate us helped tremendously. Besides, we would like to thank the authority of GLA University for providing us with a good environment and facilities to complete this project. Finally, and honourable mention goes to our families and friends for their understanding and support in completing this project. Without help from those mentioned above, this project could not have been completed.

Prateek Dubey (181500494)
Shivam Samadhiya (181500670)
Harshita Katara (181500263)
Harshit Verma (181500260)

Mini-Project Synopsis

Project Information:

Title of Project	Smart Parking System
Technology Used	• Internet of Things • Machine Learning

Technical Details	**Hardware Requirements :**
	
	• Infra-Red (IR) sensors
	• Personal Computer
	• LCD Display
	• Servo Motor
	• Connecting Wires
	• Raspberry Pi (Mini-Computer)
	• Webcam
	
	Software Requiremnts :
	
	• **APPLICATION SOFTWARE**
	  ○ Arduino IDE (Version 1.8.13)
	  ○ Raspbian OS: A free Debian-based OS optimized Pi's with all basic programs and utilities we expect from a general-purpose operating system.
	  ○ Pycharm
	
	• **MACHINE LEARNING**
	  ○ OpenCV
	
	• **Language**
	  ○ Python
	
	• **Back-end**
	  ○ Firebase

Contents

Abstract ……	
1. **Introduction**…… 1.1 General Introduction of the topic……... 1.2 Area of Computer Science……. 1.2.1 Internet of Things. 1.2.2 Machine Learning (Open CV) 1.2.3 Google Firebase	
2. Problem Definition…….	
3. Objectives.....	
4. Software & Hardware Requirement Analysis…… 4.1 Activity Diagram 4.2 DFD	
5. Implementations Details……	
6. Some Screenshots……..	
7. References……....	

Abstract:

With the increase in vehicle production and world population, more and more parking spaces and facilities are required. In today's technological world the concept of smart city has become an area of interest. Concern to parking impending in an urban area. The parking space problem can be turn into a new opportunity brought by the recent trends to meet the world's connected continuum. In the past, there have been many works done on smart parking system approaching an even smarter system in where researches have been done and still being done to create a system which is not technologically savvy but also at ease. This project makes easy for the user to find automatically a free space at the low cost and without consuming time and fuel. The WebApp is also provided to user to check the availability of free space for parking. This project is aimed to create a system that helps people with personal vehicles to find for parking easily. Both software and hardware platform have been developed in this system.

This system uses ultrasonic sensors to detect either vehicle park occupancy or not. Features of Smart Parking System include vacant parking space detection, display of available parking spaces, and directional indicators toward vacant parking spaces, through the use of specific LEDs.

We implemented a full-fledged prototype system for parking management to realize the design functionalities and features mentioned. Our preliminary test results show that the performance of this i.e. internet of thing IOT based system can effectively satisfy the needs and requirements of existing parking hassles thereby minimizing the time consumed to find vacant parking lot, real-time information rendering.

1. Introduction

1.1 General Introduction of the topic

Currently, most of the existing car parks do not have a systematic system. Most of them are manually managed and a little inefficient. The problem that always occurs at the car park is time being wasted in searching for the available parking spaces. Users will keep on circling the parking area until they found a vacant parking slot. This problem usually occurs in urban areas, where number of vehicles is higher as compared to the availability of parking spaces. These ineffective conditions happened because of the lack of implementation in technologies which are available in the market today. In this current era of modern world, almost everyone owns a personal vehicle and it has become a basic need for the humans. Hence, it has been proven statistically that the usage of vehicles is increasing rapidly yearly. Due to the growth, it is very difficult to find parking slots in cities, especially during the peak time.

This creates a necessity to introduce an automated system that allows users to see their spot just by making a few clicks through a custom made Web Application. This serves to hassle free situation for each and every users.

The development of this project prototype can act as way-finder to guide car driver inside the car park to parking slot available inside car park and guides car driver to go there. It is a Raspberry Pi, Arduino microcontroller based project. It uses the infrared sensor to detect the vacancy of each parking slot at a level of car park, sending signal wireless to microcontroller to process and display total of available parking slot on 16x2 LCD displays. At the same time, it also displays identify the Vehicle Number using OpenCV.

This project is developed based on the research in existing parking system at the crowded parking area such as shopping complex or mall. Knowing that some parking areas are hard to find an available parking lots, this project is purposely build to solve the problem. This project focus on finding the best way to guide drivers and vehicle’s users to get a free parking lot in short of time Smart Parking System can inform the drivers which parking zones are available and the number of free parking space so that they can make good decision about where they wish to park, and thus find the free parking conveniently.

1.2 Area of Computer Science

1.2.1 Internet of Things

The internet of things, or IOT, is a system of interrelated computing devices, mechanical and digital machines, or people that are provided with unique identifiers (UIDs) and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction.

An IOT ecosystem consists of web-enabled smart devices that use embedded systems, such as processors, sensors and communication hardware, to collect, send and act on data they acquire from their environments. IOT devices share the sensor data they collect by connecting to an IOT gateway or other edge device where data is either sent to the cloud to be analyzed or analyzed locally. Sometimes, these devices communicate with other related devices and act on the information they get from one another. The devices do most of the work without human intervention, although people can interact with the devices -- for instance, to set them up, give them instructions or access the data. The connectivity, networking and communication protocols used with these web-enabled devices largely depend on the specific IOT applications deployed. IOT can also make use of artificial intelligence (AI) and machine learning to aid in making data collecting processes easier and more dynamic.

The internet of things helps people lives and work smarter, as well as gain complete control over their lives. In addition to offering smart devices to automate homes, IOT is essential to business. IOT provides businesses with a real-time look into how their systems really work, delivering insights into everything from the performance of machines to supply chain and logistics operations.

IOT enables companies to automate processes and reduce labour costs. It also cuts down on waste and improves service delivery, making it less expensive to manufacture and deliver goods, as well as offering transparency into customer transactions.

As such, IOT is one of the most important technologies of everyday life, and it will continue to pick up steam as more businesses realize the potential of connected devices to keep them competitive.

Description of some devices used in this project:

I) **Infra-Red Sensors:** An **infrared sensor** is an electronic device that emits in order to sense some aspects of the surroundings. An **IR sensor** can measure the heat of an object as well as detects the motion. These types of **sensors** measure only **infrared** radiation, rather than emitting it that is called a passive **IR sensor**.

II) **LED Lights:** A light-emitting diode is a semiconductor light source that emits light when current flows through it. Electrons in the semiconductor recombine with electron holes, releasing energy in the form of photons.

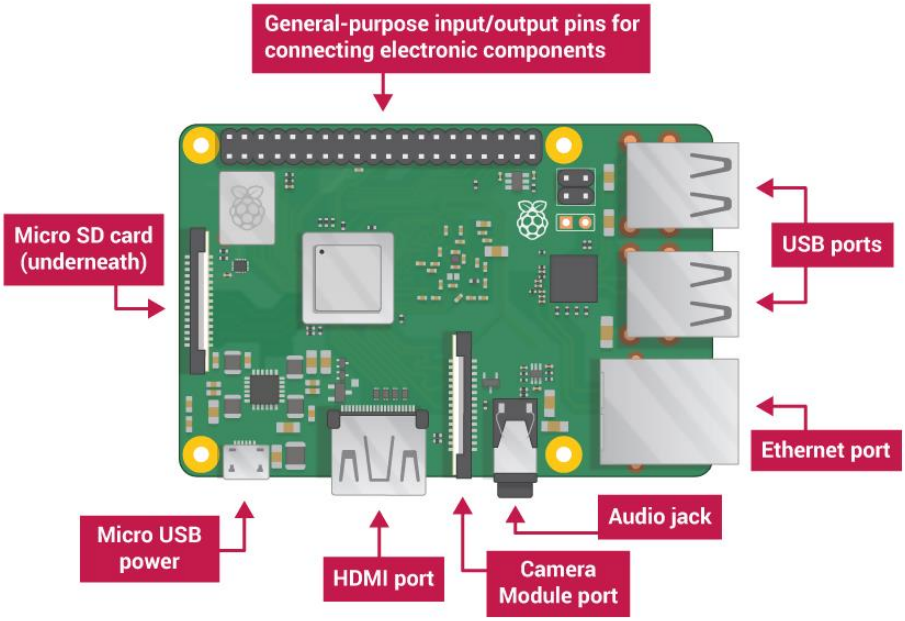
III) **LCD Display:** An **LCD** (**Liquid Crystal Display**) **screen** is an electronic **display** module and has a wide range of applications. A **16x2 LCD display** is very basic module and is very commonly

used in various devices and circuits. A **16x2 LCD** means it can **display** 16 characters per line and there are 2 such lines.

IV) Servo motor: A **servo motor** is a type of **motor** that can rotate with great precision. Normally this type of **motor** consists of a control circuit that provides feedback on the current position of the **motor** shaft; this feedback allows the **servo motors** to rotate with great precision.

V) Connecting Wires: Connecting wires provide a medium to an electrical current so that they can travel from one point on a circuit to another. In the case of computers, **wires** are embedded into circuit boards to carry pulses of electricity.

VI) **Raspberry Pi:** The Raspberry Pi is a low cost, **credit-card sized computer** that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is a capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python. It's capable of doing everything you'd expect a desktop computer to do, from browsing the internet and playing high-definition video, to making spread sheets, word-processing, and playing games.



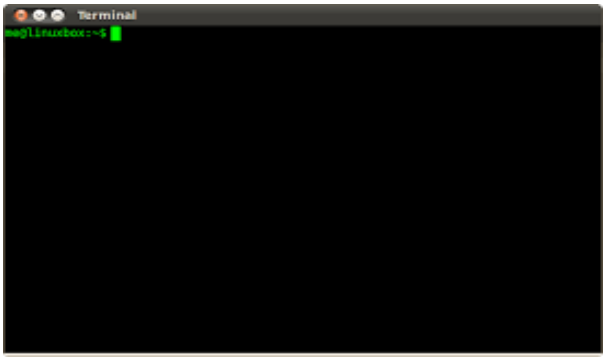
1.2.2 OpenCV

OpenCV was started at Intel in 1999 by **Gary Bradsky**, and the first release came out in 2000. **Vadim Pisarevsky** joined Gary Bradsky to manage Intel's Russian software OpenCV team. In 2005, OpenCV was used on Stanley, the vehicle that won the 2005 DARPA Grand Challenge. Later, its active development continued under the support of Willow Garage with Gary Bradsky and Vadim Pisarevsky leading the project. OpenCV now supports a multitude of algorithms related to Computer Vision and Machine Learning and is expanding day by day. OpenCV supports a wide variety of programming languages such as C++, Python, Java, etc., and is available on different platforms including Windows, Linux, OS X, Android, and iOS. An interface for high-speed GPU operations based on CUDA and OpenCL are also under active development. OpenCV-Python is the Python API for OpenCV, combining the best qualities of the OpenCV C++ API and the Python language.

OpenCV-Python is a library of Python bindings designed to solve computer vision problems. **OpenCV** is **used** for all sorts of image and video analysis, like facial recognition and detection, license plate reading, photo editing, advanced robotic vision, optical character recognition, and a whole lot more. **OpenCV is Open Source Commuter Vision Library**, so it can be easily installed in Raspberry Pi with Python and Linux environment. And Raspberry Pi with OpenCV and attached camera can be used to create many real-time image processing applications like Face detection, face lock, object tracking, car number plate detection, Home security system etc.

How To Install OpenCV In Raspberry PI 3

As OpenCV requires so many packages on the Raspberry Pi, we will install these in a step by step process**.**



You can update the currently installed packages by running the following two commands

```
sudo apt update
sudo apt upgrade
```

2. Now we can start the process of installing all the packages we need for OpenCV to compile.

To start, run the command below. This command will install the packages that contain the tools needed to compile the OpenCV code.

```
sudo apt install cmake build-essential pkg-config git
```

3. Next, we are going to install the packages that will add support for different image and video formats to OpenCV.

Install these libraries to your Raspberry Pi with the following command

```
sudo apt install libjpeg-dev libtiff-dev libjasper-dev libpng-dev libwebp-dev libopenexr-dev

sudo apt install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev libxvidcore-dev
libx264-dev libdc1394-22-dev libgstreamer-plugins-base1.0-dev libgstreamer1.0-dev
```

4. Our next step is to install all the packages needed for OpenCV's interface by using the command below.

```
sudo apt install libgtk-3-dev libqtgui4 libqtwebkit4 libqt4-test python3-pyqt5
```

5. These next packages are crucial for OpenCV to run at a decent speed on the Raspberry Pi.

You can install these packages by running the following command.

```
sudo apt install libatlas-base-dev liblapacke-dev gfortran
```

6. The second last lot of packages that we need to install relate to the Hierarchical Data Format (HDF5) that OpenCV uses to manage data.

Install the HDF5 packages to your Pi by using the command below

```
sudo apt install libhdf5-dev libhdf5-103
```

7. Finally, we can install the final few packages by using the command below.

These last few packages will allow us to compile OpenCV with support for Python on our Raspberry Pi.

```
sudo apt install python3-dev python3-pip python3-numpy
```

Before proceeding to the next section, make sure all the packages installed successfully.

Preparing your Raspberry Pi for Compiling OpenCV

1.1 OpenCV Raspberry Pi

1. With all the required packages to compile OpenCV on our Raspberry Pi now installed, we need to do some preparatory work before we can start the compilation process.

We will now need to temporarily increase the size of the swap space to help the process of compiling OpenCV on the Raspberry Pi.

The swap space is used by the operating system when the device has run out of physical RAM. While swap memory is a lot slower than RAM, it can still be helpful in certain situations.

Begin modifying the swap file configuration by running the following command.

```
sudo nano /etc/dphys-swapfile
```

2. While we are within this file, we need to find and replace the following line.

Find

```
CONF_SWAPSIZE=100
```

Replace With

```
CONF_SWAPSIZE=2048
```

Once changed, save the file by pressing CTRL+X followed by Y then Enter.

3. As we have made changes to the swapfile configuration, we need to restart its service by utilizing the command below.

```
sudo systemctl restart dphys-swapfile
```

By restarting the service, we are forcing it to recreate the swap file.

4. Next, let's go ahead and clone the two OpenCV repositories we need to our Raspberry Pi.

Running these two commands will retrieve the latest available version of OpenCV from their git repository.

```
git clone https://github.com/opencv/opencv.git

git clone https://github.com/opencv/opencv_contrib.git
```

As these repositories are quite large, they may take some time to clone to your Raspberry Pi.

Compiling OpenCV on your Raspberry Pi

1.2 Raspberry Pi zero OpenCV

1. Let's start by creating a directory called "build" within the cloned "opencv" folder and then changing the working directory to it.

```
mkdir ~/opencv/build

cd ~/opencv/build
```

In this folder, we will be compiling OpenCV on your Raspberry Pi.

2. Now that we are within our newly created build folder, we can now use cmake to prepare OpenCV for compilation on our Raspberry Pi.
Run the following command to generate the required makefile.

```
cmake -D CMAKE_BUILD_TYPE=RELEASE \
    -D CMAKE_INSTALL_PREFIX=/usr/local \
    -D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib/modules \
    -D ENABLE_NEON=ON \
    -D ENABLE_VFPV3=ON \
    -D BUILD_TESTS=OFF \
    -D INSTALL_PYTHON_EXAMPLES=OFF \
    -D OPENCV_ENABLE_NONFREE=ON \
    -D CMAKE_SHARED_LINKER_FLAGS=-latomic \
    -D BUILD_EXAMPLES=OFF ..
```

3. Once the make file has successfully finished generating, we can now finally move on to compiling OpenCV by running the command below.

We use the argument -j$(nproc) to tell the compiler to run a compiler for each of the available processors.

Doing this will significantly speed up the compilation process and allow each core on the Raspberry Pi to work on compiling OpenCV.

```
make -j$(nproc)
```

Please note that the compilation process can take considerable time. On our Raspberry Pi 4, this process took about 1 hour to complete.

4. When the compilation process finishes, we can then move on to installing OpenCV.

Luckily for us, this is a reasonably straightforward process and requires you to run the following command.

```
sudo make install
```

This command will copy all the required files into there needed locations automatically.

5. Now we also need to regenerate the operating systems library link cache.

The Raspberry Pi won't be able to find our OpenCV installation if we don't run the following command.

```
sudo ldconfig
```

Cleaning up after Compilation

1.4 Python OpenCV test

1. To test whether OpenCV is now installed to our Raspberry Pi, we will make use of our Python 3 installation.

Launch into the Python terminal by running the command below.

```
python3
```

2. While we are within Python, we can now import the OpenCV Python module using the command below.

By importing the module, we can first check to see if OpenCV will even load on our Pi.

```
>>>  import cv2
```

3. With the OpenCV module now imported, we should be able to retrieve its version.

To retrieve OpenCV's version, use the following command.

```
>>> cv2.__version__
```

4. If everything is now working as intended and OpenCV has been successfully installed to your Raspberry Pi, you should see text like the following appear in the command line.

```
'4.1.2'
```

Hopefully, at this point we will have OpenCV up and running.

1. Segmentation and contours

Image segmentation is a process by which we partition images into different regions. Whereas the **contours** are the continuous lines or curves that bound or cover the full boundary of an object in an image. And, here we will use **image segmentation technique called contours** to extract the parts of an image.

Also contours are very much important in

- **Object detection**
- **Shape analysis**

And they have very much broad field of application from the real world image analysis to medical image analysis such as in MRI's

2. Hierarchy and Retrieval Mode

Retrieval mode defines the hierarchy in contours like sub contours, or external contour or all the contours.

Now there are four retrieval modes sorted on the hierarchy types.

cv2.RETR_LIST – retrieves all the contours.

cv2.RETR_EXTERNAL – retrieves external or outer contours only.

cv2.RETR_CCOMP – retrieves all in a 2-level hierarchy.

cv2.RETR_TREE – retrieves all in a full hierarchy.

Hierarchy is stored in the following format [Next, Previous, First child, parent]

3. Approximating Contours and Finding their convex hull

In approximating contours, a contour shape is approximated over another contour shape, which may be not that much similar to the first contour shape.

For approximation we use ***approxPolyDP*** function of OpenCV which is explained below

```
cv2.approxPolyDP (contour, approximation accuracy, closed)
```

Parameters:

- **Contour** – is the individual contour we wish to approximate.

- **Approximation Accuracy**- important parameter in determining the accuracy of approximation, small value give precise approximation, large values gives more generic information. A good thumb rule is less than 5% of contour perimeter.
- **Closed** – a Boolean value that states whether the approximate contour could be open or closed.

4. Convex Hull

Convex hull is basically the outer edges, represented by drawing lines over a given figure.

It could be the smallest polygon that can fit around the object itself**.**

```
import cv2

import numpy as np

image=cv2.imread('star.jpg')

gray=cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)

cv2.imshow('original image',image)

cv2.waitKey(0)
```

Threshold the image

```
ret, thresh=cv2.threshold(gray,176,255,0)
```

Find contours

```
_, contours, hierarchy=cv2.findContours (thresh.copy (), cv2.RETR_LIST, cv2.CHAIN_APPROX_NONE)
```

Sort the contours by area and then remove the largest frame contour

```
n=len(contours)-1

contours=sorted(contours, key=cv2.contourArea,reverse=False)[:n]
```

Iterate through the contours and draw convex hull

For c in contours:

```
hull=cv2.convexHull(c)

cv2.drawContours (image,[hull],0,(0,255,0),2)

cv2.imshow ('convex hull', image)

cv2.waitKey (0)

cv2.destroyAllWindows ()
```

5. Matching Contour by shapes

```
cv2.matchShapes (contour template, contour method, method parameter)
```

Output – match value (lower value means a closer match)

Contour template – This is our reference contour that we are trying to find in a new image.

Contour – The individual contour we are checking against.

Method – Type of contour matching (1, 2, 3).

Method parameter – leave alone as 0.0 (not utilized in python OpenCV)

```
import cv2

import numpy as np
```

Load the shape template or reference image

```
template= cv2.imread ('star.jpg',0)

cv2.imshow ('template', template)

cv2.waitKey (0)
```

Load the target image with the shapes we are trying to match

```
target=cv2.imread ('shapestomatch.jpg')

gray=cv2.cvtColor (target, cv2.COLOR_BGR2GRAY)
```

Threshold both the images first before using *cv2.findContours*

```
ret,thresh1=cv2.threshold(template,127,255,0)

ret,thresh2=cv2.threshold(gray,127,255,0)
```

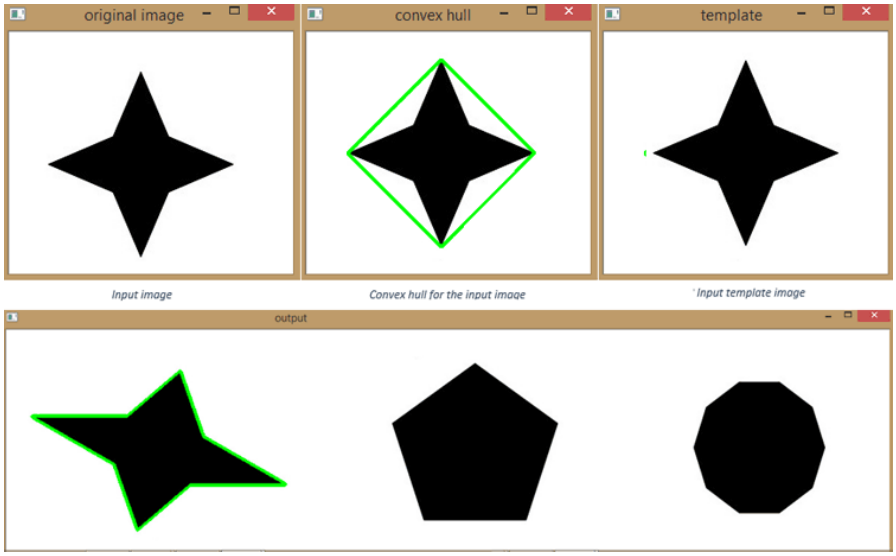
Find contours in template

```
_, contours, hierarhy=cv2.findContours (thresh1, cv2.RETR_CCOMP, cv2.CHAIN_APPROX_SIMPLE)

#we need to sort the contours by area so we can remove the largest contour which is
```

Image outline

```
sorted_contours=sorted (contours, key=cv2.contourArea, reverse=True)

#we extract the second largest contour which will be our template contour

tempelate_contour=contours [1]

#extracts the contours from the second target image

_, contours, hierarchy=cv2.findContours(thresh2,cv2.RETR_CCOMP,cv2.CHAIN_APPROX_SIMPLE)



for c in contours:

    #iterate through each contour in the target image and use cv2.matchShape to compare the contour shape

    match=cv2.matchShapes (tempelate_contour, c,1,0.0)

    print ("match")

    #if match value is less than 0.15

    if match<0.16:

        closest_contour=c

    else:

        closest_contour=[]



cv2.drawContours (target,[closest_contour],-1,(0,255,0),3)

cv2.imshow ('output', target)
```

```
cv2.waitKey(0)

cv2.destroyAllWindows ()
```



6. Identifying Shapes (circle, rectangle, triangle, square, star)

OpenCV can also be used for detecting different types of shapes automatically from the image. By using below code we will be able to detect circle, rectangle, triangle, square and stars from the image.

```
import cv2

import numpy as np
```

Load and then gray scale images

```
image=cv2.imread('shapes.jpg')

gray=cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)

cv2.imshow('identifying shapes',image)

cv2.waitKey(0)

ret, thresh=cv2.threshold(gray,127,255,1)
```

Extract contours

```
_,contours,hierarchy=cv2.findContours(thresh.copy(),cv2.RETR_LIST,cv2.CHAIN_APPROX_NONE)
```

For cnt in contours:

```
Get approximate polygons
  approx = cv2.approxPolyDP(cnt,0.01*cv2.arcLength(cnt,True),True)
  if len(approx)==3:
     shape_name="Triangle"
     cv2.drawContours(image,[cnt],0,(0,255,0),-1)
```

find contour center to place text at the center

```
     M=cv2.moments(cnt)
     cx=int(M['m10']/M['m00'])
     cy=int(M['m01']/M['m00'])
     cv2.putText(image,shape_name,(cx-50,cy),cv2.FONT_HERSHEY_SIMPLEX,1,(0,0,0),1)
  elif len(approx)==4:
     x,y,w,h=cv2.boundingRect(cnt)
     M=cv2.moments(cnt)
     cx=int(M['m10']/M['m00'])
     cy=int(M['m01']/M['m00'])
```

Check to see if that four sided polygon is square or rectangle

```
     #cv2.boundingRect return the left width and height in pixels, starting from the top
     #left corner, for square it would be roughly same
     if abs(w-h) <= 3:
        shape_name="square"
        #find contour center to place text at center
        cv2.drawContours(image,[cnt],0,(0,125,255),-1)
        cv2.putText(image,shape_name,(cx-50,cy),cv2.FONT_HERSHEY_SIMPLEX,1,(0,0,0),1)
     else:
```

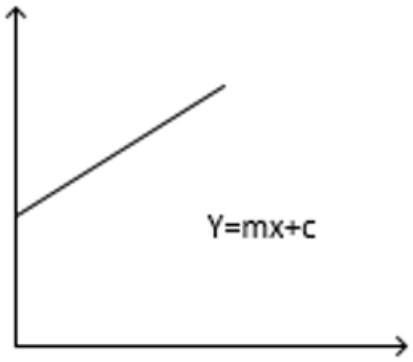
```
            shape_name="Reactangle"
            #find contour center to place text at center
            cv2.drawContours(image,[cnt],0,(0,0,255),-1)
            M=cv2.moments(cnt)
            cx=int(M['m10']/M['m00'])
            cy=int(M['m01']/M['m00'])
            cv2.putText(image,shape_name,(cx-50,cy),cv2.FONT_HERSHEY_SIMPLEX,1,(0,0,0),1)
    elif len(approx)==10:
        shape_name='star'
        cv2.drawContours(image,[cnt],0,(255,255,0),-1)
        M=cv2.moments(cnt)
        cx=int(M['m10']/M['m00'])
        cy=int(M['m01']/M['m00'])
        cv2.putText(image,shape_name,(cx-50,cy),cv2.FONT_HERSHEY_SIMPLEX,1,(0,0,0),1)
    elif len(approx)>=15:
        shape_name='circle'
        cv2.drawContours(image,[cnt],0,(0,255,255),-1)
        M=cv2.moments (cnt)
        cx=int(M['m10']/M['m00'])
        cy=int(M['m01']/M['m00'])
        cv2.putText(image,shape_name,(cx-50,cy),cv2.FONT_HERSHEY_SIMPLEX,1,(0,0,0),1)
    cv2.imshow('identifying shapes', image)
    cv2.waitKey(0)
cv2.destroyAllWindows()
```

7. Line Detection

Line detection is very much important concept in **OpenCV**, and has a promising use in the real world. Autonomous cars use line detection algorithms for the detection of lanes and roads.

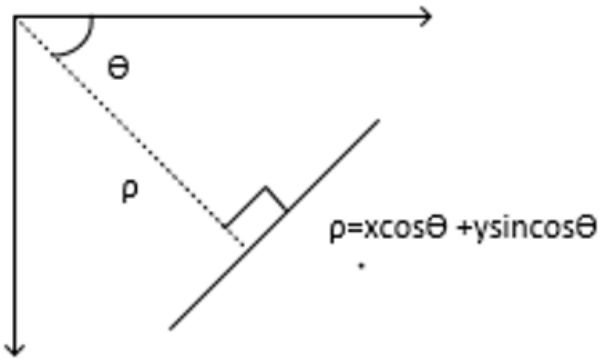
In line detection we will deal with two algorithms,

- **Hough Line Algorithm**
- **Probalistic Hough Line Algorithm**.



Normal equation of line

However, in OpenCV line is represented by another way.



Opencv's equation of line

The equation above ρ=xcosΘ +ysincosΘ is the OpenCV representation of the line, wherein ρ is the perpendicular distance of line from origin and Θ is the angle formed by the normal of this line to the origin (measured in radians, wherein 1pi radians/180 = 1 degree).

The OpenCV function for the detection of line is given as

cv2.HoughLines (binarized image, ρ accuracy, Θ accuracy, threshold), wherein threshold is minimum vote for it to be considered a line.

Now let's detect lines for a box image with the help of Hough line function of opencv.

```
import cv2

import numpy as np

image=cv2.imread('box.jpg')
```

Grayscale and canny edges extracted

```
gray=cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)

edges=cv2.Canny(gray,100,170,apertureSize=3)
```

Run Hough lines using rho accuracy of 1 pixel

```
#theta accuracy of (np.pi / 180) which is 1 degree

#line threshold is set to 240(number of points on line)

lines=cv2.HoughLines(edges, 1, np.pi/180, 240)

#we iterate through each line and convert into the format

#required by cv2.lines(i.e. requiring end points)

for i in range(0,len(lines)):

    for rho, theta in lines[i]:

        a=np.cos(theta)

        b=np.sin(theta)

        x0=a*rho

        y0=b*rho

        x1=int(x0+1000*(-b))

        y1=int(y0+1000*(a))

        x2=int(x0-1000*(-b))

        y2=int(y0-1000*(a))

        cv2.line(image,(x1,y1),(x2,y2),(0,255,0),2)



cv2.imshow ('hough lines',image)
```

```
cv2.waitKey(0)

cv2.destroyAllWindows()
```

Now let's repeat above line detection with other algorithm of probabilistic Hough line.

The idea behind probabilistic Hough line is to take a random subset of points sufficient enough for line detection.

The OpenCV function for probabilistic Hough line is represented as ***cv2.HoughLinesP (binarized image, ρ accuracy, Θ accuracy, threshold, minimum line length, max line gap)***

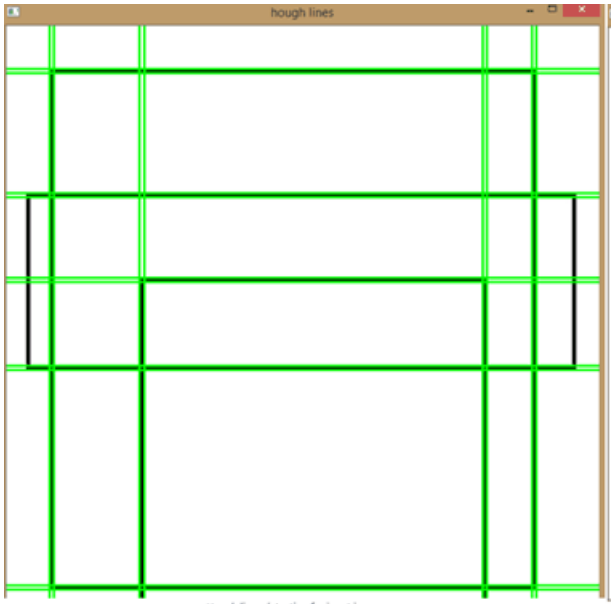
Now let's detect box lines with the help of probabilistic Hough lines.

```
import cv2

import numpy as np
```

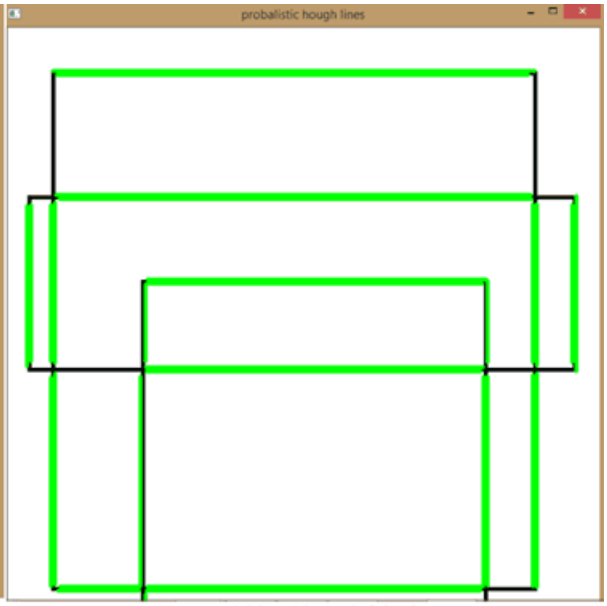
Grayscale and canny edges Extracted

```
image=cv2.imread('box.jpg')

gray=cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)



edges=cv2.Canny(gray,50,150,apertureSize=3)

#again we use the same rho and theta accuracies

#however, we specify a minimum vote(pts along line) of 100

#and min line length of 5 pixels and max gap between the lines of 10 pixels



lines=cv2.HoughLinesP(edges,1,np.pi/180,100,100,10)

for i in range(0,len(lines)):

    for x1,y1,x2,y2 in lines[i]:

        cv2.line(image,(x1,y1),(x2,y2),(0,255,0),3)



cv2.imshow('probalistic hough lines',image)
```

```
cv2.waitKey(0)

cv2.destroyAllWindows
```



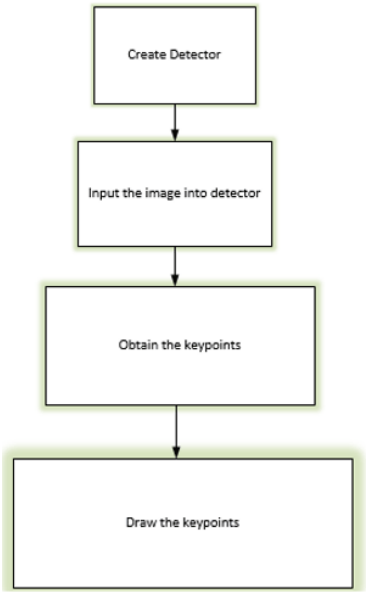
Hough lines detection for input image



Probalistic Hough line detection for input image

8. Blob Detection

Blobs can be described as a group of connected pixels that all share a common property. The method to use OpenCV blob detector is described through this flow chart.



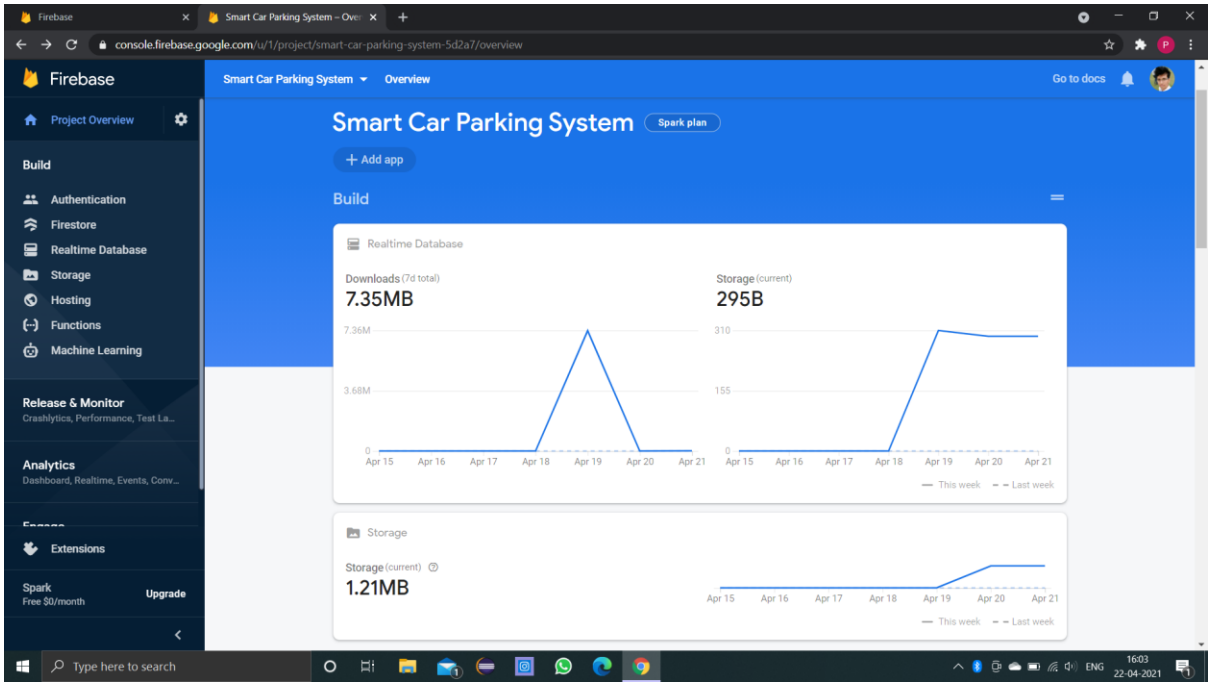
Flow chart for blob detection

1.2.3 Firebase (Back-end)

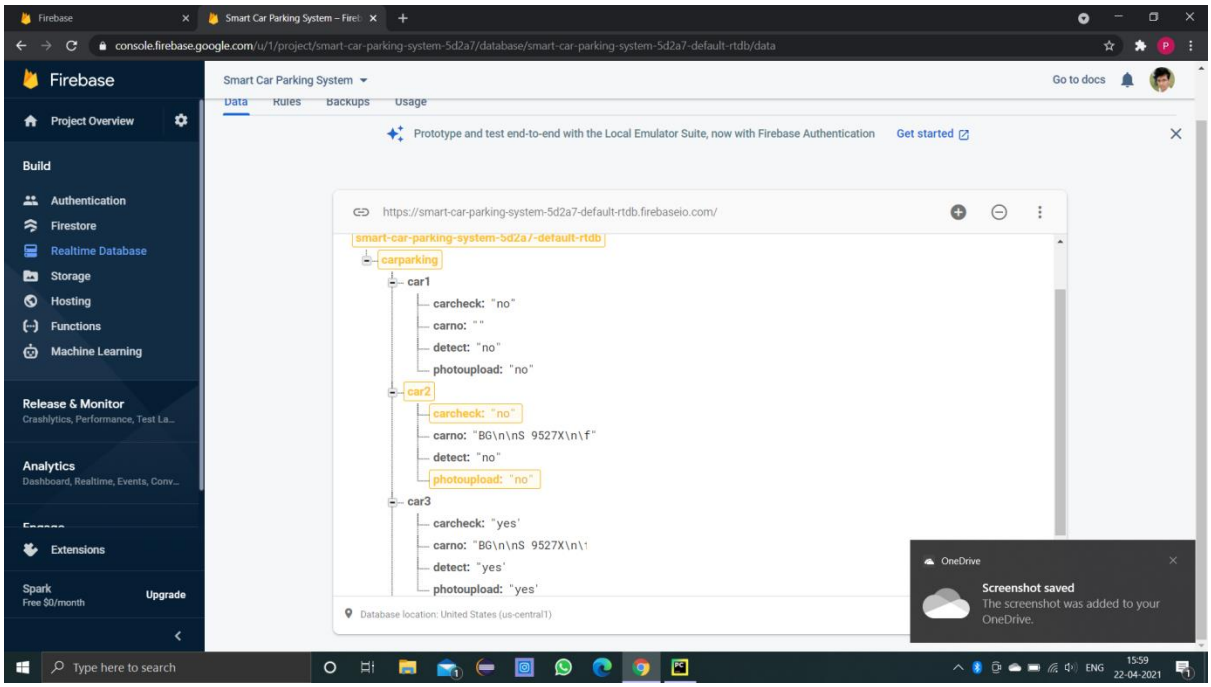
Cloud Firestore is a flexible, scalable database for mobile, web, and server development from Firebase and Google Cloud Platform. In order to structure your data, you define collections (similar to tables in SQL) which contain documents (similar to rows). Each document contains fields that contain the actual data. You can reference an individual document using its unique path, or you can query a collection for documents whose fields contain the data you're looking for. Using the Cloud console, you can browse data in the Cloud Firestore database in your project Like Realtime Database, Cloud Firestore uses data synchronization to update data on any connected device. However, it's also designed to make simple, one-time fetch queries efficiently.

How Cloud Firestore Works :-

Like Realtime Database, Cloud Firestore uses data synchronization to update data on any connected device. However, it's also designed to make simple, one-time fetch queries efficiently. Cloud Firestore caches data that your app is actively using, so the app can write, read, listen to, and query data even if the device is offline. When the device comes back online, Cloud Firestore synchronizes any local changes back to Cloud Firestore.



Asynchronous listeners: Data stored in a Firebase Realtime Database is retrieved by attaching an asynchronous listener to a database reference. The listener is triggered once for the initial state of the data and again anytime the data changes. An event listener may receive several different types of events.



Blocking reads: Data stored in a Firebase Realtime Database is retrieved by invoking a blocking method on a database reference, which returns the data stored at the reference. Each method call is a onetime operation. That means the SDK does not register any callbacks that listen to subsequent data updates. This model of data retrieval is supported in Python and Go Admin SDKs

2. Problem Definition

Nowadays most of the car parks require user's initiative to search for empty space to park their car. This will cause problems when it is too many cars and it makes them wasting their time and energy. One of the factors that contribute to this problem is because of lack of information that given at parking lot. So, one system has to be design to solve this parking problem which will include the information interface criteria.

Nowadays in most of the countries, Parking Information and Guidance (PGI) system have been put into practice in Europe, United State, Japan, and China. Number of cars on the road is increasing while parking spaces are becoming increasingly scarce. Usually during school break or holiday, the numbers of cars that use the parking space in the shopping complex become higher compared to during working day. This will make the parking space become full and the driver need to drive slowly in order for them to check for the free space parking in the shopping complex. This is time consuming and people will become more impatient. Besides that, there are also problems of the parking space which are located far from the entering zone or the building or destination. This situation makes the drivers to choose the nearest parking space as they do not want to walk far.

As the fuel price is keep increase nowadays, users will try their best to save their vehicle energy. At the peak hour, they need to wait for a long time at the entrance gate before they can find the empty parking space. So the users will waste their time and energy to find a free space. Last but not least, the major issue of the parking system is the insufficient parking spaces provided for the user. This may cause by the improper planning by developers of the places. When the parking bay is on its peak hour, the user will search the parking space at the same area again and again before they found it, only if they are lucky. Sometimes, the car park management did not put a notice that the area was full and no more space for parking.

3. Objectives

The objectives of this research project are as follows: -

1- To design an automatic parking system that provides information of empty parking spaces in the indoor parking area .

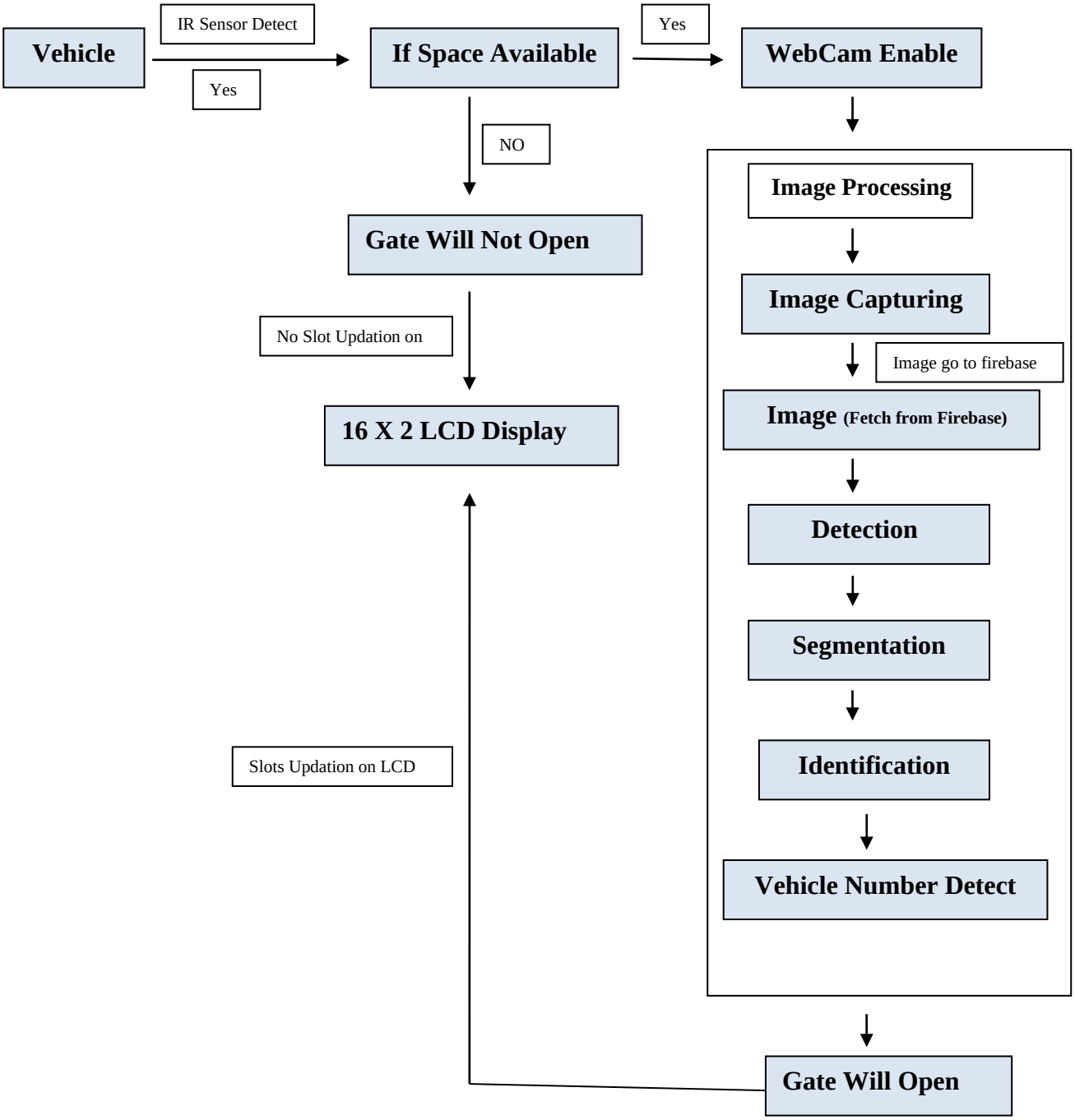
2- To develop a prototype of smart parking system using Raspberry Pi micro controller, ir sensor and servo motor

3- To evaluate the performance of developed prototype to provide information and facilitate the users to the location of empty parking spaces.

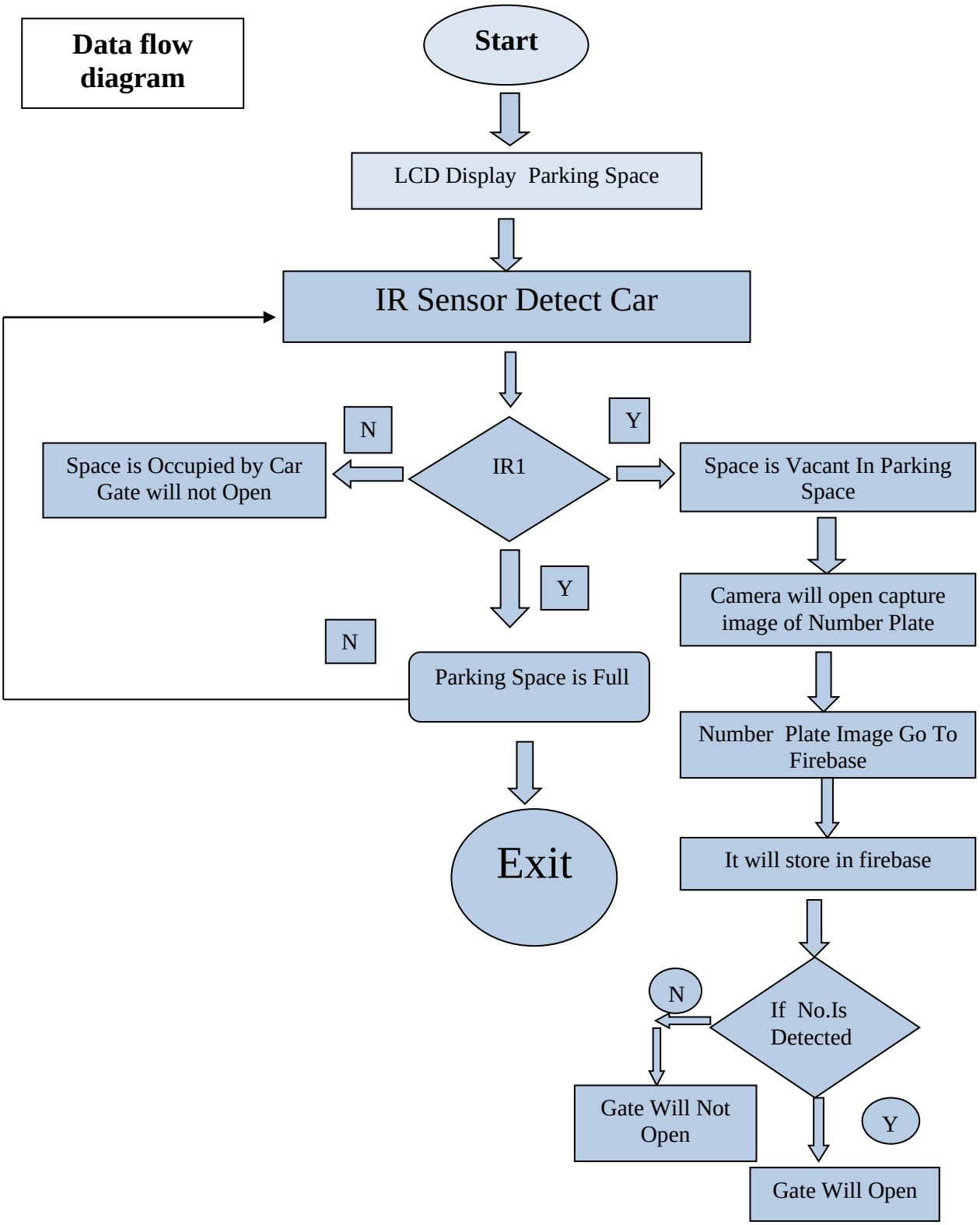
4- To detect the vehicle number plate and send the data to the firebase

4. Software & Hardware Requirement Analysis

System Block Diagram

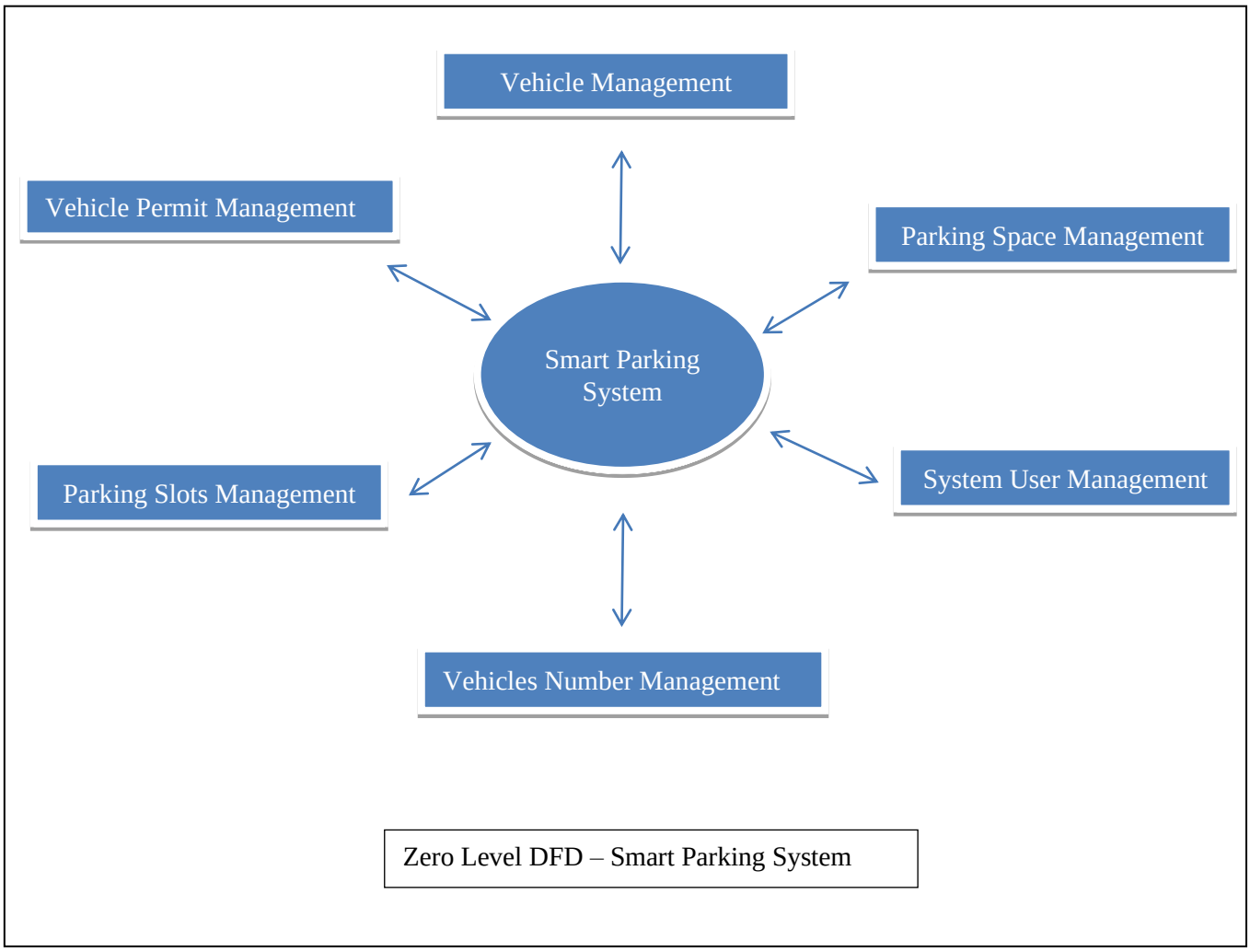


Data flow diagram



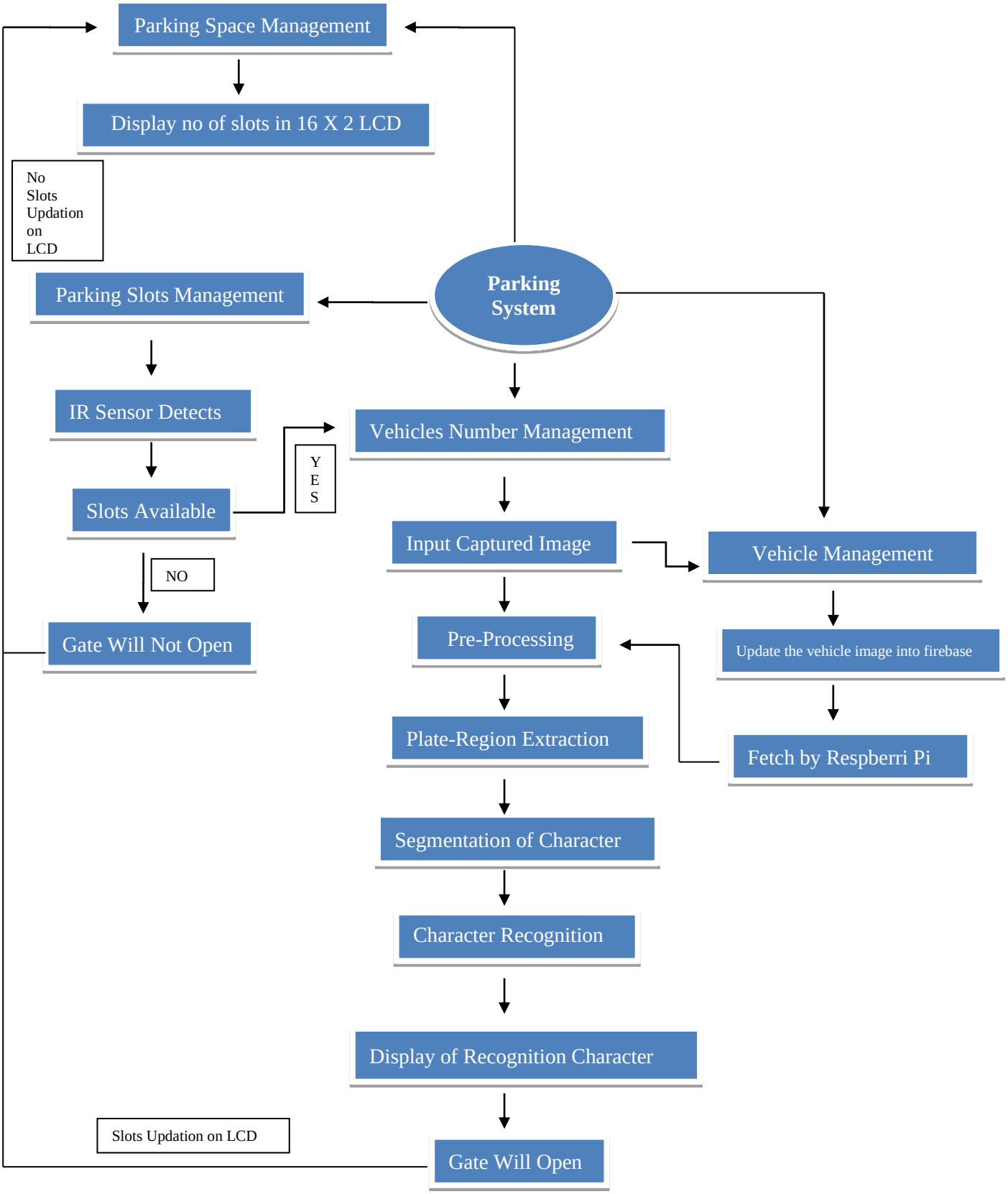
Zero-level DFD

This is the Zero Level DFD of Online Car Parking System, where we have elaborated the high level process of Car Parking. It's a basic overview of the whole Online Car Parking System or process being analyzed or modeled. It is designed to be an at-a-glance view of Parking Fees, Car owner and car Number showing the system as a single high-level process, with its relationships to external entities of Car, Parking Space and Parking Fees in zero level DFD of Online Car Parking System.



Zero Level DFD – Smart Parking System

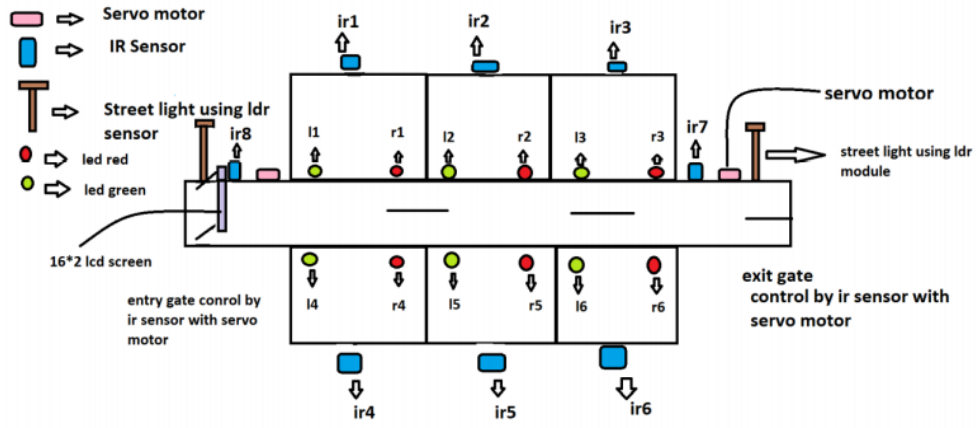
First Level DFD



5. Implementation Details

Raspberry Pi will control the complete process and also send the parking availability information to 16X2 LCD Display so that it can be monitored from outside the Parking System. IR sensors are used at entry and exit gate to detect the presence of car and automatically open or close the gate. IR Sensor is used to detect any object by sending and receiving the IR rays. Servos will act as entry and exit gate and they rotate to open or close the gate. Finally sensor is used to detect if the parking slot is available or occupied and send the data to microcontroller accordingly.

When car enters, driver see the LCD screen that is fixed before the barrier of the parking this LCD 16X2 display the vacant parking slot .If there is no parking slot is available then it display 0 slot and our parking barrier (connect to the IR sensor and servo motor that is connect to the Raspberry Pi) cannot open in this condition. If there is vacant space available then IR sensor detect the vehicle. After vehicle detection WebCam enables then image of vehicle will captured using the WebCam and send this image to Google Firebase and then the image is fetchd from the firebase and then the image processing is done on the captured image. If vehicle number detected using OCR then barrier will open and car will enter in parking. In each parking slot there is a IR sensor that detect car is available in the slot or not and display combine data of all IR sensor on the LCD screen and display the data on computer screen.



Steps involved in License Plate Recognition:

1. License Plate Detection: The first step is to detect the License plate from the car. We will use the contour option in OpenCV to detect for rectangular objects to find the number plate. The accuracy can be improved if we know the exact size, color and approximate location of the number plate. Normally the detection algorithm is trained based on the position of camera and type of number plate used in that particular country. This gets trickier if the image does not even have a car, in this case we will an additional step to detect the car and then the license plate.

2. Character Segmentation: Once we have detected the License Plate we have to crop it out and save it as a new image. Again this can be done easily using OpenCV.

3. Character Recognition: Now, the new image that we obtained in the previous step is sure to have some characters (Numbers/Alphabets) written on it. So, we can perform OCR (Optical Character Recognition) on it to detect the number

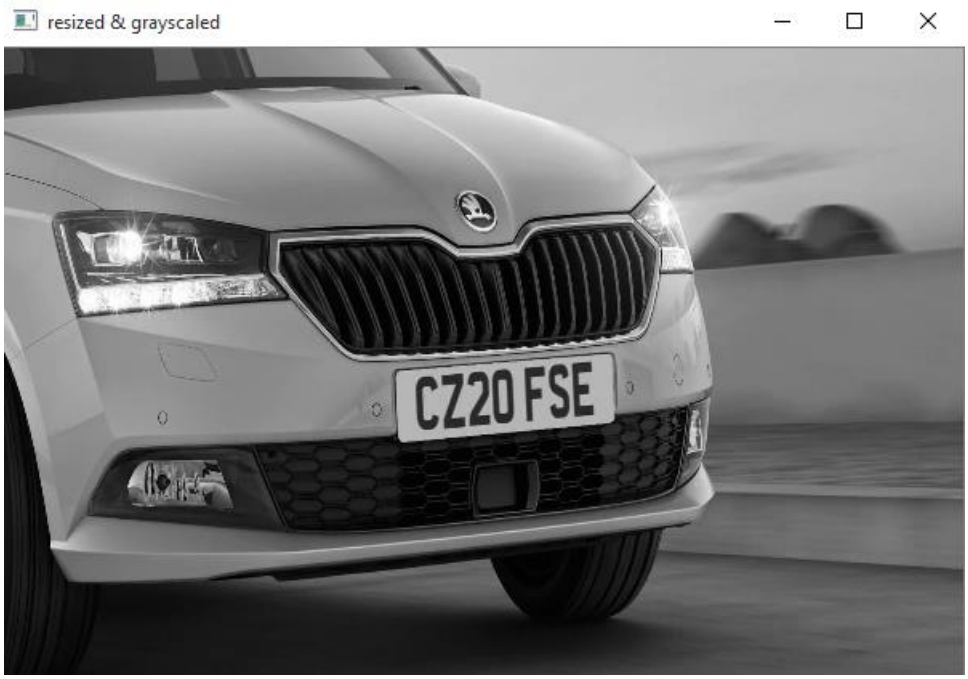
License Plate Detection:

Let's take a sample image of a car and start with detecting the License Plate on that car. We will then use the same image for Character Segmentation and Character Recognition as well.
The test image that we have used for demo is shown below.

Step 1: Resize the image to the required size and then grayscale it. The code for the same is given below

```
Img = cv2.resize(img, (620,480) )
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) #convert to grey scale
```

Resizing we help us to avoid any problems with bigger resolution images, make sure the number plate still remains in the frame after resizing. Gray scaling is common in all image processing steps. This speeds up other following process sine we no longer have to deal with the color details when processing an image. The image would be transformed something like this when this step is done.



Step 2: Every image will have useful and useless information, in this case for us only the license plate is the useful information the rest are pretty much useless for our program. This useless information is called noise. Normally using a bilateral filter (Blurring) will remove the unwanted details from an image. The code for the same is blurred

```
Gray = cv2.bilateralFilter (gray, 13, 15, 15)
```

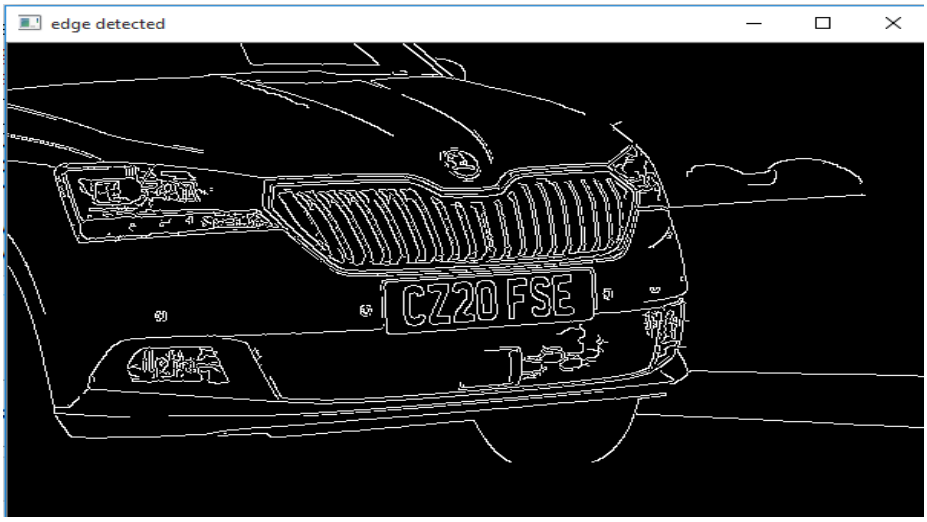
Syntax is ***destination image = cv2.bilateralFilter (source_image, diameter of pixel, sigmaColor, sigma Space).*** You can increase the sigma color and sigma space from 15 to higher values to blur out more background information, but be careful that the useful part does not get blurred. The output image is shown below; as you can see the background details (tree and building) are blurred in

this image. This way we can avoid the program from concentrating on these regions later.

Step 3: The next step is interesting where we perform **edge detection**. There are many ways to do it; the easiest and popular way is to use the **canny edge method from OpenCV.** The line to do the same is shown below

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

The syntax will be *destination image = cv2.Canny (source_image, threshold Value 1, threshold Value 2).* The Threshold Vale 1 and Threshold Value 2 are the minimum and maximum threshold values. Only the edges that have an intensity gradient more than the minimum threshold value and less than the maximum threshold value will be displayed. The resulting image is shown below



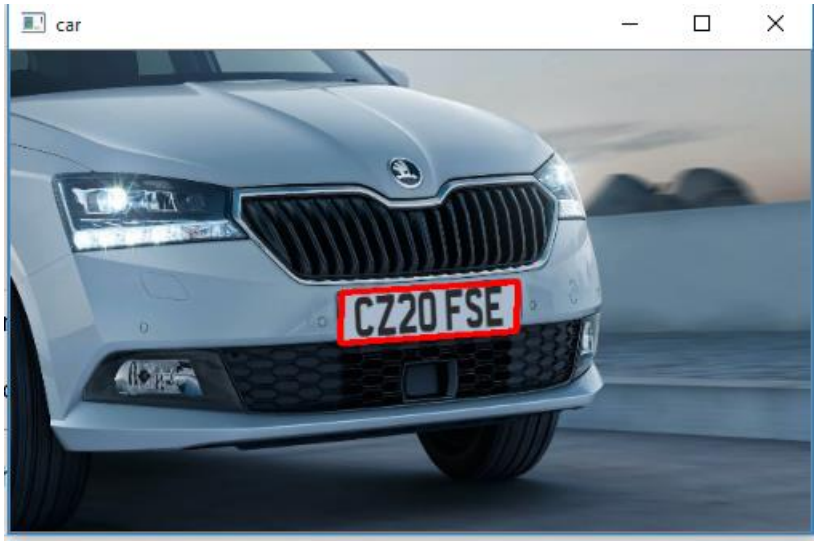
Step 4: Now we can start **looking for contours on our image**

```
contours=cv2.findContours(edged.copy(),cv2.RETR_TREE,
                                cv2.CHAIN_APPROX_SIMPLE)
contours = imutils.grab_contours(contours)
contours = sorted(contours,key=cv2.contourArea, reverse = True)[:10]
screenCnt = None
```

Once the counters have been detected we sort them from big to small and consider only the first 10 results ignoring the others. In our image the counter could be anything that has a closed surface but of all the obtained results the license plate number will also be there since it is also a closed surface.
To filter the license plate image among the obtained results, we will loop though all the results and check which has a rectangle shape contour with four sides and closed figure. Since a license plate would definitely be a rectangle four sided figure.

```
for c in cnts:
    # approximate the contour
    peri = cv2.arcLength(c, True)
    approx. = cv2.approxPolyDP(c, 0.018 * peri, True)
    # if our approximated contour has four points, then
    # we can assume that we have found our screen
    if len(approx.) == 4:
        screenCnt = approx.
        break
```

Once we have found the right counter we save it in a variable called *screenCnt* and then draw a rectangle box around it to make sure we have detected the license plate correctly.


car
CZ20 FSE

Step 5: Now that we know where the number plate is, the remaining information is pretty much useless for us. So we can **proceed with masking the entire picture except for the place where the number plate is.** The code to do the same is shown below

```
# Masking the part other than the number plate
mask = np.zeros(gray.shape,np.uint8)
new_image = cv2.drawContours(mask,[screenCnt],0,255,-1,)
new_image = cv2.bitwise_and(img,img,mask=mask)
```

The masked new image will appear something like below:

2. Character Segmentation

The next step in **Number Plate Recognition** is to segment the license plate out of the image by **cropping it and saving it as a new image.** We can then use this image to detect the character in it. The code to crop the roi (Region of interest) image form the main image is shown below

```
# Now crop
(x, y) = np.where(mask == 255)
(topx, topy) = (np.min(x), np.min(y))
(bottomx, bottomy) = (np.max(x), np.max(y))
Cropped = gray[topx:bottomx+1, topy:bottomy+1]
```

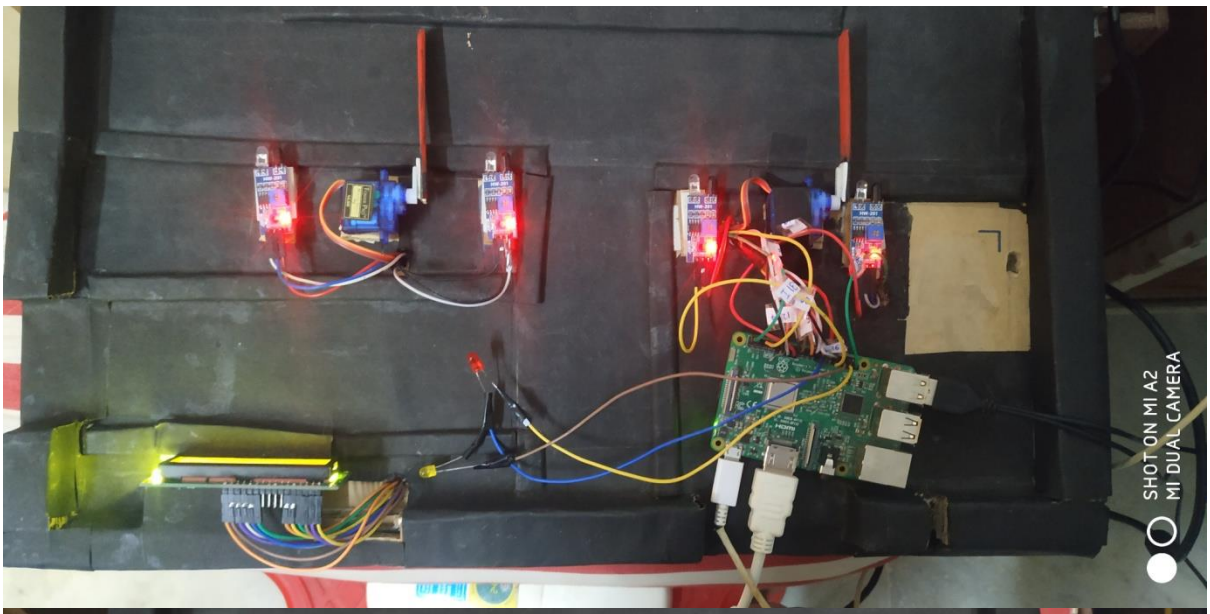
The resulting image is shown below. Normally added to cropping the image, we can also gray it and edge it if required. This is done to improve the character recognition in next step. However I found that it works fine even with the original image.

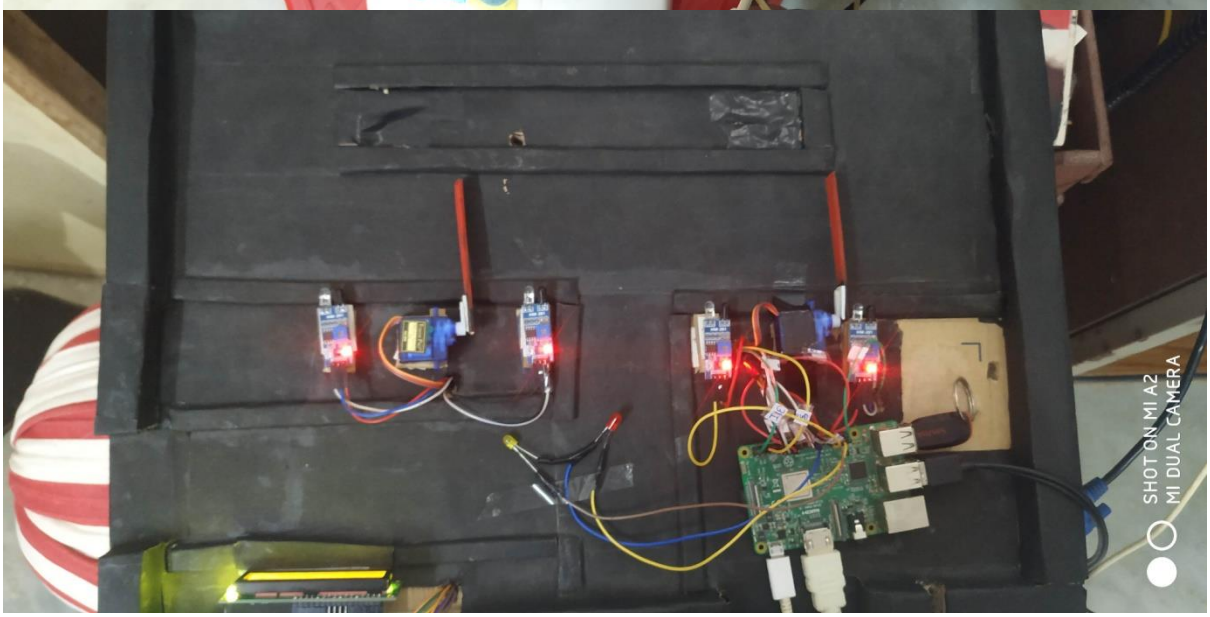
3. Character Recognition

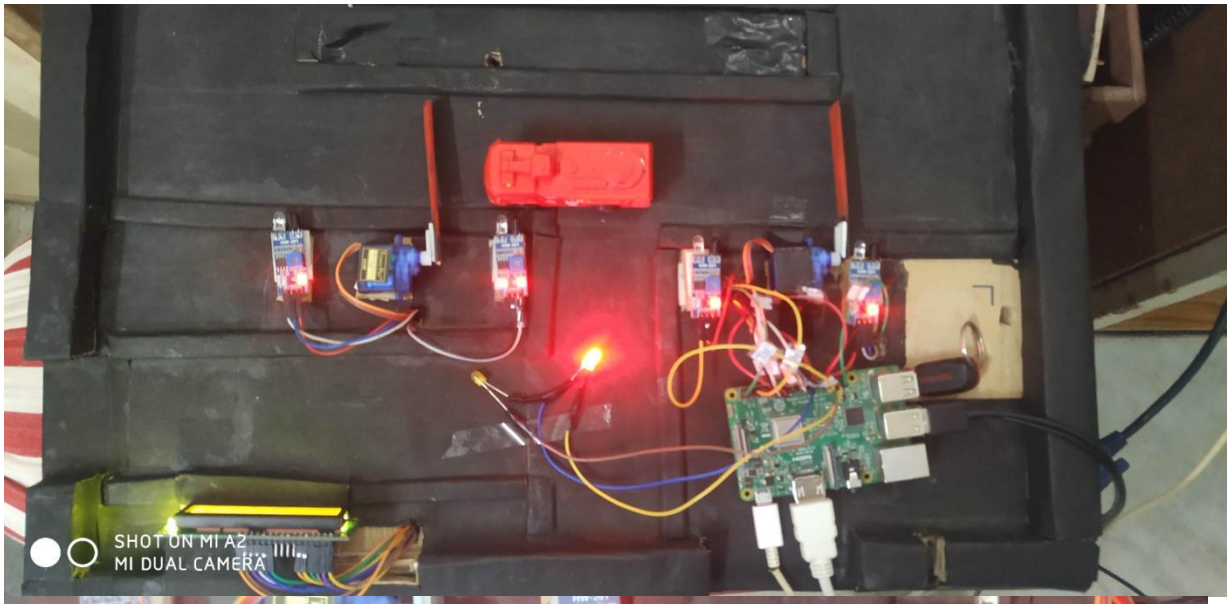
The Final step in this **Number Plate Recognition** is to actually **read the number plate information from the segmented image**. We will use the ***pytesseract*** package to read characters from image, just like we did in previous tutorial. The code for the same is given below

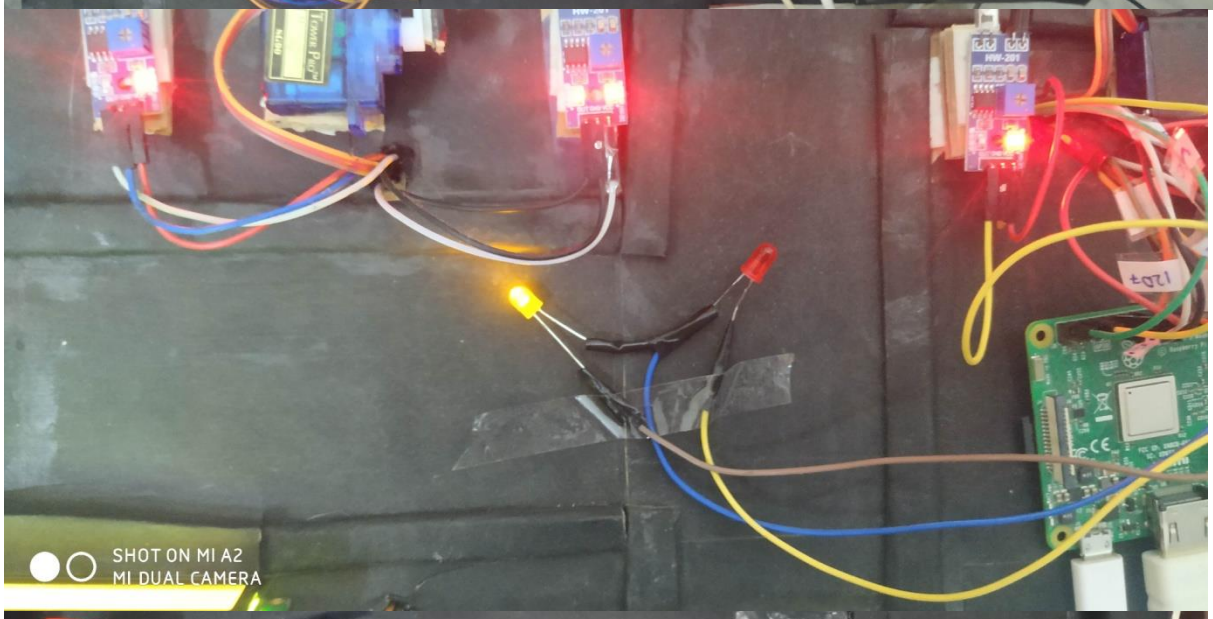
```
#Read the number plate
text = pytesseract.image_to_string(Cropped, config='--psm 11')
print("Detected license plate Number is:",text)
```

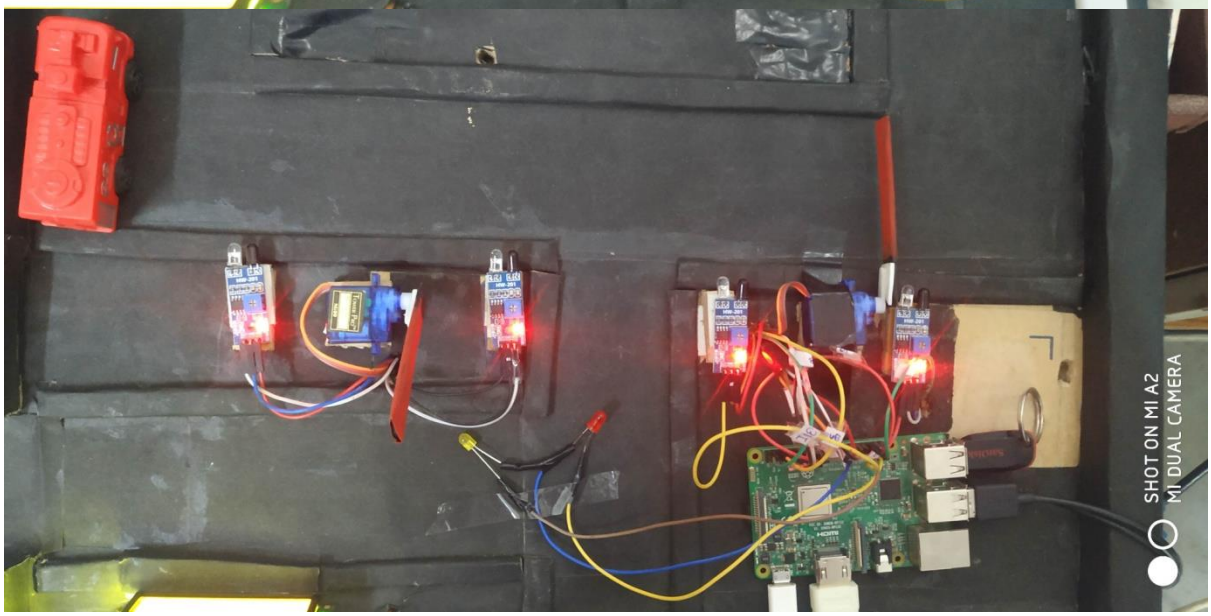
Some Screenshots of the project:






SHOT ON MI A2
MI DUAL CAMERA


SHOT ON MI A2
MI DUAL CAMERA


SHOT ON MI A2
MI DUAL CAMERA

```python
import RPi.GPIO as GPIO
from time import sleep
from google.cloud import storage
from firebase import firebase
import imutils
import numpy as np
import pytesseract
import cv2
import time
import os
from PIL import Image

#==============================
# GPIO to LCD mapping
LCD_RS = 26 # Pi pin 26
LCD_E = 24 # Pi pin 24
LCD_D4 = 22 # Pi pin 22
LCD_D5 = 18 # Pi pin 18
LCD_D6 = 16 # Pi pin 16
LCD_D7 = 12 # Pi pin 12
```

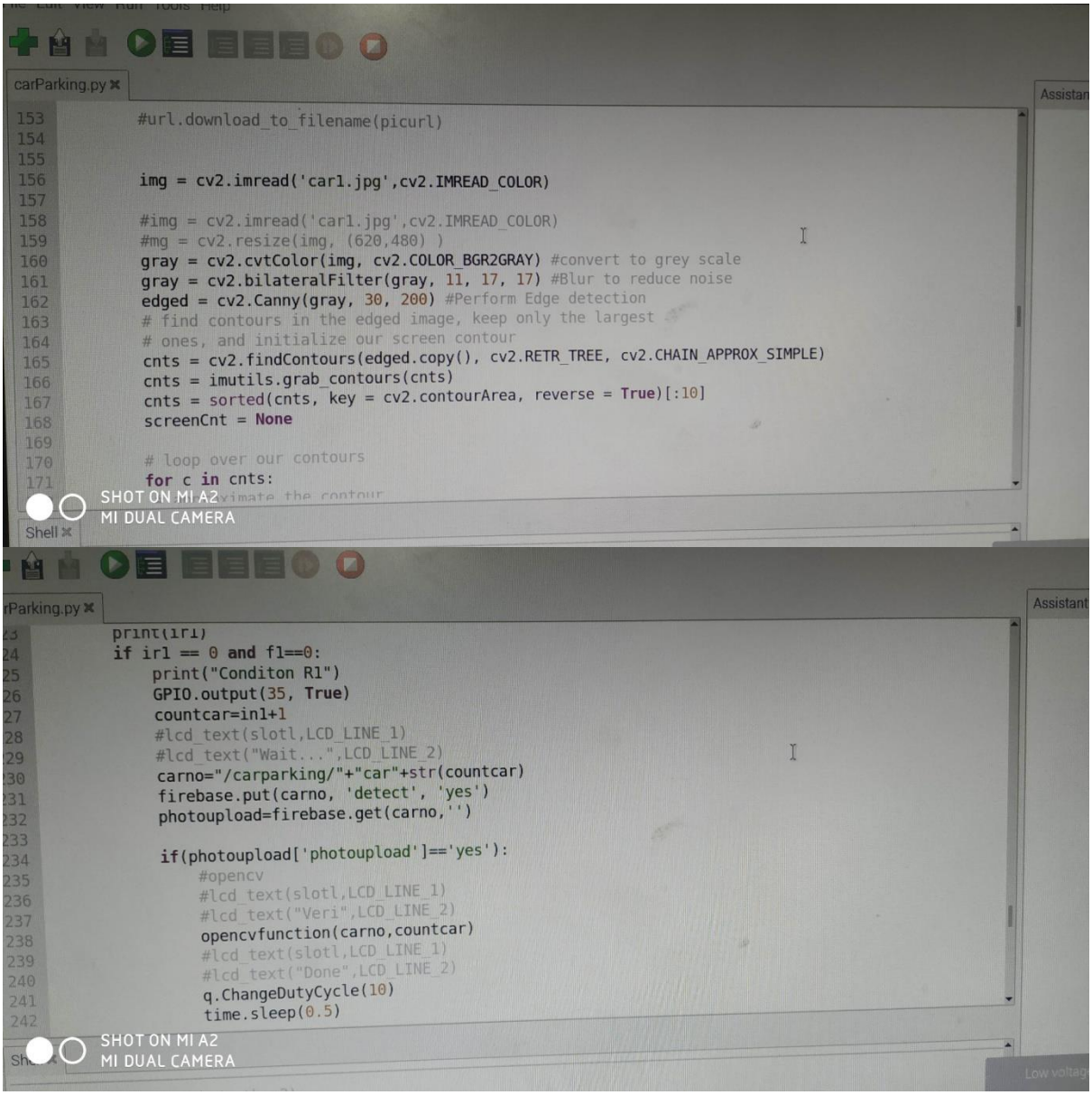
Shell


```python
GPIO.setup(servoPin1,GPIO.OUT)
q=GPIO.PWM(servoPin1,50)
q.start(2.5)
GPIO.setup(11,GPIO.IN)
GPIO.setup(13,GPIO.IN)
GPIO.setup(15,GPIO.IN)
GPIO.setup(31,GPIO.IN)
GPIO.setup(35,GPIO.OUT) #stop
GPIO.setup(33,GPIO.OUT)  #g



# Use BCM GPIO numbers
GPIO.setup(LCD_E, GPIO.OUT) # Set GPIO's to output mode
GPIO.setup(LCD_RS, GPIO.OUT)
GPIO.setup(LCD_D4, GPIO.OUT)
GPIO.setup(LCD_D5, GPIO.OUT)
GPIO.setup(LCD_D6, GPIO.OUT)
GPIO.setup(LCD_D7, GPIO.OUT)
```


Parking.py

```python
#slot_left=0
exitt=0
countcar=0
slotl=""

#====================================Variable

os.environ['GOOGLE_APPLICATION_CREDENTIALS'] = '/home/pi/SmartCarParking/smart-car-parking-system-5d2a7-fir
firebase = firebase.FirebaseApplication('https://smart-car-parking-system-5d2a7-default-rtdb.firebaseio.com
client = storage.Client()
bucket = client.get_bucket('smart-car-parking-system-5d2a7.appspot.com')



def lcd_init():
    lcd_write(0x33,LCD_CMD) # Initialize
    lcd_write(0x32,LCD_CMD) # Set to 4-bit mode
    lcd_write(0x06,LCD_CMD) # Cursor move direction
    lcd_write(0x0C,LCD_CMD) # Turn cursor off
    lcd_write(0x28,LCD_CMD) # 2 line display
```


Python 3.7.3 (/usr/bin/python3)



carParking.py

```
153         #url.download_to_filename(picurl)
154
155
156         img = cv2.imread('car1.jpg',cv2.IMREAD_COLOR)
157
158         #img = cv2.imread('car1.jpg',cv2.IMREAD_COLOR)
159         #mg = cv2.resize(img, (620,480) )
160         gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) #convert to grey scale
161         gray = cv2.bilateralFilter(gray, 11, 17, 17) #Blur to reduce noise
162         edged = cv2.Canny(gray, 30, 200) #Perform Edge detection
163         # find contours in the edged image, keep only the largest
164         # ones, and initialize our screen contour
165         cnts = cv2.findContours(edged.copy(), cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
166         cnts = imutils.grab_contours(cnts)
167         cnts = sorted(cnts, key = cv2.contourArea, reverse = True)[:10]
168         screenCnt = None
169
170         # loop over our contours
171         for c in cnts:
```

SHOT ON MI A2
MI DUAL CAMERA

Shell

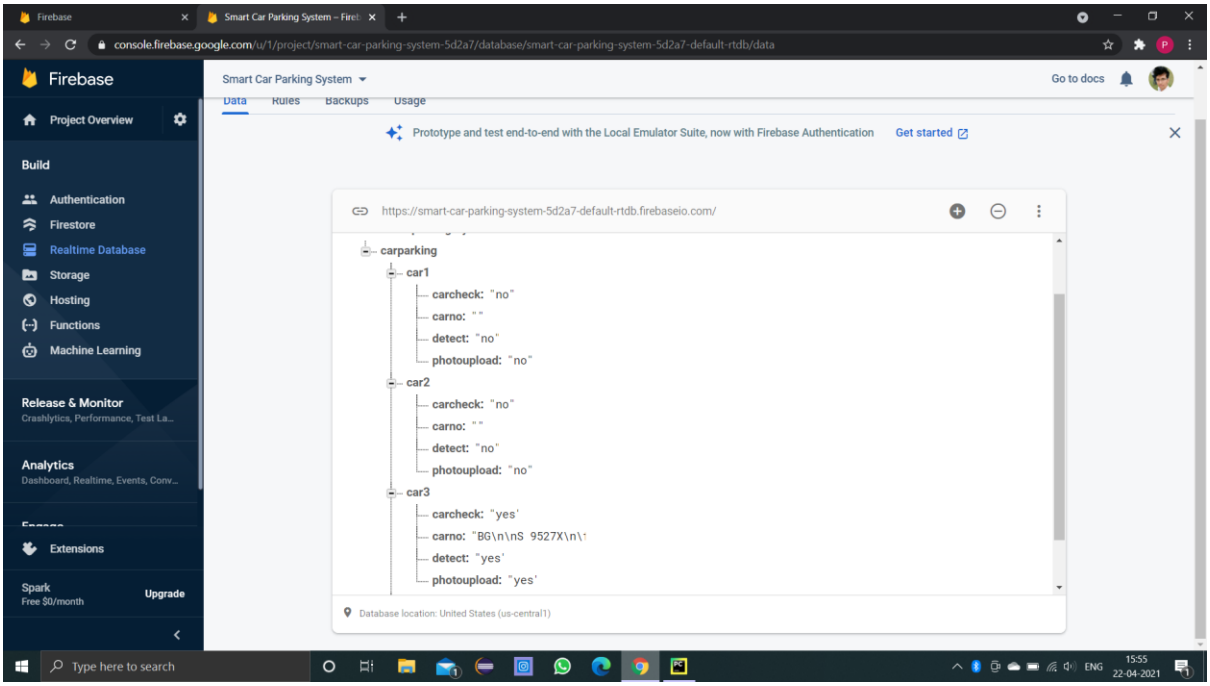
rParking.py

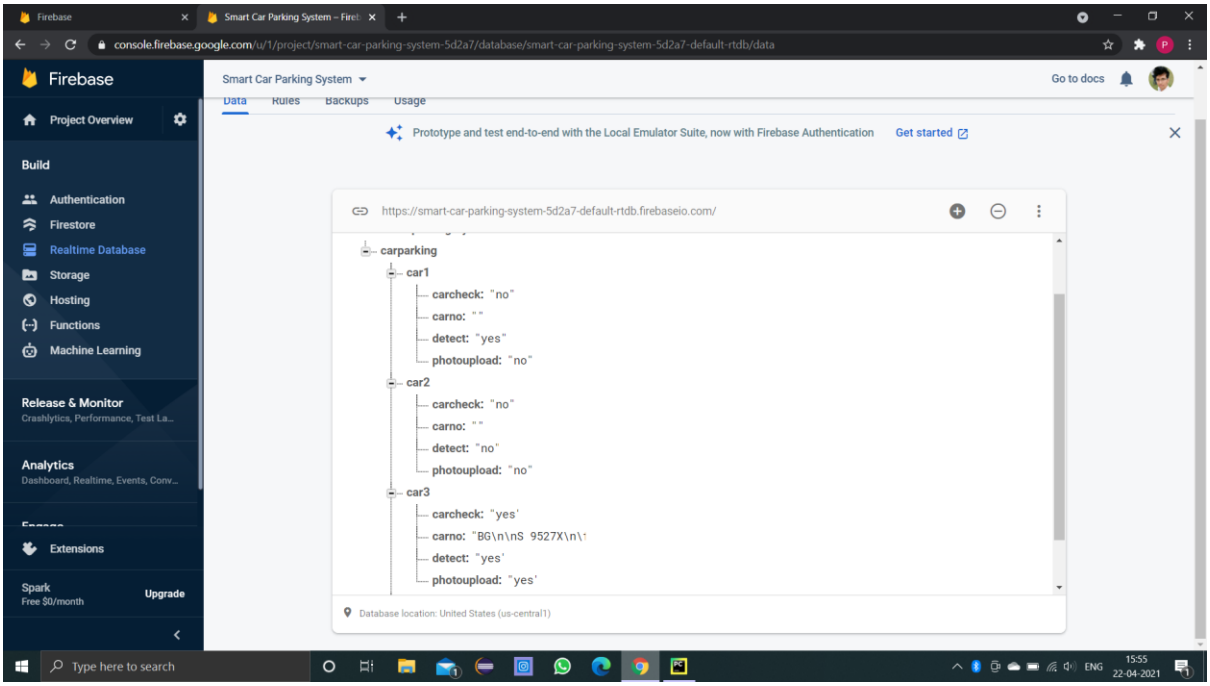
Assistant

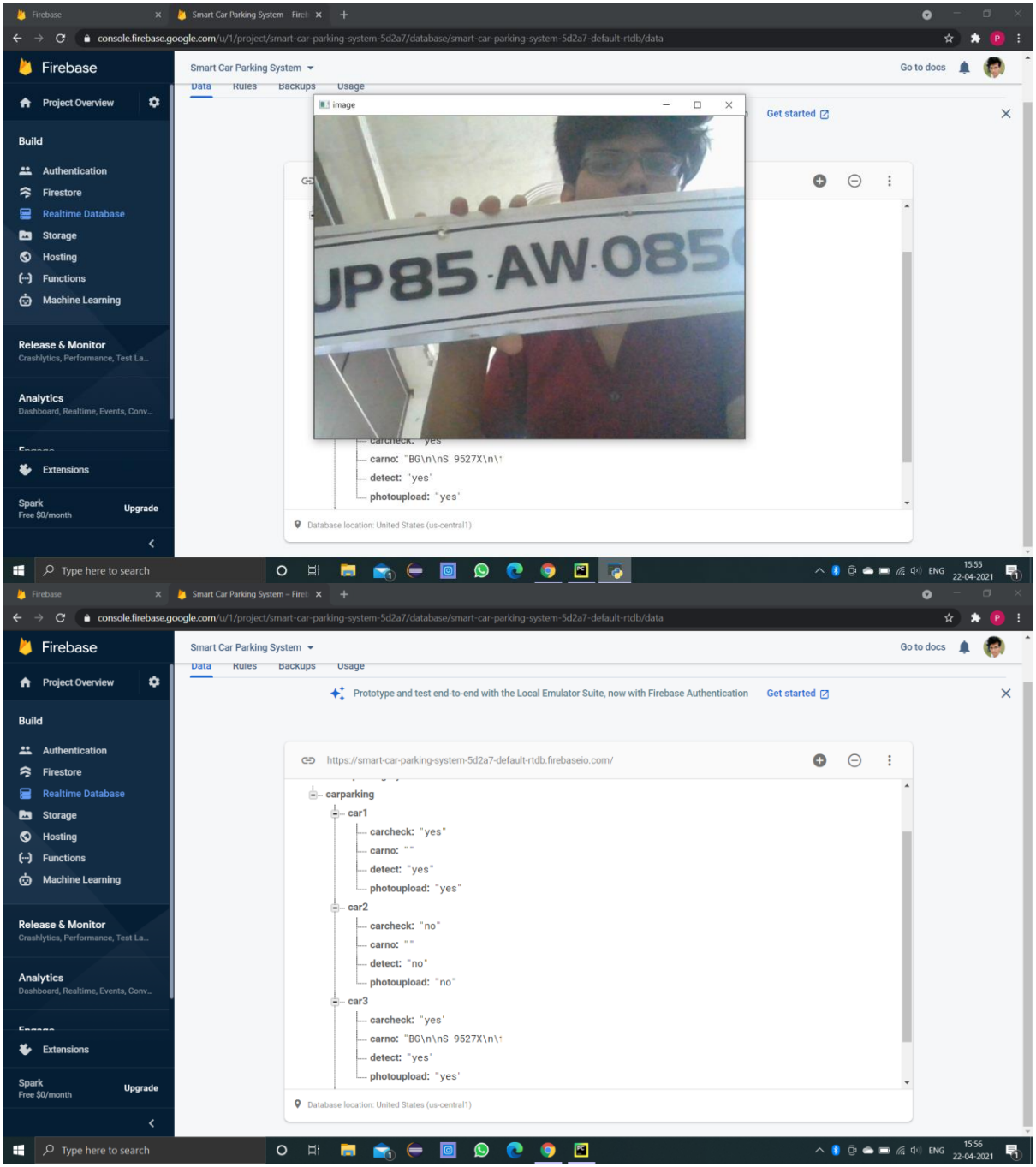
```
23          print(ir1)
24          if ir1 == 0 and f1==0:
25              print("Conditon R1")
26              GPIO.output(35, True)
27              countcar=in1+1
28              #lcd_text(slotl,LCD_LINE_1)
29              #lcd_text("Wait...",LCD_LINE_2)
30              carno="/carparking/"+"car"+str(countcar)
31              firebase.put(carno, 'detect', 'yes')
32              photoupload=firebase.get(carno,'')
33
34              if(photoupload['photoupload']=='yes'):
35                  #opencv
36                  #lcd_text(slotl,LCD_LINE_1)
37                  #lcd_text("Veri",LCD_LINE_2)
38                  opencvfunction(carno,countcar)
39                  #lcd_text(slotl,LCD_LINE_1)
40                  #lcd_text("Done",LCD_LINE_2)
41                  q.ChangeDutyCycle(10)
42                  time.sleep(0.5)
```

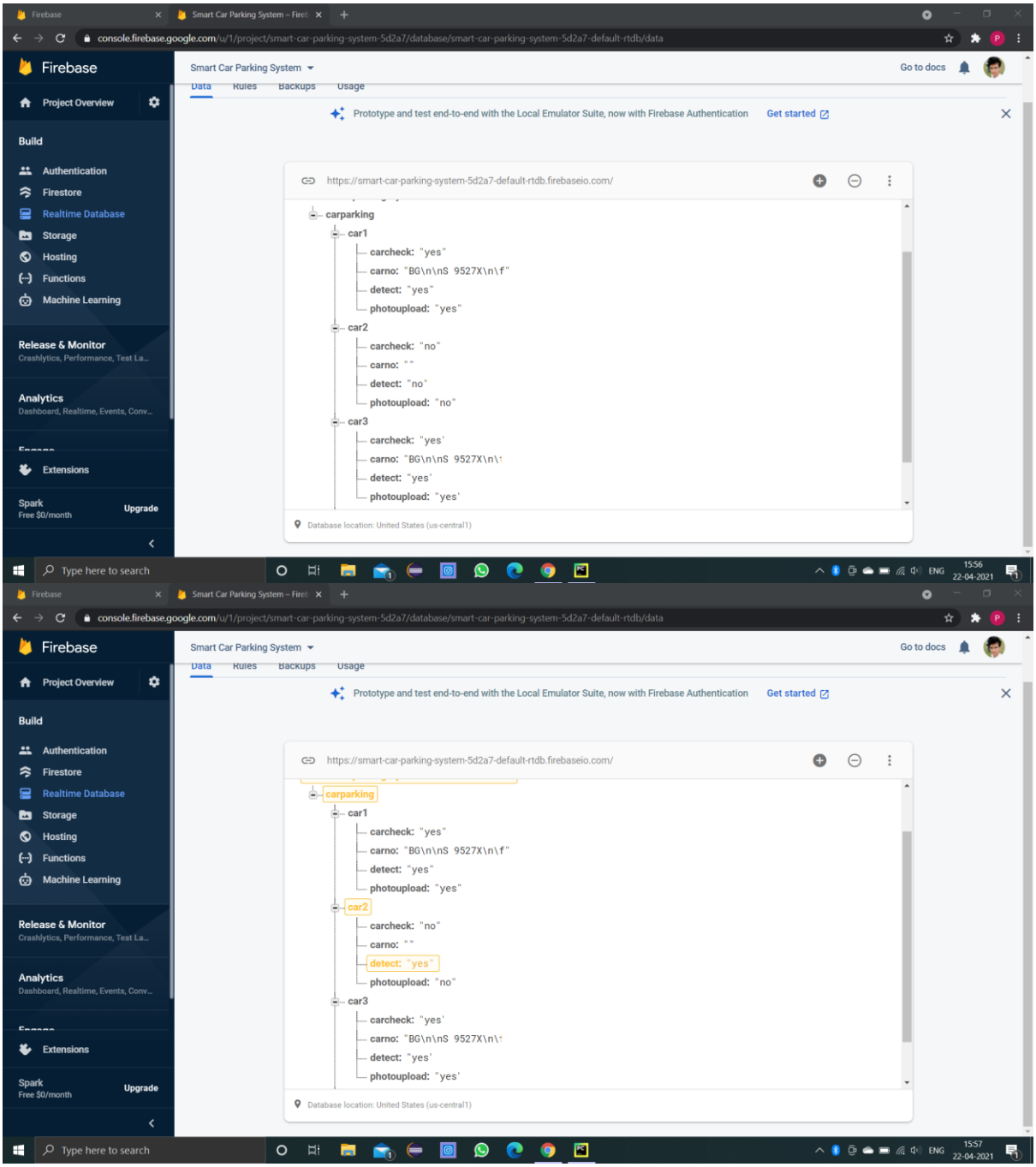
SHOT ON MI A2
MI DUAL CAMERA

Low voltage

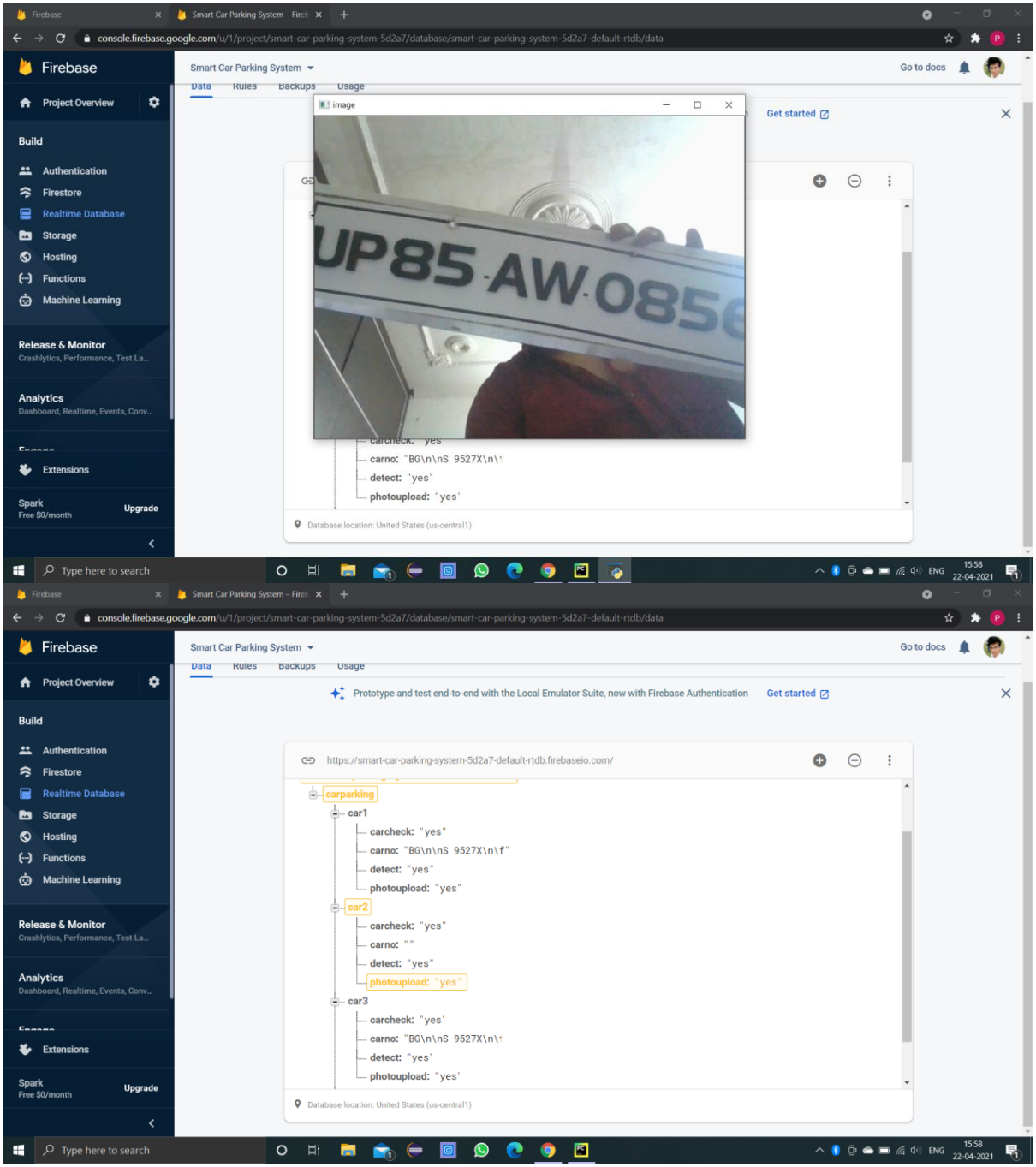

Firebase
Smart Car Parking System – Firebase
console.firebase.google.com/u/1/project/smart-car-parking-system-5d2a7/database/smart-car-parking-system-5d2a7-default-rtdb/data
Smart Car Parking System
Go to docs
Project Overview
Build
Authentication
Firestore
Realtime Database
Storage
Hosting
Functions
Machine Learning
Release & Monitor
Crashlytics, Performance, Test La...
Analytics
Dashboard, Realtime, Events, Conv...
Extensions
Spark
Free $0/month
Upgrade
Data
Rules
Backups
Usage
Prototype and test end-to-end with the Local Emulator Suite, now with Firebase Authentication
Get started
https://smart-car-parking-system-5d2a7-default-rtdb.firebaseio.com/
carparking
car1
carcheck: "no"
carno: ""
detect: "no"
photoupload: "no"
car2
carcheck: "no"
carno: ""
detect: "no"
photoupload: "no"
car3
carcheck: "yes"
carno: "BG\n\nS 9527X\n\
detect: "yes"
photoupload: "yes"
Database location: United States (us-central1)
Type here to search
ENG
15:55
22-04-2021


Firebase
Smart Car Parking System – Firebase
console.firebase.google.com/u/1/project/smart-car-parking-system-5d2a7/database/smart-car-parking-system-5d2a7-default-rtdb/data
Smart Car Parking System
Go to docs
Project Overview
Build
Authentication
Firestore
Realtime Database
Storage
Hosting
Functions
Machine Learning
Release & Monitor
Crashlytics, Performance, Test La...
Analytics
Dashboard, Realtime, Events, Conv...
Extensions
Spark
Free $0/month
Upgrade
Data
Rules
Backups
Usage
Prototype and test end-to-end with the Local Emulator Suite, now with Firebase Authentication
Get started
https://smart-car-parking-system-5d2a7-default-rtdb.firebaseio.com/
carparking
car1
carcheck: "no"
carno: ""
detect: "yes"
photoupload: "no"
car2
carcheck: "no"
carno: ""
detect: "no"
photoupload: "no"
car3
carcheck: "yes"
carno: "BG\n\nS 9527X\n\
detect: "yes"
photoupload: "yes"
Database location: United States (us-central1)
Type here to search
ENG
15:55
22-04-2021

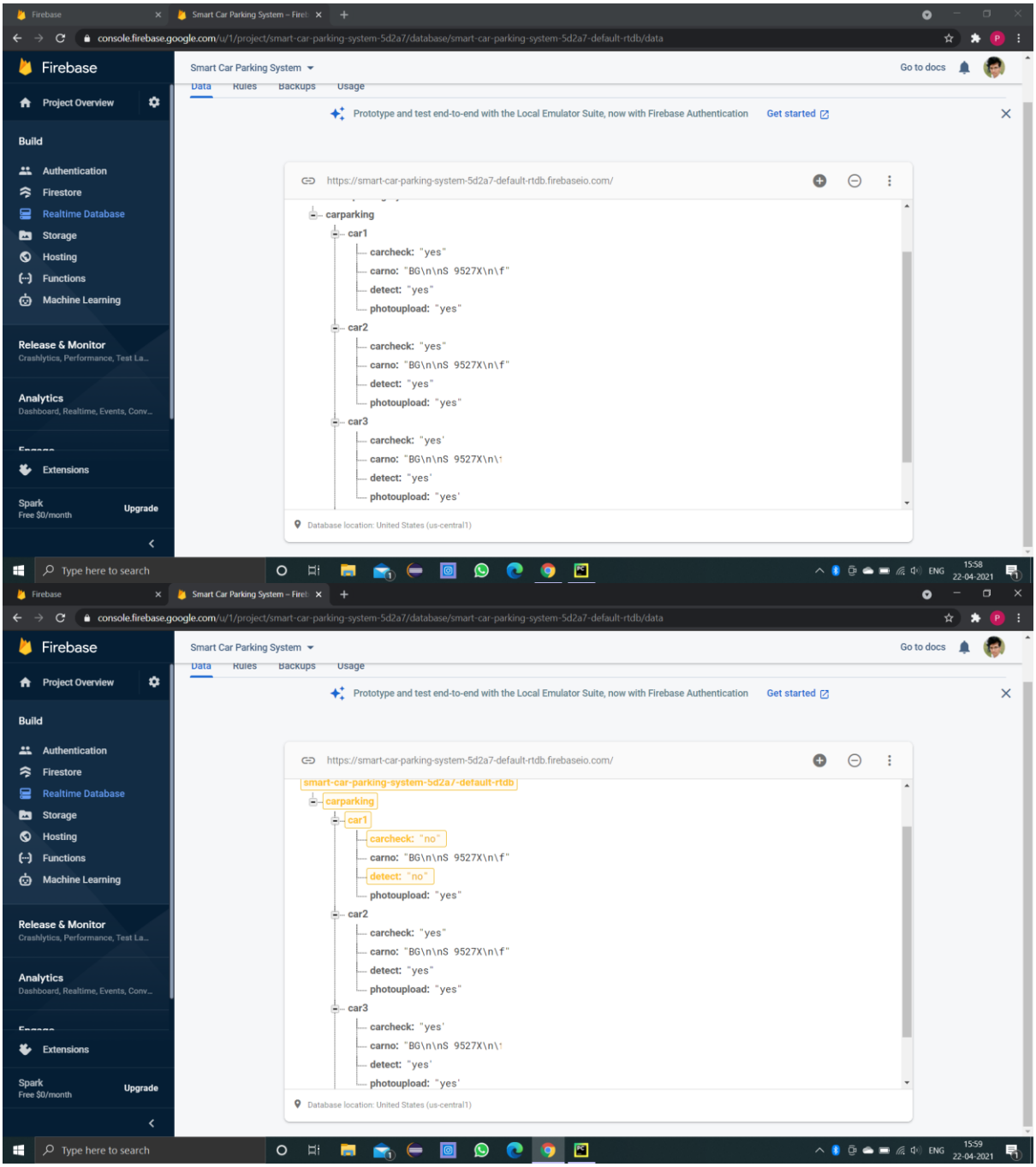

Firebase
Smart Car Parking System – Firebase
console.firebase.google.com/u/1/project/smart-car-parking-system-5d2a7/database/smart-car-parking-system-5d2a7-default-rtdb/data
Smart Car Parking System
Go to docs
Data
Rules
Backups
Usage
Get started
image
UP85 AW 085
Project Overview
Build
Authentication
Firestore
Realtime Database
Storage
Hosting
Functions
Machine Learning
Release & Monitor
Crashlytics, Performance, Test La...
Analytics
Dashboard, Realtime, Events, Conv...
Extensions
Spark
Free $0/month
Upgrade
carcheck: "yes"
carno: "BG\n\nS 9527X\n\
detect: "yes"
photoupload: "yes"
Database location: United States (us-central1)
Type here to search
ENG
15:55
22-04-2021
Prototype and test end-to-end with the Local Emulator Suite, now with Firebase Authentication
https://smart-car-parking-system-5d2a7-default-rtdb.firebaseio.com/
carparking
car1
carcheck: "yes"
carno: ""
detect: "yes"
photoupload: "yes"
car2
carcheck: "no"
carno: ""
detect: "no"
photoupload: "no"
car3
carcheck: "yes"
carno: "BG\n\nS 9527X\n\
detect: "yes"
photoupload: "yes"
15:56
22-04-2021


Firebase
Smart Car Parking System – Firebase
console.firebase.google.com/u/1/project/smart-car-parking-system-5d2a7/database/smart-car-parking-system-5d2a7-default-rtdb/data
Smart Car Parking System
Go to docs
Project Overview
Build
Authentication
Firestore
Realtime Database
Storage
Hosting
Functions
Machine Learning
Release & Monitor
Crashlytics, Performance, Test La...
Analytics
Dashboard, Realtime, Events, Conv...
Extensions
Spark
Free $0/month
Upgrade
Data
Rules
Backups
Usage
Prototype and test end-to-end with the Local Emulator Suite, now with Firebase Authentication
Get started
https://smart-car-parking-system-5d2a7-default-rtdb.firebaseio.com/
carparking
car1
carcheck: "yes"
carno: "BG\n\nS 9527X\n\f"
detect: "yes"
photoupload: "yes"
car2
carcheck: "no"
carno: ""
detect: "no"
photoupload: "no"
car3
carcheck: "yes"
carno: "BG\n\nS 9527X\n\
detect: "yes"
photoupload: "yes"
Database location: United States (us-central1)
Type here to search
ENG
15:56
22-04-2021

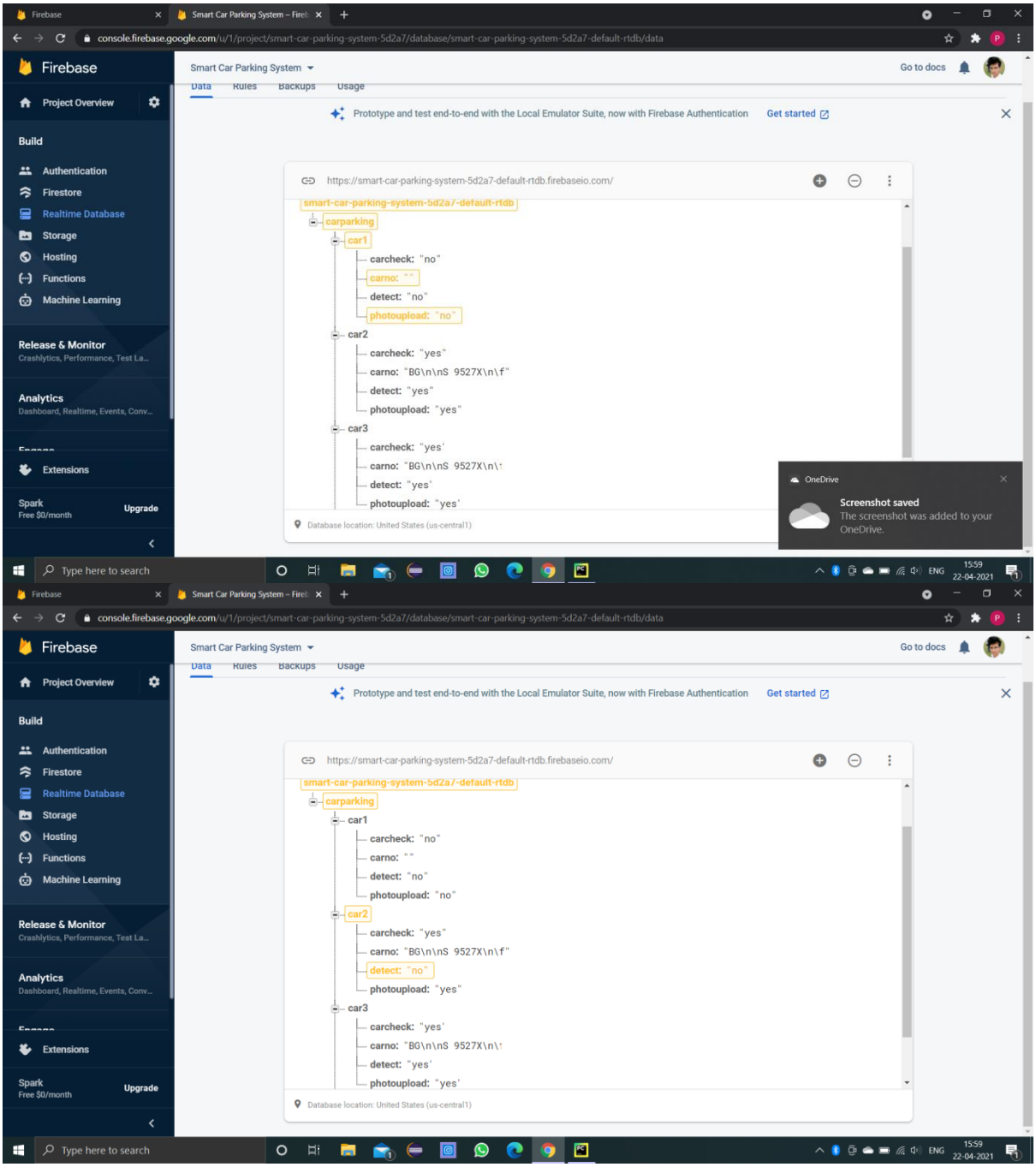
Firebase
Smart Car Parking System – Firebase
console.firebase.google.com/u/1/project/smart-car-parking-system-5d2a7/database/smart-car-parking-system-5d2a7-default-rtdb/data
Smart Car Parking System
Go to docs
Project Overview
Build
Authentication
Firestore
Realtime Database
Storage
Hosting
Functions
Machine Learning
Release & Monitor
Crashlytics, Performance, Test La...
Analytics
Dashboard, Realtime, Events, Conv...
Extensions
Spark
Free $0/month
Upgrade
Data
Rules
Backups
Usage
Prototype and test end-to-end with the Local Emulator Suite, now with Firebase Authentication
Get started
https://smart-car-parking-system-5d2a7-default-rtdb.firebaseio.com/
carparking
car1
carcheck: "yes"
carno: "BG\n\nS 9527X\n\f"
detect: "yes"
photoupload: "yes"
car2
carcheck: "no"
carno: ""
detect: "yes"
photoupload: "no"
car3
carcheck: "yes"
carno: "BG\n\nS 9527X\n\
detect: "yes"
photoupload: "yes"
Database location: United States (us-central1)
Type here to search
ENG
15:57
22-04-2021


Firebase
Smart Car Parking System – Firebase
console.firebase.google.com/u/1/project/smart-car-parking-system-5d2a7/database/smart-car-parking-system-5d2a7-default-rtdb/data
Smart Car Parking System
Go to docs
Data
Rules
Backups
Usage
Get started
Project Overview
Build
Authentication
Firestore
Realtime Database
Storage
Hosting
Functions
Machine Learning
Release & Monitor
Crashlytics, Performance, Test La...
Analytics
Dashboard, Realtime, Events, Conv...
Extensions
Spark
Free $0/month
Upgrade
image
UP85 AW 0856
carno: "BG\n\nS 9527X\n\
detect: "yes"
photoupload: "yes"
Database location: United States (us-central1)
Type here to search
ENG
15:58
22-04-2021

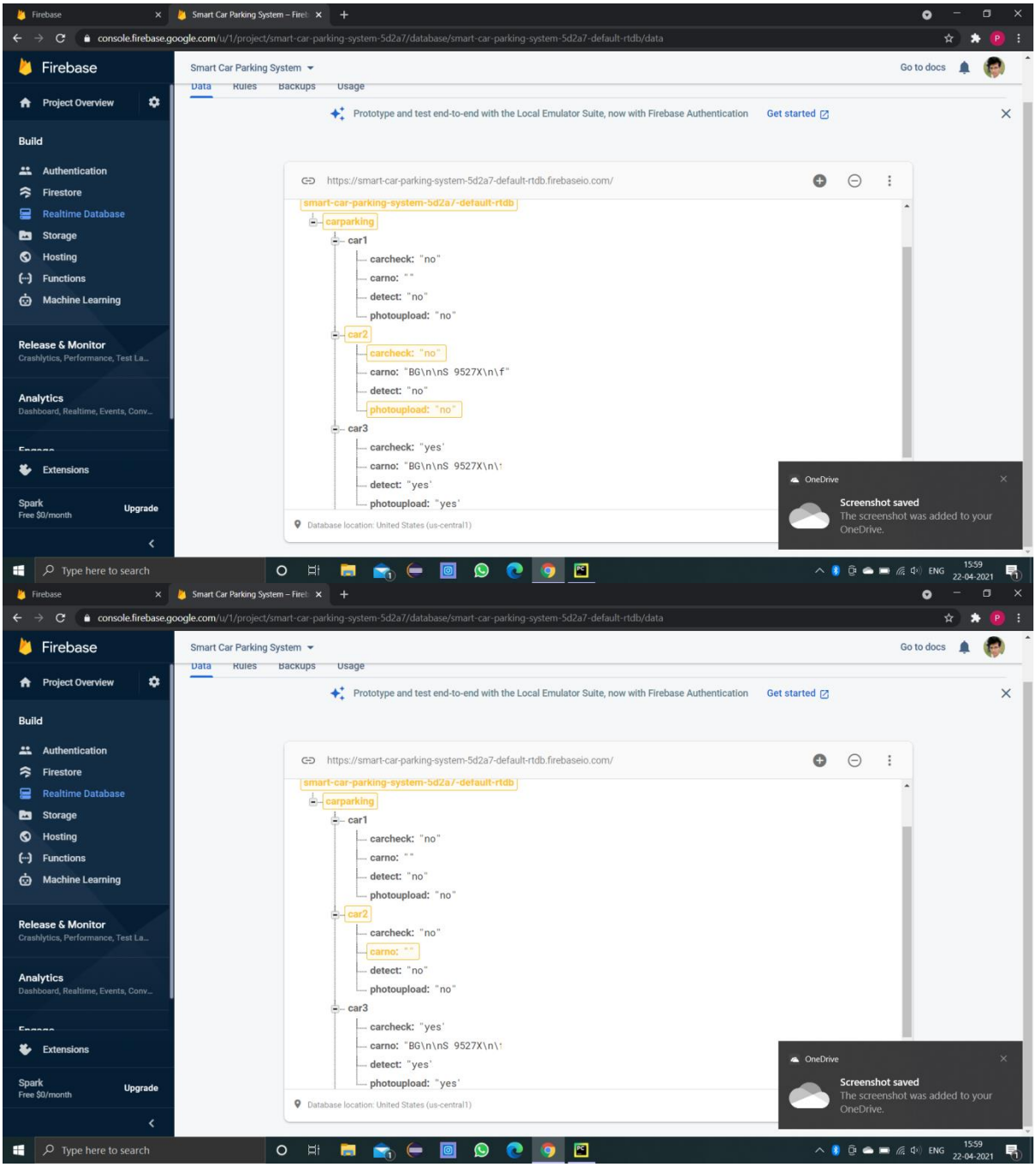
Firebase
Smart Car Parking System – Firebase
console.firebase.google.com/u/1/project/smart-car-parking-system-5d2a7/database/smart-car-parking-system-5d2a7-default-rtdb/data
Smart Car Parking System
Go to docs
Data
Rules
Backups
Usage
Prototype and test end-to-end with the Local Emulator Suite, now with Firebase Authentication
Get started
Project Overview
Build
Authentication
Firestore
Realtime Database
Storage
Hosting
Functions
Machine Learning
Release & Monitor
Crashlytics, Performance, Test La...
Analytics
Dashboard, Realtime, Events, Conv...
Extensions
Spark
Free $0/month
Upgrade
https://smart-car-parking-system-5d2a7-default-rtdb.firebaseio.com/
carparking
car1
carcheck: "yes"
carno: "BG\n\nS 9527X\n\f"
detect: "yes"
photoupload: "yes"
car2
carcheck: "yes"
carno: ""
detect: "yes"
photoupload: "yes"
car3
carcheck: "yes"
carno: "BG\n\nS 9527X\n\
detect: "yes"
photoupload: "yes"
Database location: United States (us-central1)
Type here to search
ENG
15:58
22-04-2021


Firebase
Smart Car Parking System – Firebase
console.firebase.google.com/u/1/project/smart-car-parking-system-5d2a7/database/smart-car-parking-system-5d2a7-default-rtdb/data
Smart Car Parking System
Go to docs
Project Overview
Build
Authentication
Firestore
Realtime Database
Storage
Hosting
Functions
Machine Learning
Release & Monitor
Crashlytics, Performance, Test La...
Analytics
Dashboard, Realtime, Events, Conv...
Extensions
Spark
Free $0/month
Upgrade
Data
Rules
Backups
Usage
Prototype and test end-to-end with the Local Emulator Suite, now with Firebase Authentication
Get started
https://smart-car-parking-system-5d2a7-default-rtdb.firebaseio.com/
carparking
car1
carcheck: "yes"
carno: "BG\n\nS 9527X\n\f"
detect: "yes"
photoupload: "yes"
car2
carcheck: "yes"
carno: "BG\n\nS 9527X\n\f"
detect: "yes"
photoupload: "yes"
car3
carcheck: "yes"
carno: "BG\n\nS 9527X\n\
detect: "yes"
photoupload: "yes"
Database location: United States (us-central1)
Type here to search
ENG
15:58
22-04-2021

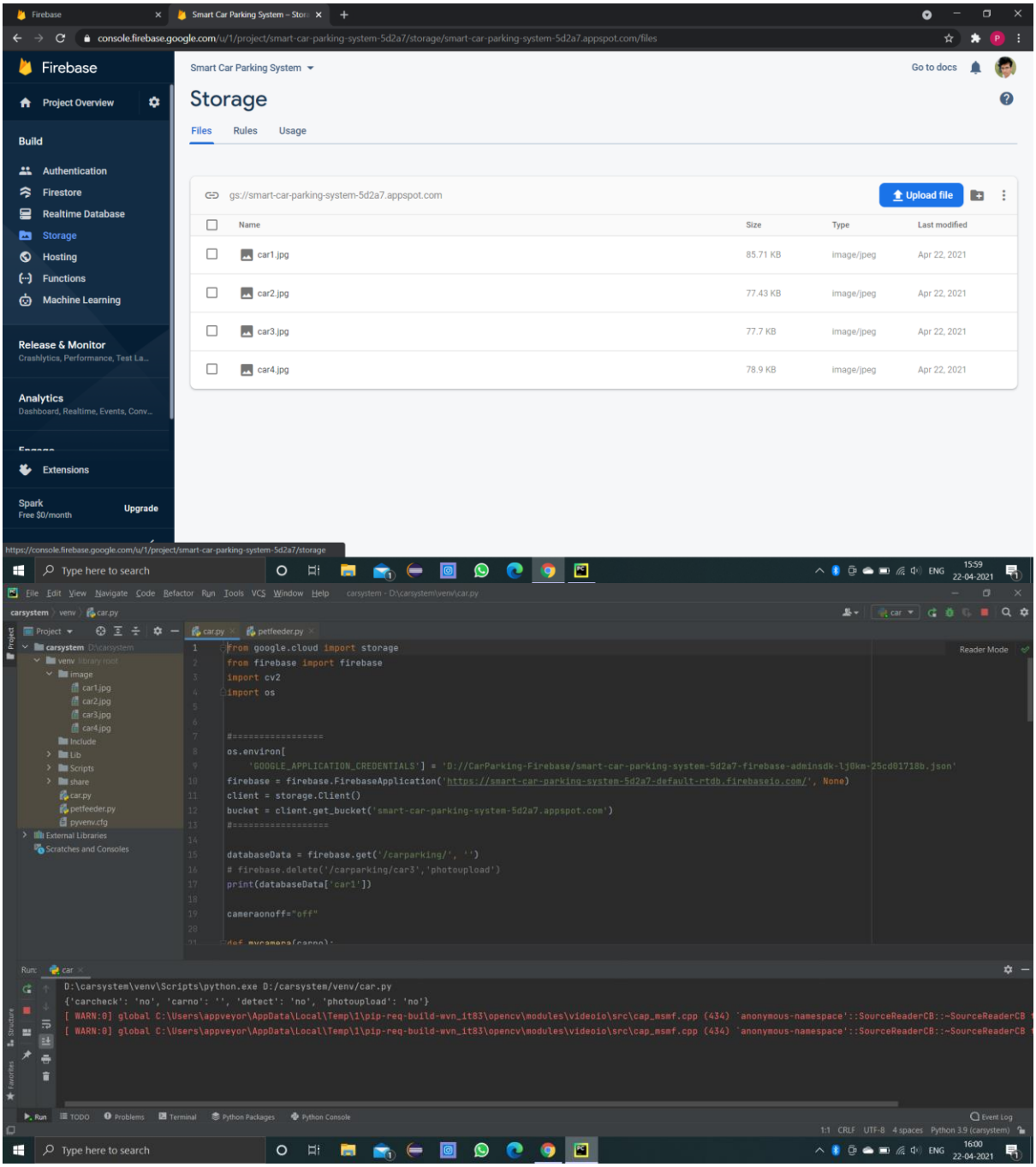
Firebase
Smart Car Parking System – Firebase
console.firebase.google.com/u/1/project/smart-car-parking-system-5d2a7/database/smart-car-parking-system-5d2a7-default-rtdb/data
Smart Car Parking System
Go to docs
Project Overview
Build
Authentication
Firestore
Realtime Database
Storage
Hosting
Functions
Machine Learning
Release & Monitor
Crashlytics, Performance, Test La...
Analytics
Dashboard, Realtime, Events, Conv...
Extensions
Spark
Free $0/month
Upgrade
Data
Rules
Backups
Usage
Prototype and test end-to-end with the Local Emulator Suite, now with Firebase Authentication
Get started
https://smart-car-parking-system-5d2a7-default-rtdb.firebaseio.com/
smart-car-parking-system-5d2a7-default-rtdb
carparking
car1
carcheck: "no"
carno: "BG\n\nS 9527X\n\f"
detect: "no"
photoupload: "yes"
car2
carcheck: "yes"
carno: "BG\n\nS 9527X\n\f"
detect: "yes"
photoupload: "yes"
car3
carcheck: "yes"
carno: "BG\n\nS 9527X\n\
detect: "yes"
photoupload: "yes"
Database location: United States (us-central1)
Type here to search
ENG
15:59
22-04-2021

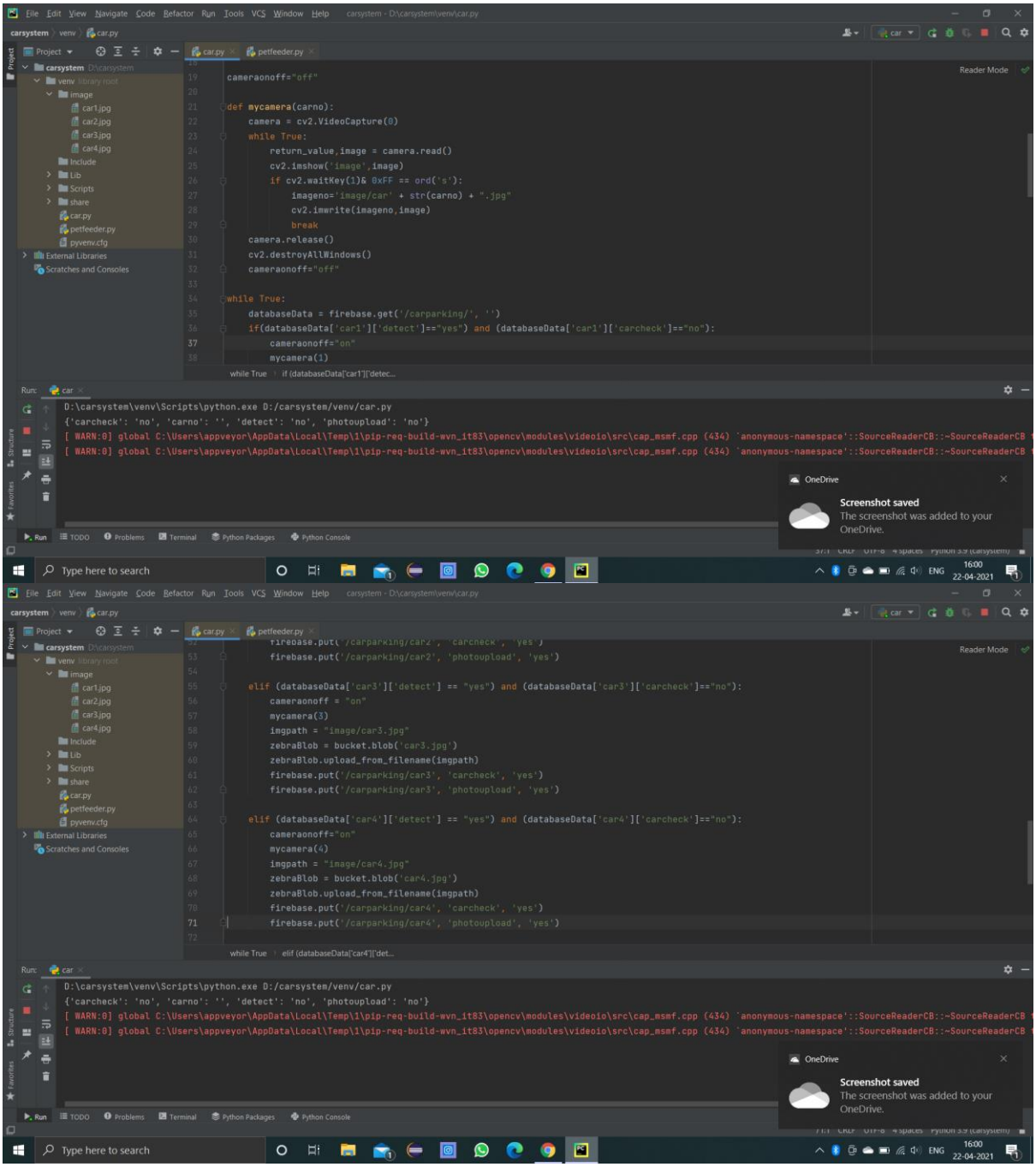

Firebase
Smart Car Parking System – Firebase
console.firebase.google.com/u/1/project/smart-car-parking-system-5d2a7/database/smart-car-parking-system-5d2a7-default-rtdb/data
Smart Car Parking System
Go to docs
Data
Rules
Backups
Usage
Prototype and test end-to-end with the Local Emulator Suite, now with Firebase Authentication
Get started
Project Overview
Build
Authentication
Firestore
Realtime Database
Storage
Hosting
Functions
Machine Learning
Release & Monitor
Crashlytics, Performance, Test La...
Analytics
Dashboard, Realtime, Events, Conv...
Extensions
Spark
Free $0/month
Upgrade
https://smart-car-parking-system-5d2a7-default-rtdb.firebaseio.com/
smart-car-parking-system-5d2a7-default-rtdb
carparking
car1
carcheck: "no"
carno: ""
detect: "no"
photoupload: "no"
car2
carcheck: "yes"
carno: "BG\n\nS 9527X\n\f"
detect: "yes"
photoupload: "yes"
car3
carcheck: "yes"
carno: "BG\n\nS 9527X\n\
detect: "yes"
photoupload: "yes"
Database location: United States (us-central1)
OneDrive
Screenshot saved
The screenshot was added to your OneDrive.
Type here to search
ENG
15:59
22-04-2021

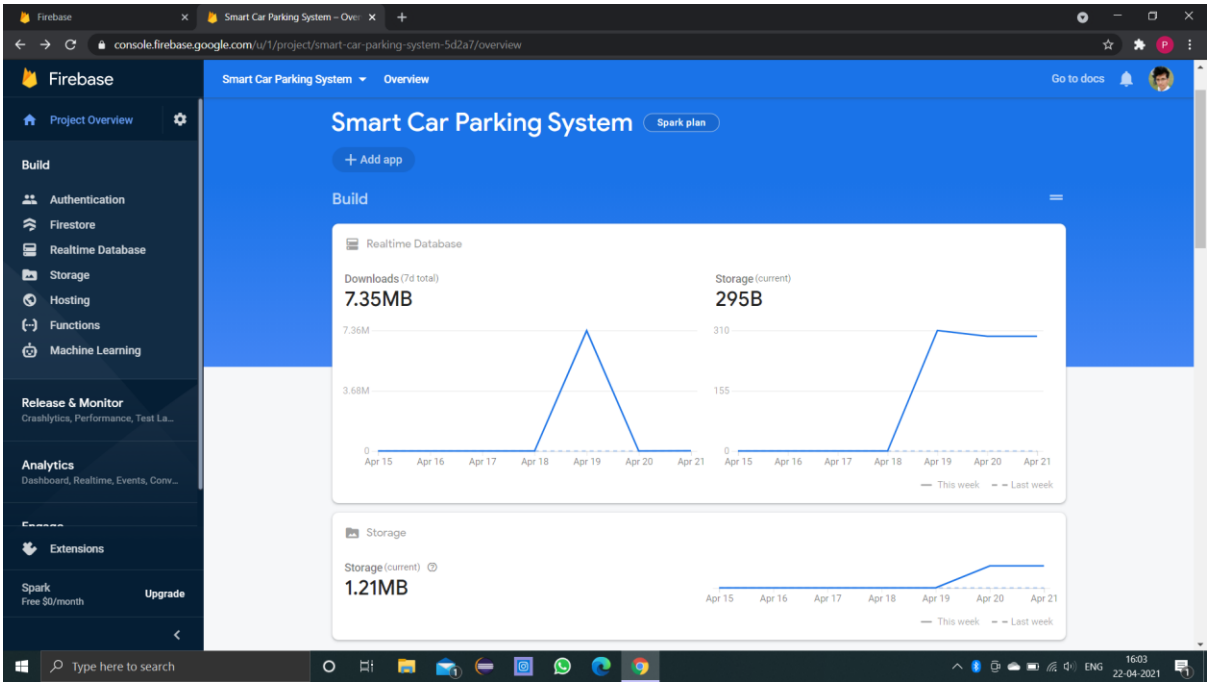
Firebase
Smart Car Parking System – Firebase
console.firebase.google.com/u/1/project/smart-car-parking-system-5d2a7/database/smart-car-parking-system-5d2a7-default-rtdb/data
Smart Car Parking System
Go to docs
Data
Rules
Backups
Usage
Prototype and test end-to-end with the Local Emulator Suite, now with Firebase Authentication
Get started
Project Overview
Build
Authentication
Firestore
Realtime Database
Storage
Hosting
Functions
Machine Learning
Release & Monitor
Crashlytics, Performance, Test La...
Analytics
Dashboard, Realtime, Events, Conv...
Extensions
Spark
Free $0/month
Upgrade
https://smart-car-parking-system-5d2a7-default-rtdb.firebaseio.com/
smart-car-parking-system-5d2a7-default-rtdb
carparking
car1
carcheck: "no"
carno: ""
detect: "no"
photoupload: "no"
car2
carcheck: "yes"
carno: "BG\n\nS 9527X\n\f"
detect: "no"
photoupload: "yes"
car3
carcheck: "yes"
carno: "BG\n\nS 9527X\n\
detect: "yes"
photoupload: "yes"
Database location: United States (us-central1)
Type here to search
ENG
15:59
22-04-2021


Firebase
Smart Car Parking System – Firebase
console.firebase.google.com/u/1/project/smart-car-parking-system-5d2a7/database/smart-car-parking-system-5d2a7-default-rtdb/data
Smart Car Parking System
Go to docs
Data
Rules
Backups
Usage
Prototype and test end-to-end with the Local Emulator Suite, now with Firebase Authentication
Get started
Project Overview
Build
Authentication
Firestore
Realtime Database
Storage
Hosting
Functions
Machine Learning
Release & Monitor
Crashlytics, Performance, Test La...
Analytics
Dashboard, Realtime, Events, Conv...
Extensions
Spark
Free $0/month
Upgrade
https://smart-car-parking-system-5d2a7-default-rtdb.firebaseio.com/
smart-car-parking-system-5d2a7-default-rtdb
carparking
car1
carcheck: "no"
carno: ""
detect: "no"
photoupload: "no"
car2
carcheck: "no"
carno: "BG\n\nS 9527X\n\f"
detect: "no"
photoupload: "no"
car3
carcheck: "yes"
carno: "BG\n\nS 9527X\n\
detect: "yes"
photoupload: "yes"
Database location: United States (us-central1)
OneDrive
Screenshot saved
The screenshot was added to your OneDrive.
Type here to search
ENG
15:59
22-04-2021

Firebase
Smart Car Parking System – Firebase
console.firebase.google.com/u/1/project/smart-car-parking-system-5d2a7/database/smart-car-parking-system-5d2a7-default-rtdb/data
Smart Car Parking System
Go to docs
Data
Rules
Backups
Usage
Prototype and test end-to-end with the Local Emulator Suite, now with Firebase Authentication
Get started
Project Overview
Build
Authentication
Firestore
Realtime Database
Storage
Hosting
Functions
Machine Learning
Release & Monitor
Crashlytics, Performance, Test La...
Analytics
Dashboard, Realtime, Events, Conv...
Extensions
Spark
Free $0/month
Upgrade
https://smart-car-parking-system-5d2a7-default-rtdb.firebaseio.com/
smart-car-parking-system-5d2a7-default-rtdb
carparking
car1
carcheck: "no"
carno: ""
detect: "no"
photoupload: "no"
car2
carcheck: "no"
carno: ""
detect: "no"
photoupload: "no"
car3
carcheck: "yes"
carno: "BG\n\nS 9527X\n\
detect: "yes"
photoupload: "yes"
Database location: United States (us-central1)
OneDrive
Screenshot saved
The screenshot was added to your OneDrive.
Type here to search
ENG
15:59
22-04-2021


Firebase
Smart Car Parking System – Stor
console.firebase.google.com/u/1/project/smart-car-parking-system-5d2a7/storage/smart-car-parking-system-5d2a7.appspot.com/files
Smart Car Parking System
Go to docs
Project Overview
Build
Authentication
Firestore
Realtime Database
Storage
Hosting
Functions
Machine Learning
Release & Monitor
Crashlytics, Performance, Test La...
Analytics
Dashboard, Realtime, Events, Conv...
Extensions
Spark
Free $0/month
Upgrade
Storage
Files
Rules
Usage
gs://smart-car-parking-system-5d2a7.appspot.com
Upload file
Name
Size
Type
Last modified
car1.jpg
85.71 KB
image/jpeg
Apr 22, 2021
car2.jpg
77.43 KB
image/jpeg
Apr 22, 2021
car3.jpg
77.7 KB
image/jpeg
Apr 22, 2021
car4.jpg
78.9 KB
image/jpeg
Apr 22, 2021
https://console.firebase.google.com/u/1/project/smart-car-parking-system-5d2a7/storage
Type here to search
ENG
15:59
22-04-2021
File Edit View Navigate Code Refactor Run Tools VCS Window Help
carsystem - D:\carsystem\venv\car.py
carsystem
venv
car.py
Project
car.py
petfeeder.py
carsystem D:\carsystem
venv library root
image
car1.jpg
car2.jpg
car3.jpg
car4.jpg
Include
Lib
Scripts
share
car.py
petfeeder.py
pyvenv.cfg
External Libraries
Scratches and Consoles
Reader Mode
from google.cloud import storage
from firebase import firebase
import cv2
import os
#==================
os.environ[
'GOOGLE_APPLICATION_CREDENTIALS'] = 'D://CarParking-Firebase/smart-car-parking-system-5d2a7-firebase-adminsdk-lj0km-25cd01718b.json'
firebase = firebase.FirebaseApplication('https://smart-car-parking-system-5d2a7-default-rtdb.firebaseio.com/', None)
client = storage.Client()
bucket = client.get_bucket('smart-car-parking-system-5d2a7.appspot.com')
#==================
databaseData = firebase.get('/carparking/', '')
# firebase.delete('/carparking/car3','photoupload')
print(databaseData['car1'])
cameraonoff="off"
Run:
car
D:\carsystem\venv\Scripts\python.exe D:/carsystem/venv/car.py
{'carcheck': 'no', 'carno': '', 'detect': 'no', 'photoupload': 'no'}
[WARN:0] global C:\Users\appveyor\AppData\Local\Temp\1\pip-req-build-wvn_it83\opencv\modules\videoio\src\cap_msmf.cpp (434) `anonymous-namespace'::SourceReaderCB::~SourceReaderCB
[WARN:0] global C:\Users\appveyor\AppData\Local\Temp\1\pip-req-build-wvn_it83\opencv\modules\videoio\src\cap_msmf.cpp (434) `anonymous-namespace'::SourceReaderCB::~SourceReaderCB
Run
TODO
Problems
Terminal
Python Packages
Python Console
Event Log
1:1 CRLF UTF-8 4 spaces Python 3.9 (carsystem)
Type here to search
ENG
16:00
22-04-2021


File Edit View Navigate Code Refactor Run Tools VCS Window Help
carsystem - D:\carsystem\venv\car.py
carsystem venv car.py
Project
car.py
petfeeder.py
Reader Mode
carsystem D:\carsystem
venv library root
image
car1.jpg
car2.jpg
car3.jpg
car4.jpg
Include
Lib
Scripts
share
car.py
petfeeder.py
pyvenv.cfg
External Libraries
Scratches and Consoles
cameraonoff="off"
def mycamera(carno):
camera = cv2.VideoCapture(0)
while True:
return_value,image = camera.read()
cv2.imshow('image',image)
if cv2.waitKey(1)& 0xFF == ord('s'):
imageno='image/car' + str(carno) + ".jpg"
cv2.imwrite(imageno,image)
break
camera.release()
cv2.destroyAllWindows()
cameraonoff="off"
while True:
databaseData = firebase.get('/carparking/', '')
if(databaseData['car1']['detect']=="yes") and (databaseData['car1']['carcheck']=="no"):
cameraonoff="on"
mycamera(1)
while True
if (databaseData['car1']['detec...
Run: car
D:\carsystem\venv\Scripts\python.exe D:/carsystem/venv/car.py
{'carcheck': 'no', 'carno': '', 'detect': 'no', 'photoupload': 'no'}
[WARN:0] global C:\Users\appveyor\AppData\Local\Temp\1\pip-req-build-wvn_it83\opencv\modules\videoio\src\cap_msmf.cpp (434) `anonymous-namespace'::SourceReaderCB::~SourceReaderCB
[WARN:0] global C:\Users\appveyor\AppData\Local\Temp\1\pip-req-build-wvn_it83\opencv\modules\videoio\src\cap_msmf.cpp (434) `anonymous-namespace'::SourceReaderCB::~SourceReaderCB
Run
TODO
Problems
Terminal
Python Packages
Python Console
OneDrive
Screenshot saved
The screenshot was added to your OneDrive.
Type here to search
ENG
16:00
22-04-2021
File Edit View Navigate Code Refactor Run Tools VCS Window Help
carsystem - D:\carsystem\venv\car.py
firebase.put('/carparking/car2', 'carcheck', 'yes')
firebase.put('/carparking/car2', 'photoupload', 'yes')
elif (databaseData['car3']['detect'] == "yes") and (databaseData['car3']['carcheck']=="no"):
cameraonoff = "on"
mycamera(3)
imgpath = "image/car3.jpg"
zebraBlob = bucket.blob('car3.jpg')
zebraBlob.upload_from_filename(imgpath)
firebase.put('/carparking/car3', 'carcheck', 'yes')
firebase.put('/carparking/car3', 'photoupload', 'yes')
elif (databaseData['car4']['detect'] == "yes") and (databaseData['car4']['carcheck']=="no"):
cameraonoff="on"
mycamera(4)
imgpath = "image/car4.jpg"
zebraBlob = bucket.blob('car4.jpg')
zebraBlob.upload_from_filename(imgpath)
firebase.put('/carparking/car4', 'carcheck', 'yes')
firebase.put('/carparking/car4', 'photoupload', 'yes')
while True
elif (databaseData['car4']['det...
Run: car
D:\carsystem\venv\Scripts\python.exe D:/carsystem/venv/car.py
{'carcheck': 'no', 'carno': '', 'detect': 'no', 'photoupload': 'no'}
[WARN:0] global C:\Users\appveyor\AppData\Local\Temp\1\pip-req-build-wvn_it83\opencv\modules\videoio\src\cap_msmf.cpp (434) `anonymous-namespace'::SourceReaderCB::~SourceReaderCB
[WARN:0] global C:\Users\appveyor\AppData\Local\Temp\1\pip-req-build-wvn_it83\opencv\modules\videoio\src\cap_msmf.cpp (434) `anonymous-namespace'::SourceReaderCB::~SourceReaderCB
OneDrive
Screenshot saved
The screenshot was added to your OneDrive.
Type here to search
ENG
16:00
22-04-2021


Firebase
Smart Car Parking System – Over
console.firebase.google.com/u/1/project/smart-car-parking-system-5d2a7/overview
Firebase
Project Overview
Build
Authentication
Firestore
Realtime Database
Storage
Hosting
Functions
Machine Learning
Release & Monitor
Crashlytics, Performance, Test La...
Analytics
Dashboard, Realtime, Events, Conv...
Extensions
Spark
Free $0/month
Upgrade
Smart Car Parking System
Overview
Go to docs
Smart Car Parking System
Spark plan
Add app
Build
Realtime Database
Downloads (7d total)
7.35MB
Storage (current)
295B
7.36M
3.68M
0
310
155
0
Apr 15
Apr 16
Apr 17
Apr 18
Apr 19
Apr 20
Apr 21
This week
Last week
Storage
Storage (current)
1.21MB
Type here to search
ENG
16:03
22-04-2021

THANK YOU