AL-AZHAR UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF ELECTRONICS AND COMMUNICATION

GRADUATION PROJECT

Smart Parking System

SUPERVISED BY:

DR: USAMA ABDELFATTAH
DR: AHMED ZAKZOUK

PRESENTED BY:

Nariman El-Sayed Mohammed

Alaa Abdul Wahab Eid

Hager Mohamed Ibrahim

Mariam Youssef Mohammed

Alaa Gabr Abdul Baki

Mariam Hesham Salah El-Deen

ACADEMIC YEAR ***2021-2022***

Acknowledgment

At the beginning, we must thank God for our success in completing this project despite all the difficult circumstances and obstacles that we faced - which of course are beyond our control - throughout the year.

We would like to express our sincere gratitude to our supervisor **Dr. Usama Abdel Fattah** and **Dr. Ahmed Zakzouk** in for the continuous support along the year, for their patience, motivation, enthusiasm, and immense knowledge. Their guidance helped us in all the time of project and writing of this book.

They also helped us in doing a lot of Research and giving us so many ideas widening our horizon of thinking and increasing our problem solving abilities. we came to know about so many new things. We are really thankful to them.

Last but not the least, we would like to thank our families for always supporting us spiritually and mentally throughout our life and witnessing our great success.

Abstract

Smart Parking System is an intelligent system that reside inside the parking and assist the driver in a variety of ways using main website.

adapting with this hurried world and full of new technologies, we have two solutions you can park without getting a ticket and you can park easily by guiding to the nearest empty parking location using the enough needed technologies to do that.

This system be used to provide vital information about car Id and entrances and exits time. Therefore, it has the potential to optimize security inside the parking.

This system is also used for check-outs and shows the car owner how much time has been spent and how much money has to be paid.

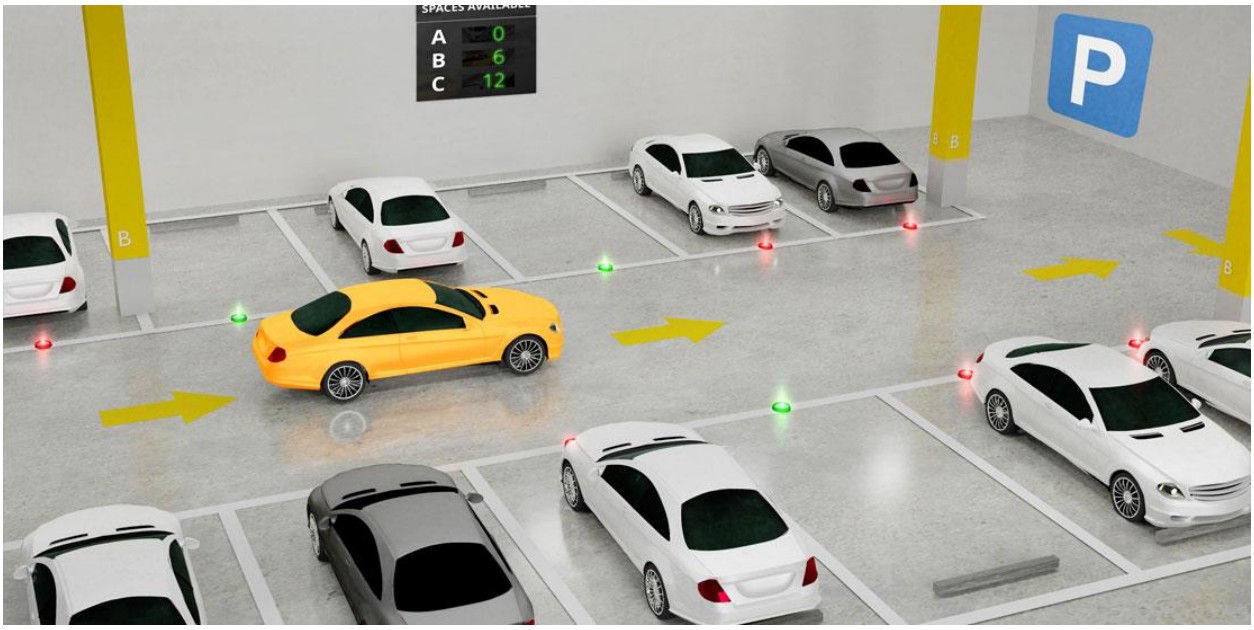
The greatest advantage of using the smart parking system is that the ticketless parking you can park without the need of queuing and the fuss to get a ticket to enter the parking.

Table of Contents

List of figures

List of Tables

Chapter 1
Introduction



Drivers searching for parking are estimated to be responsible for about 30% of traffic congestion in cities. Historically, cities, businesses, and property developers have tried to match parking supply to growing demand for parking spaces. It has become clear, though, that simply creating more parking spaces is not sufficient to address the problem of congestion. New approaches using smart parking systems look to provide a more balanced view of parking that better manages the relationship between supply and demand.

Smart parking can be defined as the use of advanced technologies for the efficient operation, monitoring, and management of parking within an urban mobility strategy. The global market for smart parking systems reached $93.5 million, with the United States representing 46% of the market share and offering a strong growth opportunity for companies offering services in the United States and overseas.

A number of technologies provide the basis for smart parking solutions, including vehicle sensors, wireless communications, and data analytics. Smart parking is also made viable by innovation in areas such as smartphone apps for customer service, mobile payments, and in-car navigation systems. At the heart of the smart parking concept is the ability to access, collect, analyze, disseminate, and act on information on parking usage. Increasingly, this information is provided in real-time by intelligent devices that enable both parking managers and drivers to optimize the use of parking capacity.

1.1 Motivation to use Smart parking:

1. Optimized parking: Users find the best spot available, saving time, resources, and effort. The parking lot fills up efficiently and space can be utilized properly by commercial and corporate entities.

2. Reduced traffic —Traffic flow increases as fewer cars are required to drive around in search of an open parking space.

3. Reduced pollution: Searching for parking burns around one million barrels of oil a day. An optimal parking solution will significantly decrease driving time, thus lowering the amount of daily vehicle emissions and ultimately reducing the global environmental footprint.

4. Enhanced User Experience: A smart parking solution will integrate the entire user experience into a unified action. The driver's payment, spot identification, location search, and time notifications all seamlessly become part of the destination arrival process.

5. New Revenue Streams – Many new revenue streams are possible with smart parking technology. For example, lot owners can enable tiered payment options dependent on parking space location. Also, reward programmes can be integrated into existing models to encourage repeat users.

6. Integrated Payments and POS: Returning users can replace daily, manual cash payments with account invoicing and application payments from their

phone. This could also enable customer loyalty programmes and valuable user feedback.

7. Increased Safety —Parking lot employees and security guards have access to real-time lot data that can help prevent parking violations and suspicious activity. License plate recognition cameras can gather pertinent footage. Also, decreased spot-searching traffic on the streets can reduce accidents caused by the distraction of searching for parking.

8. Real-Time Data and Trend Insight: Over time, a smart parking solution can produce data that uncovers correlations and trends of users and lots. These trends can prove to be invaluable to lot owners as to how to make adjustments and improvements for drivers.

9. Decreased Management Costs—More automation and less manual activity saves on labor costs and resource exhaustion.

1.2 Solution

The overview of this project is to make a smart parking system using modern technologies like IOT and AI. Using AI algorithms and deep learning, we implemented a ticketless parking system you can park without the need to queue and the fuss of getting a ticket to enter the parking. And an easy guide to an empty parking slot using a website for that. It is a system that collects data about car entrances and exits that may help in many analytics. It also helps for security purposes.

1.3 Need for project

In this advanced world, we need all the places for homes, hospitals, streets and garages.

And we see around us nowadays smart cities with all their parts connected to one system, which makes life easier with the aid of technology. So in these smart cities, we also need smart garages to adapt to these cities and make parking easier and also reduce time for parking.

1.4 Organization

The following chapters will discuss the work done to achieve the proposed system for improved parking. The two major points of investigation and discussion in this system will be the plate recognition module and the parking free locations module. Each of them will be discussed separately and in detail.

The flow will include the plate recognition module in chapter 2, This happens when entering or exiting the car to or from the parking lot. Starting from some basic knowledge about the deployed technology, literature survey, system details, reached results, and some suggested future work to improve the system.

Then localhost server section will be discussed in chapter 3 shows how the recognized-plate-number be stored in database and car-entry and exit-time, and the website can show to the car's owner the money to paid. Then the parking free locations module system, which shows the free and busy locations in the parking website interface.

Then the Hardware used will be discussed in chapter 4 in detail.

And how to connect these systems will be discussed in chapter 5.

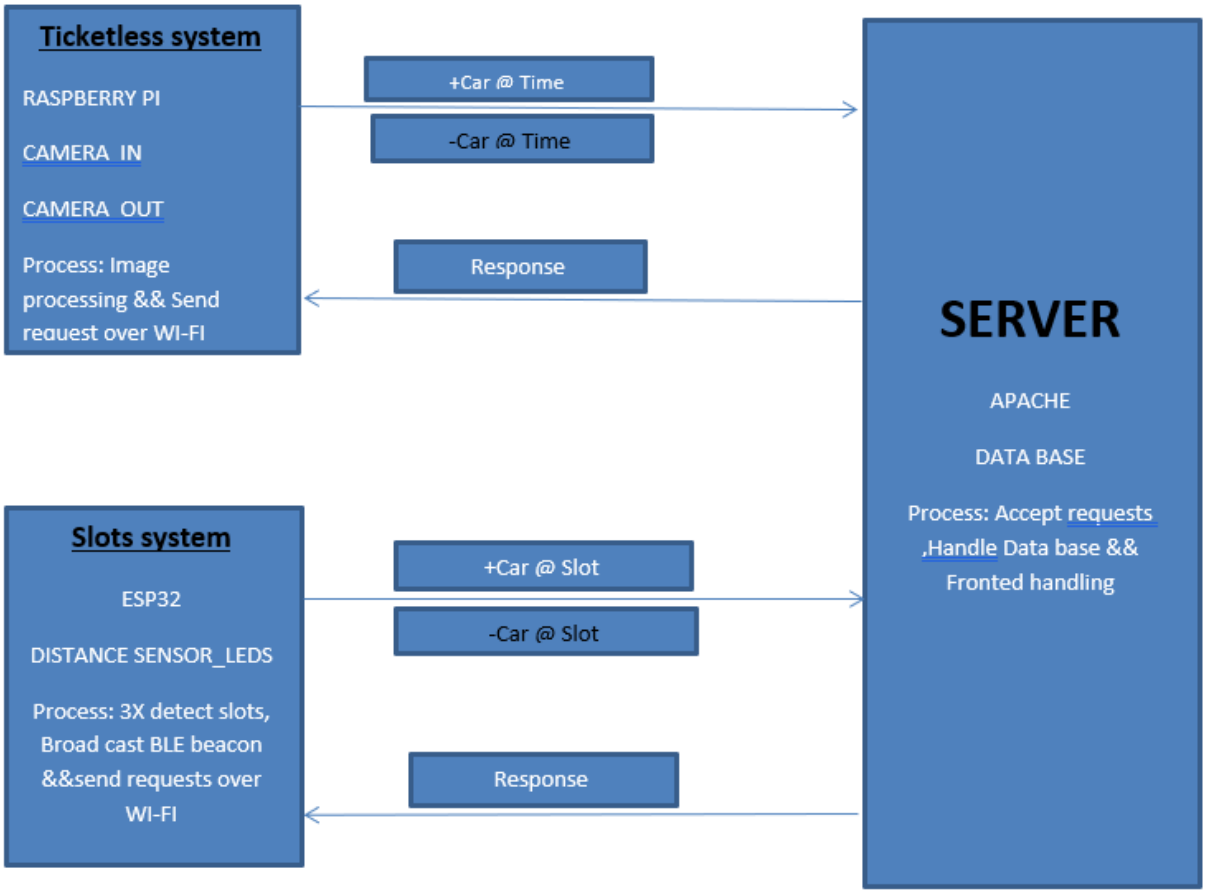


Figure 1.1: Block diagram of Smart Parking System

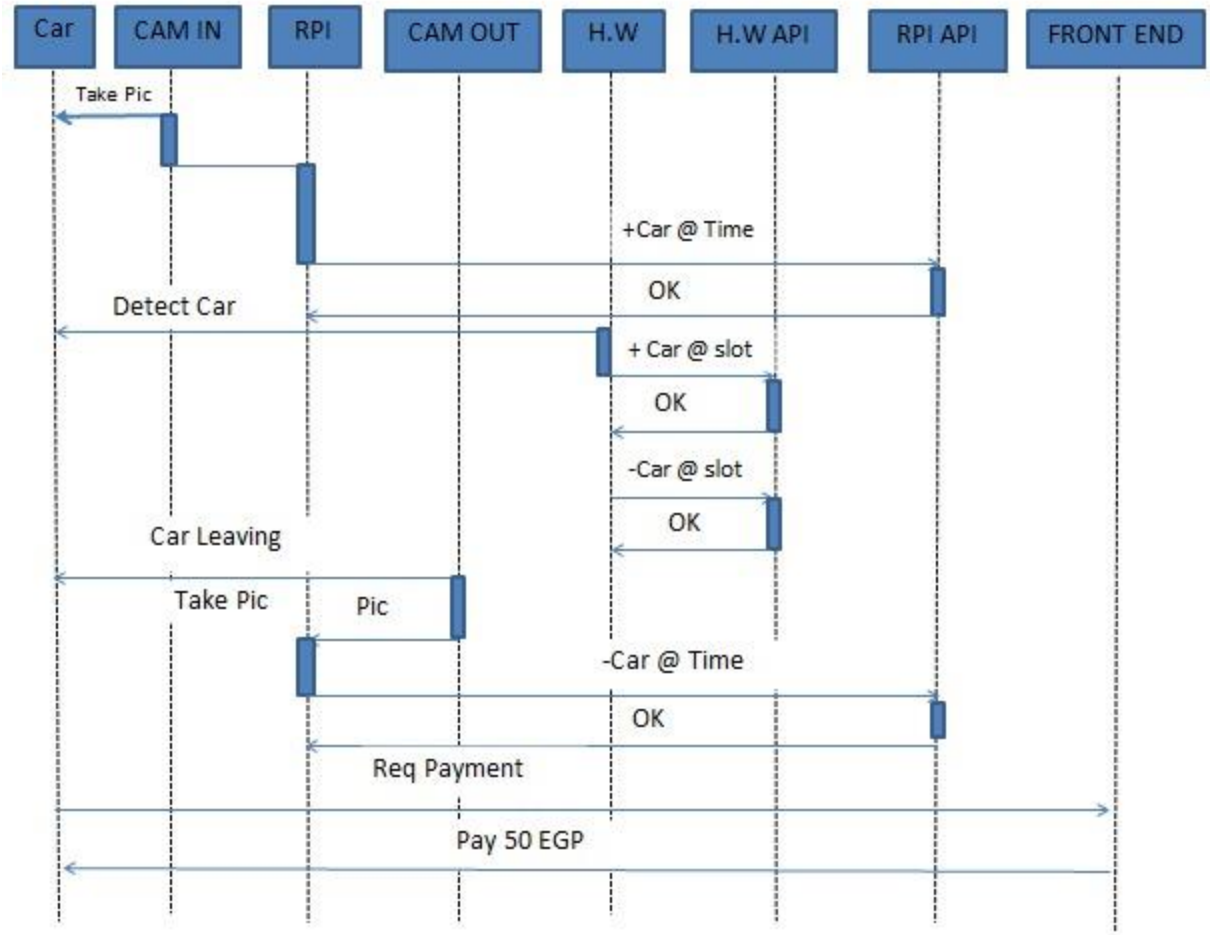


Figure 1.2: Sequence diagram of Smart Parking System

Chapter 2
Plate Detection and Recognition

AI part:

AI has many fields such as natural language processing, speech processing, planning and computer vision.

Computer vision is the technology we used to implement our project

is a field of artificial intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos, and other visual inputs — and take actions or make recommendations based on that information. If AI enables computers to think, computer vision enables them to see, observe and understand

We used computer vision principles to help us in our project. Computer vision uses deep learning to perform tasks such as image processing, image classification, object detection, and object segmentation.

We are concerned about image detection, which is the final stage of image processing which is one of the most important computer vision tasks.

Image recognition uses deep learning to teach the computer how to recognize images, numbers, or even parts of an image, like a smile in the face or a person in a group of photos. if we compare it to how we humans recognize images or people in photos, it is easy for us, but for computers it is not that easy to comprehend and distinguish.

So, how do we teach a computer to recognize or identify something in a photo?

Machine learning works by taking data as an input, applying various ML algorithms to the data to interpret it, and giving an output. Deep learning is different than machine learning because it employs a layered neural network.

Neural networks, works as our brain does it allows computers to recognize patterns and solve problems in AI field. The layers of it are input, hidden, and output layers. [1]

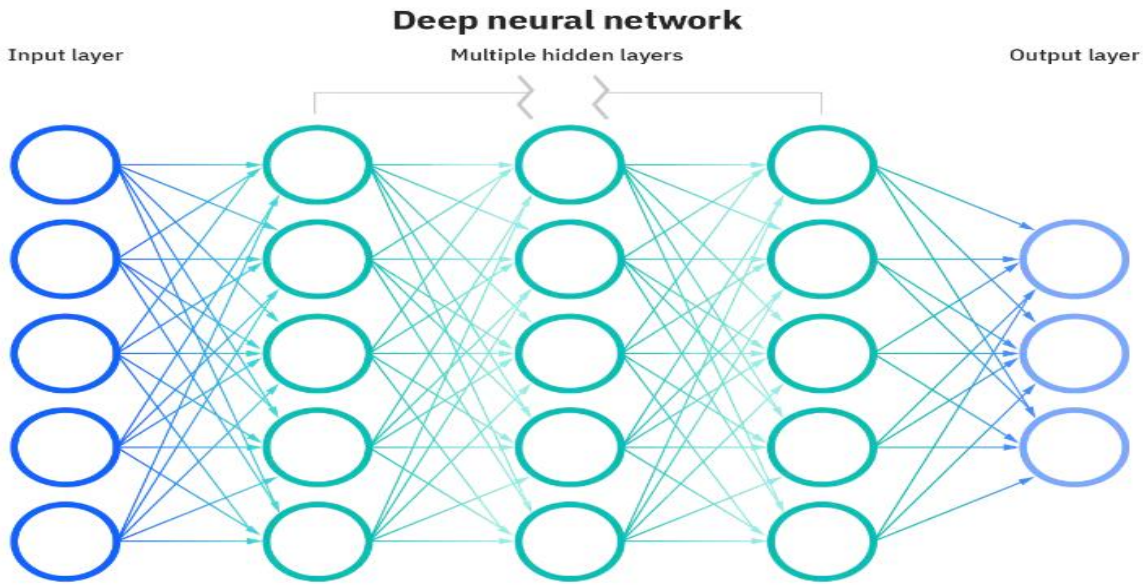


Figure 2.1: Deep neural network

2.1 AI for object detection

We can employ two deep learning techniques to perform object recognition. One is to train a model from scratch, and the other is to use an already trained deep learning model. Based on these models, we can build many useful object recognition applications.

In our project, we used the second one. we used a trained model to recognize the plates of a car.

From the idea of a project, from the entry of a car to the exit of it:

The car enters the garage without needing to wait and queue to get a ticket. We just capture the photo of the car and do processing to get the plate numbers and store it on the server.

When exiting the car, there will be a gate that will allow you to pass when checking that you have paid for your parking time. We check if you have paid or not by capturing the image of the car when exiting, then take its plate number and give it to the server and wait for it to tell us if it has paid its tolls or not.[2]

2.2 Theory of Plate Detection Classifiers

2.2.1 Haar Cascade Classifiers using OpenCV

Haar Cascades: It is an Object Detection Algorithm used to identify faces in an image or a real-time video.

Example:

Each feature is a single value obtained by subtracting the sum of pixels under the white rectangle from the sum of pixels under the black rectangle.

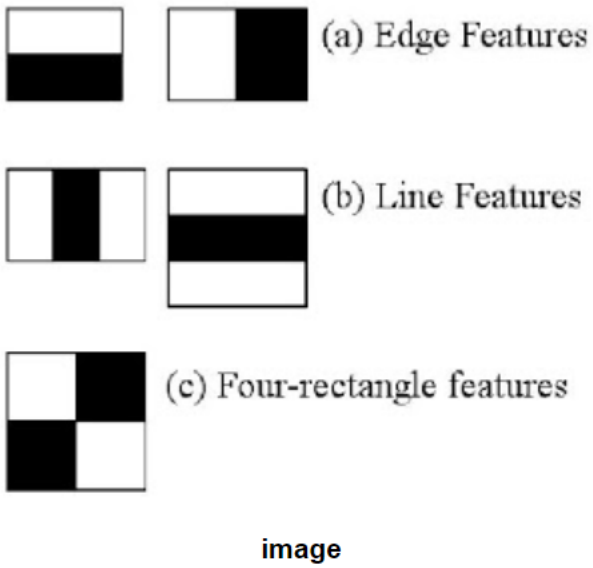


Figure2.2 Object detection using training Haar cascades

For each feature calculation, we need to find the sum of the pixels under white and black rectangles. To solve this, they introduced the integral image. However large your image is, it reduces the calculations for a given pixel to an operation involving just four pixels.[3]

2.2.1.1 Implementing Haar-cascades in OpenCV

1-The OpenCV library manages a repository containing all popular haar cascades that can be used for:

- Human face detection
- Eye detection
- Nose / Mouth detection
- Vehicle detection

We will use it in the project for vehicle detection.

Figure2.3 Haar cascade algorithms

Haar cascades are XML files that can be used in OpenCV to detect specified objects.

2- Installing OpenCV in Python:

Installing OpenCV is easy using the pip installer.

```
!pip install opencv-python
#---OR ---
!pip install opencv-contrib-python
```

3- Loading Haar Cascade in OpenCV

We can load any haar-cascade XML file using cv2.CascadeClassifier function

```
face_detector=cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
eye_dectector = cv2.CascadeClassifier('haarcascade_eye.xml')
```

Once cascade is loaded in OpenCV, we can call the detector function.

```
results = face_detector.detectMultiScale(gray_img, scaleFactor=1.05,minNeighbors=5,minSize=(30,
```

results It lists coordinates (x, y, w, h) of bounding boxes around the detected object.

Car number plate recognition using OpenCV

Recognize and manipulate cars' number plates using openCv and haar cascades

Features

- Recognize cars' plates
- Blur Car plates after detection
- Detects plates from image/video/webcam

2.2.1.2 Implemented Libraries in Code

```
import cv2
```

OpenCV: It is an open source library which is supported by multiple platforms, including Windows. It is most commonly used in Python for machine learning applications, specifically in the computer vision domain.

```
import numpy as np
```

NumPy: (Numerical Python) is an open-source library for the Python programming language. It is used for scientific computing and working with arrays.

```
import pytesseract
```

Python-tesseract: It is an optical character recognition (OCR) tool for Python. That is, it will recognize and "read" the text embedded in images.

```
import urllib.request
```

urllib.request is a Python module for fetching URLs (Uniform Resource Locators). It offers a very simple interface in the form of the *urlopen* function. This is capable of fetching URLs using a variety of different protocols.

```
plateClassifier = cv2.CascadeClassifier("haarcascade_russian_plate_number.xml")
```

Using haarcascade to detect licence plates with OpenCV and Python.

2.3 Image processing Using OpenCV

OpenCV, or the Open Source Computer Vision library, started out as a research project at Intel. It's currently the largest computer vision library in terms of the sheer number of functions it holds.

OpenCV contains implementations of more than 2500 algorithms! It is freely available for commercial as well as academic purposes. And the joy doesn't end there! The library has interfaces for multiple languages, including Python, Java, and C++. The first OpenCV version, 1.0, was released in 2006, and the OpenCV community has grown by leaps and bounds since then.

Now, let's turn our attention to the idea behind this article –the plethora of functions OpenCV offers! We will be looking at OpenCV from the perspective of a data scientist and learning about some functions that make the task of developing and understanding computer vision models easier.[4]

2.3.1 Reading, Writing and Displaying Images

Machines see and process everything using numbers, including images and text. How do you convert images to pixel values:

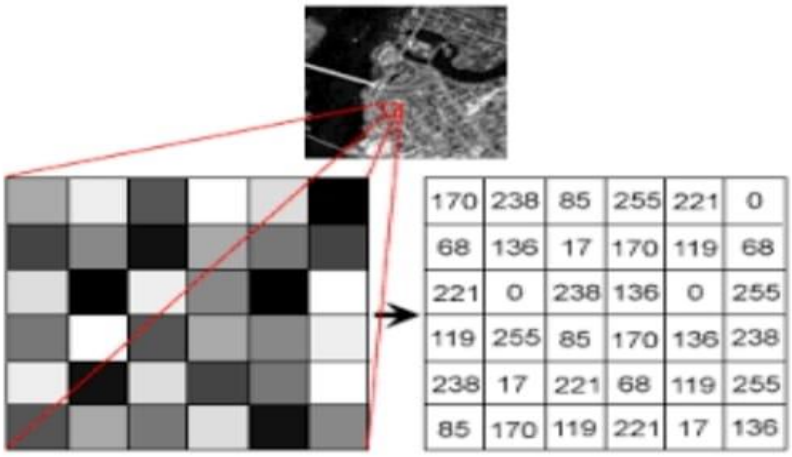


Figure2.4 Digitization and pixel values of an image

Every number represents the pixel intensity at that particular location. In the above image, I have shown the pixel values for a grayscale image where every pixel contains only one value, i.e., the intensity of the black color at that location. Color images will have multiple values for a single pixel. These values represent the intensity of respective channels–Red, Green, and Blue channels for RGB images, for instance. Reading and writing images is essential to any computer vision project. And the OpenCV library makes this function a whole lot easier.

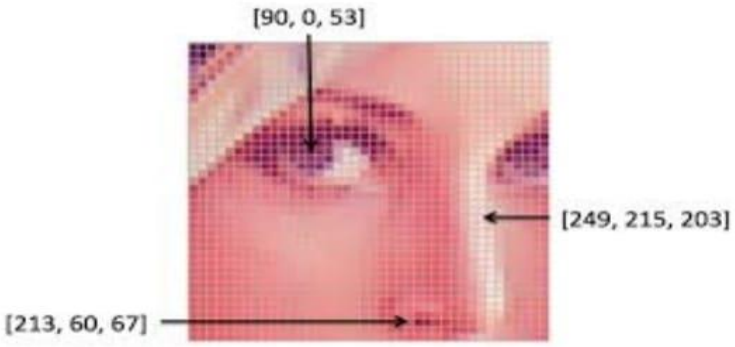


Figure2.5 pixel in image processing

let’s see how to import an image into machines using OpenCV

```python
import cv2
import pytesseract

img = cv2.imread(
    "img/Cilisos-Featured-Image-Car-Numberplate-MAHAL-1024x575.jpg")

cv2.imshow("img", img)


cv2.waitKey(0)

```

By default, the imread function reads images in the BGR (Blue-Green-Red) format. We can read images in different formats using extra flags in the imread function:

cv2.IMREAD_COLOR: Default flag for loading a color image.
cv2.IMREAD_GRAYSCALE: Loads images in grayscale format.
cv2.IMREAD_UNCHANGED: Loads images in their given format, including the alpha channel. The Alpha channel stores the transparency information—the higher the value of the alpha channel, the more opaque the pixel.

2.3.1.1 Reading an images

Use the imread () function in OpenCV to read an image. Here's the syntax:

imread(filename, flags)

It takes two arguments:

1- The first argument is the image name, which requires a fully qualified pathname to the file.

2- The second argument is an optional flag that lets you specify how the image should be represented. OpenCV offers several options for this flag, but those that are most common include:

cv2.IMREAD_UNCHANGED or -1

cv2.IMREAD_GRAYSCALE or 0

cv2.IMREAD_COLOR or 1

The default value for flags is 1, which will be read in the image as a Colored image.

OpenCV reads color images in BGR format, whereas most other computer vision libraries use the RGB channel format.

As shown in the code sections below, we will first read the test image, using all three flag values described above.

2.3.1.2 Displaying an Image

In OpenCV, you display an image using the imshow() function. Here's the syntax:

imshow(window_name, image)

This function also takes two arguments:

1- The first argument is the window name that will be displayed on the window.

2- The second argument is the image that you want to display.

To display multiple images at once, specify a new window name for every image you want to display.

The imshow() function is designed to be used along with the waitKey() and destroyAllWindows() / destroyWindow() functions.

The waitKey() function is a keyboard-binding function.

It takes a single argument, which is the time (in milliseconds) for which the window will be displayed.

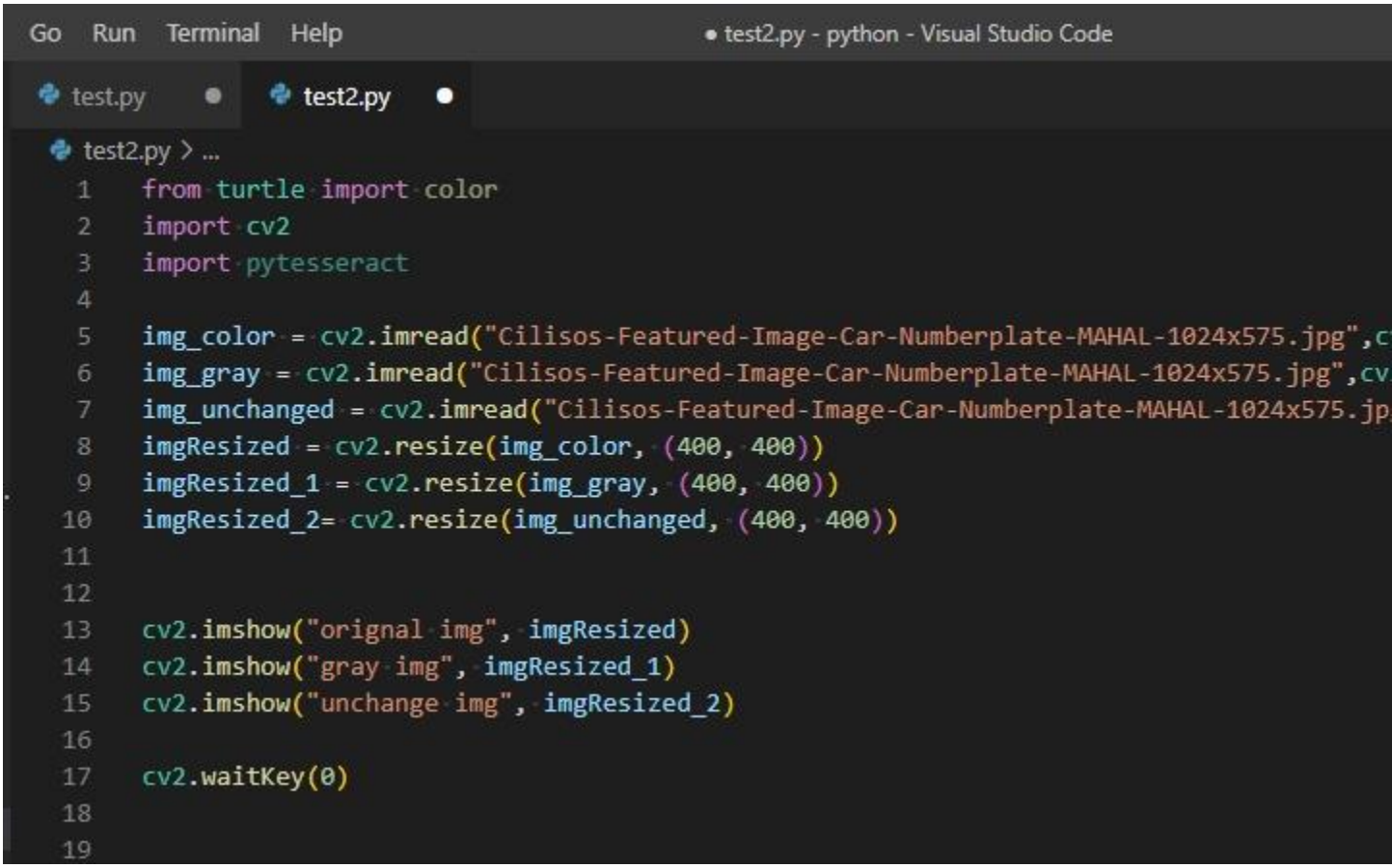
If the user presses any key within this time period, the program continues.

If 0 is passed, the program waits indefinitely for a keystroke.

can also set the function to detect specific keystrokes like the Q key or the ESC key on the keyboard, thereby telling more explicitly which key will trigger which behavior.

The function destroyAllWindows() destroys all the windows we created. If a specific window needs to be destroyed, give that exact window name as the argument. Using destroyAllWindows() also clears the window or image from the main memory of the system.

The code examples below show how the imshow() function is used to display the images you read in.



Output:

Python

```
1 cv2.imwrite('grayscale.jpg',img_grayscale)
```

2.3.2 Reading, Writing and Displaying a video using OpenCV

Reading and writing videos in OpenCV is very similar to reading and writing images. A video is nothing but a series of images that are often referred to as frames. So, all you need to do is loop over all the frames in a video sequence and then process one frame at a time. In this post, we will demonstrate how to

read, display, and write videos from a file and an image sequence. Let's see by an example how to capture video from the camera
and display it.

```
1  import numpy as np
2  import cv2
3
4  cap = cv2.VideoCapture(0)
5
6  while(True):
7      ret, frame = cap.read()
8
9      cv2.imshow('frame',frame)
10     if cv2.waitKey(1) & 0xFF == ord('q'):
11         break
12
13 cap.release()
14 cv2.destroyAllWindows()
```

cv2.VideoCapture(0) will open the default camera.

select the second camera by passing 1, third by passing 2 and so on.

This creates a VideoCapture object ("cap" here).

cap.read() capture frame by frame. This returns two values,

frame and ret. If the frame is read correctly, ret will be True otherwise

False.

cv2.waitKey(1) & 0xFF == ord(„q‟) will exit the video when

„q‟ is pressed.

cap.release() closes video file or capturing device.

If you want to play a video from a file, just change the

cv2.VideoCapture(0) function in the above code by giving the file

path as **cv2.VideoCapture(F:/downloads/Python.mp4‟).**

2.3.3 Resizing Images

convert the image to grayscale from RGB color. By essentially doing this, we are reducing the number of colors in the image, which might disturb the OCR detection. Since we need to focus more on the edges of the image, using a grayscale image helps us to do that more efficiently.

Grayscale imaging is common in all image processing steps. This speeds up the following process since we no longer have to deal with the color details when processing an image.

Code for that:

test3.py > ...

```
1   import cv2
2   import pytesseract
3
4   img = cv2.imread(
5       "img/Cilisos-Featured-Image-Car-Numberplate-MAHAL-1024x575.jpg")
6
7   cv2.imshow("original imag", img)
8
9   # resize image
10  imgResized = cv2.resize(img, (640, 480))
11  cv2.imshow("car plate", imgResized)
12
13  cv2.waitKey(0)
14
```

Output:

2.3.4 Blurring images to remove noise

Image blurring is achieved by convolving the image with a low-pass filter kernel. It is useful for removing noise. It actually removes high frequency content (e.g., noise, edges) from the image. So edges are blurred a little bit in this operation (there are also blurring techniques which don't blur the edges). OpenCV provides four main types of blurring techniques.

1. Averaging

This is done by convolving an image with a normalized box filter. It simply takes the average of all the pixels under the kernel area and replaces the central element. This is done by the function **cv.blur()** or **cv.boxFilter()**.

2. Gaussian Blurring

In this method, instead of a box filter, a Gaussian kernel is used. It is done with the function, **cv.GaussianBlur()**. We should specify the width and height of the kernel, which should be positive and odd. We also specify the standard deviation in the X and Y directions, sigmaX and sigmaY, respectively. If only sigmaX is specified, sigmaY is taken as the same as sigmaX. If both are given as zeros, they

are calculated from the kernel size. Gaussian blurring is highly effective in removing Gaussian noise from an image.

3. Median Blurring

The function **cv.medianBlur()** takes the median of all the pixels under the kernel area, and the central element is replaced with this median value. This is highly effective against noise in an image. Interestingly, in the above filters, the central element is a newly calculated value, which may be a pixel value in the image or a new value. But in median blurring, the central element is always replaced by some pixel value in the image. It reduces the noise effectively. Its kernel size should be a positive odd integer.

4. Bilateral Filtering

cv.bilateralFilter() It is highly effective in noise removal while keeping edges sharp. But the operation is slower compared to other filters. We already saw that a Gaussian filter takes the neighbourhood around the pixel and finds its Gaussian weighted average. This Gaussian filter is a function of space alone; that is, nearby pixels are considered while filtering. It doesn't consider whether pixels have almost the same intensity. It doesn't consider whether a pixel is an edge pixel or not. So it blurs the edges also, which we don't want to do.

Bilateral filtering also takes a Gaussian filter in space, but one more Gaussian filter, which is a function of pixel difference. The Gaussian function of space makes sure that only nearby pixels are considered for blurring, while the Gaussian function of intensity difference makes sure that only those pixels with similar intensities to the central pixel are considered for blurring. So it preserves the edges since pixels at the edges will have large intensity variations.

We used bilateral filter in the code.

Example:

test3.py > ...

```python
import cv2
import pytesseract

img = cv2.imread(
    "img/Cilisos-Featured-Image-Car-Numberplate-MAHAL-1024x575.jpg")

cv2.imshow("original imag", img)

# remove noise
blurredImg = cv2.bilateralFilter(img, 10, 20, 20)
cv2.imshow("blurredImg", blurredImg)


cv2.waitKey(0)

```

Original image:

Blurred image:

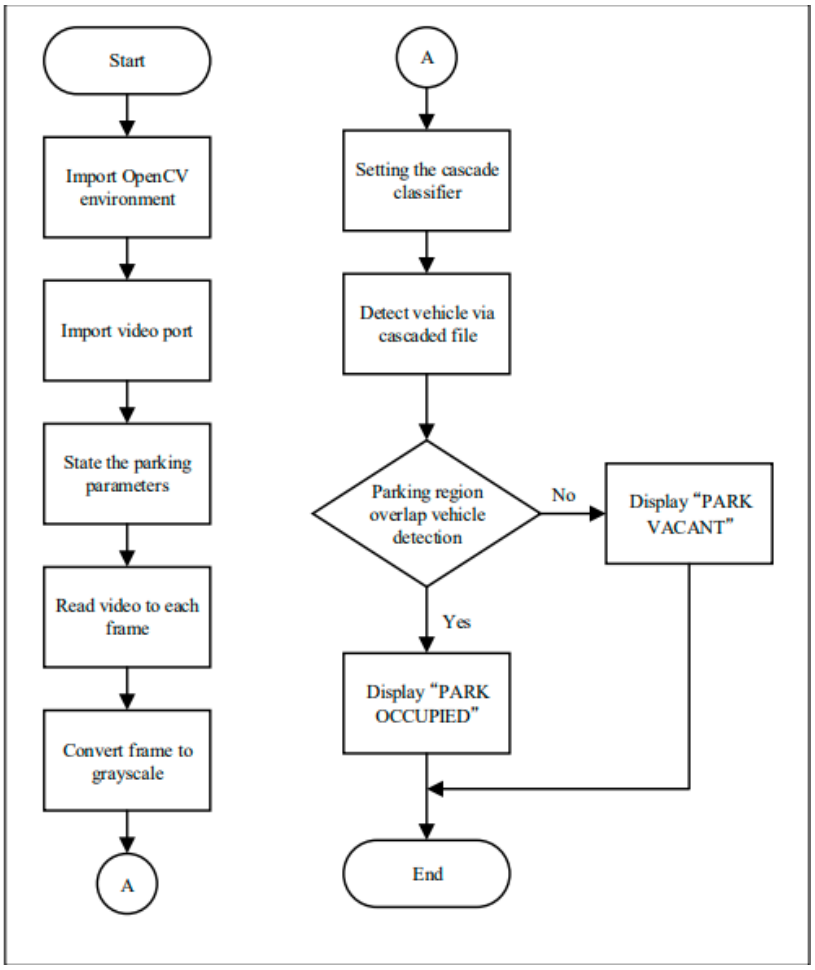


Figure 2.6 Flowchart of Python file for parking detection.

2.4 Plate Recognition Using Python

2.4.1 Tesseract: The Tesseract package contains an OCR engine – libtesseract – and a command line programme – tesseract. Tesseract has Unicode (UTF-8) support and can

It recognizes more than 100 languages "out of the box". It can be trained to recognize other languages. Tesseract supports various output formats: plain-text, hocr (html), pdf. Tesseract is available for Linux, Windows, and Mac OS X. However, due to limited resources, only Windows and Ubuntu are rigorously tested by developers.

Optical character recognition (OCR), sometimes called optical character reading, is the process of reading and converting text from images into a machine-readable format like a string. Images of old documents, receipts, plates, and house numbers can all contain useful text. Reading this text manually from images can be time-consuming and labor-intensive. This is where OCR comes into play. OCR is an important task in computer vision as it allows automatic undefined of text from various sources.[5]

IMPLEMENTATION:

When the car enters the parking lot, the image of the licence plate will be captured. By using image processing methods, the captured image will be processed. After finding the contour of the plate The processed image is then fed to the Tesseract algorithm for character recognition. The extracted plate number is then checked to see whether it exists in the database or not. If the plate already exists in the database, we extract the information from it and produce a bill based on the time of stay. If the plate does not exist in the database, we upload it with the time of entry to the database.

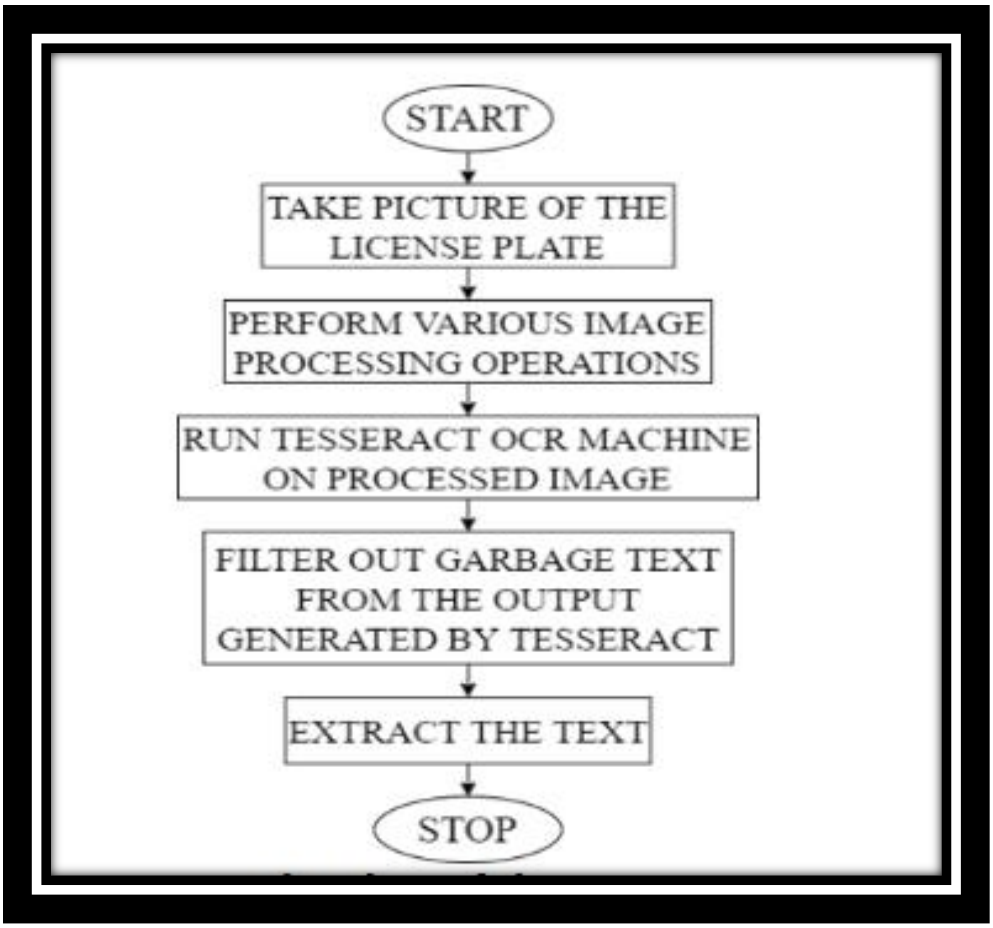


Figure2.7 Flowchart of Plate Recognition

2.4.2 Extracting features from Plate

License plate detection is identifying the part of the car that is predicted to be the number plate. Recognition is identifying the values that make up the license plate.

License plate detection and recognition is the technology that uses computer vision to detect and recognize a license plate from an input image of a car.

This technology applies in many areas. On roads, it is used to identify cars that are breaking the traffic rules. In security, it is used to capture the license plates of the vehicles getting into and out of certain premises. In parking lots, it is used to capture the license plates of the cars being parked.

Python gives us the ability to create our license plate detection and recognition program. We achieve this by using three of its libraries; pytesseract, imutils, and OpenCv.

Reading a Car License Plate

we'll read the text from the following foreign car license plate image

Figure 2.8 Normal image captured by camera

To read the text from an image the first step is to open the image. You can do so via the open() method from the image object of the Pillow library. Next, to actually read the text from an image, you need to pass the image object you just opened to the image_to_string() method of the Pytesseract module. The image_to_string() method converts the image text into a Python string which you can then use however you want. We're simply going to print the string to our screen using the print() method. Execute the following script to read the text from the car number plate image.

```
1  import pytesseract
2
3  print(pytesseract.image_to_string("vodz4gdb.jpg"))
4
```

Output:

```
cv2.waitKey(0)
```

```
MAH 41
```

Figure2.9 Data after detection and recognition

The output shows that though Tesseract OCR is capable of reading text from an image, it is not 100% correct. An accuracy rate of less than 100% is typical with all OCR engines.

As we have seen from the above, we did the ticketless parking in three steps:

Step 1: we gave the trained data of plates to our algorithm

Step 2: The algorithm can recognize plates given to it from the pictures of the cars we are getting from the camera.

Step3: we give the plate number photo to another library (pytesseract) to get numbers and alphabets from it, to send the result to be stored in the server.

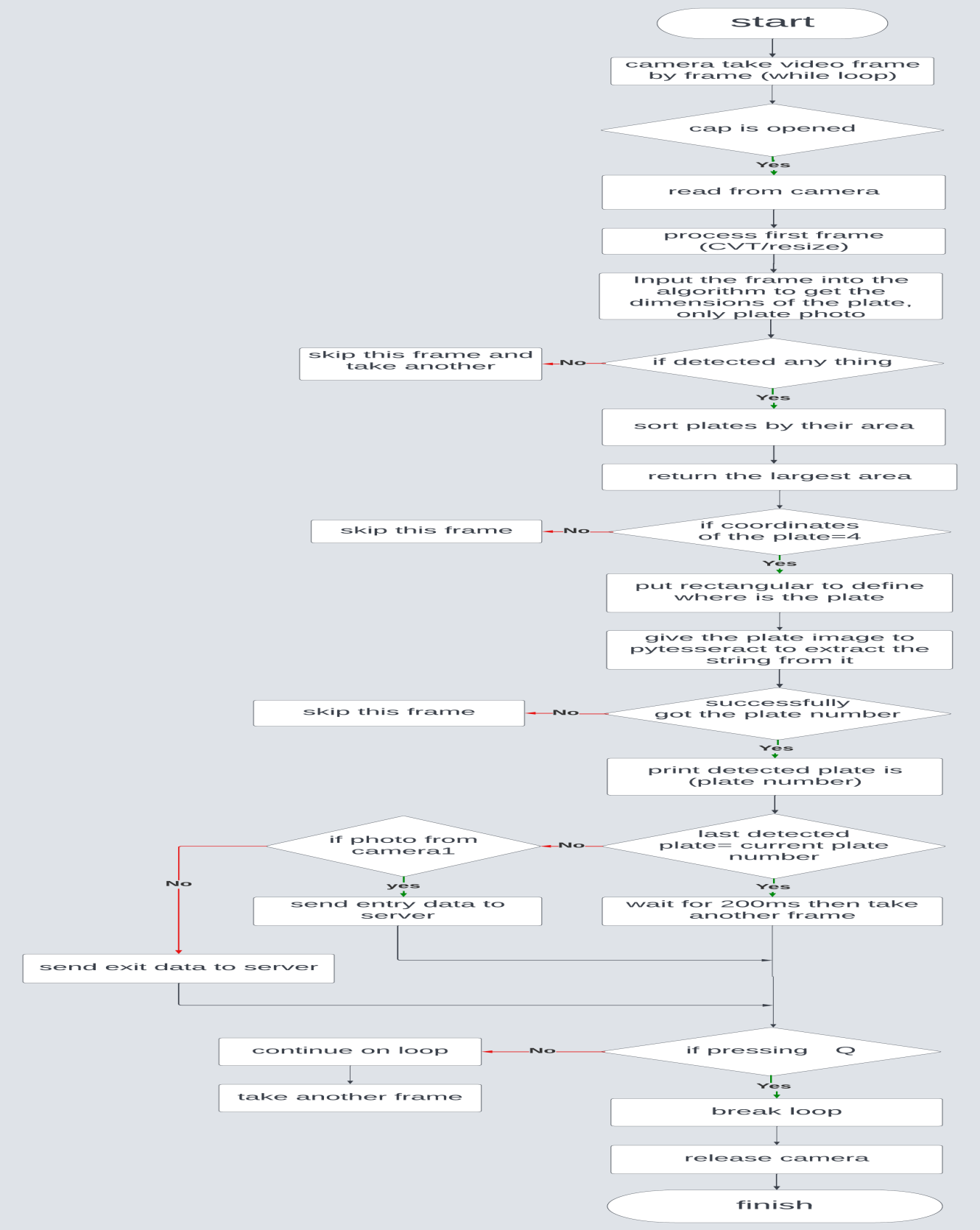


Figure2.10 Flow chart of the final code implemented

2.5 Software Implementation

1- Started by importing libraries needed and defining variables

```
In [1]: import pytesseract
        import cv2
        import urllib.request
        from concurrent.futures import process
        import numpy as np
        import time
```

```
In [3]: # configuration variables
        skipCount = 200
        waitInterval = 1
        hostname = '192.168.1.8'
```

1- Defining a function that gives the trained data to the algorithm and checks if it detects any plate, after that it sorts the plates ascendingly return the largest plate area
It returns four coordination of the plate the x and width (w) coordinate, the y and the height(h) coordinate.

```
def detectPlate(img):
    plateClassifier = cv2.CascadeClassifier('haarcascade_russian_plate_number.xml')

    # detect plates
    plates = plateClassifier.detectMultiScale(img)

    # sort by area
    if len(plates) > 0:
        # sort plates by size in ascending order
        plates = sorted(plates, key = lambda plate: plate[2]*plate[3])

        # select the plate with the biggest area and extract its coordinates
        return plates[-1]

    return ()
```

2- Start reading from camera frame after frame by implementing a while loop.
3- Processing the photo like mentioned in the opencv part.

4- Give the photo to the algorithm and extract the plate if found, or skip the iteration of the loop and take another frame.

```
while True:
    # iterate over the defined capture devices
    for i, cap in enumerate(caps):
        # attempt to read image if the capture device is open
        if cap.isOpened():
            # read image
            success, img = cap.read()

            # image read successfully
            if success:
                cv2.imshow(f"cam{i}", img)

                # if in timeout period, skip this frame
                if timeouts[i] > 0:
                    timeouts[i] -= 1
                else:
                    # gray scale
                    greyImg = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

                    # smooth the image to remove noise
                    greyImg = cv2.bilateralFilter(greyImg, 5, 15, 15)

                    # perform detection
                    coordinates = detectPlate(greyImg)
```

5- Put rectangular into the image and define the recognized plate, then show it.

```
        # process the plate img if one was found
        if len(coordinates) == 4:
            (x, y, w, h) = coordinates

            # display original image with ROI
            cv2.rectangle(img, (x, y), (x+w, y+h), (0, 0, 255), 2)
            cv2.putText(img, "Plate", (int(x*0.75), int(y*0.95)), cv2.FONT_HERSHEY_COMPLEX, 1, (0, 0, 255), 2)
            cv2.imshow(f"cam{i}", img)
```

6- Giving the recognized plate image to pytesseract to get the numbers and characters from the plate. If you couldn’t get the plate number, skip the frame and take another

```
        # extract plate from img
        plateImg = greyImg[y:y+h,x:x+w]

        # convert to binary image
        (threshold, plateImg) = cv2.threshold(plateImg, 127, 255, cv2.THRESH_OTSU)

        # extract plate number using tesseract
        plateNumber = pytesseract.image_to_string(plateImg)

        # strip whitespace characters
        plateNumber = plateNumber.strip()

        if not len(plateNumber):
            continue

        print("Detected license plate Number is:", plateNumber)
```

7- If the detected plate is equal to the last detected plate, delay for 200ms. We did that to give time to the current car to pass without continuing to take pictures of it and tiring the processor without any need.

```
    if lastDetectedPlates[i] == plateNumber:
        timeouts[i] = skipCount
        print(f"Duplicate plate number detected!, skipping {skipCount*waitInterval}ms")
    else:
```

8- If the detected plate is not equal to the last detected plate,

A) If the outcome is from camera 0, then send the plate number to the server as entry data

B) If the outcome is from camera 1, then send the data as exit data.

We implemented try, except in our code, to be able to continue code even if there is an error in sending data to the server.

```
lastDetectedPlates[i] = plateNumber
# contact server to report detected plate number
if i == 0: # entry camera
    print("Sending entry data to server")
    try: print(urllib.request.urlopen(f"http://{hostname}/SmartParking/api/carEntry.php?car_plate_number={plateNumber}").read())
    except: print("No response from server")
else: # exit camera
    print(f"Sending exit data to server")
    try: print(urllib.request.urlopen(f"http://{hostname}/SmartParking/api/carExit.php?car_plate_number={plateNumber}").read())
    except: print("No response from server")
```

9- We want to break the loop and stop the camera. We assigned a button "q" to exit the loop and release the camera.

10- Otherwise, continue on looping and take another frame.

```
    if cv2.waitKey(waitInterval) == ord('q'):
        break

# release
for cap in caps:
    cap.release()
```

So, by these steps we implemented a ticketless parking

The final code:

```
In [1]: import pytesseract
        import cv2
        import urllib.request
        from concurrent.futures import process
        import numpy as np
        import time
```

In [*]:
```
# configuration variables
skipCount = 200
waitInterval = 1
hostname = '192.168.1.8'
caps = [
    cv2.VideoCapture(0)#, cv2.CAP_DSHOW),
]


def detectPlate(img):
    plateClassifier = cv2.CascadeClassifier('haarcascade_russian_plate_number.xml')

    # detect plates
    plates = plateClassifier.detectMultiScale(img)

    # sort by area
    if len(plates) > 0:
        # sort plates by size in ascending order
        plates = sorted(plates, key = lambda plate: plate[2]*plate[3])

        # select the plate with the biggest area and extract its coordinates
        return plates[-1]

    return ()

# runtime variables
timeouts = [ 0 ] * len(caps)
lastDetectedPlates = [ '' ] * len(caps)
# # read image
# img = cv2.imread('img/car0.jpg')




# # resize image
# img = cv2.resize(img, (640, 480))
```

```
# read
while True:
    # iterate over the defined capture devices
    for i, cap in enumerate(caps):
        # attempt to read image if the capture device is open
        if cap.isOpened():
            # read image
            success, img = cap.read()

            # image read successfully
            if success:
                cv2.imshow(f"cam{i}", img)

                # if in timeout period, skip this frame
                if timeouts[i] > 0:
                    timeouts[i] -= 1
                else:
                    # gray scale
                    greyImg = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

                    # smooth the image to remove noise
                    greyImg = cv2.bilateralFilter(greyImg, 5, 15, 15)

                    # perform detection
                    coordinates = detectPlate(greyImg)

                    # process the plate img if one was found
                    if len(coordinates) == 4:
                        (x, y, w, h) = coordinates

                        # display original image with ROI
                        cv2.rectangle(img, (x, y), (x+w, y+h), (0, 0, 255), 2)
                        cv2.putText(img, "Plate", (int(x*0.75), int(y*0.95)), cv2.FONT_HERSHEY_COMPLEX, 1, (0, 0, 255), 2)
                        cv2.imshow(f"cam{i}", img)

                        # # display plate
                        # cv2.imshow(f"plate{i}", plateImg)
```

```
                    # strip whitespace characters
                    plateNumber = plateNumber.strip()

                    if not len(plateNumber):
                        continue

                    print("Detected license plate Number is:", plateNumber)
                    time.sleep(3)
                    if lastDetectedPlates[i] == plateNumber:
                        timeouts[i] = skipCount
                        print(f"Duplicate plate number detected!, skipping {skipCount*waitInterval}ms")
                    else:
                        lastDetectedPlates[i] = plateNumber
                        # contact server to report detected plate number
                        encodedPlateNumber = urllib.parse.quote(plateNumber)
                        if i == 0: # entry camera
                            print("Sending entry data to server")
                            try: print(urllib.request.urlopen(f"http://{hostname}/SmartParking/api/carEntry.php?car_plate_nu
                            except: print("No response from server")
                        else: # exit camera
                            print(f"Sending exit data to server")
                            try: print(urllib.request.urlopen(f"http://{hostname}/SmartParking/api/carExit.php?car_plate_numb
                            except: print("No response from server")



    if cv2.waitKey(waitInterval) == ord('q'):
        break

# release
for cap in caps:
    cap.release()


# # process image
# detectPlate(img)

cv2.destroyAllWindows()
```

CONCLUSION:

After performing these steps, we can easily detect the license number from the image or video of the car through the camera. After that, the information about the vehicle and its owner will be directly saved to the database. Hence, through this, we can automate the car parking process and allow entry only to the authorized vehicles.

So, by these steps, we implemented ticketless parking.

Chapter 3

Web Services

3.1 Description

A website has been developed to receive data from an embedded device through a WIFI module and show it. Using this website, the user (car owner) can know whether there is a free location in the parking or not, and also where the free location is in the parking and the nearest route to park. He can also know how much money he must pay if the car is parked for a period of time in the parking.

3.2 Background and Related Knowledge

There are some concepts we should know to understand how our website works, such as: front-end, back-end, and database.

3.2.1 Frontend Development

Front-end development refers to that area of web development that focuses on what the users see on their end. It involves transforming the code built by backend developers into a graphical interface, making sure that the data is presented in an easy-to-read and understand format. Without frontend development, all you would see on a website or web application would be undecipherable codes (unless you're a developer, too, of course). But because of front-end developers, people with no coding background can easily understand and use web applications and websites. Everything you see when you visit Google Apps, Canva, Facebook, and other web applications is a product of backend and frontend developers' working together.[6]

The very first website, created in 1991, was equal parts revolutionary and boring. Tim Berners-Lee invented the World Wide Web in 1989 while working as a software engineer at the European Organization for Nuclear Research (CERN). By 1990, Berners-Lee created three fundamental technologies that are still used today—Hyper Text Markup Language (HTML), Uniform Resource Identifier (URI or better known as URL), and Hypertext Transfer Protocol (HTTP). HTML is the language of the web, the URL serves as the unique address of a resource, and HTTP is the protocol that determines how a resource is retrieved. This first website ever consisted of a few links about the world wide web project. While a front-end developer was not needed yet, there was a language and protocol that served as the foundation for more exciting websites to come. HTML tables were an evolutionary step to organize content on websites and graphics added visual appeal in the early 1990s. Early websites were still clunky and unintuitive by today's standards, but it was an important step in web development. Soon, front-end development would flourish with the introduction JavaScript in 1995 and Flash in 1996. By 1998, Cascading Style Sheets (CSS) were created and all the tools a front-end developer would need were now available to make modern sites.[7]

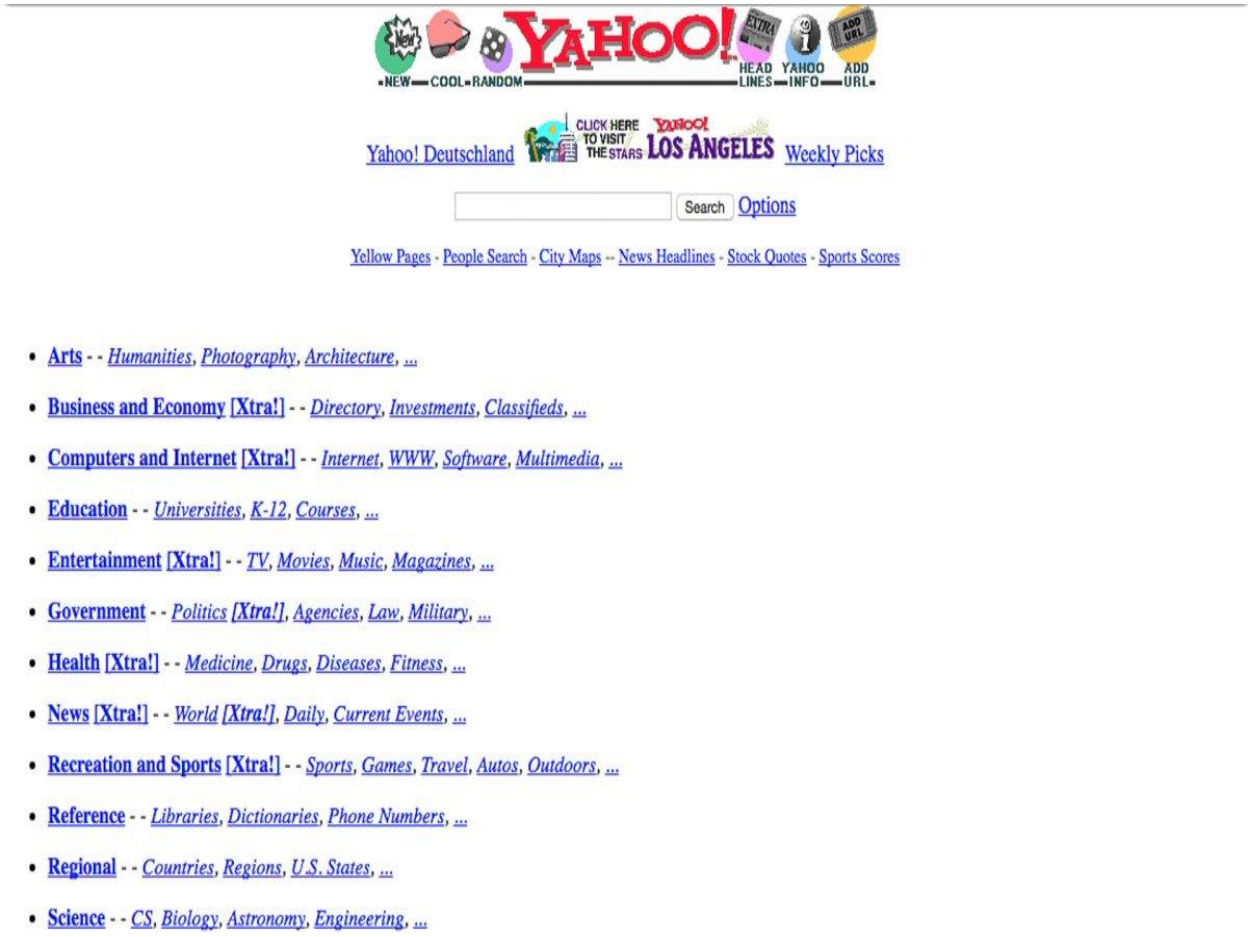


Figure 3.1: Yahoo Website Before Front-End Development

Web Technologies Involved in Frontend Development

Frontend developers use several web technologies to transform coded data into user-friendly interfaces. Among these are Hyper Text Markup Language (HTML), Cascading Style Sheets (CSS), and JavaScript. Below are brief descriptions of the three technologies that front-end developers must be familiar with.

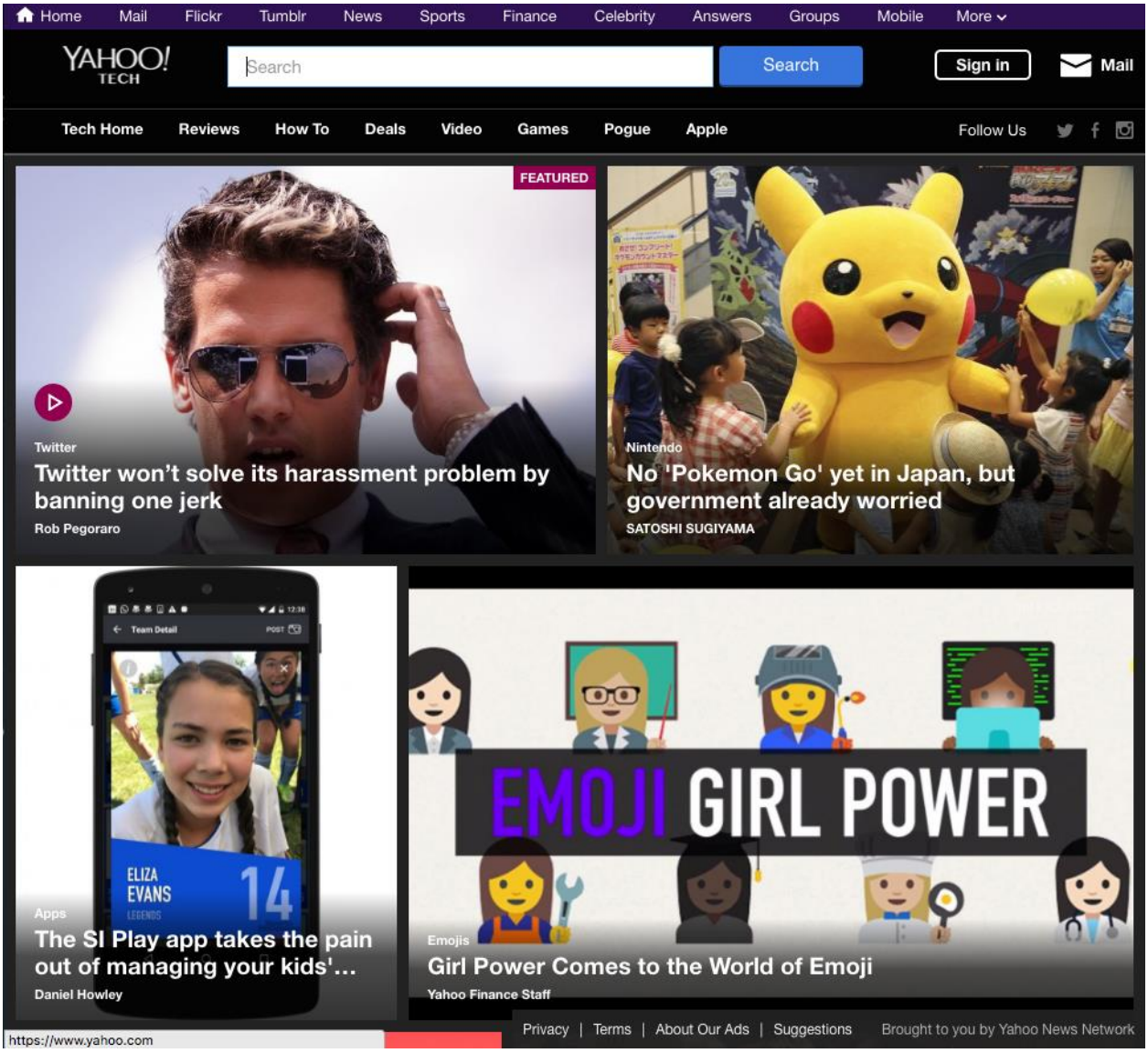


Figure 3.2: Yahoo Website After Front-End Development

1. HTML

HTML is the building block of websites. It is the programming language used to describe and mark content, so a browser displays it correctly. For example, an image on a blog post would appear as an <img> in HTML code, so browsers will know that they need to display an image .[6]

2. CSS

CSS looks more like a set of instructions that control a webpage's style and structure than a programming language. It helps developers manage a website or web application's formatting, presentation, and layout. While HTML defines elements on a page, CSS dictates how users see the content. For instance, it controls the size, border, and alignment of an image in a blog post [6].

3. JavaScript

Front-end developers can already create websites using HTML and CSS. In fact, it wasn't until 1995 that JavaScript emerged. However, it is now difficult to imagine websites without JavaScript as it enables developers to make sites interactive. The programming language can change website content based on a user's action. Techslang's Weekly Poll, for example, was created using JavaScript. Selecting an answer and clicking "Vote" would display the total number of votes for each option [6].

4.DOM

When writing web pages and apps, one of the most common things you'll want to do is manipulate the document structure in some way. This is usually done by using the Document Object Model (DOM), a set of APIs for controlling HTML and styling information that make heavy use of the document object along with some other interesting APIs that can alter your environment in interesting ways.

3.2.2 Backend Development

Backend development is the skill that powers the web. Yet it does it modestly, without fanfare—allowing people to browse their favorite sites without even knowing about all the work put in by the backend developer or team. Backend development languages handle the 'behindthe-scenes' functionality of web applications. It's code that connects the web to a database, manages user connections, and powers the web application itself. Backend development works in tandem with the front end to deliver the final product to the end user. Backend developers are primarily focused on how a website works. They write code that focuses on the functionality and logic powering the application they're working on, and the technology they work on is never directly seen by users [8].

The tech of the back end is a combination of servers, applications, and databases. Responsibilities of backend programmers could involve writing APIs, writing code to interact with a database, creating libraries, working on business processes and data architecture, and much more [8].

Web Technologies Involved in Backend Development

1. Node.js

Node.js is a server-side platform built on Google Chrome's JavaScript Engine (V8.engine). Node.js was developed by Ryan Dahl in 2009 and its latest version is v0.10.36. The definition of Node.js as supplied by its official documentation is as follows:

Node.js is a platform built on Chrome's JavaScript runtime for easily building fast and scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

Node.js is an open-source, cross-platform runtime environment for developing server-side and networking applications. Node.js applications are written in JavaScript and can be run within the Node.js runtime on OS X, Microsoft Windows, and Linux. Node.js also provides a rich library of various JavaScript modules, which simplifies the development of web applications using Node.js to a great extent. [9]

2. Express.js

Express.js, or simply Express, is a back-end web application framework for Node.js, released as free and open-source software under the MIT License. It is designed for building web applications and APIs. It has been called the de facto standard server framework for Node.js.[10]

Express is a minimal and flexible Node.js web application framework that provides a robust set of features to develop web and mobile applications. It facilitates the rapid development of Node-based Web applications. The following are some of the core features of the Express framework:

- It allows you to set up middleware to respond to HTTP requests.
- defines a routing table which is used to perform different actions based on HTTP

The method and URL

• Enables dynamic HTML page rendering by passing arguments to templates. [11]

3. HTTP

What is HTTP?

The Hypertext Transfer Protocol (HTTP) is the foundation of the World Wide Web and is used to load web pages using hypertext links. HTTP is an application layer protocol designed to transfer information between networked devices and runs on top of other layers of the network protocol stack. A typical flow over HTTP involves a client machine making a request to a server, which then sends a response message.

What's in an HTTP request?

An HTTP request is the way internet communications platforms such as web browsers ask for the information they need to load a website.

Each HTTP request made across the Internet carries with it a series of encoded data that carries different types of information. A typical HTTP request contains:

• HTTP version type

• a URL

• an HTTP method

• HTTP request headers

• Optional HTTP body.

What's an HTTP method?

An HTTP method, sometimes referred to as an HTTP verb, indicates the action that the HTTP request expects from the queried server. For example, two of the most common HTTP methods are 'GET' and 'POST'; a 'GET' request expects information back in return (usually in the form of a website), while a 'POST' request typically indicates that the client is submitting information to the web server (such as form information, e.g. a submitted slot number and car plate number).

What's in an HTTP request body?

The body of a request is the part that contains the "body" of information the request is transferring. The body of an HTTP request contains any information

being submitted to the web server, such as a slot number and car plate number, or any other data entered into a form.

What's in an HTTP response?

An HTTP response is what web clients (often browsers) receive from an Internet server in answer to an HTTP request. These responses communicate valuable information based on

What was asked for in the HTTP request?

A typical HTTP response contains:

• an HTTP status code

• HTTP response headers

• Optional HTTP body [12]

4. Ajax

Asynchronous JavaScript and XML (Ajax) refer to a group of technologies that are used to develop web applications. By combining these technologies, web pages appear more responsive since small packets of data are exchanged with the server and web pages are not reloaded each time a user makes an input change. Ajax enables a web application user to interact with a web page without the interruption of constant web page reloading. Website interaction happens quickly, with only portions of the page reloading and refreshing.

Ajax is made up of the following technologies:

• XHTML and CSS for presenting information.

• uses the Document Object Model (DOM) for dynamically interacting with and displaying the presented information.

• XML Http request object to manipulate data asynchronously with the web server.

• XML, HTML, and XSLT for data interchange and manipulation.

• JavaScript for binding data requests and information display.

Ajax incorporates these technologies to create a new approach to developing web Applications .Ajax defines a method of initiating client to server

communication without page reloads. It provides a way to enable partial page updates. From a web page user perspective, it means improved interaction with a web application, which gives the user more control of their environment, similar to that of a desktop application.

In a traditional web application, HTTP requests that are initiated by the user's interaction with the web interface are made to a web server. The web server processes the request and returns an HTML page to the client. During HTTP transport, the user is unable to interact with the web application.

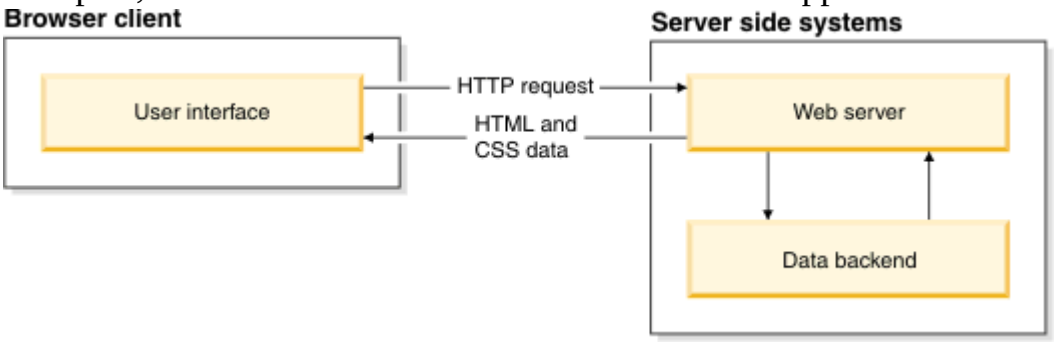


Figure 3.3: Traditional WebApp Model

In an Ajax web application, the user is not interrupted in interactions with the web application. The Ajax engine, or JavaScript interpreter, enables the user to interact with the web application independent of HTTP transport to and from the server by rendering the interface and handling communications with the server on the user's behalf. [13]

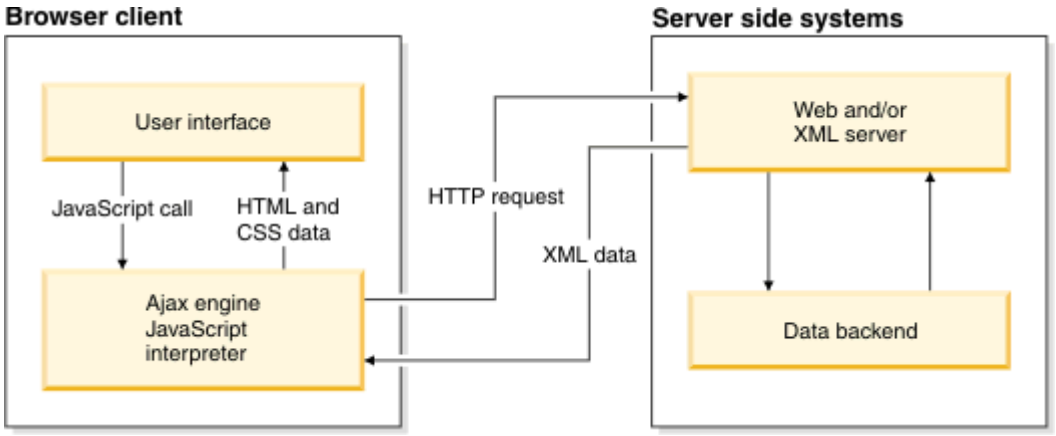


Figure 3.4: Ajax WebApp Model

5. Database

A database is an organized collection of structured information, or data, typically stored electronically on a computer system. A database is usually controlled by a database management system (DBMS). Together, the data and the DBMS, along with the applications that are associated with them, are referred to as a database system, often shortened to just "database."

Data within the most common types of databases in operation today is typically modelled in rows and columns in a series of tables to make processing and data querying efficient. The data can then be easily accessed, managed, modified, updated, controlled, and organized. Most databases use a structured query language (SQL) for writing and querying data [14].

But for the facility, we used a [mysql data structure] as a database. This can be considered an in-memory database that gets reinitialized when we restart the program..

3.3 Implementation

XAMPP:

We use the laptop as a local server using XAMPP. It allows us to work on a local server and test local copies of websites using PHP code and MySQL databases. Once xampp is running, you can use a browser to access the local copy by entering an URL such as http://localhost / or http://127.0.0.1 / (both of which mean the same thing).XAMPP is a free and open-source cross-platform web server solution stack package. Any HTML page with CSS (and Javascript) can be edited and viewed in any browser without any special tools. You just open a file with your browser to view it and use any editor to edit your HTML file.

Steps:

Install XAMPP, paste the web file to htdocs, open the browser, type "localhost", and then you can access the website that you pasted directly from your browser.

XAMP server can help in running

- Apache
- MySQL
- Filezilla
- TomCat
- Mercury

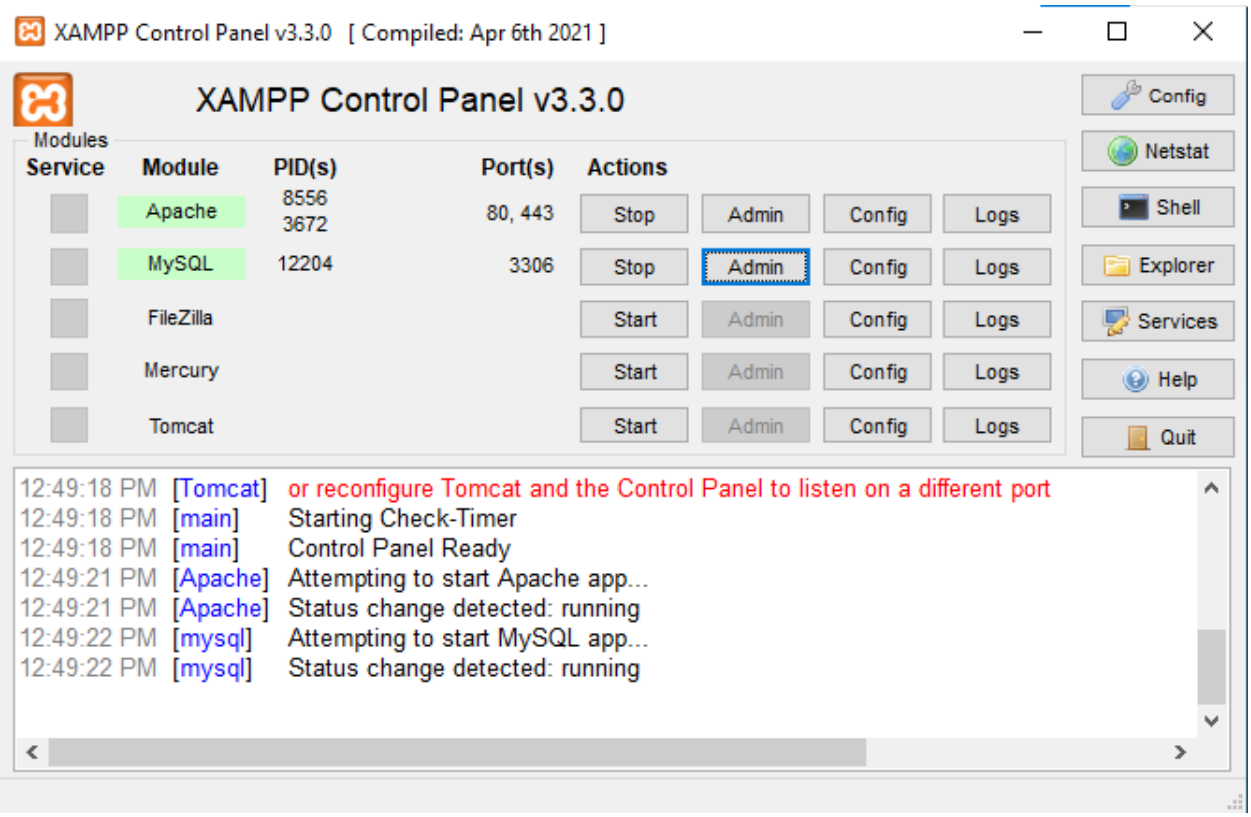


Figure3.5 XAMPP Control Panel

Apache:

It is a server that can run PHP.

We use it because it is open source software, available for free, fast, reliable, and secure. [15].

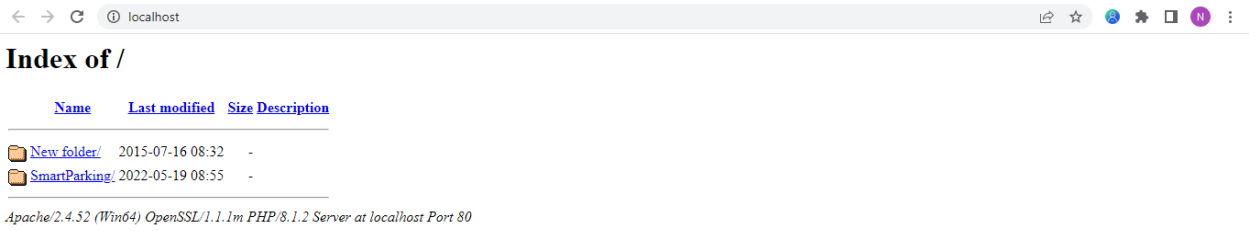


Figure 3.6 Implemented Server

MYSQL:

It is a database server and can store data.
We use it because it is open source, very fast, reliable, scalable, and easy to use. Using [SQL] Structured Query Language, you can access and manipulate MYSQL databases. [16].

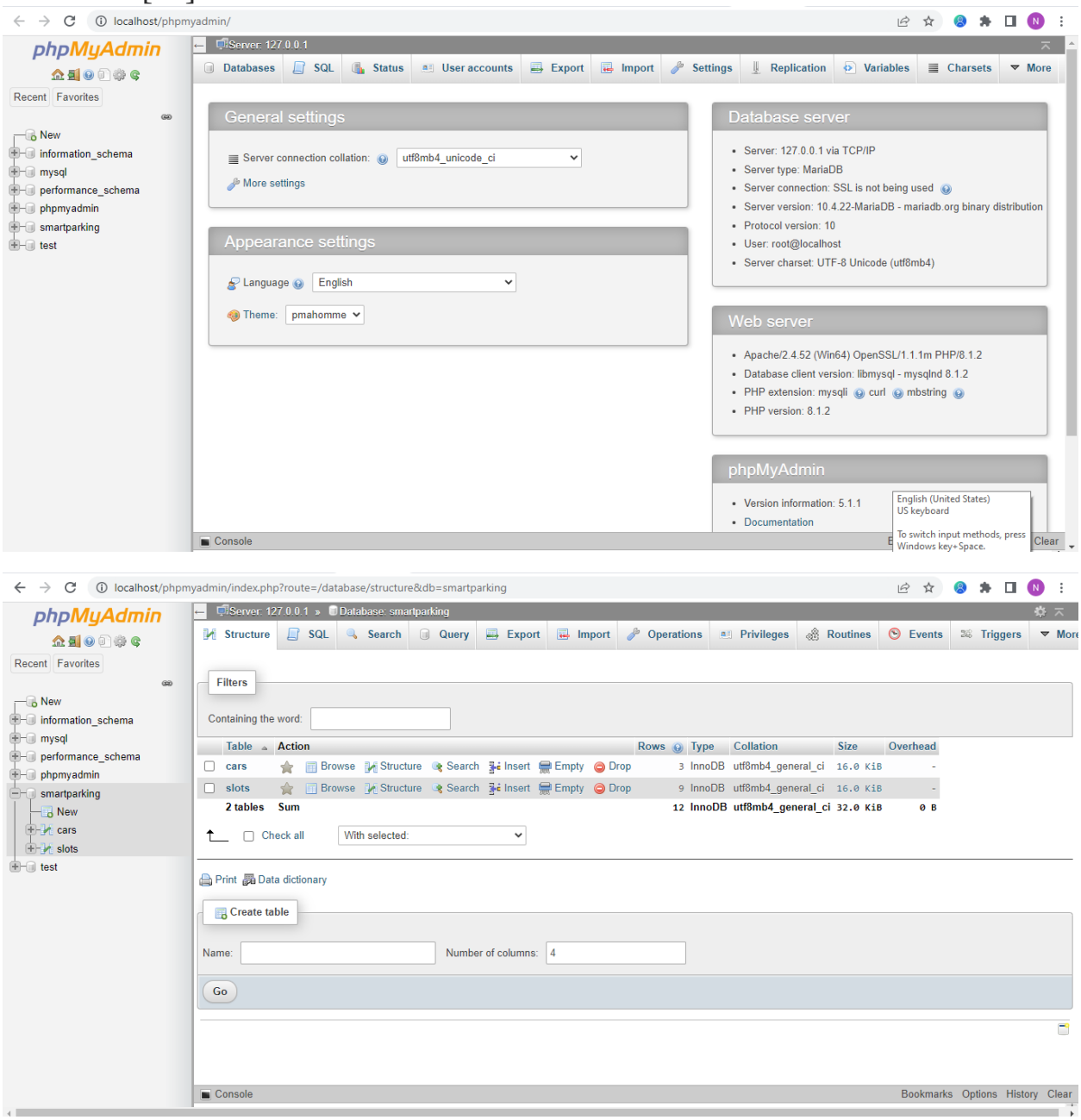


Figure 3.7 Implemented Database

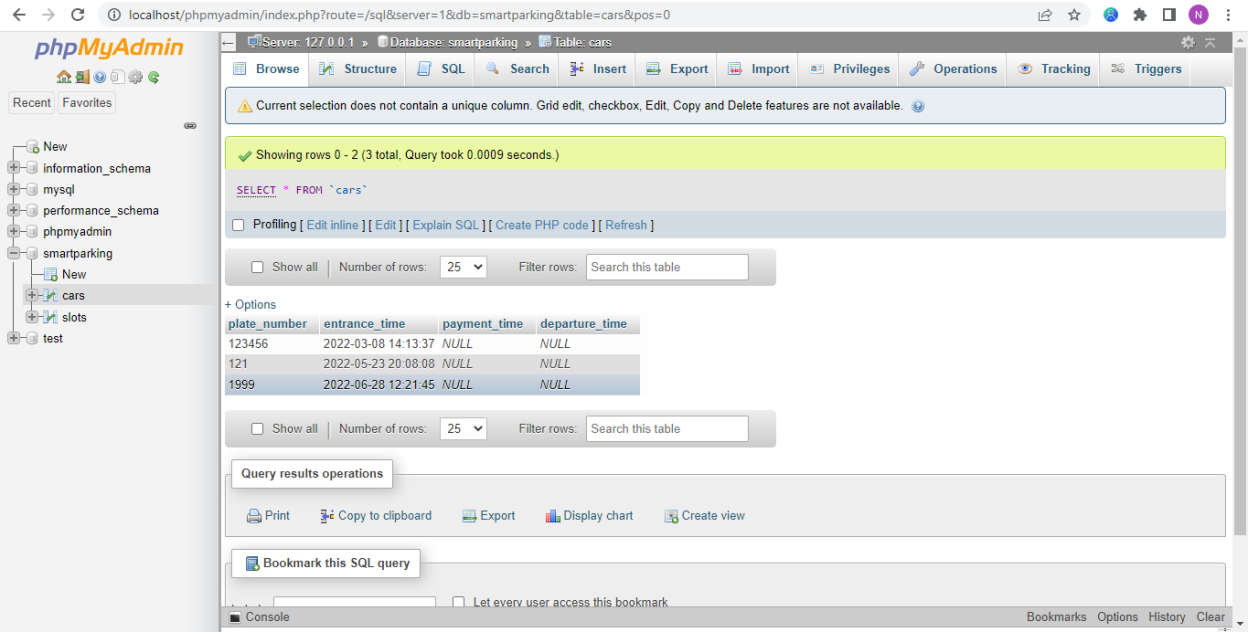


Figure 3.8 Cars Database

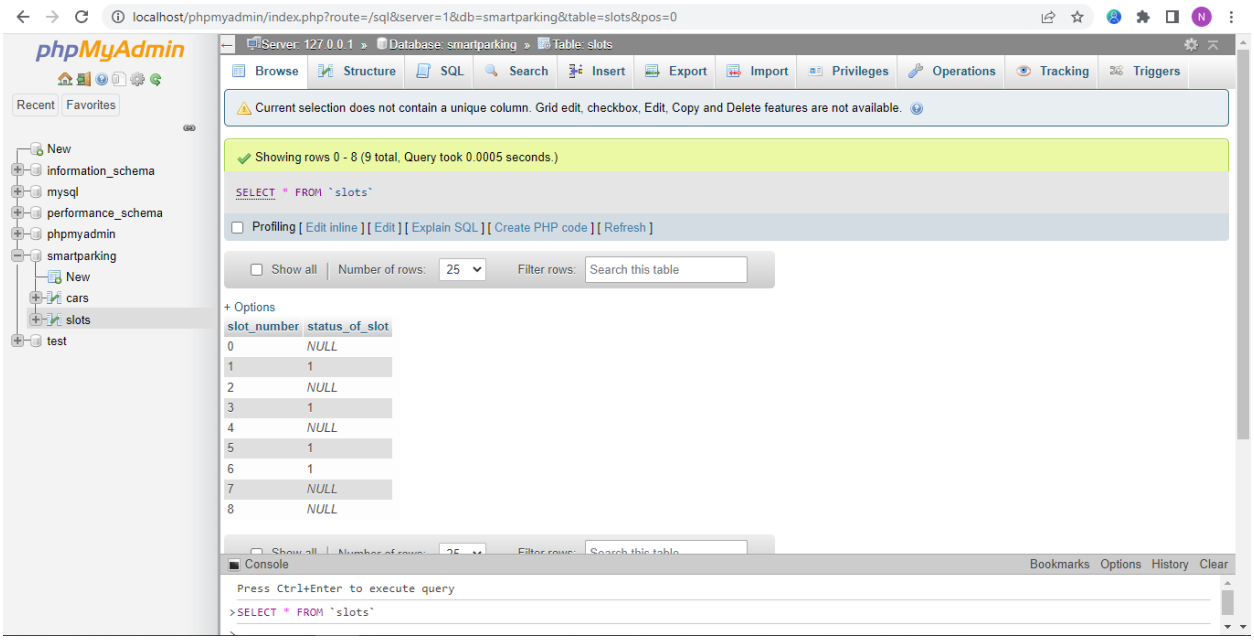


Figure 3.9 Slots Database

3.3.1 Backend

APIs

API stands for Application Programming Interface. In basic terms, APIs are a set of functions and procedures that allow for the creation of applications that access data and features of other applications, services, or operating systems. APIs

for [receiving data from the embedded device and storing it] and [sending it back to the frontend when it sends a request] were created.

The first API: car Entry

carEntry.php

C: > xampp > htdocs > webservier > api > carEntry.php

```php
<?php
    require_once('../database.php');

    DATABASE::connect('localhost', 'smartparking', '123456', 'smartparking');

    DATABASE::query("Insert into cars (plate_number) values (\"" . $_GET['car_plate_number'] . "\")");
?>

```

***We have three ways to add to the file and they are:**

1-include

Syntax: include 'filename';

2-require

Syntax: require 'filename';

3-require_once

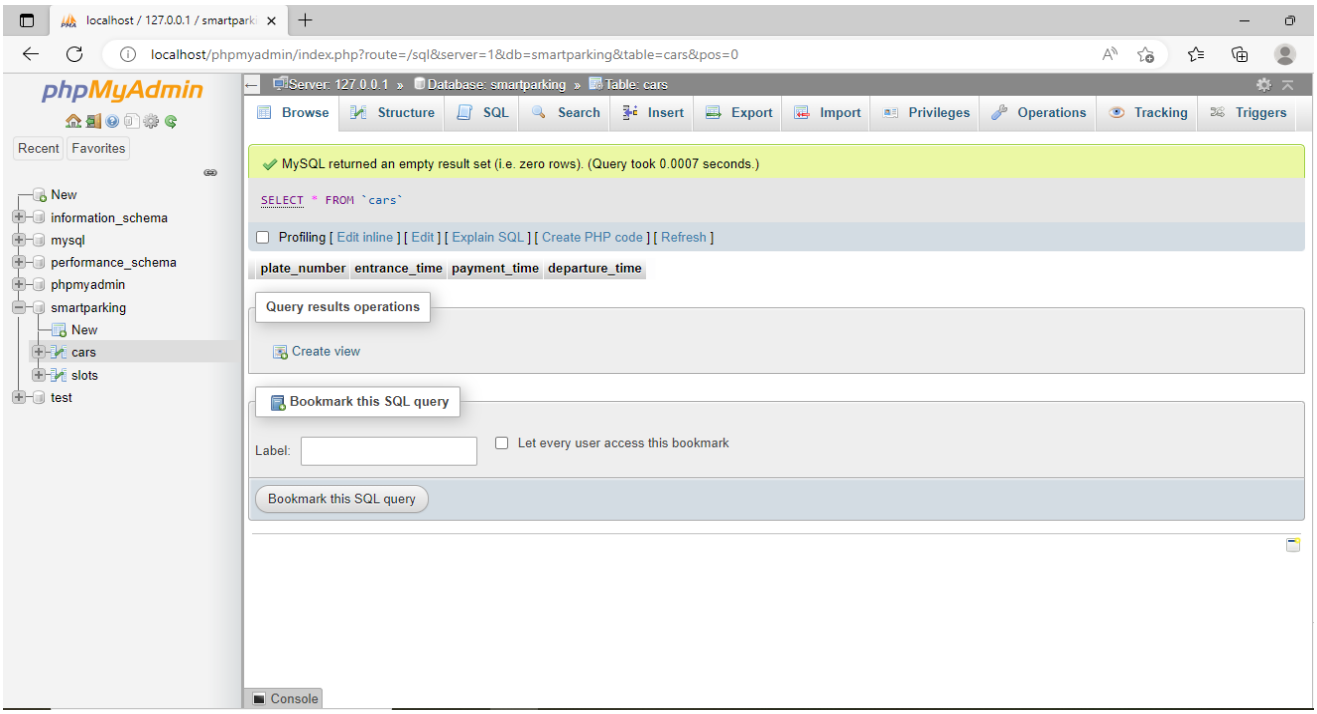
Syntax: require_once 'filename';

And we used the request once because if the file is not found, a fatal error is thrown and the programme stops, and if the file was already included previously, this statement will not include it again.

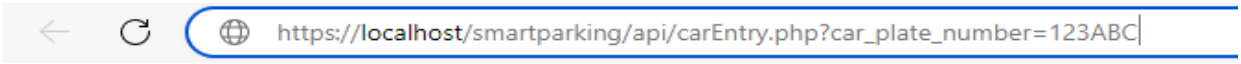
*We can connect to a database by using the host, username, password and database name.

Also, we built the API and determined its "method" (GET) and its "route" or "path (http://localhost/smartparking/api/carEntry.php?car_plate_number={PlateNumber}). (The plate number variable is captured by the camera).When the frontend sends a http request to the previous URL (path), its method is (GET) and this[happen when the car be entered to the parking] to read car plate number ,we send back “data” as a response to the frontend in order to use it.

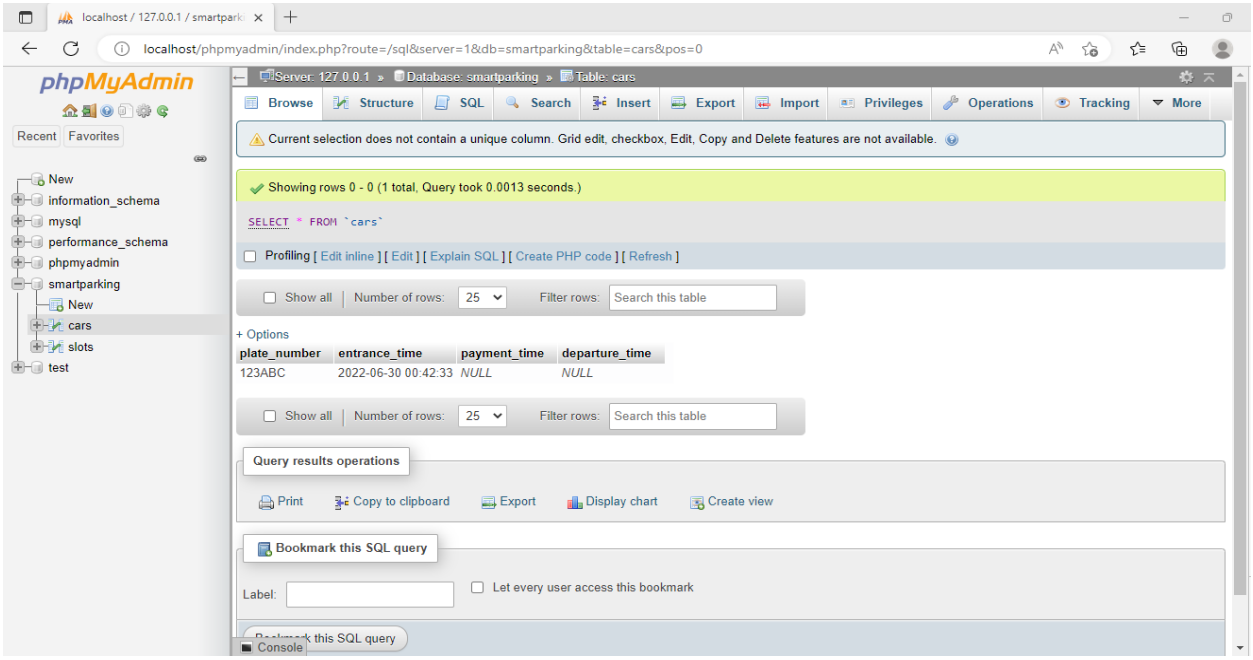
before receiving the data:



When put the url in the browser and press enter, the data will be transferred to the database.



After receiving the data:



The second API: car Exit

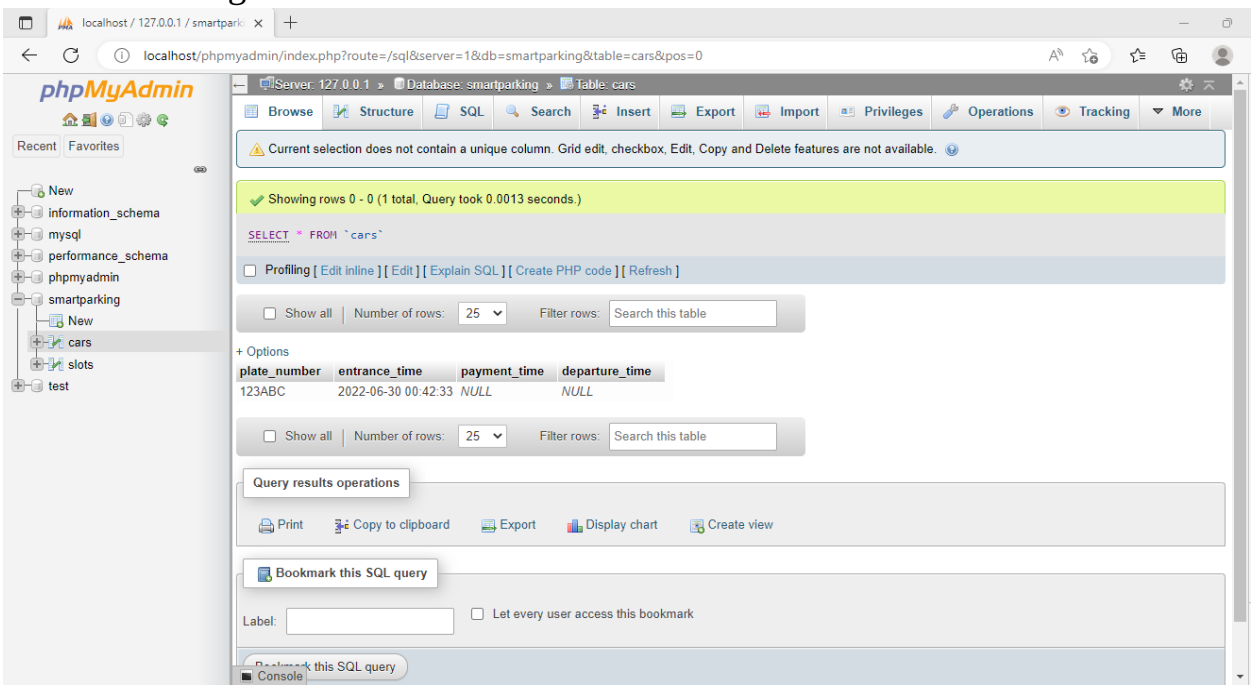
We can connect to a database by using the host, username, password and database name.

Also, we built the API and determined its "method" (GET) and its "route" or "path" (http://localhost/smartparking/api/carExit.php?car_plate_number={plateNumber}), the plateNumber variable that was captured by the camera.

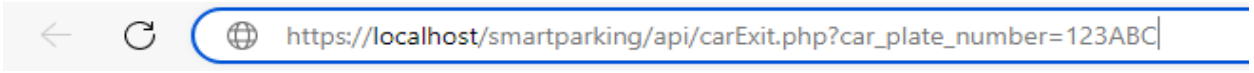
When the frontend sends a http request to the previous URL (path), its method is (GET), and this [happen when the owner's car needs to exit] to read the payment_time and update the departure_time with the current time, we send back "data" as a response to the frontend in order to use it.

And "car_state" stores the received data from the device, which is array (1) [0]=> array (1) ["payment_time"]=> NULL}}.

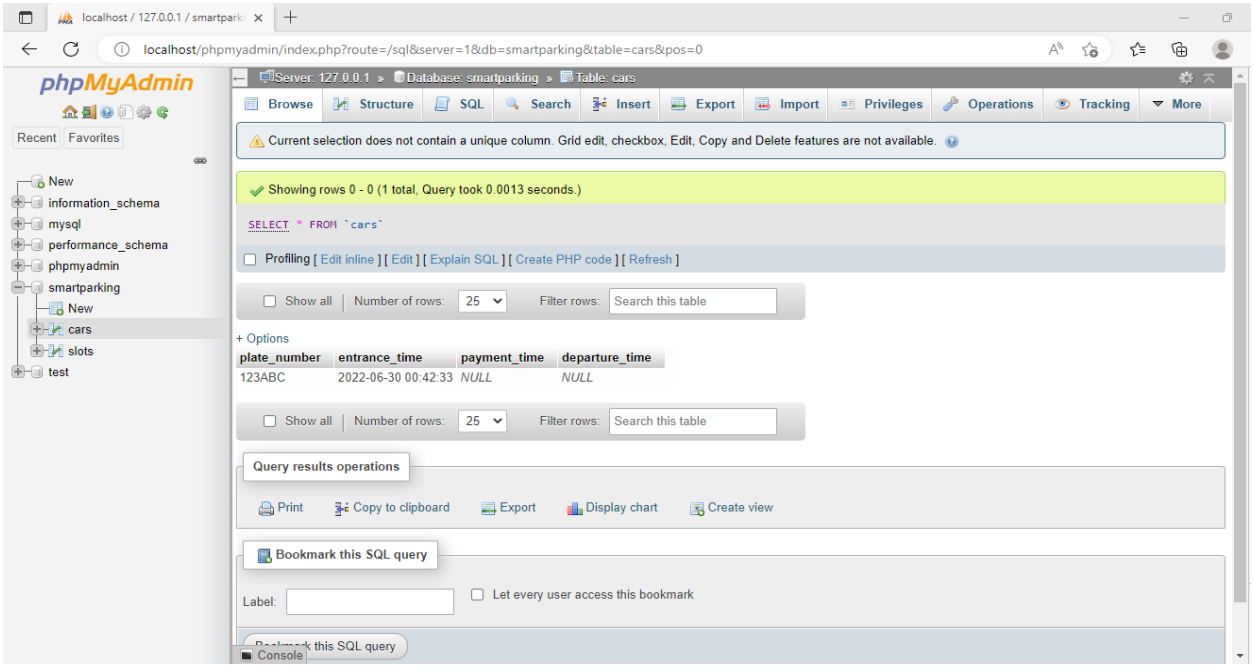
before receiving the data:



When put the url in the browser and press enter, the data will be transferred to the database.



After receiving the data:



The Third API: slot Entry

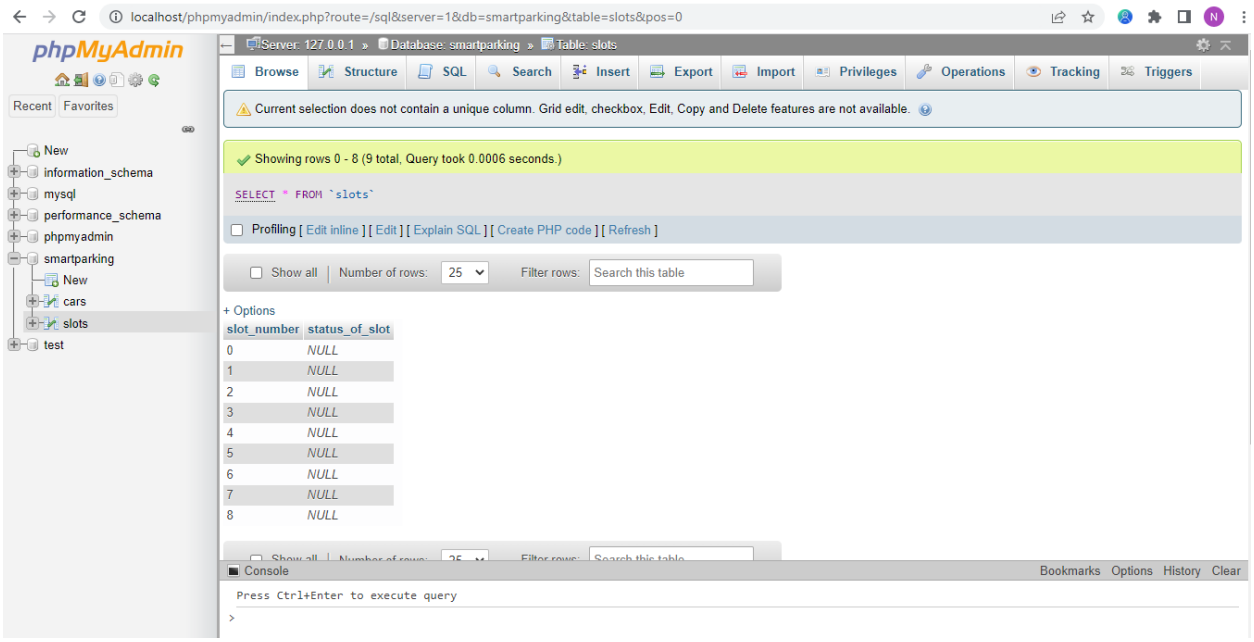
slotEntry.php

C: > xampp > htdocs > SmartParking > api > slotEntry.php

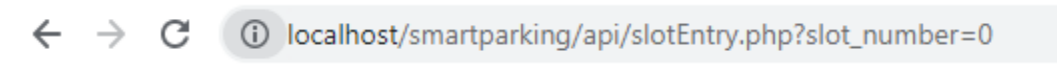
```php
<?php
    require_once('../database.php');

    DATABASE::connect('localhost', 'smartparking', '123456', 'smartparking');

    DATABASE::query("UPDATE slots set status_of_slot=1 WHERE slot_number=\"" . $_GET['slot_number'] . "\"");

```

Also, we built the API and determined its "method" (GET) and its "route" or "path (http://localhost/smartparking/api/slotEntry.php?slot_number={slotNumber}).(the variable slot number obtained by ultrasonic)
When the frontend sends a http request to the previous URL (path), its method is (GET), and this [happens when the owner's car needs to enter] to read the car plate number and slot number, we send back "data" as a response to the frontend in order to use it.

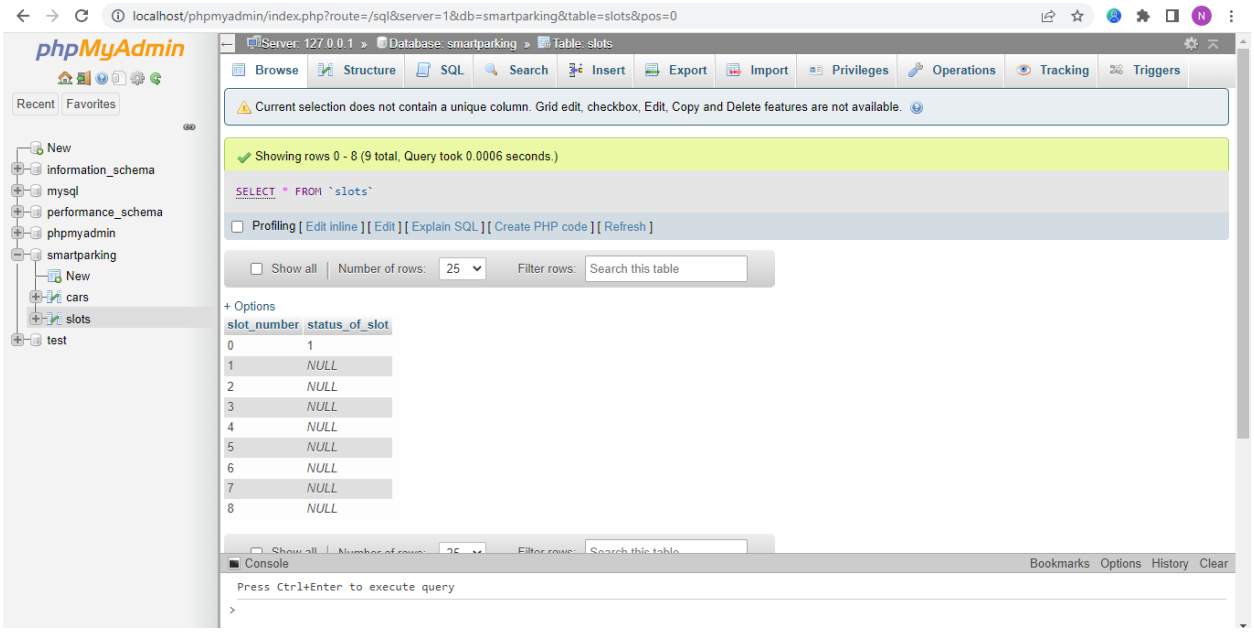
before receiving the data:



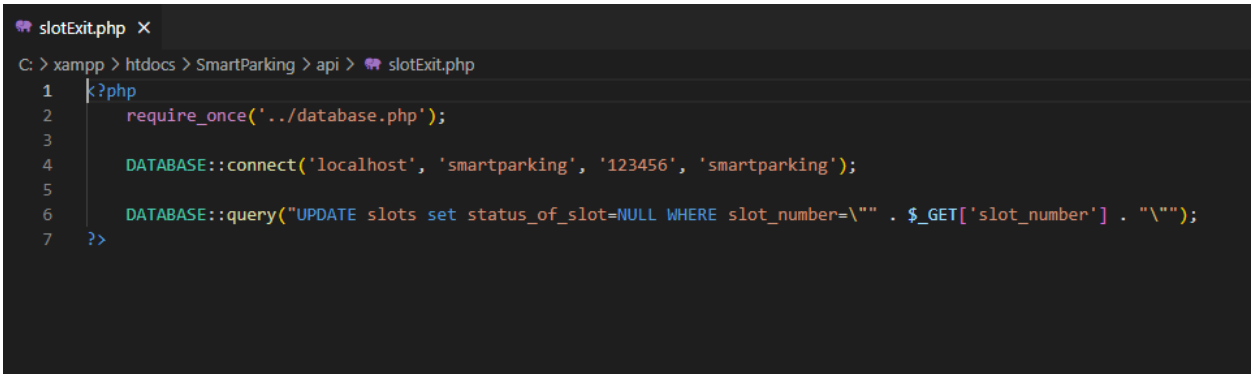
When put the url in the browser and press enter, the data will be transferred to the database.



After receiving the data:



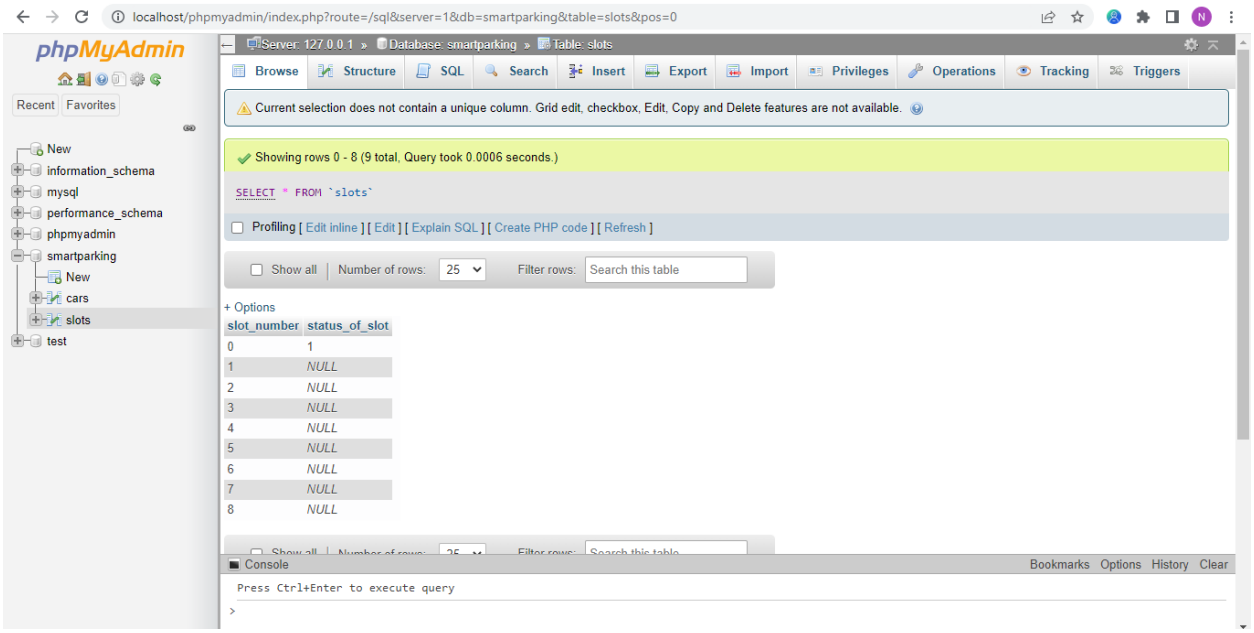
The fourth API: slot Exit



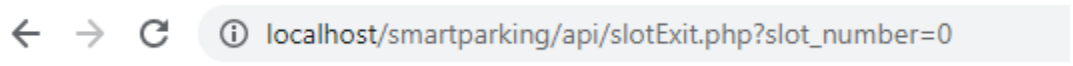
Also we built the API and determined its "method" (GET) and its "route" or "path"(http://localhost/smartparking/api/slotExit.php?slot_number={slotNumber}), the slot number variable which get by ultrasonic.

When the frontend sends a http request to the previous URL (path), its method is (GET), and this [happens when the owner's car needs to exit] to get the slot free, we send back "data" as a response to the frontend in order to use it.

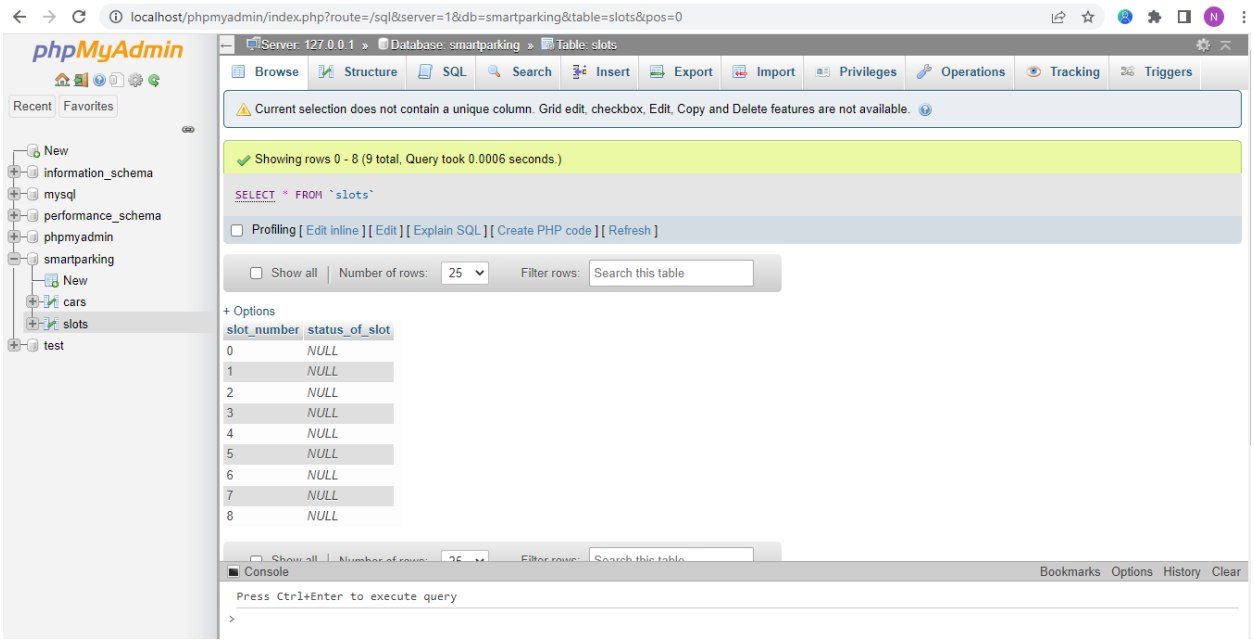
before receiving the data:



When put the url in the browser and press enter, the data will be transferred to the database.



After receiving the data:



3.3.2 Backend and Frontend in Entry page:

*The main purpose of the backend side is to connect to the database. and receive the data (free or busy locations) from embedded device in the top of the parking and save it and send it back to the frontend so we can use it.

Let us show how we did this.

entry.php

C: > xampp > htdocs > SmartParking > Website > entry.php

```php
<?php
  require_once('database.php');

  DATABASE::connect('localhost', 'smartparking', '123456', 'smartparking');

  $slots = DATABASE::query("select * from slots");
?>
```

entry.php

C: > xampp > htdocs > SmartParking > Website > entry.php

```php
<?php
$numberOfColumns = 3;
$numberOfRows = ceil(sizeof($slots)/$numberOfColumns);

for($r = 0; $r < $numberOfRows; $r++) {
  echo '<div class="row"><div class="hline"></div>';
  for($c = 0; $c < $numberOfColumns; $c++) {
    $slotId = $c + $numberOfColumns*$r;

    if($slotId >= sizeof($slots)) break;

    $color = $slots[$slotId]['status_of_slot'] === NULL ? 'red' : 'green';

    echo "<div class='slot' style='background-color: $color;'>Slot $slotId</div>";

    if($c < $numberOfColumns - 1)
      echo "<div class='vline'></div>";
  }
  echo '<div class = "hline"></div>';
  echo '<div class="street"><div class="dots"></div><div class="dots"></div><div class="dots"></div></div>';
  echo '</div>';
}
?>
```

```php
<?php
```

When PHP parses a file, it looks for opening and closing tags, which are <?php and?> which tell PHP to start and stop interpreting the code between them.

```php
$numberOfColumns = 3;
$numberOfRows = ceil(sizeof($slots)/$numberOfColumns);
```

Define 2 variables in php called numberOfColumns and numberOfRows and give each one a value.

```php
for($r = 0; $r < $numberOfRows; $r++) {
  echo '<div class="row"><div class="hline"></div>';
```

After each row there is a horizontal line.

```php
for($c = 0; $c < $numberOfColumns; $c++) {
  $slotId = $c + $numberOfColumns*$r;
```

Slot numbering starts from slot 0.

```php
if($slotId >= sizeof($slots)) break;
$color = $slots[$slotId]['car_plate_number'] === NULL ? 'red' : 'green';
echo "<div class='slot' style='background-color: $color;'>Slot $slotId</div>";
```

If the slot has a car (Busy), Database will make the color of the slot is green,

Else (Empty), It will make the color of the slot is red.

```php
  if($c < $numberOfColumns - 1)
    echo "<div class='vline'></div>";
}
echo '<div class = "hline"></div>';
```

Between every 2 slots, Put a vertical line.

*The main purposes of the frontend side is to keep information organized

Let us show how we did this.

The <style> tag is used to define style information (CSS) for a document

Inside the <style> element you specify hoe HTML elements should render in browser.

```css
<style>
  * {
    background-color: #222;
    /* border: 2px solid; */
  }
```

Container: is a boxy item that has a unique ID, starting with #, and contains the basics that make up the shape of the front end.

```css
#container {
    border: 2px solid;
    width: 550px;
    height: 500px;
    /* top: 50%;
    left: 50%; */
}
```

The border property is a shorthand property for: border-width, border-style, and border-color. If border-color is omitted, the color applied will be the color of the text.

Px Unit: Pixels (px) are relative to the viewing device (1px = 1/96th of 1in)

```css
border: 2px solid;
```

The width property defines the width of the content area of an element.

```css
width: 550px;
```

The height property sets the height of an element.

The height of an element does not include padding, borders, or margins!

```css
height: 500px;
```

On the inside of the container there are 3 rows. Each row contains 2 horizontal lines, 3 slots, dotted lines, and 2 vertical lines between each pair of slots.

Classes:

Slot class:

The location that the car can park.

```css
.slot {
    display: inline-block;
    background-color: white;
    width: 6em;
    height: 2em;
    text-align: center;
    padding-top: 1em;
    margin: 2em;
    box-shadow: black 10px 10px 10px;
}
```

The display: inline-block allows to set the width and height of the element. The top and bottom margins/paddings are respected. Also, display list items horizontally instead of vertically.

```
display: inline-block;
```

The background-color property sets the background color of an element. The background of an element is the total size of the element, including padding and border (but not the margin), Use a background color that makes the text easy to read.

```
background-color: white;
```

The width property defines the width of the content area of an element.

```
width: 6em;
```

The height property defines the height of the content area of an element.

```
height: 2em;
```

The text-align property specifies the horizontal alignment of text in an element.

```
text-align: center;
```

The padding property sets the padding area on all four sides of an element at once.

```
padding-top: 1em;
```

The margin properties are used to create space around elements outside of any defined borders. There are properties for setting the margin for each side of an element (top, right, bottom, and left).

```
margin: 2em;
```

The box-shadow property is used to apply one or more shadows to an element.

```
box-shadow: black 10px 10px 10px;
```

Finally the slots is designed.

Vline class:

A class, "vline" is to separate between two slots, which is a place the car can park.

```
.vline {
  display: inline-block;
  background-color: white;
  height: 3em;
  width: 3px;
  transform: translate(0, 40%);
}
```

display: inline-block is to display list items horizontally instead of vertically.

```
display: inline-block;
```

The background-color property sets the background color of an element. The background of an element is the total size of the element, including padding and border (but not the margin), Use a background color that makes the text easy to read.

```
background-color: white;
```

The height property sets the height of an element.

The height of an element does not include padding, borders, or margins!

em Unit: is Relative to the font-size of the element (3em means 3 times the size of the current font).

```
height: 3em;
```

The width of a block-level element will prevent it from stretching out to the edges of its container

Px Unit: Pixels (px) are relative to the viewing device (1px = 1/96th of 1in)

```
width: 3px;
```

The translate() method moves an element from its current position (according to the parameters given for the X-axis and the Y-axis).

```
transform: translate(0, 40%);
```

Finally the slots were separated from each other by a vertical line

Classes, "hline" and "dots" to make a street between each row and the next to facilitate the movement of cars inside the parking.

Hline class:

```
.hline {
    background-color: white;
    margin: auto;
    width: 90%;
    height: 1%;
}
```

The background-color property sets the background color of an element. The background of an element is the total size of the element, including padding and border (but not the margin).

```
background-color: white;
```

The margin properties are used to create space around elements outside of any defined borders. There are properties for setting the margin for each side of an element (top, right, bottom, and left).

```
margin: auto;
```

The width property defines the width of the content area of an element.

```
width: 90%;
```

The height property defines the height of the content area of an element.

```
height: 1%;
```

Dots class:

```
.dots {
    display: inline-block;
    border-top: 2px dashed white;
    width: 5em;
    margin-top: 3em;
    margin-bottom: 3em;
    margin-left: 12.5%;
}
```

The display: inline-block allows to set a width and height of the element. The top and bottom margins/paddings are respected. Also, display list items horizontally instead of vertically.

```
display: inline-block;
```

The border-top shorthand property sets all the top border properties in one declaration and give the white color.

```
border-top: 2px dashed white;
```

The width property defines the width of the content area of an element.

```
width: 5em;
```

The margin properties are used to create space around elements outside of any defined borders.

There are properties for setting the margin for each side of an element (top, right, bottom, and left).

```
margin-top: 3em;
margin-bottom: 3em;
margin-left: 12.5%;
```

Finally by using these two classes the street was made like this

3.3.3 Backend and Frontend in Exit page:

*The main purpose of the backend side is to connect to database and receive the data (The number of parking hours and the amount of money to be paid) from the saving time get from an embedded device putting in the entering of parking.

Let us show how we did this..

exit.php

C: > xampp > htdocs > SmartParking > Website > exit.php

```php
<?php
  require_once('database.php');

  DATABASE::connect('localhost', 'smartparking', '123456', 'smartparking');

  $cost_per_hour = 50;
  $car_data = [];
  $car_id = '';

  if(isset($_GET['carId'])) {
    #var_dump($_GET);
    $car_id = $_GET['carId'];
    #var_dump($car_id);
    $car_data = DATABASE::query("select HOUR(timediff(NOW(), entrance_time)) as number_of_hours from cars where plate_number=
    \"{$_GET['carId']}\" AND departure_time is NULL LIMIT 1");
    #var_dump($car_data);
  }
?>
```

exit.php

C: > xampp > htdocs > SmartParking > Website > exit.php

```php
<?php
  if(isset($_GET['carId'])) {
    #var_dump($_GET);
    if(sizeof($car_data)) {
      DATABASE::query(
        "UPDATE cars set payment_time = current_timestamp() WHERE plate_number = \""
        . $_GET['carId']
        . "\""
        );
      #var_dump($_GET);
      $number_of_hours = $car_data[0]['number_of_hours'];
      $total_cost = $number_of_hours * $cost_per_hour;
      echo "<p> Number of hours : {$number_of_hours} hours</p>";
      echo "<p>Amount to be paid: {$total_cost} </p>";
    }
    else {
    echo '<p class = "pp">CAR NOT FOUND</p>';
    }
  }
?>
```

*The main purpose of the front end is to keep information organized. Let us show how we did this..

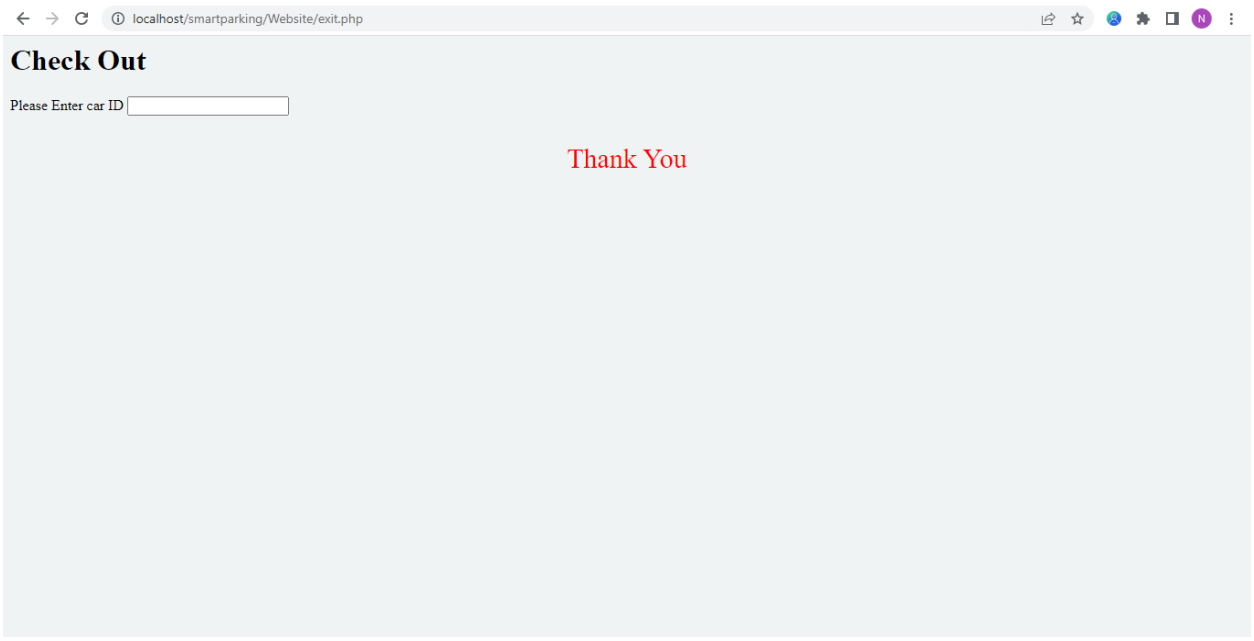
exit.php

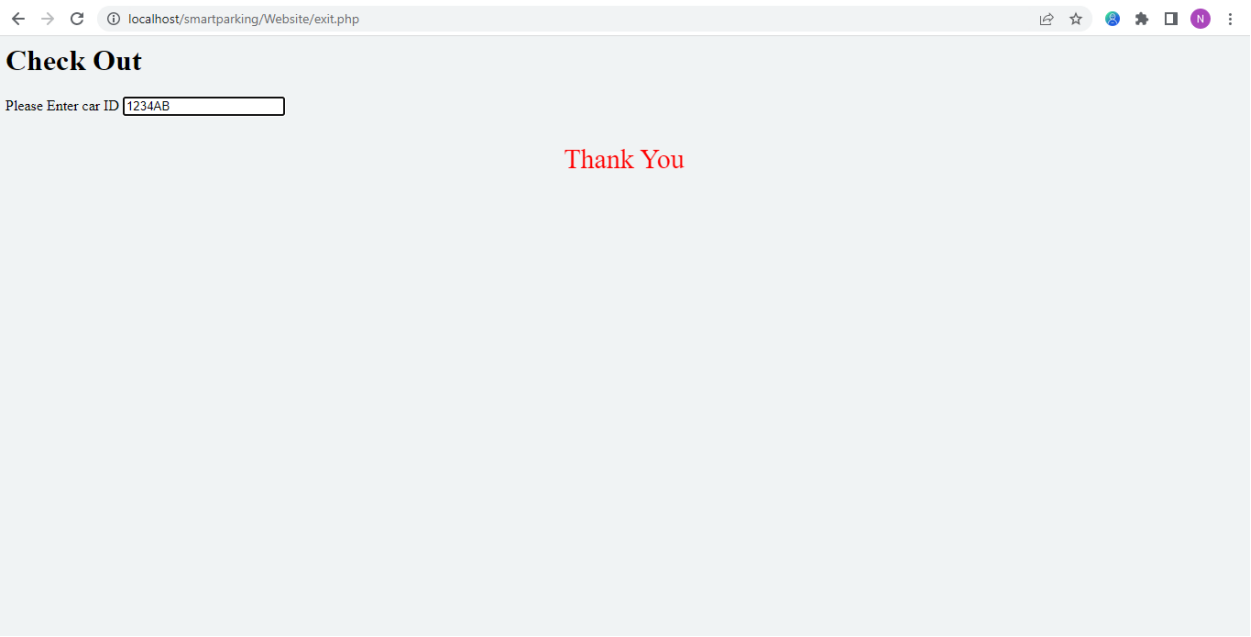
C: > xampp > htdocs > SmartParking > Website > exit.php

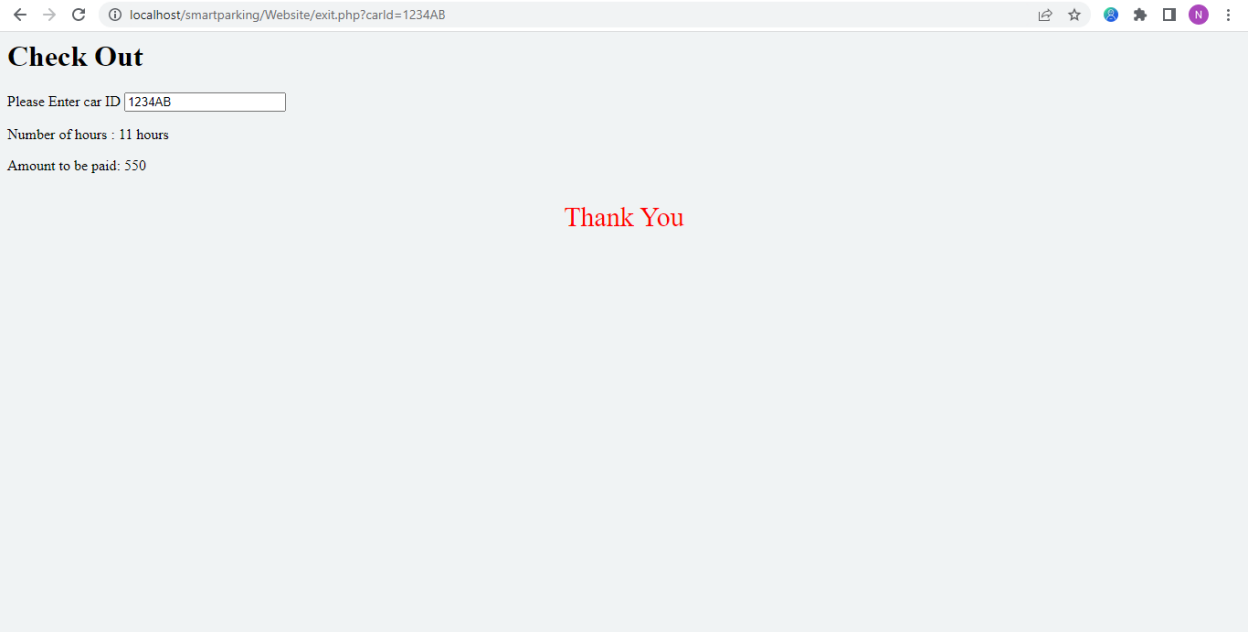
```
19  <style>
20    body {
21      background-color: #F0F3F4;
22    }
23    .pp {
24      text-align: center;
25      color: red;
26      font-size: 30px;
27    }
28  </style>
29  <h1> Check Out </h1>
30  <form>
31    <label> Please Enter car ID </label>
32    <input type="Text" name="carId" value="<?php echo $car_id; ?>">
33    <submit>
34  </form>
```

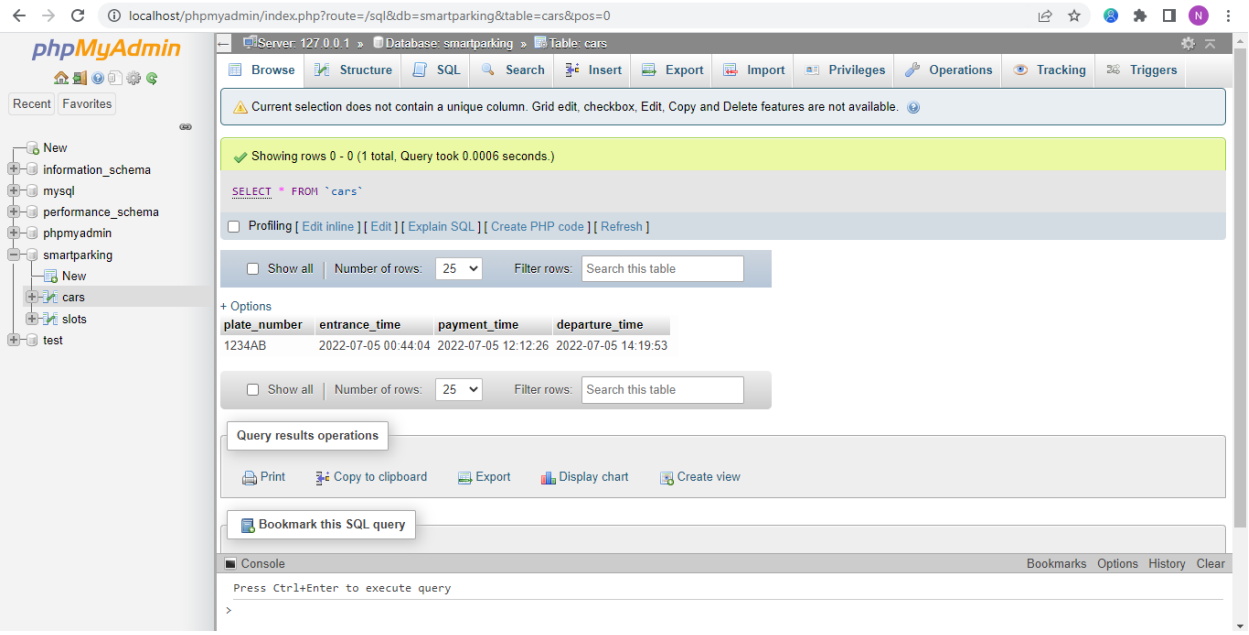
```
56  <p class = "pp"> Thank You </p>
```

At the end of all this explanation, we get

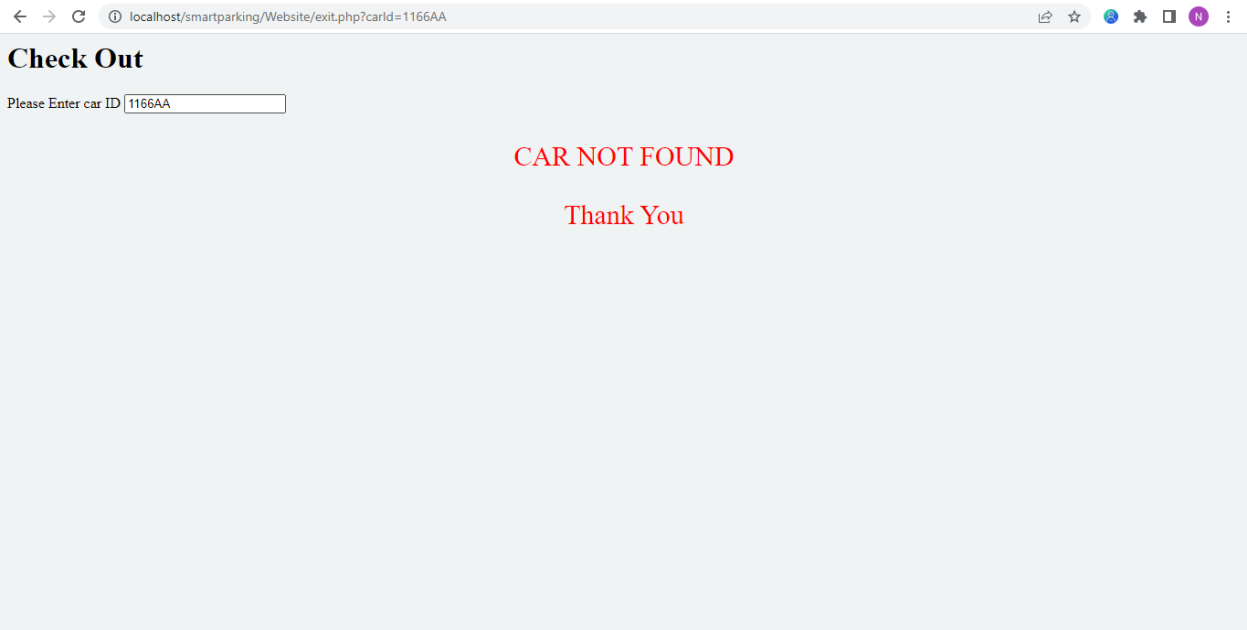



localhost/smartparking/Website/exit.php
Check Out
Please Enter car ID 1234AB
Thank You


localhost/smartparking/Website/exit.php?carId=1234AB
Check Out
Please Enter car ID 1234AB
Number of hours : 11 hours
Amount to be paid: 550
Thank You



If the user enters a wrong car ID

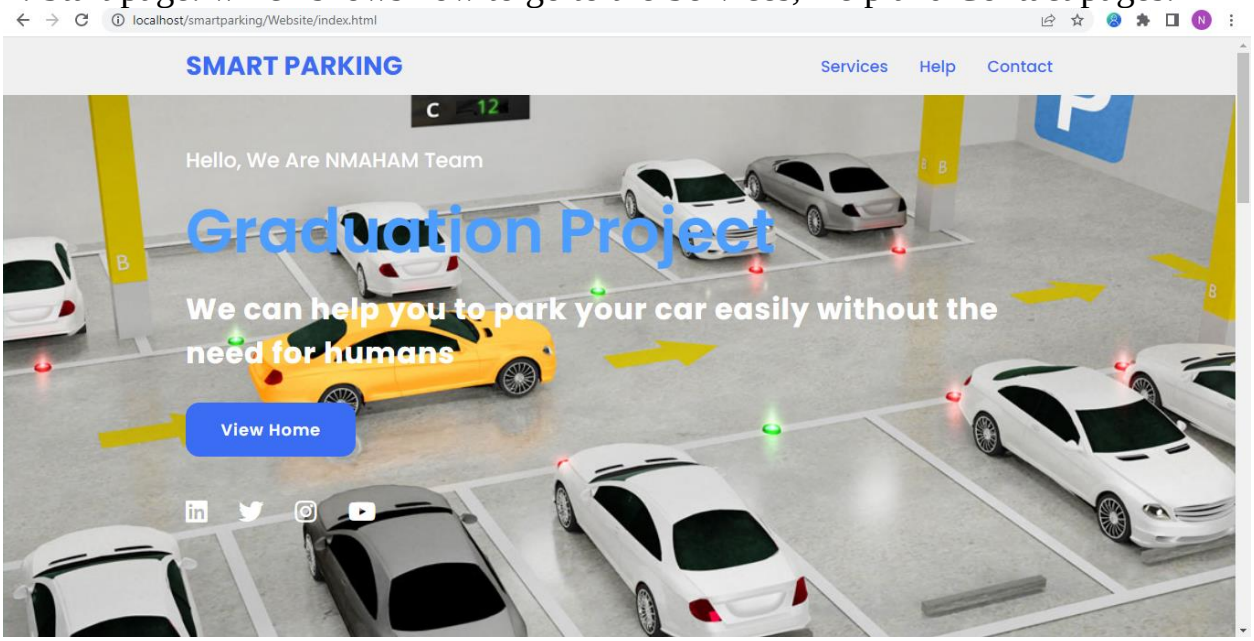


3.3.4 Frontend

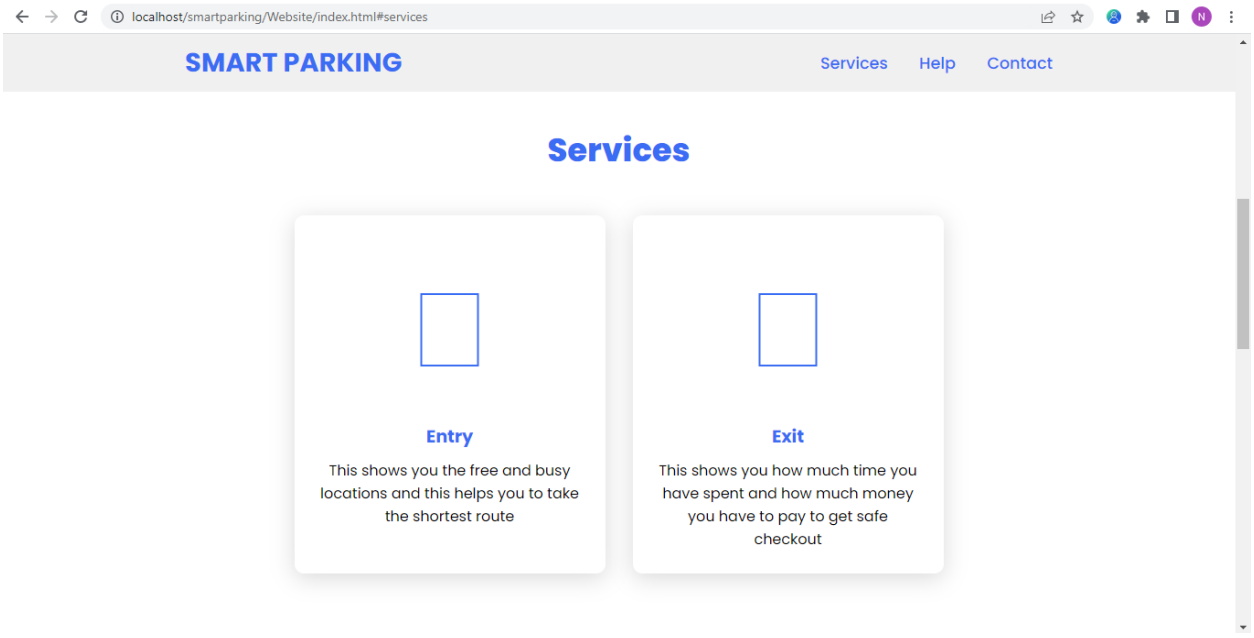
Frontend Design

There are 4 pages in the website :

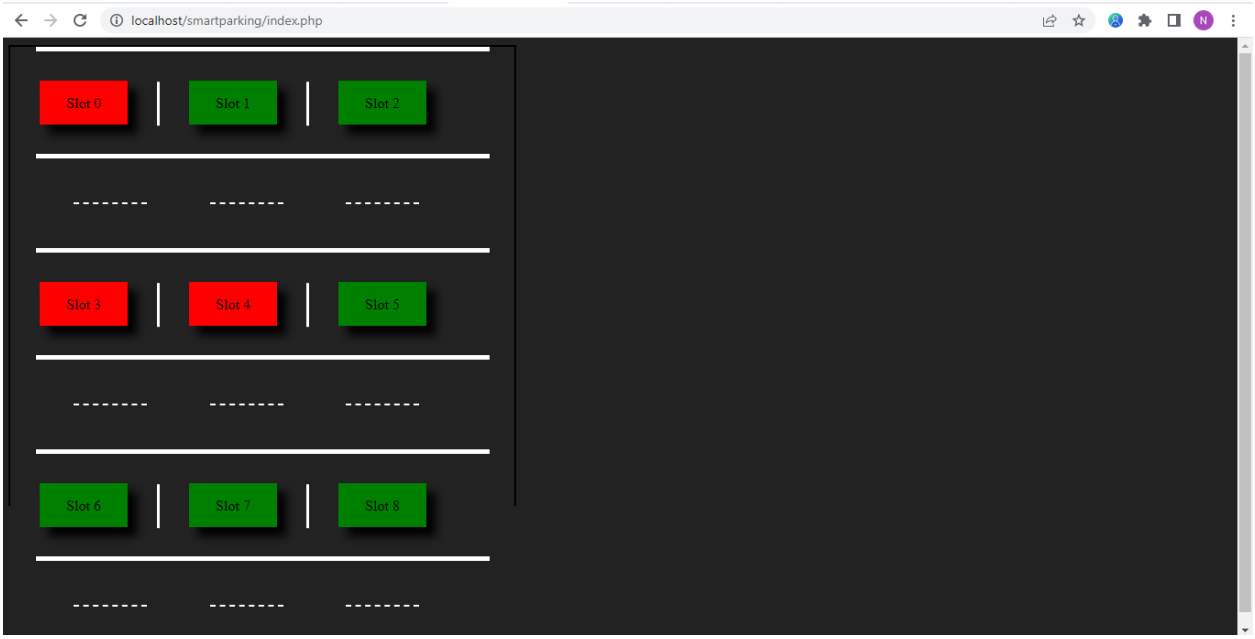
1. Start page: which shows how to go to the Services, Help and Contact pages.



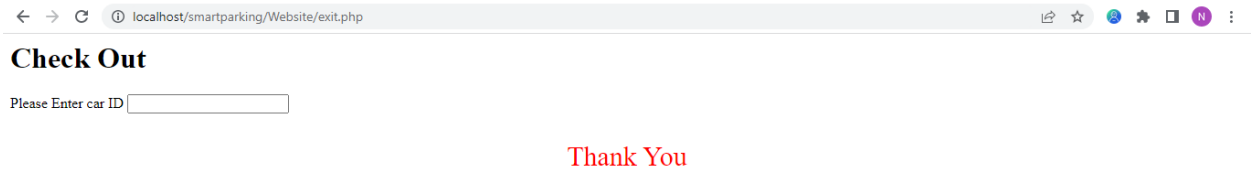
2.Services page:which shows the free and busy locations when the owner's car clicks the Entry button. and shows the time taken and the money to be paid when the owner's car clicks the Exit button.



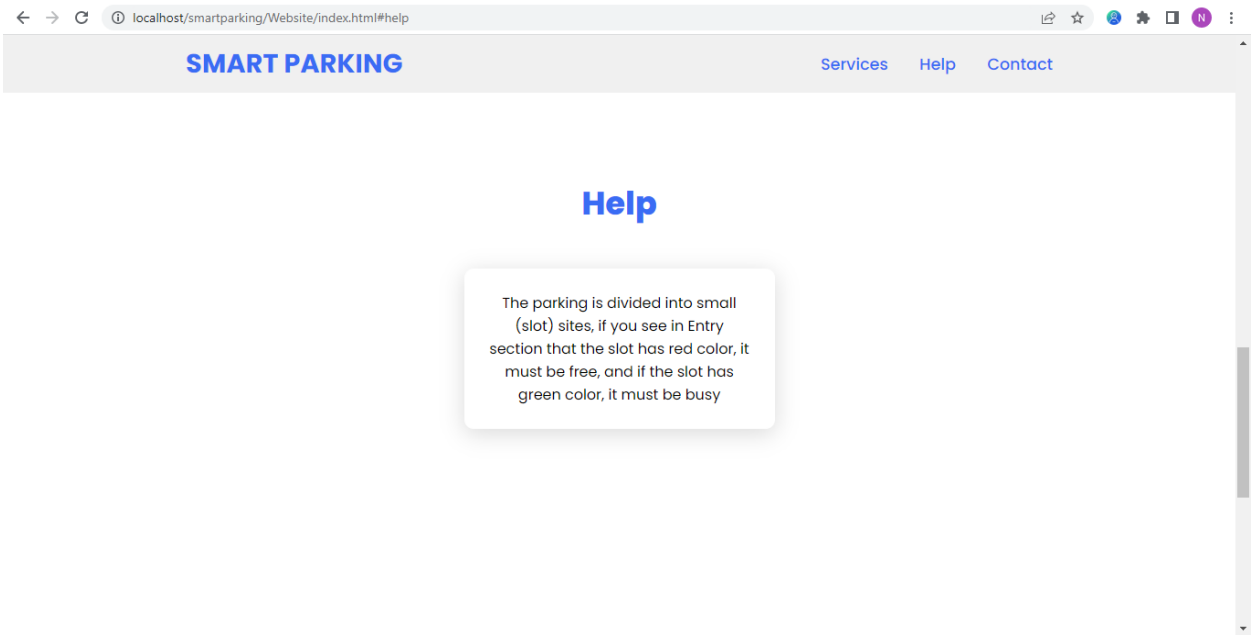
when owner’s car clicks the Entry button



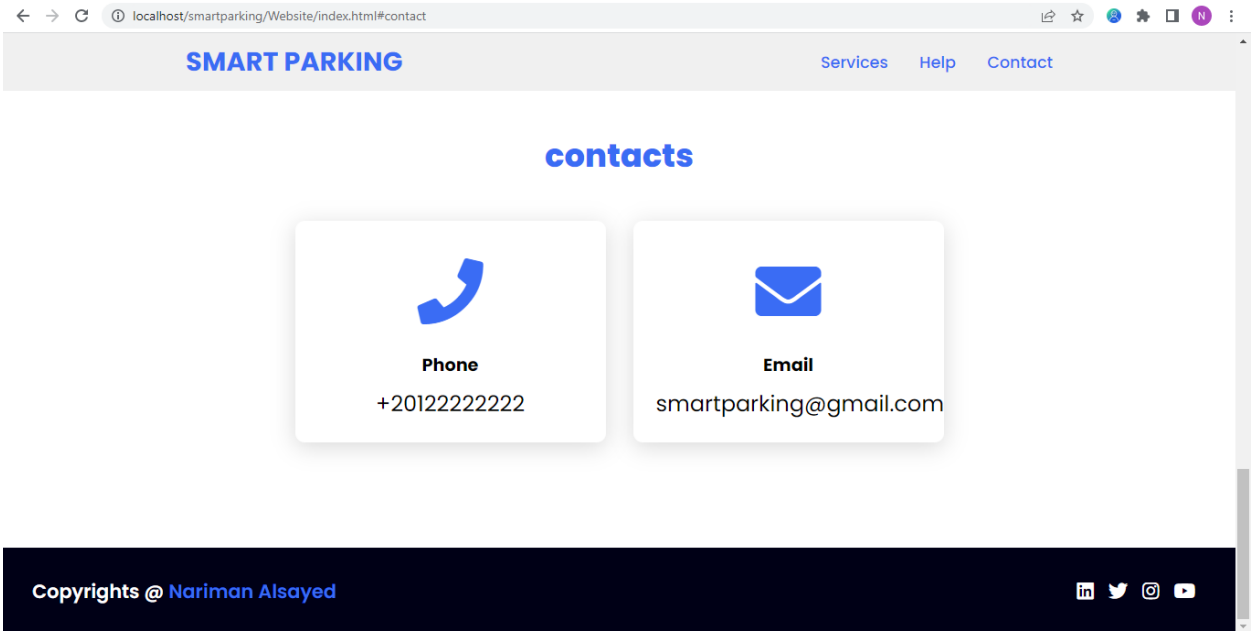
when owner’s car clicks the Exit button



3.Help page:



4.Contact page:



Chapter 4

Hardware

4.1 Introduction

This working relates to a general smart parking system and in particular an intelligent and electrical parking system based on working many objects in that park Without any human employee in it.

4.2 Project Components

4.2.1 Raspberry pi:

4.2.1.1 Introduction:

4.2.1.1.1 What is a Raspberry Pi used for?

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.

Getting started, hardware of raspberry pi:

First, we need to setup the Raspberry Pi either:

1- Monitor, HDMI cable, mouse, and keyboard, and connect them to the Raspberry Pi or

2-You can connect it to a regular PC and access it using remote desktop and using jack Ethernet to connect it or connect it wirelessly.

And in both you will need a good power supply to the Raspberry pi.

Figure 4.1: Raspberry Pi

4.2.1.1.2 Operation:

We first install the operating system on a 32 gigabyte SD card.

Steps:

Step 1: Download and Install Raspberry Pi Imager

Step 2: Run Raspberry Pi Imager and Format the SD Card

Step 3: Burn the Raspberry Pi OS Image to the SD Card

4.2.1.1.3 What have we used it for?

We used it to connect the cameras and the exit gate to it and to run the python code.

4.2.1.2 Software:

Setting up raspberry pi software:

1-update packages repo

Implementation: sudo apt update

2-upgrade system packages

Implementation: sudo apt upgrade

3-install python, xrdp and tesseract

Implementation: sudo apt install -y python3 xrdp tesseract-ocr

4-install libraries needed by opencv

Implementation: sudo apt install -y libatlas-base-dev

5-upgrade pip

Implementation: python3 -m pip install -U pip

6-install needed python packages

Implementation: python3 -m pip install opencv-python==4.2.0.34 numpy pytesseract

7- use rdp to copy files over to the rpi

Implementation: download v4 traineddata in source code tessdata folder

wget https://github.com/tesseract-ocr/tessdata/raw/4.00/eng.traineddata

8- make tesseract use local trained data dir

Implementation: pytesseract. image_to_string (plateImg, 'eng', config='--tessdata-dir ./tessdata')

9- create service script

nano myscript. service

Implementation: install service

sudo cp myscript. service /etc./system/system/myscript.service

10- test the service

Implementation: sudo systemctl start myscript. service

11- set the service to run at boot

Implementation: sudo systemctl enable myscript. Service

4.2.2 Cameras:

Cameras and **vision systems** for parking

4.2.2 .1 Introduction:

A **technologically advanced** smart parking system must be able to integrate the **mobility needs of the future.**

In addition to sensors**, Smart Parking Systems** are also using **vision systems** to offer their interlocutors a structured solution based on the needs of the territory.

Over the last few years, the **cameras** have undergone substantial technological acceleration, developing specific algorithms that allow to **identify the phenomena involved with precision and timeliness.**

4.2.2 .2 Why cameras?

1.Reduced hardware investments.

2.existing cameras can be used to reduce the economic impact of the implementation.

3.simple integration with existing software and platforms through APIs.

The cameras have become today very reliable and low-power tools that can respond, better than other technologies, to certain needs.

For example, think of an open space, like a square, where there are no visual obstacles and where you can then analyze the status of several parking spaces with a single camera.

If the conditions for implementation were optimal, it could cover up to 300 cars.

4.2.2.3 Operation:

The images are acquired and immediately converted into data (free-busy), then into algorithms, without being recorded in our parking management software.

After importing the data, the software automatically destroys the images so that there is no trace of sensitive content.

To further enhance the security of the users, the process of recording images takes place at low resolution. In this way, sensitive elements such as licence plates or people’s faces can’t be recognized.

4.2.2.4 Camera (Web camera):

Figure 4.2: Camera

A web camera is used to get the images of the parking area and image processing techniques are used to detect the presence or absence of cars to count and locate the available parking spaces which achieved the average performance is 99.5 % and is very high as compared with other parking lot detections applications. The result shows that when the captured images of the parking lot are not clear because of less lighting or occlusions, the efficiency decreases and the accuracy for detection's reduces.

4.2.2.5 Implementation:

1. Get the raspberry Pi ready.

2. Assemble the raspberry Pi and camera, Connect the camera module to the PI's camera connector using the USB provided with the camera.

3. Configure the camera software

4. Automating the camera

Figure 4.3: Connection raspberry pi with camera

4.2.2.6 Methodology

A car enters the parking lot, and the parking lot is checked for empty slots. If there exists any then the number plate of the car is scanned using the first camera and stored in cloud and let into the parking lot by displaying the slots available. If there are no empty slots available, then the same will be displayed. When the car exits, the number plate is scanned again using the second camera and compared with the time of its arrival. A bill is generated based on the time of the vehicle.

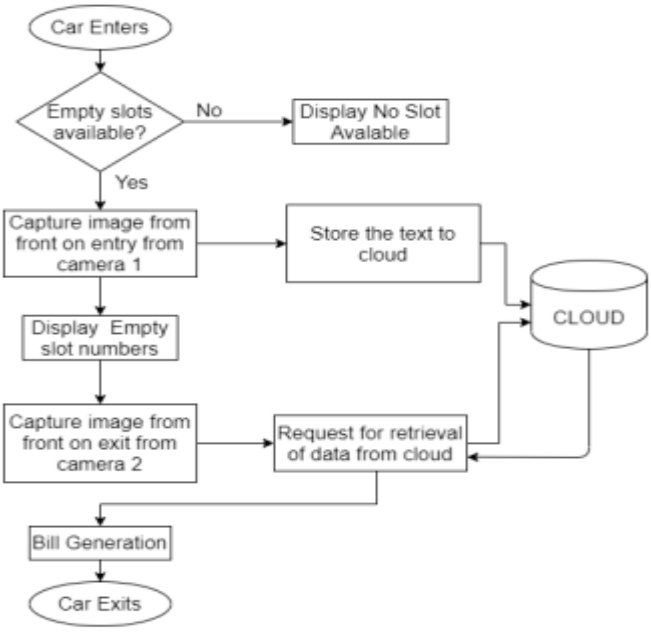


Figure 4.4: Flow chart raspberry pi with camera

4.2.3 ESP32 module

4.2.3.1 Introduction:

ESP32 is a single 2.4 GHz Wi-Fi-and-Bluetooth combo chip designed with the TSMC ultra-low-power 40 nm technology. It is designed to achieve the best power and RF performance, showing robustness, versatility and reliability in a wide variety of applications and power scenarios.

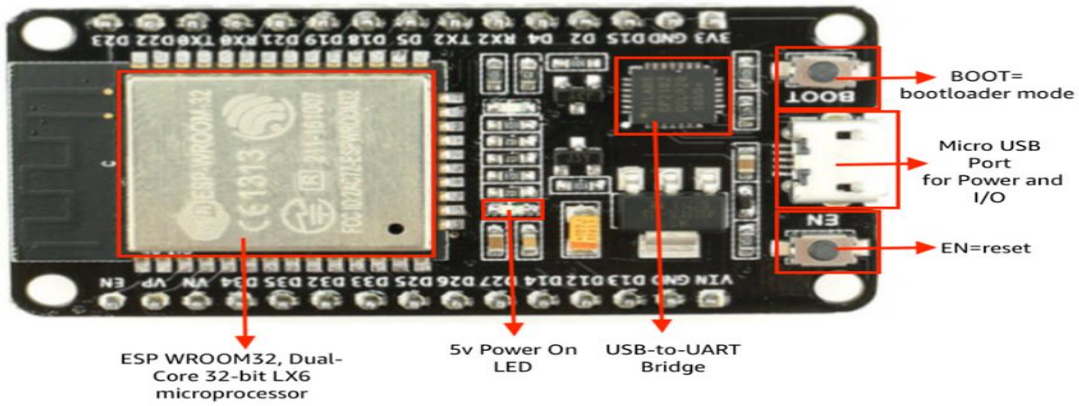


Figure 4.5: ESP32 DEVKIT DOIT V1

4.2.3.2 Features of ESP 32 module:

• Single or Dual-Core 32-bit LX6 Microprocessor with clock frequency up to 240 MHz.

• 520 KB of SRAM, 448 KB of ROM and 16 KB of RTC SRAM.

• Supports 802.11 b/g/n Wi-Fi connectivity with speeds up to 150 Mbps.

• Support for both Classic Bluetooth v4.2 and BLE specifications.

• 34 Programmable GPIOs.

• Up to 18 channels of 12-bit SAR ADC and 2 channels of 8-bit DAC

• Serial Connectivity include 4 x SPI, 2 x I2C, 2 x I2S, 3 x UART.

• Ethernet MAC for physical LAN Communication (requires external PHY).

• 1 Host controller for SD/SDIO/MMC and 1 Slave controller for SDIO/SPI.

• Motor PWM and up to 16-channels of LED PWM.

• Secure Boot and Flash Encryption.

• Cryptographic Hardware Acceleration for AES, Hash (SHA-2), RSA, ECC and RNG.

4.2.3.3 Why we choose the ESP 32 module?

ESP32 is a highly-integrated solution for Wi-Fi-and-Bluetooth IoT applications, with around 20 external components. ESP32 integrates an antenna switch, RF balun, power amplifier, low-noise receive amplifier, filters, and power management modules. As such, the entire solution occupies minimal Printed Circuit Board (PCB) area.

ESP32 uses CMOS for single-chip fully-integrated radio and baseband, while also integrating advanced calibration circuitries that allow the solution to remove external circuit imperfections or adjust to changes in external conditions.

4.2.3.4 Block Digram:

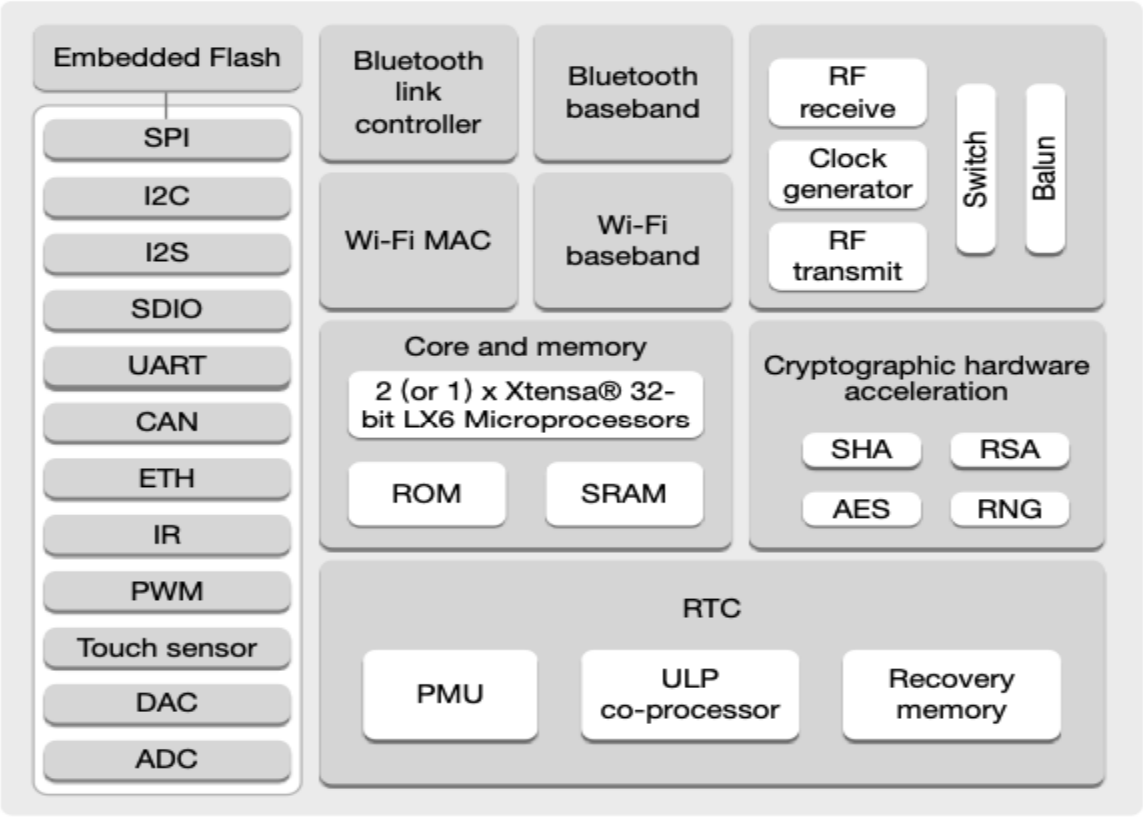


Figure 4.6: Functional Block Diagram ESP32

4.2.1.5 software

The ESP Development Board can be easily programmed with the Arduino IDE since it is easy to use.

Programming ESP32 with the Arduino IDE will hardly take 5-10 minutes. All you need is the Arduino IDE, a USB cable and the ESP board itself.

4.2.3.6 ESP32 Pinout:

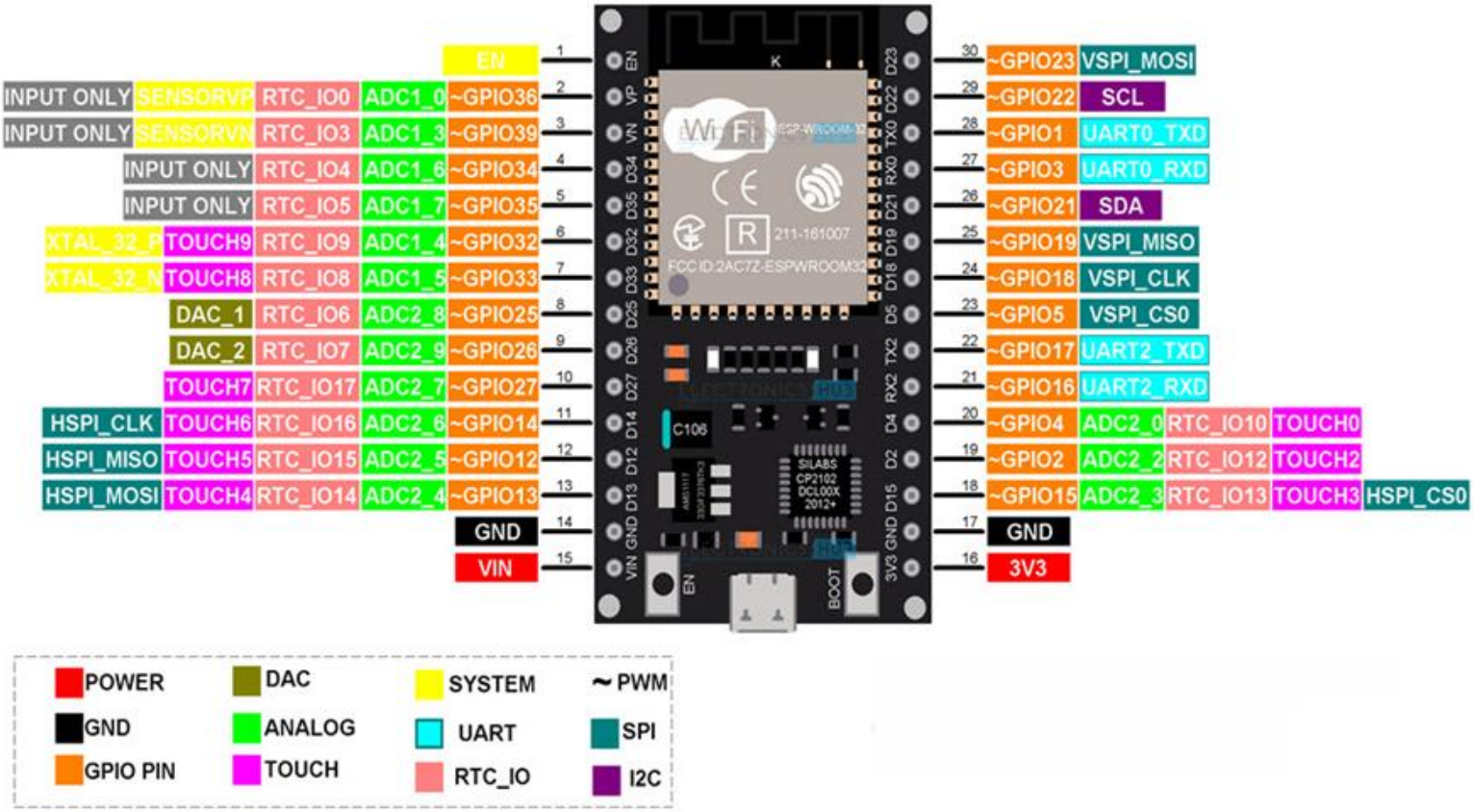


Figure 4.7: ESP32 PINOUT

4.2.3.7 Power Requirement

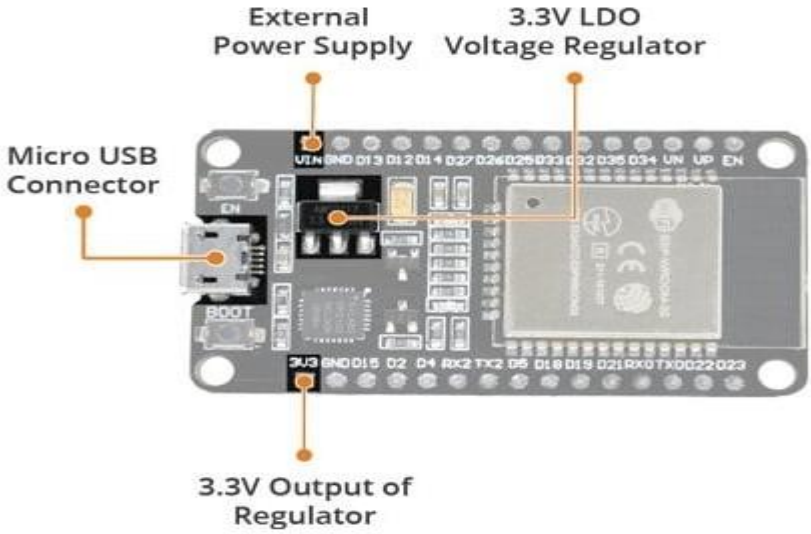


Figure4.8: ESP32 DEVKIT DOIT V1 Power Requirements

The ESP32 Board operates between 2.2V to 3.6V. But we supply 5V from Micro-USB port. For 3.3V there is already an LDO voltage regulator to keep the voltage steady at 3.3V. ESP32 can be powered using Micro USB port and VIN pin (External Supply Pin).

The power required by ESP32 is 600mA, as ESP32 pulls as much as 250mA during RF transmissions. During boot or WIFI operation it's drawing more than 200mA current.

4.2.4 Ultrasonic Sensor

When needing to input from the real world to data collection system, there are several options to choose from:

- **Light detection** (infrared sensors, light-dependent resistors, and computer vision setups)
- . **Sensors based on magnetic fields**
- **Ultrasonic sensors**

4.2.4.1 Principle of operation

Distance measurement sensor is a low-cost full functionality solution for distance measurement applications. An ultrasonic sensor is a type of electronic device that uses ultrasonic sound waves to measure the distance between two objects and converts the reflected sound into an electrical signal. Ultrasonic waves travel faster than the speed of audible sound (i.e. the sound that humans can hear). The transmitter (which generates sound using piezoelectric crystals) and the receiver are the two primary components of ultrasonic sensors (which encounters the sound after it has travelled to and from the target). To compute the distance between the sensor and the item, the sensor measures the time it takes from the transmitter's sound emission to its collision with the receiver. **D = ½ T x C** (where D is the distance, T is the time, and C is the sound speed of 343 meters/second) is the formula for this computation. The module conducts these computations and emits a pulse width that is proportional to the observed distance; this pulse can be simply interfaced to any microcontroller.

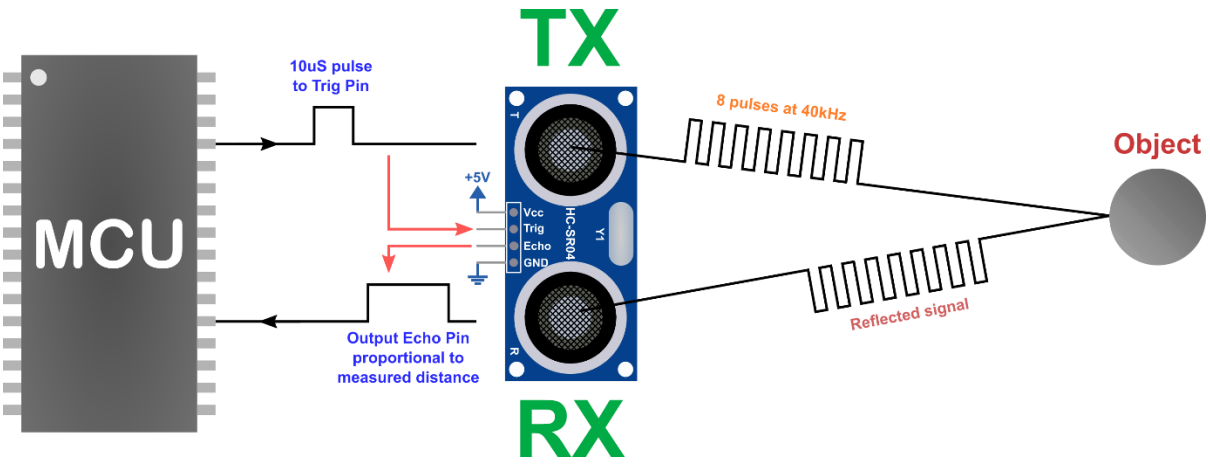


Figure 4.9: Ultrasonic Sensor Principle of Operation

4.2.4.2 Software:

for getting the distance we have to program the controller for the following:

1.Triggering the sensor by pulling up the trigger pin for at least 12uS.

2. Once echo goes high, we get an external interrupt, and we are going to start a counter (enabling a counter) in the ISR (Interrupt Service Routine), which is executed right after an interrupt triggered.

3Once echo goes low again, an interrupt is generated. This time we are going to stop the counter (disable the counter).

4. So for a pulse high to low at the echo pin, we have started a counter and stopped it. This count is updated to memory for getting the distance, as we have the width of echo in count now.

5. We are going to do further calculations in the memory to get the distance in cm.

4.2.4.3 Structure

The HC-SR04 ultrasonic sensor, like bats, uses sonar to calculate distance to an object. It offers excellent non-contact range detection with high accuracy and stable readings in an easy-to-use package. From 2cm to 400 cm or 1" to 13 feet. Its operation is not affected by sunlight or black material like Sharp rangefinders are (although acoustically soft materials like cloth can be difficult to detect). It comes complete with an ultrasonic transmitter and receiver module. Two metal cylinders protrude from the front of the ultrasonic range finder. These are called transducers. Mechanical forces are converted into electrical impulses via transducers. There are two transducers in the ultrasonic range finder: one transmitting and one receiving. The transmitting transducer turns an electrical signal into an ultrasonic pulse, and the receiving transducer translates the reflected ultrasonic pulse back into an electrical signal. If you look at the back of the range finder, you will see an IC behind the transmitting transducer labelled MAX3232. This is the IC that controls the transmitting transducer. Behind the receiving transducer is an IC labelled LM324. This is a quad Op-Amp that amplifies the signal generated by the receiving transducer into a signal that's strong enough to transmit to the microcontroller.

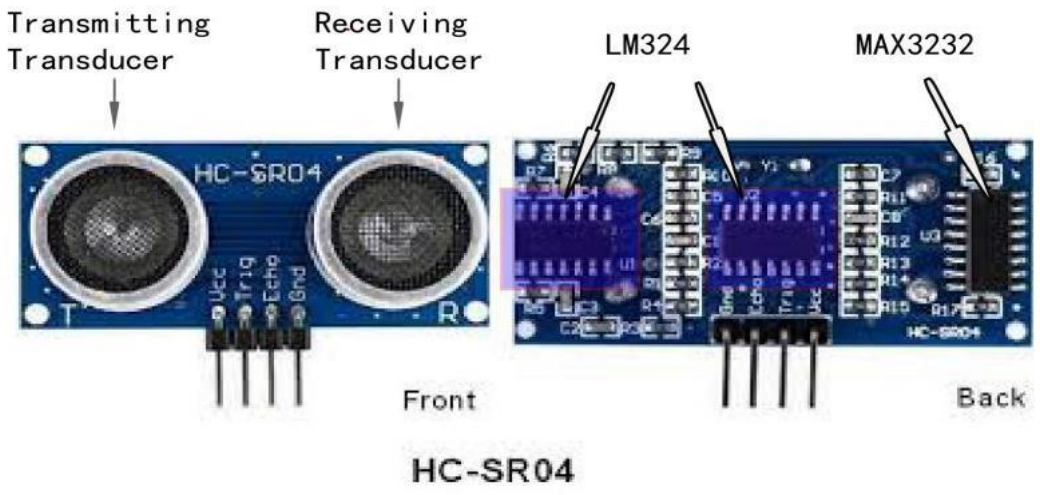


Figure 4.10 Ultrasonic Sensor Structure

Figure 4.11: Ultrasonic Sensor Pins

	Pin Symbol	Pin Function Description
1	VCC	5V power supply
2	Trig	Trigger Input pin
3	Echo	Receiver Output pin
4	GND	Power ground

Table 4.1: Ultrasonic Sensor Pin description

4.2.4.4 TIMING DIAGRAM

The HC-SR04 timing diagram is displayed. To begin measuring, the SR04's Trig must receive a high (5V) pulse for at least 10us. This will cause the sensor to broadcast an 8-cycle ultrasonic burst at 40kHz and wait for the reflected ultrasonic burst. When the sensor detects an ultrasonic signal from the receiver, it sets the Echo pin to high (5V) and delays for a duration (width) proportional to distance. To obtain the distance, measure the width (Ton) of Echo pin.

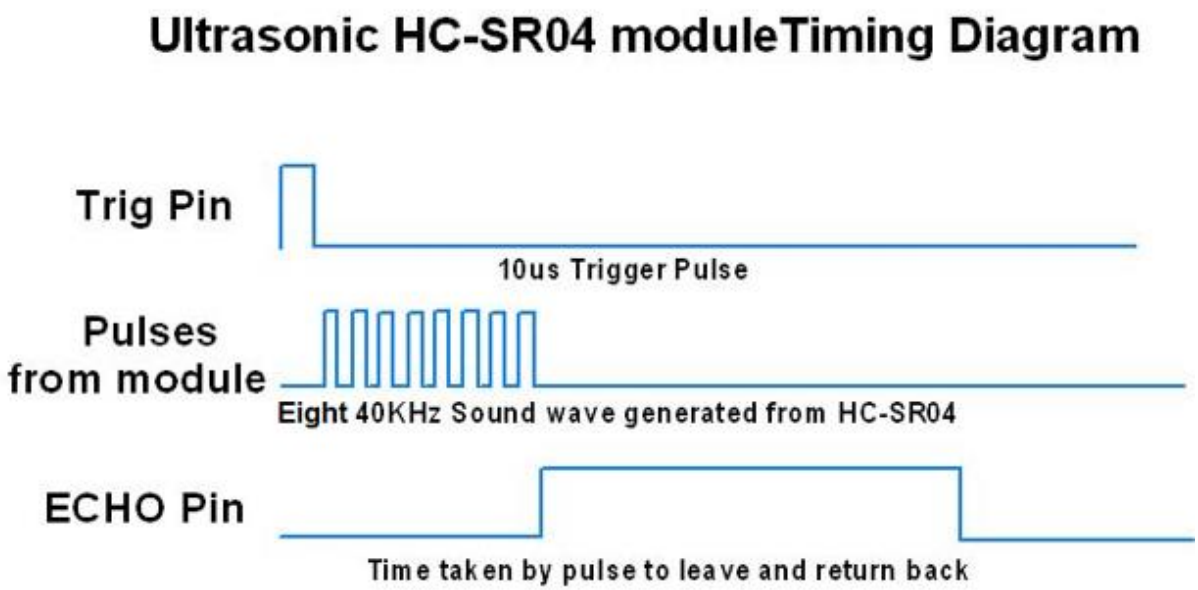


Figure 4.12: TIMING DIAGRAM

- Time = Width of Echo pulse, in us (micro second)
- Distance in centimeters = Time / 58
- Distance in inches = Time / 148

Alternatively, because sound travels through air at approximately 344 m/s (1129 ft./s), you can take the time it takes for the sound wave to return and multiply it by 344 meters (or 1129 feet) to find the total round-trip distance of the sound wave. Round-trip means that the sound wave travelled 2 times the distance to the object before it was detected by the sensor; it includes the 'trip' from the sonar sensor to the object AND the 'trip' from the object to the Ultrasonic sensor (after the sound wave bounced off the object). To find the distance to the object, simply divide the round-trip distance in half.

$$distance = \frac{speed\ of\ sound\ \times time\ taken}{2}$$

4.2.4.5 Specifications and Dimensions

- Power Supply: +5Vdc
- Supply Current: 10mA
- Measurement distance Range: 2cm to 400cm
- Trigger Input Pulse width: 10uS
- Size: 44.5mm W x 20mm H x 15mm D.
- Interface connector: 4-pin header SIP, 0.1” spacing.
- Operating temperature range: 0° - 70° C
- Resolution: 0.3 cm

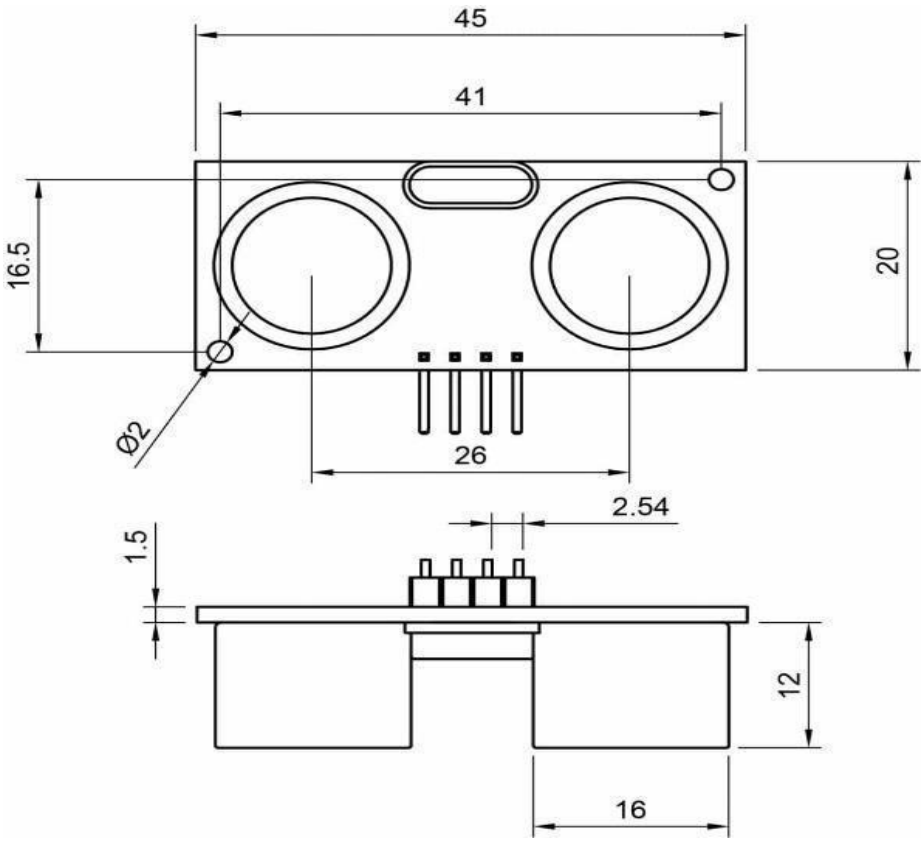


Figure 4.13: Ultrasonic Sensor Structure

4.2.4.6 Connection

Used four ultrasonic and their connections as shown in the following simulation :

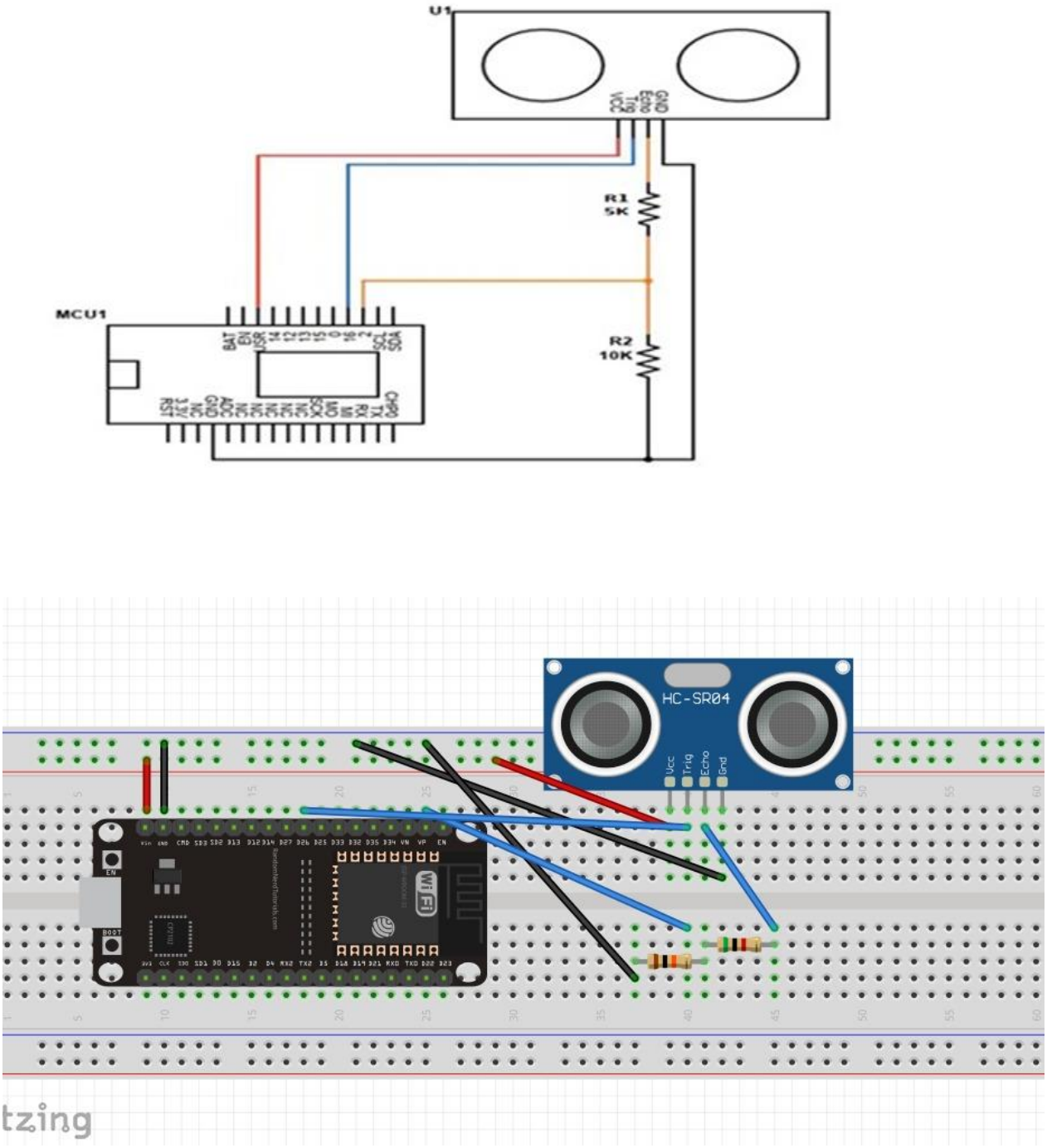


Figure 4.14: connection ESP32 with ultrasonic

4.3 Other Components

The project uses some other electronic components like:

1- 10 kΩ Resistors:

Figure 4.15: Resistor

2- Breadboard:

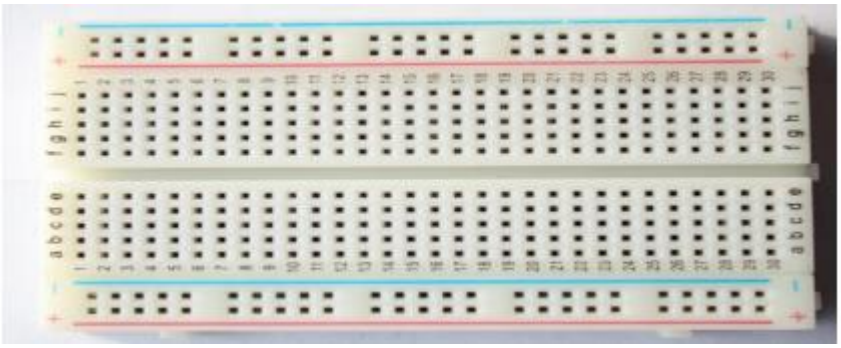


Figure 4.16: Breadboard

3- Connecting wires:

Figure 4.17: Connecting wire

Chapter 5

Experimental Work

5.1 Introduction

Network devices are physical devices that are required for communication and interaction between hardware on a network.

Figure 5.1: router device

5.1.1 Types of network devices

- Hub
- Switch
- Router
- Bridge
- Gateway
- Modem
- Repeater
- Access Point

And we use routers and access points from this list to make connections between the devices we use.

5.1.1.1 Router

Routers help transmit packets to their destinations by charting a path through the sea of interconnected networking devices using different network topologies. Routers are intelligent devices, and they store information about the networks they're connected to. Most routers can be configured to operate as packet-filtering firewalls and use access control lists (ACLs).

Routers are also used to translate from LAN framing to WAN framing. This is needed because LANs and WANs use different network protocols. Such routers are known as border routers. They serve as the outside connection of a LAN to a WAN, and they operate at the border of a network.

Routers are also used to divide internal networks into two or more subnetworks. Routers can also be connected internally to other routers, creating zones that operate independently. Routers establish communication by maintaining tables about destinations and local connections. A router contains information about the systems connected to it and where to send requests if the destination isn't known. [17]

5.1.1.2 Access Point

While an access point (AP) can technically involve either a wired or wireless connection, it commonly means a wireless device. An AP works at the second OSI layer, the Data Link layer, and it can operate either as a bridge connecting a standard wired network to wireless devices or as a router passing data transmissions from one access point to another.

Wireless access points (WAPs) consist of:

A transmitter and receiver (transceiver) device are used to create a wireless LAN (WLAN). Access points typically are separate network devices with a built-in antenna, transmitter and adapter. APs use the wireless infrastructure network mode to provide a connection point between WLANs and a wired Ethernet LAN. They also have several ports, giving you a way to expand the network to support additional clients. Depending on the size of the network, one or more APs might be required to provide full coverage. Additional APs are used to allow access to more wireless clients and to expand the range of the wireless network. Each AP is limited by its transmission range — the distance a client can be from an AP and still obtain a usable signal and data process speed. The actual distance depends on the wireless standard, the obstructions and environmental conditions between the

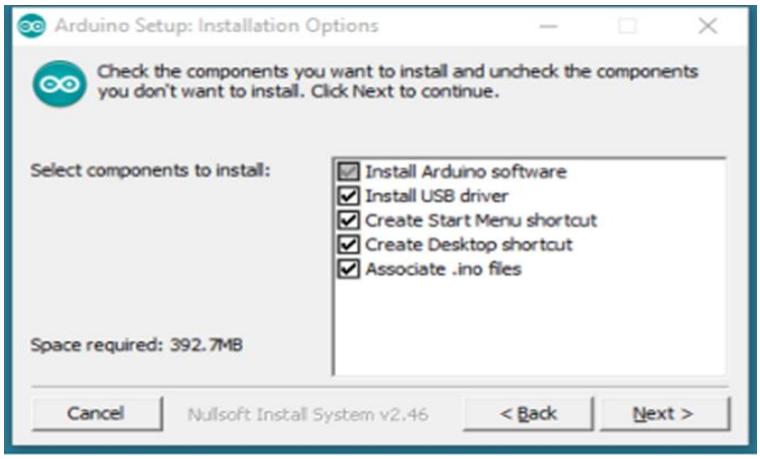
client and the AP. Higher end APs have high-powered antennas, enabling them to extend how far the wireless signal can travel. [17]

5.2 Download the Arduino Software (IDE)

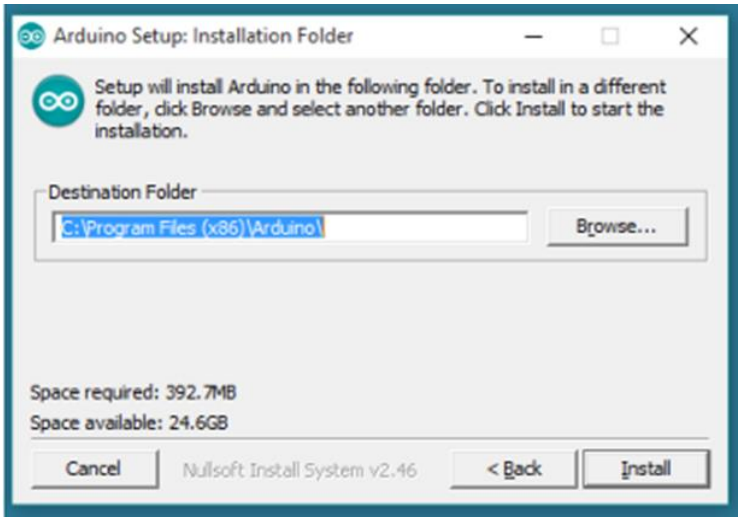
Installation can be done by downloading (.exe) and the Zip Packages. by using the first method installs directly everything you need to use the Arduino Software (IDE), including the drivers. With the Zip package you need to install the drivers manually. The Zip file is also useful if you want to create a portable installation.

When the download finishes, proceed with the installation and please allow the driver installation process when you get a warning from the operating system.

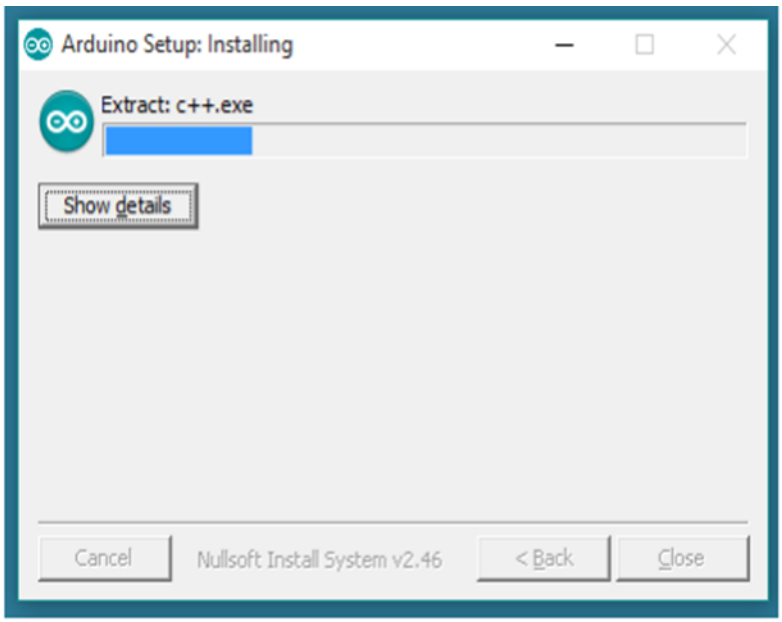
Choose the components to install.



Choose the installation directory.



The process will extract and install all the required files to execute properly the Arduino Software (IDE)

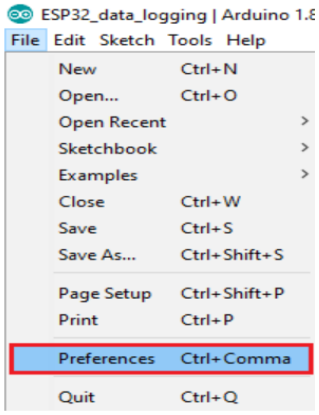


Proceed with board specific instructions. When the Arduino Software (IDE) is properly installed, you can go back to the Getting Started Home and choose your board from the list on the right of the page.

5.3 Installing the ESP 32 board in Arduino IDE:

To install the ESP32 board in your Arduino IDE, follow these next instructions:

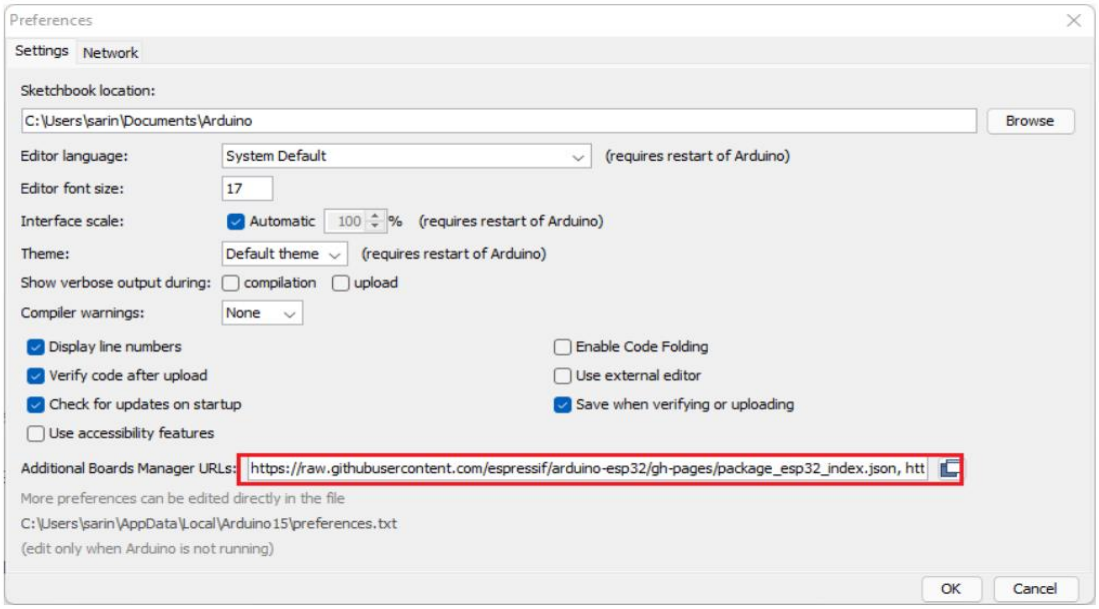
1- In your Arduino IDE, go to **File** then choose **Preferences**



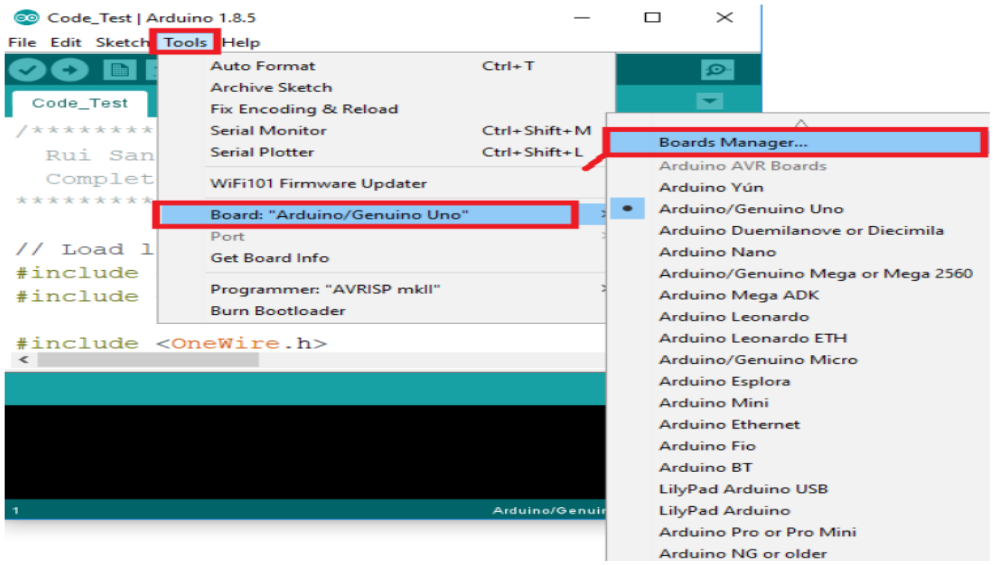
2- Enter the following into the 'Additional Board Manager URLs' field:

https://raw.githubusercontent.com/espressif/arduino-esp32/gh pages/package_esp32_index.json

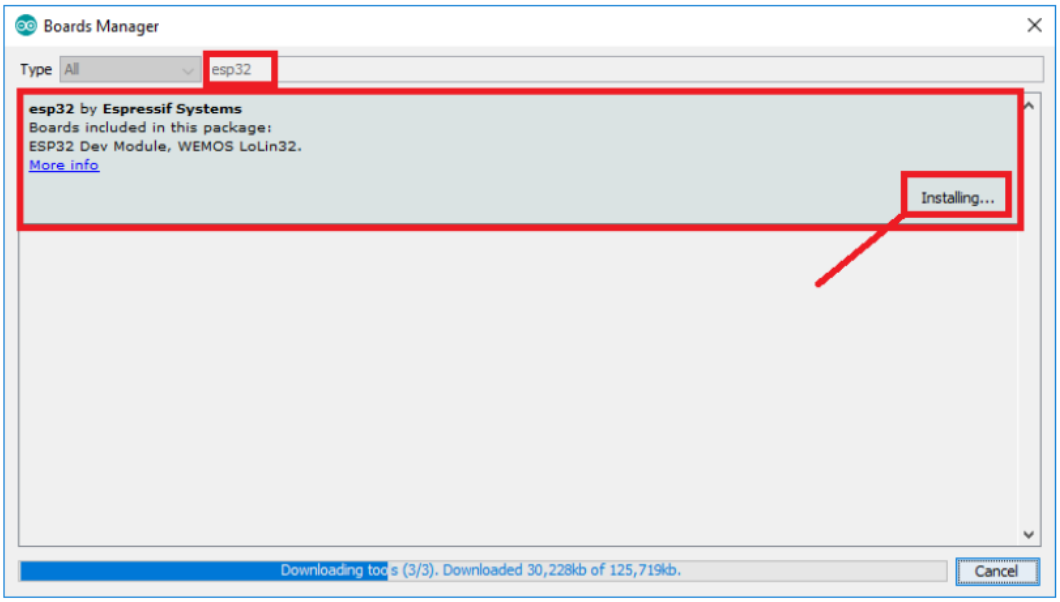
Then, click the 'OK' button:



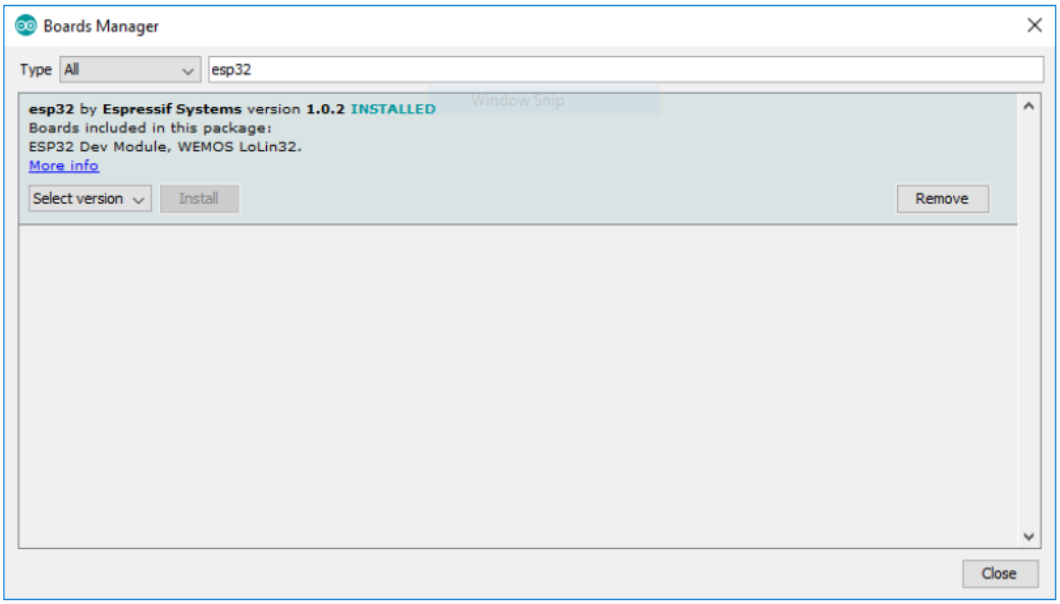
3- Open the Boards Manager. Go to **Tools** > **Board** > **Boards Manager…**



4- Search for **ESP32** and press install button for the '**ESP32 by Espressif Systems**':

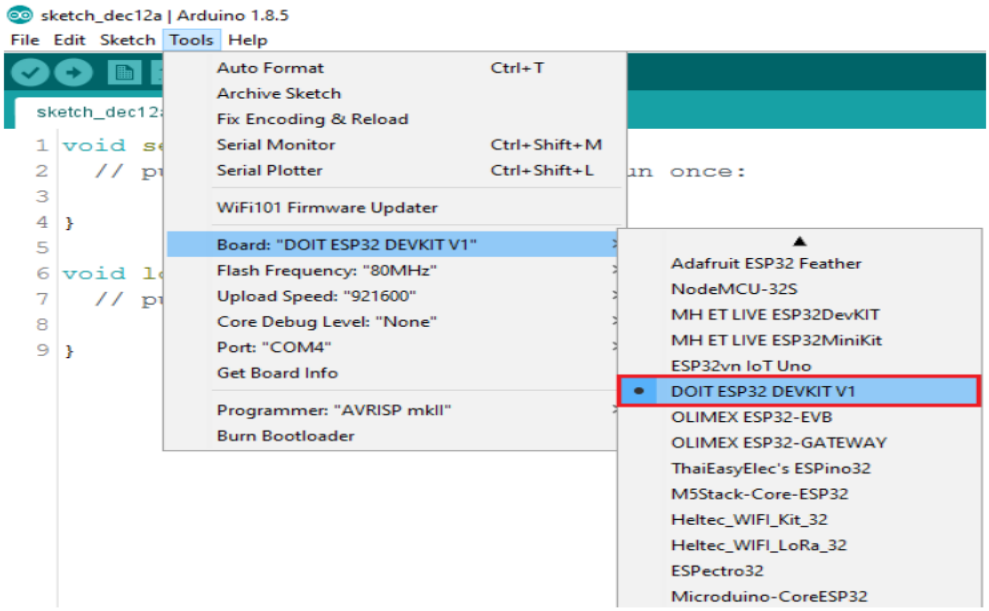


5- That's it. It should be installed after a few seconds:

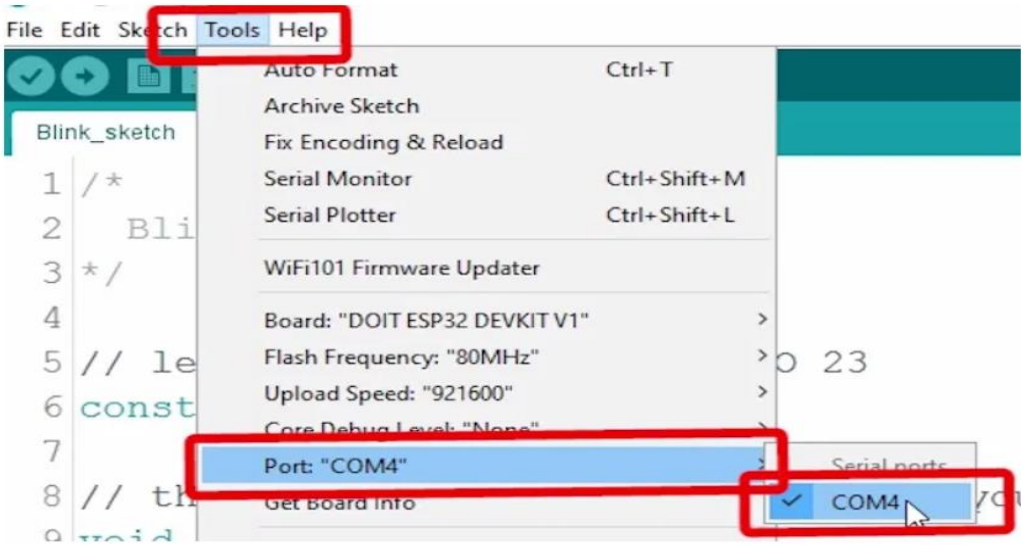


After the Installation:

1- Select your Board in **Tools** > **Board** menu (choose the **DOIT ESP32 DEVKIT V1**):



2- Select the Port (if you don't see the COM Port in your Arduino IDE, you need to install the [CP210x USB to UART Bridge VCP Drivers]):



5.4 Connection ESP 32 with server:

Step 1: we need to modify the following lines with our network credentials: SSID and password.

```
const char* ssid     = "WE_E66D88";
const char* password = "jbnl6065";
```

Step 2: we will open a serial connection to output some results from our program and then we will connect the ESP32 to the WiFi network.

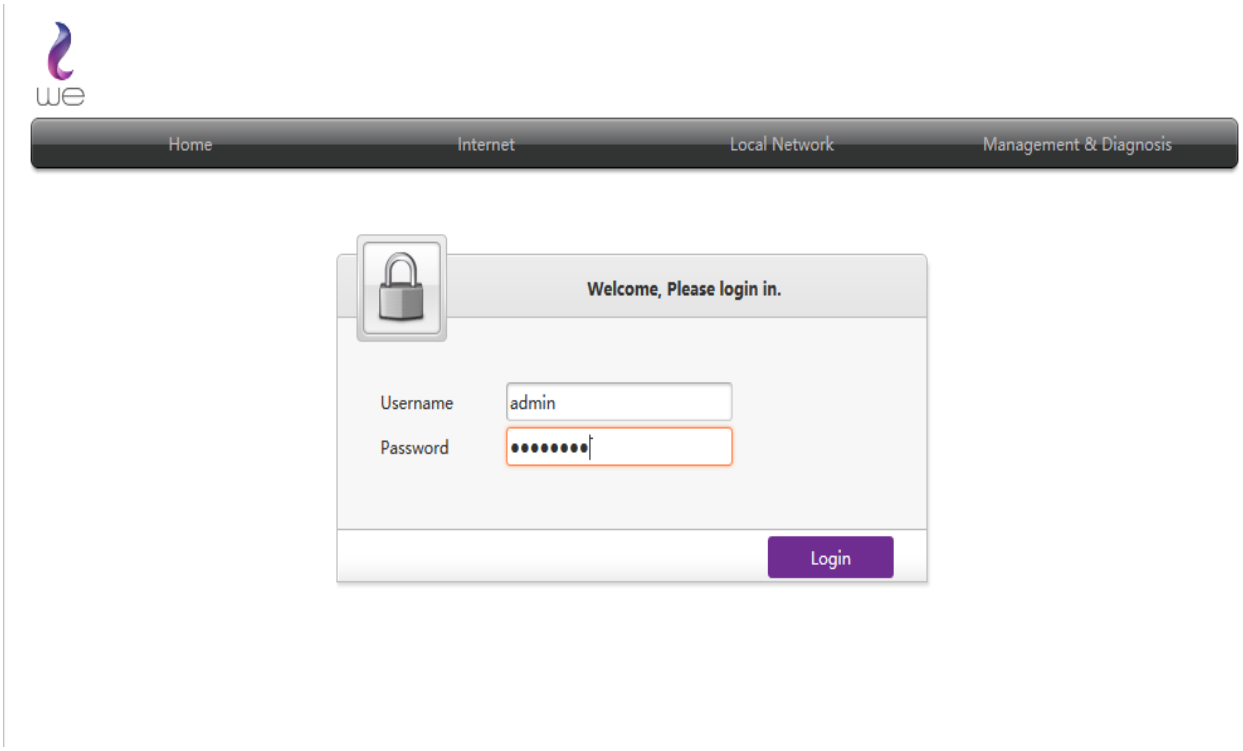
```
Serial.print("Connecting to ");
Serial.println(ssid);
WiFi.begin(ssid, password);

while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
}
```

Step 3: Add the IP address of the server:

We make it static it from router settings To remain constant and not change every time by follow the following settings:

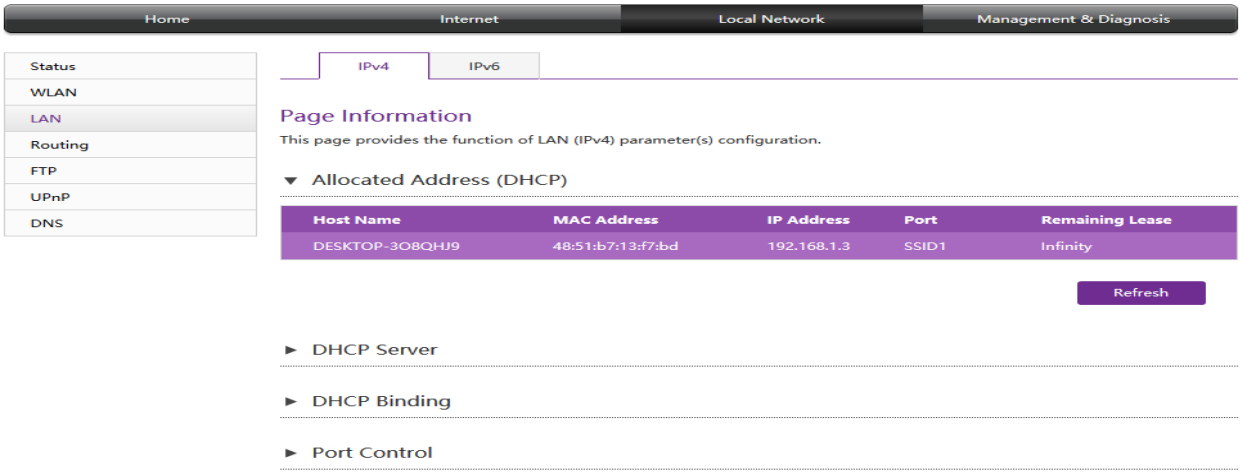
1- Identify the IP address of the router. Most routers are manufactured to use a default address such as 192.168.1.1.
2- In a web browser, such as Chrome, or Firefox, request a connection to the router. Do this by typing the router's IP address (in the format http://192.168.1.1) in the browser's address bar.

3- Enter the administrative login information—username and password—to authenticate and access the admin settings.



Then enter login.

4- Open the LAN settings and enter then DHCP Binding.



5- Then put the MAC address to the device we want to make it remain still and IP that we want.

Then enter Apply.

```
const char* httpHost = "192.168.1.3";
```

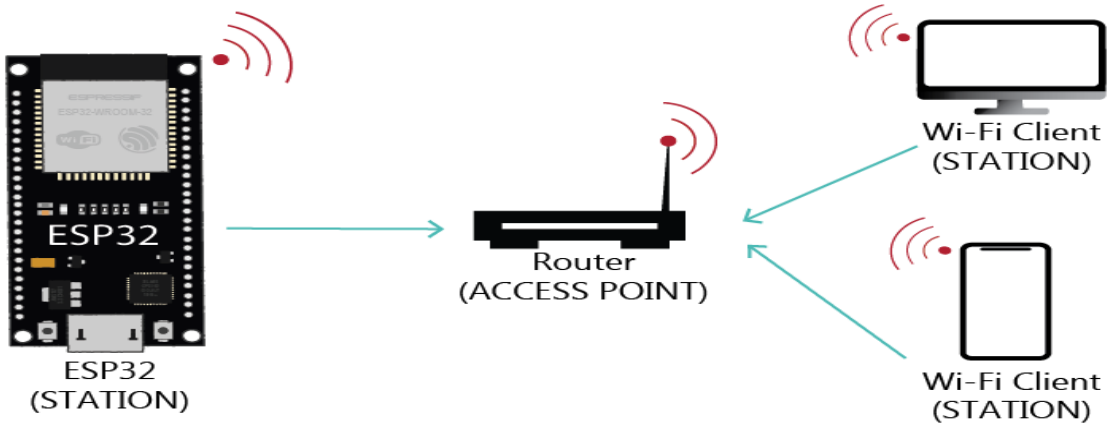
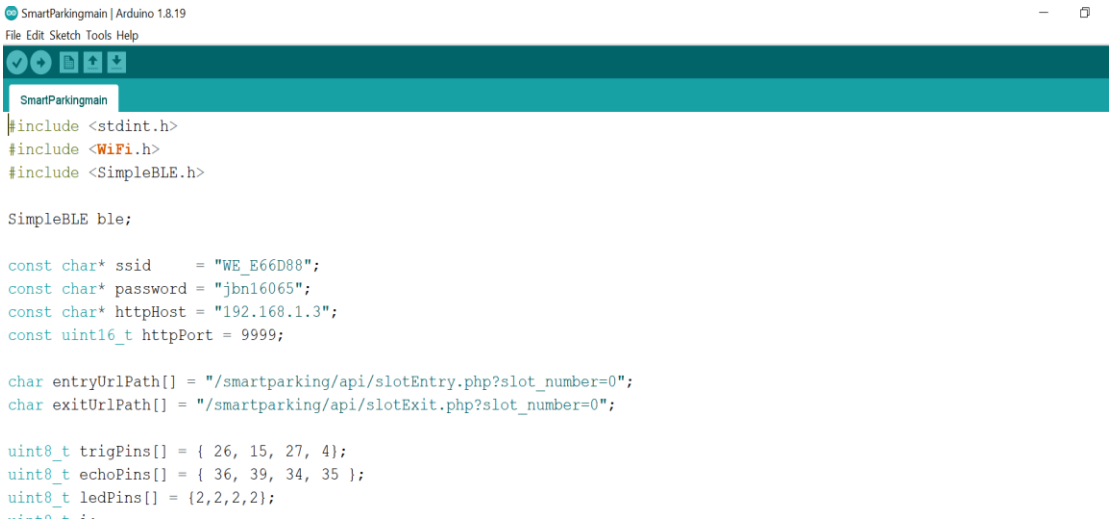


Figure 5.2: Connection ESP with server

5.5: code ESP32 with ultrasonic sensor:

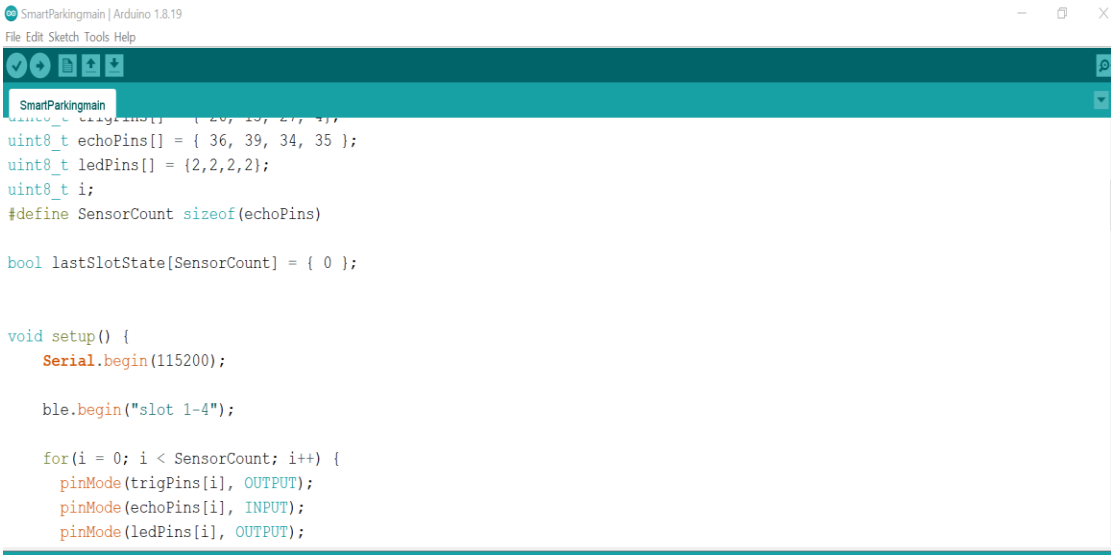



```
#include <stdint.h>
```

The **stdint.h** header defines integer types, limits of specified width integer types, limits of other integer types and macros for integer constant expressions.

```
#include <WiFi.h>
```

This library allows an Arduino board to connect to the internet. It can serve as either a server accepting incoming connections or a client making outgoing ones.



SmartParkingmain | Arduino 1.8.19

File Edit Sketch Tools Help

SmartParkingmain

```
    pinMode(ledPins[i], OUTPUT);
  }

  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

void loop() {
```

SmartParkingmain | Arduino 1.8.19

File Edit Sketch Tools Help

SmartParkingmain

```
void loop() {
  for(i = 0; i < SensorCount; i++) {
    float d;
    bool slotEmpty = 0;

    digitalWrite(trigPins[i], HIGH);
    delay(1);
    digitalWrite(trigPins[i], LOW);
    noInterrupts();
    d = pulseIn(echoPins[i], HIGH, 23529.4);
    interrupts();
    d /= 58.8235;
    slotEmpty = d > 100;
    digitalWrite(ledPins[i], slotEmpty ? HIGH : LOW);

    Serial.print("Slot ");
    Serial.print(i);
```


SmartParkingmain | Arduino 1.8.19
File Edit Sketch Tools Help
SmartParkingmain
Serial.print(i);
Serial.println(":");
Serial.print("D ");
Serial.print(d);
Serial.print("\tStatus: ");
Serial.println(lastSlotState[i] ? "Empty" : "Full");
if(slotEmpty != lastSlotState[i]) {
// select the correct url depending on the event (entry or exit)
char *path = slotEmpty ? exitUrlPath : entryUrlPath;
// get the selected url length
char pathLen = slotEmpty ? sizeof(exitUrlPath) - 1 : sizeof(entryUrlPath) - 1;
// modify the last character in the url (which is the slot number)
// by adding the integer value of the slot number to the character value 0
// which effectively gives us a character that represents the slot number
path[pathLen - 1] = '0' + i;
Serial.print("Updating status to: ");


SmartParkingmain | Arduino 1.8.19
File Edit Sketch Tools Help
SmartParkingmain
Serial.print("Updating status to: ");
Serial.println(slotEmpty ? "Empty" : "Full");
if(sendSlotStatus(httpHost, httpPort, path)) {
lastSlotState[i] = slotEmpty;
}
}
Serial.println();
delay(250);
}
// delay(10000);
}
bool sendSlotStatus(const char* httpHost, uint16_t httpPort,const char* url) {
WiFiClient client;

SmartParkingmain | Arduino 1.8.19

File Edit Sketch Tools Help

SmartParkingmain

```
  WiFiClient client;

  if (!client.connect(httpHost, httpPort)) {
      Serial.println("connection failed");
      return false;
  }

  Serial.print("Requesting URL: ");
  Serial.println(url);

  // send the request to the server
  client.print(String("GET ") + url + " HTTP/1.1\r\n"
               + "Host:" + httpHost + "\r\n"
               + "Connection: close\r\n\r\n");

//  uintmax_t timeout = millis();

//  while(client.available()) {
```

SmartParkingmain | Arduino 1.8.19

File Edit Sketch Tools Help

SmartParkingmain

```
//  while(client.available()) {
//    if (millis() - timeout > 5000) {
//        Serial.println(">>> Client Timeout !");
//        client.stop();
//        return false;
//    }
//  }
  delay(5000);

  // Read all the lines of the reply from server and print them to Serial
  Serial.println("Server Reponse:");
  while(client.available()) {
      String line = client.readStringUntil('\r');
      Serial.print(line);
  }
```

SmartParkingmain | Arduino 1.8.19

File Edit Sketch Tools Help

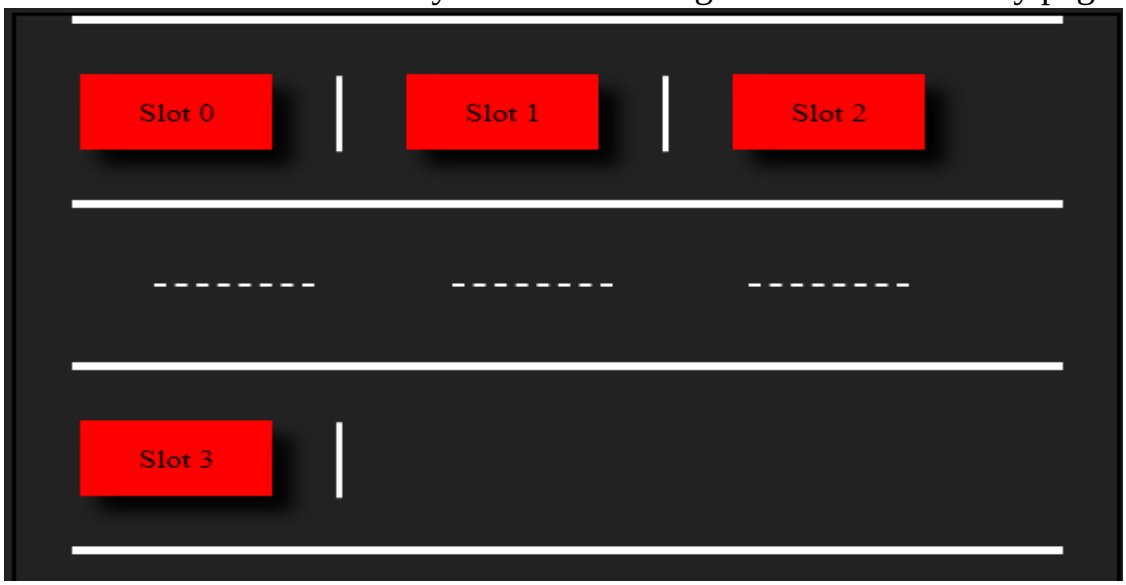
SmartParkingmain

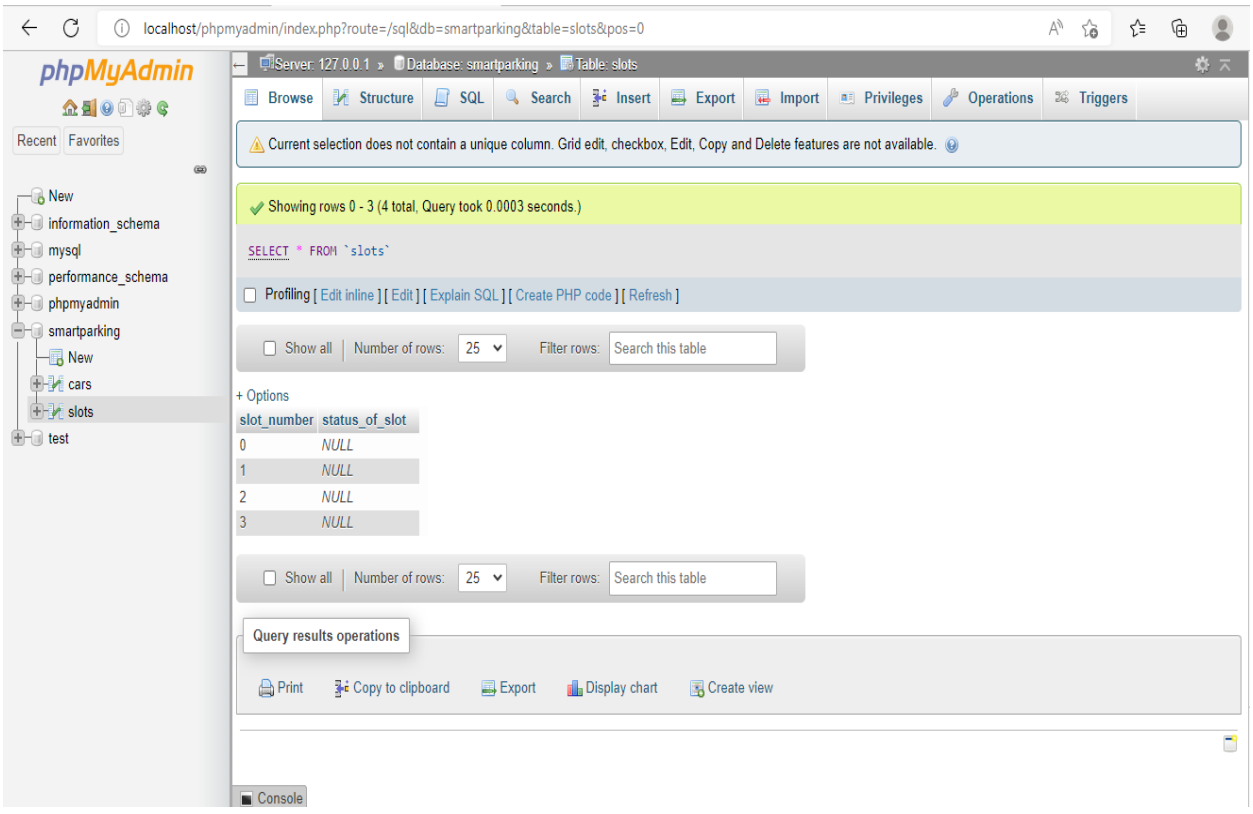
```
//        Serial.println(">>> Client Timeout !");
//        client.stop();
//        return false;
//    }
//  }
  delay(5000);

  // Read all the lines of the reply from server and print them to Serial
  Serial.println("Server Reponse:");
  while(client.available()) {
      String line = client.readStringUntil('\r');
      Serial.print(line);
  }

  client.stop();

  return true;
}
```

5. Press the **Upload** button in the Arduino IDE. Wait a few seconds while the code compiles and uploads to your board.

6. If everything went as expected, you should see a '**Done uploading.**' message.

Before sensor detect car entry there is no change in database or entry page:





After the sensor detect car entry, there is a change in the database and entry page:

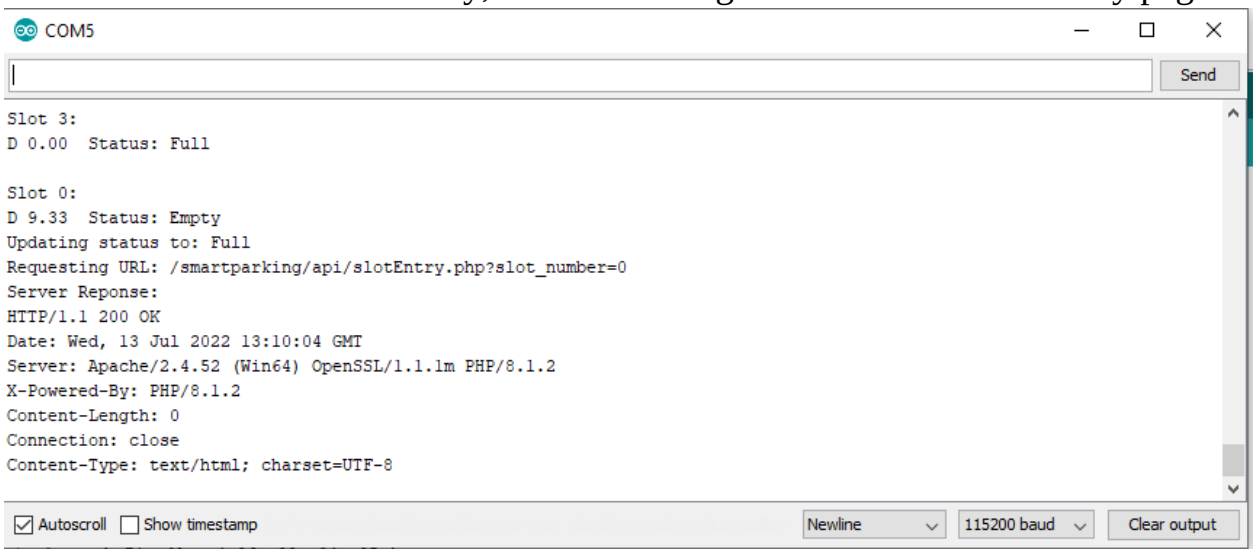
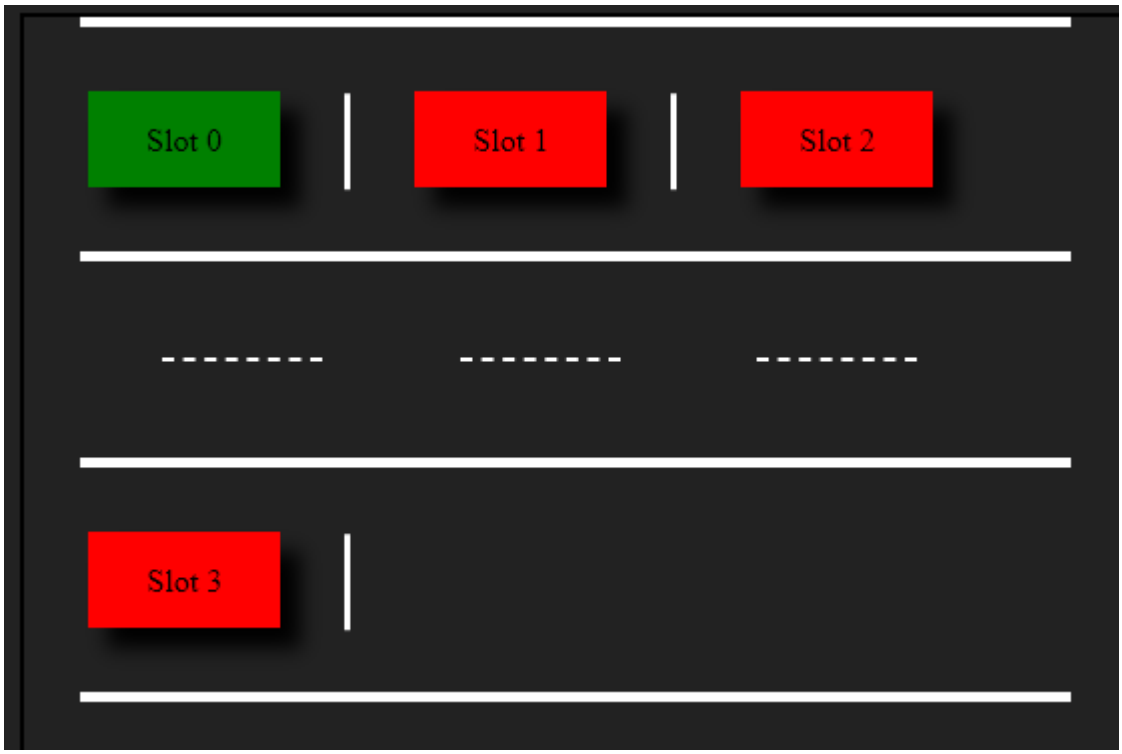
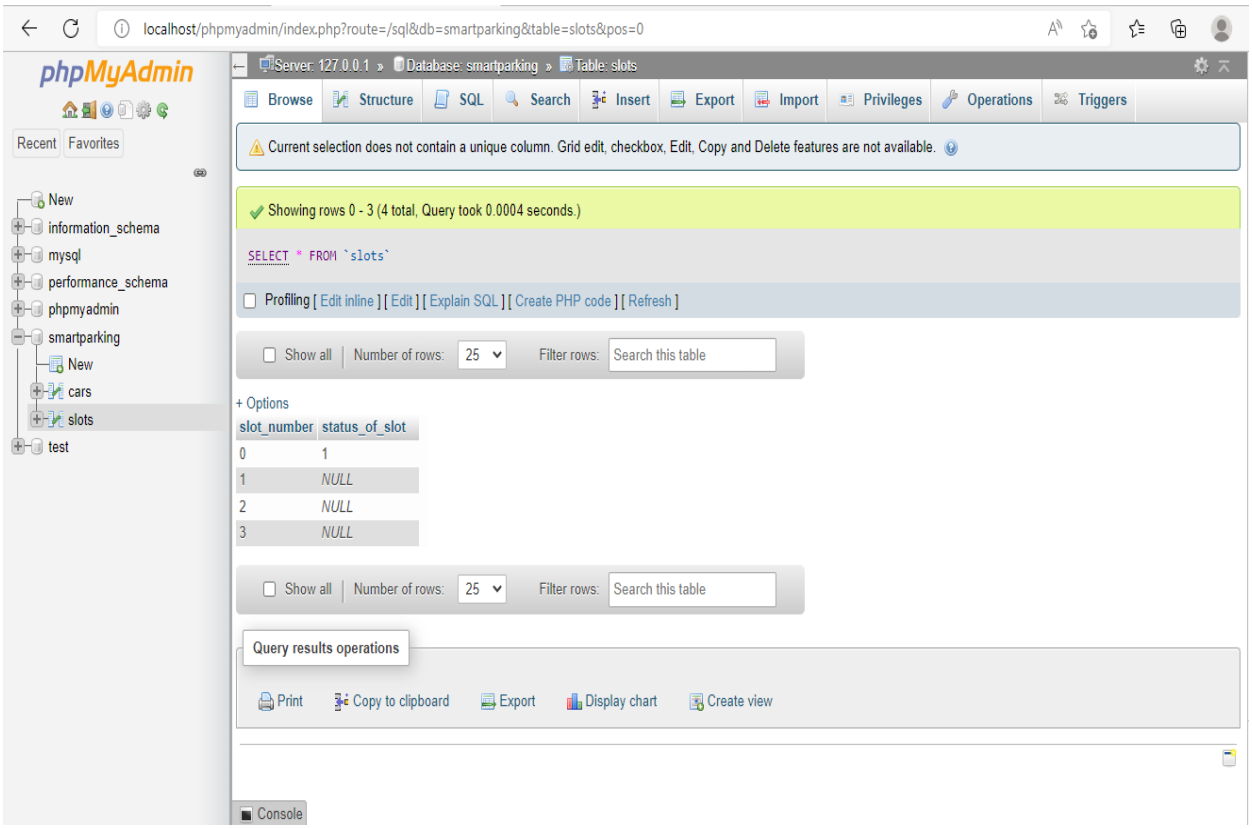


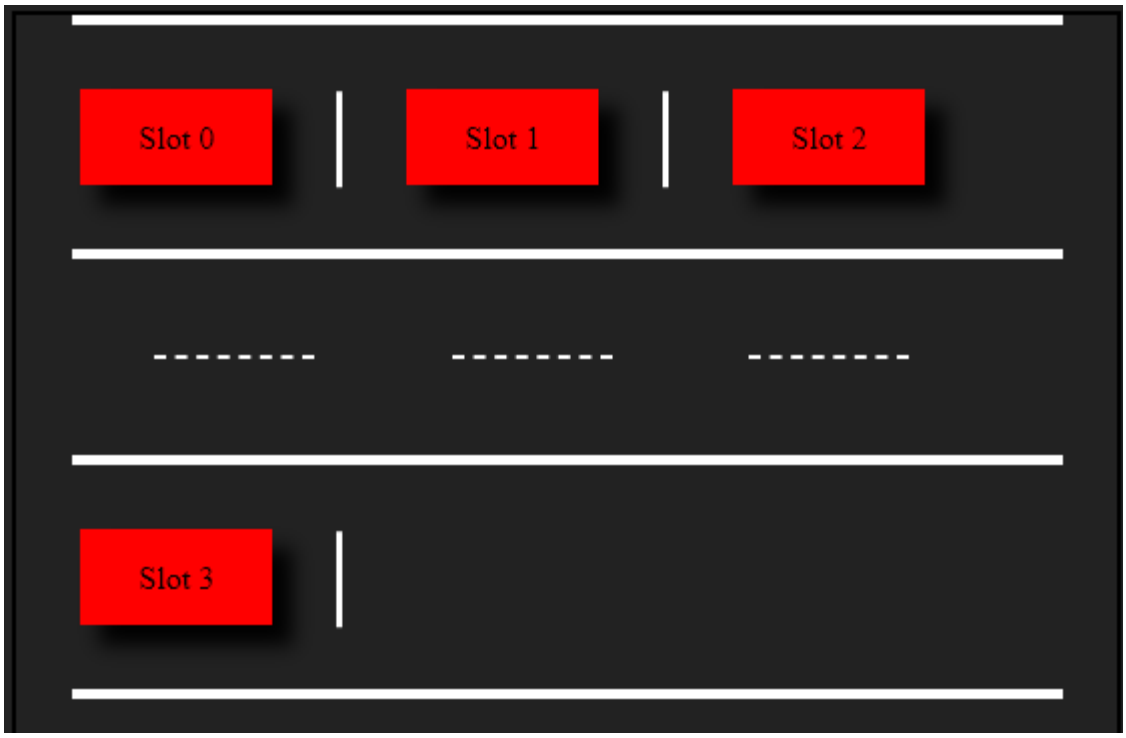
Figure 5.3: Sensor detect car entry

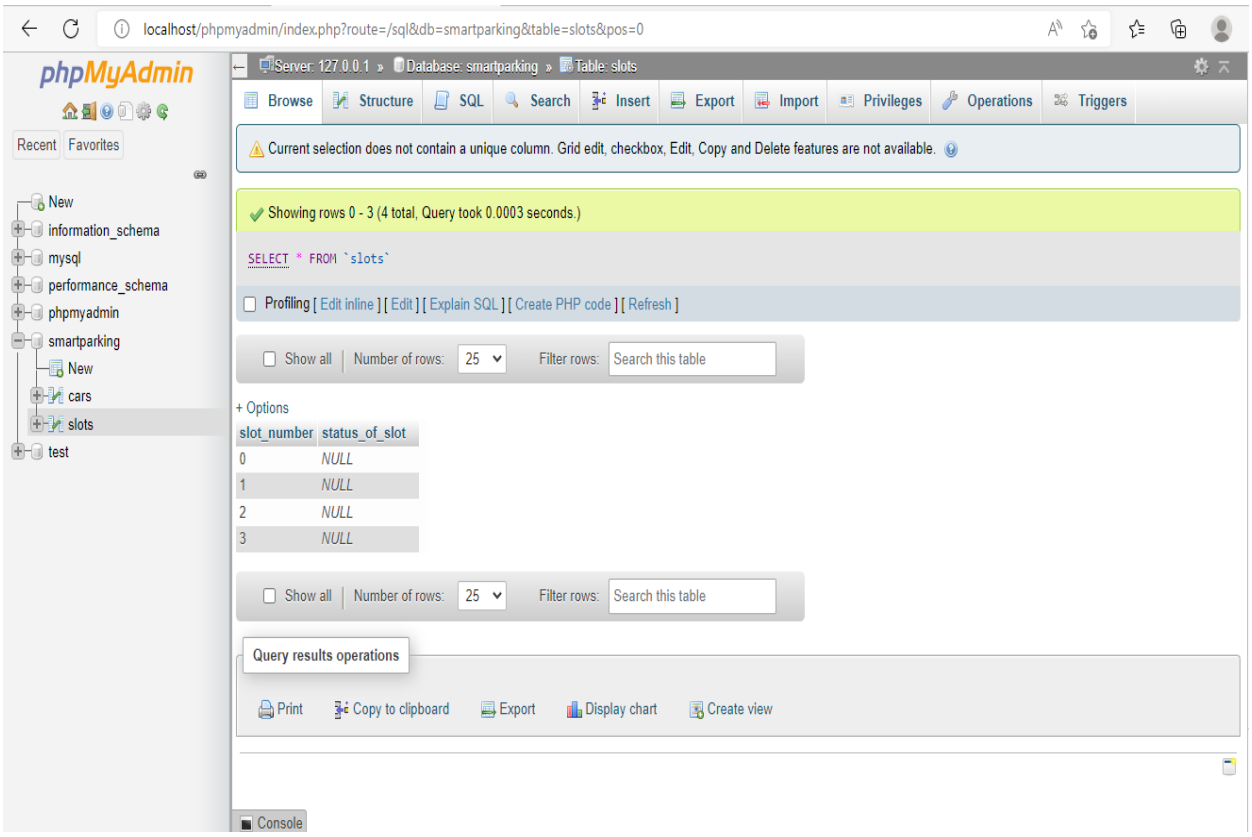




When the senser detect car leave:

Figure 5.4: Sensor detect car leave


Slot 0
Slot 1
Slot 2
Slot 3


localhost/phpmyadmin/index.php?route=/sql&db=smartparking&table=slots&pos=0
phpMyAdmin
Recent
Favorites
New
information_schema
mysql
performance_schema
phpmyadmin
smartparking
New
cars
slots
test
Server: 127.0.0.1 » Database: smartparking » Table: slots
Browse
Structure
SQL
Search
Insert
Export
Import
Privileges
Operations
Triggers
Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.
Showing rows 0 - 3 (4 total, Query took 0.0003 seconds.)
SELECT * FROM `slots`
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all
Number of rows: 25
Filter rows: Search this table
+ Options
slot_number
status_of_slot
0 NULL
1 NULL
2 NULL
3 NULL
Query results operations
Print
Copy to clipboard
Export
Display chart
Create view
Console

5.6 Connection Raspberry pi with server:

4.6.1 Urllib request

Extensible library for opening URLs
The urllib.request module defines functions and classes which help in opening URLs (mostly HTTP).

HTTP request is being made to a local server. The local server is requested to forward the request or service from a valid cache and return the response.

4.6.2 Implementation

contact the server to save the detected plate number.

To get started, you will make a request to the server, and the server will return an HTTP message. we're using Python 3 or above, and then use the urlopen() function from urllib.request.

For entry camera:

```
if i == 0:  # entry camera
    print("Sending entry data to server")
    try:
        print(urllib.request.urlopen(
            f"http://{hostname}/SmartParking/api/carEntry.php?car_plate_number={plateNumber}").read())
    except:
        print("No response from server")
```

For exit camera:

```
else:  # exit camera
    print(f"Sending exit data to server")
    try:
        print(urllib.request.urlopen(
            f"http://{hostname}/SmartParking/api/carExit.php?car_plate_number={plateNumber}").read())
    except:
        print("No response from server")
```

Then, the PlateNumber detected will be transferred to the server

Conclusion

Finally, and after we explained in detail what the idea and implementation of the project were, we would like to emphasize the importance of smart garages nowadays using the technologies that we used. We believe that the features we suggest are among the best for dealing with the inconvenience of parking.Also, to make a smart parking suitable to the smart cities arising now.

Future work: currently pytesseract is capable of reading texts from images but isn't accurate one hundred percent and also can't recognize Arabic alphabets, and we noticed that pytesseract is more accurate when the font is black and the background is white, so we can make adjustments to the plate image to increase the efficiency of pytesseract.

We hope that you like our work.

References

[1] Vehicle License Plate Detection and Recognition ,Guanghan Ning ,University of Missouri--Columbia,2013

[2] Visual Object Recognition,Kristen Grauman, Bastian Leibe,Morgan & Claypool Publishers , 2011, 1598299689, 9781598299687

[3]Proceedings of International Conference on Communication and Artificial Intelligence: ICCAI 2021,Vishal Goyal,Springer Nature, 9811909768, 9789811909764

[4]Practical Machine Learning and Image Processing: For Facial Recognition, Object Detection, and Pattern Recognition Using Python,Himanshu Singh,Apress, 2019, 1484241495, 9781484241493

[5] Intelligent Systems Design and Applications: 21st International Conference on Intelligent Systems Design and Applications (ISDA 2021) Held During December 13–15, 2021,Ajith Abraham, Niketa Gandhi, Thomas Hanne, Tzung-Pei Hong, Tatiane Nogueira Rios ,Weiping Ding,Springer Nature, 2022, 303096308X, 9783030963088

[6] https://www.techslang.com/definition/what-is-frontend-development/

[7] https://flatironschool.com/blog/what-is-front-end-development

[8] https://learntocodewith.me/posts/backend-development/

[9] https://www.tutorialspoint.com/nodejs/nodejs_introduction.htm

[10] https://en.wikipedia.org/wiki/Express.js

[11] https://www.tutorialspoint.com/nodejs/nodejs_express_framework.htm

[12] https://www.cloudflare.com/learning/ddos/glossary/hypertext-transfer-protocol-http/

[13] https://www.ibm.com/docs/en/rational-soft-arch/9.6.1?topic=page-asynchronous-javascript-xml-ajax-overview

[14] https://www.oracle.com/database/what-is-database/

[15] https://httpd.apache.org/

[16] https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html

[17] **https://blog.netwrix.com/2019/01/08/network-devices-explained/**

جامعة الأزهر
كلية الهندسة
قسم الاتصالات و الالكترونيات

مشروع تخرج

نظام وقوف السيارات الذكى

لجنة الاشراف

د: اسامه عبد الفتاح

د: احمد زقزوق

فريق عمل المشروع:

نريمان السيد محمد
الاء عبد الوهاب عيد
هاجر محمد ابراهيم
مريم يوسف محمد
الاء جبر عبد الباقي
مريم هشام صلاح الدين

2021-2022