Fall 2021

ECEN 452: Physical Sensors, Transducers & Instru

Smart Parking using Arduino and LDR

An "ECEN 452: Physical Sensors, Transducers & Instru" Project Report by:

Adham Mohammed 18101060

Mazen Mohamed 18101368

Rania Mahmoud Mohamed Soliman 19100230

Submitted in partial fulfilment for the requirements

Of the ECEN 452 project to

Dr. Ahmed Abo Bakr

Eng. Mostafa Abd elrahman

Contents

Abstract 3

Introduction 4

Background and related work 5

Specification of the project 6

Implementation 7

Software simulation 7

Hardware implementation (Architecture) 12

Software Architecture and Program code 15

Evaluation & conclusion 24

Future Works 25

References 26

Abstract

In the recent decade, the concept of smart cities has gained huge popularity. The idea of smart city is now more achievable and requested by many clients in the market. Our project introduces the idea of the smart parking, which depends on Arduino, light dependent resistor sensors, infrared sensors and seven-segment display module with a potential to add plate recognition system as the authentication system for the first entrance gate, to proceed in crafting full-functional and practical smart parking system. The project idea is to help the drivers to know which parking has available slot and which slots precisely are available by indicating the number of the total available slots in the parking on the seven-segment display, which will function as a counter, and a line of leds indicating the available slots depending on infrared sensors.

Introduction

The project aim is to guarantee the flexibility of a parking garage to full and free its maximum capacity without any endangerment to time consuming or space clashing as in dense environments it leads to social conflicts and over gas consuming which may lead to environmental and social issues. Our project functionality begin by checking the whether there is an available slot for the incoming vehicle. If there is available slot, it will enter the parking and the number of the available slots will decrease by one (or the number of the entering vehicles). If the available slots reach zero, then the parking will not allow any vehicle to enter. Unless, the vehicles inside started to leave, which will allow another vehicles to enter. The seven-segment module displays the number of the available slots according to every check in and out.

Since there are many cars nowadays, finding a parking lot is not easy sometimes. This is a generally a frustrating and time-consuming process that many citizens must go through. In addition to parking is a daily pain, it also has a huge impact on the pollution of the land. Smart parking and smart parking sensors can be seen as part of a smart city. These smart cities are cities driven by IT infrastructure that can be used by cities to improve the quality of life of their inhabitants and improve their economic development. Being a smart city can be a good way to collect historical data in a relatively easy way. By collecting this data, cities can analyze how they can optimize processes such as parking. Because of using Smart parking, those looking for a parking space can find the parking space in the most efficient way possible, and businesses and local governments can optimize the parking space. It also makes the city more livable, safer and less crowded. Using a Smart Parking system saves a lot of time for drivers since they know where to find a vacant parking spot. The amount of time you spend while looking for a parking spot will be minimized.

In Section 1, we discuss the background and related work to the topic. In section 2, we give a detailed description of the project. Afterwards, we will discuss the implementation of the software architecture and coding, and the hardware features and its implementation. Lastly, we are discussing our future work and our scope for moving forward.

Background and related work

In order to compare our work and study it's relation with other projects we must consider the main tools that is combined to reach our aim which is identify parking occupancy information to facilitate and improve parking efficiency which are, **sensors**, **technologies**, and **applications** that used to reach our aim. However, mostly the major tool is the sensor which will based our comparison on.

Our main preference to our application is it depended on **"LDR"**, which helped us to function a stationary gate with main stationary sensors that their number will be based on the gate security logic, instead of specializing a sensor to each car like in **"Ultra-sonic"** based on project. And that with the help of the parallel function of a seven-segment display, which translated the situation externally to the driver to enrich his information about the parking status to help him in his decision. Illustrating our main toll position in the following comparison table,

	Detection	**Maintenance and Development**	**Environmental responsivity**
Smart Parking (depend on LDR Sensor)	Depend on the change of resistance as the change of the amount of it's occupying light	Does not need an expensive maintenance, just basic providing especially light source privacy.	Suitable for indoors and outdoors in the terms of considering vehicles as the only effectives of the sensor light source that will not compared with human movement
Smart Parking (depend on Infrared Sensor)	Depend on the amount of infrared energy reflected from any object or vehicle	Requirement of advanced speculation and maintenance **and** to acquire the parking status as they should be placed in all the parking spaces	sensitive to the change of an environment like rain or wind (not suitable to open areas)
Smart Parking (depend on Ultrasonic Sensor)	Depend on the amount of a limited frequency sound waves reflected from an object or vehicle	Available for low cost, multiple sensors connection **and** maintenance is expensive at the long run and To acquire the parking status, sensors will be located on all parking spaces.	More suitable to open areas than IR but rather better in the indoors as the located ceiling may be sensitive to environment

Specification of the project

The project function is to check whether there is an available slot for the incoming vehicle. If there is available slot, it will enter the parking and the number of the available slots will decrease by one (or the number of the entering vehicles). If the available slots reach zero, then the parking will not allow any vehicle to enter. Unless, the vehicles inside started to leave, which will allow another vehicles to enter. The seven-segment module displays the number of the available slots according to every check in and out. The Entrance gate has two LDR (light dependent resistor) the first one senses the vehicle and make the Arduino check if there is an available space or not. The gate opens if there is available space, the vehicle moves in, which will make it pass on the second LDR to signal the Arduino that the vehicle is already inside and we can close the gate. The Third LDR is fixed on the exit gate, which senses if there is an object in front of it, so it will open the gate to let it out.

We have learnt about many things in this project: firstly, Coding and managing the Arduino to get the desired input with the other components. Secondly, we have learnt how the LDR operates, which is dependent on the light. The intense light lowers the resistance; however, the less light concentration increases the resistance significantly. Thirdly, the seven segment has two types' common anode and common cathode. In order to operate the common anode, the terminals need to read low to light the desired edges. On the other hand, the common cathode, its terminals need to read high to light the desired edges.

Implementation

Software simulation:

Our choice to simulation software was **"Proteus"** due to its flexibility in importing our components and illustrating its functions and the main connections, our simulation functionality based on uploading a **"hex file"** our programming code to our microcontroller **"Arduino Uno"**, which we will talk about in **software architecture and program code** section.

Fig1. Proteus Simulation components.

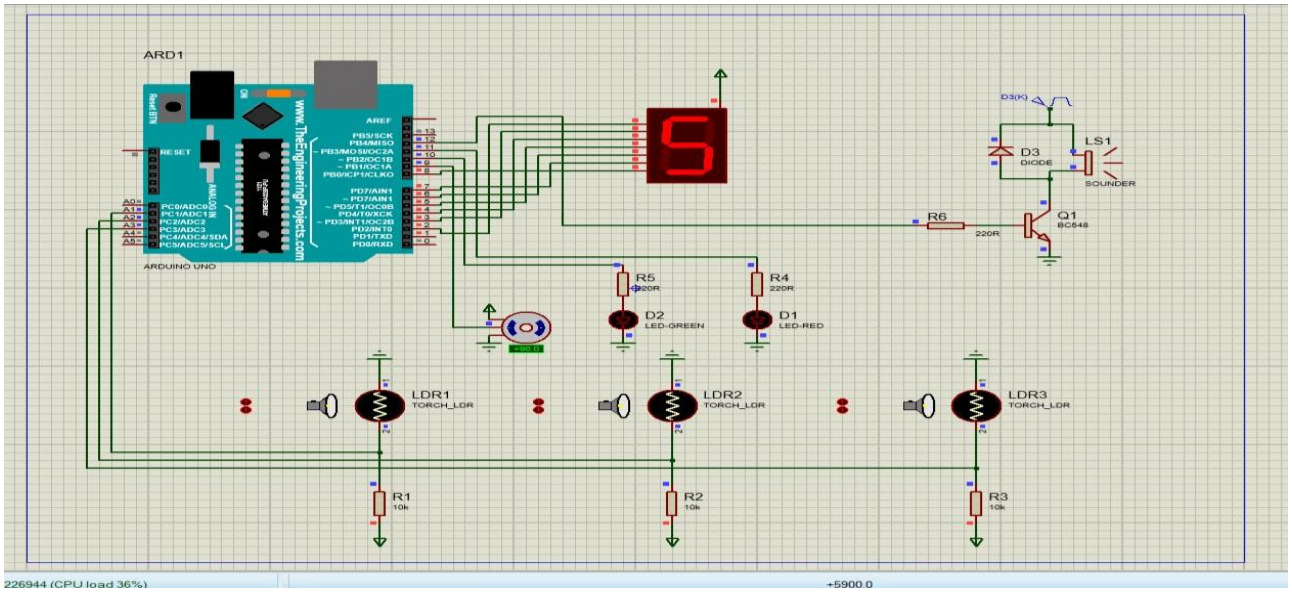
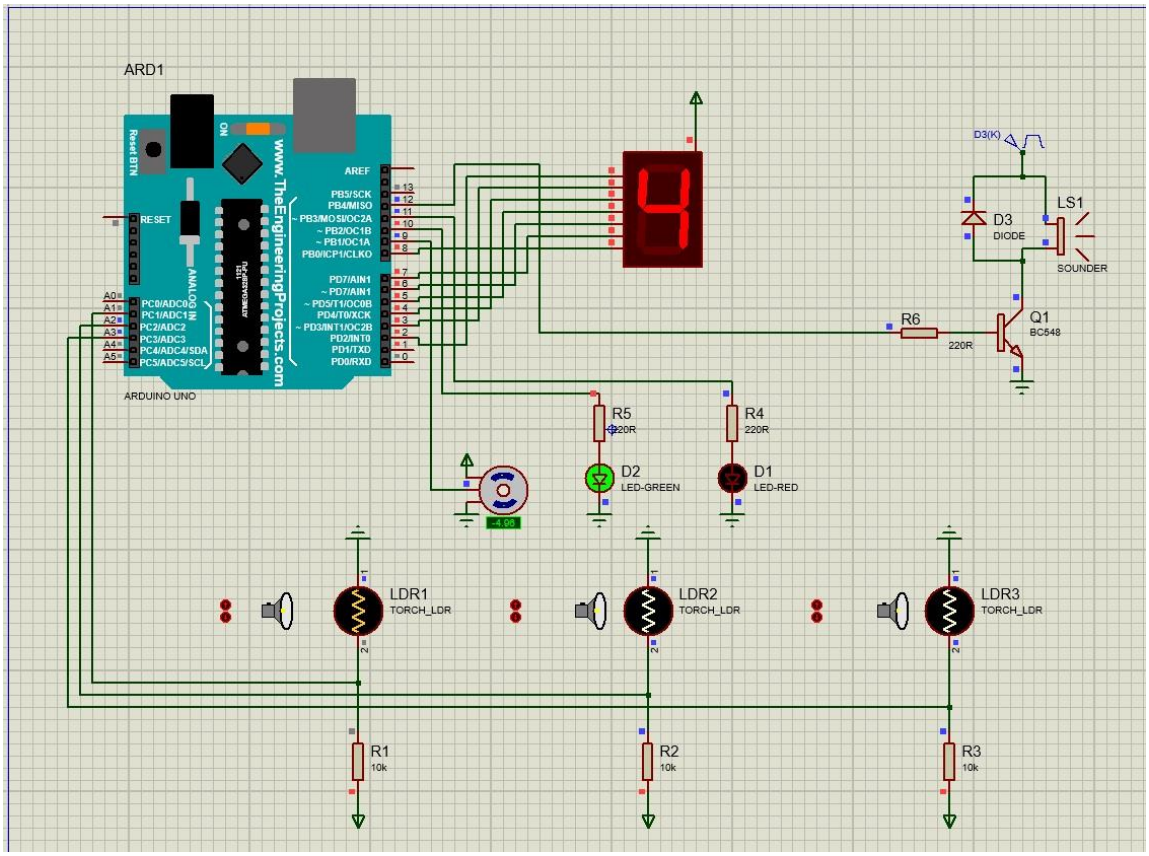


Fig2. Proteus Simulation circuit.

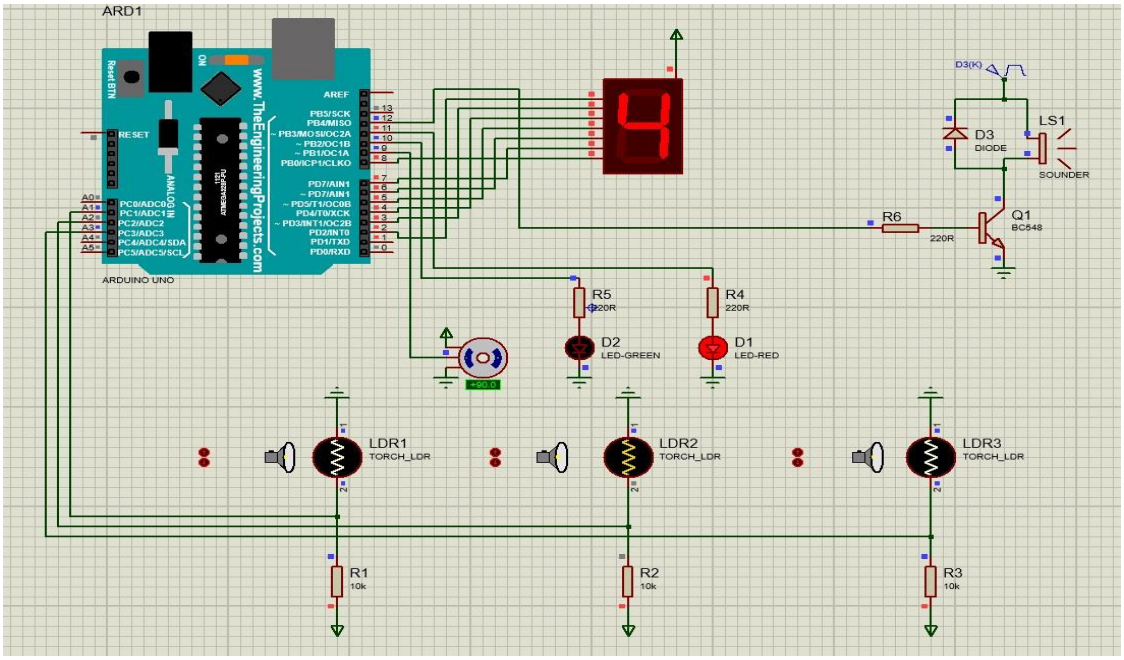
The most important components after the Arduino Uno is our **three LDR sensors** as they classify our **process of smart parking** into **three main stages**,

The **1st stage** is "gate opening",

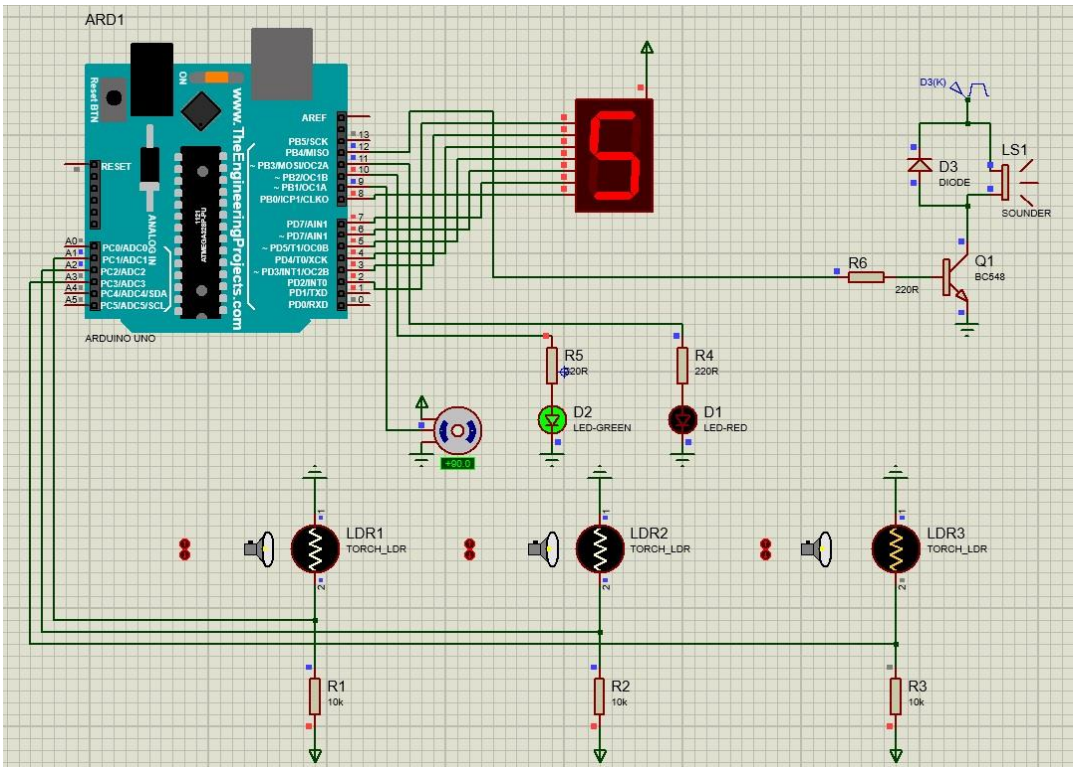


As in Fig2, the **7-segment** showing us that the **maximum capacity is 5**. Then by the passing of the first car in front of the sensor causing light change or light reducing in our cause, that will cause change in the resistance, sending signal to the Arduino causing the **servo motor to rotate and open the gate 90 degree**. **7-segment reducing the available spaces to 4**, **green light referring to a safe entering** and **finally an alerting buzzer.** Moreover, that what is the mechanism will depend mainly on in the next two stages.

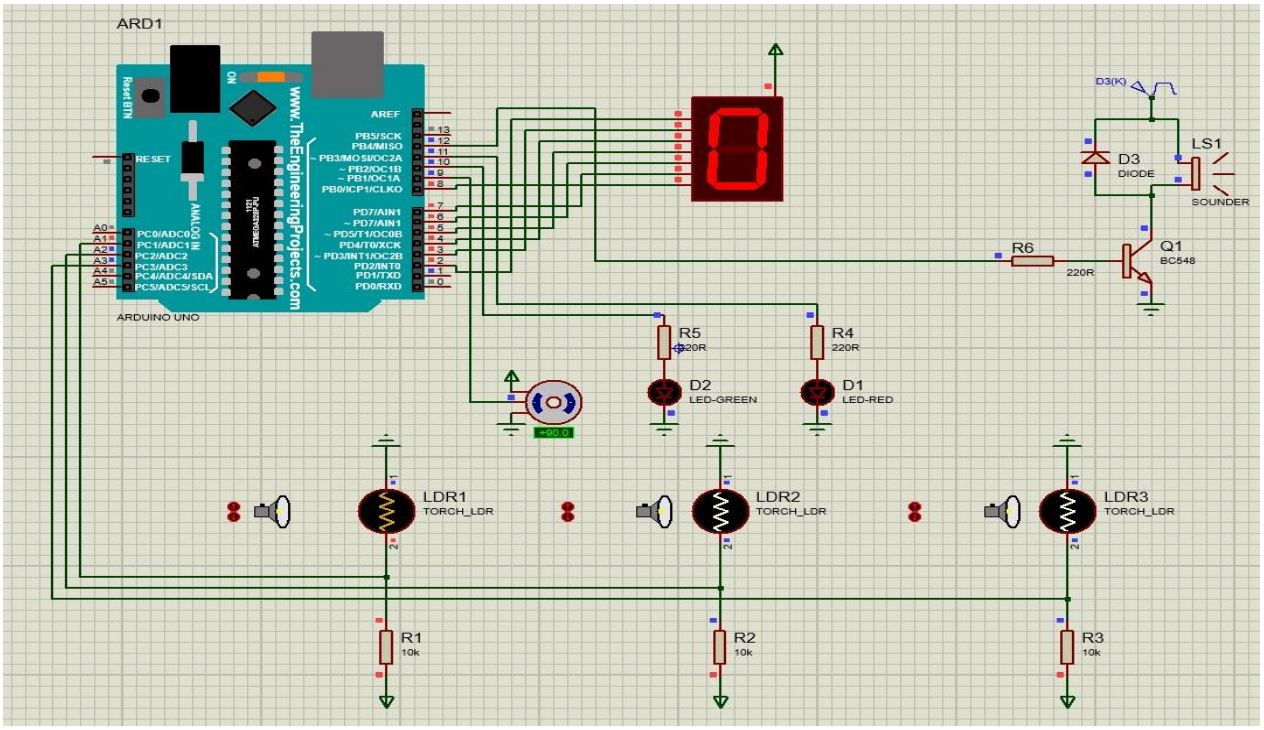
The **2nd stage** “gate closing,



Car passing in front of the 2nd sensor causing light change, red led warning to gate closing, 7segment occurring that available space had been take and alerting buzzer. The **3rd stage** “car leaving”,

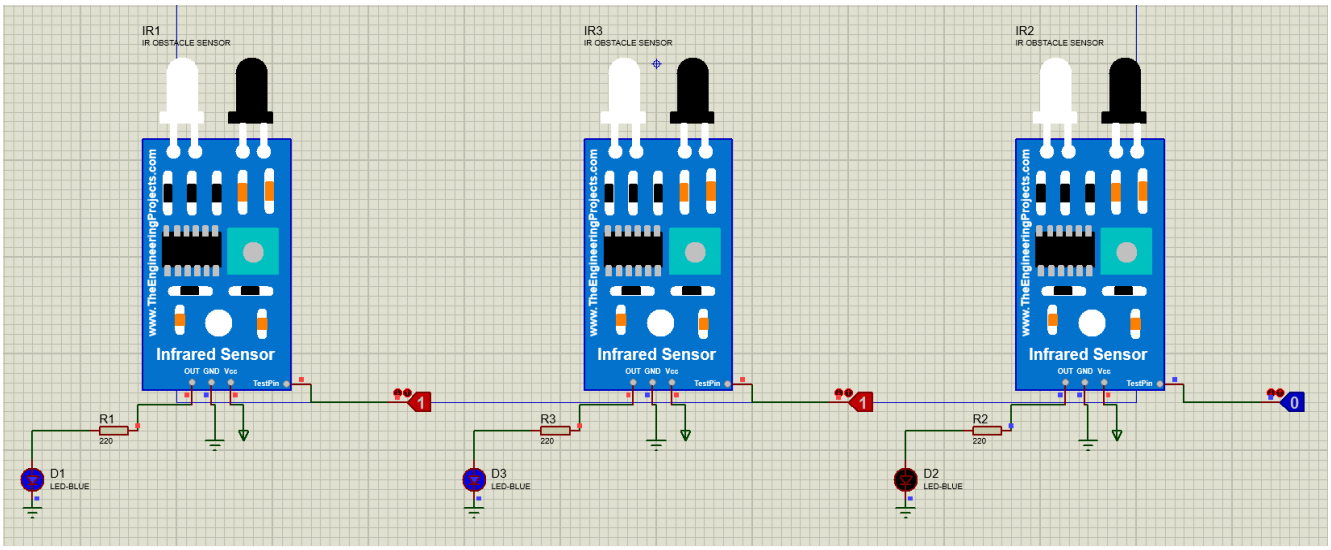


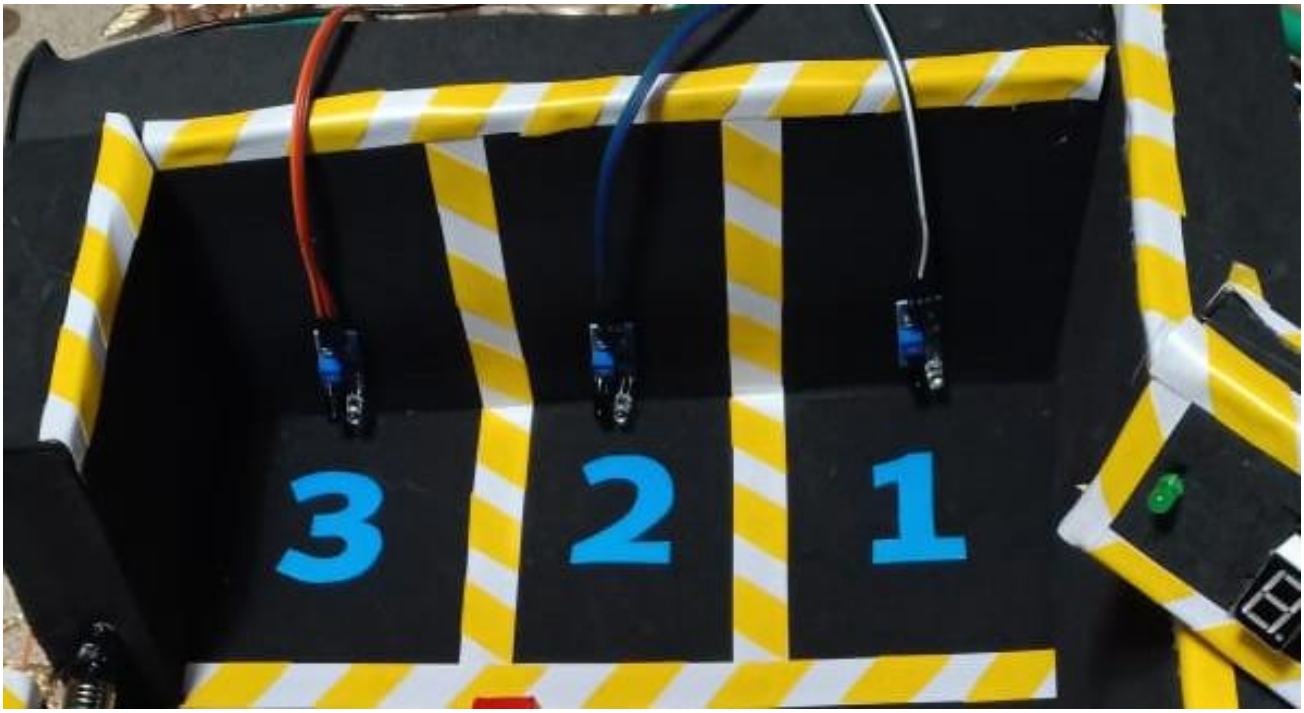
Third sensor main is to confirm that there is an available space and to occur there will be no space clashes however, the parking area space was small and if there are zero space the gate will not open however there is a change of source of the light like the following figure,



The slots localization system

Depends on the digital high or low input from IR sensors using an indication output component (LED), as high input (LED is on) indicating free space and low input (LED is off) indicating that the slot is busy, having a full info about the slots state specifically before entering considering a position reference. That resulting in provide a full scalability aspect to the system.





Hardware implementation (Architecture):

Mainly our hardware components and its implementation based on the previous Proteus simulation as it was considered as out software testbed, so we proceeded to connections as following figures,

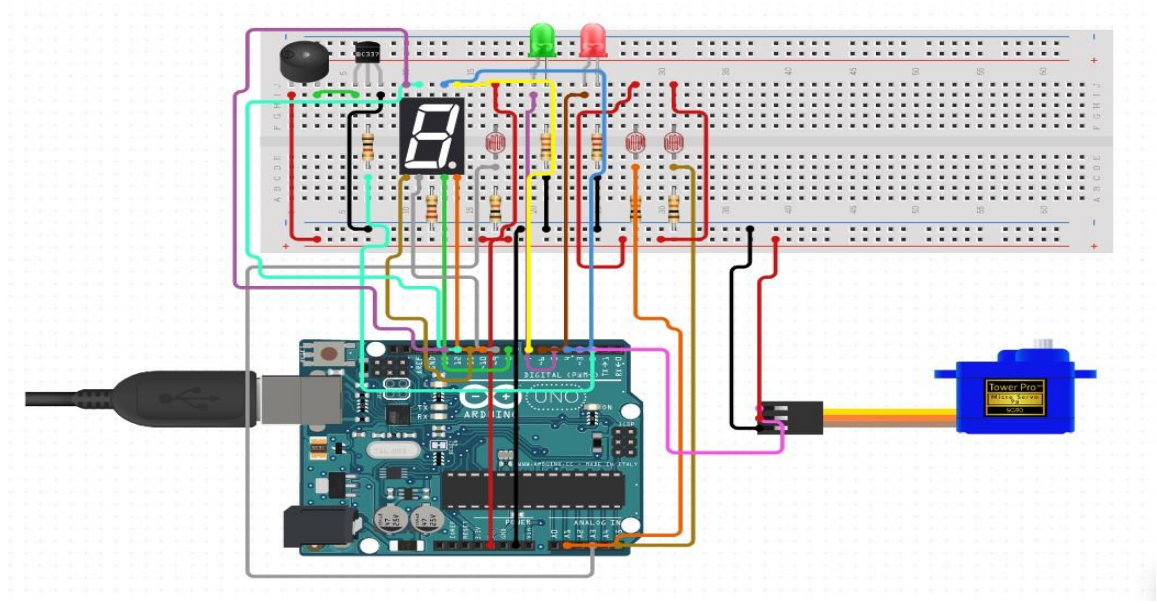


Fig3. Hardware Schematic.

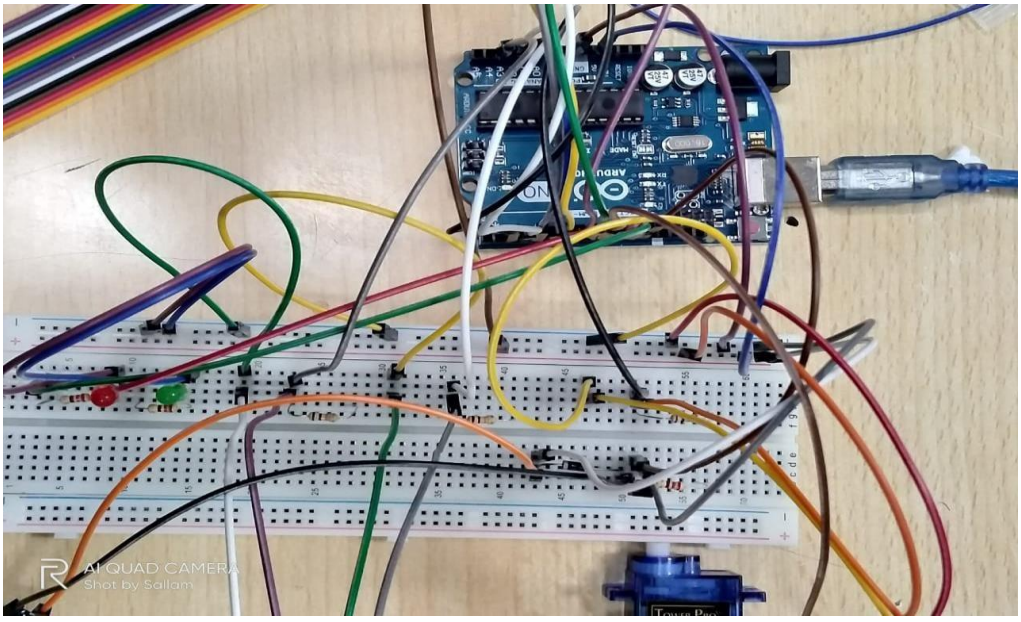
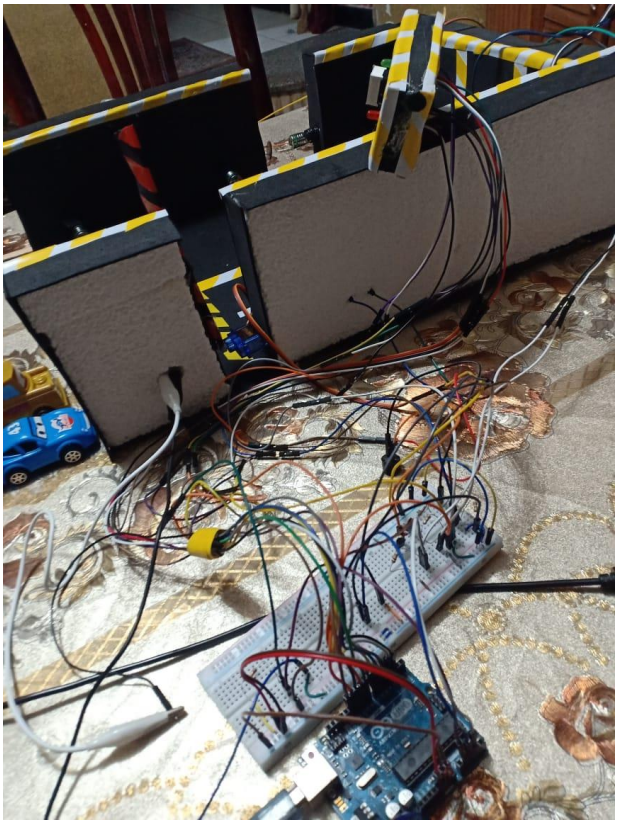


Fig4. Hardware implementation.

Hardware components:

components	Datasheet
Arduino Uno	https://components101.com/microcontrollers/arduino-uno
LDR	https://components101.com/resistors/ldr-datasheet
7 Segment	https://components101.com/displays/7-segment-displaypinout-working-datasheet
Servo Motor SG-90	https://components101.com/motors/servo-motor-basicspinout-datasheet
Raspberry Pi 4	**N/A**
IR Sensor	**N/A**
Raspberry Pi Camera	https://www.futurlec.com/Transistors/BC548.shtml
Active Passive Buzzer	https://components101.com/misc/buzzer-pinout-workingdatasheet

Prototype



Software Architecture and Program Code

The coding algorithm begins with importing important libraries like **servo library** to support our motor. After that, we begin to **define the Arduino pins** according to our components that was illustrated in the previous Proteus simulation and **defining the motor rotation angle initial position** as in "motor.write(180);", specifying that we have six cases, **0, 1, 2, 3, 4, 5.** Where **0** means 1'no available spaces" and **5** is our "maximum capacity", which is available.

PROGRAMME_PARKING_FINAL | Arduino 1.8.15

File Edit Sketch Tools Help

PROGRAMME_PARKING_FINAL

```
#include <Servo.h>
Servo motor;
#include <IRremote.h>
int RECV_PIN = 13;
IRrecv irrecv(RECV_PIN);
decode_results results;

int LDR1 = A1;
int LDR2 = A2;
int LDR3 = A3;
int i;
int LDR1_VAL, LDR2_VAL, LDR3_VAL ;
void setup() {
  irrecv.enableIRIn(); // Start the receiver
  Serial.begin(9600);

   pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(10, OUTPUT);
  pinMode(11, OUTPUT);
  pinMode(12, OUTPUT);

i=5;
  motor.attach(9);
  motor.write(180);
  //place[0];
  affichage(i);
}
```

In the following part we define our 6 cases display based on **transcodeur BCD vers 7 segments,**

Nombre Décimal	Entrées					Sorties						
	E_3	E_2	E_1	E_0	BI	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	1	0
1	0	0	0	1	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	1	0	0	1
4	0	1	0	0	1	0	1	1	0	0	1	1
5	0	1	0	1	1	1	0	1	1	0	1	1

Fig5. transcodeur BCD vers 7 segments table.

```
    switch (i)   {
      case 0:
  digitalWrite(2, LOW);   //0
  digitalWrite(3, LOW);
  digitalWrite(4, LOW);
  digitalWrite(5, LOW);
 digitalWrite(6, LOW);
  digitalWrite(7, LOW);
  digitalWrite(8, HIGH);
        break;


      case 1:
    digitalWrite(2, HIGH);   //1
  digitalWrite(3, LOW);
  digitalWrite(4, LOW);
  digitalWrite(5, HIGH);
 digitalWrite(6, HIGH);
  digitalWrite(7, HIGH);
  digitalWrite(8, HIGH);
        break;


      case 2:
    digitalWrite(2, LOW);   //2
  digitalWrite(3, LOW);
  digitalWrite(4, HIGH);
  digitalWrite(5, LOW);
 digitalWrite(6, LOW);
  digitalWrite(7, HIGH);
  digitalWrite(8, LOW);
        break;
```

```
      case 3:
  digitalWrite(2, LOW);    //3
  digitalWrite(3, LOW);
  digitalWrite(4, LOW);
  digitalWrite(5, LOW);
 digitalWrite(6, HIGH);
  digitalWrite(7, HIGH);
  digitalWrite(8, LOW);
        break;


      case 4:
    digitalWrite(2, HIGH);  //4
  digitalWrite(3, LOW);
  digitalWrite(4, LOW);
  digitalWrite(5, HIGH);
 digitalWrite(6, HIGH);
  digitalWrite(7, LOW);
  digitalWrite(8, LOW);
        break;


      case 5:
  digitalWrite(2, LOW);         //5
  digitalWrite(3, HIGH);
  digitalWrite(4, LOW);
  digitalWrite(5, LOW);
 digitalWrite(6, HIGH);
  digitalWrite(7, LOW);
  digitalWrite(8, LOW);
        break;
  }
}
```

Fig6. 7-segement cases transcodeur BCD algorith display.

In this small section, the code indicates zeroing all external alerting devices, which is the two leds and the buzzer, reads the value from the specified LDR pin to export it in the Arduino.

```
void loop() {
digitalWrite(12, LOW);
digitalWrite(11, LOW);
digitalWrite(10, LOW);
  LDR1_VAL = analogRead(LDR1);
  LDR2_VAL = analogRead(LDR2);
  LDR3_VAL = analogRead(LDR3);
```

This the parts where our alerting devices functions the red, green led and the buzzer functions, as follows in the first figure is were the green led function instantly and the car go to the second process were red led function and returning to the green led function again at the exit. Buzzer functions for a very little amount of time "delay(20);" due to not cause distribution then stops.

```
digitalWrite(10, LOW);
digitalWrite(11, HIGH);
digitalWrite(12, HIGH);
delay(20);
digitalWrite(12, LOW);
motor.write(180);
delay(1000);
        break;
```

```
  i--;
digitalWrite(11, LOW);
digitalWrite(10, HIGH);
digitalWrite(12, HIGH);
delay(20);
digitalWrite(12, LOW);
motor.write(90);
affichage(i);
delay(2000);
  }
        break;
```

Finally, to the LDR Sensors function part of the code, for the **first LDR**, it goes through an if condition algorithm checks if there is a change in value if it take our variable **"i"**, which defined before as the **value 5** and subtract an available space through **"i--"** and opens the gate at **90 degree rotation.**

```
if((LDR1_VAL >800)&& (i> 0))
{
   i--;
   digitalWrite(11, LOW);
   digitalWrite(10, HIGH);
   digitalWrite(12, HIGH);
   delay(200);
   digitalWrite(12, LOW);
   motor.write(90);
   affichage(i);
   delay(2000);
```

Continuing with if condition, it checks if there as value change in the **second LDR** it closes the gate returing the 90 degree rotation to 180 degree position “motor.write(180);”

```
if(LDR2_VAL >300)
{
digitalWrite(10, LOW);
digitalWrite(11, HIGH);
digitalWrite(12, HIGH);
delay(20);
digitalWrite(12, LOW);
motor.write(180);
delay(1000);

  }
```

Finally, with the last conditions when a car passes through the **third LDR** it indicates available space through **“i++;”** all three conditions attached with the alerting devices we discussed previously.

```
if ((LDR3_VAL >500)&& (i<5))
{
    i++;
digitalWrite(11, LOW);
digitalWrite(10, HIGH);
digitalWrite(12, HIGH);
delay(20);
digitalWrite(12, LOW);
affichage(i);
delay(1500);
```

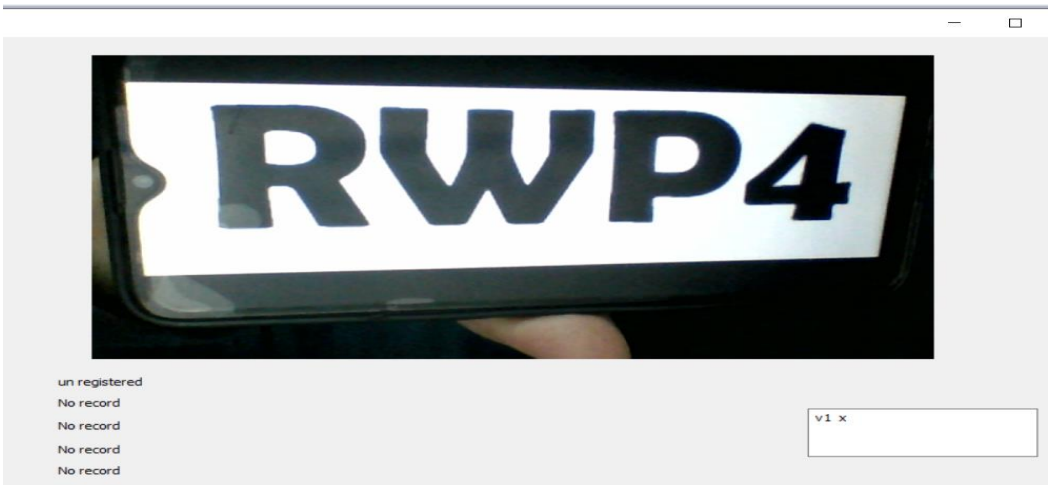
A Comparison between different approaches of OCR

	OCR Library, engine Library	**Code architecture and window interface**	**Serial communication** between controller and operating system
Microsoft Visual Studio 2010 Express	EMGU-CV is cross platform for Open CV Library. Tesseract engine library for Text recognition.	Very flexible code flow with an easy window interface design, doesn't give any compile errors. Unstable functionality in running as it have run time errors.	Based on if conditions to check the availability of the serial port and connection between the code of controller and the .NET Language and based on that it will function.
MATLAB	Self-built library, using extraction built-in function for detection and segmentation methods based on fixed sized characters templates for character matching.	Un-limited feature interface with an ease of access capable of multi-tasking, but very complicated and hard to build code architecture unlike Visual Basic.	As MATLAB have a Support Package for Arduino, you can internally communicate with your Arduino board, serial communication is a simple implantation depending on the USB wireless connection or WI-FI wireless.
Raspberry PI	Open cv library for simple detection and segmentation function for the 1st phase of the ocr and tesseract engine library for the last phase	Based on python, very clear code architecture depending on a major if condition that divides to more two if conditions control the whole flow of the image processing and I/O functions.	There is no serial communication as it's a microcomputer the operating system and the GPIO of the inputs and outputs components work within the same device.

Microsoft visual basic OCR:

```
ElseIf InStr(data, "ABC1") Or InStr(data, " abc1") Or InStr(data, "Abc1") Then
            Label1.Text = "registered"
            Label2.Text = "Afaq"
            Label3.Text = "13401-6756284-2"
            Label4.Text = "HONDA 2004"
            Label5.Text = "552Cbv76676"

            CheckBox1.Checked = True

        ElseIf InStr(data, "RWP4") Or InStr(data, " RWP 4") Or InStr(data, "rwp 4") Then
            Label1.Text = "un registered"
            Label2.Text = "No record"
            Label3.Text = "No record"
            Label4.Text = "No record"
            Label5.Text = "No record"
```

"ABC1" plate is registered in the "if conditions" cases but "RWP4" was registered as not.



Serial communication between Arduino and Visual Basic methodology base on if conditions to check the availability of the serial port and the validity of the condition,

Firstly, it check the validity of the checkbox as a method to control the gate and it's linked as a true to the registered plate,

```
Private Sub CheckBox1_CheckChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Label1.Click
        If CheckBox1.Checked = True Then
            SerialPort1.Open()
            SerialPort1.Write("c")
            SerialPort1.Close()
        Else
            SerialPort1.Open()
            SerialPort1.Write("d")
            SerialPort1.Close()
        End If
    End Sub
```

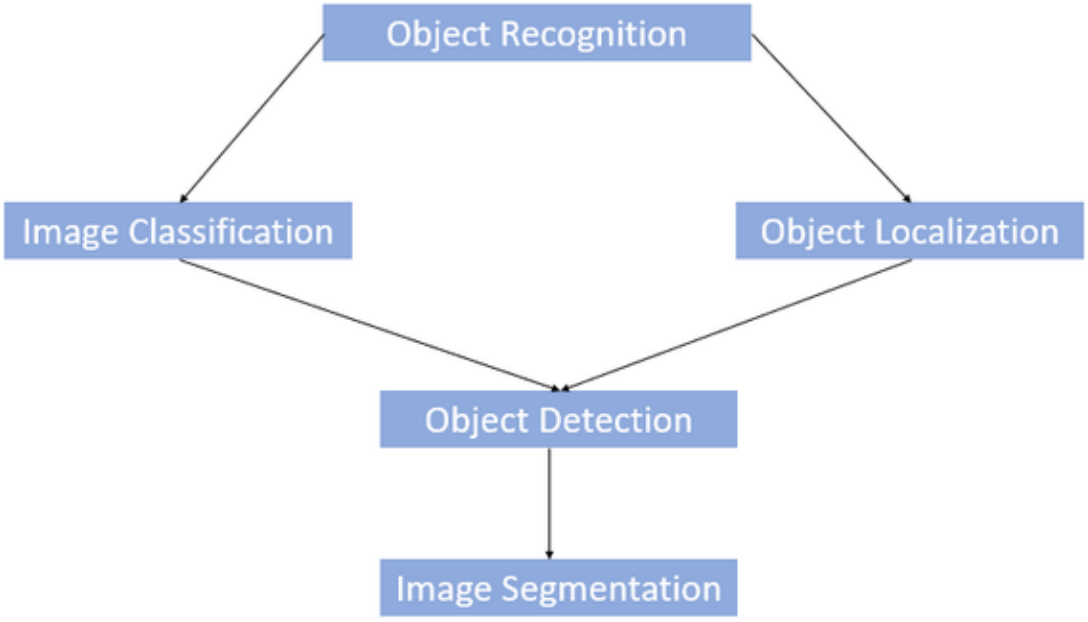
Secondly, the Arduino checks the validity of the the serial port and if true it proceeds to real the serial information and then functions based on the related current condition,

```
if ( Serial.available() > 0 )
{

pos = Serial.read();

if ( pos == 'c' )
{
  motor.write(90);
}
```

```
if ( pos == 'd' )
{
  motor.write(180);
}
}
```

Optical Character Recognition Flow:

It's to be referred that the process of plate recognition in the python implementation is divided considerably into **two Phases**,



The 1st phase is about recognition of **a "plate"** the standard rectangular shape that used in any car, which will be done manually and fully constructed depending on **"OpenCV"** functions of image processing. The flow of it starts directly after the start of the flow, which is the "if condition" controlling the start of capturing the frame that will be processed. **Firstly** starting by using a bilateral filter to remove the unwanted details and canny edge method to preform edge detection,

```
if key == ord("s"):

        gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY) #convert to grey scale

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

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

Secondly preforming **Image Classification** by finding multiple contours and filter the license plate contour by searching for a rectangle shape contour with four sides and a closed figure,

```
for c in cnts:
        peri = cv2.arcLength(c, True)
        approx = cv2.approxPolyDP(c, 0.018 * peri, True)
        if len(approx) == 4:
          screenCnt = approx
          break
```

Object Localization by finding the license plate and mask everything except it,

```
mask = np.zeros(gray.shape,np.uint8)
new_image = cv2.drawContours(mask,[screenCnt],0,255,-1,)
new_image = cv2.bitwise_and(image,image,mask=mask)
```

The **two main process** of the **2nd step in the flow** represents and results of the **3rd step** which is **Object Detection**, as we have achieved to recognize a plate from the whole frame.

Finally, the 4th step in the flow which is **Image Segmentation** as after masking the entire frame except the license plate area, it will be cropped and saved as a new image,

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

The **2nd phase** is mainly about depending **on tesseract library** to read the characters from the license plate image and store the recognized characters in the 'text' variable. There is also an IoT extension, which is responsible for sending the detected plate number to your mail using **Simple Mail Transfer Protocol** (**SMTP**)

Evaluation & Conclusion

The project is implemented and functioning+ mainly as it was planned. The LDR conditions for the light are sometimes not working properly. Moreover, sometimes it has delay issues, and the vehicle has to wait. Moreover, the OCR sometimes read garbage value depending on image orientation and quality, rather than that everything is working perfectly fine with each other. The Arduino is taking the readings from the LDRs, and based on it takes the required actions from changing or keeping the reading on the seven segment. Moreover, it signals to the servomotor, which is the gate mechanism, to open and close depending on these readings.

KPI Name	Description (What?)	Relevance (Why?)	Methodology (How?)
Efficiency of the use of parking spaces	This KPI measures the usage of parking spaces, in order to evaluate which parking areas are more flexible and demanded by clients.	This measurement can help to re-determine the order and the function of the processes of parking, as its target the family's home and the ease of their accessibility.	By access to the parking occupation information regarding the exact time of a day or an hour which the value of measurements highly depends on.
Service management satisfaction	This indicator aim is to evaluate the satisfaction level using the technical Staff that are responsible of the management parking area.	Due to the reliability of the information provided by the sensors deployed, as it is not defined by human error and the variability of it can be controlled.	The methodology depends on discussions with the responsible for reviewing the flow of process, like technicians and parking, security of areas. Who observes and states the degree of achievement improvement by of the technologies in its tasks and the relation with human methods to perform similar tasks.

Table1. Smart parking key performance indicators evaluated.

Future Works

We have learned the validation of smart city concept and its facilities especially the smart parking project and the importance of the concept micro-controllers in our educational project and the development of the concept of controlled gates and mainly controlled civilian projects depended on smart system concepts in modern era and cities development. However, we are satisfied with concept of our project; we may look forward to develop a security dependent system in the process of opening and closing the gates to provide a controlled privacy to the parking area owners. We will look forward to add a LCD display as an advanced display for the indicator instead of constant number of leds to expand a wide range of scalability varying between different floors or even connecting different parking locations and different parking slots. Creating a wider range of database to be capable of knowing info of different parking locations far from you using an IoT concept.

References

[1] Lanza, J., Sánchez, L., Gutiérrez, V., Galache, J., Santana, J., Sotres, P., & Muñoz, L. (2016). Smart City Services over a Future Internet Platform Based on Internet of Things and Cloud: The Smart Parking Case. *Energies*, *9*(9), 719.

[2] Alam, M.(2018). Real-Time Smart Parking Systems Integration in Distributed ITS for Smart Cities. Pisa, Italy: Institute of Information Science and Technologies

[3] Khanna, A.(2016). IoT based Smart Parking System. Pune: Conference: International Conference on Internet of Things and Applications

[4] Smart parking sensors, technologies and applications for open parking lots: a review. (n.d.). https://ietresearch.onlinelibrary.wiley.com/doi/epdf/10.1049/iet-its.2017.0406

[5] Revathi, G. (2012). Smart parking systems and sensors. Dindigul, India: IEEE

[6] Yamani,M.(2009). Car Park System: A Review of Smart Parking System and its Technology. Information Technology Journal

.