Date:31/10/2023

Project Id:Proj_223339_Team_3

Project Title :Smart Parking

Smart parking

Phase 5

Project Objectives:

1. And to save time for the vehicle owner to search the parking slots.

2. The aim of implementing Smart parking system is **to provide secured parking platform given for the vehicle owners.**

3. Another objective of project is to save the fuel, which is consuming while owner searching for the right parking slot.

4. An IoT -based smart parking system is a decent solution for businesses and consumers, providing real-time data on parking space availability, pricing, payments, and more.

5. It can positively impact the environment and traffic.

IOT Sensor Setup:

- Smart Parking sensors are IoT sensors that are installed in parking spaces to detect the presence of a parked vehicle using ultrasonic or radar technology.

- Smart parking sensors are easy to install, battery-powered and operate wirelessly.

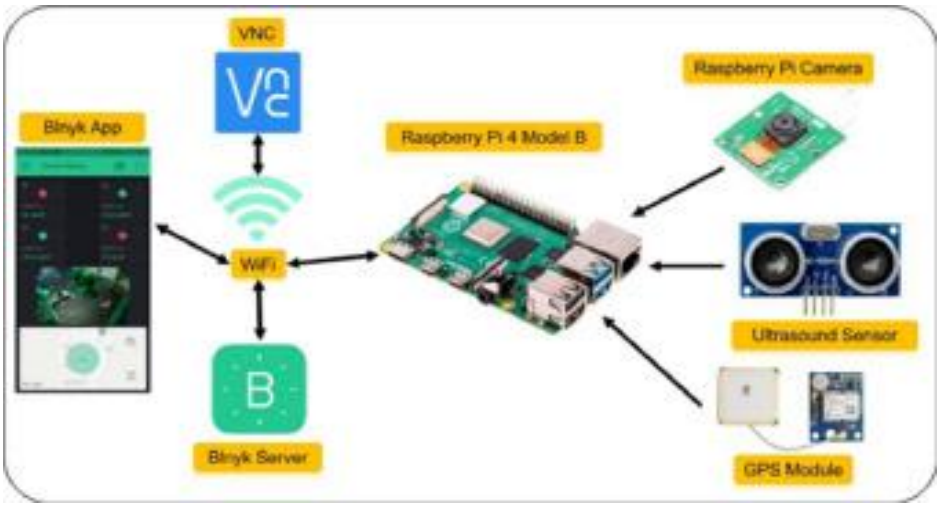
Mobile App Development

- Best mobile app developers focus on creating software through various methods, such as sensors that detect when a parking space is occupied
- The smart parking app shows drivers where available parking spaces are located, and automatic guidance systems can guide drivers directly to an open spot.

Raspberry Pi Integration:

- This system implemented by using infrared sensors in every bay which are then connected to a Raspberry Pi.

- The raspberry Pi transfers all the data to a server, which is open to users using a mobile application.



Smart parking
Implementation code

```
import processing.serial.*;
Serial myPort;
void setup()
{
size(720,600);
```

```
background(0,0,0);
println(Serial.list());
/* I know that the first port in the serial list on my PC
is always com4 the usb port, so I open Serial.list()[1].
Open whatever port is the one you're using.
*/
myPort = new Serial(this, Serial.list()[1], 9600);
 PImage b;
// Images must be in the "data" directory to load correctly
b = loadImage("parking_lot.jpg");
image(b, 0, 0,720,600);
}
float x = 0;
float y = 0;
Int car2 = 0;
float x2 = 0;
float y2 = 200;
int space = 0;
int spotFound = 0;
int delay = 5;
void draw()
{
 PImage b;
```

```
// Images must be in the "data" directory to load correctly
b = loadImage("parking_lot.jpg");
image(b, 0, 0,720,600);
 PImage green_Arrow;
 // Images must be in the "data" directory to load correctly
 green_Arrow = loadImage("green_Arrow.bmp");
 PImage red_Arrow;
 red_Arrow = loadImage("red_Arrow.bmp");
 PImage mustang;
 mustang = loadImage("Mustang.jpg");
 while (myPort.available() > 0) {
 int inByte = myPort.read();
 char SpotAvailable = char(inByte);
 println(SpotAvailable);
 fill(255,255,255);
 rect(10,10,90,150);
 rect(110,10,90,150);
 rect(210,10,90,150);
 rect(310,10,90,150);
 rect(410,10,90,150);
 rect(510,10,90,150);


 if(x2 != 15 && x2 != 115 && x2 != 215 && x2 != 315 && x2 != 415 && x2
!= 515 && spotFound
```

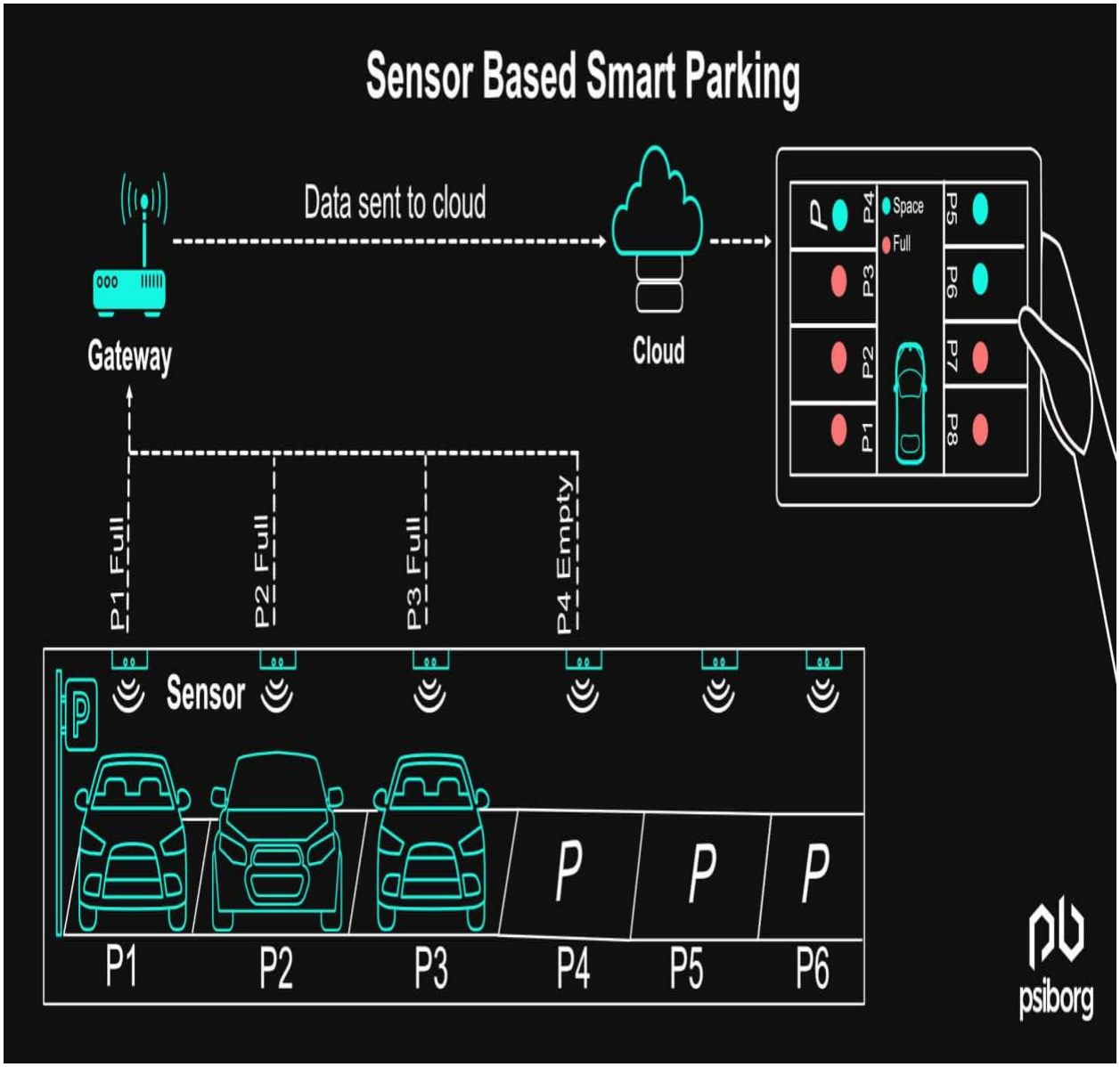
```
== 0){// && SpotAvailable == 0 && spotFound == 0){
 x2 += 5;
 fill(25,255,25);
 image(mustang,x2,200,80,100); //Ask Pushkin how do you set the frame rate or whatever it is
. // pulse whose duration is the time (in microseconds) from the sending
 // of the ping to the reception of its echo off of an object.
 pinMode(pingPin, INPUT);
 duration = pulseIn(pingPin, HIGH);
 // convert the time into a distance
 inches = microsecondsToInches(duration);
 cm = microsecondsToCentimeters(duration);
 if(inches >= 36) {
 Serial .print(0);
 }
 else if(inches <= 36) {
 Serial.print(1);
 }
 else {
 Serial .print(0);
 }
 //Serial.print(inches);
 //Serial.print("in, ");
```

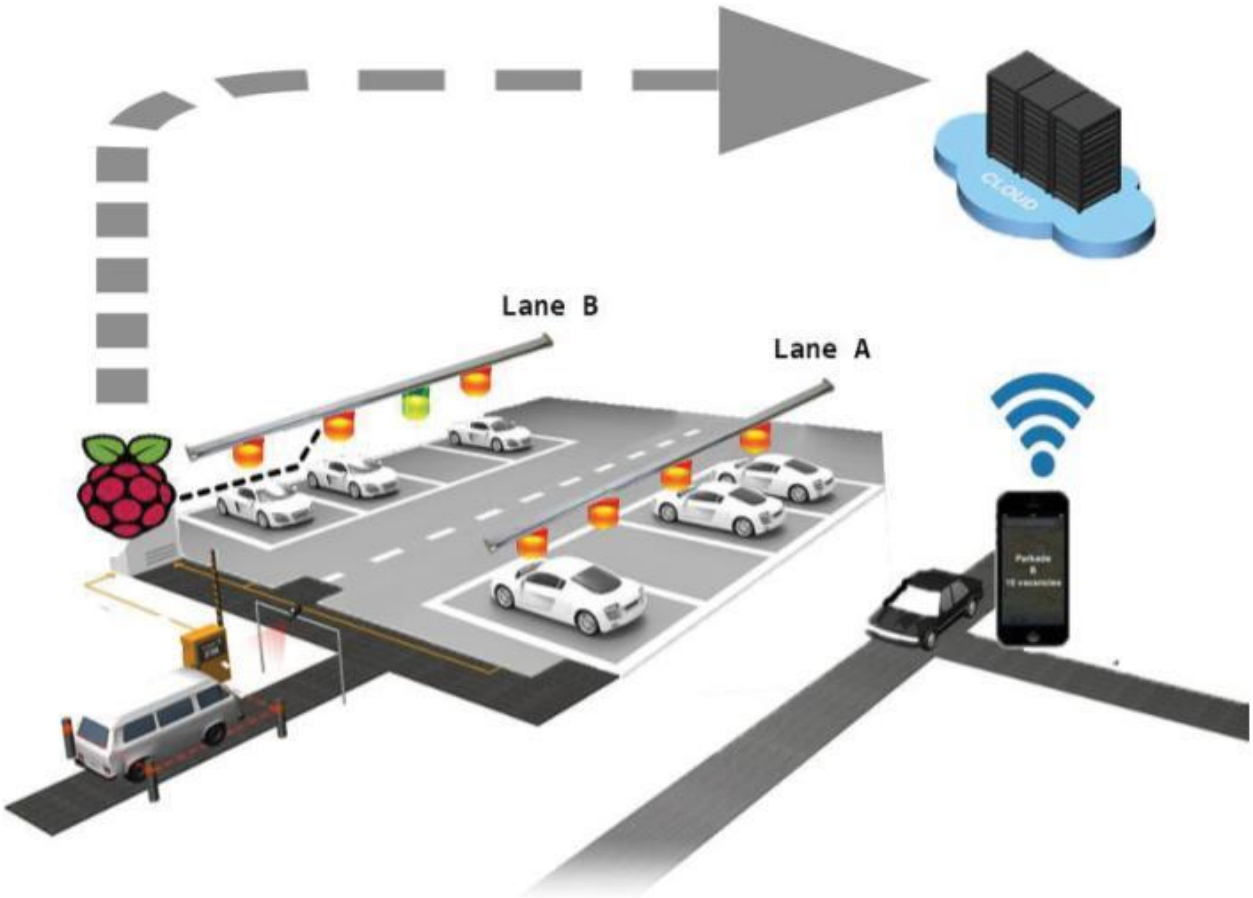
```
 //Serial.print(cm);
 //Serial.print("cm");
 //Serial.print();

 delay(100);
}
long microsecondsToInches(long microseconds)
{
 // According to Parallax's datasheet for the PING))), there are
 // 73.746 microseconds per inch (i.e. sound travels at 1130 feet per
 // second). This gives the distance travelled by the ping, outbound
 // and return, so we divide by 2 to get the distance of the obstacle.
 // See: http://www.parallax.com/dl/docs/prod/acc/28015-PING-
v1.3.pdf
 return microseconds / 74 / 2;
}
long microsecondsToCentimeters(long microseconds)
{
 // The speed of sound is 340 m/s or 29 microseconds per centimeter.
 // The ping travels out and back, so to find the distance of the
 // object we take half of the distance travelled.
 return microseconds / 29 / 2;
}
Else{
```

```
 Fill(25,255,25);
 Image(mustang,x2,y2,80,100);
 //rect(x2,y2,80,100);
 Car2 = 1; Break;
 } }
} Else{
 Println(“else x2 “ + x2);  X2 += 5;
}
 }
}
```

Real Time Parking availability:

- A real-time parking availability system provides real-time information about the available parking spots.
- It combines the data generated by smart navigation systems, cameras, predictive analytics and other transportation systems to deliver accurate information to drivers finding a parking space.


Sensor Based Smart Parking
Data sent to cloud
Gateway
Cloud
P1 Full
P2 Full
P3 Full
P4 Empty
Space
Full
P1
P2
P3
P4
P5
P6
P7
P8
Sensor
P1
P2
P3
P4
P5
P6
psiborg



Smart Parking System

Conclusion:

- As a conclusion, this project will help in reducing the amount of time a driver has to spend around the parking just to find an available spot, reducing the amount of traffic around the parking and also reducing the bad parking around the parking space.