SMART PARKING SYSTEM

*What is smart parking*?

In a nutshell, Smart Parking is a parking solution that can include in-ground Smart Parking sensors, cameras or counting sensors. These devices are usually embedded into parking spots or positioned next to them to detect whether parking bays are free or occupied.  This happens through real-time data collection. The data is then transmitted to a smart parking mobile application or website, which communicates the availability to its users. Some companies also offer other in-app information, such as parking prices and locations. This gives you the possibility to explore every parking option available to you.

Smart Parking and its Smart Parking Sensors can be seen as a part of smart cities. These smart cities are cities that are driven by an IT infrastructure and by using this infrastructure, cities can enhance the quality of life and improve economic development for its inhabitants. Becoming 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 processes, like parking can be optimized.

As a result of using Smart Parking, people who are looking to find a parking spot will find it in the most efficient way possible and companies or municipalities can optimize their parking territories. It also makes cities more livable, safer and less congested.

## Types of smart parking:

## we will divide smart parking systems into 3 main types:

1. Ground sensor technology
2. Counter technology
3. Overhead sensor or camera-based technology

Why is smart parking system important?

It reduces carbon emissions from vehicles by decreasing the congestion and mobility of the vehicles in search of parking. When the drivers move from one place to another for parking, this enhances the individual environmental footprint.



Benefits analysis of Smart Parking Technology:

* Optimized parking.
* Reduced traffic.
* Reduced pollution.
* Enhanced User Experience.
* Integrated Payments and POS.
* Increased Safety.
* Real-Time Data and Trend Insight.
* Decreased Management Costs.

FLOWCHART:

**Update display**

**Increment counter value**

**Going in?**

**Is there a car?**

**Decrement counter value**

**Initializing The Sensors**

NO

YES

NO

ADVANTAGES OF SMART PARKING:

* It helps to reduce traffic congestion by guiding drivers to available parking spots,saving time and fuel.
* It also improves parking management,allowing for better utilization of parking spaces.
* Smart parking can provide real time information on parking availability,making it easier for drivers to find parking quickly.

DISADVANTAGES OF SMART PARKING:

* They require initial investment and maintenance costs,which could be a concern for some.
* Additionally,reliance on technology means that system malfunctions or power outages could disrupt the parking process.
* However ,overall,the benefits of smart parking outweigh these potential drawbacks.

***Data sheet for smart parking:***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ExitCamID** | **EntryCamID** | **CheckedOut Time** | **CheckedIn Time** | **Mobile** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  | 1157006 | 2017.11.17 16:06:09.647 | 2017.11.17 14:56:09.647 | 112233 |
|  | 1157006 | 2017.11.17 16:06:09.647 | 2017.11.17 14:56:09.647 | 112233 |
|  | 1157006 | 2017.11.17 14:56:08.581 | 2017.11.17 14:56:08.581 | 1234567890 |
|  | 1157006 | 2017.11.17 14:56:08.581 | 2017.11.17 14:56:08.581 | 1234567890 |
| 1157006 | 1157006 | 2017.11.17 14:56:05.248 | 2017.11.17 14:56:05.248 | 112233 |
| 1157006 | 1157006 | 2017.11.17 14:56:05.248 | 2017.11.17 14:56:05.248 | 112233 |
|  | 1157006 | 2017.11.17 15:56:08.581 | 2017.11.17 14:56:08.581 | 112233 |
|  | 1157006 | 2017.11.17 15:56:08.581 | 2017.11.17 14:56:08.581 | 112233 |
|  | 1157006 | 2017.11.17 15:56:08.581 | 2017.11.17 14:56:08.581 | 112233 |
|  | 1157006 | 2017.11.17 15:56:08.581 | 2017.11.17 14:56:08.581 | 112233 |
|  |  |  |  |  |
|  |  |  |  |  |
| 2 | 1 | 14-02-2018 13:03 | 2018.02.14 13:03:36:596 | 8807550969 |
|  |  |  |  |  |
|  |  |  |  |  |
| 1157006 | 1157006 | 2017.11.17 14:56:05.248 | 2017.11.17 14:56:05.248 | 112233 |
| 2 | 1 | 14-02-2018 13:03 | 2018.02.14 13:03:36.596 | 112233 |
| 2 | 1 | 14-02-2018 13:03 | 2018.02.14 13:03:36.596 | 112233 |
| 2 | 1 | 14-02-2018 13:03 | 2018.02.14 13:03:36.596 | 8807550969 |
| 2 | 1 | 14-02-2018 13:03 | 2018.02.14 13:03:36.596 | 9985878021 |
|  | 1157006 | 2017.11.17 16:06:09.647 | 2017.11.17 14:56:09.647 | 112233 |
|  | 1157006 | 2017.11.17 14:56:08.581 | 2017.11.17 14:56:08.581 | 1234567890 |
| 2 | 1 | 14-02-2018 13:03 | 2018.02.14 13:03:36.596 | 9985878021 |
| 2 | 1 | 14-02-2018 13:03 | 2018.02.14 13:03:36.596 | 8807550969 |
|  | 1157006 | 2017.11.17 16:06:09.647 | 2017.11.17 14:56:09.647 | 112233 |
|  | 1157006 | 2017.11.17 14:56:08.581 | 2017.11.17 14:56:08.581 | 112233 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Created\_At** | **Unique\_ID** | **Receipt\_No** | **Amount** | **Payment\_ID** | **Car number** |
| 2017.11.1717:36:08.718 | 6.02202E+12 | rcpt\_AP 31 HY 1235 | 20 | payid\_AP 31 HY 1235 | AP31HY1235 |
| 2017.11.1717:36:08.718 | 6.02202E+12 | rcpt\_AP 31 HY 1235 | 20 | payid\_AP 31 HY 1235 | AP31HY1235 |
| 2017.11.1717:36:08.718 | 6.02202E+12 | rcpt\_AP 31 HY 1235 | 10 | payid\_AP 31 HY 1235 | AP31HY1235 |
| 2017.11.1717:36:08.718 | 6.02202E+12 | rcpt\_06022018095808 | 150 | payid\_06022018095803 | AP53UP3135 |
| 2017.11.1717:36:08.581 | 6.02202E+12 | rcpt\_06022018095802 | 50 | payid\_06022018095802 | AP53UP3141 |
| 2017.11.1717:36:08.718 | 6.02202E+12 | rcpt\_AP 31 HY 1235 | 10 | payid\_AP 31 HY 1235 | AP31HY1235 |
| 2017.11.1717:36:08.581 | 6.02202E+12 | rcpt\_06022018095802 | 50 | payid\_06022018095802 | AP53UP3131 |
| 2017.11.1717:36:08.718 | 6.02202E+12 | rcpt\_AP 31 HY 1235 | 20 | payid\_AP 31 HY 1235 | AP31HY1235 |
| 2017.11.1717:36:08.718 | 6.02202E+12 | rcpt\_AP 31 HY 1235 | 20 | payid\_AP 31 HY 1235 | AP31HY1235 |
| 2017.11.1717:36:08.718 | 6.02202E+12 | 6022018095902 | 5 | 6.02202E+12 | AP30CL4567 |
| 2017.11.1717:36:08.718 | 6.02202E+12 | 6022018095832 | 10 | 6.02202E+12 | IND00UP1122 |
| 2017.11.1717:36:08.718 | 6.02202E+12 | 6022018095832 | 0 | 6.02202E+12 | IND00UP1122 |
| 2017.11.1717:36:08.718 | 6.02202E+12 | 6022018095802 | 9 | 6.02202E+12 | AP53UP3131 |
| 2017.11.1717:36:08.718 | 6.02202E+12 | 6022018095802 | 4 | 6.02202E+12 | AP53UP3131 |
| 2017.11.1717:36:08.718 | 1.4022E+11 | 14022018115138 | 30 | 1.4022E+13 | AP |
| 2018.02.19 17:56:08.718 | 1.4022E+11 | 14022018115138 | 10 | 1.4022E+13 | AP |
| 2018.02.19 17:56:08.718 | 6.02202E+12 | 6022018095902 | 5 | 6.02202E+12 | AP30CL4567 |
| 2018.02.19 17:56:08.718 | 6.02202E+12 | 6022018095902 | 5 | 6.02202E+12 | AP30CL4567 |
| 2018.02.19 17:56:08.718 | 6.02202E+12 | 6022018095902 | 1 | 6.02202E+12 | AP30CL4567 |
| 2018.02.19 17:56:08.718 | 6.02202E+12 | 6022018095902 | 1 | 6.02202E+12 | AP30CL4567 |
| 2018.02.19 17:56:08.718 | 6.02202E+12 | rcpt\_AP 31 HY 1235 | 20 | payid\_AP 31 HY 1235 | AP31HY1235 |
| 2018.02.19 17:56:08.718 | 6.02202E+12 | rcpt\_AP 31 HY 1235 | 20 | payid\_AP 31 HY 1235 | AP31HY1235 |
| 2018.02.19 17:56:08.718 | 1.40222E+11 | 1.4022E+13 | 10 | 1.4022E+13 | AP35AF4234 |
| 2018.02.19 17:56:08.718 | 1.40222E+11 | 6.02202E+12 | 10 | 6.02202E+12 | AP30CL4567 |
| 2018.02.19 17:56:08.718 | 1.40222E+11 | 6.02202E+12 | 2 | 6.02202E+12 | AP30CL4567 |
| 2018.02.19 17:56:08.718 | 6.02202E+12 | 1.4022E+13 | 10 | 1.4022E+13 | AP |
| 2018.02.19 17:56:08.718 | 6.02202E+12 | 1.4022E+13 | 4 | 1.4022E+13 | IND00UP1122 |
| 2017.11.1717:36:08.718 | 6.02202E+12 | 1.4022E+13 | 10 | 1.4022E+13 | IND00UP1122 |
| 2017.11.1717:36:08.718 | 6.02202E+12 | 1.4022E+13 | 50 | 1.4022E+13 | AP35AF4234 |
| 2017.11.1717:36:08.718 | 6.02202E+12 | 1.4022E+13 | 15 | 1.4022E+13 | AP35AF4234 |
| 2017.11.1717:36:08.718 | 6.02202E+12 | 6.02202E+12 | 20 | 6.02202E+12 | IND00UP1122 |
| 2017.11.1717:36:08.718 | 6.02202E+12 | 6.02202E+12 | 4 | 6.02202E+12 | AP53UP3131 |
| 2017.11.1717:36:08.718 | 6.02202E+12 | 1.4022E+13 | 15 | 1.4022E+13 | AP35AF4234 |
| 2017.11.1717:36:08.718 | 6.02202E+12 | 1.4022E+13 | 10 | 1.4022E+13 | AP35AF4234 |
| 2017.11.1717:36:08.718 | 1.4022E+11 | 6.02202E+12 | 0 | 6.02202E+12 | IND00UP1122 |
| 2017.11.1717:36:08.718 | 1.40222E+11 | 6.02202E+12 | 2 | 6.02202E+12 | AP30CL4567 |

***Program:***

**if** sys.version\_info[0] == 2:

input = raw\_input

**class** ParkingLot:

**def** init (self):

self.capacity = 0

self.slotid = 0

self.numOfOccupiedSlots = 0

**def** createParkingLot(self,capacity):

self.slots = [-1] \* capacity self.capacity = capacity **return** self.capacity

**def** getEmptySlot(self):

**for** i **in** range(len(self.slots)):

**if** self.slots[i] == -1:

# return i

**def** park(self,regno,color):

**if** self.numOfOccupiedSlots < self.capacity:

slotid = self.getEmptySlot() self.slots[slotid] = Vehicle.Car(regno,color) self.slotid = self.slotid+1

self.numOfOccupiedSlots = self.numOfOccupiedSlots + 1

**return** slotid+1

# else:

**return** -1

**def** leave(self,slotid):

**if** self.numOfOccupiedSlots > 0 **and** self.slots[slotid-1] != -1: self.slots[slotid-1] = -1

self.numOfOccupiedSlots = self.numOfOccupiedSlots - 1

**return** True

# else:

**return** False

**def** status(self):

**print**("Slot No.**\t**Registration No.**\t**Colour")

**for** i **in** range(len(self.slots)):

**if** self.slots[i] != -1:

**print**(str(i+1) + "**\t\t**" +str(self.slots[i].regno) + "**\t\t**" + str(self.slots[i].color))

# else:

**continue**

**def** getRegNoFromColor(self,color): regnos = []

**for** i **in** self.slots:

**if** i == -1:

# continue

**if** i.color == color: regnos.append(i.regno)

**return** regnos

**def** getSlotNoFromRegNo(self,regno):

**for** i **in** range(len(self.slots)):

**if** self.slots[i].regno == regno:

**return** i+1

# else:

**continue return** -1

**def** getSlotNoFromColor(self,color): slotnos

**for** i **in** range(len(self.slots)):

**if** self.slots[i] == -1:

# continue

**if** self.slots[i].color == color:

slotnos.append(str(i+1))

**return** slotnos

**def** show(self,line):

**if** line.startswith('create\_parking\_lot'):

n = int(line.split(' ')[1]) res = self.createParkingLot(n)

**print**('Created a parking lot with '+str(res)+' slots')

**elif** line.startswith('park'):

regno = line.split(' ')[1]

color = line.split(' ')[2] res = self.park(regno,color) **if** res == -1:

**print**("Sorry, parking lot is full")

# else:

**print**('Allocated slot number: '+str(res))

**elif** line.startswith('leave'):

leave\_slotid = int(line.split(' ')[1]) status = self.leave(leave\_slotid)

**if** status:

**print**('Slot number '+str(leave\_slotid)+' is free')

**elif** line.startswith('status'):

self.status()

**elif** line.startswith('registration\_numbers\_for\_cars\_with\_colour'): color = line.split(' ')[1]

regnos = self.getRegNoFromColor(color)

**print**(', '.join(regnos))

**elif** line.startswith('slot\_numbers\_for\_cars\_with\_colour'):

color = line.split(' ')[1]

slotnos = self.getSlotNoFromColor(color)

**print**(', '.join(slotnos))

**elif** line.startswith('slot\_number\_for\_registration\_number'):

regno = line.split(' ')[1]

slotno = self.getSlotNoFromRegNo(regno)

**if** slotno == -1:

**print**("Not found")

# else:

**print**(slotno)

**elif** line.startswith('exit'):

exit(0)

**def** main():

parkinglot = ParkingLot()

parser = argparse.ArgumentParser()

parser.add\_argument('-f', action="store", required=False, dest='src\_file', help="Input File") args = parser.parse\_args()

**if** args.src\_file:

**with** open(args.src\_file) **as** f:

**for** line **in** f:

line = line.rstrip('**\n**') parkinglot.show(line)

# else:

**while** True:

line = input("$ ") parkinglot.show(line)

**if** name == ' main ': main()

***HTML Code:***

<!DOCTYPE html>

<html>

<head>

<title>Smart Parking System</title>

</head>

<body>

<h1>Smart Parking System</h1>

<form id="parking-form">

<label for="command">Enter Command:</label>

<input type="text" id="command" name="command" placeholder="Command (e.g., park ABC123 Red)">

<input type="button" value="Submit" onclick="executeCommand()">

</form>

<h2>Output:</h2>

<div id="output"></div>

<script>

function executeCommand() {

var commandInput = document.getElementById("command");

var command = commandInput.value;

commandInput.value = '';

// Implement the logic to send the command to your parking lot system

// You may use AJAX to send the command to the server and receive the response.

// For simplicity, we'll just display the command as output for now.

var outputDiv = document.getElementById("output");

outputDiv.innerHTML += command + "<br>";

}

</script>

</body>

</html>

***Explaination for smart parking system:***

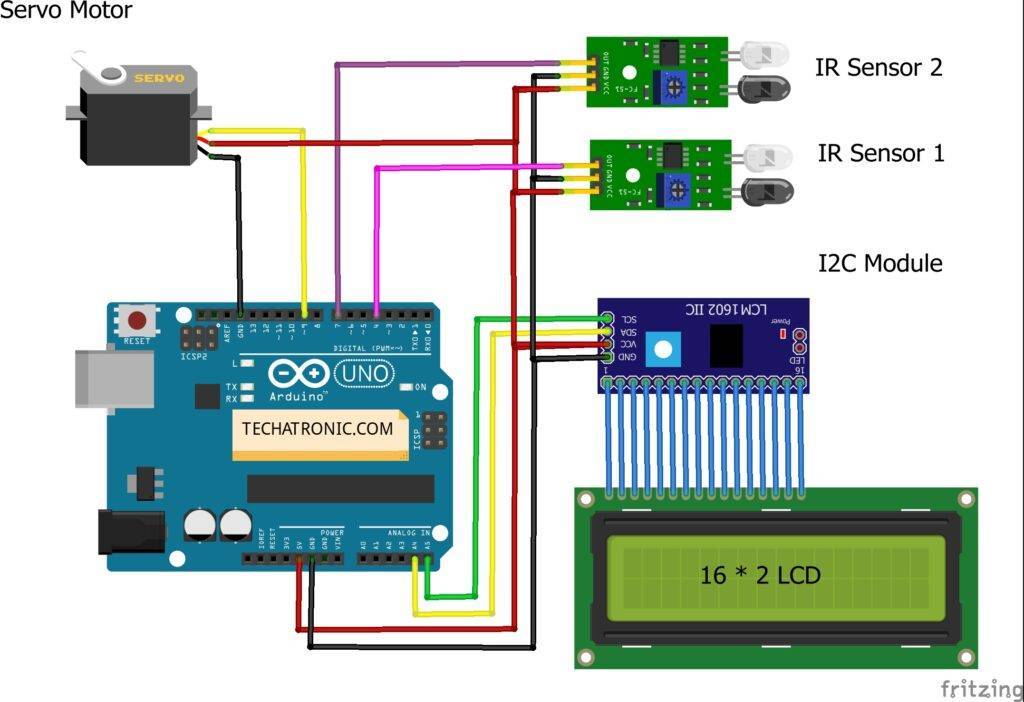


Car parking is a major issues in modern congested cities of today. There simply are too many vehicles on the road and not enough parking space. This has led to the need for efficient parking management systems. Thus we demonstrate the use of IOT based parking management system that allows for efficient parking space utilization using IOT technology. To demonstrate the concept we use IR sensors for sensing parking slot occupancy along with a dc motors to simulate as gate opener motors. We now use a wifi modem for internet connectivity and an AVR microcontroller for operating the system. We use IOTGecko for online connectivity and IOT management GUI design. The system detects if parking slots are occupied using IR sensors. Also it uses IR technology to sense if a vehicle has arrived on gate for automated gate opening. The system reads the number of parking slots available and updates data with the cloud server to allow for checking parking slot availability online. This allows users to check for available parking spaces online from anywhere and avail hassle free parking. Thus the system solves the parking issue for cities and get users an efficient IOT based parking management system.

**Hardware Specifications**

* Atmega Microcontroller
* I R Transmitter Receiver
* DC Motor
* LCD Display
* Crystal Oscillator
* Resistors
* Capacitors
* Transistors
* Cables and Connectors
* Diodes
* PCB and Breadboards
* LED
* Transformer/Adapter
* Push Buttons
* Switch
* IC
* IC Sockets
* **Software Specifications**
* Arduino Compiler
* MC Programming Language: C
* IOTGecko

## automatic car parking Circuit Diagram



Before starting please check the [address of the I2C module](https://techatronic.com/i2c-scanner/) that you are using and modify the code accordingly.

Then connect the 5 volts pin of the Arduino with the VCC pin of the I2C module, the red wire of the servo motor, and the VCC pin of both the IR sensors.

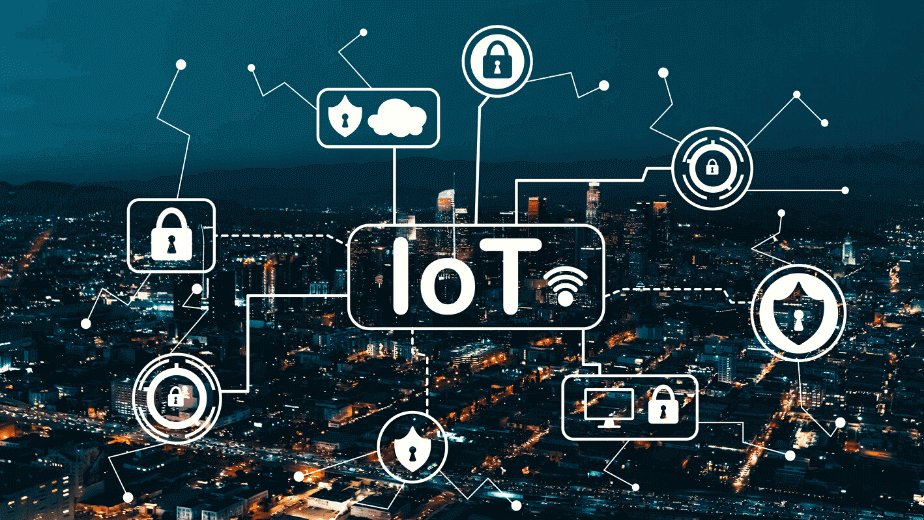
### **Access Control and Management**

Parking lot owners can use access control and management systems to allow only authorized users to enter a parking area. Here’s how it works.

The car parking system using IoT takes a user authorization mechanism through a mobile app or license plate scanning. At the same time, the controller on the barrier or gate may allow or refuse drivers to park their cars according to the set parameters.

WebbyLab’s project [Propuskator is an example of such an access control and management system](https://webbylab.com/cases/acs-remote-controller/). It uses a controller connected to the gate or barrier and paired with a [2Smart Cloud mobile application](https://webbylab.com/cases/2smart/) to grant or restrict access to the territore.

***IOT Device***



## **What is IoT?**

IoT is an umbrella term that refers to the billions of physical objects or “things” connected to the Internet, all collecting and exchanging data with other devices and systems over the Internet.

While IoT has been in existence since the 90s, recent advances in a number of different technologies have made it more practical, such as:

* Access to affordable and reliable sensors
* Increase in the availability of [cloud computing](https://www.simplilearn.com/tutorials/cloud-computing-tutorial/what-is-cloud-computing) platforms
* Advances in [machine learning and AI technologies](https://www.simplilearn.com/how-to-build-career-in-ai-and-machine-learning-article).

IoT devices can range from small ordinary household cooking appliances to sophisticated industrial tools. Each IoT component has a Unique Identifier (UID) and they can also transmit data without the assistance of humans.

## **Types of IoT**

The networking, communication and connectivity protocols depend largely on the specific [IoT application](https://www.simplilearn.com/iot-applications-article) deployed. Just as there are many different IoT devices, there are many types of IoT applications based on their usage. Here are some of the most common ones:

* Consumer IoT - Primarily for everyday use. Eg: home appliances, voice assistance, and light fixtures.
* Commercial IoT - Primarily [used in the healthcare](https://www.simplilearn.com/using-iot-applications-to-manage-the-spread-of-covid-19-article) and transport industries. Eg: smart pacemakers and monitoring systems.
* Military Things (IoMT) - Primarily used for the application of IoT technologies in the military field. Eg: surveillance robots and human-wearable biometrics for combat.
* [Industrial Internet of Things](https://www.simplilearn.com/ways-businesses-use-iot-article) (IIoT) - Primarily used with industrial applications, such as in the manufacturing and energy sectors. Eg: Digital control systems, smart agriculture and industrial big data.
* Infrastructure IoT - Primarily used for connectivity in smart cities.  Eg: infrastructure sensors and management systems.