Smart Parking System

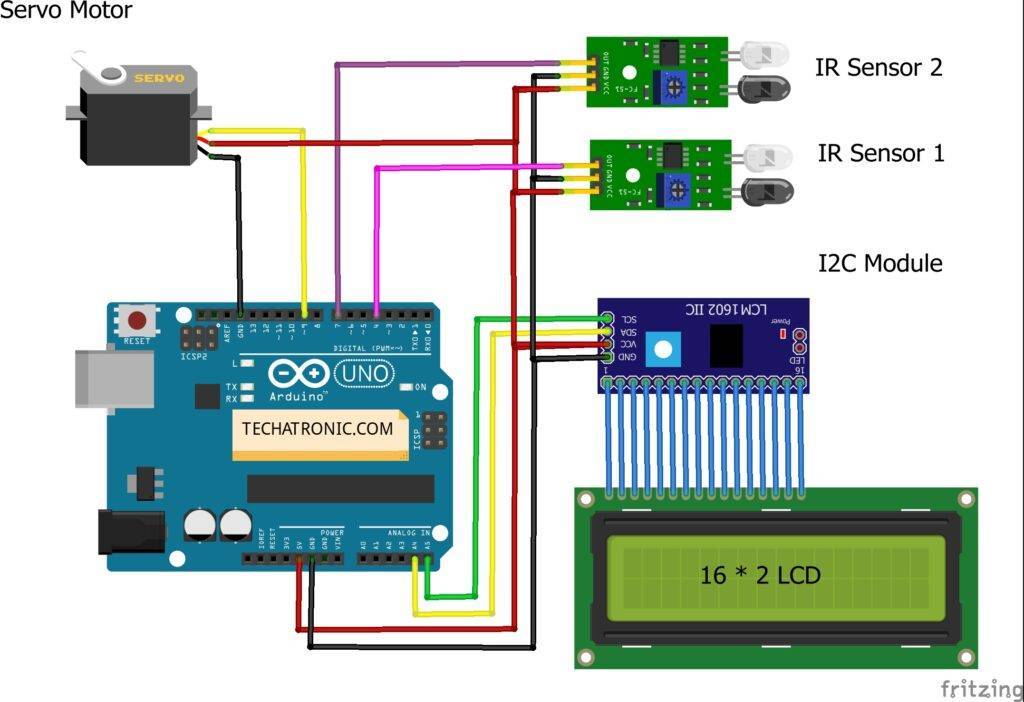


Car parking is a major issues in modern congested cities of today. There simply are too many vehicles on the road and not enough parking space. This has led to the need for efficient parking management systems. Thus we demonstrate the use of IOT based parking management system that allows for efficient 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

**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()