SMART PARKING

**Domain – IOT**

**Abstract :**

Smart Car Parking IOT: Revolutionizing Urban Parking. Our IOT-based system streamlines urban parking.It uses sensors and cameras to relay real-time parking availability, making it easy for users to find and reserve spots via mobile apps. Payment processing is automated, reducing wait times. This innovative system improves urban mobility and sustainability, reducing congestion and enhancing convenience.

**Problem Statement :**

The traditional urban parking management system is inefficient and often leads to congestion, wasted time, and environmental concerns. This project aims to address these issues by implementing an IOT- based Smart Car Parking system that offers real-time parking space availability information, reservation options, and automated payment processing, ultimately improving urban mobility and sustainability.

**Design Thinking :**

Design Thinking is a human-centered approach to problem-solving and innovation that places a strong emphasis on empathy, creativity, and iteration. It typically involves the following stages:

1. Empathize: Understand the needs and perspectives of the people you are designing for. This stage often involves research, interviews, and observation to gain deep insights into usersexperiences and challenges.

2. Define: Clearly articulate the problem or challenge based on the insights gained during the empathy stage. This involves synthesizing information to create a specific problem statement.

3. Ideate: Generate a wide range of creative ideas to address the defined problem. Encourage brainstorming and free thinking to come up with innovative solutions.

4. Prototype: Create tangible representations of your ideas. These can be rough sketches, physical models, or digital prototypes. The goal is to quickly test and iterate on different concepts.

5. Test: Gather feedback by testing your prototypes with users. This helps refine and improve the solutions. This stage often involves multiple iterations and adjustments.

6. Implement: Once a refined solution is identified through testing, it is ready for implementation. This may involve developing a final product, service, or process.

7. Iterate: Design Thinking is an iterative process, and it’s important to revisit and refine solutions based on real-world feedback and changing needs. Design Thinking is a versatile methodology that can be applied to various fields, from product design to service improvement and social innovation. It encourages collaboration, creativity, and a deep understanding of the end-users’ needs and perspectives to develop solutions that truly meet their requirements.

**INNOVATION OF SMART PARKING:**

* IOT is often used in smart parking systems to connect various

devices and sensors, allowing for real-time data exchange and

efficient management. This connectivity enables better monitoring,

control, and automation of parking operations.

* An IOT based smart parking system, also known as a connected parkingsystem, is a centralized management system that allows drivers to use a smartphone app to search for and reserve a parking spot.

**PROJECT PLANNING:**

The traditional urban parking management system is inefficient and often leads to congestion, wasted time, and environmental concerns. This project aims to address these issues by implementing an IOT-based Smart Car Parking system that offers real- time parking space availability information, reservation options, and automated payment processing, ultimately improving urban mobility and sustainability.

**HARDWARE COMPONENTS:**

* Choose the necessary IOT hardware components are Parking Sensors,Communication Devices , Servers and Databases , Network Infrastructure ,Display Devicesand  Power Supply.
* Select sensors based on the parameters you to plan the parking slot to park thevehicles in that place.

**SOFTWARE COMPONENTS:**

Choose the necessary IOT Software components are Sensor Data ManagementSoftware, Mobile Application, Data Analytics and Reporting,  Integration APIs,  Security and

* **Authentication.**
* **Sensors**
* **Mobile Applications**
* **Data Analytics**
* **Guidance Systems**
* **Signage and display**

**Controlling parking systems using sensors :**

**1. Sensor Installation**

**2. Data Collection**

**3. Occupancy Monitoring**

**4. Information Display**

**5. Navigation and Reservation**

**6. Automated Enforcement**

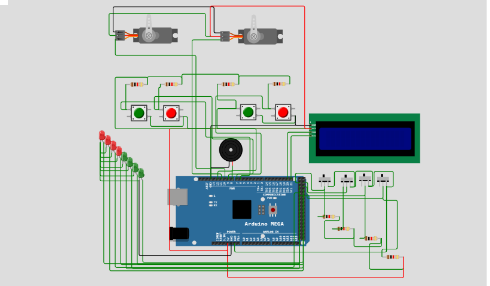
**Components Required** **:**

|  |  |  |
| --- | --- | --- |
| **S. No** | **Components** | **Quantity** |
| 1. | Arduino or Microcontroller | 1 |
| 2. | ICD 1602 | 1 |
| 3. | Servo Motors | 2 |
| 4. | Register | 8 |
| 5. | Buzzer | 1 |
| 6. | Push Button | 4 |
| 7. | LED Bulb | 8 |
| 8. | Slide Switch | 4 |
| 9. | Cables and Connectors | As Needed |

**Tools Requried:**

|  |  |  |
| --- | --- | --- |
| **S. No** | **Tools** | **Example** |
| 1. | Arduino IDE | Ardunio IDE |
| 2. | CAD Software (optional) | Autodesk Eagle |
| 3. | Version Control System | GitHub, GitHub Desktop |
| 4. | Project Management Software | Thinkspeak |
| 5. | Simulation Software (optional) | Wokwi |
| 6. | Data Analysis Tool | Excel |
| 7. | Communication Tools | Microsoft Teams, Discord, Zoom |

**Circuit Diagram of Smart Parking:**

****

**Working principle:**

The working principle of a typical car parking system involves several

components and technologies:

* Entry and Exit Gates: The system usually has entry and exit gates

controlled by sensors or attendants. These gates regulate the flow of vehicles in and out of the parking facility.

* Sensors: Sensors, such as ultrasonic or infrared sensors, are used to detect the presence of vehicles in parking spaces. These sensors send signals to a central control system.
* Central Control System: This system processes the data from the sensors and manages the availability of parking spaces. It directs incoming vehicles to open spots and keeps track of occupied spaces.
* Remote Management: parking systems are remotely monitored and managed through software, enabling operators to track occupancy, revenue, and perform maintenance tasks.

**Arduino Code:**

#include <Servo.h>

#include <LiquidCrystal\_I2C.h>

uint8\_t indoor1 = 6;

uint8\_t indoor2 = 8;

uint8\_t outdoor1 = 7;

uint8\_t outdoor2 = 9;

uint8\_t buzzer = 12;

uint8\_t sncar[]={22,23,24,25};

uint8\_t outputgreen[]={26,27,28,29};

uint8\_t outputred []={30,31,32,33};

int count1 = 0;

boolean enablecount = false;

byte datax = 0b0000;

byte indexcar = 0x00;

LiquidCrystal\_I2C lcd(0x27,16,2);

Servo myservo1;

Servo myservo2;

void setup() {

Serial.begin(9600);

Serial.println(&quot;program start&quot;);

for(uint8\_t index = 0; index&lt;4; index+=1){

pinMode(sncar[index],INPUT);

pinMode(outputgreen[index],OUTPUT);

pinMode(outputred[index],OUTPUT);

digitalWrite(sncar[index],LOW);

digitalWrite(outputgreen[index],LOW);

digitalWrite(outputred[index],LOW);

}

pinMode(indoor1,INPUT);

pinMode(indoor2,INPUT);

pinMode(outdoor1,INPUT);

pinMode(outdoor2,INPUT);

pinMode(buzzer,OUTPUT);

myservo1.attach(10);

myservo2.attach(11);

lcd.init(); // initialize the lcd

lcd.backlight();

lcd.clear();

delay(100);

}

void loop() {

/\* static unsigned long timer1 = millis();

if((millis()-timer1)&gt;=1000){

timer1 = millis();

Serial.print(&quot;timeq1 = &quot;);

Serial.print(timer1/1000);

Serial.println(&quot; sec&quot;);

}\*/

if(enablecount==true){

// Serial.println(&quot;yeet&quot;);

count1+=1;

if(count1&gt;=100){

myservo1.write(90);

enablecount = false;

Serial.println(&quot;ok&quot;);

count1=0;

}

}

for(byte i =0; i&lt;4;i+=1){

bitWrite(datax,i,digitalRead(sncar[i]));

}

for(byte i =0; i&lt;4;i+=1){

bitWrite(datax,i,digitalRead(sncar[i]));

//Serial.print(bitRead(datax,i),BIN);

indexcar = indexcar+!bitRead(datax,i);

if(i==3){

//Serial.println(&quot;&quot;);

lcd.setCursor(0,0);

lcd.print(&quot;Total : &quot;+String(indexcar)+&quot; car&quot;);

lcd.setCursor(0,1);

if(indexcar==4){

lcd.print(String(!bitRead(datax,0))+&quot; &quot;+String(!bitRead(datax,1))+&quot;

&quot;+String(!bitRead(datax,2))+&quot; &quot;+String(!bitRead(datax,3))+&quot; &quot;+&quot;FULL

CAR&quot;);

}

else{

lcd.print(String(!bitRead(datax,0))+&quot; &quot;+String(!bitRead(datax,1))+&quot;

&quot;+String(!bitRead(datax,2))+&quot; &quot;+String(!bitRead(datax,3))+&quot;

&quot;+&quot;Emptycar&quot;);

}

indexcar = 0;

}

}

if(digitalRead(indoor1)==0 &amp;&amp; datax!= 0b0000){

myservo1.write(180);

count1=0;

}

if(digitalRead(outdoor1)==0 || datax == 0b0000 ){

enablecount = true;

}

if(digitalRead(indoor2)==0){

myservo2.write(180);

}

if(digitalRead(outdoor2)==0){

myservo2.write(90);

}

for(uint8\_t index = 0; index&lt;4; index+=1){

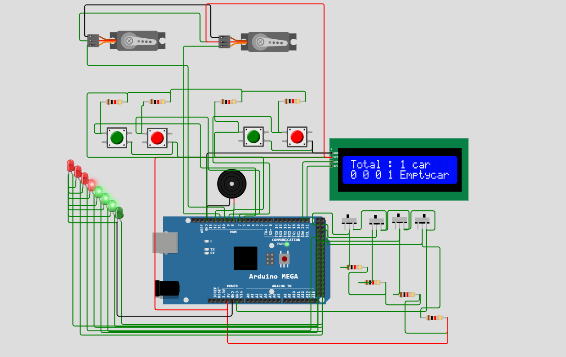
digitalWrite(outputgreen[index],!digitalRead(sncar[index]));

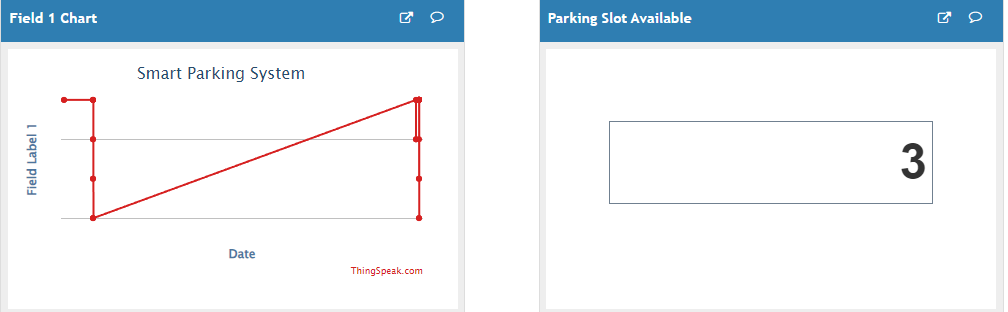
digitalWrite(outputred[index],digitalRead(sncar[index]));

}

delay(10);

}

**Simulation:**

****