GNANAMANI COLLEGE OF TECHNOLOGY(Pachal,Namakkal.)

DEPARTMENT OF BIO MEDICAL

ENGINEERING

(Third Year)

**Title**: **Smart parking**

**Team Members**: **B.Kavinila(620821121048)**

**M.Nandhini(620821121072)**

**V.Abinayasri(620821121006)**

**K.Abinaya(620821121004)**

**T.Gowriamirtha(620821121028)**

**BY**

**Gowri amirtha. T**

SMART PARKING

**PROBLEM :**

Difficulty in Accurate parking Space Detection-One common challenge in smart parkingSystems is accurately detecting whether a parking space is occupied or vacind in accutare detection can lead to confusion and insufficiency

**KEYWORDS:**

1)Internet of things

2)Smart city

3) Messaging protocols

4)Standerdization

5)Introperability

6)Protuct life cycle management

**INTRODUCTION:**

**\* .** An IOT based parking system is a vechile parking management system to is the search for the vacant parking spot in a parking lot through a smart phone.

\*. The system utilizes various sensors and microcontrollers with internet capability for decting parked vechiles and to uptate the data in real time on internet .

**DESIGN OF SMART PARKING:**

**\*.** As Mentioned above the proposed smart parking lot circuit will be equipped

With several sensors

\*. Inexpensive microcontrollers and wifi module using which a car/any vechile won car check if there is a vacant space in a parking lot using his /her phone or tabelet or even on computer.

**DATA FUSION:**

Compain data from multiple sensor to increase accuracy for example,you can use both infrared and ultra sonicsensors in tandem by cross referencing their

Data can reduce false readings .

**MACHINE LEARNING:**

**\***.Implement machine learning algorithms to analog sensor data

\*.Machine learning can help in fine tuning occupancy detection by accounting for various factors like sesor noise lighting conditions and environmentalChanges.

**REAL TIME UPDATES:**

\*. Connect the sensor data to a central system or a mobile app that provides

Realtime updates to users

\*.Indicating available parking space this ensures that drivers are directed to the nearest vacant spot

**MAINTENANCE AND CALIBRATION:**

\*.Regularly maintain and calibration the sensors to ensure their accuracy over time . this includes checking for sensor malfunctions or obstructions

\*.By addressing the accuracy of parking space detection through IOT Sensors

And data processing.you can improve the efficiency and user experience of your

SMART PARKING PROJECT.

**SOFTWARE :**

Software used for arduino based smart parking system project is arduino IDE

**ARDUINO IDE:**

This is the primary software for programming Arduino boards you can downloade it from the official arduino web site.

PHASE-2

INNOVATION

**1. \*\*IoT Connectivity:\*\***

- Utilize IoT modules (such as ESP8266 or ESP32) with Arduino to connect the system to the internet.

- Enable bidirectional communication, allowing the system to send data to the cloud and receive commands or updates.

**2. \*\*Soil Moisture Sensing:\*\***

- Implement soil moisture sensors in key locations to measure the moisture content of the soil.

- Use capacitive soil moisture sensors for accurate readings, and calibrate them to specific soil types.

**3. \*\*Data Transmission:\*\***

- Establish a secure connection to an IoT platform (like ThingSpeak, Blynk, or AWS IoT) to transmit real-time data.

- Ensure data encryption for privacy and security.

**4. \*\*Cloud-Based Analytics:\*\***

- Implement cloud-based analytics to process and analyze the collected data.

- Utilize machine learning algorithms to predict future soil moisture levels based on historical data, weather forecasts, and other relevant parameters.

**5. \*\*Mobile Application:\*\***

- Develop a user-friendly mobile app for farmers or users to monitor and control the system remotely.

- Include features such as real-time soil moisture levels, historical data graphs, and the ability to adjust irrigation settings.

**6. \*\*Automated Irrigation Control:\*\***

- Implement an automated irrigation system that adjusts water flow based on real-time sensor data.

- Include features like scheduling, threshold alerts, and emergency shutdown in case of sensor malfunctions or extreme conditions.

**7. \*\*Energy Efficiency:\*\***

- Design the system to be energy-efficient by using low-power components and optimizing the communication protocols.

8**. \*\*Scalability:\*\***

- Ensure that the system is scalable, allowing users to expand the coverage area or add more sensors as needed.

**9. \*\*Weather Integration:\*\***

- Integrate weather APIs to incorporate forecast data into the decision-making process.

- Adjust irrigation schedules based on upcoming weather conditions to avoid unnecessary watering during or after rainfall.

**10. \*\*Community and Data Sharing:\*\***

- Allow for community-based data sharing where users can contribute anonymized data for broader analysis.

- Promote a collaborative approach to water management, especially in regions facing water scarcity

**PHASE-3**

**DEVELOPMENT-1**

**1. \*\*Hardware Setup\*\*:**

- Install sensors (e.g., ultrasonic, infrared, or magnetic) in parking spaces to detect vehicle presence.

- Deploy cameras for visual monitoring and license plate recognition.

- Set up a microcontroller or IoT device (e.g., Raspberry Pi, Arduino, or specialized hardware) to connect and manage the sensors and cameras.

**2. \*\*Connectivity\*\*:**

- Establish a reliable internet connection, either through Wi-Fi, cellular, or a dedicated network for your IoT devices.

- Ensure proper security measures for data transmission.

**3. \*\*Data Collection\*\*:**

- Collect data from the sensors and cameras, such as occupancy status and license plate information.

- Send this data to a central server or cloud platform for processing and analysis.

**4. \*\*Data Processing and Storage\*\*:**

- Process the incoming data to determine parking space occupancy.

- Store historical data for trend analysis and reporting.

**5. \*\*User Interface\*\*:**

- Develop a user-friendly mobile app or web interface for users to check parking availability, reserve spots, and pay for parking.

**6. \*\*Notifications\*\*:**

- Implement real-time notifications for users, such as alerts when a parking spot becomes available or when a reservation is about to expire.

**7. \*\*Payment Integration\*\*:**

- Integrate payment gateways for users to pay for parking using various methods, such as credit cards, mobile wallets, or prepaid accounts.

**8. \*\*Security\*\*:**

- Implement security measures to protect the IoT devices and data, including encryption, access control, and device authentication.

**9. \*\*Analytics\*\*:**

- Use data analytics to gather insights on parking space utilization and optimize parking management.

**10. \*\*Maintenance and Monitoring\*\*:**

- Set up monitoring tools to track the health and status of IoT devices.

- Regularly maintain and calibrate sensors and cameras to ensure accuracy.

**11. \*\*Scalability\*\*:**

- Design the system to be scalable, allowing for easy expansion to more parking spaces or locations.

**12. \*\*Regulatory Compliance\*\*:**

- Ensure compliance with local regulations and privacy laws, especially regarding data collection and user privacy.

**13. \*\*Testing and Deployment\*\*:**

- Thoroughly test the system in a controlled environment before deploying it in a real-world setting.

**14. \*\*Feedback and Improvement\*\*:**

- Continuously gather user feedback to improve the system's features and performance.

**15. \*\*Integration with Smart City Initiatives\*\*:**

- Explore opportunities to integrate your smart parking system with broader smart city initiatives, like traffic management and sustainability efforts.

**PHASE-4**

**DEVELOPMENT-2**

**1. \*\*Feature Engineering\*\*:**

   - \*\*Sensor Data\*\*: Collect data from sensors (e.g., cameras, ultrasonic sensors) to monitor parking spaces.

   - \*\*Time Features\*\*: Incorporate time-related features such as day of the week, time of day, and holidays to capture patterns.

   - \*\*Weather Data\*\*: Include weather conditions (e.g., rain, snow) as they can affect parking behavior.

   - \*\*Historical Data\*\*: Use historical occupancy data to establish trends and patterns.

   - \*\*External Data\*\*: Integrate traffic data and event schedules that may impact parking demand.

**2. \*\*Model Training\*\*:**

   - \*\*Classification Models\*\*: Train classification models like Random Forest, SVM, or Neural Networks to predict parking space occupancy (occupied or vacant).

   - \*\*Regression Models\*\*: If you want to predict the duration of parking, use regression models.

   - \*\*Time Series Analysis\*\*: For time-dependent patterns, use techniques like ARIMA or LSTM.

   - \*\*Reinforcement Learning\*\*: RL can be used to optimize parking strategies over time, e.g., pricing or guidance to users.

   - \*\*Anomaly Detection\*\*: Detect anomalies or unauthorized parking using anomaly detection models.

**3. \*\*Evaluation\*\*:**

   - \*\*Metrics\*\*: Use metrics like accuracy, precision, recall, F1-score for classification models, or RMSE for regression models.

   - \*\*Cross-Validation\*\*: Employ cross-validation to assess model generalization.

   - \*\*Real-time Testing\*\*: Test the model in a real-time environment, comparing predictions with actual occupancy.

   - \*\*User Feedback\*\*: Collect feedback from users to gauge system satisfaction and effectiveness.

   - \*\*Optimization\*\*: Measure the efficiency of the parking system by evaluating whether it reduces congestion and optimizes space usage.

**PHASE-5**

**Project documentation and submission**

1. \*\*Sensor Simulation\*\*: Simulate parking sensors to detect vehicle presence. For simplicity, we’ll use a list to represent parking spaces where ‘0’ means vacant, and ‘1’ means occupied.

```python

Parking\_spaces = [0, 0, 0, 0, 0] # Sample parking spaces

```

1. \*\*User Interface\*\*: Create a basic user interface to display parking information and allow users to interact with the system.

```python

Def display\_parking\_status():

For I, space in enumerate(parking\_spaces):

Status = “Occupied” if space == 1 else “Vacant”

Print(f”Space {I + 1}: {status}”)

Def user\_interface():

While True:

Print(“\nSmart Parking System”)

Display\_parking\_status()

Action = input(“Enter ‘P’ to park, ‘L’ to leave, or ‘Q’ to quit: “).upper()

If action == ‘P’:

# Implement parking logic

Elif action == ‘L’:

# Implement leaving logic

Elif action == ‘Q’:

Break

Else:

Print(“Invalid input. Please try again.”)

If \_\_name\_\_ == “\_\_main\_\_”:

User\_interface()

```

1. \*\*Parking Logic\*\*: Implement parking and leaving logic to update the parking spaces.

```python

Def park\_vehicle():

For I, space in enumerate(parking\_spaces):

If space == 0:

Parking\_spaces[i] = 1

Print(f”Vehicle parked in space {I + 1}.”)

Break

Else:

Print(“No vacant spaces available.”)

Def leave\_space():

Space\_number = int(input(“Enter the space number to leave: “))

If 1 <= space\_number <= len(parking\_spaces) and parking\_spaces[space\_number – 1] == 1:

Parking\_spaces[space\_number – 1] = 0

Print(f”Space {space\_number} is now vacant.”)

Else:

Print(“Invalid space number or space is already vacant.”)

```

**Conclusion**:

In conclusion, a smart parking system offers an innovative and efficient solution for urban parking challenges. By harnessing technology such as sensors, data analysis, and user-friendly interfaces, it enhances the overall parking experience, reduces congestion, and contributes to a more sustainable and convenient urban environment.