**SMART PARKING**

**PHASE 5 - SUBMISSION DOCUMENT**

**ABSTRACT:**

Monitoring ridership, tracking locations, and predicting arrival times in public transportation vehicles have become essential for enhancing the efficiency and user experience of urban transit systems**.**

**PROJECT OBJECTIVES:**

* Real-time parking space monitoring
* Mobile app integration
* Efficent parking guidance

**REAL-TIME PARKING SPACE MONITORING:**

Real-time parking space monitoring involves the use of technology to track and manage the availability of parking spaces in real-time. Here's an overview of how this is achieved.sensors and cameras,datacollection,datatransmission,dataprocessing, information display,payment integration,alerts and notification, analytics and optimization.

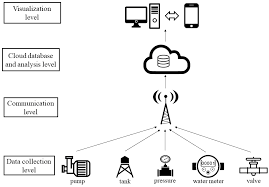
**MOBILE APP INTEGRATION:**

Mobile app integration is the process of connecting a mobile application with other systems, services, or platforms to enhance its functionality, features, and user experience. Here are some common aspects of mobile app integration.API integration,payment gateway integration.

**EFFICIENT PARKING GUIDANCE:**

Efficient parking guidance systems are designed to help drivers find available parking spaces quickly and reduce congestion in parking facilities. Here are key elements and technologies used to achieve efficient parking guidance.

**DESIGING THE IOT SENSOR SYSTEM**:



Designing an IoT sensor system involves careful planning and consideration of various components and aspects to ensure it meets the intended objectives. Here's an outline of the steps involved:

**1**. **\*Define Objectives\*:**

- Determine the purpose of the IoT sensor system. What data do you need to collect? What problems will it address?

**2. \*Select Sensors\*:**

- Choose appropriate sensors based on your objectives. Consider factors like sensor type (e.g., temperature, humidity, motion), accuracy, and communication protocols (e.g., Wi-Fi, Bluetooth, LoRa, Zigbee).

**3. \*Power Source\*:**

- Decide on the power source. It can be battery-powered, solar-powered, or connected to a reliable power grid.

**Real-time transit information platform:**

Developing a real-time transit information platform requires careful planning, integration of various technologies, and a user-centered approach. Here are the steps involved in creating such a platform:

**1. \*Define Objectives\*:**

- Determine the primary goals of your transit information platform. What are you trying to achieve? Improved user experience, reduced wait times, increased efficiency?

**2. \*Data Sources\*:**

- Identify the data sources needed, such as GPS data from vehicles, traffic data, schedules, and occupancy information. Establish partnerships with transit agencies to access their data.

**3. \*Data Integration**\*:

- Create data pipelines to collect, integrate, and process real-time data from various sources. Use ETL (Extract, Transform, Load) processes to ensure data quality.

**Integrating APPROCH:**

To use a Raspberry Pi to collect data from sensors and update a mobile app, you'll need to set up a system that involves sensor data acquisition, data processing, and communication between the Raspberry Pi and the mobile app. Here's a step-by-step guide:

**1. \*Select Sensors**\*:

- Choose the appropriate sensors for your project. Depending on your requirements, this could include temperature sensors, motion sensors, humidity sensors, etc.

**2. \*Connect Sensors to Raspberry Pi\*:**

- Connect the selected sensors to the GPIO pins of the Raspberry Pi. Make sure to follow the wiring and power requirements for each sensor.

**3. \*Install Necessary Libraries\*:**

- Install the required libraries and drivers for the sensors on the Raspberry Pi. This may involve using Python libraries, C libraries, or other programming languages depending on the sensor.

**\*Algorithm for Smart Parking:\***

STEP1:\*Initialization\*:

- Initialize the system, including the parking lot layout, sensor data, and available parking spaces.

STEP2: \*Sensor Data Collection\*:

- Continuously collect data from sensors installed at each parking space to detect occupancy (occupied or vacant).

STEP3:\*User Request\*:

- When a user requests parking, either through a mobile app or on-site device:

- Identify the user's location and desired parking duration.

- Check the availability of nearby parking spaces.

STEP4: \*Search for Available Space\*:

- Iterate through the parking spaces to find the nearest available parking spot that meets the user's criteria.

- Consider proximity to the user's current location.

- Ensure the parking space is vacant.

- Check if it meets the user's desired duration.

STEP5:\*Reserve Space\*:

- Reserve the selected parking space for the user, marking it as temporarily occupied. This prevents others from occupying it during the reservation period.

STEP6: \*User Confirmation\*:

- Notify the user of the reserved parking space, including its location, availability, and any relevant details.

STEP7: \*Parking Guidance\*:

- Provide turn-by-turn directions to the reserved parking spot if the user is on-site.

STEP8: \*User Arrival\*:

- Confirm the user's arrival at the parking spot using location data, a QR code scan, or other methods.

STEP9: \*Occupancy Update\*:

- Mark the parking space as occupied in the system.

STEP10:\*Billing and Payment\*:

- Calculate the parking fee based on the duration and rate. Charge the user through a payment gateway integrated into the system.

STEP11: \*Real-time Updates\*:

- Continuously update the parking availability information on mobile apps and digital displays to inform other users.

STEP12: \*End of Parking\*:

- Monitor the user's parking duration, and when it expires, mark the parking space as vacant again.

STEP13:\*Feedback and Reviews\*:

- Allow users to provide feedback or reviews about their parking experience, contributing to system improvement.

STEP14:\*System Optimization\*:

- Use historical data to optimize the allocation of parking spaces, predict high-demand periods, and improve the overall efficiency of the system.

STEP15:\*Maintenance and Alerts\*:

- Implement alerts for system maintenance, sensor malfunction, or other issues that require attention.

**PROGRAM :**

# Define the parking lot layout (for demonstration purposes)

total\_spaces = 20

parking\_spaces = [False] \* total\_spaces # False represents a vacant space

def display\_parking\_lot():

print("Parking Lot Status:")

for i, space in enumerate(parking\_spaces, 1):

status = "Occupied" if space else "Vacant"

print(f"Space {i}: {status}")

def find\_available\_space():

for i, space in enumerate(parking\_spaces):

if not space:

return i

return -1

def reserve\_parking\_space(space\_number):

if parking\_spaces[space\_number]:

print("Space is already occupied.")

return False

else:

parking\_spaces[space\_number] = True

print(f"Space {space\_number + 1} reserved.")

return True

def release\_parking\_space(space\_number):

parking\_spaces[space\_number] = False

print(f"Space {space\_number + 1} released.")

# Simulation

while True:

display\_parking\_lot()

action = input("Select an action (1: Reserve, 2: Release, 0: Exit): ")

if action == '0':

break

elif action == '1':

space\_number = find\_available\_space()

if space\_number != -1:

reserve\_parking\_space(space\_number)

else:

print("No available parking spaces.")

elif action == '2':

space\_number = int(input("Enter the space number to release: ")) - 1

if 0 <= space\_number < total\_spaces:

release\_parking\_space(space\_number)

else:

print("Invalid space number.")

print("Smart Parking System Exiting.")

GIT UP LINK:

<https://github.com/prabhakararaj20/prabhakararaj.git>

**SENSOR:**

Monitoring ridership, tracking vehicle locations, and predicting arrival times in public transportation can be greatly improved through the use of sensor technology. Here's how it can work:

**1. \*GPS and Location Sensors\*:** Equipping public transportation vehicles with GPS and location sensors allows real-time tracking of their positions. This information can be shared with passengers through mobile apps and used to optimize routes.

**2. \*Passenger Count Sensors\*:** Install sensors at vehicle entrances to count passengers boarding and alighting. This data can help estimate occupancy and inform waiting passengers about the capacity.

**3. \*Smart Payment Systems\*:** Implement contactless payment systems (e.g., RFID cards, mobile apps) that not only improve efficiency but also track entry and exit times of passengers.

**4. \*Traffic and Weather Sensors\*:** Integrate traffic and weather sensors to predict delays and adjust schedules in real time.

**5. \*Predictive Analytics\*:** Utilize machine learning algorithms to analyze historical and real-time data, predicting arrival times more accurately based on current conditions.

**6. \*Communication Systems\*:** Provide passengers with real-time information through displays at stops, mobile apps, or even text message alerts, regarding delays, expected arrival times, and route changes.

**7. \*Maintenance Sensors\*:** Use sensors to monitor the health of the vehicle itself, enabling predictive maintenance to reduce breakdowns.

**8. \*Energy Efficiency Sensors\*:** Monitor fuel consumption, electric consumption, and other energy-related metrics to optimize operations and reduce costs.

**9. \*Surveillance Cameras\*:** Include surveillance cameras for security purposes, which can also be used to assess passenger load in different sections of the vehicle.

**10. \*Data Integration\*:** Collect all this data and integrate it into a centralized system that transportation authorities, operators, and passengers can access.

By incorporating these sensor technologies, public transportation systems can enhance the rider experience, increase efficiency, reduce wait times, and ultimately promote the use of public transportation.

**PROTOCOL:**

To monitor ridership, track locations, and predict arrival times in public transportation vehicles using a protocol, you can consider implementing an IoT (Internet of Things) solution. Here's how it can work**:**

**1. \*Choose a Communication Protocol\*:** Select a suitable communication protocol for connecting and transmitting data between sensors and a central system. Common options include MQTT, CoAP, HTTP, or proprietary protocols.

**2. \*GPS and Location Sensors\*:** Equip vehicles with GPS and location sensors that communicate their coordinates through the chosen protocol to a central server.

**3. \*Passenger Count Sensors\*:** Install sensors at vehicle entrances to count passengers boarding and alighting. These sensors transmit data via the chosen protocol to track occupancy.

**4. \*Smart Payment Systems\*:** Utilize contactless payment systems that communicate transaction data securely using the chosen protocol to track passenger entry and exit times**.**

**5. \*Traffic and Weather Sensors\*:** Integrate traffic and weather sensors to monitor road conditions. These sensors can periodically send updates via the protocol.

**6. \*Predictive Analytics\*:** Implement a predictive analytics system that uses the collected data to estimate arrival times. The results can be communicated to passengers and central systems via the protocol.

**7. \*Communication Systems\*:** Develop communication systems that relay real-time information about delays, expected arrival times, and route changes to passengers through mobile apps or displays at stops using the chosen protocol.

**8. \*Maintenance Sensors\*:** Install sensors to monitor vehicle health and send maintenance alerts via the protocol, allowing for predictive maintenance.

**9. \*Energy Efficiency Sensors\*:** Use sensors to track energy consumption and transmit this data via the protocol for analysis and optimization.

**10. \*Data Integration\*:** Ensure that all the data from various sensors is collected, processed, and integrated into a central server through **the** chosen communication protocol. This server can be accessed by transportation authorities, operators, and passengers.

By using a well-defined communication protocol, you can establish a reliable and secure connection between sensors on public transportation vehicles and the central monitoring system, enabling efficient data transfer and real-time updates for ridership tracking, location monitoring, and arrival time prediction.

**COMMUNICATION:**

Monitoring ridership, tracking vehicle locations, and predicting arrival times in public transportation vehicles using communication technologies involves creating a system that collects and shares real-time data. Here's how it can be done**:**

**1. \*GPS and Location Data\*:** Equip vehicles with GPS receivers and location tracking systems. These systems continuously collect location data, which can be communicated via various communication methods.

**2. \*Passenger Count Sensors**\*: Install sensors at entrances and exits to count passengers boarding and alighting. These sensors can be connected to a central system via communication technology to track occupancy.

**3. \*Smart Payment Systems**\*: Implement contactless payment systems (RFID cards, mobile apps) that not only improve efficiency but also communicate transaction data to a central server for tracking entry and exit times.

**4. \*Traffic and Weather Data\*:** Integrate traffic and weather data sources, which can provide real-time information on road conditions and weather-related issues. This data can be communicated through various methods.

**5. \*Predictive Analytics\*:** Employ predictive analytics to estimate arrival times based on real-time and historical data. The results can be communicated to passengers and central systems.

**6. \*Communication Systems\*: Develop communication systems that relay real-time information to passengers through mobile apps, displays at stops, or text message alerts regarding delays, expected arrival times, and route changes.**

**7. \*Maintenance Alerts\*:** Install sensors to monitor vehicle health and communicate maintenance alerts via various communication methods, such as emails or SMS, to maintenance teams**.**

**8. \*Energy Efficiency Data\*:** Monitor energy consumption data and communicate it to a central system for analysis and optimization**.**

**9. \*Data Integration\*:** Collect, process, and integrate all the data from different sources into a central system that transportation authorities, operators, and passengers can access through communication technologies.

Common communication technologies for transmitting this data include cellular networks (3G, 4G, 5G), Wi-Fi, LoRa, Zigbee, and IoT-specific communication protocols**.**

**OBJECTIVES FOR SMART PARKING:**

* EFFICIENT SPACE UTILIZATION
* REAL-TIME AVAILABILITY
* REDUCED ENVIRONMENTAL IMPACT
* REVENUE GENERATION
* ENHANCED USER EXPERIENCE
* DATA COLLECTION
* SECURITY

**1. Efficient Space Utilization:** Maximize the utilization of parking spaces, reducing congestion and minimizing the time it takes to find a parking spot.

**2. Real-time Availability:** Provide users with real-time information on available parking spaces, reducing the frustration of searching for parking.

**3. Reduced Environmental Impact:** Minimize fuel consumption and emissions by helping users find parking quickly.

**4. Revenue Generation:** Increase revenue for parking operators through better management and pricing strategies.

**5. Enhanced User Experience:** Make parking convenient and user-friendly through mobile apps, payment options, and navigation assistance.

**6. Data Collection:** Gather data on parking patterns to support urban planning and transportation decisions.

**7. Security:** Ensure the safety and security of parked vehicles through monitoring and surveillance.

**COMPONENT FOR SMART PARKING:**

**1. \*Sensors\*:** These can be in-ground sensors, ultrasonic sensors, or camera-based systems that detect the presence of vehicles in parking spaces. They provide real-time data on space occupancy.

**2. \*Communication Network\***: A robust network, often using IoT (Internet of Things) technology, to transmit data from sensors to a central server or cloud-based platform.

**3. \*Data Processing and Storage\*:** A central server or cloud infrastructure to process, store, and analyze the data received from sensors. This is where algorithms calculate parking availability and other insights.

**4. \*User Interface\*:** Mobile apps or websites that allow users to check parking availability, reserve spots, and make payments. These interfaces often include navigation features to guide users to available spaces.

**5. \*Payment and Access Control Systems\*:** Integration with payment gateways for users to pay for parking. Access control systems can include gates, barriers, and license plate recognition for entry and exit.

**6. \*Analytics and Reporting Tools\***: Data analytics tools to generate reports and insights for parking operators. This can help in optimizing pricing, improving user experience, and making data-driven decisions.

**7. \*Signage and Guidance Systems\*:** LED signs and displays that provide real-time information on available parking spaces and guide drivers to vacant spots.

**8. \*Security and Surveillance\*:** CCTV cameras and security systems to monitor the parking facility, ensuring the safety of vehicles and users.

**9. \*Backend Software\*:** Software to manage the entire parking system, including space allocation, pricing strategies, and operational management.

**10. \*Integration with Urban Planning\*:** Sharing data with city or transportation authorities to assist in city planning and traffic management.

**11. \*Maintenance and Support\*:** Regular maintenance and technical support to ensure the system runs smoothly and addresses any issues promptly.

**12. \*Environmental Sensors (Optional)\*:** Some systems include environmental sensors to monitor air quality or weather conditions, which can impact parking management decisions.

**COMPONENT REQUIRES:**

1. SENSORS
2. COMMUNICATION NETWORK
3. DATA PROCESSING AND STORAGE
4. USER INTERFACE
5. PAYMENT AND ACCESS CONTROL SYSTEM
6. ANALYTICS AND REPORTING TOOLS
7. SIGNAGE AND GUIDANCE SYSTEMS
8. SECURITY AND SURVILLANCE
9. BACKEND SOFTWARE
10. INTEGRATION WITH URBAN PLANNING
11. MAINTENANCE AND SUPPORT

**SENSOR USED FOR SMART PARKING**:

Smart parking systems utilize various sensors to monitor and manage parking spaces effectively. Some of the commonly used sensors include:

**1. \*Ultrasonic Sensors:\*** These sensors use sound waves to detect the presence of vehicles in parking spaces. They are often placed above or below parking spaces and can accurately determine whether a spot is occupied or vacant.

**2. \*Infrared Sensors:\*** Infrared sensors use infrared light to detect the presence of vehicles. They are capable of differentiating between stationary and moving objects, making them useful for smart parking.

**3. \*Magnetic Sensors:\*** Magnetic sensors rely on changes in the Earth's magnetic field caused by the presence of vehicles. They are typically embedded in the road or parking surface, making them less visible and durable.

**4. \*Camera-based Sensors:\*** Cameras, often equipped with image recognition software, capture real-time images of parking areas. These images are processed to determine parking space occupancy.

**5. \*Wireless Sensors:\*** These sensors are often integrated into parking meters or other infrastructure. They use wireless communication to relay data about parking space occupancy to a central system.

**6. \*LIDAR Sensors:\*** LIDAR (Light Detection and Ranging) sensors use laser beams to measure distances and create detailed 3D maps of their surroundings. They can be used to monitor the availability of parking spaces and provide more detailed information about vehicle positions.

**7. \*Acoustic Sensors:\*** Acoustic sensors can detect sounds or vibrations associated with parkingevents. They are useful for monitoring large parking lots .

**ULTRASONIC SENSOR :**

**DEFINATION:**

An ultrasonic sensor is a device that uses high-frequency sound waves (ultrasonic waves) to detect the distance or presence of objects. It emits ultrasonic pulses and measures the time it takes for the sound waves to bounce back after hitting an object, allowing it to calculate the distance to the object based on the speed of sound in the surrounding medium. These sensors are commonly used in applications such as distance measurement, object detection, and obstacle avoidance in robotics and automation**.**

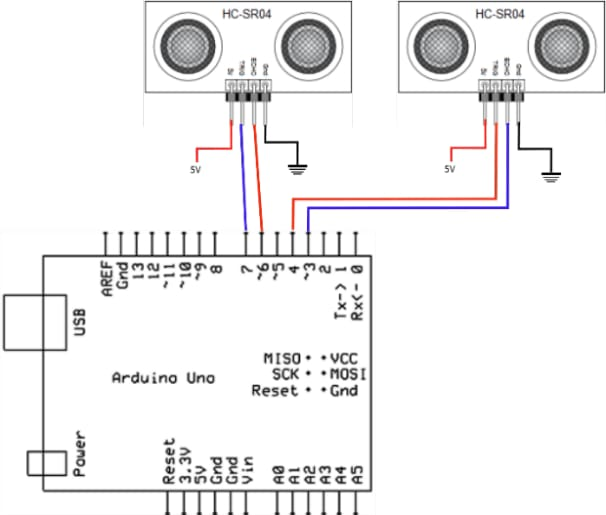


**PIN CONFIGURATION FOR ULTRASONIC SENSOR :**

**1. VCC (Voltage):** This is the power supply pin. You connect it to a voltage source, often +5V or +3.3V, depending on the sensor's specifications.

**2. GND (Ground**): This is the ground or 0V reference connection. It should be connected to the ground of your power supply.

**3. SIG (Signal**): This pin is used for both triggering the ultrasonic pulse and receiving the echo signal. It is the pin from which you can measure the distance or detect objects. You'll connect this pin to a digital input/output pin on your microcontroller or other control system.

  **INFRARED SENSOR:**

**DEFINATION:**

Infrared (IR) sensors are devices that can detect or sense infrared radiation, which is electromagnetic radiation with wavelengths longer than those of visible light. These sensors are designed to capture and interpret the heat radiation or thermal emissions from objects and living beings, as they emit infrared radiation as a function of their temperature.

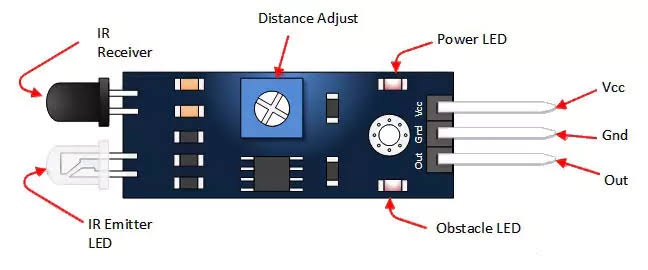


**PIN CONFIGURATION FOR INFRARED SENSOR:**

**1. VCC (Power**): This pin is typically connected to the positive supply voltage (e.g., 5V) to power the sensor.

**2. GND (Ground):** This pin is connected to the ground or 0V reference.

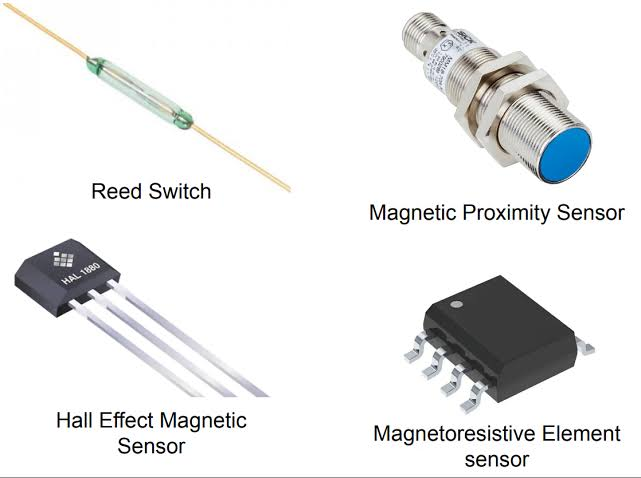
**3. OUT (Output):** The output pin typically provides a signal that changes in response to detected infrared radiation. It may be an analog voltage or a digital signal, depending on the sensor type.



**MAGNETIC SENSOR:**

**DEFINATION:**

A magnetic sensor is a device that measures and detects magnetic fields. It can determine the presence, strength, and direction of magnetic fields, making it useful in various applications, including navigation, automotive systems, industrial machinery, and electronic devices. These sensors can be based on different technologies, such as Hall effect sensors, magnetoresistive sensors, or fluxgate sensors, depending on their specific functionality and sensitivity to magnetic fields.

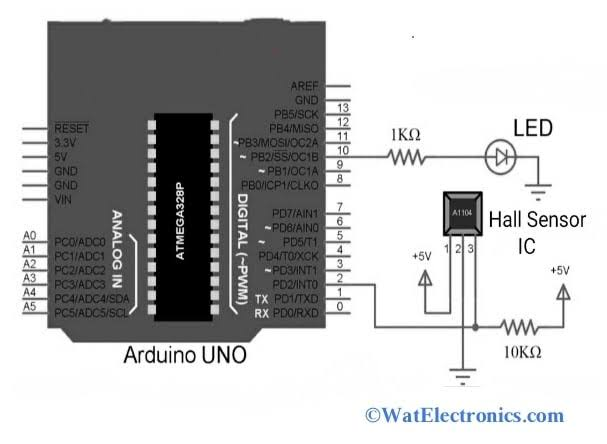


**PIN CONFIGURATION FOR MAGNETIC SENSOR:**

**1. \*Power Supply Pins (Vcc or Vin):\*** These pins provide the voltage required to operate the sensor. You'll typically connect the positive (Vcc) and negative (GND) power supply pins to your power source.

**2. \*Output Pin:\*** This pin provides the output signal that reflects the magnetic field's strength or orientation. It can be analog (voltage or current) or digital (e.g., open-drain output). The type of output can vary.

**3. \*Ground (GND) Pin**:\* This is the common ground reference for the sensor and should be connected to the ground of your power supply and the system .



**CAMERA -BASED SENSOR:**

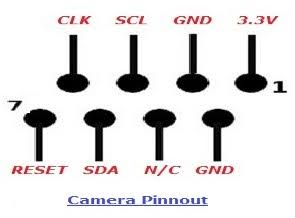
**DEFINATION:**

A camera-based sensor is a device that uses one or more cameras to capture visual information from the surrounding environment. These sensors can detect and process images or video in real-time, often used for applications such as photography, video recording, computer vision, object recognition, and more. They are widely employed in fields like robotics, autonomous vehicles, security systems, and various industrial and consumer applications to gather visual data and make informed decisions based on that data.



**PIN CONFIGURATION FOR CAMERA-BASED SENSOR:**

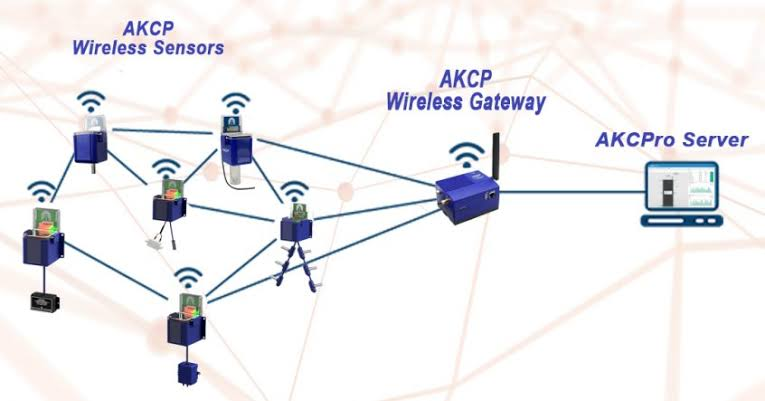
The pin configuration for a camera-based sensor can vary significantly depending on the specific sensor model, manufacturer, and interface it uses. Common interfaces for camera-based sensors include USB, MIPI, I2C, and GPIO. To determine the pin configuration for your particular camera-based sensor, you should consult the datasheet or user manual provided by the manufacturer. These documents typically include detailed information on the pinout, electrical connections, and how to interface with the sensor. You may also need to consider power supply voltage, ground connections, data lines, and any additional control or communication pins specific to your sensor.



**WIRELESS SENSOR:**

**DEFINITION:**

It seems like there might be a typo in your question. Did you mean to ask about a "wireless sensor" and its definition? If so, a wireless sensor is a device that can collect data and transmit it wirelessly, typically using technologies like Wi-Fi, Bluetooth, or other wireless communication protocols. These sensors are often used in various applications, including environmental monitoring, home automation, and industrial settings.

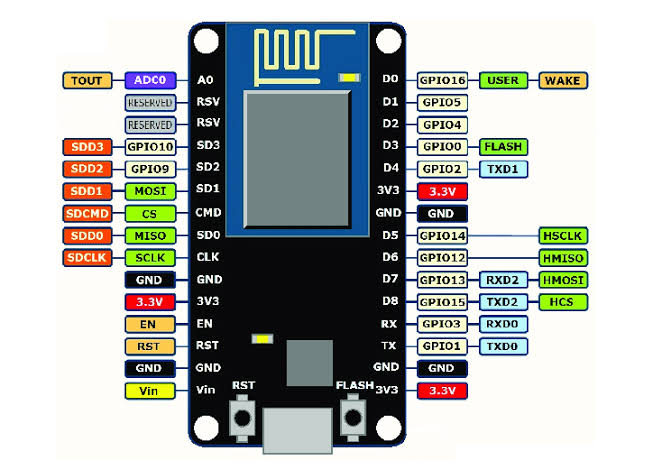


**PIN CONFIGURATION FOR WIRELESS SENSOR:**

**1. \*Power Supply**\*: This pin is used to provide the necessary voltage to the sensor. It's usually labeled as VCC or VDD.

**2. \*Ground\*:** This pin is for the ground connection, labeled as GND or GROUND.

**3. \*Data or Signal Pin**\*: This pin is used for transmitting data from the sensor. It can be labeled as DATA, TX (transmit), or a similar designation.



**LIDAR SENSOR:**

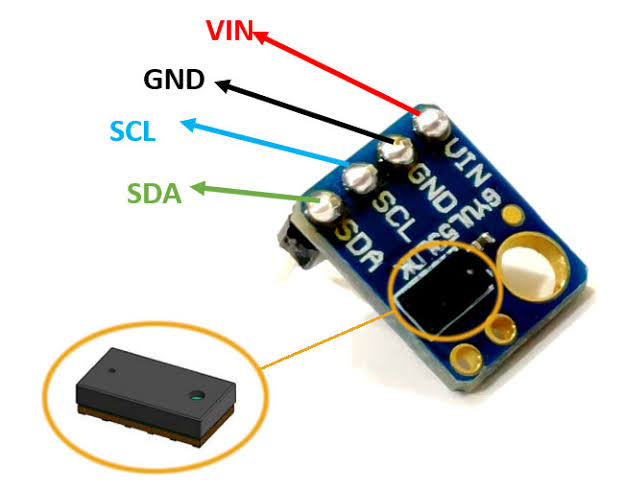
**DEFINITION:**

Lidar, short for Light Detection and Ranging, is a remote sensing technology that uses laser light to measure distances and create detailed three-dimensional representations of the environment. It works by emitting laser pulses and measuring the time it takes for the light to bounce back after hitting an object. Lidar sensors are commonly used in applications like autonomous vehicles, surveying, forestry, and archaeology for mapping and detecting objects in their surroundings with high precision.



**PIN CONFIGURATION FOR LIDAR SENSOR:**

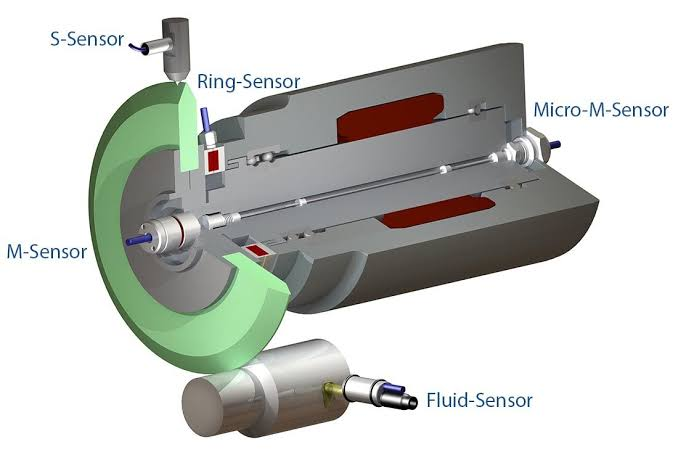
The pin configuration for a LiDAR sensor can vary depending on the specific model and manufacturer. Generally, LiDAR sensors have several pins, including power supply, ground, data communication, and sometimes additional pins for specific functions or features. To get the precise pin configuration for your LiDAR sensor, you should refer to the datasheet or user manual provided by the manufacturer. These documents typically contain detailed information about the pinout and how to connect and use the sensor.



**ACOUSTIC SENSOR:**

**DEFINITION:**

An acoustic sensor, also known as a sound sensor or microphone sensor, is a device designed to detect and measure sound or acoustic waves in the surrounding environment. It converts sound waves, which are variations in air pressure, into electrical signals that can be processed and analyzed by electronic circuits or microcontrollers. Acoustic sensors are used in a wide range of applications, including audio recording, noise monitoring, voice recognition, and various industrial and scientific applications where the detection of sound or vibrations is required. These sensors are essential components in devices like microphones, ultrasonic sensors, and other sound-related equipment.

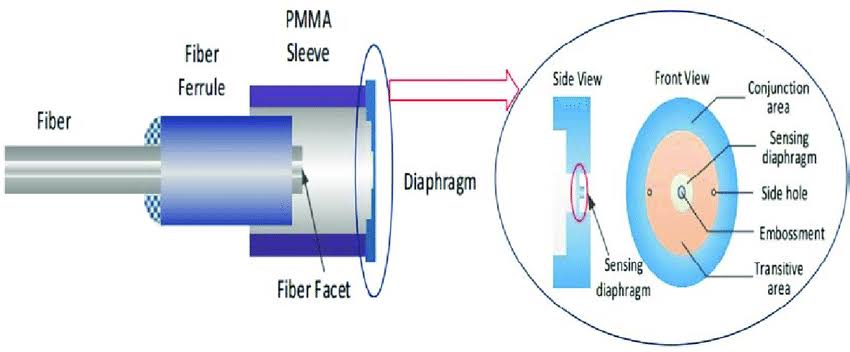


**PIN CONFIGURATION FOR ACOUSTIC SENSOR:**

**1. Signal Output:** This pin provides the analog or digital signal output that represents the detected sound or vibration.

**2. Power Supply (Vcc):** This pin is used to provide the necessary voltage to the sensor. The voltage requirements can vary, so check the datasheet for the specific sensor.

**3. Ground (GND):** This is the common ground reference for the sensor.



**PROCEDURE:**

**1. \*Gather the Necessary Hardware:\***

Raspberry Pi board with an internet connection.

- Sensors compatible with Raspberry Pi (e.g., DHT22 for temperature and humidity).

**2. \*Set Up Raspberry Pi:\***

- Install a compatible operating system on your Raspberry Pi (e.g., Raspberry Pi OS).

- Configure Wi-Fi or Ethernet for internet connectivity.

**3. \*Install Required Libraries:\***

- Open a terminal on your Raspberry Pi and install the necessary Python libraries for your sensors (e.g., Adafruit\_DHT for DHT sensors) and internet communication (e.g., requests). bashpip install Adafruit\_DHT requests

**4. \*Write Python Script:\***

- Create a Python script to read data from your sensors and send it to the cloud or mobile app server. Here's a basic example:

**5. \*Security Considerations:\***

- Implement security measures, such as using HTTPS for communication and handling authentication if required.

**6. \*Error Handling and Logging:\***

- Implement error handling and logging mechanisms to troubleshoot issues.

**7. \*Testing and Deployment:\***

- Test the system thoroughly on your Raspberry Pi and deploy it in a stable environment.

**8. \*Maintenance:\***

- Regularly update and maintain your system to ensure it continues to work reliably.

**ALGORITHM :**

**STEP 1:**

Set up your Raspberry Pi with an operating system (e.g., Raspbian).

Connect the sensors (e.g., temperature, humidity, light, etc.) to the GPIO pins of the Raspberry Pi.

**STEP 2:**

Install necessary libraries for interacting with sensors, such as RPi.GPIO or Adafruit libraries.

**STEP 3:**

Write code to read data from the connected sensors. This will depend on the specific sensors you're using.

**STEP 4:**

Process the sensor data if needed (e.g., convert units, filter noise, etc.).

**STEP 5:**

Choose a cloud service or set up your own server for data storage and processing. Obtain necessary credentials (API keys, URLs, etc.) to interact with your chosen cloud service or server.

**STEP 6:**

Use a suitable protocol (e.g., HTTP, MQTT, or other IoT protocols) to send the sensor data to the cloud or server.

**STEP 7:**

Implement security measures like encryption and authentication to protect the data in transit.

**STEP 8:**

Log data locally on the Raspberry Pi in case of network disruptions or server downtime.

**STEP 9:**

Implement error handling to manage network issues or data transmission failures.

**STEP 10:**

Run the sensor data collection and transmission code continuously as a background process.

**STEP 11:**

The app and set up communication protocols (e.g., REST API) to fetch data from the cloud or server.If you want to display data in a mobile app, develop

**STEP 12:**

Use visualization tools or libraries to present sensor data in a user-friendly format on the mobile app or a web dashboard.

**STEP 13:**

Test the system thoroughly and debug any issues.

**STEP 14:**

Optimize your code and architecture for scalability if you plan to collect data from multiple sensors or devices.

**STEP 15:**

Document your code, hardware connections, and configurations for future reference.

**PROGRAM:**

# Configure DHT sensor

sensor = Adafruit\_DHT.DHT22

pin = 4 # GPIO pin

# Configure AWS IoT

iot\_client = boto3.client('iot-data', region\_name='your-region')

while True:

humidity, temperature = Adafruit\_DHT.read\_retry(sensor, pin)

if humidity is not None and temperature is not None:

data = {'temperature': temperature, 'humidity': humidity}

iot\_client.publish(

topic='your-topic',

payload=json.dumps(data),

qos=1

)

`**Output:**

Published data: {'temperature': 23.5, 'humidity': 45.3}

Published data: {'temperature': 23.6, 'humidity': 45.2}

Published data: {'temperature': 23.5, 'humidity': 45.1}

...

**DEVELOPMENT PART-2 PHASE -4 :DOCUMENT SUBMISSION**

To develop the web and android application for displays the displays real-time restroom availability and cleanliness data.

**WEB DEVELOPMENT USING HTML CODE: HTML CODE:**

<!DOCTYPE *html*>

<html>

<head>

        <title>smart parking</title>

  <link *rel*="stylesheet" *href*="style.css">

</head>

<body>

    <h1><font *color*="blue"><center><a *href*=></a>SMART PARKING</a></center></font></h1>

    <center><img *src*="https://urbanupdate.in/wp-content/uploads/2019/07/E5ccmfalj3QPUk-TBkulPkzn2ztS8lOqJZGHlf\_oBQ8-1280x960.jpg"*width*="600"*height*="400"></center>

  <script *src*="script.js"></script>

</body>

</html>

<!DOCTYPE *html*>

<html>

<head>

<title>Page Title</title>

</head>

<body>

<h1>Need of smart public restroom</h1>

<P>Smart parking refers to the use of technology to improve the management and efficiency of parking spaces. This can include features like sensors to detect available parking spots, mobile apps to help users find parking, and data analytics to optimize parking operations. It can reduce congestion, save time for drivers, and improve the utilization of parking facilities</P>

<center><h2><font *color*="green">OUR TEAM MEMBERS ARE</font></h2></center>

<li>Prabhakararaj(TEAM HEAD)</li>

        <li>vijiyaprabhakaran</li>

        <li>sakthivel</li>

        <li>mohanraj</li>

        <li>sridhar</li>

        <li>mahenderan</li>

<head>

<style>

  .increased-font{font-size: :20px;}

</style>

</head>

<body><strong><h2><font *color*="blue">For example</font></h2></strong>

Smart parking refers to the use of technology to improve the management and efficiency of parking spaces. This can include features like sensors to detect available parking spots, mobile apps to help users find parking, and data analytics to optimize parking operations. It can reduce congestion, save time for drivers, and improve the utilization of parking facilities.

   <img *src*="https://www.anabon.com/wp-content/uploads/2023/08/smart-restroom-2-copy.webp"*width*="600"*height*="400">

<body>

   <!DOCTYPE *html*>

<html>

<head>

<style>

table {

  font-family: arial, sans-serif;

  border-collapse: collapse;

  width: 100%;

}

td, th {

  border: 1px solid #dddddd;

  text-align: left;

  padding: 8px;

}

tr:nth-child(even) {

  background-color: #dddddd;

}

</style>

</head>

<body>

<center><h2><font *color*="green">ROOMS AVAILABILITY TABLE</font></h2></center>

<table>

  <tr>

    <th>AREA</th>

    <th>NO.OF ROOMS</th>

    <th>ROOMS AVAILABILITY</th>

  </tr>

  <tr>

    <td>CITY-1</td>

    <td>20</td>

    <td>15</td>

  </tr>

  <tr>

    <td>CITY-2</td>

    <td>12</td>

    <td>10</td>

  </tr>

  <tr>

    <td>CITY-3</td>

    <td>18</td>

    <td>15</td>

  </tr>

  <tr>

    <td>CITY-4</td>

    <td>23</td>

    <td>20</td>

  </tr>

  <tr>

    <td>CITY-5</td>

    <td>28</td>

    <td>20</td>

  </tr>

  <tr>

    <td>CITY-6</td>

    <td>15</td>

    <td>12</td>

  </tr>

</table>

</body>

</html>

</body>

</html>