**Building a smart car parking system involves integrating various technologies.Here’s a high-level overview of the process:**

Sensors:

Install sensors (e.g., ultrasonic, infrared, or cameras) at parking spaces to detect the presence of vehicles.

Data Connectivity:

Establish a network to connect the sensors to a central control system. This can be wired or wireless, such as Wi-Fi, Bluetooth, or cellular connectivity.

Central Control System:

Develop a software platform that receives data from sensors, processes it, and manages parking spaces.

Mobile App/Website:

Create a user-friendly interface for drivers to check parking availability, reserve spots, and make payments.

Data Processing and Analysis:

Use algorithms to analyze sensor data and determine parking space availability in real-time.

Payment Integration:

Integrate payment gateways to allow users to pay for parking through the app or website.

Reservation System:

Implement a reservation system to allow users to book parking spots in advance.

Security and Surveillance:

Install cameras for security and surveillance purposes, and integrate them with the system.

User Notifications:

Send notifications to users about available parking spaces, reservation confirmations, and payment receipts.

Machine Learning and AI:

Utilize machine learning and AI algorithms to predict parking demand, optimize pricing, and improve system efficiency.

Feedback and Reviews:

Include features for users to provide feedback and rate their parking experience.

Maintenance and Support:

Ensure regular maintenance and customer support for system reliability.

Regulatory Compliance:

Comply with local regulations and privacy laws, especially when handling user data.

Scalability:

Design the system to be scalable to accommodate more parking spaces and users as needed.

Testing and Quality Assurance:

Thoroughly test the system to ensure it functions correctly and efficiently.

Monitoring and Analytics:

Implement a system to monitor the performance and gather analytics for continuous improvement.

**Components**:

IR Sensor:

1.Connect Vcc of the IR sensor to the output pin of the 7805 voltage regulator.

2.Connect GND of the IR sensor to the ground rail of the circuit.

3.Connect the OUT pin of the IR sensor to a GPIO pin on your microcontroller (e.g., Raspberry Pi).

7805 Voltage Regulator:

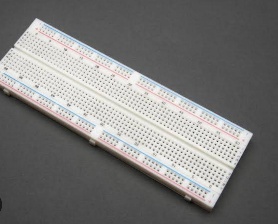
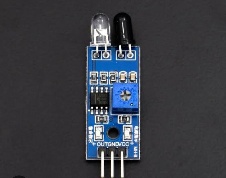
1.Connect the input pin of the 7805 to your power source (e.g., 9V battery or a suitable power supply).

2. Connect the ground pin of the 7805 to the ground rail of the circuit.

3.Connect the output pin (5V) of the 7805 to the Vcc pins of the IR sensor and LEDs.

LEDs:

1.Connect the anode (longer leg) of each LED through a current-limiting resistor to the 5V output of the 7805. 2.Connect the cathode (shorter leg) of each LED to a separate GPIO pin on your microcontroller (these will be used to control the LED).





Resistor:

A 1k ohm resistor (1,000 ohms) has various uses in electronic circuits.

Here are a few common applications:

Current Limiting:

It’s often used to limit the current flowing through an LED to prevent it from burning out.

For example, when connecting an LED to a power source, a 1k ohm resistor can be used in series to protect the LED.

Voltage Division:

In voltage divider circuits, a 1k ohm resistor can be used to create a reference voltage.

By connecting it in series with another resistor, you can divide the input voltage and obtain a specific output voltage.

Pull-Up or Pull-Down Resistor:

It’s commonly used in digital logic circuits as a pull-up or pull-down resistor.

These resistors ensure that a digital input has a defined voltage level when it’s not actively being driven high or low.

Biasing Transistors:

In transistor amplifier circuits, 1k ohm resistors are often used to set the biasing point, ensuring that the transistor operates in its active region.

Filter Circuits:

They can be used in RC filter circuits to set the cutoff frequency for filtering applications.

General Signal Conditioning:

1k ohm resistors are versatile and can be used for various signal conditioning purposes, like voltage level shifting, scaling, or impedance matching.

Breadboard:

A breadboard is a handy tool for building and testing electronic circuits without soldering. Here’s a basic guide on how to use one:

Components: Gather your components like resistors, LEDs, wires, and integrated circuits that you want to connect in your circuit.

Breadboard Layout: A standard breadboard has rows of holes separated by a central trench. Each row of holes is connected internally, and there are often power rails on the sides.

Power Rails: The long rows on the sides of the breadboard are typically used as power rails. The left side is usually for ground (GND), and the right side for voltage (VCC).Component Placement: Insert your components into the breadboard. The component’s leads should go into the holes. Make sure you follow the component’s datasheet or circuit diagram for proper connections.

Wiring: Use jumper wires to connect components by plugging them into different holes on the breadboard. Make sure to connect components as per your circuit diagram.

Testing: Power your circuit by connecting a voltage source (e.g., a battery or a power supply) to the appropriate power rails. Use a multimeter to check voltages and currents if necessary.

Prototyping: Breadboards are great for prototyping and testing circuits. You can easily modify your circuit by repositioning components and wires.

Double-Check: Ensure that there are no loose connections or short circuits. Double-check your connections to avoid damaging components.

Documentation: It’s a good practice to document your circuit as you build it. Draw a schematic or take notes for future reference.

Soldering: Once you’ve tested and finalized your circuit on the breadboard, you can move to soldering it on a perfboard or a PCB for a more permanent solution.

**Circuit connection steps:**

• Place the IR Sensor.

• Label the top pin as “Vcc,” the middle pin as “OUT,” and the bottom pin as “GND.”

• To represent the 7805 Voltage Regulator,

Label the left pin as “Input,” the middle pin as “GND,” and the right pin as “Output (5V).”

• To represent the LEDs.Label each LED with “LED1,” “LED2,” and “LED3.”

**Connection of the components**

• Connect the “Vcc” pin of the IR Sensor to the “Output (5V)” pin of the 7805 Voltage Regulator.

• Connect the “GND” pin of the IR Sensor to the “GND” pin of the 7805 Voltage Regulator.

• Connect the “OUT” pin of the IR Sensor to a GPIO pin on your microcontroller (e.g., Raspberry Pi).

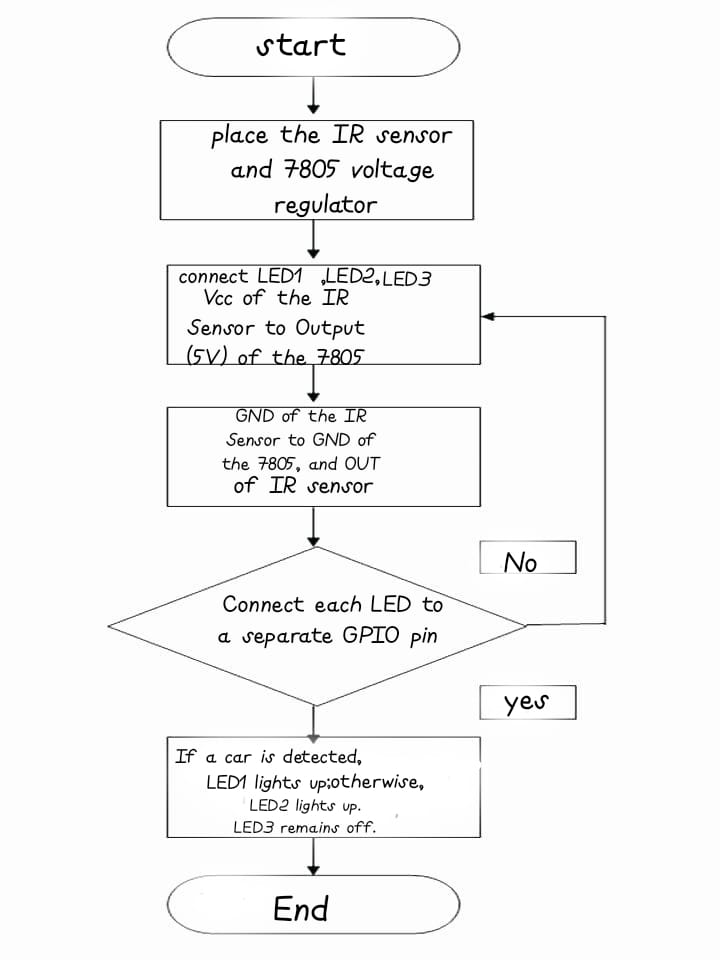
• Connect the “Input” pin of the 7805 Voltage Regulator to your power source.

• Connect the “GND” pin of the 7805 Voltage Regulator to the ground rail of the circuit.

• Connect the “Output (5V)” pin of the 7805 Voltage Regulator to the “Vcc” pins of the IR Sensor and LEDs.

• Connect the anode (longer leg) of each LED through a current-limiting resistor to the “Output (5V)” of the 7805 Voltage Regulator.

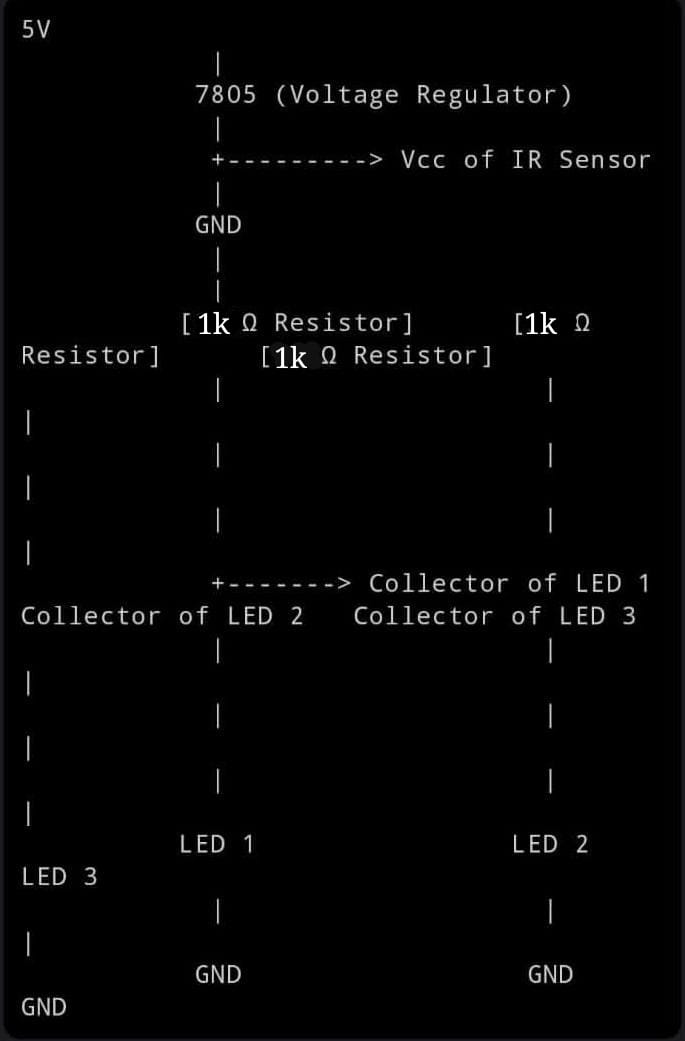
• Connect the cathode (shorter leg) of each LED to separate GPIO pins on your microcontroller.



**Flow chart**:

**Diagram**:

:

****

**Codings:**

Import RPi.GPIO as GPIO

Import time

# Set up GPIO

GPIO pin, adjust as needed

IR\_SENSOR\_PIN = 17 # Example GPIO pin, adjust as needed

LED1\_PIN = 18

LED2\_PIN = 19

LED3\_PIN = 20

# Initialize GPIO pins

GPIO.setup(IR\_SENSOR\_PIN, GPIO.IN)

GPIO.setup(LED1\_PIN, GPIO.OUT)

GPIO.setup(LED2\_PIN, GPIO.OUT)

GPIO.setup(LED3\_PIN, GPIO.OUT)

Def check\_parking\_status():

If GPIO.input(IR\_SENSOR\_PIN) == GPIO.LOW:

Return True # Car detected

Else:

Return False # No car detected

Def indicate\_parking\_status(status):

If status:

GPIO.output(LED1\_PIN, GPIO.HIGH) # LED1 ON

GPIO.output(LED2\_PIN, GPIO.LOW) # LED2 OFF

GPIO.output(LED3\_PIN, GPIO.LOW) # LED3 OFF

Else:

GPIO.output(LED1\_PIN, GPIO.LOW) # LED1 OFF

GPIO.output(LED2\_PIN, GPIO.HIGH) # LED2 ON

GPIO.output(LED3\_PIN, GPIO.LOW) # LED3 OFF

Try:

While True:

Status = check\_parking\_status()

Indicate\_parking\_status(status)

Time.sleep(1)

Except KeyboardInterrupt:

GPIO.cleanup()

**Working**:

* The IR sensor is used to detect the presence of a car. If a car is detected, LED1 lights up; otherwise, LED2 lights up.
* LED3 remains off.
* Make sure to adjust the GPIO pin numbers and connections according to your specific setup.
* Remember to use appropriate current-limiting resistors for the LEDs to prevent them from burning out.
* Additionally, ensure that your power supply can handle the load of the IR sensor and the LEDs.