Sample IoT Design Report

Smart Obstacle Avoiding Car Using IoT Sensors

1. Purpose & Requirements Specification

The primary purpose of this project is to design a theoretical IoT-based smart car system capable of autonomously avoiding obstacles in real time. The system is intended for educational use, leveraging the Freenove 4WD Smart Car Kit with a Raspberry Pi to simulate basic intelligent vehicle behavior.

Requirements include:

- Autonomous navigation in an indoor environment.

- Use of ultrasonic sensors for obstacle detection.

- Real-time distance measurement and response.

- Basic decision-making logic to change direction or stop.

- Battery-powered and portable operation.

2. Process Specification

The smart car continuously monitors its environment using the ultrasonic sensor mounted at the front. When the sensor detects an obstacle within a defined proximity (e.g., 20 cm), the Raspberry Pi triggers a decision-making process that either halts the car or redirects it by turning left or right. The car resumes forward movement once the obstacle is no longer in range.

Use Case Scenario:

1. The car is powered on and begins moving forward.

2. The sensor measures distance periodically.

3. If an obstacle is detected:

   - Stop.

   - Evaluate space on left and right.

   - Turn to avoid obstacle.

4. Continue movement after path is clear.

3. Domain & Information Model

The domain model includes the following primary entities:

- SmartCar: The 4WD vehicle chassis with attached sensors and motors.
- SensorModule: HC-SR04 ultrasonic sensor responsible for environment scanning.
- MotorController: L298N driver circuit controlled via GPIO.
- ControlUnit: Raspberry Pi microcontroller executing logic.
- PowerUnit: Rechargeable battery powering all components.

Relationships:

- SmartCar contains SensorModule and MotorController.
- ControlUnit interfaces with both SensorModule and MotorController.
- SensorModule provides data (distance) to ControlUnit.
- ControlUnit sends commands (forward, stop, turn) to MotorController.

4. Service Specification

The smart car provides the following services:

1. ObstacleDetectionService:
   - Input: Distance data from ultrasonic sensor.
   - Output: Trigger stop or turn command.
   - Frequency: Continuous loop (real-time).

2. MotionControlService:
   - Input: Decision command (move, stop, turn).
   - Output: Motor GPIO signals.

3. StatusUpdateService:

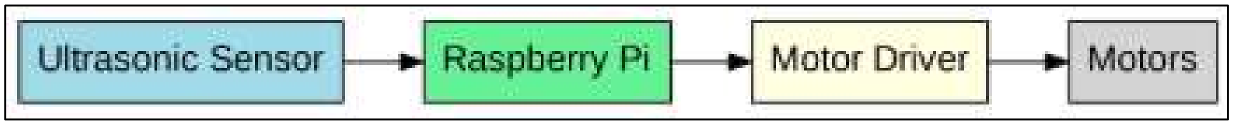
- Input: Car's current action.

- Output: Logs state (optional for future improvement).

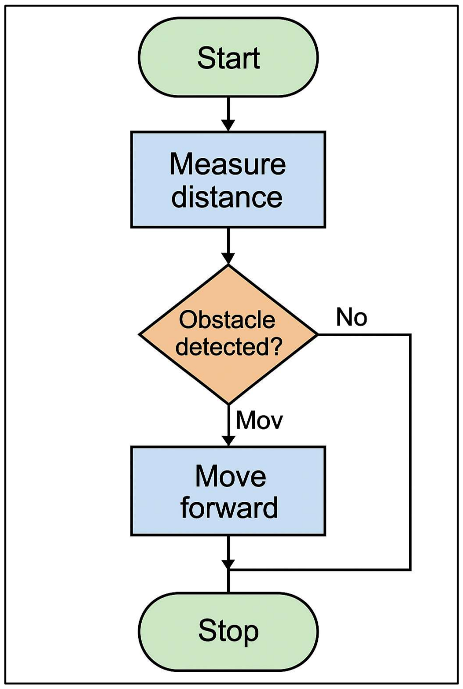
5. Application Design & Integration

The system is designed around a modular IoT architecture, using a Raspberry Pi to integrate sensor readings and motor actuation. Components are connected via GPIO pins, and the logic is executed in Python using simple loops.

5.1. Block Diagram:



5.2. Flow Chart and Pseudocode :



Pseudocode:

```
while True:
    distance = sensor.get_distance()
    if distance < 20:
        stop_motors()
        turn_left()
    else:
        move_forward()
```

This basic loop ensures the car remains responsive to its environment and avoids obstacles as long as it is active.

5.3. How the system works

The smart obstacle-avoiding car operates based on sensor input and programmed logic. When powered on, the system enters a loop that continuously monitors distance data from the ultrasonic sensor. If the sensor detects an object within the threshold (e.g., 20 cm), the Raspberry Pi triggers a response to avoid the obstacle. This may involve stopping the car and turning left or right. If no obstacle is detected, the car continues to move forward.

This behavior is implemented using simple Python code running on the Raspberry Pi, and real-time GPIO interactions with the motor driver module. The modular design allows the components to operate together seamlessly, enabling autonomous movement.

5.4. Components

- **Ultrasonic Sensor:** A sensor that measures distance by using sound waves. It emits an ultrasonic pulse and then listens for the echo; the built-in control circuit measures the round-trip time of the echo to calculate the distance to an objectt. This provides the Raspberry Pi with environmental data (e.g. how far away an obstacle is).
- **Raspberry Pi (Controller):** A small computer that acts as the "brain" of the system. It reads the sensor's input and runs the decision-making logic (software). Based on the ultrasonic sensor's data (and potentially other inputs or programmed criteria), the Raspberry Pi determines what actions to take. It then outputs appropriate control signals through its GPIO (General Purpose Input/Output) pins or interfaces.
- **Motor Driver:** An interface module (typically an electronic circuit or chip, like an H-bridge driver) that receives low-power control signals from the Raspberry Pi and translates them into higher-power signals to drive the motors. The Raspberry Pi's GPIO pins by themselves cannot safely power a motor (they supply only a few milliamps at 3.3 V, and direct connection could damage the Pi due to voltage spikes). Therefore, an external motor driver is used as a bridge, allowing the Pi to control the motors without providing power directly. In other words, the motor driver takes commands from the Pi and switches the necessary voltage/current to the motors from a separate power source.
- **Motors (Actuators):** The motors are the actuators that physically turn wheels or move parts of the system. They convert electrical energy into mechanical motion. When they receive power via the motor driver, they rotate or move accordingly. In IoT and embedded systems, motors carry out the physical action commanded by the controller – for example, driving a robot forward or adjusting a mechanism. (Actuators like motors are the components that *"move or control"* a system based on the controller's signals.)

Conclusion

This report provides a comprehensive academic design of a Smart Obstacle Avoiding Car IoT system. The design integrates key concepts of sensor-based automation, modular architecture, and real-time control. Although theoretical, this model offers foundational understanding necessary for implementing practical IoT-based robotics.

Future extensions include adding camera vision, GPS for outdoor use, or Bluetooth control via mobile.