Unmanned Smart Parking System

BCSE310L– IoTArchitectures and Protocols

Project REPORT

By

C. Jyohti Swaroop(22BCE3467)

K. Mukesh raja(22BCE2672)

CH. Mokesh Lakshman(22BCE2811)

School of Computer Science and Engineering

APRIL 2025

1. Introduction

As urbanization accelerates and the number of vehicles on the road continues to rise, cities face increasing challenges related to traffic management, environmental sustainability, and efficient use of infrastructure. One of the most pressing and often overlooked issues is the inefficiency in traditional parking systems. Drivers frequently spend excessive time searching for vacant parking spots, which not only leads to traffic congestion but also increases fuel consumption, emissions, and stress levels.

In recent years, the **Internet of Things (IoT)** has emerged as a transformative technology capable of addressing these urban challenges. IoT refers to a network of interconnected physical devices that collect, exchange, and analyze data with minimal human intervention. In the context of parking systems, IoT can enable the development of smart, automated, and real- time parking management solutions.

The **motivation** behind this work stems from the need to reduce the inefficiencies and frustrations associated with traditional parking methods. By leveraging IoT, we aim to build a system that not only monitors parking space occupancy but also improves decision-making for both drivers and city planners.

The **objective** of this project is to design and implement a **Unmanned smart parking system** that:

- Utilizes proximity or ultrasonic sensors to **detect the presence** of vehicles in individual parking slots.
- Uses an Arduino Uno microcontroller to process sensor data locally and control hardware components such as the servo motor and LCD display, without the need for wireless communication.
- Processes and stores real-time occupancy data locally on the Arduino board using internal variables and direct display, eliminating the need for a cloud-based server or database.

This system demonstrates the practical application of IoT technologies in **smart city**

infrastructure. It is intended to enhance urban mobility, reduce environmental impacts, and pave the way for more sustainable and efficient city planning.

2. System Requirements

To successfully implement an IoT-based parking system, the following hardware and software components are required:

1. Hardware Requirements

Component	Description
Arduino Uno Board	The central microcontroller that processes data and controls all components.
IR Sensors (x4)	Detect vehicle presence in individual parking slots.
Ultrasonic Sensors	Measure the distance of vehicles at the entry and exit points.
Servo Motor	Controls the gate for vehicle entry and exit automation.
LCD Display (I2C)	Displays real-time parking slot availability and system messages.
Breadboard	Used for prototyping and connecting components without soldering.
Jumper Wires	Connect sensors and components to the Arduino and breadboard.
Power Supply	Can be a USB cable or external battery pack to power the Arduino board.

2. Software Requirements

Component	Description
Arduino IDE	Used to program the microcontroller (ESP32/Arduino) with sensor logic and communication protocols.
Arduino Uno Board Drivers	Ensures proper communication between your computer and the Arduino Uno.
Servo.h	Controls the servo motor for gate operation.

Wire.h	Enables I2C communication between Arduino and the LCD display.
LiquidCrystal_I2C.h	Controls the I2C LCD to display real-time slot availability.
Serial Monitor (Arduino IDE)	Used for debugging sensor readings and monitoring system behavior.

Design of the Proposed System

The proposed **IoT-based Parking System** is designed to monitor the occupancy status of parking slots in real-time and communicate this information to users through a web or mobile interface. The design is modular, scalable, and suitable for implementation in both indoor and outdoor environments.

1. System Architecture Overview

The system consists of three main layers:

A. Sensing Layer (Edge Devices)

- **Ultrasonic/Proximity Sensors** are installed at each parking slot to detect whether a vehicle is present.
- These sensors are connected to a **microcontroller** (e.g., ESP8266/ESP32), which reads distance values and determines the slot status (occupied or free).

B. Application Layer (Arduino-Based Implementation)

- The application layer in this system is implemented through a local LCD display that shows the real-time status of parking slots directly to users on-

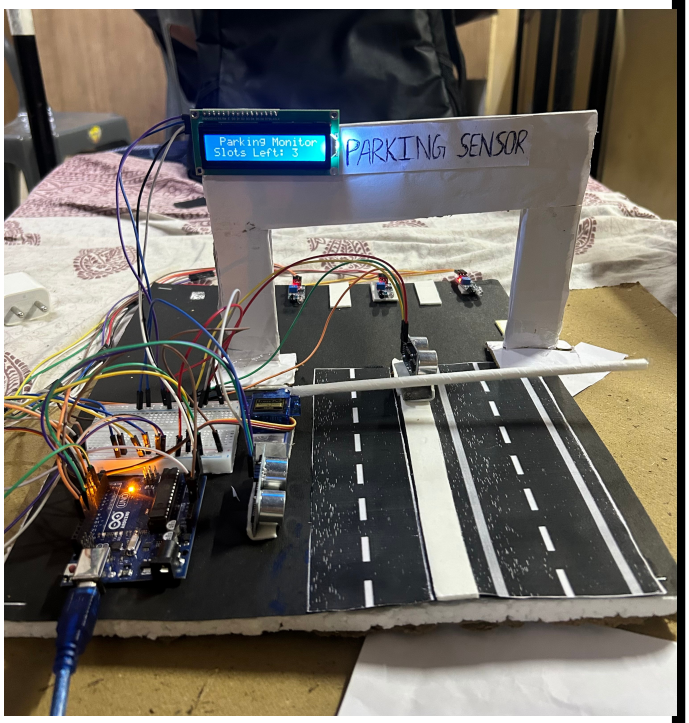
site.

- The LCD is connected to the Arduino Uno via I2C and continuously updates messages such as **"SLOTS LEFT: X"** or **"LOT IS FULL"**, based on sensor inputs.
- This layer allows users (e.g., drivers or attendants) to make immediate parking decisions without needing mobile apps, cloud dashboards, or internet connectivity.
- It also provides feedback on system status during entry and exit, such as gate operation messages or system initialization info.

2. Workflow of the System

1. **Sensor Detection**: The sensor continuously measures the distance between itself and any object (like a vehicle) in front of it.
2. **Data Processing**: If the measured distance is less than a predefined threshold, the microcontroller marks the slot as "occupied"; otherwise, it's "free".
3. **Data Transmission**: The Arduino board processes the slot status and directly updates the local display without requiring Wi-Fi or external database communication.
4. **Data Storage**: Slot status is tracked in real-time on the Arduino board, and changes are reflected instantly on the LCD display without the use of timestamps or external logging.
5. **User Access**: An onboard LCD display connected to the Arduino shows the real-time number of available parking slots directly at the parking location.
6. **Status Feedback (Optional)**: LEDs can show slot status on-site—Green for free, Red for occupied.

3. Features of the Design



- **Modular**: Each parking slot sensor operates independently and can be scaled for more slots.
- **Real-Time Updates**: Changes in parking status are reflected instantly on the connected LCD module, providing real-time feedback directly from the Arduino board.
- **Low Power Consumption**: Especially when using energy-efficient MCUs like ESP32 with sleep modes.
- **Mobile-Friendly**: End users can access the system via smartphones to save time and fuel.

4. Proposed System Explanation

The proposed system is an **automated smart parking system** built using an Arduino microcontroller, ultrasonic sensors, IR sensors, a servo motor, and an I2C LCD. It is designed to detect vehicles entering or exiting a parking area, monitor the availability of parking slots, and control access through a servo-operated gate.

Objectives

- **Monitor parking slots** in real-time.
- **Control gate access** based on slot availability.

- **Display available slots** to the user.
- **Automate entry and exit** based on vehicle detection.

CODE:

```
#include <Wire.h>
#include
<LiquidCrystal_I2C.h>
#include <Servo.h>
// I2C LCD
LiquidCrystal_I2C lcd(0x27, 16, 2);
// Servo
motor Servo
myservo;
// Ultrasonic pins (entry and
exit) const int trigPinEntry = 9;
const int echoPinEntry =
10; const int trigPinExit
= 11; const int
echoPinExit =12;
// IR sensor pins for slots
c o n s t  i n t
IR_SLOT1 = 5;
c o n s t  i n t
IR_SLOT2 = 6;
c o n s t  i n t
IR_SLOT3 = 7;
c o n s t  i n t
IR_SLOT4 = 8;
// Servo pin
```

```
const int SERVO_PIN = 4;
// Total slots
const int totalSlots
= 4; int
availableSlots = 4;
void setup() {
  Serial.begin(9600
  ); lcd.init();
  lcd.backlight();
  pinMode(IR_SLOT1,
  INPUT);
  pinMode(IR_SLOT2,
  INPUT);
  pinMode(IR_SLOT3,
  INPUT);
  pinMode(IR_SLOT4,
  INPUT);
  pinMode(trigPinEntry, OUTPUT);
  pinMode(echoPinEntry, INPUT);
  pinMode(trigPinExit, OUTPUT);
  pinMode(echoPinExit, INPUT);
  myservo.attach(SERVO_PIN);
  myservo.write(100); // Initial gate closed
  lcd.setCursor(0, 0);
  lcd.print(" ARDUINO PARKING ");
  lcd.setCursor(0, 1);
  lcd.print("   SYSTEM READY  ");
  delay(2000);
  lcd.clear();
}
```

```
void loop(){ updateAvailableSlots();
 displaySlots();
  int entryDistance = getDistance(trigPinEntry,
  echoPinEntry); int exitDistance =
  getDistance(trigPinExit, echoPinExit);
  Serial.print("Entry Distance: ");
  Serial.print(entryDistance);
  Serial.print(" cm, Exit
  Distance: ");
  Serial.println(exitDistance);
  // Entry logic — only if distance is between 2cm and
  4cm if (entryDistance >= 2 && entryDistance <=
  4) {
    if (availableSlots > 0)
      { openGate();
      delay(500); // Wait for vehicle to pass
      updateAvailableSlots(); // Re-check slot sensors
      after entry
    } else
      { lcd.clear()
      ;
      lcd.setCursor(0, 0);


      lcd.print("  LOT IS FULL
      "); delay(1000);
    }
  }

  // Exit logic — only if distance is between 2cm
  and 4cm if (exitDistance >= 2 && exitDistance
```

```
  <= 4)
  { openGate();
    delay(500); // Wait for vehicle to leave
    updateAvailableSlots(); // Re-check slot sensors
    after exit
  }
  delay(500);
}

int getDistance(int trigPin, int echoPin)
  { digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  long duration = pulseIn(echoPin, HIGH, 30000); // Optional
  timeout if (duration == 0) return -1; // No echo detected
  int distance = duration * 0.034 / 2; // Convert to
  cm return distance;
}

void updateAvailableSlots() {
  i n t          s 1          =
  digitalRead(IR_SLOT1);
  i n t          s 2          =
  digitalRead(IR_SLOT2);
  i n t          s 3          =
  digitalRead(IR_SLOT3);
  i n t          s 4          =
```

```
  digitalRead(IR_SLOT4);
  availableSlots = 0;
  if ( s 1 == H I G H )
  availableSlots++; if (s2 ==
  HIGH) availableSlots++; if
  ( s 3 = = H I G H )
  availableSlots++; if(s4 ==
  HIGH) availableSlots++;
}

void openGate()
  { myservo.write(0); //
  Open gate delay(3000); //
  Time to pass
  myservo.write(100); //
  Close gate delay(500);
}
void
displaySlots() {
lcd.clear();
 lcd.setCursor(0, 0);
  lcd.print(" Parking Monitor
  "); lcd.setCursor(0, 1);
  lcd.print("Slots Left: ");
  lcd.print(availableSlots);
}
```

5. Results and Discussion

The IoT-based Smart Parking System was successfully implemented using ultrasonic sensors for vehicle detection, IR sensors for slot monitoring, a servo motor for gate control, and an LCD display for real-time slot availability. The system was designed to

run independently on an Arduino, using local components for monitoring and display, without relying on external cloud platforms.

Key Results:

1. **Slot Detection Accuracy:**
    - The IR sensors accurately detected the presence or absence of vehicles in each of the 4 parking slots.
    - Sensor readings were reliable under normal lighting and indoor conditions.
2. **Distance Measurement:**
    - The ultrasonic sensors measured distances at the entry and exit gates to detect incoming and outgoing vehicles.
    - Vehicles within the threshold range (2–4 cm) triggered the gate mechanism.
3. **Gate Automation:**
    - The servo motor gate opened only when a vehicle was detected and parking space was available.
    - The gate remained closed when the lot was full, preventing unauthorized entry.
4. **Real-Time Display:**
    - The LCD module updated the number of available slots in real time.
    - Display messages like "SLOTS LEFT: X" and "LOT IS FULL" improved user interaction.

6. CONCLUSION

The proposed IoT-based Smart Parking System effectively addresses the challenges of traditional parking management by providing a **low-cost**, **automated**, and **real-time**

solution. Using **Arduino**, **IR sensors**, **ultrasonic sensors**, **servo motors**, and **LCD display**, the system detects vehicle presence, monitors slot availability, controls gate access, and displays the current status to users.

This standalone Arduino-based system provides a compact and scalable solution for small- scale parking management, with potential for future cloud integration if needed. The prototype successfully demonstrates how IoT technology can be used to improve parking efficiency, reduce traffic congestion, and enhance user convenience.

The system is **modular**, **scalable**, and can be upgraded with features like mobile app control, automated billing, and advanced data analytics for future implementation.

7. References

1. Arduino Official Documentation. *Arduino.cc*. Retrieved from: https:// www.arduino.cc/ en/Guide

2. HC-SR04 Ultrasonic Sensor Datasheet. Retrieved from: https:// components101.com/ sensors/hc-sr04-ultrasonic-sensor

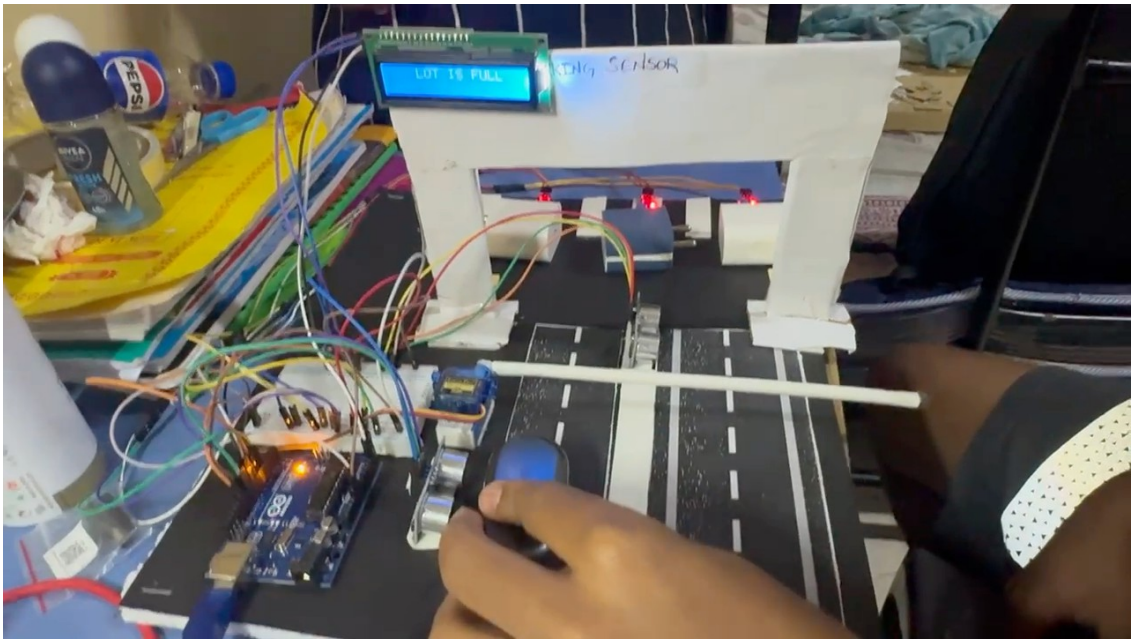
3. IR Sensor Working and Applications. *Electronics Hub*. Retrieved from: https:// www.electronicshub.org/ir-sensor/

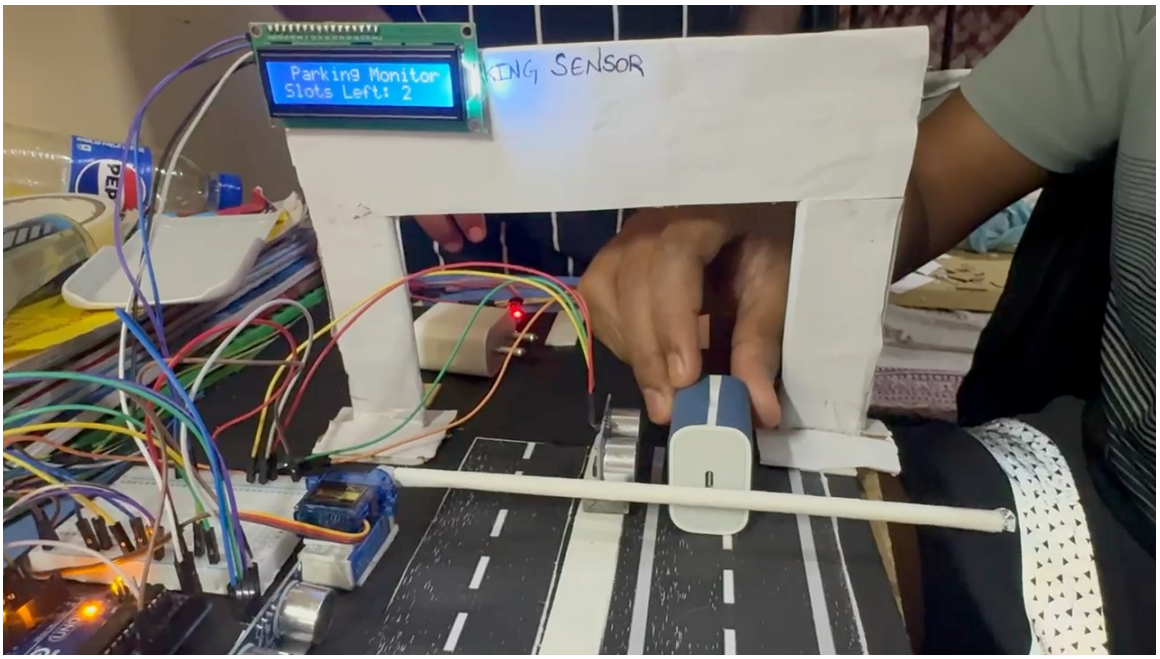
4. Servo Motor Basics. *SparkFun Electronics*. Retrieved from: https:// learn.sparkfun.com/tutorials/servo-motors/all

5. Chien, S., Ding, Y., & Wei, C. (2015). *Parking space detection and management using IoT technology*. International Journal of Smart Home, 9(1), 35-50.

6. Kumar, R., & Mallick, B. (2018). *The Role of IoT in Smart Cities: A Review*. *International Journal of Engineering Research & Technology (IJERT)*, 7(11), 1-4.

IMAGES





VIDEO LINK 1 : https://drive.google.com/file/d/1vgclOCBA7sqpFRF-jaTt2hZ6KnRQyCpL/view?usp=drivesdk

VIDEO LINK 2: https://drive.google.com/file/d/1Vyd6dA-4ee-8wsq4hN41biQL0UL4cT6c/view?usp=drivesdk