Military Institute of Science and Technology

DEPARTMENT OF ELECTRICAL ELECTRONIC AND COMMUNICATION ENGINEERING

COURSE: EECE 304

DIGITAL ELECTRONICS LABORATORY

LAB PROJECT REPORT

GROUP NO: 01

Project Title: Intelligent Car Parking System & Billing Framework

Submitted By:

(Name)	(Student ID)
Rabeya Bushra Sujana Badhan	202116191
Pretom Das	202216072
Samiha Tabassum Mehreen	202216083
Tanvir Ahammed Rahat	202216103
Zuairah Zakir Sarah	202216113
Kazi Suraiya Khanum	202216212
Md Rizit Hossain Rubai	202116166

Course Instructors:

i. Prof. Dr. Md. Golam Mostofa
ii. Lec. Lamiya Mahbub
iii. Lec. Moumita Datta
iv. Lec. Nishat Salsabil
v. Lec. Raihan Habib Himel
vi. Lec. Tahmina Tabassum Trina

Table of Contents

Abstract

The project focuses on an intelligent car parking system designed to ease the challenge of finding parking in crowded urban areas. The system provides real-time parking availability, allowing drivers to quickly locate available spots, reducing frustration and traffic congestion. Additionally, our smart parking system includes an automatic bill generation feature, making the process more efficient by calculating parking fees based on the duration of stay and providing the bill to drivers seamlessly. This solution aims to improve the overall parking experience while addressing the issues of time wastage and increased traffic.

1. Introduction

The intelligent car parking system is designed to streamline the process of parking management in urban areas. It uses logic gates and sensors to automate the detection of parked vehicles, display available spots, and generate automatic billing based on the duration of the stay. The system provides a real-time update of parking status using LED indicators and a seven-segment display, ensuring a seamless parking experience for drivers.

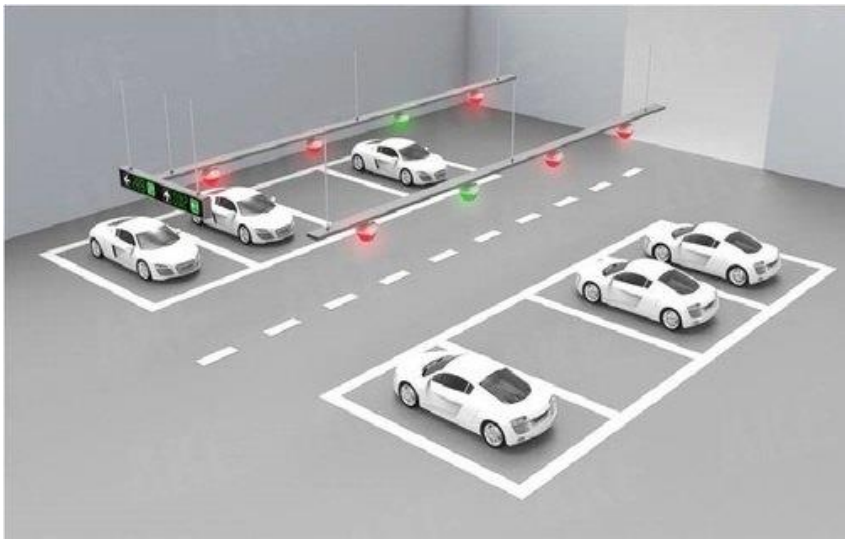


Fig-01: 2D model a smart car parking system

1.1 Theory Overview

The intelligent car parking system presented here is an innovative approach to addressing common issues in urban parking, such as space congestion, time wastage, and inefficient manual operations.

This system automates the entire process, from detecting the presence of cars in parking spaces to displaying available spots and generating automatic billing based on the vehicle's duration of stay. The system utilizes logic gates, IR sensors, and layered circuits to provide real-time parking management. Additionally, an **Arduino microcontroller** is integrated to handle automatic bill generation, calculating parking fees based on the duration of a vehicle's stay and providing a seamless billing process. Furthermore, **Verilog** is employed for hardware description and design, enabling efficient implementation of digital circuits that form the backbone of the parking management system. This integration of modern technologies not only enhances the user experience but also helps reduce traffic congestion and the time spent searching for parking.

1.2 System Architecture and Key elements

The intelligent car parking system consists of various hardware components, including logic gates, sensors, and an Arduino, that work together to offer seamless parking management. Below are the primary components and their roles:

1. **Infrared (IR) Sensors**: Each parking spot is equipped with IR sensors that detect the presence of a vehicle. The sensor outputs a signal—0 if a car is present and 1 if the spot is empty. This binary data is crucial for monitoring the parking space's availability in real-time.

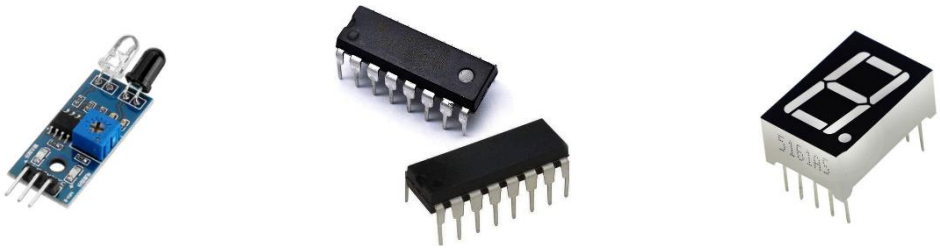


Fig-02: Some key elements

2. **Layered Circuit Design**:

    - **First Layer (Half-Adder)**: This layer sums the outputs from the 8 IR sensors, producing four 2-bit numbers that serve as the base data for further processing
    - **Second Layer (4-Bit Adder)**: Using dual-input 4-bit adder ICs (74LS83), this layer adds the 4-bit numbers from the previous stage to generate a simplified representation of the occupied spots
    - **Third Layer (4-Bit Adder)**: Another 4-bit adder further condenses the data, outputting a single 4-bit number representing the number of parked cars
    - **Fourth Layer (Subtractor Circuit)**: This layer calculates the number of free parking spots by subtracting the number of occupied spots from the total

available spots using a subtractor circuit built with NOT gates and 4-bit adder ICs.

- **Fifth Layer (BCD to 7-Segment Decoder)**: The processed data is converted into a human-readable format by a BCD to 7-segment decoder, which displays the number of free spots on a seven-segment display

3. **Arduino for Bill Generation**:

   An Arduino microcontroller automates the billing process by tracking the duration a car remains parked. Upon entry, the Arduino records the time, and upon exit, it calculates the parking fee based on a predefined rate. It interfaces with a real-time clock (RTC) module for accurate time tracking and uses a display or printer to provide the bill, ensuring a fully automated billing system.

2. Circuit Diagram

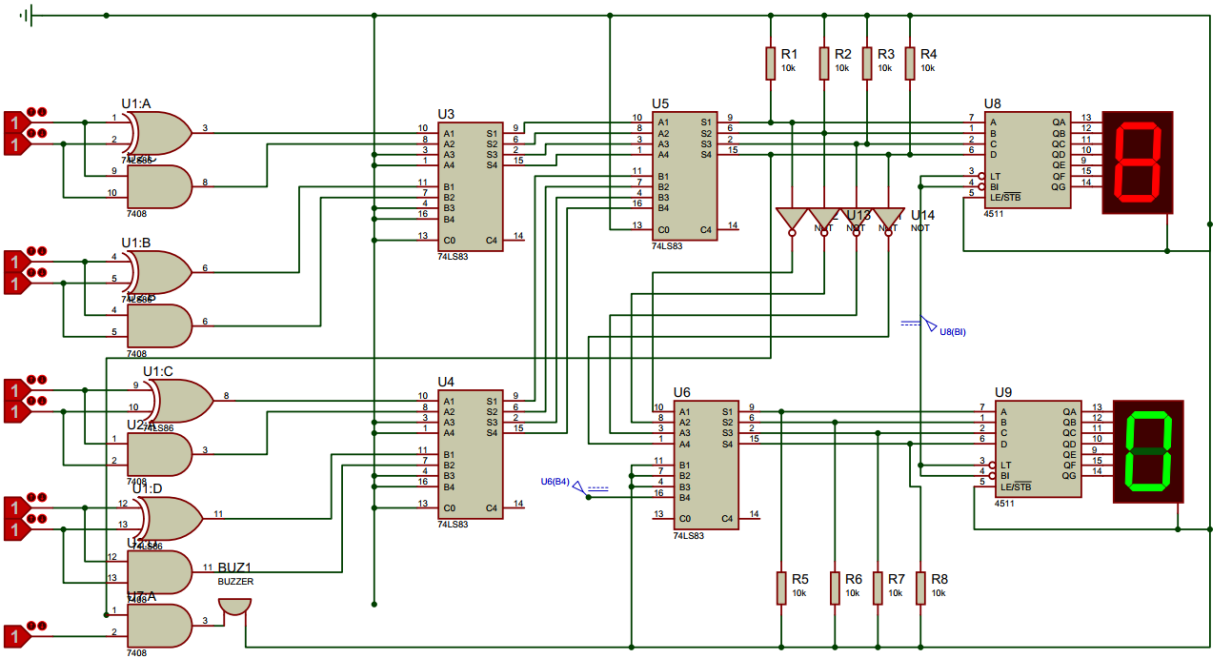


Fig-03: Circuit Diagram of Car Counting System

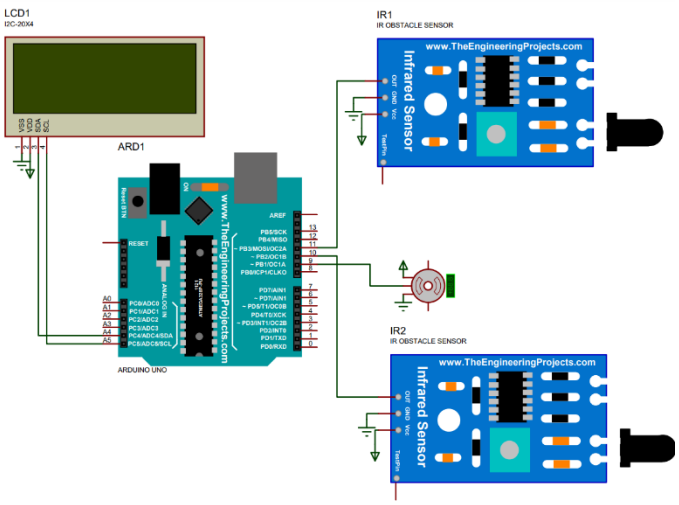


Fig-04: Schematic of bill generation with Arduino

3. Project Components List

In order to design and implement the intelligent parking system following components were used and connected as per the circuit diagram provided in **figure.03 and figure.04**

Table-01: List of Components

Component	Quantity
Dual Input XOR Gate (74LS86)	2
Dual Input AND Gate (74LS08)	2
Dual Input 4-bit Adder (74LS83)	4
7 Segment Display (Common Cathode)	2
NOT Gate (74LS04)	2
BCD to 7 Segment Decoder (CD4511)	2
10k Resistors	8
IR Sensors	9
Arduino UNO Rev3	1
Servo Motor	1
Buzzer	1
LED	9
Jumper Wires	1
Breadboard	3
Voltage Supply Module	1
3.7 Volt Battery	2

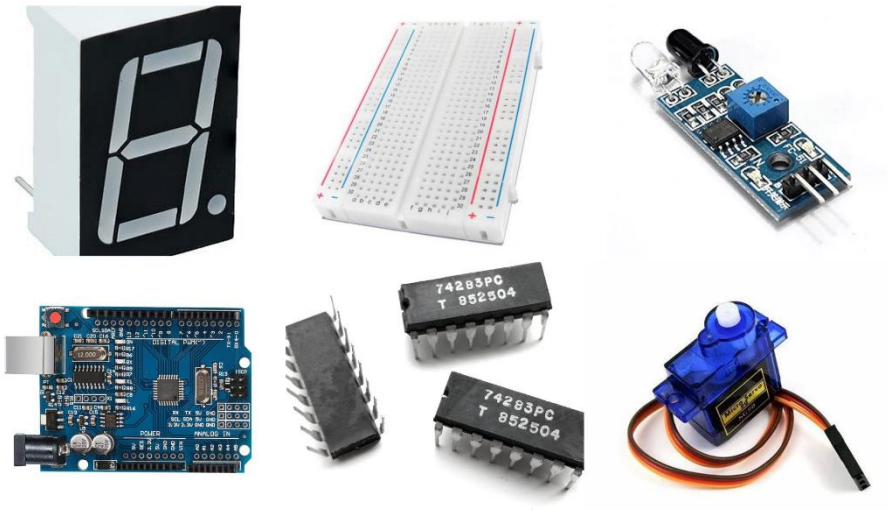


Fig-05: Required Components for the construction of the system

4. Code Section

Arduino codes

- **Adding Library and Pins initiating**

```
#include <Wire.h> // Required for I2C communication
#include <LCD_I2C.h>
LCD_I2C lcd(0x27);
#include <Servo.h>

Servo myServo; // Create a servo object
const int servoPin = 9; // Pin connected to the servo
const int bill_ir = 11; // ir responsible for bill generation,
const int ex_ir = 10; // for external IR sensor
const int max = 8; // if the capacity is full
unsigned long startTime = 0; // To store start time
unsigned long elapsedTime = 0; // Elapsed time in milliseconds
float billAmount = 0.0; // Bill amount (initialized to 0)
```

- **Pin mode and Display setup**

```
void setup() {
  pinMode(bill_ir, INPUT); // Set pin 11 as an input
  pinMode(servoPin, OUTPUT); // Set pin 9 as an input for servo
  pinMode(ex_ir, INPUT); // Set pin 11 as an input
  pinMode(max, INPUT); // Set pin 8 as an input
  lcd.begin();
  lcd.backlight();
  lcd.setCursor(4, 0);
  lcd.print("Group-1");
  myServo.attach(servoPin);
  myServo.write(0); // Set initial position (0 degrees)

}
```

- **Entry gate logic (Servo motor coding)**

```
void servo () {
  int state1 = digitalRead(ex_ir); // Read the state of pin 9 for
servo ir
  int state_max = digitalRead(max);
  // red is vcc, brown is gnd and orange is output
// for servo motor
  if (state1 == HIGH) {
    // Pin 9 is high, move servo 90 degrees clockwise
    myServo.write(90);
  } else {
    // Pin 9 is low, return servo to initial position
    myServo.write(0);
  }
  delay(1000);
}
```

- **Parking time calculation and Bill Generation**

```
void loop () {
  servo();
  int state = digitalRead(bill_ir); // Read the state of pin 11
```

```
// for bill generation
  if (state == HIGH) {
    // Pin 11 is high, start the timer
    startTime = millis();
  } else {
    // Pin 11 is low, stop the timer
    elapsedTime = millis() - startTime;
    billAmount = 0.01667 * (elapsedTime / 1000.0);


    // Display elapsed time and bill amount
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Time(s): ");
    lcd.setCursor(0, 1);
    lcd.print(elapsedTime / 1000.0, 2); // Print elapsed time with 2
decimal places

    lcd.setCursor(9, 0);
    lcd.print("Bill:");
    lcd.setCursor(9, 1);

    lcd.print(billAmount, 2); // Print bill amount with 2 decimal
places
    billAmount = 0.0; // Reset bill amount
  }
}
```

Verilog Codes

- **Car parking adder logic**

```
module Parking_System(
    input [7:0] car,
    output [6:0] car_count_display,
    output [6:0] empty_space_display
);

    wire [3:0] car_count;
    wire [3:0] empty_spaces;
```

```
    // Sum the car inputs using an adder
    assign car_count = car[0] + car[1] + car[2] + car[3] + car[4] +
car[5] + car[6] + car[7];

    // Calculate empty spaces (8 - car_count)
    assign empty_spaces = 4'b1000 - car_count;

    // Instantiate 7-segment display drivers for car count and empty
spaces
    SevenSegDriver display1 (.binary_in(car_count),
.seg_out(car_count_display));
    SevenSegDriver display2 (.binary_in(empty_spaces),
.seg_out(empty_space_display));

endmodule
```

- **7 segment displays**

```
// 7-segment display driver module (BCD to 7-segment)
module SevenSegDriver(
    input [3:0] binary_in,
    output reg [6:0] seg_out
);
    always @(*) begin
        case (binary_in)
            4'b0000: seg_out = 7'b1000000; // 0
            4'b0001: seg_out = 7'b1111001; // 1
            4'b0010: seg_out = 7'b0100100; // 2
            4'b0011: seg_out = 7'b0110000; // 3
            4'b0100: seg_out = 7'b0011001; // 4
            4'b0101: seg_out = 7'b0010010; // 5
            4'b0110: seg_out = 7'b0000010; // 6
            4'b0111: seg_out = 7'b1111000; // 7
            4'b1000: seg_out = 7'b0000000; // 8
            default: seg_out = 7'b1111111; // Blank/Error
        endcase
    end
```

```
endmodule
```

The **Arduino** section manages hardware control, including sensors, a servo motor for the entry gate, and an LCD that displays parking time and billing information. It uses infrared sensors to detect car entry and exit, calculating the parked time and generating bills based on elapsed time. This part controls the physical components and real-time data display. Meanwhile, the **Verilog** section handles the logic for counting cars and displaying available parking spaces on a 7-segment display. It processes binary inputs to track the number of cars and ensure accurate parking availability updates. Together, both sections provide an integrated system for automated parking management and billing.

5. Methodology

5.1) Working Principle

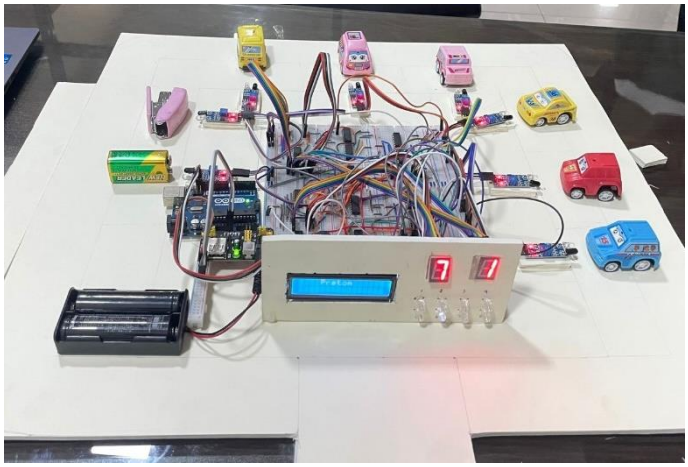


Fig-06: Practical demostration of the intelligent parking system

The working principle of the intelligent car parking system involves real-time detection of vehicle occupancy using IR sensors, layered logic circuits for spot calculation, and automated billing and display systems for seamless parking management.

a. Sensor-based Detection: The system uses infrared (IR) sensors to detect the presence of vehicles in each parking spot. Each sensor sends a binary signal: 0 when a car is present, and 1 when the spot is empty.

b. Layered Circuit Design: The signals from the sensors are processed through multiple layers of logic gates and adders.

Layer 1: A half-adder adds the binary outputs from the sensors, generating the total number of occupied spots.
Layer 2: A 4-bit adder further processes the data, condensing the information into manageable binary numbers.
Layer 3: Another 4-bit adder finalizes the total count of occupied spots.

c. Subtraction for Available Spots: The number of available parking spots is calculated by subtracting (Layer-4) the occupied spots from the total number of spots. This is done through a subtractor circuit, which uses NOT gates and adders.

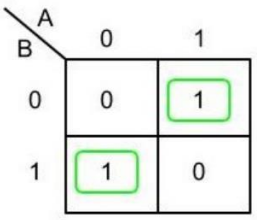
d. Display of Available Spots: The available spots are displayed on a seven-segment display via a Binary-Coded Decimal (BCD) to seven-segment decoder (Layer-5). The system updates this display in real-time.

e. Automated Billing: An Arduino microcontroller handles billing by tracking the time a vehicle remains parked. IR sensors at the entrance and exit gates trigger time recording, and the Arduino calculates the total parking fee. The fee is displayed on an LCD screen or printed for the user.

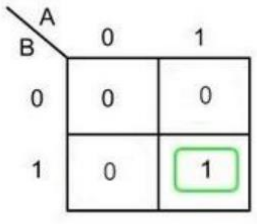
f. Gate Operation: A servo motor, controlled by the Arduino, operates the entry and exit gates. When a vehicle is detected by the IR sensor at the gate, the servo motor moves to open or close the gate.

5.2) 1st Layer: Half-adder

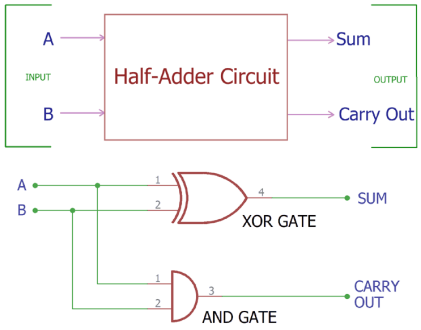
A half adder is a basic combinational circuit that performs binary addition of two single-bit inputs, A and B, producing two outputs: SUM and CARRY. The SUM is the XOR of A and B, representing the least significant bit (LSB) of the result, while the CARRY, which is the AND of A and B, indicates any carry-over from the addition. The half adder requires one XOR gate for the SUM and one AND gate for the CARRY, making it the simplest building block for more complex adder circuits.



For Sum, **S=A XOR B**



For Carry, **C=A AND B**



A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Fig-07: Half-adder circuit, Boolean expressions and truth table

- **Role of 1st Layer (Half Adder) in the Intelligent Car Parking System-**

- The half adder is utilized in the **first layer** of the circuit.

- It sums the **binary outputs** from the IR sensors, which detect whether a parking spot is occupied (1 for free, 0 for occupied).

- The half adder **adds the signals** from the 8 parking spots to produce a sum that represents the total number of occupied spots.

- This sum is processed by subsequent layers of adders and subtractors to **calculate the number of free spots**.

- The half adder plays a **crucial role** in combining the sensor outputs before passing the data to the next stages for further `processing`.

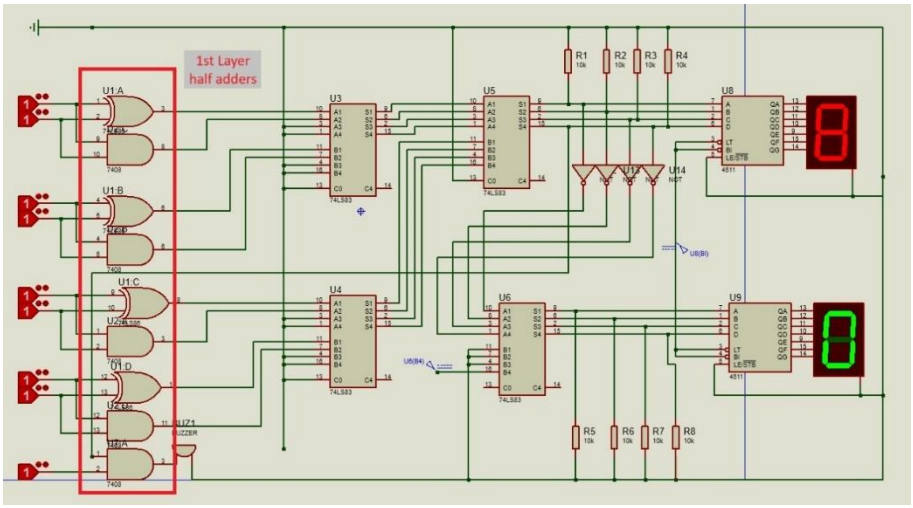
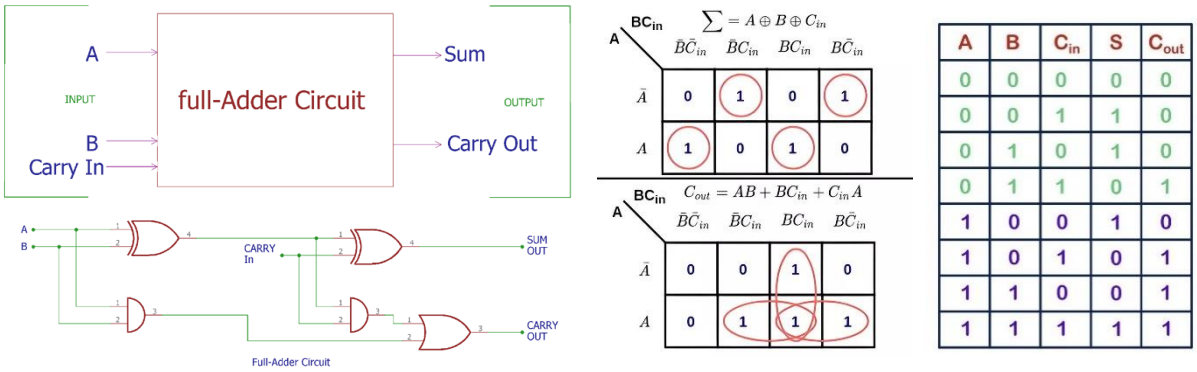


Fig-08: 1st Layer

5.3) 2nd Layer: 4-bit Adder-01

A full adder takes three inputs—A, B, and a carry-in (C-IN)—and generates two outputs: SUM (S) and carry-out (C-OUT). The carry-out signals when more than one input is high. Full adders are necessary when a carry-in bit is involved, as they allow the carry bit to propagate between adders, which half adders cannot handle. This makes full adders crucial for adding multi-bit binary values.



A	B	C_{in}	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Fig-09: Full-adder circuit, Boolean expressions and truth table

- **Role of the 2nd Layer (Adder-01) in the Intelligent Car Parking System-**

- The **second layer** uses a **dual-input 4-bit adder (IC 74LS83)** to process the binary data from the first layer.

- It combines the **four 2-bit numbers** generated in the first layer, representing the parking spot status.

- The full adder condenses the results into **two 4-bit numbers**, making the data more manageable for further processing.

- This layer reduces **data complexity**, preparing it for the next stage of the system.

- By handling the **carry-in** from previous calculations, the full adder ensures accurate computation of occupied parking spots.

- The output from this layer is critical for determining the **total number of occupied spots** in the parking system.

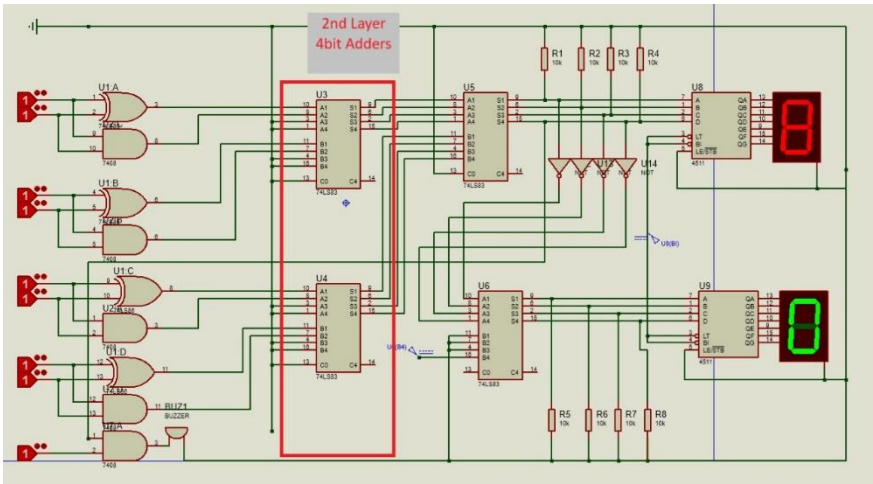


Fig-10: 2nd Layer

5.3) 3rd Layer: 4-bit Adder-02

- **Role of the 3rd Layer in the Intelligent Car Parking System**

- The **third layer** utilizes another **dual-input 4-bit adder (IC 74LS83)** to further process the output from the second layer.

- It adds the **two 4-bit numbers** obtained from the previous layer, consolidating the data for final calculations.

- This layer produces a **single 4-bit output** that represents the total count of occupied parking spots in the system.

- By efficiently managing carry bits from earlier additions, the third layer enhances the accuracy of the overall computation.

- The output from this layer is essential for the subsequent layer, as it feeds into the subtractor circuit to calculate the number of free parking spots.

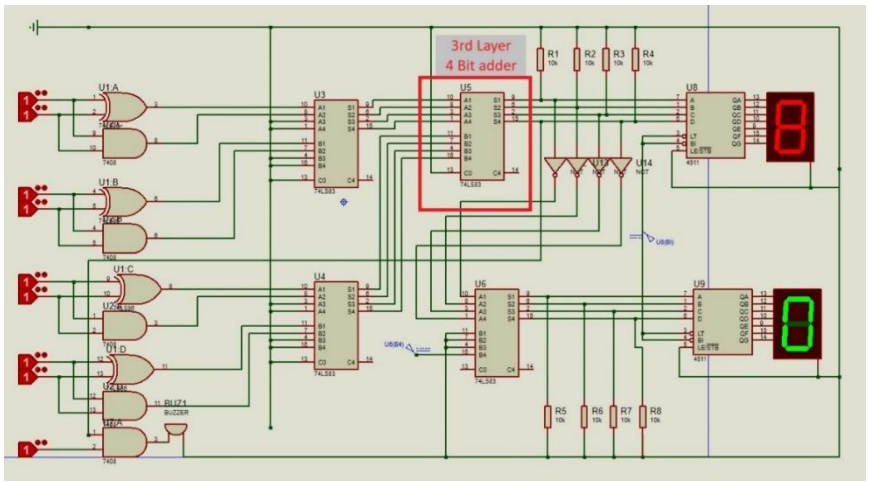
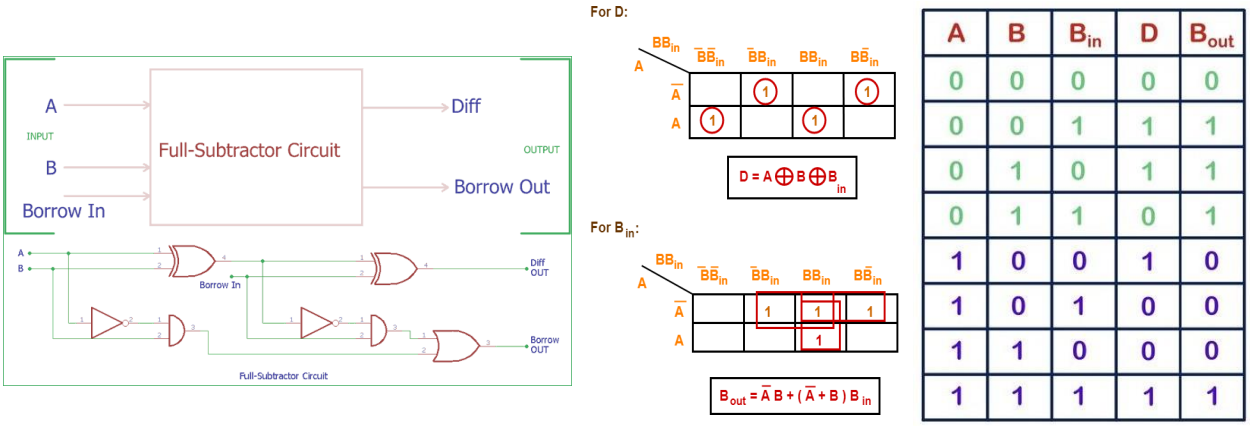


Fig-11: 3rd Layer

5.4) 4th Layer: Subtractor

A full subtractor is a combinational circuit that subtracts two bits, factoring in a previous borrow. It has three inputs: A (minuend), B (subtrahend), and Bin (borrow in), and two outputs: D (difference) and Bout (borrow out). Although subtraction can be done by adding the subtrahend's complement, it's useful to examine the truth table and logic for direct subtraction.



A	B	B_{in}	D	B_{out}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Fig-12: Full-subtractor circuit, Boolean expressions and truth table

- **Role of the 4th Layer in the Intelligent Car Parking System**

- The **fourth layer** consists of a **subtractor circuit** using a **dual-input 4-bit adder (IC 74LS83)** and a **NOT gate.**

- This layer subtracts the total number of occupied parking spots (from the third layer) from the total available spots (8).

- The result is a **single 4-bit output** representing the number of free parking spots in the system.

- The subtractor circuit ensures the system accurately calculates and displays the available spaces in real-time.

- The output of this layer is crucial for the final display, as it provides the number of available spots to drivers.

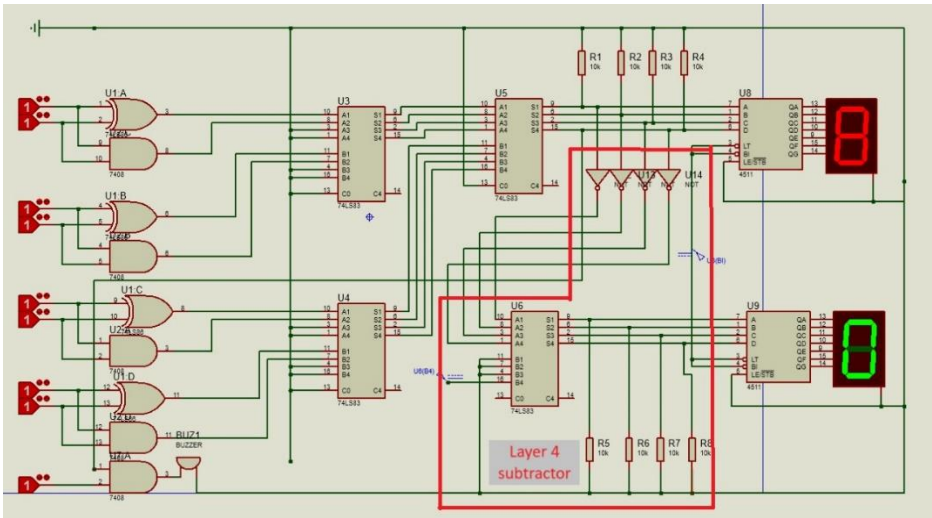


Fig-13: 4th Layer

5.5) 5th Layer: BCD to 7-Segment Decoder

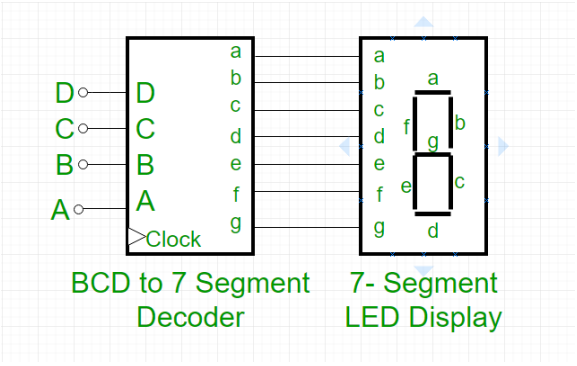
Binary Coded Decimal (BCD) is a 4-bit encoding scheme where each decimal digit (0-9) is represented by its binary equivalent.

A seven-segment display uses seven LEDs arranged in a pattern to display decimal digits (0-9) using BCD input. There are two types of displays:

- Common Cathode: All LED cathodes are connected to ground, and digits are displayed when a 'HIGH' signal is applied to the anodes.

- Common Anode: All LED anodes are connected to +Vcc, and digits are displayed when a 'LOW' signal is applied to the cathodes.

A BCD to seven-segment decoder converts the 4-bit BCD input (A, B, C, D) into signals for the seven LED segments (a, b, c, d, e, f, g), which then display the corresponding decimal digit.



A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1

Fig-14: BCD to 7-Segment Decoder block diagram and truth table

- **Role of the 5th Layer in the Intelligent Car Parking System**

- The **fifth layer** employs a **BCD to 7-segment decoder (IC 4511)** to convert the binary output from the subtractor in the fourth layer into a displayable format.

- It processes the **4-bit binary output** that represents the number of available parking spots.

- The IC drives the **7-segment display,** allowing it to show the number of free parking spaces in a human-readable format.

- This layer ensures that the parking information is presented clearly to drivers in real time.

- The **BCD to 7-segment decoder IC** plays a key role in translating the system's calculations into a visible and understandable form.

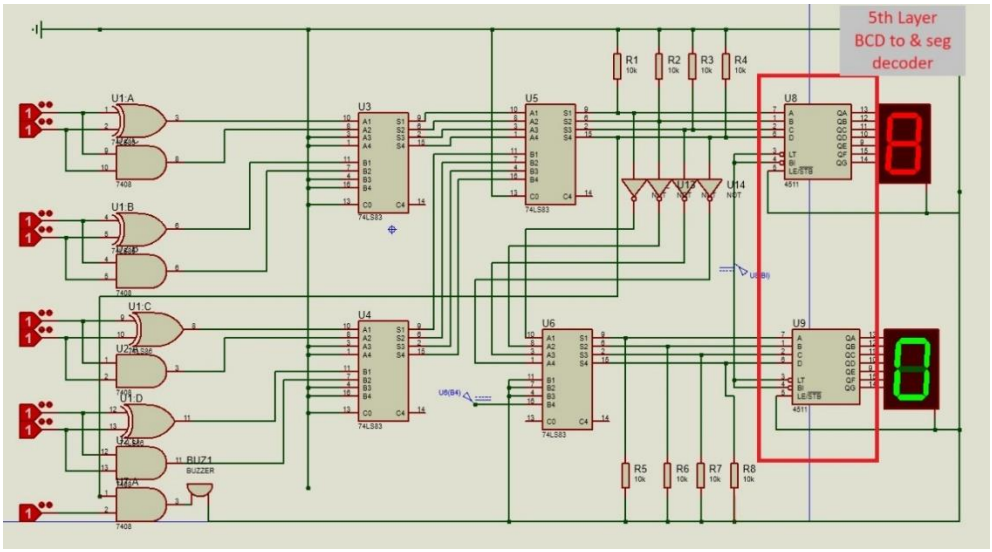


Fig-15: 5th Layer

5.6) Arduino-Based Billing System, LCD Display, and Servo Control Framework

Based on the Arduino code provided in the document, here's how the Arduino contributes to creating the billing structure, controlling the LCD display, and functioning the servo motor:

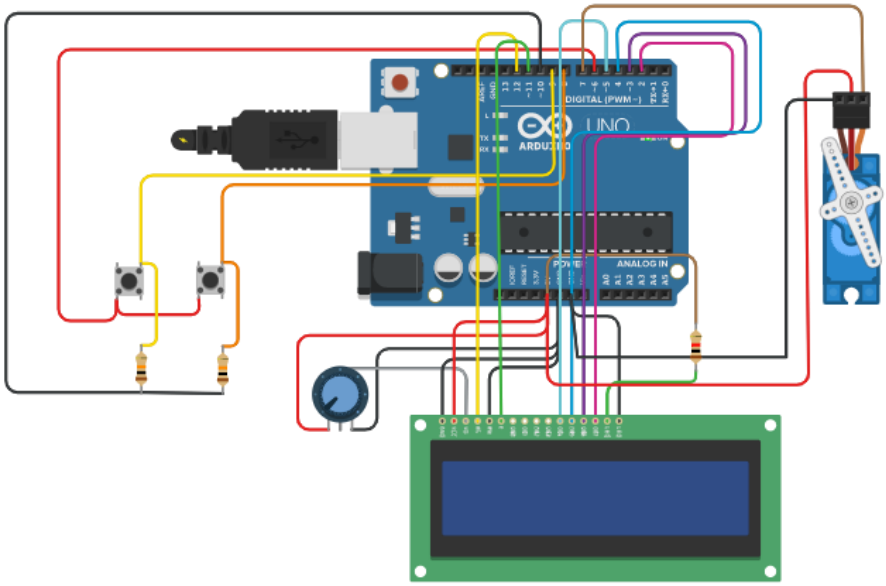


Fig-16: Connection Diagram of Arduino-Based Billing System with LCD Display and Servo Motor

a) Billing Structure-

- IR Sensor (Pin 11): The system uses an IR sensor to start the billing process when a car is detected.

- Timer Start and Stop: The billing begins as soon as the IR sensor detects an object and ends when the object is no longer detected.

- Time Calculation: The elapsed time is calculated using the `**millis ()** ` function, which measures the total parking time in milliseconds.

- Billing Rate: A formula calculates the bill amount based on the time parked'

$$\boldsymbol{billAmount = 0.1667 \times (\frac{elapsedTime}{1000})}$$

This formula multiplies the parking time in seconds by a fixed rate (0.01667).

- Display on LCD: The total time and bill amount are displayed on the LCD after the vehicle exits.

b) LCD Display-

- Library Integration: The code includes the `**Wire.h**` and `**LCD_I2C.h**` libraries to control the LCD via I2C communication.

- Setup: The LCD is initialized in the `setup () ` function using `**lcd.begin()**` and the backlight is enabled with `**lcd.backlight()**`.

- Information Display: The LCD shows two lines of information:

- The elapsed parking time (in seconds).

- The total bill amount, updated in real-time as the parking duration increases.

- Cursor Positioning: The `**lcd.setCursor()**` function is used to position the displayed content on the screen.

c) Servo Motor Functionality-

- Servo Library: The `**Servo.h**` library controls the servo motor.

- IR Sensor Activation (Pin 10): An external IR sensor detects when a car is present at the entry gate, which triggers the servo motor.

- Servo Movement: When a car is detected, the servo motor moves to 90 degrees (opening the gate). After a delay or when the car passes, the motor returns to 0 degrees (closing the gate).

- Servo Pin: The servo is connected to Pin 9, which controls its operation based on the sensor input.

This system integrates sensors, an LCD display, and a servo motor to manage parking, calculate bills, and handle gate operations.

5.7) Verilog Code Functionality in the Intelligent Car Parking System

The Verilog code in the intelligent car parking system focuses on managing car count and displaying available parking spaces using a 7-segment display. It utilizes an 8-bit input to track the number of cars, with each bit representing a car in a parking slot. An adder logic sums the occupied slots and calculates the available spaces by subtracting the car count from the total parking capacity (8). The results are then fed to two 7-segment display drivers: one showing the current car count and the other showing the available empty spaces. This system provides real-time updates on parking availability. Basically, the Verilog functions as-

- Purpose: Manages car count and displays available parking spaces.

- Input: Uses an 8-bit input to represent the presence of cars in parking slots.

- Car Count Calculation: An adder logic sums the occupied slots.

- Available Spaces Calculation: Computes empty spaces by subtracting the car count from the total capacity (8).

- Display Outputs: Utilizes two 7-segment display drivers,

- One driver shows the current number of parked cars.

- The other displays the number of available parking spaces.

- **Real-Time Updates**: Provides continuous monitoring of parking availability.

6. Project Flow Diagram

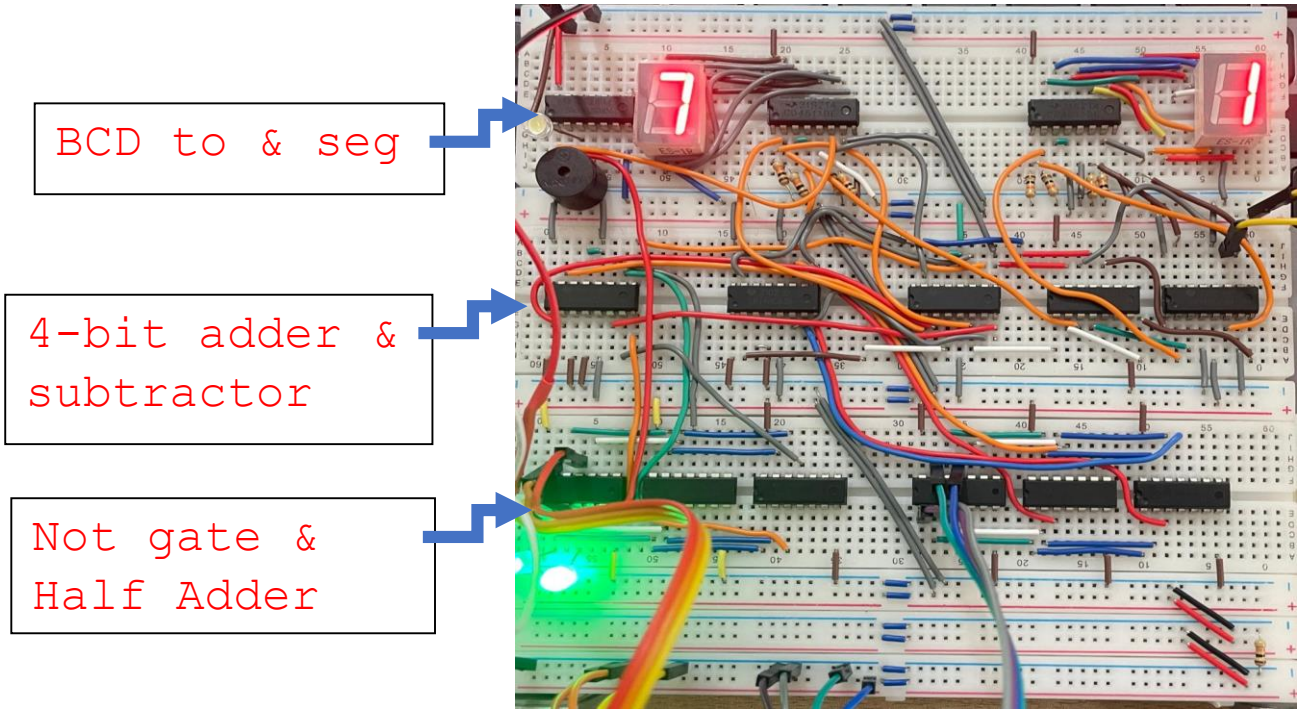


Fig-17: Layered Circuit Diagram

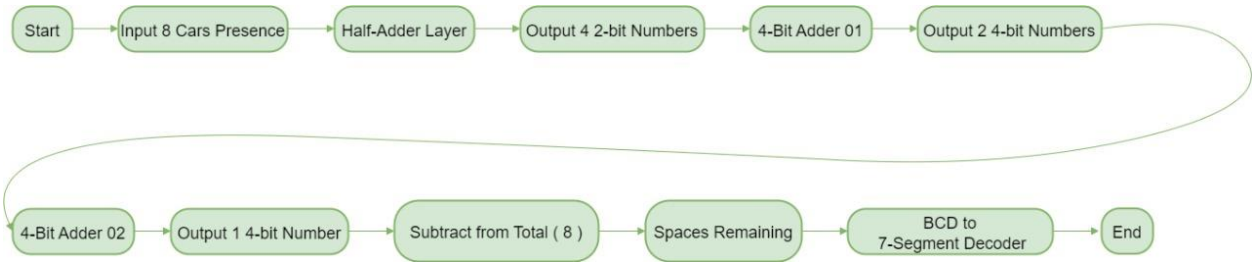


Fig-18: Block Diagram Layered Circuit

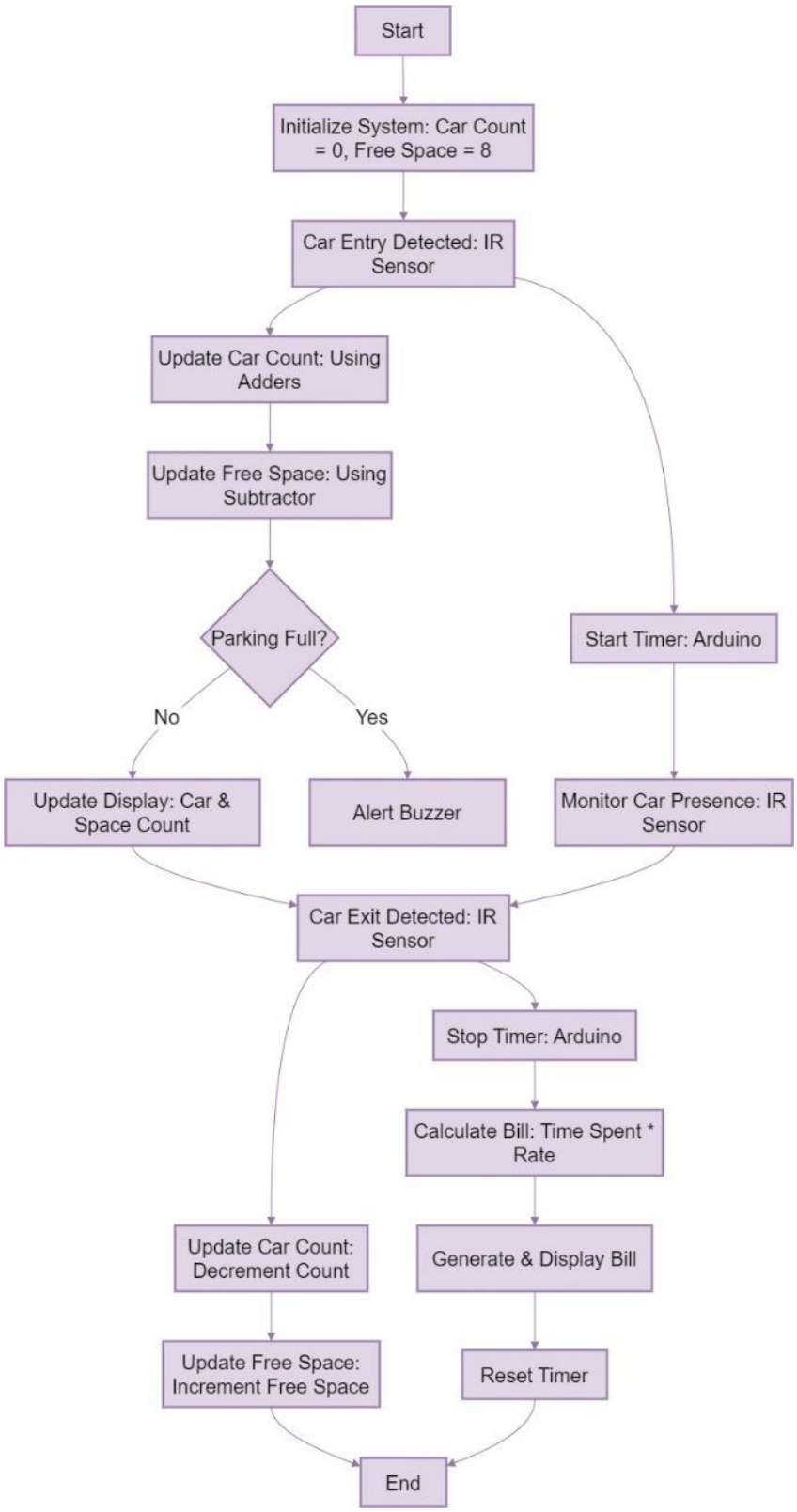


Fig-19: Block diagram for the intelligent car parking system and billing framework

7. Result and Findings

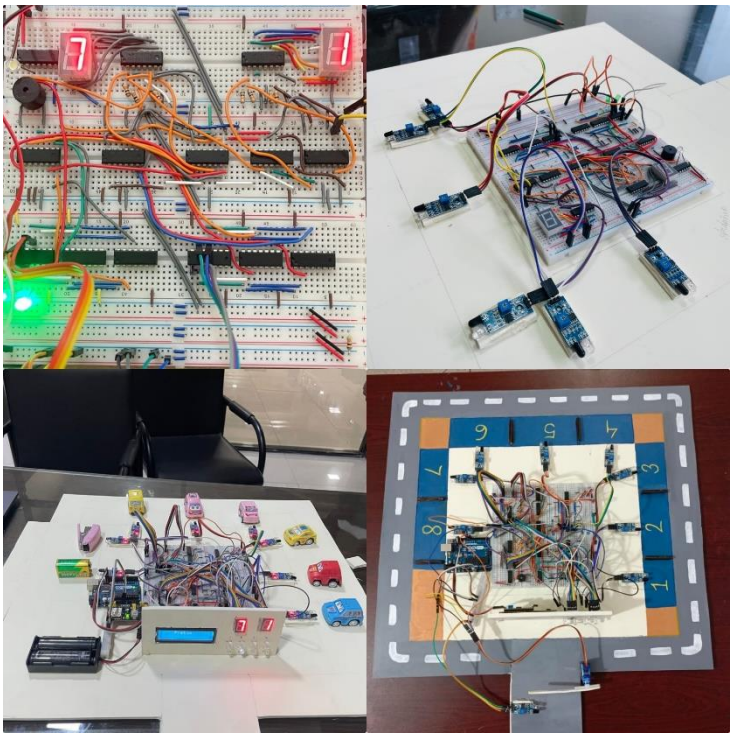


Fig-20: Initial to final phase of the constructed intelligent car parking system

Positive Impact and Benefits of the Intelligent Car Parking System-

The intelligent car parking system provides significant benefits in managing parking spaces efficiently. Real-time detection of vehicle presence through IR sensors and immediate updates on the availability of spots streamline the parking process, reducing the time drivers spend searching for a space. The system's automated billing mechanism, using an Arduino, enhances user convenience by calculating parking fees based on the time spent without manual intervention. Additionally, the smooth operation of the automated gates ensures a seamless entry and exit experience for users, minimizing congestion and improving the overall flow of traffic within parking facilities.

Why We Used the Half-Adder Layer First? -

The decision to use a half-adder layer at the beginning of the circuit design is driven by the need to process single-bit inputs efficiently. Each parking spot is monitored by an IR sensor, which provides binary data (1 for empty, 0 for occupied). A half-adder is well-suited for this stage because it combines two single-bit inputs, simplifying the initial data processing. Using a 4-bit adder at this stage would introduce unnecessary complexity, as it is designed for multi-bit inputs. The half-

adder layer allows for a more modular and manageable system, breaking down the problem into smaller steps before moving on to more complex operations with multi-bit adders in later stages.

Limitations of the System -

Despite its advantages, the intelligent car parking system has some limitations. The system's scalability is restricted, as expanding to larger parking lots would require significant changes in both hardware and software. Additionally, the accuracy of the system heavily relies on the proper functioning of IR sensors, which can be affected by environmental factors such as dust or poor alignment. Maintenance is also a concern, as the sensors and Arduino must be regularly checked to ensure the system operates smoothly. Lastly, the initial setup cost can be relatively high due to the number of components required, which may limit its adoption in certain contexts.

Practical Application of Learned Concepts-

In our EECE-304 lab course, we conducted experiments involving various digital electronic components and concepts, including Verilog programming, half-adders, full adders, subtractors, and BCD to 7-segment decoders. These hands-on experiences provided us with a solid understanding of digital logic design and circuit implementation. The knowledge gained from these experiments has been directly applied in the intelligent car parking system, where we utilized integrated circuits (ICs) and other digital electronic elements to create a functional and efficient parking management solution. This practical application not only reinforced our theoretical knowledge but also allowed us to develop a comprehensive system that integrates multiple digital components seamlessly.

8. Real-life Implications

The intelligent car parking system significantly impacts urban mobility by reducing traffic congestion and enhancing the user experience. By providing real-time information on parking availability, it minimizes the time drivers spend searching for spots, leading to a smoother traffic flow and reduced emissions. The automated billing framework streamlines the payment process, ensuring accurate fee collection and increasing revenue for parking facility operators. This efficiency not only enhances customer satisfaction but also encourages more people to utilize parking facilities, optimizing resource management.

Additionally, the system aligns with smart city initiatives by leveraging technology to improve urban living. It facilitates data collection on parking patterns and occupancy rates, which can inform urban planning and resource allocation. Furthermore, by promoting sustainable practices

and reducing operational costs, the intelligent parking system contributes to creating greener cities. Overall, this innovative solution represents a significant step toward efficient and sustainable urban mobility management .

Conclusion

- The intelligent car parking system provides a smart solution to urban parking challenges by automating parking space management and billing.
- It enhances user convenience and supports sustainable urban development.

In conclusion, the intelligent car parking system offers a comprehensive solution to the challenges of urban parking by integrating real-time vehicle detection, automated billing, and efficient parking management. It significantly improves user experience, reduces traffic congestion, and aligns with smart city goals by promoting sustainability and data-driven urban planning. With its practical and scalable design, the system has the potential to transform parking facilities into more efficient, user-friendly, and environmentally conscious spaces.

Reference

[1] *Elsonbaty, Amira & Shams, Mahmoud. (2020). The Smart Parking Management System. 10.48550/arXiv.2009.13443.*

[2] GeeksforGeeks. (n.d.). *GeeksforGeeks*. https://www.geeksforgeeks.org/