# Advanced & Smart Parking System

*in*

**Electronics and Communication Engineering**

*Submitted by*

**Karteek Dokala (21ECE1015)**

**Ruthala Jayanth (21ECE1029)**

*Under the supervision of*

# Dr. Trilochan Panigrahi

# Associate Professor

# Dr. Shivnarayan Patidar

# Assistant Professor

# National Institute of Technology Goa - Wikipedia

Department of Electronics and Communication Engineering

National Institute of Technology Goa

May 1, 2024

**ABSTRACT**

The Advanced & Smart Parking System (ASPS) represents a groundbreaking solution designed to revolutionize conventional parking infrastructure by integrating cutting-edge technologies and intelligent management systems. This project aims to address the challenges associated with urban congestion, inefficient space utilization, and manual parking management, while enhancing user experience and optimizing resource allocation.

ASPS leverages a combination of sensor technologies, real-time data analytics, and automated control mechanisms to provide a seamless parking experience. By deploying sensors at strategic locations within parking lots and utilizing machine learning algorithms, ASPS accurately detects and monitors parking space availability in real-time. This information is then communicated to drivers through mobile applications or electronic signage, enabling them to quickly locate vacant parking spots and reduce the time spent searching for parking.

One of the key advantages of ASPS is its scalability and adaptability to diverse parking environments, including commercial complexes, residential areas, and public facilities. Through cloud-based infrastructure and scalable architecture, ASPS can seamlessly integrate with existing parking systems or be deployed as a standalone solution, offering flexibility and ease of implementation.

In conclusion, the Advanced & Smart Parking System represents a paradigm shift in parking management, offering unparalleled efficiency, convenience, and sustainability. By harnessing the power of technology and data-driven insights, ASPS not only addresses the challenges of urban mobility but also lays the foundation for smarter and more connected cities of the future.

**TABLE OF CONTENTS**

1.Abstract

2.Introduction

3.Hardware description

4.Block diagram

5.Circuit diagram

6.Working principle

7.Testing

8.Advantages

9.Applications

10.Conclusion

**INTRODUCTION**

The Advanced & Smart Parking System (ASPS) is a pioneering initiative aimed at revolutionizing traditional parking infrastructure through the integration of advanced technologies and intelligent management systems. In an era characterized by rapid urbanization and increasing vehicular congestion, the need for innovative solutions to optimize parking resources and enhance user experience has become paramount. ASPS emerges as a comprehensive solution designed to address these challenges by leveraging state-of-the-art sensor technologies, real-time data analytics, and automated control mechanisms.

Traditional parking systems often suffer from inefficiencies such as limited visibility into parking space availability, time-consuming entry and exit procedures, and suboptimal space utilization. These inefficiencies not only lead to frustration among drivers but also contribute to traffic congestion and environmental degradation. ASPS seeks to overcome these limitations by providing accurate and up-to-date information on parking space availability, optimizing traffic flow within parking facilities, and streamlining entry and exit procedures.

An Advanced and smart Parking System utilizing Arduino Mega, ultrasonic sensors, and ESP8266 has been developed to efficiently manage a parking lot with 10 spaces. The Arduino Mega serves as the central control unit, orchestrating the interaction between the ultrasonic sensors and the ESP8266 module. Ultrasonic sensors are strategically placed in each parking space to detect the presence or absence of vehicles.

The ultrasonic sensors utilize sound waves to measure the distance between the sensor and the vehicle, providing accurate and real-time occupancy information. This data is then transmitted to the Arduino Mega for processing. The ESP8266 module, equipped with Wi-Fi capabilities, enables wireless communication between the parking system and a central monitoring station.

A user-friendly interface can be implemented, allowing drivers to check the availability of parking spaces in real-time through a dedicated mobile app or a web-based platform. The system can also incorporate smart features, such as notifying users about available parking spaces and guiding them to the optimal location within the parking lot.

In addition to efficient parking space utilization, the Advanced & Smart Parking System promotes sustainability by reducing unnecessary fuel consumption and emissions associated with searching for parking spaces. The integration of Arduino Mega, ultrasonic sensors, and ESP8266 exemplifies a cost-effective and scalable solution for managing parking facilities with enhanced precision and convenience.

**HARDWARE DESCRIPTION**

**List of Components**

1. Arduino Mega 2560 R3

2. ESP-01 ESP8266-01 WiFi Transceiver Module

3. Ultrasonic Sensor Module - HC-SR04

4. 20x4 Graphical LCD 2004A

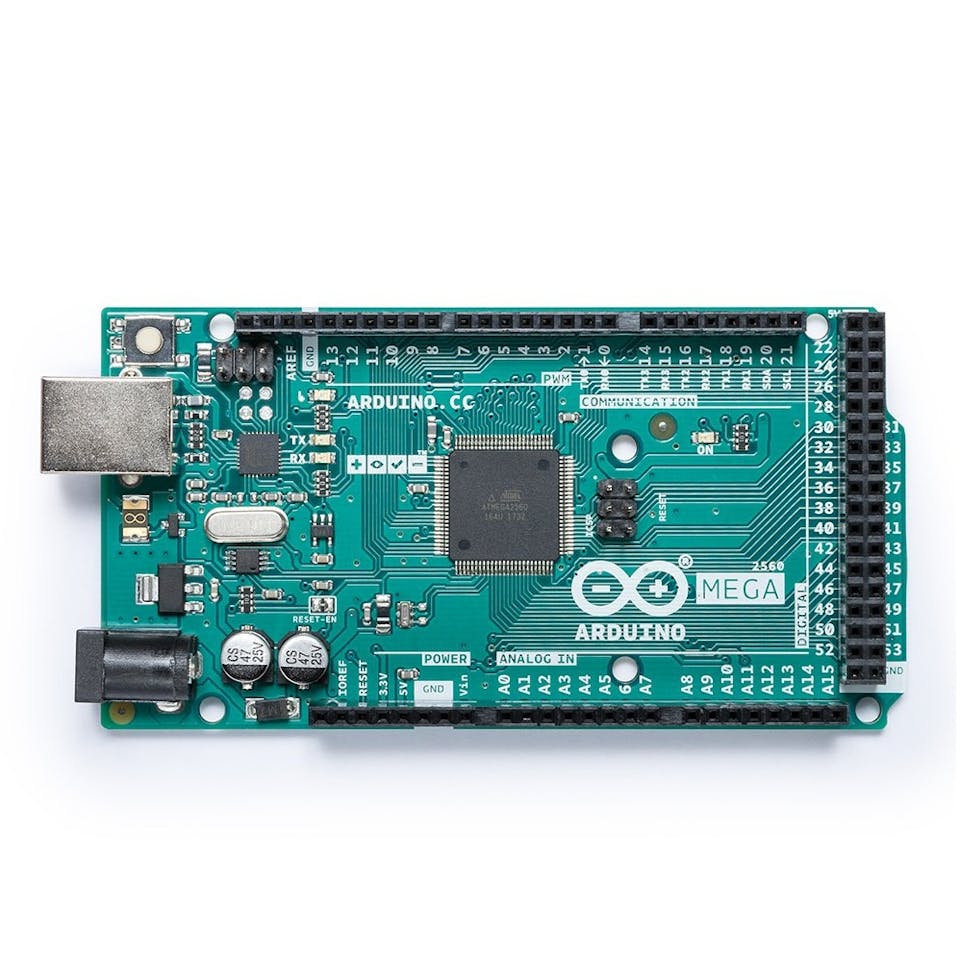
5. PCB board

6. connecting wires

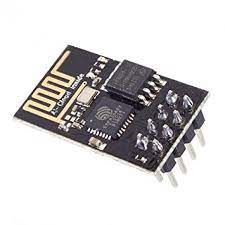
**Components Description**

***Arduino Mega 2560 R3***

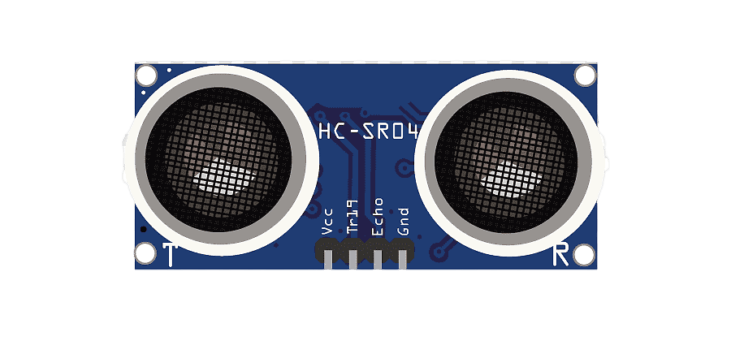
The Arduino Mega 2560 is an open-source microcontroller board based on ATmega2560. Along with this, it also has 54 digital I/O pins, 16 Analog Inputs, 4 UARTs, ICSP Header, 16 MHz Crystal Oscillator, a reset button, USB port, and a DC Power Jack. It is designed for projects which require more I/O lines, more sketch memory, and more RAM. It is best suited for Robotics, 3D printers, RC planes, etc. projects which require many sensors with different communication protocols and actuators to work in tandem. It gets programmed with the help of Arduino IDE and you do not need to attach any other components to program them. Its uses include in DIY projects, IoT based projects, Robotics, Prototyping for Electronic components and systems. The Arduino Microcontroller boards such as Arduino UNO, Arduino Nano, Arduino Leonardo, Arduino Pro Mini have revitalized the Automation Industry with their sheer simplicity and easy to use interface.



***ESP-01 ESP8266-01 WiFi Transceiver Module***

The ESP8266-01 is a low cost and user-friendly device to provide internet connectivity to your projects. This module can be used as a Wi-Fi hotspot or can connect to any WI-Fi network. We use this module to make our projects IoT enabled and gather and process the data easily. The ESP-01 can also fetch data from the internet and it can be used in your project. Its most important feature is that it can be programmed with Arduino IDE, which makes it a lot more user friendly. One drawback of this module is that it only has two GPIO pins, so you have to use it along with another controller like Arduino, or go for other modules like NodeMCU, ESP-12, or ESP-32. It finds uses in Portable Electronics, Robotics, Home Automation, Wireless Data Logging, etc.

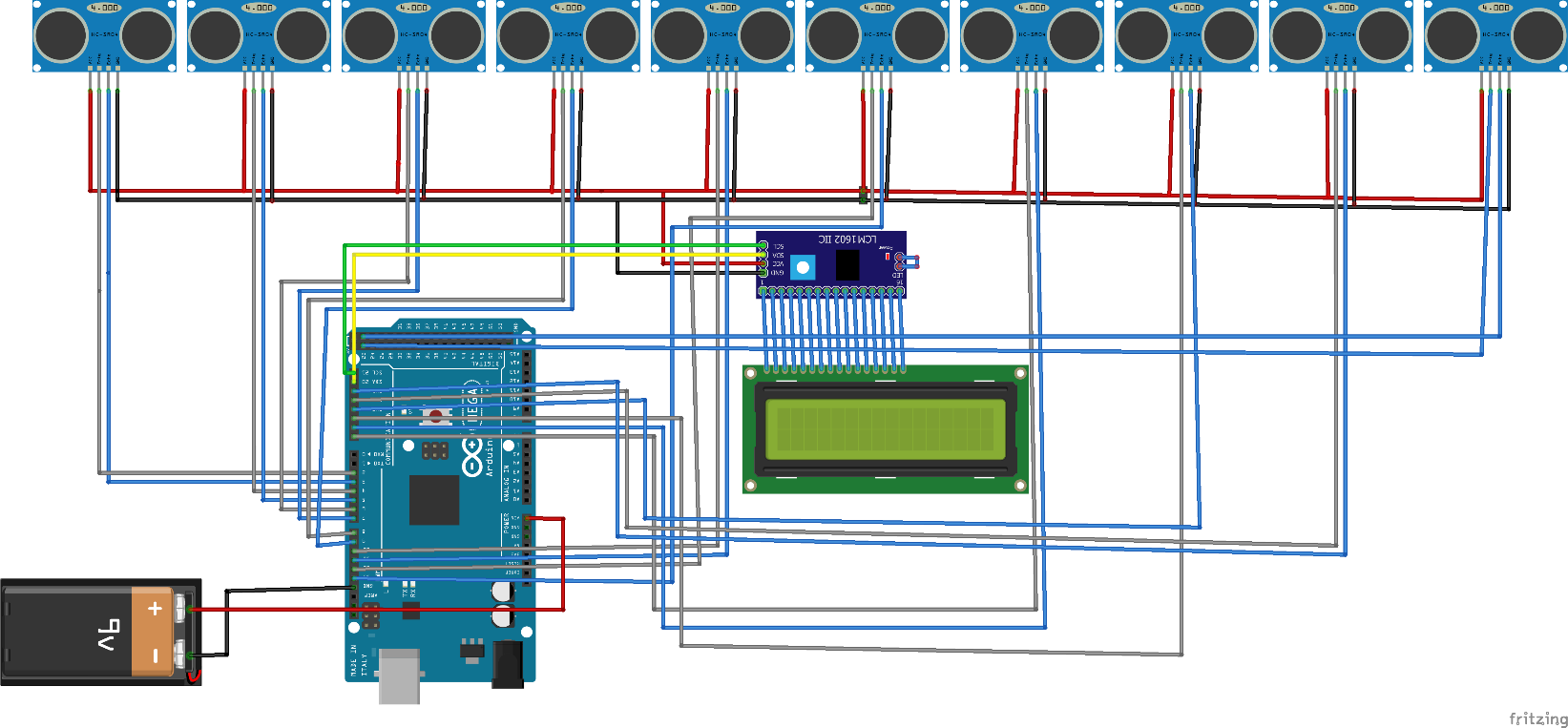
***Ultrasonic Sensor Module - HC-SR04***

The Ultrasonic Sensor uses Ultrasonic waves to determine the distance of an object like Bats, hence it can be used as a distance measuring sensor. There are two Ultrasonic Transducers present in which one acts as a Transmitter which transmits a high frequency Ultrasonic signal and other acts as a receiver which will wait for the receiving of echo signal which gets reflected by any object in its path. The time between the two signals when divider by speed of sound gives us the distance of the object. Theoretically the sensor claims to have a measuring distance of 2cm to 400cm. However, a range up to 75-80cm can be easily achieved practically. They are cheap, easy to interface and require low power to operate. They can be used to measure the depth of water as waves can travel in water, detect and avoid obstacles in the path of a robot and also as a parking assist sensor.

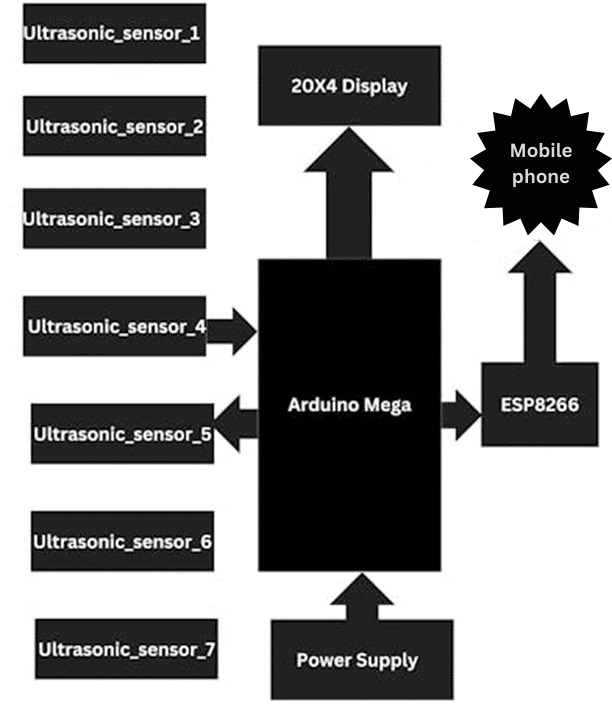
***20x4 Graphical LCD 2004A***

LCD modules are very commonly used in most embedded projects, the reason being its cheap price, availability, and programmer-friendly. Most of us would have come across these displays in our day to day life, either at PCO’s or calculators. Alphanumeric Graphical LCD (2004A) is a 20x4 Blue/Green Coloured Liquid Crystal Display. It can display 4 lines of text and each line can have up to 20 characters in it, which is bigger than the 16x2 LCD display but the programming is almost same. These characters can either be text, numbers, graphical symbols or even custom characters. It can be used in DIY projects, to display the data on IoT projects, etc.



**CIRCUIT DIAGRAM**

**Block Diagram**



**WORKING PRINCIPLE**

The Advanced & Smart Parking System utilizing Arduino Mega, ultrasonic sensors, and ESP8266 operates through a series of coordinated steps to efficiently manage a parking lot with 10 spaces:

* **Ultrasonic Sensor Detection:** Ultrasonic sensors are installed at each parking space to detect the presence or absence of vehicles. These sensors use ultrasonic waves to measure the distance between the sensor and the nearest obstacle, which in this case is a vehicle.
* **Data Collection:** The ultrasonic sensors continuously collect data regarding the occupancy status of each parking space. When a vehicle enters or exits a parking space, the corresponding sensor detects the change in distance and signals the Arduino Mega.
* **Arduino Mega Processing:** The Arduino Mega serves as the central processing unit for the parking system. It receives data from all the ultrasonic sensors and processes this information to determine the real-time occupancy status of each parking space.
* **Communication with ESP8266:** Once the Arduino Mega has processed the data, it communicates with the ESP8266 module. The ESP8266 is equipped with Wi-Fi capabilities, allowing it to establish a wireless connection with a central monitoring station or a user interface.
* **Wireless Transmission:** The Arduino Mega sends the occupancy data to the ESP8266, which wirelessly transmits this information to a central server or a cloud-based platform. This enables remote monitoring and management of the parking spaces
* **User Interface:** A user-friendly interface, such as a mobile app or a web-based platform, allows users to access real-time information about parking space availability. Drivers can check the status of the parking lot before arriving and make informed decisions about where to park.
* **Central Monitoring:** The central monitoring station, connected to the cloud or server, can keep track of parking space occupancy, trends, and system health. This information can be used for analytics and to improve the efficiency of the parking management system.

By integrating these components and functionalities, the Advanced & Smart Parking System provides an intelligent and responsive solution for optimizing parking space utilization while offering a seamless experience for users.

**Code for Arduino mega**

#include <LiquidCrystal\_I2C.h>

#include <SoftwareSerial.h>

SoftwareSerial mySerial(52, 53);

LiquidCrystal\_I2C lcd(0x27, 20, 4);

const int trig\_1 = 2;

const int echo\_1 = 3;

const int trig\_2 = 4;

const int echo\_2 = 5;

const int trig\_3 = 6;

const int echo\_3 = 7;

const int trig\_4 = 8;

const int echo\_4 = 9;

const int trig\_5 = 10;

const int echo\_5 = 11;

const int trig\_6 = 12;

const int echo\_6 = 13;

const int irSensor1 = 200;  // IR sensor for entry

const int irSensor2 = 500; // IR sensor for exit

const int servoPin = 900;   // Servo motor control pin

const int rotationAngle = 120;

const int carDetectLED\_1 = A0;

const int carDetectLED\_2 = A1;

const int carDetectLED\_3 = A2;

const int carDetectLED\_4 = A3;

const int carDetectLED\_5 = A4;

const int carDetectLED\_6 = A5;

float distanceCM\_1 = 0, resultCM\_1 = 0;

float distanceCM\_2 = 0, resultCM\_2 = 0;

float distanceCM\_3 = 0, resultCM\_3 = 0;

float distanceCM\_4 = 0, resultCM\_4 = 0;

float distanceCM\_5 = 0, resultCM\_5 = 0;

float distanceCM\_6 = 0, resultCM\_6 = 0;

long Time\_1, Time\_2, Time\_3, Time\_4, Time\_5, Time\_6;

float car\_1, car\_2, car\_3, car\_4, car\_5, car\_6;

float Dist\_1 = 50.0, Dist\_2 = 50.0, Dist\_3 = 50.0, Dist\_4 = 50.0, Dist\_5 = 50.0, Dist\_6 = 50.0;

int total = 0, timer\_cnt = 0;

float parkingFee = 50.0;  // Parking fee

void setup()

{

  Serial.begin(9600);

  mySerial.begin(115200);

  pinMode(trig\_1, OUTPUT);

  pinMode(trig\_2, OUTPUT);

  pinMode(trig\_3, OUTPUT);

  pinMode(trig\_4, OUTPUT);

  pinMode(trig\_5, OUTPUT);

  pinMode(trig\_6, OUTPUT);

  pinMode(echo\_1, INPUT);

  pinMode(echo\_2, INPUT);

  pinMode(echo\_3, INPUT);

  pinMode(echo\_4, INPUT);

  pinMode(echo\_5, INPUT);

  pinMode(echo\_6, INPUT);

  digitalWrite(trig\_1, LOW);

  digitalWrite(trig\_2, LOW);

  digitalWrite(trig\_3, LOW);

  digitalWrite(trig\_4, LOW);

  digitalWrite(trig\_5, LOW);

  digitalWrite(trig\_6, LOW);

  pinMode(carDetectLED\_1, OUTPUT);

  pinMode(carDetectLED\_2, OUTPUT);

  pinMode(carDetectLED\_3, OUTPUT);

  pinMode(carDetectLED\_4, OUTPUT);

  pinMode(carDetectLED\_5, OUTPUT);

  pinMode(carDetectLED\_6, OUTPUT);

  lcd.init();

  lcd.backlight();

  lcd.setCursor(0, 0);

  lcd.print("    IoT ENABLED");

  Serial.print("    IoT ENABLED");

  lcd.setCursor(0, 1);

  lcd.print("  PARKING SYSTEM");

  Serial.print("  PARKING SYSTEM");

  lcd.setCursor(0, 2);

  lcd.print("Jayanth (ECE)");

  Serial.print("Jayanth (ECE)");

  lcd.setCursor(0, 3);

  lcd.print(" NIT GOA");

  Serial.print(" NIT GOA");

  delay(4000);

  lcd.clear();

}

void loop()

{

  total = 0;

  car\_1 = sensor\_1();

  car\_2 = sensor\_2();

  car\_3 = sensor\_3();

  car\_4 = sensor\_4();

  car\_5 = sensor\_5();

  car\_6 = sensor\_6();

  lcd.setCursor(0, 0);

  lcd.print("P1:");

  if (car\_1 <= Dist\_1)

  {

    digitalWrite(carDetectLED\_1, HIGH);

    lcd.print("F,");

    Serial.print("F,");

  }

  else

  {

    total += 1;

  }

  if (car\_1 > Dist\_1)

  {

    digitalWrite(carDetectLED\_1, LOW);

    lcd.print("E,");

    Serial.print("E,");

    // displayParkingFee(1);

  }

  // lcd.setCursor(6, 0);

  lcd.print("P2:");

  if (car\_2 <= Dist\_2)

  {

    digitalWrite(carDetectLED\_2, HIGH);

    lcd.print("F,");

    Serial.print("F,");

  }

  else

  {

    total += 1;

  }

  if (car\_2 > Dist\_2)

  {

    digitalWrite(carDetectLED\_2, LOW);

    lcd.print("E,");

    Serial.print("E,");

    // displayParkingFee(2);

  }

  // lcd.setCursor(12, 0);

  lcd.print("P3:");

  if (car\_3 <= Dist\_3)

  {

    digitalWrite(carDetectLED\_3, HIGH);

    lcd.print("F,");

    Serial.print("F,");

  }

  else

  {

    total += 1;

  }

  if (car\_3 > Dist\_3)

  {

    digitalWrite(carDetectLED\_3, LOW);

    lcd.print("E,");

    Serial.print("E,");

    // displayParkingFee(3);

  }

  // lcd.setCursor(18, 0);

  lcd.print("P4:");

  if (car\_4 <= Dist\_4)

  {

    digitalWrite(carDetectLED\_4, HIGH);

    lcd.print("F");

    Serial.print("F,");

  }

  else

  {

    total += 1;

  }

  if (car\_4 > Dist\_4)

  {

    digitalWrite(carDetectLED\_4, LOW);

    lcd.print("E");

    Serial.print("E,");

    // displayParkingFee(4);

  }

  lcd.setCursor(0, 1);

  lcd.print("P5:");

  if (car\_5 <= Dist\_5)

  {

    digitalWrite(carDetectLED\_5, HIGH);

    lcd.print("F,");

    Serial.print("F,");

  }

  else

  {

    total += 1;

  }

  if (car\_5 > Dist\_5)

  {

    digitalWrite(carDetectLED\_5, LOW);

    lcd.print("E,");

    Serial.print("E,");

    // displayParkingFee(5);

  }

  // lcd.setCursor(0, 5);

  lcd.print("P6:");

  if (car\_6 <= Dist\_6)

  {

    digitalWrite(carDetectLED\_6, HIGH);

    lcd.print("F,");

    Serial.print("F,");

  }

  else

  {

    total += 1;

  }

  if (car\_6 > Dist\_6)

  {

    digitalWrite(carDetectLED\_6, LOW);

    lcd.print("E,");

    Serial.print("E,");

    // displayParkingFee(6);

  }

  // lcd.setCursor(0, 4);

  // lcd.print("E:");

  // lcd.print(total);

  // if (timer\_cnt >= 50)

  {

    mySerial.print('\*');

    Serial.print('\*');

    mySerial.print(total);

    Serial.print(total);

    mySerial.println('#');

    Serial.println('#');

    timer\_cnt = 0;

  }

  timer\_cnt += 1;

  lcd.setCursor(0, 3);

  lcd.print("  EMPTY PARKING:");

  Serial.println("  EMPTY PARKING:");

  lcd.print(total);

  Serial.println(total);

  if (timer\_cnt >= 50)

  {

    mySerial.print('\*');

    Serial.print('\*');

    mySerial.print(total);

    Serial.print(total);

    mySerial.println('#');

    Serial.println('#');

    timer\_cnt = 0;

  }

  timer\_cnt += 1;

  delay(200);

}

float sensor\_1(void)

{

  digitalWrite(trig\_1, HIGH);

  delayMicroseconds(10);

  digitalWrite(trig\_1, LOW);

  Time\_1 = pulseIn(echo\_1, HIGH);

  distanceCM\_1 = Time\_1 \* 0.034;

  return resultCM\_1 = distanceCM\_1 / 2;

}

float sensor\_2(void)

{

  digitalWrite(trig\_2, HIGH);

  delayMicroseconds(10);

  digitalWrite(trig\_2, LOW);

  Time\_2 = pulseIn(echo\_2, HIGH);

  distanceCM\_2 = Time\_2 \* 0.034;

  return resultCM\_2 = distanceCM\_2 / 2;

}

float sensor\_3(void)

{

  digitalWrite(trig\_3, HIGH);

  delayMicroseconds(10);

  digitalWrite(trig\_3, LOW);

  Time\_3 = pulseIn(echo\_3, HIGH);

  distanceCM\_3 = Time\_3 \* 0.034;

  return resultCM\_3 = distanceCM\_3 / 2;

}

float sensor\_4(void)

{

  digitalWrite(trig\_4, HIGH);

  delayMicroseconds(10);

  digitalWrite(trig\_4, LOW);

  Time\_4 = pulseIn(echo\_4, HIGH);

  distanceCM\_4 = Time\_4 \* 0.034;

  return resultCM\_4 = distanceCM\_4 / 2;

}

float sensor\_5(void)

{

  digitalWrite(trig\_5, HIGH);

  delayMicroseconds(10);

  digitalWrite(trig\_5, LOW);

  Time\_5 = pulseIn(echo\_5, HIGH);

  distanceCM\_5 = Time\_5 \* 0.034;

  return resultCM\_5 = distanceCM\_5 / 2;

}

float sensor\_6(void)

{

  digitalWrite(trig\_6, HIGH);

  delayMicroseconds(10);

  digitalWrite(trig\_6, LOW);

  Time\_6 = pulseIn(echo\_6, HIGH);

  distanceCM\_6 = Time\_6 \* 0.034;

  return resultCM\_6 = distanceCM\_6 / 2;

}

**Idea about making this Project**

***Efficient Space Utilization:*** The goal of such a system is to optimize the utilization of parking spaces. By implementing smart technologies, you can reduce the time spent by drivers searching for available parking spots and minimize congestion in parking lots.

***Resource Conservation:*** Smart parking systems contribute to resource conservation by reducing fuel consumption and emissions associated with circling a parking lot in search of a space. This aligns with environmental sustainability goals.

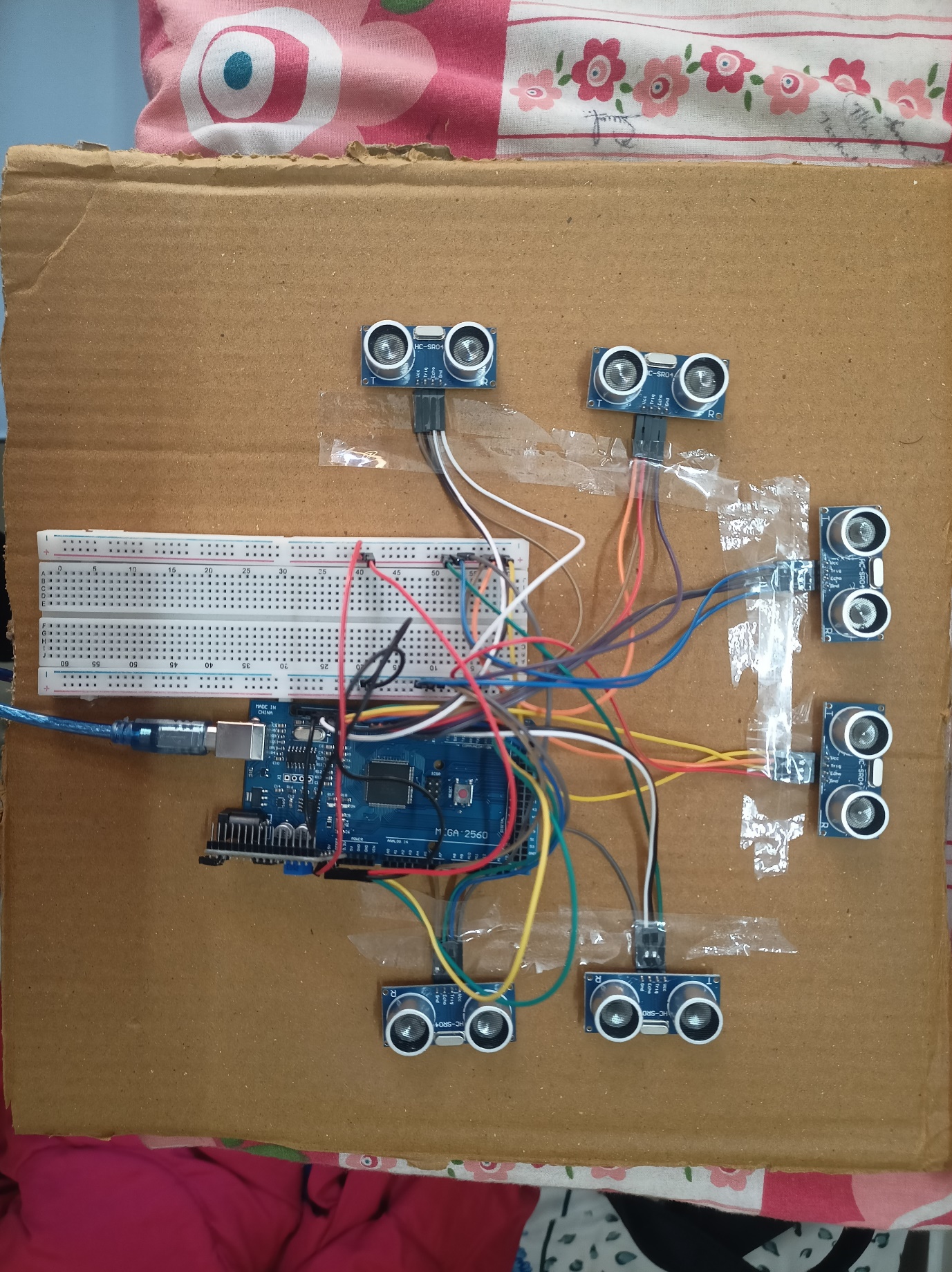
***Technological Innovation:*** Building a parking system with Arduino Mega, ultrasonic sensors, and ESP8266 showcases technological innovation. It allows enthusiasts or developers to explore the capabilities of these devices and learn more about the integration of hardware and software.

***User Convenience:*** The system provides a convenient way for users to check parking space availability in real-time through a mobile app or web interface. This can enhance the overall user experience and make parking more convenient.

***Data Insights:*** By collecting and analyzing data on parking space occupancy, the system can offer valuable insights for future planning and decision-making. For example, trends in parking demand can inform the expansion or optimization of parking facilities.

***Demonstration of IoT Concepts:*** The integration of the ESP8266 module demonstrates the concept of IoT by enabling wireless communication between the parking system and a central monitoring station. This showcases the potential of connected devices in real-world applications.

Ultimately, the decision to create an Advanced & Smart Parking System may stem from a combination of environmental concerns, a desire for technological exploration, and a commitment to improving user experiences in parking facilities.

**TESTING**

**CONCLUSION**

In conclusion, the development of a smart parking system using an Arduino Mega and ultrasonic-sensors presents a practical and innovative solution to address parking management challenges. Through the integration of sensor technology and microcontroller programming, this project offers real-time monitoring and efficient utilization of parking spaces.

The implementation process involves careful sensor placement, code development for accurate detection, and optional features such as user interfaces and gate control integration. By leveraging the versatility of Arduino platforms and the reliability of ultrasonic sensors, the system can effectively detect parking spot occupancy and display status information to users.

Furthermore, the scalability and customization options of this project allow for adaptation to various parking environments, from small-scale lots to large parking structures. With additional features like RFID or NFC integration and power-saving mechanisms, the system can be tailored to meet specific requirements and enhance user experience.

Overall, the smart parking system provides a cost-effective and sustainable solution for optimizing parking space utilization, improving traffic flow, and enhancing convenience for users. As technology continues to advance, further enhancements and refinements can be implemented to meet evolving needs and ensure the system's long-term effectiveness. Through continued innovation and implementation, smart parking systems contribute to smarter, more efficient urban environments.