PHASE -5
SMART PARKING
ON
ULTRASONIC SENSOR

- *project objectives*
- *IOT device setup*
- *platform development*
- *program*
- *circuit diagram*
- *output*
- *schematic*
- *IOT device*

PROJECT OBJECTIVE

1. ***Optimize Parking Space Utilization***: *Design a system that efficiently utilizes parking spaces by providing real-time information on available spots.*
2. ***Real-time Parking Availability***: *Develop a system that continuously monitors parking spaces and updates a central database or display with the availability status.*
3. ***Reduce Traffic Congestion***: *Aim to decrease traffic congestion by guiding drivers directly to available parking spaces, minimizing unnecessary circling.*
4. ***Enhance User Convenience***: *Create a user-friendly interface, possibly a mobile app, that allows drivers to easily access information about available parking spaces.*
5. ***Minimize Environmental Impact***: *By reducing the time spent searching for parking, the system can help decrease emissions and fuel consumption.*

6. ***Integration with IoT Platform***: *Enable integration with Internet of Things (IoT) platforms for seamless data sharing and remote monitoring.*
7. ***Occupancy Monitoring***: *Implement a feature to detect if a vehicle is parked incorrectly or if a spot becomes occupied after initially being marked as vacant.*
8. ***Alerts and Notifications***: *Provide notifications to drivers about the availability of parking spaces through various communication channels (app, SMS, etc.).*
9. ***Security and Safety***: *Ensure the system's reliability and accuracy to prevent accidents or conflicts over parking spaces.*
10. ***Data Analysis and Reporting***: *Incorporate data analytics to gather insights about parking patterns and trends, which can be used for future city planning or improvements to the system.*

IOT DEVICE SETUP:

1. ***Gather Required Components***:
   - *Ultrasonic Sensors (for detecting parked vehicles)*
   - *Microcontroller (such as Arduino, Raspberry Pi, or similar)*
   - *Power supply for the sensors and microcontroller*
   - *Connecting wires*
2. ***Connect Ultrasonic Sensors***:
   - *Connect the ultrasonic sensors to the microcontroller. Generally, ultrasonic sensors have at least four pins: VCC, GND, TRIG, and ECHO. Connect VCC to a power source, GND to ground, TRIG to a digital pin for triggering, and ECHO to a digital pin for receiving.*
3. ***Install Necessary Libraries***:
   - *Depending on your microcontroller, you might need to install specific libraries to interface with the ultrasonic sensors. For example, for Arduino, you can use libraries like "NewPing" or "Ultrasonic."*

4. ***Write Firmware***:
    - *Develop the firmware (code) that reads data from the ultrasonic sensors and communicates it to the central system. Use the microcontroller's programming language (e.g., C++ for Arduino).*
5. ***Test Sensors and Microcontroller***:
    - *Write a simple test program to ensure that the sensors are functioning correctly and providing accurate readings.*
6. ***Establish Communication***:
    - *Decide on the communication protocol to transmit data from the IoT device to the central server or cloud platform. Common protocols include MQTT, HTTP, or LoRa.*
7. ***Connect to Internet***:
    - *Set up the IoT device to connect to the internet. This might involve configuring Wi-Fi or other networking options, depending on your chosen microcontroller.*

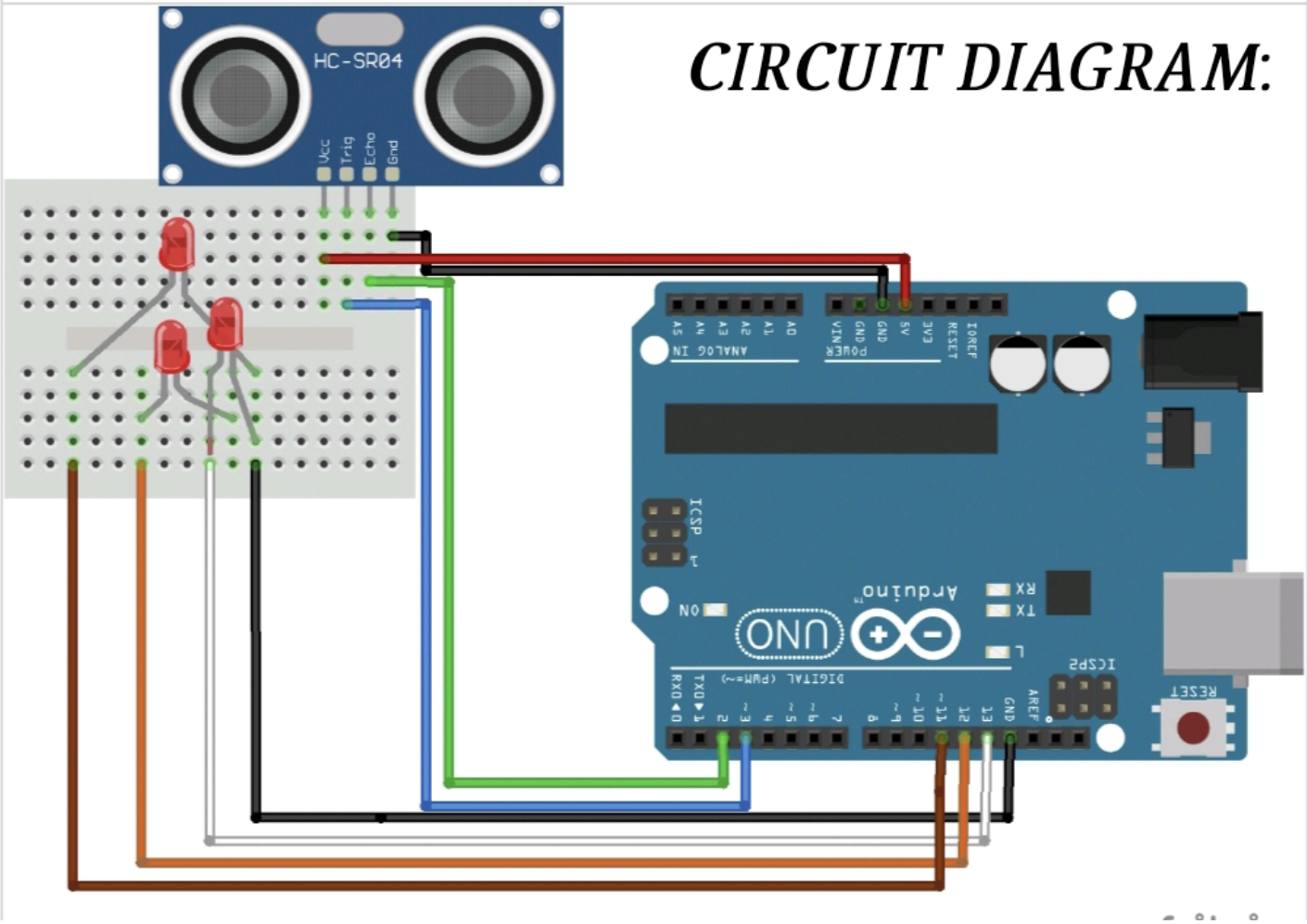
PLATFORM

1. **Define Requirements**:
   - Clearly outline the features and functionalities you want in the platform. This may include real-time parking availability, user authentication, notifications, etc.
2. **Choose a Technology Stack**:
   - Decide on the programming languages, frameworks, and tools you'll use for backend, frontend, and database development. For example, you might use Python, Django or Node.js for backend, HTML/CSS/ JavaScript for frontend, and a database like MySQL or PostgreSQL.
3. **Design the Database**:
   - Create a database schema to store information about parking spaces, sensor data, user accounts, and any other relevant data.
4. **Set Up Backend**:
   - Develop the backend logic for processing sensor data, managing user accounts, handling authentication, and interacting with the database.

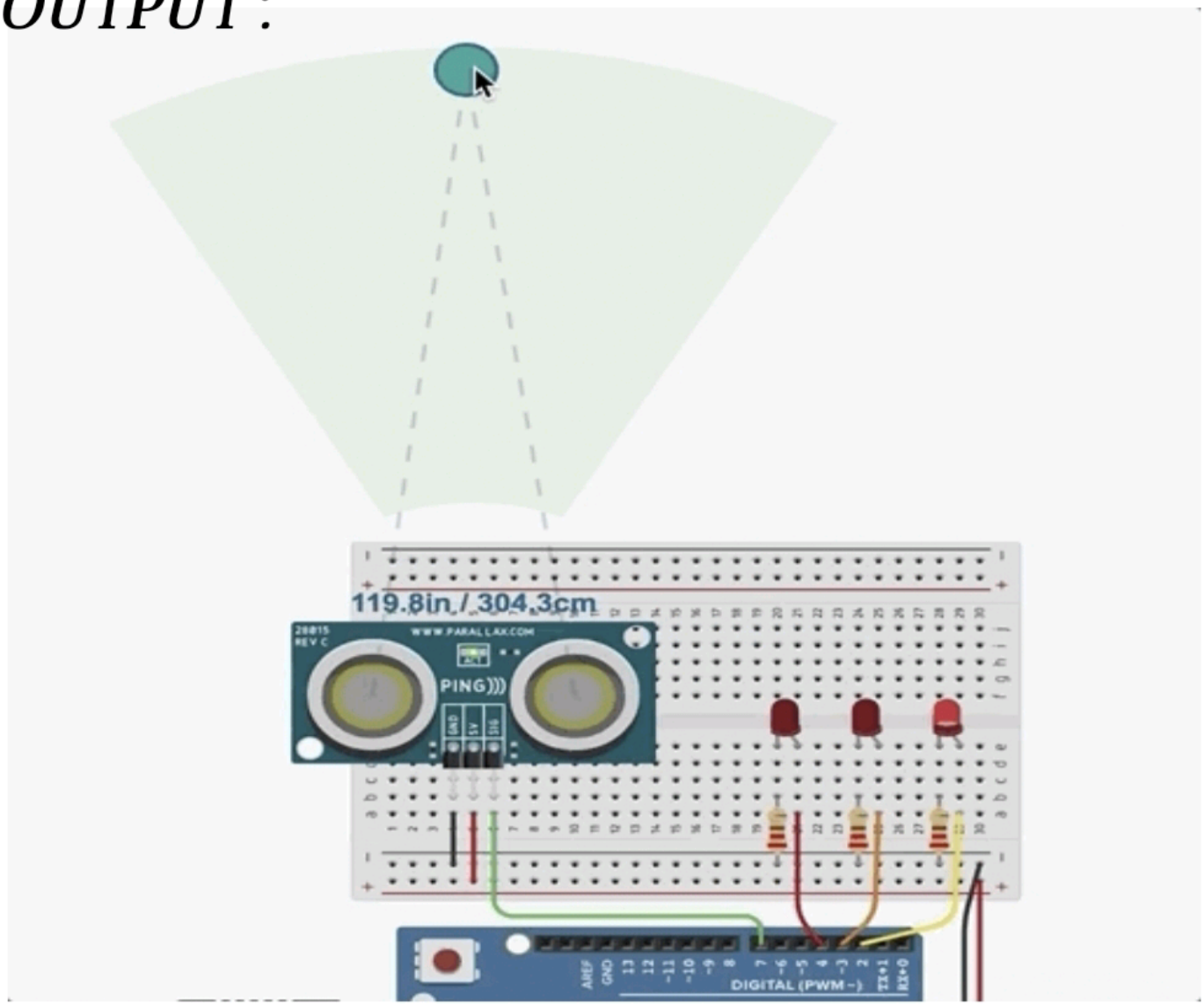
5. ***Integrate Ultrasonic Sensors***:
    - *Write code to interface with the ultrasonic sensors, read data, and send it to the backend for processing. This could be done using microcontroller programming (e.g., Arduino or Raspberry Pi).*
6. ***Real-time Data Processing***:
    - *Implement algorithms to process the sensor data and determine parking space availability.*
7. ***Implement User Authentication***:
    - *Create a system for user registration, login, and authentication. This is crucial for user-specific features and data privacy.*
8. ***User Interface Development***:
    - *Design and develop a user-friendly interface for accessing parking information. This could be a web application or a mobile app.*

```
int trigPin = 9; // Trigger Pin of Ultrasonic
Sensor
int echoPin = 8; // Echo Pin of Ultrasonic
Sensor
void setup( )
{
  Serial.begin(9600);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}
void loop( ){
  long duration, inches, cm;
digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
inches = duration/74 /2;  //convert
microseconds into inches
  cm = duration/29 /2;   //convert
microseconds into cm
  Serial.print(inches);
  Serial.print(" inches ");
  Serial.print(cm);
  Serial.print(" cm");
  Serial.println( );
  delay(200);
}
```

CIRCUIT DIAGRAM:



OUTPUT :



Here's a basic schematic:

1. ***Components:***
    - *Ultrasonic Sensor (e.g., HC-SR04)*
    - *Microcontroller (e.g., Arduino Uno)*
    - *Power Supply (for both the microcontroller and the sensor)*
    - *Connecting Wires*
2. ***Connections:***
    - *Connect the VCC (power) pin of the ultrasonic sensor to a power source (e.g., 5V output from the microcontroller).*
    - *Connect the GND (ground) pin of the ultrasonic sensor to the ground (GND) pin on the microcontroller.*
    - *Connect the TRIG (trigger) pin of the ultrasonic sensor to a digital pin on the microcontroller (e.g., D2).*
    - *Connect the ECHO (echo) pin of the ultrasonic sensor to another digital pin on the microcontroller (e.g., D3).*
    - *Connect the VCC and GND pins of the microcontroller to a suitable power source.*

1. ***Power Supply***:
    - *Ensure that you have a stable power supply for both the microcontroller and the sensor. Consider using a separate power supply for the sensor if needed.*
2. ***Testing***:
    - *Upload the code to the microcontroller and observe the serial monitor to see the distance readings from the ultrasonic sensor.*

IOT DEVICE :

1. ***Define Data Sharing Requirements***:
   - *Determine what data you want to share (e.g., real-time parking availability, historical data, user preferences) and who the target audience is (e.g., drivers, city planners).*
2. ***Select a Cloud Platform or Server***:
   - *Choose a cloud platform (e.g., AWS, Google Cloud, Azure) or set up a dedicated server to host the data sharing platform.*
3. ***Set Up a Database***:
   - *Create a database to store parking availability data. Choose a database system that suits your needs (e.g., SQL, NoSQL).*
4. ***Develop API Endpoints***:
   - *Create API endpoints to allow IoT devices (ultrasonic sensors) and the user interface to interact with the database. Use technologies like RESTful APIs or GraphQL.*

5. ***Data Processing and Aggregation***:
    - *Write backend code to process incoming data from the sensors. This might involve aggregating data, calculating statistics, and preparing it for storage in the database.*
6. ***User Authentication and Authorization***:
    - *Implement a system for user authentication and authorization to control access to the data sharing platform.*
7. ***Real-time Data Updates***:
    - *Set up mechanisms to provide real-time updates to the user interface and any connected applications.*
8. ***Security Measures***:
    - *Implement security measures to protect user data and the platform itself. This includes encryption, secure APIs, and secure coding practices.*

EXPLANATION:

An ultrasonic sensor is used to measure the distance to an object using sound waves. It provides measurements of the time that takes the sound to fling something and return it to the sensor.

It works on sound wave frequency like sonar. They also cannot detect some objects.

HC-SR04 ultrasonic can measure a range of 2 cm to 400 cm (4 m).

This sensor has 2 openings on its front. One is the transmitter which transmits ultrasonic waves like a speaker and the other is a receiver that receives them like a microphone.

Mainly there operations are not affected by sunlight or black material, but acoustically, soft materials like cloth can be difficult to detect.

It works on sound wave frequency like sonar. The ultrasonic sensor also cannot detect some objects.

HC-SR04 ultrasonic can measure a range of 2 cm to 400 cm (4 m).

This sensor has 2 openings on its front. One is the transmitter which transmits ultrasonic waves like a speaker and the other is a receiver that receives them like a microphone.

Hardware Connections

- **Vcc** to **5V** of Arduino.
- **GND** to **GND** of Arduino.
- **Trig** to digital pin **13** of Arduino.
- **Echo** to digital pin **12** of Arduino.

HC-SR04 Specifications

Board Size	43 x 20 x 15 mm
Working voltage	5V DC
Operating voltage	5V DC
Working Frequency	40Hz
Range	2 cm to 400 cm (4 m)

Measuring Angle	*15 degree*
Trigger Input Signal	*10 µs TTL pulse*
Echo Output Signal	*Input TTL lever signal and the range in proportion*

Uses of Ultrasonic Sensor

- *For Measuring Distance.*
- *Robotic sensing.*
- *Vehicle detection for car wash and automotive assembly.*
- *Use in Parking Areas.*
- *Presence detection.*

Pin Connections

Ultrasonic Pins	***Arduino Pins***
5v or Vin	*5V*
GND	*GND*
Trigger	*9*
Echo	*8*