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.*

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.

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.*

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.*

Components of the Smart Parking System:

1. **Ultrasonic Sensors:** These sensors are installed in each parking space to detect the presence of a vehicle.
2. **Microcontroller (e.g., Arduino or Raspberry Pi):** The microcontroller processes the sensor data and communicates with the central server.
3. **Central Server:** This server stores and manages data about available parking spaces and communicates with the user interface.
4. **User Interface:** Users can access the system through a web or mobile app to check parking availability and reserve spots.
5. **Database:** The system should have a database to store parking space information and reservations.

IOT DEVICE :

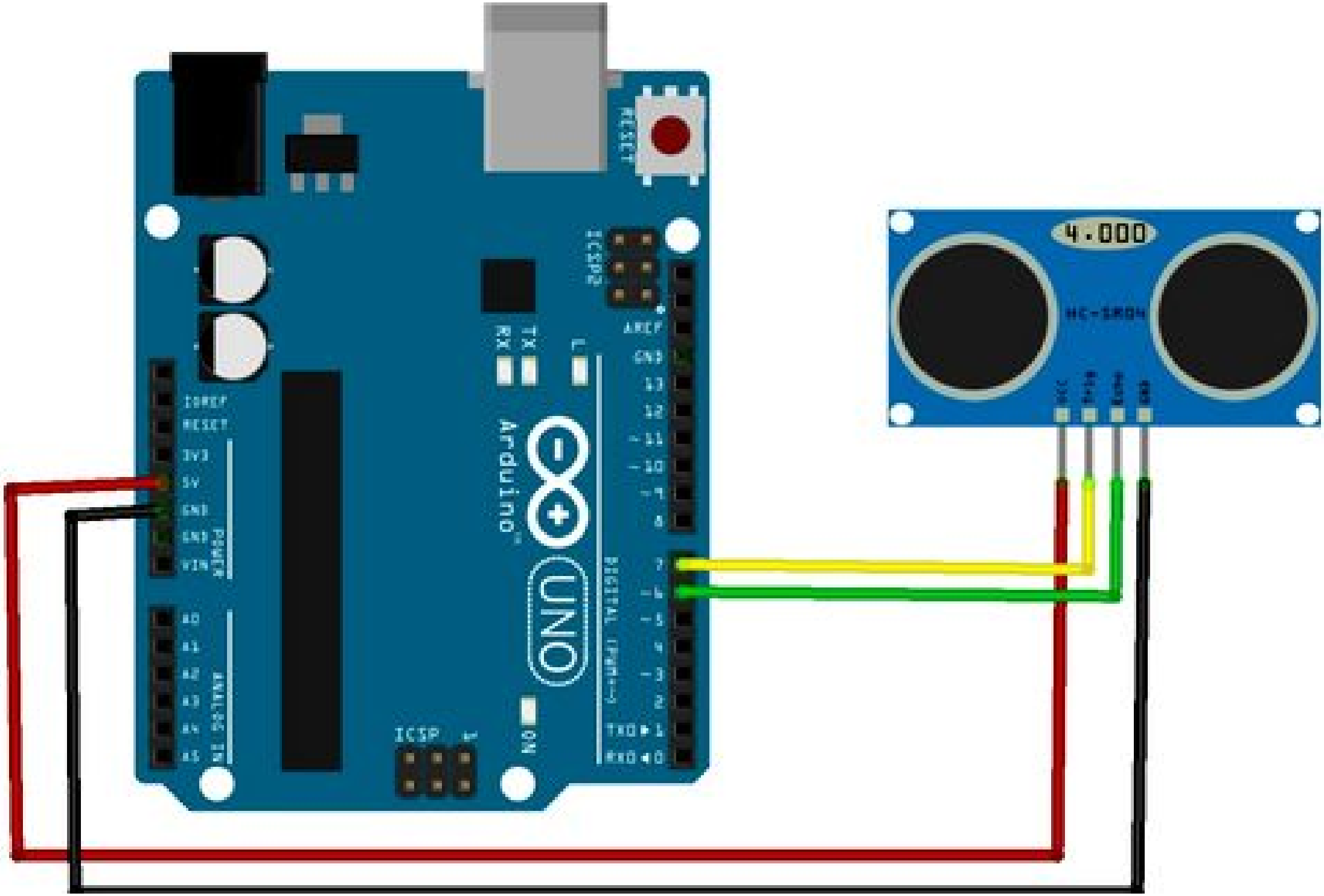
1. ***Define Data Sharing Requirements***:
   - *Determine what data you want to share (e.g., real-time parking availability, historical data, user preferences) and who the target audience is (e.g., drivers, city planners).*
2. ***Select a Cloud Platform or Server***:
   - *Choose a cloud platform (e.g., 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.*

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.

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.

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*


RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM~)
ICSP
L
TX
RX
ON
POWER
ANALOG IN
IOREF
RESET
3V3
5V
GND
GND
VIN
A0
A1
A2
A3
A4
A5
4.000
HC-SR04
Vcc
Trig
Echo
Gnd

```
#include <NewPing.h>

#define TRIGGER_PIN 7  // Arduino pin
connected to sensor's trigger pin.
#define ECHO_PIN 6     // Arduino pin
connected to sensor's echo pin.
#define MAX_DISTANCE 200 // Maximum
distance in centimeters (adjust
according to your setup).

NewPing sonar(TRIGGER_PIN, ECHO_PIN,
MAX_DISTANCE); // Create a NewPing
object.

void setup() {
  Serial.begin(9600);
}

void loop() {
  delay(100); // Wait for 100
milliseconds between readings.

  unsigned int distance =
sonar.ping_cm(); // Get distance in
centimeters.

  if (distance == 0) {
    Serial.println("Out of range");
  } else {
    Serial.print("Distance: ");
    Serial.print(distance);
    Serial.println(" cm");
  }
```