**SMART PARKING**

**PHASE 3**

**SENSOR DESIGN SIMULATION USING WOKWI**

**INTRODUCTION:**

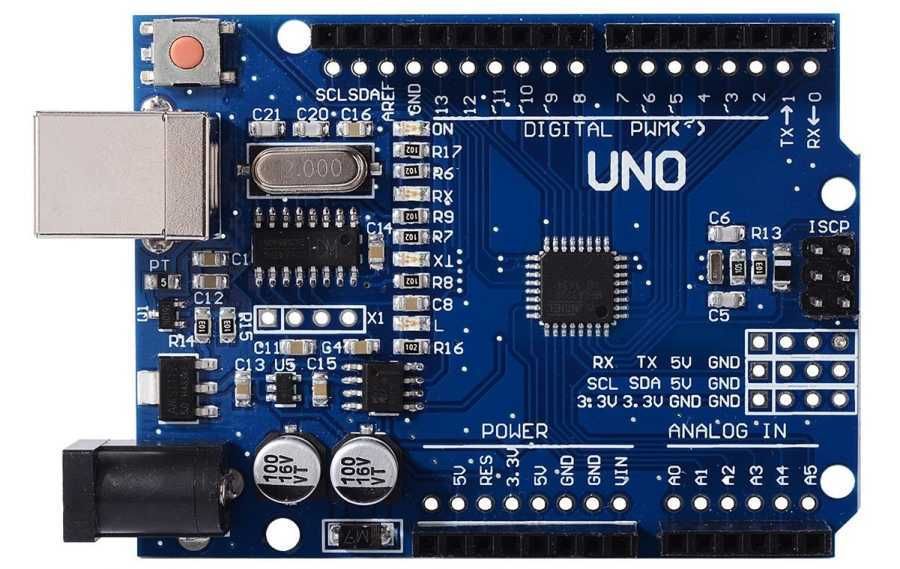
In phase 3, we have interfaced the ultrasonic sensor(HC-SR04) with the Arduino Uno. The simulation of the circuit was done using WOKWI simulator. Our project primarily focuses on techniques of smart parking and detection of distances between two cars and if there is enough distance available to park your cars. The output is displayed using LED display.

**REQUIREMENTS:**

Creating a complete smart parking system in an online simulation platform like Wokwi requires multiple components, including sensors, a microcontroller (such as an Arduino), and a way to visualize and control the system.

* Arduino Uno:

[**Arduino UNO**](https://www.theengineeringprojects.com/2018/06/introduction-to-arduino-uno.html) is a low-cost, flexible, and easy-to-use programmable open-source microcontroller board that can be integrated into a variety of electronic projects. This board can be interfaced with other Arduino boards, Arduino shields, Raspberry Pi boards and can control relays, LEDs, servos, and motors as an output.



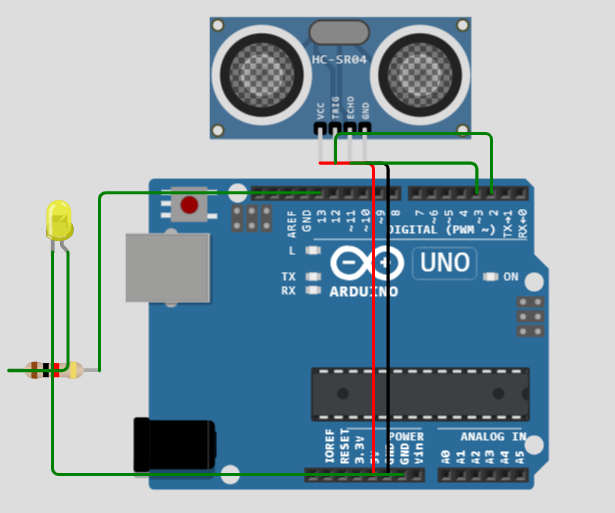
* Ultrasonic sensor (HC-SR04):

The HC-SR04 sensor is a distance sensor.This economical sensor provides 2cm to 400cm of non-contact measurement functionality with a ranging accuracy that can reach up to 3mm. Each HC-SR04 module includes an ultrasonic transmitter, a receiver and a control circuit.

There are only four pins that you need to worry about on the HC-SR04: VCC (Power), Trig (Trigger), Echo (Receive), and GND (Ground).

* LEDs (to represent parking spots)
* Breadboard and wires
* Resistor : 1kΩ

**CIRCUIT DESIGN:**



**PROGRAM:**

#define TRIGGER\_PIN 9

#define ECHO\_PIN 10

const int numParkingSpots = 4; // Number of parking spots

bool isOccupied[numParkingSpots] = {false}; // Array to track parking spot status

void setup() {

Serial.begin(9600);

pinMode(TRIGGER\_PIN, OUTPUT);

pinMode(ECHO\_PIN, INPUT);

for (int i = 2; i <= 5; i++) {

pinMode(i, OUTPUT);

}

}

void loop() {

for (int spot = 0; spot < numParkingSpots; spot++) {

long duration, distance;

digitalWrite(TRIGGER\_PIN, LOW);

delayMicroseconds(2);

digitalWrite(TRIGGER\_PIN, HIGH);

delayMicroseconds(10);

digitalWrite(TRIGGER\_PIN, LOW);

duration = pulseIn(ECHO\_PIN, HIGH);

distance = (duration / 2) / 29.1;

if (distance < 10) {

isOccupied[spot] = true;

} else {

isOccupied[spot] = false;

}

// Update LED for the current parking spot

digitalWrite(spot + 2, isOccupied[spot] ? HIGH : LOW);

}

// Print parking spot status

for (int spot = 0; spot < numParkingSpots; spot++) {

Serial.print("Spot ");

Serial.print(spot + 1);

Serial.print(": ");

Serial.println(isOccupied[spot] ? "Occupied" : "Empty");

}

delay(1000);

}

```cpp

#define TRIGGER\_PIN 9

#define ECHO\_PIN 10

const int numParkingSpots = 4; // Number of parking spots

bool isOccupied[numParkingSpots] = {false}; // Array to track parking spot status

void setup() {

Serial.begin(9600);

pinMode(TRIGGER\_PIN, OUTPUT);

pinMode(ECHO\_PIN, INPUT);

for (int i = 2; i <= 5; i++) {

pinMode(i, OUTPUT);

}

}

void loop() {

for (int spot = 0; spot < numParkingSpots; spot++) {

long duration, distance;

digitalWrite(TRIGGER\_PIN, LOW);

delayMicroseconds(2);

digitalWrite(TRIGGER\_PIN, HIGH);

delayMicroseconds(10);

digitalWrite(TRIGGER\_PIN, LOW);

duration = pulseIn(ECHO\_PIN, HIGH);

distance = (duration / 2) / 29.1;

if (distance < 10) {

isOccupied[spot] = true;

} else {

isOccupied[spot] = false;

}

// Update LED for the current parking spot

digitalWrite(spot + 2, isOccupied[spot] ? HIGH : LOW);

}

// Print parking spot status

for (int spot = 0; spot < numParkingSpots; spot++) {

Serial.print("Spot ");

Serial.print(spot + 1);

Serial.print(": ");

Serial.println(isOccupied[spot] ? "Occupied" : "Empty");

}

delay(1000);

}

**EXPLANATION:**

In this code, we use an ultrasonic sensor to detect the presence of a vehicle in each parking spot. The Arduino controls LEDs that represent the status of each parking spot (occupied or empty). The status is displayed in the serial monitor.

**1. Variable and Constant Definitions:**

- `TRIGGER\_PIN` and `ECHO\_PIN` are defined to specify the Arduino pins connected to the ultrasonic sensor's trigger and echo pins, respectively.

- `numParkingSpots` is set to 4, indicating that there are four parking spots to monitor.

- An array `isOccupied` is defined to keep track of the status of each parking spot. Initially, all spots are assumed to be empty (`false`).

**2. Setup Function:**

- `Serial.begin(9600)` initializes the serial communication for debugging, allowing you to view the status of parking spots on the serial monitor.

- `pinMode(TRIGGER\_PIN, OUTPUT)` and `pinMode(ECHO\_PIN, INPUT)` set the trigger pin as an output and the echo pin as an input for the ultrasonic sensor.

- A `for` loop is used to set pins 2, 3, 4, and 5 as OUTPUT. These pins are connected to LEDs that represent the parking spot status.

**3. Loop Function:**

- The `loop` function is where the continuous monitoring of parking spots takes place.

- A `for` loop iterates through each parking spot (from 0 to 3).

- An ultrasonic sensor is used to measure the distance to an object in front of it. The steps include triggering a pulse, measuring the time it takes for the pulse to bounce back (echo), and converting it into a distance in centimeters.

- If the measured distance is less than 10 centimeters (you can adjust this threshold), it's considered that a vehicle is occupying the spot, and `isOccupied[spot]` is set to `true`. Otherwise, the spot is considered empty (`false`).

- LEDs connected to pins 2, 3, 4, and 5 are updated to indicate the status of each parking spot (lit if occupied, off if empty).

**4. Status Printing:**

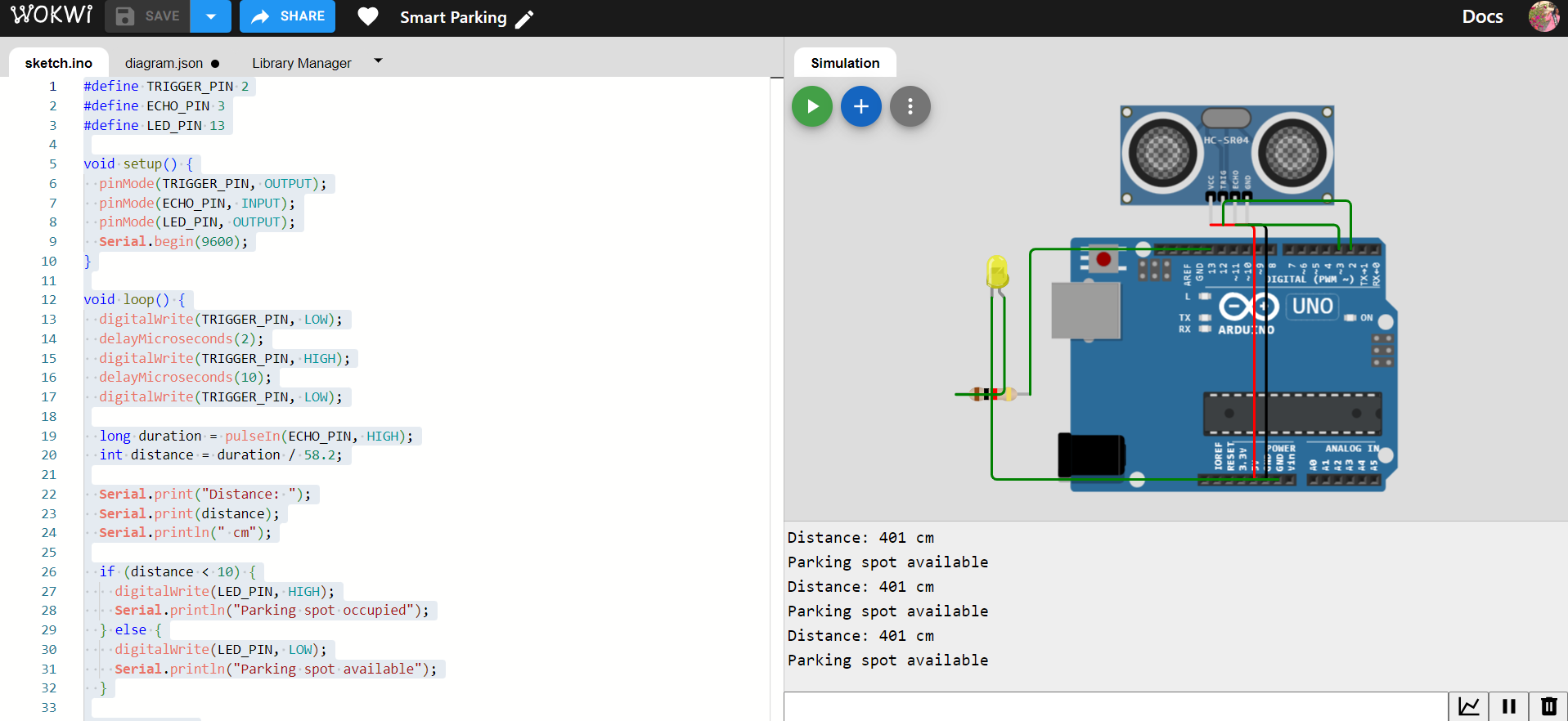
- After checking the status of all parking spots, a second `for` loop is used to print the status of each spot on the serial monitor. It will display the spot number and whether it is "Occupied" or "Empty."

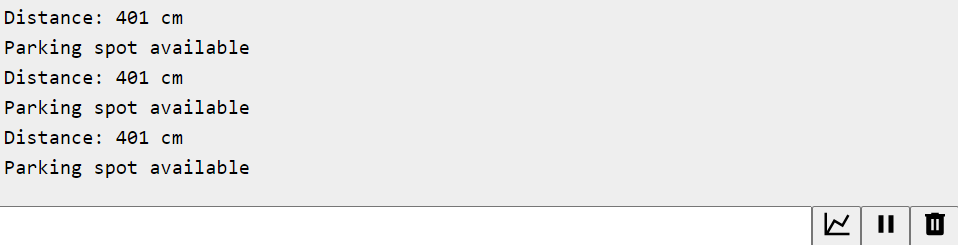
**5. Delay:**

- A delay of 1000 milliseconds (1 second) is added to prevent constant readings and updates. This provides a more manageable output on the serial monitor and avoids rapid changes in the LED status.

This code continuously monitors the distance in front of the ultrasonic sensor for each parking spot and updates the status of the parking spots (occupied or empty) on the LEDs and in the serial monitor. This is a basic example of a smart parking system.

**OUTPUT:**

****

****