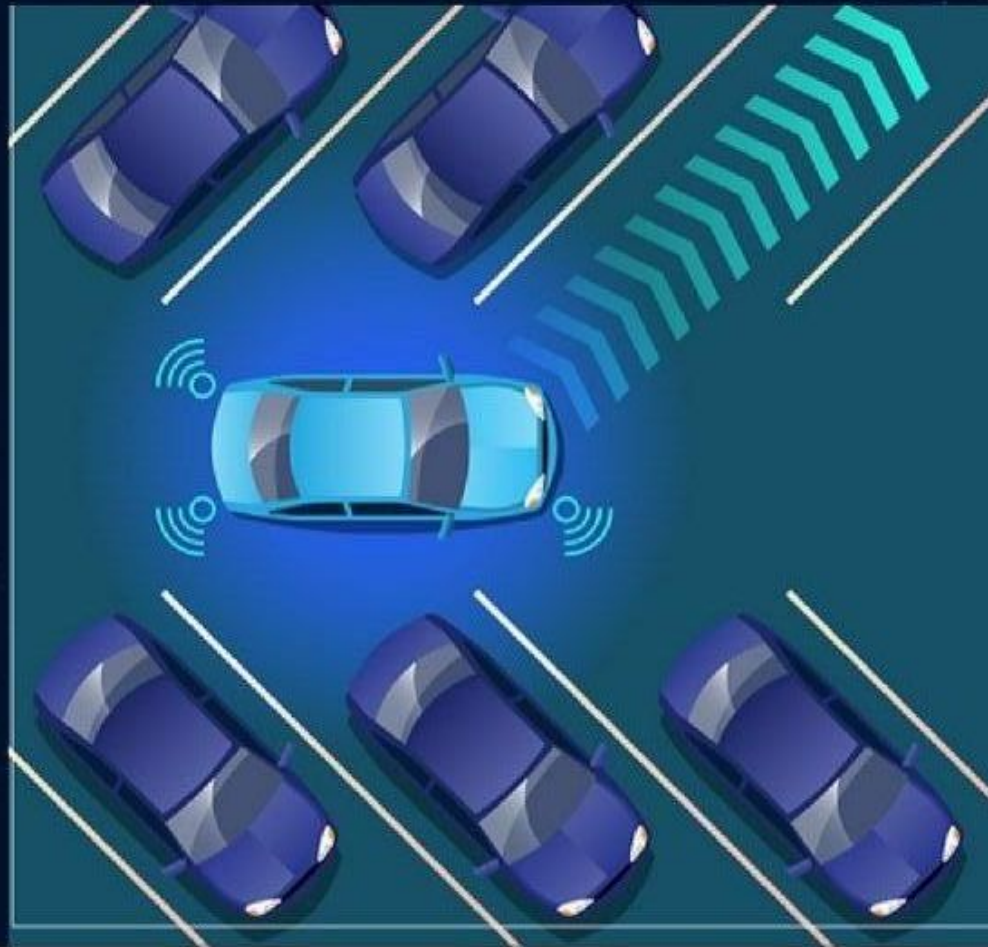
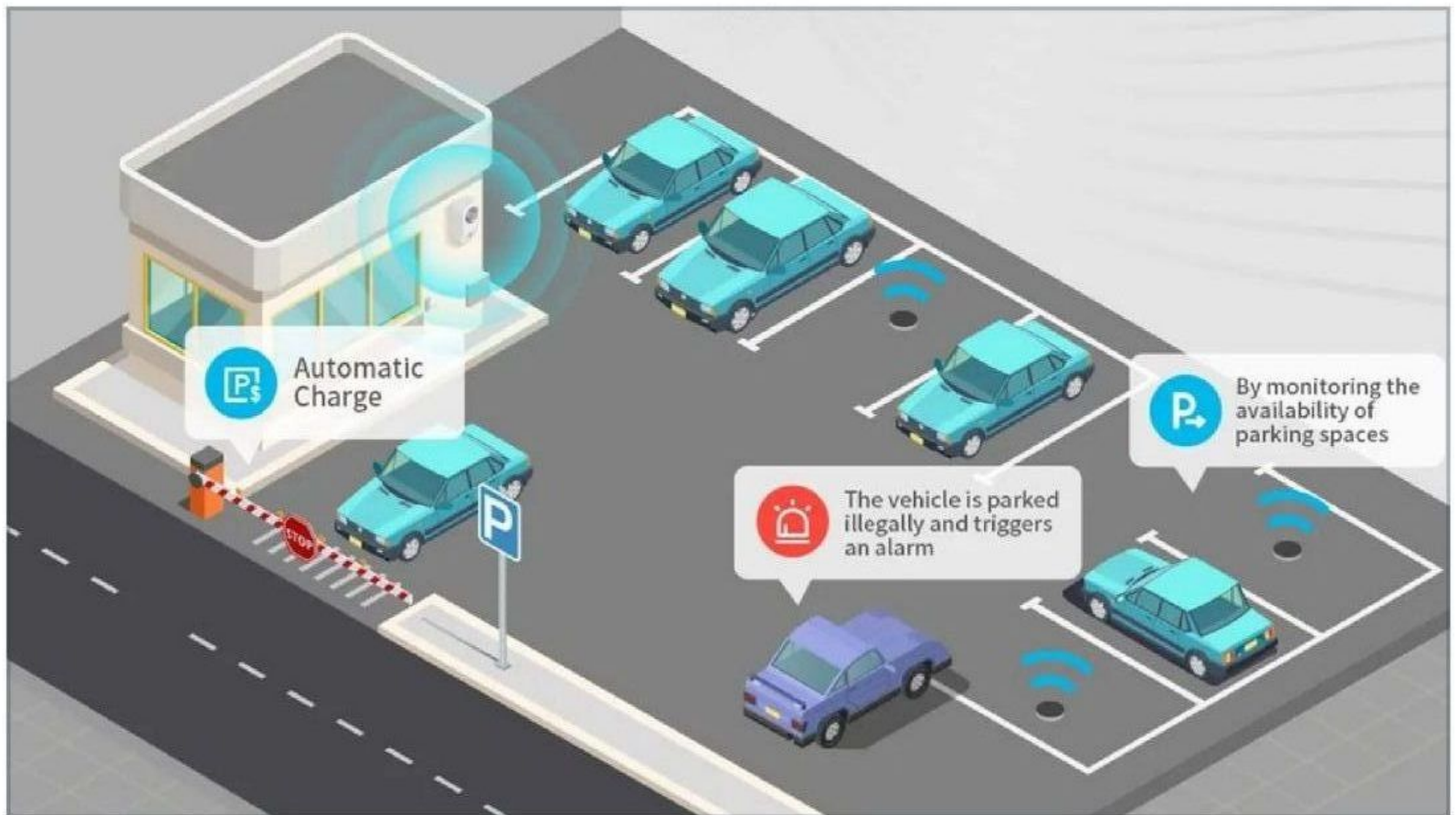


Smart Parking



**IoT-Based
Smart Parking**
System Development

*S. Mohamed
Dhanish*



Hint

Meaning of Smart Parking

How Smart Parking Works

Smart Parking Components

Circuit Diagram Advantages

Effective Time Management

What Is an IoT Based Parking System?

- *An IoT based smart parking system, also known as a connected parking system, is a centralized management system that allows drivers to use a smartphone app to search for and reserve a parking spot.*
- *The system's hardware features sensors that detect available parking slots and communicate this information to all drivers in the area.*

This data is updated in real-time, which means drivers never have to worry about not finding an available space.



HOW DOES IT SMART PARKING WORKS ?

- *Parking systems are installed on the outside of buildings or inside of buildings. When a vehicle enters the space, sensors detect its presence and calculate available parking slots. This information is then sent to the driver's phone via an app*
- *The smart parking system also has real-time data on occupancy rates, which can be found on the app. This data is collected from each sensor and is updated every five minutes.*

What components involved in the smart parking system using IoT?

- *A sensor that can detect the presence of the vehicle.*
 - *A micro control that can help you processing the data.*
 - *A cloud platform will restore the data.*
- *A mobile application enables you to control the smart parking process*



The Operating Principle of Smart Parking

- To detect parked cars in a specific parking lot, an IoT device can use engineering technology to identify their presence and occupancy. This enables a smart parking system to provide searching, navigation, and reservation of parking lots.*

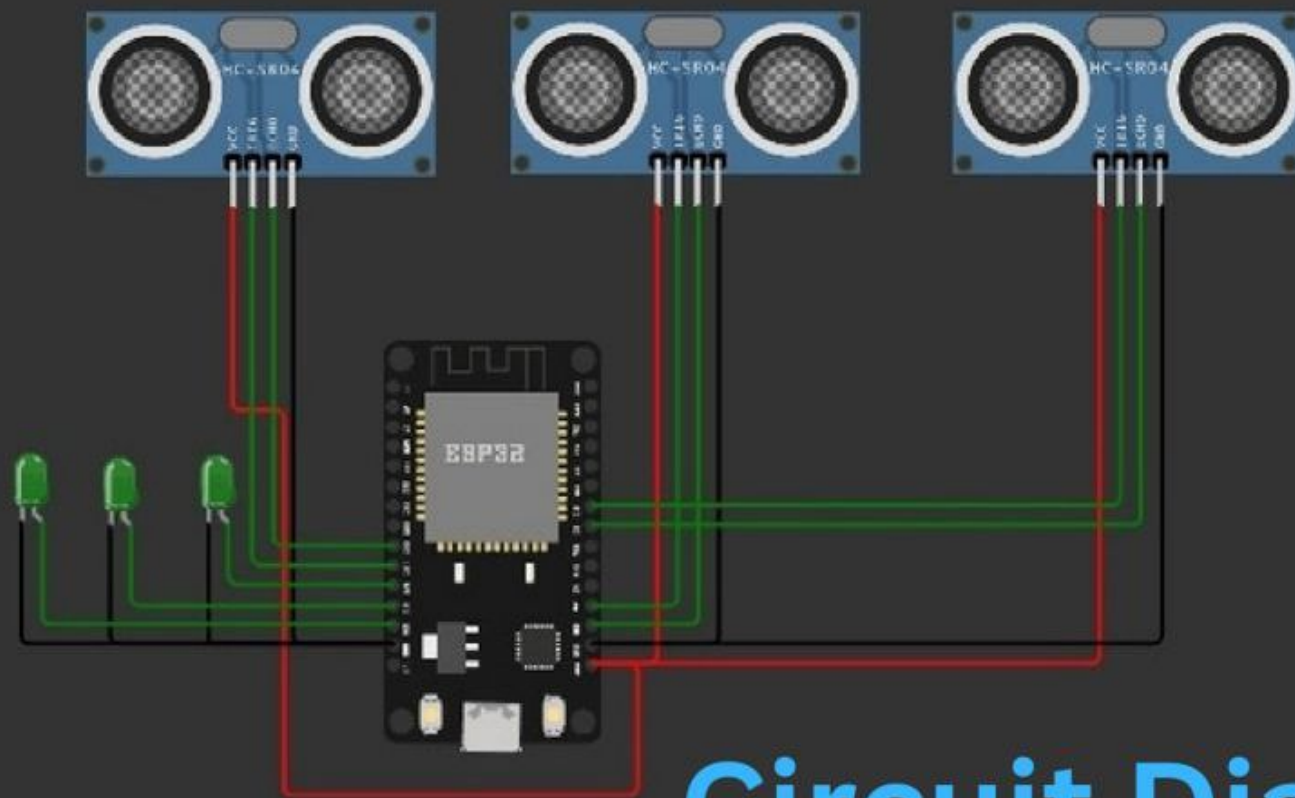


Parking spaces have occupancy sensors that detect the presence, absence, arrival, and departure of vehicles, powered by batteries.

An occupation sensor detects vehicle activity and sends a short message package via an embedded LoRa receiver to any wireless network gateway within its range.

LoRaWAN enables two-way communication, enabling parking lot administrators to request data from sensors.

PIN's parking cloud service shares real-time parking data with other smart city services for municipal and district governments. It uses information from various city infrastructure to provide unique applications, including remote parking enforcement.



Circuit Diagram

Advantages of IoT-based Smart Parking Systems


Optimizing Parking Space

**Provide electric vehicle
charging stations.**

**Locate Emergency Vehicle
Zones.**

**Special permit for loading
and unloading, taxis, and
more.**

*Smart Parking solution can
reduce*



Traffic volume: -8%

Gas Emission:-40%

Km Travelled:-30%

Time speed:-43%

Km. travelled
searching for
parking

-30%

time spent
looking for
parking

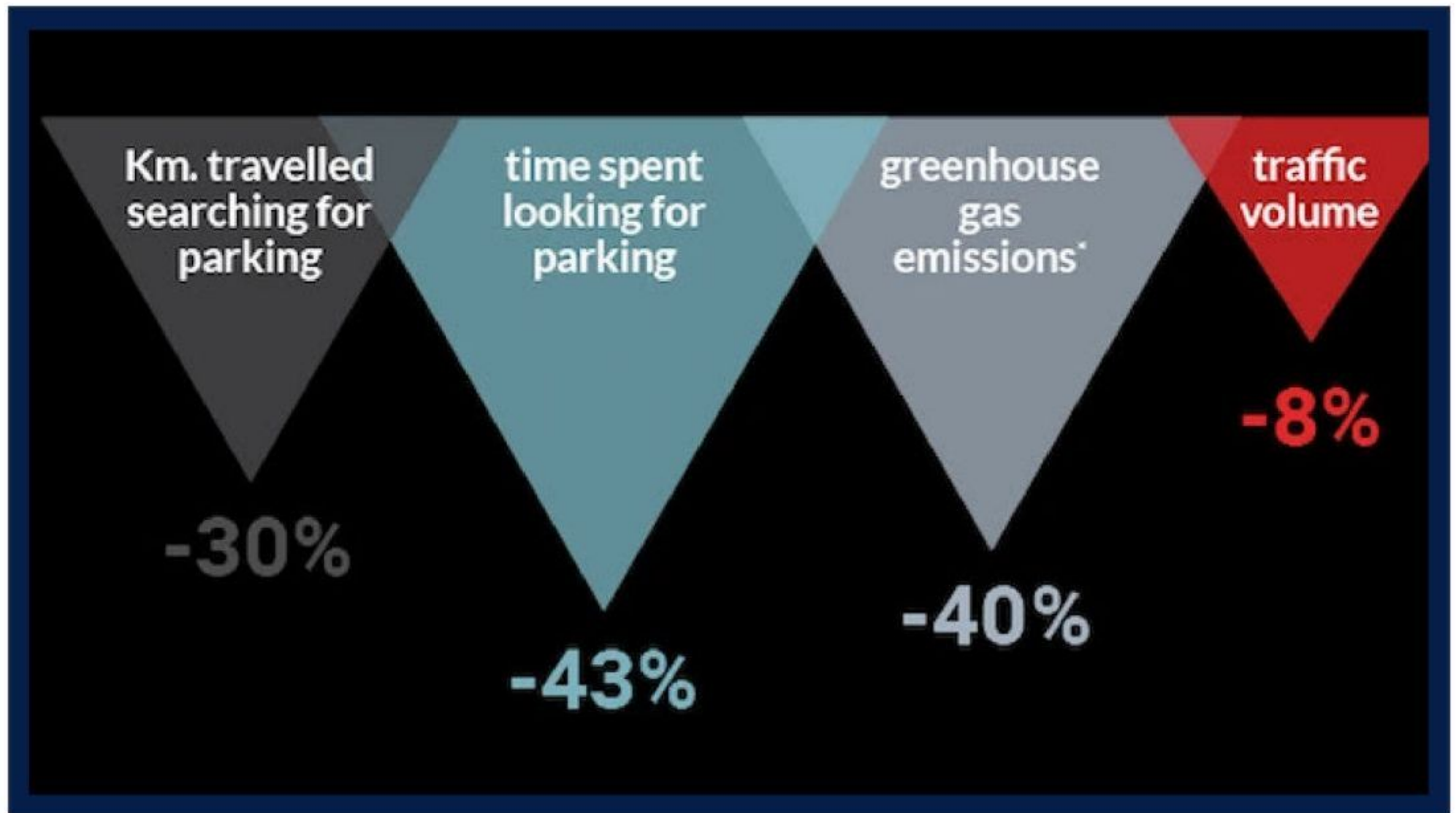
-43%

greenhouse
gas
emissions*

-40%

traffic
volume

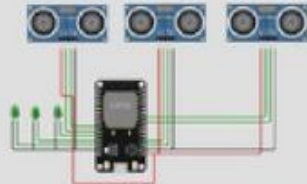
-8%



Code for Smart Parking

```
sketch.ino  diagram.json  Library Manager  Simulation

1 #define ECHO_PIN1 15 //Pin for Sensor 1
2 #define TRIG_PIN1 2 //Pin for Sensor 1
3
4 #define ECHO_PIN2 5 //Pin for Sensor 2
5 #define TRIG_PIN2 18 //Pin for Sensor 2
6
7 #define ECHO_PIN3 26 //Pin for Sensor 3
8 #define TRIG_PIN3 27 //Pin for Sensor 3
9
10
11 int LEDPIN1 = 13;
12 int LEDPIN2 = 12;
13 int LEDPIN3 = 14;
14
15 void setup() {
16   Serial.begin(115200);
17   pinMode(LEDPIN1, OUTPUT);
18   pinMode(TRIG_PIN1, OUTPUT);
19   pinMode(ECHO_PIN1, INPUT);
20
21   pinMode(LEDPIN2, OUTPUT);
22   pinMode(TRIG_PIN2, OUTPUT);
23   pinMode(ECHO_PIN2, INPUT);
24
25   pinMode(LEDPIN3, OUTPUT);
26   pinMode(TRIG_PIN3, OUTPUT);
27   pinMode(ECHO_PIN3, INPUT);
28 }
29
30 float readDistance1CM() {
31   digitalWrite(TRIG_PIN1, LOW);
32   delayMicroseconds(2);
33   digitalWrite(TRIG_PIN1, HIGH);
34   delayMicroseconds(10);
35   digitalWrite(TRIG_PIN1, LOW);
36   int duration = pulseIn(ECHO_PIN1, HIGH);
37   return duration * 0.034 / 2;
38 }
39
40 float readDistance2CM() {
41   digitalWrite(TRIG_PIN2, LOW);
42   delayMicroseconds(2);
43   digitalWrite(TRIG_PIN2, HIGH);
44   delayMicroseconds(10);
45   digitalWrite(TRIG_PIN2, LOW);
46   int duration = pulseIn(ECHO_PIN2, HIGH);
47   return duration * 0.034 / 2;
48 }
49
50 float readDistance3CM() {
51   digitalWrite(TRIG_PIN3, LOW);
52   delayMicroseconds(2);
53   digitalWrite(TRIG_PIN3, HIGH);
54   delayMicroseconds(10);
55   digitalWrite(TRIG_PIN3, LOW);
56   int duration = pulseIn(ECHO_PIN3, HIGH);
57   return duration * 0.034 / 2;
58 }
59
60 void loop() {
61   float distance1 = readDistance1CM();
62   float distance2 = readDistance2CM();
63   float distance3 = readDistance3CM();
64
65   bool isNearby1 = distance1 > 200;
```



sketch.ino

diagram.json

Library Manager

Sim

```
62 float distance2 = readDistance2CM();
63 float distance3 = readDistance3CM();
64
65 bool isNearby1 = distance1 > 200;
66 digitalWrite(LEDPIN1, isNearby1);
67
68
69 bool isNearby2 = distance2 > 200;
70 digitalWrite(LEDPIN2, isNearby2);
71
72
73 bool isNearby3 = distance3 > 200;
74 digitalWrite(LEDPIN3, isNearby3);
75
76 Serial.print("Measured distance: ");
77 Serial.println(readDistance1CM());
78 Serial.println(readDistance2CM());
79 Serial.println(readDistance3CM());
80 delay(100);
81 }
```

```

1
2 "version": 1,
3 "author": "Surya K",
4 "editor": "wokwi",
5 "parts": [
6   { "type": "wokwi-esp32-devkit-v1", "id": "esp", "top": 168.01, "left": -54.47, "attrs": {} },
7   { "type": "wokwi-hc-sr04", "id": "ultrasonic1", "top": 10.18, "left": 222.47, "attrs": {} },
8   { "type": "wokwi-hc-sr04", "id": "ultrasonic2", "top": 10.18, "left": 7.37, "attrs": {} },
9   { "type": "wokwi-hc-sr04", "id": "ultrasonic3", "top": 11.1, "left": -199.42, "attrs": {} },
10  ],
11  { "type": "wokwi-led",
12    "id": "led1",
13    "top": 215.93,
14    "left": -245.43,
15    "attrs": { "color": "green" }
16  },
17  {
18    "type": "wokwi-led",
19    "id": "led2",
20    "top": 217.94,
21    "left": -202.14,
22    "attrs": { "color": "green" }
23  },
24  {
25    "type": "wokwi-led",
26    "id": "led3",
27    "top": 216.99,
28    "left": -154.69,
29    "attrs": { "color": "green" }
30  }
31 ],
32 "connections": [
33   [ "esp:TX0", "SerialMonitor:RX", "", [] ],
34   [ "esp:RX0", "SerialMonitor:TX", "", [] ],
35   [ "ultrasonic1:VCC", "esp:3V3", "red", [ "v0" ] ],
36   [ "ultrasonic1:GND", "esp:GND.1", "black", [ "v0" ] ],
37   [ "ultrasonic2:VCC", "esp:3V3", "red", [ "v0" ] ],
38   [
39     "ultrasonic3:VCC",
40     "esp:3V3",
41     "red",
42     [ "v97.89", "h25.31", "v186.97", "h171.78", "v-63.59" ]
43   ],
44   [ "ultrasonic3:GND", "esp:GND.2", "black", [ "v0" ] ],
45   [ "led1:C", "esp:GND.2", "black", [ "v0" ] ],
46   [ "led2:C", "esp:GND.2", "black", [ "v0" ] ],
47   [ "led3:C", "esp:GND.2", "black", [ "v0" ] ],
48   [ "led1:A", "esp:D13", "green", [ "v0" ] ],
49   [ "led2:A", "esp:D12", "green", [ "v0" ] ],
50   [ "led3:A", "esp:D14", "green", [ "v0" ] ],
51   [ "ultrasonic1:TRIG", "esp:D27", "green", [ "v0" ] ],
52   [ "ultrasonic1:ECHO", "esp:D26", "green", [ "v0" ] ],
53   [ "ultrasonic2:GND", "esp:GND.1", "black", [ "v0" ] ],
54   [ "ultrasonic2:ECHO", "esp:D15", "green", [ "v0" ] ],
55   [ "ultrasonic2:TRIG", "esp:D2", "green", [ "v0" ] ],
56   [ "ultrasonic3:ECHO", "esp:D5", "green", [ "v0" ] ],
57   [ "ultrasonic3:TRIG", "esp:D18", "green", [ "v0" ] ]
58 ],
59 "dependencies": {}
60

```


Thank You