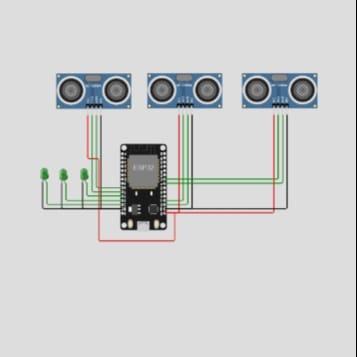
**PROJECT TITLE : SMART PRAKING**

****

**CODE:**

ECHO\_PIN1 = 15 # Pins for Sensor 1

TRIG\_PIN1 = 2 # Pins for Sensor 1

ECHO\_PIN2 = 5 # Pins for Sensor 2

TRIG\_PIN2 = 18 # Pins for Sensor 2

ECHO\_PIN3 = 26 # Pins for Sensor 3

TRIG\_PIN3 = 27 # Pins for Sensor 3

LEDPIN1 = 13

LEDPIN2 = 12

LEDPIN3 = 14

def setup():

Serial.begin(115200)

pinMode(LEDPIN1, OUTPUT)

pinMode(TRIG\_PIN1, OUTPUT)

pinMode(ECHO\_PIN1, INPUT)

pinMode(LEDPIN2, OUTPUT)

pinMode(TRIG\_PIN2, OUTPUT)

pinMode(ECHO\_PIN2, INPUT)

pinMode(LEDPIN3, OUTPUT)

pinMode(TRIG\_PIN3, OUTPUT)

pinMode(ECHO\_PIN3, INPUT)

def readDistance1CM():

digitalWrite(TRIG\_PIN1, LOW)

delayMicroseconds(2)

digitalWrite(TRIG\_PIN1, HIGH)

delayMicroseconds(10)

digitalWrite(TRIG\_PIN1, LOW)

duration = pulseIn(ECHO\_PIN1, HIGH)

return duration \* 0.034 / 2

def readDistance2CM():

digitalWrite(TRIG\_PIN2, LOW)

delayMicroseconds(2)

digitalWrite(TRIG\_PIN2, HIGH)

delayMicroseconds(10)

digitalWrite(TRIG\_PIN2, LOW)

duration = pulseIn(ECHO\_PIN2, HIGH)

return duration \* 0.034 / 2

def readDistance3CM():

digitalWrite(TRIG\_PIN3, LOW)

delayMicroseconds(2)

digitalWrite(TRIG\_PIN3, HIGH)

delayMicroseconds(10)

digitalWrite(TRIG\_PIN3, LOW)

duration = pulseIn(ECHO\_PIN3, HIGH)

return duration \* 0.034 / 2

def loop():

distance1 = readDistance1CM()

distance2 = readDistance2CM()

distance3 = readDistance3CM()

isNearby1 = distance1 > 200

digitalWrite(LEDPIN1, isNearby1)

isNearby2 = distance2 > 200

digitalWrite(LEDPIN2, isNearby2)

isNearby3 = distance3 > 200

digitalWrite(LEDPIN3, isNearby3)

print("Measured distance: ")

print(readDistance1CM())

print(readDistance2CM())

print(readDistance3CM())

time.sleep(0.1)

CODE EXPLANATION:

* The code starts by declaring the pins that are used for each sensor.
* The first two sensors use pin 1 and 2, while the third uses pin 3.
* The next line sets up a variable called "pinMode" which is used to set the digital output mode of a given pin on an Arduino board.
* In this case, all three pins are set as OUTPUTs so they can be connected to LEDs or other components in your project.
* Next, there's some setup code that starts serial communication with the computer and then initializes each of the three sensors using their respective pins (TRIG\_PIN1, ECHO\_PIN1, TRIG\_PIN2, ECHO\_PIN2) Finally there's some code that turns on LED1 and LED3 at different times during operation
* The code is used to initialize the pins for the three sensors that will be used in this project.
* The setup() function starts by initializing the serial port and then setting up each of the three sensors.
* The code starts by setting the TRIG\_PIN1 and TRIG\_PIN2 pins to LOW.
* This is done so that when the code runs, it will not interfere with any other circuits in the board.
* The next line of code sets a delayMicroseconds(10) before turning on both pins again.
* The next line of code reads how long it takes for one pulse to go from pin 1 (TRIG\_PIN1) to pin 2 (TRIG\_PIN2).
* It then divides this number by two and returns that as a distance measurement in centimeters.
* The loop() function starts at line 9, which means that every time through the loop(), we will run lines 8-9 first, then lines 10-12, etc., until we reach line 13 where we stop running through the loop().
* Line 12 calculates how many centimeters are between each pair of pins using readDistance3CM().
* Then these values are added together to get a total distance value for all three pairs of pins.
* The code will loop for a certain amount of time and then read the distance to the object.
* The first function, readDistance1CM(), is used to measure the distance from the sensor to an object.
* The second function, readDistance2CM(), is used to measure the distance from the sensor to an object.
* The third function, readDistance3CM(), is used to measure the distance from the sensor to an object.
* The code starts by declaring variables for the distance1, distance2, and distance3.
* The readDistance3CM() function is then called to get the three distances in centimeters.
* The next line declares a boolean variable named isNearby1 that will be used to determine if the first person's measured distance is within 200 cm of the second person's measured distance.
* If it is, LEDPIN1 will turn on; otherwise it won't turn on.
* Next, another boolean variable named isNearby2 will be declared and set to true if the second person's measured distance falls within 200 cm of the third person's measured distance.
* If so, LEDPIN2 turns on; otherwise it doesn't turn on.
* Finally, a third boolean variable named isNearby3 will be declared and set to true if the third person's measured distance falls within 200 cm of all other people who have been tested so far (i.e., those two previous lines).
* In this case LEDPIN3 turns on; otherwise it doesn't turn on
* The code is meant to measure the distance from three different points in centimeters and then print the result.

DESCRIPTION:

Application Layer:

# AI&DS

The application layer is the top layer of the architecture stack that allows the partici-

Pants to interact with the system that they use, the mobile application (i.e., Android and

iOS), or the Web application. Here, as highlighted by Yang et al. [23], users are capable of

searching for their preferred parking locations, and they can make reservations. Similarly,

the parking services provider can send parking-related information, e.g., parking space

availability, to the providers and the offers to the integrated systems. Since the users

interact with the integrated system directly, the layer delivers the end-users’ final service.

## Network layer:

The network layer ensures seamless communication among the various parking cen-

Ters, integrated systems, and users. The user and parking center data are transmitted to the

Integrated system through a layer. The layer contains the different types of communication

Technologies that may include LAN and WAN, which are used by the users, parking

Service providers, and the IoT devices related to the parking systems (e.g., the parking

### Physics layer:

The physical layers deal specifically with the mechanisms and the electronic anchorage

of the system. The physical layer is based on the set of the physical sensors and the data

received from the entities collected that are analyzed and used to manage the entities.

The different types of sensors are the significant elements of the layer.

Transaction layer:

This is the layer that is mandated to transact the nodes in the network. The users and

the various parking centers exchange the data more securely through the smart contract

and the consensus mechanisms. The parking center also updates the public ledger through

the layer. The transaction layer preserves the transparent quality of the transaction and the

security of the data transmission without trusted third parties, especially if they rely on

Blockchainsystems

# DAS

Smart parking refers to the use of technology to improve the efficiency and management of

parking spaces. It often involves sensors, cameras, and data analytics to provide real-time

information about parking availability. Smart parking systems can help reduce congestion, save

time for drivers, and make better use of urban space. If you have specific questions or need more

information about smart parking, please feel free to ask.

# IOT

These systems employ various sensors, communication networks, and software to provide real-

Time information about parking availability. Here’s a brief overview of how it works:

Sensor Deployment: Sensors are installed in each parking space. These can be ultrasonic,

Infrared, magnetic, or other types of sensors.

Data Collection: The sensors collect data about the status of each parking space.

Centralized System: The central server processes the best 3uhdata received from the sensors.

User Interface: A user-friendly interface is provided t to end-users, typically through a mobile app

Or a website.

Navigation and Guidance: Some systems also include navigation features to guide drivers to the

Nearest available parking spot.

Maintenance and Monitoring: Regular maintenance and monitoring of the sensors and

Communication network are crucial to ensure the system operates smoothly.

# CAD

It seems like you’re looking for information about smart parking cards. Smart parking cards can

refer to various technologies and systems used for managing parking, including RFID cards, NFC

cards, or mobile apps that facilitate parking payments and access control. If you have specific

questions or need more details, please let me know!