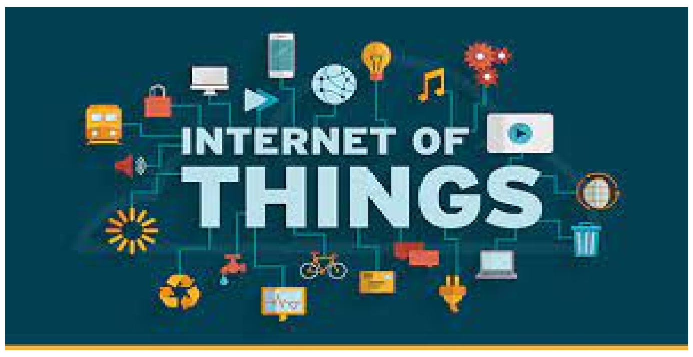
SMART PARKING SYSTEM USING IOT

PHASE -3 : DEVELOPMENT PART-1



SUBMITTED BY:

TEAM LEADER - VINITHA .V (510921106045)

TEAM MEMBER – NANDHINI .B (51092110617)

TEAM MEMBER – RAGAVI .S (510921106027)

TEAM MEMBER –GAYATHRI .M (510921106301)

Components Required:

1.**Ultrasonic Sensor**:

The ultrasonic sensor is a device that uses ultrasonic sound waves to measure distances. It typically has two main components: a transmitter and a receiver. The transmitter emits ultrasonic pulses, and the receiver detects the reflection of these pulses to calculate the distance to an object.

2. **LEDs (Light Emitting Diodes)**:

LEDs are used as visual indicators in electronic projects. They emit light when an electrical current flows through them. In your project, you have three LEDs, often categorized by their colors, such as red, green, and blue.

3.** 330-ohm Resistor**:

A resistor is an electrical component that limits the flow of electric current. In your project, the 330-ohm resistor is used to limit the current flowing through the LEDs. It's important for preventing them from burning out.

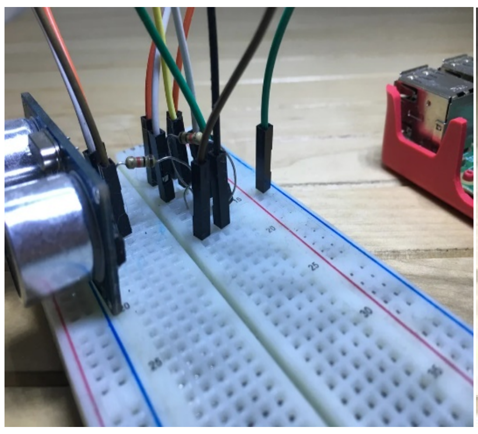
4. **10-ohm Resistor**: This resistor is likely used for a specific purpose in your project. The value of the resistor depends on the application and the specific requirements of the project. If you could provide more context, I can give more specific guidance.

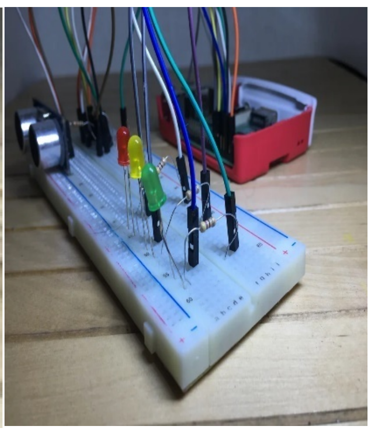
5. **Jumper Wires**:

Jumper wires are used for making electrical connections between different components on a breadboard or connecting components to a Raspberry Pi's GPIO pins. They come in various lengths and can be male-to-male, male-to-female, or female-to-female.

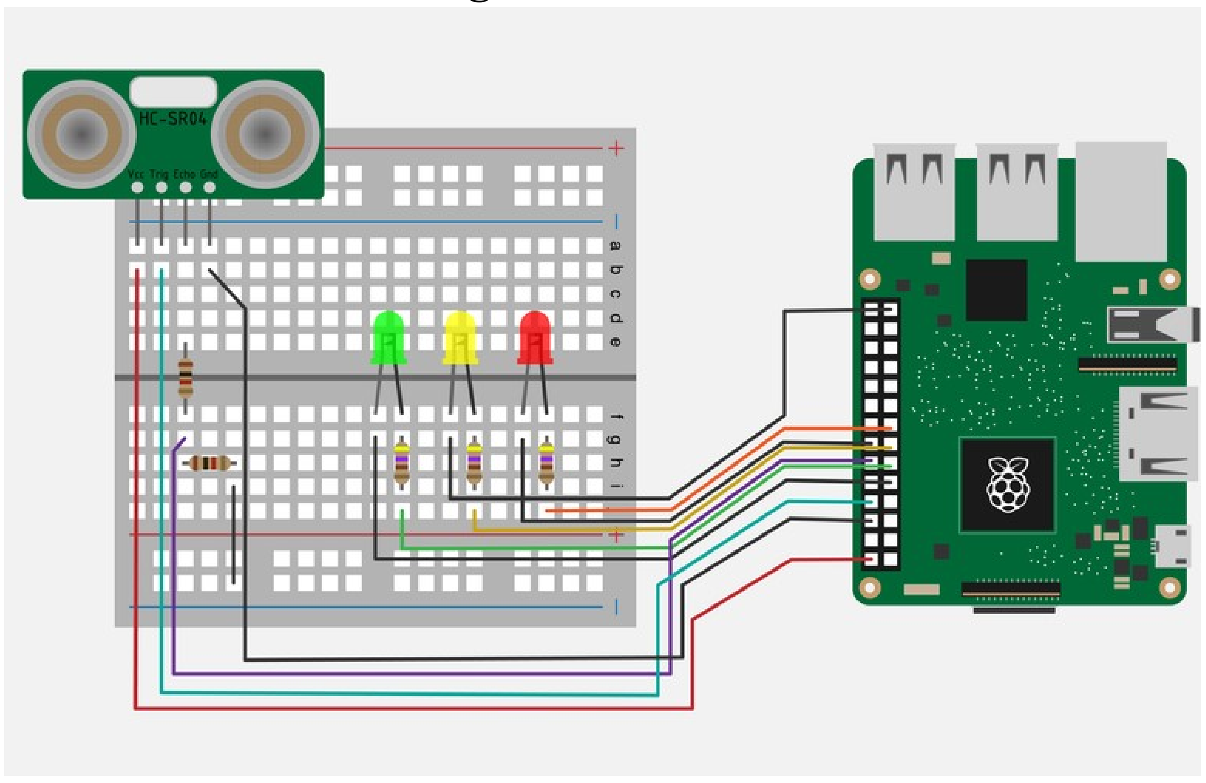
6. **Half Breadboard**:

A breadboard is a tool for prototyping electronic circuits. The half breadboard provides a platform for you to connect and test your components without soldering. It has a grid of holes for inserting components and creating electrical connections.





Circuit Simulation Diagram:



Python code using Raspberry Pi:

```
import RPi.GPIO as GPIO

import time

# Define GPIO pins

TRIG = 23  # Ultrasonic sensor trigger pin

ECHO = 24  # Ultrasonic sensor echo pin

LED1 = 17  # LED1 pin

LED2 = 27  # LED2 pin

LED3 = 22  # LED3 pin

# Set GPIO mode and setup

GPIO.setmode(GPIO.BCM)

GPIO.setup(TRIG, GPIO.OUT)

GPIO.setup(ECHO, GPIO.IN)

GPIO.setup(LED1, GPIO.OUT)

GPIO.setup(LED2, GPIO.OUT)

GPIO.setup(LED3, GPIO.OUT)

# Function to measure distance with the ultrasonic sensor

def measure_distance():

    GPIO.output(TRIG, True)

    time.sleep(0.00001)

    GPIO.output(TRIG, False)


while GPIO.input(ECHO) == 0:

        pulse_start = time.time()

 while GPIO.input(ECHO) == 1:

        pulse_end = time.time()

  pulse_duration = pulse_end - pulse_start

  distance = (pulse_duration * 34300) / 2  # Speed of sound = 34300 cm/s

  return distance
 try:
    while True:
            distance = measure_distance()
             print("Distance: {:.2f} cm".format(distance))
             if distance < 10:
                    GPIO.output(LED1, GPIO.HIGH)
                   GPIO.output(LED2, GPIO.LOW)
                    GPIO.output(LED3, GPIO.LOW)
           elif 10 <= distance < 20:
                    GPIO.output(LED1, GPIO.HIGH)


                    GPIO.output(LED2, GPIO.HIGH)
                    GPIO.output(LED3, GPIO.LOW)
        else:
           GPIO.output(LED1, GPIO.HIGH)
           GPIO.output(LED2, GPIO.HIGH)
           GPIO.output(LED3, GPIO.HIGH)


except KeyboardInterrupt:
    GPIO.cleanuop()
```

Note: This code sets up the ultrasonic sensor and three LEDs. The LEDs will light up in different patterns depending on the measured distance from the ultrasonic sensor. You can modify the distance thresholds and LED patterns to suit your project's needs.