SMART PARKING

PHASE-3 DEVELOPMENT PART 1

Team member: Abhijay.S.H

Exam no: 210521106002

Domain: IoT

# Definition:

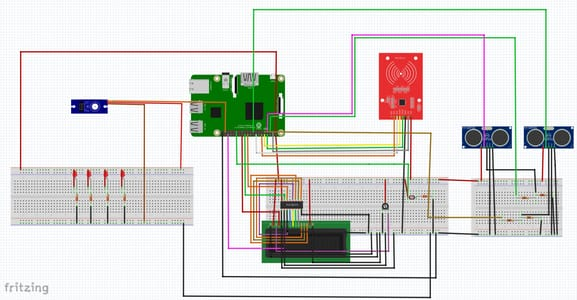
In urban areas, parking congestion has become a significant challenge, leading to frustration, wasted time, and increased pollution. The traditional methods of finding parking spaces often result in inefficiencies, traffic congestion, and environmental pollution. Drivers struggle to locate available parking spots, leading to unnecessary fuel consumption and emissions. Moreover, urban planners lack accurate, real-time data to optimize parking space allocation.

# COMPONENTS USED :

* 1. LED
  2. IR SENSOR
  3. RASPBERRY PI

# WORKING & CIRCUIT DIAGRAM :

Building an IoT sensor system for parking space occupancy detection involves integrating sensors with a Raspberry Pi and writing Python scripts to collect data and send it to the cloud or a mobile app server. Below are the steps to get you started with Phase 3: Development Part 1.



# CODING:

import RPi.GPIO as GPIO

import time

import requests

GPIO.setmode(GPIO.BCM)

TRIG = 17

ECHO = 18

GPIO.setup(TRIG, GPIO.OUT)

GPIO.setup(ECHO, GPIO.IN)

def measure\_distance():

GPIO.output(TRIG, True)

time.sleep(0.00001)

GPIO.output(TRIG, False)

pulse\_start, pulse\_end = 0, 0

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 \* 17150 # Speed of sound = 343 m/s, distance = time \* speed / 2 (since it's a round trip)

distance = round(distance, 2)

return distance

while True:

try:

distance = measure\_distance()

print(f"Distance: {distance} cm")

# Send data to the cloud or mobile app server

# Replace 'YOUR\_API\_ENDPOINT' with the actual endpoint to send data

requests.post('YOUR\_API\_ENDPOINT', json={'distance': distance})

time.sleep(1) # Wait for 1 second before taking the next reading

except KeyboardInterrupt:

GPIO.cleanup()

break