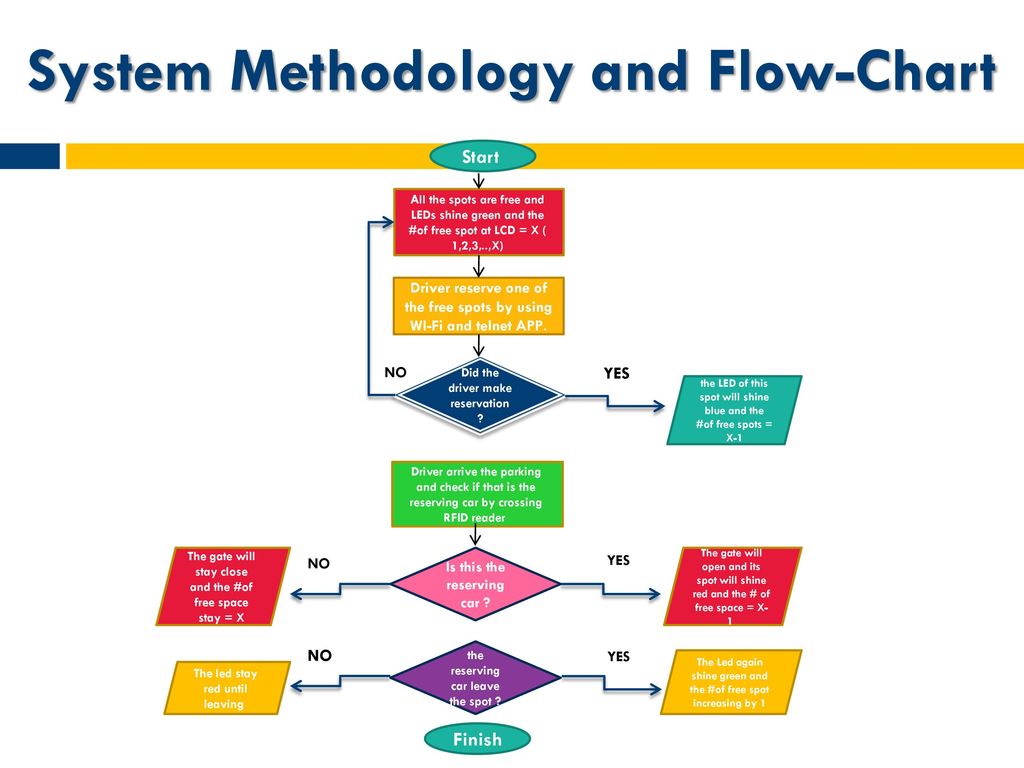
**SMART PARKING SYSTEM**

**Phase 3: Development Part 1**

***Flow chart for a smart parking system:***



***System Implementation using IoT Sensor system and Raspberry Pi Integration:***

**SENSOR MODULE**

**ANDROID MODULE**

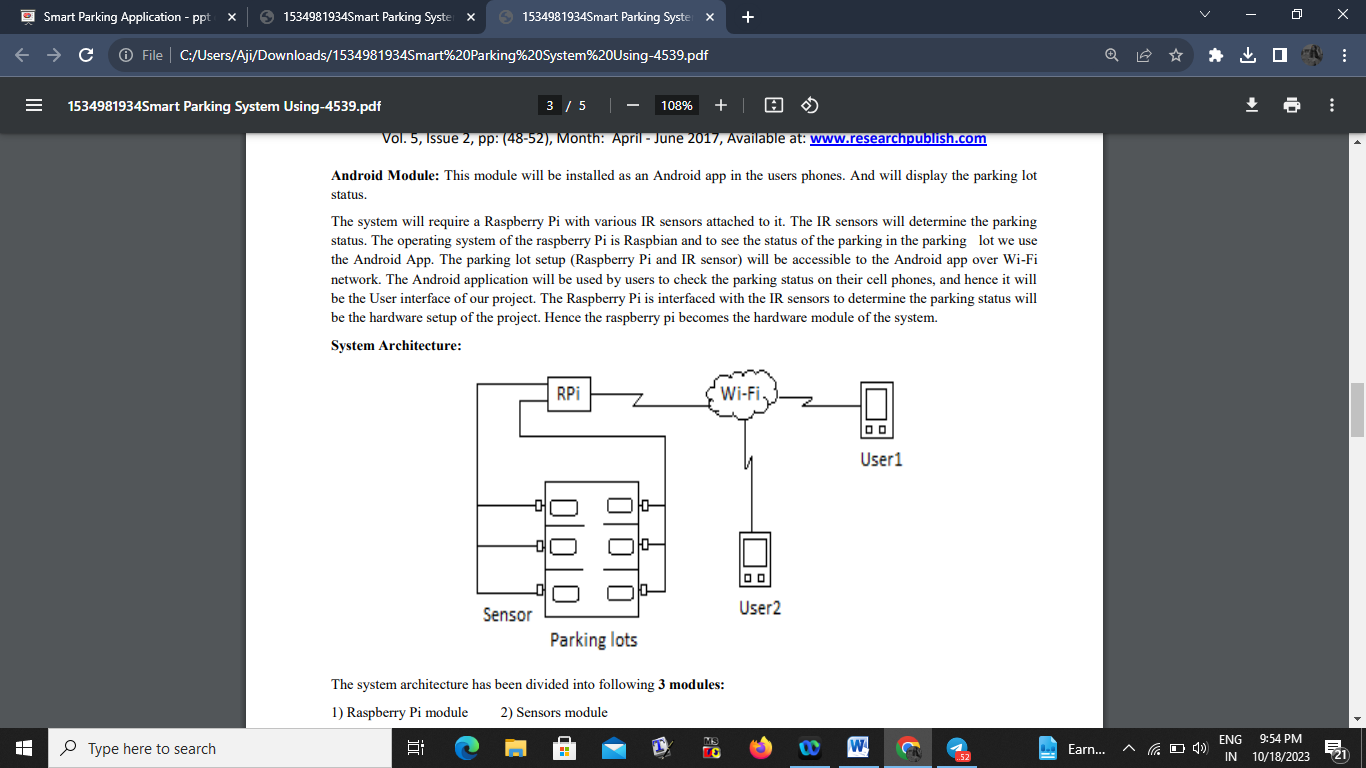
**Sensor Module:** This module will be installed in the parking place, there will be a sensor node for each parking space. The sensors which will be used are IR- sensors.

**Raspberry Pi Module:** This module will be connected to the internet and will have connections from all sensor nodes.

**Android Module:** This module will be installed as an Android app in the users phones. And will display the parking lot status.

The system will require a Raspberry Pi with various IR sensors attached to it. The IR sensors will determine the parking status. The operating system of the raspberry Pi is Raspbian and to see the status of the parking in the parking lot we use the Android App. The parking lot setup (Raspberry Pi and IR sensor) will be accessible to the Android app over Wi-Fi network. The Android application will be used by users to check the parking status on their cell phones, and hence it will be the User interface of our project. The Raspberry Pi is interfaced with the IR sensors to determine the parking status will be the hardware setup of the project. Hence the raspberry pi becomes the hardware module of the system.

***SYSTEM ARCHITECTURE:***



**Algorithm:**

**Sensor and Raspberry Pi Module:**

Step 1: Initialize the System (IR Sensors and raspberry Pi)

Step 2: Read sensor status and store in the Database

Step3: Send Sensor Status to Android Application when requested

Step 4: Continue through Step 3

Step 5: Stop

**Android Application Module:**

Step 1: Start Android App

Step 2: Check connectivity with Server (Raspberry Pi)

Step 3: Request Sensor Status after every time interval

Step4: Continue through Step 3

Step 5: Stop

***Python program for smart parking system:***

class ParkingSpace:

def \_\_init\_\_(self, name):

self.name = name

self.is\_occupied = False

def occupy(self):

self.is\_occupied = True

def vacate(self):

self.is\_occupied = False

class SmartParkingSystem:

def \_\_init\_\_(self, num\_spaces):

self.parking\_spaces = [ParkingSpace(f'Space {i+1}') for i in range(num\_spaces)]

def park\_vehicle(self, space\_number):

if 1 <= space\_number <= len(self.parking\_spaces):

space = self.parking\_spaces[space\_number - 1]

if not space.is\_occupied:

space.occupy()

print(f'Parked vehicle in {space.name}')

else:

print(f'Space {space.name} is already occupied.')

else:

print('Invalid space number.')

def vacate\_space(self, space\_number):

if 1 <= space\_number <= len(self.parking\_spaces):

space = self.parking\_spaces[space\_number - 1]

if space.is\_occupied:

space.vacate()

print(f'Vacated space {space.name}')

else:

print(f'Space {space.name} is already vacant.')

else:

print('Invalid space number.')

def check\_availability(self):

for i, space in enumerate(self.parking\_spaces):

status = "Occupied" if space.is\_occupied else "Vacant"

print(f'Space {i + 1}: {status}')

def main():

num\_spaces = 5 # Number of parking spaces

parking\_system = SmartParkingSystem(num\_spaces)

while True:

print("\nSmart Parking System Menu:")

print("1. Park Vehicle")

print("2. Vacate Space")

print("3. Check Parking Space Availability")

print("4. Exit")

choice = input("Enter your choice: ")

if choice == '1':

space\_number = int(input("Enter the space number to park the vehicle: "))

parking\_system.park\_vehicle(space\_number)

elif choice == '2':

space\_number = int(input("Enter the space number to vacate: "))

parking\_system.vacate\_space(space\_number)

elif choice == '3':

parking\_system.check\_availability()

elif choice == '4':

print("Exiting the Smart Parking System.")

break

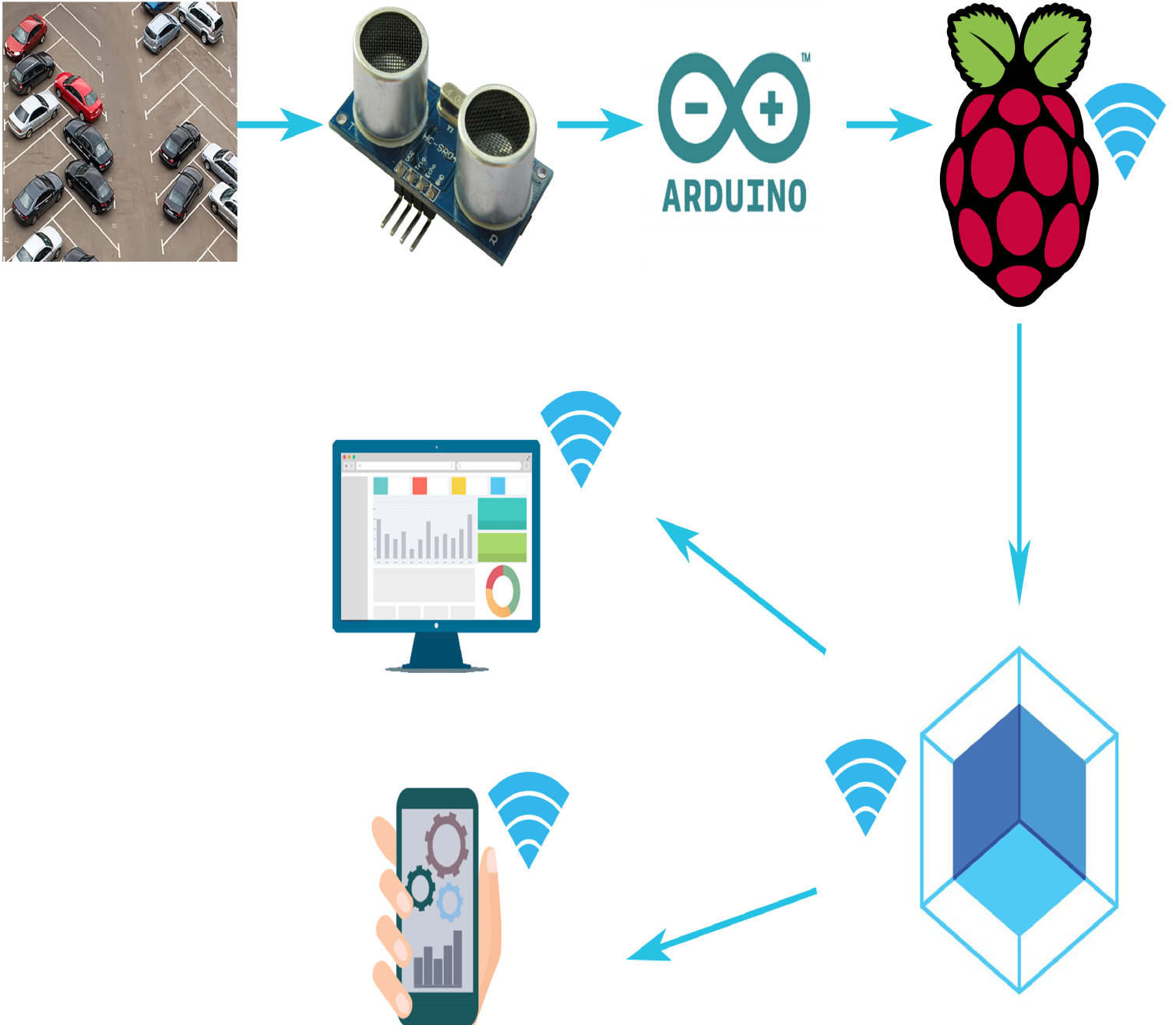
else:

print("Invalid choice. Please select a valid option.")

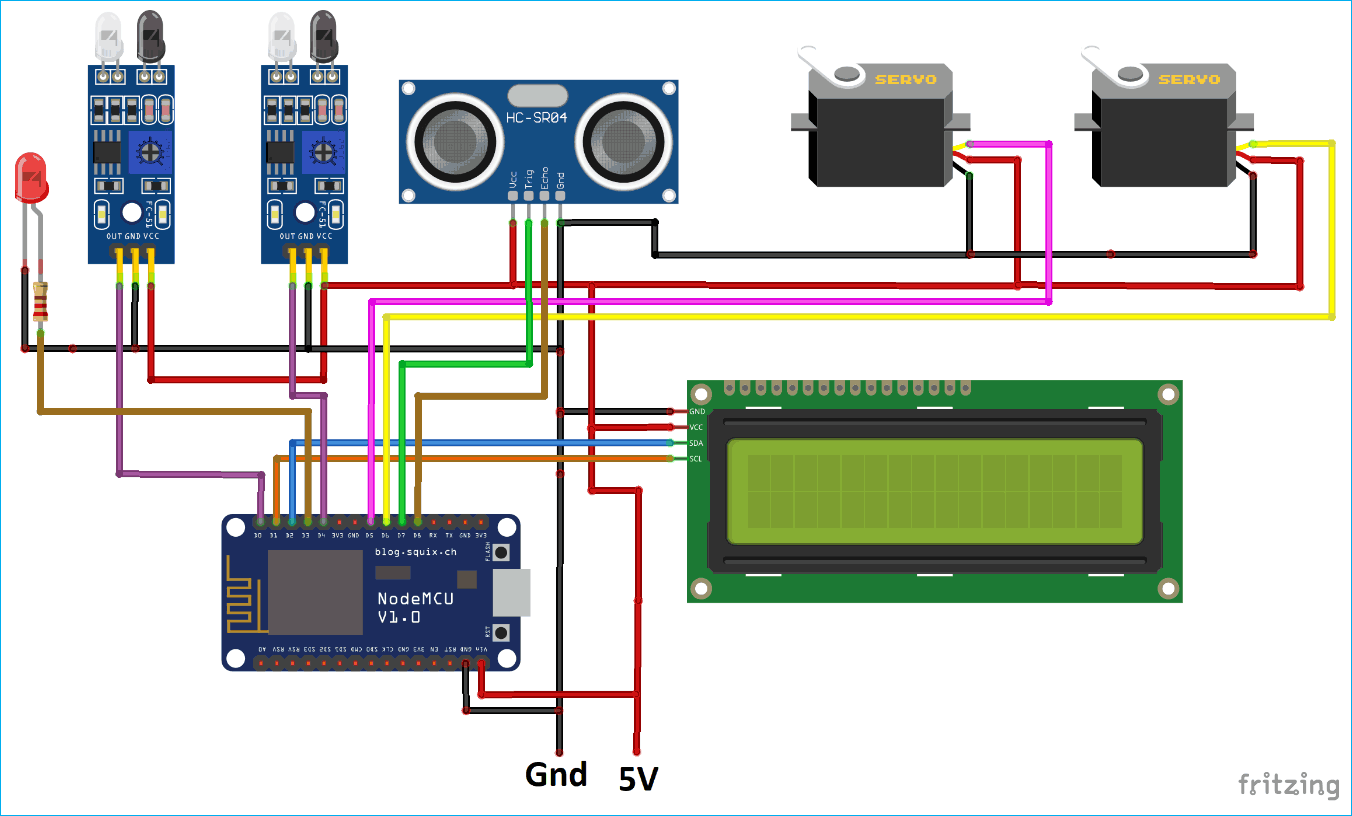
if \_\_name\_\_ == "\_\_main\_\_":

main()

***SCHEMATIC REPRESENTATION:***



***RASBPERRY PI REPRESENTATION:***



***DEMO OF SMART PARKING SYSTEM:***

