AIM:

To implement smart parking system which can provide :

a.) automatic billing generation

b.) automatic report generation.

DESCRIPTION:

Car parking is a major issue in modern congested cities of today. There are too many vehicles on the road and not enough parking spaces. Getting stuck in the vehicle because of parking is quite complicated due to less space and time. Moreover, a manually handled car parking lot can be stressful like anything. This is not only stressful for the visitors but also for every parking owner. Such a situation has led to the need for efficient parking management systems.

The system uses ultrasonic sensors to detect either car park occupancy . Different detection technologies are reviewed and compared to determine the best technology for developing SPS. Features of SPS include vacant parking space detection, a billing technology.

HARDWARE AND SOFTWARE REQUIREMENTS:

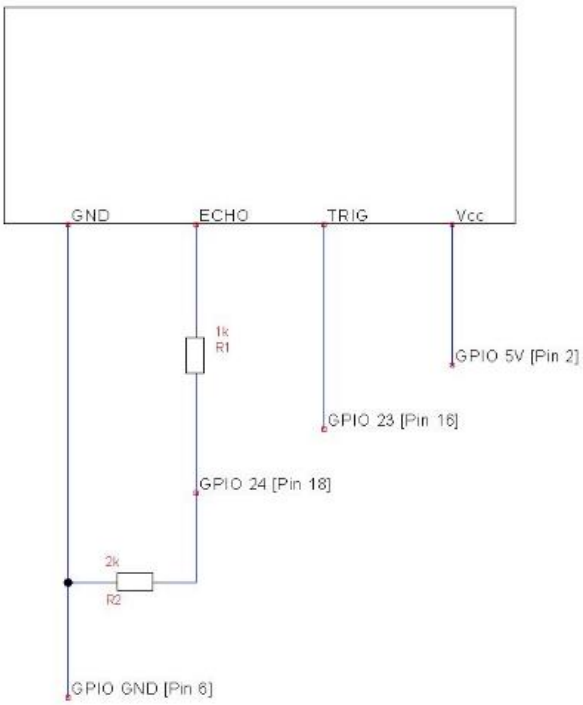
Raspberry pi

Ultrasonic sensors

male and female jumper wires

A/c adapter.

CIRCUIT DIAGRAM:



CODE:

```
#Libraries
import RPi.GPIO as GPIO
import time
#GPIO Mode (BOARD / BCM)
GPIO.setmode(GPIO.BCM)
#set GPIO Pins
GPIO_TRIGGER = 23
GPIO_ECHO = 24
StartTime = time.time()
StopTime = time.time()
#set GPIO direction (IN / OUT)
GPIO.setup(GPIO_TRIGGER, GPIO.OUT)
GPIO.setup(GPIO_ECHO, GPIO.IN)
def distance():
        # set Trigger to HIGH
        GPIO.output(GPIO_TRIGGER, True)
        # set Trigger after 0.01ms to LOW
        time.sleep(0.00001)
        GPIO.output(GPIO_TRIGGER, False)
        # save StartTime
        while GPIO.input(GPIO_ECHO) == 0:
        StartTime = time.time()


        # save time of arrival
```

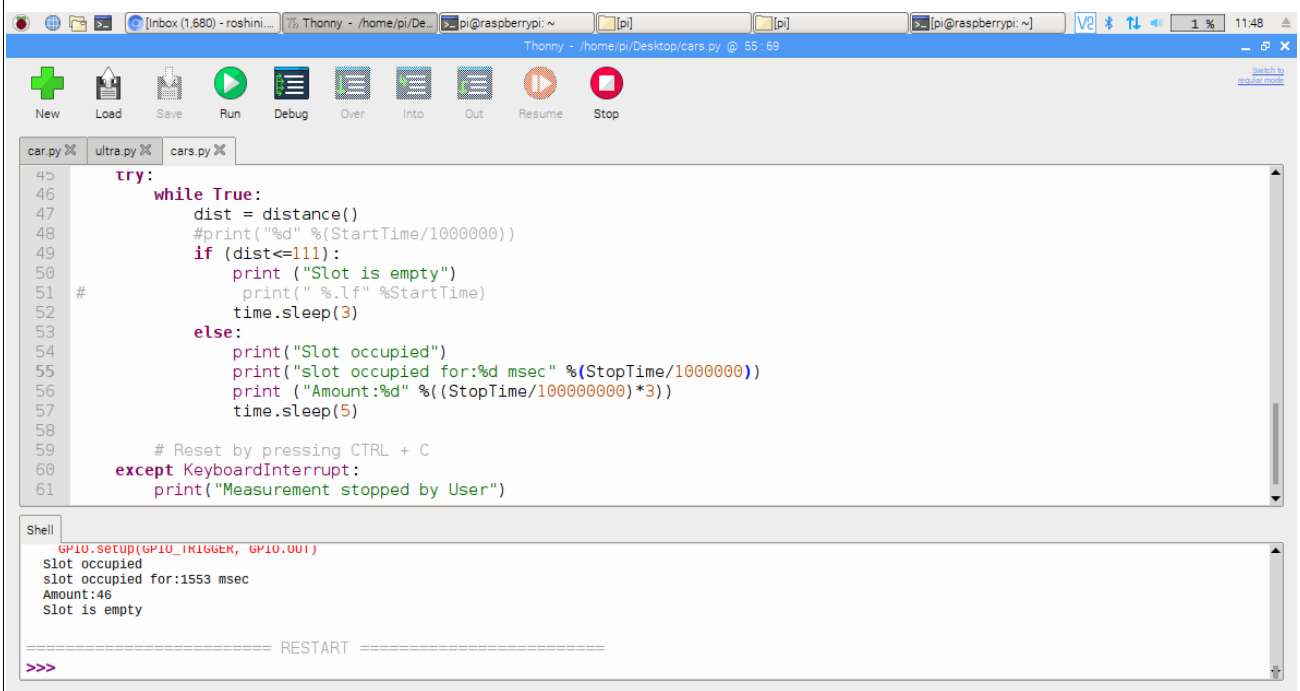
```
        while GPIO.input(GPIO_ECHO) == 1:
        StopTime = time.time()
        # time difference between start and arrival
        TimeElapsed = StopTime - StartTime
        # multiply with the sonic speed (34300 cm/s)
        # and divide by 2, because there and back
        distance = (TimeElapsed * 34300) / 2
        return distance
if __name__ == '__main__':
        try:
                while True:
                        dist = distance()
                        if (dist<=111):
                                print ("Slot is empty")
                                 time.sleep(3)
                        else:
                                print("Slot occupied")
                                print("slot occupied for:%d msec" %(StopTime/1000000))
                                print ("Amount:%d" %((StopTime/100000000)*3))
                                time.sleep(5)

        # Reset by pressing CTRL + C
        except KeyboardInterrupt:
                print("Measurement stopped by User")
                GPIO.cleanup()
```

STEPS:

1. Connect keyboard and mouse to Raspberry Pi using USB cables.
2. Connect Monitor to Raspberry Pi using HDMI cable
3. Connect Led and PIR sensor to Raspberry Pi as shown in Circuit Diagram
4. Run the Python code above.

OUTPUT:

a.) automatic billing generation



Thonny - /home/pi/Desktop/ultra.py @ 17 : 26

New Load Save Run Debug Over Into Out Resume Stop

car.py ultra.py

```python
    return distance

if __name__ == '__main__':
    try:
        while True:
            dist = distance()
            if (dist<=111):
                print ("Slot is empty")
            else:
                print("Slot occupied")
                time.sleep(1)

        # Reset by pressing CTRL + C
    except KeyboardInterrupt:
        print("Measurement stopped by User")
        GPIO.cleanup()
```

Shell

```
Slot is empty
Slot is empty
Slot is empty
Slot occupied
Slot is empty
Slot is empty
Slot is empty
Slot occupied
```