Implementation of IoT with Raspberry Pi:
Introduction to Implementation of data sharing between server & client

Networking in Python

- Networking in Python involves using modules like
  socket for low-level network communication
  socketserver, asyncio for higher-level libraries
  third-party frameworks like Twisted or Tornado for more complex networking tasks.
- Python also has libraries that provide higher-level access to specific application-level network protocols, such as FTP, HTTP, and so on.

Sockets

- Sockets are the endpoints of a bidirectional communications channel. Sockets may communicate within a process, between processes on the same machine, or between processes on different continents.
- Sockets may be implemented over a number of different channel types: Unix domain sockets, TCP, UDP, and so on.
- The socket library provides specific classes for handling the common transports as well as a generic interface for handling the rest.

Socket module

- The socket module in Python provides low-level networking interfaces that allow you to create and use network sockets.
- Sockets are the endpoints for communication between two machines over a network.

Types of Sockets

TCP Sockets (Stream Sockets):

- Used for reliable, two-way communication streams.
- Example: Client-server applications, HTTP, FTP.

UDP Sockets (Datagram Sockets):

- Used for connectionless communication.

Example: DNS, video streaming, online gaming.

Some terms and description

- **Domain:** The family of protocols the is used as a transport mechanism. These values are

constant such as AF_INET, PF_INET, PF_UNIX, PF_X25 ans so on.

- **Type:** the type of communication between two endpoints typically SOCK_STREAM for connection-oriented protocols and SOCK_DGRAM for connectionless protocols.
- **Protocol:** Typically, zero, this may be used to identify a variant of protocol within a domain and type.
- **Hostname:** the identifier of network interface.

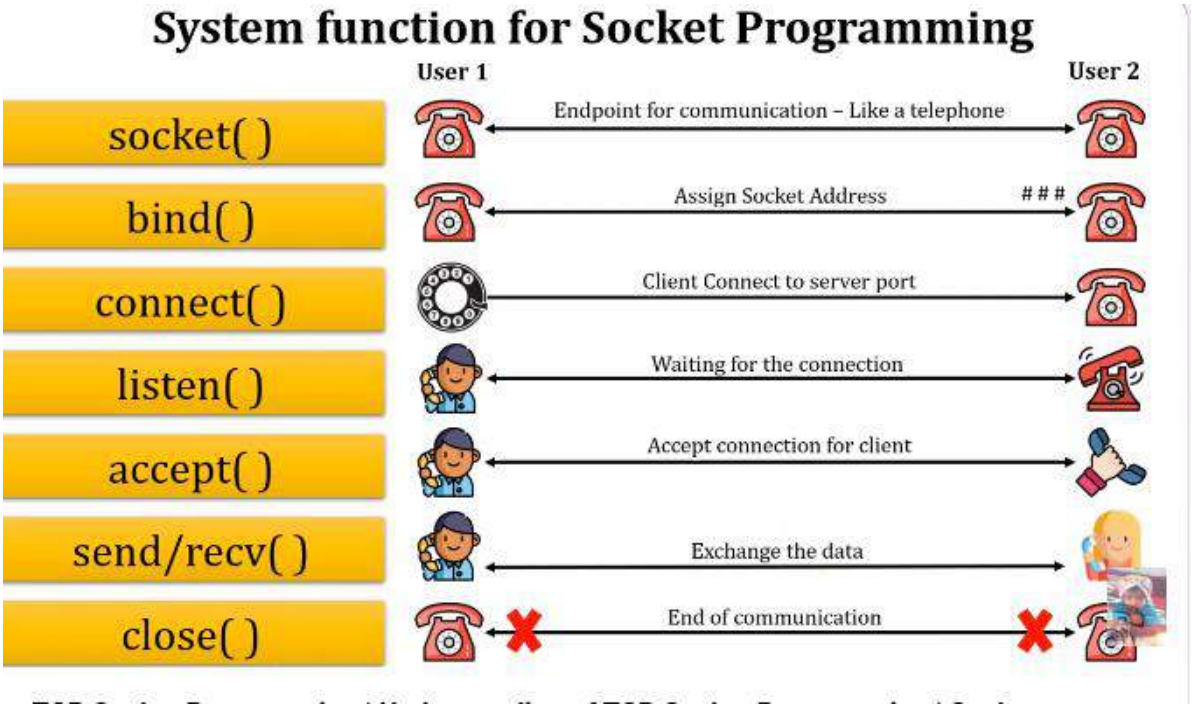
A string which can be a host name, a dotted quad address, or an IPV6 address in colon notation.
A string "<broadcast>", which specifies an INADDR_BROADCAST address.
An integer interpreted as binary address in host byte order.

- **Port:** Ports allow multiple applications or services to operate simultaneously on a single device, each using its own unique port number to distinguish one communication channel from another. A port is a virtual endpoint for communication in an operating system.

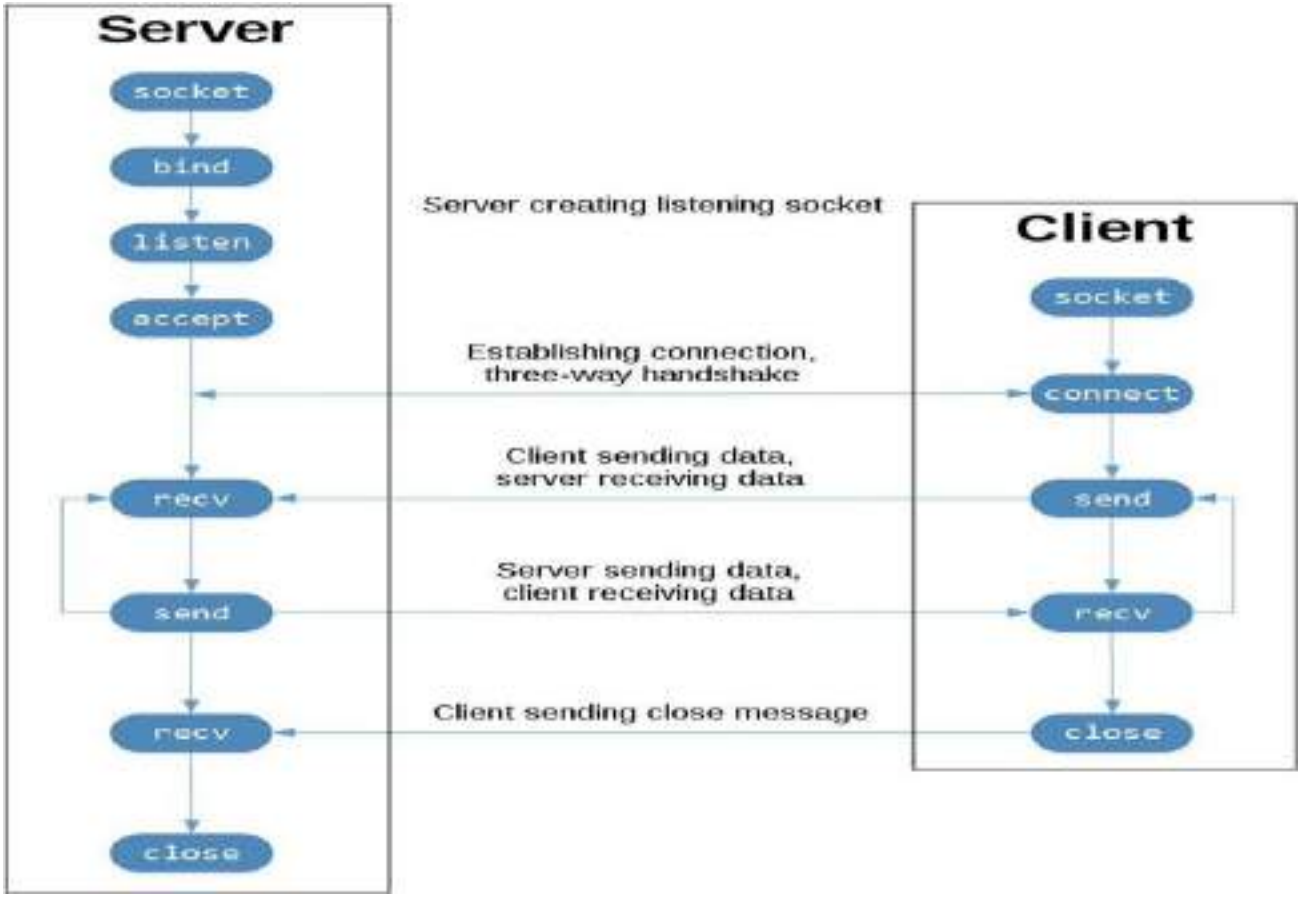
SOCKET PROGRAMMING

- **Socket programming is a way of connecting two nodes on a network to communicate with each other.**
- **The main functions in <sys/socket.h> are:**
- **socket()**
- **bind()**
- **listen()**
- **connect()**
- **accept()**
- **send()/recv()/sendto()/receivefrom()**
- **close()**

understand all the system from telephone analogue

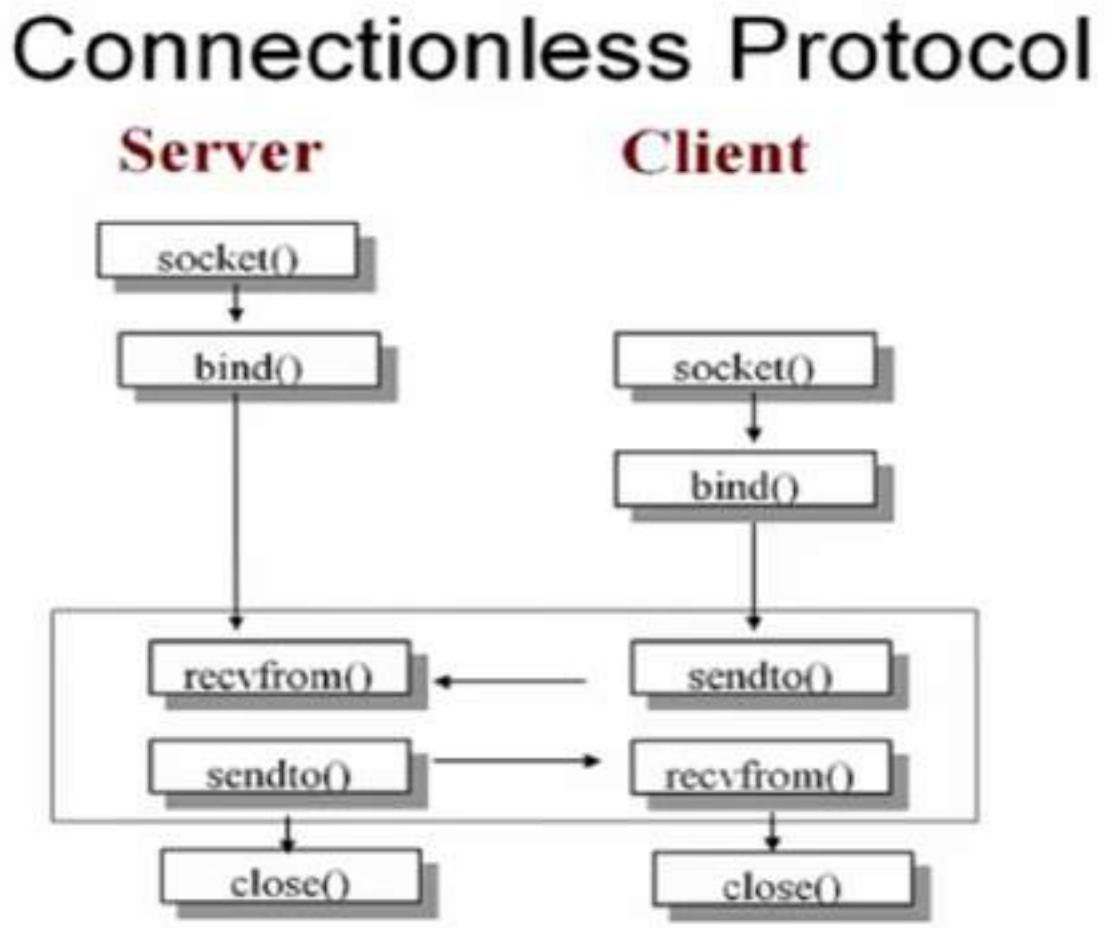


TCP SOCKET FLOW



- TCP socket programming is used in client server architecture.
- In TCP socket programming client and server programs are separate.
- First at server-side server will create socket after creating socket server will bind the services running on server machine so using the bind() function server will bind IP address and port number with specific services.
- The purpose of listen() is listen incoming connection from the client, server can handle multiple clients at a time.
- accept() is block until the connection from the client at client side list of all you should write socket() function for creating a socket then next function is connect()
- Using connect() function client can try to connect with the server port so after receiving connection request at that time accept() function is executed at server side so it means connection is established between client and server
- After connection next process is data transfer so it means we have to write read/write or send/receive function at both client and the server.
- Client send request to the server, server will take request and read that request so after executing request server will process the request and use send function or write function to the client, now client receive replay from server so in this flow write means send function and read means receive unction.
- After exchanging the data server and client use close() function to close socket at both client and server side.

UDP SOCKET FLOW



- UDP socket programming is used in client server architecture.
- In UDP socket programming client and server programs are separate.
- Socket is created at both server and client
- bind() function is to bind the IP address and port number with specific services running on server.
- receivefrom() function is blocked until the datagram received from the client, so here accept() and listen() functions are not used because UDP provide connectionless service.
- In client program first function used is socket() for creating socket at client side
- sendto() function client send a request to server first server will receive that request and process the request after processing request server will send replay to the client so client will receive through the receivefrom() function.
- When client receive all the required data from server then client use close() function after that server is in waiting state for client request but server is not get an request from client so automatically server close the socket at server side.

Socket Programming:

Creates a two-way communication between two nodes in a network
The nodes are termed as Server and Client

Creating a socket:

s = socket.socket (SocketFamily, SocketType, Protocol=0)

SocketFamily can be AF_UNIX or AF_INET
SocketType can be SOCK_STREAM or SOCK_DGRAM
Protocol is set default to 0

Lab Experiment 9:

Monitor data remotely using TCP/UDP Protocol using socket programming

Server Program

```
import socket
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
                                        #Bind the socket to the  port
server_address = ("192.168.0.2",2000)
sock.bind(server_address)
while True:
    data,address = sock.recvfrom(4096) with open("Datalog.txt","a") as f:
mess=str(data)
f.write(mess)
print(mess)
f.close()
```

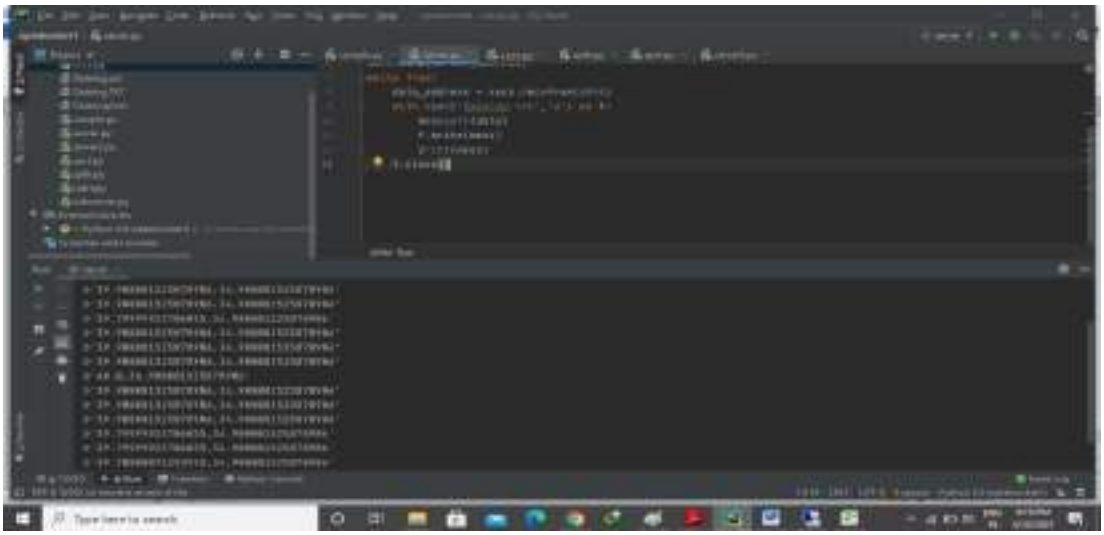
Client Program:

```
import socket import sys
import RPi.GPIO as GPIO from time
import sleep
import Adafruit_DHT

def sensordata():
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
Sensor=Adafruit_DHT.DHT22
humidity, temperature = Adafruit_DHT.read_retry(Sensor,4)
return(humidity, temperature)
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)          #create UDP socket
server_address = ("192.168.43.234",2000)
try:
while (1):
```

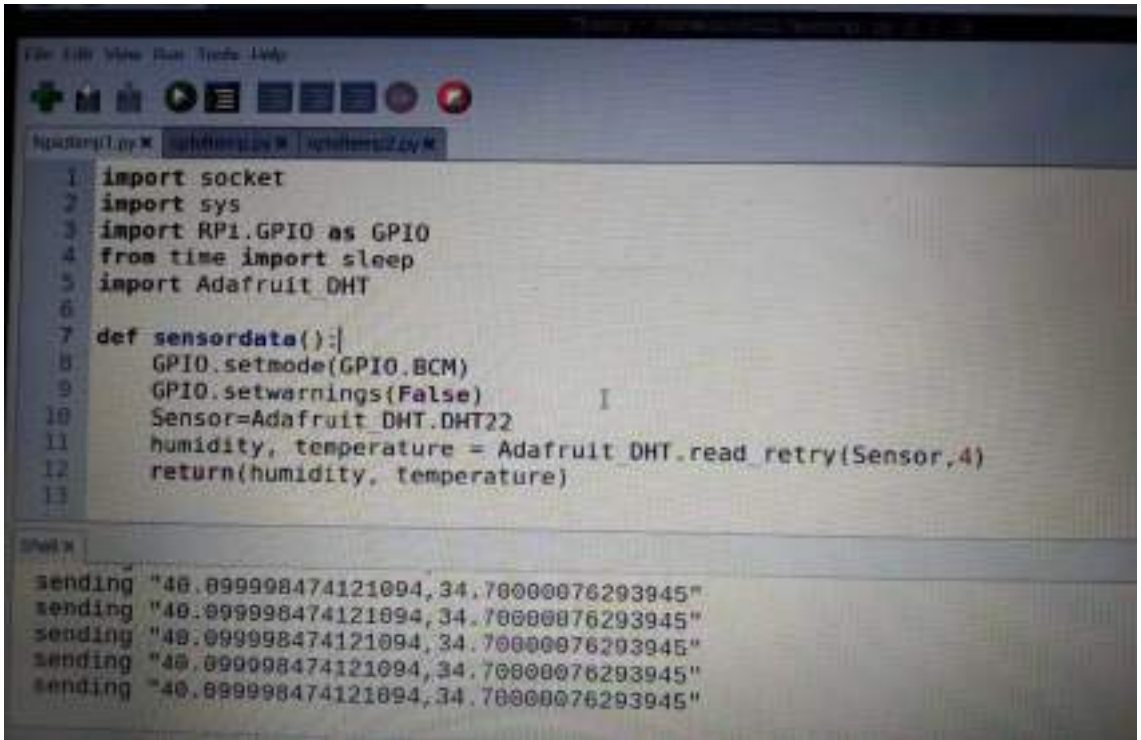
```
h,t = sensordata()
message = str(h)+','+str(t)
                                                    #Send data
print (sys.stderr, 'sending "%s"' % message)
sent = sock.sendto(message.encode(), server_address)
finally:
print (sys.stderr, 'closing socket')
sock.close()
```

OUTPUT:

SERVER Receiving Data:



CLIENT Sending Data:



Traffic Light System:

- Traffic lights are a set of three lights that signal whether one grouping of Vehicles can go or if they need to stop by combining these lights in series at an intersection a smooth flow of traffic can be ensured.
- Green signals “GO”, Yellow or Amber signals to “Slow Down and Prepared to Stop” and Red means “Stop”.
- As a vehicle approaches a traffic light at a typical intersection it has the choice of turning left, right or going straight.
- These choices have to be addressed by the signal from the traffic light.

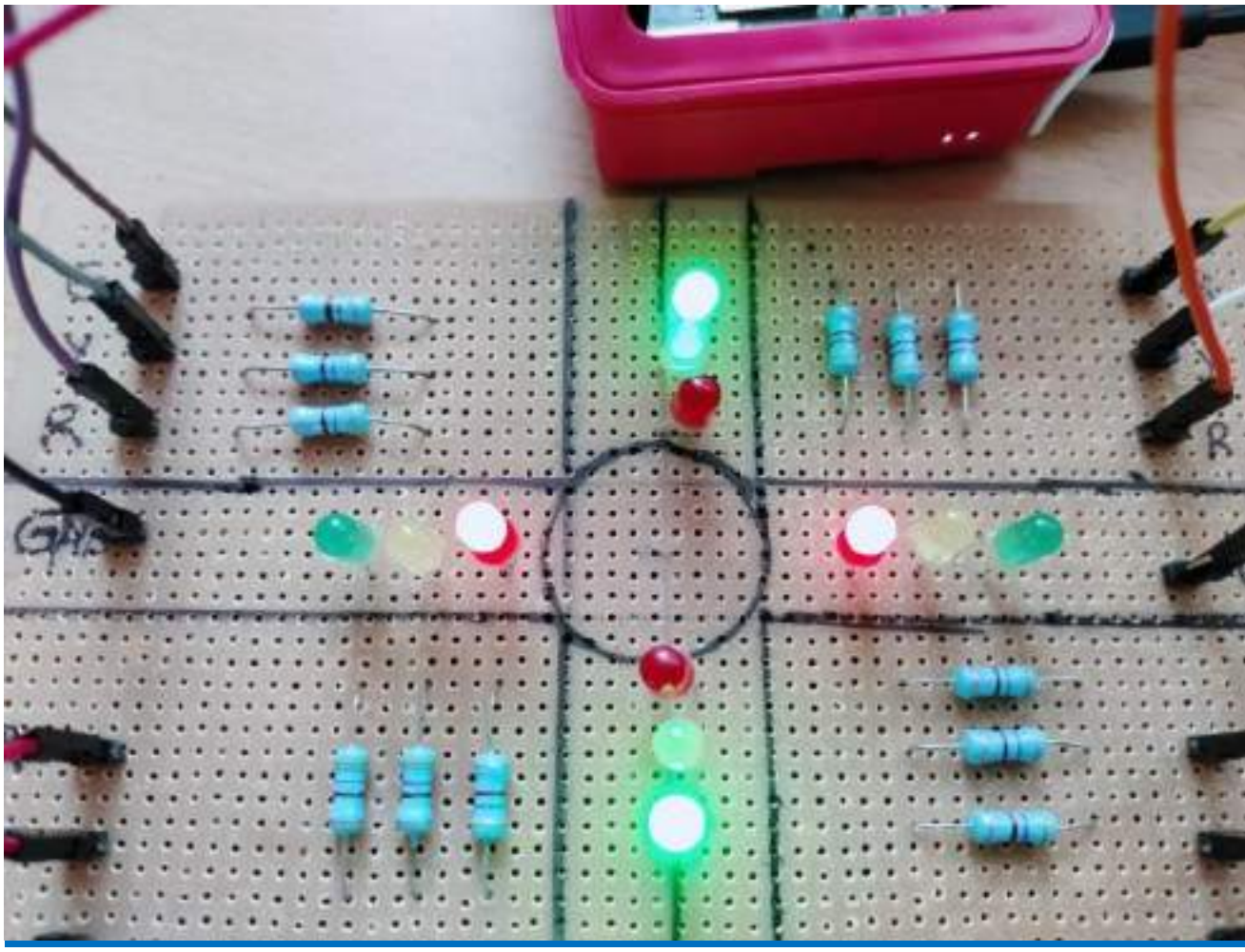
LabExperiment7: Implementation of Traffic Light system with Raspberry Pi

Program:

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
GPIO.setup(2,GPIO.OUT)
GPIO.setup(3,GPIO.OUT)
GPIO.setup(4,GPIO.OUT)

GPIO.setup(17,GPIO.OUT)
GPIO.setup(27,GPIO.OUT)
GPIO.setup(22,GPIO.OUT)

GPIO.setup(10,GPIO.OUT)
GPIO.setup(9,GPIO.OUT)
GPIO.setup(11,GPIO.OUT)

GPIO.setup(13,GPIO.OUT)
GPIO.setup(6,GPIO.OUT)
GPIO.setup(5,GPIO.OUT)

while True:
   GPIO.output(2,False)
   GPIO.output(3,False)
   GPIO.output(4, False)
   GPIO.output(17, False)
   GPIO.output(27, False)
   GPIO.output(22, False)
```

```
    GPIO.output(10, False)
    GPIO.output(9, False)
    GPIO.output(11, False)
    GPIO.output(5, False)
    GPIO.output(6, False)
    GPIO.output(13, False)
    time.sleep(2 )
    print(" Green Signal for Lane 2 & Lane 4 & Red signal for Lane 1 & Lane
3")
    GPIO.output(2, True)
    GPIO.output(10, True)
    GPIO.output(13, True)
    GPIO.output(22, True)
    time.sleep(8)
    print(" Yellow Signal for Lane 2 & Lane 4" )
    GPIO.output(2, False)
    GPIO.output(10, False)
    GPIO.output(13, False)
    GPIO.output(22, False)
    GPIO.output(27, True)
    GPIO.output(6, True)
    time.sleep(6)
    print(" Green Signal for Lane 1 & Lane 3 & Red signal for Lane 2 & Lane
4")
                GPIO.output(27,False)
   GPIO.output(6, False)
   GPIO.output(17, True)
   GPIO.output(5, True)
   GPIO.output(4, True)
   GPIO.output(11, True)
   time.sleep(8)
   print(" Yellow Signal for Lane 1 & Lane 3“)
   GPIO.output(17, False)
   GPIO.output(5, False)
   GPIO.output(4, False)
   GPIO.output(11, False)
   GPIO.output(9, True)
   GPIO.output(3, True)
 time.sleep(4)
```

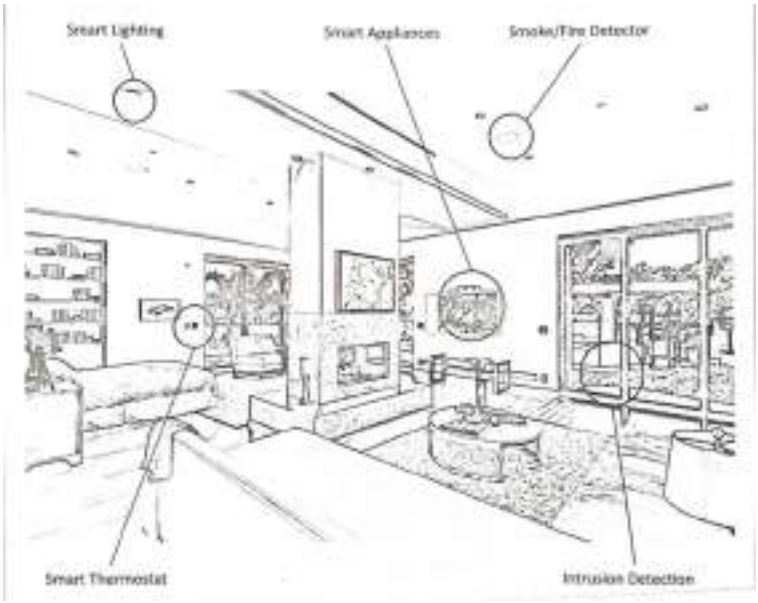
OUTPUT:



IoT applications: HOME AUTOMATION

IoT applications for smart homes:

- Smart Lighting
- Intrusion Detection
- Smoke / Gas Detectors



Smart Lighting

- Smart lighting achieves energy savings by sensing the human movements and their environments and controlling the lights accordingly.
- Key enabling technologies for smart lighting include:
  - Movement Based Light ON/OFF or Door OPEN/CLOSE
    IP-enabled lights
- Wireless-enabled and Internet connected lights can be controlled remotely from IoT applications such as a mobile or web application.

Intrusion Detection

- Home intrusion detection systems use security cameras and sensors to detect intrusions and raise alerts.
- The form of the alerts can be in form:
  - Buzzer/LED ON
  - SMS
  - Email
  - Image grab or a short video clip as an email attachment

Smoke / Gas Detectors

- Smoke detectors are installed in homes and buildings to detect smoke that is typically an early sign of fire.
- It uses optical detection, ionization or air sampling techniques to detect smoke
- The form of the alert can be in form:
- Signals that send to a fire alarm system/ Buzzer Alarm
- Gas detector can detect the presence of harmful gases such as carbon monoxide (CO2), liquid petroleum gas (LPG), etc.

PIR SENSOR for Home Automation

- A passive infrared sensor (PIR sensor) is an electronic sensor that measures infrared (IR) light radiating from objects in its field of view(range)
- PIR sensors are commonly used in security alarms and automatic lighting applications.
- PIR sensors detect general movement, but do not give information on who or what moved.
- PIN DESCRIPTION
- Pin1 corresponds to the drain terminal of the device, which connected to the positive supply 5V DC.
- Pin2 corresponds to the source terminal of the device, which connects to the ground terminal via a 100K or 47K resistor. The Pin2 is the output pin of the sensor.
- Pin3 of the sensor connected to the ground.

Program

```
import RPi.GPIO as GPIO import time
GPIO.setwarnings(False) GPIO.setmode(GPIO.BOARD)
GPIO.setup(11, GPIO.IN)      #Read ouput from PIR motion sensor
GPIO.setup(3, GPIO.OUT)      #LED output pin
While True:
        i=GPIO.input(11)
        if i == 0:           #When output from motion sensor is LOW print ("No Person
                             detected",i)
        GPIO.output(3,0) time.sleep(0.1)
        elif i == 1:         #When output from motion sensor is HIGH print ("A Person is
                             detected",i)
        GPIO.output(3,1)     #Turn ON LED
        time.sleep(0.1)
```

OUTPUT

IoT applications in healthcare:

IoT applications in smart health & lifestyle:

1. Health & Fitness Monitoring
2. Wearable Electronics

Health & Fitness Monitoring

- Wearable IoT devices allow to continuous monitoring of physiological parameters such as blood pressure, heart rate, body temperature, etc than can help in continuous health and fitness monitoring.
- It can analyze the collected health-care data to determine any health conditions.
- The wearable devices may can be in various form such as:
    - Belts
    - Wrist-bands

Wearable Electronics

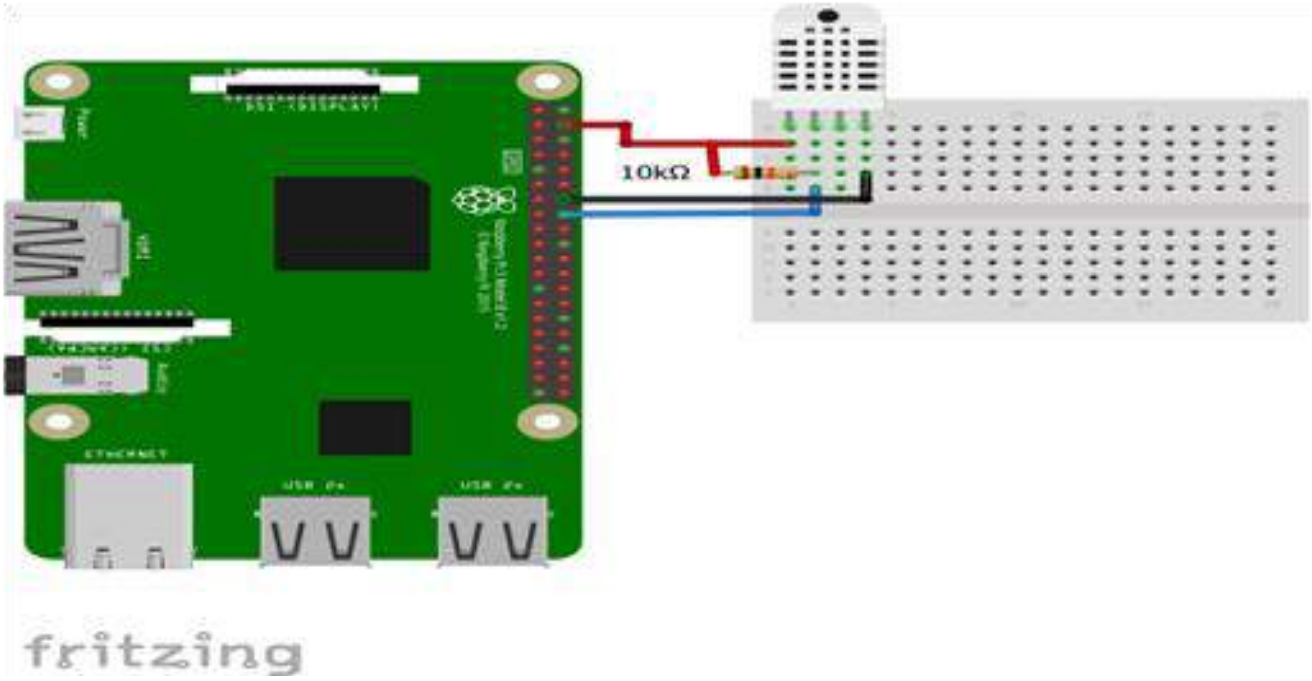
- Wearable electronics such as wearable gadgets (smart watch, smart glasses, wristbands, etc) provide various functions and features to assist us in our daily activities and making us lead healthy lifestyles.
- Using the smart watch, the users can search the internet, play audio/video files, make calls, play games, etc.
- Smart glasses allow users to tae photos and record videos, get map directions, check flight status or search internet using voice commands
- Smart shoes can monitor the walking or running speeds and jumps with the help of embedded sensors and be paired with smart-phone to visualize the data.
- Smart wristbands can tract the daily exercise and calories burnt.

IoT Implementation in Healthcare

ThingSpeak:

- ThingSpeak is an open-source Internet of Things application and API to store and retrieve data from things using the HTTP and MQTT protocol over the Internet or via a Local Area Network.
- ThingSpeak is an IoT analytics platform service that allows you to aggregate, visualize and analyze live data streams in the cloud. You can send data to Thing Speak from your devices, create instant visualization of live data and send alerts.

Connections



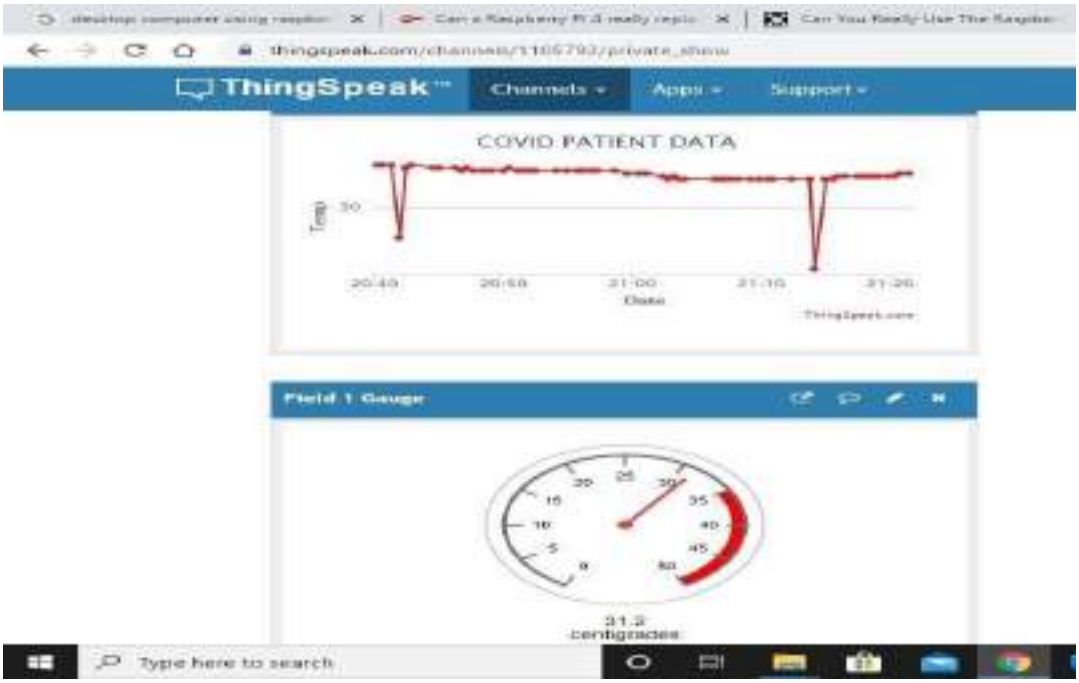
Program:

```
import sys
import urllib.request
from time import sleep import Adafruit_DHTas dht
                #EnterYour APIkey here myAPI='T736KR658LSN9USN'
                 #URL where we will send the data, don't change it
baseURL='https://api.thingspeak.com/update?api_key=%s'%myAPI

defDHT22_data():
                    # Reading from DHT22 and storing the temperature
                      and humidity
humi,temp= dht.read_retry(dht.DHT22,4)
return humi,temp
```

```
whileTrue: try:
humi, temp = DHT22_data()
#IfReading is valid
if isinstance(humi,float)and isinstance(temp,float): # Formatting to two decimal places
humi= '%.2f' % humi temp='%.2f' % temp    print('temperature:',temp)
                                        #Sending the data to thingspeak
 conn = urllib.request.urlopen(baseURL + '&field1=%s&field2=%s' % (temp, humi))
 res=conn.read()
print(res)
                                # Closing the connection
conn.close()

else:
print('Error')
                    #DHT22 requires 2 seconds to give a reading,so
                      make sure to add delay of above 2 seconds.
 sleep(20)
 except:break
```

Output:

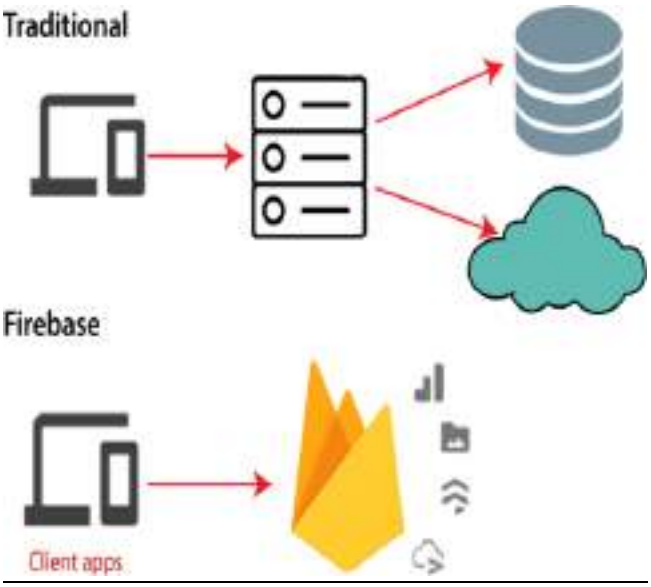
Temperature Monitoring using IOT Dashboard



Firebase Introduction and Account creation

Firebase:

- In the era of rapid prototyping, we can get bright ideas, but sometimes they are not applicable if they take too much work. Often, the back-end is the limiting factor - many considerations never apply to server-side coding due to lack of knowledge or time.
- Firebase is a Backend-as-a-Service(BaaS) which started as a YC11 startup. It grew up into a next-generation app-development platform on Google Cloud Platform. Firebase (a NoSQLjSON database) is a real-time database that allows storing a list of objects in the form of a tree. We can synchronize data between different devices.
- Google Firebase is Google-backed application development software which allows developers to develop Android, IOS, and Web apps. For reporting and fixing app crashes, tracking analytics, creating marketing and product experiments, firebase provides several tools



- Firebase has three main services, i.e., a real-time database, user authentication, and hosting. We can use these services with the help of the Firebase iOS SDK to create apps without writing any server code.

Why use Firebase:

- Firebase manages real-time data in the database. So, it easily and quickly exchanges the data to and from the database. Hence, for developing mobile apps such as live streaming, chat messaging, etc., we can use Firebase.
- Firebase allows syncing real-time data across all devices - iOS, Android, and Web –without refreshing the screen.
- Firebase provides integration to Google Advertising, AdMob, Data Studio, Big Query Double Click, Play Store, and Slack to develop our apps with efficient and accurate management and maintenance.

- Everything from databases, analytics to crash reports are included in Firebase. So, the app development team can stay focused on improving the user experience.

Pros:

- Firebase is a real-time database.
- It has massive storage size potential.
- Firebase is serverless.
- It is highly secure.
- It is the most advanced hosted BaaS solution.
- It has minimal setup.

- It provides three-way data binding via angular fire.
- It provides simple serialization of app state.
- We can easily access data, files, auth and more.
- There is no server infrastructure required to power apps with data.
- It has JSON storage, which means no barrier between data and objects.

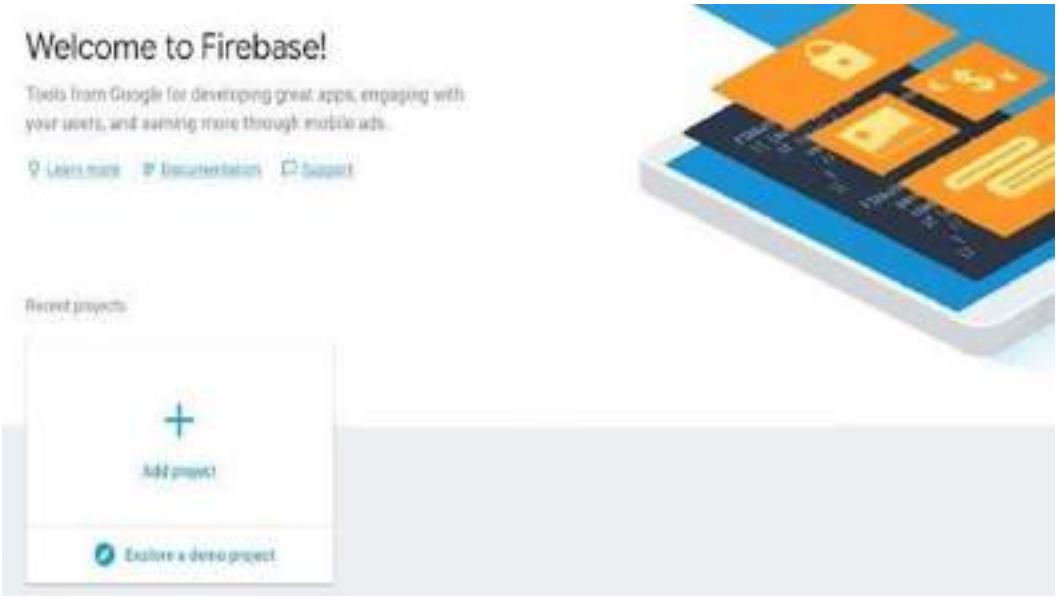
Cons:

- Firebase is not widely used, or battle-tested for enterprises.
- It has very limited querying and indexing.
- It provides no aggregation.
- It has no map-reduce functionality.
- It cannot query or list users or stored files.

Procedure to Create Firebase Account:

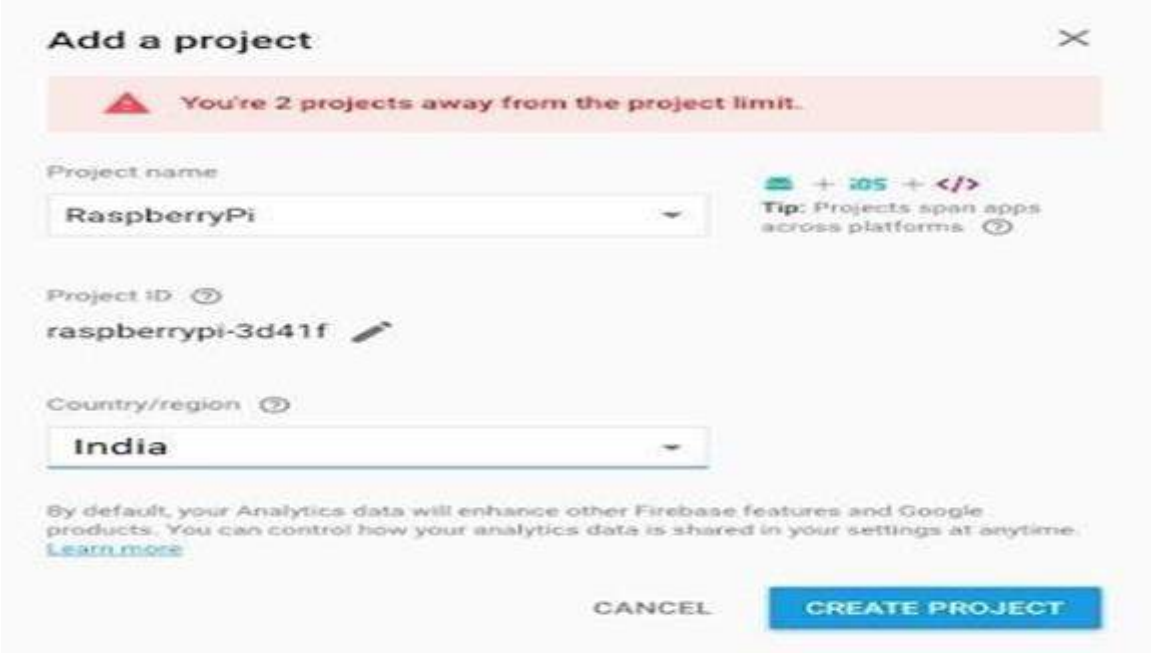
Step1:

- Login to your Firebase account (*https://firebase.google.com/*), then go to the Firebase console application.



Step2:

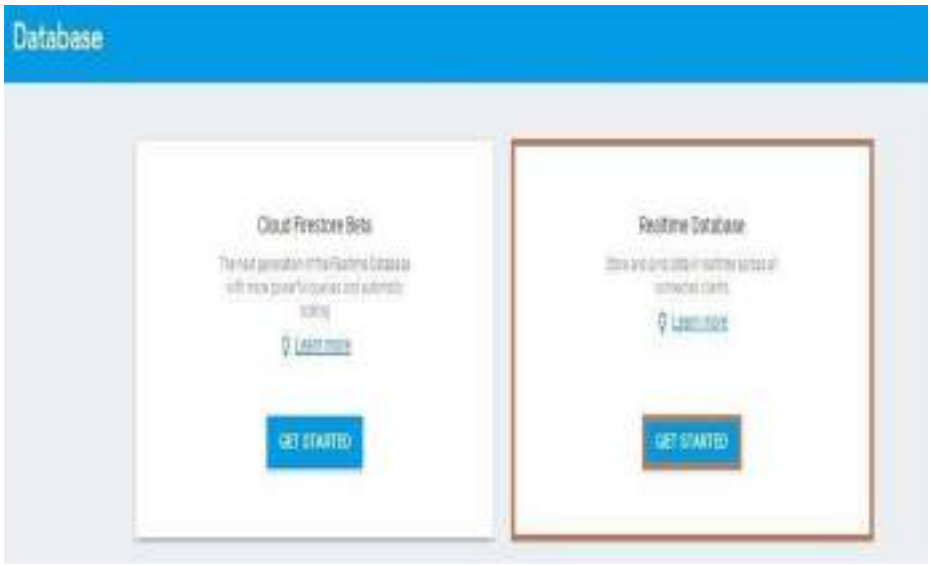
- Create a new project and fill in the necessary details.



Step 3

Next, goto select Develop and then click on Database in your project.

After that, the Database appears and select the "Real-time Database Get Started" button



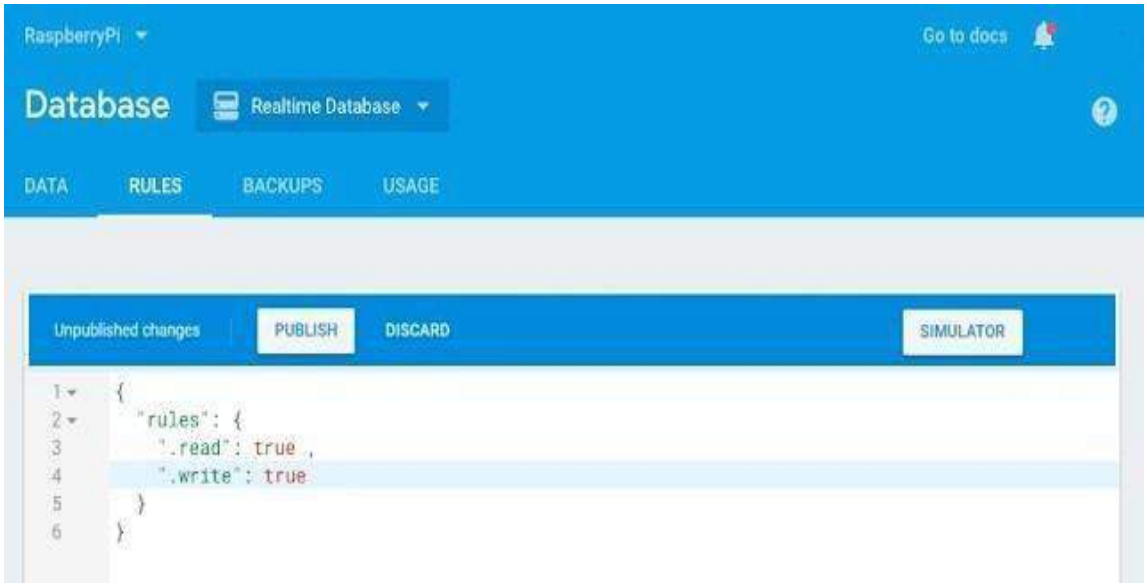
Step 4:

Next, gotorules>>changerules. Just replace that with the following code.

Default Rules

```
// require
{

    ".read":"auth!=null"
    ".write":"auth!=null"
  }
}
```

Change to these rules



This is for the first test. Anyone can read and write your database without authentication.

```
//Not require
{

    ".read":true,
    ".write":true
  }
}
```

- Apply the above code to Firebase rule, then click the publish button to deploy the rules. After the rules are published anyone can read and write to your data.
- Goto database>>real-timedatabase>> data

Step 5:

Open the python IDE, File >> New. Save the name as dht-firebase.py.py, then follow the given code.

Experiment-9: Creating a firebase cloud account and connect to cloud
services to monitor device data.

Program:

```
import RPi.GPIO as GPIO
GPIO.setwarnings(False)

GPIO.setmode(GPIO.BOARD)
GPIO.setup(10,GPIO.IN,pull_up_down=GPIO.PUD_DOWN) from datetime
                    # to monitor the state of GPIO pin 10
                   and print the current time whenever its state changes.
import datetime
from picamera import PiCamera from time  # to capture images periodically with timestamps
import sleep


import os
import pyrebase
firebase Config={
'apiKey' : "AIzaSyABtUYpOCaczxhfx1MNEtPX70ohAKDk1EY",
'authDomain ': "rpi- testing-e3379.firebaseapp.com",
'databaseURL' : "https://rpi-testing-e3379-default-rtdb.firebaseio.com",
'projectId' : "rpi- testing-e3379",
storageBucket' : "rpi-testing-e3379.appspot.com",
'messagingSenderId' : "943281949971",
 'appId' : "1:943281949971:web:2b765d4d08eba7c5dad ee6",
'measurementId' : "G-TMFQNXZCHE"
}

# initialize the firebebase application and firebase storage
firebase = pyrebase.initialize_app(firebaseConfig)
storage= firebase.storage()

#db is to send the sensor data of firebase.
db=firebase.database()     #To constantly update the sensor data in the cloud storage
 db.child("<NAME>").set(<sensor data>)
#To push the data to the storage name= "google.jpeg"
db.child("<NAME>").push(<sensor data>)

 #camera.capture(name)
 #print(name+"saved")
 # initialize the firebebase application and firebase storage
 firebase = pyrebase.initialize_app(firebaseConfig)
```

```
storage= firebase.storage()

#db is to send the sensor data of firebase.
db=firebase.database()     #To constantly update the sensor data in the cloud storage
db.child("<NAME>").set(<sensor data>)
#To push the data to the storage name= "google.jpeg"
db.child("<NAME>").push(<sensor data>)

 #camera.capture(name)
 #print(name+"saved")
```

In this script:

- GPIO is set up for pin 10 with a pull-down resistor.
- Firebase is initialized with your configuration.
- The PiCamera is initialized.
- The get_sensor_data function is a placeholder for fetching actual sensor data.
- The capture_and_upload_image function captures an image, uploads it to Firebase Storage, and removes the local file.
- The send_sensor_data_to_firebase function sends sensor data to Firebase Realtime Database.
- The script runs in an infinite loop, capturing and uploading an image and sending sensor data every 10 seconds, until interrupted by pressing CTRL+C.

Make sure your Raspberry Pi has the necessary libraries installed (pyrebase, picamera, RPi.GPIO, etc.) and that it is connected to the internet for Firebase operations.

IoT Applications: Smart Deceives
IOT GARBAGE MONITORING USING RASPBERRY PI

Smart Appliances/Smart Devices

- Smart appliances make the management easier and provide status information of appliances to the users locally or remotely.
- E.g: Smart Dustbin, Remotely controlled Home Appliances (AC,Bulbs,Geyser)
- smart washer/dryer that can be controlled remotely and notify when the washing/drying cycle is complete.
- Open Remote is an open-source automation platform for smart home and building that can control various appliances using mobile and web applications.
- It comprises of three components:
    - a Controller manages scheduling and runtime integration between devices.
    - a Designer allows to create both configuration for the controller and user interface designs.

Smart Parking

- Finding the parking space in the crowded city can be time consuming and frustrating
- Smart parking makes the search for parking space easier and convenient for driver.
- It can detect the number of empty parking slots and send the information over the Internet to the smart parking applications which can be accessed by the drivers using their smartphones, tablets, and in car navigation systems.
- Sensors are used for each parking slot to detect whether the slot is empty or not, and this information is aggregated by local controller and then sent over the Internet to database.

Smart Lighting for Roads

- It can help in saving energy
- Smart lighting for roads allows lighting to be dynamically controlled and also adaptive to ambient conditions.
- Smart light connected to the Internet can be controlled remotely to configure lighting schedules and lighting intensity.
- Custom lighting configurations can be set for different situations such as a foggy day, a festival, etc.

IOT GARBAGE MONITORING USING RASPBERRY PI

- The IOT garbage monitoring using the Raspberry Pi project consists of a power supply circuit, ultrasonic level sensors, a Raspberry Pi digital controller, an LCD display, and a remote server.
- The power supply is responsible for supplying a stable 5V DC voltage to all the DC-powered devices in the electronic system.
- It consists of a full-wave diode bridge rectifier, a voltage regulator IC and passive electronic components.
- The power supply converts high-level AC voltage to low-level DC voltage.

Operation

- The IOT garbage monitoring using Raspberry Pi project uses two ultrasonic sensors to monitor the garbage

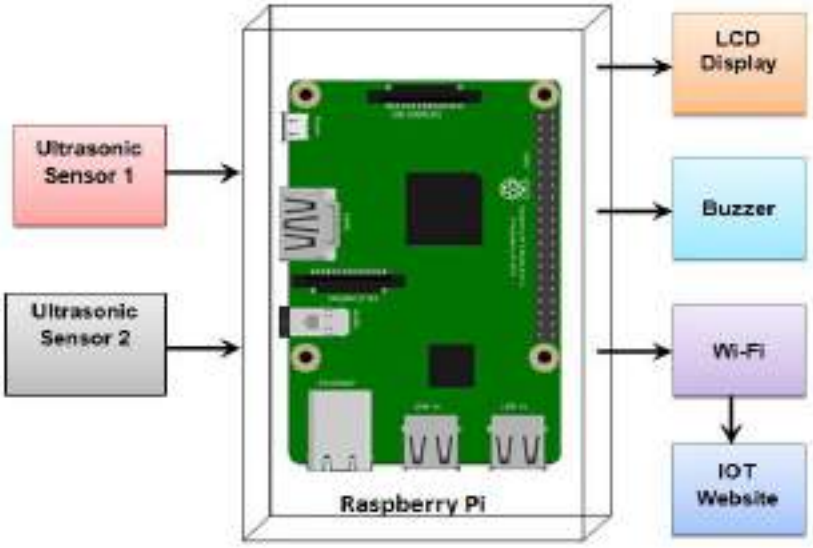
level of two bins.

- The ultrasonic sensors consist of a transmitter and a receiver. They work on the principle of Doppler's effect. The sensor generates an ultrasonic wave that reflects back after colliding with an obstacle or an object.
- The time-lapse between the transmission and reception of the ultrasonic wave is measured by the ultrasonic sensor and the distance calculation is made based on this time-lapse value.
- The distance measured by the sensor in our case is the distance between the sensor and the surface level of the garbage.
- The garbage fill level of the bins as measured by the ultrasonic sensors is continuously fed to the Raspberry Pi single-board computer.
- The Raspberry Pi controller processes this data and transmits it to the output devices as well as the remote servers.

Data sent over the cloud using IOT:

- The real-time garbage fill level of each bin is displayed on an LCD display. In addition to that, the data is also transmitted to a remote server by means of IOT protocols.
- An IOT platform like ThingSpeak is used to receive, store, and visualize this data.
- A graphical user interface is built using the ThingSpeak IOT platform, which visualizes the sensor data with respect to time, in the form of a graph.

Block Diagram

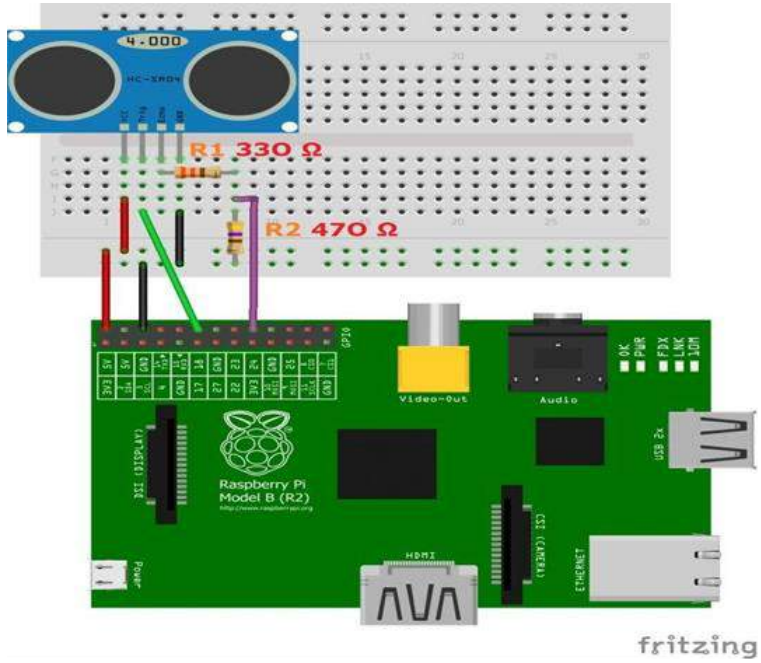


Connections

There are four pins on the ultrasound module that are connected to the Raspberry:

- VCC to Pin 2(VCC)
- GND to Pin 6(GND)

- TRIG to Pin12(GPIO18)
- Connect the 330Ω resistor to ECHO. On it send you connect it to Pin18 (GPIO24) and through a 470Ω resistor you connect it also to Pin6(GND).



Program:

```
import RPi.GPIO as GPIO import time
                                                    #GPIOMode(BOARD/ BCM)
GPIO.setmode(GPIO.BCM)                              #set GPIO Pins
GPIO_TRIGGER=18
GPIO_ECHO=24
                                                    #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 to HIGH
GPIO.output(GPIO_TRIGGER,True)
                                                                # set Trigger after 0.01ms to LOW
 time.sleep(0.00001)
GPIO.output(GPIO_TRIGGER,False)
 StartTime = time.time()
 StopTime=time.time()
                                                            # 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 sonicspeed(34300cm/s)
                                        #and divide by2, because there and back
distance = (TimeElapsed * 34300) / 2

return distance

if name== 'main':
try:
 while True:
        dist=distance()
        print ("Measured Distance = %.1f cm" % dist) time.sleep(1)
# Reset by pressing CTRL + C
        except: Keyboard Interrupt:
        print("Measurement stopped by User") GPIO.cleanup()
```

OUTPUT: