Gas Monitoring Dashboard

Tools & Components:

- Raspberry Pi
- Ultrasonic sensor
- Jumper Wires

Procedure:

1. Connect the laptop to Raspberry Pi with the LAN cable. Use Putty for SSH connection. Type “raspberrypi.local”. Once connection is established, enter the credentials. Use the command “sudo raspi -config” to do the following changes:

   -Update

   -System Options-Boot/Auto Login-Desktop (enable) -Interface-VNC (enable).
2. Then run the command “ifconfig” to get the IP address. Open VNC viewer and paste the IP address to run Raspberry Pi.
3. Open a new file and name the file with the extension .py. Write a python code for Ultrasonic sensor and save it.
4. Open Terminal in the VNC and write the following commands in it.
   →python3 -m venv path/to/venv

   →path/to/venv/bin/pip install RPi.GPIO

   →path/to/venv/bin/pip install paho-mqtt

   →path/to/venv/bin/python Exp8.py
5. Run the code.
6. Open Node-red and insert mqtt in, function node, Gauge node, Chart (line & Bar) node and connect them accordingly.
7. In MQTT node, change Server name and topic according to the python code.
8. In Function Node, write the json code.

9. In Gauge Node, edit group and Range accordingly.
10. In Chart Node, edit size and type (both line and bar) should be added.
11. Deploy the model and go to dashboard for the result.

PIN CONFIGURATION:

RASPBERRY PI	DHT 11
5V	VCC
GPIO 4	TRIG
GPIO 5	ECHO
GND	GND

Raspberry Pi Program:

```
import RPi.GPIO as GPIO
import time
import paho.mqtt.client as mqtt

# Set GPIO pin numbers
trig_pin = 14
echo_pin = 18

# MQTT Broker connection parameters
mqtt_broker = "broker.hivemq.com"
mqtt_port = 1883
mqtt_topic_distance = "sensor/ultrasonic/distance"

# Create MQTT client instance
client = mqtt.Client()

# Connect to MQTT broker
client.connect(mqtt_broker, mqtt_port)

def measure_distance():
```

```
    # Set up GPIO
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(trig_pin, GPIO.OUT)
    GPIO.setup(echo_pin, GPIO.IN)

    # Ensure the trigger pin is low initially
    GPIO.output(trig_pin, False)
    time.sleep(0.1)

    # Trigger ultrasonic sensor
    GPIO.output(trig_pin, True)
    time.sleep(0.00001)
    GPIO.output(trig_pin, False)

    # Measure time for echo
    while GPIO.input(echo_pin) == 0:
        pulse_start = time.time()

    while GPIO.input(echo_pin) == 1:
        pulse_end = time.time()

    # Calculate distance
    pulse_duration = pulse_end - pulse_start
    speed_of_sound = 34300  # Speed of sound in cm/s
    distance = (pulse_duration * speed_of_sound) / 2

    # Cleanup GPIO
    GPIO.cleanup()

    return distance

try:
    while True:
        # Measure distance
        distance = measure_distance()

        # Publish distance data to MQTT topic
        distance_payload = f"{distance:.2f}"
        client.publish(mqtt_topic_distance, distance_payload)
        print("Published distance to MQTT:", distance_payload)
```

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

except KeyboardInterrupt:
    print("\nProgram stopped by the user")
    client.disconnect()
```

Json Code:

```
 // Assuming the incoming message contains a payload like "distance"
var payload = msg.payload;

// Extract distance
var distance = parseFloat(payload); // Assuming distance is the first value

// Set the extracted distance value into the message payload
msg.payload = distance;

return msg;
```

Output

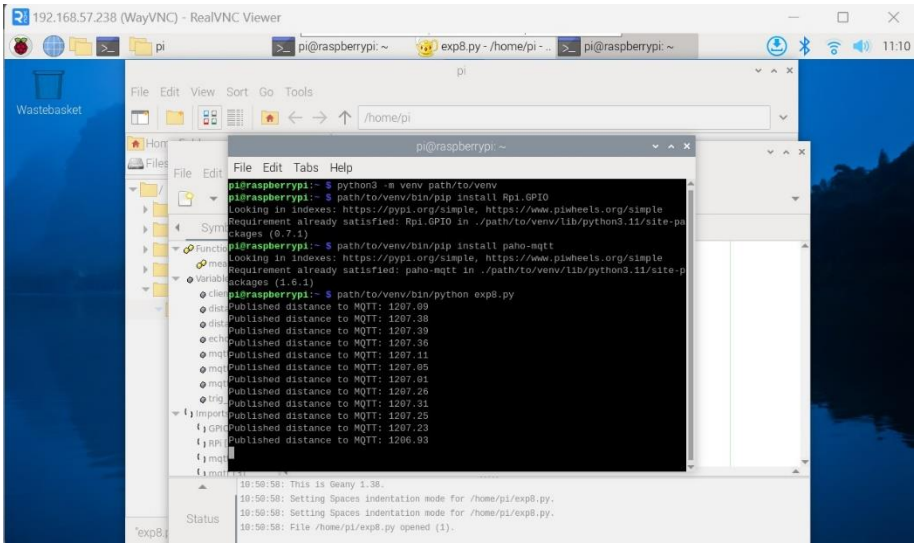

Fig. VNC Viewer

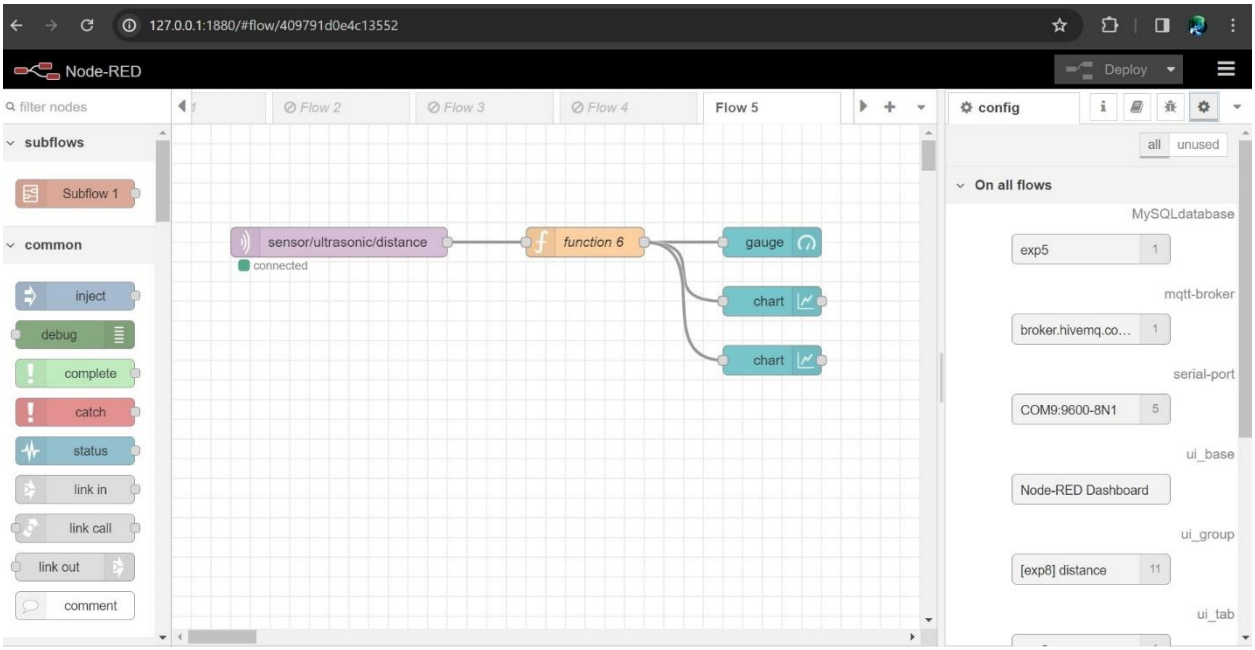

Fig. Node-Red Flow diagram

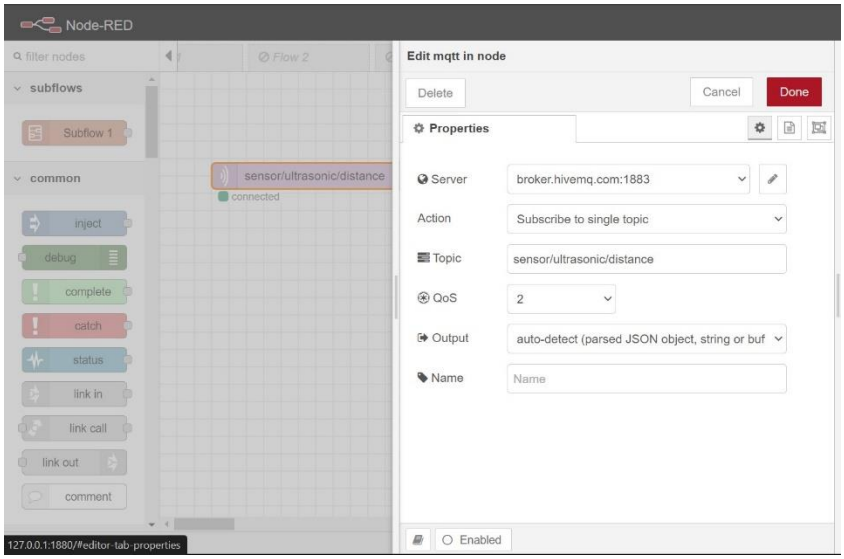

Fig. MQTT IN

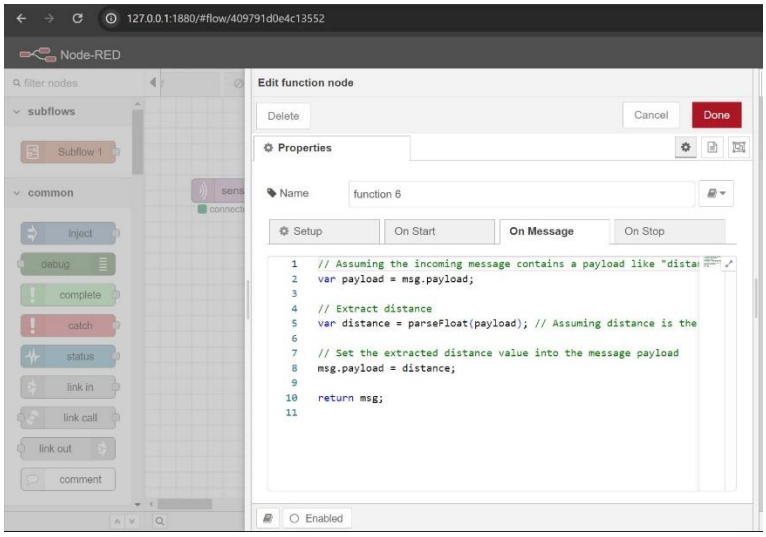

Fig. Function Node

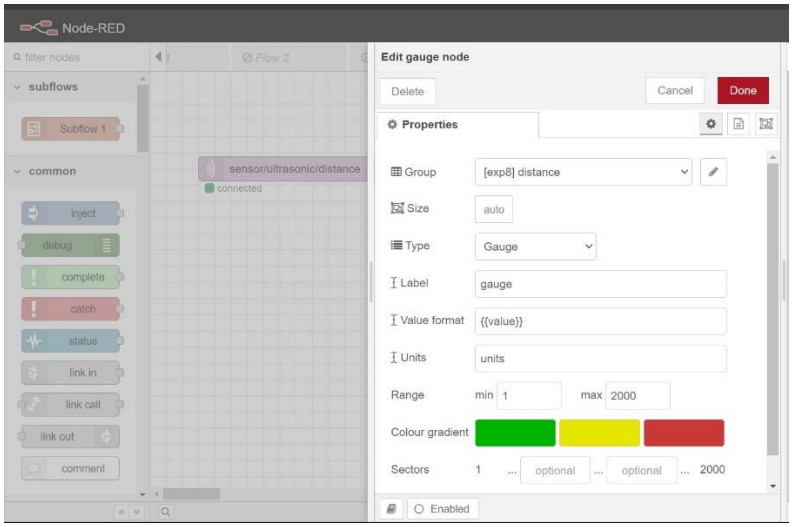

Fig. Gauge Node

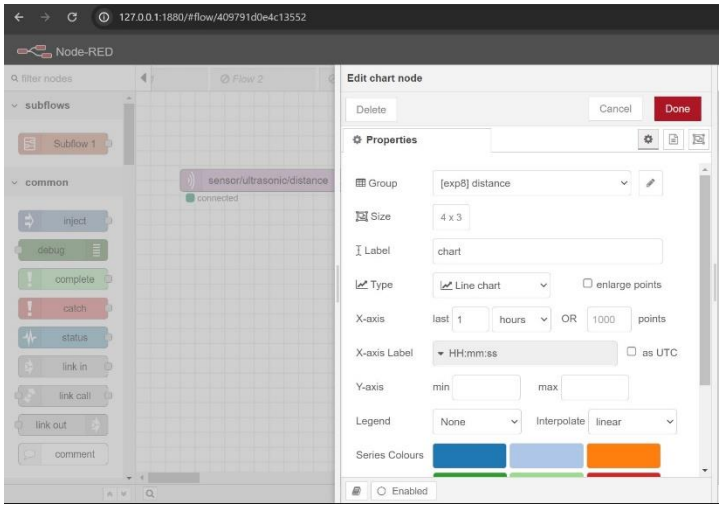

Fig. Line Chart Node

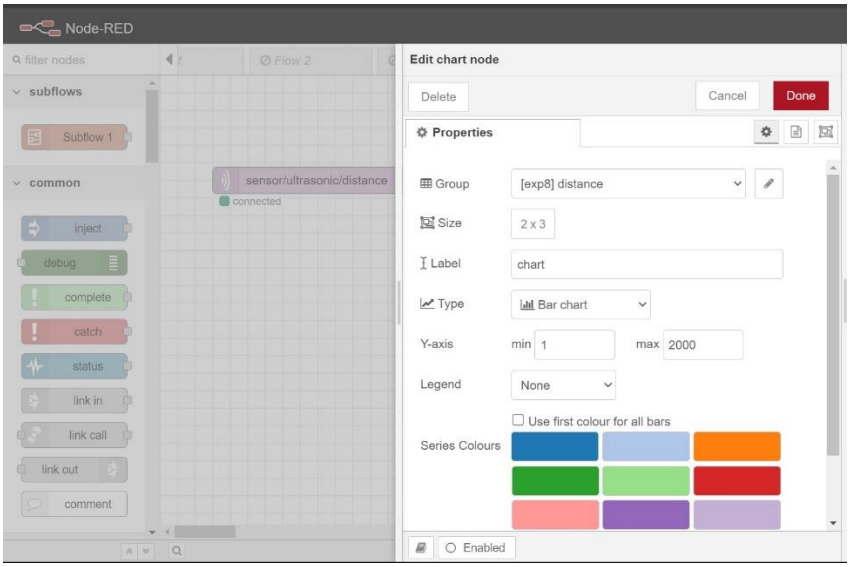

Fig. Bar Chart Node

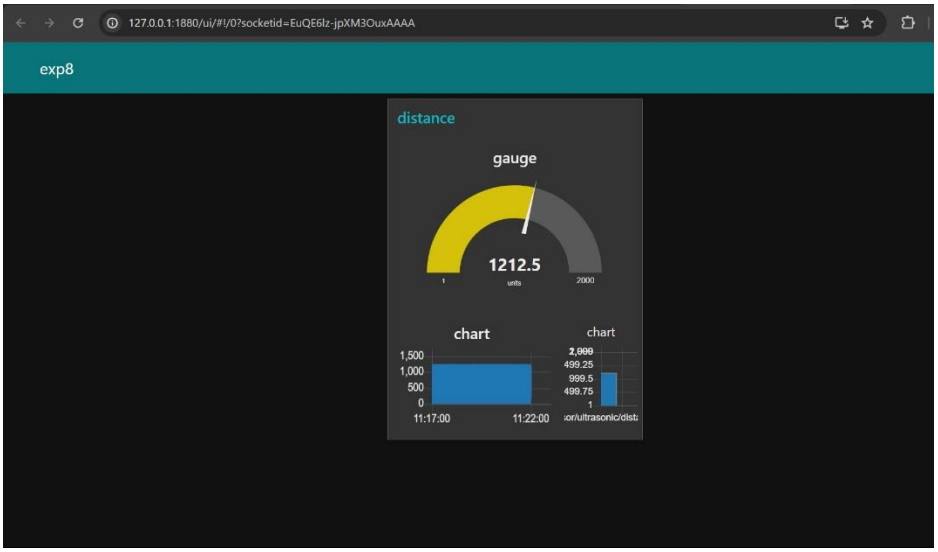

Fig. Node RED Dashboard

Result: Node Red was set up and MQTT protocol was used to turn on a digital output successfully.