|  |
| --- |
| Difference Between NAAC & NBA Accreditation - Haq Se EngineerPREC LONIJai Shriram Engineering College (@JSREC09) / Twitter**JAI SHRIRAM ENGINEERING COLLEGE**  **TIRUPPUR – 638 660**  Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai  Recognized by UGC & Accredited by NAACandNBA (CSE and ECE) |

**DEPARTMENT OF**

**ELECTRONICS AND COMMUNICATION ENGINEERING**

**IBM - Naan Mudhalvan**

**Internet of Things**

**Group 3**

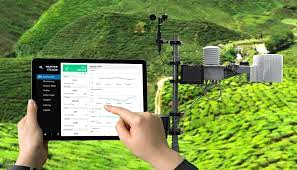
**Phase 3 - Project Submission**

**NAME : RAGUL.S.M**

**NM ID : au711221106026**

**YEAR : III**

**Environmental Sensors**

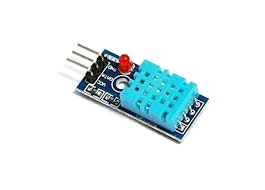


Environmental sensors measure the environmental conditions of the data center such as the temperature and humidity. Real-time data from environmental sensors is collected, monitored, and reported on by Data Center Infrastructure Management (DCIM) software to help data center managers see trends, get alerts, save energy, and increase uptime.

The types of environmental sensors, and the benefits of environmental monitoring with DCIM software, are:

**Temperature sensors:**By monitoring the temperature in your environment, you can identify the formation of hot spots that can damage equipment and cause downtime or know if you are overcooling the data center and overspending on cooling. Some make the mistake of only monitoring the temperature at the room level, while the American Society of Heating, Refrigerating, and Air-Conditioning Engineers (ASHRAE) recommends placing no fewer than six temperature sensors in each rack, mounted at the top, middle, and bottom at both the front and back of the racks. The recommended temperature range at the intake is between 18°-27°C / 65°-80°F and the exhaust temperature should be no less than 20°C / 35°F compared to the intake.

**Humidity sensors:**When the environment is too dry, static electricity can build up. Too humid, and equipment can begin to corrode. Since extreme humidity levels either way can damage equipment and cause downtime and humidity can change depending on outdoor conditions, you must monitor for humidity and maintain a stable environment. ASHRAE recommends that the relative humidity in the data center should be between 40% and 60%. Since humidity does not vary as quickly as temperature, fewer sensors are needed. Typically, one humidity sensor is deployed for every five racks and placed in the front of a rack, but more can be used for extra coverage.



**Airflow sensors:**Proper airflow in a data center will help avoid hot spots and maintain a stable ambient temperature, but cabling and other obstructions can build up over time and disrupt airflow. Monitor airflow to ensure cold air is efficiently cooling the environment or to understand if you need to make adjustments. The recommended airflow sensor placement is one sensor for each cold air supply and one for each hot air return.

**Pressure sensors:**For hot aisle/cold aisle deployments using variable speed fans, the difference in airflow between the aisles can potentially cause certain partitions like plastic curtains to be draw into an aisle creating an air leak that can lead to inefficient cooling and hot spots. Similarly, a chimney above a high-heat, high air pressure rack may leak hot air into a neighboring chimney above a low-heat, low air pressure rack. Differential air pressure sensors are typically placed at the top and bottom of racks, between aisles, between underfloor perforated titles, and in vents and air plenums.



**Vibration sensors:**Vibrations in the data center can potentially damage disk drives over time and cause downtime. When vibration sensor data is charted over time in DCIM software with a parts management feature, makes and models of disk drives that have a higher probability of failure can easily be identified in every piece of equipment across an organization’s entire global infrastructure.

**Water leak detection sensors:** Early detection of water in the data center can give you enough time to prevent a potential disaster. Consider deploying water sensors to avoid downtime caused by undetected air conditioning leakage, condensation, burst pipes, or local plumbing failures.

**Contact closure sensors:** Keep your data center secure with contact closures to know if your cabinet doors are open or closed. Often, contact closures are connected to third-party sensors, such as a smoke detection sensor or a webcam that takes a picture when a cabinet door is opened.



Environmental sensors can be deployed as plug-and-play devices that connect to certain models of intelligent rack PDUs, rack controllers, inline meters, branch circuit monitors, and gateway devices. From there, the sensor data can be collected, analyzed, and reported on by your DCIM software or other analytics application. Sensor connectivity can be either wired or wireless. Wired systems are typically fast, reliable, unimpeded by electronic interference or distance, but are expensive and time-consuming to install and are vulnerable to damage and loosened or disconnected wires. Wireless systems are cost-effective, can be placed throughout the facility with no cabling required, and are easily scalable, but may be affected by interference or long distances between sensors

A Python script on the IOT devices to send realtime environmental data to the monitoring platform.

import paho.mqtt.client as mqtt

import json

import time

import random

# Define the MQTT broker's address and port

broker\_address = "[mqtt.broker.com](http://mqtt.broker.com/)"

port = 1883

# Define the topic to which you want to publish the environmental data

topic = "environmental\_data"

# Function to simulate environmental data (replace with your actual sensor readings)

def get\_environmental\_data():

    temperature = random.uniform(20, 30)

    humidity = random.uniform(40, 60)

    data = {"temperature": temperature, "humidity": humidity}

    return json.dumps(data)

# Callback when the client connects to the MQTT broker

def on\_connect(client, userdata, flags, rc):

    if rc == 0:

        print("Connected to MQTT broker")

    else:

        print(f"Connection failed with error code {rc}")

# Create an MQTT client

client = mqtt.Client("IoTDevice")

# Set the callback functions

client.on\_connect = on\_connect

# Connect to the MQTT broker

client.connect(broker\_address, port)

# Loop to continuously publish data

while True:

    data = get\_environmental\_data()

    client.publish(topic, data)

    print(f"Published data: {data}")

    time.sleep(5) # Adjust the interval as needed

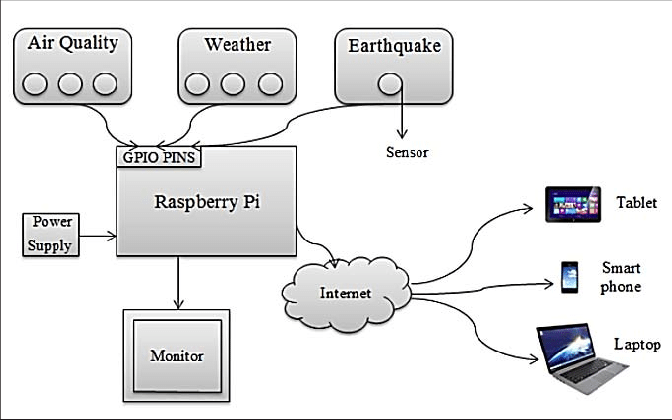
# Disconnect from the broker when done

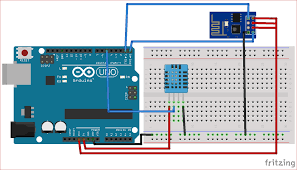
client.disconnect()

In this script, we simulate environmental data (temperature and humidity) using random values. You should replace this with actual sensor data from your IoT device. Make sure to replace the broker\_address and port with the address and port of your MQTT broker.

You'll need to install the paho-mqtt library to run this script. You can install it using pip:

pip install paho-mqtt

Once you've adapted the script to your specific hardware and sensors, it will publish environmental data to the MQTT broker at regular intervals, which can then be consumed by your monitoring platform.



Hardware implementation for This proposed system is shown in above with the blocks. Raspberry Pi is the processor and its relevant components. The Wi-Fi is used for wireless communication for IoT and Wi-Fi USB module is interfaced to Raspberry Pi’s USB port and depends upon sensor’s data is the notification is pushed in to web server and monitoring by camera and LCD is used for display the Sensors data and sensors.