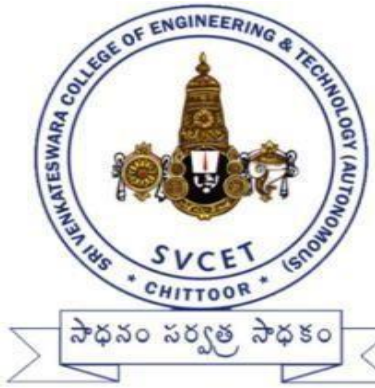


SRI VENKATESWARA COLLEGE OF ENGINEERING & TECHNOLOGY

(AUTONOMOUS)

RVS NAGAR, CHITTOOR – 517127 (AP)

(Approved by AICTE, New Delhi, Affiliated to JNTUA, Anantapur)

(Accredited by NBA, New Delhi & NAAC, Bangalore)

(An ISO 9001:2000 Certified Institution)

https://www.svcetedu.org

BACHELOR OF ELECTRONICS & COMMUNICATION ENGINEERING

PROJECT

Intelligent Bathroom Ventilation Fan Controller

Group Description

Section: A

Batch No.: 8(3)

Members Details:

S. No.	Roll No.	Name of the Student
1	21781A0470	G. Naga Lakshmi
2	21781A0477	G. Ayesha
3	21781A0478	G. Lakshmi Anjan Karthikeya

Intelligent Bathroom Ventilation Fan Controller

Aim of the Project:

The aim of this project is to design and implement an intelligent bathroom ventilation fan controller that automatically manages the ventilation based on humidity and user presence. The goal is to enhance comfort, prevent Mold growth, and improve energy efficiency.

Problem Statement & Solution:

Problem Statement

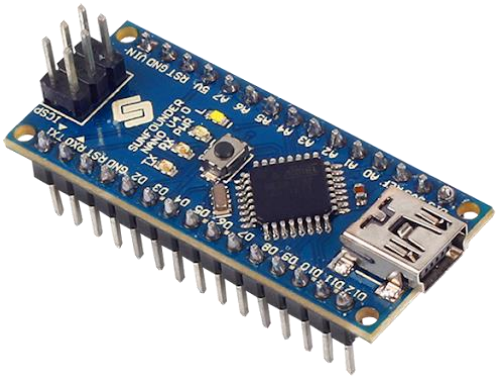
Bathrooms are prone to high humidity levels, which can lead to Mold growth and damage to fixtures. Traditional ventilation fans are either manually operated or run on timers, which may not efficiently manage humidity levels.

Solution

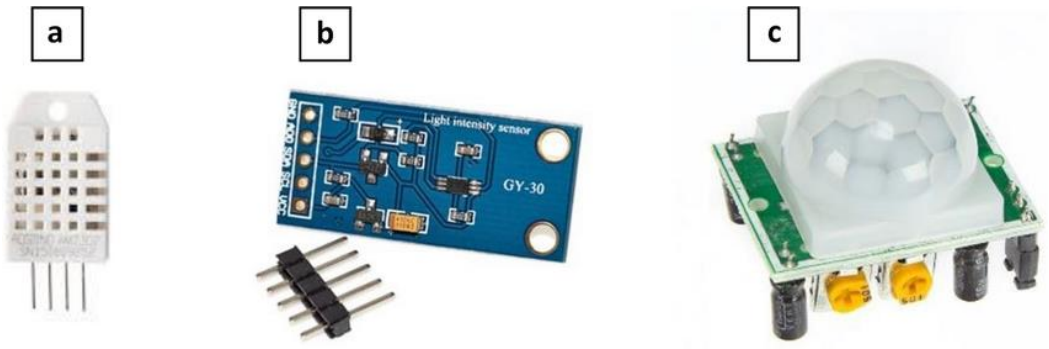
An intelligent ventilation fan controller that uses sensors to monitor humidity and presence, automatically adjusting the fan speed and operation time to maintain optimal conditions.

Project Design Specification:

Microcontroller: Arduino or ESP8266/ESP32



Sensors: Humidity sensor (DHT22), Motion sensor (PIR)



Actuators: Relay module for fan control

Power Supply: 5V/12V power supply

Connectivity: Optional Wi-Fi module for remote monitoring

User Interface: Basic LED indicators or LCD display

Project Architecture:

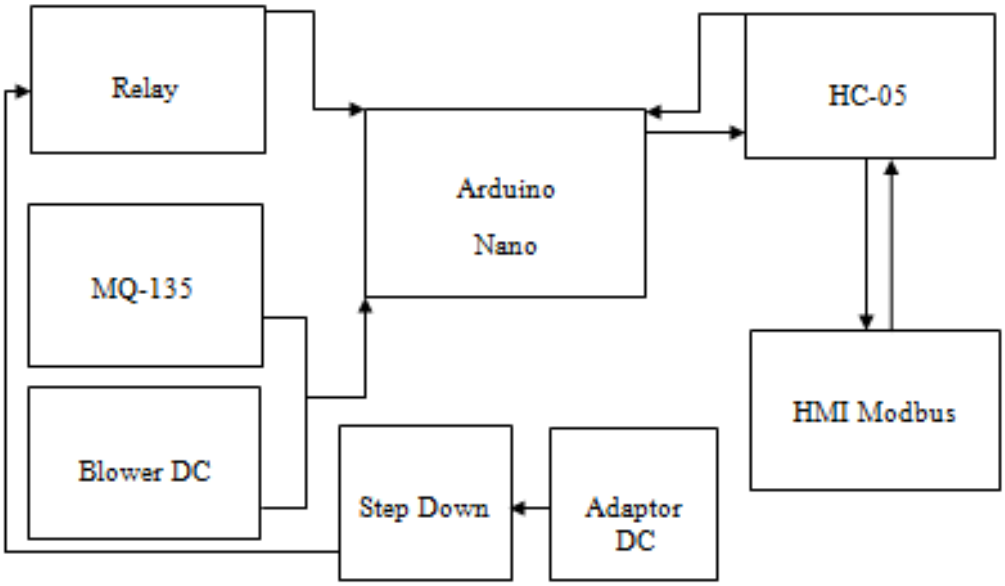


Figure 1: Architecture

Microcontroller Unit (MCU): Central processing unit for sensor data and fan control

Sensors: DHT22 for humidity, PIR sensor for presence detection

Actuators: Relay to control fan operation

Power Supply: Ensures stable power to MCU and sensors

Connectivity Module: Optional Wi-Fi module for remote access

User Interface: LEDs or LCD for displaying status

Flow Explanation:

1. Initialization: MCU initializes sensors and actuators.

2. Data Collection: Periodically read data from humidity and motion sensors.

3. Decision Making: Determine fan operation based on sensor data:

- If humidity exceeds threshold, turn on fan.
- If motion is detected, turn on fan.

4. Fan Control: Adjust fan speed or operation duration based on data.

5. Status Update: Display status on LEDs/LCD and optionally send data to a remote server.

Automatic Flow:

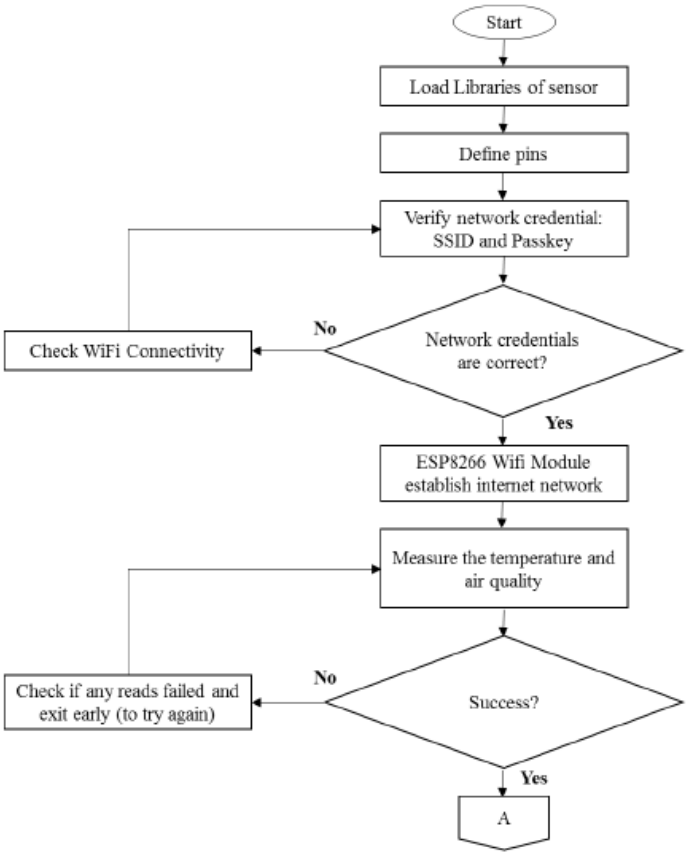


Figure 2: Process flowchart (automatic mode)

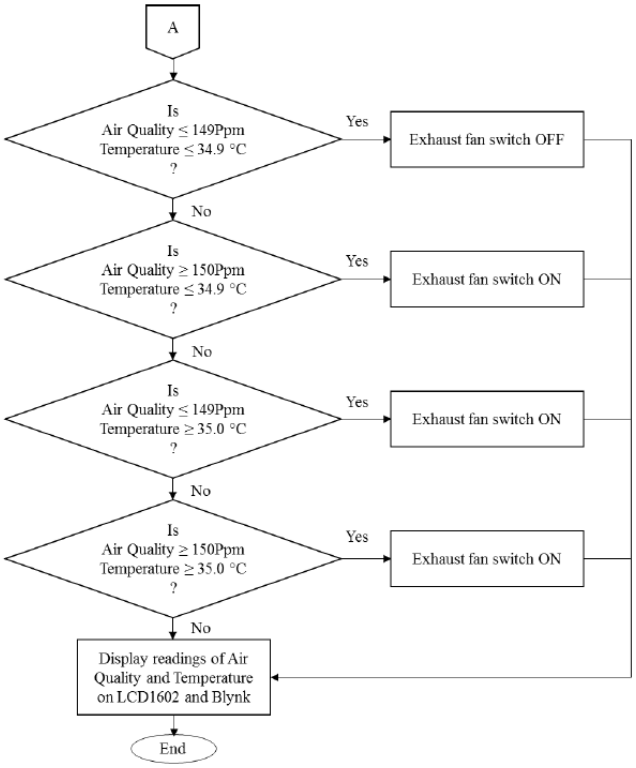


Figure 3: Process flowchart (automatic mode)

Manual Flow:

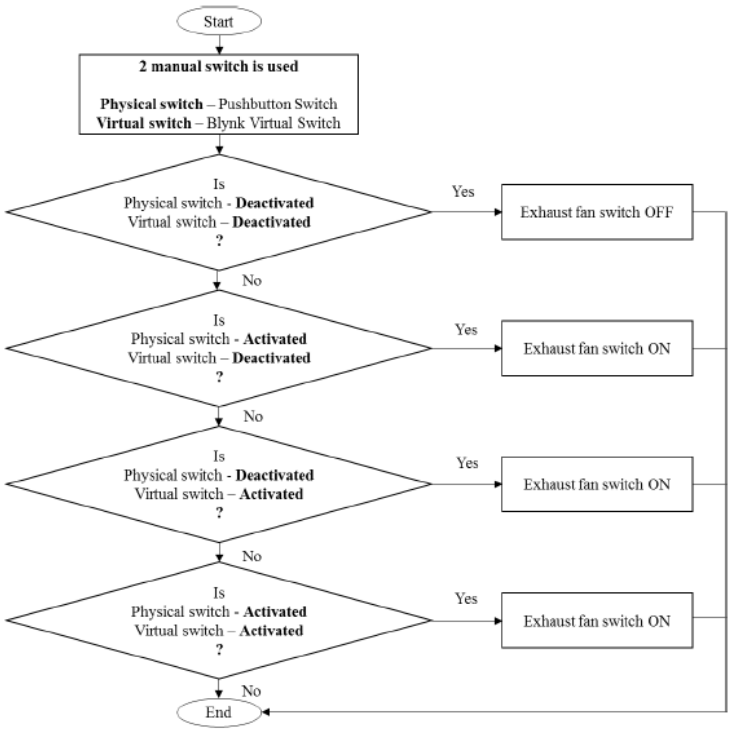
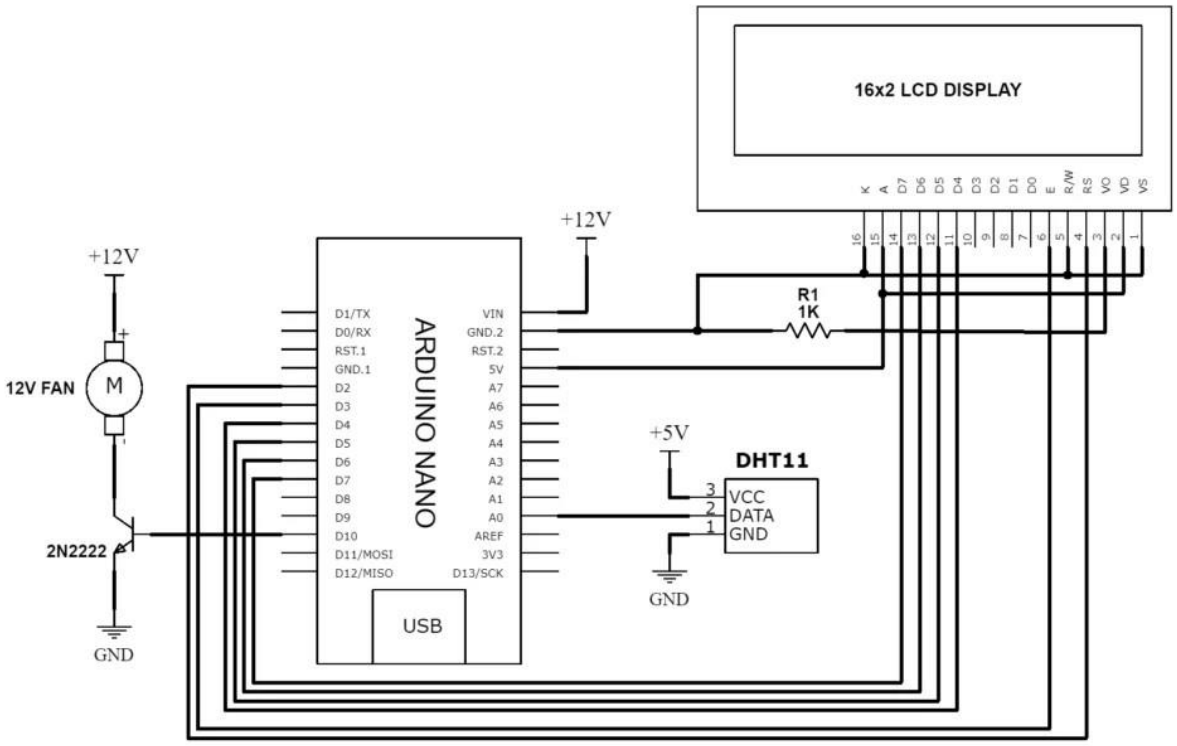


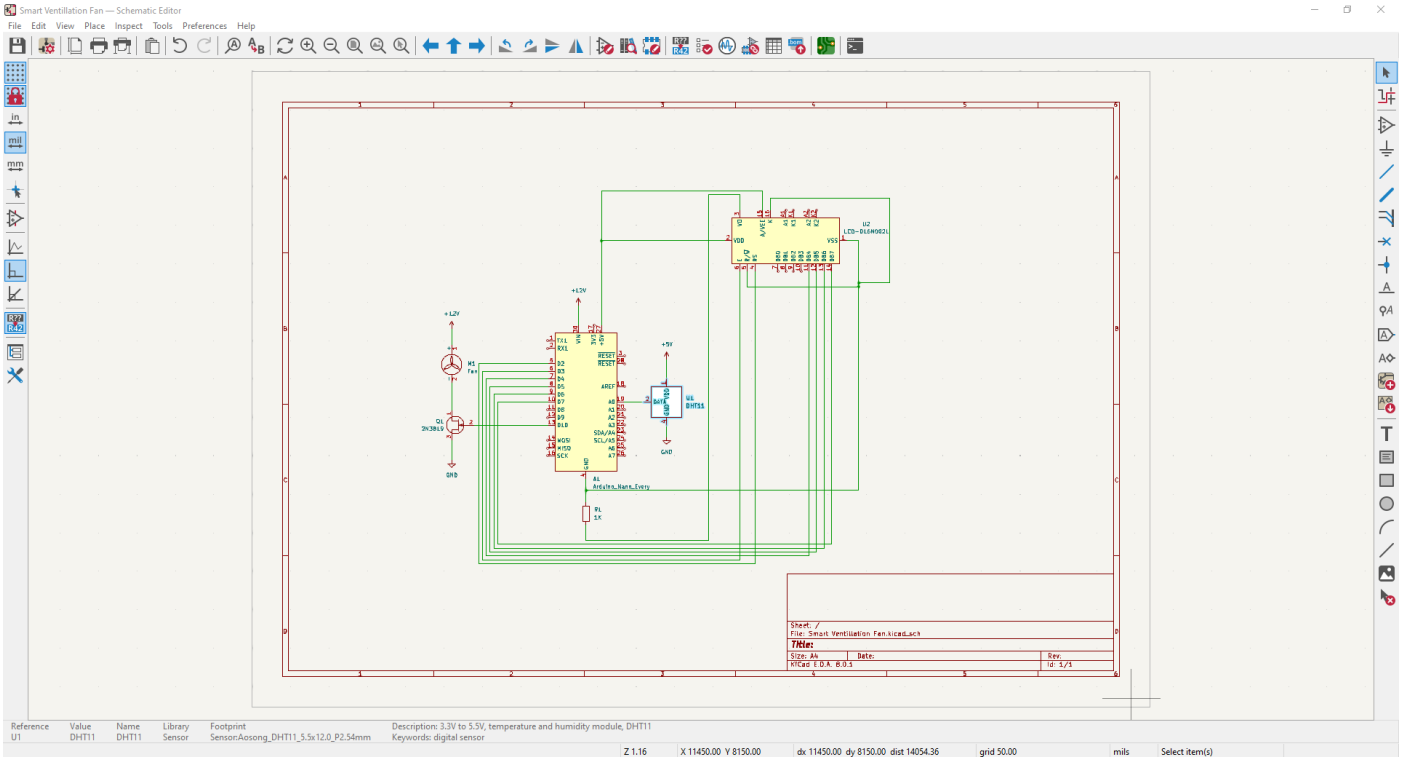
Figure 4: Process flowchart (manual mode)

Wiring Diagram:

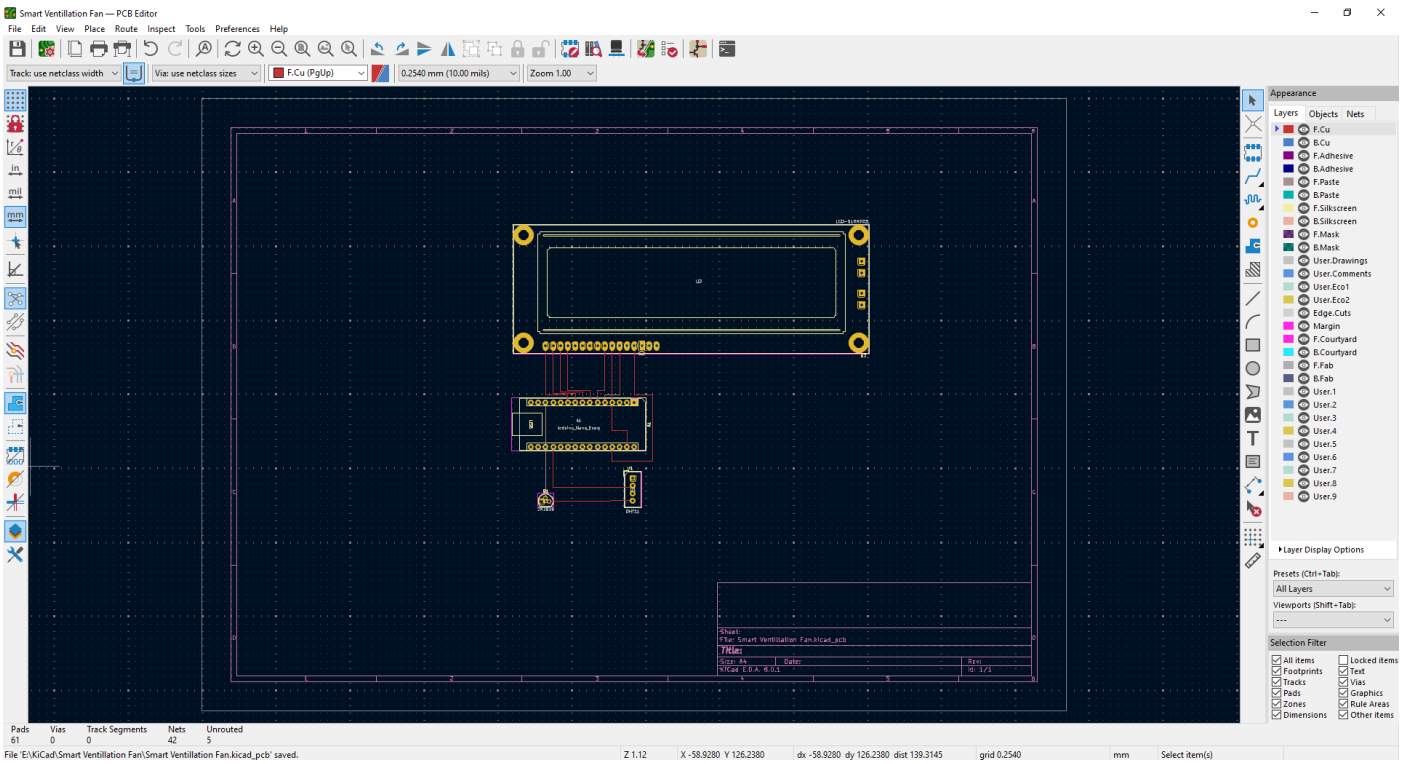


PCB Design:

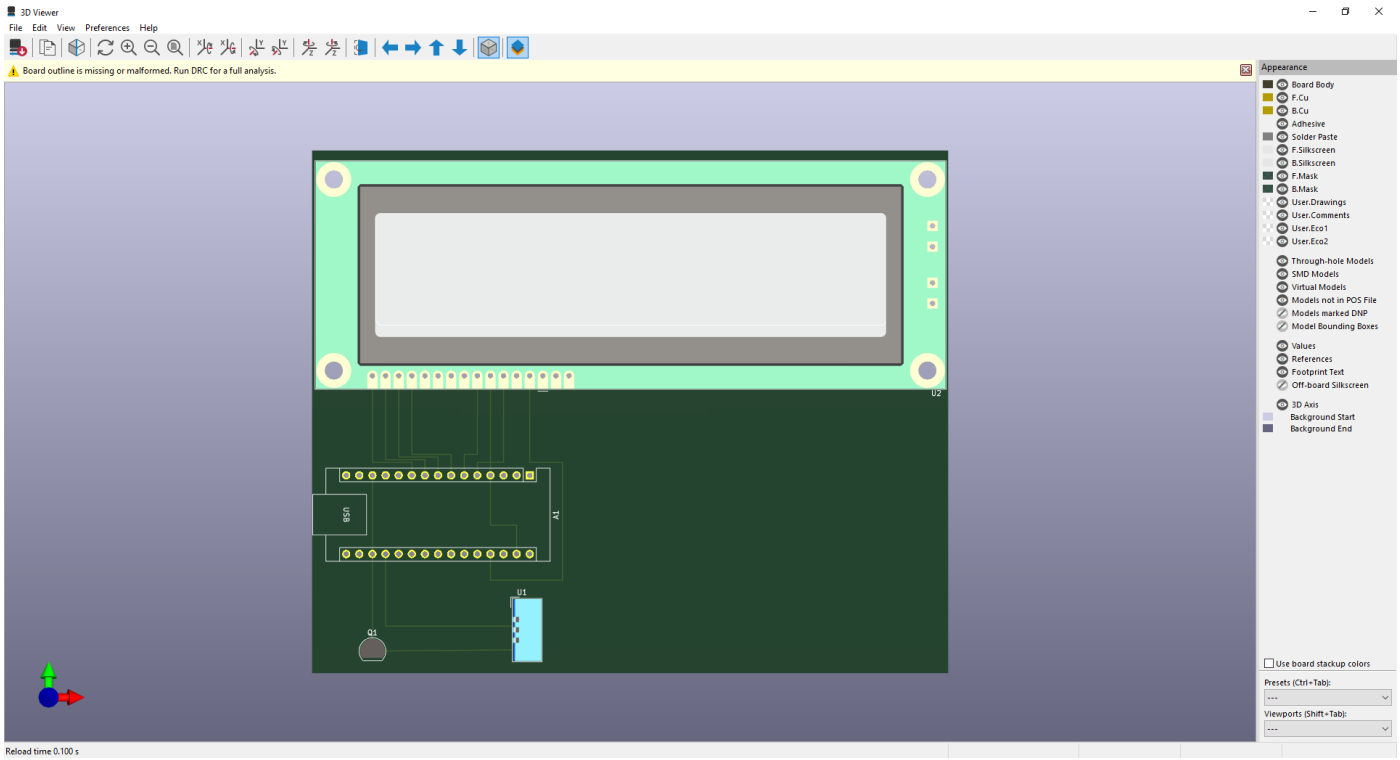
Schematic Diagram: Detailed electronic schematic of the project.



PCB Layout: Well-organized PCB design with clear component placement.



KiCad Gerber File:



Components Working Principles/Functionality:

Humidity Sensor (DHT22)

- Measures relative humidity and temperature.

- Outputs digital signal to MCU.

Motion Sensor (PIR)

- Detects human presence through infrared radiation.

- Outputs digital signal when motion is detected.

Relay Module

- Electrically operated switch.

- Allows MCU to control high-power fan using low-power signals.

Microcontroller (ESP8266/ESP32)

- Processes sensor data.

- Executes control logic and drives the relay.

Software Used:

- KiCad 8.0

- Thonny

Program/Code:

```
// Template and device credentials for Blynk
#define BLYNK_TEMPLATE_ID "TMPL6YAS5_ss3"
#define BLYNK_TEMPLATE_NAME "IoT Smart Exhaust Fan"
#define BLYNK_PRINT Serial  // Directs Blynk messages to the serial port

#include <WiFi.h>  // Include WiFi library for ESP32
#include <BlynkSimpleEsp32.h>  // Include Blynk ESP32 library
#include "DHT.h"  // Include DHT sensor library

// Blynk and WiFi credentials
const char auth[] = "*****************";  // Blynk authentication token
const char ssid[] = "*****************";  // WiFi SSID
const char pass[] = "*****************";  // WiFi password

// Pin configuration
#define DHTPIN 22  // DHT sensor pin
#define DHTTYPE DHT11  // Type of DHT sensor

const int gasSensorPin = 34;  // Gas sensor pin
const int relayPin = 23;  // Relay module pin

const int gasThreshold = 20;  // Gas level threshold for triggering the relay

DHT dht(DHTPIN, DHTTYPE);  // Initialize DHT sensor

bool manualMode = false;  // Flag to track if the relay is in manual mode

void setup()
{
  Serial.begin(115200);  // Start serial communication at 115200 baud rate
  dht.begin();  // Initialize DHT sensor
  Blynk.begin(auth, ssid, pass);  // Connect to Blynk using WiFi
```

```
  pinMode(relayPin, OUTPUT);  // Set relay pin as output
  pinMode(gasSensorPin, INPUT);  // Set gas sensor pin as input

   digitalWrite(relayPin, LOW);  // Ensure relay is off initially

  delay(2000);  // Wait for 2 seconds before proceeding
}

// Function to handle manual relay control from Blynk app
BLYNK_WRITE(V4)
{
  int relayControl = param.asInt();  // Get value from Blynk app
  manualMode = (relayControl == 1);  // Set manual mode based on app input

  if (manualMode)
  {
    digitalWrite(relayPin, HIGH);  // Turn on relay in manual mode
  }
  else
  {
    digitalWrite(relayPin, LOW);  // Turn off relay when exiting manual mode
  }

  Blynk.virtualWrite(V3, relayControl);  // Update relay status on Blynk
}

void loop()
{
  Blynk.run();  // Run Blynk

  int sensorValue = analogRead(gasSensorPin);  // Read gas sensor value
```

```
  int gas_percentage = map(sensorValue, 0, 4095, 0, 100);  // Convert to
percentage


  float humidity = dht.readHumidity();  // Read humidity from DHT sensor
  float temperature = dht.readTemperature();  // Read temperature from DHT
sensor


  // Check if sensor readings are valid
  if (isnan(humidity) || isnan(temperature))
  {
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
  }


  // Print sensor values to the Serial Monitor
  Serial.print("Humidity: ");
  Serial.print(humidity);
  Serial.println("%");


  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.println("°C ");


  Serial.print("Gas sensor value: ");
  Serial.println(sensorValue);


  Serial.print("Gas Percentage: ");
  Serial.print(gas_percentage);
  Serial.println("%");


  Serial.println();


  // Automatic control logic based on gas levels
  if (!manualMode)
```

```
  {
    if (gas_percentage > gasThreshold)
    {
      digitalWrite(relayPin, HIGH);  // Activate relay if gas above threshold
      Blynk.virtualWrite(V3, HIGH);  // Update Blynk app
    }
    else
    {
      digitalWrite(relayPin, LOW);  // Deactivate relay if gas below threshold
      Blynk.virtualWrite(V3, LOW);  // Update Blynk app
    }
  }

  // Send sensor values to Blynk
  Blynk.virtualWrite(V0, gas_percentage);  // Send gas percentage
  Blynk.virtualWrite(V1, temperature);  // Send temperature
  Blynk.virtualWrite(V2, humidity);  // Send humidity

  delay(1000);  // Wait for a second before next loop iteration
}
```

Bill of Materials (BoM): List of all components with specifications.

"Reference","Value","Datasheet","Footprint","Qty","DNP"

"A1","Arduino_Nano_Every","https://content.arduino.cc/assets/NANOEveryV3.0_sch.pdf","Module:Arduino_Nano","1",""

"M1","Fan","~","","1",""

"Q1","2N3819","https://my.centralsemi.com/datasheets/2N3819.PDF","Package_TO_SOT_THT:TO-92","1",""

"R1","1K","~","","1",""

"U1","DHT11","http://akizukidenshi.com/download/ds/aosong/DHT11.pdf","Sensor:Aosong_DHT11_5.5x12.0_P2.54mm","1",""

"U2","LCD-016N002L","http://www.vishay.com/docs/37299/37299.pdf","Display:LCD-016N002L","1",""

Project Outcome:

Functional Testing: Verify the operation of the intelligent fan controller.

Efficiency: Assess the energy savings and improved humidity control.

User Feedback: Collect feedback from users to refine the design.

Appendices

Code Listings: Include source code for the microcontroller.

Schematics and Layouts: Detailed images of schematics and PCB layouts.

Datasheets: Provide datasheets for key components.