ECE 445

SENIOR DESIGN LABORATORY

FINAL PRESENTATION HANDOUT

Automatic Humidity Sensing and Water Refilling Cool-Mist Humidifier

---

Team No. 11

ANDREW SHERWIN
(zyxie2@illinois.edu)
Jalen Chen
(jalenc3@illinois.edu)
Woojin Kim
(wkim51@illinois.edu)

TA: Surya Vasanth
Professor: Dr. Jonathon Schuh

April 29, 2024

Abstract

Improper indoor humidity levels are a cause for health issues, such as dry skin or inflamed sinus passages. With the majority of humidifiers on the market being manually controlled, this causes an issue when no operator is present. A room can be under-humidified or over-humidified when there is no one to control the humidity output. As a result, we propose a cost-effective humidifier that will mitigate this problem. Concerned about the health of the operator, a cool-mist humidifier design was chosen. This humidifier communicates via 2.4GHz WIFI with multiple remote humidity sensors before determining the need to turn itself on or off. When the humidifier filter needs more water, the humidifier will activate a water valve that will irrigate the filter to the perfect amount. With all the components combined, this humidifier brings a new perspective to the humidifiers already on the market, guaranteeing the safety and comfort of the user. A truly luxurious experience.

Table of Contents

Introduction

Problem and Solution

The United States Environmental Protection Agency (EPA) strongly suggests the regulation of indoor humidity to be between 30%-50%. When indoor humidity falls within these ranges, pollutants, such as chemicals, gasses, mold, and other airborne particles, are minimized in the air. This ultimately helps with allergies, respiratory illnesses, and other health issues.

To maximize the health benefits of our product, we decided to implement a cool mist humidifier. Unlike ultrasonic humidifiers and warm mist humidifiers, this product does not disperse bacterial matters, minerals, or cause your nasal passages, as the US Food and Drug Administration (FDA) suggests. To further improve humidifiers on the market, our humidifier features an automatic-on function and an automatic water tank filling function. The solution has three remote humidity sensors. These sensors detect the humidity and communicate the humidity readings with the humidifier. Using the average of these readings, with the temperature accounted for, the humidifier decides whether it is necessary to turn itself on or off. Inside the water tank of the humidifier are two water sensors. One sensor detects whether more water is needed, while the other sensor detects when to stop automatically adding water. Furthermore, a webpage for the humidifier allows the user to read the current humidity and temperature and has options to turn the humidifier and water valve on manually. With all these improvements, we push out a revolutionary humidifier system onto the market that has a high level of autonomy, providing a luxurious product to the end-user.

High Level Requirements

Our project would need to achieve a multitude of high-level goals to be sufficiently complete. Some goals would include:

1. The ESP32 can read data from the humidity sensor using the ESP32 microcontroller communication through the three different sensors placed around the room, communicating all with ESPs via 2.4 GHz Wi-Fi with continuous readings. This will allow us to use average humidity to determine a good range for on/off functionality.

2. The filter irrigation system refills the water tank with water depending on signals from two water level sensors at top and bottom of the tank. The ESP32 will activate the 12V DC water valve, creating a flow of water which will turn off when the top sensor detects water. The measured max capacity of the water tank, according to manufacturer specifications, should be 4.16 liters every 24 hours and the filter should absorb around 150 milliliters ± 5 milliliters.

3. The humidifier should be operating between 35% and 50% humidity. If the value is beyond 50, the humidifier turns off, and if below 35, it turns on. With remote communication between the humidity sensor's ESP32 microcontroller and the humidifier's ESP32 microcontroller, there is a theoretical estimation of 44ns response time to the activation and the deactivation of the humidifier's fans, when algorithm and code run-times are not considered. Furthermore, high voltage parts of the PCB will not affect the ESP32 and other low voltage parts.

RV Tables

Humidifier Subsystem

Requirements	Verification
• **Provide 3.3V +/- 0.5% from an input voltage of 5V +/- 0.1% DC source** • **Thermal stability maintains below 120'C**	• Use multimeter to test and take average voltage output over 1-hour to test stability • Monitor heat dissipation with thermal laser gun over an average of 1-hour use time, expected +/- 5'C under load
• **ESP32 communication with remote sensor ESP32 utilizing 2.4GHz Wi-Fi**	• Verify functionality of ESP32 Wi-Fi component by creating an access point, and accessing access point to remotely turn on LED lights • Verify communication between two ESP32 boards by sending command to turn on LED light from one ESP32 board • Verify thermal performance of ESP32 chip during operation by probing with a laser thermal gun
• **ESP32 control of activation and deactivation of water valve** • **ESP32 control of activation and deactivation of fan**	• Verify functionality of ESP32 communication with DC relay by sending activation signal to water valve/fan DC relay controller, and when signal is received, DC relay controllers output a lit LED light • Probe the output voltage of the DC relay controllers to make sure it is 12V +/- 0.5%

• **ESP32 detects signals from two water-level sensors**	• Contact/No Contact water to the water-level sensors and check through Arduino IDE to see if ESP32 receives contact/no contact signal from the two water-level sensors • Water level sensor low signal when in contact with water, and high signal when not in contact with water

Humidity Sensor Subsystem

Requirements	Verification
• **Provide 3.3V +/- 0.5% from an input voltage of 5V +/- 0.1% DC source** • **Thermal stability maintains below 120'C**	• Use multimeter to test and take average voltage output over 1-hour to test stability • Monitor heat dissipation with thermal laser gun over an average of 1-hour use time, expected +/- 5'C under load
• **ESP32 communication with humidity sensor receiving humidity data at least once in a minute** • **Humidity sensor measuring humidity data at least between 25% - 55%**	• Verify communication between ESP32 and humidity sensor Arduino IDE by checking humidity data ESP32 receives • Compare the humidity data received by ESP32 to the commercial humidity sensor and check if the humidity is within +/- 3% • Make the air dry/moist to see if the sensor can measure humidity between 25% - 55% • Exhale on sensor to see humidity levels are raised • Compare humidity reading with reference humidity reader

References

[1] F. Hecht, “Wifi Propagation,” *freedom*, 2017. https://doc.freefem.org/tutorials/wifiPropagation.html (accessed Mar. 29, 2024).
[2] M. Marwell, “Issues with the I^2C (Inter-IC) Bus and How to Solve Them,” *DigiKey*, Aug. 09, 2018. Accessed: Mar. 29, 2024. [Online]. Available: https://www.digikey.com/en/articles/issues-with-the-i2c-bus-and-how-to-solve-them
[3] MetaGeek, “Wi-Fi and Non Wi-Fi Interference,” *MetaGeek*, 2024. https://www.metageek.com/training/resources/wifi-and-non-wifi-interference/ (accessed Mar. 29, 2024).
[4] A. V. Arundel, E. M. Sterling, J. H. Biggin, and T. D. Sterling, “Indirect health effects of relative humidity in indoor environments.,” *Environmental Health Perspectives*, vol. 65, no. 65, pp. 351–361, Mar. 1986, doi: 10.1289/ehp.8665351.
[5] V. Perrone *et al.*, “The Epidemiology, Treatment Patterns and Economic Burden of Different Phenotypes of Multiple Sclerosis in Italy: Relapsing-Remitting Multiple Sclerosis and Secondary Progressive Multiple Sclerosis,” *Clinical Epidemiology*, vol. Volume 14, no. 14, pp. 1327–1337, Nov. 2022, doi: 10.2147/clep.s376005.
[6] IEEE, “IEEE Code of Ethics,” *IEEE Code of Ethics*, 2020. https://www.ieee.org/about/corporate/governance/p7-8.html (accessed Mar. 29, 2024).
[7] University of Illinois, “Salary Averages,” *UIUC*, 2022. https://ece.illinois.edu/admissions/why-ece/salary-averages (accessed Mar. 29, 2024).
[8] Environmental Protection Agency, “Care for your air: A guide to indoor air quality,” *US EPA*, Aug. 07, 2023. https://www.epa.gov/indoor-air-quality-iaq/care-your-air-guide-indoor-air-quality (accessed Mar. 25, 2024).
[9] Sensirion, “Datasheet -SHT4x,” *Sensirion*, Aug. 2023. https://sensirion.com/media/documents/33FD6951/6555C40E/Sensirion_Datasheet_SHT4x.pdf
[10] A. Sherwin, *ECE445 Team 11 - Sensor Subsystem Schematic*. 2024.
[11] A. Sherwin, *ECE445 Team 11 - Humidifier Subsystem Schematic*. 2024.
[12] Espressif, “ESP32-S3-WROOM-1 ESP32-S3-WROOM-1U Datasheet,” *Espressif*, 2023. https://www.espressif.com/sites/default/files/documentation/esp32-s3-wroom-1_wroom-1u_datasheet_en.pdf
[13] IEEE, “IEEE Code of Ethics,” *IEEE Code of Ethics*, Jun. 2020. https://www.ieee.org/about/corporate/governance/p7-8.html (accessed Mar. 26, 2024).
[14] Espressif, “sch_esp32-s3-devkitc-1_v1_20210,” *Espressif*, Apr. 13, 2022. https://dl.espressif.com/dl/schematics/SCH_ESP32-S3-DevKitC-1_V1.1_20220413.pdf (accessed Mar. 26, 2024).
[15] A. Sherwin, *Sensor Subsystem Block Diagram*. 2024.
[16] A. Sherwin, Humidity Subsystem Block Diagram. 2024.
[17] D. Workshop, “ESP NOW - Peer to Peer ESP32 Communications,” DroneBot Workshop, Apr. 03, 2022. https://dronebotworkshop.com/esp-now/
[18] K. Rembor, “Adafruit Sensirion SHT40, SHT41 & SHT45 Temperature & Humidity Sensors,” Adafruit Learning System, Feb. 04, 2021. https://learn.adafruit.com/adafruit-sht40-temperature-humidity-sensor/arduino

[19] cplusplus, “std::chrono::high_resolution_clock::now,” cplusplus. https://cplusplus.com/reference/chrono/high_resolution_clock/now/ (accessed Apr. 20, 2024).
[20] cplusplus, “std::chrono::nanoseconds,” cplusplus. https://cplusplus.com/reference/chrono/nanoseconds/ (accessed Apr. 20, 2024).
[21] me-no-dev, “ESPAsyncWebServer/README.md at master · me-no-dev/ESPAsyncWebServer,” GitHub. https://github.com/me-no-dev/ESPAsyncWebServer/blob/master/README.md
[22] Tea, “ESP32 Web Server periodic updating problem,” Stack Overflow. https://stackoverflow.com/questions/64610221/esp32-web-server-periodic-updating-problem
[23] antepher, “ESP32 Arduino: HTTP server over soft AP,” techtutorialsx, Jan. 07, 2018. https://techtutorialsx.com/2018/01/07/esp32-arduino-http-server-over-soft-ap/
[24] ESPRESSIF, “ESP-NOW - ESP32 - — ESP-IDF Programming Guide v5.2.1 documentation,” espressif.com. https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-reference/network/esp_now.html