Kwame Nkrumah University Of Science and Technology

Microprocessors -Group 16

Project Topic: A smart Poultry Incubator

Group Members:

- Akatey Collins 1815322
- Ibida Margaret-Mary Kanulia 1822822
- fazl Maltiti yahaya froko-1829722
- Sarfo Nana Kofi Asante -1827922
- Copson Wallace Walter - 1820622
- Afua Twumasi Britwum - 1829222
- Boakye Yiadom Kwadwo Wiredu - 1819922
- Gbadago Bill Etornam Kwame - 1822222
- Aning Akwasi Acheampong – 1817022
- Nisy Adjei Acheampong – 1813022

Contents

Introduction 3

Problem Statement 3

Methodology 3

Hardware Components 3

i. ESP32 Microcontroller 3

ii. Humidity and Temperature Sensor: DHT22 4

iii. LCD Display 4

iv. LED 5

v. Resistors 5

vi. Jumper Wires 6

Software Components 6

i. Wokwi Software 6

ii. DHTesp Library: 6

iii. Blynk IOT 6

Circuit Design 6

System Functionality 7

Demonstration/Simulation 8

Conclusion 10

Appendix 10

The code for the project 10

Reference 11

Introduction

Incubation plays a crucial role in poultry farming, ensuring a controlled environment for egg hatching. Traditional incubation methods often rely on manual temperature and humidity control, leading to inconsistencies and lower hatch rates. This **Smart Poultry Incubator** automates the process using an ESP32 microcontroller, a DHT22 sensor, and an LCD display, maintaining optimal conditions for successful incubation.

The Smart Poultry Incubator is an Arduino-based automated system designed to regulate temperature and humidity for optimal incubation conditions. The system continuously monitors these parameters using a DHT22 sensor and adjusts the heating and cooling mechanisms accordingly.

Problem Statement

Maintaining optimal temperature and humidity levels is crucial for successful egg incubation in poultry farming. Traditional incubators require frequent manual monitoring and adjustments, leading to inconsistencies in environmental conditions that can reduce hatch rates and increase labor demands.

This project aims to develop a smart poultry incubator using an ESP32 microcontroller and DHT22 sensor to continuously monitor temperature and humidity levels. The system will automatically regulate heating and cooling mechanisms based on real-time sensor readings. An LCD display will provide real-time feedback, and an automated control system will ensure that the incubator maintains stable conditions within the desired thresholds. This smart approach enhances incubation efficiency, improves hatch rates, and reduces human intervention.

Methodology

Hardware Components

i. ESP32 Microcontroller

The ESP32 microcontroller serves as the brain of the smart poultry incubator, handling real-time monitoring and automated control of temperature and humidity to create optimal incubation conditions. It reads data from the DHT22 sensor, processes it, and determines whether to activate the heater or cooling fan based on predefined thresholds. Additionally, it controls two LCD screens, displaying real-time temperature, humidity, and system status updates for easy monitoring. With its fast-processing speed, multiple GPIO pins, and built-in WiFi and Bluetooth, the ESP32 ensures efficient automation while allowing for future IoT integration, enabling remote monitoring and control. Its low power consumption and versatility make it ideal for this project, reducing manual intervention while improving incubation success rates.

Figure 1 Esp32

ii. Humidity and Temperature Sensor: DHT22

The DHT22 sensor is a crucial component of the smart poultry incubator, responsible for accurately measuring temperature and humidity inside the incubation chamber. It provides real-time data to the ESP32 microcontroller, which processes this information and decides whether to activate the heater or cooling fan to maintain optimal conditions for egg incubation. The sensor ensures precise environmental control, helping to improve hatch rates by keeping temperature and humidity within the required range. Its high accuracy and reliability make it an excellent choice for this automated incubation system.

Figure 2 DHT22 Sensor

iii. LCD Display

The two 20x4 I2C LCD displays play a vital role in the smart poultry incubator by providing real-time feedback on system status. One display shows the temperature and humidity readings from the DHT22 sensor, allowing users to monitor incubation conditions at a glance. The second display indicates the status of the heating and cooling system, informing whether the heater or fan is active or if the temperature is stable. Using I2C communication, these LCDs require minimal wiring and improve user interaction, making it easy to track the incubator's performance without additional hardware or software interfaces.

Figure 3 LCD

iv. LED

The system includes two LEDs for manual simulation: a red LED to indicate when the heating system is active and a blue LED to indicate when the cooling system is active.

Figure 4 LED

v. Resistors

Incorporating two resistors in series with the red and blue LEDs serves a crucial function: limiting the current flowing through each LED to prevent damage and ensure proper operation. LEDs are designed to operate within specific current ranges, and exceeding these limits can lead to overheating, reduced lifespan, or immediate failure.

Figure 5 Resistors

vi. Jumper Wires

They are essential components in electronics prototyping, consisting of insulated conductors with connectors at each end. They enable quick and reliable connections between different points in a circuit without the need for soldering. In your smart poultry incubator project, jumper wires facilitate the interconnection of components such as the ESP32 microcontroller, DHT22 sensor, LCD displays, LED indicators, and other elements on a breadboard or within the circuit setup.

Software Components

i. Wokwi Software

Wokwi is a free, web-based simulator designed for prototyping and testing embedded systems projects directly within your browser.
It was used in the simulation of the project and also writing of the codes.

ii. DHTesp Library:

For handling temperature and humidity data from the DHT22 sensor.

iii. Blynk IOT

In our project, we utilized the Blynk IoT platform to facilitate real-time monitoring of temperature and humidity data on a mobile device. By integrating the ESP32 microcontroller with a DHT22 sensor, we were able to capture environmental readings and transmit them to the Blynk application over Wi-Fi. This setup allowed for instantaneous data visualization and remote access, enhancing user convenience and system responsiveness. The Blynk platform's customizable widgets enabled the creation of an intuitive mobile dashboard, displaying real-time temperature and humidity metrics. This integration not only streamlined the monitoring process but also provided a scalable solution for future IoT applications.

Circuit Design

A comprehensive circuit diagram was developed, detailing the interconnections between the ESP32, sensors, LEDs, resistors, and LCD displays.

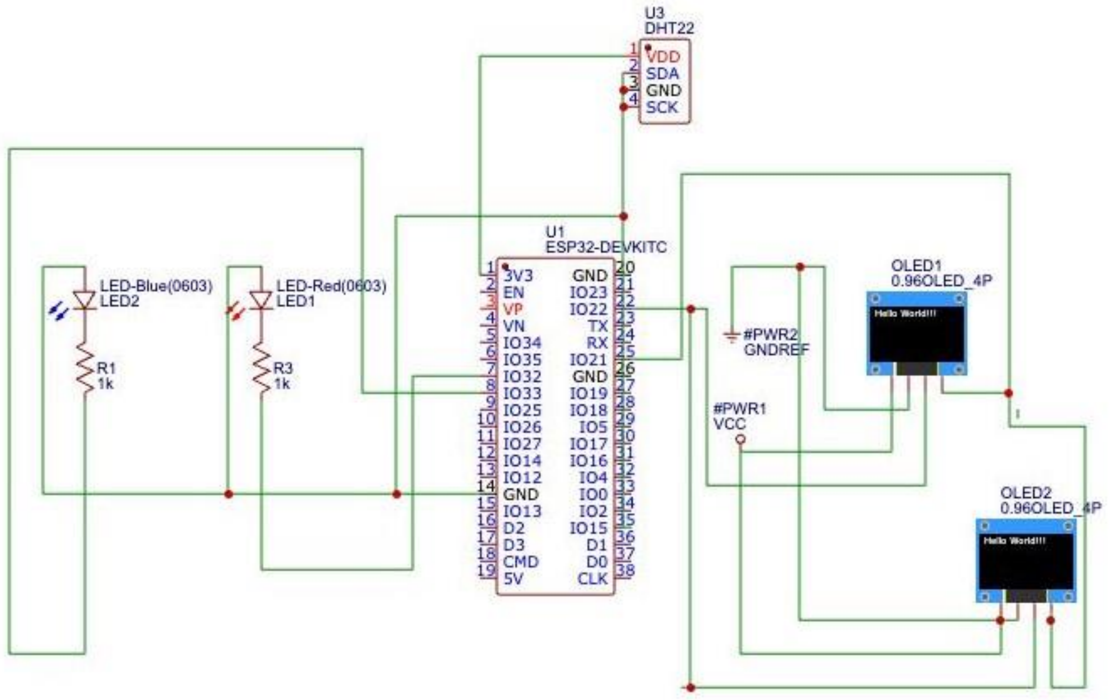


Figure 6 Circuit diagram of the project

System Functionality

i. **Sensor Integration:** Libraries for the DHT22 sensor were utilized to facilitate accurate data acquisition.

ii. Actuator Control Logic: Conditional statements were employed to activate heating or cooling mechanisms based on manual control, maintaining the target temperature of 37.5°C with a tolerance of ±0.5°C. Similarly, humidity control ensures a target of 55°C with a tolerance of ±5°C.

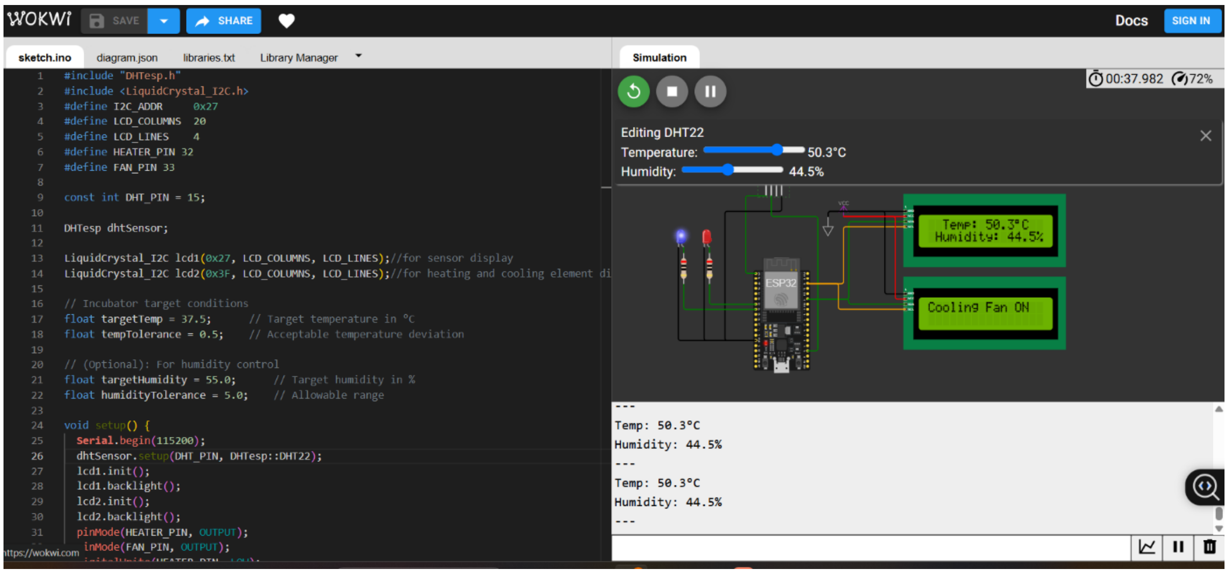
iii. **Indicator Activation:** The red and blue LEDs were programmed to illuminate corresponding to the activation of heating and cooling systems, respectively. The red light illuminates when the temperature is low compared to the target temperature which indicates that heater is On. The blue light illuminates when the temperature of the incubator is higher than the targeted temperature which indicates that the cooling system is active.

iv. **Display Updates:** The LCD displays were configured to present real-time temperature, humidity, and system status information.

v. **Blynk Integration:** The Blynk IoT platform was integrated to enable remote monitoring. The ESP32 was connected to the Blynk application over Wi-Fi, transmitting sensor data for real-time visualization on a mobile device. This setup allowed for remote access and monitoring of the incubator's environmental conditions.

Demonstration/Simulation

In **Fig. 6** below, the incubator's temperature is manually adjusted to exceed the target temperature. One of the LCDs displays the temperature and humidity values, while the blue LED is illuminated, indicating that the cooling system is active.



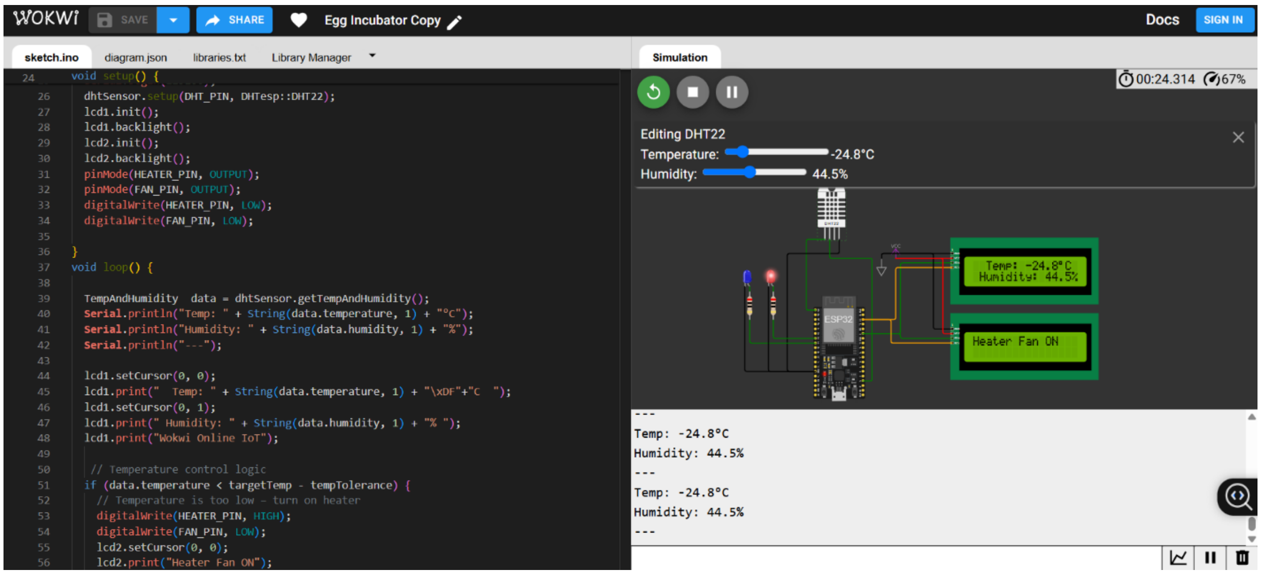
```
#include "DHTesp.h"
#include <LiquidCrystal_I2C.h>
#define I2C_ADDR     0x27
#define LCD_COLUMNS  20
#define LCD_LINES    4
#define HEATER_PIN 32
#define FAN_PIN 33

const int DHT_PIN = 15;

DHTesp dhtSensor;

LiquidCrystal_I2C lcd1(0x27, LCD_COLUMNS, LCD_LINES);//for sensor display
LiquidCrystal_I2C lcd2(0x3F, LCD_COLUMNS, LCD_LINES);//for heating and cooling element di

// Incubator target conditions
float targetTemp = 37.5;      // Target temperature in °C
float tempTolerance = 0.5;    // Acceptable temperature deviation

// (Optional): For humidity control
float targetHumidity = 55.0;      // Target humidity in %
float humidityTolerance = 5.0;    // Allowable range

void setup() {
  Serial.begin(115200);
  dhtSensor.setup(DHT_PIN, DHTesp::DHT22);
  lcd1.init();
  lcd1.backlight();
  lcd2.init();
  lcd2.backlight();
  pinMode(HEATER_PIN, OUTPUT);
  inMode(FAN_PIN, OUTPUT);
```

Editing DHT22
Temperature: 50.3°C
Humidity: 44.5%

```
---
Temp: 50.3°C
Humidity: 44.5%
---
Temp: 50.3°C
Humidity: 44.5%
---
```

Figure 7 When the Cooling system is On

In **Fig. 7** below, the incubator's temperature is manually adjusted to be lower than the target temperature. One of the LCDs displays the temperature and humidity values, while the red LED is illuminated, indicating that the heating system is active.

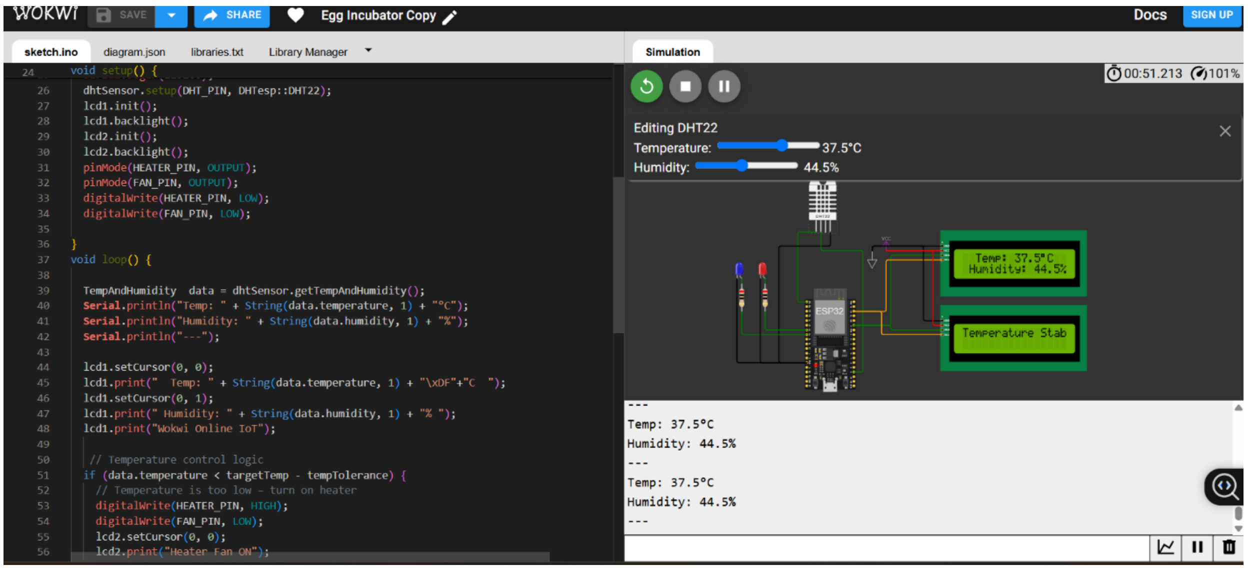


```
void setup() {
  dhtSensor.setup(DHT_PIN, DHTesp::DHT22);
  lcd1.init();
  lcd1.backlight();
  lcd2.init();
  lcd2.backlight();
  pinMode(HEATER_PIN, OUTPUT);
  pinMode(FAN_PIN, OUTPUT);
  digitalWrite(HEATER_PIN, LOW);
  digitalWrite(FAN_PIN, LOW);

}
void loop() {

  TempAndHumidity  data = dhtSensor.getTempAndHumidity();
  Serial.println("Temp: " + String(data.temperature, 1) + "°C");
  Serial.println("Humidity: " + String(data.humidity, 1) + "%");
  Serial.println("---");

  lcd1.setCursor(0, 0);
  lcd1.print("  Temp: " + String(data.temperature, 1) + "\xDF"+"C  ");
  lcd1.setCursor(0, 1);
  lcd1.print(" Humidity: " + String(data.humidity, 1) + "% ");
  lcd1.print("Wokwi Online IoT");

   // Temperature control logic
  if (data.temperature < targetTemp - tempTolerance) {
    // Temperature is too low - turn on heater
    digitalWrite(HEATER_PIN, HIGH);
    digitalWrite(FAN_PIN, LOW);
    lcd2.setCursor(0, 0);
    lcd2.print("Heater Fan ON");
```

```
---
Temp: -24.8°C
Humidity: 44.5%
---
Temp: -24.8°C
Humidity: 44.5%
---
```

Figure 8 When the heating system is On

In **Fig. 8** below, the incubator's temperature is manually adjusted to be equal to the target temperature. One of the LCDs displays the temperature and humidity values, while the red and blue LED remains Off, indicating that the temperature and the humidity of the system is stable.



```
void setup() {
  dhtSensor.setup(DHT_PIN, DHTesp::DHT22);
  lcd1.init();
  lcd1.backlight();
  lcd2.init();
  lcd2.backlight();
  pinMode(HEATER_PIN, OUTPUT);
  pinMode(FAN_PIN, OUTPUT);
  digitalWrite(HEATER_PIN, LOW);
  digitalWrite(FAN_PIN, LOW);

}
void loop() {

  TempAndHumidity  data = dhtSensor.getTempAndHumidity();
  Serial.println("Temp: " + String(data.temperature, 1) + "°C");
  Serial.println("Humidity: " + String(data.humidity, 1) + "%");
  Serial.println("---");

  lcd1.setCursor(0, 0);
  lcd1.print("  Temp: " + String(data.temperature, 1) + "\xDF"+"C  ");
  lcd1.setCursor(0, 1);
  lcd1.print(" Humidity: " + String(data.humidity, 1) + "% ");
  lcd1.print("Wokwi Online IoT");

   // Temperature control logic
  if (data.temperature < targetTemp - tempTolerance) {
    // Temperature is too low - turn on heater
    digitalWrite(HEATER_PIN, HIGH);
    digitalWrite(FAN_PIN, LOW);
    lcd2.setCursor(0, 0);
    lcd2.print("Heater Fan ON");
```

```
---
Temp: 37.5°C
Humidity: 44.5%
---
Temp: 37.5°C
Humidity: 44.5%
---
```

Figure 9 The system is stable

Conclusion

This Smart Poultry Incubator provides an efficient, automated system to ensure optimal incubation conditions. By leveraging **ESP32, DHT22, and LCD displays**, it maintains a precise balance between heating and cooling. With further improvements, the project can be adapted for large-scale incubation setups.

Appendix

The code for the project

```
#include "DHTesp.h"
#include <LiquidCrystal_I2C.h>
#define I2C_ADDR     0x27
#define LCD_COLUMNS  20
#define LCD_LINES    4
#define HEATER_PIN 32
#define FAN_PIN 33

const int DHT_PIN = 15;

DHTesp dhtSensor;

LiquidCrystal_I2C lcd1(0x27, LCD_COLUMNS, LCD_LINES);//for sensor display
LiquidCrystal_I2C lcd2(0x3F, LCD_COLUMNS, LCD_LINES);//for heating and cooling element
display

// Incubator target conditions
float targetTemp = 37.5;      // Target temperature in °C
float tempTolerance = 0.5;    // Acceptable temperature deviation

// (Optional): For humidity control
float targetHumidity = 55.0;      // Target humidity in %
float humidityTolerance = 5.0;    // Allowable range

void setup() {
  Serial.begin(115200);
  dhtSensor.setup(DHT_PIN, DHTesp::DHT22);
  lcd1.init();
  lcd1.backlight();
  lcd2.init();
  lcd2.backlight();
  pinMode(HEATER_PIN, OUTPUT);
  pinMode(FAN_PIN, OUTPUT);
  digitalWrite(HEATER_PIN, LOW);
  digitalWrite(FAN_PIN, LOW);

}
void loop() {

  TempAndHumidity  data = dhtSensor.getTempAndHumidity();
  Serial.println("Temp: " + String(data.temperature, 1) + "°C");
  Serial.println("Humidity: " + String(data.humidity, 1) + "%");
  Serial.println("---");

  lcd1.setCursor(0, 0);
  lcd1.print("  Temp: " + String(data.temperature, 1) + "\xDF"+"C  ");
  lcd1.setCursor(0, 1);
  lcd1.print(" Humidity: " + String(data.humidity, 1) + "% ");
  lcd1.print("Wokwi Online IoT");

   // Temperature control logic
  if (data.temperature < targetTemp - tempTolerance) {
    // Temperature is too low – turn on heater
    digitalWrite(HEATER_PIN, HIGH);
    digitalWrite(FAN_PIN, LOW);
    lcd2.setCursor(0, 0);
    lcd2.print("Heater Fan ON");
  } else if ( data.temperature > targetTemp + tempTolerance) {
    // Temperature is too high – activate cooling fan
    digitalWrite(HEATER_PIN, LOW);
    digitalWrite(FAN_PIN, HIGH);
    lcd2.setCursor(0, 0);
    lcd2.print("Cooling Fan ON");
  } else {
    // Temperature is within the target range – turn off both
    digitalWrite(HEATER_PIN, LOW);
    digitalWrite(FAN_PIN, LOW);
    lcd2.setCursor(0, 0);
    lcd2.print("Temperature Stable");
  }

  delay(1000);
}
```

Reference

I. ***ESP32 Technical Reference Manual*** *- https://www.espressif.com/en/products/socs/esp32/resources*

II. ***DHT22 Sensor Datasheet*** *- https://www.adafruit.com/product/385*

III. ***Wokwi Simulator*** *- https://wokwi.com/*

IV. ***LiquidCrystal_I2C Library*** *- https://github.com/johnrickman/LiquidCrystal_I2C*

V. ***DHTesp Library for ESP32*** *- https://github.com/beegee-tokyo/DHTesp*