CHANAKYA
UNIVERSITY

IOT Project Final Report

[CSY407]

Touch Based Fan Regulator

Submitted To: Prof. Arun Vijaykumar

Submitted By: Aditi Balaji

CU22BCA005A

[In lieu of partial fulfilment

of the course requirements Internet Of Things (CSY407)

BCA Semester 7

School of Engineering]

Introduction

Modern smart-home systems increasingly aim to simplify everyday tasks by replacing traditional mechanical switches with intuitive, touch-based controls. This project focuses on developing a touch-controlled fan speed and power management system using an ESP32 microcontroller. Instead of using physical switches or knobs, the user can control the fan with capacitive touch sensors connected to dedicated GPIO pins. Each sensor is mapped to a specific fan state, enabling seamless interaction through simple touch gestures.
The system also leverages PWM (Pulse Width Modulation) to regulate fan speed electronically, making the fan both efficient and flexible. This project demonstrates how microcontrollers can modernize household appliances with minimal hardware and low-cost components.

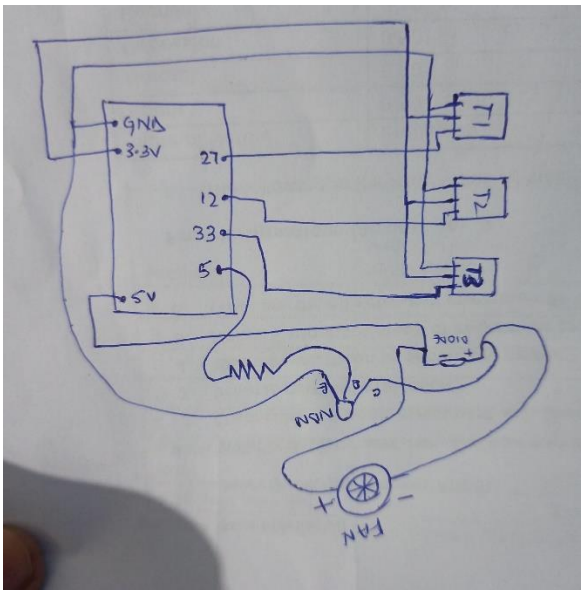
Objective

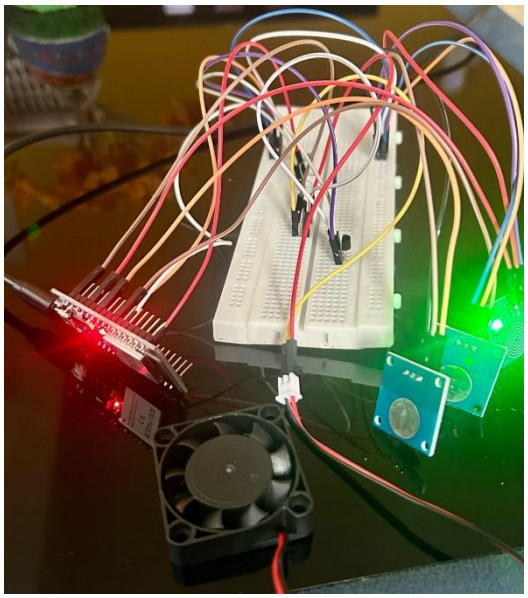
1. To design and implement a touch-based control system for operating a DC fan.
2. To replace mechanical buttons with capacitive touch inputs.
3. To use PWM to vary fan speed across multiple levels.
4. To toggle each fan speed state with a single touch (touch to ON at given speed, touch again to OFF).
5. To develop a compact, reliable user interface suitable for smart-home automation.

Components Used

1. **ESP32 Development Board**
   Used as the main controller for reading capacitive touch sensors and generating PWM signals for fan control.
2. **Capacitive Touch Sensors (TTP223 or ESP32 inbuilt touch pins)**
   Three sensors connected to GPIO 27, 32, and 33 for speed 1, speed 2, and speed 3 operations respectively.
3. **NPN Transistor (BC548)**
   Used as a low-side switch to drive the DC fan since the fan draws more current than the ESP32 can supply.
4. **DC Fan (5V or 12V depending on design)**
   The load controlled using PWM.
5. **External Power Supply**
   For powering the fan independently while interfacing the ground with ESP32.
6. **Connecting Wires, Breadboard, Resistors**
   General prototyping components.

Hardware design:





Logic Flow

1. Initialization

The ESP32 configures three pins as touch inputs and sets up a PWM channel on the fan output pin.

2. Touch Detection

Each touch pin is continuously monitored in the loop.
When a touch is detected:

- The software checks the current state of that speed level (ON or OFF).
- If OFF, the fan turns ON at that specific PWM level.
- If ON, the fan turns OFF.

3. Speed Logic

- If Speed 1 sensor is touched:
    - PWM = Low speed (e.g., duty cycle 150)
- If Speed 2 sensor is touched:
    - PWM = Medium speed (e.g., duty cycle 200)

- If Speed 3 sensor is touched:
    - PWM = High speed (e.g., duty cycle 250)

Each acts as a toggle, independent of the others.

4. Conflict Handling

If a higher or lower speed is selected while another is active:

- The system shuts off the previous speed.
- Then activates the newly requested speed.
  Only one speed remains active at a time.

5. Fan OFF State

If any active speed sensor is touched again, PWM = 0.

Results

1. The system successfully detects touch inputs from each of the three sensors.
2. The fan turns ON and OFF reliably at all three speed levels.
3. The transition between speeds happens smoothly using PWM without mechanical noise.
4. The fan remains OFF until a speed level is selected through a touch gesture.
5. The system demonstrates fast response time and stable performance.

Program

```
#include <Arduino.h>

// Touch sensor pins

const int touchPin1 = 27;

const int touchPin2 = 12;

const int touchPin3 = 33;

// Fan pin

const int fanPin   = 5;
```

```
// Speeds (0 to 255)
const int speed1 = 150;     // Sensor 1
const int speed2 = 200;    // Sensor 2
const int speed3 = 250;    // Sensor 3

// Toggle states
bool state1 = false;
bool state2 = false;
bool state3 = false;
// Debounce timing
unsigned long lastT1 = 0;
unsigned long lastT2 = 0;
unsigned long lastT3 = 0;
const unsigned long debounceTime = 300;

void setup() {
  Serial.begin(115200);
  pinMode(touchPin1, INPUT);
  pinMode(touchPin2, INPUT);
  pinMode(touchPin3, INPUT);
  pinMode(fanPin, OUTPUT);
}

void loop() {
  unsigned long now = millis();
  int s1 = digitalRead(touchPin1);
  int s2 = digitalRead(touchPin2);
```

```
int s3 = digitalRead(touchPin3);

// SENSOR 1
if (s1 == HIGH && now - lastT1 > debounceTime) {
  state1 = !state1;
  lastT1 = now;
  // Disable other states
  state2 = false;
  state3 = false;
  Serial.println("Sensor 1 toggled");
}

// SENSOR 2
if (s2 == HIGH && now - lastT2 > debounceTime) {
  state2 = !state2;
  lastT2 = now;
  state1 = false;
  state3 = false;
  Serial.println("Sensor 2 toggled");
}

// SENSOR 3
if (s3 == HIGH && now - lastT3 > debounceTime) {
  state3 = !state3;
  lastT3 = now;
  state1 = false;
  state2 = false;
```

```
    Serial.println("Sensor 3 toggled");
  }
  // FAN SPEED SELECTION
  int speed = 0;
  if (state1) speed = speed1;
  if (state2) speed = speed2;
  if (state3) speed = speed3;
  analogWrite(fanPin, speed);
  Serial.print("States: ");
  Serial.print(state1); Serial.print(" ");
  Serial.print(state2); Serial.print(" ");
  Serial.print(state3); Serial.print(" | Speed: ");
  Serial.println(speed);
  delay(30);
}
```

Use Cases

1. **Smart Homes**
   Wall-mounted touch panels to replace traditional fan regulators.
2. **Child-safe Fan Controls**
   No mechanical switches that kids can pull or break.
3. **Silent Operation**
   PWM-based speed control, no noisy mechanical regulators.
4. **Energy-efficient Appliances**
   Fine-grained control allows better power management.

Future Improvements

1. Add Wi-Fi / Bluetooth Control
Enable smartphone or voice assistant integration.

2. Temperature-based Automatic Speed Control
Use a DHT11/DHT22 or LM35 sensor to adjust fan speed automatically.

3. OLED Display
Show current speed level and system status.

4. Touch Debouncing & Smoothing
Improve accuracy using software filtering for noisy environments.

5. Overcurrent Protection
Use a MOSFET or driver module for higher-power fans.

6. Multi-device Expansion
Add support for lights or multiple fans using the same touch interface.

Conclusion

The project demonstrates that capacitive touch sensors can effectively replace physical switches for controlling a DC fan. The ESP32 proved capable of handling multiple inputs and generating PWM output for speed adjustment. The hardware setup operated safely and efficiently with minimal components. The resulting system delivers a modern, contactless, and customizable control interface.