

SMART ICE DETECTION AND AUTO DE-ICING WING SYSTEM

S Ramasubramanian¹, Vigneshwaran V², Pravin Singh³, Prankrishna Bhuyan³, Vijaykrishna Venu³, Mohammed Sayeed T R³

¹Associate Professor, Department of Aviation, Vels Institute of Science, Technology & Advanced Studies, Chennai 600117, Tamil Nadu, India

²Program Coordinator, MH Cognition Private Limited – 600117, Tamil Nadu, India

³Student, Department of Aviation, Vels Institute of Science, Technology & Advanced Studies, Chennai 600117, Tamil Nadu, India

Abstract

Aircraft icing is one of the most dangerous environmental hazards affecting flight safety and aerodynamic efficiency. Ice accumulation on aircraft wings disturbs airflow, reduces lift, increases drag, and may lead to loss of control. Conventional de-icing systems often depend on manual activation and continuous power usage, resulting in reduced efficiency. This paper proposes a **Smart Ice Detection and Auto De-Icing Wing System** that integrates temperature sensing, microcontroller-based decision making, and automated heating elements to detect icing conditions and remove ice in real time. The system uses sensors to continuously monitor surface temperature and automatically activates heating elements when icing thresholds are reached. The prototype demonstrates improved response time, reduced pilot workload, and optimized energy consumption through selective heating. This low-cost intelligent system can be extended to aircraft, UAVs, and other cold-environment applications.

Keywords: Aircraft icing, Smart de-icing, Arduino, Sensors, Aviation safety, Automation

I. Introduction

Aircraft wings are carefully designed airfoils that generate lift through smooth airflow. When ice forms on the leading edge of wings, airflow is disrupted, resulting in aerodynamic degradation. Even a small ice layer can significantly reduce lift and increase drag. In severe conditions, icing may cause stalling, unstable flight, or engine performance issues.

Traditional ice protection systems include pneumatic boots, thermal anti-icing systems, and chemical de-icing fluids. However, these systems suffer from several drawbacks:

- Manual operation by pilots
- High energy consumption
- Delayed response
- Uniform heating even when unnecessary

Therefore, an intelligent automated solution is required to detect ice formation early and respond efficiently.

II. Literature Review

Recent research has explored smart icing solutions using advanced materials and artificial intelligence.

- **Graphene-based sensors** have shown excellent sensitivity for ice detection.
- **YOLOv8 vision models** have been used for real-time detection of snow and ice on aircraft surfaces.
- **Deep learning models** achieved over 97% accuracy in wing icing detection.
- **Electro-thermal systems** using Nichrome and Kanthal materials improved heating efficiency.
- **Ultrasonic guided waves** enabled early-stage ice detection at micron levels.

These studies confirm that intelligent sensing and automation can greatly improve aircraft de-icing systems.

III. Proposed Methodology

The proposed system consists of three functional units:

A. Sensing Unit

Temperature and humidity sensors continuously monitor environmental conditions.

B. Control Unit

An Arduino Uno microcontroller processes sensor inputs and compares values with predefined icing thresholds.

C. Actuation Unit

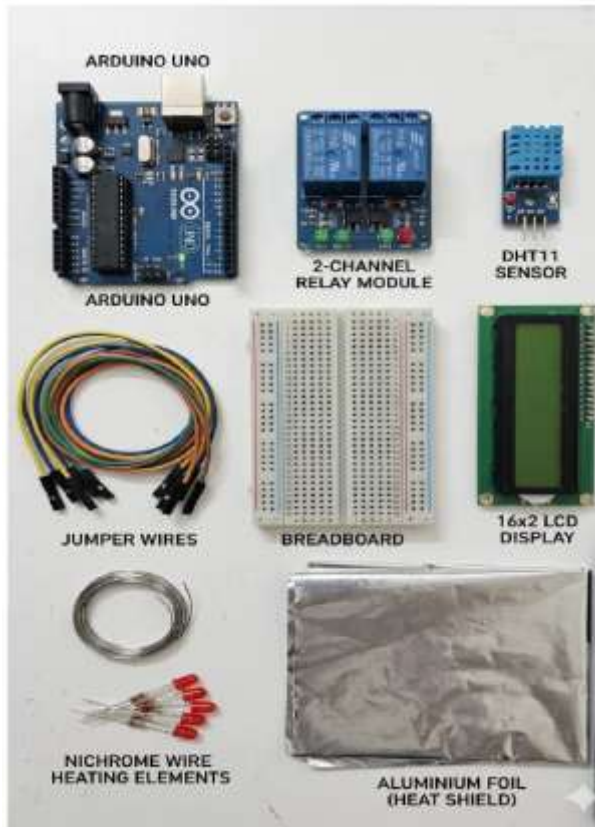
Nichrome wire heating elements installed along the wing leading edge are activated automatically through a relay module.

Working Process

1. Sensor detects low temperature.
2. Microcontroller verifies icing condition.
3. Relay activates heating system.
4. Ice melts from wing surface.
5. Heating turns OFF automatically after safe temperature restoration.

IV. Hardware Components

Component	Function
Arduino Uno	Central controller
DHT11 Sensor	Temperature & humidity sensing
Relay Module	Switches heater ON/OFF
Nichrome Wire	Heating element
LCD Display	Shows system status
Breadboard & Wires	Circuit connection
Battery Supply	Power source



V. Algorithm

```

Start
Initialize Sensor & Relay
Read Temperature
If Temp ≤ Threshold:
    Turn Heater ON
Else:
    Turn Heater OFF
Display Status
Repeat
    
```

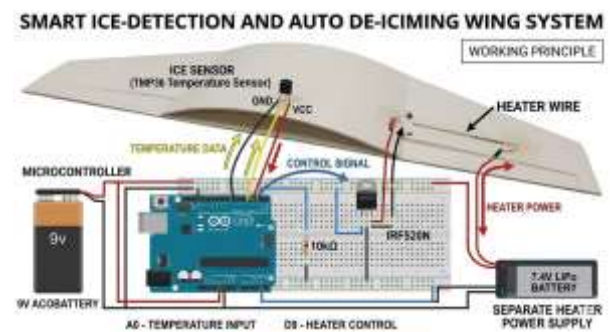
Hysteresis logic is used to avoid frequent switching and improve stability.

VI. Experimental Setup

A prototype wing model was built using foam material covered with aluminum foil. Sensors were mounted near the leading edge, and heating wire was fixed on the surface. Ice cubes were placed near the sensor to simulate icing conditions.

Observations

- Sensor detected temperature drop quickly
- Heater activated automatically
- Ice melted efficiently
- LCD displayed live system status



CODE DONE TO RUN

```

#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);
// Pin definitions
int tempPin = A0;
int relayPin = 7;
int ledPin = 8;
// Temperature control settings
const float DE_ICE_THRESHOLD = 0.0;
// Temperature to turn ON (°C)
const float HYSTERESIS = 2.0; //
Hysteresis to prevent chattering (°C)
const float TEMP_OFFSET = 0.0; //
Calibration offset for LM35
// Timing settings
const unsigned long READ_DELAY =
2000; // Read temperature every 2
seconds
    
```

```
const unsigned long
DEBOUNCE_DELAY = 500; // Debounce
time for sensor
// Safety settings
const float MAX_SAFE_TEMP = 50.0; //
Maximum safe temperature
const float MIN_SAFE_TEMP = -20.0; //
Minimum safe temperature
// System state variables
bool relayState = false;
bool ledState = false;
unsigned long lastReadTime = 0;

unsigned long lastDisplayUpdate = 0;

int sensorErrorCount = 0;
void setup() {
// Initialize LCD
lcd.init();
lcd.backlight();
lcd.clear();
// Initialize pins
pinMode(relayPin, OUTPUT);
pinMode(ledPin, OUTPUT);
// Start serial for debugging
Serial.begin(9600);
Serial.println("De-icing Control System
Started");
// Initial display
lcd.setCursor(0, 0);
lcd.print("System Ready");
lcd.setCursor(0, 1);
lcd.print("Temp: --. C");
delay(2000);
lcd.clear();
}
void loop() {
unsigned long currentMillis = millis();
// Read temperature at regular intervals
if (currentMillis - lastReadTime >=
READ_DELAY) {
lastReadTime = currentMillis;
// Read and process temperature
float rawTemp = readTemperature();
```

```
float          filteredTemp          =
filterTemperature(rawTemp);

float calibratedTemp = filteredTemp +
TEMP_OFFSET;
// Validate sensor reading
if (isValidTemperature(calibratedTemp)) {
sensorErrorCount = 0;
// Control de-icing system with hysteresis
controlDeIcing(calibratedTemp);
// Update display
updateDisplay(calibratedTemp,
relayState);
// Log to serial
logToSerial(calibratedTemp, relayState);
} else {
// Handle sensor error
sensorErrorCount++;
handleSensorError();
}
}
// Periodic display refresh (prevents
flickering)
if (currentMillis - lastDisplayUpdate >=
500) {
lastDisplayUpdate = currentMillis;
if (relayState) {
blinkLED();
}
}
}
float readTemperature() {
// Take multiple readings for better
accuracy
const int numSamples = 10;
long sum = 0;

for (int i = 0; i < numSamples; i++) {

sum += analogRead(tempPin);
delay(10);
}
int avgValue = sum / numSamples;
```

```
// Convert to temperature (LM35: 10mV
per °C)
// Formula: (Analog reading * Reference
voltage * 100) / 1024
float temperature = (avgValue * 5.0 *
100.0) / 1024.0;
return temperature;
}

float filterTemperature(float rawTemp) {
// Simple moving average filter
static float tempBuffer[5] = {0};
static int bufferIndex = 0;
static bool bufferFull = false;
tempBuffer[bufferIndex] = rawTemp;
bufferIndex = (bufferIndex + 1) % 5;
if (bufferIndex == 0) {
bufferFull = true;
}
int numValues = bufferFull ? 5 :
bufferIndex;
float sum = 0;
for (int i = 0; i < numValues; i++) {
sum += tempBuffer[i];
}
return sum / numValues;
}

bool isValidTemperature(float temp) {

if (temp < MIN_SAFE_TEMP || temp >
MAX_SAFE_TEMP) {

Serial.print("Warning: Invalid temperature
reading: ");
Serial.println(temp);
return false;
}

return true;
}

void controlDeIcing(float currentTemp) {
static bool lastRelayState = false;
static unsigned long lastSwitchTime = 0;
// Check if enough time has passed since
last switch (prevents rapid switching)
```

```
if (millis() - lastSwitchTime <
DEBOUNCE_DELAY) {
return;
}

bool newRelayState = relayState;
// Hysteresis control logic
if (!relayState && currentTemp <=
DE_ICE_THRESHOLD) {
// Turn ON when temperature drops below
threshold
newRelayState = true;
}
else if (relayState && currentTemp >
(DE_ICE_THRESHOLD +
HYSTERESIS)) {
// Turn OFF when temperature rises above
threshold + hysteresis
newRelayState = false;
}
// Apply new state if changed
if (newRelayState != relayState) {
lastSwitchTime = millis();
relayState = newRelayState;

ledState = relayState;

digitalWrite(relayPin, relayState ? HIGH :
LOW);
digitalWrite(ledPin, ledState ? HIGH :
LOW);
// Log state change
Serial.print("System state changed: ");
Serial.println(relayState ? "DE-ICING
ON" : "SYSTEM OFF");
}
}

void updateDisplay(float temp, bool
systemActive) {
lcd.setCursor(0, 0);
lcd.print("Temp: ");
// Format temperature with 1 decimal place
char tempStr[8];
dtostrf(temp, 5, 1, tempStr);
lcd.print(tempStr);
```

```
lcd.print(" C "); // Extra spaces to clear old
characters
lcd.setCursor(0, 1);
if (systemActive) {
lcd.print("De-Icing ON ");
// Show running time on second line if
space allows
lcd.setCursor(12, 1);
static unsigned long startTime = millis();
unsigned long runningSeconds = (millis() -
startTime) / 1000;
lcd.print(runningSeconds % 60);
} else {
lcd.print("System OFF ");
}

}

void logToSerial(float temp, bool
systemActive) {

Serial.print("Temperature: ");

Serial.print(temp, 1);
Serial.print(" C | System: ");
Serial.println(systemActive ? "ACTIVE" :
"OFF");
}

void handleSensorError() {
if (sensorErrorCount >= 3) {
// Critical sensor error - shut down system
for safety
digitalWrite(relayPin, LOW);
digitalWrite(ledPin, LOW);
relayState = false;
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("SENSOR ERROR!");
lcd.setCursor(0, 1);
lcd.print("System Halted ");
Serial.println("CRITICAL ERROR:
Sensor failure! System halted.");
// Blink error pattern
for (int i = 0; i < 10; i++) {
```

```
digitalWrite(ledPin, HIGH);
delay(100);
digitalWrite(ledPin, LOW);
delay(100);
}
} else {
// Temporary error - show warning
lcd.setCursor(0, 1);
lcd.print("Sensor unstable");
Serial.println("Warning: Unstable sensor
reading");
}

}

void blinkLED() {
// Blink LED when system is active for
better visibility
static unsigned long lastBlink = 0;
static bool blinkState = false;
if (millis() - lastBlink >= 1000) {
lastBlink = millis();
blinkState = !blinkState;
digitalWrite(ledPin, blinkState ? HIGH :
LOW);
}

}
```

VII. Results and Discussion

The prototype successfully demonstrated automatic ice detection and removal. Major benefits observed:

- Fast response time
- Reduced manual intervention
- Low-cost implementation
- Energy-efficient selective heating
- Reliable repeated operation

Compared with conventional systems, the proposed model provides better automation and lower power consumption.

VIII. Advantages

- Improves aircraft safety
- Reduces pilot workload
- Prevents aerodynamic loss
- Lower maintenance cost
- Scalable for UAVs and commercial aircraft
- Supports future AI integration

IX. Future Scope

The system can be enhanced by integrating:

- AI-based predictive icing analysis
- IoT monitoring systems
- Wireless cockpit alerts
- Multiple distributed sensors
- Lightweight graphene heaters
- UAV autonomous deployment

X. Conclusion

This paper presented a Smart Ice Detection and Auto De-Icing Wing System for improving aviation safety and efficiency. By combining real-time sensing, embedded control, and automatic heating, the system effectively removes ice with minimum power consumption. Experimental results validate the feasibility of the concept. The proposed technology offers a promising solution for next-generation intelligent aircraft systems.

References

1. Project Report: Smart Ice Detection and Auto De-Icing Wing System.
2. Pathan et al., AI-Enhanced Ice Detection Using Graphene Sensors, 2024.
3. Wang et al., Wing Icing Detection using Deep Learning, 2025.
4. Kumar et al., Thermal-Electric Analysis of Aircraft De-Icing Systems, 2025.
5. Bragg, M. B., Broeren, A. P., & Blumenthal, L. A., "Iced-Airfoil Aerodynamics," *Progress in Aerospace Sciences*, vol. 41, no. 5, pp. 323–362, 2005.
6. Gent, R. W., Dart, N. P., & Cansdale, J. T., "Aircraft Icing," *Philosophical Transactions of the Royal Society A*, vol. 358, no. 1776, pp. 2873–2911, 2000.
7. Thomas, S. K., Cassoni, R. P., & MacArthur, C. D., "Aircraft Anti-Icing and De-Icing Techniques and Modeling," *Journal of Aircraft*, vol. 33, no. 5, pp. 841–854, 1996.
8. Lynch, F. T., & Khodadoust, A., "Effects of Ice Accretions on Aircraft Aerodynamics," NASA Technical Report, NASA/CR-2001-210031, 2001.
9. Cao, Y., Tan, W., & Wu, Z., "Aircraft Icing: An Ongoing Threat to Aviation Safety," *Aerospace Science and Technology*, vol. 75, pp. 353–385, 2018.
10. Hansman, R. J., & Turnock, S. R., "Investigation of Surface Roughness Effects Caused by Ice Accretion on Airfoils," *Journal of Aircraft*, vol. 26, no. 12, pp. 1069–1075, 1989.
11. Wright, W. B., "User's Manual for LEWICE Ice Accretion Prediction Program," NASA CR-2002-211793, 2002.
12. Vargas, M., & Reshotko, E., "Heat Transfer Characteristics in Electro-Thermal Wing Anti-Icing Systems," *Journal of Thermophysics and Heat Transfer*, vol. 22, no. 4, pp. 689–697, 2008.

13. Parent, O., & Ilinca, A., “Anti-Icing and De-Icing Techniques for Wind Turbines: Critical Review,” *Cold Regions Science and Technology*, vol. 65, no. 1, pp. 88–96, 2011.
14. Meuler, A. J., Smith, J. D., Varanasi, K. K., Mabry, J. M., & Cohen, R. E., “Relationships Between Water Wettability and Ice Adhesion,” *ACS Applied Materials & Interfaces*, vol. 2, no. 11, pp. 3100–3110, 2010.