Development of Heateye: A Smart Application an Arduino-Based Heat Detector Device for Household Appliances

Jenghis Judah L. Alcantara[1], Jumaine P. Amoncio[2], Noemi M. Enriquez[3], John Christian E. Gasmido[4], Jenny Rose B. Mecasio[5], Engr. Jose C. Felipe Jr.[6]

Department of Computer Engineering, Eulogio "Amang" Rodriguez Institute of Science and Technology, Manila, Philippines

DOI: https://doi.org/10.51244/IJRSI.2026.1306000254

Received: 14 June 2026; Accepted: 20 June 2026; Published: 03 July 2026

ABSTRACT

In the Philippines, household appliances are often used for extended periods, which may lead to overheating, electrical hazards, and, eventually, fire-related incidents. Existing temperature monitoring systems typically offer only basic alert mechanisms, limiting their effectiveness in preventing accidents and ensuring appliance safety. This study focuses on the development of HeatEye, an Arduino-based heat detector integrated with a mobile application for real-time temperature monitoring and remote appliance control. The system uses an Arduino Uno, an LM35 temperature sensor, an HC-05 Bluetooth module, and a relay module to detect abnormal heat levels and automatically respond when the temperature exceeds the programmed threshold. The mobile application provides live temperature updates, overheating notifications, ON/OFF remote operation, timer automation, and historical temperature logs with graphical visualization.

INTRODUCTION

In the Philippines, the continuous advancement and widespread use of electrical appliances have greatly improved the quality of daily living. However, it has also raised safety concerns, particularly regarding overheating and equipment malfunctions. In many households, appliances are frequently operated for extended periods without proper monitoring, leading to excessive heat buildup that often goes unnoticed. This condition poses a significant risk, as overheating contributes to electrical hazards and fire-related incidents. In a setting where electrical devices are widely used, the need for a more efficient and responsive safety mechanism has become increasingly evident. Current temperature detection systems offer some level of protection, but many rely on basic alert features that need user input. These systems often alert users only after a critical point, potentially delaying the necessary actions and reducing their effectiveness in preventing danger. Additionally, the absence of real-time monitoring and accessibility limits users' ability to consistently check appliance conditions, especially when they are not physically present. These constraints underscore the necessity for a more comprehensive approach that integrates detection, monitoring, and immediate response.

In response to these concerns, this study focuses on developing an enhanced heat detection and monitoring system that integrates automation and application-based control. The proposed system enables continuous temperature monitoring and provides real-time feedback through a connected mobile application, allowing users to remotely monitor their appliances and receive timely notifications when abnormal heat levels are detected. By integrating monitoring and accessibility into a single platform, the system supports a more proactive approach to safety, reducing reliance on delayed responses and manual intervention.

Through this approach, the study aims to improve the overall management of overheating risks in household appliances by promoting early detection and immediate response. The system is designed not only to enhance user awareness but also to provide a more reliable and efficient way to prevent potential hazards associated

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue VI June 2026

with excessive heat. While the scope of this research is limited to the development and evaluation of the proposed system, it demonstrates the potential of combining real-time monitoring, automation, and user-centered application design in addressing a common yet critical safety concern. Ultimately, the study seeks to help reduce appliance-related risks and support the creation of safer household environments.

LITERATURE REVIEW

This section presents the related literature and studies that serve as the foundation for the development of the HeatEye system. It discusses existing technologies and research related to temperature monitoring, fire prevention, home automation, and mobile-based monitoring systems.

Household electrical fires remain a serious safety concern. According to the National Fire Protection Association (2023), overheating appliances, overloaded circuits, and electrical malfunctions are among the leading causes of residential fires. These incidents highlight the need for systems that can detect overheating early and help prevent fire-related accidents.

Arduino-based monitoring systems have been widely used for household safety applications. Banzi and Shiloh (2022) explained that Arduino is a low-cost and flexible platform suitable for monitoring and automation projects. A local study by Casinillo et al. (2024) developed an Arduino-based high-heat detector that could detect excessive temperatures and automatically turn off appliances using a relay module. Similarly, the ODET system developed by De La Salle University (2022) focused on detecting overheating conditions in household appliances.

Although these systems successfully detected high temperatures and protected appliances, they mainly relied on local alarms and automatic shutdown functions. Users needed to be near the device to monitor its status. The proposed HeatEye system builds on these features by providing real-time temperature monitoring via a mobile application, allowing users to view temperature readings, receive overheating alerts, control appliances remotely, and access historical temperature records.

Several studies have also explored wireless home automation systems. Kumar and Kumar (2018), Bhoyar and Sahare (2019), and Ghosh and Das (2020) showed that Bluetooth technology can be effectively used to control and monitor appliances through mobile applications. Similar to these studies, HeatEye uses the HC-05 Bluetooth module to establish communication between the Arduino-based prototype and the mobile application.

Temperature monitoring systems commonly use sensors to accurately detect heat levels. The LM35 temperature sensor is widely used because of its reliability and ease of integration with Arduino systems (Texas Instruments, n.d.). Studies by Kumar and Kumar (2020) and Prasad and Rao (2021) demonstrated that temperature sensors are effective for real-time monitoring and safety applications. HeatEye utilizes the LM35 sensor to continuously monitor appliance temperature and trigger safety actions when necessary.

Research on smart monitoring systems also emphasizes the importance of mobile applications. Juwariyah et al. (2021) found that mobile-based monitoring systems improve user awareness by providing notifications, data logs, and real-time updates. Unlike many existing monitoring solutions that only provide alarms or automatic shutdown, HeatEye combines several features in a single system, including real-time temperature monitoring, overheating notifications, appliance ON/OFF control, timer automation, graphical data logging, and emergency hotline access.

Based on the reviewed literature, existing systems have proven effective in detecting overheating and improving household safety. However, many of them focus only on monitoring or automation. The HeatEye system integrates monitoring, control, notification, and data logging features into a single mobile-based platform, providing a more convenient and comprehensive solution for preventing appliance-related overheating incidents.

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue VI June 2026

Arduino-Based Overheat Detectors for Fire Prevention

The development of Arduino-based overheat detection systems as a fire prevention strategy has been documented in several previous studies. One such study conducted by De La Salle University researchers (2022) introduced ODET: An Arduino-based Overheat Detector for Household Appliances, addressing the increasing incidence of electrical fires in residential settings. The study highlighted that overheated appliances are a significant contributor to fire-related incidents in the Philippines, underscoring the need for low-cost preventive measures. The prototype used an Arduino Uno, a LM35 temperature sensor, a relay module, and a buzzer to detect abnormal heat levels and automatically disconnect power to the appliance when unsafe temperatures were reached (De La Salle University, 2022). The ODET system demonstrated that Arduino-based configurations can effectively monitor temperature and respond to overheating conditions with reliable performance. However, its functionality was limited to local alerts via buzzers and onboard indicators, which restricted its effectiveness when users were not physically near the device. This limitation reduced its practicality in real-world household environments where remote monitoring is often necessary. Similarly, related studies on Arduino-based temperature control systems found that such prototypes typically achieved acceptable accuracy in detecting temperature changes but still relied heavily on manual supervision. Some versions of these systems reported average error rates below 2%, indicating strong measurement reliability of the LM35 sensor when properly calibrated (Juwariyah et al., 2021). However, variations in sensor placement and threshold configuration significantly affected detection consistency.

Furthermore, Juwariyah et al. (2021) emphasized the importance of integrating Internet of Things (IoT) capabilities into fire detection systems. Their study demonstrated that combining Arduino-based hardware with mobile applications significantly improves real-time monitoring and emergency response by enabling notifications and data logging. In comparison, while ODET and similar Arduino-based systems effectively demonstrate automated overheating protection, they lack advanced user interaction features such as mobile control, real-time remote monitoring, and data logging visualization. These limitations suggest that traditional Arduino-based detectors are primarily reactive rather than proactive. The HeatEye system addresses this gap by integrating mobile app functionality, Bluetooth communication, and real-time data visualization, creating a more comprehensive, user-centered approach to appliance safety and fire prevention.

METHODOLOGY

This research leveraged the Arduino-based heat detector prototype developed by the STEM students of the Liceo de Cagayan University. The researchers enhanced the existing prototype by integrating a compatible mobile software application. To systematically design and construct the integrated software, the study adopted the modified research and development (R&D) method designed by Borg and Gall. The testing focused on evaluating the software application's functionality and accuracy, as well as the hardware's response time when executing the application's commands.

Hardware Architecture

The study's main focus was on integrating a mobile software application that has control and monitoring capabilities (HeatEYE) with the Arduino-based temperature-controlled prototype device intended for household appliances. As a proof of concept for the project, the researchers used a specific device for testing and development: a Clip Fan.

The researchers constructed a diagram of the enhanced prototype to illustrate the wiring connections between its components. As shown in Figure 1, the improved prototype featured the main components of an Arduino Uno, an LM35 temperature sensor, an HC-05 Bluetooth Module, and a 1-Channel (SRD-05VDC-SL-C) Relay Module. Other components used in the prototype included a DC fan motor and a breadboard with jumper wires for circuit connections.

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue VI June 2026

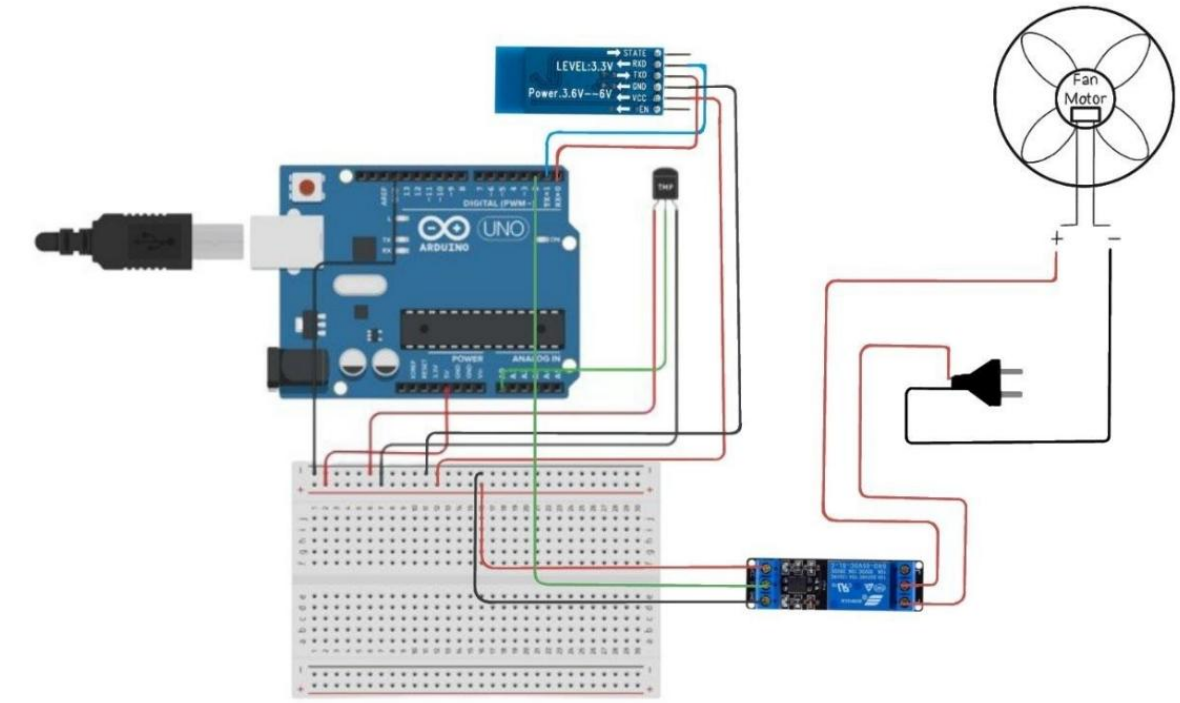

Figure 1. Diagram of the enhanced Arduino-based heat detector temperature control prototype

Hardware Components

- **Arduino UNO**: The Arduino served as the system's main controller. It was responsible for processing temperature data from the LM35 temperature sensor and controlling the appliance's operational state via the relay module.
- **LM35 Temperature Sensor**: The LM35 temperature sensor was used to detect and measure the appliance's temperature. The sensor sent the measured temperature as an analog signal to the Arduino for data processing and monitoring.
- **HC-05 Bluetooth Module:** HC-05 is a low-cost Bluetooth module with a communication range of up to 10 meters in an open space. This module was utilized in the prototype to establish a wireless connection between the prototype device and the HeatEye mobile application. Through this module, users can remotely monitor the appliance's temperature readings and status, and control its operation via a smartphone.
- **SRD-05VDC-SL-C Relay Module**: The SRD-05VDC-SL-C Relay Module was used in the prototype as an operational electrical switch, allowing the Arduino Uno to control the power supplied to the appliances. In the conducted research, the relay module used the Arduino's low-voltage output signal to safely control the fan's higher power requirements, enabling efficient switching between the Arduino and the fan.

Diagram Explanation

Figure 1 illustrates the wiring connections and the flow of the enhanced Arduino-based temperature-controlled prototype. The Arduino is connected to the LM35 temperature sensor, the HC-05 Bluetooth module, and the relay module to establish communication and control within the system.

The LM35 temperature sensor continuously measures temperature and outputs an analog signal to the Arduino Uno via the analog input pin. The Arduino Uno processes the transmitted signals and compares temperature readings with the programmed threshold value. If the detected temperature exceeds the threshold, the Arduino deactivates the relay module, cutting off the fan's power supply and shutting the fan down. On the other hand, if the temperature falls below the threshold value, the Arduino will activate the relay module, allowing current to flow to the fan motor, which will automatically switch the fan ON to provide cooling. The fan motor is connected to an external power supply through the relay terminals, ensuring a stable operation within the hardware system.

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue VI June 2026

Arduino Coding

The Arduino source code was developed in the Arduino IDE using C++. As shown in Figure 2, the Arduino was programmed to read the temperature data from the LM35 sensor, control the relay temperature to switch the fan operational states based on the predefined thresholds (warm at 50°C, overheat at 65°C), and implement a safety procedure that keeps the fan off for 5 seconds after the overheating events to ensure a safety cooldown period for the appliance. The Arduino was also programmed to handle Bluetooth remote commands (ON/OFF/AUTO) from the developed mobile application, ensuring that it overrides automatic control only when safe and transmits temperature data and appliance status to the application every second.

```
const int sensor = A0;
#define RELAY 2

// Temperature thresholds in °C
const int WARM_THRESHOLD = 50;
const int OVERHEAT_THRESHOLD = 65;
// --- NEW: 5-minute delay (300000 ms) after overheating ---
const unsigned long OVERHEAT_DELAY_MS = 5000;   // 5 minutes
bool inCooldown = false;        // TRUE = waiting for delay to finish
unsigned long cooldownStartTime = 0;
bool wasOverheated = false;     // remembers that an overheat just ended
// ------------------------------------------------------------

float tempc;
int fanStatus = 0;     // 0=NORMAL, 1=OVERHEAT, 2=WARM
bool manualMode = false;
String command = "";

void setup() {
  pinMode(RELAY, OUTPUT);
  Serial.begin(9600);

  // Start with fan ON (AUTO default ON)
  digitalWrite(RELAY, LOW);
}

void loop() {
  // ----------------------------
  // READ BLUETOOTH COMMAND
  // ----------------------------
  if (Serial.available()) {
    command = Serial.readStringUntil('\n');
    command.trim();

    if (command == "ON") {
      manualMode = true;
      // Do NOT change relay yet – will be set later based on temperature safety
    }
    else if (command == "OFF") {
      manualMode = true;
      // Do NOT change relay yet – will be set later based on temperature safety
    }
    else if (command == "AUTO") {
      manualMode = false;
    }
  }
    if (!inCooldown)
      digitalWrite(RELAY, LOW);  // Fan ON to cool
    else
      digitalWrite(RELAY, HIGH); // Stay OFF during cooldown
    }
    else {
      fanStatus = 0;             // NORMAL
      // --- NEW: only turn ON if not in cooldown ---
      if (!inCooldown)
        digitalWrite(RELAY, LOW);  // Fan ON (normal cooling)
      else
        digitalWrite(RELAY, HIGH); // Stay OFF during cooldown
    }
  }
  else {
    // MANUAL mode (and temperature below overheat threshold)
    // Honour the last user command (ON/OFF) but only if safe.
    if (command == "ON") {
      // --- NEW: block manual ON during cooldown ---
      if (!inCooldown)
        digitalWrite(RELAY, LOW);   // Fan ON
      else
        digitalWrite(RELAY, HIGH);  // Force OFF during cooldown
    }
    else if (command == "OFF") {
      digitalWrite(RELAY, HIGH);  // Fan OFF
    }
    // If no command received yet, keep previous state (do nothing)
    // Override fanStatus to reflect actual temperature (not manual command)
    if (tempc >= OVERHEAT_THRESHOLD) {
      fanStatus = 1;
    } else if (tempc >= WARM_THRESHOLD) {
      fanStatus = 2;
    } else {
      fanStatus = 0;
    }
  }
  // ----------------------------
  // SEND DATA TO ANDROID
  // ----------------------------
  Serial.print(tempc);
  Serial.print(",");
  Serial.println(fanStatus);

  delay(1000);
}
  }
  // ----------------------------
  // READ TEMPERATURE (LM35)
  // ----------------------------
  int rawvoltage = analogRead(sensor);
  float millivolts = (rawvoltage / 1023.0) * 5000.0;
  tempc = millivolts / 10.0;
  // ----------------------------
  // COOLDOWN STATE MANAGEMENT (NEW)
  // ----------------------------
  if (tempc >= OVERHEAT_THRESHOLD) {
    // Currently overheating
    wasOverheated = true;        // mark that we had an overheat
    inCooldown = false;          // no cooldown while still overheated
  } else {
    // Temperature safe
    if (wasOverheated) {
      // Just transitioned from overheat → safe: start the 5-minute delay
      inCooldown = true;
      cooldownStartTime = millis();
      wasOverheated = false;
      // Fan will be forced OFF by the cooldown flag (see below)
    }
    // If already in cooldown, check if the delay has finished
    if (inCooldown && (millis() - cooldownStartTime >= OVERHEAT_DELAY_MS)) {
      inCooldown = false;        // cooldown over – fan may turn on again
    }
  }
  // ----------------------------
  // RELAY CONTROL WITH OVERHEAT SAFETY (ALWAYS ENFORCED)
  // ----------------------------

  // SAFETY FIRST: if temperature exceeds OVERHEAT_THRESHOLD, force relay OFF
  if (tempc >= OVERHEAT_THRESHOLD) {
    digitalWrite(RELAY, HIGH);   // Fan OFF (safety cut)
    fanStatus = 1;               // Report OVERHEAT
  }
  else if (!manualMode) {
    // AUTO mode (and temperature below overheat threshold)
    if (tempc >= WARM_THRESHOLD) {
      fanStatus = 2;             // WARM
      // --- NEW: only turn ON if not in cooldown ---
      if (!inCooldown)
        digitalWrite(RELAY, LOW);  // Fan ON to cool
      else
```

Figure 2. Arduino Code

Software Procedural Design

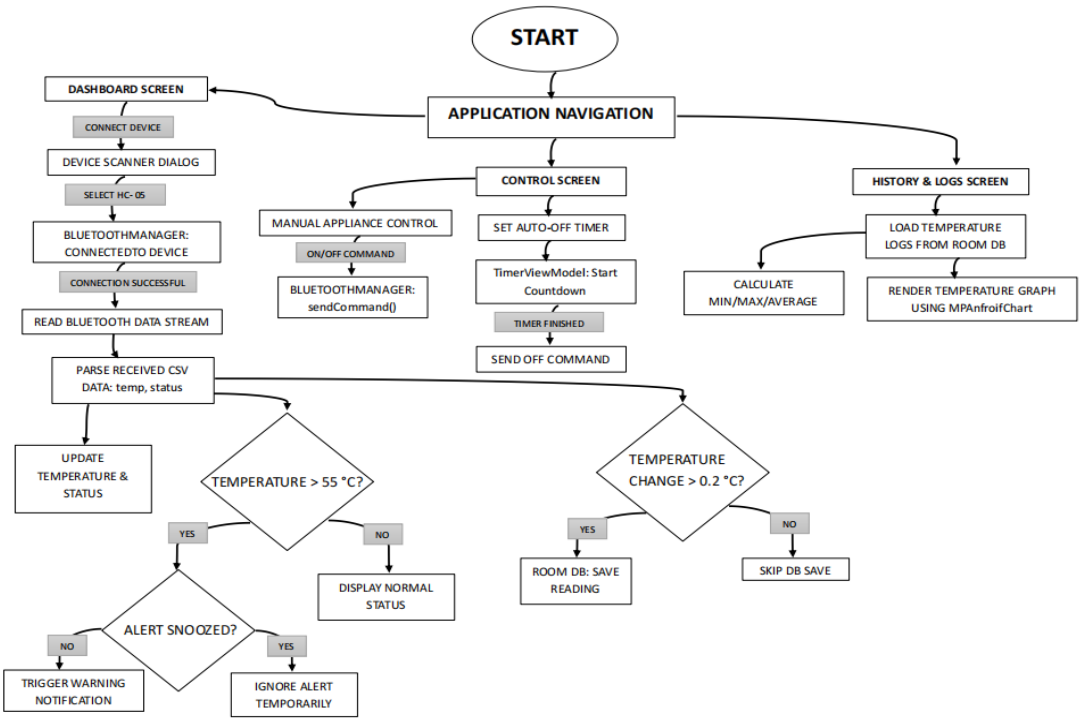

Figure 3. Flowchart of the HeatEye mobile application

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue VI June 2026

Figure 3 illustrates the logical flow of the HeatEye mobile application. The application begins on the Dashboard Screen, where the user's device connects to the prototype via the HC-05 Bluetooth module. The BluetoothManager manages device connectivity, and once a connection is established, the application continuously receives temperature data from the LM35 sensor and updates the temperature display interface in real time, enabling users to monitor the temperature and status of the appliances. The application also includes a Control Screen, where users can remotely control the appliance's operational states and configure an automatic shutdown timer. The automatic shutdown timer allows the user to select a countdown of 15, 30, or 60 minutes, or a custom countdown. The TimerViewModel automatically sends an OFF command to the Arduino when the timer finishes, triggering the appliance's automatic switch-off. In addition, the application stores all temperature readings locally using a Room Database. The collected data is displayed on the History & Logs screen as a visualized, timestamped line graph, historical logs of all recorded temperature readings, and the minimum, maximum, and average recorded temperatures.

Navigation between the Dashboard, Control, and History & Logs screens is managed through the application's bottom navigation interface or via AppNavigation.

Software Features

- **Bluetooth Connectivity**: The HeatEYE mobile application establishes a wireless serial communication with the Arduino-based heat detector prototype using an HC-05 Bluetooth module. This enables real-time data transmission between the hardware and the mobile application, as well as remote command execution without requiring physical interaction.
- **ON/OFF Remote Operation**: The application allows users to remotely control the connected appliance through an ON/OFF control interface. Commands are transmitted from the mobile application to the Arduino via the HC-05 Bluetooth module. Upon receiving the command, the Arduino activates or deactivates the appliance using an SRD-05VDC-SL-C relay module.
- **Overheating Notification & Alert**: When the measured temperature reaches or exceeds 65 °C, the application automatically triggers an overheating alert. This includes a push notification and an audible alarm to warn the user. A persistent background service, implemented through a BluetoothForegroundService, ensures continuous monitoring and alert delivery even when the application is running in the background.
- **Real-time Temperature Monitor**: The application continuously displays live temperature data collected from the LM35 sensor and transmitted via Bluetooth. The system classifies temperature levels into three categories: Normal (≤49 °C), Warm (50–64 °C), and Overheating (≥65 °C). Each category is represented using a color-coded interface: green for Normal, yellow for Warm, and red for Overheating. In the Overheating state, the system also activates alerts and notifications to enhance user awareness and safety.
- **Temperature Data Graphing and Logs**: All temperature readings are stored locally using a Room Database. These recorded values are used to generate time-stamped line graphs and historical logs within the application. This feature allows users to monitor temperature trends and review past overheating events for analysis.
- **Timer Automation Control**: The application includes a timer-based automation feature that allows users to set a countdown for automatic system deactivation. Users may select preset durations of 15, 30, or 60 minutes, or define a custom duration. The TimerViewModel handles the timer logic and executes the corresponding OFF command once the timer expires.
- **Emergency Hotline**: The application provides a built-in emergency hotline directory. When selected, it redirects the user to the device's dialer application with a pre-configured emergency number for quick access during critical situations.

Testing Method

1. **Functionality Test:** The functionality of the HeatEYE system was evaluated by testing each core feature ten (10) times. The number of successful executions was recorded for each feature. A feature was considered functional if it successfully performed its intended operation in at least nine (9) out of ten (10)

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue VI June 2026

trials. Otherwise, the system was reviewed for possible errors in hardware integration or software implementation.

2. **Accuracy Test:** The accuracy of the HeatEYE prototype was evaluated by comparing temperature readings from the LM35 temperature sensor with those obtained from a calibrated digital thermometer, which served as the reference instrument. Prior to the experiment, the digital thermometer was verified based on the manufacturer's specifications to ensure measurement reliability and consistency.

During testing, both the LM35 sensor and the digital thermometer probe were immersed in the same water sample to ensure identical temperature conditions. This setup allowed for a direct and controlled comparison between the two measuring devices, minimizing environmental variations that could affect the results.

As illustrated in Figure 11, the experimental setup consisted of the HeatEYE prototype equipped with the LM35 sensor and a digital thermometer used as the reference instrument. Three (3) trials were conducted with water samples at different temperatures to evaluate the prototype's accuracy under varying thermal conditions.

The temperature readings from both instruments were recorded for each trial. The difference between the digital thermometer readings and the HeatEYE prototype readings was computed, and the percent error was determined to quantify the system's accuracy. This procedure ensured that the prototype's performance was systematically assessed under controlled conditions.

3. **Response Time Test:** The response time of the HeatEYE prototype was evaluated by issuing ON and OFF commands through the mobile application and observing the corresponding response of the connected appliance. Multiple trials were conducted to assess the consistency and reliability of communication between the HeatEYE application and the Arduino-based prototype.

The response time was measured as the interval between pressing a control command in the mobile application and the actual activation or deactivation of the relay-controlled appliance (fan). Observations were recorded throughout all trials to determine system responsiveness.

The results showed that the prototype consistently executed commands within approximately 1–2 seconds after being issued from the mobile application. No communication failures or incorrect command executions were observed during testing.

These findings indicate that the HeatEYE system provides fast, reliable, real-time control of connected appliances. The minimal delay between command transmission and execution demonstrates effective Bluetooth communication and hardware responsiveness, making the system suitable for practical monitoring and control applications.

RESULTS & DISCUSSION

Hardware Setup and Performance

The development of the enhanced prototype involved analyzing and selecting the essential components from the original Arduino-based heat detector. The researchers made several improvements to the original prototype to enhance the hardware's efficiency and to incorporate the proposed HeatEye mobile application software into the prototype device. Key improvements included removing several hardware components from the original prototype, such as the push buttons, buzzer, LEDs, resistors, and LCD. The researchers made this decision after identifying that the functions of the following components can be effectively executed within the mobile application. Another improvement made to the prototype was the integration of the HC-05 Bluetooth Module. These are conducted to ensure a wireless connection between the mobile application and the Arduino-based heat detector prototype.

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue VI June 2026

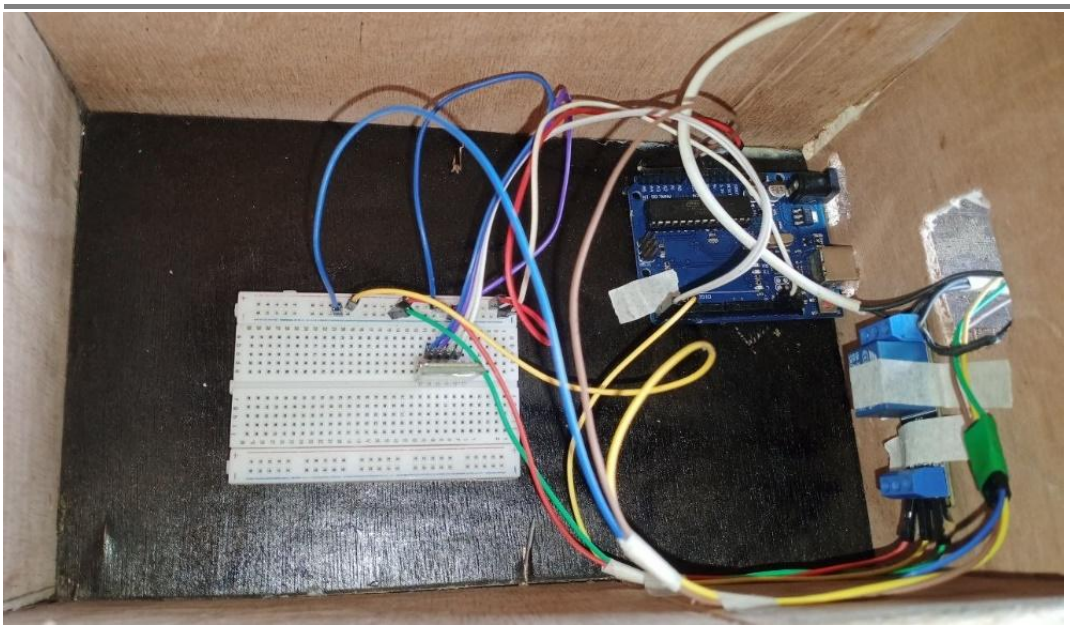

Figure 4. Arduino-Based Heat Detector Prototype Design

As shown in Figure 4, the enhanced prototype now consisted of four (4) main components: the Arduino Uno, which served as the prototype's main controller, an LM35 sensor for the temperature detection, an HC-05 Bluetooth module for the device's wireless communication, and a relay module for the appliance operational switching. Based on the performance testing conducted, as shown in Figure 5, it was confirmed that this enhanced setup was operating as planned. Examples are the successful execution of the automatic overheating shutdown after detecting that the temperature exceeded the programmed threshold, and the effective execution of remote commands from the HeatEYE application by successfully actuating the relay.

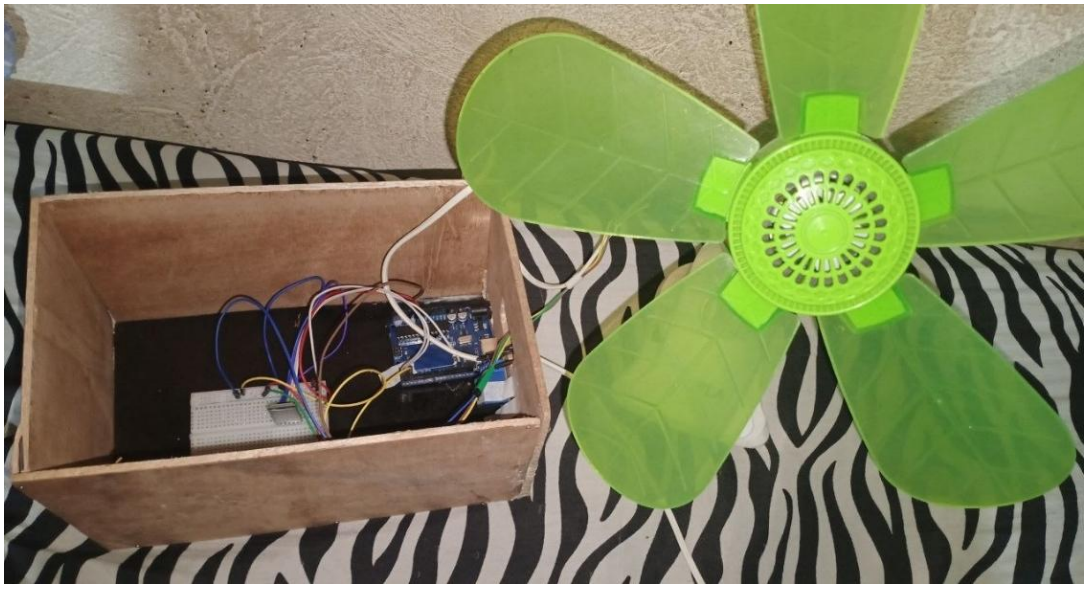

Figure 5. Prototype testing

Design and Development of HeatEYE Application UI & Logic

The HeatEye mobile application was developed in Android Studio using Kotlin as the primary programming language for the application's user interface and functionality.

The software application was programmed to continuously display LM35 temperature readings, control the appliance's operational state, store readings locally in a Room database, visualize a timestamped line graph and historical logs of temperature readings, and contact the emergency hotline via the device's dialer application.

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue VI June 2026

The HeatEYE application was designed to be simple and minimalistic, with three main navigation buttons for easy navigation and understanding, improving accessibility, and giving the application a user-friendly interface.

- **Dashboard Screen**

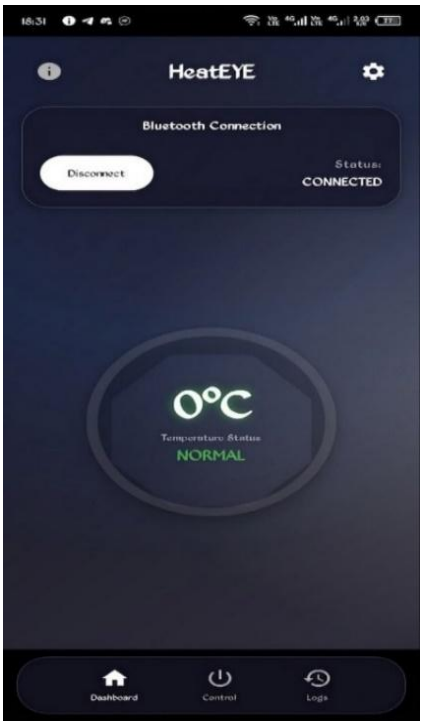

Figure 6. Dashboard Screen of the HeatEye mobile application

As shown in Figure 6, the dashboard displayed the application's Bluetooth connectivity and a live temperature readout.

- **Control Screen**

Figure 7. HeatEye application Control Screen

Figure 7 depicts the application's Control screen, which features automated controls for the appliance's operational state and the auto-off timer.

- **History & Logs Screen**

Figure 8. HeatEye's History & Logs Screen: Graphical Representation and List

The application line graph and historical logs for all recorded temperature data were featured in the History & Logs Screen, as illustrated in Figure 8.

- **Settings Screen**

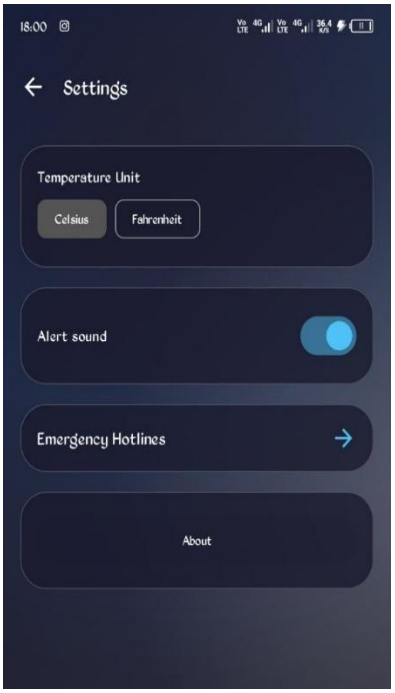

Figure 9. Settings Screen of the mobile application

The application Settings Screen contained two HeatEye features. One is the Temperature Unit, which allows the user to switch between Celsius and Fahrenheit, as illustrated in Figure 9. Moreover, the second, as shown in Figure 10, was the Emergency Screen, which allows the user to contact an Emergency Hotline via a preconfigured number in the device's dialer.

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue VI June 2026

Figure 10. HeatEye's Emergency Hotline Feature

Functionality Test

As described in the Methodology, the app functionality was tested by executing each core feature ten times. See Table 1 for the results of the functionality test.

Table 1. Summary of the Functionality Test Results

Application Core Features	**Expected Outcome**	**Results**
Bluetooth connection or disconnection	The device successfully pairs and reliably connects with the enhanced prototype	10/10
Remote appliance ON/OFF control	The fan or appliances automatically switch ON/OFF after issuing the command from the application	10/10
Auto-Off timer (15, 30, 60 min or custom)	The fan or appliances automatically turn off after the set duration	9/10
Real temperature display and live updates	The dashboard screen shows changing temperature and the temperature status every second.	10/10
Overheating notification (≥65°C)	A pop-up notification and audible alert were triggered	10/10
Data logging and graph generation	Display a timestamped and accurate line-graphed log with continuous updates	10/10
Emergency hotline redirection	A device dialer with a pre-configured hotline number is opened	10/10
Conversion of temperature Unit	Accurately converting the Celsius temperature to Fahrenheit	10/10

As shown in Table 1, all eight features successfully passed the functionality test. Seven features achieved a perfect success rate of 100% (10/10), while the Auto-off Timer feature achieved a success rate of 90% (9/10). The Auto-off Timer feature failed once due to a programming issue that caused the timer to stop when the user navigated to a different screen. To address this problem, the researchers implemented a TimerViewModel, which ensured that the timer continued its countdown even when the application switched between screens.

Aside from the Auto-off Timer issue, no other failures were observed during testing. These results confirm that the HeatEYE mobile application reliably and efficiently performs its intended functions under normal operating conditions.

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue VI June 2026

Accuracy Test

Figure 11. Experimental setup for the accuracy testing of the HeatEye prototype

Before the accuracy testing was conducted, the digital thermometer was verified against the manufacturer's specifications and served as the reference instrument throughout the experiment.

The temperature accuracy of the HeatEye prototype was evaluated by comparing the LM35 temperature sensor readings with those from the digital thermometer. During each trial, both the LM35 sensor and the thermometer probe were immersed in the same water sample to ensure that they were exposed to identical temperature conditions. This setup allowed a direct comparison of the readings from the two instruments.

Figure 11 shows the experimental setup used during testing. The digital thermometer served as the reference instrument, while the LM35 sensor was connected to the HeatEye prototype. Three trials were conducted with water samples at different temperatures to evaluate the prototype's ability to accurately measure and monitor temperature changes.

Table 2. Temperature Readings Obtained During Accuracy Testing

Trial	Digital Thermometer (°C)	HeatEye Reading (°C)	Difference (°C)	Percent Error (%)
1	86.0	86.0	0.0	0.00
2	101.3	101.0	0.3	0.54
3	53.1	52.9	0.2	0.30
Average	80.13	79.97	0.17	0.28

The difference between the digital thermometer readings and the HeatEye prototype readings was calculated for each trial. Percent error was also calculated to assess the prototype's accuracy relative to the reference instrument.

The results indicate that the HeatEye prototype measured temperature with a high degree of accuracy. The temperature readings obtained from the prototype were very close to those recorded by the digital thermometer, resulting in an average percent error of only 0.28%. The highest percent error observed during the test was 0.54%, which is significantly lower than the commonly accepted 5% error limit for sensor-based monitoring systems.

The small differences between the readings of the HeatEye prototype and the digital thermometer may be attributed to factors such as sensor tolerance, response time, slight variations in sensor positioning, and environmental conditions during testing. Despite these minor variations, the prototype consistently produced reliable and accurate temperature measurements across a range of temperatures.

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XIII Issue VI June 2026

Overall, the findings demonstrate that the HeatEye prototype accurately detects and monitors temperature changes. The low percent error observed during testing confirms the reliability of the LM35 sensor and the system's effectiveness in measuring temperature. Therefore, the prototype is suitable for electrical safety monitoring applications, particularly for detecting abnormal temperature increases and potential overheating.

Response Time Test

In accordance with the Response Time Testing Method described in the methodology, the researchers evaluated the performance of the developed system by issuing ON and OFF commands through the mobile application and observing the corresponding response of the enhanced HeatEye prototype. Multiple test trials were conducted to assess the consistency and reliability of the communication between the application and the prototype device.

The results showed that the prototype consistently executed the commands within 1–2 seconds of being issued by the mobile application. During all test trials, the appliance connected to the system responded correctly to the commands without any noticeable delays or communication failures. This demonstrates that the prototype's wireless communication and control mechanism functioned effectively throughout testing.

The observed response time indicates that the system can provide fast, reliable real-time control of connected appliances. The minimal delay between command transmission and execution enhances system usability and ensures prompt user control. Based on these findings, the HeatEye prototype demonstrated satisfactory response time and is suitable for practical household appliance monitoring and control applications.

CONCLUSION

The developed heat detection and temperature monitoring system demonstrated its intended purpose of monitoring appliance temperature and responding effectively to abnormal heat conditions. Based on the evaluation and testing conducted, the system demonstrated the ability to accurately detect temperature changes and provide reliable monitoring during operation. The integration of real-time monitoring and automated response improved the system’s ability to manage overheating, making it more efficient in supporting appliance safety and reducing the potential risks associated with excessive heat.

The results from the testing phase also indicated that the system consistently detected high-temperature conditions and provided timely notifications through the connected mobile application. The inclusion of application-based monitoring improved accessibility and allowed users to observe appliance conditions more conveniently, even without direct physical supervision. Through this approach, the study promoted a more proactive approach to handling overheating situations, compared to conventional systems that mainly rely on manual monitoring and delayed responses.

Furthermore, the study's findings confirmed that the researchers' objectives were successfully achieved. The developed system integrated temperature detection, monitoring, and user accessibility into a single platform to support safer appliance use. Although the research focused solely on the development and evaluation of the proposed system, the results highlighted its potential as a practical tool for improving household safety and minimizing risks from overheated appliances. The study also highlights the potential for future improvements and the wider application of the system to advance safety and monitoring technologies.

LIMITATIONS AND FUTURE IMPROVEMENTS

This section discusses the essential considerations and limitations identified in the conducted study. Despite the successful development of the HeatEYE system, several constraints were observed that may affect its further enhancement and scalability. One major limitation is the restricted range of the current wireless communication components, which prevents the system from functioning as a fully reliable smart heat monitoring and control solution. In addition, the absence of a fail-safe mechanism when the connection is lost or when the mobile application is closed limits the system’s ability to continuously send overheating notifications and execute remote commands. The prototype is also limited in hardware scalability, as it currently supports only one temperature sensor, allowing it to monitor a single appliance at a time.

Furthermore, because only one prototype unit was available, the researchers found that conducting a full-scale user acceptance test was highly impractical within the study's scope and timeline. Since the HeatEYE mobile application operates in conjunction with an Arduino-based hardware device, a comprehensive usability evaluation involving multiple households was not feasible. As a result, user acceptance testing across broader real-world household conditions could not be performed.

To address these limitations and improve the system's overall performance, functionality, and applicability, several recommendations are proposed for future research. It is suggested that future developers adopt Wi-Fi-based modules, such as the ESP32 or ESP8266, to improve communication range and eliminate distance constraints. The implementation of a fail-safe mechanism is also recommended to ensure users continue to receive alerts even when the application disconnects. Lastly, expanding the system to support multiple temperature sensors is advised to enable simultaneous monitoring and control of multiple appliances, allowing for wider and more practical household deployment.

ACKNOWLEDGEMENT

The researchers would like to sincerely thank Engr. Jose C. Felipe Jr. for his guidance, technical help, patience, and useful feedback throughout the development of this study. His support and expertise greatly helped improve and complete this research, and the researchers truly appreciate the time and effort he devoted to checking and improving the work. The researchers also thank all the people who shared their ideas, knowledge, and assistance during the project, as their help was important in finishing the study. Lastly, the researchers are grateful to their families and friends for their continued support and encouragement, which motivated them throughout the research process.

REFERENCES

1. AlQahtani, A. A. S., Sulaiman, M., Alshayeb, T., & Alamleh, H. (2025). From inception to innovation: A comprehensive review and bibliometric analysis of IoT-enabled fire safety systems. Safety, 11(2), 41. https://doi.org/10.3390/safety11020041
2. Arafat, M. Y., Hossain, M. S., & Rahman, M. M. (2020). Smart fire detection system using IoT technology. International Journal of Advanced Computer Science and Applications, 11(5), 570–576.
3. Banzi, M., & Shiloh, M. (2022). Getting started with Arduino (4th ed.). Maker Media.
4. Bhoyar, K., & Sahare, P. (2019). Home automation using Bluetooth and Arduino. International Journal of Innovative Technology and Exploring Engineering, 8(6), 1200–1204.
5. Casinillo, R. M. L., So, A. L. A., Mandaya, M. V., Dabalos, S. A. J., Enriquez, M. C. S., & Cane, J. F. (2024). Development of Arduino-based high heat detector temperature control prototype for household appliances. IAES International Journal of Robotics and Automation (IJRA), 13(2), 140–159. https://ijra.iaescore.com/index.php/IJRA/article/view/20620
6. Chauhan, S., & Singh, R. (2021). Smart appliance monitoring system using embedded technology. International Journal of Engineering Research & Technology, 10(7), 332–336.
7. Components101. (n.d.). HC-05 Bluetooth module. https://components101.com/wireless/hc-05-bluetooth-module
8. Components101. (n.d.). LM35 temperature sensor pinout and features. https://components101.com/sensors/lm35-temperature-sensor
9. Components101. (n.d.). Relay module pinout and working. https://components101.com/switches/5v-single-channel-relay-module-pinout-features-applications-working-datasheet
10. Danaryani, S., Agustina, V., & Syamsudin, F. B. H. (2019). Fire detection based on IoT through fiber to the home (FTTH). Jurnal Poli-Teknologi, 17(2). https://doi.org/10.32722/pt.v17i2.1316
11. De La Salle University. (2022). ODET: An Arduino-based overheat detector for household appliances. https://animorepository.dlsu.edu.ph/conf_shsrescon/2022/paper_csr/5/
12. Electronics Hub. (n.d.). Relay module working principle. https://www.electronicshub.org/relay-module/
13. Firdaus, Tjandi, Y., & Pratama, A. (2023). Design of a fire detection system using fire and smoke sensors based on Arduino microcontroller. Jurnal Media Elektrik, 21(2). https://doi.org/10.59562/metrik.v21i2.2211

14. Ghosh, A., & Das, R. (2020). Android-based home automation system using Bluetooth. International Journal of Scientific Research in Computer Science Engineering and Information Technology, 6(3), 269–274.
15. Hasan, M. R., Islam, S., & Rahman, A. (2022). Design and implementation of smart home safety system using IoT. Heliyon, 8(6), e09650. https://doi.org/10.1016/j.heliyon.2022.e09650
16. Ifani, A. Z., Fajar, M. I. N., & Badila, A. A. (2026). IoT-based early fire detection system uses MQ-2 smoke sensor and DS18B20 temperature sensor. Infact: International Journal of Computers, 10(1), 22–29. https://doi.org/10.61179/infact.v10i01.763
17. Islam, M. T., Rahman, M. A., & Hasan, M. K. (2021). IoT-based intelligent fire detection and alarm system. International Journal of Electrical and Computer Engineering, 11(2), 1452–1460.
18. Juwariyah, T., Prayitno, S., Krisnawati, L., & Sulasminingsih, S. (2021). Design of IoT-based home fire detection system equipped with a data logger. IOP Conference Series: Materials Science and Engineering, 1125, 012079. https://doi.org/10.1088/1757-899X/1125/1/012079
19. Kaur, A., & Kaur, P. (2019). Smart home automation using IoT and Android application. International Journal of Computer Applications, 178(7), 1–5.
20. Khalaf, O. I., Abdulsahib, G. M., & Zghair, N. A. K. (2019). IoT fire detection system using sensor with Arduino. Revista AUS, 26(1), 26–30.
21. Khalid, A., et al. (2020). IoT-based fire detection and temperature monitoring system. Procedia Computer Science, 170, 1171–1176. https://www.sciencedirect.com/science/article/pii/S1877050920312087
22. Kumar, N., & Kumar, P. (2018). Smart home automation system using Bluetooth technology and Arduino. ResearchGate. https://www.researchgate.net/publication/328073462_Smart_Home_Automation_System_Using_Bluetooth_Technology_and_Arduino
23. Kumar, S., & Kumar, V. (2020). Automatic temperature monitoring and controlling system using Arduino. International Journal of Engineering Applied Sciences and Technology, 5(4), 334–338.
24. Li, S., Xu, L. D., & Zhao, S. (2015). The Internet of Things: A survey. Information Systems Frontiers, 17(2), 243–259. https://doi.org/10.1007/s10796-014-9492-7
25. Malik, S., & Ahmad, N. (2021). Development of a low-cost smart fire alarm system using Arduino. International Journal of Advanced Research in Computer and Communication Engineering, 10(5), 45–50.
26. Microsoft. (n.d.). Bluetooth overview. https://learn.microsoft.com/en-us/windows/uwp/devices-sensors/bluetooth
27. Mishra, P., & Sharma, R. (2022). IoT-enabled smart monitoring system for household appliances. Materials Today: Proceedings, 56, 2150–2155. https://doi.org/10.1016/j.matpr.2021.11.420
28. National Fire Protection Association. (2023). Electrical fires. https://www.nfpa.org/
29. Perilla, F. S., Villanueva, G. R., & Cacanindin, N. M. (2018). Fire safety and alert system using Arduino sensors with IoT integration. Proceedings of the 7th International Conference on Software and Computer Applications, 199–203. https://doi.org/10.1145/3185089.3185121
30. Prasad, K. N., & Rao, B. P. (2021). Real-time temperature monitoring system using wireless sensor network. International Journal of Scientific & Technology Research, 10(2), 80–84.
31. Rajput, P., & Yadav, S. (2020). Smart home automation and security system using Arduino and Android application. International Research Journal of Engineering and Technology, 7(6), 4210–4215.
32. Saeed, F., Paul, A., Rehman, A., Hong, W. H., & Seo, H. (2018). IoT-based intelligent modeling of smart home environment for fire prevention and safety. Journal of Sensor and Actuator Networks, 7(1), 11. https://doi.org/10.3390/jsan7010011
33. Sarhan, Q. I. (2020). Systematic survey on smart home safety and security systems using the Arduino platform. IEEE Access, 8, 128362–128384. https://doi.org/10.1109/ACCESS.2020.3008610
34. Sharma, A., & Gupta, R. (2021). Embedded systems for home automation and safety monitoring. International Journal of Recent Technology and Engineering, 10(1), 55–60.
35. Singh, H., & Verma, A. (2020). Wireless sensor-based temperature monitoring system for smart homes. International Journal of Innovative Research in Technology, 7(2), 112–118.
36. Suklabaidya, S., & Das, I. (2021). IoT as a platform: For smart home analysis and monitoring of fire parameters. In Advanced Computing and Intelligent Technologies (pp. 329–338). Springer.

https://doi.org/10.1007/978-981-16-2164-2_27
37. Texas Instruments. (n.d.). LM35 precision centigrade temperature sensors datasheet. https://www.ti.com/product/LM35
38. Texas Instruments. (n.d.). LM35 temperature sensor technical documentation. https://www.ti.com/lit/ds/symlink/lm35.pdf
39. Vernekar, H. G. (2023). Detection of fire and its control using Arduino and Proteus. Journal of Electronics and Informatics, 5(1), 54–62. https://doi.org/10.36548/jei.2023.1.004
40. World Health Organization. (2022). Burns and household fire injuries. https://www.who.int/news-room/fact-sheets/detail/burns
41. Yerpude, S., & Singhal, T. K. (2018). Smart home system using Internet of Things. International Journal of Advanced Research in Computer Science, 9(1), 181–184.
42. Kliuba, M., & Likhouzova, T. (2023). Software tools for creating interfaces for interaction with Arduino via Bluetooth. Adaptive Systems of Automatic Control, 2(43), 3–11.
43. Roque, G., & Padilla, V. S. (2020). LPWAN based IoT surveillance system for outdoor fire detection. IEEE Access, 8, 114900–114909.
44. Paglinawan, C. C., Bacabac, M. L. C., & Garcia, T. J. D. (2022). Electrical fault detection and analysis Arduino-based preventive device for household appliances. 2022 IEEE Region 10 Symposium (TENSYMP), 1–6.
45. Emets, S. V., Polischouk, I. N., & Kudayarov, V. N. (2020). Processing calibration results for measuring transducers with an integrated sensor. Journal of Physics: Conference Series, 1582(1), 012026.
46. Rathore, R. V., Dixit, S., & Arvindhan, A. (2024). Home automation using Bluetooth HC-05 and ATmega328. Journal of Engineering Research and Applications, 14(5), 28-34.