Vilnius University

Faculty of Physics

Waqas Ahmad

Application of infrared thermal camera in Microcontroller based systems

Master's final thesis

Electronics and Telecommunications Technologies

Student	Waqas Ahmad
Approved	22-05-2026
Academic supervisor	Assoc.Prof. Vytautas Jonkus
Director/ representative of the Institute	Prof. Robertas Grigalaitis

Vilnius 2026

Contents

Abstract

Infrared thermography is non contact temperature measurement technique that enables visualization of heat distribution by detecting infrared radiation emitted from objects. In recent years, the integration of thermal sensors with embedded systems has enabled the development of compact and cost-effective thermal imaging solutions.

In this thesis which presents the design, implementation, and optimization of a microcontroller based thermal imaging system developed across multiple stages. In the initial phase, a low-resolution MLX90640 infrared sensor was interfaced with an ESP32 microcontroller to acquire thermal data and transmit it wirelessly through a web-based interface. This stage primarily focused on data acquisition, calibration, and real-time visualization.

It was extended to next phase for practical applications in electronic circuit analysis. Thermal imaging was used to study the behavior of printed circuit boards (PCBs) under different operating conditions, including fault detection scenarios. The results showed ability of system to detect hotspots and identify abnormal thermal behavior.

In the final stage of the project, the system was upgraded to more advanced platform using the STM32N6 Nucleo-144 microcontroller together with the ATI320 thermal camera. As compared to the earlier MLX90640-based setup, ATI320 provides high resolution RAW14 thermal image data due to this image processing was required before the thermal data could be displayed properly.

Several processing methods were implemented to improve image visualization, including linear compression, histogram based image enhancement, percentile clipping and dynamic range compression. During testing, these techniques helped improve image contrast and reduced the effect of extremely hot objects dominating the scene.

1. Introduction

Non-contact thermography is a way of measuring surface temperature without touching the object being measured. Instead of physical contact, it relies on detecting the infrared radiation naturally emitted by surfaces. Over the last few years, this approach has become common in areas such as medical screening, monitoring of industrial machinery, and inspecting buildings for heat loss. [2]

Compared with traditional contact sensors, thermographic systems can capture the temperature distribution of a whole area in one shot which makes it easier to detect hotspots, air leaks, and other thermal anomalies that might detect faults or anomalies. [2]

In this work development of thermal imaging system through several stages. It started with basic wireless thermal imaging platform and gradually upgraded into high performance embedded thermal imaging system capable of real time processing and improved visualization.

The main objective of this thesis is to design and implement an efficient thermal imaging pipeline that improves visualization quality, supports multiple object detection in real time and detects abnormal behavious in electronics circuit design.

2. Literature Overview

2.1 Fundamentals of Infrared Radiation

Non-contact infrared thermometry relies fundamentally on blackbody radiation theory. An ideal black body absorbs all incident radiation and re-emits energy according to its absolute temperature. [2] Planck's law describes the spectral radiance $L(\lambda, T)$ of a black body at temperature T and wavelength λ:

$$L(\lambda, T) = \left(\frac{C_1}{\lambda^5}\right) \cdot \frac{1}{\left(e^{\frac{C_2}{(\lambda T)}} - 1\right)} \quad \text{.....(1)}$$

where $C1 = 3.74 \times 10^{-16}\ Wm^2$ and $C2 = 1.44 \times 10^{-2}\ Km$. By integrating $L(\lambda, T)$ over all wavelengths, one obtains the Stefan–Boltzmann law,

$$M = \sigma T^4 \quad \text{……. …….(2)}$$

With $\sigma = 5.67 \times 10^{-8}\ Wm^{-2}K^{-4}$, giving total power radiated per unit area. [2]

2.2 Surface Emissivity

Real objects deviate from the ideal blackbody; they are grey bodies with emissivity $\varepsilon < 1$. Emissivity is the ratio of radiation emitted by the object to that of a black body at the same T. It depends on material, surface finish, wavelength and temperature. Metallic surfaces typically have low, wavelength-dependent emissivity, whereas non-metallic (e.g., plastics, paint) often exhibit high, near-constant emissivity in the long-wave IR band. [2]

The relation between emissivity, reflectivity and transmissivity is given by: [2]

$$\varepsilon + \phi + \tau = 1 \text{ ……..…….(3)}$$

where ε is emissivity, ϕ is reflectivity and τ is transmissivity. For most opaque materials used in electronic circuits, transmissivity is negligible; therefore: [2]

$$\varepsilon + \phi = 1 \text{ ……...……(4)}$$

In electronic circuits, different materials such as PCB silkscreen, integrated circuit packages, relays, and metallic connectors exhibit different emissivity values. Materials with higher emissivity produced stronger thermal radiation, while metallic surfaces such as solder joints showed lower infrared response due to reflective behavior The accuracy of temperature measurements can be impacted by changes in emissivity.

Component	Material Type	Emissivity (Typical)
PCB Substrate	FR-4 with Silkscreen	High (0.8-0.9)
IC Packages	Plastic(epoxy)	High (≈0.9)
Solder Joints and leads	Metal	Low-Medium (0.2-0.4)
Power Components	Mixed(Plastic + Metal)	Medium-High (0.4-0.6)

Table 1: Typical emissivity values of Common PCB Components

2.3 Infrared Thermography in Electronic Systems

Infrared thermography has become very good for monitoring thermal behavior in electronic systems. It enables:

- Detection of localized hotspots
- Identification of short circuits and overheating components
- Visualization of heat distribution across PCBs

Unlike contact-based sensors (e.g., thermocouples), infrared imaging provides full-field temperature mapping without interfering with system operation which makes it particularly suitable for real time diagnostics and fault detection. [4]

2.4 Thermopile Sensor Operation

Thermopile array works using micro-fabricated thermopile elements. Each pixel consists of series of thermocouples placed on a suspended membrane when infrared radiation is absorbed, the temperature of the membrane increases, which generates a voltage signal. [1] A cold junction located on the substrate acts as a reference point, so the produced voltage is proportional to the temperature difference between the object and the reference. [1]

Based on the Stefan Boltzmann law, the electrical output of the thermopile detector is proportional to the emissivity of the object and to the fourth power of its absolute temperature. [2]

$$U \sim \varepsilon T_{obj}^4 \quad \text{.....(5)}$$

where $U$is the detector output voltage, ε is the emissivity of the object, and T_{obj} is the object temperature. The analog voltage signal generated by the thermopile array is processed using internal digital filtering and memory blocks, and the temperature data is communicated through the I2C interface. [2]

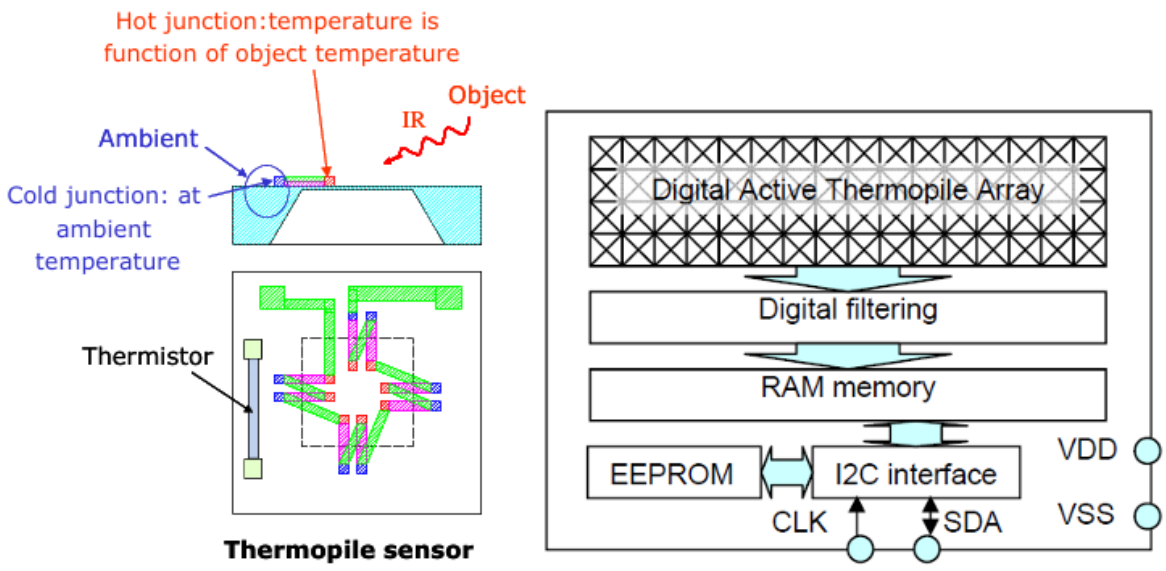


Figure 1: Principle of a single pixel thermopile (left); 4x16 digital active thermopile array (MLX90620) concept (right) [2]

2.5 Calibration and Correction Techniques in Thermal Imaging Systems

Thermal imaging systems need calibration and correction to give accurate and stable results. Infrared sensors do not measure the exact temperature directly. Instead, they produce raw output signals that are related to the infrared radiation coming from the object. These raw signals depend on different factors such as sensor non uniformity and changes in the surrounding environment. For this reason, calibration and correction are important parts of any thermal imaging system

2.6 Thermal Image Formation and Linear Scaling

Thermal cameras usually produce high dynamic range data, such as 14-bit data, while most display devices use 8-bit format.Therefore, a mapping function is required to convert raw sensor values into displayable grayscale intensities. The most basic method is linear min–max scaling, defined as: [5]

$$I_{gray} = \frac{(I_{raw} - I_{min})}{(I_{max} - I_{min})} \times 255 \quad \text{.....(6)}$$

where:

- I_{raw}= raw pixel value (14-bit)
- I_{min}, I_{max}= minimum and maximum values in the frame

Limitations

Although simple, this approach has major drawbacks:

- Highly sensitive to outliers
- Dominance of high-temperature objects
- Background suppression (appears dark)
- Poor visualization of multiple objects

These limitations were observed experimentally, motivating the use of more advanced methods.

2.7 Histogram-Based Enhancement Techniques

Histogram based enhancement techniques are widely used to overcome the limitations of linear compression. These methods analyzed the distribution of pixel intensities rather than using extreme values.

2.7.1 Histogram Representation

A histogram divides the intensity range into discrete bins, the 14-bit raw data is reduced to 8-bit bins: [6]

$$Bin = \frac{I_{raw}}{2^6} = I_{raw} >> 6 \quad \text{.....(7)}$$

This creates a 256-bin histogram, where each bin counts the number of pixels within that intensity range.

2.7.2 Percentile-Based Clipping

Instead of using absolute minimum and maximum values, percentile based clipping improves intensity scaling by excluding extreme values from the selected thermal range.:

$$I_{min} = P_{low}, I_{max} = P_{high} \quad \text{.....(8)}$$

where:

- P_{low}= lower percentile (e.g., 15%)
- P_{high}= upper percentile (e.g., 85%)

In implementation, this is obtained by cumulative histogram summation:

- Find bin where cumulative count ≥ 15% of pixels
- Find bin where cumulative count ≥ 85% of pixels

This approach removes:

- Noise (low values)
- Extreme hotspots (high values)

It works much better because:

- Ignores outliers
- Preserves majority of scene
- Enables multiple objects to be visible

2.8 Dynamic Range Compression and Intensity mapping

Even after percentile clipping, scenes with large temperature differences can still suffer from imbalance. To address this, dynamic range compression are applied.

2.8.1 Clamped Linear Scaling

After defining $I_{min}$and I_{max}, values are clamped:

$$I_{clamped} = \begin{cases} I_{min}, & I_{raw} < I_{min} \\ I_{max}, & I_{raw} > I_{max} \\ I_{raw}, & \text{otherwise} \end{cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(9)$$

Then normalized:

$$I_{norm} = \frac{(I_{clamped} - I_{min})}{(I_{max} - I_{min})} \times 255 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(10)$$

2.8.2 Dynamic range compression

In practical scenarios, it was observed that even after normalization, very hot or close objects could dominate the image. To overcome this limitation, a dynamic range compression technique was implemented.

High intensity values are selectively compressed, This approach reduces the influence of extreme intensity values while preserving overall contrast.[7]

2.8.3 Practical Effect

The combination of Percentile clipping, clamping results in Multiple thermal objects visible simultaneously, reduced dominance of very hot objects, improved background visibility and better edge definition.

3. Project Evolution Across Semesters

3.1 Initial Phase: Thermal Data Acquisition and Streaming

In initial stage the system was made to get thermal data from the MLX90640 sensor and transmit it over Wi-Fi using the ESP32 microcontroller.

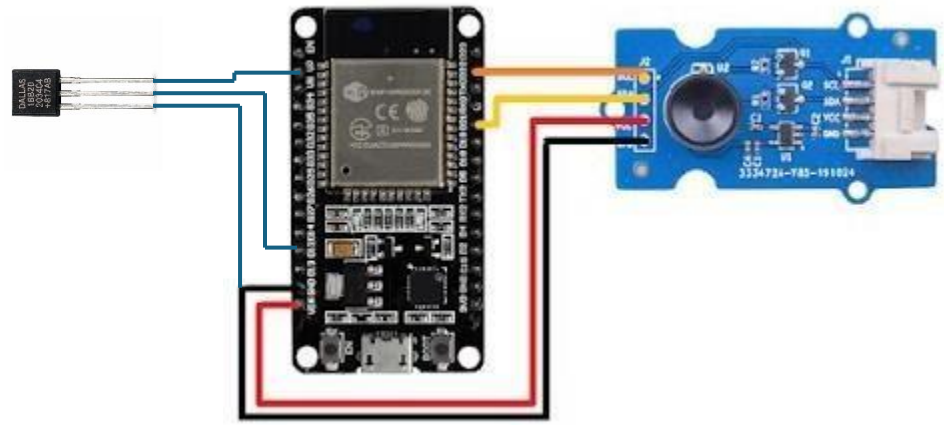


Figure 2: Schematic Diagram

A web interface was developed to visualize the thermal data in real time using a color-coded heatmap. The ESP32 host HTTP server that provided temperature data in JSON format.

This phase successfully demonstrated:

- Real time thermal data acquisition
- Wireless streaming
- Web-based visualization

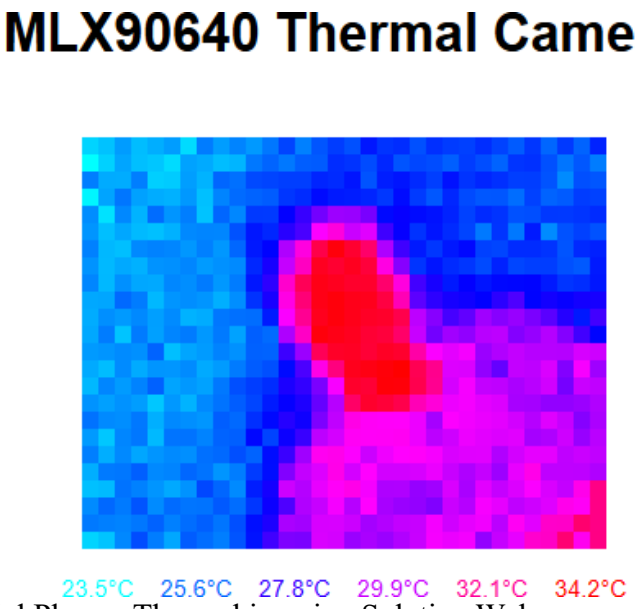


Figure 3: Initial Phase: Thermal imaging Solution Webpage

However, the focus was mainly on functionality rather than image quality.

Figure2 below presents the block diagram of the complete system.

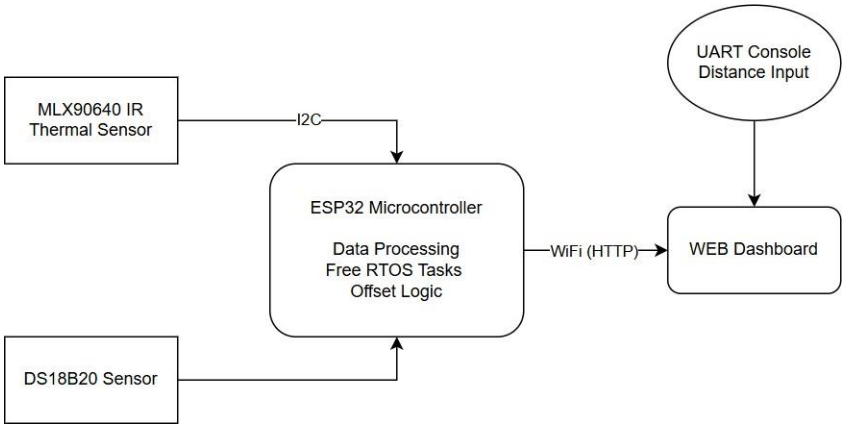


Figure 4: High Level Block diagram of the complete system

3.2 Intermediate Phase: PCB Fault Detection

The thesis was extended for practical application in next phsae. Thermal behavior of the PCB was evaluated under different conditions, including unpowered, normal operating, and short-circuit fault scenarios.

The experimental setup consisted of an MLX90640 infrared thermal sensor positioned above the PCB under test at controlled distances ranging from 5 cm to 30 cm. Thermal data was acquired with a resolution of 32 × 24 pixels. A DS18B20 contact temperature sensor was connected to the PCB surface and used as a reference sensor for temperature correction and validation.

Different materials on PCB have different emissivity characteristics which directly affect infrared temperature measurement. During experimental testing variation were observed between PCB silkscreen, IC packages, solder joints, and power components. Although it successfully detects thermal hotspots, measurement accuracy is affected by emissivity variation, reflective PCB surfaces, sensor distance, and the limited spatial resolution of the MLX90640 sensor.

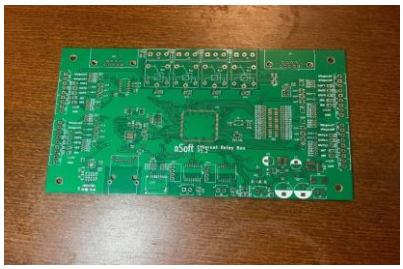
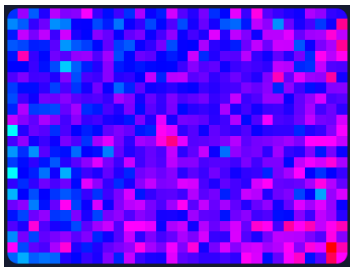
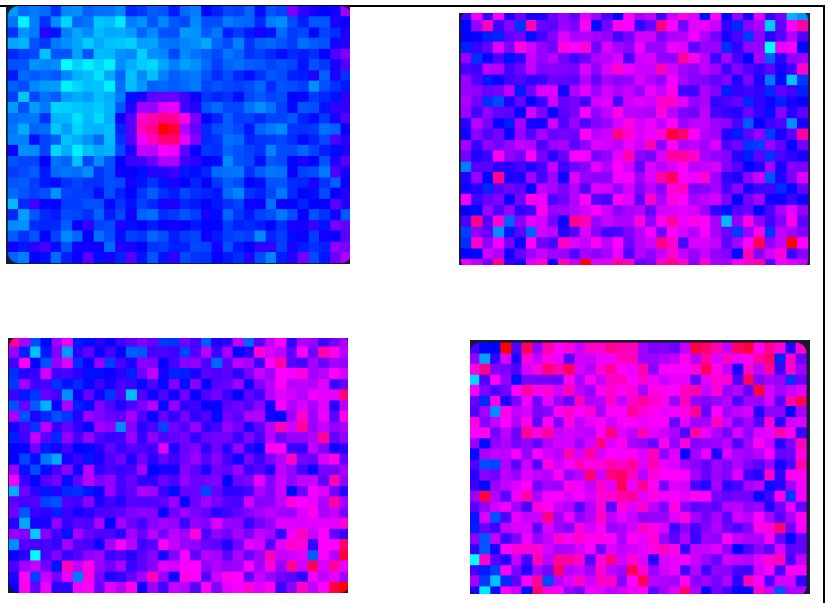


Figure5:PCB Silkscreen for testing



Thermal Visualization of PCB Silkscreen

Figure6: PCB Board for Testing



Thermal Visualization: Shiny Surface(Top Left),Power Component(Top Right),IC Package(Bottom Left),whole pcb board(Bottom right)

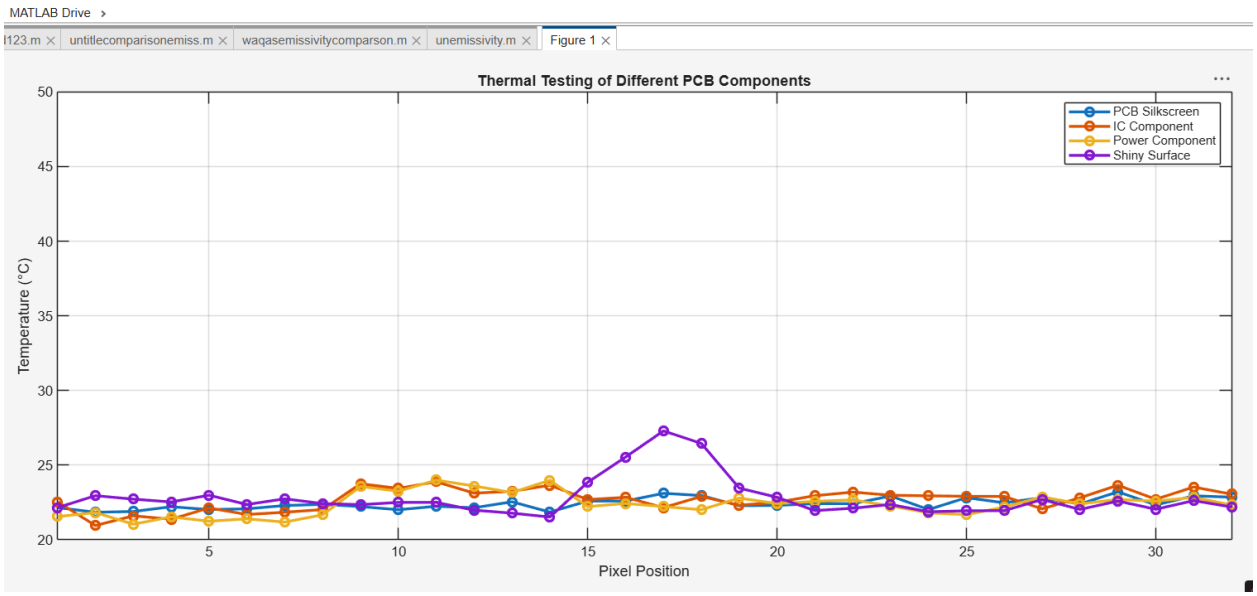


Figure 7: Thermal Response Comparison of Different PCB Materials

As shown in Figure 7, the normal PCB silkscreen region exhibits relatively stable temperature behavior with only minor fluctuations. the IC and power components demonstrates a moderate temperature increase.

However, the shiny or reflective surface shows irregular temperature variations compared to other components. This behavior is caused by low emissivity and infrared reflection effects, which influence the accuracy of thermal measurements. The results demonstrate that PCB material properties significantly affect infrared temperature measurements.

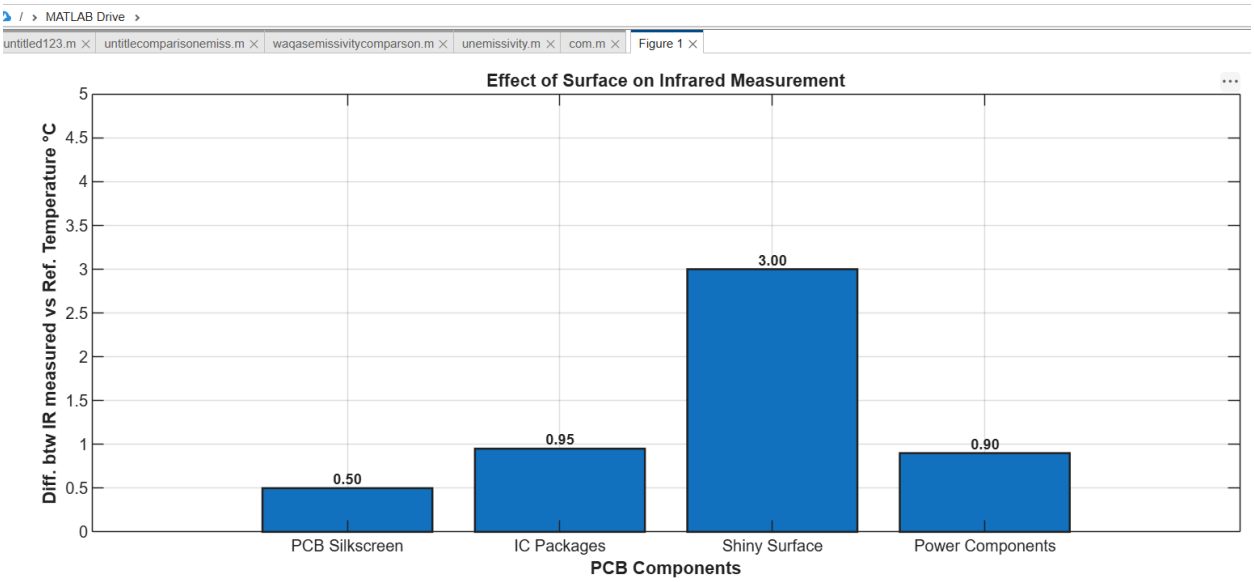


Figure 8: Effect of Surface Material on Infrared Measurements

Infrared temperature measurements obtained from the MLX90640 sensor were effected by sensor to PCB distance. To improve the consistency of the measurements, a reference-based correction approach was implemented using a DS18B20 contact temperature sensor mounted on the PCB.

The each pixel of sensor represents the temperature measured by an individual thermopile element. For each acquired frame, a total of 768 temperature values are obtained and denoted as $TIR(i, j)$.

To obtain a representative temperature value for the entire PCB surface, the average infrared temperature is calculated as

$$T_{\{avg\}} = \frac{1}{768} \sum_{\{i,j\}} T_{\{IR\}(i,j)} \quad \text{.....(11)}$$

At the same time, the DS18B20 sensor measures the contact temperature of the PCB. This value is used as a reference temperature and is denoted as $Tref$.

A correction is applied by comparing the average infrared temperature with the reference temperature. If the infrared temperature is lower than the reference, a correction offset is added. If it is higher, the offset is subtracted.

During experimental testing, it was observed that the sensor to object distance also affects the measured temperature values. For this reason, the distance between the MLX90640 sensor and the PCB was entered manually by the user via a UART interface. Based on this distance, predefined correction offsets were applied.

A correction offset was applied based on experimental measurements performed on the PCB under testing:

- For distances up to 20 cm, correction offset of ±1.0 °C was used.
- For distances between 20 cm and 30 cm, the offset was set to ±1.5 °C
- For distances greater than 30 cm, an offset of ±2.0 °C was applied.

The selected correction offset is applied uniformly to the thermal data. This preserves relative temperature differences and hotspot patterns across the PCB.

Figures 9 and 10 shows software architecture which includes the high level block diagram and code flow. The main components in code flow are sensor initialization, data acquisition, processing, and web server handling, thermal data from the sensor is processed and served to the browser through HTTP endpoints for real-time visualization.

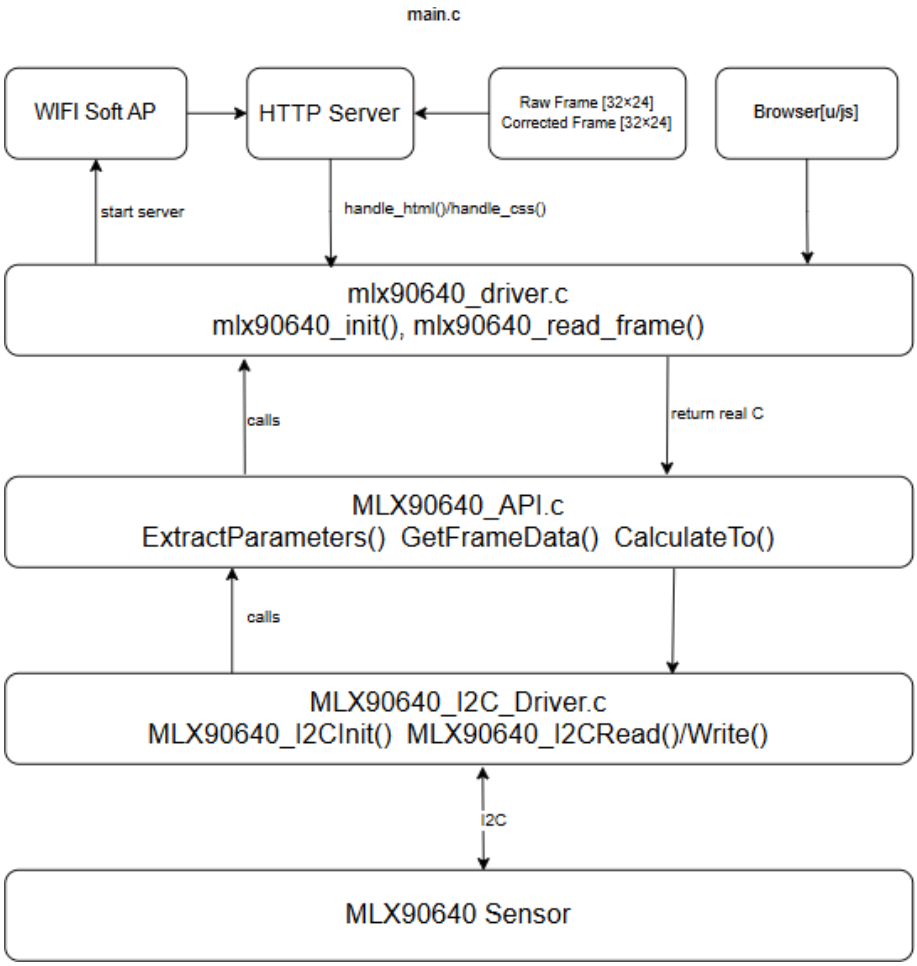


Figure 9: Software Architecture and Module Interaction

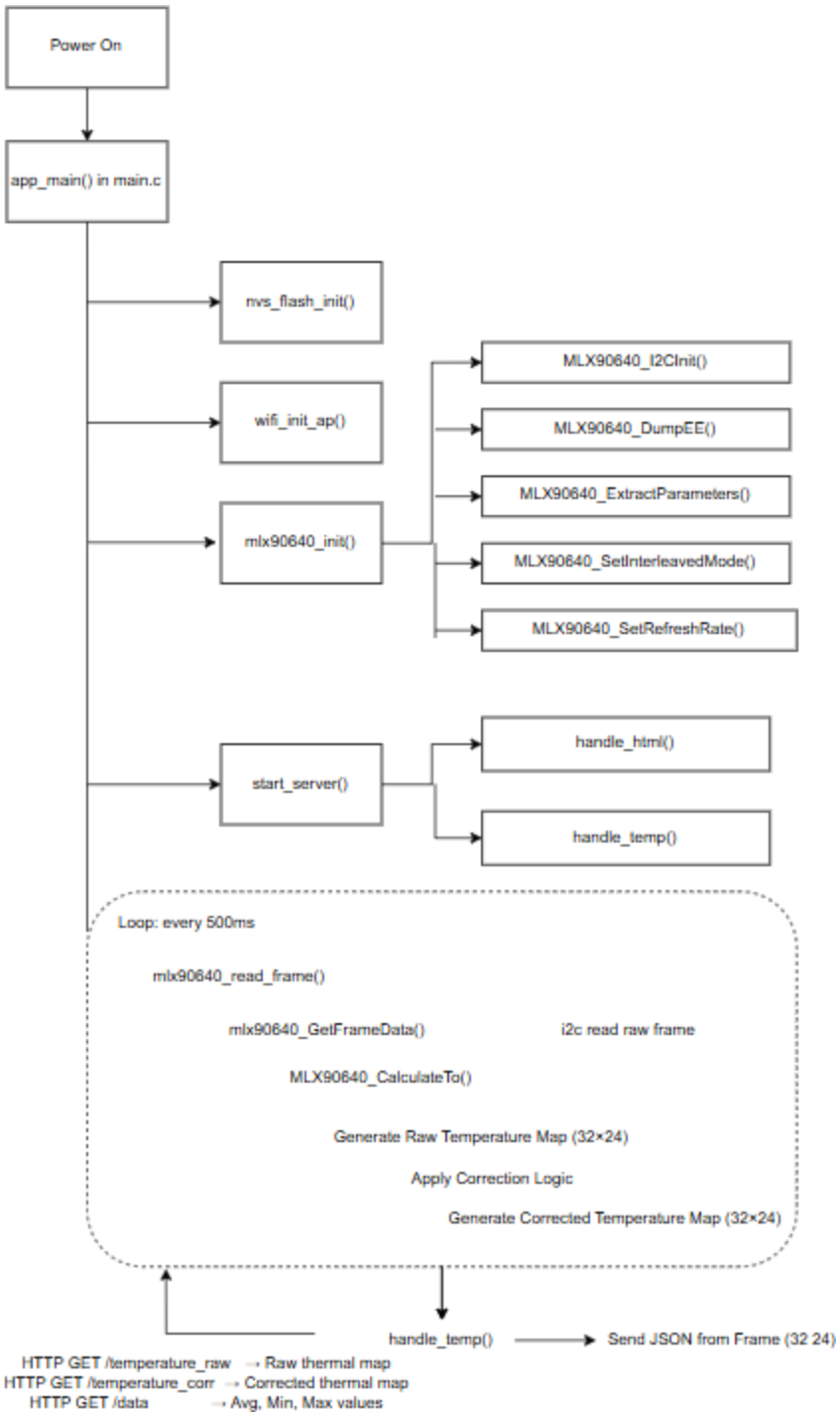


Figure 10: Code Flow for Real-Time PCB Thermal Measurement

A short circuit near the power supply of the board was intentionally created by shorting the diode to test the fault detection ability of the system. These kind of problems typically happen in real electronic systems so it is useful test case.

Under normal operating conditions before the fault was introduced, the PCB did not show any unusual heating, and the temperature map was fairly even across the board, as shown in Figure 12. After adding the short-circuit, the infrared thermal images showed a clear local hotspot in the affected area. In Figures 13,14 and 15, the temperature around the shorted diode goes up much more than the surrounding parts of the PCB, while most other components stay relatively cooler.

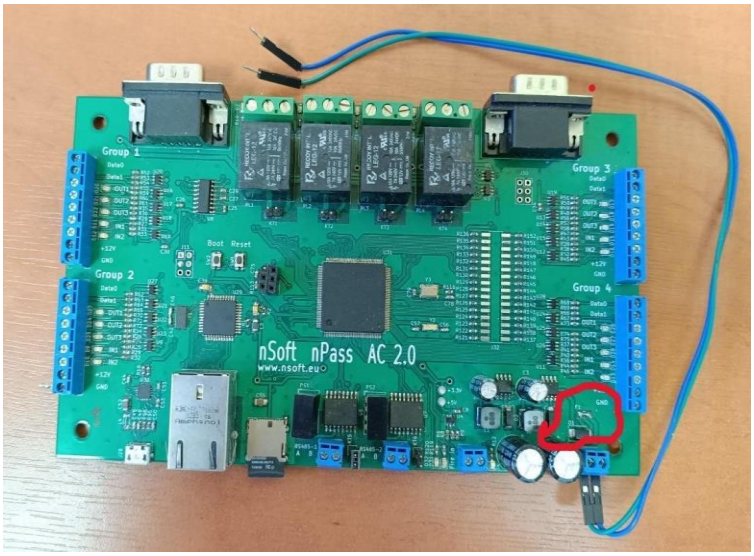


Figure 11: Short Circuit near Powered PCB Board for Testing

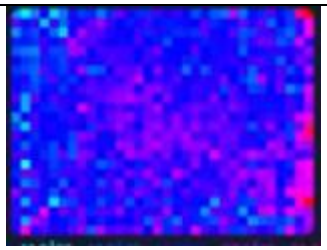
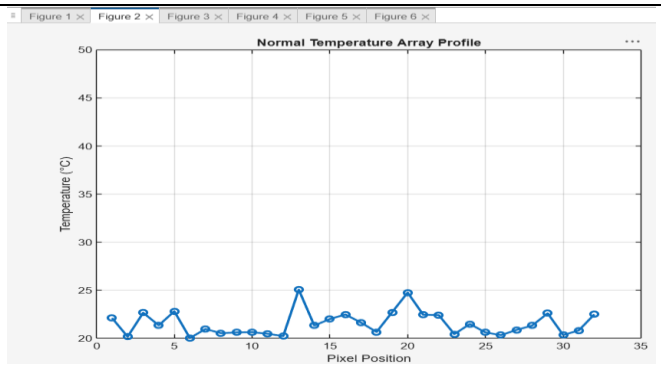


Figure 12: PCB Board operating at normal state



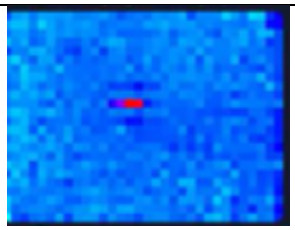
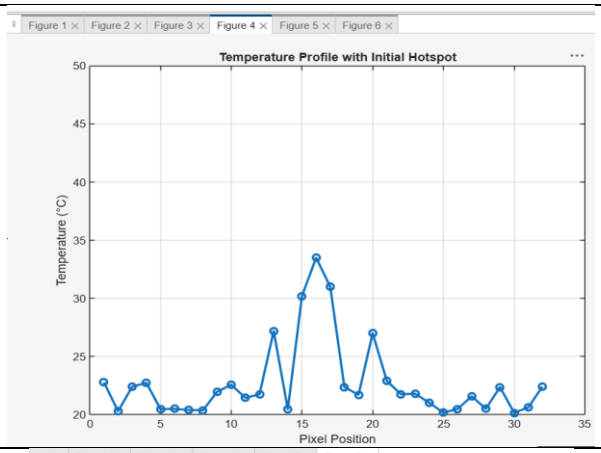


Figure 13: PCB Board just powered on state(fault detected as it is clearly visible)



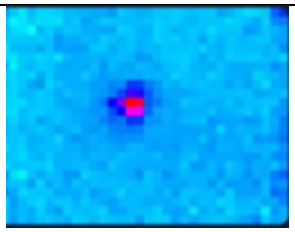
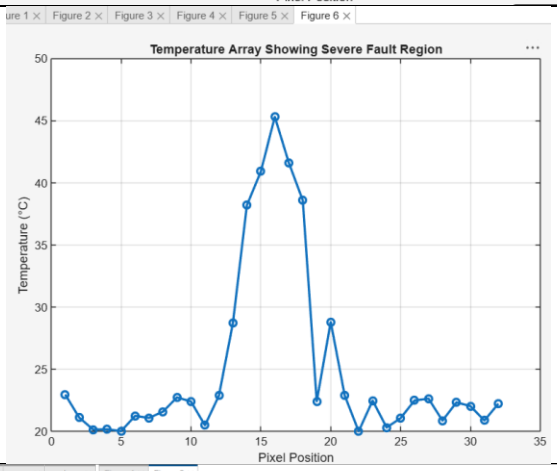


Figure 14: Hotspot visible (noticeable increase in temperature at that region **)**



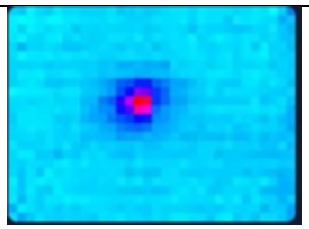
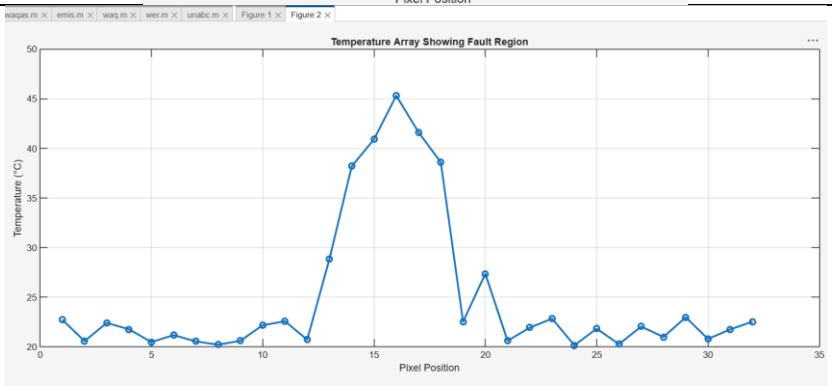


Figure15: Hotspot clearly visible



The corresponding temperature array profiles shown beside each thermal image further confirm the thermal behavior observed on the PCB surface. Under normal operating conditions, the temperature distribution remains relatively stable with only small variations between neighboring pixels.

After the fault condition was introduced, the temperature profile shows sudden increase in a localized region of the array. As the fault becomes more severe, the hotspot intensity increases significantly, producing a clear peak in the temperature graph. The surrounding pixels also experience moderate heating due to the fault location..

From these results, it can be seen that the infrared thermal monitoring setup is able to detect local abnormal heating caused by faults even though a low cost infrared sensor array was used the system was still able to find faulty region just by looking at the thermal behaviour of the board. This shows that infrared based monitoring can be used for electronic circuit fault detection.

Experimental results showed that:

- Faults produced clear thermal hotspots
- The system could detect abnormal heating
- Thermal imaging is effective for PCB diagnostics

3.3 Final Phase: Advanced Thermal Image Processing

The final system was a major improvement in both hardware and processing performance. It used the STM32N6 microcontroller together with the ATI320 thermal camera, which produce RAW14 image data because of this full image processing pipeline had to be developed to turn the raw sensor output into clear visual images.

Some of the main challenges which are addressed here are:

- High dynamic range data handling
- Real-time processing
- Visualization quality improvement
- Multiple object detection

4. Hardware Design and Components

4.1 System Overview

The system consists of a thermal camera connected to a high-performance microcontroller, which processes and streams the data.

Figure 16: STM32 board integrated with ATI320

4.2 STM32N6 Nucleo-144

The STM32N6 Nucleo-144 development board based on the STM32N6 microcontroller which feature ARM Cortex M55 processor. This is used for high performance embedded applications requiring real-time data acquisition, image processing, and high-speed communication.
Unlike conventional low-power microcontrollers typically used for sensor interfacing, the STM32N6 provides sufficient computational capability to process high-resolution thermal image data in real time. This makes it suitable for advanced thermal imaging applications involving RAW image processing, histogram analysis, adaptive gain control, and real-time streaming.
The reason for using the STM32N6 microcontroller is the ability to handle continuous real time image streaming directly on the microcontroller.
The STM32N6 also provides:

- Large SRAM and memory resources for frame buffering
- High-speed USB interface for video transmission
- DMA-based data transfer for efficient memory operations
- RTOS support for multitasking and synchronized frame processing

- Hardware interfaces suitable for high-bandwidth camera communication

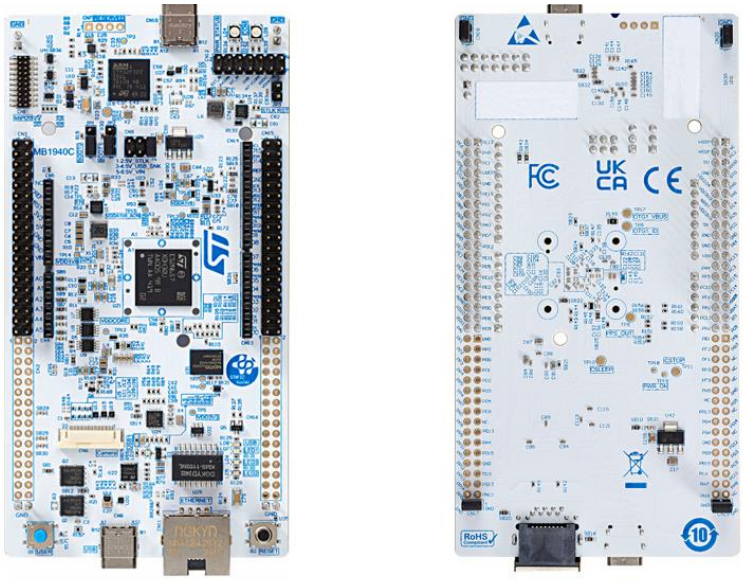


Figure 17: NUCLEO-N657X0-Q top view(left view) . NUCLEO-N657X0-Q (bottom view)

4.3 ATI320 Thermal Camera

The ATI320 camera provides high-resolution thermal imaging.

Specifications:

- Resolution: 320 × 240
- Output: RAW14 (14-bit)
- Frame rate: up to 60 FPS

Figure 18: ATI320 Camera

5. Methodology

5.1 System Overview

The system integrates an ATI320 thermal camera with an STM32N6 Nucleo-144 development board, enabling fast processing of high resolution thermal data.

The main objective of system is to convert RAW thermal data into visually interpretable images while preserving important features such as multiple heat sources and background details. To achieve this, a multi-stage processing pipeline was implemented, including linear compression, adaptive gain control and dynamic range compression.

5.2 Software Architecture and Task Design

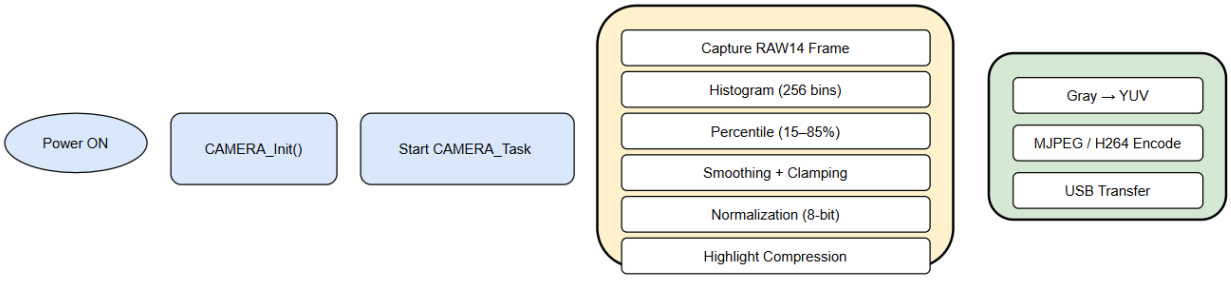


Figure 19 : Low Level Code Flow of system showing the complete pipeline from RAW14 acquisition to thermal visualization.

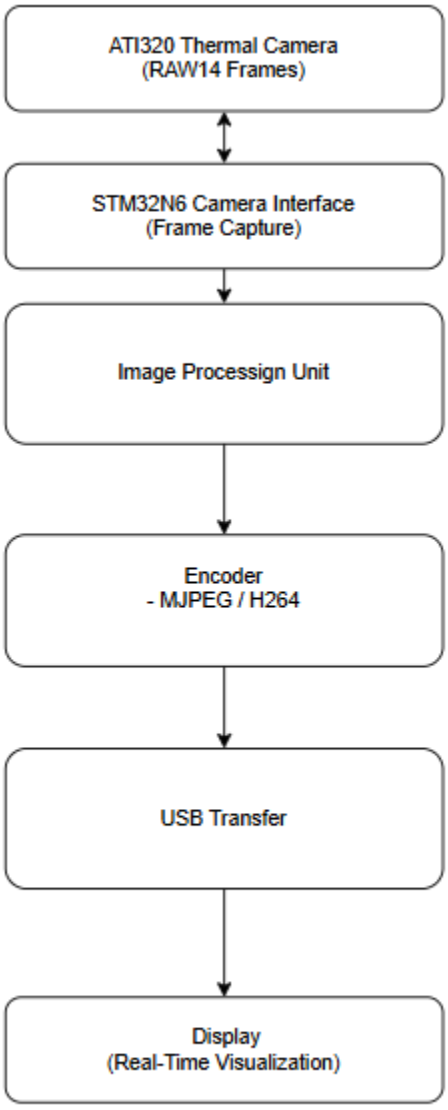


Figure 20 : High Level Block Diagram of the system.

5.3 Thermal Data Acquisition

The infrared camera provides thermal data in RAW14 format where each pixel is represented by 14 bits. The data is received as a byte and combine to obtain each RAW14 pixel value.

The extraction of RAW14 pixel values is implemented as:

```
uint16_t raw = ((uint16_t)p[2*i+1] << 8) | p[2*i];
raw &= 0x3FFF;
```

In above step it combines two bytes into a 16-bit value and masks the lower 14 bits, ensuring only valid thermal data is used.

5.4 Development of Image Processing Algorithm

During the development of image processing algorithm, multiple methods were tested to improve thermal visualization.

Initial Approach: Direct Downscaling

This approach reduced the 14-bit thermal data to 8-bit grayscale values using a simple bit-shift operation. Although this method was simple and fast, the generated images had very low contrast and limited thermal detail.

A simple bit-shift operation was used:

```
uint8_t gray = (raw & 0x3FFF) >> 6;
```

Min–Max Scaling

A linear scaling approach was applied where the thermal intensity values were adjusted according to the minimum and maximum intensity levels present in each frame.

```
uint8_t gray = (uint8_t)((raw - min) * 255 / (max - min));
```

Although this improved contrast, it causes major limitation as when a hot object place in the scene, it dominated the entire image, causing other objects to become nearly invisible.

Clipped Scaling

To address this issue, this method is tested by removing extreme values from the range. This provided some improvement but was still not sufficient for stable visualization in dynamic scenes.

Final Approach: Histogram-Based Adaptive Gain Control

To improve the thermal imaging visualization, an adaptive histogram based method was tested. Instead of depending only on the minimum and maximum intensity values, the method evaluates the overall intensity distribution of the image.

5.5 Histogram-Based Enhancement

In the implemented method, the RAW14 data is first reduced into 256 intensity levels to build a histogram representing the distribution of pixel values in each frame as shown in code snippet below:

```
uint16_t hist[256] = {0};


for(uint32_t i = 0; i < pixels; i++)
{
  uint16_t raw = ((uint16_t)p[2*i+1] << 8) | p[2*i];
  raw &= 0x3FFF;

  uint8_t bin = raw >> 6;
  hist[bin]++;
}
```

This converts RAW14 data into a 256-bin histogram.

Instead of using absolute minimum and maximum values, percentile-based thresholds are used. Specifically, lower and upper bounds are selected based on the percentage of pixels within the frame (e.g., 15%–85%).

The lower and upper bounds are computed using cumulative histogram counts:

```
uint32_t low_count  = total * 15 / 100;
uint32_t high_count = total * 85 / 100;
```

Then the corresponding bins are found:

```
for(int i = 0; i < 256; i++)
{
   sum += hist[i];
   if(sum >= low_count)
   {
      low_bin = i;
      break;
   }
}
```

This ensures that:

- Extreme values do not dominate the image
- Most of the scene information is preserved
- Multiple thermal objects can be visualized simultaneously

5.6 Temporal Stability

To avoid sudden brightness fluctuations between frames, temporal smoothing is applied to the calculated minimum and maximum values. Instead of updating them directly, a weighted averaging method is used.

```
min = (min * 7 + cur_min) / 8;
max = (max * 7 + cur_max) / 8;
```

This significantly improves visual stability and prevents flickering, especially in dynamic environments.

5.7 Dynamic Range Compression

This method is used because the main challenge during testing was that very hot objects could still dominate the image especially when located close to the camera.

In this method, high-intensity values are selectively compressed after normalization. Instead of allowing bright regions to saturate, their intensity growth is reduced, which prevents them from overwhelming the rest of the scene.

```
uint32_t norm = (raw - min) * 255 / (max - min);


if(norm > 180)
{
   norm = 180 + (norm - 180) / 3;
}
```

As a result:

- Hot objects remain visible but not dominant
- Background details are preserved
- Multiple objects at different temperatures can be observed clearly

This ste shows key improvement in the system and makes overall visualization better.

5.8 Grayscale Conversion and Visualization

After processing, the thermal data is converted into an 8-bit grayscale image. This image represents the normalized thermal intensity and serves as the base for visualization.

```
p8[i] = (uint8_t)norm;
```

For enhanced interpretability, an optional **color mapping function** can be applied, converting grayscale values into a pseudo-color thermal palette.

The grayscale image is converted into YUV format:

```
res = COLOR_Gray8_to_YUV(&Buffer_CameraPreEncode, &Buffer_Temp);
```

Optional thermal color mapping can be applied:

```
res = COLOR_test(&Buffer_CameraPreEncode);

//  COLOR PALETTE

                /*

                res = COLOR_test(&Buffer_CameraPreEncode);

                if (res) {

                    CAM_ReportFail1("[CAM] COLOR_test() fail", res);

                    continue;

                }
```

This improves the ability of users to distinguish temperature differences visually.

5.9 Data Conversion and Streaming

The processed grayscale image is converted into YUV format for compatibility with video streaming standards. The system supports real-time transmission via USB and Ethernet.

The processed frame is transmitted via USB:

```
tud_video_n_frame_xfer(0, 0, (void*) buff->data, buff->data_len);
```

For compressed streaming:

```
VENC_JpegEncodeFrame(&Buffer_CameraJpeg, &Buffer_CameraPreEncode);
```

Depending on the configuration, frames can be encoded using MJPEG or H.264 before transmission.

5.10 System Validation and Debugging

To verify the performance of the implemented algorithms, debugging tool is used. Key parameters such as minimum and maximum intensity values, percentile limits, and pixel values were monitored during runtime.

Debugging was performed using runtime logs:

```
printf("LOW=%u HIGH=%u MIN=%u MAX=%u RAW=%u GRAY=%u\r\n",
low_bin, high_bin, min, max, debug_raw, debug_gray);
```

These debug outputs helped in:

- Validating algorithm behavior
- Tuning parameters (e.g., percentile range)
- Ensuring stable real-time streaming.

6 Experimental Results

Experiments were conducted under different conditions to test performance of the implemented image processing algorithms. The test scenarios included:

- Single heat source (e.g., hot cup)
- Multiple heat sources at equal distance
- Objects at different distances (foreground vs background)
- Human face combined with hot objects
- Indoor ambient environment

These scenarios were selected to test the system's ability to handle dynamic range variations and maintain visibility across different temperature levels.

6.1 Evaluation of Initial Approaches

6.1.1 Direct RAW14 Downscaling

The initial implementation used simple bit-shifting to convert RAW14 data into 8-bit grayscale.

Observation:

- Very low contrast
- Poor visibility of thermal details
- Background and objects were not distinguishable

This approach was computationally efficient but unsuitable for practical visualization.

6.1.2 Min–Max Scaling

Linear min–max normalization was implemented to improve contrast.

Observation:

- Improved brightness distribution
- Clear visibility of a single heat source

However, a major limitation was observed:

When a high-temperature object (e.g., a hot cup) was introduced, it dominated the image.

Other objects, such as a human face, became much darker and less visible.

This confirmed that min–max scaling is highly sensitive to outliers.

c 

Figure 21: Visual representation of hot object dominating the scene

6.2.3 Clipped Scaling

To reduce the effect of extreme values, clipped scaling was tested.

Observation:

- Slight improvement over min–max
- Reduced impact of extreme values

However:

- Still unstable in dynamic scenes
- Did not fully solve multi-object visibility

6.3 Performance of Histogram-Based Approach

The final implementation applied to histogram based enhancement approach using percentile clipping together with adaptive gain control.

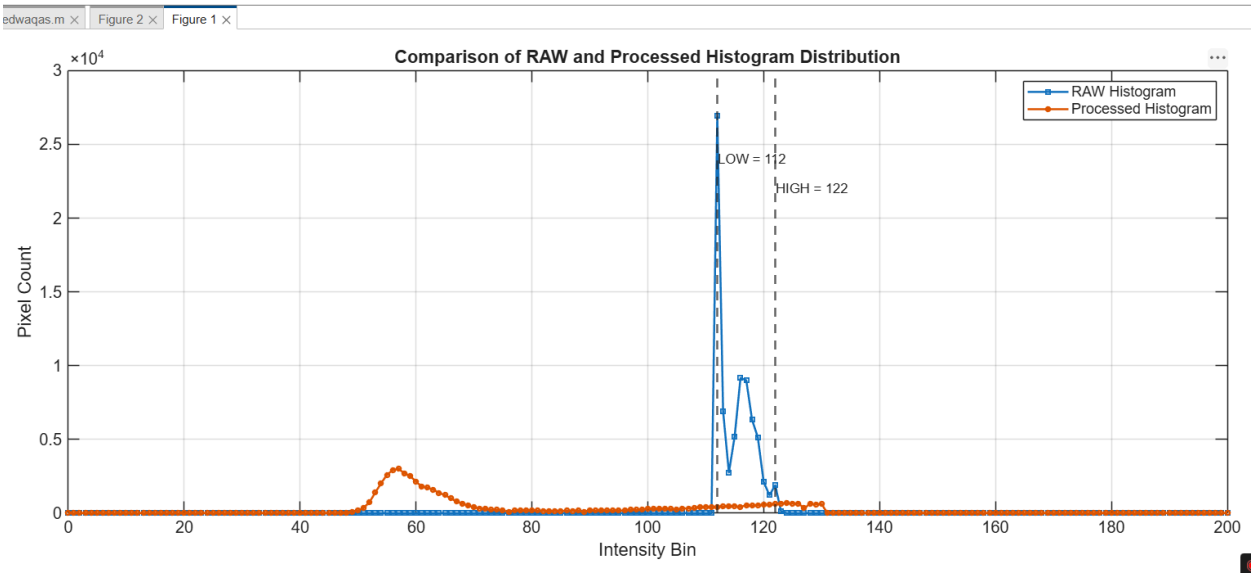


Figure 22 : Comparison of RAW and Processed Histogram Distribution

Figure 22 compares the original RAW histogram with the processed histogram after image enhancement. In the original RAW data most intensity values were grouped within a small range which resulted in low contrast and poor visibility of thermal details. After processing, the intensity values became more spread across the available range, making different thermal regions easier to distinguish. This improvement helped produce clearer thermal images and allowed multiple objects in the scene to remain visible at the same time.

Observation:

- Multiple objects became clearly visible simultaneously
- Background details were preserved
- Reduced influence of extreme temperature values

In scenarios where two heated objects were placed at equal distance, both objects were clearly distinguishable, unlike in previous methods.

6.4 Effect of Percentile Clipping

Different percentile ranges were tested to evaluate performance.

Case 1: Wide Range (1%–99%)

- High dynamic range preserved
- Slight dominance of hot objects

Case 2: Moderate Range (15%–85%)

- Best balance between contrast and stability
- Multiple objects clearly visible
- Background not suppressed

This range was selected as the optimal configuration.

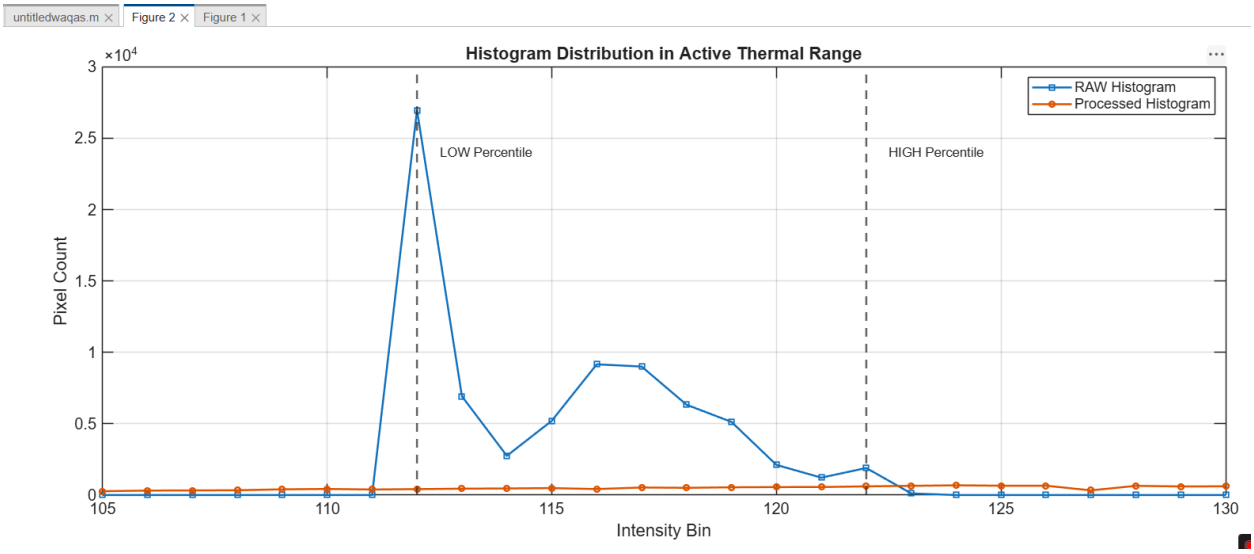


Figure 23 — Histogram Distribution in Active Thermal Range

Figure 23 shows the active thermal range selected by the percentile clipping algorithm. The low and high percentile boundaries automatically exclude extreme outlier values while keeping the majority of thermal data which enables stable contrast enhancement and reduces dominance of extremely hot regions.

6.5 Dynamic Range Compression Results

During testing, it was observed that object distance had a strong effect on the thermal image intensity. When a hot object, such as a cup, was placed closer to the camera, it is little brighter than surrounding objects, including the human face. Because of this, the face appeared darker and some background details were less visible even after histogram based enhancement.

To improve the balance of the thermal image, dynamic range compression was applied to reduce the dominance of very bright regions.

After Compression:

- Bright regions were controlled
- Background objects remained visible
- Balanced visualization across different distances

Result:

Figure 24: Visual Representation of hot object and human subject monitoring

- The system successfully displayed both near and far objects
- Reduced dominance of very hot or close objects
- Improved overall image perception

This represents a significant improvement over traditional approaches.

6.6 Application: PCB Thermal Analysis and Fault Detection

By creating a short circuit across diode near the power supply area to produce abnormal heating, which is a common issue in electronic board failures such as short circuit components or damaged power paths.

The infrared camera successfully detected clear thermal hotspot in the affected area which showed noticeable temperature rise around the faulty diode while the other regions of the PCB remained relatively cool which is shown in Figure 25.

Overall this testing proves the system ability to detect abnormal thermal behaviour in electronic circuit fault detection.

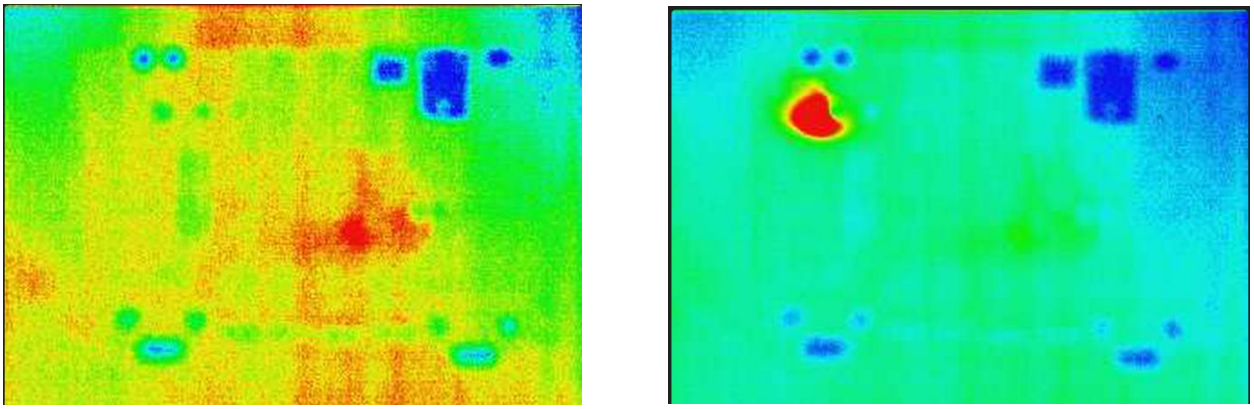


Figure 25: Hotspot detected near short circuit region(on the right)

Effect of Image Processing

The implemented histogram-based AGC and dynamic range compression played a critical role in improving visualization:

- Prevented overheating components from dominating the image
- Preserved visibility of surrounding components
- Enhanced contrast between active and inactive regions

Without these enhancements, smaller thermal variations would have been difficult to observe.

Fault Detection Potential

The system demonstrated the capability to identify abnormal thermal patterns, which can indicate:

- Overheating components
- Possible short circuits
- Inefficient heat dissipation

This highlights the potential of the system for non-contact diagnostics and preventive maintenance in electronic systems.

6.7 Performance Comparison of algorithms

The performance of different methods can be shown in Table2:

Method	Performance
RAW Downscaling	Poor contrast
Min–Max Scaling	Good for single object detection
Clipped Scaling	Moderate improvement
Histogram AGC	Slight improvement in multi-object visibility
Histogram + Compression (Final)	Best overall performance

Table 2: Comparison of Different Methods based on performance

6.8 Debug Validation

To validate the correctness of the implemented algorithm, debugging was performed. Parameters such as minimum and maximum values, histogram limits, and pixel intensities were monitored.

Example debug output:

```
LOW=XX HIGH=XX MIN=XXXX MAX=XXXX RAW=XXXX GRAY=XXX
```

These logs confirmed:

- Correct histogram behavior
- Stable min/max adaptation
- Proper scaling of pixel values

7. Conclusion

In this work, embedded thermal imaging solution was developed using the STM32N6 microcontroller and ATI320 thermal camera. Histogram based image enhancement approach with percentile clipping and dynamic range compression was implemented to improve thermal visualization and final system provides improved contrast, better visualization of multiple thermal objects, and stable performance during human subject testing. Experimental results show successful detection of thermal hotspots on PCB, demonstrating the practical applications of the system.

It also extends the initial work based on the MLX90640 thermal sensor array. In that work, low-resolution thermal monitoring was successfully used for PCB temperature analysis and basic fault detection. Experimental results demonstrated that thermal hotspots generated by faulty electronic components could be identified.

Overall, the developed system demonstrates that advanced thermal image enhancement can be successfully implemented on an embedded platform while maintaining efficient real-time operation for monitoring and diagnostic applications.

8.References

[1] Melexis (n.d.) White Paper: MLX90620 for Smart and Green HVAC Applications.

[2] Micro-Epsilon (n.d.) Infrared Temperature Measurement Basics.

[3] Branchitta, F., Diani, M., Corsini, G. and Porta, A. (2007) 'Dynamic Range Compression in Infrared Imaging'. doi:10.1117/1.2956655

[4] Firmansyah, H. et al. (2023) 'IoT-Based Thermal Monitoring Using MLX90640'. doi:10.26418/positron.v15i1.89878

[5] Zakaria, M.F., Ibrahim, H. and Suandi, S.A. (2014) 'A Review: Image Compensation Techniques'. doi:10.1109/ICCET.2010.5485499

[6] Gaber, R., Ali, A. and Ahmed, K. (2022) 'Performance Evaluation of Infrared Image Enhancement Techniques', International Journal of Advances in Computer Science and Technology, 11(2). doi:10.30534/ijacst/2022/011122022.

[7] Li, S., Jin, W., Li, J. and Li, Y. (2018) 'An Improved Contrast Enhancement Algorithm for Infrared Images Based on Adaptive Double Plateaus Histogram Equalization', Infrared Physics & Technology, 90, pp. 164–174. doi:10.1016/j.infrared.2018.03.010.

[8] Pizer, S.M., Amburn, E.P., Austin, J.D., Cromartie, R., Geselowitz, A., Greer, T., ter Haar Romeny, B., Zimmerman, J.B. and Zuiderveld, K. (1987) 'Adaptive Histogram Equalization and Its Variations', Computer Vision, Graphics, and Image Processing, 39(3), pp. 355–368.doi: 10.1016/S0734-189X(87)80186-X

[9] Wu, Y. et al. (2019) 'Prediction of Thermal Sensation Using Low-Cost IR Sensors', IOP Conference Series. doi: 10.1088/1757-899X/609/3/032002

[10] Pizer, S.M., Johnston, R.E., Ericksen, J.P., Yankaskas, B.C. and Muller, K.E. (1990)

Contrast-Limited Adaptive Histogram Equalization: Speed and Effectiveness.

Medical Image Display Research Group, University of North Carolina, Chapel Hill, NC, USA.

[11] Wang, L. and Yan, J. (2012) 'Method of infrared image enhancement based on histogram',

Journal of Infrared and Millimeter Waves,31(1), pp. 74–78. doi:10.1007/s11801-011-9085-3.

[12] Gonzalez, R.C. and Woods, R.E. (2018) Digital Image Processing. 4th edn. Pearson.

[13] Vollmer, M. and Möllmann, K.-P. (2017) Infrared Thermal Imaging: Fundamentals, Research and Applications. 2nd edn. Wiley-VCH.

[14] Melexis (2024) MLX90640 Far Infrared Thermal Sensor Array Datasheet.

[15] Espressif Systems (2024) ESP-IDF Programming Guide.

[16] STMicroelectronics (2024) STM32N6 Series Reference Manual.

[17] Guide Sensmart (2024) ATI320 Thermal Camera User Manual.

Faculty of Physics

Institute of Applied Electrodynamics and Telecommunications

Waqas Ahmad

Application of infrared thermal camera in Microcontroller based systems

Supervisor: Assoc.Prof. Vytautas Jonkus

Summary

In this work the main focus was on development of an embedded infrared thermal imaging system for monitoring and diagnostic applications. The work was carried out in several stages, starting with a low cost thermal monitoring solution based on the MLX90640 infrared sensor and ESP32 microcontroller. In this stage, thermal data acquisition, wireless transmission, and real time visualization through a web interface were successfully implemented.

The system used for printed circuit board (PCB) thermal analysis and fault detection. Experimental testing showed that thermal imaging could be used to identify abnormal behaviour and detect fault conditions such as short circuits. The results also showed the influence of emissivity, reflective surfaces, sensor distance, and sensor resolution on measurement accuracy.

In the final system it was upgraded using the STM32N6 microcontroller and ATI320 thermal camera due to the higher resolution and RAW14 image output of the camera, a complete image processing pipeline was developed. Multiple approaches were investigated, including direct downscaling, min-max scaling, clipped scaling, and histogram-based enhancement. The final implementation combined percentile-based histogram enhancement, temporal smoothing, and dynamic range compression to improve thermal image quality.The results shows thate proposed method gives better contrast, improved visibility of multiple thermal objects and reduced the dominance of very hot region.

Overall, the results of this work show that advanced thermal image enhancement techniques can be successfully implemented and the final solution shows improved thermal image quality, making it suitable for monitoring and diagnostic applications.

Faculty of Physics

Institute of Applied Electrodynamics and Telecommunications

Waqas Ahmad

Application of infrared thermal camera in Microcontroller based systems

Supervisor: Assoc.Prof. Vytautas Jonkus

Santrauska

Šiame darbe pagrindinis dėmesys buvo skirtas įterptinės infraraudonųjų spindulių terminio vaizdinimo sistemos kūrimui stebėsenos ir diagnostikos taikymams. Darbas buvo vykdomas keliais etapais, pradedant nebrangiu terminio stebėjimo sprendimu, pagrįstu MLX90640 infraraudonųjų spindulių jutikliu ir ESP32 mikrovaldikliu. Šiame etape buvo sėkmingai įgyvendintas šiluminių duomenų surinkimas, belaidis duomenų perdavimas bei realaus laiko vizualizavimas internetinėje sąsajoje.

Vėliau sistema buvo pritaikyta spausdintinių plokščių (PCB) šiluminei analizei ir gedimų aptikimui. Eksperimentiniai tyrimai parodė, kad terminis vaizdinimas gali būti naudojamas nenormaliam kaitimui nustatyti ir tokiems gedimams kaip trumpieji jungimai aptikti. Gauti rezultatai taip pat atskleidė paviršiaus emisijos koeficiento, atspindinčių paviršių, jutiklio atstumo ir jutiklio skiriamosios gebos įtaką temperatūros matavimų tikslumui.

Baigiamajame darbo etape sistema buvo atnaujinta naudojant STM32N6 mikrovaldiklį ir ATI320 terminę kamerą. Dėl didesnės kameros skiriamosios gebos ir RAW14 formato vaizdų išvesties buvo sukurta pilna vaizdo apdorojimo grandinė. Buvo ištirti keli metodai, įskaitant tiesioginį mastelio mažinimą, min–max mastelio keitimą, ribotą mastelio keitimą ir histograma pagrįstą vaizdo gerinimą. Galutiniame sprendime buvo sujungtas procentilėmis pagrįstas histogramų gerinimas, laikinis glodinimas ir dinaminio diapazono suspaudimas, siekiant pagerinti terminio vaizdo kokybę.

Eksperimentiniai rezultatai parodė, kad pasiūlytas metodas užtikrina geresnį kontrastą, pagerina kelių šiluminių objektų matomumą, sumažina labai karštų sričių dominavimą ir išlaiko stabilų vaizdo atvaizdavimą realiuoju laiku. Be to, sistema sėkmingai aptiko elektroninių grandinių šiluminius karštuosius taškus, taip parodydama savo potencialą bekontakčio stebėjimo, diagnostikos ir gedimų aptikimo srityse.

Apibendrinant galima teigti, kad sukurta sistema parodė, jog pažangūs terminio vaizdo gerinimo metodai gali būti sėkmingai įgyvendinti įterptinėje platformoje, išlaikant efektyvų veikimą realiuoju laiku stebėsenos ir diagnostikos uždaviniams spręsti

11. Appendix

- ## Main Code block: STM32+ATI320

Histogram Final Approach

```
uint8_t *p  = Buffer_CameraInput.data;
            uint8_t *p8 = Buffer_Temp.data;

            uint32_t pixels = Buffer_CameraInput.image_width *
Buffer_CameraInput.image_height;

            //  SAFETY
            if(p == NULL || p8 == NULL)
            {
                CAM_ReportFail("[CAM] NULL buffer");
                continue;
            }

            //  HISTOGRAM
            uint32_t hist[256] = {0};

            // FINAL OUTPUT HISTOGRAM (FOR GRAPH)
            uint32_t hist_final[256] = {0};

            for(uint32_t i = 0; i < pixels; i++)
            {
                uint16_t raw = ((uint16_t)p[2*i+1] << 8) | p[2*i];
                raw &= 0x3FFF;

                uint8_t bin = raw >> 6;   // 14-bit to 8-bit

                hist[bin]++;
            }

            //  PERCENTILE (1% - 99%)
            uint32_t total = pixels;

            uint32_t low_count  = total * 1 / 100;
            uint32_t high_count = total * 99 / 100;
```

```
            uint32_t sum = 0;

            uint8_t low_bin = 0;
            uint8_t high_bin = 255;

            // LOW
            for(int i = 0; i < 256; i++)
            {
                sum += hist[i];

                if(sum >= low_count)
                {
                    low_bin = i;
                    break;
                }
            }

            // HIGH
            sum = 0;

            for(int i = 255; i >= 0; i--)
            {
                sum += hist[i];

                if(sum >= (total - high_count))
                {
                    high_bin = i;
                    break;
                }
            }

            //  RAW RANGE
            uint16_t cur_min = ((uint16_t)low_bin) << 6;
            uint16_t cur_max = ((uint16_t)high_bin) << 6;

            //   SMOOTHING
            static uint16_t min = 0;
```

```
            static uint16_t max = 0;

            if(min == 0 && max == 0)
            {
                min = cur_min;
                max = cur_max;
            }
            else
            {
                min = (min * 7 + cur_min) / 8;
                max = (max * 7 + cur_max) / 8;
            }

            //  MIN RANGE PROTECTION
            if((max - min) < 1000)
            {
                uint16_t mid = (min + max) / 2;

                min = mid - 500;
                max = mid + 500;
            }

            // safety
            if(max <= min)
            {
                max = min + 1;
            }

            //  DEBUG
            static uint32_t frame_cnt = 0;
            frame_cnt++;

            uint16_t debug_raw = 0;
            uint8_t  debug_gray = 0;

            //  SCALING + GAMMA
            for(uint32_t i = 0; i < pixels; i++)
            {
```

```
                uint16_t raw = ((uint16_t)p[2*i+1] << 8) | p[2*i];

                raw &= 0x3FFF;

                // clamp
                if(raw < min)
                    raw = min;

                if(raw > max)
                    raw = max;

                uint32_t norm =
                    (raw - min) * 255 / (max - min);

                // compress highlights
                if(norm > 180)
                {
                    norm = 180 + (norm - 180) / 3;
                }

                uint8_t gray = (uint8_t)norm;

                // FINAL HISTOGRAM
                hist_final[gray]++;

                p8[i] = gray;

                if(i == 0)
                {
                    debug_raw = raw;
                    debug_gray = gray;
                }
            }

            // DEBUG PRINT + GRAPH EXPORT
            if(frame_cnt % 300 == 0)
            {
                printf("\r\nGRAPH_START\r\n");
```

```
                printf("BIN,RAW,PROCESSED\r\n");

                for(int i = 0; i < 256; i++)
                {
                    printf("%d,%u,%u\r\n",
                            i,
                            hist[i],
                            hist_final[i]);
                }

                printf("GRAPH_END\r\n");

                printf("LOW=%u HIGH=%u MIN=%u MAX=%u RAW=%u
GRAY=%u\r\n",
                        low_bin,
                        high_bin,
                        min,
                        max,
                        debug_raw,
                        debug_gray);
            }

            //  OUTPUT BUFFER
            Buffer_Temp.color = BUFCOLOR_GRAY_8BITS;

            Buffer_Temp.data_len = pixels;

            Buffer_Temp.image_width  =
Buffer_CameraInput.image_width;
            Buffer_Temp.image_height =
Buffer_CameraInput.image_height;

            //  PIPELINE
            res = COLOR_Gray8_to_YUV(
                    &Buffer_CameraPreEncode,
                    &Buffer_Temp);
```

```
            if (res)
            {
                CAM_ReportFail1(
                    "[CAM] COLOR_Gray8_to_YUV() fail",
                    res);

                continue;
            }

            //  COLOR PALETTE
            /*
            res = COLOR_test(&Buffer_CameraPreEncode);

            if (res)
            {
                CAM_ReportFail1(
                    "[CAM] COLOR_test() fail",
                    res);

                continue;
            }
*/
            // FINAL CONVERSION
            res = COLOR_UsbColorConversion(
                    &Buffer_CameraPreEncode,
                    &Buffer_CameraPreEncode);

            CameraStatus &= ~CAM_FRAME_READY;

            if (res)
            {
                CAM_ReportFail1(
                    "[CAM] UsbColorConversion() fail",
                    res);

                continue;
            }
```

- Full Source Code Final Work STM32+ATI320

  https://mega.nz/folder/83wmXC4J#SJsLNuqonVlS

  -ill5dU2sg

- Full Source Code Initial Work

  MLX90640+ESP32

  https://mega.nz/folder/82YR2Czb#dBWVx3lj8pGdt

  BBFFVirCw

- Video Demonstration of Previous Results output

  https://drive.google.com/file/d/1yrNNmd5ywFdyJs64GG6VnKzbZB1tegsU/view?usp=drive_lin k

- Debugging Data for figures20,21

```
GRAPH_START
BIN,RAW,PROCESSED0,0,01,0,02,0,03,0,04,0,05,0,06,0,0
7,0,08,0,09,0,010,0,011,0,012,0,013,0,014,0,015,0,016,0,017,0,0
18,0,019,0,020,0,021,0,022,0,023,0,024,0,025,0,026,0,027,0,028,0,0
29,0,030,0,031,0,032,0,033,0,034,0,035,0,036,0,037,0,038,0,039,0,04
0,0,041,0,042,0,043,0,144,0,045,0,0
46,0,347,0,648,0,2049,0,5150,0,17251,0,36952,0,75053,0,1400
54,0,203255,0,259356,0,291857,0,301558,0,269659,0,251560,0,2154
61,0,179262,0,172963,0,159764,0,136865,0,123666,0,100167,0,806
68,0,64269,0,52170,0,41071,0,32572,0,29373,0,21674,0,22375,0,19676,
0,8877,0,17278,0,18079,0,18080,0,17481,0,16982,0,15883,0,142
84,0,15185,0,15186,0,16187,0,15788,0,17889,0,8990,0,18791,0,168
92,0,19293,0,17794,0,17195,0,18096,0,17197,0,22798,0,22399,0,230100
,0,316101,0,29102,0,296103,0,304104,0,275105,0,268106,0,313
```

107,0,324108,0,336109,0,399
110,0,426 111,16,387 112,26966,415 113,6918,447 114,2747,463
115,5204,487 116,9158,421 117,9003,528 118,6326,506 119,5123,542
120,2112,558 121,1219,562 122,1895,608 123,113,647 124,0,681
125,0,655 126,0,655 127,0,327 128,0,646 129,0,603 130,0,611
131,0,584 132,0,557 133,0,595 134,0,636 135,0,571 136,0,537
137,0,573 138,0,571 139,0,636 140,0,295 141,0,621 142,0,546
143,0,499 144,0,525 145,0,460 146,0,471 147,0,424 148,0,441
149,0,443 150,0,432 151,0,433 152,0,451 153,0,449 154,0,419
155,0,395 156,0,423 157,0,350 158,0,341 159,0,328 160,0,336
161,0,318 162,0,285 163,0,314 164,0,316 165,0,294 166,0,301
167,0,324 168,0,317 169,0,349 170,0,324 171,0,342 172,0,402
173,0,399 174,0,357 175,0,358 176,0,289 177,0,223 178,0,105
179,0,180 180,0,463 181,0,447 182,0,450 183,0,317 184,0,260
185,0,218 186,0,172 187,0,199 188,0,226 189,0,320 190,0,273
191,0,332 192,0,351 193,0,420 194,0,418 195,0,268 196,0,63 197,0,26
198,0,11 199,0,2 200,0,0 201,0,0 202,0,0 203,0,0 204,0,0 205,0,0
206,0,0 207,0,0 208,0,0 209,0,0 210,0,0 211,0,0 212,0,0 213,0,0
214,0,0 215,0,0 216,0,0 217,0,0 218,0,0 219,0,0 220,0,0 221,0,0
222,0,0 223,0,0 224,0,0 225,0,0 226,0,0 227,0,0 228,0,0 229,0,0
230,0,0 231,0,0 232,0,0 233,0,0 234,0,0 235,0,0 236,0,0 237,0,0
238,0,0 239,0,0 240,0,0 241,0,0 242,0,0 243,0,0 244,0,0 245,0,0
246,0,0 247,0,0 248,0,0 249,0,0 250,0,0 251,0,0 252,0,0 253,0,0
254,0,0 255,0,0
GRAPH_END
LOW=112 HIGH=122 MIN=6977 MAX=7977 RAW=7306 GRAY=83