Human-computer interaction tool development for skin cancer diagnosis on a low-cost platform

Desarrollo de una herramienta de interacción humano-computador para el diagnóstico de cáncer de piel en una plataforma de bajo costo

Desenvolvimento de uma ferramenta de interação humano-computador para diagnóstico de câncer de pele em uma plataforma de baixo custo

Carlos Vicente Niño-Rondón[1]
Sergio Alexander Castro-Casadiego[2]

Received: August 10th, 2025
Accepted: November 20th, 2025
Available: January 13th, 2026

How to cite this article:
C. V. Niño-Rondón and S. A. Castro-Casadiego, "Human-computer interaction tool development for skin cancer diagnosis on a low-cost platform," *Revista Ingeniería Solidaria,* vol. 22, no. 1, 2026.
doi: https://doi.org/10.16925/2357-6014.2026.01.07

Research article. https://doi.org/10.16925/2357-6014.2026.01.07

[1] Departamento de Electricidad y Electrónica. Facultad de Ingeniería. Universidad Francisco de Paula Santander, Cúcuta, Colombia.

E-mail (Carlos Vicente Niño-Rondón): carlosvicentenr@ufps.edu.co.

ORCID: https://orcid.org/0000-0002-3781-4564

CvLAC: https://scienti.minciencias.gov.co/cvlac/visualizador/generarCurriculoCv.do?-cod_rh=0001703664

[2] Departamento de Electricidad y Electrónica. Facultad de Ingeniería. Universidad Francisco de Paula Santander, Cúcuta, Colombia.

E-mail (Sergio Alexander Castro-Casadiego): sergio.castroc@ufps.edu.co.

ORCID: https://orcid.org/0000-0003-0962-9916

CvLAC: https://scienti.minciencias.gov.co/cvlac/visualizador/generarCurriculoCv.do?-cod_rh=0001132598

Abstract

Introduction: This paper is the result of the research project "Deep learning for the identification of non-melanocytic lesions on dermoscopic images," conducted at Francisco de Paula Santander University in 2024. A computer-aided diagnostic tool is proposed for skin cancer classification, aimed at facilitating the automatic analysis of skin lesions on low-cost platforms.

Methods: A graphical user interface was developed in Python, integrating a convolutional neural network model implemented on a Raspberry Pi. The system's performance was evaluated through real-world testing.

Results: The average diagnosis time was 13.6 seconds. Mean CPU usage was 32.65%, RAM usage reached 46.70%, and processor temperature peaked at 55 °C under maximum load.

Conclusions: The system operates effectively on embedded devices, with good performance and moderate resource consumption, making it suitable for resource-constrained environments.

Originality: The authors declare that the work is original and unpublished.

Limitations: The study focuses on the development of the human-computer interaction tool, not on the development of the computational model.

Keywords: Skin cancer, artificial intelligence, aided diagnosis, open source, low cost, embedded systems.

Resumen

Introducción: Se propone una herramienta de diagnóstico asistido por computador para la clasificación de cáncer de piel, orientada a facilitar el análisis automático de lesiones cutáneas en plataformas de bajo costo.

Métodos: Se desarrolló una interfaz gráfica en Python que integra un modelo de clasificación basado en redes convolucionales, implementado en una Raspberry Pi. Se evaluó el rendimiento del sistema mediante pruebas de uso real.

Resultados: El tiempo promedio de diagnóstico fue de 13.6 segundos. El uso medio de CPU fue de 32.65%, la RAM alcanzó 46.70% y la temperatura del procesador llegó hasta 55 °C en carga máxima.

Conclusiones: El sistema es funcional en dispositivos embebidos, con buen desempeño y consumo moderado de recursos, lo que lo hace apto para entornos con limitaciones tecnológicas.

Originalidad: Los autores declaran que el trabajo es original e inédito.

Limitaciones: El estudio se centra en el desarrollo de la herramienta de interacción humano-computador, y no en el desarrollo del modelo computacional.

Palabras clave: cáncer de piel, inteligencia artificial, diagnóstico asistido, código abierto, bajo costo, sistemas embebidos.

Resumo

Introdução: Propõe-se uma ferramenta de diagnóstico assistido por computador para classificação de câncer de pele, visando facilitar a análise automática de lesões cutâneas em plataformas de baixo custo.

Métodos: Uma interface gráfica foi desenvolvida em Python, integrando um modelo de classificação baseado em redes neurais convolucionais, implementado em um Raspberry Pi. O desempenho do sistema foi avaliado por meio de testes de uso em situações reais.

Resultados: O tempo médio de diagnóstico foi de 13,6 segundos. O uso médio da CPU foi de 32,65%, o uso da RAM atingiu 46,70% e a temperatura do processador chegou a 55 °C sob carga máxima.

Conclusões: O sistema é funcional em dispositivos embarcados, com bom desempenho e consumo moderado de recursos, tornando-o adequado para ambientes com limitações tecnológicas.

Originalidade: Os autores declaram que o trabalho é original e inédito.

Limitações: O estudo concentra-se no desenvolvimento da ferramenta de interação humano-computador e não no desenvolvimento do modelo computacional.

Palavras-chave: câncer de pele, inteligência artificial, diagnóstico assistido, código aberto, baixo custo, sistemas embarcados.

1. INTRODUCTION

Skin cancer is a disease characterized by the abnormal growth of skin cells [1], typically resulting from prolonged and cumulative exposure to ultraviolet (UV) radiation [2], primarily from the sun or from artificial sources like tanning beds [3]. It is classified into various types, the most common being basal cell carcinoma, squamous cell carcinoma, and melanoma, the latter having a greater potential for metastasis and clinical severity. The early detection of these lesions is fundamental for effective treatment, as the prognosis is usually favorable in initial stages [4]. However, visual diagnosis can be challenging due to the morphological variability of the lesions and their similarity to other benign skin conditions. Therefore, the development of complementary diagnostic methods, especially those incorporating automated image analysis, has gained importance in clinical settings as a supporting tool for medical decision-making [5], [6].

Computer-Aided Diagnosis (CAD) refers to the use of computational systems designed to support healthcare professionals in the detection, evaluation, and classification of pathologies based on clinical data and medical images [7], [8]. These systems integrate digital processing algorithms, machine learning, and, particularly, deep neural networks to identify relevant patterns that might go unnoticed in conventional visual assessments [9], [10]. In the specific case of dermatological diseases [11], CAD has demonstrated utility in the analysis of skin lesion images, facilitating the differentiation between benign and malignant lesions based on morphological, textural, and color features [12], [13]. While CAD does not replace clinical judgment, its incorporation allows for a reduction in the margin of diagnostic error, the standardization of evaluations, and improved efficiency in high-workload settings or those with limited availability of specialists [14]. The implementation of these tools on embedded devices represents a step toward portability and access to real-time diagnostic solutions in diverse clinical contexts [15].

Low-cost platforms, such as those based on ARM architectures, have consolidated as a viable alternative for implementing technological solutions in resource-constrained settings [16]. These systems offer a favorable balance between cost, energy consumption, and processing capability, making them suitable candidates for deploying real-time embedded applications [17], [18]. In the medical field, their use has expanded into tasks such as monitoring [19], signal acquisition, process automation, and, more recently, the execution of artificial intelligence models for aided diagnosis [20]. Despite their restrictions compared to conventional workstations, these platforms allow for the integration of sensors, cameras, and displays through GPIO, I2C, or SPI interfaces [21], facilitating the development of portable and autonomous devices [22], [23]. Their versatility, commercial availability, and technical support community also represent key advantages for their adoption in applied research projects [24], functional prototyping, and field deployment, especially in rural settings or health centers without advanced computational infrastructure [25].

This article presents the development of a human-computer interaction tool as an aided diagnostic strategy for the classification of skin cancer on a low-cost device. The document describes the process of hardware selection, the design and development of the Graphical User Interface (GUI), and system validation. The tests were executed on a Raspberry Pi embedded board.

2. METHODOLOGY

The proposed methodology for the development of the skin cancer diagnostic tool is structured into three fundamental stages. First, the hardware platform selection is performed using a prioritization matrix, in which relevant technical criteria are defined and assigned a specific weighting. The second stage corresponds to the design of the Graphical User Interface (GUI) [26], which is aimed at facilitating intuitive and accessible interaction with the diagnostic system. Finally, in the third stage, system validation is carried out by evaluating the tool's performance on the selected platform through functional testing.

2.1. TOOL SELECTION

The device selection was performed using the prioritization matrix technique, in which selection parameters were defined and assigned a specific weighting. The assignment of comparative values between parameters was based on the following scale: 10 if a parameter is much more important than the one being compared, 5 if it is more

e-ISSN 2357-6014 / Vol. 22, no. 1 / january-april 2026 / Bogotá D.C., Colombia
Universidad Cooperativa de Colombia

important, 1 if they have the same importance, 1/5 if it is less important, and 1/10 if it is much less important. Once the comparisons were completed, the obtained values for each parameter were summed, thus allowing the calculation of their specific weight within the decision process [27].

Table 1 presents the comparison, where the parameters are denoted as follows: "A" for processing speed, "B" for availability of input/output interfaces, "C" for energy consumption, "D" for heat dissipation capability, and "E" for price and market availability. The results show that the most relevant parameters in the selection of the embedded device for executing the skin cancer classification system are energy consumption (C) and processing speed (A), both with a normalized importance factor of 0.3508. Second in importance are the availability of input/output interfaces (B) and the heat dissipation capability (D), with normalized factors of 0.1403, indicating moderate importance in the decision. Finally, price and market availability (E) was the parameter with the lowest relevance, with a factor of 0.0178, reflecting a comparatively low weight in the selection process.

Table 1. Parameter Weighting for Tool Selection

	A	**B**	**C**	**D**	**E**	**Total**	**Factor**
A	X	5	1	5	5	16	0,3508
B	1/5	X	1/5	1	5	6.4	0,1403
C	1	5	X	5	5	16	0,3508
D	1/5	1	1/5	X	5	6.4	0,1403
E	1/5	1/5	1/5	1/5	X	0.8	0,0178
			Total			45,6	1

Source: own work

In addition, Table 2 presents a comparison between various hardware platforms in relation to parameter "A," corresponding to processing speed. The evaluated boards are: Raspberry Pi 4B [28], BeagleBone Black, NVIDIA Jetson Nano, and Odroid M1s [29]. Among these, the NVIDIA Jetson Nano is positioned as the option with the best performance, obtaining a normalization factor of 0.5446, which reflects its superiority in terms of processing capability. It is followed by the Raspberry Pi 4B, with a value of 0.3311, which also demonstrates noteworthy performance. In contrast, the BeagleBone Black and Odroid M1s platforms obtain significantly lower normalization factors, suggesting a lower level of suitability regarding this specific criterion.

e-ISSN 2357-6014 / Vol. 22, no. 1 / january-april 2026 / Bogotá D.C., Colombia
Universidad Cooperativa de Colombia

Table 2. Comparison Regarding Processing Speed

	Raspberry Pi 4B	BeagleBone Black	NVIDIA Jetson Nano	Odroid M1s	Total	Factor
Raspberry Pi 4B	X	5	1/5	10	15.2	0.3311
BeagleBone Black	1/5	X	1/10	5	5.3	0.1154
NVIDIA Jetson Nano	5	10	X	10	25	0,5446
Odroid M1s	1/10	1/5	1/10	X	0.4	0,0089
		Total			45.9	1

Source: own work

Similarly, Table 3 shows the comparison of the hardware platforms based on parameter "B," corresponding to the availability of input/output (I/O) interfaces. In this category, the Raspberry Pi 4B stands out as the most favorable option, with a normalization factor of 0.6079, which reflects a broad availability of interfaces. In second place are both the NVIDIA Jetson Nano and the Odroid M1s, both with a value of 0.1885, which indicates an intermediate capacity regarding I/O. For its part, the BeagleBone Black presents a normalization factor of barely 0.0151, positioning it as the alternative with the lowest availability of input/output interfaces among the evaluated devices.

Table 3. Comparison Regarding Input and Output Interface Availability

	Raspberry Pi 4B	BeagleBone Black	NVIDIA Jetson Nano	Odroid M1s	Total	Factor
Raspberry Pi 4B	X	10	5	5	20	0,6079
BeagleBone Black	1/10	X	1/5	1/5	0.5	0.0151
NVIDIA Jetson Nano	1/5	5	X	1	6.2	0.1885
Odroid M1s	1/5	5	1	X	6.2	0.1885
		Total			32.9	1

Source: own work

Additionally, Table 4 presents the comparison of the hardware platforms according to parameter "C," corresponding to energy consumption. Both the Raspberry Pi 4B and the BeagleBone Black obtain the highest value in this category, each with a normalization factor of 0.4435, which indicates that both are equally energy efficient. In contrast, the NVIDIA Jetson Nano and the Odroid M1s register a factor of 0.0565, which demonstrates a significantly higher consumption. These results suggest that,

e-ISSN 2357-6014 / Vol. 22, no. 1 / january-april 2026 / Bogotá D.C., Colombia
Universidad Cooperativa de Colombia

in terms of energy efficiency, the Raspberry Pi 4B and the BeagleBone Black widely surpass the other two options, being approximately seven times more efficient.

Table 4. Comparison Regarding Potential Energy Consumption

	Raspberry Pi 4B	BeagleBone Black	NVIDIA Jetson Nano	Odroid M1s	Total	Factor
Raspberry Pi 4B	X	1	5	5	11	0.4435
BeagleBone Black	1	X	5	5	11	0.4435
NVIDIA Jetson Nano	1/5	1/5	X	1	1.4	0.0565
Odroid M1s	1/5	1/5	1	X	1.4	0.0565
		Total			24.8	1

Source: own work

Table 5 presents the comparison of the hardware platforms in relation to parameter "D," corresponding to heat dissipation capability. In this evaluation, the NVIDIA Jetson Nano clearly stands out with a normalization factor of 0.6613, which reflects its high efficiency in thermal management during operation. The Raspberry Pi 4B shows a moderate dissipation capacity, with a factor of 0.2698. On the other hand, the BeagleBone Black and Odroid M1s obtain the lowest values in this category, demonstrating limited performance regarding thermal dissipation compared to the other alternatives.

Table 5. Comparison Regarding Heat Dissipation Capability

	Raspberry Pi 4B	BeagleBone Black	NVIDIA Jetson Nano	Odroid M1s	Total	Factor
Raspberry Pi 4B	X	5	1/5	5	10.2	0.2698
BeagleBone Black	1/5	X	1/10	1	1.3	0.0343
NVIDIA Jetson Nano	5	10	X	10	25	0.6613
Odroid M1s	1/5	1	1/10	X	1.3	0.0343
		Total			37.8	1

Source: own work

Regarding parameter "E," which evaluates price and market availability, Table 6 presents the corresponding comparison. The Raspberry Pi 4B stands out with a

e-ISSN 2357-6014 / Vol. 22, no. 1 / january-april 2026 / Bogotá D.C., Colombia
Universidad Cooperativa de Colombia

normalization factor of 0.6613, which reflects its clear advantage in terms of economic accessibility and commercial availability. The BeagleBone Black and Odroid M1s boards share second place, both with a factor of 0.1884, indicating that they are slightly more costly and less common in the market. Finally, the NVIDIA Jetson Nano is positioned last due to its considerably higher price compared to the other options, which is reflected in a normalized factor of 0.0149.

Table 6. Comparison Regarding Price and Market Availability

	Raspberry Pi 4B	BeagleBone Black	NVIDIA Jetson Nano	Odroid M1s	Suma	Factor
Raspberry Pi 4B	X	5	10	5	20	0.6079
BeagleBone Black	1/5	X	5	1	6.2	0.1884
NVIDIA Jetson Nano	1/10	1/5	X	1/5	0.5	0.0149
Odroid M1s	1/5	1	5	X	6.2	0.1884
		Total			32.9	1

Source: own work

GRAPHICAL USER INTERFACE DESIGN

An aided diagnostic tool based on a user-centric graphical interface was designed. For its development, QtDesigner was utilized, a visual design tool included in the Qt framework, which facilitated the organization and arrangement of the graphical elements necessary for the assisted diagnosis. Furthermore, this platform allowed for subsequent integration with Python code, providing full functionality to the interface [30]. The tool is cross-platform and compatible with operating systems such as Windows, Linux, and macOS.

The first stage of the tool focused on designing a login screen, as skin cancer diagnosis and classification handle sensitive and confidential patient information. The authentication system, through login, also allows for maintaining a detailed record of the actions performed by each user, which is essential for audits and access control [31]. In this way, the risk of unauthorized access is minimized, ensuring data integrity and confidentiality [32].

To develop the graphical user interface, it was necessary to use the library packages detailed in Table 7. This table includes the name of each library, the version used,

e-ISSN 2357-6014 / Vol. 22, no. 1 / january-april 2026 / Bogotá D.C., Colombia
Universidad Cooperativa de Colombia

its size, and the function it performs within the interface designated for skin cancer diagnosis.

Table 7. Libraries Used to Develop the GUI

Library	Version	Size	Functionality
PyQt5	PyQt5 5.15.7 Qt 5.15.2	50.1 MB	Provides tools and classes for graphical interface design, event management, printing, among other functions.
TensorFlow	2.15.0- dev20230920	300.9 MB	Loading of previously trained deep learning models.
OpenCV	4.8.0	38.6 MB	Processing of skin lesion images.
Pillow	9.4.0	16.4 MB	Facilitates image manipulation, including opening, modification, and saving in different formats.
Numpy	1.24.3	15.8 MB	Handles multidimensional arrays and matrices, in addition to providing mathematical functions.
Pandas	1.5.3	12.3 MB	Analysis and manipulation of tabular data.

Source: own work

Additionally, Table 8 presents a functional breakdown of the main operations implemented in the graphical interface developed for skin lesion diagnosis. Each row represents a specific function within the application's logical flow, grouped into key tasks such as component initialization, user interaction, image processing, diagnostic execution, and data management. Initially, essential libraries are imported, followed by the definition of the main interface class and its configuration methods. Subsequently, functions oriented toward user interaction are incorporated, such as image loading, information display, and action confirmation. Methods dedicated to the execution of the deep learning-based diagnostic model are also included, along with functions for storing patient information. Finally, the program execution and the start of the event loop—necessary for the continuous functioning of the interface—are completed. This modular structure allows for a clean, organized, and functional design, ensuring an efficient user experience and effective integration with the artificial intelligence model.

Table 8. Pseudocode with Graphical User Interface Functionalities

N. º	Functionality
1	Load the required libraries for the system's operation.
2	Establish the main class responsible for the graphical interface.
3	Create the method that configures and starts the user interface.

(continúa)

(viene)

N. º	Functionality
4	Incorporate visual components such as buttons, labels, and other elements.
5	Create a method that allows exporting the interface in PDF format.
6	Take a screenshot of the interface and save it as a PDF file.
7	Establish the method that allows loading images from the user's system.
8	Display a pop-up window that facilitates the selection and incorporation of an image into the interface.
9	Create a method that contains detailed information about the developers.
10	Present an informational box with data about the development team.
11	Establish a procedure to confirm the application closure.
12	Display a confirmation box that allows closing the application if the user decides.
13	Design a method to issue a help message to the user.
14	Display a message with instructions on how to use the application.
15	Create the procedure responsible for executing the diagnostic analysis of the image.
16	Load the previously trained machine learning model and apply it for diagnosis.
17	Implement a method that records patient data in a CSV file.
18	Obtain and save the patient's data in a structured format.
19	Execute the program's main function.
20	Display the interface on the screen.
21	Activate the event loop that keeps the application running.

Source: own work

Similarly, Table 9 presents a summary of the main use cases associated with the application's graphical interface. These represent the essential actions that the user can carry out during interaction with the system. The process begins with the "Launch Application" use case, which marks the start of the tool's use. The user can then record patient data through the "Register Patient Data" function and continue by selecting relevant risk factors using the corresponding option. Following this, the option is offered to load an image of the lesion so that the system can execute the analysis via the "Automated Lesion Diagnosis" function. Once the analysis is complete, the user can review the obtained results. The collected information is automatically saved in a CSV file, and there is also the option to generate a PDF file with a visual summary of the results. Finally, the session can be concluded using the "Log Out" action.

e-ISSN 2357-6014 / Vol. 22, no. 1 / january-april 2026 / Bogotá D.C., Colombia
Universidad Cooperativa de Colombia

Table 9. Graphical User Interface Use Cases

Main Action	Actor	Functionality Description
Launch Application	User	The user executes the application from the device to start the process.
Register Patient Data	User	The user enters the patient's personal and medical information.
Select Risk Factors	User	Allows the user to select the applicable risk factors.
Upload Lesion Image	User	The user attaches an image of the lesion for analysis.
Automated Lesion Diagnosis	System	The system processes the data and image to generate a diagnosis.
Display Diagnosis Result	User	The diagnosis obtained by the system is presented to the user.
Save Information in CSV Format	System	The entered information and the diagnosis result are saved in a CSV file.
Export Summary in PDF	System	A PDF file is generated with a visual summary of the processed and displayed information.
Log Out / End Session	User	The user closes the session, concluding the use of the application.

Source: own work

3. VALIDATION

First, the implementation of the aided diagnostic tool on two embedded system platforms is considered. This allows for simulating usage scenarios in practical contexts with hardware constraints typical of portable or low-power devices. This phase seeks to ensure that the system not only functions in development environments but also maintains its performance when deployed on hardware with real limitations.

Subsequently, the system execution time is measured on these embedded environments to compare these results with the metrics obtained during the model training phase in more controlled or powerful settings. This comparison is essential for determining the impact of environmental conditions on the speed and efficiency of the diagnostic system.

Finally, an exhaustive evaluation of computational resource usage—particularly CPU and RAM consumption—is carried out to assess the energy and operational efficiency of the system on low-power devices. This stage is necessary to guarantee that the proposed solution is viable in contexts where resource optimization is a priority, such as in mobile applications, portable health systems, or technologies applied in areas with limited infrastructure.

e-ISSN 2357-6014 / Vol. 22, no. 1 / january-april 2026 / Bogotá D.C., Colombia
Universidad Cooperativa de Colombia

4. RESULTS

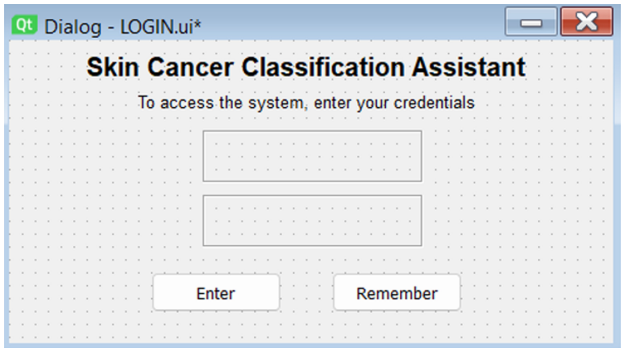
Table 10 presents the matrix used for selecting the low-power hardware intended for the execution of the skin cancer classification system. This matrix allows for the comparison of different technological alternatives based on multiple relevant criteria. To determine the total value associated with each evaluated tool, a weighting methodology is applied in which each value assigned to a specific parameter per tool is multiplied by that parameter's relative weight. The result of this operation is summed to obtain a global score, which allows for identifying the most suitable option according to the system's needs, considering both performance and energy efficiency in the evaluated hardware.

Table 10. Decision Matrix for Hardware Tool Selection.

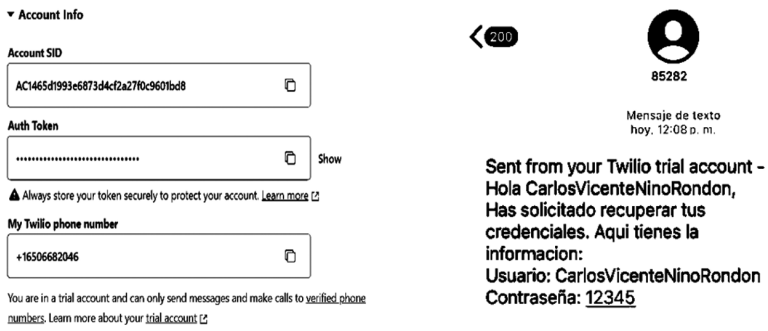
	A	B	C	D	E	**Total**
Raspberry Pi 4B	0.3311 x 0,3508	0,6079 x 0,1403	0.4435 x 0,3508	0.2698 x 0,1403	0.6079 x 0,0178	0.4054
BeagleBone Black	0.1154 x 0,3508	0.0151 x 0,1403	0.4435 x 0,3508	0.0343 x 0,1403	0.1884 x 0,0178	0.2062
NVIDIA Jetson Nano	0,5446 x 0,3508	0.1885 x 0,1403	0.0565 x 0,3508	0.6613 x 0,1403	0.0149 x 0,0178	0.3329
Odroid M1s	0,0089 x 0,3508	0.1885 x 0,1403	0.0565 x 0,3508	0.0343 x 0,1403	0.1884 x 0,0178	0.0555

Source: own work

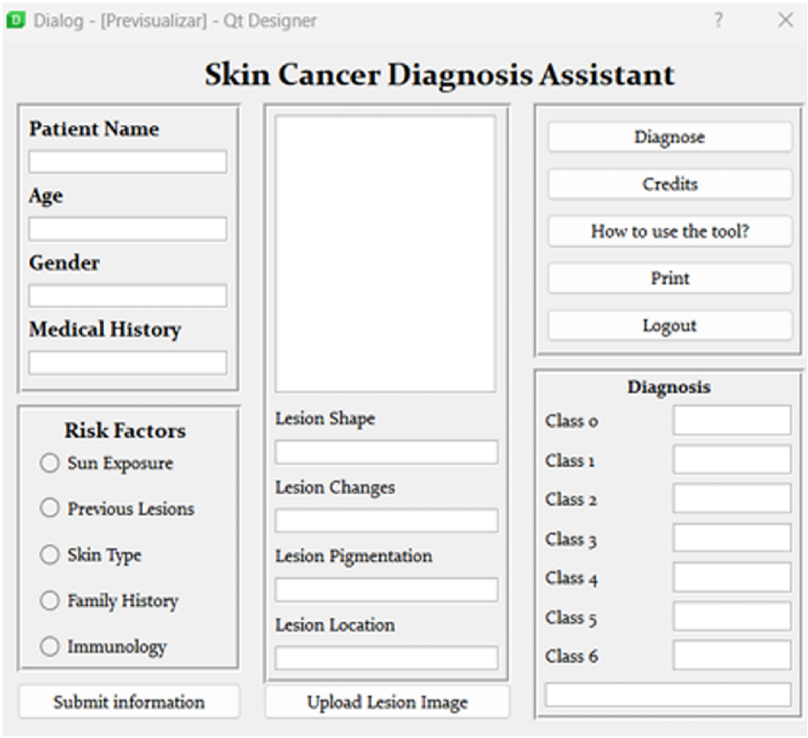
Similarly, Figure 1 shows the Graphical User Interface (GUI) and its functionalities. Section (a) presents the login interface, section (b) shows the simulation for access credential recovery, and section (c) displays the developed graphical interface with all its functionalities.



(a)



(b)



(c)

Figure 1. Developed Graphical User Interface.
Source: own work

Additionally, Figure 2 comprehensively represents the functional architecture of the Graphical User Interface (GUI) designed to run on an embedded system, specifically a Raspberry Pi, as a low-power platform for computer-aided skin cancer diagnosis. This tool integrates both the hardware and the system's logical flow, illustrating the complete process from initial user interaction to obtaining the diagnosis.

The first block shows the Raspberry Pi 4 connected to basic peripherals such as a monitor, keyboard, mouse, and wireless network. This hardware serves as the main processing unit on which the GUI and the diagnostic model are executed. The second block describes the user credential validation process, which is the first step after starting the system. Depending on whether the validation is successful (Case 2) or failed (Case 1), the system acts accordingly. In case of error, an alert is generated via the Twilio SMS messaging service; if the authentication is correct, the system allows progression to the next module.

The third block shows the interface's logical flow, starting with the lesion image upload, followed by the entry of the patient's clinical information and the selection of risk factors. This data is integrated and sent to the pre-trained model (h5 file) to carry out the automated skin lesion diagnosis. Finally, the result is classified as benign or malignant and presented to the user through the interface.

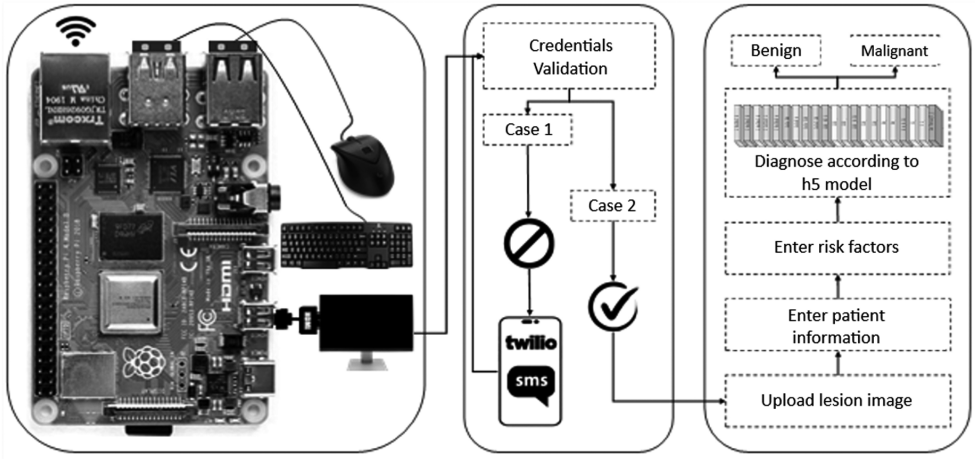


Figure 2. Functional Architecture of the Graphical User Interface.
Source: own work

Figure 3 presents a summary of the performance of the classification system executed on a Raspberry Pi 3B+ board, highlighting the times associated with three key processes: model loading, information storage, and diagnostic execution.

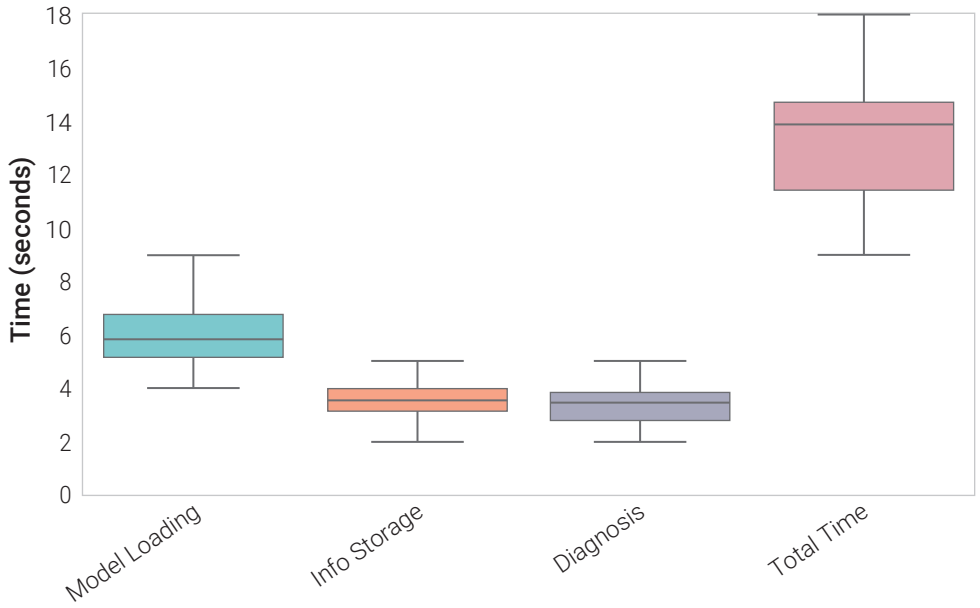


Figure 3. Distribution of Execution Times.
Source: own work

The time required to load the convolutional model ranges between 4 and 9 seconds, with an average of 6.35 seconds and a median of 6.5 seconds, which indicates a central tendency close to 6–7 seconds. The standard deviation is approximately 1.867 seconds, denoting moderate variability. Additionally, the Interquartile Range (IQR) is 2.75 seconds, implying that half of the values are concentrated within that margin.

Regarding the information storage process, the times are between 2 and 5 seconds, with a mean of 3.6 seconds and a median of 4 seconds. The data dispersion is somewhat lower, with a standard deviation of 1.187 seconds and an IQR of 2 seconds. The time needed to perform the diagnosis behaves similarly to storage, also varying between 2 and 5 seconds, with a mean and median of 3.6 seconds. The standard deviation is approximately 1.059 seconds, which indicates low dispersion, and the interquartile range is also 2 seconds. Finally, the total time per test fluctuates between 9 and 18 seconds, with a mean of 13.6 seconds and a median of 13 seconds. In this case, the standard deviation is 2.732 seconds, representing moderate variability, while the IQR is 3.75 seconds, indicating the range where half of the total measurements are concentrated.

Figure 4 illustrates the behavior of the classification system executed in an embedded environment, with special attention to CPU and RAM usage during the operation of the different functionalities of the graphical user interface. Regarding processor (CPU) usage, it is observed that the percentages recorded in the tests performed vary

between 22% and 42%, with an average utilization of 32.65%. The standard deviation, approximately 5.74%, indicates moderate variability in processor load levels. In general, this suggests that the system requires about one-third of the total processing capacity, with a data distribution that tends to be symmetrical, although with a slight inclination toward higher values. As for RAM usage, the values range between 34% and 56%, with an average of 46.70%. The standard deviation, close to 6.01%, reflects moderate data dispersion. This indicates that, during operation, the system consumes about half of the available RAM, which is relevant in resource-constrained embedded environments. Similarly, the distribution of RAM usage values also appears to be symmetrical, with a slight inclination toward the higher levels.

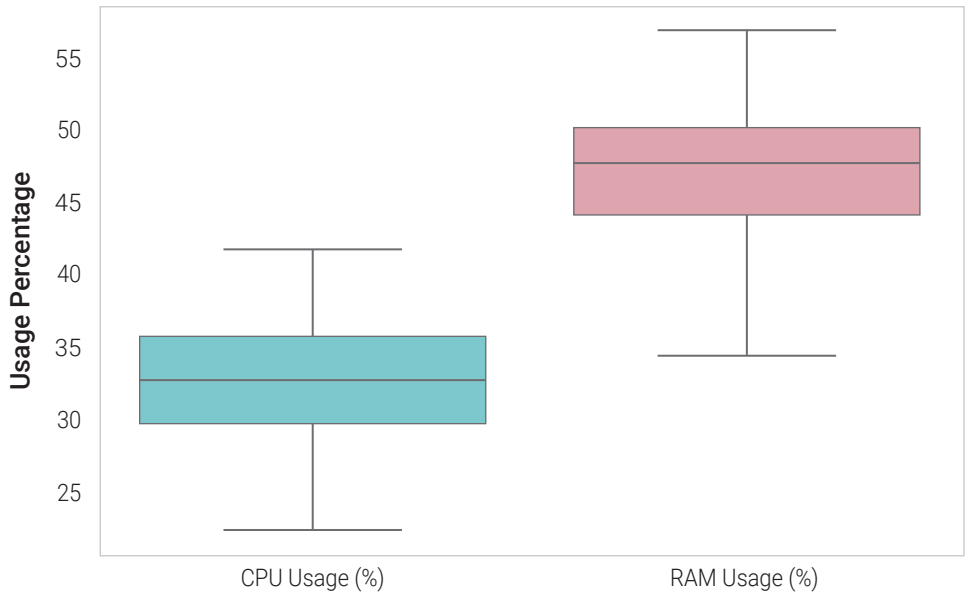


Figure 4. CPU and RAM Usage Percentage During System Execution.
Source: own work

Finally, Figure 5 presents a detailed analysis of the thermal behavior of the embedded system's processor during the execution of the classification tool. The study is divided into four consecutive stages, each associated with a specific system functionality and represented by a distinctive color.

During the initial phase, where the system is simply powered on but not executing the graphical interface, a base temperature is recorded, fluctuating between 0°*C* and 12°*C*, with a mean of 6.4°*C*. This condition is visually represented in green and reflects the passive state of the device.

e-ISSN 2357-6014 / Vol. 22, no. 1 / january-april 2026 / Bogotá D.C., Colombia
Universidad Cooperativa de Colombia

With the execution of the computational model loading, a significant temperature increase is observed, reaching a range between and 6.3°*C*, and 27°*C* a mean value of 24°*C*. This second phase is represented in blue and demonstrates the computational effort required by the model during its initialization.

Subsequently, during the information storage process, illustrated in violet tones, the temperature continues to rise, reaching peaks of up to 40°*C* in some executions.

Finally, in the diagnosis stage, the system reaches its highest thermal level, with temperatures soaring up to 55°*C*, which highlights the high resource consumption during this critical phase of the process.

Figure 5. Relationship of Embedded System Temperatures During Execution.
Source: own work

5. DISCUSSION AND CONCLUSIONS

The implementation of the classification system for skin cancer diagnosis was successfully carried out through the development of an aided diagnostic tool equipped with a user-oriented Graphical User Interface (GUI). This solution, built using Python and QtDesigner, offers an efficient and practical alternative for skin lesion analysis, while simultaneously ensuring patient data privacy and allowing for a detailed record of all user interactions. The system incorporates a login module that restricts access exclusively to authorized personnel, such as medical professionals, thereby strengthening the security and confidentiality of the managed medical information.

The tool is cross-platform compatible, which facilitates its integration into different operating environments and increases its applicability. The interface design focuses on ease of use and functionality, integrating components such as buttons, labels, and input fields to facilitate user interaction and ensure an intuitive experience. For its development, libraries such as PyQt5, TensorFlow, and OpenCV were employed, enabling the implementation of advanced capabilities such as loading deep learning models and processing medical images. Furthermore, tools like Pillow for image manipulation and Pandas for data management and analysis were integrated, thereby optimizing the system's overall performance and diagnostic accuracy.

Key functionalities of the tool include the ability to capture and export the interface in PDF format, load lesion images for analysis, display diagnostic results, and manage patient information in a structured manner.

Through a process of technical evaluation and criteria prioritization, the Raspberry Pi 4B was identified as the most suitable hardware platform to execute the system, considering factors such as processing speed, input/output interfaces, energy efficiency, heat dissipation capability, cost, and availability. The comparative analysis performed demonstrated the advantages of this board across various aspects, establishing it as a balanced and robust option for system deployment.

The performance tests conducted on the Raspberry Pi 4B indicated that the system's total execution time fluctuated between 9 and 18 seconds, with a mean of 13.6 seconds. Regarding resource usage, an average of 32.65% CPU utilization and 46.70% RAM utilization was observed, reflecting a moderate consumption of the device's resources. Finally, thermal monitoring revealed a progressive increase in the processor temperature during the execution of the system's different functionalities, reaching a maximum of 55 °C during the diagnosis phase. These results constitute a solid foundation for the implementation of computer-aided diagnostic systems in the medical field, demonstrating their technical viability and their potential impact on clinical practice.

REFERENCES

[1] M. Chen, P. Zhou, D. Wu, L. Hu, M. M. Hassan, and A. Alamri, "AI-Skin: Skin disease recognition based on self-learning and wide data collection through a closed-loop framework," *Information Fusion*, vol. 54, pp. 1–9, 2020. https://doi.org/10.1016/j.inffus.2019.06.005

[2] T. C. Pham, A. Doucet, C. M. Luong, C. T. Tran, and V. D. Hoang, "Improving skin-disease classification based on customized loss function combined with balanced mini-batch logic and

real-time image augmentation," *IEEE Access*, vol. 8, pp. 150725–150737, 2020. https://doi.org/10.1109/access.2020.3016653

[3] W. Gouda, N. U. Sama, G. Al-Waakid, M. Humayun, and N. Z. Jhanjhi, "Detection of skin cancer based on skin lesion images using deep learning," *Healthcare (Switzerland)*, vol. 10, no. 7, Jul. 2022. https://doi.org/10.3390/healthcare10071183

[4] K. Ali, Z. A. Shaikh, A. A. Khan, and A. A. Laghari, "Multiclass skin cancer classification using EfficientNets – a first step towards preventing skin cancer," *Neuroscience Informatics*, vol. 2, no. 4, p. 100034, Dec. 2022. https://doi.org/10.1016/j.neuri.2021.100034

[5] S. Jinnai, N. Yamazaki, Y. Hirano, Y. Sugawara, Y. Ohe, and R. Hamamoto, "The development of a skin cancer classification system for pigmented skin lesions using deep learning," *Biomolecules*, vol. 10, no. 8, pp. 1–13, Aug. 2020. https://doi.org/10.3390/biom10081123

[6] S. S. Chaturvedi, K. Gupta, and P. S. Prasad, "Skin lesion analyser: an efficient seven-way multi-class skin cancer classification using MobileNet," in *Advances in Intelligent Systems and Computing*, Springer, 2021, pp. 165–176. https://doi.org/10.1007/978-981-15-3383-9_15

[7] R. P. Widhianto et al., "Hardware design of skin cancer detection device," in *9th International Conference on Engineering and Emerging Technologies (ICEET)*, IEEE, May 2024, pp. 1–6. https://doi.org/10.1109/iceet60227.2023.10525755

[8] S. A. A. Ahmed, B. Yanikoglu, O. Goksu, and E. Aptoula, "Skin lesion classification with deep CNN ensembles," in *28th Signal Processing and Communications Applications Conference (SIU)*, Oct. 2020. https://doi.org/10.1109/siu49456.2020.9302125

[9] Z. Huang et al., "The correlation of deep learning-based CAD-RADS evaluated by coronary computed tomography angiography with breast arterial calcification on mammography," *Scientific Reports*, vol. 10, no. 1, pp. 1–8, 2020. https://doi.org/10.1038/s41598-020-68378-4

[10] V. Mohite, A. B. Deoghare, and K. M. Pandey, "Modeling of human airways CAD model using CT scan data," *Materials Today: Proceedings*, vol. 22, pp. 1710–1714, 2019. https://doi.org/10.1016/j.matpr.2020.02.189

[11] A. Kumar and A. Vatsa, "Untangling classification methods for melanoma skin cancer," *Frontiers in Big Data*, vol. 5, pp. 1–11, Mar. 2022. https://doi.org/10.3389/fdata.2022.848614

[12] M. K. Monika et al., "Skin cancer detection and classification using machine learning," *Materials Today: Proceedings*, pp. 4266–4270, 2020. https://doi.org/10.1016/j.matpr.2020.07.366

[13] J. Rashid et al., "Skin cancer disease detection using transfer learning technique," *Applied Sciences (Switzerland)*, vol. 12, no. 11, Jun. 2022. https://doi.org/10.3390/app12115714

[14] Ş. Öztürk and U. Özkaya, "Skin lesion segmentation with improved convolutional neural network," *Journal of Digital Imaging*, vol. 33, no. 4, pp. 958–970, May 2020. https://doi.org/10.1007/s10278-020-00343-z

[15] A. Imran et al., "Skin cancer detection using combined decision of deep learners," *IEEE Access*, vol. 10, pp. 118198–118212, 2022. https://doi.org/10.1109/access.2022.3220329

[16] Q. et al., "A two-stage low-altitude remote sensing *Papaver somniferum* image detection system based on YOLOv5s + DenseNet121," *Remote Sensing*, vol. 14, no. 8, pp. 1–18, 2022. https://doi.org/10.3390/rs14081834

[17] M. Z. U. Rehman et al., "Classification of skin cancer lesions using explainable deep learning," *Sensors*, vol. 22, no. 18, Sep. 2022. https://doi.org/10.3390/s22186915

[18] A. Imran et al., "Skin cancer detection using combined decision of deep learners," *IEEE Access*, vol. 10, pp. 118198–118212, 2022. https://doi.org/10.1109/access.2022.3220329

[19] S. Jain et al., "Deep learning-based transfer learning for classification of skin cancer," *Sensors*, vol. 21, no. 23, Dec. 2021. https://doi.org/10.3390/s21238142

[20] S. Jain et al., *Deep learning-based transfer learning for classification of skin cancer*, *Sensors*, vol. 21, no. 23, Dec. 2021. https://doi.org/10.3390/s21238142

[21] H.-W. Huang, B. W.-Y. Hsu, C.-H. Lee, and V. S. Tseng, "Development of a lightweight deep learning model for cloud applications and remote diagnosis of skin cancers," *Journal of Dermatology*, vol. 48, no. 3, pp. 310–316, Mar. 2021. https://doi.org/10.1111/1346-8138.15683

[22] T. Guergueb and M. A. Akhloufi, "Melanoma skin cancer detection using recent deep learning models," in *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBS)*, 2021, pp. 3074–3077. https://doi.org/10.1109/embc46164.2021.9631047

[23] J. Ramya, H. C. Vijaylakshmi, and H. Mirza Saifuddin, "Segmentation of skin lesion images using discrete wavelet transform," *Biomedical Signal Processing and Control*, vol. 69, Aug. 2021. https://doi.org/10.1016/j.bspc.2021.102839

[24] N. A. Ramírez Pérez, E. Gómez Vargas, and H. Vacca González, "Automatic learning model to predict transparency indicators for effective management of public resources," *Ingeniería Solidaria*, pp. 1–21, Sep. 2023. https://doi.org/10.16925/2357-6014.2023.03.09

[25] P. Varalakshmi, V. Aruna Devi, M. Ezhilarasi, and N. Sandhiya, "Enhanced dermatoscopic skin lesion classification using machine learning techniques," in *2021 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, pp. 68–71. https://doi.org/10.1109/wispnet51692.2021.9419466

[26] Y. Jin, M. Ma, and Y. Zhu, "A comparison of natural user interface and graphical user interface for narrative in HMD-based augmented reality," *Multimedia Tools and Applications*, vol. 81, no. 4, pp. 5795–5826, 2022. https://doi.org/10.1007/s11042-021-11723-0

[27] C. V. Niño-Rondón, S. A. Castro-Casadiego, B. Medina-Delgado, and D. Guevara-Ibarra, "Análisis de viabilidad y diseño de un sistema electrónico para el seguimiento de la dinámica poblacional en la ciudad de Cúcuta," *Ingenierías USBMed*, vol. 11, no. 1, pp. 56–64, 2020. https://doi.org/10.21500/20275846.4489

[28] M. Mallegowda, A. Anithakanavlli, and A. M. P. Amrutha, "Design and integration of middleware for IoT devices towards solar panel monitoring based on Raspberry Pi," *Journal of Seybold Report*, 2020, p. 19.

[29] N. Onizawa, S. C. Smithson, B. H. Meyer, W. J. Gross, and T. Hanyu, "In-hardware training chip based on CMOS invertible logic for machine learning," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 5, pp. 1541–1550, May 2020. https://doi.org/10.1109/tcsi.2019.2960383

[30] Y. Batko, G. Melnyk, and O. Pitsun, "Graphical interface of hybrid intelligent systems for biomedical imaging analysis," in *2016 IEEE 1st International Conference on Data Stream Mining and Processing (DSMP)*, 2016, pp. 121–124. https://doi.org/10.1109/dsmp.2016.7583521

[31] S. Jiang, H. Li, and Z. Jin, "A visually interpretable deep learning framework for histopathological image-based skin cancer diagnosis," *IEEE Journal of Biomedical and Health Informatics*, vol. 25, no. 5, pp. 1483–1494, May 2021. https://doi.org/10.1109/jbhi.2021.3052044

[32] N. Kausar et al., "Multiclass skin cancer classification using ensemble of fine-tuned deep learning models," *Applied Sciences (Switzerland)*, vol. 11, no. 22, Nov. 2021. https://doi.org/10.3390/app112210593