

UNIVERSIDAD DEL VALLE DE GUATEMALA  
Facultad de Ingeniería



**Implementación de un Sistema de Visión por Computadora  
para el Reconocimiento Facial y de Emociones para el Rostro  
Animatrónico de la Universidad del Valle de Guatemala**

Trabajo de graduación presentado por Jorge Antonio Lorenzana Ramos  
para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

2022







UNIVERSIDAD DEL VALLE DE GUATEMALA  
Facultad de Ingeniería



**Implementación de un Sistema de Visión por Computadora  
para el Reconocimiento Facial y de Emociones para el Rostro  
Animatrónico de la Universidad del Valle de Guatemala**

Trabajo de graduación presentado por Jorge Antonio Lorenzana Ramos  
para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

2022



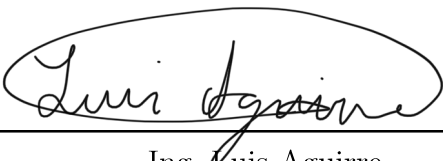
Vo.Bo.:

(f)   
Ing. Kurt Kellner

Tribunal Examinador:

(f)   
Ing. Kurt Kellner

(f)   
Msc. José Morales

(f)   
Ing. Luis Aguirre

Fecha de aprobación: Guatemala, 5 de Enero de 2022.





Este trabajo de graduación es el reflejo de la búsqueda constante de retos que superar, de mi pasión por la tecnología y el querer derribar las barreras modernas. Desde pequeño he tenido un interés por los androides y la inteligencia artificial, siendo admirador de las películas de ciencia ficción, lo que me ha llevado a trabajar en un rostro animatrónico autónomo.

Agradezco el apoyo incondicional de las personas que estuvieron a mi lado durante toda mi carrera universitaria. Especialmente a mis padres, Karlo Lorenzana y Sofía Ramos, quienes me apoyaron tanto económica como emocionalmente durante estos cinco años de carrera. A mi hermano, Alejandro Lorenzana, por alentarme a seguir y brindarme su consejo para el desarrollo de software. A mis amigos que vivieron conmigo este proceso de preparación profesional, por su apoyo, consejo y comprensión en momentos precisos. A mi asesor, el ingeniero Kurt Kellner, por guiarme y brindarme su conocimiento para el desarrollo de este trabajo de graduación. A la Universidad del Valle de Guatemala por darme la oportunidad de formar parte de su comunidad y brindarme el material necesario para prepararme profesionalmente.

Finalmente, dedicar este trabajo a mi persona, por el esfuerzo entregado durante estos años. Por haber superado las dificultades presentadas y nunca haberme rendido. Y por finalmente ver los frutos de la preparación que he recibido.



<b>Prefacio</b>	<b>v</b>
<b>Lista de figuras</b>	<b>x</b>
<b>Lista de cuadros</b>	<b>xi</b>
<b>Resumen</b>	<b>xiii</b>
<b>Abstract</b>	<b>xv</b>
<b>1. Introducción</b>	<b>1</b>
<b>2. Antecedentes</b>	<b>3</b>
2.1. Sistema de detección de rostro y reconocimiento de gestos para robot anima- trónico . . . . .	3
2.2. Detector de emociones utilizando OpenCV . . . . .	4
<b>3. Justificación</b>	<b>7</b>
<b>4. Objetivos</b>	<b>9</b>
4.1. Objetivo general . . . . .	9
4.2. Objetivos específicos . . . . .	9
<b>5. Alcance</b>	<b>11</b>
<b>6. Marco teórico</b>	<b>13</b>
6.1. Las 7 emociones básicas . . . . .	13
6.1.1. Las señales faciales de las emociones . . . . .	14
6.2. Cómo reconocer las emociones en los demás . . . . .	16
6.2.1. Alegría . . . . .	16
6.2.2. Sorpresa . . . . .	16
6.2.3. Tristeza . . . . .	16
6.2.4. Miedo . . . . .	17
6.2.5. Ira . . . . .	17

6.2.6.	Asco . . . . .	18
6.2.7.	Desprecio . . . . .	18
6.3.	Visión por computadora . . . . .	19
6.3.1.	OpenCV . . . . .	19
6.3.2.	SimpleCV . . . . .	19
6.3.3.	Accord.NET Framework . . . . .	20
6.3.4.	BoofCV . . . . .	20
6.4.	Aprendizaje automático . . . . .	21
6.4.1.	Keras . . . . .	21
6.4.2.	TensorFlow . . . . .	21
6.4.3.	PyTorch . . . . .	22
6.5.	Kivy . . . . .	22
6.6.	CUDA . . . . .	23
6.7.	Aprendizaje automático . . . . .	24
6.7.1.	Aprendizaje Profundo . . . . .	24
<b>7.</b>	<b>Modelo 3D</b>	<b>25</b>
<b>8.</b>	<b>Software de detección de emociones</b>	<b>31</b>
8.1.	Modelo de reconocimiento . . . . .	31
8.1.1.	Trade Study . . . . .	32
8.1.2.	Desarrollo del modelo . . . . .	33
8.2.	Detector de emociones . . . . .	37
8.2.1.	Trade Study . . . . .	38
8.2.2.	Desarrollo del software . . . . .	39
<b>9.</b>	<b>Respuesta a las emociones</b>	<b>43</b>
9.1.	Imitación . . . . .	43
9.2.	Reacción . . . . .	45
<b>10.</b>	<b>GUI</b>	<b>47</b>
<b>11.</b>	<b>Conclusiones</b>	<b>53</b>
<b>12.</b>	<b>Recomendaciones</b>	<b>55</b>
<b>13.</b>	<b>Bibliografía</b>	<b>57</b>
<b>14.</b>	<b>Anexos</b>	<b>59</b>

---

## Lista de figuras

---

1.	Detección de rostros con ángulo de visión de 45°. [1]	4
2.	Resultado del detector de emociones de Karan Sethi. [2].	5
3.	Gestos de distintas emociones [4].	14
4.	Rasgos de alegría [5].	16
5.	Rasgos de sorpresa [5].	16
6.	Rasgos de tristeza [5].	17
7.	Rasgos de Miedo [5].	17
8.	Rasgos de ira [5].	17
9.	Rasgos de asco [5].	18
10.	Rasgos de desprecio [5].	18
11.	OpenCV logo [6]	19
12.	SimpleCV logo [7]	19
13.	Accord.NET Framework logo [8]	20
14.	BoofCV logo [9]	20
15.	Keras logo [10]	21
16.	TensorFlow logo [11]	21
17.	PyTorch logo [12]	22
18.	Kivy logo [13]	23
19.	Nvidia logo [14]	23
20.	Red Neuronal como representación del aprendizaje profundo [16]	24
21.	Base de la fase anterior en las condiciones en las que se me entregó.	26
22.	Base modificada para sujetar servo-motores.	26
23.	Base rotativa para asegurar el ensamblaje de la base para sujetar los servo-motores.	27
24.	Base cojinete con crush ribs.	28
25.	Sujetador para le cojinete con crush ribs.	28
26.	Renderizado de la base.	29
27.	Ejemplo de una red neuronal y su proceso de entrenamiento [17].	31
28.	Ejemplo de imágenes contenidas en el dataset de FER-2013 a escala 1:1 [18].	33
29.	Capas red neuronal.	34

30.	Hardware utilizado. . . . .	35
31.	Ejemplo de imágenes contenidas en el dataset de AffectNet a escala 1:1 [19] .	36
32.	Imágenes de AffectNet modificadas. . . . .	37
33.	Ejemplo del detector de emociones entrenado con FER-2013. . . . .	38
34.	Detector de rostros utilizando Haar Cascade. . . . .	39
35.	Detector de emociones utilizando Haar Cascade y el modelo de reconocimiento creado por nosotros entrenado con AffectNet. . . . .	40
36.	Resultado de emociones detectadas (alegría, sorpresa, enojo, disgusto, miedo, triste) con modelo entrenado con el conjunto de datos de AffectNet. . . . .	41
37.	Conexión del circuito con los motores para pruebas. . . . .	44
38.	Árbol de decisiones. . . . .	45
39.	Diseño de las ventanas. . . . .	48
40.	Interfaz gráfica con el detector de emociones implementado. . . . .	48
41.	Interfaz gráfica con el árbol de decisiones implementado. . . . .	49
42.	Interfaz gráfica para la base de datos. . . . .	50
43.	Interfaz gráfica para el modo de reacción. . . . .	50
44.	Interfaz gráfica para el modo de imitación. . . . .	51
45.	Interfaz gráfica para la conexión del puerto COM. . . . .	51

---

## Lista de cuadros

---

1.	Lista de unidades de acción y descripciones de acción. . . . .	15
2.	Trade Study de los distintos software de Machine Learning. . . . .	32
3.	Peso de características de los software de deep learning y sus resultados. . . . .	32
4.	Parámetros y resultados para las distintas funciones de activación con dropout de 0.2. . . . .	36
5.	Comparación entre el modelo entrenado con FER-2013 y el entrenado con AffectNet. . . . .	37
6.	Trade Study de los distintos software de visión por computadora. . . . .	38
7.	Peso de características de los software de visión por computadora y sus resultados. . . . .	39
8.	Codificación de las emociones en el microcontrolador programado por Christopher Jenatz. . . . .	44
9.	Campos de las tablas en la base de datos. . . . .	46
10.	Campos finales de las tablas en la base de datos. . . . .	46





El presente trabajo de graduación tiene como objetivo principal el desarrollo e implementación de un sistema de visión por computador de reconocimiento facial y de emociones para la interacción con el rostro animatrónico de la Universidad del Valle de Guatemala. Para esto se seleccionó Keras para realizar el modelo de reconocimiento, OpenCV para la visión por computadora y Kivy para la interfaz gráfica del usuario. Asimismo, se buscó programar una respuesta congruente del rostro para la emoción que este detecte y crear una interacción Humano-Robot.

Para lograr una detección de emociones acertada se requirió de una detallada investigación acerca de los gestos faciales que el ser humano presenta según su estado de ánimo. Debido a que el rostro humano tiene múltiples unidades de acción no se podría asignar una sola para cada expresión. Por ello hay distintas configuraciones que combinan diferentes unidades de acción que se le asignan a cada emoción. Basándonos en estos criterios se buscó un conjunto de datos que contuviera la información necesaria que representara las emociones que queríamos.

El desarrollo del modelo de reconocimiento de emociones se realizó en Python haciendo uso de la API Keras para el aprendizaje profundo. Se realizó una red neuronal iterativa con distintas funciones de activación para encontrar el mejor método de entrenamiento. Se utilizaron dos conjuntos de imágenes para el desarrollo del modelo, el primero era pequeño y su función era encontrar la mejor forma de entrenar el modelo y el segundo era más grande y su función era mejorar los resultados para generar el modelo de reconocimiento final.

El reconocimiento de emociones fue programado en Python utilizando la librería de OpenCV y la API Keras. Se utilizó un código base de reconocimiento facial que utiliza un clasificador *Haar Cascade* y se modificó para aplicar el modelo de reconocimiento que desarrollamos. Para que el programa de reconocimiento funcione se necesita una cámara conectada a la computadora.

Finalmente, se procedió a crear una interfaz gráfica para el usuario que fuera sencilla de manipular y entender. El objetivo de esta era poder monitorear la información que se obtenía, la información que se enviaba y la respuesta del robot. Esta interfaz fue desarrollada en Python utilizando, la biblioteca de desarrollo de GUI, Kivy.



The main objective of this graduation work is the development and implementation of a computer vision system for facial and emotion recognition for interaction with the animatronic face of the Universidad del Valle de Guatemala. For this, Keras was selected for the recognition model, OpenCV for the computer vision and Kivy for the graphical user interface. Likewise, it was sought to program a congruent response of the face for the emotion that it detects and create a Human-Robot interaction.

To achieve an accurate detection of emotions, a detailed investigation was required about the facial gestures that the human being presents according to his mood. Because the human face has multiple units of action, a single one could not be assigned to each expression. For this reason, there are different configurations that combine different units of action that are assigned to each emotion. Based on these criteria, a data set was searched that contained the necessary information that represented the emotions we wanted.

The development of the emotion recognition model was carried out in Python using the Keras API for deep learning. An iterative neural network with different activation functions was performed to find the best training method. Two sets of images were used for the development of the model, the first was small and its function was to find the best way to train the model and the second was larger and its function was to improve the results to generate the final recognition model.

Emotion recognition was programmed in Python using the OpenCV library and the Keras API. A facial recognition base code using a *Haar Cascade* classifier was used and modified to apply the recognition model we developed. For the recognition program to work you need a camera connected to the computer.

Finally, we proceeded to create a graphical user interface that was easy to manipulate and understand. The objective of this was to be able to monitor the information that was obtained, the information that was sent and the response of the robot. This interface was developed in Python using the GUI development library, Kivy.



El objetivo principal de este proyecto es el desarrollar e implementar un sistema de visión por computadora que nos permita realizar el reconocimiento facial y de emociones para poder realizar interacciones con el rostro animatrónico que ha sido diseñado y fabricado en la Universidad del Valle de Guatemala. El rostro animatrónico contaba con un diseño anatómicamente correcto y movimientos acertados al de un rostro humano. Lo que se buscó con este proyecto fue desarrollar un comportamiento humano en cuanto a acciones se refiere.

Un rostro humano puede representar su estado de ánimo por medio de expresiones faciales. Entre estas emociones hay siete principales que son la alegría, la tristeza, el miedo, el enojo, el asco, el desprecio y la sorpresa. Cada una de estas cuenta con una combinación de unidades de acción del rostro que permite identificarlas. Estas emociones básicas tienen funciones sociales y de supervivencia. Es por eso que pueden variar según el entorno que nos rodea, sucesos o eventos que se experimentan e incluso una conversación puede cambiar el estado de una persona.

Si bien la Universidad del Valle de Guatemala cuenta con un rostro animatrónico con la anatomía humana correcta, no se ha implementado un comportamiento para la interacción con los usuarios. Por eso haciendo uso de los avances tecnológicos se ha buscado la manera de darselo y que sea lo más humano posible. Se buscó darle la capacidad de poder reaccionar a la percepción psicológica que este obtiene de las personas para poder entablar una interacción Humano-Robot. Esto incluye comportamientos específicos que estén dentro de las acciones que un ser humano tomaría al encontrarse en situaciones similares.

En este trabajo de graduación se detalla el proceso de diseño y desarrollo para realizar un rostro animatrónico con capacidad de reaccionar a las emociones. En estos procedimientos principalmente se buscó mejorar tanto funcional como estéticamente el modelo 3D rediseñando piezas que necesitaban ser modificadas. También incluye el desarrollo del detector de emociones que abarca la creación y entrenamiento de un modelo de reconocimiento, la implementación en un programa de visión por computadora, el árbol de decisiones que tiene el rostro y el desarrollo de una interfaz gráfica para el usuario.



## 2.1. Sistema de detección de rostro y reconocimiento de gestos para robot animatrónico

Anteriormente en la Universidad del Valle de Guatemala (UVG), el estudiante Luis Eduardo Ruano Argueta trabajó un software para la detección del rostro y emociones. Para este proyecto se utilizó Python, OpenCV, Base de datos de rostros, entre otros. Se utilizan dos algoritmos distintos para clasificar las distintas emociones y ellos utilizan una cámara Logitech 920 para obtener las imágenes en tiempo real [1].

El objetivo principal de este proyecto es implementar por medio de software un programa que pueda darle seguimiento a los rostros y el reconocimiento de expresiones faciales de las personas. Para esto se realizó una comparación entre dos algoritmos. Uno que utiliza algoritmos en cascada y otro que utiliza las marcas de cara. Se analizaron ambos algoritmos en dos ambientes diferentes, uno controlado y uno no controlado para evaluar el desempeño de ambos en aspectos como la distancia máxima, movimientos laterales, agresividad de movimientos y rotación con respecto al ángulo de visión. Entre los resultados destaca el alcance máximo del algoritmo *Haar cascade*, con un alcance máximo de 5.31m en ambientes controlados y abiertos. Mientras que el algoritmo de marcas de cara destaca en cuando a la detección cuando existe un ángulo de rotación mayor a 45° al ángulo de visión respecto a la cámara. [1].

Los algoritmos seleccionados para el *Haar cascade* y el algoritmo de marcas de cara son *Fisher Face* y SVM respectivamente. Para el algoritmo *Fisher Face* se realizó reconstrucción de imágenes y representación de estas por medio de dispersión de datos. Para el algoritmo SVM se utilizó marcas de cara para generar entrenamientos [1].

Para las bases de datos se utilizó una base de datos descargada llamada *Cohn-Kanade* y las bases generadas *UVG*, *UVG 2m*, *Ambiente 2* y *Personalizada*. Estas bases de datos se utilizaron para entrenar los algoritmos para trabajar en los distintos ambientes, ya sean controlados o no, cantidad de iluminación, distancia, entre otros [1].

Se tuvieron problemas al momento de reconocer ciertas expresiones, ya que los algoritmos las confundían con otras. Al hacer las pruebas se limitó la cantidad de expresiones para detectar a tres: Feliz, enojo y sorpresa. Al realizar esta modificación se tuvo una mejora en los resultados teniendo un porcentaje de éxito por encima del 68 %, donde el más bajo fue de 68.21 % para el ambiente UVG 2m con el algoritmo *Fisher Face* [1].

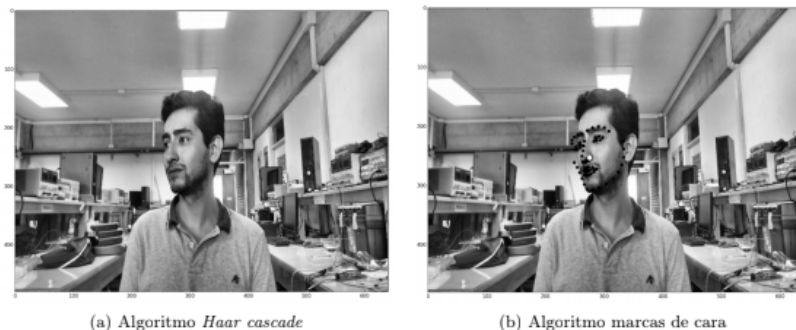


Figura 1: Detección de rostros con ángulo de visión de 45°. [1]

## 2.2. Detector de emociones utilizando OpenCV

El usuario de medium.com, Karan Sethi, publicó un artículo donde presenta el procedimiento y resultados de la detección de su rostro y el reconocimiento de cinco emociones. El usuario utilizó el software de OpenCV para la visión por computadora y Keras para el aprendizaje automático. El procedimiento que utilizó se reduce en dos grandes etapas, la creación del modelo, que incluye el entrenamiento del mismo, y la implementación del modelo para el reconocimiento de emociones en tiempo real [2].

En el artículo, Karan, dice explícitamente que los prerequisites para poder realizar este proyecto es tener conocimiento básico de los siguientes temas:

- Python
- OpenCV
- Red neuronal de convolución (CNN)
- numpy

El objetivo principal de Karan es crear una red neuronal de convolución utilizando Keras, la API para Python sobre *Deep Learning*, para detectar emociones en tiempo real a través de la retroalimentación al sistema de una cámara en tiempo real [2].

Para esto él realiza la primera etapa, la creación del modelo, en cinco tareas. La primera tarea es importar todos los módulos necesarios en el proyecto, declarar algunas variables como la cantidad de emociones, los píxeles de las imágenes que se estarán utilizando y la cantidad de muestras que se tomarán antes de que el modelo se actualice. Seguido de eso, realiza la segunda tarea, la cual es importar el *dataset* que se estará utilizando. Él emplea



el conjunto de datos de **fer2013** que es de libre uso alojado en la plataforma de **kaggle**. Este contiene siete clases, las cuales corresponden a las siete emociones básicas, sin embargo, Karan aprovecha solo cinco de esas clases. Ahora continuamos con la tercera tarea la cual consiste en expandir el conjunto de imágenes que se está manejando de manera artificial. Ahora, la cuarta tarea es crear el cerebro de nuestro modelo, es decir, la red neuronal de convolución. Se define el modelo que se aplicará que, en el caso de Karan, es el modelo secuencial que define que todas las capas en la red serán sucesivas y las almacenará en una variable del modelo. Por último, la quinta tarea es compilar y entrenar el modelo. Para esto se importan algunas librerías, se crean algunas funciones y se ajusta el modelo. Con esto ya solo quedaría implementar el modelo para el reconocimiento de emociones en tiempo real [2].

Ahora, Karan, realiza la segunda etapa. La cual consta de crear el controlador para usar el modelo creado anteriormente. Lo primero es importar algunos módulos necesarios para el funcionamiento del código, como las herramientas de OpenCV. También hay que cargar el modelo y el algoritmo de clasificación, él utiliza **Haar Cascade** que es un algoritmo para identificar objetos en imágenes o videos. Después de programar y terminar el código el detector de emociones estaría terminado. Y como resultado se tendría lo mostrado en la Figura 2 [2].

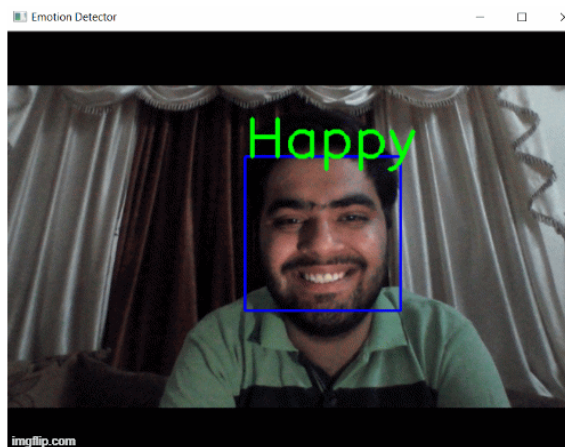


Figura 2: Resultado del detector de emociones de Karan Sethi. [2].



El ámbito de la animatrónica en Guatemala no está desarrollado. Siendo algo que se puede aplicar en varios campos de trabajo como lo es la industria del entretenimiento, el turismo o cualquiera que necesite de dar información. Si bien un animatrónico no puede sustituir por completo un recurso humano, si que puede ser de utilidad para cumplir ciertas funciones. Este rostro podrá ser utilizado en la Universidad del Valle de Guatemala para brindar al usuario información útil acerca de la misma, reaccionando al estado de ánimo. Así como esta aplicación, se puede aplicar en museos, para dar información acerca de las distintas piezas que se exponen. De esta manera ayudar a instituciones a darse a basto para cumplir con la demanda de información y evitar que el personal humano realice la misma actividad multiples veces. Además de empezar con el desarrollo y aprendizaje de creaciones de inteligencias pseudoartificiales.

El desarrollo de animatrónicos es un buen ejercicio para un ingeniero que busca especializarse en el campo de la robótica, automatización e incluso en biomédica, ya que cuenta con replicar movimientos humanos y diseñar en base a la anatomía humana. Es un proceso muy complejo que requiere de conocimientos de las distintas especializaciones para llevarse a cabo. Por otro lado, el desarrollo del detector de emociones es un proceso que necesita de programación y psicología, dos campos completamente distintos. Esto porque hay que realizar un software capaz de capturar imágenes y procesarlas y hay que tener conocimientos en el área de psicología para poder interpretar la emoción que se está mostrando y como responder ante esta.

Así pues, el desarrollo de este proyecto es una oportunidad que abarca muchas áreas de aprendizaje y de desarrollo, tanto para las instituciones como para el país. Además de ser un proyecto que puede seguir creciendo con el tiempo, de manera que abre el camino para seguir aprendiendo e implementando nuevos sistemas, nuevas interacciones e incluso nuevas respuestas.



### 4.1. Objetivo general

Desarrollar e implementar un sistema de visión por computador de reconocimiento facial y de emociones para la interacción con el rostro animatrónico de la Universidad del Valle de Guatemala.

### 4.2. Objetivos específicos

- Identificar, por medio de las herramientas de software seleccionadas, las siete emociones básicas: alegría, tristeza, miedo, enojo, asco, desprecio, sorpresa.
- Implementar un sistema en el que el rostro animatrónico imite directamente los gestos de una persona.
- Implementar un sistema de respuestas pre-programadas para interactuar con el usuario según la emoción que percibe el rostro animatrónico.
- Implementar una interfaz gráfica de uso sencillo que sirva de puente entre el hardware y los sensores.
- Desarrollar un árbol de decisiones para la continua interacción Humano-Robot.



El rostro animatrónico propone el avance en el tema de inteligencia artificial, la programación de interfaces gráficas y la comunicación entre microcontroladores y computadoras implementado a un sistema mecánico desarrollado previamente en la Universidad del Valle de Guatemala.

El alcance de este trabajo abarca la investigación de las redes neuronales para el desarrollo de un sistema de visión por computadora capaz de identificar las emociones que presenta un ser humano a través de los gestos faciales. En este proceso se involucra el entrenamiento y validación del modelo para la detección de siete distintas. Estas son analizadas para que el rostro animatrónico pueda responder adecuadamente según lo que presenta el usuario.

Dentro del detector de emociones se propone una red neuronal capaz de entrenar el modelo con conjuntos de datos para la identificación de estas utilizando Keras. Para validar este aspecto se implementa el uso de OpenCV y así poner a prueba el modelo entrenado. Verificando que sea capaz de identificar todas las emociones a través de gestos humanos en tiempo real.

El desarrollo de las acciones del rostro incluye la programación en lenguaje de Python realizando la comunicación serial con el respectivo microcontrolador para el movimiento del rostro. Además, utilizando el mismo lenguaje se desarrolla un árbol de emociones para realizar la interacción con el usuario según la emoción que este muestra, dando así frases coherentes al comportamiento del usuario.

En el diseño de la interfaz gráfica se abarca la programación tanto de Python como de lenguaje Kivy, para obtener un *software* agradable a la vista del usuario y que sea simple de utilizar. Esta interfaz cubre desde lo que se está recibiendo del detector de emociones, hasta lo que se está enviando para que el rostro interactúe con el usuario.





## 6.1. Las 7 emociones básicas

Las emociones evolucionaron para cumplir la función de promover la supervivencia física, pero el desarrollo de la cultura humana también ha impulsado la evolución de las emociones en el sentido de que estas cumplan funciones relativas a metas sociales, tales como llevarse bien con los demás y avanzar en la comunidad. Las emociones básicas tienen funciones sociales además de las de supervivencia [3].

Estas siete emociones básicas son:

- **Alegría:** Sensación dichosa de placer y bienestar [3].
- **Tristeza:** Sensación opresiva de pérdida o carencia que produce desánimo [3].
- **Miedo:** Sensación de agitación causada por una percepción de peligro debida a riesgos físicos, morales, o la presencia de dolor [3].
- **Enojo:** Sensación perturbadora que resulta de una ofensa, una torpeza propia o un obstáculo natural. Generalmente incluye el deseo de reaccionar agresivamente [3].
- **Asco:** Sensación de repugnancia debida a la percepción de un estímulo desagradable a los sentidos [3].
- **Desprecio:** Sensación de rechazo o desestimación hacia otra persona o cosa, por considerarla inferior, indigna o carente de valor [3].
- **Sorpresa:** Sensación súbita e inesperada de asombro [3].



Figura 3: Gestos de distintas emociones [4].

### 6.1.1. Las señales faciales de las emociones

Las señales faciales son las configuraciones de los distintos rasgos particulares de cada emoción, que son producidas por movimientos involuntarios en los músculos del rostro. Según Ekman, las expresiones de las emociones nos dan información acerca de lo que está ocurriendo dentro de la persona, lo que posiblemente ocurrió antes de que se desarrollara la emoción y aquello que puede llegar a pasar en consecuencia de la emoción. Para esto se utiliza el sistema FACS (Facial Action Coding System), que es un sistema de codificación que recoge todos movimientos expresivos del rostro en unidades de acción (AU) [4].

<b>AU</b>	<b>Acción</b>
1	Levantamiento interior de ceja
2	Levantamiento exterior de ceja
4	Bajar cejas
5	Levantamiento del párpado superior
6	Levantamiento de mejillas
7	Apretar párpados
8	Labios encimados uno de otro
9	Arrugar nariz
10	Levantamiento del labio superior
11	Profundidad nasolabial
12	Tiramiento labial esquinial
13	Tiramiento labial frontal
14	Hoyuelo facial
15	Depresión labial esquinial
16	Depresión labial frontal
17	Levantamiento de barbilla
18	Arruga labial
19	Muestrero de lengua
20	Apretar los labios
21	Apretamiento de cuello
22	Embudo labial
23	Morder labios
24	Presión labial
25	Deslizamiento labial
26	Caída de la mandíbula
27	Apretamiento bucal
28	Lamido labial
29	Tracción de la mandíbula
30	Deslizamiento de mandíbula
31	Contracción mandibular
32	Mordida labial
33	Succión de mejillas
34	Inflar mejillas
35	soplido de mejillas
36	Protuberancia de lengua
37	Limpieza labial
38	Dilatado nasal
39	Compresión nasal

Cuadro 1: Lista de unidades de acción y descripciones de acción.

## 6.2. Cómo reconocer las emociones en los demás

### 6.2.1. Alegría

El truco para reconocer correctamente la alegría está en los ojos. Concretamente, en las temidas patas de gallo. Si no aparecen estas características arrugas en el contorno exterior de los ojos, la sonrisa no se considera espontánea, sino una sonrisa social o intencionada. Los dos movimientos musculares o unidades de acción más características de la alegría son la elevación simétrica de las comisuras de los labios (AU12) y el ascenso de las mejillas (AU6) [5].



Figura 4: Rasgos de alegría [5].

### 6.2.2. Sorpresa

Las tres unidades de acción más características de la sorpresa son la elevación simétrica de las cejas hacia el exterior (AU2), la apertura desorbitada de los párpados (AU5) y la caída de la mandíbula (AU26). El truco para reconocer correctamente la sorpresa está en los ojos y en la mandíbula. Los ojos parecen desorbitar, por efecto de la subida de los párpados superiores. Y lo que es más curioso, queda al descubierto la parte blanca de la esclerótica por encima del iris, que normalmente no vemos [5].



Figura 5: Rasgos de sorpresa [5].

### 6.2.3. Tristeza

Las unidades de acción más características de la tristeza son la elevación de las cejas hacia el interior (AU1), la caída de las comisuras de los labios (AU15), y la subida del mentón (AU17). El truco para reconocer correctamente la tristeza está en el comportamiento de las cejas. Lo más habitual es que, al subir hacia el interior, la activación del músculo frontal forme unas arrugas horizontales en el centro de la frente [5].



Figura 6: Rasgos de tristeza [5].

#### 6.2.4. Miedo

El truco para reconocer correctamente el miedo está en fijarnos bien en los ojos, y no confundirlos con los de la sorpresa. Las dos unidades de acción más características del miedo son la elevación de los párpados superiores (AU5), y la retracción o estiramiento horizontal de los labios (AU20) [5].



Figura 7: Rasgos de Miedo [5].

#### 6.2.5. Ira

Las tres unidades de acción más características de la ira son juntar y bajar las cejas sobre la nariz (AU4), la tensión en los párpados inferiores (AU7) y la proyección de la mandíbula hacia adelante (AU29). El truco para reconocer correctamente la ira está en el ceño fruncido, formado por las típicas arrugas sobre la nariz al juntar y bajar las cejas [5].

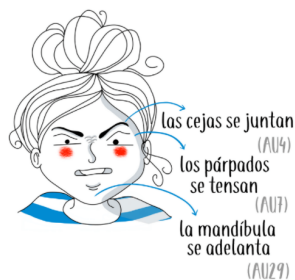


Figura 8: Rasgos de ira [5].

### 6.2.6. Asco

Las dos unidades de acción más características del asco son arrugar la nariz (AU9), y elevar el labio superior (AU10). El truco para reconocer correctamente el asco está en la activación del pliegue nasolabial, fácilmente reconocible porque el labio superior asciende y la nariz se arruga [5].



Figura 9: Rasgos de asco [5].

### 6.2.7. Desprecio

La única unidad de acción características del desprecio es la retracción de una de las comisuras de los labios hacia la mejilla (AU14), formando el típico hoyuelo en un solo lado de la cara, o acentuando su existencia. El truco para reconocer correctamente el desprecio está en el típico hoyuelo, formado en una sola de las mejillas cuando los labios se retraen hacia un lado de la cara. El problema está en que no siempre se aprecia con claridad, sobre todo si la microexpresión es leve y rápida [5].



Figura 10: Rasgos de desprecio [5].

## 6.3. Visión por computadora

### 6.3.1. OpenCV

OpenCV es una biblioteca de software de visión por computadora y de aprendizaje automático (*Machine learning*) de código abierto. OpenCv se creó para aplicaciones de visión por computadora y para acelerar el uso de la percepción de la máquina en productos comerciales. Al ser un producto con licencia BSD, facilita que las empresas utilicen y modifiquen el código. La biblioteca tiene más de 2500 algoritmos optimizados que incluyen un conjunto completo de algoritmos de aprendizaje automático y visión por computadora clásicos y de última generación. Estos algoritmos se pueden utilizar para detectar y reconocer rostros, identificar objetos, clasificar acciones humanas en videos, entre otros. OpenCV tiene más de 47 mil usuarios en la comunidad y un número estimado de descargas superior a 18 millones. La biblioteca se utiliza ampliamente en empresas, grupos de investigación y organismos gubernamentales. Tiene interfaces C ++, Python, Java y MATLAB y es compatible con Windows, Linux, Android y Mac OS [6].



Figura 11: OpenCV logo [6]

### 6.3.2. SimpleCV

SimpleCV es un marco de código abierto para crear aplicaciones de visión por computadora. Con él, obtiene acceso a varias bibliotecas de visión por computadora de alta potencia, como OpenCV, sin tener que aprender primero sobre profundidades de bits, formatos de archivo, espacios de color, administración de búfer, valores propios o almacenamiento de matriz versus mapa de bits. Esta es la visión por computadora simplificada [7].

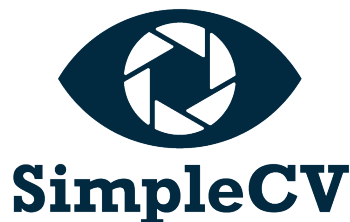


Figura 12: SimpleCV logo [7]

### 6.3.3. Accord.NET Framework

Accord.NET Framework es un marco de aprendizaje automático .NET combinado con bibliotecas de procesamiento de audio e imagen completamente escritas en C#. Es un marco completo para crear aplicaciones de visión por computadora, audición por computadora, procesamiento de señales y estadísticas, incluso para uso comercial. Un conjunto completo de aplicaciones que proporciona un comienzo rápido para comenzar a funcionar rápidamente, y una extensa documentación y wiki ayudan a completar los detalles [8].



Figura 13: Accord.NET Framework logo [8]

### 6.3.4. BoofCV

BoofCV es una biblioteca de código abierto escrita desde cero para la visión por computadora en tiempo real. Su funcionalidad cubre una variedad de temas, procesamiento de imágenes de bajo nivel, calibración de la cámara, detección / seguimiento de funciones, estructura desde el movimiento, detección fiducial y reconocimiento. BoofCV ha sido lanzado bajo una licencia Apache 2.0 para uso académico y comercial [9].



Figura 14: BoofCV logo [9]



## 6.4. Aprendizaje automático

### 6.4.1. Keras

Keras es una API de aprendizaje profundo escrita en Python, que se ejecuta sobre la plataforma de aprendizaje automático TensorFlow. Fue desarrollado con un enfoque en permitir una experimentación rápida. Ser capaz de pasar de la idea al resultado lo más rápido posible es clave para hacer una buena investigación. Ofrece APIs consistentes y simples, minimiza la cantidad de acciones del usuario necesarias para casos de uso comunes y proporciona mensajes de error claros y procesables. También cuenta con una extensa documentación y guías para desarrolladores [10].

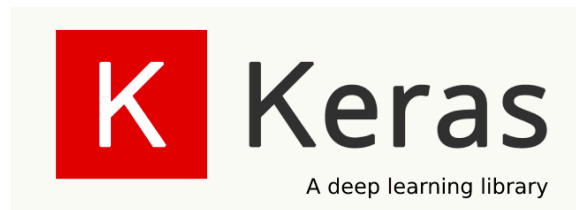


Figura 15: Keras logo [10]

### 6.4.2. TensorFlow

TensorFlow es una plataforma de código abierto de extremo a extremo para el aprendizaje automático. Cuenta con un ecosistema integral y flexible de herramientas, bibliotecas y recursos de la comunidad que les permite a los investigadores innovar con el aprendizaje automático y, a los desarrolladores, compilar e implementar con facilidad aplicaciones con tecnología de AA. TensorFlow ofrece varios niveles de abstracción para que puedas elegir el que se adecue a tus necesidades. Compila y entrena modelos mediante la API de alto nivel de Keras, que ayuda a que los primeros pasos con TensorFlow y el aprendizaje automático sean sencillos [11].



Figura 16: TensorFlow logo [11]

### 6.4.3. PyTorch

Es un paquete de Python diseñado para realizar cálculos numéricos haciendo uso de la programación de tensores. Además permite su ejecución en GPU para acelerar los cálculos. Normalmente PyTorch es usado tanto para sustituir numpy y procesar los cálculos en GPU como para la investigación y desarrollo en el campo del machine learning, centrado principalmente en el desarrollo de redes neuronales. PyTorch es una librería muy reciente y pese a ello dispone de una gran cantidad de manuales y tutoriales donde encontrar ejemplos. Además de una comunidad que crece a pasos agigantados. PyTorch dispone una interfaz muy sencilla para la creación de redes neuronales pese a trabajar de forma directa con tensores sin la necesidad de una librería a un nivel superior como pueda ser Keras para Theano o Tensorflow [12].



Figura 17: PyTorch logo [12]

### 6.5. Kivy

Kivy es un proyecto comunitario, dirigido por desarrolladores de software profesionales. Es una biblioteca de desarrollo de GUI multiplataforma de código abierto para Python y puede ejecutarse en iOS, Android, Windows, OS X y GNU/Linux. Ayuda a desarrollar aplicaciones que hacen uso de una interfaz de usuario multitáctil innovadora. La idea fundamental detrás de Kivy es permitir que el desarrollador cree una aplicación una vez y la use en todos los dispositivos, haciendo que el código sea reutilizable e implementable, lo que permite un diseño de interacción rápido y fácil y una rápida creación de prototipos [13].

Kivy es 100% gratuito, bajo una licencia MIT a partir de la versión 1.7.2 y LGPL 3 para las versiones anteriores. El conjunto de herramientas está desarrollado, respaldado y utilizado profesionalmente. Puede utilizarlo en un producto comercial. El marco es estable y tiene una API bien documentada, además de una guía de programación para ayudarlo a comenzar [13].

El motor gráfico está construido sobre OpenGL ES 2, utilizando una canalización de gráficos moderna y rápida. El kit de herramientas viene con más de 20 widgets, todos muy extensibles. Muchas partes están escritas en C usando Cython y probadas con pruebas de regresión [13].



Figura 18: Kivy logo [13]

## 6.6. CUDA

CUDA es una plataforma de computación paralela y un modelo de programación desarrollado por NVIDIA para computación general en unidades de procesamiento gráfico (GPU). Con CUDA, los desarrolladores pueden acelerar drásticamente las aplicaciones informáticas aprovechando la potencia de las GPU [14].

En las aplicaciones aceleradas por GPU, la parte secuencial de la carga de trabajo se ejecuta en la CPU, que está optimizada para el rendimiento de un solo subproceso, mientras que la parte de procesamiento intensivo de la aplicación se ejecuta en miles de núcleos de GPU en paralelo. Al usar CUDA, los desarrolladores programan en lenguajes populares como C, C ++, Fortran, Python y MATLAB y expresan el paralelismo a través de extensiones en forma de algunas palabras clave básicas. El kit de herramientas CUDA de NVIDIA incluye bibliotecas aceleradas por GPU, un compilador, herramientas de desarrollo y el tiempo de ejecución de CUDA [14].



Figura 19: Nvidia logo [14]

## 6.7. Aprendizaje automático

Aprendizaje automático es la programación de computadoras para optimizar el rendimiento de un criterio utilizando datos de ejemplo o experiencia pasada. Tenemos un modelo definido hasta algunos parámetros, y el aprendizaje es la ejecución de un programa informático para optimizar los parámetros del modelo utilizando los datos de entrenamiento o experiencia pasada. El modelo puede ser predictivo para hacer predicciones en el futuro o descriptivo para adquirir conocimiento a partir de datos, o ambos [15].

El aprendizaje automático utiliza la teoría de la estadística en la construcción de modelos matemáticos, porque la tarea principal es hacer inferencias a partir de una muestra. La función de la informática es doble: primero, en la formación, necesitamos algoritmos eficientes para resolver el problema de optimización, así como para almacenar y procesar la enorme cantidad de datos que tenemos en general. En segundo lugar, una vez que un modelo se aprende, su representación y solución algorítmica para necesidades de inferencia para ser eficiente también. En ciertas aplicaciones, la eficiencia del algoritmo de aprendizaje o inferencia, es decir, su complejidad espacial y temporal, puede ser tan importante como su precisión predictiva [15].

### 6.7.1. Aprendizaje Profundo

El aprendizaje profundo es el subcampo de la inteligencia artificial que se centra en la creación de grandes modelos de redes neuronales que son capaces de tomar decisiones precisas basadas en datos. El aprendizaje profundo es particularmente adecuado para contextos donde los datos son complejos y donde hay grandes conjuntos de datos disponibles. Hoy en día, la mayoría de las empresas en línea y las tecnologías de consumo de alta gama utilizan el aprendizaje profundo. Entre otras cosas, Facebook utiliza el aprendizaje profundo para analizar texto en conversaciones en línea. Google utiliza el aprendizaje profundo para la búsqueda de imágenes y también para la traducción automática. Todos los teléfonos inteligentes modernos tienen sistemas de aprendizaje profundo que se ejecutan en ellos; por ejemplo, el aprendizaje profundo es la tecnología estándar para el reconocimiento de voz y también para la detección de rostros en cámaras digitales. [16].



Figura 20: Red Neuronal como representación del aprendizaje profundo [16]

Para poder desarrollar por completo los objetivos del proyecto, era necesario utilizar el rostro animatrónico diseñado en la fase anterior. En la fase anterior, se diseñó la rotación del cuello para que esta sucediera en un cojinete axial. Utilizando un servo-motor de alto torque fijado en una base estática que, a su vez, hacía girar una pieza sujeta al cojinete axial. Este cojinete estaba montado sobre la base fija haciendo uso de pegamento, lo cual volvía frágil la estructura. El cojinete estaba sujeto a la pieza rotativa por medio de una pieza 3D que entraba a presión en el cojinete, pero que era fácil de remover, cabe recalcar que esta pieza 3D también se encontraba sujeta con pegamento hacia la base rotativa. El cuello tiene dos movimientos más que fueron replicados utilizando otros dos actuadores de alto torque logrando de esta forma que el cuello cuente con tres grados de libertad y logre replicar movimientos bastante verosímiles. Para implementar estos últimos dos servo-motores, se diseñó un sujetador que los mantuviera fijos a la base rotativa, lamentablemente el diseño utilizado era muy frágil y además también estaba sujeto a la base rotativa con pegamento. Al momento de tomar el proyecto era evidente el mal estado de la base del cuello, como se puede ver en la Figura 21, lo que nos llevó a detectar los errores antes mencionados, sin contar con algunos errores de dimensionamiento.

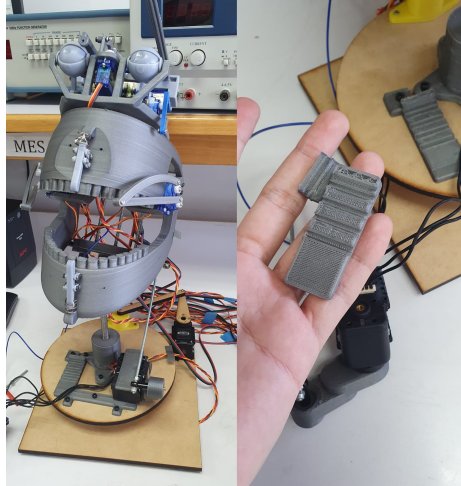


Figura 21: Base de la fase anterior en las condiciones en las que se me entregó.

Como la base en ese momento no era funcional, se decidió rediseñarla para que su funcionamiento cumpliera con la tarea inicial y asegurarnos que su ensamble fuese más seguro y profesional. Así que empezamos por modificar el sistema de sujeción de los servo-motores que estaban montados en la base rotativa. Nuestra intención era diseñar un soporte para los servo-motores que fuese resistente, fácil de montar y que unificara la sujeción de la barra roscada que sostenía el rostro. Por eso optamos por diseñar tres piezas nuevas, una base que se sujetara a la base rotativa, dos carriles que encajaran con la parte inferior de los servo-motores, estos irían pegados a la pieza anterior ya que no nos interesaba que se desmontaran, y una pieza que funcionara como capucha para cubrir las piezas anteriores y que permitiera montar la barra roscada en si para sujetar el rostro, todas las piezas sujetadas por medio de tornillos. Adicionalmente, la capucha de los servo-motores cuenta con un cilindro en el centro que contiene una tuerca M10 para poder artornillar la barra roscada que sujeta el rostro. De esta manera cumplimos con el objetivo de realizar un montaje más profesional y nos aseguramos de tener piezas más resistentes. Como resultado obtuvimos una base como la mostrada en la Figura 22.

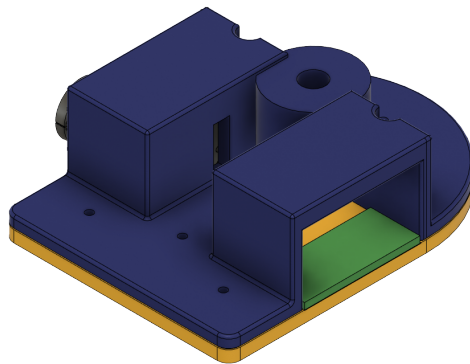


Figura 22: Base modificada para sujetar servo-motores.

Una vez rediseñada la sujeción de estos servo-motores, tuvimos que modificar la base rotativa. El objetivo de modificar esta era ajustar los agujeros para atornillar la base para sujetar servo-motores y aprovechar para cambiar la medida de los tornillos a utilizar, los tornillos que seleccionamos son M3x16. Estos tornillos tienen la longitud necesaria para poder sostener todas las piezas junto con su respectiva tuerca. El resultado final se muestra en la Figura 23.

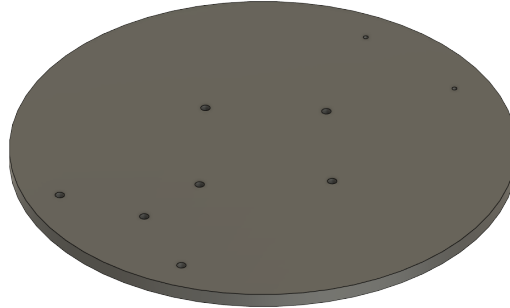


Figura 23: Base rotativa para asegurar el ensamblaje de la base para sujetar los servo-motores.

Con estos dos conflictos resueltos, solo quedaba mejorar el sistema de sujeción del cojinete axial que se encargaba de girar el cuello. Para esto nos basamos en el diseño de la fase anterior y realizamos pequeños cambios en cuanto al dimensionamiento y diseño de las piezas. Principalmente queríamos eliminar el uso de pegamento para sujetar el cojinete así que decidimos ensamblar las piezas a presión. Primero diseñamos la pieza que sujetaba la base rotativa con el cojinete, una pequeña pieza circular impresa en 3D que tenía agujeros que encajaban a la perfección con los agujeros de la base rotativa. Para poder sujetar esta pieza, nos aseguramos que tuviese los encajes para las tuercas M3 donde se enrosacarían los tornillos M3x16. Adicional esta pieza contaba con un pequeño cilindro que sobresalía de la pieza, en este cilindro encajaría el cojinete. Para garantizar que se sujetara de manera satisfactoria utilizamos *crush ribs* y así ensamblarlo a presión, la pieza final se muestra en la Figura 24.

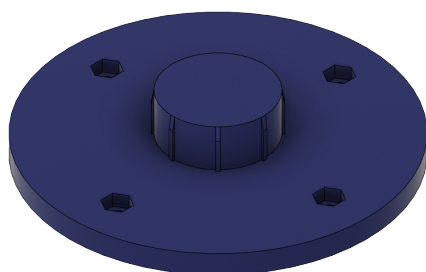


Figura 24: Base cojinete con crush ribs.

La misma idea se utilizó para el sujetador del cojinete que va sujeto a la base fija del proyecto. Esta pieza se diseñó como un rectángulo con un círculo en medio. Este círculo tenía el diámetro exterior del cojinete como medida. Para que el cojinete se sujetara a presión se volvieron a utilizar *crush ribs*. Esta pieza se fabricó en MDF con corte láser. La pieza final se muestra en la Figura 25.

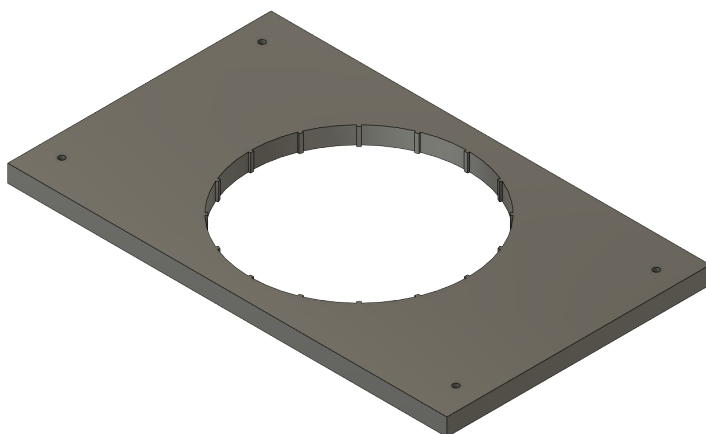


Figura 25: Sujetador para le cojinete con crush ribs.



Finalmente, el ensamblaje de toda la base modificada cumplió con el objetivo principal de haberla rediseñado. Tuvo un acabado más profesional, la sujeción era más segura y las piezas eran más resistentes. A continuación en la Figura 26 se muestra un renderizado del montaje utilizando Autodesk Fusion 360.

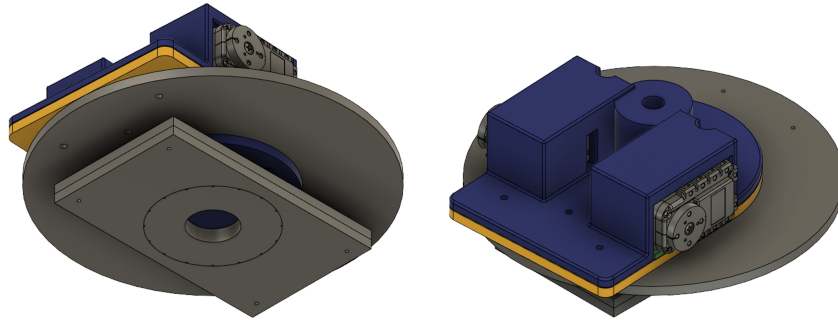


Figura 26: Renderizado de la base.



### 8.1. Modelo de reconocimiento

Para poder reconocer las emociones del usuario se tuvo que crear un clasificador. Este clasificador es un modelo que se tuvo que entrenar con información existente acerca de las emociones que se deseaban. Para poder desarrollar este modelo y su entrenamiento se requirió de herramientas diseñadas para *deep learning*. Se diseñó una red neuronal capaz de recibir información, analizarla y reconocer sus características y finalmente capaz de clasificar la información de entrada en una de las siete clases definidas. Estas clases corresponden a las siete emociones que se deseaban reconocer que son alegría, tristeza, miedo, enojo, asco, neutral y sorpresa. Además se necesitó de un *dataset* con la información que se utilizó tanto como para entrenarlo como para verificar su funcionamiento. Luego, se probaron distintas funciones de activación para el entrenamiento y se verificó cual de estas funciones de activación daba como resultado un mejor clasificador basado en la precisión del mismo.

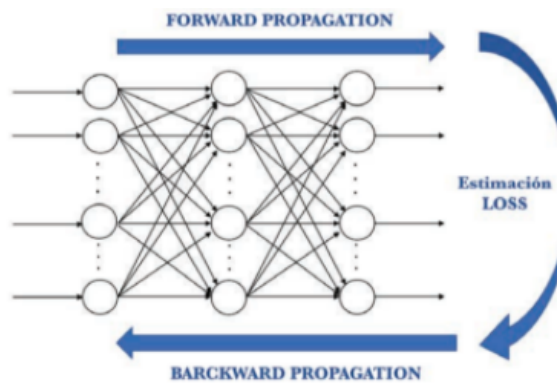


Figura 27: Ejemplo de una red neuronal y su proceso de entrenamiento [17].

### 8.1.1. Trade Study

Característica	Keras	PyTorch	TensorFlow
Nivel API	Alto	Bajo	Alto y Bajo
Arquitectura	Simple, conciso, legible	Complejo, poco legible	No es fácil de usar
Dataset	Datasets pequeños	Datasets grandes, alto rendimiento	Datasets grandes, alto rendimiento
Depuración	Red simple, por lo que la depuración no suele ser necesaria	Buenas capacidades de depuración	Difícil de realizar una depuración
Modelos entrenados	Sí	Sí	Sí
Popularidad	Segundo más popular	Tercero más popular	Más popular
Velocidad	Lento	Rápido	Rápido
Lenguaje	Python	Lua	C++, Python

Cuadro 2: Trade Study de los distintos software de Machine Learning.

Como se puede observar en el Cuadro 2 cada uno de los *software* de aprendizaje profundo cuenta con distintas características que, según la aplicación, puede traer ventajas o desventajas. Para poder realizar la decisión de que herramienta utilizar para crear el modelo de reconocimiento de emociones, se tuvo que dar un peso o prioridad a las características listadas. El peso de cada característica se muestra en el Cuadro 3 junto con el resultado de cada software.

Característica	Keras	PyTorch	TensorFlow
Nivel API	2	1	3
Arquitectura	3	2	1
Dataset	1	2	2
Depuración	3	2	1
Modelos entrenados	1	1	1
Popularidad	2	1	3
Velocidad	1	2	2
Lenguaje	3	1	2
<b>Total</b>	16	13	15

Cuadro 3: Peso de características de los software de deep learning y sus resultados.

Basado en las cualidades de cada software, se le asignó un puntaje de 1 a 3 a cada característica según cual de los softwares era mejor para nuestra aplicación. Las propiedades más importantes en nuestro caso son la popularidad y el lenguaje de programación. La popularidad tiene peso en la decisión ya que al ser una herramienta popular no solo cuenta con la documentación oficial de los desarrolladores, sino que también cuenta con muchos ejemplos y foros de la comunidad que resultaron útiles al momento del desarrollo del modelo. Además, el lenguaje de programación debía ser un lenguaje con el que estuviéramos familiarizados.

### 8.1.2. Desarrollo del modelo

Para el modelo se usó la API para entrenamiento profundo Keras. Lo primero que se hizo fue conseguir un conjunto de imágenes para entrenar y validar el modelo. El conjunto de imágenes utilizado para las primeras pruebas fue el FER-2013. El conjunto de datos consiste en imágenes de rostros con una dimensión de 48x48 píxeles en escala de grises. En la Figura 28 se muestra nueve ejemplos de estas. Entre estas imágenes se cuenta con siete categorías que son enojo, disgusto, miedo, alegría, tristeza, sorpresa y neutral. El total de imágenes es de 28,709 para entrenamiento y 3,589 para validación. Seguido de esto, se procedió a la creación del código del modelo.

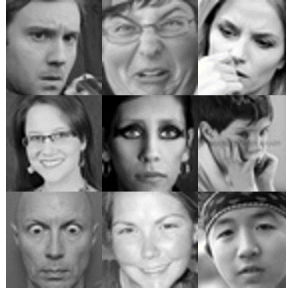


Figura 28: Ejemplo de imágenes contenidas en el dataset de FER-2013 a escala 1:1 [18].

Primero se importaron las librería de Keras, se definió la cantidad de clases que tendrá el modelo. Estas clases son las emociones que se querían detectar, en este caso eran siete clases. Se definió el tamaño del arreglo de la imagen, es decir la cantidad de píxeles que la conforman. Para el caso del conjunto de imágenes de FER-2013 es de 48x48 píxeles. Y definimos la cantidad de muestras que se procesan antes de actualizar el modelo, mejor conocido como *batch size*, utilizamos el valor predeterminado de Keras que es de 32. Seguido se cargó la dirección del conjunto de datos en variables que pudieramos utilizar.

Se procedió a expandir el conjunto de imágenes de manera artificial haciendo algunas modificaciones como: rotar, voltear o reescalar. De esta manera se obtienen más datos desde los que se tienen en propiedad. Seguido, se estandarizaron las cualidades de las imágenes que fueron procesadas para que todas contaran con parámetros similares como: el tamaño, la escala de colores o la clasificación del conjunto de datos. Luego, se definió el tipo de modelo que se utilizó. En este caso se utilizó un modelo secuencial, ya que se quería que la capas se ejecutaran una después de la otra en el orden que se estableció.

El siguiente paso fue crear la red neuronal para el aprendizaje profundo. Para ello trabajamos la red neuronal en siete bloques. El primer bloque cuenta con una capa de convolución con 32 filtros y la función de activación escogida. Las funciones de activación utilizadas para la capas de convolución fueron las funciones elu, relu, selu, sigmoide, softmax, softsign, tanh y la exponencial, cada una de estas funciones tuvieron un resultado diferente al momento de entrenar el modelo. Luego de las capa de convolución se realizó una capa de *maxpooling* con un *dropout* de 0.2. Los bloques dos, tres y cuatro son exactamente iguales, con la única diferencia que cuentan con 64, 128 y 256 filtros respectivamente. El quinto bloque cuenta con una capa *flat*, una capa densa con función de activación elu y un *dropout* de 0.5. El sexto bloque cuenta con una capa densa con función de activación elu y un *dropout* de 0.5. El séptimo bloque es una capa densa con función de activación softmax. En la Figura 29 se

muestra un diagrama de bloques con las respectivas capas de la red neuronal.

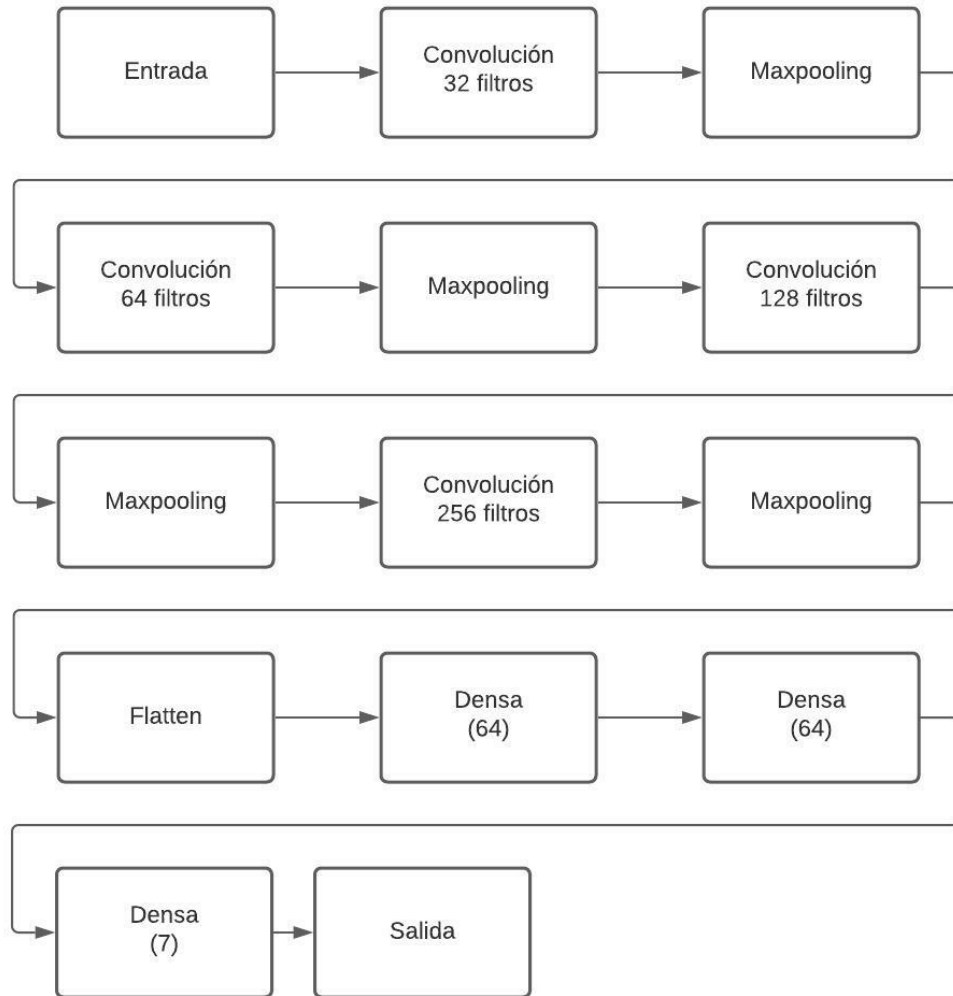


Figura 29: Capas red neuronal.

Antes de poder entrenar el modelo se definieron algunos parámetros acerca de como queríamos que avanzara el entrenamiento. Entre estos parámetros se configuró un *early stop* donde al no tener una mejora durante tres épocas el entrenamiento se detenía. Otro de los parámetros fue el criterio de mejora, para que el modelo tuviera una mejora significativa tenía que tener una mejora en el valor de pérdida arriba de 0.0001. Por último, la cantidad máxima de épocas que iba a realizar el entrenamiento si este no tenía un *early stop*. Con esto se procedió a ejecutar el código y obtener los resultados del modelo.

Es importante tomar en cuenta el *hardware* que se usó, pues esto afecta directamente el tiempo de entrenamiento. Durante el entrenamiento inicial se utilizó un CPU Intel Core i5-9600k de 3.7GHz y un disco duro interno de 2 TB marca Seagate Barracuda de 3.5" a 7200 RPM con una velocidad de transferencia de 220 MB/s. Esto quiere decir que el procesamiento de imágenes y entrenamiento lo ejecutaba el CPU por completo y la transferencia de datos, imágenes en este caso, era lenta. El primer modelo, utilizando el *dataset* de FER-2013, tomó

aproximadamente 25 horas haciendo uso del 70 % del CPU.

Luego se mejoró el *hardware* utilizado para el entrenamiento del modelo a un CPU Intel Core i5-9600k de 3.7GHz, una unidad de almacenamiento sólido NVMe con velocidad de lectura de 1700 MB/s y de escritura de 1550 MB/s marca Gigabyte y una GPU Nvidia 1660 Super de 6GB marca EVGA como se muestra en la Figura 30. En este caso se utilizó la plataforma de computación paralela, CUDA, para procesar las imágenes con el motor gráfico de la GPU, de esta manera se le quitaban procesos al CPU. Además se tenía una transferencia de archivos siete veces más rápida. El mismo modelo con las mismas imágenes tardó aproximadamente 6 minutos haciendo uso del 50 % del CPU.



Figura 30: Hardware utilizado.

En el Cuadro 4 se puede observar que la función de activación con mejores resultados es la Elu. Con 18 épocas realizadas, esto indica que su última mejora fue en la época 15, con una pérdida de 1.3856 y una precisión de 52.18%. También se observa que la función de activación Relu realizó 22 épocas, pero tuvo resultados más bajos, esto nos indica que la relación de mejora por época es distinta para cada función. Para obtener estos resultados, se entrenó el modelo y se realizó una verificación con una porción de imágenes que eran nuevas para el modelo, obteniendo la información de pérdida y precisión del mismo. Así con todos los modelos obtenidos.

<b>Función de activación</b>	<b>Paciencia</b>	<b>Épocas</b>	<b>Loss</b>	<b>Accuracy</b>
<b>Elu</b>	3	18	1.3856	0.5218
<b>Relu</b>	3	22	1.3997	0.4873
<b>Selu</b>	3	14	1.4711	0.4759
<b>Sigmoid</b>	3	8	1.7883	0.2385
<b>Softmax</b>	3	14	1.6411	0.3491
<b>Softsign</b>	3	8	1.7970	0.2435
<b>Tanh</b>	3	7	1.7897	0.2396
<b>Exponential</b>	3	3	nan	nan

Cuadro 4: Parámetros y resultados para las distintas funciones de activación con dropout de 0.2.

Para el modelo final se utilizó un conjunto de datos más grande que contenía aproximadamente 300,000 imágenes. Estos archivos tenían una dimensión de 224x224 píxeles y un formato RGB, como se puede ver en la figura 31. Esto no era conveniente ya que al ser de mayor dimensión y con un formato rgb se tenían que procesar más píxeles, lo que hacía que el entrenamiento mucho más lento. Además, para poder implementar el modelo al detector de emociones, se requería que se entrenara en escala de grises, en el siguiente capítulo entraremos más a detalle. Otra dificultad era que los datos no venían clasificados según la clase, a diferencia del conjunto de datos de FER-2013. En este caso se tenía una carpeta con los archivos y una con la información de cada archivo en formato NPY. Por otro lado, la cantidad de imágenes que se tenían por emoción también fue un problema, ya que en algunas se superaban las 100,000 imágenes, mientras que en otras no se superaba el valor de 4,000. Así que para poder utilizar este conjunto de datos se realizaron modificaciones.



Figura 31: Ejemplo de imágenes contenidas en el dataset de AffectNet a escala 1:1 [19]

Primero se ajustó el tamaño de las imágenes a 48x48 píxeles y se convirtieron a escala de grises. Para esto se usó un código en Python que recorría la carpeta de archivos, tomaba una imagen y le modificaba el tamaño, luego cambiaba el color y la guardaba en una carpeta nueva. El resultado final de esta operación se muestra en la figura 32. Seguido de eso, se realizó la clasificación de imágenes por emoción. Se realizó un código en Python que recorría cada uno de los archivos dentro de la nueva carpeta, seleccionaba una imagen y luego buscaba en la carpeta de información el archivo NPY que le correspondía. Dicho archivo contenía un



array con un número de clasificación para cada emoción (0: neutral, 1: alegría, 2: tristeza, 3: sorpresa, 4: miedo, 5: asco, 6: enojo). Luego, se guardó la imagen en una nueva carpeta con el nombre de la emoción. Finalmente, para que el conjunto de datos fuera lo más homogéneo posible, se igualó la cantidad de datos de todas las clases con la clase que menos datos tenía y se separó un 80 % para entrenamiento y un 20 % para validación, esto de manera manual.



Figura 32: Imágenes de AffectNet modificadas.

El objetivo de utilizar un conjunto de datos más grande, era mejorar el entrenamiento y por ende mejorar los resultados del modelo. Finalmente, se compararon los resultados de ambos modelos entrenados con la función de activación Elu. Para obtener estos resultados se verificó un conjunto de datos, el mismo para ambos modelos, que era completamente nuevo. Como se puede ver en el Cuadro 5 el modelo entrenado con AffectNet tiene una pequeña mejora de 0.48 % en la precisión, sin embargo hay una mejora significativa en la pérdida, mejorando un 19.14 %.

Dataset	Loss	Accuracy
<b>FER-2013</b>	1.9457	0.3640
<b>AffectNet</b>	1.5732	0.3688

Cuadro 5: Comparación entre el modelo entrenado con FER-2013 y el entrenado con AffectNet.

## 8.2. Detector de emociones

El objetivo principal del proyecto era responder al usuario según la emoción que este presente. Para eso se tuvo que generar un detector de emociones utilizando la biblioteca de software de visión por computadora, OpenCV. El software creado requiere del clasificador Haar Cascade, que es un clasificador para detectar rostros y el modelo de reconocimiento de emociones creado en el capítulo anterior. Para poder hacer uso del modelo que se había creado, se necesita la implementación de Keras y no solo de OpenCV. Además se necesita una entrada de video para poder obtener visión de lo que se está analizando. El procedimiento a grandes rasgos para este código era reconocer el rostro haciendo uso de Haar Cascade, luego interpretar e identificar la emoción presentada utilizando el modelo de reconocimiento desarrollado y por último mostrar la emoción detectada.

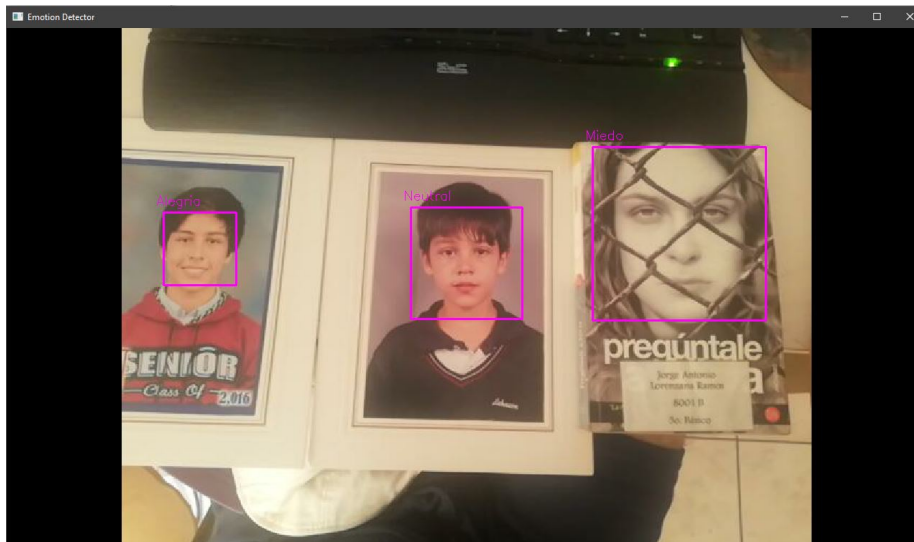


Figura 33: Ejemplo del detector de emociones entrenado con FER-2013.

### 8.2.1. Trade Study

Característica	OpenCV	SimpleCV	BoofCV	Accord.NET
Popularidad	Más popular	Segundo más popular	Tercero más popular	Cuarto más popular
Lenguaje	C++, Python, Java y MATLAB	Python	Java	C#
Última actualización	2021	2015	2021	2017
Documentación	Alta	Media	Baja	Baja
Sistema Operativo	Android, Linux, Windows, Mac	Linux, Windows, Mac	Chrome OS, Android, Linux, Windows, Mac	Linux, Windows

Cuadro 6: Trade Study de los distintos software de visión por computadora.

Como se puede observar en el Cuadro 6 se tomaron cuatro opciones para el desarrollo del software de detector de emociones. Cada una de estas librerías tiene aspectos que se tomaron en cuenta para escoger cual sería el utilizado para nuestra aplicación. De igual manera que para la API escogida en el capítulo anterior, se le dio un peso a cada cualidad de las librerías y en una ponderación se escogió el que tuviera el valor más alto. Esa información la mostramos en el Cuadro 7.

Característica	OpenCV	SimpleCV	BoofCV	Accord.NET
Popularidad	4	3	2	1
Lenguaje	4	4	1	1
Última actualización	4	1	4	2
Documentación	4	3	1	1
Sistema Operativo	4	4	4	4
<b>Total</b>	20	15	12	9

Cuadro 7: Peso de características de los software de visión por computadora y sus resultados.

En el Cuadro 7 se puede observar que la librería más conveniente era la de OpenCV. Las cualidades más importantes a tomar en cuenta para la selección de nuestra librería son el lenguaje y la última actualización. El lenguaje es importante por una razón y es que queríamos mantener la homogeneidad en cuanto al software desarrollado, por eso al haber desarrollado el modelo de reconocimiento en Python se buscaba desarrollar el detector de emociones en el mismo lenguaje de programación. También era importante el año de la última actualización, de esta manera podíamos saber si la librería contaba con soporte de parte de los desarrolladores y las limitantes que esta podría tener. Por estas características se escogió OpenCV, ya que se puede programar en Python y es una librería que se mantiene actualizada además de tener una alta documentación.

### 8.2.2. Desarrollo del software

Para el detector de emociones, primero se desarrolló un código en Python que detectara rostros. Para esto lo primero que se hizo fue buscar un clasificador de rostros, los propios desarrolladores de OpenCV tienen varios clasificadores *Haar Cascade* en su GitHub. Se buscó el archivo “haarcascade\_frontalface\_default.xml” y se descargó. Luego se importó el clasificador utilizando la librería de OpenCV. Seguido, utilizando la entrada de video se captura imagen en tiempo real. Para que el clasificador funcione correctamente se tiene que convertir la imagen a escala de grises. Luego se utiliza el clasificador y se dibuja un cuadrado en la posición donde el clasificador detecta un rostro. Finalmente, se muestra en pantalla la captura de video original con el rectángulo dibujado encima como se muestra en la Figura 34.

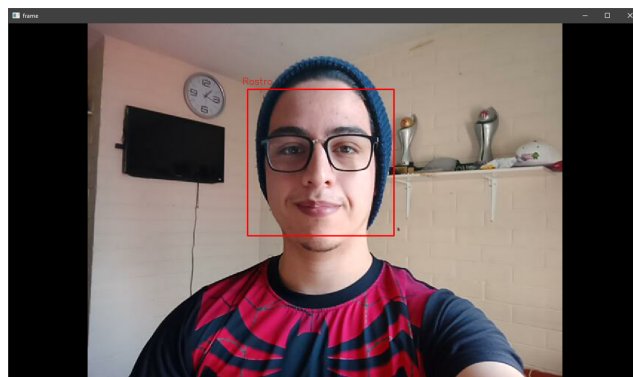


Figura 34: Detector de rostros utilizando Haar Cascade.

Una vez que se tenía el detector de rostros se continuó a programar el detector de emociones. Para esto se tomó el código previamente desarrollado y se le agregó un nuevo clasificador. Este clasificador sería el modelo de reconocimiento creado en el capítulo anterior, era importante que el modelo que creamos se entrenara en escala de grises, ya que el clasificador *Haar Cascade* trabaja con este formato. También se le agregó un arreglo de datos con los nombres de las emociones, es importante agregar los textos en el orden alfabético de las clases originales. Es decir en orden alfabético de las carpetas que contenían las imágenes de entrenamiento. Se tenían las clases con el nombre de la emoción en inglés, por eso el vector de clases quedó en este orden: enojo, disgusto, miedo, alegría, neutral, tristeza, sorpresa. Finalmente, al momento en el que el clasificador *Haar Cascade* detectara un rostro, se tenía que ajustar a una imagen de 48x48 píxeles debido a que el modelo de reconocimiento fue entrenado para esas dimensiones. Luego, se le aplicó reconocimiento de emociones al rostro ajustado y se le agrega un texto sobre el rectángulo, que dibujamos anteriormente, con el texto que corresponde a la emoción detectada como se observa en la Figura 35.

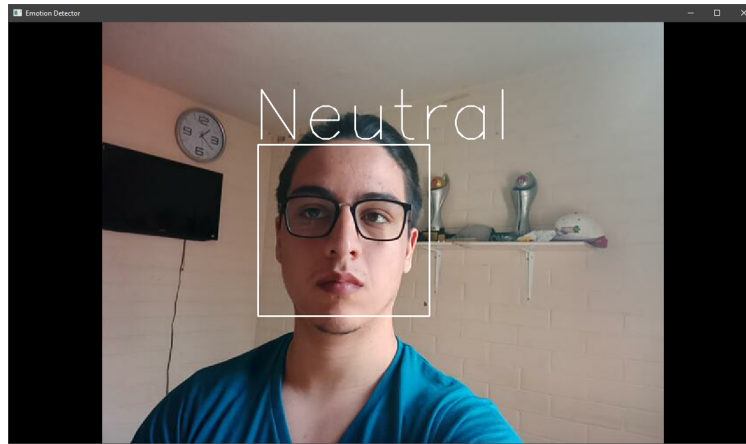


Figura 35: Detector de emociones utilizando Haar Cascade y el modelo de reconocimiento creado por nosotros entrenado con AffectNet.

Finalmente se obtuvo como resultado un programa que es capaz de detectar siete emociones distintas. Cabe mencionar que por falta de datos acerca de la emoción asco, esta no se puede detectar. Sin embargo, fue reemplazada por la emoción neutral. Como se puede ver en la Figura 35 y en la Figura 36 el programa cumple con su objetivo. Adicional a esto, cabe resaltar que utilizando el modelo entrenado con Fer-2013 solo se detectaban cinco emociones, esto por el sesgo que existía al tener menos información de las otras emociones.



Figura 36: Resultado de emociones detectadas (alegría, sorpresa, enojo, disgusto, miedo, triste) con modelo entrenado con el conjunto de datos de AffectNet.



---

### Respuesta a las emociones

---

Se pre-programaron acciones para que el rostro realizara según la emoción que detectara. El rostro tiene dos modos de funcionamiento:

- Modo imitación, con el objetivo de imitar la emoción del usuario.
- Modo reacción, con el objetivo de reaccionar a la emoción del usuario.

#### 9.1. Imitación

Para el modo de imitación, se recibe la emoción detectada y luego se envía por comunicación serial un número que identifica esa emoción. Esta comunicación serial es con el OpenCm9.04 de ROBOTIS que controla el rostro. Este microcontrolador fue programado por Christopher Jenatz, para su trabajo de graduación, con siete configuraciones para los servomotores, una para cada emoción como se muestra en el Cuadro 8. Estas configuraciones cuentan con 19 datos, 4 son para los servomotores XL-320, 3 para los AX-12A y 12 datos para controlar los servomotores TowerPro.

Emoción	Codificación
Neutral	1
Enojo	2
Miedo	3
Felicidad	4
Tristeza	5
Sorpresa	6
Disgusto	7

Cuadro 8: Codificación de las emociones en el microcontrolador programado por Christopher Jenatz.

Para realizar pruebas del correcto funcionamiento de la comunicación, se utilizaron servomotores ajenos al rostro. La única diferencia entre los que estaban en el rostro respecto a los usados para las pruebas, eran los TowerPro que fueron cambiados por servomotores Hitec HS-322HD, que utilizan el mismo protocolo, como se muestra en la Figura 37. Para el control de servomotores tradicionales se utilizó el código existente. Sin embargo, para los motores Dynamixel XL-320 y AX-12A se implementó un procedimiento adicional.

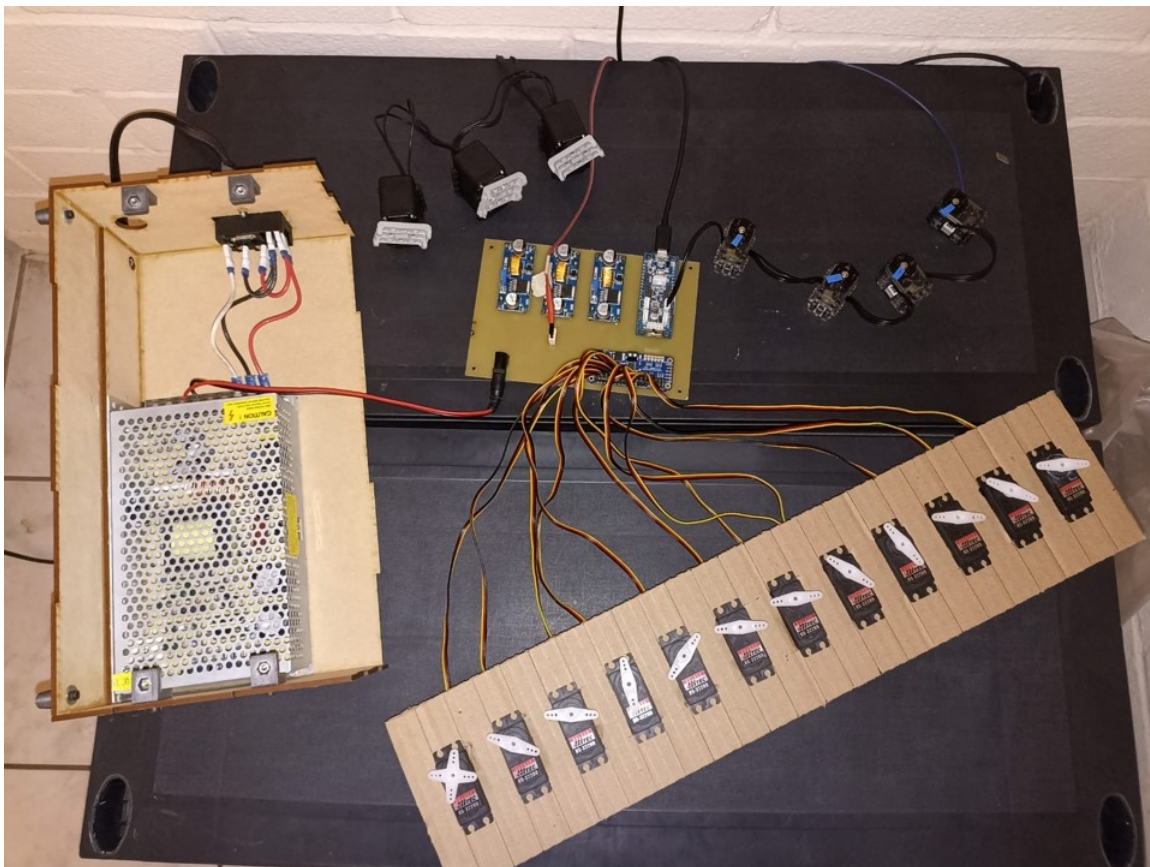


Figura 37: Conexión del circuito con los motores para pruebas.

Los motores Dynamixel pueden conectarse entre sí, creando un bus con el que se pueden controlar todos de manera independiente, gracias a que cuentan con su propio lazo de conexión para comunicación serial asíncrona *Half Duplex*. Para poder enviar datos a un ser-



motor específico, se tiene que indicar el número de identificación de este. Este número es configurado por defecto con valor 1 y para poder utilizar el programa de Jenatz teníamos que cambiar ese valor. Para esto utilizamos el IDE de ROBOTIS OpenCM y el ejemplo `b_setID` de la categoría *Dynamixel Easy* que nos proporcionan. En este código basta con cambiar la variable correspondiente por el número de identificación que queremos asignar. Con este código se va a cambiar el ID de todos los Dynamixel que estén conectados, así que se recomienda conectarlos de uno en uno.

Para verificar que los Dynamixel cuenten con el número de identificación correcto, se puede utilizar el ejemplo `a_getModelNumber` de la misma categoría. Este código despliega todos los motores conectados en el bus con su ID y su modelo.

## 9.2. Reacción

Para el modo reacción, se realizó un árbol de decisiones en Python, un código que recibiera un texto, indicando la emoción detectada y enviara una acción de todas las posibles para esa emoción. El código consiste en una serie de condicionales anidados para seleccionar la emoción. Dentro de cada condicional, se genera un número aleatorio que esté dentro del rango de posibles decisiones y se escoge la reacción.

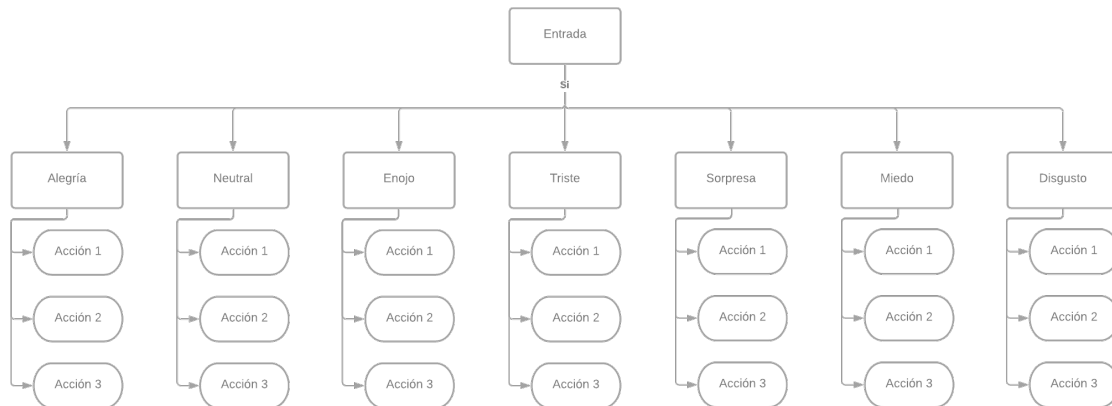


Figura 38: Árbol de decisiones.

Como este código se tenía que ejecutar junto con el detector de emociones y no se quería saturar un archivo de Python con tantas líneas de código, se dejó en un archivo aparte y se volvió función. De esta manera en el código principal solo se tendría que importar desde el archivo del árbol de decisiones la función creada. Luego se llama a la función enviándole la emoción detectada como parámetro y se recibe la acción tomada. El árbol de decisiones se diseñó de manera que el proceso de agregar y modificar respuestas sea simple y directo. Así que esto fue solo un prototipo que tendría que cambiarse para poder ser más amigable con el usuario.

Seguido, se realizó una base de datos en SQLite con siete tablas distintas, una por cada emoción. Para implementarla se creó un código en Python, donde este recibiera un texto indicando la emoción que detecta y luego busca en la base de datos una respuesta para esa

emoción de manera aleatoria.

Lo primero que se realizó fue conectar la base de datos, si la base de datos no está creada, la crea. Seguido de eso se crearon las siete tablas correspondientes a las siete emociones básicas. En cada tabla se agregaron tres posibles respuestas de manera automática, de esta manera la base de datos tiene información predeterminada para poder responder a las emociones. Era importante que se pudieran mostrar las respuestas almacenadas en la base de datos y que estas se pudieran modificar, ya sea agregar o eliminar. En el Cuadro 9 se puede observar los campos que tiene cada tabla de la base de datos

<b>ID</b>	<b>Respuesta</b>
1	Respuesta 1
2	Respuesta 2
3	Respuesta 3

Cuadro 9: Campos de las tablas en la base de datos.

Para mostrar las respuestas se agregaron siete opciones, una para cada emoción y al seleccionar esa emoción se despliega la tabla con las respuestas que se tienen hasta el momento. Para agregar una respuesta nueva se agregó una octava opción que guardaba lo que se ingresaba en la última tabla que se había mostrado. Y por último, para eliminar una respuesta se agregó una novena opción que recibe el número de identificación de la fila y elimina la respuesta con ese número de identificación de la última emoción mostrada.

Finalmente, se tenía que mostrar una respuesta según la emoción que se recibe, para esto se recibe un texto correspondiente a la emoción detectada, es decir alegría, miedo, enojo, tristeza, neutral, sorpresa y disgusto. Con este texto selecciona la tabla correspondiente a esa emoción y selecciona una fila de esa tabla de manera aleatoria. Seguido se muestra que seleccionó la respuesta que corresponde a una fila, especificando la fila que es, de la emoción, especificando la emoción, y mostrando en otro texto la respuesta que dará el rostro animatrónico.

Con esto, se podía tener respuestas de parte del rostro, pero para darle un comportamiento más humano se decidió agregarle movimiento. Para esto se usó el mismo procedimiento que en el modo imitación, enviar por comunicación serial un número que el OpenCM9.04 interpretara como un gesto. Se decidió agregar un campo más a las tablas de la base de datos, como se aprecia en el Cuadro 10, que sería el movimiento del rostro según la respuesta. Este movimiento puede ser cualquiera de las emociones ya codificadas o se podrían agregar más movimientos modificando el código del OpenCM9.04.

<b>ID</b>	<b>Respuesta</b>	<b>Movimiento</b>
1	Respuesta 1	0
2	Respuesta 2	5
3	Respuesta 3	2

Cuadro 10: Campos finales de las tablas en la base de datos.

El objetivo de la interfaz gráfica es monitorear el comportamiento del rostro, es decir qué está viendo y qué está haciendo. Para desarrollarla se utilizó la biblioteca de desarrollo de GUI para Python, Kivy. Es una herramienta que tiene buenos resultados estéticos y simples. La interfaz gráfica contaba con dos ventanas, una para el modo imitación del rostro y la otra para el modo reacción que sirve para interactuar con el usuario. Se utilizan dos lenguajes de programación que son Kivy y Python.

Lo primero que se hizo, fue darle forma a la interfaz gráfica en un archivo de lenguaje Kivy. En este archivo se definieron las ventanas que se iban a estar utilizando. Luego, se le agregaron los artilugios que cada ventana iba a tener. Para la primera ventana se tiene un texto que muestra que está en modo imitación, una imagen que se utiliza para mostrar el video obtenido con OpenCV, un texto que despliega la emoción que se detecta y un botón que permite cambiar al modo reacción. Para la segunda ventana se agregaron los mismos artilugios, realizando el respectivo cambio al texto que indica el modo en el que está operando, y se agregaron dos rectángulos para mostrar la decisión que se tomó del árbol de decisiones y para mostrar la conversación que está teniendo el usuario con el robot.

Lo siguiente a realizar, fue el código de Python para ejecutar la interfaz gráfica. Empezando por importar todas las librerías necesarias de Kivy para desplegar la aplicación y los artilugios. Luego se crearon cuatro clases, una para cada ventana de la interfaz gráfica, una para el administrador de ventanas y una para construir la aplicación. Las primeras tres clases solo se declararon mas no se programó nada dentro de ellas. La cuarta clase es donde se construye la aplicación, esta incluye una función donde se agregaron las ventanas y artilugios al administrador de ventanas, lo cual construye la aplicación. De esta manera se creó el diseño de la interfaz gráfica.

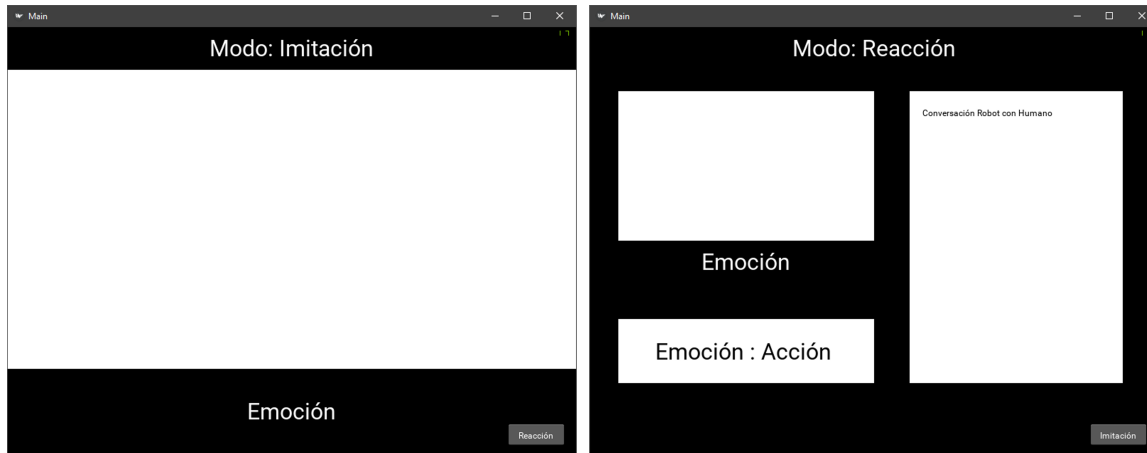


Figura 39: Diseño de las ventanas.

Con el diseño realizado, procedimos a realizar la implementación del detector de emociones. Para esto se agregaron las librerías de OpenCV y Keras, para poder utilizar el código anterior. Se importaron los clasificadores, se agregaron los textos de las emociones y finalmente se agregaron dos funciones más a última clase. Lo primero que se hizo fue agregar una línea de código en la función que ya teníamos definida para poder ejecutar el código del detector emociones en un hilo separado, de esta manera no interrumpir la interfaz gráfica como tal. Luego, se creó una función donde se agregaba exclusivamente el código principal del detector de emociones, de esta manera se ejecutaba como si fuese un programa aparte. La siguiente función toma el video de OpenCV y lo transforma a un formato que el artilugio de imagen pueda desplegar. En la Figura 40 se observa el detector de emociones implementado.

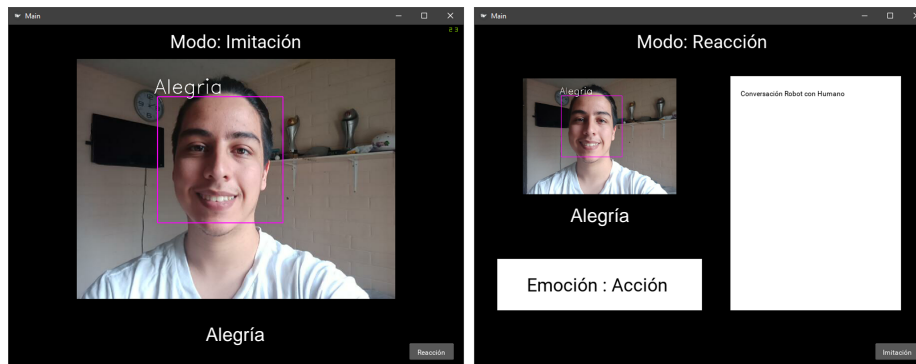


Figura 40: Interfaz gráfica con el detector de emociones implementado.

Lo siguiente en realizarse fue la implementación del árbol de decisiones para el modo de reacción. Primero se importó la base de datos con las decisiones y se agregó un botón en la respectiva ventana. Cada vez que se presionara ese botón tenía que detectar la emoción que se presentara y enviarla a la función para recibir una respuesta. Al presionar el botón se levanta una bandera, cuando esta bandera está arriba el detector de emociones envía la primera emoción que detecta a la función, retorna una acción y la despliega en el rectángulo que está debajo de la imagen. A continuación se muestra el resultado de esta implementación en la Figura 41.



Figura 41: Interfaz gráfica con el árbol de decisiones implementado.

Adicionalmente se implementó una ventana para modificar la base de datos de las respuestas. Así que primero se agregó una nueva ventana con sus respectivos artilugios al archivo kivy del proyecto. Estos son diez botones, uno para cada emoción, uno para agregar respuestas, uno para eliminar respuestas y otro para regresar. Se le agregaron tres entradas de texto, dos para la respuesta a agregar y una para la respuesta a eliminar. De esta manera presionando los botones de las emociones se mostraban las posibles respuestas para cada una y se permitía modificar cada tabla. Para agregar datos, se presiona el botón de una emoción, se introduce la nueva respuesta y presiona el botón de agregar. Para eliminar, se selecciona la emoción correspondiente, se ingresa el identificador de la respuesta a eliminar y se presionaba el botón de eliminar. La ventana para modificar la base de datos se observa en la Figura 42.

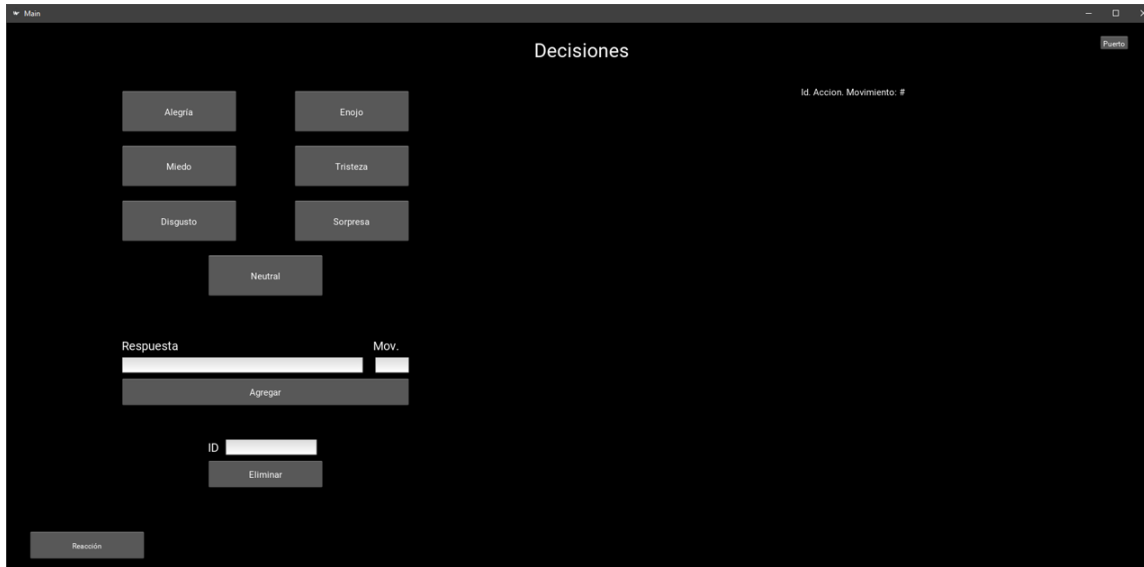


Figura 42: Interfaz gráfica para la base de datos.

Además se modificó la segunda ventana para que tuviera una estética más agradable y se le agregó un botón para poder navegar a la vista de la base de datos. Como se puede observar en la Figura 43, se eliminaron los rectángulos y se muestra la respuesta del rostro.

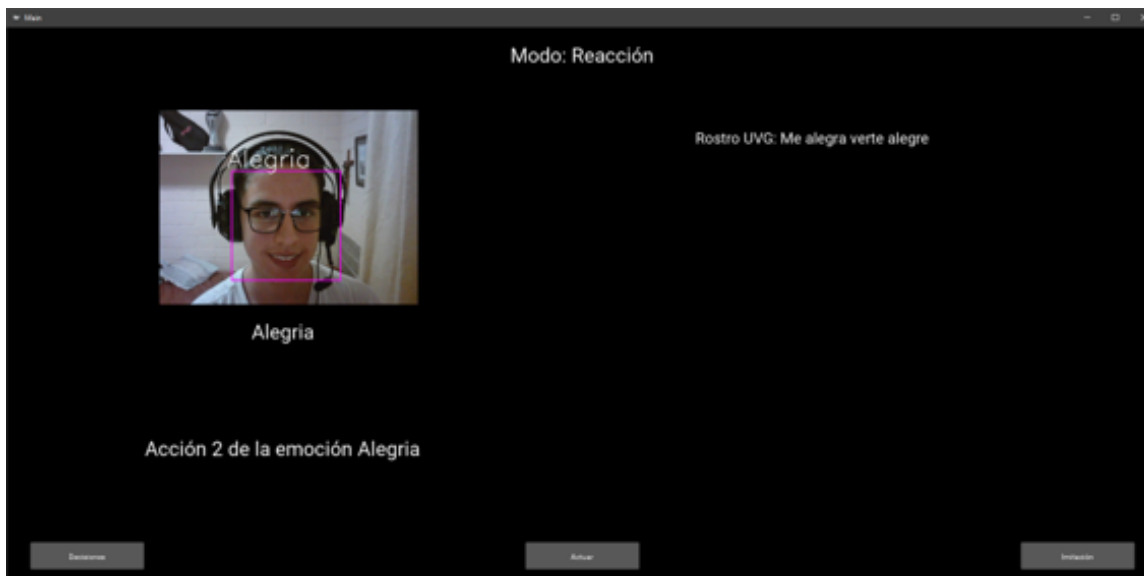


Figura 43: Interfaz gráfica para el modo de reacción.

Luego se trabajó con el modo imitación. A esta ventana se le agregó un botón extra, como se observa en la Figura 44, el cual indica que el rostro debe realizar la imitación de la emoción. Cuando se presiona se levanta una bandera y el rostro va a identificar la primera emoción que detecte. Con esto, va a codificar a un valor numérico la emoción, como se mencionó en el capítulo anterior, y lo va a enviar por comunicación serial.

Hasta este punto la conexión con el rostro se hacía desde la consola, lo que no era muy

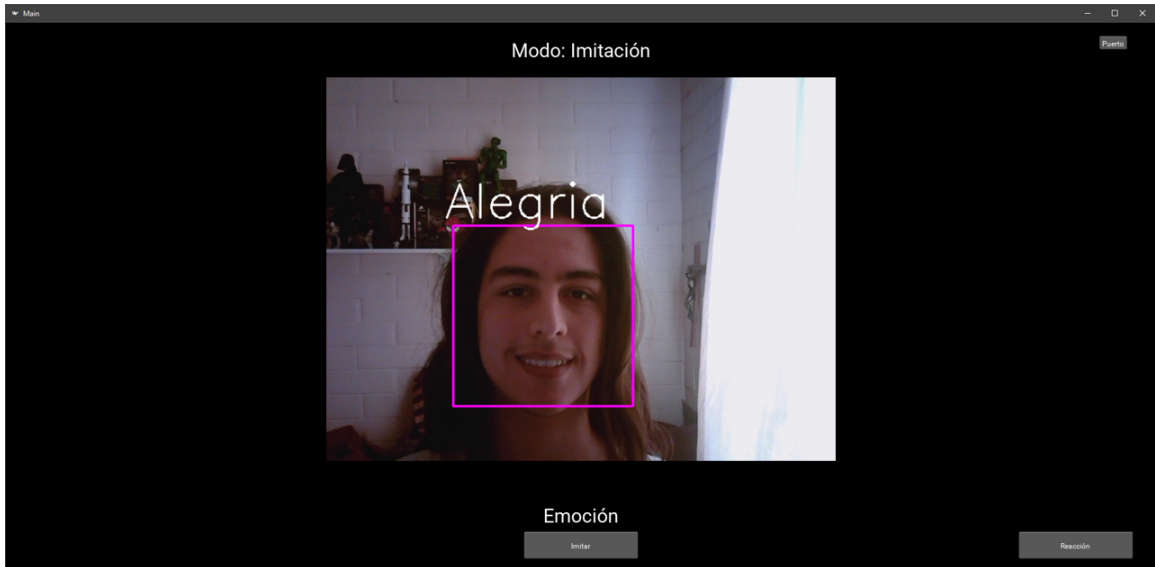


Figura 44: Interfaz gráfica para el modo de imitación.

amigable con el usuario, así que se decidió realizarla desde la interfaz gráfica. Para esto se agregó una ventana más, se se puede observar en la Figura 45 donde se ingresa el puerto que se está utilizando para la comunicación serial. De esta manera se podía conectar con el microcontrolador y poder realizar los movimientos. Esta ventana cuenta con una entrada de texto, un botón para iniciar la conexión, uno para cerrar la ventana y un indicador para saber si la conexión ha sido exitosa. Adicionalmente, se agregó un boton en el resto de ventanas para poder ingresar a esta en cualquier momento. De esta manera, se podrá ejecutar el software sin necesidad de tener la conexión del hardware.

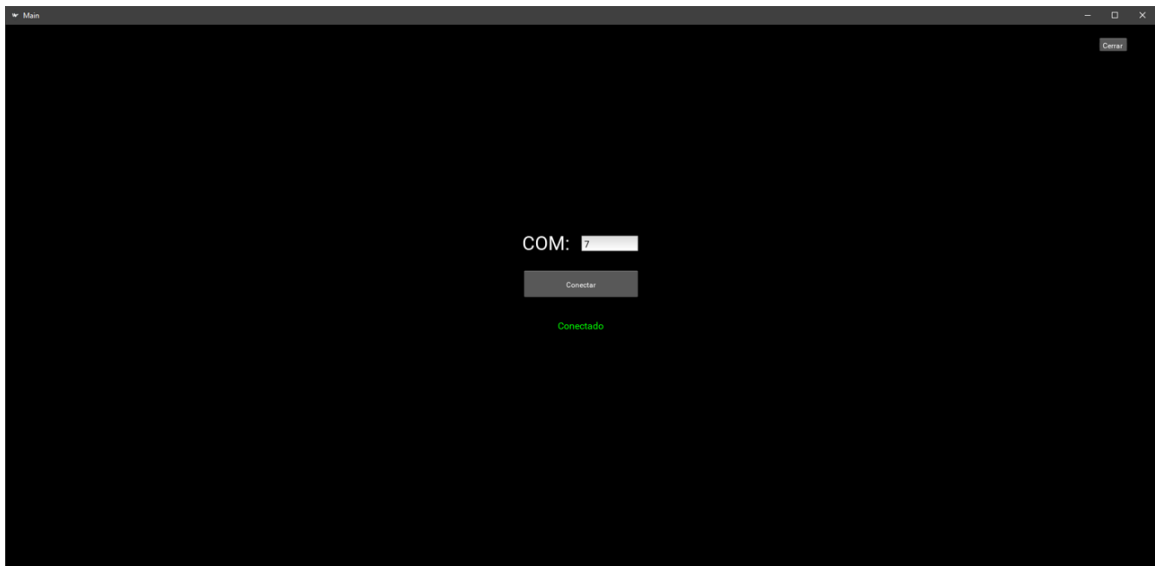


Figura 45: Interfaz gráfica para la conexión del puerto COM.





- Se lograron identificar siete emociones distintas: alegría, tristeza, miedo, enojo, desprecio, sorpresa y neutral.
- Se sustituyó la emoción “asco” por la emoción “neutral” debido a la falta de imágenes en el conjunto de datos utilizados.
- La función de activación Elu en todas las capas de convolución presenta mejores resultados que las funciones: relu, selu, sigmoide, softmax, softsign, tanh y exponencial.
- Se implementó un sistema en el cual el rostro es capaz de imitar la emoción del usuario.
- Se desarrolló un árbol de decisiones con respuestas pre-programadas para cada emoción, que puede ser modificada desde la interfaz gráfica del usuario.
- Se implementó una interfaz gráfica capaz de monitorear el funcionamiento del rostro, la imagen que está recibiendo, cómo lo está interpretando y que respuesta está brindando.



---

### Recomendaciones

---

- Implementar una campaña dentro de la Universidad del Valle de Guatemala para que tanto los alumnos que están inscritos como los que se inscribirán en un futuro sean fotografiados mostrando las siete emociones básicas. De esta manera, crear un conjunto de imágenes propio con suficiente información para mejorar el rendimiento del detector de emociones. Además, de contar con esta información para futuros proyectos de aprendizaje profundo.
- Montar un ambiente controlado con la suficiente iluminación para que los gestos del rostro tengan un mejor contraste. Esto incluye aislar al usuario para que no exista inconvenientes con quienes transitan por el ambiente. Implementar un fondo liso que evite una vista a contraluz del objetivo y que tampoco cree sombras que el software pueda confundir. Que sea capaz de transportarse y montarse por una sola persona. De esta manera tener un proyecto que pueda ser expuesto en distintas partes dentro del campus, como fuera.
- Desarrollar un sistema de detector de audio y un sistema de voz para el rostro animatrónico. De esta manera, poder reconocer e interpretar diálogos del usuario y poder dar respuestas a este de manera auditiva. Implementando micrófonos y altavoces que permitan recibir y transmitir de manera clara. Esto con el objetivo de poder entablar conversaciones Humano-Robot y crear una experiencia más real.
- Aprovechar el underskull y colocar más puntos de referencia para la colocación de una cubierta de látex. Esto dará al rostro una mejor imagen al usuario y ayudará a interpretar sus movimientos, tales como el de los labios al hablar o la emoción que el rostro está mostrando.



- 
- [1] L. Ruano, “Sistema de detección de rostro y reconocimiento de gestos para robot animatrónico,” Tesis de licenciatura, Universidad Del Valle de Guatemala, 2019.
  - [2] *Emotion Detection Using OpenCV and Keras*, <https://medium.com/swlh/emotion-detection-using-opencv-and-keras-771260bbd7f7>, Accessed: 2021-05-20.
  - [3] S. Rulicki y M. Cherny, *CNV comunicación no-verbal: cómo la inteligencia emocional se expresa a través de los gestos*, ép. Management (Granica).: Comunicación. Advanced Marketing s De Rl De Cv, 2012, ISBN: 9789506414979. dirección: <https://books.google.com.gt/books?id=ui7ZKBtZQQ0C>.
  - [4] P. Ekman y J. Serra, *El rostro de las emociones: Cómo leer las expresiones faciales para mejorar sus relaciones*. RBA Libros, 2017, ISBN: 9788490568859. dirección: <https://books.google.com.gt/books?id=HOTODwAAQBAJ>.
  - [5] L. Romero, *Cómo reconocer las 7 emociones básicas de la comunicación no verbal*, <https://www.analisisnoverbal.com/las-7-emociones-basicas-en-la-comunicacion-no-verbal/>, Accessed: 2021-04-09.
  - [6] *About - OpenCv*, <https://opencv.org/about/>, Accessed: 2021-05-06.
  - [7] *SimpleCV*, <http://simplecv.org>, Accessed: 2021-05-06.
  - [8] *Accord.NET Machine Learning*, <http://accord-framework.net>, Accessed: 2021-05-06.
  - [9] *BoofCV*, [http://boofcv.org/index.php?title=Main\\_Page](http://boofcv.org/index.php?title=Main_Page), Accessed: 2021-05-06.
  - [10] *About Keras*, <https://keras.io/about/>, Accessed: 2021-05-06.
  - [11] *Por qué TensorFlow*, <https://www.tensorflow.org/about?hl=es-419>, Accessed: 2021-05-06.
  - [12] *Features/PyTorch*, <https://pytorch.org/features/>, Accessed: 2021-05-06.
  - [13] *Kivy: Cross-platform Python Framework for NUI Development*, <https://kivy.org/#home>, Accessed: 2021-09-09.
  - [14] *CUDA Zone | NVIDIA Developer*, <https://developer.nvidia.com/cuda-zone>, Accessed: 2021-09-20.

- [15] E. Alpaydin, *Introduction to Machine Learning*, ép. Adaptive Computation and Machine Learning series. MIT Press, 2020, ISBN: 9780262043793. dirección: <https://books.google.com.gt/books?id=tZnSDwAAQBAJ>.
- [16] J. Kelleher, *Deep Learning*, ép. MIT Press Essential Knowledge series. MIT Press, 2019, ISBN: 9780262537551. dirección: <https://books.google.com.gt/books?id=b06qDwAAQBAJ>.
- [17] J. Torres, *DEEP LEARNING Introducción práctica con Keras*, ép. Watch this space. Lulu.com, 2018, ISBN: 9780244078959. dirección: <https://books.google.com.gt/books?id=ju1mDwAAQBAJ>.
- [18] *fer2013/Kaggle*, <https://www.kaggle.com/deadskull17/fer2013>, Accessed: 2021-08-02.
- [19] A. Mollahosseini, B. Hasani y M. H. Mahoor, “AffectNet: A Database for Facial Expression, Valence, and Arousal Computing in the Wild,” *IEEE Transactions on Affective Computing*, vol. PP, n.º 99, págs. 1-1, 2017.

## CAPÍTULO 14

---

Anexos

---

Repositorio de GitLab: <https://gitlab.com/kekellner/rostro-animatronico-jorge-lorenzana/-/tree/lor17302-main-patch-85275>

