Implementación de una interfaz de uso y control para el rostro animatrónico de la Universidad del Valle de Guatemala

Jorge Fernando Lanza Salguero

UVG UNIVERSIDAD DEL VALLE DE GUATEMALA

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Implementación de una interfaz de uso y control para el rostro animatrónico de la Universidad del Valle de Guatemala

Trabajo de graduación presentado por Jorge Fernando Lanza Salguero
para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

2024

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Implementación de una interfaz de uso y control para el rostro animatrónico de la Universidad del Valle de Guatemala

Trabajo de graduación presentado por Jorge Fernando Lanza Salguero
para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

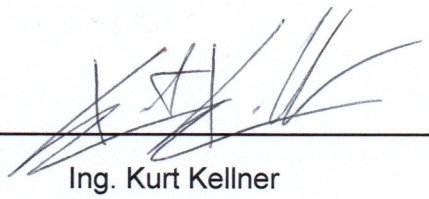
2024

Vo.Bo.:

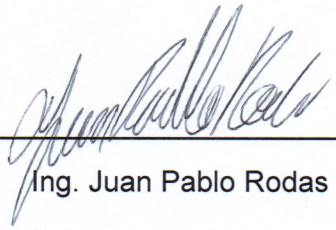
(f) 

Ing. Kurt Kellner

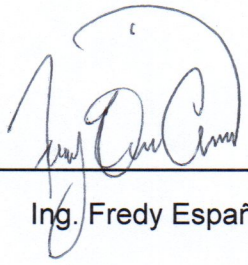
Tribunal Examinador:

(f) 

Ing. Kurt Kellner

(f)) 

Ing. Juan Pablo Rodas

(f) 

Ing. Fredy España

Fecha de aprobación: Guatemala, 13 de enero de 2024

Prefacio

La ejecución de este trabajo de graduación no se hubiera podido llevar a cabo sin el apoyo incondicional de las personas que se relacionan conmigo directamente. Primeramente quisiera agradecer la provisión de mi padre, quien me brindó desde apoyo emocional hasta apoyo económico a lo largo de todo mi trayecto universitario. Agradezco todos los consejos que me ha dado y su guía en momentos difíciles durante la carrera. Así mismo, quiero agradecerle a mi madre, quien con mucho esfuerzo y cariño me a apoyado emocionalmente con consejos, frases y su presencia.

Debo reconocer el apoyo que he tenido de mis compañeros Fernando Caceros, Fredy Godoy y Alejandro Rodríguez durante los 5 años de carrera. Ellos me apoyaron e inspiraron durante toda la carrera para obtener mejores resultados en proyectos personales. Agradezco profundamente su guía y apoyo al igual que su amistad y compañerismo.

Expreso mi gratitud a mi asesor y catedráticos que me han apoyado en la realización de este trabajo de graduación. Su guía fue indispensable para los buenos resultados obtenidos, al igual que su comprensión y apoyo.

Índice

Lista de figuras

Lista de cuadros

Resumen

Actualmente existen dos grandes programas de software en el rostro animatrónico de la Universidad del Valle de Guatemala. Estos son: un sistema de reconocimiento de emociones por medio de una cámara, y un sistema de *chatbot* con reconocimiento de voz y capaz de reproducir audio. Estos programas funcionan por separado, pero aún no existe una integración para que el rostro animatrónico posea ambas características al mismo tiempo. Así mismo, tampoco se cuenta con una interfaz gráfica de usuario unificada para su uso.

El presente trabajo tiene como objetivo el diseño de una aplicación que contenga integrado ambos programas al igual que una interfaz gráfica de usuario más formal y amigable para un evento o una presentación. También se busca una forma de empaquetamiento del programa para poder ser ejecutado en diferente hardware y así asegurar que pueda correr en distintas computadoras. Como último objetivo, se busca crear un protocolo de comunicación entre la computadora y el microcontrolador del rostro animatrónico.

Para lograr esto, primeramente se realizaron unas pruebas a los códigos existentes con el objetivo de encontrar los escenarios ideales para la mejor ejecución de dichos programas. Luego se realizó una transcripción de los programas basándose en la hoja de estilización PEP-8 ya que al ser un programa tan grande y un proyecto que seguirá siendo trabajado, se debe procurar que el código sea lo más comprensible posible. La creación de la interfaz gráfica fue realizada con el lenguaje de Python utilizando la librería Kivy.

La interfaz gráfica se dividió en tres pantallas para una mejor presentación de todas las partes de este proyecto. En la primera pantalla se colocaron la integración de ambos sistemas existentes. En la segunda se colocó el estado de los motores al igual que una interacción manual con los servomotores físicos. Por último se colocó la modificación de las base de datos al igual que el entrenamiento de la red neuronal del *chatbot*.

Se utilizó la comunicación serial como vía para la creación del protocolo. Esta comunicación se hace a través del puerto UART del microcontrolador y un puerto UART de la computadora. Así mismo se programó en un archivo ino el comportamiento del microcontrolador para la recepción de datos. Esto para lograr que el microcontrolador comprenda a la perfección el protocolo de comunicación creado.

Por último, se empaquetó todas las librerías para poder ser ejecutadas en un ambiente virtual de Python también conocida como *venv*. Se diseño un manual para poder instalarlo en cualquier máquina que utilice Windows 10 u 11 y que cuente con permisos de administrador. Al seguir los pasos sugeridos todas las librerías de Python se instalan automáticamente de manera local en la carpeta seleccionada por lo que no afectará el funcionamiento de la máquina en general.

Abstract

Currently there are two major software programs in the animatronic face of the Universidad del Valle de Guatemala. These are: an emotion recognition system using a camera, and a chatbot system with voice recognition and capable of playing audio. These programs work separately, but there is still no integration so that the animatronic face has both characteristics at the same time. Likewise, there is no unified graphical user interface for one use either.

The objective of this work is to design an application that contains both programs integrated as well as a more formal and friendly graphical user interface for an event or presentation. A form of packaging the application is also being sought to be able to run on different hardware and thus ensure that it can run on different computers. As a final objective, the aim is to create a communication protocol between the computer and the microcontroller of the animatronic face.

To achieve this, tests were first carried out on the existing codes with the aim of finding the ideal scenarios for the best execution of said programs. Then a transcription of the programs was made based on the PEP-8 stylization sheet since this is a proyect that will be continue to be worked on, we must ensure that the code is as understandable as possible. The creation of the graphical interface was carried out with the Python language using the Kivy library.

The graphical interface was divided into three screens for a better presentation of all parts of this project. The integration of both existing systems was placed on the first screen. In the second, the state of the motors was placed as well as a manual interaction with the physical servomotors. Finally, the modification of the database was carried out as well as the training of the neural network of the chatbot.

Serial communication was used as a way to create the protocol. This communication is done through the UART port of the microcontroller and a UART port of the computer. Likewise, the behavior of the microcontroller for receiving data was programmed in an ino file. This is to ensure that the microcontroller perfectly understands the communication protocol created.

Finally, all the libraries were packaged to be able to be executed in a virtual Python environment also known as *venv*. A manual was designed to be able to install it on any machine that uses Windows 10 or 11 and that has administrator permissions. By following the suggested steps, all Python libraries are automatically installed locally in the selected folder, so it will not affect the operation of the machine in general.

CAPÍTULO 1

Introducción

Actualmente la Universidad del Valle de Guatemala se encuentra desarrollando un proyecto de un rostro animatrónico. Este proyecto ha sido trabajado por 4 alumnos a lo largo de los años. Estos han aportado diseños de mecanismos, control de servomotores, un sistema de reconocimiento de voz con un *chatbot* y un sistema de reconocimiento de emociones por medio de visión por computadora. Con estos sistemas se ha avanzado cada vez más para lograr un animatrónico totalmente autónomo capaz de interactuar con las personas y su entorno, pero dichos sistemas son independientes entre sí y no pueden ser presentados como un solo proyecto.

Este trabajo de graduación busca unir todos estos sistemas en una sola interfaz. Se quiere crear un rostro animatrónico con subsistemas complementarios y que colaboren entre sí. Para esto se creó una interfaz gráfica de usuario ejecutado por una computadora, capaz de tener todos los sistemas y programas existentes hasta el momento. Se buscó integrar el sistema de visión por computadora junto con el *chatbot* para obtener así un animatrónico más realista. Con esta interfaz gráfica se pretende no sólo servir como guía de los sistemas presentes, sino también una forma de explicar diferentes ramas de ingeniería que la carrera Mecatrónica utiliza. Por ejemplo se puede observar la programación de redes neuronales para los sistemas computarizados, la creación de sistemas mecánicos para lograr los movimientos de un rostro humano, la electrónica para la alimentación y control de servomotores al igual que programación de microcontroladores.

Para lograr una correcta integración del proyecto en la interfaz se utilizó una hoja de estilización llamada PEP-8. Esta hoja consiste en un reglamento para que la programación de aplicaciones sea más legible y comprensible por otros desarrolladores. Esto se realizó con el objetivo de que se siga desarrollando el proyecto en el futuro y los estudiantes se puedan enfocar en la implementación de nuevas características y no se pierda tiempo en la comprensión del programa. Así mismo, se utilizó la librería de Kivy del lenguaje de Python para la programación de la interfaz. Esta librería se especializa en la creación de interfaces gráficas debido a su amplio catálogo de elementos donde también permite una versatilidad en la ejecución del programa en diferentes dispositivos.

CAPÍTULO 2

Antecedentes

2.1. Diseño de cabeza y cuello animatrónicos con 18 grados de libertad

Jenatz realizó el proyecto de un rostro animatrónico como trabajo de graduación en la Universidad del Valle de Guatemala. El investigador buscó la creación y el diseño mecánico y electrónico de un animatrónico conformado por un rostro y un cuello. El autor planteó como objetivo que este tuviera 18 grados de libertad, y sus movimientos sean lo más suavizados posibles. [1]

Mediante software CAD Autodesk Fusion 360 se realizó un diseño modular para el mecanismo del proyecto. El animatrónico presentado por el autor posee movimientos de: mandíbula inferior, labios, cejas, párpados, ojos en dos direcciones y rotaciones para el cuello. Para poder realizar estos movimientos, el investigador utilizó servo motores TowerPro SG90, Dynamixel XL-320 y Dynamixel AX-12A en donde algunos de estos últimos requirieron de un controlador PID. El autor programó cada actuador para poder realizar movimientos parecidos a los de una persona, para esto, utilizó protocolos Dynamixel 1.0 para los AX-12A, Dynamixel 2.0 para los XL-30 e I2C para los SG90. [1]

2.2. Rediseño e implementación de rostro animatrónico

En el año 2023 una tesis aún no publicada realiza por Soto tenía como objetivo el rediseño del sistema mecánico del rostro animatrónico de la Universidad del Valle de Guatemala. El investigador buscó el rediseño del sistema mecánico de las cejas, párpados y cuello del proyecto. Para lograr dichos objetivos primeramente el autor migró los archivos existentes de Fusion 360 a inventor. Posteriormente, Soto procedió a realizar cambios al sistema mecánico de las partes anteriormente mencionadas ya que estas fallaban o no eran representativos para los movimientos de un rostro real. [2]

2.3. Implementación de un chatbot en tiempo real en rostro animatrónico

Fuentes propuso una investigación con el objetivo de desarrollar un sistema de interacción verbal en tiempo real para el rostro animatrónico de la Universidad del Valle de Guatemala. Para poder lograr esto, primeramente, el investigador implementó un sistema de reconocimiento de voz con la técnica Voz a Texto para poder lograr una interacción en tiempo real entre animatrónico y usuario. El autor, utilizó una librería de Python llamada *Speech_ Recognition* junto con una Interfaz de Programación de Aplicaciones (API, por sus siglas en ingles) de Google, permitiendo así que funcione tanto en inglés como en español latino. [3]

Posteriormente, el investigador procedió al diseño y creación del *chatbot* utilizando un modelo basado en recuperación mediante machine learning. Este consistía en respuestas predeterminadas, guardadas en una base de datos con formato JSON y una estructura tipo diccionario. Ya que una pregunta puede ser formulada de distintas maneras igualmente correctas por lo que cada diccionario de la base de datos cuenta con un identificador que el investigador utilizó para entrenar la red neuronal. El autor utilizó técnicas creadas en una rama de inteligencia artificial llamada Procesamiento de Lenguaje Natural para procesar los datos del reconocimiento de voz, y seleccionar la respuesta predeterminada correcta. [3]

2.4. Diseño de un sistema para reconocimiento facial, gestos y voz para un rostro animatrónico

González planteó una tesis que tenía como objetivo diseñar e implementar un software para la detección facial, gestos y voz en el rostro animatrónico de la Universidad del Valle de Guatemala. Para lograr dicho objetivo, el investigador utilizó Python como lenguaje del software y utilizó distintas librerías ya creadas en este lenguaje: *OpenCV*, *Speech_ Recognition* y *Pyttsx3x* para poder programar las tareas de su software. En la tarea del reconocimiento facial y gestos utilizó la librería *OpenCV* ya que esta genera marcas en rostros para una mejor detección. Junto con dicha librería y algoritmos de clasificación como Haar cascade, Histogramas de Gradientes Orientados y Detección Linear mediante Máquinas de soporte Vectorial, el autor logró identificar 6 emociones: alegría, sorpresa, ira, tristeza, asco y miedo. Con la implementación de estos recursos, el autor menciona que su software tiene la capacidad de reconocer el gesto con una precisión de un 64.75 % en promedio ya que algunas emociones presentan un mayor índice de reconocimiento que otras.[4]

Por otro lado, para la tarea de reconocimiento de voz, el investigador utilizó la librería *Speech_ Recognition* junto con *PocketSphinx*. Este último es un motor de reconocimiento de voz de código abierto el cual tiene la ventaja de ser totalmente offline, el autor lo adaptó al idioma español ya que el motor no contaba con esto. Por el contrario, para la generación una salida de texto a voz utilizó la librería *Pyttsx3x*. El autor menciona que esta función de reconocimiento de voz posee una limitante en la longitud de las frases para detectar.[4]

2.5. Sistema de reconocimiento facial y emociones para rostro animatrónico

El proyecto tenía como objetivo el desarrollo e implementación de un sistema de visión por computador capaz de reconocer emociones. Se buscaba implementar el software en un rostro animatrónico de la Universidad del Valle de Guatemala para que este fuera capaz de generar una respuesta congruente al momento de una interacción Humano-Robot. Para lograr esto, Lorenzana utilizó el lenguaje de Python ya que en este existen diferentes librerías que podía utilizar como *OpenCV* para el reconocimiento facial y *Kivy* para el desarrollo de una interfaz gráfica. [5]

Para el modelo de reconocimiento de emociones, el investigador implementó Keras. Esta es una *API* de aprendizaje profundo escrita en Python la cual se ejecuta sobre la plataforma de *TensorFlow*. El objetivo de utilizar esto era crear una red neuronal que, junto con el algoritmo de clasificación *Haar cascade*, fuera capaz de detectar siete emociones: alegría, tristeza, miedo, enojo, desprecio, sorpresa y neutral. Para el entrenamiento de esta red neuronal, el autor utilizó la base de datos FER-2013 para el primero modelo de la red y *AffectNet* para el segundo modelo. Este último poseía 300,000 imágenes de rostros a color a diferencia de la primera base de datos utilizada. Al final del entrenamiento de ambos modelos, el autor menciona que eran capaces de detectar las siete emociones planteadas con una presición del 36.4 % y una pérdida del 1.94 % para el primer modelo, una presición del 36.88 % y una pérdida del 1.57 % para el segundo.

2.6. Hablando con emociones

Los autores buscan la creación de un agente virtual capaz de interactuar con los usuarios de manera natural y emocional. Con la tecnología actual, los agentes virtuales son capaces de mantener una conversación muy realista, mencionan los autores. Por el contrario, toda la comunicación no verbal aún no se ha realizado de tal manera que sea natural para las personas interactuar con una inteligencia artificial. Para los autores las expresiones faciales y las emociones son muy importantes en la comunicación no verbal y no existe una solución exacta para programar el comportamiento de estas. [6]

Los investigadores mencionan que, existen aspectos clave para lograr una expresión facial natural: el tiempo y la intensidad de estas. Para la sincronización, los autores determinaron tres momentos clave, siendo el primero *onset*. Este es el tiempo que le tomaría a los músculos de pasar de una expresión facial neutral a la deseada. Luego llega el momento *apex*, el tiempo en que se mantiene la expresión facial. Por último el momento *offset*, que al contrario de *onset*, es el tiempo que le toman a los músculos pasar de una expresión facial a una neutral. Los investigadores mencionan que estos tiempos pueden variar según la emoción que se este representando, pero todas poseen este mismo patrón. Por otro lado, para el aspecto de la intensidad, los autores mencionan que esta puede variar según el movimiento de los músculos ya que estos se pueden mover de posición sólo un poco o pueden llegar hasta donde fisiológicamente sea posible. También mencionan que, dependiendo de la intensidad, no necesariamente todos los músculos que podrían llegar a participar en la expresión facial de la emoción se muevan. [6]

2.7. Cabeza mecatrónica con inteligencia emocional y artificial

Los autores plantearon como objetivo en su investigación la creación de una cabeza mecatrónica realista con alta capacidad gestual y apariencia de un hombre de la tercera edad capaz de poseer inteligencia artificial y emocional. Para el desarrollo de la cabeza, los investigadores tomaron en cuenta los músculos más significativos del rostro, involucrados en la expresión facial siendo estos: frontal, occipital, orbicularis oculi, cigomático, orbicularis oris, temporal y master. Para el movimiento de estos músculos de manera realista se utilizaron 12 servomotores que asociaron a los diferentes músculos anteriormente mencionados obteniendo así 23 grados de libertad. [7]

Para este proyecto, los autores utilizaron una programación SMACH. Esto es una librería ROS-independiente de Python para la construcción de máquinas de estados jerárquicos. Este tipo de programación es orientado a eventos le permite al animatrónico utilizar el reconocimiento facial y de voz para poder interactuar de forma más realista y no necesita siempre regresar a un mismo punto o tener una programación cíclica. Con esta programación los autores concluyen un notable grado de realismo en la interacción humano-robot obteniendo una mejora en este aspecto del 35 %. [7]

2.8. VisualHFSM 5: mejoras en la programación de robots con autómata en JdeRobot

Rey y Cañas tenían como objetivo la demostración de la mejora de una herramienta llamada VisualHFSM. Esta provee el uso de máquinas de estados finitos jerárquicos en robots autómatas para programar su comportamiento. Los autores mencionan que esta técnica de programación permite una conexión entre estados muy buena, esto quiere decir transición de tareas del autómata muy eficiente y un mejor comportamiento del mismo. Los investigadores mencionan que en esta nueva versión de la herramienta existen mejoras en la representación gráfica de los estados y en la generación automática de código para este. Los autores para poder probar estas mejoras, realizaron pruebas tanto en simulaciones como físicas utilizando drones. Los autores concluyen que el uso de máquinas de estados finitos jerárquicos es una herramienta muy poderosa para representar comportamientos complejos de una manera eficiente y precisa. [8]

CAPÍTULO 3

Justificación

La Universidad del Valle de Guatemala siempre ha buscado llegar a interesar a los jóvenes en investigación y desarrollo para fomentar más proyectos en Guatemala. Una de las maneras de poder lograr esto es a través de actividades de captación, en donde se presentan las carreras pertenecientes a la universidad y actividades representativas de estas. El objetivo general del proyecto "Rostro animatrónico" es precisamente esto, poder convencer y animar a más jóvenes a participar en estas carreras de investigación y desarrollo con demostraciones en la rama animatrónica de alto nivel.

Actualmente, el proyecto cuenta con diferentes programas creados por estudiantes durante su trabajo de graduación. Existen códigos para: reconocimiento de voz, reconocimiento facial, detección de emociones y un *chatbot* para que el usuario interactúe con el animatrónico. Todos estos son muy útiles, pero son totalmente independientes entre sí y no existe forma que, el animatrónico, pueda tener todas estas funciones al mismo tiempo ni una interfaz de usuario que pueda ser útil para el propósito del reclutamiento.

Es por ello que se considera indispensable la integración de todos los programas en un solo código. Con este proyecto de graduación se pretende tener una interfaz gráfica que sea fácil de entender para los jóvenes con pocos conocimientos en esta área, pero sea lo suficientemente completa para abarcar todo lo que el animatrónico es capaz de realizar. También se busca que este código sea fácil de instalar en computadoras ya que no siempre se cuenta con el mismo equipo todo el tiempo o para todas las actividades.

CAPÍTULO 4

Objetivos

4.1. Objetivo general

Implementar una interfaz de uso y control para el rostro animatrónico de la Universidad del Valle de Guatemala.

4.2. Objetivos específicos

- Validar el sistema de reconocimiento de rostro y emociones implementado en el rostro animatrónico
- Validar el sistema de reconocimiento de voz implementado en el rostro animatrónico
- Implementar una interfaz gráfica que muestre el reconocimiento facial y voz, estado de motores y sensores del rostro animatrónico
- Implementar una interfaz que permita la modificación y expansión de la base de datos de respuestas del *chatbot*
- Diseñar un protocolo de comunicación entre la interfaz implementada y el hardware
- Empaquetar todas las librerías y códigos para lograr una instalación rápida en distintas computadoras

CAPÍTULO 5

Alcance

El alcance de este trabajo de graduación abarca la creación de una interfaz gráfica de usuario. Se buscó poder presentar de manera ordenada y clara todos los sistemas que abarcan el rostro animatrónico. Los sistemas presentados son: sistema de identificación de emociones, sistema de *chatbot*, estado de motores y la modificación de la base de datos del *chatbot* al igual que su entrenamiento.

Así mismo también se propuso un protocolo de comunicación para obtener un enlace eficiente entre la interfaz de usuario y el microcontrolador. Se buscó que el microcontrolador entienda en todo momento lo que esta ocurriendo con la interfaz gráfica utilizando la comunicación serial. Es por ello que se propuso un código para el microcontrolador capaz de entender dicho protocolo.

Por otro lado, se busca una forma de instalación rápida del programa para que cualquiera pueda utilizar el rostro animatrónico. Se propone una guía de instalación para computadoras utilizando programas externos. Para la instalación de este programa se requiere de una máquina que contenga como software de Windows 10 u 11, posea Python 3.9 o superior, tenga acceso a internet y el usuario cuente con permisos de administrador.

CAPÍTULO 6

Marco teórico

6.1. Interfaz gráfica de usuario

Según Lauesen, la interfaz gráfica de usuario (GUI por sus siglas en inglés) es aquella parte del sistema que podemos ver, escuchar y sentir. Esta tiene como objetivo volver sistemas interactivos, es decir, aquel sistema el cual existe una interacción humano-computadora y/o viceversa. Generalmente, la GUI en sistemas poco complejos consta de componentes de interacción simples, por ejemplo, en una computadora sería el teclado, el ratón y las bocinas. Por otro lado, en sistemas más complejos se pueden tener componentes más avanzados como micrófonos, sensores, cámaras, entre otros [9].

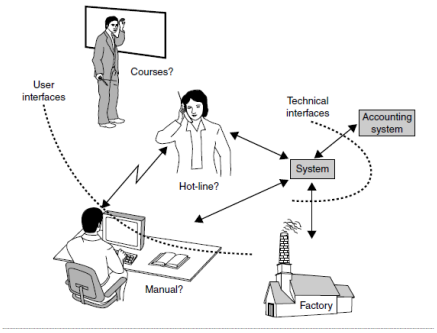


Figura 1: Sistema de interfaces [9]

6.2. Kivy

Kivy es un software de código abierto dirigido por desarrolladores profesionales. Es considerada un proyecto comunitario ya que cualquiera puede realizar un aporte al código y

los desarrolladores que trabajan con la compañía se encargan de aprobarlo. Kivy es una biblioteca de desarrollo GUI multiplataforma para python el cual se puede ejecutar en: iOS, Android, Windows y GNU/Linux. El objetivo de este proyecto es ayudar a desarrollar aplicaciones que utilizan una interfaz de usuario multitáctil de manera innovadora y fácil. La idea detras de esto es para que el desarrollador solamente cree un único código y este se pueda ejecutar en todas las plataformas anteriormente mencionadas, así lograr mayor versatilidad con menor esfuerzo. Cuenta con guías y tutoriales al igual que códigos de ejemplo totalmente gratis para ayudar a desarrolladores. [10].

Figura 2: Logo de Kivy [10]

6.2.1. ScreenManager

ScreenManager o administrador de pantallas, se utiliza en Kivy para poder manejar múltiples pantallas en la aplicación. Este módulo permite mostrar una pantalla a la vez, pero se puede hacer transición a la pantalla que se desee en cualquier momento con la función *manager.current*. Se cuentan con varios comandos para ejecutar funciones en momentos específicos como *on_pre_enter* ó *on_enter* la cual se ejecuta antes de entrar a la pantalla. O en caso contrario se tiene *on_pre_leave* ó *on_leave* la cual se ejecuta antes de salir de la pantalla. Al utilizar *ScreenManager* se debe definir un identificador para cada pantalla a utilizar con un nombre. Dicho identificador se utilizará si se desea hacer un cambio de pantalla o acceder a las funciones de esta. Cada pantalla cuenta con su grupo de funciones, *widgets* e identificadores los cuales se pueden definir dentro del objeto *Screen*. [10]

6.2.2. Properties

Kivy cuenta con un soporte para la creación de clases. Utilizado el módulo *Properties* se pueden conseguir distintas declaraciones de variables al momento de instanciar una clase. Un ejemplo de esto sería *NumericProperty* la cual coloca un observador en la variable para que este sea únicamente un número. Este módulo facilita la utilización de propiedades en los archivos kv de Kivy al igual que acceder a sus valores o modificarlos. [10]

6.2.3. Popup

El módulo *Popup* de Kivy permite la creación de ventanas emergentes modales. Por defecto, la ventana emergente cubrirá toda la pantalla padre al momento de abrirse, pero se puede modificar tanto su tamaño como su posición. Esta ventana puede tratarse como una pantalla cualquiera, pero esta no conserva valores ingresados o modificados una vez se cierre. Al momento de abrirse, por defecto activa una máscara negra transparente al resto de la pantalla en caso su tamaño sea menor al de la pantalla padre. El módulo incorpora funciones como *.open()* y *.dismiss()* las cuales permiten abrir y cerrar la ventana emergente de forma inmediata. [10]

Figura 3: Ventana emergente de Kivy [10]

6.2.4. ScrollView

El módulo de *ScrollView* permite crear una pantalla deslizante del tamaño que se desee. Esta pantalla puede deslizarse de horizontalmente, verticalmente o ambas. Permite deslizarse con la rueda del ratón o con los dedos si el dispositivo es táctil. Este módulo permite configuraciones como la dirección para deslizarse, la distancia mínima para deslizarse por paso, el periodo máximo para deslizarse entre otros. Cabe recalcar que este módulo únicamente permite que exista un objeto en él y por ello se debe utilizar algún tipo de *Layout* para poder añadir más de un elemento. [10]

6.2.5. Layouts

Los *Layouts* son contenedores que permiten arreglar y ordenar los elementos y *widgets* que se añadan a este. Permite, de forma automática, añadirlos y ordenarlos de cierta manera según el contenedor que se escoja. Algunos tipos de *Layouts* son: [10]

BoxLayouts

Permite la colocación de elementos de manera vertical u horizontal uno a la par del otro como se observa en la Figura 4. Esta disposición es la más sencilla de utilizar ya que los elementos distribuyen, por defecto, el máximo tamaño posible del contenedor padre de forma equitativa. [10]

Figura 4: Kivy Box Layout [10]

FloatLayouts

Permite colocar un elemento en cualquier parte de la pantalla. Para el uso de esta disposición se deben utilizar parámetros de posición dada por la función *pos* ó *pos_ hint* los cuales deben tener valores de los ejes "x" y "y". Para la primera función deben colocarse los valores en píxeles, esta función permite colocar el elemento en su posición exacta dentro de la pantalla. Por otro lado, la segunda función permite colocar el elemento en una posición relativa al tamaño del contenedor padre por lo que los valores deben de ir como porcentaje. [10]

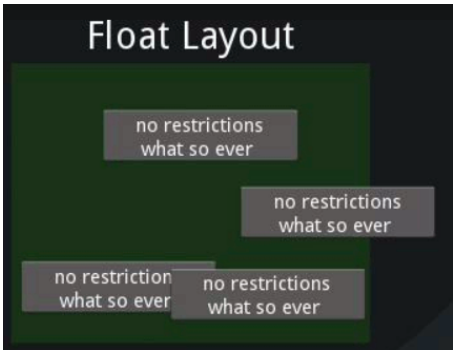


Figura 5: Kivy Float Layout [10]

GridLayouts

Permite ordenar los elementos de una manera más estructurada. Para esta disposición se deben definir las cantidades de columnas y filas que se deseen con los parámetros *cols* y *rows*. La distancia entre columnas y filas puede ser modificable por el usuario, pero por

defecto toman los valores para ocupar el tamaño máximo posible dado por el contenedor padre. [10]

Figura 6: Kivy Gird Layout [10]

AnchorLayouts

Esta disposición permite colocar el elemento en 9 posiciones diferentes. Como se puede observar en la Figura 7, se divide el contender padre en una cuadrícula de 3 por 3. Luego por medio de los parámetros *anchor_ x* y *anchor_ y* se define en cuál de esos recuadros estará el elemento. [10]

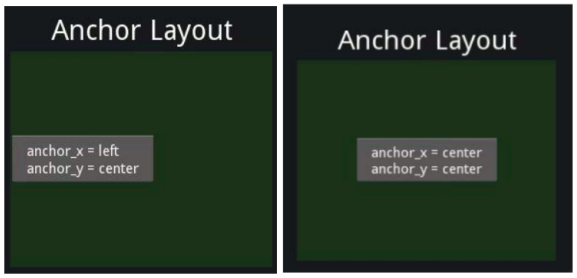


Figura 7: Kivy Anchor Layout [10]

6.2.6. Widgets

Los *widgets* son elementos de una interfaz gráfica que forman parte de la experiencia del usuario, es decir, elementos que permiten una interacción con el usuario ya sea de forma directa o indirecta. Kivi cuenta con distintos tipos de *widgets* que permiten una buena interacción con el usuario y se pueden utilizar en las aplicaciones que se creen cone sta librería.[10]

Image

Este elemento permite colocar una imagen en la aplicación. Este elemento puede configurarse distintos parámetros como el tamaño, el ángulo y la ruta de donde proviene la imagen. Esta última puede cambiarse en cualquier momento permitiendo así actualizar automáticamente la textura y obtener diferentes imágenes en la misma posición según ciertos criterios o eventos que sucedan en la aplicación. [10]

Label

Este permite renderizar texto en la aplicación. Este elemento permite modificar parámetros como el tipo, tamaño y color de letra así como aplicar negrilla, subrayado o cursiva. Se pueden utilizar las fuentes de familia o descargar o diseñar una propia. [10]

Figura 8: Etiqueta de Kivy [10]

Button

El botón es un módulo de etiqueta, pero tiene asociada una acción al momento de presionarse o dejar de ser presionado. Por lo tanto, si se inserta un botón y no se le asocia alguna acción, simplemente será un elemento de texto. Para poder asociarle una acción se le debe añadir el parámetro *on_press*, *on_release* o ambos. A estos parámetros se les debe colocar una función a ejecutar o instrucciones directas. [10]

Figura 9: Botón de Kivy [10]

Spinner

Ese elemento permite una rápida selección entre un conjunto de valores. Por defecto, el *spinner* muestra el último valor seleccionado ó el valor inicial de este. Al momento presionar el elemento despliega un menú con todas las opciones vinculadas a este. Si se presiona alguna opción este oculta las opciones y despliega la opción seleccionada. El elemento permite modificar tanto el estilo del valor desplegado como las opciones vinculadas a este. [10]

Figura 10: Kivy Spinner [10]

Slider

Este elemento permite la creación de un control deslizante que se puede configurar de manera horizontal o vertical. Este elemento permite configurar su tamaño, valor mínimo y máximo, valor inicial, su color y la textura del control. Existen una función asociadas a este objeto llamada *on_value* la cual se ejecuta al momento de cambiar el valor del elemento por medio del control deslizante. [10]

Figura 11: Control deslizante de Kivy [10]

TextInput

Permite la creación de una entrada de texto del usuario. Este elemento crea una caja con texto editable como se puede observar en la Figura 12. Del texto, se pueden configurar que es aceptable como entrada, es decir, se puede restringir la entrada a solo letras, solo números y caracteres especiales. Visualmente se puede configurar color de fondo de la caja, color del borde, el color y tamaño de texto, entre otras. Otro aspecto configurable son los comandos para manipular texto, por ejemplo, el copiar y pegar texto, deshacer una acción, seleccionar texto, entre otros. [10]

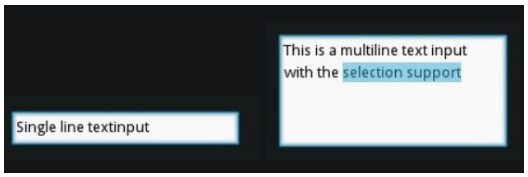


Figura 12: Entrada de texto de Kivy [10]

6.3. PEP 8

En 2001, Guido van Rossum creó un ensayo en donde mencionaba recomendaciones para la escritura del código en Python, en donde más adelante Barry Warsaw y Nick Coghlan agregarían ciertas reglas creando así PEP 8. Esta es una guía de convenciones estilísticas para escribir códigos en Python. Se trata de un conjunto de recomendaciones que, bajo criterio de los autores, hace más legibles y entendibles los códigos. Cuenta con recomendaciones desde el número máximo de caracteres por línea, hasta convenciones para el nombramiento de funciones y variables. PEP 8 se encuentra vigente actualmente como una estandarización para códigos escritos en Python y su última actualización fue realizada en 2013. [11]

Reglamentos o estandarizaciones en códigos permiten una lectura y entendimiento del programa de manera más sencilla y eficiente. Así mismo, estos permiten una mejor compresión entre equipos de trabajo cuando se realizan proyectos de más de una persona. El proyecto del rostro animatrónico de la Universidad del Valle de Guatemala permite una continua evolución, mejoras o creaciones de nuevas características por lo que es muy importante que estudiantes futuros sean capaces de comprender y continuar el código de manera eficiente.

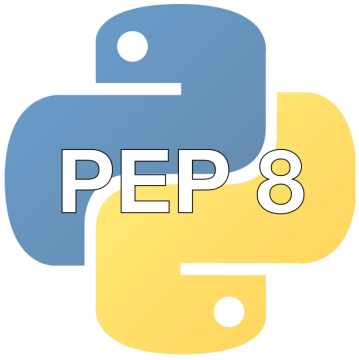


Figura 13: Logo PEP 8 para Python creado por la comunidad [12]

CAPÍTULO 7

Validación de sistemas existentes

7.1. Programa Reconocimiento de emociones

El primer sistema con el que se contaba para el proyecto del rostro animatrónico de la Universidad del Valle de Guatemala era la visión por computadora. Esta es especializada en detectar rostros con el objetivo de diferenciar 7 emociones diferentes. Surgió la necesidad de encontrar las condiciones ideales para un mejor funcionamiento del sistema, tomando en cuenta parámetros de iluminación y distancia lineal de la cámara al rostro detectado.

7.1.1. Prueba realizada

Los materiales de la prueba, sin contar con la interfaz gráfica, son los tres: una cámara Logitech C270 HD [15], un bombillo LED de 12 Watts y una hoja de 60 gramos blanca. Para la preparación de la prueba se debe colocar la cámara en un lugar fijo para poder variar la distancia moviendo al usuario a detectar y cambiar así entre los escenarios. Para poder variar la iluminación, en el escenario con el foco directo al rostro se debe colocar como se observa en la Figura 14 para que la hoja blanca actúe como un difusor y no sobre exponer el rostro. Esta lámpara se debe colocar a no más de 10 centímetros lineales del rostro.

Figura 14: Bombillo de 12 watts con difusor para escenario con diferente iluminación

Figura 15: Cámara Logitech C270 HD

La prueba se realizó en la primera pantalla de la interfaz con el siguiente procedimiento: Como primer paso, se debe presionar el botón de la Figura 16 para iniciar la prueba. Este botón activa un contador descendente de 3 segundos. Transcurrido este tiempo la interfaz desplegará una foto tomada al igual que un emoticono representativo para cada emoción como se puede observar en la Figura 17.

Figura 16: Botón para detectar una emoción

Figura 17: Contador descendente de 3 segundos

Como segundo paso se debe evaluar lo que el sistema esta detectando en tiempo real. Como se observa en la Figura 18 el sistema coloca un cuadrado magenta alrededor del rostro detectado con una etiqueta blanca con el nombre de la emoción detectada. Para la prueba se debe observar si el sistema detecta alguna otra emoción a parte de la deseada durante el funcionamiento del contador.

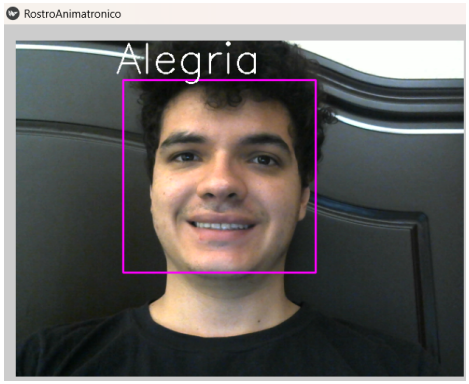


Figura 18: Detección de emoción en tiempo real

Como último paso se debe observar la emoción detectada transcurridos los 3 segundos con el emoticono que el sistema despliegue como se puede observar en la Figura 19. Únicamente existe un emoticono para cada emoción los cuales pueden observarse en la Figura 20

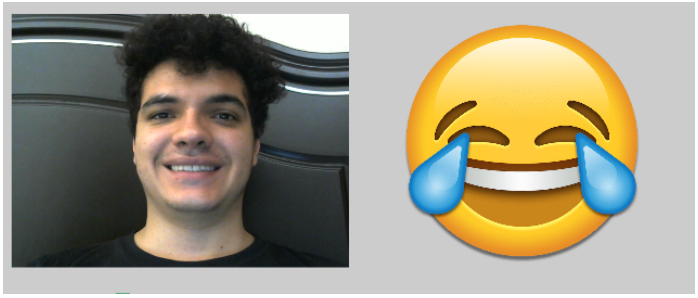


Figura 19: Detección de emoción luego de 3 segundos con emoticono obtenido de [13]

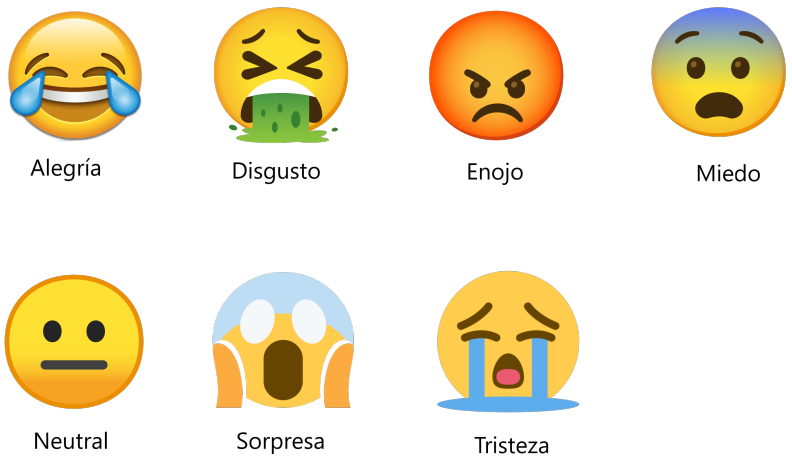


Figura 20: Emoticono para cada emoción detectable obtenido de: [13][14][15][16]

7.1.2. Resultados de pruebas realizadas

Se realizaron 10 pruebas para cada una de las 7 emociones en 6 escenarios distintos. Los escenarios eran los siguientes: 0.5 metros desde la cámara con luz ambiental, 0.5 metros desde la cámara con luz directo al rostro, 1 metro desde la cámara con luz ambiental, 1 metro desde la cámara con luz directo al rostro, 1.5 metros desde la cámara con luz ambiental y 1.5 metros desde la cámara con luz directo al rostro.

El resultado de cada prueba puede obtener tres valores distintos. Se colocó el valor de 0 en el caso que el rostro, luego de los 3, segundos no detectara la emoción deseada como en la Figura 19. Un valor de 0.5 en el caso que satisfactoriamente detectara la emoción deseada, pero en la detección en tiempo real de la Figura 18, la etiqueta mostrara otra emoción que no sea la deseada. El último posible valor es 1 en caso si se detecte la emoción deseada luego de los tres segundos y durante ese proceso, en tiempo real, únicamente detecte la emoción deseada.

Con luz ambiental			
Emoción / Distancia	0.5 m	1 m	1.5 m
Enojo	73 %	76 %	75 %
Disgusto	56 %	74 %	73 %
Miedo	85 %	84 %	76 %
Alegría	85 %	96 %	92 %
Neutral	43 %	56 %	49 %
Triste	27 %	50 %	38 %
Sorpresa	50 %	73 %	60 %

Cuadro 1: Cuadro de detección de imágenes con luz ambiental. El cuadro representa el porcentaje de detección para cada una de las 7 emociones con tres distancias de la cámara hacia el rostro únicamente utilizando luz ambiental

Luz con difusor apuntando al rostro			
Emoción / Distancia	0.5 m	1 m	1.5 m
Enojo	77 %	29 %	32 %
Disgusto	65 %	64 %	63 %
Miedo	44 %	67 %	27 %
Alegría	88 %	89 %	91 %
Neutral	41 %	47 %	22 %
Triste	36 %	64 %	73 %
Sorpresa	83 %	90 %	52 %

Cuadro 2: Cuadro de detección de imágenes utilizando luz con difusor apuntando al rostro. El cuadro representa el porcentaje de detección para cada una de las 7 emociones con tres distancias de la cámara hacia el rostro utilizando una bombillo LED de 12 vatios con un difusor apuntando directo al rostro

De la información obtenida de los Cuadros 1 y 2 a pesar de tener resultados muy variados en cada escenario se pueden observar ciertas tendencias. Una muy clara es que, sin importar el escenario, la emoción de alegría tiene un alto porcentaje de detección siendo el peor un 85 %. Por el lado contrario, la emoción neutral tiende a tener porcentajes muy bajos siendo el mejor un 56 %. Estas tendencias pueden deberse a la base de datos utilizada ya que FER-2013 no posee la misma cantidad de imágenes para cada emoción. Debido a la variabilidad de la mejora o empeora del porcentaje de detección para cada emoción se decidió calcular el promedio del sistema en general para encontrar el escenario ideal de este sistema.

Detección del sistema completo			
Iluminación / Distancia	0.5 m	1 m	1.5 m
Sin luz en el rostro	59.83 %	72.71 %	66.14 %
Con luz en el rostro	62 %	64.29 %	51.43 %

Cuadro 3: Cuadro de detección del sistema. El cuadro representa el porcentaje de detección para cada escenario

Como se puede observar en el Cuadro 3 al realizar un promedio de los porcentajes de las 7 emociones se tiene más claro el rendimiento del sistema en cada escenario. Se observa que el caso ideal es posicionar el rostro a un metro de la cámara sin tener ninguna luz en el rostro obteniendo un rendimiento del 72.71 %. Por el otro extremo se tiene el peor escenario posible el cual es a 1.5 metros utilizando la luz con difusor directamente al rostro con un rendimiento del 51.43

7.2. Programa *chatbot*

El segundo sistema con el que se contaba para el proyecto del rostro animatrónico de la Universidad del Valle de Guatemala era el reconocimiento de voz junto con un *chatbot* para lograr una interacción con el usuario. Se cuenta con métodos como voz-a-texto, el cual con técnicas de procesamiento de lenguaje natural y una red neuronal entrenada, detecta 24 temas distintos. Cada tema tenía vinculadas diferentes respuestas en donde, por medio

de métodos texto-a-voz, se lograba responder sobre el tema detectado. El objetivo de esta prueba era medir el impacto que tiene el ruido en la detección de temas en el *chatbot*.

7.2.1. Pruebas realizadas

Los materiales de la prueba, sin contar con la interfaz gráfica, fueron dos: audífonos Jabra EVOLVE 20 [21] y un celular Motorola G20 con Android 11. Para la disposición de la prueba el celular y el micrófono de los audífonos deben posicionarse lo más cercanos posibles para que la medición de los decibeles sea lo más parecido posible a lo que el micrófono experimente. Se debe colocar una bocina lo más cercano posible a estos dos objetos e ir variando el volumen hasta que la aplicación del celular mida los decibeles del escenario deseado.

Figura 21: Audífonos con micrófono Jabra EVOLVE 20

Primeramente, se asigno un número a cada tema o categoría existente en la base de datos del *chatbot* para facilitar la selección. Se puede observar el número asignado en el Cuadro 4. Para cada prueba se generaron cincuenta números aleatorios entre el 1 y el 24 representando alguno de los temas o categorías de la base de datos. Se realizó esto para 3 escenarios con distintos niveles de ruido medidos con la aplicación Herramientas inteligentes descargable de la tienda de aplicaciones de Android (*PlayStore*) [17]. Esta aplicación cuenta con distintas herramientas de medición habilitadas dependiendo los sensores con los que el celular cuente como se observa en la Figura 23. En este caso se utilizó la herramienta de sonido la cual es capaz de medirlos decibeles que el micrófono detecta como se puede observar en la Figura 22.

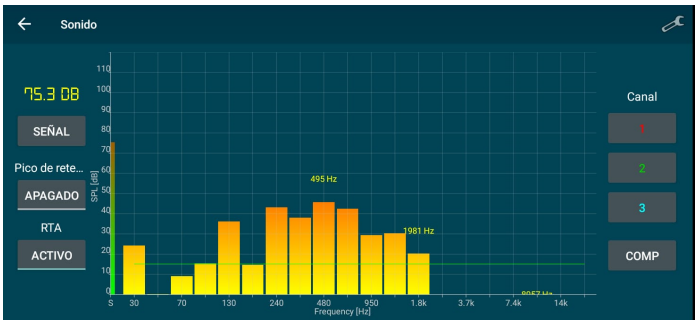


Figura 22: Herramienta de medición de sonido de la aplicación Herramientas inteligentes [17]

Figura 23: Herramientas de medición de la aplicación Herramientas inteligentes [17]

Número asignado	Tema / categoría
1	Saludo
2	Despedida
3	Edad
4	Creadores
5	Definición mecatrónica
6	Materiales
7	Mejor carrera en UVG
8	Comida Favorita
9	Mascarilla
10	Color favorito
11	Flor favorita
12	Definición electrónica
13	Diferencia entre electrónica y mecatrónica
14	Cantidad de carreras en la UVG
15	Donde vive
16	Carrera favorita
17	Becas en la UVG
18	Donde se puede comer en la UVG
19	Cantidad de charlas delvas
20	Bebida favorita
21	Datos sobre Guatemala
22	División de carreras en la UVG
23	Deporte favorito
24	Animal favorito

Cuadro 4: Cuadro del número asignado a cada tema. El cuadro indica el número asignado a cada tema de la base de datos para la realización de la prueba

Como segundo paso, se tomó un número de los generados, se buscaba en la lista a que categoría pertenece y se pensó en una frase correspondiente al tema. Luego se debía mantener presionado el botón de la Figura 24 que se encontraba en la primera pantalla de la interfaz gráfica de usuario mientras se decía la frase seleccionada. Se debe decir la frase una vez este botón se vuelva rojo y aparezca una imagen de una oreja como se observa en la Figura 25. Si se dice la frase antes de esto es probable que no se capte toda la frase por lo tanto falle la detección de la categoría.

Figura 24: Botón para hablar con el chatbot

Figura 25: Estado del chatbot en donde se presenta una buena detección de voz con icono modificado de [18]

Por último, al momento de terminar la frase se soltó el botón y se esperó a que el sistema genere la respuesta como se observa en la Figura 26 y el *chatbot* reproduzca el texto con el método de texto-a-voz. Una vez esto suceda se puede evaluar la prueba con un valor de 0 en el caso la respuesta seleccionada no corresponde a la categoría deseada o 1 en el caso que sí.

Figura 26: Detección de voz-a-texto y respuesta seleccionada por chatbot sobre el tema con icono de celular obtenido de [19]

7.2.2. Resultados de pruebas realizadas

Como se puede observar en el cuadro 5 el sistema es muy sensible al nivel del ruido. Se puede observar que con 50 Db de ruido, el sistema posee un 94 % de precisión lo cual es muy bueno, pero al llegar a 89 Db únicamente presenta un 71 %. Esta caída del 23 % es significativa y puede entorpecer una interacción fluida entre el usuario y el rostro animatrónico. Tomando en cuenta que el proyecto del rostro animatrónico se desea utilizar en espacios lleno de estudiantes, se debe tener en consideración este alto impacto del ruido en el sistema.

Db	puntaje obtenido
50 Db	94 %
75 Db	80 %
89 Db	71 %

Cuadro 5: Cuadro de detección del sistema. El cuadro indica el porcentaje de detección del sistema bajo distintos niveles de ruido

CAPÍTULO 8

Desarrollo de la interfaz gráfica de usuario

8.1. Primera pantalla de la interfaz gráfica de usuario

8.1.1. Integración de sistemas existentes

Al momento de la realización de este trabajo el proyecto del rostro animatrónico contaba con dos programas existentes: sistema de reconocimiento de emociones y un *chatbot*. Ambos programas poseían su propia interfaz gráfica totalmente independiente una a la otra. Todos los programas fueron trabajados con el lenguaje de python y sus respectivas interfaces con la librería llamada Kivy. Por ello, la interfaz gráfica de usuario creada en este trabajo fue utilizando la librería Kivy 2.2.1. La pantalla en donde se integraron estos dos sistemas es la primera.

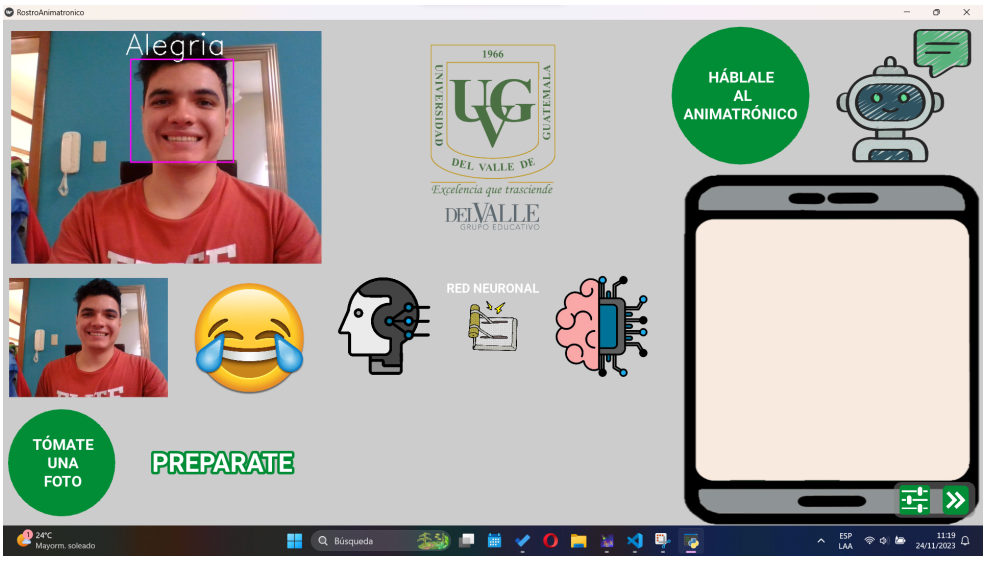


Figura 27: Pantalla donde se integraron los sistemas existentes

Sistema de reconocimiento de emociones

El trabajo realizado por Lorenzana contaba con una interfaz gráfica con 2 pantallas. La primera era el modo imitación el cual contaba con una cámara en tiempo real que colocaba un cuadrado al rededor de la cara con una etiqueta con el nombre de la emoción detectada. La segunda contaba, de nuevo, con la cámara en tiempo real del sistema, un botón y un cuadro de texto. Al presionar este botón capturaba la emoción que el sistema detectaba en ese instante y colocaba en el cuadro de texto un mensaje relacionada con esta emoción. El investigador creó una base de datos con respuestas predeterminadas para cada emoción por lo que al presionar dicho botón se seleccionaba una de estas respuestas y se desplegaba. [5]

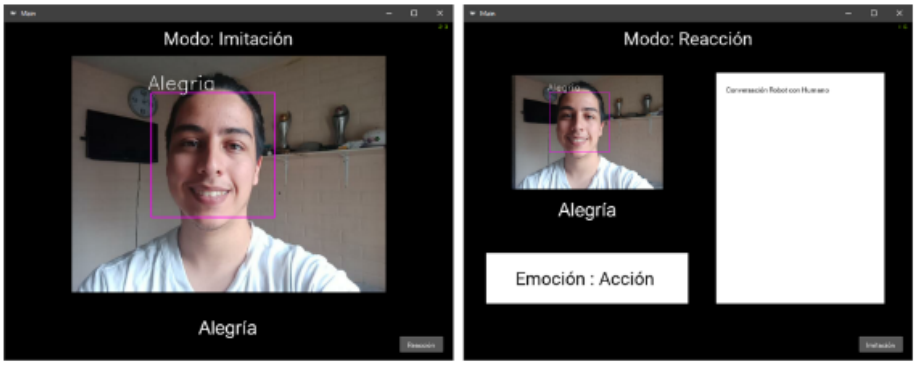


Figura 28: Interfaz gráfica antigua del sistema de detección de emociones [5]

Para la nueva interfaz gráfica en la pantalla 1 se integró el sistema de detección de emociones del lado izquierdo. Se utilizó únicamente la pantalla de reacción de la interfaz anterior por lo que se cuenta con la cámara en tiempo real y el botón de detección de emociones. Se agregó un temporizador de 3 segundos para la detección instantánea de la

emoción la cual luego del tiempo transcurrido toma una foto y la despliega junto con un emoticono correspondiente a la emoción como se puede observar en la Figura 19. Por otro lado, se eliminó el cuadro de texto que se tenía para el despliegue de la respuesta en la detección de la emoción. Esta se reemplazó por el método texto-a-voz del sistema del *chatbot* para una interacción más realista.

Para lograr este diseño se programó la pantalla 1 en su propio archivo kv con un *GridLayout* de 3 columnas. En la primera columna se integró este sistema con un *BoxLayout* en orientación vertical. Se colocó un *widget* de imagen en donde con el módulo de *Texture* de Kivy se actualiza la textura colocando lo que la cámara inicializada por opencv detecta. Esta textura se añade luego de que se coloque el recuadro detectando el rostro al igual que la etiqueta de la emoción detectada.

Como segundo elemento, se colocó un nuevo *BoxLayout* en orientación horizontal el cual contiene dos imágenes. La textura de estas imágenes se actualizan luego que el temporizador de 3 segundos termine. En la primera imagen se coloca, así como en la imagen de tiempo real, lo captado por la cámara, pero se coloca antes de que *opencv* dibuje la detección de rostro. Esto se colocó así para poder simular una fotografía tomada con una cámara o celular. Por otro lado, lo que se actualiza para cambiar la textura del emoticono es la ruta del archivo a utilizar.

Como último elemento se volvió a colocar un *BoxLayout* con orientación vertical. Este contiene un botón y una etiqueta como se observa en la Figura 29. Al momento de presionar el botón, se utilizó el módulo de *Clock* para colocar eventos que se ejecutaran luego de cierto tiempo. Estos eventos es el cambio de texto de la etiqueta en donde cambia a: "3" luego de un segundo, "2" luego de dos segundos, "1" luego de tres segundos, "cheese" luego de 3.5 segundos y "prepárate" luego de 5.5 segundos.

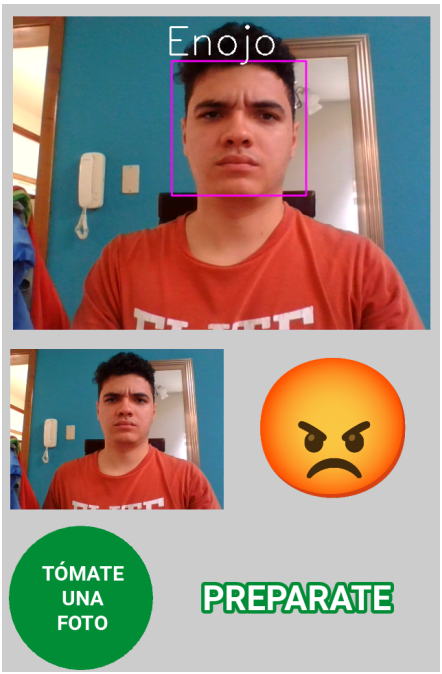


Figura 29: Interfaz gráfica nueva del sistema de detección de emociones

Sistema de chatbot

El segundo sistema con el que se contaba era un programa de reconocimiento de voz capaz de identificar 24 categorías o temas distintos. Este programa poseía un archivo *.json* el cual fungía como base de datos. En dicho archivo se guardan 3 parámetros: tag, es el nombre de la categoría o tema que se desee identificar; patterns, se refiere a frases que contengan los patrones referentes al tema con las que se entrena la red neuronal; responses, se refiere a las respuestas que el *chatbot* dará una vez identificado el tema. [3]

La interfaz gráfica que creó el ingeniero Fuentes consistía en una sola pantalla. Dicha interfaz buscaba parecerse a un aplicación de mensajería por texto en donde se desplegara lo que detectó el reconocimiento de voz así como la contestación del *chatbot*. Esta interfaz mantenía el micrófono abierto en donde intentaba reconocer algún tipo de mensaje luego de cierto tiempo. Si reconocía algo, se deplegaba en la interfaz haciendo alusión a un mensaje enviado por el usuario, el programa detectaba la categoría y elegía una respuesta al azar almacenada en la base de datos. Una vez hecha esta selección, desplegaba el mensaje haciendo alusión a un mensaje recibido por el usuario y al mismo tiempo, con una técnica de texto a voz, decía la respuesta. Cada vez que se detectaba un mensaje nuevo este reemplazaba el mensaje anterior por lo que los recuadros que se observan en la Figura 30 son estáticas.

Figura 30: Interfaz gráfica antigua del sistema de *chatbot*[3]

Para la nueva interfaz gráfica se intentó mantener esta idea de una aplicación de mensajería por texto. Se agregó un botón con el cuál comienza una grabación de audio que se almacena en un archivo *wav*. Este archivo se le carga al programa de voz-a-texto para poder realizar todo el proceso. Se agregó de esta manera ya que con la interfaz anterior no se tenía control ni indicación de cuando estaba comenzando a escuchar, lo que causaba que no

siempre se captara la frase entera que se decía. Por otro lado, se agregó una funcionalidad de desplazamiento vertical al "celular". Cada vez que se presiona el botón para hablar se crea un nuevo mensaje con el texto detectado al igual que la contestación del *chatbot* como se puede observar en la Figura 31.

Para que sea más fácil para el usuario saber si el sistema esta escuchando o no se colocaron 3 estados visuales en la interfaz como se observa en la Figura 32. El primer estado es el neutral, este se encuentra al momento de iniciar la aplicación o cambiar de página. El estado neutral indica que no se a utilizado el sistema del *chatbot* y tiene la imagen de un robot. El segundo estado es escuchando, aquí se pinta el botón de rojo y se cambia la imagen por una oreja. Este estado se mantiene activo mientras el botón este siendo presionado. Es aquí donde el archivo de audio se esta generando. Por último, se tiene el estado hablando en donde el botón vuelve a ser verde y se cambia la imagen por una persona hablando. En este estado se crean los mensajes de texto del usuario y del *chatbot*. Es también aquí donde el método de texto a voz es utilizado y el sistema responde.

Figura 31: Ejemplo de desplazamiento vertical en historial de conversación

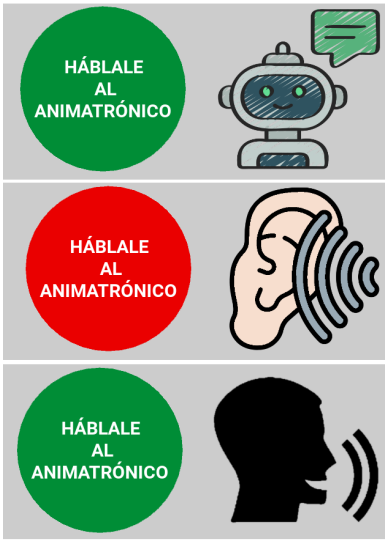


Figura 32: Estados del *chatbot* con iconos obtenidos de: [20] [18] [21]

Para lograr la creación de esta interfaz se utilizó la tercera columna del *GridLayout* de la primera pantalla. Se creó un *BoxLayout* con orientación vertical en donde se agregaron 3 elementos. El primer elemento fue un *BoxLayout* con orientación horizontal. En este se agregó el botón para cambiar los estados previamente mencionados del *chatbot* al igual que una *widget* de imagen representando los estados. Para actualizar este estado se hace el cambio de la ruta de la imagen a utilizar.

Como segundo elemento se agregó un *BoxLayout* con orientación vertical. Se agregó esto ya que se deseaba tener el fondo de una imagen de un celular. Como único elemento de esta distribución es un *ScrollView* ya que permite colocar desplazamientos horizontales o verticales. Como se desea dar la ilusión de una aplicación de mensajería de texto se utilizó el desplazamiento vertical. Cabe recalcar que este *ScrollView* comienza vació y mediante una función propia de Kivy llamada *add_widget* se le agregaban los elementos de texto correspondientes. Como último elemento, se agregó el menú para cambio de pantallas el cuál se detallará más adelante.

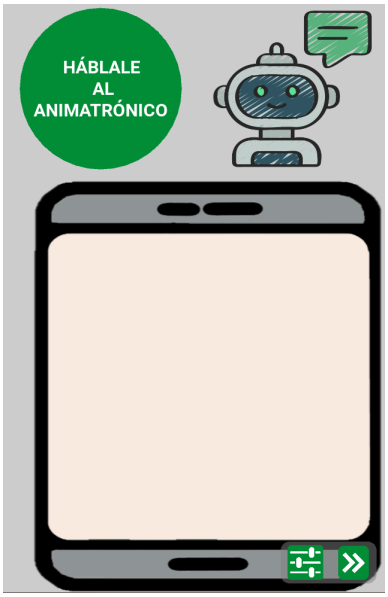


Figura 33: Interfaz gráfica nueva del sistema de *chatbot*

8.1.2. Sistema para cambio de inteligencia

Paralelo a este trabajo realizado, el investigador Mazariegos trabajó en la investigación titulada "Implementación de la Inteligencia Artificial en rostro animatrónico capaz de crear una conversación". Dicho trabajo buscó implementar una alternativa al sistema de *chatbot* que se tiene actualmente. Se implementó un sistema para el cambio de inteligencia entre el *chatbot* y el sistema planteado por Mazariegos como se puede observar en la Figura 34.

Esta interfaz se programó en la primera pantalla en la columna 2. Se utilizó un *BoxLayout* con orientación vertical que contiene dos elementos. El primer elemento es una *widget* de imagen con el logotipo de la Universidad del Valle de Guatemala. El segundo elemento es un *BoxLayout* con orientación horizontal que contiene 4 elementos. Dos de estos elementos son imágenes representativos de cada inteligencia. El primero representa una red neuronal y el otro a la inteligencia artificial. Por último tenemos un botón que tiene una imagen como textura el cual, en programación, cambia la inteligencia que se utiliza. Ambas inteligencias tienen como entrada texto por lo que al hacer el cambio de inteligencia se cambia a que sistema se le envía el texto creado por el método de voz-a-texto.

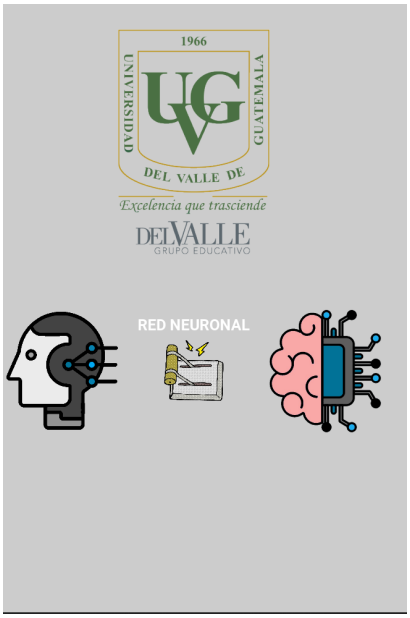


Figura 34: Interfaz gráfica para cambio de inteligencia con iconos obtenidos de: [22] [23]

8.2. Segunda pantalla de la interfaz gráfica de usuario

El proyecto del rostro animatrónico cuenta con 19 servomotores dividiéndose en: 12 motores SG90, 3 dynamixel AX-12A y 4 dynamixel XL-320. Se deseaba desplegar el estado de estos motores en la interfaz gráfica en donde se pudieran modificar manualmente. Para esto se creó la segunda pantalla como se puede ver en la Figura 35. Para la creación de esta pantalla se utilizó un *GridLayout* de 2 columnas. Al entrar a esta pantalla el animatrónico entra en modo manual y despliega el estado de los 19 motores en la segunda columna. Por otro lado, en esta pantalla se implementó un sistema de gestos para que el usuario pueda interactuar con el animatrónico.

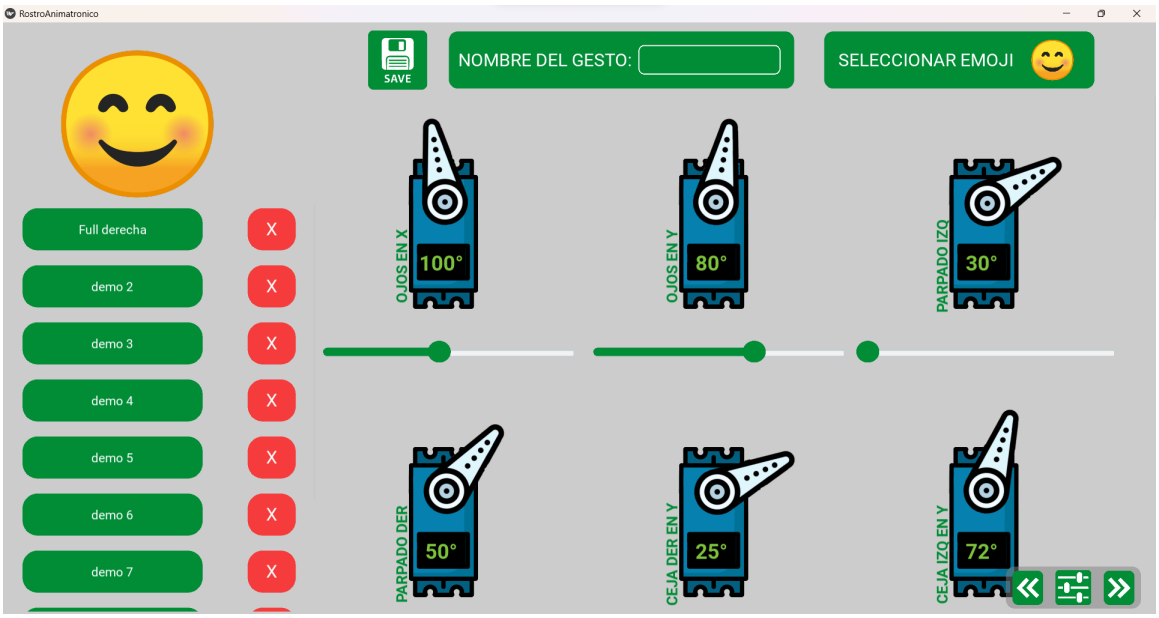


Figura 35: Pantalla 2 de la interfaz gráfica

8.2.1. Despliegue del estado de motores

En la segunda columna de la pantalla, se colocó un *BoxLayout* en donde como segundo elemento se agregó un *ScrollView* para poder desplegar todos los 19 motores como se observa en las figuras 36 y 36. Se colocó un deplazamiento vertical en donde se colocan 3 motores por fila. Para lograr esto se colocó un *BoxLayout* con orientación horizontal en cada fila y se creó una clase de servomotor individual. Esta clase contenía diferentes *widgets* de imágenes para el cuerpo y el brazo móvil del servomotor, un control deslizante para poder modificar el valor en grados del motor y un texto con el nombre del servomotor. Se creó un modelo diferente para cada tipo de servomotor utilizado en el proyecto para poder tener una mejor representación del estado de los motores.

Para la obtención de los datos, Kivy cuenta con una función llamada *on_pre_enter* la cual se ejecuta antes de cargar la pantalla correspondiente. En esta función se le pide el estado de los motores al microcontrolador por medio de comunicación serial. Al momento de recibir los datos se le asigna su valor a cada motor modificando propiedades creadas por el módulo de *Properties* de Kivy la cual permite crear propiedades numéricas o de texto para las clases propias.

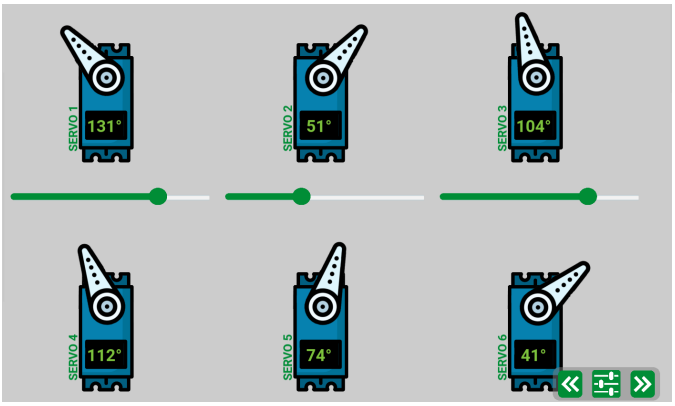


Figura 36: Despliegue de motores SG90 con figuras obtenidas de [24]

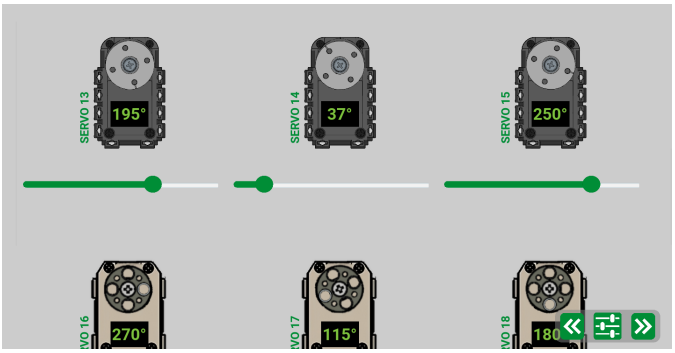


Figura 37: Despliegue de motores dynamixel AX-12A y XL-320 con figuras obtenidas de: [25] [26]

8.2.2. Sistema de modificación individual de motores

Como se mencionó anteriormente, se creó una clase de motor individual para la optimización y mejor comprensión del código. Esta clase puede llevar elementos propios de Kivy por lo que se agregaron 2 imágenes, 2 textos y un control deslizante. Primeramente se colocó un *BoxLayout* con orientación vertical la cual tenía como fondo la primera imagen representando el cuerpo del servomotor. Se guardaron en una carpeta los 3 modelos de cuerpo para cada servomotor y se utilizó la propiedad *StringProperty* de Kivy para poder modificarla al momento de agregar la clase y colocar la ruta correspondiente.

Luego, como primer elemento, se colocó de nuevo una imagen representando el brazo móvil de cada servomotor. Para este también se creo una *StringProperty* que debe contener la ruta del brazo a utilizar. Para esta imagen se creó otra propiedad utilizando *NumericProperty* de Kivy para poder modificar el ángulo de este. Si se comparan las figuras 38 y 39 se puede ver que al momento de modificar el valor del control deslizante el brazo móvil también se moverá de posición. Esto con el fin de tener una mejor representación del movimiento que se esta realizando al servomotor.

Los siguientes elementos colocados fueron de texto. En el primero se colocó el nombre representativo del motor en donde se giró 90° por cuestiones estéticas. El otro texto representa los grados actuales que tiene el motor. Esta etiqueta toma el valor inicial obtenido

del microcontrolador antes de entrar a la pantalla, pero se actualiza conforme se modifica el control deslizante. De nuevo, si se comparan los valores de esta etiqueta en las figuras 38 y 39 se puede observar dicha actualización de datos. Para lograr esta actualización se creó un *StringProperty* el cual toma el valor de los grados del control deslizante y le agrega el caracter "°" para una mejor comprensión del dato que se despliega.

Como último elemento de esta clase se colocó el control deslizante con el módulo *Slider*. A este módulo se le modificó el color a la paleta utilizada en la interfaz gráfica al igual que el mínimo y el máximo. Para estos últimos parámetros se crearon dos *StringProperty* las cuales toman su valor inicial obteniéndolos de la base de datos. Esta base de datos es un archivo *.json* el cuál se modifica en la tercera pantalla que se explicará más adelante. Al presionar este control deslizante se va actualizando automáticamente los grados que se despliegan, la rotación de la imagen del brazo y se envía el valor, por medio de comunicación serial, al microcontrolador.

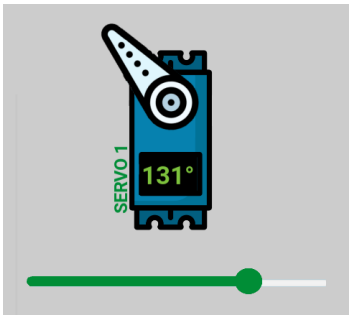


Figura 38: Control individual de los motores

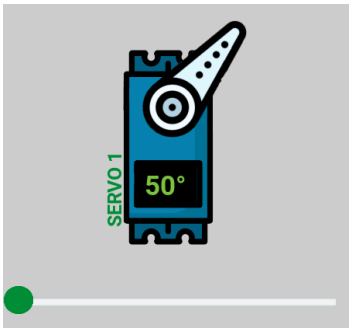


Figura 39: Cambios al momento de mover el control deslizante

8.2.3. Sistema de gestos

Para una mayor interacción con el usuario en la presentación de los estados de los motores se creó el sistema de gestos. Esto consiste en almacenar los valores de los motores con un nombre y un emoticono. Esta actividad busca que el usuario mueva manualmente todos los motores hasta hacer un gesto con el animatrónico. Una vez el usuario este a gusto con el gesto realizado, puede proceder a colocarle un nombre en el lugar que indica la Figura 40. También se puede seleccionar un emoticono que el usuario desee el cuál generara una ventana emergente como se ve en la Figura 41. Una vez seleccionado estos parámetros se

puede presionar el botón de guardar el cual almacenará todo en la base de datos que se encuentra en un archivo *.json*.

Para lograr esto, en la columna 2 de la segunda pantalla se colocó un *BoxLayout* con orientación horizontal en el cual se colocaron 3 elementos. El primero es un botón que tiene como textura una imagen de disquete el cual al ser presionado se activa la función de escritura en el archivo *.json* que actúa como base de datos. Una vez actualizada la base de datos se agrega un *widget* de botón al *ScrollView* que despliega todos los gestos, este se explicará más adelante.

El segundo elemento es un *BoxLayout* con orientación horizontal que contiene una etiqueta de texto y una entrada de texto llamada *TextInput* propia de Kivy. Esta entrada acepta cualquier caracter, tanto números, letras o caracteres especiales. Es aquí donde el usuario debe ingresar el nombre que quiere colocarle al texto. Este nombre es el que aparecerá en el botón desplegado.

Por útlimo, se agregó un *BoxLayout* con orientación horizontal el cual contiene un botón y una imagen. El botón activa una ventana emergente que se logró con el módulo *PopUp* de Kivy. Este módulo permite crear una ventana con tamaño y posición modificables colocando un filtro al resto de la pantalla que no sea la ventana como se observa en la Figura 41. A esta ventana se le colocó dos *BoxLayout* con orientación horizontal, uno para cada fila de emoticonos. A cada uno se le agregaron 6 botones cuyas texturas tienen el emoticono que representan. Al presionar algún boton la ventana emergente se cierra y modifica la imagen que se encuentra en el botón que abre esta ventana emergente.

Figura 40: Creación de gestos y almacenamiento en la base de datos con icono obtenido de [27]

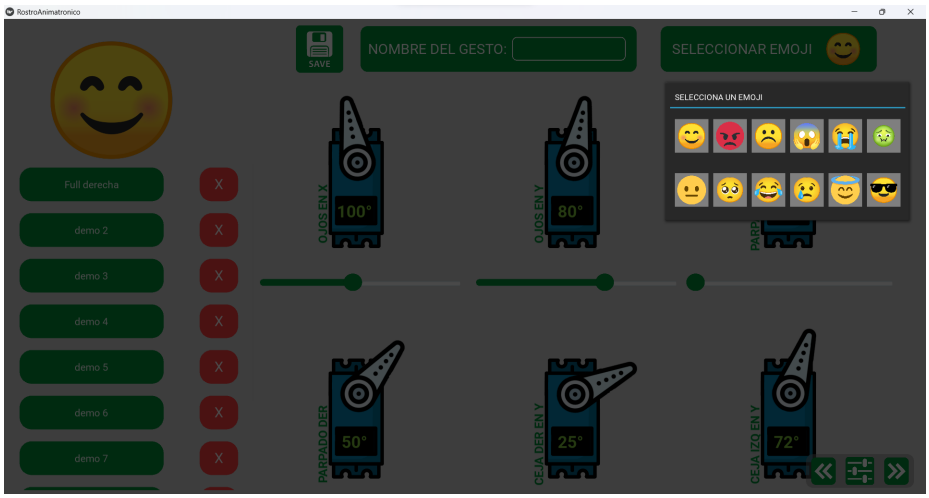


Figura 41: Selección de emoticono con iconos obtenidos de [28]

Por último, tenemos la presentación de los gestos almacenados los cuales se colocaron en

la columna 1 de la pantalla. A esta columna se le agregó un *BoxLayout* con dos elementos. El primero es una imagen la cuál despliega el emoticono guardado para el gesto. Esta imagen se actualiza al presionar cualquier botón de los gestos almacenados actualizando la ruta de la imagen. Como segundo elemento tenemos un *ScrollView* con desplazamiento horizontal para así poder guardar todos los gestos que se deseen sin restricción alguna. Es aquí donde, con la función *on_ pre_ enter* de Kivy podemos cargar todos los gestos de la base de datos y colocarlos como botones. También se agregan automáticamente cuando se presiona el botón de guardar de la Figura 40. Al presionar cualquiera de estos botones se actualizan los valores de todos los motores y el emoticono que se ve hasta arriba de la Figura 42. Esta acción también hace que el sistema envíe los valores de los motores al microcontrolador utilizando el protocolo adecuado para lograr así una sincronización con los motores físicos.

Figura 42: Sistema de gestos

8.3. Ventana de configuración

Se creó una ventana para poder configurar el puerto USB al que estará conectado el microcontrolador y la selección de la cámara a utilizar en el sistema de reconocimiento de emociones. Para poder acceder a esta se debe presionar el botón central del menú que se puede ver en la Figura 45. Una vez presionado aparecerá una ventana emergente como la que se puede observar en la Figura 43 en donde se puede ingresar el puerto serial o la cámara a utilizar. Si se desea salir de esta ventana se debe presionar la x roja que esta en la parte superior derecha.

Para la creación de esta ventana primeramente se utilizó el módulo de *PopUp* de Kivy. A este módulo se le colocó un *BoxLayout* con orientación vertical en donde se agregaron 3 elementos. El primer elemento es un *BoxLayout* con orientación horizontal el cual tiene 3 elementos: una imagen con el logo de la Universidad de Valle de Guatemala como textura,

una etiqueta con el título de la ventana y un botón con una x la cual se programó para cerrar la ventana. El siguiente elemento es otro *BoxLayout* con 3 elementos: el primero es una etiqueta indicando que se puede agregar el puerto serial a utilizar, el siguiente es un elemento de *TextInput* de Kivy configurado para únicamente aceptar números y por último se tiene un botón. Este al ser presionado se envía una señal y se espera la contestación en donde de ser así aparece un mensaje de éxito, de lo contrario aparece un mensaje de falla. Por último se tiene un elemento de *Spinner* el cual se colocaron como opciones la cámara por defecto de la computadora o la cámara USB como se puede observar en la Figura 44.

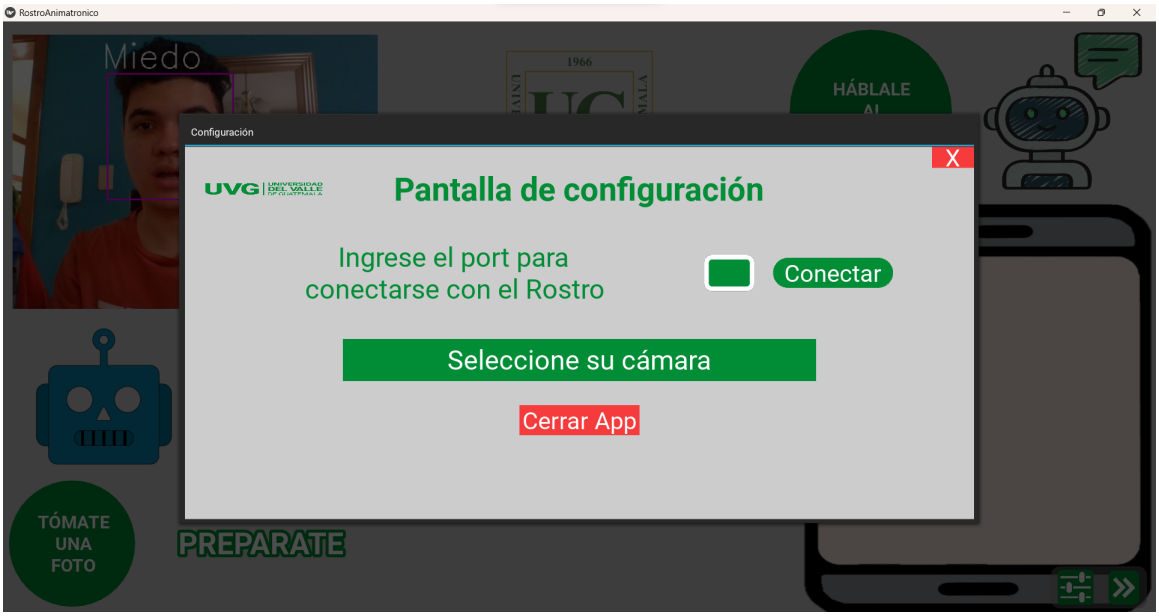


Figura 43: Pantalla de configuración

Figura 44: Selección de cámara a utilizar

8.4. Menú para cambio de pantallas

Para la interfaz gráfica se tienen 3 pantallas distintas por lo que se crearon botones y para la movilización entre estas. Como se puede observar en la Figura 45 son tres botones programados para este menú. Las flechas observables son paras los cambios de pantalla donde únicamente se puede navegar de la siguiente manera: pantalla 1 hacia 2, pantalla 2 hacia 1, pantalla 2 hacia 3, pantalla 3 hacia 2. Es por estas opciones que existen 3 diseños diferentes del menú. Por otro lado, el botón del centro es para abrir la ventana de configuración que se detallará más adelante.

Para la creación de este menú se utilizó un *FloatLayout* para permitir la superposición sobre los elementos de la pantalla con un *BoxLayout* con orientación horizontal. Dentro de este se encuentran 3 *widgets* de botones. Para el cambio de pantallas se utilizó una función propia de Kivy llamada "root.current". Dicha función permite al administrador de pantallas cambiar la pantalla actual.

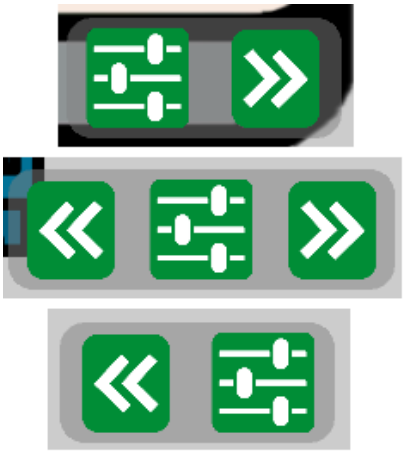


Figura 45: Botones del menú en distintas pantallas con iconos obtenidos de: [29] [30]

CAPÍTULO 9

Desarrollo de la interfaz de control

Como última pantalla se tiene la interfaz de control observable en la Figura 46. Esta interfaz busca poder modificar aspectos más específicos de los motores y bases de datos. En este caso se desea controlar los mínimos y máximos de cada motor, al igual que la base de datos utilizada para el *chatbot*. Para la creación de esta pantalla se utlizó un *GridLayout* con 2 columnas. En la primera columna se encuentra la modificación de mínimos y máximos, y en la segunda la modificación de la base de datos.

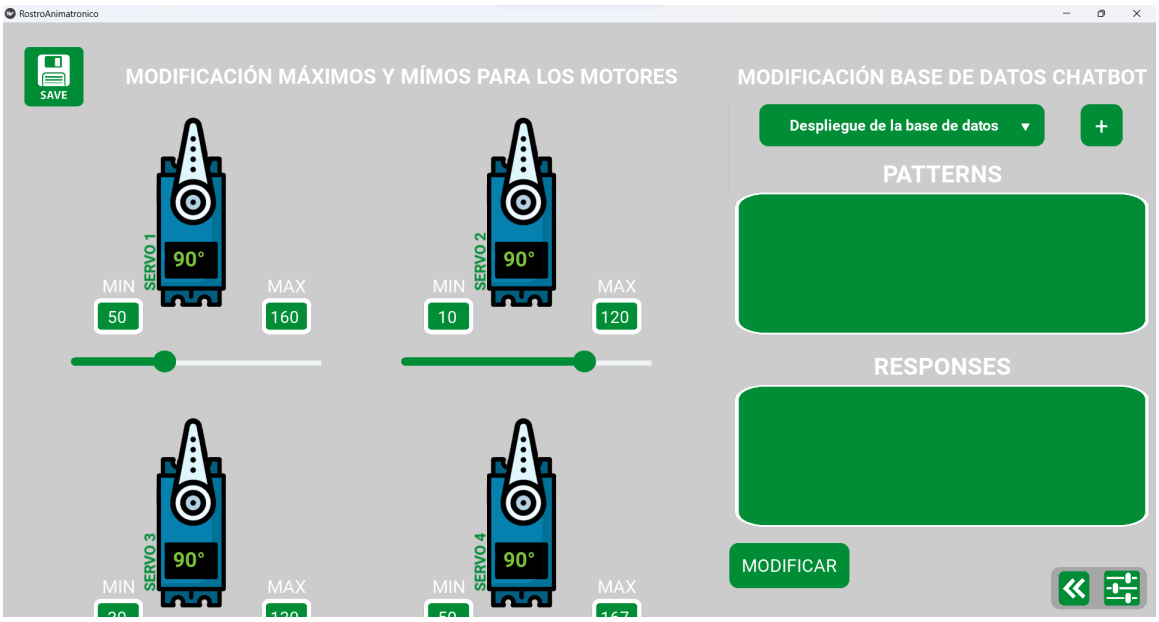


Figura 46: Interfaz de control dentro de la interfaz gráfica de usuario

9.1. Modificación de mínimos y máximos de los servomotores

Para la modificación de los mínimos y máximos de los servomotores se utilizó una estructura muy similar a la clase creada para el despliegue del estado de los motores. Como se puede observar en la Figura 47 se tiene de nuevo un *BoxLayout* con el fondo del cuerpo del servomotor, una imagen con el brazo móvil del motor que se mueve junto con el control deslizante, una etiqueta con el nombre del servomotor, una etiqueta con los grados que tiene el servomotor y un control deslizante. Adicionalmente a esta estructura que ya conocemos, se agregaron dos elementos de *TextInput* que aceptan únicamente números junto con sus etiquetas indicando si es el mínimo o el máximo. En este caso, al igual que en la segunda pantalla, se tiene un *ScrollView* con desplazamiento vertical para poder acceder a todos los motores, pero en este caso se tiene únicamente dos motores por fila como se observa en la Figura 48.

En esta pantalla se tiene la misma funcionalidad que en el despliegue del estado de motores ya que con el control deslizante también podemos controlar los servomotores físicos, pero el objetivo es diferente. En esta pantalla se busca poder modificar los mínimos y máximos de cada motor para tener la mayor movilidad posible sin comprometer los mecanismos. Actualmente la base de datos cuentan con los valores mínimos y máximos recomendados por el ingeniero Soto. [2]

El proceso que se debe llevar a cabo debe ser tres simples pasos. Primeramente se debe modificar los mínimos y máximos de cada motor a los deseados. Luego se debe verificar si estos mínimos y máximos son los correctos o deseados moviendo el control deslizante. Una vez se esta satisfecho con los valores se debe presionar el botón de guardar ubicado en la esquina superior izquierda como se puede observar en la Figura 46. Una vez presionado este botón se guardarán en un archivo *.json* los valores actuales y automáticamente se modificarán los valores en la segunda pantalla.

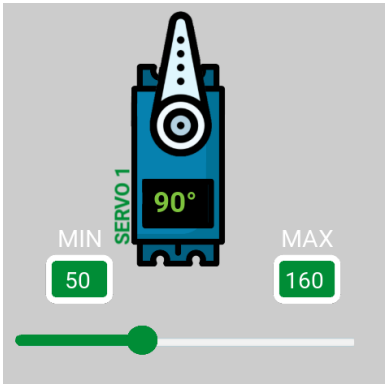


Figura 47: Modificación de mínimos y máximos

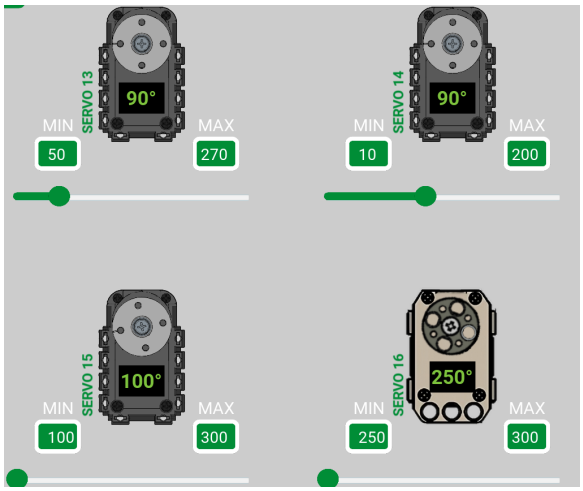


Figura 48: Despliegue mínimos y máximos con otros modelos de motores

9.2. Modificación de la base de datos del *chatbot*

Anteriormente el sistema de *chatbot* no poseía ninguna interfaz gráfica para la modificación de la base de datos y el reentrenamiento del mismo. Si se deseaba hacer alguna modificación tanto a las respuestas de cada categoría o la modificación de los patrones para volver a entrenarlo debía hacerse de forma manual en alguna aplicación para programación. Es por ello que se buscó facilitar este proceso y se agregó en la tercera pantalla. Como se puede observar en la Figura 49 se tiene un selector en donde presionarlo se despliegan todos los títulos de las categorías que están almacenadas como "tags" en la base de datos. Al presionar algún título se colocan los patrones y respuestas en sus lugares correspondientes como se observa en la Figura 50. Una vez se encuentran así se puede hacer la modificación, ya sea cambiar, quitar o agregar un patrón o respuesta. Para poder agregar se debe tomar en cuenta siempre iniciarla en una nueva línea y terminar con una coma (",") y la creación de una nueva línea. Al momento de estar satisfecho con la modificación se debe presionar el botón de "modificar" el cual empezará un entrenamiento de la red neuronal.

Para lograr esto se utilizó primeramente un *BoxLayout* en donde se agregaron 7 elementos. El primero es únicamente una etiqueta con el nombre de la funcionalidad de esta parte de la interfaz. El segundo elemento es otro *BoxLayout* con 2 elementos, el primero un módulo de *Spinner* para la creación del selector y luego un botón que abre la funcionalidad de adición de categorías nuevas a la base de datos que se explicará en detalle más adelante. Como tercer elemento tenemos de nuevo una etiqueta indicando que se desplegarán los patrones. El cuarto elemento es un módulo de *TextInput* el cual toma como valor inicial los patrones contenidos en la base de datos de la categoría seleccionada. El quinto elemento es nuevamente una etiqueta indicando que se desplegarán las respuestas. Como sexto elemento se tiene nuevamente un *TextInput* en donde, al igual que el anterior, toma como valor inicial las respuestas almacenadas en la base de datos de la categoría seleccionada. Por último, se tiene el botón para entrenar la red neuronal la cual, al ser presionada, utiliza una función creada por el ingiero Fuentes para esta tarea y modifica el archivo *.json* que funge como base de datos del sistema.

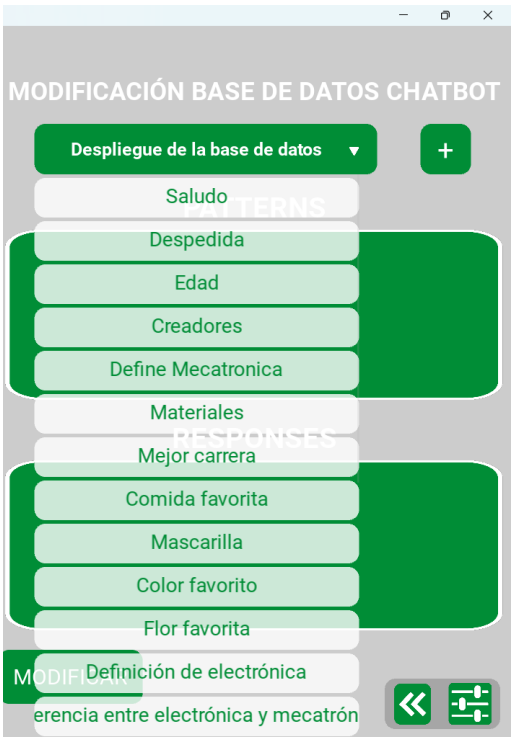


Figura 49: Despliegue de categorías almacenadas en la base de datos

Figura 50: Ejemplo de selección de categoría para modificación

9.3. Adición de categoría a la base de datos del *chatbot*

El ingeniero Fuentes creó una segunda interfaz gráfica para la tarea de la adición de una categoría a la base de datos. Como se puede observar en la Figura 51 se debía colocar el nombre de la categoría, 4 patrones distintos y 2 respuestas distintas. Al momento de presionar el botón "Actualizar" se ejecutaba una función de entrenamiento a al red neuronal y se modificaba el archivo *.json* de la base de datos. El autor menciona que esta interfaz no puede ejecutarse al mismo tiempo que la interfaz gráfica principal.

Figura 51: Interfaz gráfica antigua para la adición de una categoría a la base de datos del *chatbot* [3]

Para la nueva interfaz gráfica se buscó integrar este sistema para una fácil utilización del mismo y se pueda ejecutar junto al código principal como se observa en la Figura 52. La manera en que se integró fue por medio de una ventana emergente la cual se abre al presionar el botón "+" al lado del selector de categorías que se observa en la Figura 50. En esta ventana se tienen los mismos tres parámetros necesarios para almacenarlos en la base de datos: nombre, patrones y respuestas. Se quitó el límite que se tenía de 4 patrones y 2 respuestas para tener mayor versatilidad y personalización. Se utilizó el mismo formato que la modificación de la base de datos en donde para agregar un patrón o respuesta se debe finalizar con una coma y un *enter*. Al momento de presionar el botón añadir ubicado en la parte inferior izquierda, se ejecuta la misma función de entrenamiento que en la modificación y se agrega el tema a la base de datos. Si ya no se desea utilizar la ventana se debe presionar el botón de cancelar.

Para la creación de esta ventana, de nuevo, se utilizó el módulo de *PopUp* de Kivy para la ventana emergente. Se agregó un *BoxLayout* con 7 elementos: 3 elementos de etiqueta para indicar que se debe ingresar, 3 módulos de *TextInput* configurados para aceptar cualquier tipo de caracter y por último un *BoxLayout* con orientación horizontal que contiene dos botones.

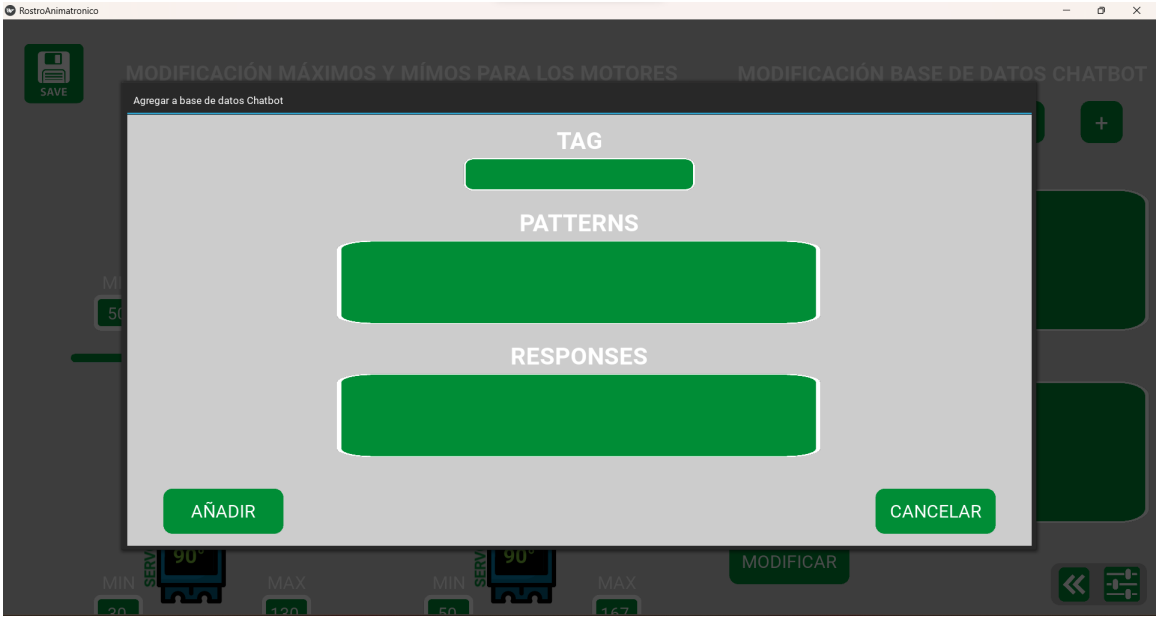


Figura 52: Interfaz nueva para añadir categoría a la base de datos

9.4. Modificación de la base de datos del reconocimiento de emociones

Así como el sistema del *chatbot*, la detección de emociones también tiene respuestas dependiendo de la emoción que se detecte al presionar el botón de la Figura 45. Al detectar una emoción, el sistema escoge una respuesta aleatoria entre las posibles opciones de la base de datos y utiliza el sistema texto a voz para poder mencionarla al usuario. La modificación de esta base de datos se puede acceder al deslizar hacia abajo en donde se encuentra la modificación de la base de datos del *chatbot* como se observa en la Figura 53. Se puede seleccionar cualquiera de las 7 emociones que es capaz de detectar el sistema como se observa en la Figura 54 y se debe modificar de la misma manera anteriormente mencionada en el capítulo 9.2. Para realizar los cambios se debe presionar el botón de modificar y en este caso únicamente sobrescribe el archivo *json* ya que al ser únicamente respuestas no se debe re-entrenar la red neuronal.

Figura 53: Despliegue de las 7 emociones de la base de datos del sistema de detección de emociones

Figura 54: Despliegue de respuestas de la emoción seleccionada

CAPÍTULO 10

Protocolo de comunicación serial

Se diseñó un protocolo de comunicación entre la interfaz gráfica y el microcontrolador encargado de los motores físicos. Dicho protocolo utiliza la comunicación serial vía UART por medio de un traductor de UART a USB para poder conectarlo a la computadora. Para poder inicializar la comunicación se debe indicar el puerto serial a utilizar en la ventana de configuración que se puede observar en la Figura 43.

Nombre del caso	caracter seleccionado
Envió valor de un motor	#
Modo manual	*
Envió respuesta de sistemas	$
Solicitud estado de motores	&
Envió mínimos y máximos de cada motor	%

Cuadro 6: Caracteres especiales que se envían como segundo byte en cada caso utilizando codificación utf-8

Existen cinco casos distintos que se tomaron en cuenta para el protocolo de comunicación en donde cada uno posee una estructura específica interna. Esto se refiere que para todos los casos se utilizó una estructura más general en donde existen dos caracteres de inicio y fin siendo estos "<" y ">" respectivamente utilizando una codificación utf-8. Luego del byte de inicio, se seleccionaron un caracter especial para cada caso como se puede observar en el Cuadro 6. Estos caracteres sirven para que el microcontrolador sea capaz de tomar decisiones dependiendo de cada caso y así modificar los valores de un motor o varios, enviar o recibir datos y conocer si debe estar en modo manual o automático. Para la decodificación del protocolo se realizó un código en Arduino el cual contiene una respuesta distinta para cada caso que se tomó en cuenta en el protocolo.

10.1. Envío del estado de un motor con el control deslizante

Cuando se entra a la segunda pantalla y el microcontrolador ya se encuentra en modo manual se utiliza este protocolo. Se envía mientras el control deslizante de alguno de los servomotores de la segunda pantalla se esté presionando como el de la Figura 38 o cuando se presione el botón de algunos de los gestos como se observa en la Figura 42. El protocolo se puede observar en el Cuadro 7 en donde se observa los caracteres de inicio en la primera posición, el caracter especial seleccionado del Cuadro 6 en la segunda posición y el caracter de finalización en la última posición.

El protocolo específico utilizado es el enviar la letra "M" luego del identificador del motor que se puede observar en el Cuadro 8. En el ejemplo del Cuadro 7 se observa que se envía M19, esto quiere decir que el valor que se esta enviando pertenece al motor de la rotación del cuello. Estos identificadores se utilizan también para que el microcontrolador modifique el valor del motor deseado. Luego de este identificador se utiliza el caracter ":" para poder separar el valor del identificador del valor del estado del motor, en este caso, el valor es de 180°.

Byte	0	1	2	3	4	5	6	7	8	9
Caracter	<	#	M	1	9	:	1	8	0	>

Cuadro 7: Cuadro del protocolo para envío del estado de un motor. El cuadro indica el orden de los caracteres que se deben enviar por comunicación serial al microcontrolador

Tipo de servomotor	Servo motor	Identificador
SG90	Ojos en x	1
SG90	Ojos en Y	2
SG90	Parpado izquierdo	3
SG90	Parpado derecho	4
SG90	Ceja derecha en Y	5
SG90	Ceja izquierda en Y	6
SG90	Ceja derecha en X	7
SG90	Ceja izquierda en X	8
SG90	Labio lateral izquierdo delantero	9
SG90	Labio lateral izquierdo trasero	10
SG90	Labio lateral derecho delantero	11
SG90	Labio lateral derecho trasero	12
XL-320	Motor derecho mandíbula	13
XL-320	Motor izquierdo mandíbula	14
XL-320	Labio frontal maxilar	15
XL-320	Labio frontal mandíbula	16
AX-12A	Motor derecho cuello	17
AX-12A	Motor izquierdo cuello	18
AX-12A	Rotación cuello	19

Cuadro 8: Cuadro que muestra el identificador de cada motor para el envío del protocolo con su nombre correspondiente [2]

10.2. Entrada y salida del modo manual

Se estableció que las pantallas 2 y 3 de la interfaz gráfica pertenecerían al modo manual ya que en estas se modifican manualmente cada servomotor a placer. Es por ello que se consideró necesario que el microcontrolador conozca en que momentos se le enviará valores de motores en específico o no. Para esto se realizó el protocolo de modo manual en donde únicamente se envían tres caracteres con codificación utf-8 como se observa en el Cuadro 9. El primer y último caracter son los anteriormente mencionados para iniciar y detener la comunicación con el microcontrolador. El segundo caracter es el escogido para este caso que se mencionó en el Cuadro 6. Este protocolo se envía al salir y entrar de la segunda pantalla.

Byte	0	1	2
Caracter	<	*	>

Cuadro 9: Cuadro del protocolo para enviar el estado del modo manual

10.3. Envío de la respuesta de los sistemas

Debido a la alto coste computacional de los sistemas de detección de emociones y el *chatbot* se decidió enviar el texto de respuesta de estos sistemas al microcontrolador para que más adelante este texto pueda ser convertido a posiciones de servo para simular el habla del rostro animatrónico en el microcontrolador. Este protocolo se envía al obtener una respuesta del *chatbot* luego de hablarle ó cuando se detecta una emoción luego de tomarse la foto con el reconocimiento de emociones. El protocolo a utilizar se puede ver en el Cuadro 10 en donde nuevamente podemos ver como primer y último byte los caracteres de inicio y fin de la comunicación y como segundo byte el seleccionado para este caso del Cuadro 6.

En este protocolo se espera enviar el texto obtenido de la selección aleatoria de respuestas ya sea del sistema del *chatbot* o de la detección de emociones. Por ejemplo, en la Figura 54 se puede observar que a primera respuesta a escoger en caso se detecte la emoción de enojo es "¿Por qué estás enojado?". Siguiendo el protocolo del cuadro 10 se espera que el protocolo envíe: "<$¿Por qué estás enojado?>"

Byte	0	1	2	3	4	5	6	7	8	9	10	11
Caracter	<	$	R	e	s	p	u	e	s	t	a	>

Cuadro 10: Cuadro del protocolo para envío de la respuesta de los sistemas. El cuadro indica el orden de los caracteres que se deben enviar por comunicación serial al microcontrolador

10.4. Solicitud del estado de los motores

Al ingresar a la segunda pantalla se necesitan los valores actuales de cada motor por lo que solicitan dichos valores al microcontrolador. El protocolo utilizado se puede observar en el Cuadro 11 para la solicitud del estado de los 19 motores. Al momento de enviar dicho protocolo, la interfaz queda a la espera de estos valores. Para el envió del estado de los 19

motores, el microcontrolador utiliza el protocolo del Cuadro 12. Se puede observar que se utiliza la misma generalidad que los protocolos anteriores, en donde se tiene el mismo byte de inicio junto a su caracter especial seleccionado para este caso, pero se tiene la diferencia del byte de finalización. Para este caso se tiene como finalización el byte de nueva linea como se observa en el cuadro, esto es porque se utiliza la función de la librería pyserial llamada *readline()*. Esta función lee el buffer de la comunicación serial hasta encontrarse con los bytes de nueva línea. Como se observa en el cuadro, se espera recibir el valor del estado de cada motor en grados separados por comas. El orden en que se deben enviar los valores es el mismo que el Cuadro 8 en donde el identificador es el orden en que se envía siendo de menor a mayor.

Byte	0	1	2
Caracter	<	&	>

Cuadro 11: Cuadro del protocolo para solicitar el estado de los motores. El cuadro indica el orden de los caracteres que se deben enviar por comunicación serial al microcontrolador

Byte	0	1	2	3	4	5	6	7	8		n-2	n-1	n
Caracter	<	&	1	8	0	,	4	5	,	...	>	x0D	x0A

Cuadro 12: Cuadro del protocolo que utiliza el microcontrolador para enviar el estado de los 19 motores

10.5. Envío de mínimos y máximos de los servomotores

Ya que se decidió que el microcontrolador debe mover los servomotores según las respuestas de los sistemas, este debe conocer los mínimos y máximos de cada servomotor que se establecen en la interfaz. Este protocolo se envía automáticamente luego de una conexión exitosa establecida la cuál se puede observar en el Cuadro 13. Como se observa, se utiliza los bytes de inicio y final como en los otros protocolos y como segundo byte el caracter especial seleccionado para este caso en el Cuadro 6. En este caso se espera enviar el valor mínimo en grados separado por una coma y seguidos del valor máximo en grados. Para la separación entre cada motor se envía el caracter de punto y coma. El orden en que se envían los datos es el mismo que en el Cuadro 8 en donde el identificador es el orden en que se envía siendo de menor a mayor.

Byte	0	1	2	3	4	5	6	7	8	9		n
Caracter	<	%	m	i	n	,	m	a	x	;	...	>

Cuadro 13: Cuadro del protocolo para enviar los mínimos y máximos de cada servomotor. El cuadro indica el orden de los caracteres que se deben enviar por comunicación serial al microcontrolador

CAPÍTULO 11

Manual de instalación

La instalación de la aplicación fue diseñada para poder seguir desarrollando el proyecto de una manera cómoda y sencilla. Se busca poder disminuir el tiempo invertido en hacer correr la aplicación por primera vez y que pueda ser ejecutada sin importar la máquina que se utilice. Para este proceso de instalación se necesita que la máquina posea lo siguiente: sistema operativo de Windows 10 u 11, contar con permisos de administrador en el usuario, conexión a internet, tener instalado Python 3.9 o superior y tener instalado github en la computadora. La forma de empaquetar las librerías de Python utilizadas es por medio de un ambiente virtual de desarrollo ligero. A continuación se describirán el proceso de instalación que se debe seguir para poder correr la aplicación de manera exitosa.

Para poder crear ambientes virtuales ligeros de Python debemos asegurarnos que la máquina tenga habilitado la ejecución de *scripts*.

1. En el buscador de Windows buscar la aplicación de PowerShell y ejecutarla como administrador

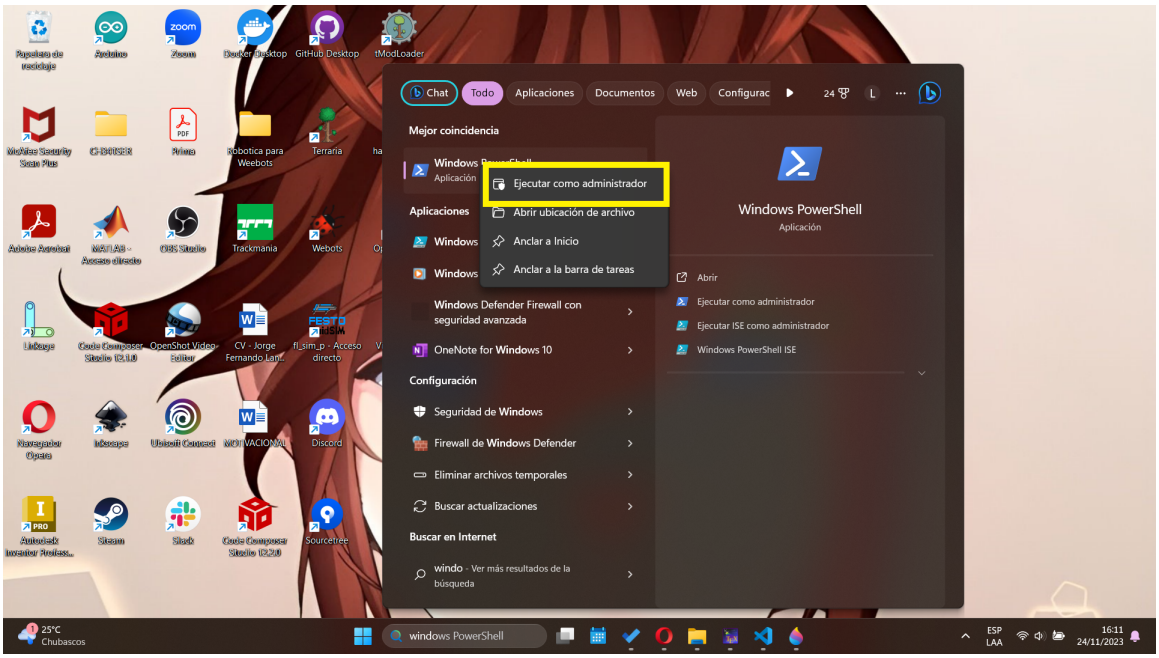


Figura 55: Ejecución de PowerShell como administrador

2. Ejecutar el siguiente comando dentro de la terminal, se debe verificar si en la lista el CurrentUser aparece como Undefined o RemoteSigned. En caso aparezca como en la Figura 56 se debe proceder al paso 3, si ya se tiene como RemoteSigned se debe pasar al paso 4.

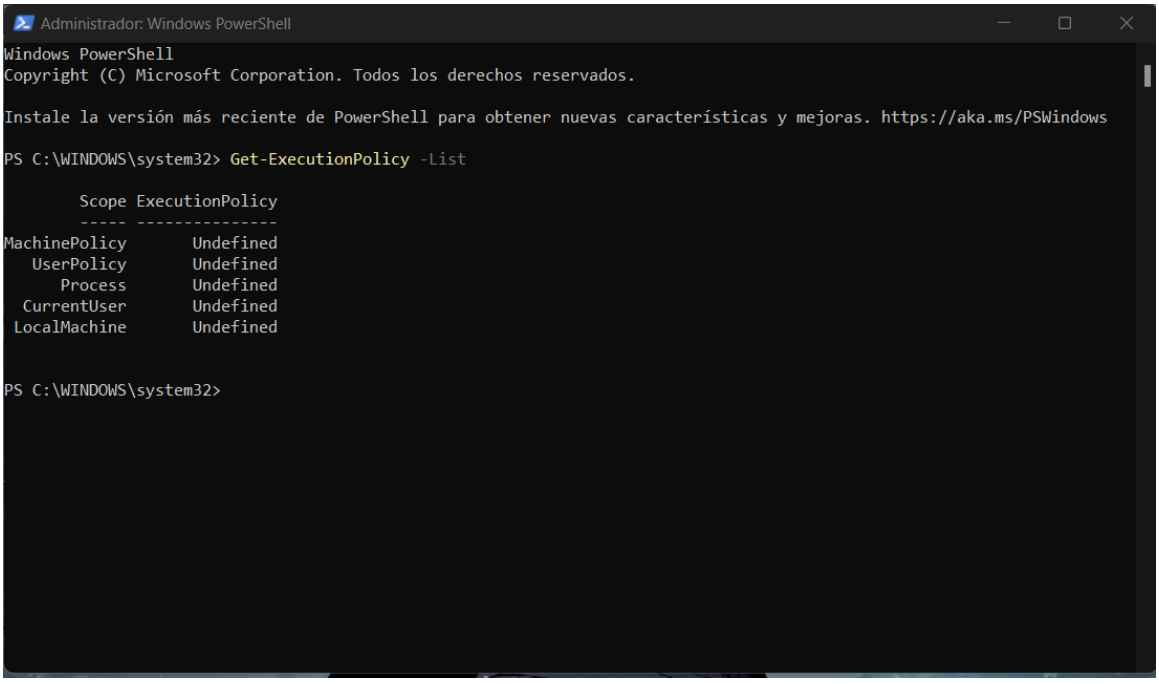


Figura 56: Lista de ejecución de scripts

3. Ejecutar el comando que aparece en la Figura 57 y luego ejecutar el comando del paso

2, ahora debe de salir la lista como se observa en la Figura 57

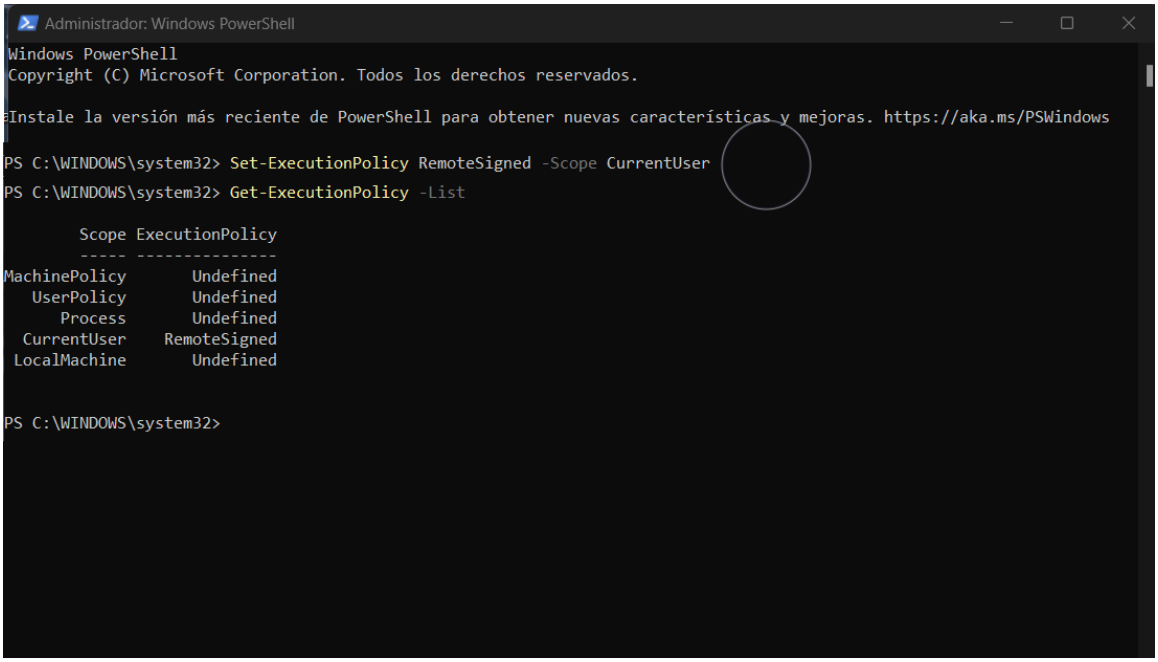


Figura 57: Habilitación de ejecución de scripts en el usuario

4. Clonar el repositorio de github del siguiente enlace `https://github.com/kekellner/DI-rostro-animatronico/tree/19175JorgeLanza/RostroAnimatronico2023` en donde se desee almacenar el proyecto y abrir una terminal de comandos en ese directorio

5. En la consola colocar el siguiente comando, esto creará un ambiente virtual de Python con el nombre de myenv en una carpeta llamada con el mismo nombre como se observa en la Figura 59

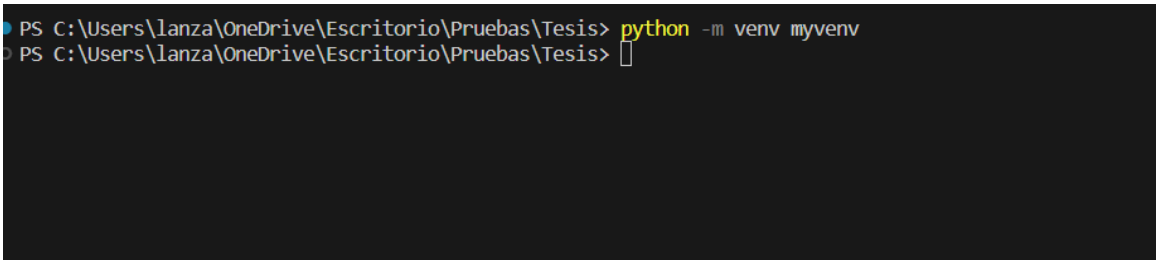


Figura 58: Creación de un ambiente virtual de desarrollo de python

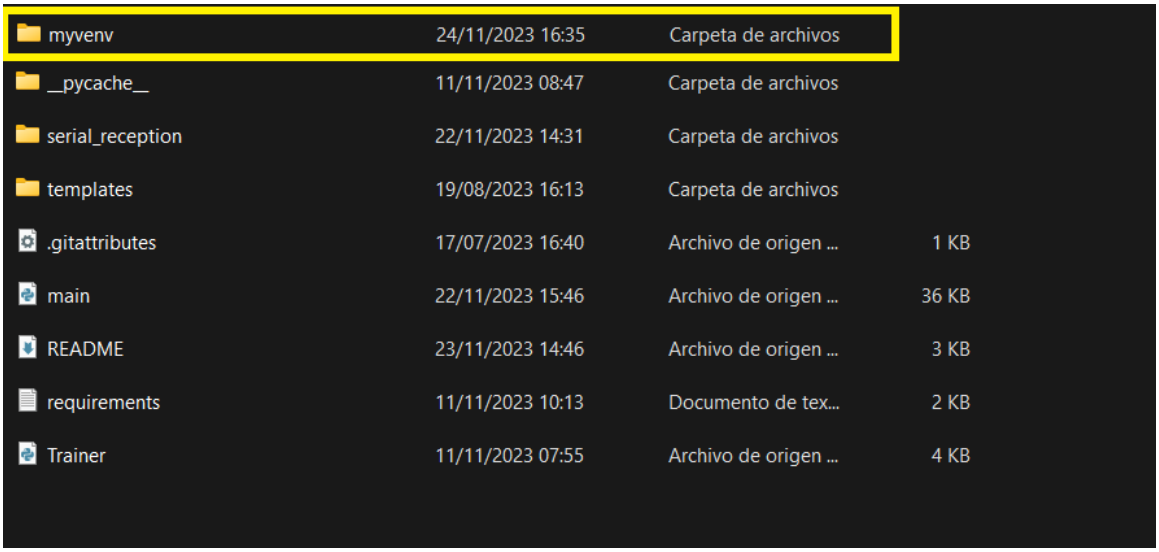


Figura 59: Creación de la carpeta del ambiente virtual

6. Se debe activar el ambiente virtual de Python para poder trabajar sobre el por lo que se debe ejecutar el siguiente comando. Debe aparecer el nombre del ambiente virtual antes de la ruta para saber que ya esta activada como se observa en la Figura 60

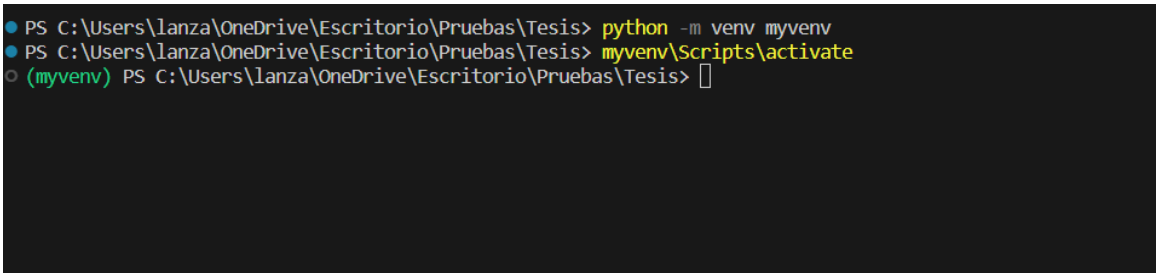


Figura 60: Activación de ambiente virtual

7. Ahora se debe instalar las librerías necesarias para el proyecto, para eso se debe ejecutar el siguiente comando. Esto instalará todas las librerías automáticamente por lo que puede tardar un poco.

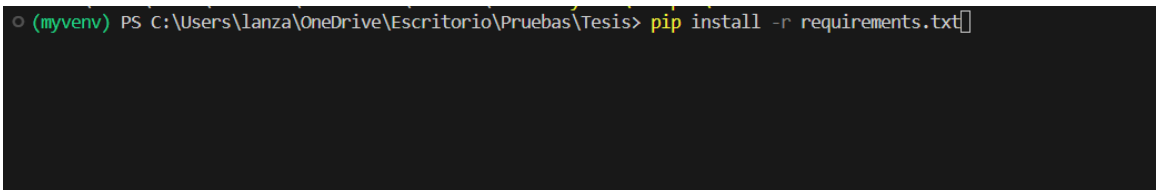


Figura 61: Instalación de librerías en el ambiente virtual de Python

8. Debido a un problema de incompatibilidad se debe bajar de versión la librería numpy de Python por lo que se debe ejecutar el siguiente comando

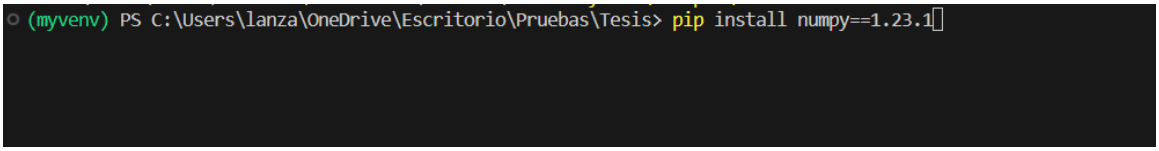


Figura 62: Instalación de versión anterior de librería numpy

9. Por último, ya se puede correr la aplicación ejecutando el siguiente comando

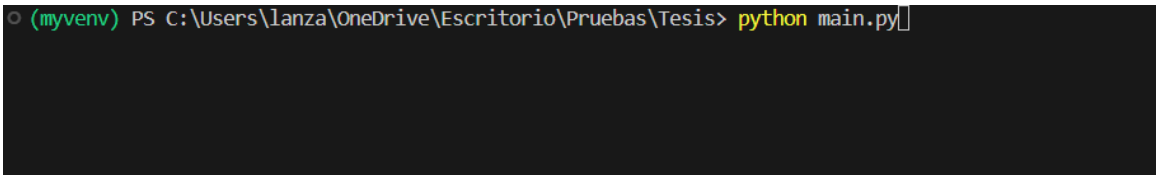


Figura 63: Ejecución de aplicación

CAPÍTULO 12

Conclusiones

- Se encontró el escenario óptimo para el sistema de reconocimiento de emociones siendo este en un lugar con mucha luz ambiental y teniendo la persona a identificar a una distancia de un metro desde la cámara que se este utilizando.
- El sistema que utiliza el *chatbot* para traducir la voz a texto posee una peor porcentaje de detección si se encuentra en un ambiente con mucho ruido.
- Se integraron correctamente los sistemas de reconocimiento de emociones, el sistema de *chatbot* y el despliegue del estado de motores satisfactoriamente en una sola interfaz gráfica
- Es posible realizar la modificación de la base de datos del sistema de *chatbot* por medio de la interfaz gráfica.
- Se diseñó un protocolo de comunicación capaz de enviar y recibir los datos de los motores por medio de UART utilizando comunicación serial
- Se logró la creación de un proceso de instalación sencillo y rápido en diferentes computadoras

CAPÍTULO 13

Recomendaciones

- Crear una función capaz de recibir texto y traducirlo a valores de los 19 servomotores que se ejecute en el microcontrolador. Se espera que esta función sea capaz de sincronizar la respuesta de los sistemas con movimientos similares a un humano hablando.
- Desarrollar un sistema de inteligencia artificial para volver más realista la interacción del animatrónico con el usuario. Se recomienda utilizar sistemas más sofisticados como ChatGPT para tener una mayor fluidez en las respuestas y no depender únicamente de las opciones predefinidas o de temas seleccionados limitados.
- Suprimir el ruido lo más posible para evitar perdidas en la detección del *chatbot*. Se demostró en este trabajo la influencia en el rendimiento que tiene el ruido en el sistema de *chatbot* por lo que se recomienda el uso de un micrófono con buenas características o un programa que pueda suprimir el ruido.
- Acomodar lo mejor posible el ambiente alrededor del rostro animatrónico a su condición ideal. Se espera tener buena luz ambiental y poco ruido para el mejor desempeño posible de sus sistemas.
- Se recomienda continuar con la investigación de este proyecto para poder crear un producto final y llamativo en actividades de captación de la universidad. Actualmente no se realizaron pruebas para la sincronización entre la interfaz gráfica y la parte física del proyecto. Se debe procurar una buen comportamiento al momento que el *chatbot* este hablando con un movimiento natural de la boca en el animatrónico.
- Actualmente la instalación de la aplicación esta enfocada a un ambiente de desarrollo, al momento de finalizarla y crear un proyecto final se recomienda volver la aplicación un ejecutable para una mayor facilidad en su ejecución al igual que explorar la posibilidad de que la aplicación corra en máquina diferentes a Windows como por ejemplo Linux. Se recomienda seguir el siguiente tutorial para hacerlo `https://www.youtube.com/watch?v=cewc02KuHoo`

CAPÍTULO 14

Bibliografía

[1] C. A. Jenatz, "*Diseño de cabeza y cuello animatrónicos con dieciocho grados de libertad con movimientos suaves*," Tesis de licenciatura, Universidad Del Valle de Guatemala, 2020.

[2] M. A. Soto, "*Rediseño e implementación de rostro animatrónico de Universidad del Valle de Guatemala*," Tesis de licenciatura no publicada, Universidad Del Valle de Guatemala, 2023.

[3] D. E. Fuentes, "*Implementación de un chatbot a través de reconocimiento de voz en tiempo real entre el usuario y el rostro animatrónico de la Universidad del Valle de Guatemala*," Tesis de licenciatura, Universidad Del Valle de Guatemala, 2021.

[4] K. D. González, "*Diseño e Implementación de un Sistema para Reconocimiento Facial, de Gestos y de Voz para un Rostro Animatrónico*," Tesis de licenciatura, Universidad Del Valle de Guatemala, 2021.

[5] J. A. Lorenzana, "*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*," Tesis de licenciatura, Universidad Del Valle de Guatemala, 2021.

[6] E. Bevacqua, M. Mancini y C. Pelachaud, "*Speaking with emotions*," en *Proceedings of the AISB Symposium on Motion, Emotion and Cognition*, 2004, págs. 197-214.

[7] S. L. S. López, E. Zalama y J. G. García-Bermejo, "CABEZA MECATRÓNICA CON INTELIGENCIA EMOCIONAL Y ARTIFICIAL,"

[8] S. Rey y J. M. Canas, "VisualHFSM 5: recent improvements in programming robots with automata in JdeRobot," *Málaga, Spain-June 2016*, pág. 155,

[9] S. Lauesen, *User interface design: a software engineering perspective*. Pearson Education, 2005.

[10] Kivy, *Kivy: Cross-platform Python Framework for NUI*. dirección: `https://kivy.org/`, Accessed: 13.05.2023.

[11] *PEP 8 – Style Guide for Python Code*. dirección: `https://peps.python.org/pep-0008/`, Accessed: 13.05.2023.

[12] *PEP8-Python Code Writing Guide*. dirección: https://strateh76.medium.com/pep8-python-code-writing-guide-1788093205f8, Accessed: 14.05.2023.

[13] C. C. R.-C. 3. Unported, *Emoticono de alegría*. dirección: https://www.fundeu.es/consulta/emoticono-emoji/, Accessed: 30.09.2023.

[14] R. Randría, *Emoticono de disgusto*. dirección: https://www.alucare.fr/es/cual-es-el-significado-del-smiley/, Accessed: 30.09.2023.

[15] EmojiTerra, *Emoticono de enojo, miedo, sorpresa y tristeza*. dirección: https://emojiterra.com/es/cabreado/, Accessed: 30.09.2023.

[16] Webmaster, *Emoticono de neutral*. dirección: https://hotemoji.com/straight-face-emoji.html, Accessed: 30.09.2023.

[17] P. Mehanik, *Herramientas Inteligentes*. dirección: https://play.google.com/store/apps/details?id=com.pcmehanik.smarttoolbox&hl=es_EC, Accessed: 30.09.2023.

[18] Freepik, *Escuchando icono gratis*. dirección: https://www.flaticon.es/icono-gratis/escuchando_3898069, Accessed: 30.09.2023.

[19] icon0, *Celular*. dirección: https://es.vecteezy.com/png/9362935-smartphone-icono-signo-simbolo-diseno, Accessed: 30.09.2023.

[20] juicy_fish, *Asistente De Robot icono gratis*. dirección: https://www.flaticon.es/icono-gratis/asistente-de-robot_5828608, Accessed: 30.09.2023.

[21] PNGWING, *reconocimiento de voz ordenador iconos conversación inglés voz pasiva, habla, diverso, Inglés, cara png*. dirección: https://www.pngwing.com/es/free-png-nhniz, Accessed: 30.09.2023.

[22] Freepik, *Red Neuronal icono gratis*. dirección: https://www.flaticon.es/icono-gratis/red-neuronal_8616594, Accessed: 30.09.2023.

[23] F. Icons, *Inteligencia Artificial icono gratis*. dirección: https://www.flaticon.es/icono-gratis/inteligencia-artificial_1610171, Accessed: 30.09.2023.

[24] Freepik, *Servo icono gratis*. dirección: https://www.flaticon.es/icono-gratis/servo_6276775, Accessed: 30.09.2023.

[25] S. Electronics, *Arduino y Dynamixel AX-12*. dirección: https://savageelectronics.blogspot.com/2011/01/arduino-y-dynamixel-ax-12.html, Accessed: 30.09.2023.

[26] G. Robots, *Dynamixel XL-320 servo motor*. dirección: https://www.generationrobots.com/en/401692-dynamixel-xl-320-servo-motor.html, Accessed: 30.09.2023.

[27] FR_Design, *Guardar (icono de disquete) botón cuadrado verde suave*. dirección: https://depositphotos.com/es/photo/save-floppy-disk-icon-soft-green-square-button-131530968.html, Accessed: 30.09.2023.

[28] EmojiTerra, *Emoticono de enojo, miedo, sorpresa y tristeza*. dirección: https://emojiterra.com/, Accessed: 30.09.2023.

[29] A. Jonama, *Payklever*. dirección: https://www.crunchbase.com/person/alexis-jonama, Accessed: 30.09.2023.

[30] seekicon, *SETTINGS ICON - 7*. dirección: https://seekicon.com/free-icon/settings_7, Accessed: 30.09.2023.