Universidad Zaragoza
1542

Trabajo Fin de Grado

Desarrollo de personalidad e inteligencia emocional en un robot educativo para fomentar la interacción entre humano y robot

Development of educational robot's personality and emotional intelligence for promoting interaction between human and robot

Autor

Belén Quintas Herraiz

Director

Manuel Bernal Lecina

Ponente

Javier Civera Sancho

ESCUELA DE INGENIERÍA Y ARQUITECTURA
2022

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe remitirse a seceina@unizar.es dentro del plazo de depósito)

D./Dª. Belén Quintas Herraiz ,

en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de Estudios de la titulación de

selecciona titulación

Ingeniería electrónica y automática

(Título del Trabajo)

Desarrollo de personalidad e inteligencia emocional en un robot educativo para fomentar la interacción entre humano y robot

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 30 de agosto de 2022

Fdo:

AGRADECIMIENTOS

Quiero dar las gracias en primer lugar a mi tutor, Manuel Bernal, por todo el tiempo invertido, el apoyo y las herramientas ofrecidas para hacer posible este proyecto. También al equipo de LoCoCorp, por echarme siempre una mano cuando la necesito. A mis padres, por su confianza en mí y por el sinfín de oportunidades que me brindan para poder llegar a donde me propongo. A los que siempre están ahí para mí, sin vosotros no sería quien soy.

Gracias a todos.

RESUMEN

Hoy en día la interacción entre humanos y robots es un área de rápido crecimiento, presente en multitud de aplicaciones potenciales. El desarrollo de personalidades en robots es un factor que ayuda a entender la naturaleza de tales interacciones, ofreciendo la posibilidad de reconocer emociones y comportamientos humanos e integrarlos para mejorarlas. Este trabajo se enmarca en el campo de la robótica educativa, y su objetivo principal es el desarrollo de personalidad e inteligencia emocional en el robot LoCoQuad de la start-up LoCoCorp. Para llevarlo a cabo, será necesaria la implementación de un módulo de visión artificial que permita capturar información emocional del usuario a partir de imágenes. Otro de los objetivos es el desarrollo de algoritmos de aprendizaje automático en el lenguaje de programación Python, que hagan posible una correcta interpretación de la información recibida. Por último, se realizará la integración en el robot para ofrecer una respuesta mecánica acorde al estímulo detectado y su personalidad.

ABSTRACT

Nowadays, human-robot interaction is a rapidly growing area, present in a multitude of potential applications. The development of personality in robots helps to understand the nature of such interactions, offering the possibility to recognize human emotions and behaviors and integrating them to improve such interactions. This work is framed in the field of educational robotics, and its main objective is the development of personality and emotional intelligence in the LoCoQuad robot, developed by LoCoCorp start-up. To carry it out, it will be necessary to implement a computer vision module to capture emotional information from the user via images. Another objective is the development of Machine Learning algorithms in the Python programming language, which will enable a correct interpretation of the information received. Finally, the system will be integrated in the robot to offer a mechanical response according to the detected stimulus and its personality.

Índice

Capítulo 1

Introducción y objetivos

LoCoCorp es una start-up dedicada a la robótica educativa, cuyo propósito es la creación de robots como herramienta de aprendizaje activo para incentivar el interés por las áreas de conocimiento tecnológico y científico, o también llamadas STEAM (acrónimo en inglés para Science, Technology, Engineering, Arts and Mathematics). Actualmente, LoCoCorp se encuentra en la fase de desarrollo de la 4ª versión de su robot educativo LoCoQuad. El robot está diseñado para ofrecer una herramienta de aprendizaje a jóvenes o iniciados en tecnología, y en concreto, en el mundo de la robótica. Posee una estructura modular para facilitar el manejo y la enseñanza de todos los elementos que lo componen, además de permitir la inclusión de otros módulos externos mediante la conexión I2C.

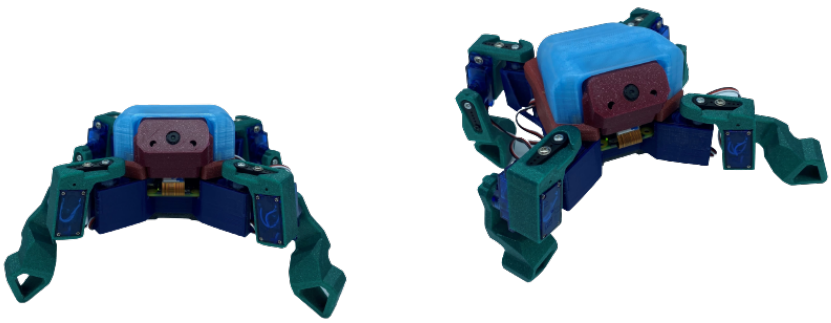


Figura 1.1: LoCoQuad V4 con cámara

Este proyecto surge de la idea de implementar métodos de visión artificial, aprovechando la característica modular del LoCoQuad, para fomentar la interacción entre usuario y robot. Proveer al robot de información visual del sujeto con el que interactúa resulta muy beneficioso en el ámbito educativo, ya que permite adaptar la enseñanza a sus necesidades individuales, además de fomentar su implicación en el proceso de aprendizaje. Dadas la numerosas aplicaciones posibles de la visión artificial en el campo de la robótica educativa, en este trabajo se ha optado por el reconocimiento de expresiones faciales asociadas a emociones durante la interacción con el robot.

El primer objetivo de este trabajo es, por tanto, la elección de componentes relativos al módulo de visión, teniendo en cuenta que deben ser compatibles con las características mecánicas y electrónicas del robot. En lo relativo al hardware, en concreto a la placa de control del módulo y la cámara, se han valorado las alternativas que ofrecen características que permitan una interacción fluida y a su vez de bajo coste, además de contar con una interfaz de comunicación compatible con la placa de desarrollo integrada. Con respecto al software, se requiere un sistema operativo de código abierto, compatible con el lenguaje de programación Python, permitiendo su uso libre a nivel educativo.

Para dotar al robot de inteligencia emocional y capacidad de reconocer emociones, es necesaria una correcta interpretación de la información visual captada. Por consiguiente, el segundo de los objetivos consiste en el desarrollo de algoritmos que permitan detectar tanto rostros como expresiones faciales concretas. Para ello, se hará uso de algoritmos de aprendizaje automático por su efectividad en aplicaciones de clasificación a partir de imágenes. El desarrollo de estos algoritmos se realizará en el lenguaje de programación Python, como se ha mencionado anteriormente. Asimismo, para la creación de personalidad, se diseñará un modelo emocional inspirado en una cadena de Markov que, mediante vectores de probabilidad, determinará la sucesión de estados emocionales del robot durante la interacción con el usuario.

Por último, y como tercer objetivo, se efectuará la implementación en el robot LoCoQuad. Con este fin, se desarrollará un programa modular, que además de captar y procesar las imágenes, permita la comunicación con el controlador de servomotores integrado en la placa de desarrollo del robot.

Capítulo 2

Trabajo Relacionado

En el campo de la interacción humano-robot (HRI, del inglés "Human Robot Interaction") existe un gran interés por los robots sociales, y en concreto, en robots afectivos que posean personalidad e inteligencia emocional. El reconocimiento de emociones y su simulación se han convertido en una importante investigación de cara a mejorar las interacciones humano-robot en multitud de ámbitos, siendo de especial interés en proyectos educativos. Según Paiva et al. [1], para obtener una interacción exitosa, los robots deben estar dotados con procesado emocional para ser capaces de responder emocionalmente a los humanos y adaptarse al contexto ambiental y moral de la sociedad. La aparición de la afectividad y la inteligencia emocional en robots educativos hace de ellos una herramienta ya no puramente cognitiva, sino capaz de ofrecer una respuesta adecuada a los estados emocionales del usuario, permitiendo una adaptación del proceso de aprendizaje a las necesidades específicas del alumno.

Uno de los robots sociales más desarrollados presente en multitud de estudios es el robot NAO, un robot humanoide programable diseñado principalmente para aplicaciones educativas y sociales. Sobre él se han realizado numerosos estudios relacionados con el reconocimiento de emociones, entre ellos uno capaz de reconocer expresiones faciales del usuario mientras este responde preguntas de un test diseñado para aprender gestión emocional [2]. Otro estudio [3] sobre NAO propone el uso de algoritmos de aprendizaje profundo (deep learning) para identificar expresiones faciales con hasta un 91 % de acierto.

Además de algoritmos de inteligencia artificial, en este otro estudio sobre el robot NAO [4] también se proponen otras técnicas como Facial Action Coding

System (FACS), un sistema de seguimiento y medida de los músculos faciales el cual demuestra ser efectivo para este problema.

Otra aplicación de sistemas de reconocimiento facial es Face API, de Microsoft [5]. Puede realizar la detección de emociones en una expresión facial basada en anotaciones percibidas por los codificadores humanos. No obstante, advierte de que las expresiones faciales solas pueden no representar necesariamente los estados internos de las personas.

Con relación a la psicología de las emociones, existe una gran evidencia de que la cara es uno de los principales recursos a la hora de detectar emociones, junto con la voz o la escritura. De acuerdo con Ekman(1999) [6] existen 6 emociones básicas y expresiones faciales diferentes para cada una de ellas. Estas emociones son: asco, ira, sorpresa, miedo, tristeza y felicidad. Dentro de las ciencias cognitivas, el profesor Jordi Vallverdú introdujo la noción de affordance en entornos HRI [7] (una affordance es algo que posibilita y a la vez puede determinar una acción). En este estudio se explica cómo la expresividad del robot puede condicionar la relación con el humano con el que interactúa, buscándose un comportamiento de lógica natural humana para su uso en entornos HRI.

Por otro lado, diferentes arquitecturas han sido propuestas para el desarrollo de un módulo emocional que genere una respuesta afectiva en robots. La mayoría proponen modelos matemáticos para determinar nuevas emociones en el proceso de interacción con humanos, como el modelo de Markov, el cual tiene la capacidad de modelar cambios dinámicos de estados emocionales e incertidumbres modificando las probabilidades de transición a medida que discurre la interacción. En este estudio [8], se propone un modelo de Markov capaz de actualizar la matriz de transición en cada bucle de procesamiento a través de un factor de influencia positivo que determina una nueva matriz de probabilidades, en función de si el robot ha cumplido sus objetivos de asistencia sanitaria o no.

Capítulo 3

Algoritmos de Aprendizaje Automático y Aprendizaje Profundo

Antes de continuar con el desarrollo del proyecto es importante explicar las diferentes técnicas de aprendizaje automático probadas e implementadas, pues contienen una base teórica que no es sencilla y que es necesario conocer para poder entender las decisiones tomadas.

3.1. Redes Neuronales Artificiales

Una red neuronal es un modelo de aprendizaje compuesto de múltiples capas de procesamiento para aprender representaciones de datos con múltiples niveles de abstracción, y que, mediante una serie de transformaciones lineales y no lineales, generan una salida que se entrena para que sea próxima a la esperada. Esta estructura se enmarca dentro del denominado aprendizaje profundo, compuesto por estructuras con múltiples capas de procesamiento compuestas a su vez de un gran número de neuronas.

3.1.1. El Perceptrón

La red más sencilla es la compuesta por una sola neurona, también llamada Perceptron. En la figura 3.1, se muestra dicho modelo de una sola neurona. En este modelo, ante una serie de entradas, la neurona asigna unos pesos a cada una de estas

y les suma un sesgo b. El resultado lo pasará a través de una función de activación no lineal que producirá un resultado entre 0 y 1 (o -1 y 1) dependiendo del tipo de función de activación asociada a la neurona.

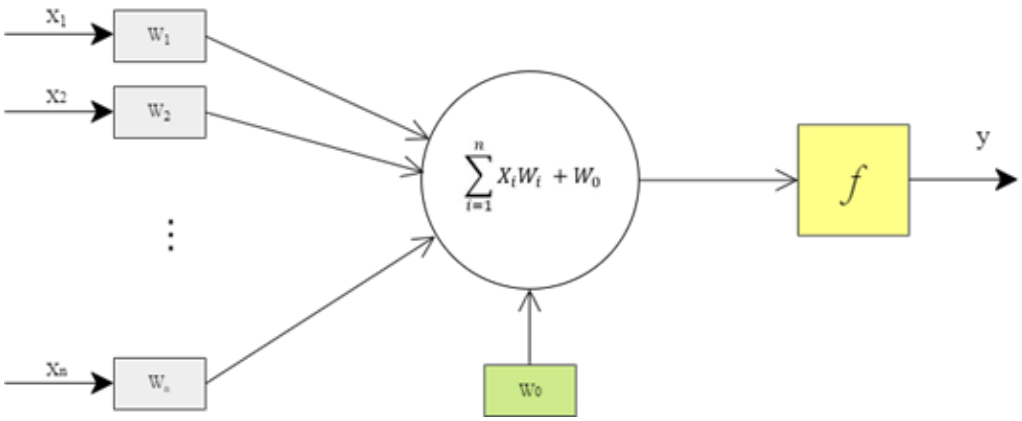


Figura 3.1: Modelo de perceptrón.

3.1.2. Función de activación

La función de activación define la salida de la neurona y se utiliza para calcular el estado de actividad de esta transformando la entrada en un valor de activación, cuyo rango suele ser de 0 a 1 o de -1 a 1. Las funciones más utilizadas son:

- Función sigmoide: rango de valores de 0 a 1. No está centrada en el cero.

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{3.1}$$

- Función tangente hiperbólica: rango de valores entre -1 y 1. Centrada en cero.

$$tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{3.2}$$

- Función ReLU (Rectified Lineal Unit): no está acotada y solo se activa en los positivos.

$$f(x) = max(0, x) = \begin{cases} 0 & si \quad x < 0 \\ \\ x & si \quad x \geq 0 \end{cases} \tag{3.3}$$

– Función Softmax: acotada entre 0 y 1. Salida con representación de probabilidad.

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad para\ i = 1, 2, \ldots, K \tag{3.4}$$

3.1.3. Aprendizaje de una Red Neuronal

Una red neuronal está formada por neuronas, como la explicada anteriormente, conectadas entre sí. Cada conexión está asociada a un peso que determina la importancia de esa relación en la neurona al multiplicarse por el valor de entrada. En el entrenamiento, los datos son propagados hacia adelante, y las neuronas realizan los cálculos necesarios para obtener una predicción. A medida que se entrena el modelo, se ajustan los pesos de las interconexiones en función de la llamada función de pérdida (ec. 3.5), que estima el error cometido en las predicciones en base a la etiqueta correcta y la predicción proporcionada. Con este proceso se busca que la función de pérdida se aproxime a cero, y para ello se hace uso de la técnica llamada descenso de gradiente.

$$-\sum_{c=1}^{M} y_{o,c} \log(p_{o,c}) \tag{3.5}$$

Con el descenso de gradiente (Gradient Descent), se van ajustando los pesos mediante el cálculo del gradiente (la derivada) de la función de pérdida, lo que permite ver en qué dirección se desciende al mínimo global. Este proceso se repite en sucesivas iteraciones (epochs) sobre lotes de datos (batches).

3.1.4. Hiperparámetros

Durante el proceso de aprendizaje existen una serie de parámetros que, mediante su correcta modificación, permiten un entrenamiento mejor y más rápido de los modelos. Estos parámetros son externos al modelo, y son especificados por el programador.

– Epochs: número de veces que los datos han pasado por la red neuronal en el entrenamiento.

– Batch size: tamaño de los lotes de datos de entrenamiento.

– Learning rate: escalar que multiplicado por la magnitud del gradiente determina el siguiente punto a calcular por el algoritmo Gradient Descent. Un valor alto puede agilizar el entrenamiento, pero se corre el riesgo de no llegar nunca a un mínimo global.

Con el estudio de estas redes, se han ido desarrollando otro tipo de redes neuronales, como son las convolucionales, muy similares a las anteriores, que se aplican a tareas de clasificación de imágenes y van a ser probadas en este proyecto. En las redes convolucionales cada capa va aprendiendo diferentes niveles de abstracción en forma de patrones locales en ventanas de dos dimensiones.

3.2. Análisis de Componentes Principales (PCA)

Principal Component Analysis (PCA) [9] es un método estadístico perteneciente a la familia de técnicas de aprendizaje no supervisado, que permite reducir la complejidad dimensional de espacios muestrales conservando la máxima información.

Dado un vector X de dimensión n, su media viene definida como

$$\mu = E\{x\} \tag{3.6}$$

La matriz de covarianza se define como

$$\Sigma_{ij} = E\{(X-\mu)(X-\mu)'\} = \begin{bmatrix} \sigma_{11}^2 & \sigma_{12}^2 & \dots & \sigma_{1n}^2 \\ \sigma_{21}^2 & \sigma_{22}^2 & \dots & \sigma_{2n}^2 \\ \vdots & \vdots & \vdots & \vdots \\ \sigma_{n1}^2 & \sigma_{n2}^2 & \dots & \sigma_{nn}^2 \end{bmatrix} \tag{3.7}$$

Donde σ_{ij} representa la covarianza entre el componente i y el número j del vector X. El método PCA consiste en encontrar los vectores propios de esta matriz y definir X en función de estos vectores. De esta forma, el vector X se puede representar mediante una combinación lineal de vectores de la forma

$$X = \sum_{i=1}^{n} y_i A_i \tag{3.8}$$

Estos vectores también son conocidos como *eigenvectors*.

Como se ha explicado, para calcular las componentes principales, primero se procede a realizar la centralización de las variables mediante la resta a cada valor de la media de la variable a la que pertenece. Como resultado, todas las variables tienen media cero. A continuación, se extraen los vectores propios de la matriz de covarianza, para lo cual existen varios algoritmos. Entre ellos, el algoritmo SVD (Single Value Decomposition) el cual descompone la matriz en un producto de otras tres matrices. Como resultado, se obtienen los *eigenvectors* de la matriz de covarianza.

Desde un punto de vista geométrico, se muestra en la siguiente imagen un conjunto de muestras de las que se dispone dos variables (dimensiones). El vector que define la primera componente principal sigue la dirección de mayor varianza en las muestras. La proyección de cada observación sobre esa dirección equivale al valor de la primera componente para dicha muestra.

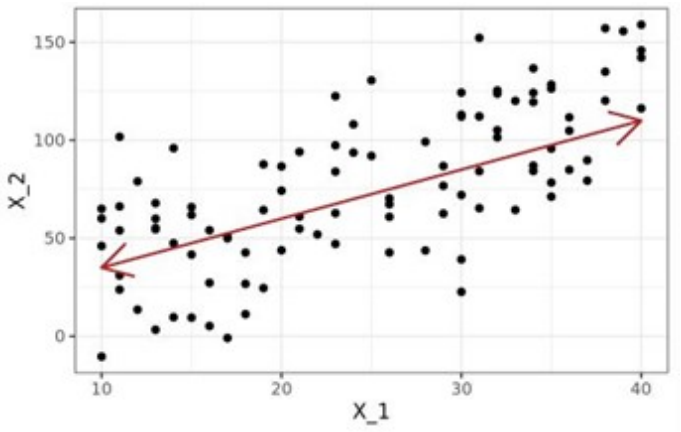


Figura 3.2: Primera componente principal
https://www.cienciadedatos.net

La segunda componente se calcula de la misma forma que la primera, pero con la condición de no correlación con la primera componente, o lo que es lo mismo, sus direcciones han de ser ortogonales.

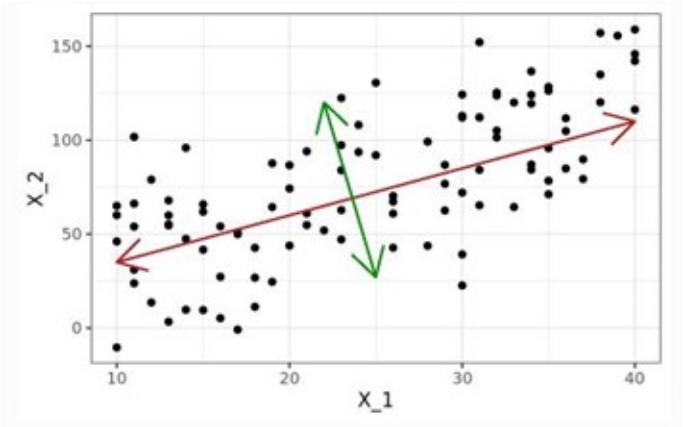


Figura 3.3: Segunda componente principal
https://www.cienciadedatos.net

Debido al uso de las varianzas, la técnica PCA es muy sensible a outliers (valores atípicos respecto a una dimensión). Por ello, lo ideal es realizar un estudio gráfico de estos, aunque en el caso de un número elevado de dimensiones de los datos, el proceso se complica de forma considerable.

3.3. Máquinas de Vectores de Soporte (SVM)

Las Máquinas de Vectores de Soporte (SVM) [10] son algoritmos de aprendizaje supervisado comúnmente utilizadas para tareas de clasificación, regresión y detección de valores atípicos (outliers). Dado un conjunto de muestras, el objetivo de esta técnica es encontrar un hiperplano, mediante la resolución de un problema de optimización, en un espacio N-dimensional que separe las muestras etiquetadas de entrenamiento. Este plano, a su vez, ha de ser aquel que maximice la distancia entre los conjuntos de muestras de cada clase. Al maximizar este margen se facilita que las futuras muestras sean clasificadas con mayor seguridad.

3.3.1. Hiperplanos

Los hiperplanos son límites de decisión que permiten clasificar las muestras dependiendo de a qué lado del hiperplano ha "caído" dicha muestra. Su dimensión depende del número de características de los datos, por ejemplo, si la dimensión de

los datos de entrada es dos, entonces el hiperplano será una línea. Si la dimensión es tres, el hiperplano será un plano de dos dimensiones. La función kernel sirve para encontrar un hiperplano en casos donde los datos no sean linealmente separables, mediante el aumento a una dimensión donde sí lo sean.

Dado un conjunto de muestras $(x_1, d_1),\ (x_2, d_2),\ ...,\ (x_n, d_n)$ $d = 1$ *o* $d = -1$, el problema de maximización de margen se formula como

$$maxQ(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j d_i d_j x_i^T x_j \tag{3.9}$$

tal que

$$\sum_{i=1}^{n} \alpha_i d_i = 0 \tag{3.10}$$

con

$$0 \leq \alpha_i \leq C \ para \ \ i = 1, 2, ..., n \tag{3.11}$$

Este método se puede extender a conjuntos de datos en espacios de dimensiones superiores $\phi :\ x_i \longmapsto \phi(x_i)$ mediante el uso del kernel

$$K(x_1, x_2) = \phi(x_1)^T \cdot \phi(x_2) \tag{3.12}$$

A continuación se calcula la máquina de vectores de soporte mediante una suma ponderada de las salidas del kernel.

$$f(x) = sign \left[\sum_{i=1}^{l} \alpha_i d_i K(x, x_i) + b \right] \tag{3.13}$$

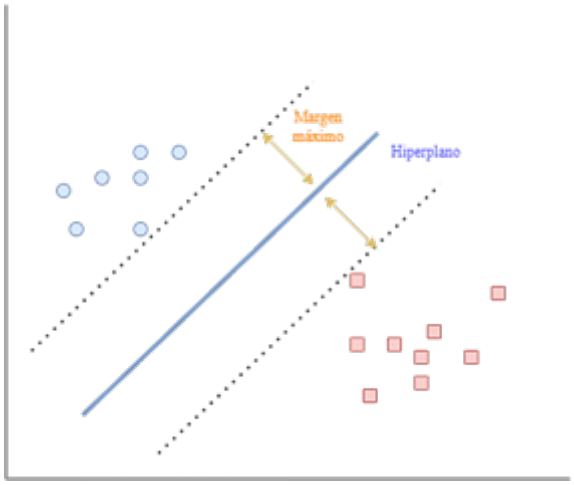


Figura 3.4: Hiperplano de dos dimensiones.

3.3.2. Vectores de Soporte

Los vectores de soporte (fig. 3.5) son el conjunto de muestras más cercanas al hiperplano, y que influencian la posición y orientación de este. Se utilizan para maximizar el margen del clasificador mediante la función de costes y actualización de gradiente. De forma similar a las redes neuronales, el objetivo es hacer que la función de costes se aproxime a 0, lo que significa que se ha encontrado el máximo margen entre clases.

Los vectores de soporte son muestras las cuales su multiplicador lagrangiano (ec 3.9) es distinto de cero.

$$\alpha_i \neq 0 \tag{3.14}$$

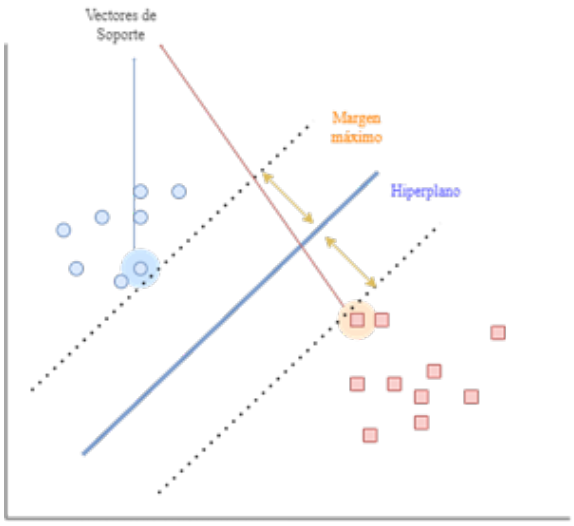


Figura 3.5: Hiperplano y vectores de soporte.

3.3.3. Parámtros C y γ

Con el fin de ajustar el margen de la manera que ofrezca un mejor resultado (problema de optimización), se deben permitir ciertos errores en la clasificación, es decir, permitir que una muestra se clasifique en la clase que no le corresponde. A este error se le asigna una penalización, que podrá ser mayor o menor dependiendo del parámetro C. A mayores valores de C, mayor penalización se le asigna al error y menor margen existirá entre clases, haciendo el clasificador más estricto y con menor varianza. También se corre el riesgo de hacer sobreajuste del modelo (clasificador muy bueno únicamente con los datos de entrenamiento). El caso contrario sucede con valores de C más bajos, donde sucederán más fallos en el entrenamiento por ser más permisivo.

Por otro lado, el parámetro γ (solo afecta en funciones kernel gaussianas, como Radial Basis Function) decide la curvatura del hiperplano. A valores mayores, el modelo será más restrictivo y con tendencia al sobreajuste, mientras que, a valores más bajos, las curvas serán más suaves y el modelo más permisivo, pudiendo obtener mayor número de errores. Ambos parámetros tienen una fuerte influencia sobre el proceso de clasificación, siendo comúnmente elegidas mediante la técnica de validación cruzada.

3.4. Clasificador Haar Cascade + Ada Boost (Viola Jones, 2001)

El clasificador basado en filtros Haar [11] es un método de aprendizaje automático en el que se dispone de una concatenación (cascada) de clasificadores, cada uno analizando una porción de una imagen. Estos clasificadores por sí mismos son débiles, teniendo una alta probabilidad de dar falsos positivos, pero, por el contrario, cuando son combinados resultan muy potentes. Al transmitir una imagen, el algoritmo inicialmente extrae las características de esta mediante el uso de un filtro (kernel) que realiza una convolución sobre los píxeles que se encuentran dentro de la ventana del filtro. Una de las características de este algoritmo es la alta velocidad del procesado, lo cual se consigue mediante la disminución de cálculos que se necesitan para evaluar cada una de las características extraídas, reduciéndolo a una operación que involucra únicamente 4 píxeles por ventana. La extracción de características se realiza por

medio del algoritmo Ada Boost.

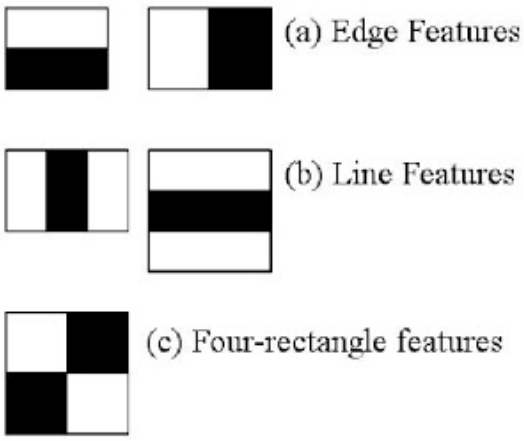


Figura 3.6: Filtros Haar.
https://docs.opencv.org/

Los filtros se disponen en cascada y son entrenados, de forma que, si un filtro determina negativo, se descarta la imagen (no hay detección) sin pasarla al siguiente filtro. Así pues, la detección es positiva cuando una imagen pasa por todos los filtros.

Capítulo 4

Requerimientos del Sistema

Una de las primeras cuestiones a plantearse antes de continuar, es qué hardware se requiere para el desarrollo de este proyecto, puesto que en función de las herramientas utilizadas se tomarán unas decisiones u otras. Se ha tratado de resolver esta cuestión teniendo en cuenta dos enfoques, el educativo y el técnico, al tratarse de un producto comercial y dedicado a la educación. Por un lado, desde un enfoque pedagógico, se precisa de una placa de control que cuente con un microcontrolador programable para que el usuario pueda ejecutar instrucciones y que la conexión con el robot no resulte demasiado compleja. Desde un enfoque técnico, la placa debe contar con una interfaz de comunicación I2C para garantizar la compatibilidad con la placa que suministra por un lado la potencia, y por otro, las conexiones con el controlador de servos y los servomotores. Asimismo, debe poseer unas dimensiones reducidas que permitan su integración en el cuerpo del robot.

En cuanto al software, el sistema operativo que contenga el microcontrolador debe ser de código abierto, para su posible comercialización, y deberá ser compatible con la programación en el lenguaje Python.

Con todas las exigencias de hardware y software planteadas, se ha determinado como mejor alternativa el uso de la placa Raspberry Pi Zero W y el módulo de cámara Rev. 1.3 Raspberry PI Zero.

4.1. Raspberry Pi Zero W

Raspberry Pi Zero W es un computador de placa reducida (SBC) orientado al sector educativo. Su software es de código abierto, el sistema operativo oficial, Raspberry Pi OS (previamente Raspbian), es una versión adaptada de Debian GNU/Linux, el cual debe ser cargado en una tarjeta MicroSD y posteriormente, esta debe ser conectada a la placa. Se detallan en la tabla 4.1 las especificaciones técnicas más relevantes [12] de la Raspberry Pi Zero W que han determinado su elección.

HARDWARE		SOFTWARE
Procesador	Dimensiones	Sistema Operativo
Single Core ARM 1GHz 512MB RAM	65mm x 30mm x 5mm	Raspberry Pi OS

Tabla 4.1: Características de la Raspberry Pi Zero W.

Figura 4.1: Raspberry Pi Zero con cámara.

4.2. Cámara Raspberry Pi Rev 1.3

La elección del módulo de la cámara, Rev. 1.3 para Raspberry PI, se ha determinado debido a la facilidad y calidad de comunicación con la placa, ya que esta ya cuenta con los drivers necesarios para efectuar dicha comunicación. Se ha intentado establecer conexión con la cámara *ArduCam*, pero debido a diversos problemas con la instalación de los drivers, se ha descartado la alternativa. Adicionalmente, las dimensiones del módulo escogido son muy reducidas, por lo que la hace apta para poder ser integrada en el robot. A continuación, se detallan algunas especificaciones técnicas de la cámara.

- Puerto CSI
- Resolución: 5MP 2592x1944
- Grabación de vídeo: 1080p HD a 30fps, 720p a 60fps, 640x480p 60/90
- Campo de visión de 53.50 grados horizontal y 41.41 grados en vertical
- Dimensiones: 25.2 x 23.8 x 8.2 mm

Capítulo 5

Desarrollo

5.1. Estructura del Sistema

La estructura del sistema se diseña para que el robot sea capaz de ejecutar 5 acciones durante la interacción con el usuario:

1. Captura de imagen
2. Detección facial
3. Detección de la emoción a través de la expresión facial del usuario
4. Generación de emoción propia
5. Ejecución de movimiento

Por ello, se opta por establecer una estructura modular, que permita dividir el problema en módulos manejables y con la posibilidad de externalizarlos. No obstante, en este trabajo se ha decidido ejecutar cada uno de ellos desde el procesador local, el de la Raspberry Pi Zero, evitando posibles latencias en el envío de datos. La entrada al sistema de control es la imagen captada del exterior, que contiene la información del estado emocional del usuario (o no, si no se detecta al usuario). Sobre esta entrada, se aplica una serie de procesos que determinarán la nueva emoción del robot y su consiguiente comportamiento.

El sistema de control consta de tres hilos de procesamiento. El primer hilo es el de captura de imagen, el cual se ejecutará constantemente captando imágenes a una

frecuencia aproximada de de 2 frames por segundo. El segundo hilo es el de procesado de imagen, en el que se ejecutarán los procesos de detección de rostro y emoción una vez haya finalizado el movimiento asociado a la emoción previa. Finalmente, el tercer hilo es el de acción. Se ejecutará el proceso una vez finalizado el procesado de imagen, y se encargará de determinar qué emoción siente el robot en ese momento en base a la emoción del usuario detectada, y ejecutará el movimiento acorde a ese estado emocional.

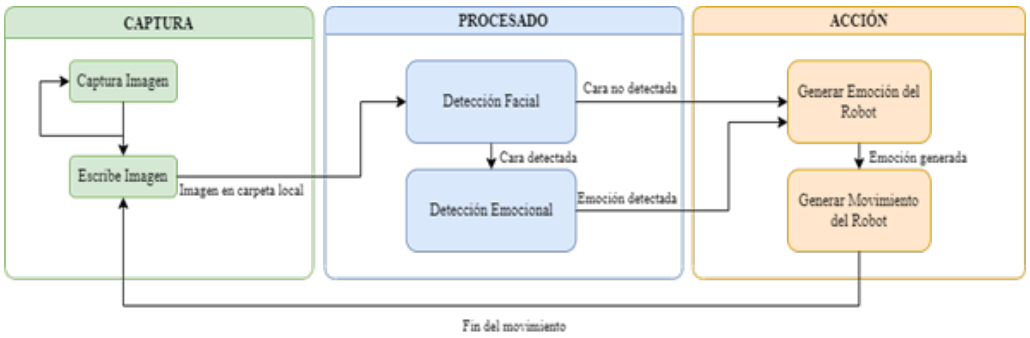


Figura 5.1: Hilos de procesamiento.

En primer lugar, para la obtención de imágenes sobre las que poder detectar una cara (y una emoción), es necesaria la creación de un módulo que se encargue de realizar la captura de imagen desde la cámara. Desde el módulo Cámara, se hace una llamada al módulo de detección facial, el cual se encarga de procesar la imagen para encontrar una cara. Si la cara es detectada, se llama al módulo de detección emocional del usuario, que ejecutará el proceso de reconocimiento de la emoción en el rostro detectado. El módulo de acción recibirá la información de la emoción del usuario detectada , y posteriormente, llamará al módulo de estado emocional del robot para determinar qué emoción siente en ese momento. Una vez generada la emoción, ejecutará los movimientos acordes a ella.

Durante el desarrollo del programa se ha tratado de garantizar un pipeline, o flujo de datos, desde la captura de imagen a la ejecución del movimiento del robot, lo más eficiente posible para ajustarnos al tiempo real lo máximo permitido por el microprocesador de la placa.

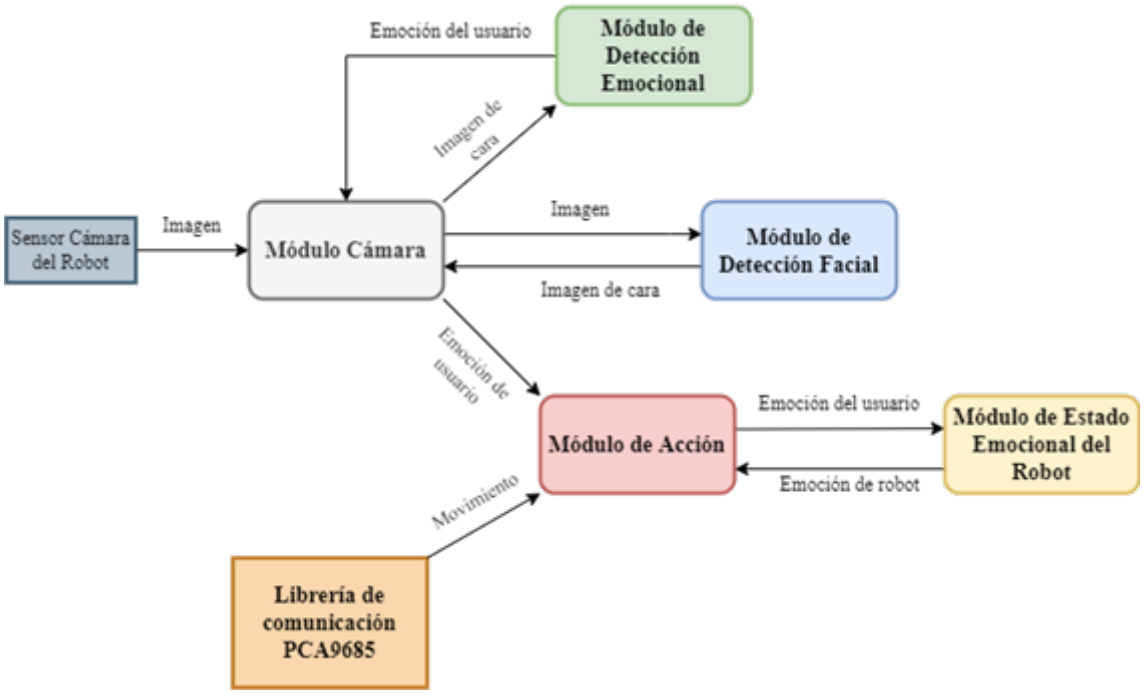


Figura 5.2: Módulos del sistema.

5.2. Captura de Imágenes

La obtención de entradas al sistema se realiza mediante la captura de imágenes en tiempo real. Para ello se han explorado dos vías con el fin de determinar cuál de ellas es más rápida y así poder ofrecer un flujo de datos lo más eficiente posible.

5.2.1. Captura a través de OpenCV

OpenCV es una biblioteca open source de visión artificial utilizada en multitud de aplicaciones, entre ellas, la detección de movimiento, reconocimiento de objetos, reconstrucción 3D entre otros. Es multiplataforma y se puede utilizar en varios lenguajes como C/C++ o Python. El método de obtención de una imagen desde OpenCV comprende los siguientes pasos:

1. Crear un objeto de captura para la cámara
2. Hacer una llamada a la función de lectura usando el objeto creado, que devuelve una variable con la imagen

Una vez capturada una imagen en una iteración del bucle, se ha determinado que la

transferencia de esta a otros módulos del programa se realice mediante la escritura de la imagen en el almacenamiento local de la Raspberry Pi. De esta forma, cualquier módulo podría acceder a esta imagen sin necesidad de complicar el proceso de flujo de datos del programa. Para este método de captura de imagen se ha medido una latencia media de 0.18 segundos (únicamente se ha medido el proceso de escritura, no el de lectura de la imagen).

5.2.2. Captura a través de PiCamera

PiCamera es una librería diseñada para el módulo de la cámara de la Raspberry Pi. Desde ella, se puede capturar una imagen o un vídeo, además de ajustar parámetros relativos a la cámara. Esta última funcionalidad es la que ha impulsado la idea de utilizar esta librería, puesto que se quiere probar a modificar estos parámetros para comprobar si ofrecen ventajas de cara a la latencia del proceso de obtención de imágenes. Se ha probado a reducir la resolución de la cámara al mínimo, 64x64 píxeles y se ha establecido una frecuencia de captura de 10 imágenes por segundo. El proceso de captura es muy similar al anterior, y los resultados muestran una mayor latencia, 0.7 segundos de media, por lo que se ha decidido descartar esta alternativa.

5.3. Módulo de Detección Facial

Constituye la primera fase del procesado de la imagen. En este módulo, se tratará de encontrar una cara sobre la que poder reconocer una emoción. Para ello, se hace uso del algoritmo de Viola-Jones, el cual utiliza clasificadores Haar-like features. Las ventajas que ofrece este método lo hacen una alternativa ideal para la implementación en el módulo de detección facial de este trabajo, pues garantiza robustez y bajo coste computacional, siendo esta última una característica fundamental en aplicaciones de tiempo real. La implementación de este método se ha realizado a través de la librería OpenCV, y se han utilizado los clasificadores entrenados para la detección de rostros desde una perspectiva frontal, *'haarcascade_frontalface'*.

En primer lugar, se crea el objeto clasificador. Posteriormente, se hace una llamada al método de detección, siendo los parámetros la imagen captada (en escala de gris), la ventana mínima de detección y el escalado de la imagen. Estos dos últimos

Figura 5.3: Proceso de la detección facial.

son parámetros muy importantes con relación a la tasa de acierto y la velocidad del detector. Cuanto menor sea la ventana de detección, mayor número de ventanas se evaluarán, aumentando significativamente el tiempo de cómputo. De la misma manera, cuanto menor sea el escalado de la imagen, mayor tiempo se requerirá para la realización de esta acción. Por último, se aplica sobre esta imagen una función de preprocesado con objeto de optimizar el clasificador de emociones, que se detallará más adelante.

(a) Imagen (b) Imagen de cara reconocida

Figura 5.4: Detección de caras con Haar-like features

5.4. Módulo de Detección Emocional

Este módulo contiene el modelo del clasificador y la función de detección de emociones. Recibirá la imagen preprocesada de la cara detectada a modo de entrada, y como salida, se determinará la emoción del usuario.

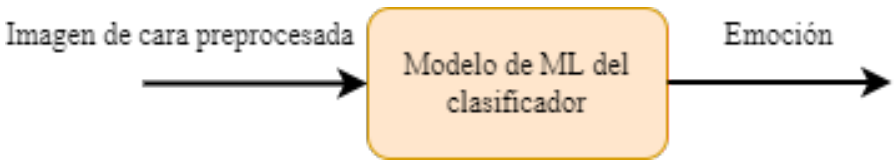


Figura 5.5: Proceso de la detección emocional.

La clasificación por imagen es un campo extensamente estudiado por la rama de la inteligencia artificial, y en concreto, por el aprendizaje automático o machine learning, ofreciendo muy buenos resultados para este tipo de aplicaciones. Por esta razón se ha decidido hacer uso de estos algoritmos en el desarrollo del clasificador.

5.4.1. Aprendizaje Automático sobre Raspberry Pi Zero W

Antes de comenzar con el desarrollo del modelo, se han de evaluar las condiciones del entorno en el que se ejecuta la aplicación, la placa Raspberry Pi Zero W. Se deberán tener en cuenta sus limitaciones, dada la escasa potencia de la GPU, escasa memoria RAM y un microprocesador de un solo núcleo no demasiado potente, de 1GHz de frecuencia.

Teniendo en cuenta las mencionadas limitaciones, se ha decidido realizar el entrenamiento y la validación de los modelos en una máquina con una GPU más potente y posteriormente integrar el modelo ya entrenado en la placa. Por otro lado, la memoria RAM del microprocesador (512 MB), es suficiente para ejecutar un modelo de aprendizaje automático ya entrenado. No obstante, se deberá tener en cuenta que la frecuencia de reloj del microprocesador no es muy alta, de 1GHz, por lo que se ha de contemplar el número de parámetros del modelo, ya que probablemente influyan en su latencia.

5.4.2. Desarrollo del modelo

Las fases de desarrollo de este módulo son 3. Se comenzará con la elección, evaluación y/o creación del dataset a utilizar por el algoritmo. Posteriormente se dará comienzo a la fase de entrenamiento, donde también se escogerá el algoritmo de aprendizaje automático que ofrezca mejores resultados, y para finalizar, se realizará la fase de pruebas (test) de los modelos.

Elección del Dataset

Para llevar a cabo el entrenamiento y el conjunto de pruebas se utilizarán tres bases de datos. Dos estándar, CK+ Dataset y FER2013 Dataset, y una propia. Es importante añadir que se ha decidido clasificar 3 (enfado, felicidad, tristeza) de las 6

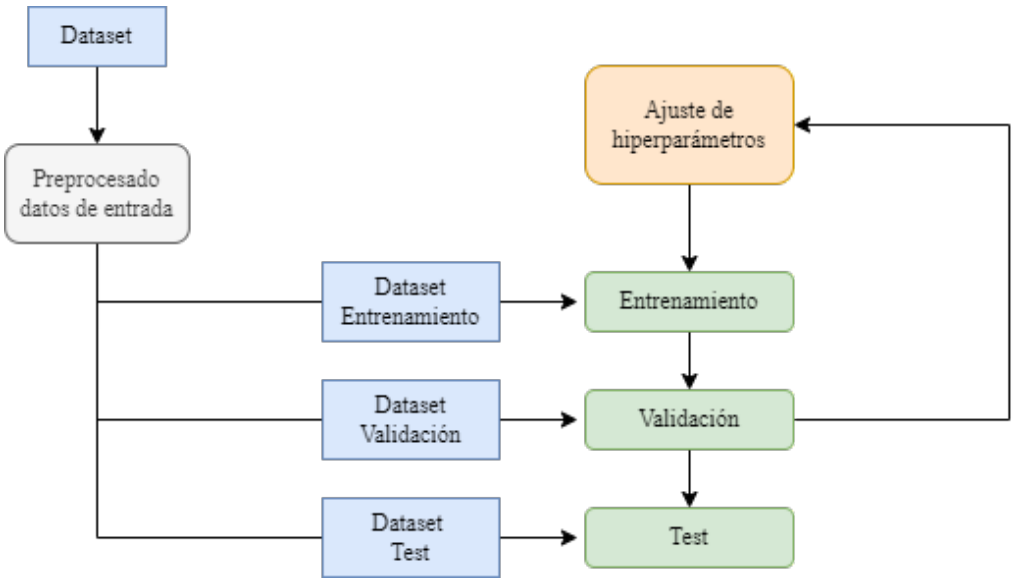


Figura 5.6: Proceso de desarrollo de un modelo de Machine Learning.

emociones básicas y un estado neutral por ser, bajo nuestro criterio, las expresiones más representativas y relevantes en el proceso de interacción con el usuario. De esta forma se agiliza tanto el proceso de creación de la base de datos como el propio entrenamiento de los modelos. Por este motivo, se harán modificaciones a los datasets mencionados a continuación para adaptar el clasificador a las necesidades de la aplicación.

– CK+ Dataset *Extended Cohn-Kanade*(CK) dataset [13] es la base de datos más utilizada para el desarrollo de algoritmos de detección facial. Contiene 593 secuencias de 123 sujetos y están etiquetadas en 7 categorías de emociones básicas: Desprecio, Enfado, Miedo, Felicidad, Tristeza y Sorpresa. Es un dataset de dominio público y se puede obtener directamente desde Kaggle, una comunidad de machine learning y ciencia de datos.

(a) (b) (c) (d) (e) (f)

Figura 5.7: Imágenes del dataset CK+.

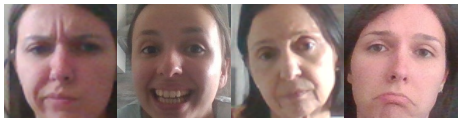
– FER2013 Dataset *Facial Expression Recognition 2013* [14] es un dataset que contiene 35887 imágenes de 48x48 píxeles de resolución, obtenidas en entornos

no controlados. Las imágenes se clasifican en 7 categorías: Enfado, Disgusto, Miedo, Felicidad, Tristeza, Sorpresa y Neutral. Comparado a otras bases de datos, FER ofrece más variabilidad en las imágenes incluyendo oclusión facial, objetos en la cara (como gafas de sol o pelo), e imágenes con poco contraste.

(a) (b) (c) (d) (e) (f) (g)

Figura 5.8: Imágenes del dataset FER2013.

– Base de datos propia Se ha construido una base de datos formada por 160 imágenes de un total de 3 sujetos. Las imágenes tienen un tamaño de 176x176 píxeles y han sido obtenidas a través de la webcam del PC. Se clasifican en 4 emociones básicas: Enfado, Felicidad, Neutral, Tristeza.



(a) (b) (c) (d)

Figura 5.9: Imágenes del dataset propio.

Entrenamiento

Durante esta etapa se han realizado los entrenamientos sobre las diferentes alternativas planteadas. Tanto los algoritmos de aprendizaje automático como su fase de pruebas, se han desarrollado desde Google Colaboratory, una versión alojada en la nube de Cuaderno Jupyter, que permite ejecutar código en Python desde el navegador. Es especialmente indicado para tareas de aprendizaje automático y análisis de datos, por lo que la hace una herramienta ideal para el desarrollo de este proyecto, evitando el gran coste computacional que le supondría a la Raspberry Pi el entrenamiento de estos modelos.

– Red Neuronal
  Como primera opción, dentro del campo del aprendizaje profundo, se ha

valorado el uso de redes neuronales convolucionales por su robustez y su capacidad para trabajar con cantidades muy altas de datos.

Las librerías utilizadas son TensorFlow y Keras, dos librerías Open Source creadas para el desarrollo de algoritmos de Deep learning.

TensorFlow es una librería de computación matemática capaz de ejecutar gráficos de flujo, operaciones matemáticas cuya entrada y salida es un vector multidimensional (o tensor) de datos.

Keras es una API de alto nivel que minimiza las acciones requeridas en el desarrollo de modelos de aprendizaje profundo y es capaz de ejecutarse sobre TensorFlow. Es una herramienta muy apropiada por su facilidad de uso y nivel de abstracción.

Para la construcción del modelo se ha utilizado la técnica transfer learning [15], que consiste en utilizar redes previamente entrenadas, sustituyendo la última capa para realizar la clasificación y entrenando las últimas capas para particularizar la red al problema específico a tratar. La ventaja de esta técnica es poder utilizar modelos muy especializados en el reconocimiento de patrones, que han sido sometidos a largos entrenamientos sin la necesidad de invertir tal cantidad de tiempo en el modelo desarrollado para esta tarea.

Existen varias opciones de modelos preentrenados disponibles, siendo Densenet169 una de las más aptas para este trabajo debido a su facilidad de entrenamiento teniendo en cuenta su gran profundidad. La red cuenta con 14.3 millones de parámetros y una top-5 accuracy del 93,1 % sobre el dataset de validación de ImageNet [16].

El modelo desarrollado contiene la base de Densenet169, al que se le han añadido 4 capas densas de 216, 512, 1024 y 4 neuronas (esta última corresponde a la de clasificación), 3 capas de dropout de factor 0.5, y una capa de pooling promedio para asegurar la compatibilidad con Densenet169.

Con el fin de aumentar el dataset para ofrecer al modelo el mayor número de casos y que este pueda realizar mejores predicciones, se hace uso de la técnica Data Augmentation, que consiste en generar más datos de entrenamiento aplicando transformaciones a los datos disponibles. Con ello, la red neuronal nunca verá la misma imagen en las diferentes iteraciones del entrenamiento (epochs). Dos de las muchas formas de transformación de imagen que existen, y que se han utilizado en el desarrollo de este modelo son:

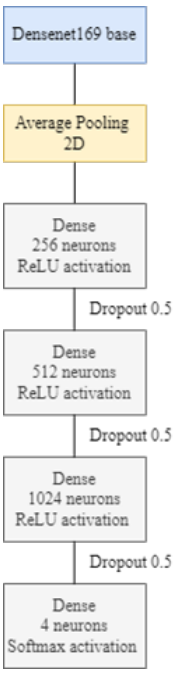


Figura 5.10: Estructura del modelo de Red Neuronal.

- Width shift range: mueve los pixeles de la imagen en dirección horizontal
- Height shift range: mueve los pixeles de la imagen en dirección vertical

El proceso de entrenamiento se efectúa en dos fases:

- Entrenamiento de las últimas capas: se comienza entrenando únicamente las últimas capas densas con el dataset FER2013 (modificado), con las capas de la red preentrenada congeladas. Esto se realiza debido a que los pesos aleatorios iniciales de las nuevas capas densas pueden afectar a los pesos de la red preentrenada durante el proceso de fine tuning. Se utiliza el optimizador SGD (Stochastic Gradient Descent) con un learning rate de 0.1.
- Fine tuning: se descongelan algunas capas de la red preentrenada y se realiza otro bucle de entrenamiento, esta vez con un learning rate menor, de 0.001.

Como resultado, se muestra en la siguiente tabla los valores finales del entrenamiento, la función de pérdida, la tasa de acierto, y las correspondientes de la validación.

– Support Vector Machines + PCA

Loss	Accuracy	Validation Loss	Validation Acuracy
0.6814	0.7552	0.7839	0.7025

Tabla 5.1: Resultados del entrenamiento de la Red Neuronal.

Otro de los métodos más utilizados para la clasificación de imágenes es el uso de Máquinas de Vectores de Soporte (SVM). Es intuitivo pensar que cuanta más información de la imagen se disponga, mejor aprenderá el modelo y más acertadas serán las predicciones. Sin embargo, no toda la información con la que se cuenta es relevante para realizar la predicción, pudiendo aumentar las probabilidades de sobreentrenamiento (overfitting). Además, la ejecución del algoritmo requeriría de mucho tiempo haciendo el sistema mucho más lento. La solución, por tanto, es reducir la información de la imagen o, en otras palabras, reducir la dimensionalidad.

Una forma de realizar esta reducción es mediante la extracción de características. La técnica empleada para ello es Principal Component Analysis (PCA), donde se eliminarán aquellas características menos relevantes conservando aquellas que aporten más valor a la predicción. Para el entrenamiento de este modelo se han fusionado las bases de datos CK+ (modificado) y la base de datos propia, ya que la primera de ellas no contiene imágenes de estado neutral.

La librería utilizada y una de las más comunes para el desarrollo de algoritmos de machine learning es Scikit Learn. Es una librería *open source* para el lenguaje Python que contiene diversas herramientas eficientes para el aprendizaje automático y modelado estadístico, como regresión, clasificación y reducción de dimensionalidad, entre otras. También, trabaja con otras librerías numéricas como NumPy.

Preprocesado de la imagen.

Se debe contemplar el hecho de que un sistema de detección de expresiones faciales puede ser, y es, sensible a variaciones en la iluminación y posición de la cara en la imagen [17].

En primer lugar, eliminar las porciones de la imagen que no aportan información de la cara hacen que el algoritmo entrene de forma más eficiente, por lo que se realiza un recorte de la imagen con las dimensiones mínimas que

muestren el rostro detectado en su totalidad. A continuación, se realiza un re-escalado de forma que todo el conjunto de datos posea el mismo tamaño de imagen. El tamaño de este escalado se ha determinado en función del tamaño de las fotos de las bases de datos utilizadas, ya que todas las imágenes de entrada al modelo deben tener las mismas dimensiones.

En cuanto a la variabilidad lumínica, se propone una normalización en la iluminación de la imagen basada en la ecualización de histograma (HE) y filtro paso bajo Gaussiano (GLPF).

La ecualización de histograma es una técnica empleada para mejorar el contraste de la imagen a través de su histograma. Se busca la existencia del mismo número de píxeles para cada nivel de gris del histograma en una imagen monocromática, es decir, un una distribución uniforme de los distintos niveles de intensidad. Para su implementación en Python, se recurre a la librería OpenCV equalizeHist() cuyo único parámetro es la imagen sobre la que se quiere aplicar el procesado.

(a) Imagen (b) Imagen ecualizada

Figura 5.11: Ecualización del histograma.

El filtro paso bajo Gaussiano es un método de procesado de imagen en el dominio de la frecuencia, usado para suavizar y reducir el ruido de la imagen. Sobre cada píxel de la imagen se aplica una ventana de un tamaño impar (kernel), de tal forma que el valor máximo aparece en el píxel central y disminuye hacia los extremos. Para su implementación, se utiliza el método GaussianBlur() de la librería OpenCV, sobre el cual es necesario definir el tamaño del kernel.

(a) Imagen (b) Imagen ecualizada (c) Imagen ecualizada y filtrada

Figura 5.12: Fases del preprocesado de la imagen.

Algoritmo PCA.

Extracción de Características.

Una vez normalizados los datos de entrada (imágenes), se extraen los valores característicos de cada imagen, también llamados *eigenvectors* o *eigenfaces*. Estos valores deben intentar caracterizar cada cara con la mayor exactitud con independencia de factores no relevantes como la iluminación, la perspectiva, etc, y deben tener capacidad de discriminación, es decir, que los valores extraídos de una cara y de otras caras, deben formar dos grupos lo más separados y compactos posible. Para obtener los *eigenfaces* se computa el algoritmo PCA sobre el dataset (los datos son tratados como no etiquetados), el cual sigue los siguientes pasos:

1. Se crean vectores cuyos valores constituyen la información de la imagen.
2. Los vectores se agrupan en una matriz Mx, cuyas columnas son las imágenes del conjunto de entrenamiento (n) y las filas (p) son las dimensiones del espacio muestral.
3. Se resta a Mx su media.
4. Se computa la técnica SVD (Singular Value Decomposition) obteniendo tres matrices, una de ellas contiene los autovectores (*eigenvectors*).

El número de componentes principales (*eigenvectors*) que se pueden calcular

es como máximo el menor de n y p (n imágenes, p dimensiones). No obstante, dado que el objetivo es reducir la dimensionalidad, es de interés escoger el menor número de componentes que resulten relevantes para la clasificación. Una forma de escoger el valor óptimo es evaluar la proporción de varianza explicada acumulada y seleccionar el número mínimo de componentes a partir del cual deja de ser sustancial el incremento.

Esta técnica se implementa desde Scikit learn mediante un escalado de los datos de 0 a 1. Posteriormente, se especifica en la función PCA la varianza explicada acumulada deseada. En este caso se ha escogido del 95 %.

Proyección de la imagen sobre los *eigenfaces*

Por último, se calcula el producto escalar de la imagen sobre la matriz (ortonormal) de los *eigenfaces*, es decir, se halla la proyección de la imagen sobre ellos. Como resultado se obtiene la representación reducida en dimensionalidad de la imagen.

Figura 5.13: *Eigenfaces* obtenidas.

Algoritmo SVM

Una vez reducidas las dimensiones del conjunto de entrenamiento, se realiza la clasificación de emociones mediante Máquinas de Vectores de Soporte (SVM).

Kernel

Para poder realizar la implementación del algoritmo en el sistema, es necesario escoger la función kernel a utilizar. Como se ha explicado en la sección X, la función kernel determina una dimensión en la que se pueda encontrar un hiperplano para separar las clases. Se han decidido implementar dos kernel, el polinomial y el RBF (Radial Basis Function) por ser dos de los más representativos. La función RBF mapea de forma no lineal las muestras, por lo que puede llegar a ser más útil en los casos donde la relación de etiquetas y atributos no sea lineal. Por otra parte, la función polinomial utiliza mayor número de hiperparámetros, por lo que el proceso de aprendizaje puede llegar a ser más complejo.

Parámetros C y γ

Se han probado varias alternativas para los parámetros C y γ, dado que para escoger el valor más indicado de cada uno de ellos es recomendable hacer una valoración cruzada, es decir, probar combinaciones con distintos parámetros y utilizar aquel que haya dado mejores resultados. La realización de la valoración cruzada, y por tanto la obtención de los parámetros óptimos, puede ejecutarse automáticamente gracias a la función GridSearchCV() de Scikit Learn, donde transmitiéndole la lista de parámetros y el modelo, realiza una validación cruzada probando y puntuando el resultado de las diferentes pruebas.

Entrenamiento

Por último, se ejecuta el entrenamiento del modelo. Este entrenamiento resulta notablemente más rápido mediante la utilización de la técnica PCA que sin ella. Al reducir las dimensiones del conjunto de datos, el clasificador cuenta con menos información que clasificar, optimizando en gran medida el proceso y permitiendo hacer más pruebas para obtener el mejor resultado.

El entrenamiento se ejecuta desde la función fit(), cuyos parámetros son el conjunto de datos de dimensiones reducidas y las etiquetas de las cuatro clases. Los parámetros que mejores resultados han proporcionado tras realizar la validación cruzada son:

C	γ	Kernel
100	0.001	RBF

Tabla 5.2: Parámetros óptimos de la SVM.

Pruebas de los Modelos

La última fase del desarrollo del detector de emociones es evaluar la ejecución de los modelos creados sobre un conjunto de datos desconocido pero clasificado. La base de datos para test se ha determinado como un porcentaje de imágenes del dataset global. Los valores comúnmente utilizados para este conjunto de datos son de alrededor de un 20-25 % del total de imágenes disponibles, siendo en la realización de este trabajo un 20 % del conjunto total.

– Test de la Red Neuronal

  Para analizar los resultados del modelo de red neuronal se ha utilizado la matriz de confusión. Es una herramienta que permite visualizar el desempeño de un algoritmo de aprendizaje supervisado, como lo son las redes neuronales. Cada columna representa el número de predicciones de cada clase, y las filas representan el número real de instancias de cada clase.

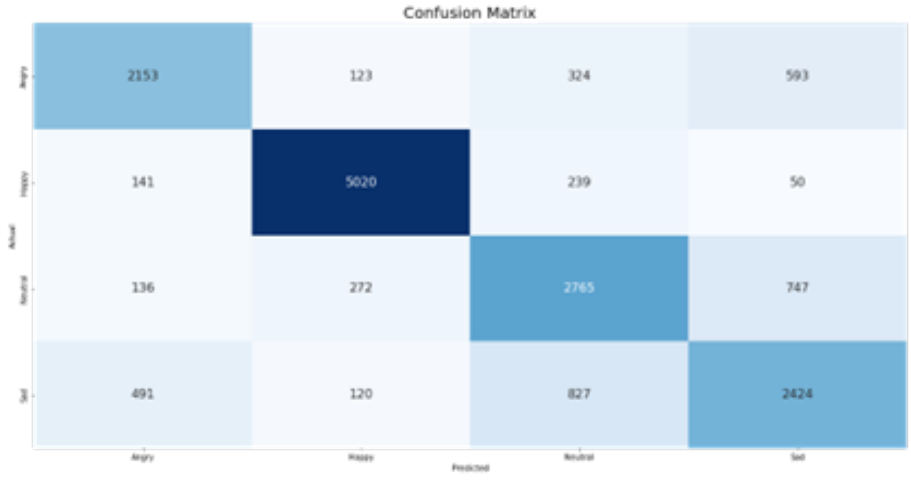


Figura 5.14: Matriz de Confusión de la Red Neuronal.

Como se puede observar, la clase que mejor clasifica el modelo es la del estado feliz. Esto no es sorprendente puesto que, analizando los datos de entrada, se observa que es el estado con más cantidad de imágenes, lo que conlleva un mejor entrenamiento de predicción de esa clase. El caso contrario sucede con el

estado de enfado, del cual se puede encontrar una cantidad considerablemente menor de ejemplos. Sobre la evaluación del dataset de test se ha obtenido un desempeño de 70 % de tasa de aciertos, siendo un resultado con posibilidades de implementación al sistema, pero indudablemente mejorable de cara a la aplicación que se está desarrollando. A raíz de este resultado, se desarrolló la otra alternativa (PCA+SVM) con el fin de mejorar la tasa de aciertos del modelo.

– Test de PCA+SVM

  La evaluación de este modelo se ha realizado de nuevo mediante una matriz de confusión.

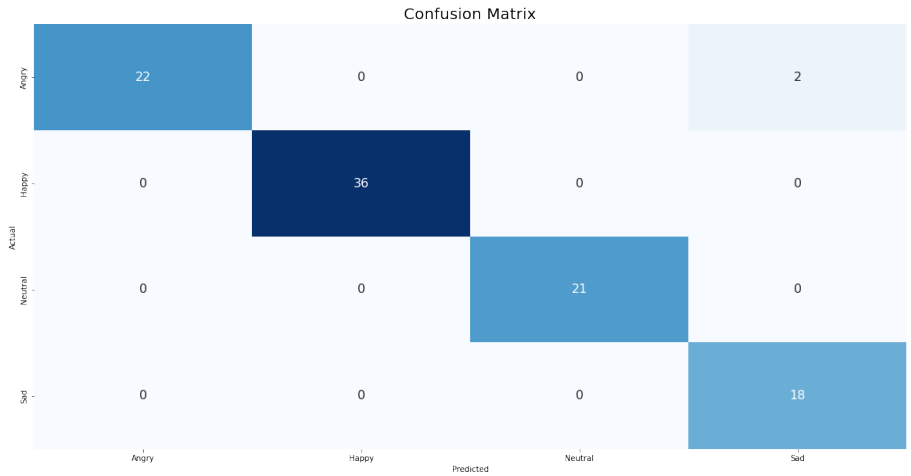


Figura 5.15: Matriz de Confusión de SVM+PCA.

Se observa un mejor desempeño con respecto a la red neuronal, con una tasa de aciertos del 97.97 % para el conjunto de datos de test. Se ha de señalar que las bases de datos no son las mismas, conteniendo este modelo un número mucho menor de muestras de test que la red neuronal. Debido a esto, no se puede llegar a realizar una comparación estricta puesto que los datos de entrada (y la cantidad de entradas) no es la misma. No obstante, de cara al objetivo de este proyecto, se han comparado ambos modelos en función de los resultados obtenidos y pruebas realizadas, y se ha determinado que el método PCA+SVM es el más indicado. Como apunte, la decisión de utilizar dos datasets distintos viene de la teoría detrás de ambos métodos de aprendizaje. En el aprendizaje profundo, a mayor cantidad de datos de entrada, mayor capacidad de generalización adquiere la red, por lo que se ha utilizado el dataset con mayor número de imágenes (FER2013). Por otro lado,

las máquinas de vectores de soporte son algoritmos computacionalmente muy complejos, por lo que un conjunto de datos demasiado extenso puede llegar a requerir una cantidad de tiempo desmesurada para efectuar el entrenamiento. Por ello, se ha utilizado una base de datos más reducida (CK+ modificada [18]).

5.5. Módulo de Estado Emocional

Una vez el sistema ha obtenido la emoción actual del usuario, se requiere una respuesta emocional por parte del robot al estímulo obtenido. En los humanos, la personalidad determina en gran medida las respuestas emocionales generadas tras la percepción de un estímulo, por lo que en este trabajo se pretende aplicar esta relación personalidad-estado emocional sobre el robot para que la interacción con el usuario resulte lo más natural posible.

Con objeto de ofrecer una interacción aproximada a la conducta natural humana, se busca un comportamiento no determinista, en la medida que el estado subsiguiente del sistema emocional se determina tanto por las emociones predecibles del proceso como por elementos aleatorios, es decir, el proceso será de naturaleza estocástica.

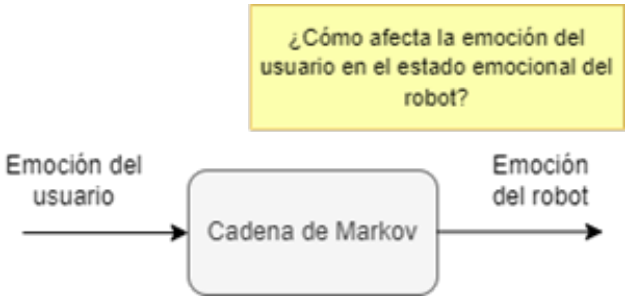


Figura 5.16: Proceso de generación de emoción.

5.5.1. Cadena de Markov

Uno de los métodos utilizados para simular estados emocionales en robots son las cadenas de Markov. Una cadena de Markov es un proceso estocástico en el que, si el estado actual X_n y los estados previos X_1, X_2, ..., X_{n-1} son conocidos, la probabilidad del estado futuro X_{n+1}:

– No depende de los estados anteriores X_1, X_2, ..., X_{n-1}

– Solamente depende del estado actual X_n

La matriz de transición, de dimensión mxm, recoge el conjunto de valores denominados probabilidades de transición, p_{ij}.

$$T = [p_{ij}] = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1m} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ p_{m1} & p_{m2} & \cdots & p_{mm} \end{bmatrix} \tag{5.1}$$

Cada fila de la matriz es una distribución de probabilidad, es decir

$$\sum_{j=1}^{m} p_{ij} = 1 \tag{5.2}$$

Para su implementación en el sistema, se han determinado cuatro estados emocionales posibles para el robot: triste, neutral, contento y excitado y cinco casos, correspondientes a 4 emociones detectadas, más el caso de no detección de la cara.

Así pues, los vectores de probabilidad para cada estado en función del estímulo percibido se recogen en cinco matrices de transición (5 estímulos posibles) de 4x4 estados cuyas distribuciones de probabilidad se han determinado bajo criterio propio en función de qué carácter se quiere modelar sobre el robot y el estímulo recibido.

Dadas las características de un proceso de Markov, se ha observado su semejanza con una máquina de estados, en la que la transición al estado siguiente depende únicamente del estado actual. Por ello, se ha implementado una máquina de estados finitos para modelar el proceso, donde la transición al estado siguiente viene determinada por el estado actual y la función random, de la librería Numpy, que escoge un valor pseudoaleatorio de una secuencia de valores dada (emociones) y sus probabilidades de ocurrencia.

Asimismo, el vector de probabilidades iniciales se determina en base a una distribución uniforme (todos poseen la misma probabilidad inicial de ocurrencia).

A continuación, se muestra un ejemplo de máquina de estados para el caso en el que el robot recibe una emoción de enfado por parte del usuario.

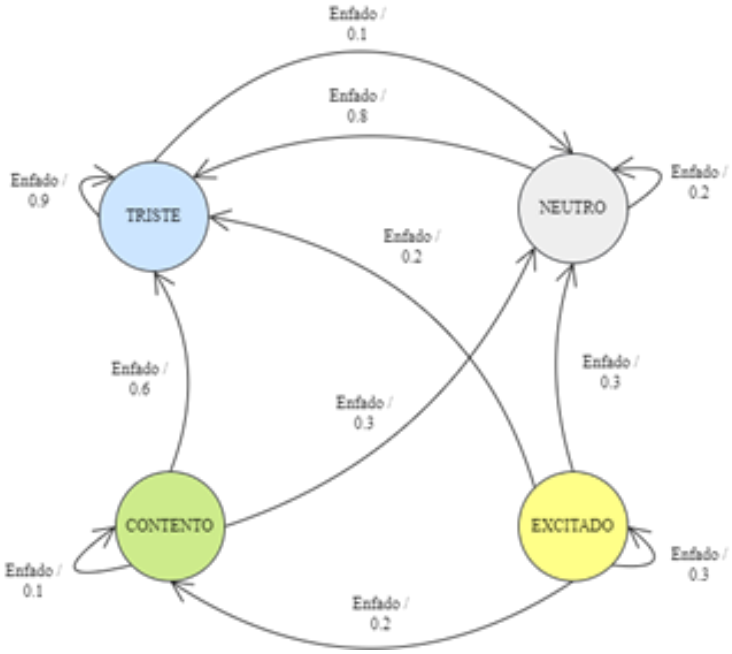


Figura 5.17: Máquina de estados para el estímulo de enfado.

5.6. Módulo de Acción

Una parte importante en lo que respecta a la interacción humano-robot, es el tipo de respuesta ofrecida por el robot ante el estímulo detectado en el usuario. Dadas las características actuales del robot sobre el que se implementa la aplicación, existe una única posibilidad de respuesta, el movimiento. Se ha relacionado cada uno de los cuatro estados emocionales del robot a un movimiento que exprese esa misma emoción en base a la observación de los comportamientos humanos.

Así pues, ante un estado de tristeza, el robot tenderá a realizar movimientos alicaídos y breves, mientras que en un estado de felicidad o excitación los movimientos serán enérgicos y de una duración superior. Además, los movimientos están programados para que la experiencia de uso sea agradable, sin movimiento constante del robot, de tal forma que algunas veces responde con movimiento y otras no.

Para ello, se ha desarrollado la clase Action, que recibirá la salida del módulo de estado emocional, una emoción, y hará una petición de movimiento haciendo una llamada a la librería de conexión con el controlador de servos.

(a) Neutral (b) Feliz (c) Triste

Figura 5.18: Ejemplos de posiciones del robot

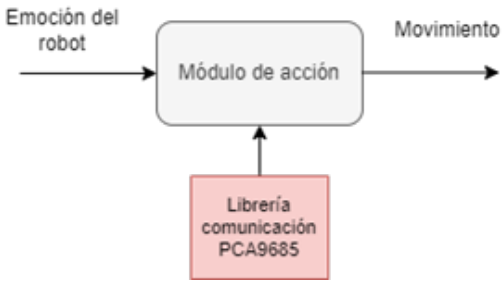


Figura 5.19: Proceso del módulo de Acción.

Capítulo 6

Integración

Como se ha detallado con anterioridad, la plataforma de integración es la Raspberry Pi Zero W, y el sistema operativo sobre el que se va a ejecutar el programa es una versión adaptada de GNU/Linux. Para establecer conexión desde el PC con la placa, se utiliza el protocolo SSH (Secure Shell) que permite controlar y modificar el servidor remoto a través de un mecanismo de autenticación, transfiriendo entradas desde cliente a host y retransmitiendo la salida de vuelta al cliente.

Los módulos del programa se han desarrollado sobre el editor de código Visual Studio Code, desde PC. Para transferir los scripts de Python a la placa se ha hecho uso de SAMBA, una implementación libre del protocolo de archivos compartidos de Microsoft Windows que posibilita que computadoras con GNU/Linux actúen como servidores o clientes en redes de Windows.

6.1. Placa de Potencia

La implementación se realiza sobre el robot LoCoQuad como se ha explicado anteriormente. Este cuenta con una placa que suministra la potencia necesaria a la placa de control, la Raspberry Pi Zero W, la cual requiere una tensión de 5V y 2.5A de corriente (aunque con la cámara conectada requerirá alrededor de 150mA más). La placa de potencia y los servomotores son alimentados por dos baterías CR126A en serie de 3.7V con capacidad de 800mAh, que proporcionarán potencia para media hora de uso del robot aproximadamente y mucho más si este se encuentra en reposo.

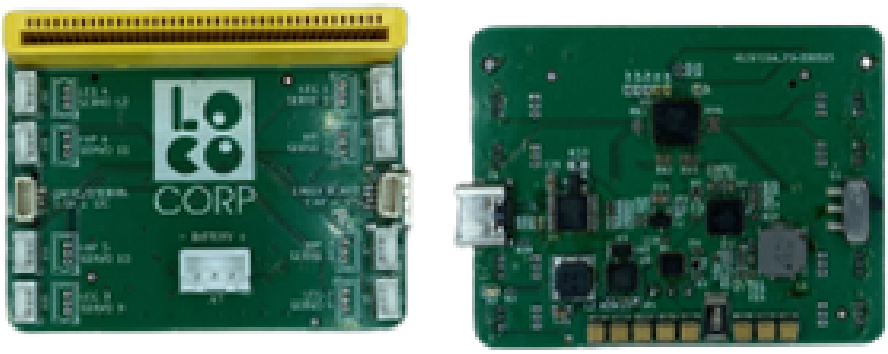


Figura 6.1: Placa de potencia.

6.2. Integración de la aplicación

La implementación de la aplicación sobre el robot sigue el esquema mostrado en la figura 6.2.

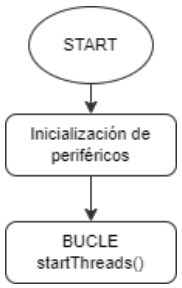


Figura 6.2: Diagrama de flujo de la implementación.

Una vez encendida la placa de potencia (mediante el interruptor integrado), se inicializan los periféricos conectados a la Raspberry Pi Zero, en este caso únicamente se hace uso de la cámara, pero se ha preferido mantener el estado de inicialización en vistas a una posible adición de otros periféricos que requieran una inicialización previa, como una pantalla, otro módulo de cámara, etcétera. A continuación, se ejecuta el bucle de procesamiento.

Para que el programa comience a ejecutarse al encender desde el interruptor, se hace una modificación del fichero *rc.local*. Este fichero es leído en el encendido (boot) de la placa y a continuación se ejecuta el programa almacenado en él.

Por último, la Raspberry Pi Zero establece conexión I2C con la placa de potencia a través de la conexión de pines I2C de ambas placas. En la imagen 6.3 se muestra esta conexión.

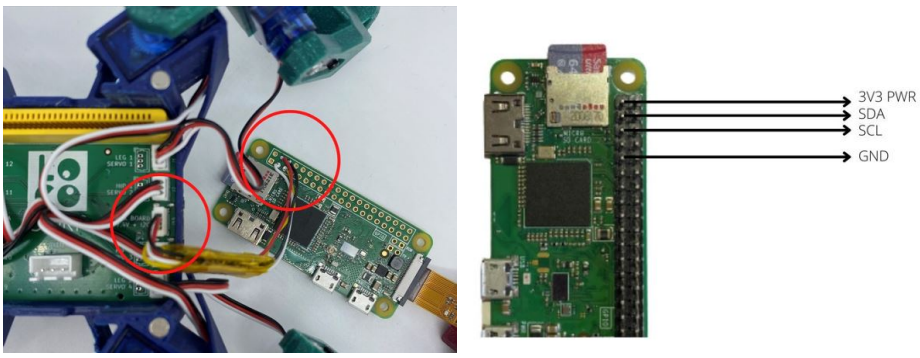


Figura 6.3: Conexión I2C con la placa de potencia.

6.2.1. Conexión con los servomotores

Una vez desarrollada la aplicación, que ofrecerá una petición de movimiento asociado a su estado emocional, el sistema debe ser capaz de responder a esta petición ejecutando el movimiento a través de los 8 servomotores integrados.

El modelo de servomotor es un microservo FS90MG, el cual irá conectado al controlador PCA9685. Este controlador permite controlar tanto LEDs como servos, ya que ambos son controlables mediante PWM (Pulse Width Modulation). En la placa de desarrollo se encuentran ya conectados los 8 servos a los pines del controlador, siendo únicamente necesario conocer la asociación pines-servos para poder controlarlos. La interfaz de comunicación es I2C, cuyos pines en el PCA9685 se encuentran también ya conectados a los del conector I2C de la placa de potencia.

Para establecer la conexión con el PCA9685 se ha desarrollado un módulo de conectividad, PCACom, que contiene la inicialización del I2C, el acceso a los canales del controlador asociados a cada servo, funciones de configuración del PWM como la frecuencia, configuración del rango de movimiento de los servos, y todos los movimientos que ejecutará el robot dispuestos como vectores de referencias de ángulos para los 8 servomotores.

Para el desarrollo de este módulo se han utilizado las librerías Adafruit-Blinka, Adafruit-PCA9685, y Adafruit-Motor.

– Adafruit-Blinka es una API de CircuitPython diseñada para ejecutarse en sistemas Linux. Esta contiene el módulo Busio (entre otros) que proporciona una interfaz de comunicación I2C con la placa.

– Adafruit-PCA9685 proporciona funciones para la configuración del controlador PCA9685 como la frecuencia, canales PWM, rango de movimiento, etcétera.

– Adafruit-Motor proporciona objetos de alto nivel para controlar motores y servos de una o más salidas PWM facilitando el proceso de envío de referencias de ángulos a cada servomotor.

Se mantendrá la frecuencia de 50Hz por defecto para el PWM siendo la estándar de los servos, y el rango de actuación también se ha establecido por defecto de 0 a 180 grados ya que cuentan con ese barrido (hay casos en los que el barrido puede ser menor).

Para el envío de referencias se hará uso de la función angle de la librería Adafruit-motor, en la que únicamente se ha de especificar el ángulo y el canal del controlador asociado al servo que se quiere controlar.

Los movimientos se han generado a partir de un repositorio de movimientos realizados por miembros de la empresa y se han decidido bajo criterio propio cuáles son los más apropiados para cada estado emocional.

Capítulo 7

Resultados

A continuación, se muestran los resultados de la detección facial y el detector de emociones para comprobar el funcionamiento individual de cada uno, y posteriormente se ejecutará una prueba global con el fin de probar el funcionamiento del sistema en su totalidad.

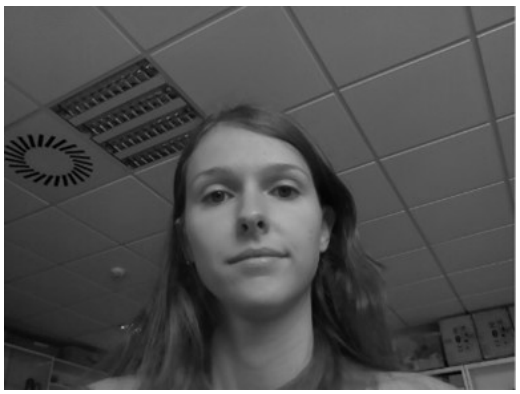
7.1. Resultados de la detección facial

Como se puede comprobar en las siguientes imágenes, el detector desarrollado mediante Haar-features muestra una robustez y una velocidad muy satisfactorias (para factor de escala 1.6 y tamaño de ventana (30x30)). Al utilizar el modelo desarrollado para caras en perspectiva frontal, este no reconoce caras posicionadas de perfil, pero esto no se ha visto como problema, puesto que la situación normal en una interacción es estar posicionado de cara al objeto de interacción.

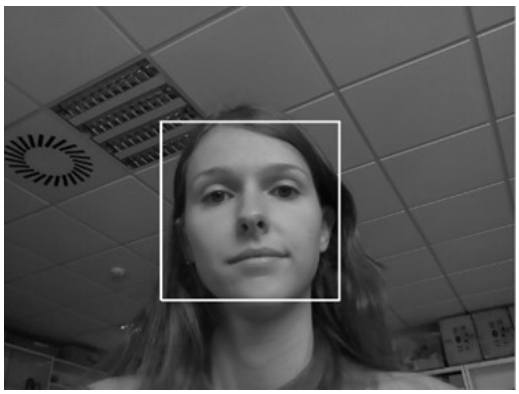
Asimismo, se muestran pruebas de cuánto afectan los parámetros de factor de escala y tamaño de ventana a la latencia del proceso, demostrando así la fuerte dependencia de la velocidad con la magnitud de dichos parámetros.

Factor de escala	Tamaño de ventana (píxeles)	Latencia media (s)
1.6	(40 x 40)	1.47
1.1	(40 x 40)	8.84
1.6	(30 x 30)	2.35
1.1	(30 x 30)	17.70

Tabla 7.1: Resultados de la modificación de parámetros del detector de caras.

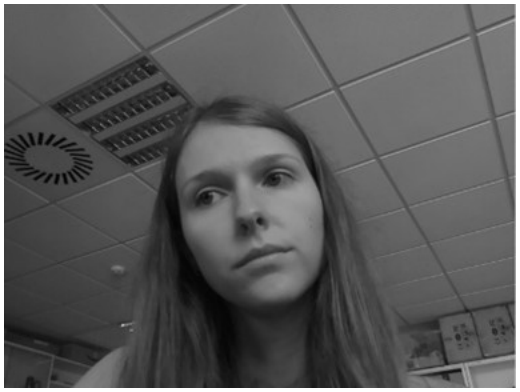


(a) Imagen sin procesar

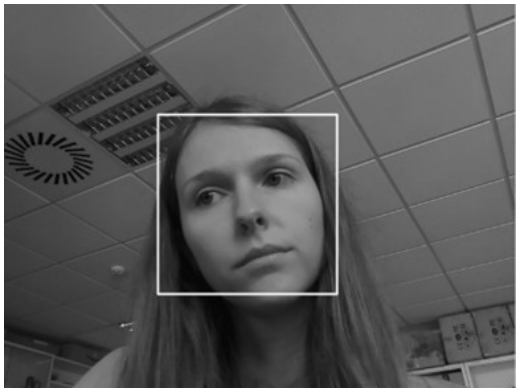


(b) Imagen de cara reconocida

Figura 7.1: Detección de cara recta con Haar-like features



(a) Imagen sin procesar



(b) Imagen de cara reconocida

Figura 7.2: Detección de cara ligeramente inclinada con Haar-like features

(a) Imagen sin procesar

(b) Imagen de cara no reconocida

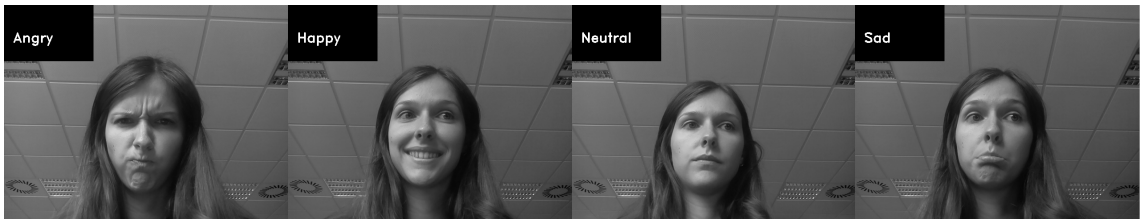
Figura 7.3: Detección de cara de perfil con Haar-like features

7.2. Resultados de la detección de emociones

Dada la alta cantidad de pruebas que se han realizado para este módulo, se muestran las más relevantes y que sirven para demostrar el correcto funcionamiento

de este. Para probar el modelo se ha ejecutado el programa guardando las imágenes tomadas y escribiendo automáticamente en ellas la emoción que ha detectado.

Como se puede observar en las imágenes, el modelo clasifica la mayoría de las expresiones faciales correctamente, y además es capaz de hacer esta clasificación con cualquier persona. Es importante añadir que, para la correcta clasificación de las emociones, las expresiones faciales han de ser bastante marcadas.



(a) Enfado (b) Felicidad (c) Neutral (d) Tristeza

Figura 7.4: Detección de emociones

Se ha observado que el correcto funcionamiento del modelo está fuertemente relacionado con la posición del robot, o más específicamente, con la diferencia de altura con respecto a la cara. Cuanto más baja se encuentre, peor rendimiento ofrece, mientras que si la cámara se acerca a la altura de la cara, acierta un número de veces considerablemente mayor. Este resultado no resulta sorprendente, puesto que en la mayoría de imágenes del dataset de entrenamiento la cámara se encuentra a la altura aproximada de la cara y como se ha explicado con anterioridad, el modelo es sensible a cambios en la posición y orientación del sujeto. Una posible solución es aumentar el dataset de entrenamiento para ofrecer una mayor variabilidad en la orientación de las caras.

	Muestras	Aciertos	Fallos	% acierto
Cámara alta	17	15	2	88.23
Cámara baja	17	10	7	58.88

Tabla 7.2: Resultados del detector de emociones.

También se ha observado una clara sensibilidad a los cambios en la iluminación. Aunque se han añadido técnicas de normalización de la imagen, el lugar desde donde procede la luz sí altera los resultados obtenidos, no siendo tan relevante la intensidad de la luz en la estancia (por supuesto garantizando un mínimo de luz).

Por otro lado, el bloque de procesamiento tarda en ejecutarse alrededor de 93 milisegundos.

7.3. Resultado del sistema global sobre LoCoQuad en tiempo real

El desempeño de la aplicación sobre el robot se puede visualizar en el vídeo recogido en el enlace de la referencia [19]. También se pueden observar otras pruebas que se han grabado del funcionamiento global del sistema en la carpeta "VIDEOS".

Como se observa en el vídeo, el robot reconoce correctamente las emociones a excepción de una.

Es importante añadir que, aunque puede dar la impresión de que una emoción se procesa dos veces, es en realidad un factor que depende del usuario. Cuando el robot termina un determinado movimiento, inmediatamente se captura otra imagen y en la mayoría de los casos el usuario mantiene la misma expresión anterior volviéndose a procesar la misma emoción. En otras palabras, cuando el usuario quiere cambiar de expresión, el sistema ya está procesando la inmediatamente anterior.

Este hecho no se ha querido corregir, pues dejar tiempo para que el usuario cambie de expresión haría el proceso mucho más lento y el objetivo es que el robot observe el estado del usuario, lentamente cambiante, no pedirle que cambie de expresión continuamente.

Un factor que sí se contempla corregir, es reducir la latencia del proceso para que se ajuste al tiempo real, pero dadas las limitaciones temporales se plantea como un posible trabajo de cara al futuro.

Tanto el código del programa (scripts de Python) como algunas pruebas documentadas que se han realizado se pueden encontrar en la carpeta a la que se llega a través del link de la referencia [20].

Capítulo 8

Conclusiones y trabajo futuro

El objetivo principal de este proyecto era el desarrollo de inteligencia emocional y personalidad en el robot LoCoQuad para la start-up LoCoCorp. Se ha llevado a cabo mediante técnicas de inteligencia artificial que han demostrado ser muy aptas para aplicaciones de reconocimiento y clasificación de imágenes, y han permitido evaluar y valorar los algoritmos que mejor se adaptan a este problema. Pese al creciente desarrollo del campo del aprendizaje profundo, se ha demostrado cómo en algunos casos el uso de otras técnicas de aprendizaje no profundo puede resultar más ventajoso, por ejemplo, en el caso de datasets poco extensos o cuando los requerimientos de tiempo son mucho más ajustados ofreciendo incluso mejores resultados que los que ofrecen las redes neuronales. Al mismo tiempo, el reconocimiento de emociones ha demostrado ser una aplicación compleja, puesto que existe una gran variabilidad en las formas de expresión facial, y supone un error asignar una expresión facial concreta a una emoción tan general como son las utilizadas en este trabajo. Además, las técnicas implementadas son muy sensibles a variaciones de iluminación y posición, haciendo que sea necesario trabajar en la generalización de los modelos. No obstante, se ha abierto una línea de investigación futura que permitirá el desarrollo y la mejora de este reconocimiento, mediante la introducción de otras emociones más concretas en un entorno más variado.

Por otro lado, dadas las limitaciones temporales presentes en el proyecto, se ha desarrollado un modelo de personalidad simple que puede resultar insuficiente teniendo en cuenta otros robots con personalidad en el mercado. No obstante, se ha conseguido mantener la naturalidad en el proceso de interacción humano-robot, aportando comportamientos no deterministas pero que han demostrado estar dentro

de una lógica emocional inspirada en la humana. De nuevo, se ha abierto otra línea de investigación dirigida a desarrollar un modelo emocional más complejo aportando nuevas emociones que otorgarán mucha más profundidad a la personalidad del LoCoQuad.

Por último, como futuro trabajo y posibles mejoras se plantea los siguiente:

- Con respecto a la detección de emociones, desarrollar un modelo más robusto que sea capaz de clasificar un mayor número de expresiones faciales. Asimismo, mejorar la calidad y aumentar el dataset empleado para poder hacer un estudio más profundo de las redes neuronales sobre esta aplicación.
- Aumentar los tipos de respuesta en la interacción, como por ejemplo la inclusión de una pantalla que muestre una expresión facial de acuerdo con las emociones del robot, o un sistema de generación de sonido, como un altavoz o zumbador con el que se pueda comunicar.
- La detección de estímulos por otras vías que no sean la visual. Es interesante aportar nuevas formas de interactuar con el robot, como por ejemplo la detección del movimiento sufrido por el robot en el caso en el que el usuario quiera levantarlo o moverlo, mediante la inclusión de un acelerómetro o brújula. O incluso una interacción verbal incluyendo un micrófono y empleando sistemas de reconocimiento por voz.
- Mejora del módulo emocional del robot. Como se ha mencionado, se ha abierto una línea de investigación para desarrollar un modelo de personalidad más complejo que permita darle mucha más profundidad al robot.
- Optimización del programa. Mejora de las latencias provocadas por procesos bloqueantes del programa, como la latencia del modelo, la ejecución de los movimientos del robot y otras funciones de procesado.

Capítulo 9

Bibliografía

[1] Sofia Petisca Filipa Correia Patrícia Alves-Oliveira Ana Paiva, Samuel Mascarenhas. Towards more humane machines: creating emotional social robots. In *New Interdisciplinary Landscapes in Morality and Emotion*, 2017.

[2] Akrivi Krouska Michalis Feidakis Cleo Sgouropoulou Iro Athina Valagkouti, Christos Troussas. Emotion recognition in human–robot interaction using the nao robot. Department of Informatics and Computer Engineering, University of West Attica, 12243 Egaleo, Greece, 2022.

[3] Chiara Filippini, David Perpetuini, Daniela Cardone, and Arcangelo Merla. Improving human–robot interaction by enhancing nao robot awareness of human facial expression. *Sensors*, 21(19), 2021.

[4] Duygun Erol Barkana Hatice Köse Fatma Göngör, Onder Tutsoy. An emotion analysis algorithm and implementation to nao humanoid robot. 2017.

[5] Craig Dunn David Britch, David Coulter. Reconocimiento de emociones percibido mediante face api, 2022. Microsoft. `https://docs.microsoft.com/es-es/xamarin/xamarin-forms/data-cloud/azure-cognitive-services/emotion-recognition`.

[6] Wallace V. Friesen Paul Ekman. *Unmasking the Face: A Guide to Recognizing Emotions from Facial Clues*, volume 4. ISHK, 2003.

[7] Gabriele Trovato Jordi Vallverdú and Lorenzo Jamone. Allocentric emotional affordances in hri: The multimodal binding. *Multimodal Technologies and Interaction*, 2018.

[8] Junichi Terao Maurizio Ficocelli and Goldie Nejat. Promoting interactions between humans and robots using robotic emotional behavior. *IEEE transactions on cibernetics*, 46(12), 2016.

[9] Joaquín Amat Rodrigo. Análisis de componentes principales (principal component analysis, pca) y t-sne, 2017. `https://www.cienciadedatos.net/documentos/35_principal_component_analysis`.

[10] Shen F. Fan H. et al. Zheng, J. An online incremental learning support vector machine for large-scale data. *Neural Comput Applic*, 2013.

[11] Cascade classifier. `https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html`.

[12] Raspberry pi zero w. `https://datasheets.raspberrypi.com/rpizero2/raspberry-pi-zero-2-w-product-brief.pdf`.

[13] Patrick Lucey, Jeffrey F. Cohn, Takeo Kanade, Jason Saragih, Zara Ambadar, and Iain Matthews. The extended cohn-kanade dataset (ck+): A complete dataset for action unit and emotion-specified expression. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops*, pages 94–101, 2010.

[14] Yousif Khaireddin and Zhuofa Chen. Facial emotion recognition: State of the art performance on fer2013. *arXiv preprint arXiv:2105.03588*, 2021.

[15] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[16] Keras applications. `https://keras.io/api/applications/`.

[17] Yu X. Ryu K.H. et al. Li, M. Face recognition technology development with gabor, pca and svm methodology under illumination normalization condition. *Cluster Comput*, 2018.

[18] Dataset de entrenamiento pca+svm. `https://drive.google.com/drive/folders/1QX19x5FMD18vTEJldz55WgFuinYqKLvc?usp=sharing`.

[19] Vídeo de la prueba del funcionamiento global. `https://drive.google.com/file/d/1WWf2f3DcHghhWxZ4rTiN1rI9GK2qrCqR/view?usp=sharing`.

[20] Carpeta con archivos del proyecto. https://drive.google.com/drive/folders/1gXOzATwZMcF--vkU0TQj-dQ8v572agag?usp=sharing.

Lista de Figuras

Lista de Tablas

Anexos

Anexos A

Movimiento con LoCoQuad

LoCoQuad es un robot cuadrúpedo de forma arácnida de 2 ejes por pata. En la figura A.1 se muestra una representación de su estructura.

Así pues, cada una de las patas del robot cuenta con un servomotor que hará la función de hombro, y otro que hará la función de rodilla, moviendo la pierna del robot en el ángulo deseado.

Para poder ejecutar el movimiento sobre el robot es necesario definir las 8 referencias de ángulos que deben adoptar cada uno de los servomotores y que posteriormente serán enviadas al PCA9685 mediante la librería Adafruit-Motor. Para ello, se ha utilizado el esquema de las figuras A.2 y A.3.

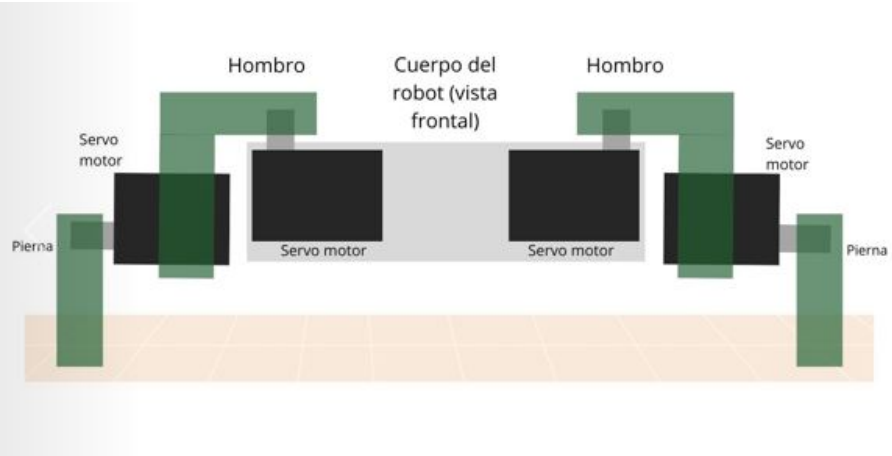


Figura A.1: Estructura de LoCoQuad.
Imagen cedida por LoCoCorp.

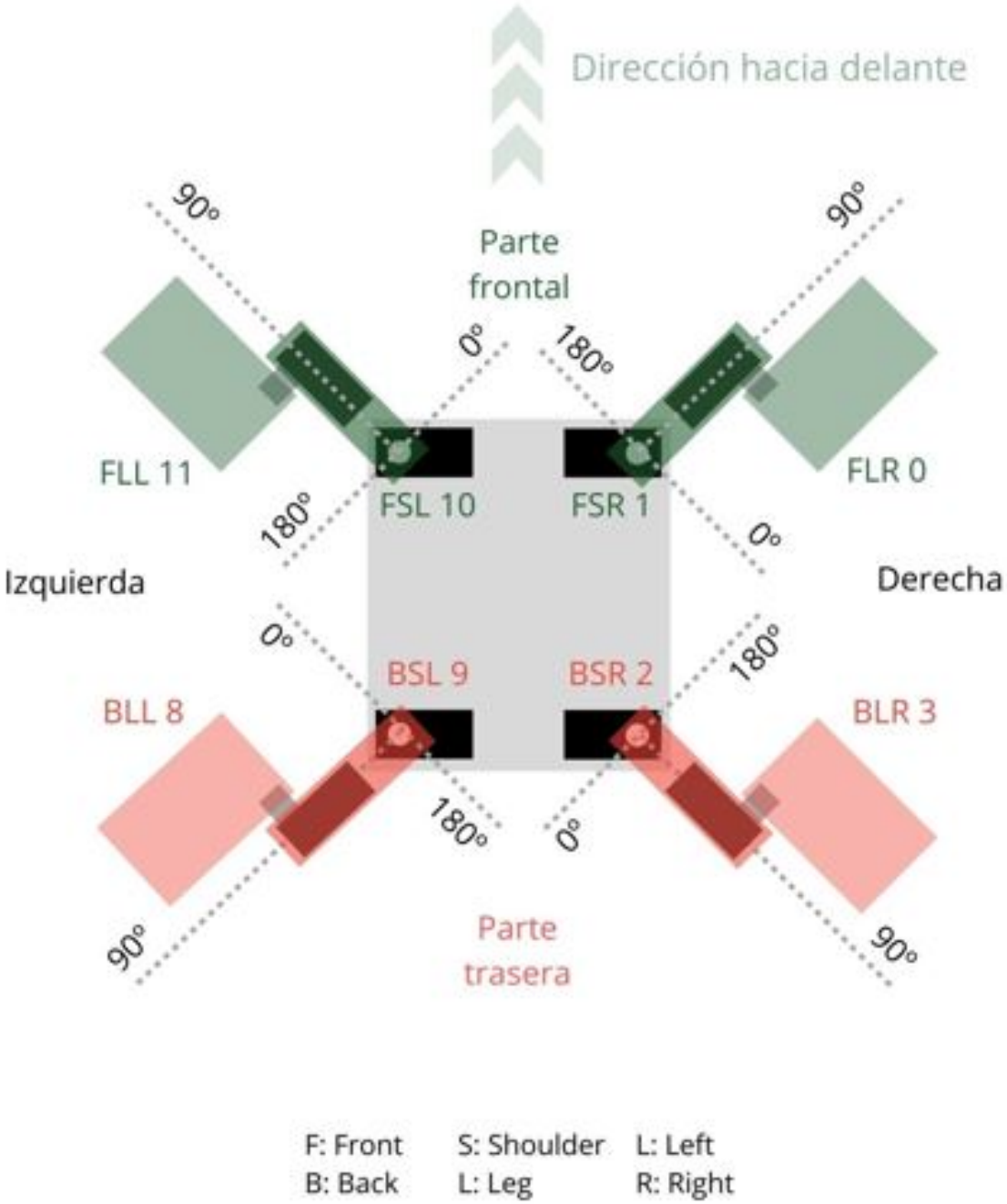


Figura A.2: Guía de referencias para hombros.
Imagen cedida por LoCoCorp.

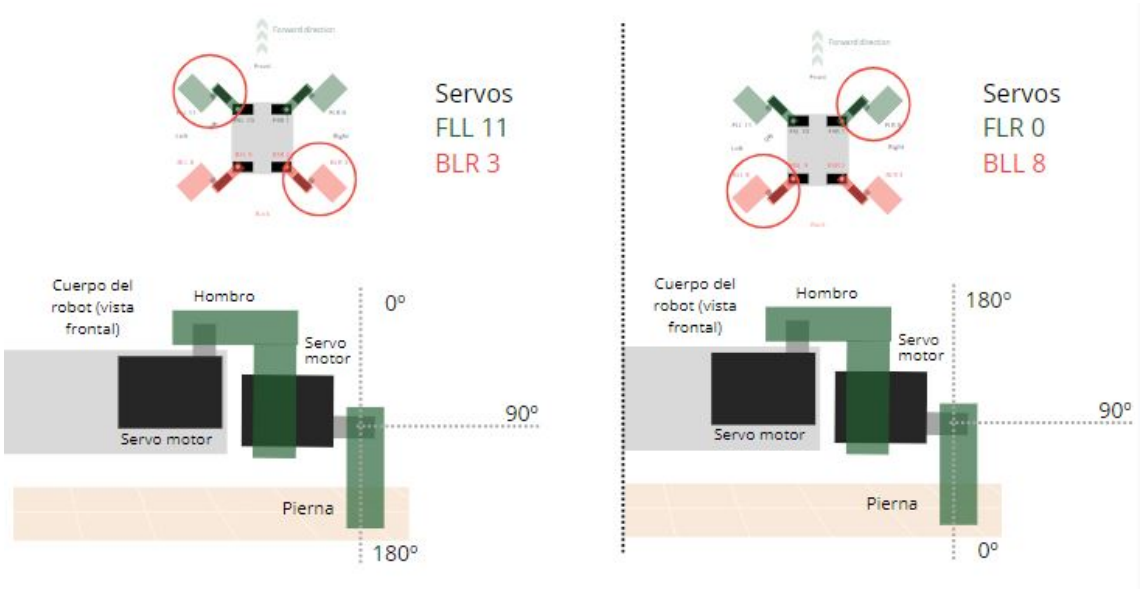


Figura A.3: Guía de referencias para piernas.
Imagen cedida por LoCoCorp.