FACULTAD DE INGENIERÍA

Carrera de Ingeniería de Sistemas Computacionales

"SOFTWARE INTELIGENTE GENERADOR DE ÓRDENES DE DESPLAZAMIENTO A PARTIR DE MOVIMIENTOS DE LA VISTA EN PERSONAS CON DISCAPACIDAD"

Tesis para optar el título profesional de:

Ingeniero de Sistemas Computacionales

Autores:

Br.Ríos Kavadoy Juan Carlos
Br. Guerrero Bello Harold Gianpierre

Asesor:

Mg. Ing. Peralta Luján José

Trujillo - Perú

2019

DEDICATORIA

A mi padre, aquel hombre que me enseñó a valorar desde lo más insignificante a lo más valioso de la vida, aquel que con sus actos me demostró su impecable responsabilidad y compromiso con nosotros sus hijos, además del rigor para cuando me desviaba del camino; mi madre por abrirme los brazos para refugiarme en el cada vez que lo necesitaba, por brindarme su cariño y amor incondicional, por demostrarme que en el mundo existen personas tan geniales como lo es ella, a mis hermanos por apoyarme a lo largo del camino académico que emprendí, por ayudarme a tomar las mejores decisiones para cuando eran necesarias, por darme ese soporte para levantarme una y otra vez para cuando las cosas no salían bien, a mis tíos, mis primos y demás familiares que con sus buenos consejos han aportado en mi formación como persona y como profesional, a mis amigos(as) y sobre todo a Richard, gran amigo desde la universidad y ahora compañero de tesis de quien he aprendido mucho gracias a sus buenos aportes.

Juan Carlos

El apoyo incondicional que recibí desde el principio de mi carrera profesional, los alientos en cada amanecida durante mi vida universitaria de mis familiares serán siempre recordados y se verá reflejado en el logro de esta nueva meta.

Gracias a Dios por permitir estar al lado de mis padres, por iluminar mi camino para no rendirme frente algún obstáculo que se me presente en mi vida personal y profesional, por siempre ayudarme en aprender de mis errores y a mejorar como persona y cristiano.

Gracias a mis padres, por confiar, creer en mí y en mis expectativas, por su apoyo y compañía en cada amanecida, trabajo y ciclo universitario donde sus oraciones y alientos eran vitales para poder obtener los resultados que hoy me permiten buscar mi titulación profesional.

Gracias a todas las personas que directamente e indirectamente me apoyaron para no bajar la guardia en mi vida universitaria, profesional y social.

Harold Gianpierre

AGRADECIMIENTO

Al Mg. Ing. José Luis Peralta por permitirnos las asesorías llevadas y su valioso aporte para el desarrollo del presente desarrollo de investigación, asimismo a nuestros docentes de la Universidad Privada del Norte

Tabla de contenidos

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

Esta investigación se basa en visión computacional y robótica, el presente trabajo se realizó con el objetivo de generar órdenes de desplazamiento usando algoritmos como suavizado, binarizado, resalte, transformada de Hough, etc, en la Universidad Privada del Norte, en el mes de marzo del presente año.

El tipo de estudio es pre-experimental; con una muestra constituida por 200 órdenes de movimiento de la vista, para la recolección de datos se aplicó la técnica del fichaje.

Las dimensiones comprendidas en la variable dependiente es el grado de inteligencia, mientras que las dimensiones comprendidas en la variable independiente órdenes visuales. El software inteligente para personas con discapacidad o falencia motora debe realizar generaciones de movimiento, durante un tiempo aceptable para generar órdenes visuales mediante la comunicación entre la cámara y el sistema generador de órdenes, es por medio de la captura de imágenes de manera constante. Los resultados obtenidos para la escena (imagen) es recortada de acuerdo a la escena del rostro, a su vez es recortada la escena de la vista, y mediante algoritmos necesarios se debe encontrar la pupila, la complejidad radica no tan solo en ubicar la pupila, si no saber en qué momento está dando una orden y en qué momento no. Finalmente, el equipo procesa la orden de movimiento (izquierda y derecha) lo realiza para ser representadas en encender/apagar leds. Con base de lo mencionado, podemos concluir que el índice de reconocimiento de órdenes de desplazamiento en 85%. Además, la cantidad de órdenes de desplazamiento generadas sean las correctas en un 84%.

.

Palabras clave: Desplazamiento, visión computacional de imágenes, pupila

ABSTRACT

This research is based on computational and robotic vision, the present work was carried out with the objective of generating displacement orders using algorithms such as smoothing, binarization, highlighting, Hough transformation, etc., at the Universidad Privada del Norte, in the month of March of this year.

The type of study was the aplicated one; with a sample constituted by 200 orders of movement of the sight, for the data collection the technique of the transfer was applied.

The dimensions included in the dependent variable is the degree of intelligence, while the dimensions comprised in the independent variable visual orders. Intelligent software for people with disabilities or motor failures must perform generations of movement, during an acceptable time to generate visual orders through communication between the camera and the order-generating system, it is through the capture of images constantly. The results obtained for the scene (image) is trimmed according to the scene of the face, in turn the scene of the view is cut out, and by means of necessary algorithms the pupil must be found, the complexity lies not only in locating the pupil , if you do not know at what time you are giving an order and at what time not. Finally, the team processes the movement order (left and right) it performs to be represented in turn on / off LEDs. Based on the aforementioned, we can conclude that the index of recognition of orders of displacement in 85%. In addition, the number of displacement orders generated are correct by 84%.

Palabras clave: Displacement, computational vision, pupil

CAPÍTULO I. INTRODUCCIÓN

1.1.Realidad problemática

La Organización Mundial de la Salud (OMS, 2011), considera que "más de mil millones de personas viven en todo el mundo con alguna forma de discapacidad; de ellas casi 200 millones experimentan dificultades considerables en su funcionamiento" y que el 3.8% de la población padece una discapacidad grave. Si es que a la letra dice (hawking, 2011) "La discapacidad no debería ser un obstáculo para el éxito".

Un aporte importante presentado en Panamá dirigida por la Ing. Anabel Broce de Tapia, es el software SOLCA, apoya a las personas con discapacidad en la utilización de las manos, para ello se agencia de la vista para poder manipular un computador.

Otro aporte relevante presentado en México por el Dr. Tomas Dávalos, diseña un software interactivo que podría ser una herramienta de apoyo en la rehabilitación de personas con algún grado de discapacidad motriz, basándose en la realidad virtual.

En nuestro país según el (INEI, 2015) existe diferentes tipos de enfermedades que impide realizar (o dar) una orden a algo, y que representa un 59.2% de la población total de discapacitados, y que les impide trasladarse de un lugar a otro, subir, etc. En la vida diaria se realiza ordenes o acciones con diferentes medios, tal como la voz, por ejemplo, para activar el celular, abrir/cerrar la puerta de la casa, etc. Y todo lo que nos ofrece el mundo de la domótica. Pero poco en el seguimiento de la vista para dar las mismas ordenes como la voz, la mano, etc.

Por tanto, queda claro que la dificultad motora (movimientos en las manos, pies, etc.) impide generar órdenes de desplazamiento y por ende tomar decisiones correctas (COHEN, 1993) . De las varias opciones para poder capturar las ordenes nacidas desde

el cerebro, una son los impulsos nerviosos según la investigación de (Wander, 2013) donde puede realizar órdenes a partir de esos impulsos, otra de las opciones es usar los movimientos oculares, tal como lo expresa (Casanova, 2010) en cómo se realiza la percepción visual. Este última opción de capturar las ordenes de la vista, se pretende realizar el presente proyecto agenciándose de 02 áreas de la Inteligencia Artificial (procesamiento digital de imágenes (K, 2001) y robótica tal como lo describe (Serrano, 2012) y (COPPIN, 2004)), encontrándose con dos problemas como realizar las ordenes por medio de la vista, el primero es poder discriminar las ordenes de las que no son, considerando que en un determinado momento puede estar visualizando un paisaje, mas no generando una orden, por tanto se desea encontrar el reconocimiento de órdenes, y segundo problema encontrado es que las órdenes de desplazamiento realizadas sean las correctas, como si es que se da la orden a la izquierda y realiza una orden diferente.

Se han considerado los siguientes estudios como antecedentes para la variable dependiente, estudio realizado por (B Wodlinger, 2014) en la investigación “Control antropomórfico de diez dimensiones del brazo en una interfaz cerebro-máquina humana: dificultades, soluciones y limitaciones”, tuvieron como objetivo generar movimiento a partir de la mente, para lograr esto tuvieron que hacer análisis de las frecuencias del cerebro para poder usar redes neuronales artificiales para aprender que acción debe realizar, teniendo los resultados de 16/18 intentos en el séptimo día de entrenamiento, llegando a la conclusión que una persona tetrapléjica puede logra mover un brazo robótico con la mente.

Estudio realizado como antecedente para la variable dependiente, estudio realizado por (CHANG TORTOLERO, 2013), cuyo título es “Detección de objetos complejos usando redes neurales y micro temblor ocular”, tuvieron como objetivo encontrar un objeto de

acuerdo a su características, usando red neuronal artificial, teniendo un resultado de una tasa de 98%, llegando a concluir que es posible descomponer el proceso de identificar a un objeto móvil y complejo en dos sub procesos neurales conectados en cascada donde el primero se controla a sí mismo y en el ínterin genera valiosa información espacio-tiempo para un segundo proceso.

Estudio realizado como antecedente para la variable dependiente, estudio realizado por (Bereciartura, 2016), cuyo título de investigación es "Aplicación a segmentación de hígado sobre imágenes de resonancia magnética multisecualcial", tuvieron como objetivo usando procesamiento digital de imágenes segmentar la imagen, de la resonancia magnética, obteniendo un valor promedio de 0.84 (ósea un 84%) de reconocimiento y segmentación, llegando a concluir que se puede reconocer características específicas del hígado en imágenes de manera autónoma.

Otro estudio realizado para la variable dependiente fue realizado por (Sevillano, 2018), cuyo título es "modelos basados en el aprendizaje automático", teniendo como objetivo aplicar diferentes modelos para poder aprender algunas acciones, para ello se consideró el modelo de la red neuronal artificial (como la red SOM), obteniendo como resultado con un 87% de efectividad. Llegando a concluir que si se puede realizar un modelo a aprendizaje basado en redes neuronales artificiales.

La presente investigación justifica porque desde el punto de vista teórico abarca conocimientos necesarios respecto al procesamiento digital de imágenes, propios de la inteligencia artificial, donde se tiene que extraer una información a partir de una escena, también tiene una justificación aplicativa puesto que se trabaja con hardware que se utilizan a las personas con falencia motora, como valorativa tenemos que puede ser escalable y de usos a mayor cantidad de personas y también tiene una justificación social,

dado que esta investigación nace de una realidad problemática para el apoyo a las personas con falencia motora.

Se encontraron las siguientes limitaciones en el desarrollo del proyecto, tal como obtener una cámara que se puede colocar al frente del rostro de la persona, otra limitación es que necesariamente debe de estar apuntando al rostro y no tener de fondo otros objetos similares al rostro, así como la tonalidad de luz.

Software inteligente

Según (Russell & Norvig, 2004) es aquel que, para cada posible secuencia de percepciones, realiza la acción que se espera que maximice su medida de rendimiento, basándose en la evidencia proporcionada por su secuencia de percepción y el conocimiento que el agente mantiene almacenado.

Por tanto, un sistema inteligente es un programa de computación que emula la manera de razonar, inferir tal como lo hace el humano o animal con su grado de inteligencia.

El sistema inteligente debe tener las siguientes características:

- **Grado de Inteligencia**: Debe cumplir con los objetivos establecidos tal como lo hace el humano de acuerdo a la tarea encomendada.
- **Sistematización**: Sus efectores deben ser los más idóneos a partir de sus sensores, donde debe capturar la mayor cantidad de información.
- **Objetivo**: Tiene como objetivo dentro de su universo de posibilidades como solución la que es racionalmente idóneo, considerando la optimización de resultados.
- **Capacidad sensorial**: Debe tener la cantidad máxima de información adquirida por medio de sus sensores, que puede ser un dispositivo hardware (cámara, sensor de calor, de presión, etc).

- **Reglas de actuación**: La representación o resultados de cada una de las acciones aprendidas, como es un motor, un parlante, una acción de computadora, etc.
- **Aprendizaje**: Bajo qué criterio va a aprender, responder frente a un conjunto de datos, por lo general se usa otras áreas de la IA, como redes neuronales artificiales, algoritmo genético, sistemas expertos, visión computacional, etc.

Robótica

Para (WorldCat, 2019) Robot Institute of America (RIA) da una descripción más precisa de los robots industriales o de la Robotica: «un robot es un manipulador reprogramable multifuncional diseñado para mover materiales, piezas o dispositivos especializados, a través de movimientos programados variables para la realización de una diversidad de tareas». Según (Gonzáles, 1989) “En suma, un robot es un manipulador reprogramable de uso general con sensores externos que pueden efectuar diferentes tareas de montaje. Con esta definición, un robot debe poseer inteligencia que se debe normalmente a los algoritmos de computador asociados con su sistema de control y sensorial”.

Según la página oficial (Arduino, 2019) en Colombia “es una plataforma de prototipos electrónica de código abierto (open-source) basada en hardware y software, flexibles y fáciles de usar, está pensado para artistas, diseñadores, como hobby y para cualquiera interesado en crear objetos o entornos interactivos”.

Para poder dar los resultados como la orden de desplazamiento como el movimiento de un motor a manejo de sensores sensibles, es necesario usar este dispositivo.

Lenguaje Python

Por otro lado se usó en lenguaje de programación Python, con la librería de visión por computadora el OpenCV, porque según (Open4U, 2019) resalta a la librería OpenCV

ayuda a los programadores a desarrollar aplicaciones 'sotisficadas' de CV (Computer Vision) rápidamente. La librería OpenCV contiene aproximadamente 500 funciones que abarcan muchas áreas de CV, incluyendo inspección de productos de fábricas, escaneo médico, seguridad, interfaces de usuario, calibración de cámaras, robótica, etc. Esta sublibrería está especializada en el reconocimiento estadístico de patrones y clustering.

Para (Rossum, 2008) Python es un lenguaje de programación poderoso y fácil de aprender. Cuenta con estructuras de datos eficientes y de alto nivel y un enfoque simple pero efectivo a la programación orientada a objetos. La elegante sintaxis de Python y su tipado dinámico, junto con su naturaleza interpretada, hacen de éste un lenguaje ideal para scripting y desarrollo rápido de aplicaciones en diversas áreas y sobre la mayoría de las plataformas.

Es un lenguaje de programación de muy alto nivel, con una sintaxis muy clara y una apuesta firme por la legibilidad del código. Sin duda es un lenguaje de programación muy versátil, fuertemente tipado, imperativo y orientado a objetos, aunque contiene también características que lo convierten en un lenguaje de paradigma funcional.

Es un lenguaje semi-interpretado. A diferencia de C, el código Python no se ejecuta directamente en la máquina destino, sino que es ejecutado por un SW intermedio (o intérprete). Sin embargo, al igual que JAVA, Python compila el código escrito en lenguaje de alto nivel para obtener un pseudo código máquina (bytecode) que es el que propiamente ejecuta el intérprete. Existen versiones del intérprete de Python para la mayor parte de las plataformas.

OpenCV

La librería grafica Opencv para (Bazaga, 2018) es una librería de visión por computador de código abierto, la librería está escrita en los lenguajes C y C++ y es compatible con

Linux, Windows y Mac OS X. Cuenta con un desarrollo activo en interfaces para Python, Ruby, Matlab y otros lenguajes.

Esta librería va a ser necesario para poder capturar y obtener la información de la escena, necesaria para poder analizarla y procesarla.

Inteligencia Artificial

Según (Norvig, 2004) la Inteligencia artificial "Existe la creencia de que el aprendizaje automático no se debe separar de la teoría de la información, que el razonamiento incierto no se debe separar de los modelos estocásticos, de que la búsqueda no se debe aislar de la optimización clásica y el control, y de que el razonamiento automático no se debe separar de los métodos formales y del análisis estático".

Por tanto, es una parte importante, poder agenciarse de la IA para encontrar soluciones de ubicación de la escena basándose en un área como es el procesamiento digital de imágenes, así como realizar el proceso de aprendizaje de la escena como lo hacen las redes neuronales artificiales.

Un sistema inteligente para (J, 1998) es un sistema inteligente es un programa de computación que reúne características y comportamientos asimilables al de la inteligencia humana o animal.

Sistema que presenta, como principal característica, su capacidad de adaptación a condiciones variables de su entorno, en pos del cumplimiento de sus objetivos. Para ello debe poseer tres capacidades básicas que es primero Razonar, para obtener conclusiones y, de ahí, tomar sus propias decisiones. Segundo Aprender, para adquirir nuevos conocimientos, a partir de sus experiencias. Y tercero Interactuar con otros Sistemas Inteligentes, mediante la comunicación y el entendimiento. Se da por supuesto que el

Sistema Inteligente posee, al menos, una mínima capacidad de memorizar, la que es un imprescindible complemento de todas estas capacidades.

De todo lo anterior surge una síntesis de características principales de los SI, que pueden ser denominadas características esenciales, a las que se agregan otras, denominadas características deseables, presentes en los sistemas biológicos, las que se detallan a continuación. Donde los Esenciales está en el Razonamiento: para obtener conclusiones y, de ahí, tomar sus propias decisiones. Aprendizaje: para adquirir nuevos conocimientos, a partir de sus experiencias. Interacción con otros Sistemas Inteligentes, mediante la comunicación y el entendimiento. Generalización: para resolver bien situaciones no presentadas durante su proceso de aprendizaje. Memoria: como imprescindible complemento de las demás capacidades. Y los deseables refleja la robustez: para poder continuar operando bien con daños parciales. Reproducción: para poder mejorar generacionalmente.

Visión computacional

Según (Gómez & Sucar, 2008) la visión computacional tiene como función principal en reconocer y localizar objetos en el ambiente mediante el procesamiento de las imágenes. Así también (Gibson, 1979) menciona que la visión computacional es recuperar de la información de los sentidos (vista) propiedades válidas del mundo exterior y construir máquinas con capacidades similares. Así refleja previo al tratamiento de la escena, para obtener información, es necesario el manejo de procesamiento digital de imágenes, para que la información sea más certera a la hora de ser extraído. La visión computacional dentro del área de la inteligencia artificial, se usa para poder identificar una información en una escena establecida.

Procesamiento digital de imágenes

Respecto al procesamiento digital de imágenes (Gutiérrez, 2003) la percepción humana "el ser humano utiliza imágenes digitales, bien para guardarlas o para modificarlas (mejorarlas), como teledetección, imágenes médicas o fotográficas, etc.".

Imágenes Digitales

Según (Gutiérrez, 2003) el sistema de percepción automática "consiste en adquirir imágenes y realizar procesos a partir de ellas. Un ejemplo es el movimiento de robots guiados por sistemas de visión o los procesos de inspección en la industria".

Para las imágenes digitales (Gutiérrez, 2003) lo considera que "Proceden del muestreo espacial y en intensidad de la imagen óptica. Están formadas por una matriz de elementos (píxeles). El píxel, picture element, es el valor de color o intensidad asociado a cada elemento de la matriz; tiene coordenadas espaciales. Puede tomar valores dentro del rango [0, 255]".

Según (García, 2008) clasifica las imágenes digitales en 2 grupos

- **Imágenes vectoriales**

  La información de cada uno de los puntos se recoge en forma de ecuación matemática que lo relaciona con el resto de los puntos que forman la imagen. Ofrece la gran ventaja de que la calidad de la imagen no varía al modificar el tamaño, ya que la información de cada punto no es absoluta sino relativa al resto de la imagen. Además, debido a su definición matemática, apenas ocupa espacio, ya que una fórmula que represente su forma es suficiente para representar todos los puntos que la componen. Es el tipo adecuado para el diseño de líneas, polígonos y figuras.

- **Imágenes Raster o mapa de bits**

  Trabaja en función de una matriz de dimensiones mxnx3, con la ventaja de que se puede extender o escalar. Utilizan una cuadrícula rectangular de elementos de

imagen (píxeles) para representar las imágenes en rangos de 0 a 255. A cada píxel se le asigna una ubicación y un valor de color específico. La ventaja que presenta este formato es la posibilidad de recoger una amplia gama tonal, por lo que es el tipo adecuado para representar imágenes captadas de la realidad. La desventaja es que se pierde la calidad si es que se escala demasiado.

Generador de ordenes

Generador de órdenes para (BienSalud, 2015) el funcionamiento adecuado del organismo humano no depende únicamente de las órdenes enviadas por el cerebro a través del sistema nervioso. Para que el sistema orgánico realice sus funciones de manera idónea, se necesita el flujo constante de información tanto transmitida como recibida. El canal de comunicación entre los órganos del cuerpo humano es la herramienta sustancial de evolución y desarrollo de la vida diaria.

Son mensajes del cerebro al resto del cuerpo, que va en función del conjunto de acciones necesarias.

El resultado es generar ordenes visuales, para ello (Querelle, 2018) menciona que el movimiento tiene que ver con la sensación de desplazamiento rápido, como ver una moto o un auto a gran velocidad, pero es provocado por un efecto invisible, que actúa sobre los cuerpos, llamado fuerza.

Fuerza y movimiento son dos eventos físicos que están ligados. Pero, aunque la fuerza puede manifestarse sola, el movimiento no es posible sin el concurso de una fuerza. Como la fuerza es invisible, alguno de los efectos producidos por esta, también son invisibles.

1.2.Formulación del problema

¿De qué manera influye el software inteligente en la generación de órdenes de desplazamiento a partir de la vista de personas con discapacidad?

1.3.Objetivos

1.3.1. Objetivos generales

Determinar la influencia del software inteligente en la generación de órdenes de desplazamiento a partir de los movimientos de la vista de personas con discapacidad basado en visión computacional.

1.3.2. Objetivos específicos

- Determinar el índice de reconocimiento de órdenes de desplazamiento a partir de los movimientos de la vista de personas con discapacidad.
- Determinar la cantidad de órdenes de desplazamiento generadas con los movimientos de la vista de personas con discapacidad que sean correctas.

1.4.Hipótesis

1.4.1. Hipótesis general

El software inteligente influye de manera significativa para generar órdenes de desplazamiento a partir de la vista de personas con discapacidad.

1.4.2. Hipótesis especifica

- El software inteligente incrementa en el índice de reconocimiento de órdenes de desplazamiento a partir de los movimientos de la vista de personas con discapacidad.
- El software inteligente incrementa en la cantidad de órdenes realizadas correctas para el desplazamiento a partir de movimientos de la vista de personas con discapacidad que sean correctas.

CAPÍTULO II. METODOLOGÍA

2.1.Tipo de investigación

Pre-Experimental

Según (Sampiere., 2011) manifiesta que "consiste en administrar un estímulo o tratamiento a un grupo y después aplicar una medición de una o más variables para observar cual es el nivel del grupo y después aplicar una medición de una o más variables para observar cual es el nivel del grupo en estas variables". No hay manipulación de la variable independiente

Por lo tanto, la presente investigación es de tipo pre-experimental, porque se aplicará un estímulo (software inteligente) para la resolución del problema.

G: O1 ➔ X ➔ O2

Donde:

G: Personas que se van a evaluar las ordenes visuales.

O1: Medición pre-experimental generar ordenes de desplazamiento

X: Sistema inteligente

O2: Medición post-experimental generar ordenes de desplazamiento

2.2. Población y muestra (Materiales, instrumentos y métodos)

Según (Lenin, 2014) manifiesta que: "Una población es cualquier grupo de elementos; los elementos son las unidades individuales que componen la población. Mientras que la población se refiere a un grupo finito, el universo se refiere a sucesos que no tienen límite, infinitos." (p.237).

Según (HERNÁNDEZ, 2014), define que: "Un estudio no será mejor por tener una población más grande; la calidad de un trabajo investigativo estriba en delimitar claramente la población con base en el planteamiento del problema." (p.174).

La población está conformada por 100 órdenes visuales realizadas, como pruebas las órdenes son izquierda y derecha. Por tanto, el objeto de estudio está dado por las órdenes visuales. Para obtener estas órdenes visuales se considerar 5 intentos de orden de cada lado, realizado a 10 personas, por tanto 10 intentos x 2 lados x 10 personas, son 200 órdenes de desplazamiento, el cual será sometida a evaluación de la certeza.

2.3. Técnicas e instrumentos de recolección y análisis de datos

Recolección de datos:

Técnica del fichaje: Según (GUILLERMO, 2017) "Es una técnica auxiliar de todas las demás técnicas; consiste en registrar los datos que se van obteniendo en los instrumentos llamados fichas, los cuales debidamente elaborados y ordenados contiene la mayor parte de la información que se recopila en una investigación por lo cual constituye un valioso instrumento auxiliar en esa tarea".

Ficha de registro: Sera una ficha de evaluación de ordenes visuales de desplazamiento generadas y valoradas según su resultado.

Validez: Según sostiene que "la validez, en términos generales, se refiere al grado en que un instrumento mide realmente la variable que pretende medir".

Análisis de datos:

Según (Cherry, 2017) la investigación aplicada se refiere al estudio científico que busca resolver problemas prácticos. Esta se utiliza para encontrar soluciones a problemas cotidianos, curar enfermedades y desarrollar tecnologías innovadoras.

2.4. Procedimiento

Para la presente investigación se tendrán en cuenta 2 técnicas: Observación y Ficha de registro, considerado para la adquisición de dato, tal como se muestra en la tabla de datos (ver anexo1), donde se observa que el observador (el presente investigador) se obtuvo los datos de los intentos de generación de ordenes visuales, identificando cuál de ellos han sido realmente ordenes visuales.

2.4.1. Recolección de datos antes de usar el software inteligente

Indicador 1: Índice de reconocimiento de órdenes de desplazamiento

Recolección de datos antes de usar el software inteligente para el indicador índice de reconocimiento de órdenes de desplazamiento (ver tabla 1).

Caso de prueba	Intentos	Total Intentos	OVGC	Índice de reconocimiento	Índice de reconocimiento (%)
Persona1	Intentos izquierda	10	6	0.60	60%
	Intentos derecha	10	4	0.40	40%
Persona2	Intentos izquierda	10	6	0.60	60%
	Intentos derecha	10	7	0.70	70%
Persona3	Intentos izquierda	10	6	0.60	60%
	Intentos derecha	10	4	0.40	40%
Persona4	Intentos izquierda	10	7	0.70	70%
	Intentos derecha	10	7	0.70	70%
Persona5	Intentos izquierda	10	8	0.80	80%
	Intentos derecha	10	6	0.60	60%
Persona6	Intentos izquierda	10	7	0.70	70%
	Intentos derecha	10	4	0.40	40%
Persona7	Intentos izquierda	10	7	0.70	70%
	Intentos derecha	10	4	0.40	40%
Persona8	Intentos izquierda	10	7	0.70	70%

	Intentos derecha	10	8	0.85	85%
Persona9	Intentos izquierda	10	8	0.80	80%
	Intentos derecha	10	4	0.40	40%
Persona10	Intentos izquierda	10	4	0.40	40%
	Intentos derecha	10	6	0.60	60%
	TOTAL(TO)	200	48		

Tabla 1. Datos Pretest del indicador índice de reconocimiento de órdenes de desplazamiento
Fuente: Elaboración propia

Indicador 2: órdenes realizadas correctas

Recolección de datos antes de usar el software inteligente para el indicador órdenes realizadas correctas (ver tabla 2).

Medición	Intentos	Total Intentos	OC	Ordenes Visuales Correctas	Ordenes Visuales Correctas(%)
Persona1	Intentos izquierda	6	2	0.3	30%
	Intentos derecha	4	1	0.3	30%
Persona2	Intentos izquierda	6	3	0.5	50%
	Intentos derecha	7	2	0.3	30%
Persona3	Intentos izquierda	6	5	0.8	80%
	Intentos derecha	4	1	0.3	30%
Persona4	Intentos izquierda	7	2	0.3	30%
	Intentos derecha	7	3	0.4	40%
Persona5	Intentos izquierda	8	3	0.4	40%
	Intentos derecha	6	3	0.5	50%
Persona6	Intentos izquierda	7	4	0.6	60%
	Intentos derecha	4	0	0.0	0%
Persona7	Intentos izquierda	7	2	0.3	30%
	Intentos derecha	4	0	0.0	0%

Persona8	Intentos izquierda	7	0	0.0	0%
	Intentos derecha	8	3	0.4	40%
Persona9	Intentos izquierda	8	2	0.3	30%
	Intentos derecha	4	0	0.0	0%
Persona10	Intentos izquierda	4	0	0.0	0%
	Intentos derecha	6	2	0.3	30%
	TOTAL(OTR)	120	38		

Tabla 2. Datos pretest del indicador cantidad de ordenes reconocidas correctas
Fuente: Elaboración propia

2.4.2. Desarrollo de la metodología del software inteligente

En esta sección se describe las fases para desarrollar un sistema generador de ordenes visual, usando visión computacional y procesamiento digital de imágenes con el software Python y hardware Arduino. Haciendo uso de algoritmos y funciones de la biblioteca de OpenCV. Este proyecto de investigación se desarrolla bajo el ciclo de vida en cascada según indica el autor: (Sommerville, 1989) “, a través las siguientes fases:

2.4.2.1.Análisis.

Para identificar las ordenes visuales vamos a analizar el movimiento de la pupila y cuando son ordenes, para ello se usa procesamiento digital de imágenes, tanto para ubicar los ojos en el rostro, así como generar ordenes visuales.

Figura 1: Ubicación de la vista
Fuente: Elaboración propia

Definición de requerimientos

- ✓ Funcionales
  - Ubicar la posición del rostro
  - Ubicar dentro del rostro a los ojos.
  - Ubicar dentro de los ojos donde está la pupila.
  - Fijar una posición de la pupila dentro de la posición central del ojo
  - Escuchar órdenes de izquierda(i) y derecha (d)
  - Encender el led definido como izquierda y derecha.
  - Procesar la escena (mejorar la imagen)
  - Identificar órdenes.
  - Encender led.
- ✓ Requerimientos no funcionales
  - Se usará el lenguaje de programación Python con OpenCV para la parte de escritorio y el procesamiento digital de imágenes respectivamente.
  - El tiempo de generar orden no debe exceder 2 segundo.
  - Las interfaces deben ser amigables y sencillas de usar.

2.4.2.2.Diseño

Para determinar los métodos y técnicas en las distintas fases del procesamiento de imagen para la generación de ordenes visuales, se asignó pesos en una escala de 1 al 5, en función a sus características:

Malo	Regular	Medio	Bueno	Excelente
1	2	3	4	5

Tabla 3: Tabla de datos para la elección del método para el análisis de la generación de ordenes visuales
Fuente: Elaboración propia

a) Captura de la escena

Para capturar la escena se ha contado con una cámara de 18 Megapixeles.

Figura 2: Proceso de captura de la escena del rostro
Fuente: Elaboración propia

Marca	Modelo	Puerto de entrada	Video
Logitech	C525 HD Web Cam	USB 2.0	720p

Tabla 4: Características de la cámara para la captura de la escena del rostro
Fuente: Elaboración propia

b) Procesamiento

Técnica para eliminar ruido

Para lograr los resultados, necesariamente se ha tenido que aplicar unos filtros, como el de suavizar la imagen:

Características	Promedio	Gaussiano	Mediana
Reducción de ruido	3	3	4
Suavizado	3	4	2
Costo computacional	3	3	3
TOTAL	9	10	9

Tabla 5: Selección de técnica para eliminar ruido
Fuente: Elaboración propia

La técnica adecuada para el proceso de mejora de la imagen, previo a la identificación de la pupila es el filtro gaussiano.

Aplicando para el proyecto el código fuente en lenguaje Python 3.7.5 que realiza el filtro gaussiano, usando la librería OpenCV:

```
imgGrayencirculado = cv2.gaussianBlur(imgGrayencirculado,(5,5),0)
```

Técnica para encontrar la pupila

Para encontrar la pupila dentro de la escena de la vista, se ha utilizado la transformada de Hough, ideal para encontrar un circulo dentro de la escena.

Características	Hough	Blood
Obtención de posición, área, etc.	5	1
Completa el circulo si no está completo.	4	2
Costo computacional	3	3
TOTAL	12	6

Tabla 6: Técnica para encontrar la pupila
Fuente: Elaboración propia

La técnica adecuada para el proceso de encontrar la pupila en una escena es el filtro Hough.

Aplicando para el proyecto el código fuente en lenguaje Python 3.7.5 que realiza el filtro Hough, usando la librería OpenCV:

```
circles=cv2.HoughCircles(imgGrayencirculado,cv2.HOUGH_GRADIENT,1,20,param1=50,param2=30,minRadius=7,maxRadius=80)
```

Técnica para realizar morfología a la escena

Para resaltar la pupila dentro de la escena de la vista y discriminando el resto de datos de la escena, lo ideal es que corte y remarque la pupila. Para ello se usó la función morfológica erode y dilate con 4 iteraciones cada uno. Así se muestra en el código fuente en Python:

```
histo = cv2.erode(histo,kernel,iterations = 4)
histo = cv2.dilate(histo,kernel,iterations = 4)
```

Diagrama de flujo del procesamiento para genera la orden

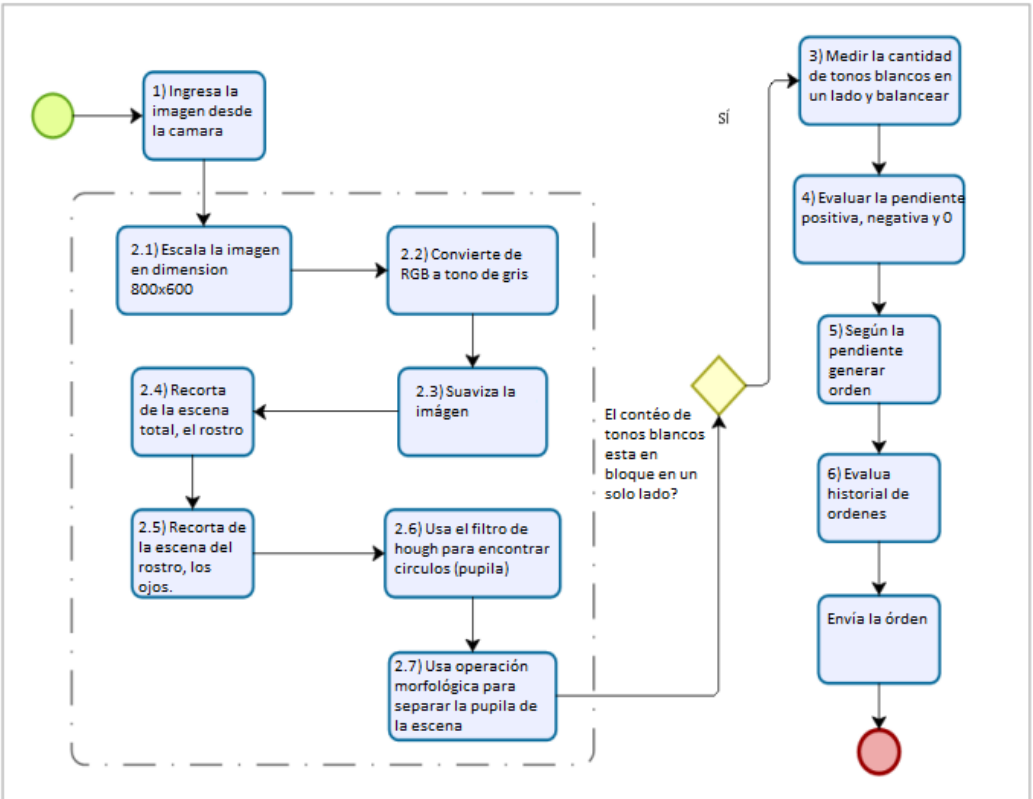

Figura 3: Diagrama de flujo del proceso de generar orden
Fuente: Elaboración propia

1. Captura la escena con la cámara enfocada al rostro de la persona. Para ello se usa también el lenguaje Python para obtener el resultado:

ret, img = cap.read()

2. Ubicar y resaltar la pupila

2.1. Escalar a imagen para poder tener una escena definida, esto porque se desee cambiar la cámara y tenga diferente resolución.

2.2. Para poder realizar el procesamiento de imágenes, es necesario trabajar con una matriz, que represente la escena, por tanto, se convierte a tono de gris.

2.3. Para eliminar ciertos ruidos presente, como 'sal y pimienta' es necesario usar unos filtros de suavizamiento como es el gaussiano.

imgGrayencirculado = cv2.gaussianBlur(imgGrayencirculado,(5,5),0)

2.4. De la escena total, es necesario recortar la escena en el cual esta solo el rostro. Para eso se usa el lenguaje Python con el siguiente código:

faces = face_cascade.detectMultiScale(gray)#, 1.3,5,minSize=(20, 20))

2.5. De la escena del rostro, es necesario encontrar los ojos, para ello se usa el siguiente código Python:

```
eyes = eye_cascade.detectMultiScale(roi_gray,1.1,1)
```

2.6. En los ojos, es necesario encontrar la pupila, para ello se usa el filtro de Hogh, ya que lo encuentra de manera encirculada. Para ello se usa el siguiente código en Python:

```
circles=cv2.HoughCircles(imgGrayencirculado,cv2.HOUGH_GRADIENT,1,
20,param1=50,param2=30,minRadius=7,maxRadius=80)
```

2.7. Una vez encontrada la pupila, es necesario separarlo, para ello es necesario realizar operaciones morfológicas, para que a la vez los datos cercanos a la pupila lo tengan en color blanco. Para ello se considera el siguiente código fuente en Python:

```
histo = cv2.erode(histo,kernel,iterations = 4)
histo = cv2.dilate(histo,kernel,iterations = 4)
```

3. Medir la cantidad de tonos blancos que hay en cada lado, es necesario para poder saber hacia dónde está mirando. Esto para tener el balance de ojos y poder generar una línea y por ende pendiente.

```
m = pendiente(suma)
```

4. Una vez que se tiene la línea generada por el balance de tonos de grises, es necesario saber si la pendiente es positiva, negativa o es paralela al eje X. y se usa el siguiente código en Python:

```
m = valorpendiente(histo)
```

5. Para los valores de la pendiente (positivo, negativo, 0) hay que generar orden, como izquierda ‘i’, derecha, ‘d’ y espera ‘e’ respectivamente. para ello se usa el siguiente código en Python:

```
if M==0 and np.median(historiaM)<0:
  estadoActual ="Derecha"
if M>0.5:
  estadoActual ="Centro"
  arduino.write(b'e')
 if estadoActual=="Izquierda":
    arduino.write(b'i')
```

c) Segmentación

Permite resaltar la escena en tono de gris, necesario para poder analizar la posición de la pupila. Para ello se utilizó el siguiente código fuente en Python:

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

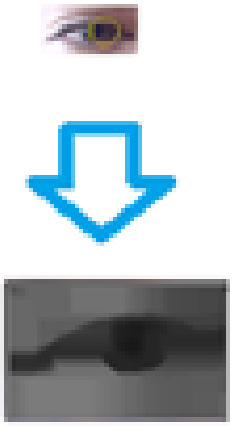

Figura 4: Gráfico de RGB a tono de gris
Fuente: Elaboración propia

d) Descripción

Técnica de extracción de características.

Esto es necesario para identificar cual es la técnica necesaria para la extracción de características particulares en una escena, y por medio de ello lograr ubicar el objeto en cuestión y ayudar a tomar decisión, para ello se apoya del modelo de hough para poder encontrar la pupila.

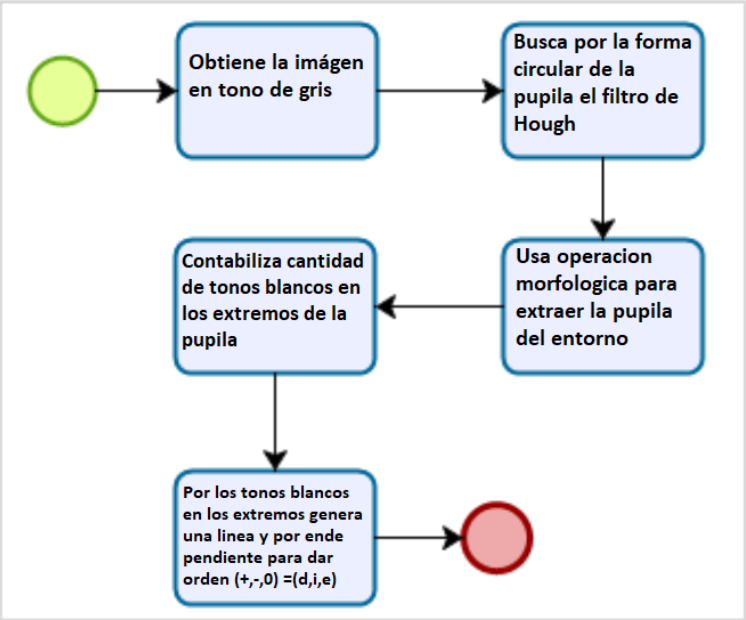

Figura 5: Diagrama de flujo de extracción de características
Fuente: Elaboración propia

e) Reconocimiento de orden visual

Considerando los valores de las ordenes de las pendientes generadas por la línea de los tonos de color blanco, para considerar una orden es necesaria tener una estadística de las mismas.

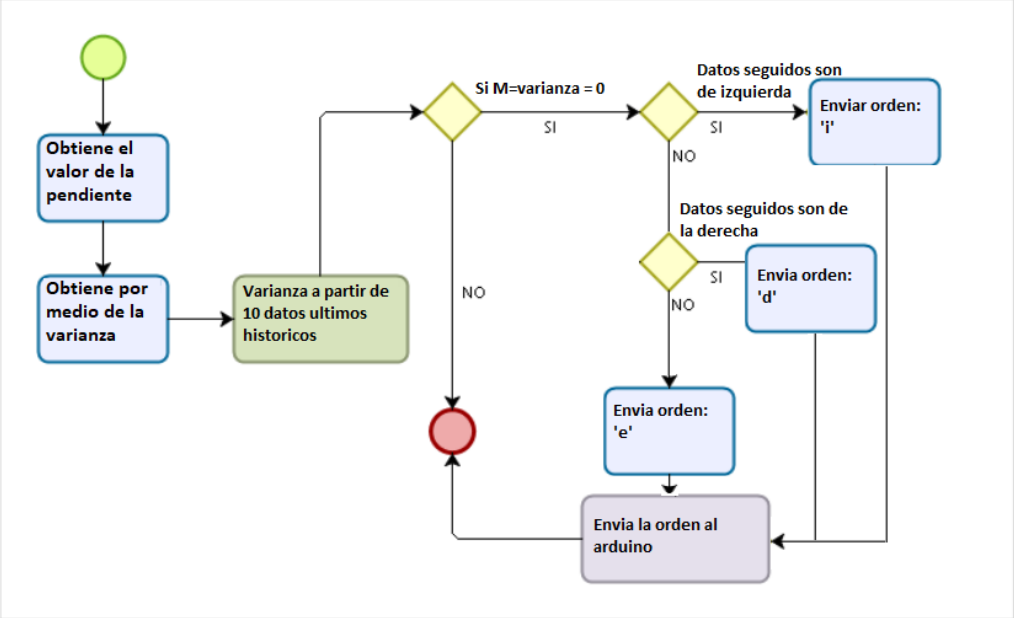

Figura 6: Diagrama de la identificación de la orden visual
Fuente: Elaboración propia

Clasificación de ordenes usando procesamiento digital de imágenes:

Para reconocer la orden visual es necesario considerar la cantidad de datos de tono de grises que están en blanco para poder identificarlos, posteriormente usar el método de la varianza para la cantidad de órdenes.

$$S_X^2 = \frac{\sum_{i=1}^{N}(X_i - \overline{x})^2}{N-1}$$

Siendo X1, X2,…,Xn, el conjunto de N datos, y x es la media de dichos datos.

Se analiza si es que S es 0, por tanto, no hay varianza, esto implica que existe una intensión de dar una orden, pero si los datos son diferentes de 0 implica que tiene diferentes tipos de órdenes, por tanto, no genera orden establecida; eso puede ser porque está mirando a un lado y luego al otro, etc.

Figura 7: Arquitectura para la identificación de la orden visual
Fuente: Elaboración propia

2.4.2.3.Implementación

Para la implementación del sistema basado en procesamiento digital de imágenes para la generación de ordenes visuales, se ha considerado lo siguiente:

a) Tecnologías empleadas

- Python 3.7.5.
- OpenCV 3.4.2

b) Funciones principales

Funciones	Interpretación
def pendiente(y): x = np.arange(len(y))	Permite encontrar la pendiente a partir de un conjunto de 'y' datos

if isnan(np.corrcoef(x, y)[0,1]):
return 0
else:
return np.corrcoef(x, y)[0,1]	

def varianza(y): return np.std(y)**0.5	Permite encontrar la varianza de ordenes visuales de un conjunto de 'y' datos.
Circles = cv2.HoughCircles (imgGrayencirculado, cv2.HOUGH_GRADIENT, 1,20, param1=50, param2=30, minRadius=7, maxRadius=80)	Permite encontrar la pupila de manera circular.

Tabla 7: Funciones del sistema
Fuente: Elaboración propia

2.4.2.4.Pruebas

Las pruebas han sido realizadas por medio de la caja negra para medir los resultados de la aplicación, detallando a continuación con un ejemplo de prueba realizada:

a) Caso evaluado a persona 1:

Medición	Intento realizado	Acción correcta
Persona1	Intento izquierda	Intento izquierda

Tabla 8: Evaluación de la generación de movimientos a la izquierda
Fuente: Elaboración propia

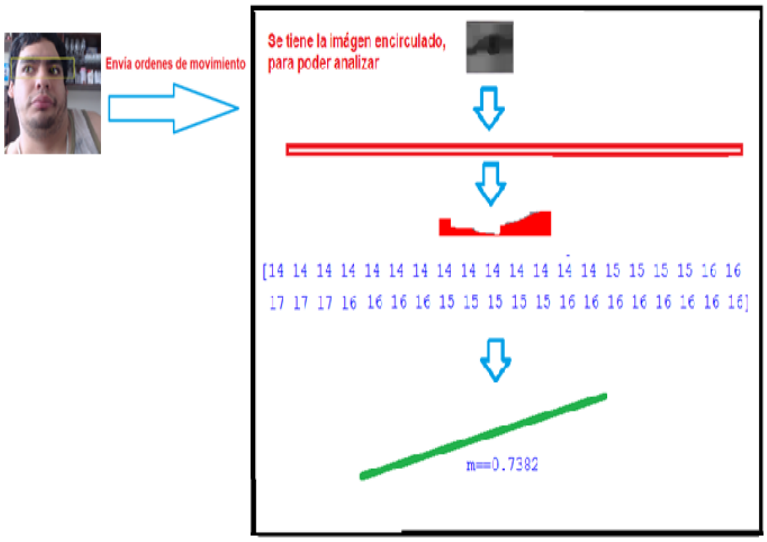

Figura 8: Resultado obtenido al realizar la prueba de generar orden visual hacia la izquierda
Fuente: Elaboración propia

b) Caso evaluado a persona 2:

Medición	Intento realizado	Acción correcta
Persona2	Intento derecha	Intento derecha

Tabla 9: Evaluación de la generación de movimientos a la derecha
Fuente: Elaboración propia

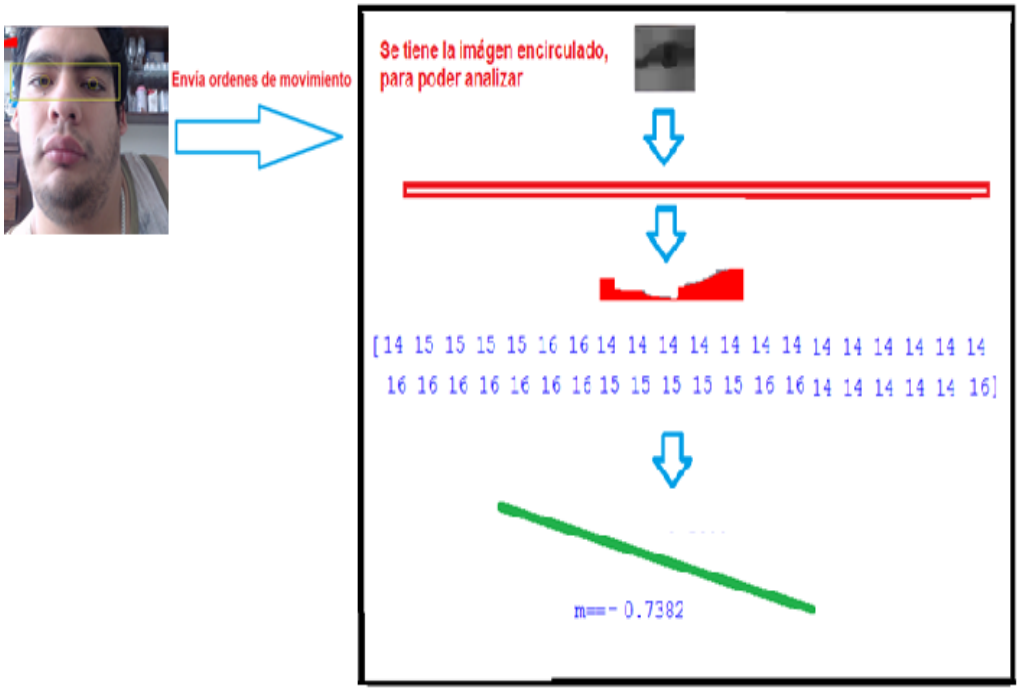

Figura 9: Resultado obtenido al realizar la prueba de generar orden visual hacia la derecha
Fuente: Elaboración propia

2.4.3. Recolección de datos después de usar el software inteligente

Indicador 1: Índice de reconocimiento de órdenes de desplazamiento

Recolección de datos después de usar el software inteligente para el indicador índice de reconocimiento de órdenes de desplazamiento (ver tabla 10).

Caso de prueba	Intentos	Total Intentos	OVGC	Índice de reconocimiento	Índice de reconocimiento (%)
Persona1	Intentos izquierda	10	9	0.9	90%
	Intentos derecha	10	8	0.8	80%
Persona2	Intentos izquierda	10	9	0.9	90%
	Intentos derecha	10	9	0.9	90%

Persona3	Intentos izquierda	10	9	0.9	90%
	Intentos derecha	10	8	0.8	80%
Persona4	Intentos izquierda	10	9	0.9	90%
	Intentos derecha	10	8	0.8	80%
Persona5	Intentos izquierda	10	10	1	100%
	Intentos derecha	10	8	0.8	80%
Persona6	Intentos izquierda	10	8	0.8	80%
	Intentos derecha	10	6	0.6	60%
Persona7	Intentos izquierda	10	8	0.8	80%
	Intentos derecha	10	6	0.6	60%
Persona8	Intentos izquierda	10	8	0.8	80%
	Intentos derecha	10	10	1	100%
Persona9	Intentos izquierda	10	10	1	100%
	Intentos derecha	10	8	0.8	80%
Persona10	Intentos izquierda	10	9	0.9	90%
	Intentos derecha	10	9	0.9	90%
	TOTAL(TO)	200	169		

Tabla 10. Datos Postest del indicador Índice de reconocimiento de órdenes de desplazamiento
Fuente: Elaboración propia

Indicador 2: órdenes realizadas correctas

Recolección de datos después de usar el software inteligente para el indicador órdenes realizadas correctas (ver tabla 11).

Casos de prueba	Intentos	Total Intentos	OC	Ordenes Visuales Correctas	Ordenes Visuales Correctas(%)
Persona1	Intentos izquierda	9	8	0.9	90%
	Intentos derecha	8	8	1.0	100%
Persona2	Intentos izquierda	9	8	0.9	90%
	Intentos derecha	9	5	0.6	60%

Persona3	Intentos izquierda	9	9	1.0	100%
	Intentos derecha	8	6	0.8	80%
Persona4	Intentos izquierda	9	8	0.9	90%
	Intentos derecha	8	6	0.8	80%
Persona5	Intentos izquierda	10	9	0.9	90%
	Intentos derecha	8	6	0.8	80%
Persona6	Intentos izquierda	8	8	1.0	100%
	Intentos derecha	6	4	0.7	70%
Persona7	Intentos izquierda	8	8	1.0	100%
	Intentos derecha	6	5	0.8	80%
Persona8	Intentos izquierda	8	8	1.0	100%
	Intentos derecha	10	8	0.8	80%
Persona9	Intentos izquierda	10	6	0.6	60%
	Intentos derecha	8	8	1.0	100%
Persona10	Intentos izquierda	9	8	0.9	190%
	Intentos derecha	9	9	1.0	100%
	TOTAL(OTR)	169	145		

Tabla 11. Datos Postest del indicador ordenes visuales correctas
Fuente: Elaboración propia

Por último, se va a realizar los cálculos en base a las fórmulas de los indicadores.

- Para el indicador: Índice de Reconocimiento de Ordenes de desplazamiento

$$\text{IRO} = \frac{OVGC}{TO} = \frac{169}{200} = 0.85$$

IRO: Índice de Reconocimiento de Ordenes de desplazamiento
OVGC: Órdenes visuales generadas correctas.
TO: Total ordenes

- Para el indicador: Ordenes visuales correctas

$$OVC = \frac{OC}{OTR} = \frac{145}{169} = 0.84$$

OVC: Órdenes visuales correctas.

OTR: Ordenes totales realizadas.

OC: Ordenes Correctas

CAPÍTULO III. RESULTADOS

3.1 ANALISIS DE RESULTADOS

Análisis descriptivo

En el estudio se aplicó un software inteligente (Variable Independiente) para evaluar el índice de reconocimiento de órdenes y las ordenes realizadas correctas para el orden de desplazamiento (Variable dependiente); para ello se aplicó un Pre-Test que permita conocer las condiciones iniciales del indicador; posteriormente se implementó el software inteligente y nuevamente se registró el índice de reconocimiento de órdenes y las ordenes realizadas correctas para el orden de desplazamiento. Los resultados descriptivos de estas medidas se observan en las Tablas 12 y 13.

- **Indicador índice de reconocimiento de órdenes**

  Los resultados descriptivos del índice de reconocimiento de órdenes, estas medidas se observan en la Tabla 12.

Estadísticos descriptivos

	N	Mínimo	Máximo	Media	Desv. Desviación
ReconocimientoDeOrdenes_Pretest	20	,40	,85	,6025	,15259
ReconocimientoDeOrdenes_Postest	20	,60	1,00	,8450	,10990
N válido (por lista)	20				

Tabla 12. Estadístico descriptivo del indicador índice de reconocimiento de ordenes
Fuente: Elaboración propia

En el caso del índice de reconocimiento de órdenes para generar ordenes de desplazamiento, en el pre-test se obtuvo un valor de 60.25%, mientras que en el post-test fue de 84.5% tal como se aprecia en la Figura 10; esto indica una gran diferencia antes y después de la implementación del software inteligente; así mismo, el índice de

reconocimiento de órdenes mínima fue del 40% antes, y 60% (ver Tabla 12) después de la implementación del software inteligente.

En cuanto a la dispersión del índice de reconocimiento de órdenes, en el pre-test se tuvo una variabilidad de 0.15; sin embargo, en el post-test se tuvo un valor de 0.11.

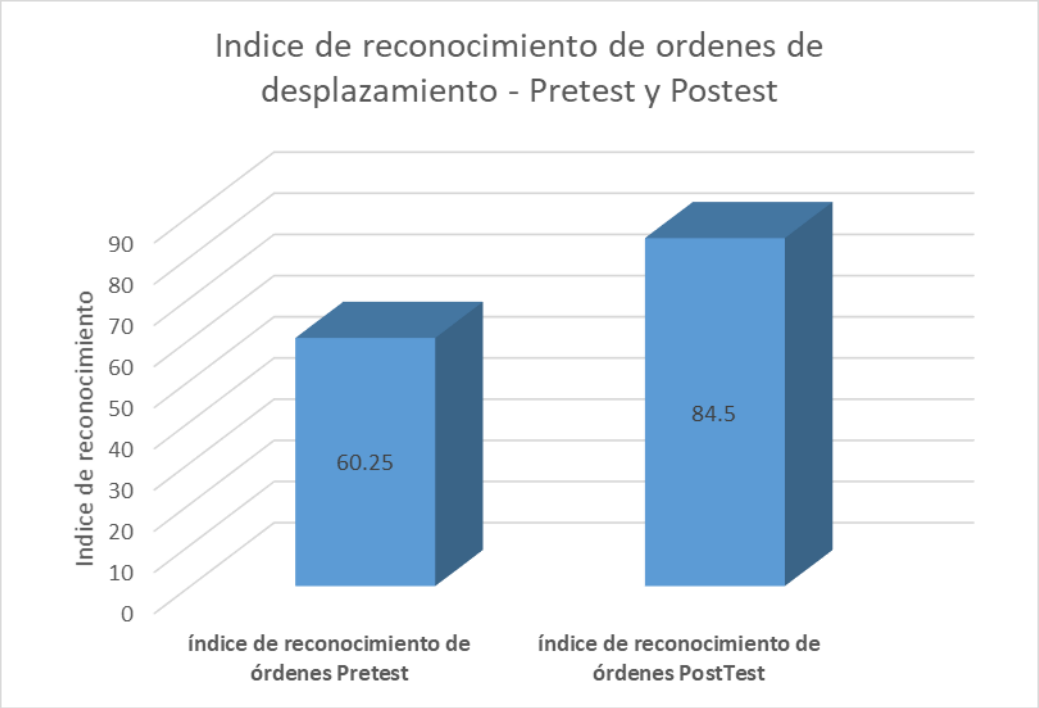

Figura 10. Índice de reconocimiento de orden de desplazamiento - Pretest y Postest
Fuente: Elaboración propia

- **Indicador: Cantidad de órdenes realizadas correctas**

  Los resultados descriptivos de la cantidad de órdenes realizadas correctas, estas medidas se observan en la Tabla 13.

Estadísticos descriptivos

	N	Mínimo	Máximo	Media	Desv. Desviación
CantidadOrdenesCorrectas_Pretest	20	,00	,80	,2950	,21392
CantidadOrdenesCorrectas_Postest	20	,60	1,00	,8600	,12937
N válido (por lista)	20				

Tabla 13. Estadístico descriptivo del indicador ordenes correctas – Pretest y Postest
Fuente: Elaboración propia

En el caso de la cantidad de órdenes realizadas correctas para generar ordenes de desplazamiento, en el pre-test se obtuvo un valor de 30%, mientras que en el post-test fue de 87 % tal como se aprecia en la Figura 11; esto indica una gran diferencia antes y después de la implementación del software inteligente; así mismo, la cantidad de órdenes realizadas correctas mínima fue del 0% antes, y 60% (ver Tabla 13) después de la implementación del software inteligente.

En cuanto a la dispersión de la cantidad de órdenes realizadas correctas, en el pre-test se tuvo una variabilidad de 0.21; sin embargo, en el post-test se tuvo un valor de 0.13.

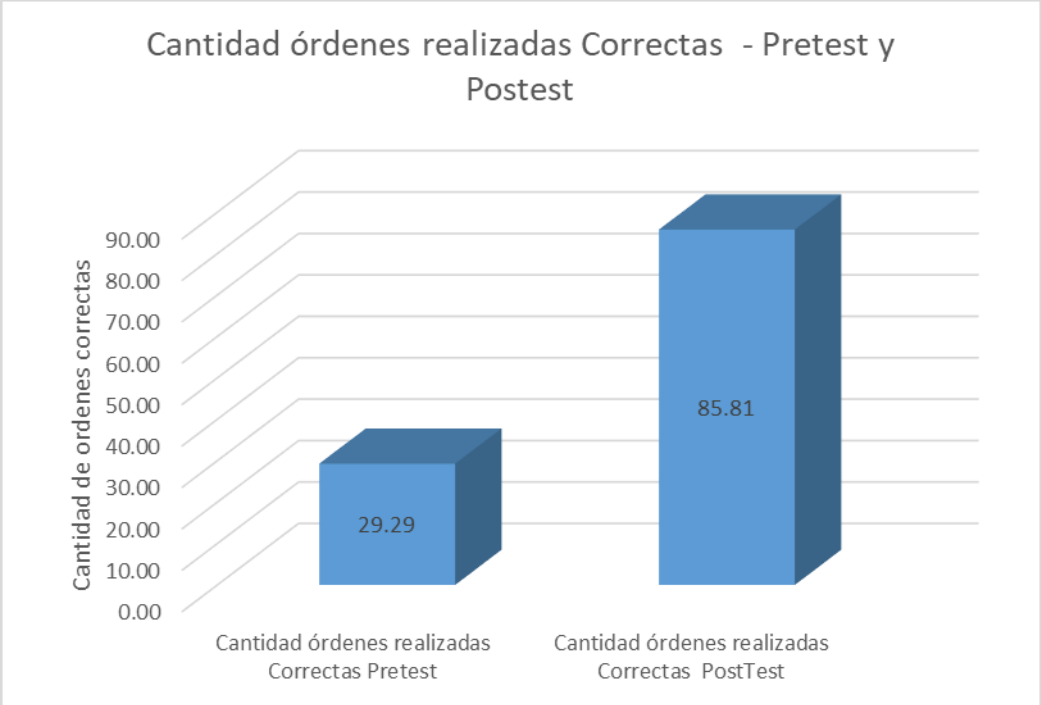

Figura 11. Cantidad de ordenes realizadas correctas, Pretest y Postest
Fuente: Elaboración propia

Análisis Inferencial

Prueba de Normalidad

Se procedió a realizar las pruebas de normalidad para los indicadores de índice de reconocimiento de órdenes y la cantidad de órdenes realizadas correctas a través del método Shapiro-Wilk, debido a que el tamaño de nuestra muestra estratificada está conformado por 20 fichas registros y es menor a 50, tal como lo indica Hernández, Fernández y Baptista (2006, p. 376). Dicha prueba se realizó introduciendo los datos de cada indicador en el software estadístico SPSS 25.0, para un nivel de confiabilidad del 95%, bajo las siguientes condiciones:

Si:

Sig. $<$ 0.05 adopta una distribución no normal.

Sig. $\geq$ 0.05 adopta una distribución normal.

Dónde:

Sig. : P-valor o nivel crítico del contraste.

Los resultados fueron los siguientes:

- **Indicador índice de reconocimiento de órdenes**

  Con el objetivo de seleccionar la prueba de hipótesis; los datos fueron sometidos a la comprobación de su distribución, específicamente si los datos del índice de reconocimiento de órdenes contaban con distribución normal.

	Shapiro-Wilk		
	Estadístico	gl	Sig.
ReconocimientoDeOrdenes_Pretest	,865	20	,010
ReconocimientoDeOrdenes_Postest	,856	20	,007

Tabla 14. Resultados del indicador Índice de reconocimiento de ordenes
Fuente: Elaboración propia

Como se muestra en la Tabla 14 los resultados de la prueba indican que el Sig. del índice de reconocimiento de órdenes en el Pre-Test fue de 0.10, cuyo valor es mayor que 0.05. Por lo tanto, el índice de reconocimiento de órdenes se distribuye normalmente. Los resultados de la prueba del Post-Test indican que el Sig. del índice de reconocimiento de órdenes fue de 0.07, cuyo valor es mayor que 0.05, por lo que indica que el índice de reconocimiento de órdenes se distribuye normalmente. Lo que confirma la distribución normal de ambos datos de la muestra, se puede apreciar en la Figura 12.

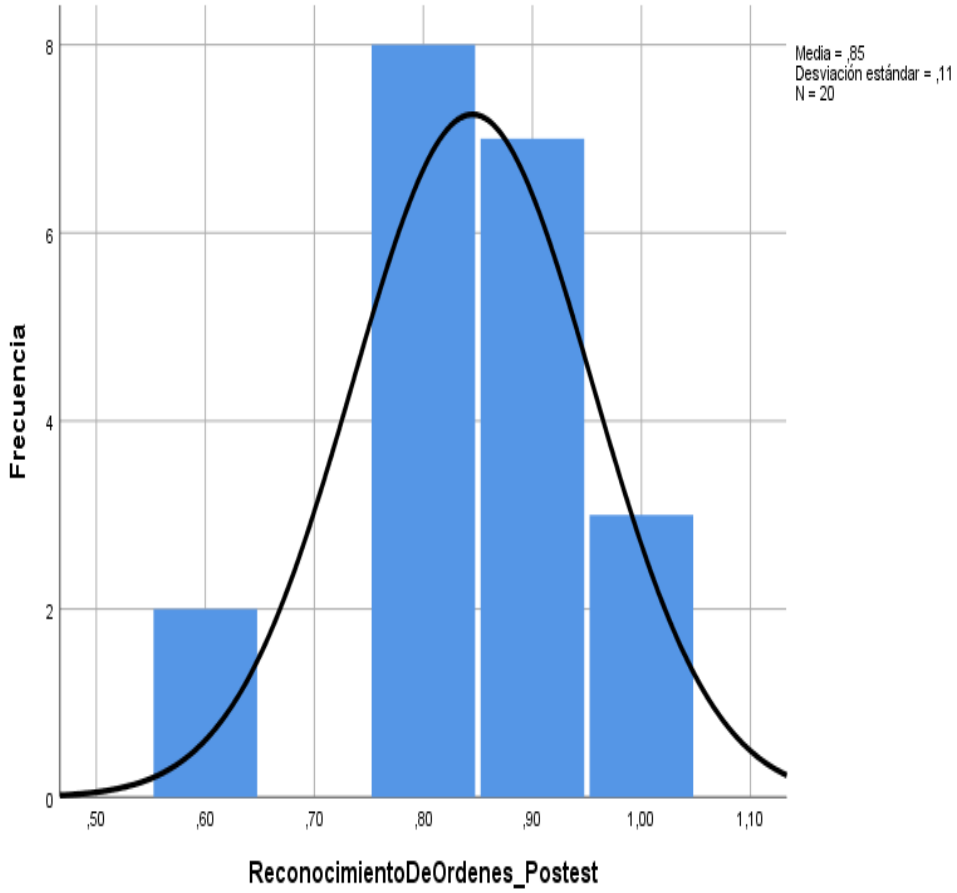

Figura 12. Campana de Gaus del indicador Reconocimiento de ordenes – Postest
Fuente: Elaboración propia

- **Indicador cantidad de órdenes realizadas correctas**

  Con el objetivo de seleccionar la prueba de hipótesis; los datos fueron sometidos a la comprobación de su distribución, específicamente si los datos de la cantidad de órdenes realizadas correctas contaban con distribución normal.

	Shapiro-Wilk		
	Estadístico	gl	Sig.
CantidadOrdenesCorrectas_Pretest	,869	20	,026
CantidadOrdenesCorrectas _Postest	,873	20	,006

Tabla 15. Resultados del indicador Ordenes realizadas correctas
Fuente: Elaboración propia

Como se muestra en la Tabla 15 los resultados de la prueba indican que el Sig. de la cantidad de órdenes realizadas correctas en el Pre-Test fue de 0.11, cuyo valor es mayor que 0.05. Por lo tanto, la cantidad de órdenes realizadas correctas se distribuye normalmente. Los resultados de la prueba del Post-Test indican que el Sig. de la cantidad de órdenes realizadas correctas fue de 0.13, cuyo valor es mayor que 0.05, por lo que indica que la cantidad de órdenes realizadas correctas se distribuye normalmente. Lo que confirma la distribución normal de ambos datos de la muestra, se puede apreciar en la Figura 13.

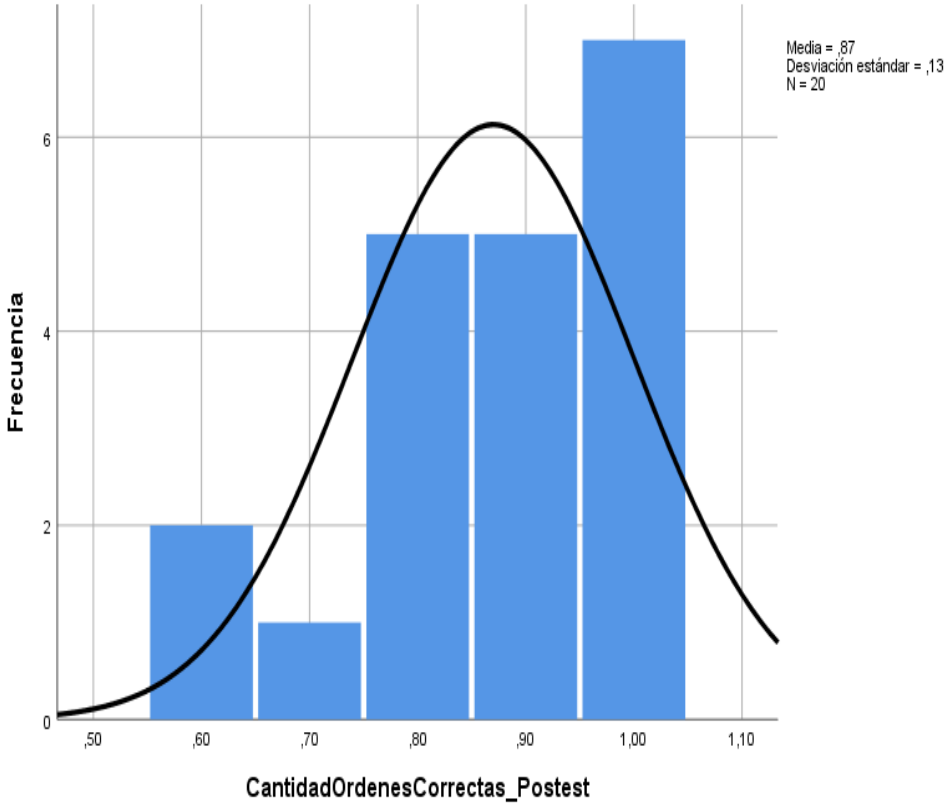

Figura 13. Campana de gauss del indicador Cantidad de ordenes correctas – Postest
Fuente: Elaboración propia

3.2 Prueba de Hipótesis

Hipótesis de Investigación 1:

- **H1:** El software inteligente incrementa el índice de reconocimiento de órdenes de desplazamiento a partir de los movimientos de la vista de personas con discapacidad.
- **Indicador: índice de reconocimiento de órdenes**

Hipótesis Estadísticas

Definiciones de Variables:

- IRa: Índice de reconocimiento de órdenes antes de usar el software inteligente.
- IRd: Índice de reconocimiento de órdenes después de usar el software inteligente.

- **H0:** El software inteligente no incrementa el índice de reconocimiento de órdenes de desplazamiento a partir de los movimientos de la vista de personas con discapacidad.

$$H0 = IRa \geq IRd$$

El indicador sin el software inteligente es mejor que el indicador con el software inteligente.

- **HA:** El software inteligente incrementa el índice de reconocimiento de órdenes de desplazamiento a partir de los movimientos de la vista de personas con discapacidad.

$$H0 = IRa < IRd$$

El indicador con el software inteligente es mejor que el indicador sin el software inteligente.

Se concluye de la Figura 10 que existe un incremento en el índice de reconocimiento de órdenes, el cual se puede verificar al comparar las medias respectivas, que asciende de 60.25% al valor de 84.5%.

En cuanto al resultado del contraste de hipótesis se aplicó la Prueba T-Student, debido a que los datos obtenidos durante la investigación (Pre-Test y Post-Test) se distribuyen normalmente. El valor de T contraste es de -9.336, el cual es claramente menor que -1.703. (Ver tabla 16).

		Prueba de T-Student		
	Media	T	gl	Sig. (bilateral)
ReconocimientoDeOrdenes_Pretest	0.6025			
ReconocimientoDeOrdenes_Postest		-9.336	19	,000
ReconocimientoDeOrdenes_Pretest	0.8450			

Tabla 16. Prueba T del indicador índice de reconocimiento de ordenes
Fuente: Elaboración propia

Entonces, se rechaza la hipótesis nula, aceptando la hipótesis alterna con un 95% de confianza. Además, el valor T obtenido, como se muestra en la Figura 14, se ubica en la zona de rechazo. Por lo tanto, El software inteligente

incrementa el índice de reconocimiento de órdenes de desplazamiento a partir de los movimientos de la vista de personas con discapacidad.

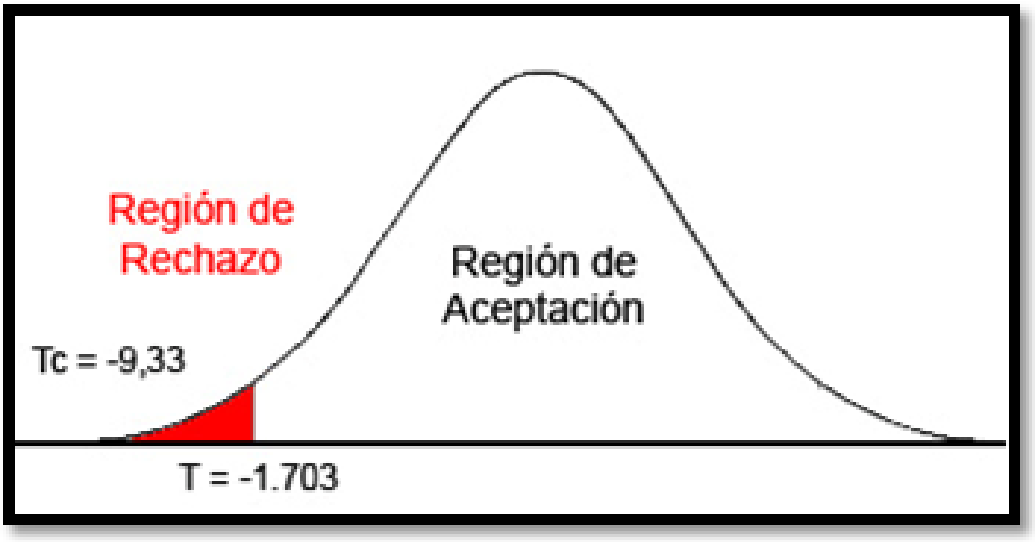

Figura 14. Área de rechazo y aceptación de la hipótesis del primer indicador
Fuente: Elaboración propia

Hipótesis de Investigación 2:

- **H1:** El software inteligente incrementa la cantidad de órdenes realizadas correctas a partir de los movimientos de la vista de personas con discapacidad.
- **Indicador: cantidad de órdenes realizadas correctas**

Hipótesis Estadísticas

Definiciones de Variables:

- ORCa: Cantidad de órdenes realizadas correctas antes de usar el software inteligente.
- ORCd: Cantidad de órdenes realizadas correctas después de usar el software inteligente.

- **H0:** El software inteligente no incrementa la cantidad de órdenes realizadas correctas de desplazamiento a partir de los movimientos de la vista de personas con discapacidad.

$$H0 = ORCa \geq ORCd$$

El indicador sin el software inteligente es mejor que el indicador con el software inteligente.

- **HA:** El software inteligente incrementa la cantidad de órdenes realizadas correctas de desplazamiento a partir de los movimientos de la vista de personas con discapacidad.

$$H0 = ORCa < ORCd$$

El indicador con el software inteligente es mejor que el indicador sin el software inteligente.

Se concluye de la Figura 11 que existe un incremento en la cantidad de órdenes realizadas correctas, el cual se puede verificar al comparar las medias respectivas, que asciende de 29.29% al valor de 85.81 %.

En cuanto al resultado del contraste de hipótesis se aplicó la Prueba T-Student, debido a que los datos obtenidos durante la investigación (Pre-Test y Post-Test) se distribuyen normalmente. El valor de T contraste es de -10.682, el cual es claramente menor que -1.703. (Ver tabla 17).

		Prueba de T-Student		
	Media	T	gl	Sig. (bilateral)
CantidadOrdenesCorrectas_Pretest	0.3000			
CantidadOrdenesCorrectas_Postest		-10.682	19	,000
CantidadOrdenesCorrectas_Pretest	0.8700			

Tabla 17. Prueba T del indicador Ordenes Correctas
Fuente: Elaboración propia

Entonces, se rechaza la hipótesis nula, aceptando la hipótesis alterna con un 95% de confianza. Además, el valor T obtenido, como se muestra en la Figura 15, se ubica en la zona de rechazo. Por lo tanto, El software inteligente incrementa la cantidad de órdenes realizadas correctas de desplazamiento a partir de los movimientos de la vista de personas con discapacidad.

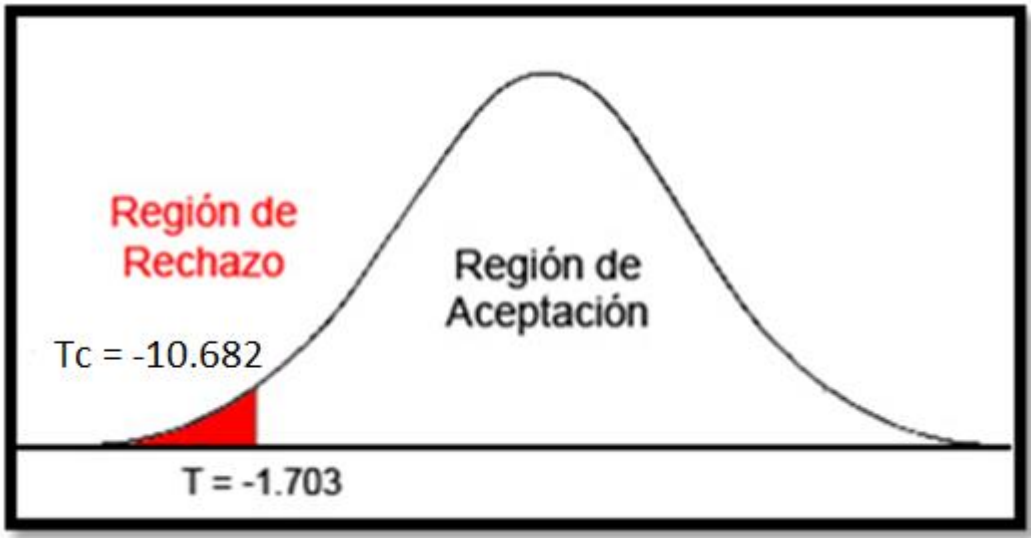

Figura 15. Área de aceptación y rechazo de la hipótesis del segundo indicador
Fuente: Elaboración propia

Análisis de la dimensión grado de inteligencia

Se aplicó el software inteligente en la dimensión grado de inteligencia donde se mide el nivel de afectación del generador de órdenes de desplazamiento donde se obtuvo resultado del indicador reconocimiento de órdenes de desplazamiento en un resultado de 0.91, haciendo uso de la recolección de datos mediante el fichaje.

VARIABLES DE MEDICIÓN PARA EL INDICADOR: INDICE DE RECONOCIMIENTO DE ORDENES:

Para poder identificar el índice de reconocimiento de ordenes según los datos de la Tabla 10 y Tabla 8, se puede representar en la Figura 16.

$$\text{IRO} = \frac{OVGC}{TO} x100 = \frac{91}{100} x100 = 91\%$$

IRO: Índice de Reconocimiento de Ordenes
OVGC: Órdenes visuales generadas correctas.
TO: Total ordenes

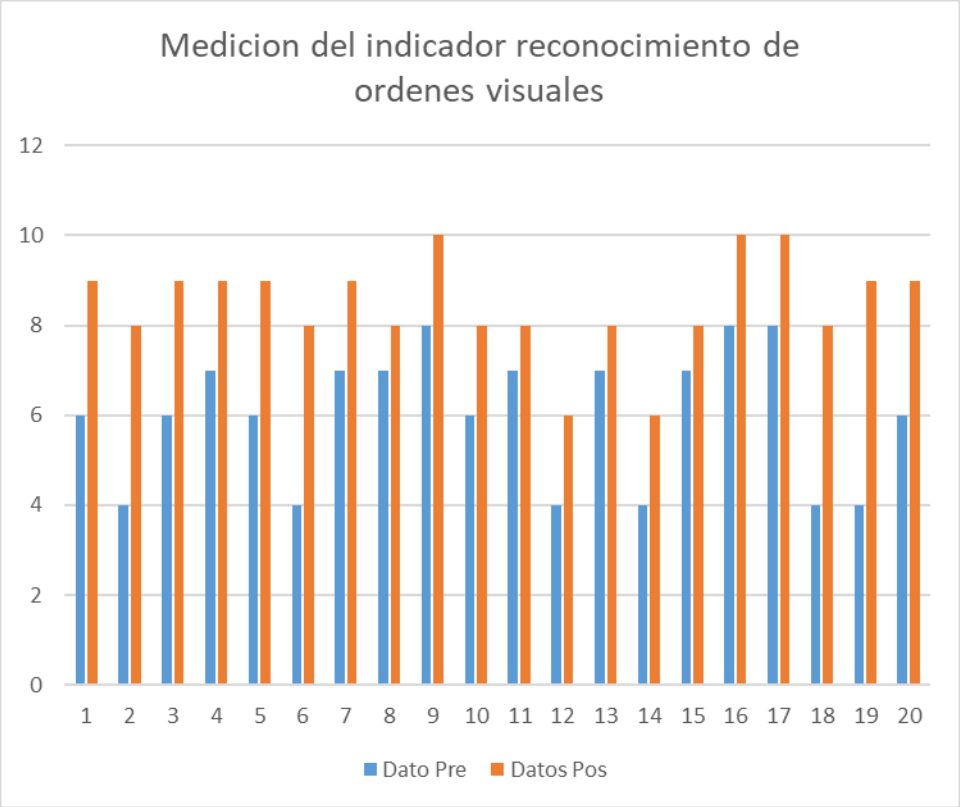

Figura 16. Medición del indicador re reconocimiento de ordenes visuales
Fuente: Elaboración propia

VARIABLES DE MEDICIÓN PARA EL INDICADOR: ÓRDENES VISUALES REALIZADAS CORRECTAS:

Para poder identificar la cantidad de reconocimientos de ordenes según los datos de la Tabla 11 y Tabla 9, se puede representar en la Figura 17

$$OVC = \frac{OC}{OTR} x100$$

OVC: Órdenes visuales correctas.
OTR: Ordenes totales realizadas.
OC: Ordenes Correctas

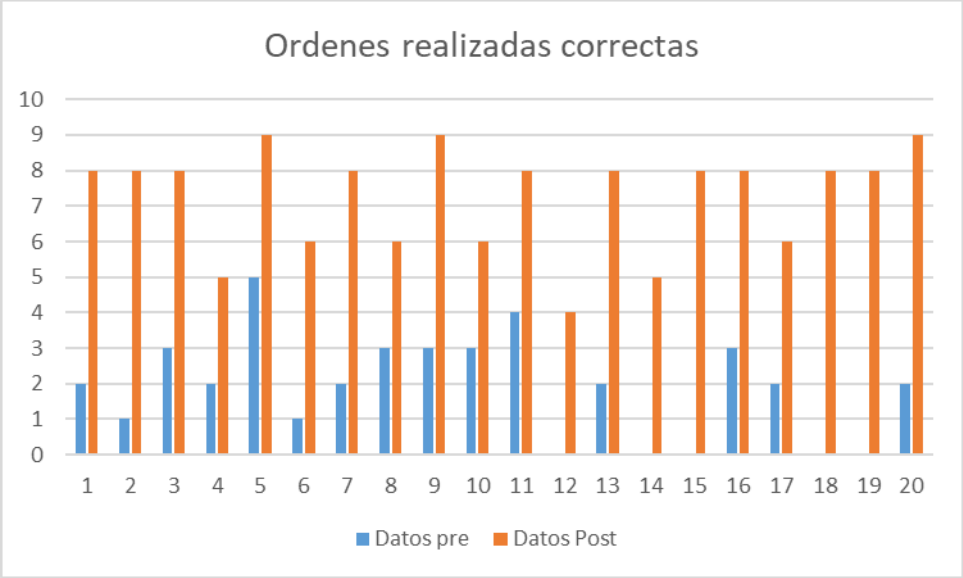

Figura 17: Medición del índice de reconocimiento de orden
Fuente: Elaboración propia

CAPÍTULO IV. DISCUSIÓN Y CONCLUSIONES

4.1. DISCUSIÓN

A continuación, se grafican los resultados obtenidos para cada indicador respaldado por la prueba de hipótesis.

INDICADOR 1: Índice de Reconocimiento de Ordenes

El resultado de la prueba de hipótesis para este indicador concluye que el índice de reconocimiento de órdenes por un usuario usando el sistema propuesto es mayor que haciendo uso del método de (Bereciartura, 2016).

Según la tabla 18 mostrada y graficada en la figura 18 podemos apreciar el valor promedio del indicador índice de reconocimiento de órdenes del software inteligente fue de 0.845 , lo que evidencia una clara mejora de 0.05 en la tasa de reconocimiento, lo que coincide con la investigación de (Bereciartura, 2016) titulada “Aplicación a segmentación de hígado sobre imágenes de resonancia magnética multisecuencial” donde se implementó un sistema para reconocer características específicas del hígado en imágenes de manera autónoma.

Resultado de Bereciartura	Investigación Propuesta	Diferencia
0.84	0.845	0.05

Tabla 18. Comparativa de Bereciartura con la investigación
Fuente: Elaboración propia

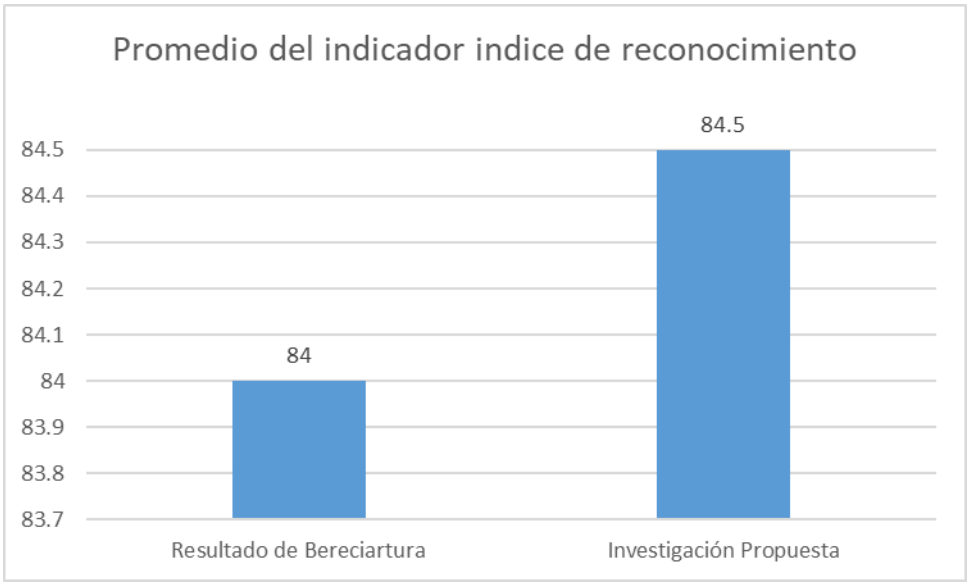

Figura 18: Promedio del indicador de reconocimiento de orden
Fuente: Elaboración propia

INDICADOR 2: Órdenes visuales realizadas correctas

El resultado de la prueba de hipótesis para este indicador concluye que el índice órdenes visuales realizadas correctas por un usuario usando el sistema propuesto es mayor el modelo propuesto por (Sevillano, 2018).

Según la tabla 19 mostrada y graficada en la figura 19 podemos apreciar el valor promedio del indicador ordenes visuales realizadas correctas con el software inteligente fue de 0.89, lo que evidencia una clara proximidad en resultados de 0.08 con la investigación por (Sevillano, 2018) con 0.97, cuyo título de investigación es "Modelos basados en el aprendizaje automático" donde se implementó un modelo para reconocer características específicas de una imagen de manera autónoma.

Resultado de Sevillano	Investigación Propuesta	Diferencia
0.87	0.85	-0.02

Tabla 19.Comparativa de Sevillano con la investigación actual
Fuente: Elaboración propia

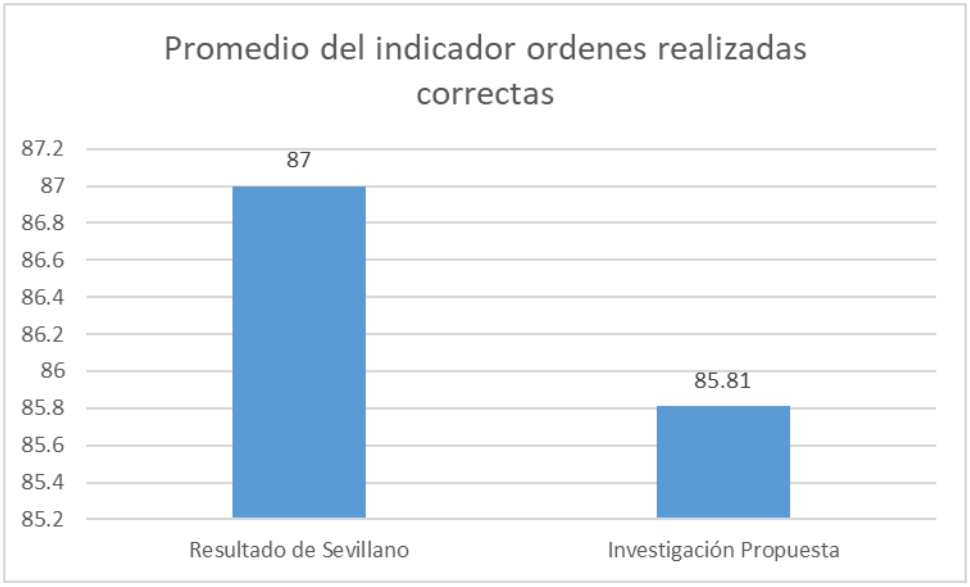

Figura 19: Promedio del indicador ordenes visuales correctas
Fuente: Elaboración propia

4.2. CONCLUSIONES

Al finalizar trabajo de investigación, se llegaron a las siguientes conclusiones:

- Se demostró la influencia del software inteligente en la dimensión grado de inteligencia del reconocimiento de órdenes visuales para personas con discapacidad según el índice de reconocimiento de órdenes visuales de 169 reconocimientos correctos de 200, por lo tanto, usando el modelo y los algoritmos adecuados se obtiene una tasa de reconocimiento de órdenes visuales de 0.85.
- Se demostró la influencia del software inteligente en la dimensión órdenes visuales del reconocimiento en las órdenes visuales correctas para personas con discapacidad según el índice de reconocimiento de órdenes visuales correctas de 145 reconocimientos correctos de 169, por lo tanto, se obtiene respecto a la tasa generación de órdenes visuales correctas de 0.86.
- En base a los resultados obtenidos, podemos deducir que la solución se puede aplicar a otra persona con discapacidad. Analizando previamente si es que hay algún detalle del ojo como cuando la persona realmente está dando una orden.
- El software inteligente tuvo una influencia positiva sobre el reconocimiento de órdenes de desplazamiento a partir de movimientos.

4.3. RECOMENDACIONES

- Usar la presente investigación como fin para el índice de reconocimiento de ordenes tomando como referencia a una tasa de efectividad de 0.845 y una tasa de órdenes realizadas correctas de 0.858 graficado, identificado en la Figura 10 y Figura 11 respectivamente.
- Usar la presente investigación como base de investigación para poder generar órdenes a partir de la vista, usando otros modelos de detección del iris como es el bloodfinder circle.
- Usar un modelo general para poder ser aplicado en software para cualquier persona que lo desee utilizar, indistinto si es que padece una falencia motora.
- Usar este prototipo como base para nuevos conocimientos referente al procesamiento digital de imágenes y dispositivos externos como sensores.
- Usar dispositivos como Raspberry Pi, para intentar independizar con un hardware más portable.

REFERENCIAS

Arduino. (01 de 04 de 2019). *Arduino*. Obtenido de http://arduino.org

B Wodlinger, J. E.-K. (2014). Ten-dimensional anthropomorphic arm control in a human brain−machine interface: difficulties, solutions, and limitations. *Journal of Neural Engineering*, 17.

Baray, H. L. (2006). *INTRODUCCIÓN A LA METODOLOGÍA DEL LA INVESTIGACIÓN.* CUAUHTEMOC, CHIHUAHUA, MEXICO: Instituto Tecnológico de Cd. Cuauhtémoc.

Bazaga, A. R. (01 de 04 de 2018). *OpenCV: Librería de Visión por Computador*. Obtenido de https://osl.ull.es/software-libre/opencv-libreria-vision-computador/

Behar Rivero, D. S. (2008). *Metodología de la Investigación.* Argentina: Shalom.

Bereciartura, A. (2016). *Aplicacion a segmentacion de higado sobre imagenes de resonancia magnetica multisecualcia.* España: Universidad de Pasco.

BienSalud, B. (25 de 08 de 2015). *MENSAJES DEL CEREBRO AL RESTO DEL CUERPO*. Obtenido de http://www.revistabiendesalud.com/general/mensajes-del-cerebro-al-resto-del-cuerpo/

Casanova, A. A. (2010). *Psicología de le percepción visual.* Barcelona: Universidad de Barcelona.

CHANG TORTOLERO, O. G. (2013). Detección de objetos complejos usando redes neurales y micro temblor ocular. *Scielo Venezuela*, 49-55.

Cherry, K. (2017). *What Is Applied Research?* EEUU: verywell.

COHEN, K. D. (1993). *Sistema de Información para la Toma de Decisiones.* México : Publicaciones Acuario Editores S.A.

COPPIN, B. (2004). *Artificial Intelligence Illuminated.* Mississauga: Jones and Bartlett Publishers.

Fernando, C. M. (2003). *El proyecto de investigación y su esquema de elaboración.* Caracas: Uyapar.

García, S. (2008). *Visión Artificial y Procesamiento Digital de Imágenes usando Matlab.* Ecuador: Ibarra.

Gibson, J. (1979). *The Ecological Approach to Visual Perception.* Boston: Houghton Houghton.

Gómez, G., & Sucar, E. (2008). *Visión Computacional.* Mexico: Helmholtz Zentrum Munchen.

Gonzáles, F. K. (1989). *Robótica: Control, deteccion, visión e inteligencia.* Mexico: McGraw-Hill Interamericana de Mexico.

GUILLERMO, H. V. (2017). *MANUAL DE TÉCNICA DE INVESTIGACIÓN CONCEPTO Y APLICACIONES.* MEXICO: IPLADEES.

Gutiérrez, E. A. (2003). *PROCESAMIENTO DIGITAL DE IMAGEN.* España: Universidad de León.

HERNÁNDEZ, R. F. (2014). *Metodología de la investigación.* México: Mc Graw Hill.

J, N. N. (1998). *Artificial Intelligence: A New Synthesis.* EEUU: Morgan Kaufmann Publishers, Inc.

K, P. W. (2001). *Digital Image Processing.* Estados Unidos: John Wiley & Songs, INC.

Lenin, N. C. (2014). *Epistemología y metodología.* Mexico: Grupo Editorial Patria.

Norvig, S. R. (2004). *Inteligencia Artificial, Un enfoque moderno.* México: Prentice Hall.

Open4U, B. (6 de 5 de 2019). *Cinco librerías en Python para científicos de datos: cómo visualizar información*. Obtenido de bbvaopen4u.com: https://bbvaopen4u.com/es/actualidad/cinco-librerias-en-python-para-cientificos-de-datos-como-visualizar-informacion

Querelle. (01 de 04 de 2018). *profesor en linea*. Obtenido de http://www.profesorenlinea.cl/

Rossum, G. v. (01 de 04 de 2008). *El tutorial de Python.* EEUU: Python Software Foundation. Obtenido de www.netbeans.org

Russell, S., & Norvig, P. (2004). *Inetigencia Artificial, un enfoque moderno.* Madrid: PEARSON PRENTICE HALL.

Sampiere., H. (2011). *Metodología de investigación científica. 5a. ed.* Mexico: Editorial cáliz,.

Serrano, A. G. (2012). *Inteligencia Artificial, Fundamentos, práctica y aplicaciones.* madrid: RC-Libros ISBN: 978-84939450-2-2.

Sevillano, V. (2018). *modelos basados en el aprendizaje automático.* Costa Rica: UNED.

Sommerville, I. (1989). *Software Engineering.* Inglaterra: Addison-Wesley.

Wander, J. (2013). *High-performance neuroprosthetic control by an individual with tetraplegia.* Pittsburgh: The Lancet.

WorldCat. (02 de 02 de 2019). *WorldCat Provider*. Obtenido de WorldCat Provider: https://www.worldcat.org/

ANEXOS

ANEXO 1: TABLA DE RESULTADOS

A. Tabla de datos

Índice de reconocimiento de ordenes

Medición	Intentos	Total Intentos	OVGC	Índice de reconocimiento (%)
Persona1	Intentos izquierda	10	6	60%
	Intentos derecha	10	4	40%
Persona2	Intentos izquierda	10	6	60%
	Intentos derecha	10	7	70%
Persona3	Intentos izquierda	10	6	60%
	Intentos derecha	10	4	40%
Persona4	Intentos izquierda	10	7	70%
	Intentos derecha	10	7	70%
Persona5	Intentos izquierda	10	8	80%
	Intentos derecha	10	6	60%
Persona6	Intentos izquierda	10	7	70%
	Intentos derecha	10	4	40%
Persona7	Intentos izquierda	10	7	70%
	Intentos derecha	10	4	40%
Persona8	Intentos izquierda	10	7	70%
	Intentos derecha	10	8	85%
Persona9	Intentos izquierda	10	8	80%
	Intentos derecha	10	4	40%
Persona10	Intentos izquierda	10	4	40%
	Intentos derecha	10	6	60%
	TOTAL(TO)	200	48	
	IRO	0.85		

Tabla 20: Datos del indicador índice de reconocimiento de ordenes visuales
Fuente: Elaboración propia

Ordenes visuales realzadas correctas antes del software

Medición	Intentos	Total Intentos	OC	Ordenes Visuales Correctas(%)
Persona1	Intentos izquierda	6	2	30%
	Intentos derecha	4	1	30%
Persona2	Intentos izquierda	6	3	50%
	Intentos derecha	7	2	30%
Persona3	Intentos izquierda	6	5	80%
	Intentos derecha	4	1	30%
Persona4	Intentos izquierda	7	2	30%
	Intentos derecha	7	3	40%
Persona5	Intentos izquierda	8	3	40%
	Intentos derecha	6	3	50%
Persona6	Intentos izquierda	7	4	60%
	Intentos derecha	4	0	0%
Persona7	Intentos izquierda	7	2	30%
	Intentos derecha	4	0	0%
Persona8	Intentos izquierda	7	0	0%
	Intentos derecha	8	3	40%
Persona9	Intentos izquierda	8	2	30%
	Intentos derecha	4	0	0%
Persona10	Intentos izquierda	4	0	0%
	Intentos derecha	6	2	30%
	TOTAL(OTR)	120	38	
	OCV	0.89		

Tabla 21: Datos del indicador ordenes visuales realizadas correctas

Fuente: Elaboración propia

B. Análisis de datos

Índice de reconocimiento de ordenes

$$\text{IRO} = \frac{OVGC}{TO} x100 = \frac{169}{200} x100 = 85\%$$

IRO: Índice de Reconocimiento de Ordenes

OVGC: Órdenes visuales generadas correctas.

TO: Total ordenes

Órdenes visuales realizadas correctas

$$OVC = \frac{OC}{OTR} x100 = \frac{145}{169} x100 = 86\%$$

OVC: Órdenes visuales correctas.

OTR: Ordenes totales realizadas.

OC: Ordenes Correctas

ANEXO 2: CAPTURA DE PANTALLAS

Interfaz para el manejo del software

Interfaz muestra la captura de la escena por medio de la cámara, para procesar la imagen, para encontrar el rostro, el ojo y la pupila, la figura hace referencia al CU01.

En la siguiente figura se muestra en el preciso momento cuando da la orden al lado izquierdo. Tener en cuenta que el lado izquierdo es con respecto de la pantalla, mas no de la persona.

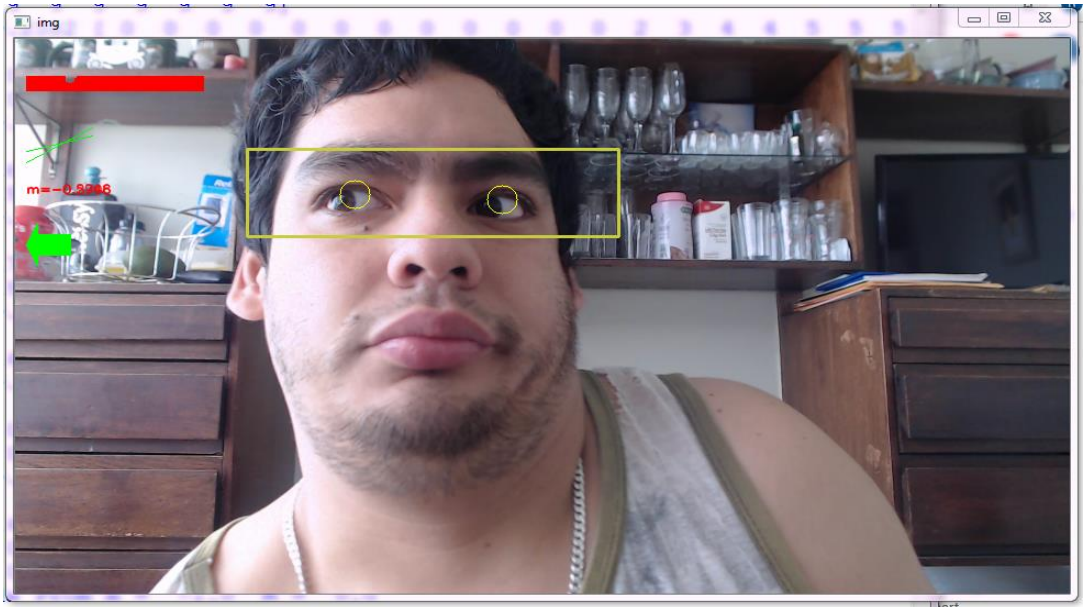

Figura 20: Interfaz de la escena como genera orden visual
Fuente: Elaboración propia

- Interfaz de la captura del ojo mostrando la vista y a pupila, y en función de ella mostrar generar las ordenes, haciendo referencia al caso de CU02.

Figura 21: Interfaz de la captura de la ubicación de los ojos
Fuente: Elaboración propia

- Interfaz para mostrar la ubicación de la pupila, y en función de ella mostrar generar las ordenes, donde se muestra la imagen en tono de grises por conveniencia para ser procesado y analizado, haciendo referencia al caso de CU02.

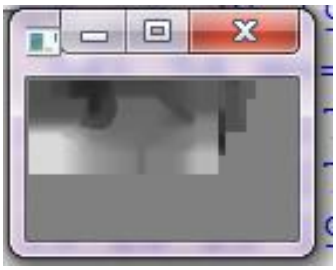

Figura 22: Interfaz de ubicación de la pupila en tono de gris
Fuente: Elaboración propia

Interfaz para el manejo del hardware: Aplicación Arduino

- Interfaz muestra la escucha las órdenes provenientes del pc, la figura hace referencia al CU03.

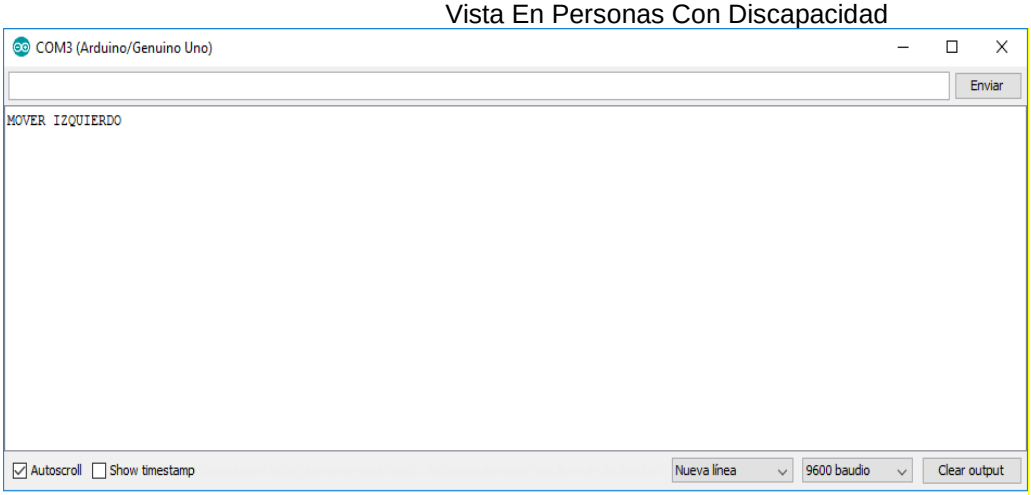

Figura 23: Escucha de las ordenes
Fuente: Elaboración propia

- Interfaz muestra la orden de encender led, la figura hace referencia al CU04.

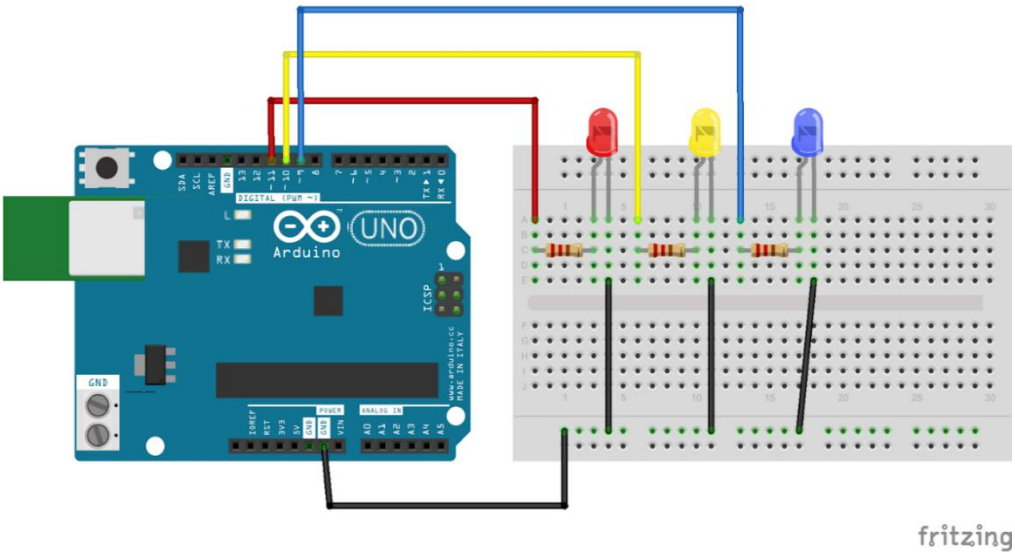

Figura 24: Diagrama del dispositivo hardware para enviar la orden
Fuente: Elaboración propia

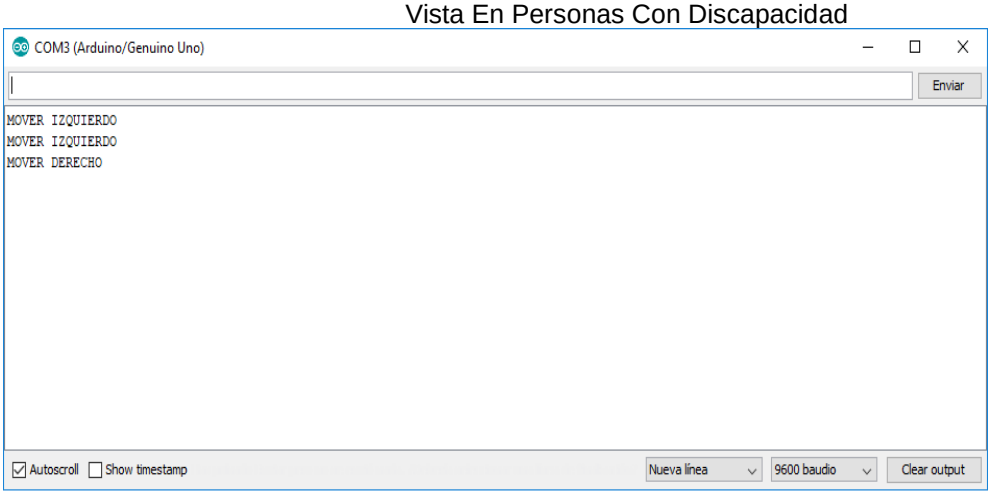

Figura 25: Muestra la orden de encender el led, dependiendo la orden
Fuente: Elaboración propia

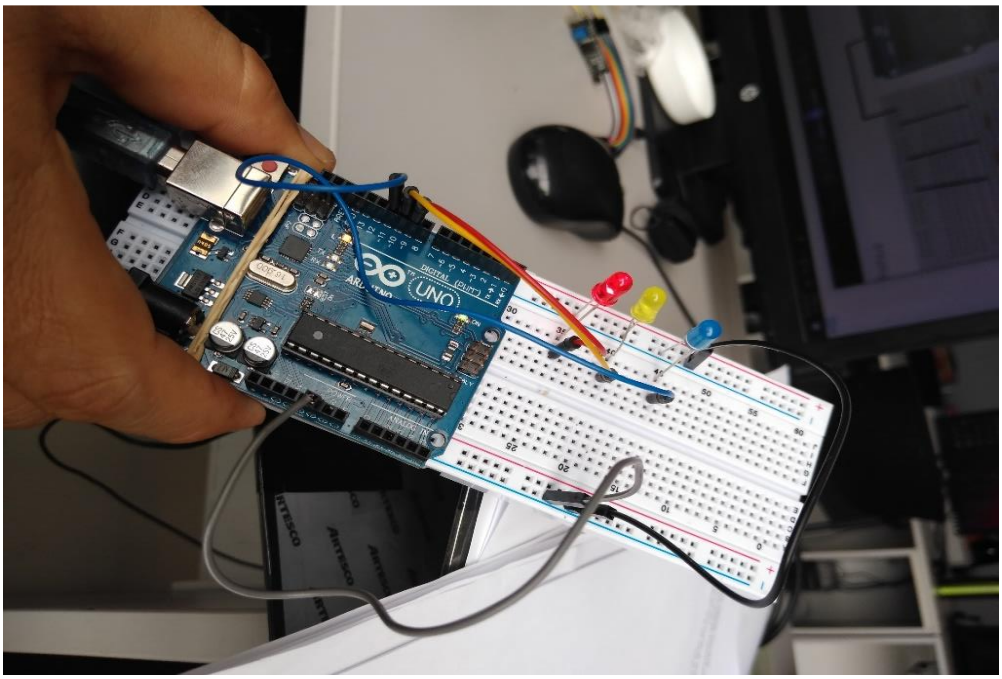

Figura 26: Muestra la orden ejecutada en izquierda (color rojo)
Fuente: Elaboración propia

Figura 27: Secuencia para procesar la escena
Fuente: Elaboración propia

- **Identificar órdenes**

  Para la identificación de las ordenes de movimiento, se deben tener en cuenta la cantidad de pixeles obtenidos como tono de grises (previamente binarizado), por tanto, se puede contabilizar por medio de la cantidad de tonos de grises 1 (o 255) y 0 (o 0), por cada columna de la imagen, lo que a su vez se puede representar por el método estadístico de dispersión, encontrar la línea que representa ese conjunto de valores estadísticos y por tanto encontrar la pendiente de esa línea creada, con respecto a la pendiente (m) se va a evaluar su inclinación, y frente a los diferentes experimentos se considera que el valor de -0.2 o menores indica que va a realizar la orden de ir a la derecha, valores mayores a 0.1 indica que va a realizar órdenes a la izquierda, para cualquier otro caso, indica que no va a realizar orden alguna.

  Cabe resaltar que las órdenes van a ser consideradas como tal, si la órden es recurrente, para este caso se ha considerado con 2 ordenes seguidas se puede considerar que realmente desea enviar una orden, porque puede mirar a un lado e inmediatamente mira a otro lado, este caso no se considera órdenes.

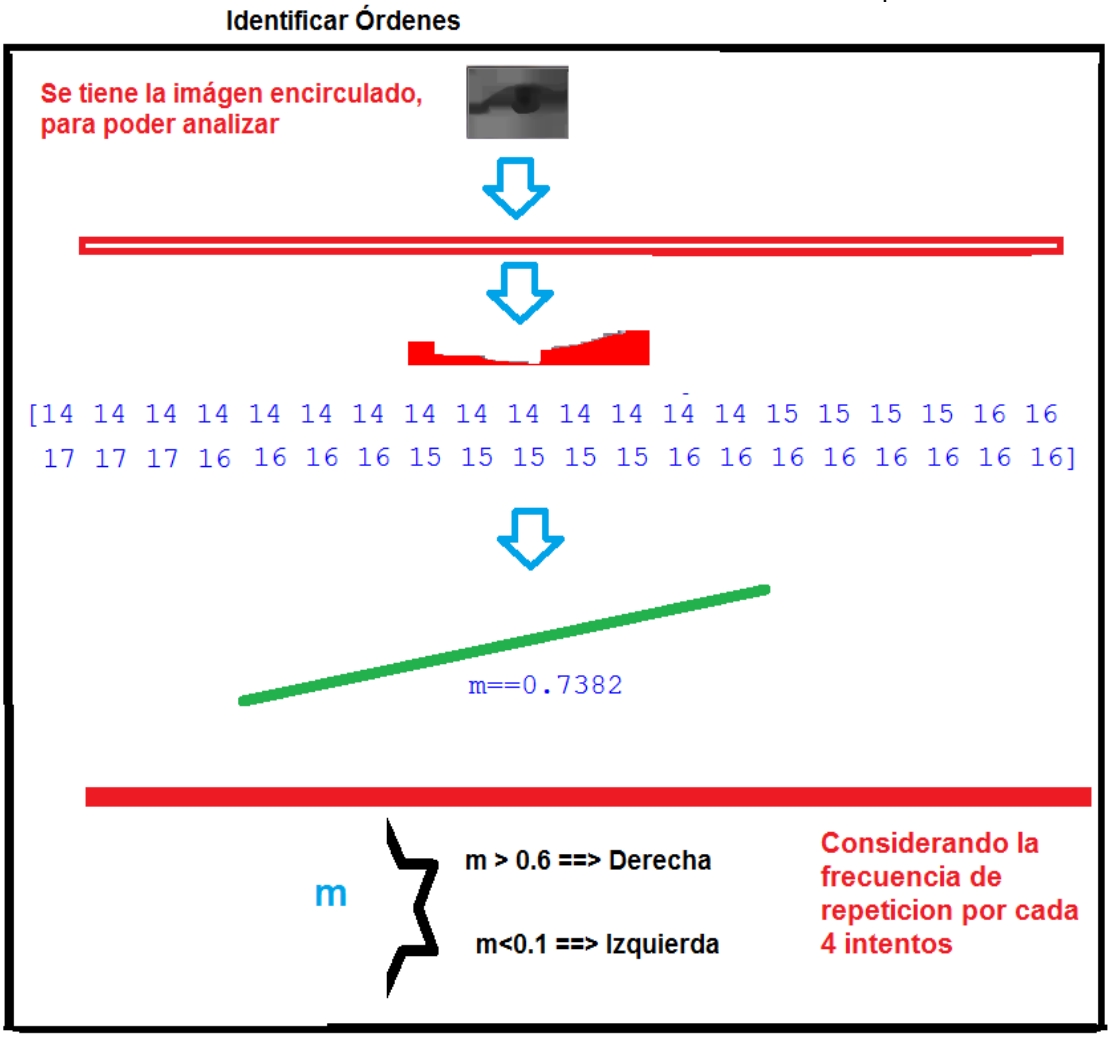

Figura 28: Procesar órdenes
Fuente: Elaboración propia

- **Escuchar órdenes**

Figura 29: Escuchar órdenes
Fuente: Elaboración propia

- **Encender led**

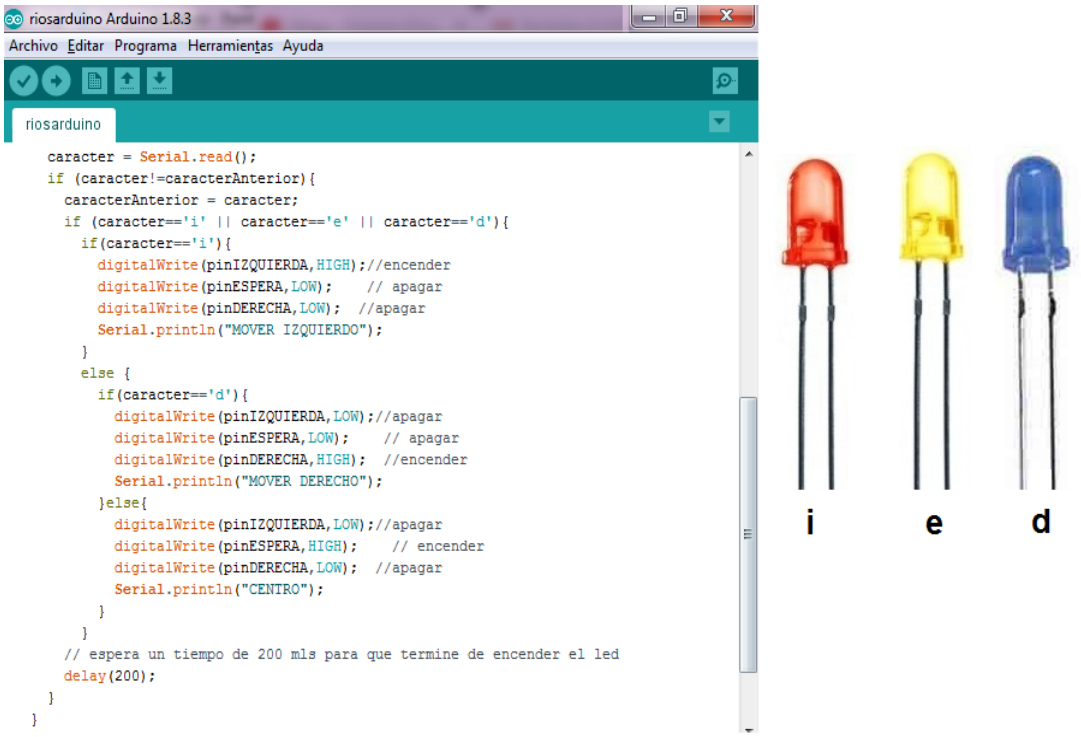

Figura 30: Encender el led según orden
Fuente: Elaboración propia

Vista lógica

Descomposición en susbsistemas

El sistema a desarrollar está compuesto por 2 subsistemas muy bien definidos, subsistema escritorio y subsistema Arduino, dado su origen es necesario la aplicación de otra metodología llamada Ropes, el mismo que se desarrollará posteriormente.

A continuación, se muestra la arquitectura general en donde se refleja la composición y relación entre el actor y los subsistemas.

Subsistema Arduino

El subsistema Arduino, se desarrollará con la estructura que ofrece el fabricante del hardware Arduino como lo define en bloques (Setup() y Loop()).

Subsistema escritorio

El subsistema escritorio se desarrollará en el lenguaje Python de manera intuitiva de programar, a continuación, se muestra un diagrama de la arquitectura y los componentes que lo conforman.

Código fuente representativo

Se analizó y se realizó las pruebas con el siguiente código fuente:

Script para el arduino Uno

Código fuente de Arduino

```
//PIN 9 IZQUIERDA
//PIN 10 ESPERA
//PIN 11 DERECHA
const int pinIZQUIERDA = 9;
const int pinESPERA = 10;
const int pinDERECHA = 11;
unsigned char caracterAnterior = 0;
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  // Habilitar el pin del arduino para que sea salida
  pinMode(pinIZQUIERDA,OUTPUT);
  pinMode(pinESPERA,OUTPUT);
  pinMode(pinDERECHA,OUTPUT);
  caracterAnterior = 'e';
}

void loop() {
  // crea una variable caracter, para recibir el "i" o "e" o "d"
  unsigned char caracter = 0;
  // Evalua si es que ha sido ingresado un valor
  if(Serial.available()>0){
    // va a leer: i : izquierda   e : espera  d : derecha
    caracter = Serial.read();
    if (caracter!=caracterAnterior){
      caracterAnterior = caracter;
      if (caracter=='i' || caracter=='e' || caracter=='d'){
        if(caracter=='i'){
          digitalWrite(pinIZQUIERDA,HIGH);//encender
          digitalWrite(pinESPERA,LOW);    // apagar
          digitalWrite(pinDERECHA,LOW);  //apagar
          Serial.println("MOVER IZQUIERDO");
        }
```

```
      else {
       if(caracter=='d'){
        digitalWrite(pinIZQUIERDA,LOW);//apagar
        digitalWrite(pinESPERA,LOW);    // apagar
        digitalWrite(pinDERECHA,HIGH);  //encender
        Serial.println("MOVER DERECHO");
       }else{
        digitalWrite(pinIZQUIERDA,LOW);//apagar
        digitalWrite(pinESPERA,HIGH);    // encender
        digitalWrite(pinDERECHA,LOW);  //apagar
        Serial.println("CENTRO");
       }
      }
     // espera un tiempo de 200 mls para que termine de encender el led
     delay(200);
    }
  }
  // de existir en cola mas ordenes mientras esperaba los 200 mls
  // los elimina y lee nuevamente en la siguiente pasada del loop
  Serial.write(0x0d);
 }
}
```

Tabla 22. Código fuente del arduino
Fuente: Elaboración propia

Script del código fuente en Python 3.7

Código en Python para analizar la orden

```
import numpy as np
import glob
import cv2
from playsound import playsound
import serial


def pendiente(y):
  x = np.arange(len(y))
  if isnan(np.corrcoef(x, y)[0,1]):
    return 0
  else:
    return np.corrcoef(x, y)[0,1]
def varianza(y):
  return np.std(y)**0.5
```

```
def isnan(value):
  try:
     import math
     return math.isnan(float(value))
  except:
     return False

arduino = serial.Serial('COM3', 9600)
## utiliza harcascade para encontrar el rostro en una escena
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
## utiliza harcascade para encontrar la vista dentro de un rostro
eye_cascade = cv2.CascadeClassifier('haarcascade_lefteye_2splits.xml')
##captura la lectura de la camara 0=primera camara, 1=segunda camara, ....
cap = cv2.VideoCapture(0)
historiaM = []
historiaMLinea = []
estadoAnterior ="Centro"
estadoActual = estadoAnterior
kernel = np.ones((5,5),np.uint8)
while (cap.isOpened()):
##  Lee y lo pasa en formato imagen la toma de ese momento la camara
   ret, img = cap.read()

##  convierte la imagen que esta en rgb a tono de gris
   gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
##  la imagen que esta en tono de gris, se busca si es que existe un rostro
   faces = face_cascade.detectMultiScale(gray)#, 1.3,5,minSize=(20, 20))
##  De existir un rostro (=faces) lo muestra cada unno de ellos
   for (x,y,w,h) in faces:
##     encontrado en la escena un rostro, lo marca con un rectangulo
      #cv2.rectangle(img,(x,y+int(h/4)),(x+w,y+int(h/2)),(62,204,201),2)
      cv2.rectangle(img,(x,y+int(h/4)),(x+w,y+int(h/2)),(62,204,201),2)
##     obtiene una porcion de la esncena en rgb

      if h<200 or w<200:
        continue

      roi_gray = gray[y+int(h/5):y+h+10, x:x+w]
```

```
        roi_color = img[y+int(h/5):y+h+10, x:x+w]
##      busca dentro de la escena de un rostro, donde esta la vista
        eyes = eye_cascade.detectMultiScale(roi_gray,1.1,1)
  ##      muestra cada una de las vistas encontradas en una escena de rostro(=face)
        for (ex,ey,ew,eh)in eyes:

            if ey<roi_gray.shape[0]/3 and ew>30 and eh>20:
              cv2.circle(roi_color,(int(ex+ew/2),int(ey+eh/2)+8),14,(0,255,255),)
              imgGrayencirculado = roi_gray[ey+15:ey+eh, ex:ex+ew].copy()
              imgColorencirculado = roi_color[ey+15:ey+eh,ex:ex+ew]
              cv2.imshow('Color',imgColorencirculado)
              if ew>30:
                imgGrayencirculado = cv2.gaussianBlur(imgGrayencirculado,(5,5),0)

circles=cv2.HoughCircles(imgGrayencirculado,cv2.HOUGH_GRADIENT,1,20,param1
=50,param2=30,minRadius=7,maxRadius=80)

                if circles is not None:
                   circles = np.uint16(np.around(circles))

                   for i in circles[0,:]:
                       if i[2] is not None:
                         if i[2]==0:
                           continue
                         if True:
                           desdeY = 0
                           if int(i[1])-int(i[2])>0:
                             desdeY = int(i[1])-int(i[2])
                           hastaY = imgGrayencirculado.shape[0]
                           if imgGrayencirculado.shape[0]>(int(i[0])+int(i[2])):
                             hastaY = int(i[0])+int(i[2])

                           desdeX = 0
                           if int(i[0])-int(i[2])>0:
                             desdeX = int(i[0])-int(i[2])

                           hastaX = imgGrayencirculado.shape[1]
                           if imgGrayencirculado.shape[1]>(int(i[1])+int(i[2])):
```

```
                    hastaX = int(i[1])+int(i[2])

                histo = imgGrayencirculado[desdeY:hastaY,desdeX:hastaX]
                if histo.shape[1]<18:
                  if len(historiaMLinea)>4:
                    historiaMLinea.pop(0)
                    historiaMLinea.append(0)
                    print("XXXXX",historiaMLinea)
                else:
                  histo = cv2.erode(histo,kernel,iterations = 4)
                  histo = cv2.dilate(histo,kernel,iterations = 4)
                  cv2.imshow('histo1',histo)
                  ret,histo                                                    =
cv2.threshold(histo,50,255,cv2.THRESH_BINARY_INV)
                  histo2 = histo.copy()

                  suma = np.sum(histo2//255,axis=1)
                  if np.sum(suma)==0:
                    continue
                  while suma[0]==0:
                    histo2 = np.delete(histo2,(0),axis=0)
                    suma = np.sum(histo2//255,axis=1)
                  histo2 = np.flip(histo2,axis=0)
                  suma = np.sum(histo2//255,axis=1)
                  while suma[0]==0:
                    histo2 = np.delete(histo2,(0),axis=0)
                    suma = np.sum(histo2//255,axis=1)
                  histo2 = np.flip(histo2,axis=0)

                  suma = np.sum(histo2//255,axis=0)
                  maximo = np.max(suma)
                  for i in range(len(suma)):
                    cv2.line(img, (2*i+10, 50-(maximo - suma[i])), (2*i+10, 50),
(0,0,255))
                    cv2.line(img, (2*i+11, 50-(maximo - suma[i])), (2*i+11, 50),
(0,0,255))

                  m = pendiente(suma)
```

```
                print("m="+str(round(m,4)),suma)

                cv2.line(img, (10, 100-int(35*m)), (70, int(25*m)+100), (0,255,0))
                cv2.putText(img,"m="+str(round(m,4)),(10,150), cv2.FONT_HERSHEY_SIMPLEX, 0.4,(0,0,255),1,cv2.LINE_AA)

                if not ((np.sum(suma[0:int(len(suma)/4)])==0 or np.sum(suma[int(3*len(suma)/4):int(len(suma))])==0) and histo2.shape[0]>10):
                  continue
                historiaMLinea.append(m)
                if len(historiaMLinea)>2:
                  historiaMLinea.pop(0)
                  mp = np.median(historiaMLinea)
                umbralIzquierda = 0.1
                umbralDerecha = -0.2
                if m>umbralIzquierda:
                  historiaM.append(1)
                else:
                   if m<umbralDerecha:
                     historiaM.append(-1)
                   else:
                     historiaM.append(m)

                M = 0
                if len(historiaM)>2:
                  historiaM.pop(0)

                  M = varianza(historiaM)
                  if M==0 and np.median(historiaM)>0:
                    estadoActual ="Izquierda"

                  if M==0 and np.median(historiaM)<0:
                    estadoActual ="Derecha"

                  if M>0.5:
                    estadoActual ="Centro"
                    arduino.write(b'e')
```

```
                                print("median=",np.median(historiaM),"m = ",m, " estado = ",
        estadoActual)
                                if  (estadoActual!=estadoAnterior):
                                   historiaMLinea.clear()
                                   historiaM.clear()
                                   if estadoActual=="Derecha":
                                      pts                                                    =
        np.array([[10,200],[20,180],[20,190],[50,190],[50,210],[20,210],[20,220]], np.int32)
                                      pts = pts.reshape((-1,1,2))
                                      cv2.fillPoly(img, np.int_([pts]), (0, 255, 0))
                                      playsound('derechox.mp3')
                                      arduino.write(b'd')
                                   if estadoActual=="Izquierda":
                                      pts                                                    =
        np.array([[10,190],[40,190],[40,180],[50,200],[40,220],[40,210],[10,210]], np.int32)
                                      pts = pts.reshape((-1,1,2))
                                      cv2.fillPoly(img, np.int_([pts]), (0, 255, 0))
                                      arduino.write(b'i')
                                      playsound('izquierdox.mp3')
                                   if estadoActual=="Centro":
                                      arduino.write(b'e')
            cv2.imshow('img',img)
            k = cv2.waitKey(30) & 0xff
            if k == 27:
               break
        arduino.close()
        cap.release()
cv2.destroyAllWindows()
```

Tabla 23. Código fuente en Python
Fuente: Elaboración propia

Plan de pruebas de integración

Propósito

Este documento describe el plan a seguir para integrar los componentes del sistema involucrados.

Alcance

Este plan de integración abarca los subsistemas Arduino con los leds.

Subsistema

Arduino

Este subsistema consta de un conjunto de leds para mostrar la ordenes realizadas (escuchadas) por parte del computador. Para ello existe una comunicación constante entre el Arduino por los pines digitales el Arduino considerados como emisores que representa los movimientos o las ordenes.

Despliegue

Integración 1

SUBSISTEMA	Funciones
Arduino	• Escuchar órdenes • Encender led
Sistema escritorio	• Procesar la escena • Identificar órdenes

Tabla 24: Integración de sistema con arduino
Fuente: Elaboración propia

Integración 2

SUBSISTEMA	Funciones
Arduino	• Escuchar órdenes • Encender led/apagar led
Sistema escritorio	• Procesar la escena • Enviar órdenes

Tabla 25: Integración con el sistema arduino con el sistema escritorio
Fuente: Elaboración propia

ANEXO 2: HARDWARE ARDUINO PARA EL PROYECTO

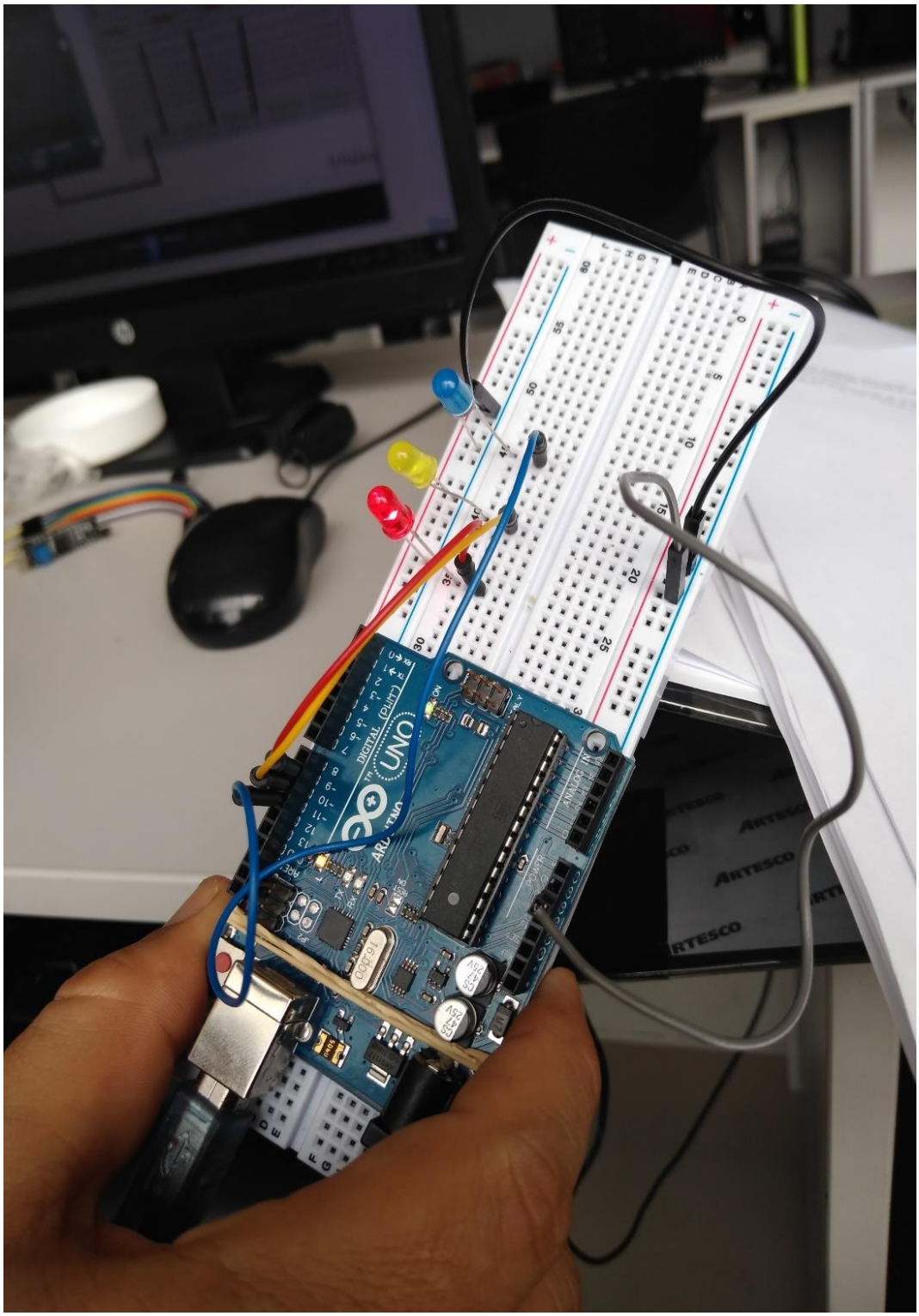

Figura 31: Orden de movimiento para la izquierda realizada
Fuente: Elaboración propia

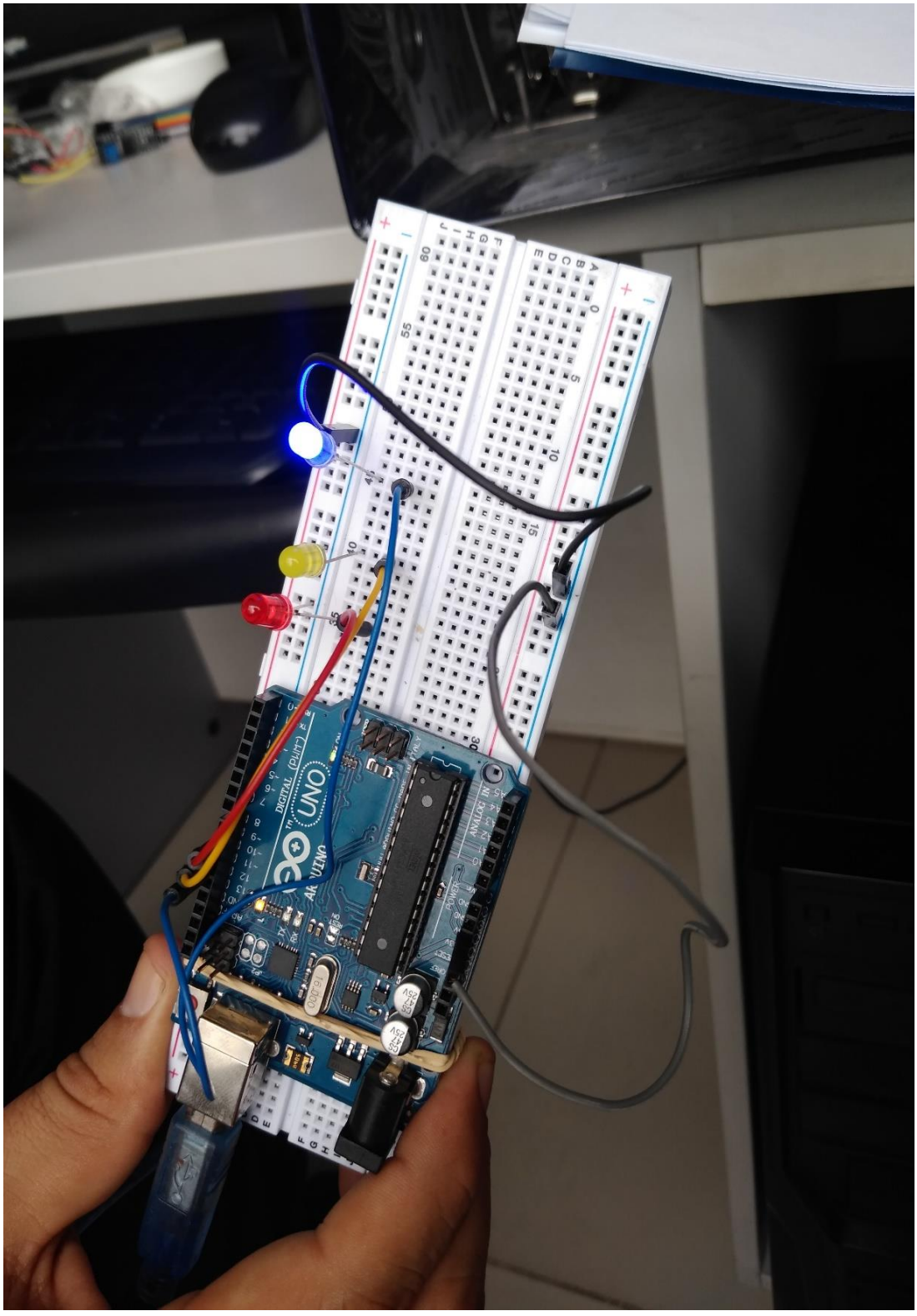

Figura 32: Orden de movimiento para la derecha realizada
Fuente: Elaboración propia

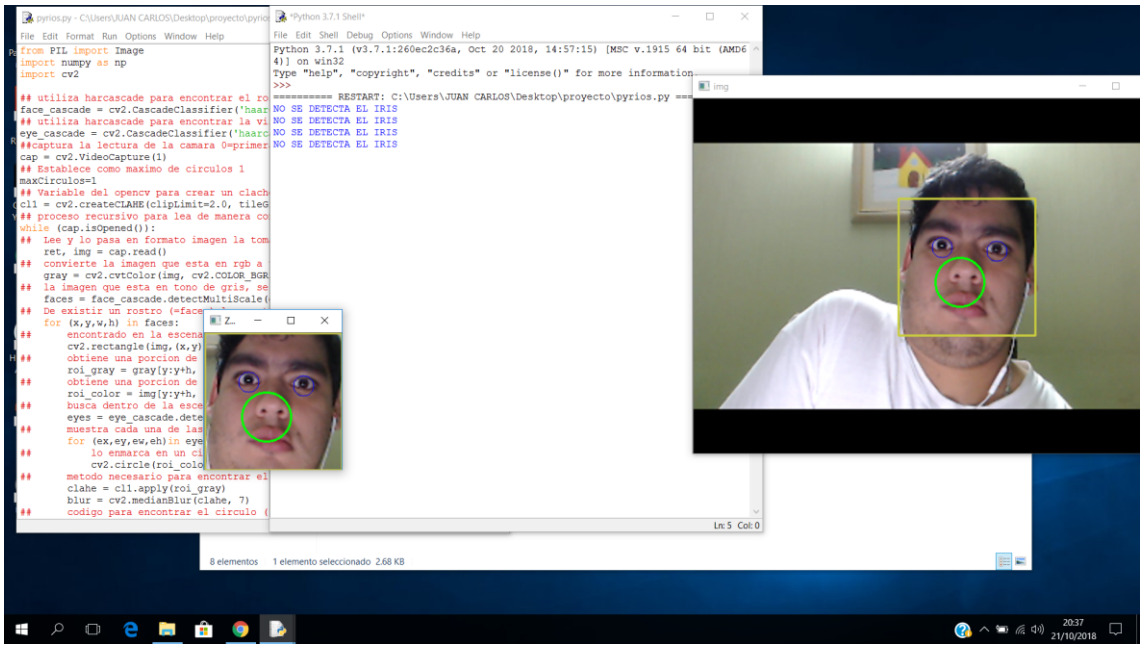

Figura 33: Captura de la escena del rostro-como parte inicial
Fuente: Elaboración propia

ANEXO 3: CODIGO FUENTE DEL SOFTWAREA ARDUINO

: Código fuente del software arduino

riosarduino Arduino 1.8.7

Archivo Editar Programa Herramientas Ayuda

riosarduino

```
//PIN 9 IZQUIERDA
//PIN 10 ESPERA
//PIN 11 DERECHA
const int pinIZQUIERDA = 9;
const int pinESPERA = 10;
const int pinDERECHA = 11;
unsigned char caracterAnterior = 0;
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  // Habilitar el pin del arduino para que sea salida
  pinMode(pinIZQUIERDA,OUTPUT);
  pinMode(pinESPERA,OUTPUT);
  pinMode(pinDERECHA,OUTPUT);
  caracterAnterior = 'e';
}

void loop() {
  // crea una variable caracter, para recibir el "i" o "e" o "d"
  unsigned char caracter = 0;
  // Evalua si es que ha sido ingresado un valor
  if(Serial.available()>0){
    // va a leer: i : izquierda   e : espera  d : derecha
    caracter = Serial.read();
    if (caracter!=caracterAnterior){
      caracterAnterior = caracter;
      if (caracter=='i' || caracter=='e' || caracter=='d'){
        if(caracter=='i'){
          digitalWrite(pinIZQUIERDA,HIGH);//encender
          digitalWrite(pinESPERA,LOW);    // apagar
          digitalWrite(pinDERECHA,LOW);  //apagar
          Serial.println("MOVER IZQUIERDO");
        }
        else {
          if(caracter=='d'){
            digitalWrite(pinIZQUIERDA,LOW);//apagar
            digitalWrite(pinESPERA,LOW);    // apagar
            digitalWrite(pinDERECHA,HIGH);  //encender
            Serial.println("MOVER DERECHO");
          }else{
            digitalWrite(pinIZQUIERDA,LOW);//apagar
            digitalWrite(pinESPERA,HIGH);    // encender
            digitalWrite(pinDERECHA,LOW);  //apagar
            Serial.println("CENTRO");
          }
        }
      // espera un tiempo de 200 mls para que termine de encender el led
      delay(200);
    }
  }
  // de existir en cola mas ordenes mientras esperaba los 200 mls
  // los elimina y lee nuevamente en la siguiente pasada del loop
  Serial.write(0x0d);
 }
}
```

Figura 34: Código fuente en arduino para evidenciar las ordenes
Fuente: Elaboración propia