Estimación de la fuerza ejercida por el brazo en la prueba *tapping-test* utilizando visión artificial

Diego Nicolás Zambrano Velasco

Universidad CESMAG

Facultad de Ingeniería

Programa de Ingeniería Electrónica

San Juan de Pasto

2022

Estimación de la fuerza ejercida por el brazo en la prueba *tapping-test* utilizando visión artificial

Diego Nicolás Zambrano Velasco

Informe Final de Trabajo de Grado presentado ante el Comité Curricular para optar por el título de Ingeniero Electrónico

Asesor:
Mario Fernando Henao Rosero

Universidad CESMAG
Facultad de Ingeniería
Programa de Ingeniería Electrónica
San Juan de Pasto
2022

Nota De Aceptación

Aprobado por el comité curricular del Programa de Ingeniería Electrónica en cumplimiento de los requisitos Exigidos por la Universidad CESMAG, para optar al título de Ingeniero Electrónico.

Mario Henao

_____

Asesor

_____

Camilo Lagos
Jurado 1

_____

Javier Narváez
Jurado 2

Página de Nota Exclusión de Responsabilidad Intelectual

"El pensamiento que se expresa en esta obra es exclusivamente responsabilidad de los autores y no compromete la ideología de la Universidad CESMAG"

Dedicatoria

Este trabajo de grado está dedicado a mi padre, quien me enseñó que el mejor conocimiento que se puede tener es el que se aprende por sí mismo. También está dedicado a mi madre, quien me enseñó que incluso la tarea más grande se puede lograr si se hace un paso a la vez.

Me gustaría agradecer en estas líneas la ayuda que muchas personas y colegas me han prestado durante el proceso de investigación y redacción de este trabajo. En primer lugar, quisiera agradecer a mis padres que me han ayudado y apoyado en todo mi producto, Robert Alexander Zambrano Castro y Ana Julia Velasco Salas, a mis asesores, Mario Henao y Francisco Checa, por haberme orientado en todos los momentos que necesité sus consejos y correcciones.

A todos mis amigos, vecinos y futuros colegas que me ayudaron de una manera desinteresada, gracias infinitas por toda su ayuda y buena voluntad.

Agradecemos a nuestros docentes de la facultad de ingeniería Electrónica, por haber compartido sus conocimientos a lo largo de la preparación de nuestra profesión.

Y en último lugar pero no menos importante a la Universidad CESMAG por haber sido la sede de todo el conocimiento adquirido en estos años recorridos.

Tabla de contenido

Indice de tablas

Índice de figuras

Indice de anexos

Introducción

La prueba de *Tapping test* mide la velocidad segmentaria de una extremidad superior. El desarrollo del ejercicio puede ser sobre una mesa o una superficie plana donde se colocarán las placas de respuesta a las fuerzas aplicadas.

Iniciando con la posición inicial el ejecutante debe colocarse delante de la mesa con los pies ligeramente separados situando su mano *no dominante* sobre la placa de marca y la otra mano sobre uno de los círculos (placa receptora con sensores previamente caracterizados y ya instalados sobre ellas).

El *Tapping test* o test de Golpeo de placas, (TT) fue desarrollado originalmente como una prueba neurofisiológica que evalúa una medida simple de la velocidad y del control motor (Mitrushina, Booney y D'Elia, 1999) y se utiliza en neuropsicología como prueba sensible para el ya conocido daño cerebral (Christianson y Leathem, 2004). Otros autores como Scherer, Bauer, Baum (1997) y Mendo et al., (2011), la consideran como una tarea de ejecución motora y un método de prueba manual con numerosas variaciones (1 a 5 zonas de *tapping*) que posee una alta sensibilidad acerca del control que tienen pacientes con distintos tipos de daño cerebral. Aunque el funcionamiento motor en humanos sea controlado por muchas áreas del cerebro (Mendo et al., 2011).

El desarrollo del ejercicio comienza al sentir una señal de inicio al momento de escuchar el sujeto la señal de comienzo deberá de tocar alternativamente los 2 círculos (placas) un total de 25 veces, cada uno con la mano dominante, tan rápido como se pueda.

La prueba finaliza en el contacto número 25, momento en el cual se detiene el cronómetro, su valoración de prueba se registraría con los segundos y décimas de segundos invertidos en la prueba.

Se busca realizar un sistema el cual automatice el proceso de toma de datos del ejercicio supliendo así el puesto del observador (profesor), ya que así se encuentre un experto evaluando, este personal no podrá estimar la fuerza que se ejerce con exactitud, además existe la varaiable de del error del ser humano tal ejemplo puede evidenciarse con o a la poca exactitud del ojo para poder percibir y discernir las distintas velocidades y aceleraciones. Un sistema que incluya un componente electrónico y un componente de procesamiento de video podrà satisfacer esa

necesidad, cuyo requerimiento será estudiar la efectividad y la estimación de la fuerza ejercida al ejecutar el ejercicio mediante visión artificial.

Esto mediante la ayuda y la efectividad de software y hardware. Sensores de recolección de datos sobre fuerza y en base a velocidad dada por imagen de una cámara de alta velocidad instalada sobre la superficie del trabajo, se buscara estimar la fuerza con la que el brazo del ejecutor está realizando el *tapping test*.

La implementación del proyecto podría tener una gran acogida en el campo de educación física y medicina ya que en sus vidas académicas rige la idea de evaluar procesos físicos, psicológicos y mentales. Al momento de querer tener valores reales de actividades las cuales lo demande sería excelente contar con un sistema el cual la toma y estudie sus datos en base en la observación de una visión artificial.

1. El problema de investigación

1.1 Objeto o tema de investigación

Sistema de estimación de la fuerza ejercida por el brazo en la prueba *tapping test* por medio de visión artificial.

1.2 Línea de investigación

Sistemas de automatización y control.

El área de sistemas de automatización y control de la Universidad CESMAG desarrolla procesos investigativos orientados al modelamiento, simulación, diseño, desarrollo y evaluación de algoritmos de control, sistemas de control, sistemas inteligentes, control de procesos industriales, sistemas embebidos, acondicionamiento y procesamiento de señales, robótica, domótica e inteligencia artificial (Programa de Ingenieria Electronica, Universidad CESMAG, 2015) (Energía, n.d.).

1.3 Sub-línea de investigación

Biometría: “La investigaciónestá ligada a esta sub-línea de biometría ya que hace referencia a la medición y análisis de características fisiológicas externas de las hojas de la planta y también en su comportamiento (cambio de forma) ’’ (Electrónica, 2015).

Parte de la investigación en cuanto al análisis de imágenes se desarrolló en la Universidad CESMAG, en el programa de ingeniería electrónica, el cual es un programa académico que busca explotar la creatividad de sus estudiantes en resolución de problemas, necesidades o requerimientos del entorno regional, nacional e internacional a través de la idea del diseño, la implementación y la intervención de los dispositivos electrónicos de acuerdo con el objeto de estudio.

1.4 Planteamiento o descripción del problema

Las pruebas de *tapping test* consisten en la medición del tiempo sobre la ejecución del ejercicio de golpes en las placas, más no otras variables tales como la fuerza ejercida por el brazo al tocar las placas. Esta información podría ser de utilidad al profesional de la educación Física y a cualquier persona, ya que así podría evaluar en mayor medida el desarrollo psicomotor del ejecutor.

El término fiabilidad es descrita en el diccionario de la RAE como "probabilidad de buen funcionamiento de algo". Por tanto, extendiendo el significado a sistemas, se dice que la fiabilidad de un sistema es la probabilidad de que ese sistema funcione o desarrolle una cierta función, bajo condiciones fijadas y durante un período determinado.

La capacidad de procesamiento y retención que tienen los seres humanos resulta limitada frente a una máquina; por supuesto que tenemos un pensamiento no lineal que redunda en la creatividad, pero es un hecho que las máquinas pueden procesar más y más rápido, aunque linealmente.

La problemática se convierte en el deseo de saber que tan satisfactorio se puede desempeñar un sistema de visión artificial en la estimación de la fuerza que ejerce el brazo al tocar las placas en la prueba *tapping* test. Se planteo la comparación entre los datos que arrojaran sensores de presión ubicados en las placas, dispuestas para la ejecución de la prueba. La estimación basada en el movimiento del brazo (velocidad y aceleración), cuando se es analizada utilizando un sistema de V.A el cual estudia también las curvas de desempeño y rendimiento.

1.5 Formulación del problema

¿Cuál es el error que se obtiene al estimar la fuerza ejercida por el brazo en la prueba *tapping-test* mediante un sistema basado en la visión artificial?

1.6 Objetivos

1.6.1 Objetivo general

Evaluar el error en la estimación de la fuerza ejercida por el brazo en la prueba *tapping test* mediante un sistema de visión artificial.

1.6.2 Objetivos específicos.

- Establecer los requisitos que debe tener el sistema basado en visión artificial para la estimación de la fuerza ejercida en la prueba de *tapping test*.
- Implementar un prototipo de sistema para estimar la fuerza ejercida en la prueba de *tapping test* utilizando visión artificial.
- Implementar un modelo de estimación de la fuerza a partir de datos de movimiento y características físicas del ejecutor.
- Evaluar el desempeño del prototipo construido.

1.7 Justificación

En este momento la evaluación de la prueba *tapping-test* la hace un experto humano quien podria tener problemas de objetividad asociados con condiciones fisiológicas como el cansancio, condiciones psicológicas, emocionales, las cuales son normales en un ser humano. De este modo si se desarrolla un sistema automatizado que evalúe dicha prueba o ayude al evaluador a aumentar esa objetividad mejorará la calidad en la evaluación.

La razón de implementar un sistema electrónico en la prueba *tapping test* será tener digitalizado la medición de variables como fuerza, velocidad y aceleración para tener una mejor objetividad de cada una de las personas a la cual se realice la prueba y así obtener mejores datos de evaluación en la prueba.

Así mismo facilitar el proceso de evaluación para el docente encargado de realizar la prueba.

La implementación de este prototipo vendría de gran ayuda al programa de educación física el cual permitiría analizar con mayor rapidez y eficacia a las pruebas que se realicen a estudiantes para evaluar su desarrollo psicomotriz.

Teniendo en cuenta una adecuada estimación de fuerza obtenida por los sensores y la cámara de alta velocidad y datos que el software genere se puede recurrir a una forma más precisa de evaluar los resultados y de reducir la variabilidad , con los cuales se podrá determinar los inconvenientes y ventajas que tengan las personas a las cuales se les realice la prueba, teniendo en cuenta los resultados arrojados por la prueba determinar un plan específico en los casos en que las pruebas arrojen resultados por abajo del promedio, como también determinar que estudiantes tienen una habilidad en algún deporte relacionado con las extremidades superiores del cuerpo humano.

1.8 Delimitación

El proyecto pretendió desarrollar un prototipo que estime la fuerza a partir del análisis de imágenes y mediciones del cuerpo del ejecutor en la prueba de *tapping test*, con la ayuda de un sistema automatizado se verificó el error en la estimación ejercida por el brazo respecto a los datos recolectados en la prueba donde se desarrolló un prototipo cuya validación fue desarrollada con la ayuda de voluntarios del programa de ingeniería electrónica.

2. Marco teorico - contextual

2.1 Contexto

La investigación es realizada en la Universidad CESMAG de la Ciudad de San Juan de Pasto, Nariño, Colombia, como proyecto de trabajo de grado para optar al título de ingeniero electrónico.

El trabajo de campo de la investigación se realizó en el municipio de San Juan de Pasto en el departamento de Nariño, en las instalaciones de la Institución educativa UNIVERSIDAD CESMAG proveyendo de los espacioes necesaria para la realización de poder implementar el sistema basado en visión artificial para la obtención de la estimación de la fuerza en las pruebas de Tapping Test,

Figura 1. Instalaciones de La universidad CESMAG.

2.2 Antecedentes

2.2.1 Reconocimiento de la dinámica de acción en la esgrima utilizando señales multimodales.

Debido a la naturaleza altamente competitiva de los deportes, los atletas y entrenadores están ansiosos por adaptar y verificar prácticamente los nuevos métodos de entrenamiento, así como las tecnologías innovadoras para el apoyo al entrenamiento. El uso de la tecnología en los deportes ha mejorado el diseño de instalaciones y equipos deportivos en una amplia gama de disciplinas deportivas. El análisis biomecánico de los movimientos deportivos permite comprender mejor cómo se comporta el cuerpo humano durante diversas acciones deportivas y, posteriormente, desarrollar mejores métodos de entrenamiento, así como reducir el riesgo de lesiones. Cada disciplina deportiva desarrolló un conjunto de ejercicios destinados a perfeccionar determinadas habilidades. El empleo de tecnologías modernas permite a los atletas dominar estas habilidades en menos tiempo y aumentar su rendimiento general.

Actualmente, un análisis de las acciones deportivas comienza a jugar un papel crucial en el proceso de formación en varias disciplinas. Su importancia proviene de la posibilidad de brindar retroalimentación relevante. El reconocimiento de acciones es uno de los temas básicos que deben abordar estas herramientas de análisis. Esto incluye tanto la identificación general de diferentes actividades deportivas como las acciones específicas de la disciplina, por ejemplo, brazadas de natación. El reconocimiento de acciones (RA) es un tema de investigación importante con muchas aplicaciones posibles en diversas áreas. Por tanto, ya se han propuesto múltiples métodos. Sin embargo, la literatura dedicada al reconocimiento de acciones tanto generales como deportivas considera solo la detección y clasificación de acciones significativamente diferentes entre sí. Aunque algunas de las bases de datos de RA más populares contienen acciones, que son similares entre sí hasta cierto punto, todas tienen trayectorias distintas y son fácilmente reconocibles por los humanos. En algunas disciplinas deportivas, sin embargo, las acciones pueden distinguirse principalmente por la dinámica del movimiento más que por la trayectoria. Estas acciones pueden ser difíciles de distinguir incluso para un ser humano, especialmente para una persona que no está familiarizada con una disciplina deportiva en particular.

En este documento se propone un enfoque novedoso para el reconocimiento de acciones deportivas. Lo cual consiste en el uso de la dinámica en el análisis de patrones de movimiento similares. Se propone descriptores de movimiento informativos y se demuestra con esto que se puede modelar mejor la dinámica y que los clasificadores que los utilizan logran resultados superiores en comparación con los algoritmos de última generación. Se propone un método para el reconocimiento de la dinámica de acción, que se basa en datos acelero métricos, características delas articulaciones del esqueleto y un descriptor novedoso basado en la profundidad, además de emplear una red para la fusión de las características de datos multimodales. Se considera 6 tipos de acciones dinámicas, que incluyen 4 tipos de estocadas de esgrima, cada una con una trayectoria promedio muy similar, pero una dinámica de movimiento considerablemente diferente. Se Pondrá a disposición del público un conjunto de datos dedicado con muestras de juego de pies de esgrima de 10 tiradores, incluidos datos de profundidad, esqueléticos e inerciales. Se mostrará que en nuestro conjunto de datos de juego de pies de esgrima, el método propuesto supera a los métodos actuales de vanguardia para el reconocimiento general de acciones.

Este artículo se relaciona con el documento que se desarrolló ya que aborda el problema del reconocimiento de acciones dinámicas en un entorno. Donde se considera el reconocimiento del básico en el movimiento de brazos sobre una base y datos de aceleración. Se descubrió que la dinámica de acción es un tema importante en el reconocimiento de acciones deportivas y difiere significativamente de los problemas típicos de reconocimiento de acciones. Las acciones dinámicas con trayectorias muy similares pero diferentes dinámicas de movimiento requieren métodos novedosos para una clasificación adecuada. Se presentó el primer conjunto de datos públicos con tales acciones: el conjunto de datos de juego de pies de esgrima. Este conjunto de datos multimodal incluye datos recopilados con sensores piezoeléctricos y dinámica delanálisis de datos mediante V.A (visión artificial), Para modelar correctamente los cambios de movimiento en acciones dinámicas, Se propuso 3 conjuntos de características, basadas en los datos del esqueleto (dinámica de articulaciones), los datos del esqueleto y la profundidad (imágenes de seguimiento local) y los datos inerciales (características de aceleración). Se evalúa todos los conjuntos de características por separado y se utiliza la red neuronal MLP para la fusión. Los resultados experimentales indican que la fusión de diferentes características mejora significativamente la precisión del reconocimiento. En el conjunto de datos de juego de brazos,

los algoritmos propuestos superan a los métodos más avanzados para el reconocimiento de acciones generales. Se creerá que el análisis de la dinámica de acciones es una dirección novedosa e interesante en el área de reconocimiento de acciones y que el conjunto de datos presentado será útil para futuras investigaciones sobre este tema (Malawski & Kwolek, 2018).

2.2.2 Inteligencia computacional para diagnóstico de entrenamiento cualitativo.

La tecnología de entrenamiento, los usables y los exer-games pueden proporcionar retroalimentación cuantitativa basada en la actividad medida, pero hay poca evidencia de retroalimentación cualitativa efectiva para ayudar a mejorar la técnica. Para lograr una retroalimentación cualitativa personalizada, demostramos un prototipo de prueba de concepto que combina kinesiología e inteligencia computacional que podría ayudar a mejorar la técnica de swing de tenis utilizando datos de movimiento de tenis tridimensionales (3D) adquiridos de video multi-cámara. El etiquetado de datos de expertos se basó en la reproducción virtual de figuras de palitos en 3D. Diversos criterios de evaluación para principiantes y aquellos con niveles de habilidad intermedios y escenarios de entrenamiento configurables combinados con una variedad de golpes de tenis (22 revés y 21 golpes de derecha), incluida una buena técnica y errores comunes. Se transfirió un conjunto de reglas de entrenamiento seleccionadas (CR) a módulos de evaluación adaptativa capaces de aprender de los datos, desarrollar sus estructuras internas y producir comentarios personalizados autónomos, incluidas señales verbales sobre la reproducción 3D de la cámara virtual y un informe de progreso al final de la sesión. El prototipo demostró una evaluación autónoma sobre datos futuros basada en el aprendizaje de ejemplos anteriores, alineada con el nivel de habilidad, escenarios de entrenamiento flexibles y RC. La retroalimentación diagnóstica intuitiva generada consistió en elementos de seguridad y rendimiento para la técnica de swing de tenis, donde cada muestra de swing se comparó con el experto. Para los aspectos de seguridad del ancho relativo de oscilación, el prototipo mostró una evaluación mejorada (del 81% al 91%) cuando se tienen en cuenta las partes ocluidas de la pelvis. Este estudio ha mostrado una prueba de concepto para una retroalimentación cualitativa personalizada. La próxima generación de sistemas de entrenamiento y ejercicio aumentados podrá ayudar a mejorar las técnicas específicas de la disciplina deportiva del usuario final. Al aprender de pequeños conjuntos de datos etiquetados por expertos y dichos sistemas podrán

adaptarse y proporcionar evaluaciones autónomas e intuitivas personalizadas y comentarios de diagnóstico alineados con un programa de entrenamiento específico y requisitos de contexto.

El sistema presentado demostró que era posible generar retroalimentación consistente en elementos de seguridad y rendimiento para ayudar al aprendizaje motor o mejorar técnicas complejas específicas del deporte como el swing de tenis. Como una contribución científica multidisciplinaria e interdisciplinaria, el sistema de prueba de concepto demostrado fue capaz de capturar conocimientos de expertos en un modelo informático y reproducir diagnósticos cualitativos en datos de movimiento nunca antes vistos similares al razonamiento humano (> 80%). La retroalimentación diagnóstica automatizada demostrada se asoció con la evaluación subjetiva del experto y la retroalimentación que contiene reglas abstractas y descriptivas de sentido común asociadas con el rendimiento y la seguridad, características críticas de la secuencia de movimientos humanos específicos del deporte que también pueden operar con sistemas basados en IA, y errores comunes y señales de atención. Para el aspecto de seguridad del ancho relativo de un swing de tenis, el prototipo demostró una evaluación mejorada (del 81% al 91%) al tener en cuenta las partes ocluidas de la pelvis en los mismos datos, lo que como evidencia también se considera un descubrimiento de conocimiento de datos que pueden informar la práctica del coaching. Para abordar la necesidad de aprendizaje adaptativo de por vida (para deportes como el tenis), los modelos de sistema y clasificador tienen propiedades que CI considera como aprendizaje adaptativo, evolutivo y de por vida a partir de datos de entrenamiento inicialmente pequeños.

Este artículo se relaciona con el documento que se desarrolló ya que se plantea con la hipótesis de que esto mejorara potencialmente. La técnica del usuario final más que utilizar la comunicación de entrenamiento existente de solo los resultados cuantitativos de los movimientos observados. El análisis cualitativo generado por máquinas para la retroalimentación del entrenamiento de patrones de movimiento complejos para mejorar la función motora, el control y la técnica. Es comúnmente aplicable a una variedad de disciplinas deportivas y escenarios de rehabilitación, la base tecnológica subyacente cubre los dispositivos de captura de movimiento existentes y futuros capaces de generar un conjunto de datos cada vez más grande como dispositivos portátiles, cámaras deportivas / de acción, teléfonos móviles, sensores conectados a equipos deportivos, controladores de juegos y de ejercicios, dispositivos de electroencefalograma o interfaces de computadora cerebral, dispositivos de electromiografía, dispositivos de

rehabilitación funcional, prótesis inteligentes y diseño de control de exoesqueleto (Bačić & Hume, 2018).

2.2.3 Detección y análisis de movimiento usando visión artificial.

Un sistema basado en conocimiento (KBS) es una herramienta software (o agente inteligente) con una cantidad significativa de conocimiento presente en una forma declarativa explicita. Las actuales KBS han alcanzado niveles de madurez importantes estando presentes en diversos entornos.

El agente reacciona de acuerdo a la información que perciba del exterior, estando compuesto internamente por el programa agente (función que implementa la transformación de secuencias percibidas en acciones) y una arquitectura (mecanismo con el cual acepta la información del exterior y genera las acciones, generalmente se asocia a un computador) (Sanabria, John, y Archila, 2011).

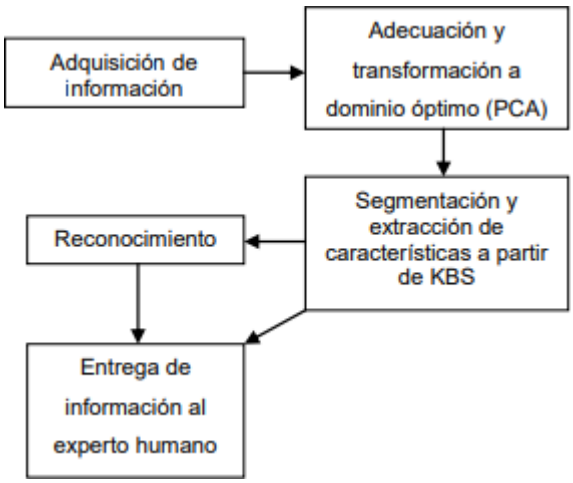


Figura 2. Elemento de un sistema de detección de movimiento

PCA: Proceso de teledetección.

KBS: Proceso con el cual se puede clasificar información obtenida en las imágenes.

Se adapta el protocolo para el análisis de imagen, antes de extraer cualquier información proveniente se deben considerar los defectos provenientes de las diversas fuentes entre los cuales es posible encontrar:

- Defectos en el sistema (ruido e interferencias que podrían estar ensuciando la información de la imagen) de adquisición lo cual originará imágenes defectuosas.
- La pérdida de datos durante la captura de la misma.
- La corrupción de información en los medios de distribución.
- Las distorsiones de escala en las imágenes, las fallas o variaciones en la iluminación empleada. Siendo necesario el acondicionamiento9 de la(s) imagen(es). Posteriormente al acondicionamiento se definen los criterios a resaltar, involucrando técnicas de mejoramiento.
- Redistribución de la información estadística de la imagen.
- La manipulación de las características del espacio de color (Intensidad, Saturación y Pureza).
- Filtrado.

El protocolo anterior se implementó para una buena obtención, detección y un buen análisis de imágenes en movimiento.

2.2.4 Sistema de reconstrucción de movimiento inteligente para swing de golf.

En la última década, el golf ha despertado un gran interés en la gente y el número de jugadores de golf ha aumentado significativamente. Por lo tanto, cómo entrenar a un golfista para hacer un swing perfecto ha atraído la atención de una extensa investigación. Entre estas investigaciones, el paso más importante es capturar y reconstruir el movimiento del swing de una manera transportable y no intrusiva. Restringido por el desarrollo de los actuales dispositivos de imágenes de profundidad, el movimiento de balanceo capturado inicial puede no ser aceptable debido a oclusiones y mezcla de partes del cuerpo. En este artículo, para restaurar la información de movimiento de la auto oclusión y reconstruir el swing de golf en 3D a partir de datos de baja resolución, se propone un algoritmo de reconstrucción de swing de golf basado en el modelo de Red Bayesiana Dinámica (DBN) para aumentar la precisión de captura considerando las similitudes espaciales y temporales del swing. Entre diferentes golfistas. Se presenta un sistema de reconstrucción de movimiento inteligente para swing de golf, SMRG, basado en el modelo DBN con un popular dispositivo de imágenes de profundidad, Kinect, como dispositivo de captura. Los resultados experimentales han demostrado que el sistema propuesto puede lograr una precisión de reconstrucción comparable al sistema comercial de subtítulos de movimiento

óptico (OMocap) y un mejor rendimiento que los algoritmos de modificación de última generación que utilizan información de profundidad

Los experimentos han demostrado que el sistema de visión artificial si puede construir con buena calidad.

El trabajo futuro será incorporar partes del cuerpo como (muñeca y columna vertebral) y el sistema de análisis de los parámetros cinemáticos generados por el sistema. También se está considerando la reconstrucción y el análisis de otros movimientos regulares utilizando dispositivos de imágenes de profundidad utilizando visión artificial (Lv et al., 2017).

La reconstrucción del movimiento es necesario ya que gracias a este se busca el patrón del seguimiento en cualquier movimiento, lo cual puede llegar a estimar un valor físico sobre una secuencia de movimientos los cuales se van a obtener de un material visual

2.2.5 Detección y análisis de movimiento mediante visión artificial

El ser humano percibe las imágenes a través de la retina, la cual se comporta como una cámara fotográfica sensible a la luz, mediante la cual, es posible obtener información visual proveniente del entorno.

Dentro del ojo se encuentran minúsculos receptores denominados bastones y conos agrupados en estructuras, existiendo dentro de estas estructuras, secciones sensibles a cada uno de los tres colores básicos del RGB. Una vez los receptores realizan su labor, la información recibida es enviada al cerebro por medio del nervio óptico, para ser procesada por el cerebro. Ya dentro del cerebro la información es analizada por millones de neuronas, altamente especializadas, distribuidas en capas. Teniendo por función la primera capa detectar los límites ‘bordes’ en las imágenes, siendo estos la manifestación de cambios importantes en la intensidad o iluminación de la imagen, mientras las otras capas detectan contornos y demás comportamientos presentes en la imagen como es el caso de profundidad y movimiento

Con el surgimiento de las computadoras en la década de los cincuentas, se genera al interior de la comunidad científica el interrogante relacionado con la posibilidad de enseñar a las computadoras a realizar tareas comúnmente asociadas con la inteligencia humana, entre las cuales se encuentra la capacidad de resolver problemas, comprender lenguajes o analizar información visual Dando origen con ello a una disciplina orientada a emular la inteligencia humana, denominada ‘inteligencia artificial’ y una miríada de aplicaciones y campos de investigación científica.

La visión artificial se realiza de manera semejante al proceso asociado con la visión humana siendo su insumo básico de entrada una imagen obtenida mediante una cámara. Siendo la

Imagen un arreglo o matriz de puntos correspondientes estos al valor de una función bidimensional f(x, y) donde x y y son coordenadas espaciales y el valor de f, representa la intensidad o brillo de la imagen, en el caso de imágenes blanco y negro y para imágenes a color, corresponde a la combinación de tres arreglos en el modelo de color aditivo denominado RGB.

Previamente a la clasificación de la información se sugiere realizar la transformación de la imagen a un dominio óptimo para el trabajo a desarrollar. Considerando como premisa la eliminación de información redundante, sin que ello implique pérdidas de información de utilidad, siendo de común utilización el análisis de componentes principales (PCA) y sus variantes.

Basándose la lógica del proceso en la toma de n muestras, representadas mediante m variables, cumpliendo (1) y (2).

$$\mathbf{m < n} \quad \mathbf{(1)}$$

$$\mathbf{l <= min(n,m)} \quad \mathbf{(2)}$$

Buscando reducir la cantidad de información a procesar, del resultado de la reducción se obtienen los ***l*** componentes principales.

A partir de los ***l*** componentes se realiza la combinación lineal de las variables originales representada por (3) siendo ellas independientes entre sí.

$$\mathbf{X} = \sum_{\mathbf{a=1}}^{\mathbf{l}} \mathbf{taPa} + \mathbf{E}\ (\mathbf{3})$$

Donde ***ta*** corresponde a los pesos; información acerca de la relación entre las muestras, ***Pa*** informa la relación entre las variables y la matriz, ***E*** corresponde al error ocasionado.

La descomposición se realiza mediante los vectores propios presentes en la matriz de covarianza calculados según (4) ó (5).

$$\mathbf{COV(X)} = \frac{x^1 - x}{n-1}\ (4)$$

$$\mathbf{COV(X)}\boldsymbol{PA} = \mathbf{YaPa} \sum_{\mathbf{a=1}}^{\mathbf{m}} \mathbf{Ya}\ (\mathbf{5})$$

Siendo ***Ya*** el valor asociado al valor propio ***Pa***.

Obteniéndose una nueva representación de la información, expresada mediante (6)

$$\mathbf{Ta = XPa\ (6)}$$

En esta nueva representación cerca del noventa (90%) por ciento de la información presente en la imagen está contenida en los tres (3) primeros componentes principales. Adicional al análisis con PCA se integran sistemas basados en conocimiento (KBS) para realizar las segmentación, clasificación y análisis de la información contenida en la imagen a partir de características (Sanabria, John y Archila, 2011).

Este artículo se relaciona con el documento que se desarrolló ya que aporta mucha investigación de varios autores donde afirman que la visión artificial si es posible en varios puntos ingenieriles fisicos donde se puede analizar, procesar y automatizar basándose en la lógica del proceso en la toma de n muestras, representadas mediante m variables.

Las herramientas de visión artificial más modernas se basan en entrenar a un algoritmo de inteligencia artificial para que sea capaz de reconocer imágenes de objetos, formas, caras, patrones y estructuras más complejas, incluso acciones que implican movimiento.

Hay diferentes grados de complejidad: desde simples sensores de visión con una capacidad de procesamiento muy limitada, hasta cámaras con gran definición y algoritmos de análisis más avanzados. Todo depende del uso que se le vaya a dar, la cantidad de imágenes que tenga que analizar por segundo o la calidad de las propias imágenes.

2.2.6 Implementación de un sistema de visión para la localización bidimensional con estimación de velocidad.

Presenta una metodología para la implementación de un sistema de localización de vehículos, en dos dimensiones, por medio de visión artificial, se plantea la utilización de una red neuronal para mapear la posición de coordenadas de la cámara de un marco ubicado en el suelo; así como la estimación de la velocidad en la que se aplica, de manera practica un diferenciador por medio de modos deslizantes de orden superior. Un sistema de localización planteado se puede utilizar con cualquier tipo de vehículo autónomo que pueda desplazarse dentro del área de alcance de la cámara.

En el método tradicionalmente utilizado para transformar coordenadas de la cámara a un marco en el mundo real, es necesario realizar un buen modelo de la cámara, esto es, conocer los parámetros intrínsecos de la cámara, Una vez que se conoce el modelo de la cámara, se aplican

operaciones matriciales, en donde estas operaciones transforman las coordenadas de pixeles a coordenadas de distancia, eliminando además los errores debidos a la distorsión radial.

Para el correcto mapeo de las coordenadas X − Y de la imagen hacia coordenadas x − y en el plano del suelo, se desarrolló una red neuronal con una capa de entrada correspondiente a 2 señales (posición X –Y.

La derivada de una función en el tiempo representa la variación de dicha función en cada intervalo infinitesimal de tiempo. La aproximación más simple de una derivada, en tiempo discreto, se obtiene calculando la diferencia entre la muestra actual y la anterior, y dividiendo entre el período de muestreo en ese instante de tiempo.

$$\frac{df(t)}{dt} = \frac{f(t) - f(t-T)}{T} \quad \textbf{(1)}$$

La desventaja de utilizar este diferenciador en el sistema de localización consiste en que la cámara llega a muestrear en ocasiones consecutivas un mismo valor de posición, aunque el objeto se esté moviendo (debido a que la cámara muestrea continuamente y almacena datos en el búfer) y el diferenciador estará interpretando en esos momentos una diferencia de cero. Otra desventaja en este tipo de diferenciador es que los valores de la derivada instantánea van a depender mucho del período de muestreo, por lo que pueden resultar valores muy irreales. Una solución al usar la diferencia hacia atrás en este tipo de sistemas es la utilización de un filtro pasa-bajas.

El filtro discreto que se utilizará se define como:

$$G(z) = \frac{(1 - e^{-\alpha * t}) z^{-1}}{1 - e^{-\alpha * t} z^{-1}} \quad \textbf{(2)}$$

Al filtrar la derivada discreta con la función de transferencia (2) se obtiene una muy aceptable aproximación de la derivada. El único problema consiste en seleccionar adecuadamente la constante de tiempo α, pues con un valor muy pequeño la derivada filtrada estará por debajo del valor instantáneo que debería tener, además de que la señal se va desfasando (atrasando con respecto a la derivada real). Si se selecciona un valor muy alto de α, la derivada filtrada conservará mucho del ruido de la señal que se está tratando de filtrar.

La técnica de modos deslizantes está relacionada a la teoría de sistemas de estructura variable. Los modos deslizantes estándar son robustos y de gran precisión, sin embargo presentan

una característica indeseada que es el llamado efecto *chattering* causado por la alternancia del control.

Este artículo se relaciona con el documento que se desarrolló ya que tiene en cuentala idea de utilizar métodos de visión artificial con la finalidad de estimar el velocidad vehicular y solucionar el reto entre vehículos provocado por la ubicación en las cámaras de video los cuales fueron varias las metodologías utilizadas activamente en el análisis de imágenes y videos aplicadas a la adquisición de parámetros para comparar la efectividad en el mapeo.

2.3 Enunciados de los supuestos teóricos

2.3.1 Prueba de tapping test.

Pertenece a una Batería de test Eurofit (realizada para niños para que participen con regularidad y placer de las actividades físicas y deportivas), El objetivo principal del test, es medir la velocidad gestual o segmentaria de la extremidad superior. La prueba *tapping-test* consiste en tocar alternativamente los 2 círculos un total de 25 veces cada uno con la mano dominante, tan rápido como sea posible. (Jos, 2019).

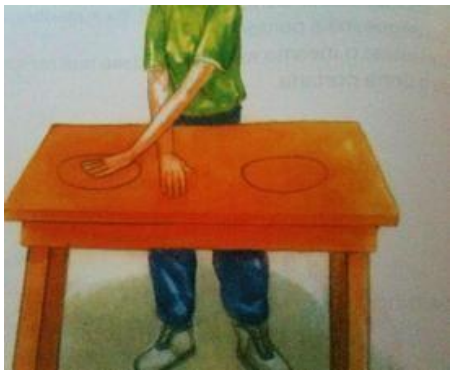


Figura 3. Prueba tapping test.

2.3.2 Laboratorios de análisis de movimientos

El análisis del movimiento humano ha interesado a muchos desde tiempos atrás, llevando al desarrollo de diferentes métodos para su estudio. Gracias al cambio tecnológico y la introducción de los sistemas computacionales, se ha logrado obtener novedosos sistemas para análisis del movimiento humano disponibles en la actualidad. Una de las áreas más desarrolladas del estudio del movimiento, corresponde al estudio de la marcha humana normal y sus alteraciones causadas por ejemplo: por problemas patológicos asociados con el movimiento, lesiones que alteren el aparato locomotor, como afecciones del sistema articular o de origen neurológico, que comprometen el buen funcionamiento de las extremidades(Mariana Haro, 2014). Estos sistemas contribuyen a la investigación, ya que generan grandes cantidades de datos que pueden ser útiles para estudios biomecánicos posteriores.

2.3.3 Procesamiento digital de imágenes

El procesamiento digital de imágenes ha adquirido, en años recientes, un papel importante en las tecnologías de la información y el cómputo. Actualmente, es la base de una creciente variedad de aplicaciones que incluyen diagnosis médica, percepción remota, exploración espacial, visión por computadora, etc.

Como resultado directo de la reducción en el precio de las computadoras, el procesamiento digital de imágenes actualmente se puede efectuar (aunque con ciertas limitantes) en una computadora personal. El presente trabajo proporciona una breve introducción a esta área de la informática y de la computación haciendo referencia a las principales teorías y métodos; asimismo, se muestran los resultados de estas teorías y métodos cuando se aplican a imágenes dadas.

Al conjunto de técnicas y procesos para descubrir o hacer resaltar información contenida en una imagen usando como herramienta principal una computadora se le conoce como procesamiento digital de imágenes (PDI). Hoy en día, el PDI es un área de investigación muy específica en computación y está muy relacionada con el procesamiento digital de señales. Esta relación estriba en el hecho de que en esencia el PDI es una forma muy especial del procesamiento digital de señales en dos o tres dimensiones(Rojas Montes, 2012)

2.3.4 Presión

La presión es una magnitud física escalar representada con el símbolo *p*, que designa una proyección de fuerza ejercida de manera perpendicular sobre una superficie, por unidad de superficie.

La presión relaciona una fuerza de acción continua y una superficie sobre la cual actúa, por lo cual se mide en el *Sistema Internacional* (SI) en pascales (*Pa*), equivalentes cada uno a un newton (*N*) de fuerza actuando sobre un metro cuadrado (*m2*) de superficie. En el sistema inglés, en cambio, se prefiere la medida de libras (*pounds*) por pulgadas (*inches*).

Se define la presión, como la cantidad de fuerza ejercida por unidad de área.

$P=F/A$

Así que, para crear una gran cantidad de presión, puedes ejercer una fuerza muy grande o ejercer una fuerza sobre unas áreas pequeñas (o ambas).(Molina Ruiz, 2010).

2.3.5 Prototipo basado en sensores inerciales para el segmento en la actividad física.

En la actualidad muchas disciplinas utilizan sistemas de análisis de movimiento para capturar la postura y el movimiento del cuerpo humano como lo son la medicina, el deporte y la industria, para los puestos de trabajo, ya que requieren el uso de registros gonio métrico de diferentes articulaciones que casi siempre son obtenidos durante movimientos dinámicos. El seguimiento de la actividad física se puede realizar mediante diferentes tecnologías como lo son los sistemas ópticos, mecánicos, magnéticos e inerciales, siendo estos últimos portátiles y menos robustos considerándose la mejor manera de hacer un análisis gonio métrico del movimiento humano, ya que estos no requieren de un lugar controlado.

Los sistemas inerciales utilizan unos pequeños sensores (normalmente acelerómetros, giroscopios y magnetómetros en tres ejes ortogonales) que recogen información sobre la aceleración y la velocidad angular del sensor y no se necesitan mecanismos externos ni unidades estacionarias, como receptores o cámaras para la recolección de datos y como en el caso de los sistemas ópticos que están acoplados a unos trajes especiales y son necesarios lugares controlados en donde se ubica también una unidad transmisora(Tamime, 2019).

2.3.6 Fuerza.

La fuerza es una magnitud vectorial que mide la razón de cambio de momento lineal entre dos partículas o sistemas de partículas. Según una definición clásica, fuerza es todo agente capaz de modificar la cantidad de movimiento o la forma de los materiales. No deben confundirse con los conceptos de esfuerzo o de energía.

En el Sistema Internacional de Unidades, la unidad de medida de la fuerza es el newton que se representa con el símbolo N, nombrada así en reconocimiento a Isaac Newton por su aportación a la física, especialmente a la mecánica clásica. El newton es una unidad derivada del Sistema Internacional de Unidades que se define como la fuerza necesaria para proporcionar una aceleración de 1 m/s^2 a un objeto de 1 kg de masa.

2.3.7 Sensor piezoeléctrico.

Es un dispositivo que utiliza el efecto piezoeléctrico para medir presión, aceleración, tensión o fuera; transformando las lecturas en señales eléctricas que exhibe una disminución en la resistencia con un aumento de la fuerza aplicada a la superficie activa. Su sensibilidad a la fuerza es optimizado para su uso en contacto humano control de dispositivos electrónicos, una celda de carga o un medidor de tensión.

Características:

- Voltaje de funcionamiento:3.3 V o 5 V
- Corriente de trabajo:< 1mA"
- Cuenta con 2 pines de conexión separados 0,1’’
- Tamaño: 4,5cm x 4,5cm
- Disco cerámico piezoeléctrico

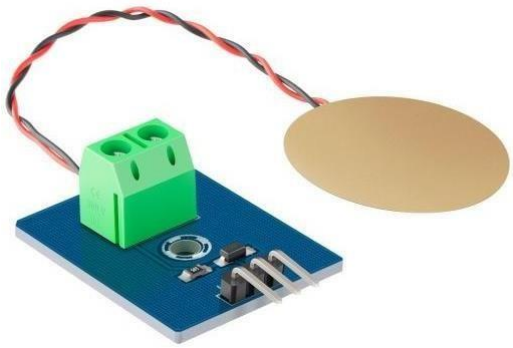


Figura 4. Sensor Piezoeléctrico

2.4 Definición de conceptos

2.4.1 Error relativo.

El porcentaje de error es la magnitud de la diferencia entre un valor exacto y uno aproximado dividida por la magnitud del valor exacto por 100 casos (tiene forma de porcentaje). Básicamente, esta medida te permite ver qué tan lejos está un valor aproximado de uno exacto a través de un porcentaje del valor exacto. El error puede deberse al método de medición (herramienta o error humano) o a las aproximaciones que se usan en el cálculo.

$$\frac{v\ real - v\ aproximado}{v\ real}\ x100\%$$

Valor real: valor tomado en la implementación de la visión aritificial.

Valor aproximado: es el cociente que se obtiene al dividir el valor absoluto entre el valor matemático exacto

2.4.2 Estimación del error.

Evaluar el error en la fuerza ejercida por el brazo en la prueba de ***tapping test,*** a partir de datos obtenidos mediante visión artificial y sensores de presión Un mismo estimador ofrece distintos valores para distintas muestras del mismo tamaño extraídas de la misma población. Por lo tanto, se debera tener una medida de la variabilidad del estimador respecto del parámetro que se trata de estimar. Esta variabilidad se mide en términos de la desviación estándar del estimador, la cual recibe el nombre del error estándar.

$$\frac{\sigma x}{\sqrt{n}} = \sigma^{\bar{X}}$$

n=el número de la población evaluada

σx= parámetro del sensor implementado

2.4.3 Presión.

Magnitud que se define como la derivada de la fuerza con respecto al área. Logo Cuando la fuerza que se aplica es normal y uniformemente distribuida sobre una superficie, la magnitud de presión se obtiene dividiendo la fuerza aplicada sobre el área correspondiente.

$$\frac{dF}{dA} = P$$

Cuando la fuerza que se aplica es normal y uniformemente distribuida sobre una superficie, la magnitud de presión se obtiene dividiendo la fuerza aplicada sobre el área correspondiente.

$$\frac{F}{A} = P$$

P: es la presión en Pa.

F: es la fuerza en N

A: es el área en m²

La presión será calculada del brazo de la persona que este efectuando el ejercicio, esto se hará con una comparación entre cámara aplicando visión artificial y con el sensor resistivo.

2.4.4 Fuerza.

En física, la fuerza es una magnitud vectorial que mide la intensidad del intercambio de momento lineal entre dos cuerpos. En el Sistema Internacional de Unidades (SI), el hecho de definir la fuerza a partir de la masa y la aceleración (magnitud en la que intervienen longitud y tiempo), conlleva a que la fuerza sea una magnitud derivada.

La unidad de medida de fuerza es el newton que se representa con el símbolo: N, nombrada así en reconocimiento a Isaac Newton por su aportación a la física. El newton es una unidad derivada del SI que se define como la fuerza necesaria para proporcionar una aceleración de 1 m/s^2 a un objeto de 1 kg de masa.

Empujar, arrastrar, sujetar, tirar, atraer. Todas estas palabras describen la acción de un cuerpo sobre otro, y en física describe a ellas con un solo término que es la fuerza. Así observando que se obtiene fuerzas por las deformaciones o los cambios de velocidad que producen estas fuerzas en los cuerpos.

$$F = (m)\ (a)$$

m : masa

a : aceleración

La fuerza será calculada del brazo de la persona que este efectuando el ejercicio, esto se hizo con una comparación entre cámara aplicando visión artificial y con el sensor resistivo.

2.4.5 Velocidad.

La velocidad es una magnitud física que expresa la relación entre el espacio recorrido por un objeto, el tiempo empleado para ello y su dirección. La palabra proviene de latín *velocĭtas*, *velocitātis*.

Debido a que la velocidad también considera la dirección en que se produce el desplazamiento de un objeto, es considerada una magnitud de carácter vectorial.

Así, la velocidad implica el cambio de posición de un objeto en el espacio dentro de determinada cantidad de tiempo, es decir, la rapidez, más la dirección en que se produce dicho movimiento. De allí que velocidad y rapidez no sean lo mismo.

Su unidad en el Sistema Internacional de Unidades es el metro por segundo (m/s), e incluye la dirección del desplazamiento.

$$V = \frac{d}{t}$$

d : distancia

t : tiempo

La velocidad será calculada del brazo de la persona que este efectuando el ejercicio, esto se hará con una comparación entre cámara aplicando visión artificial y con el sensor resistivo.

2.4.6 Aceleración.

Es el nombre que le damos a cualquier proceso en donde la velocidad cambia. Como la velocidad es una rapidez y una dirección, solo hay dos maneras para que se acelere; cambia la rapidez o cambia la dirección (o cambia ambas).

$$vf = vi + a\Delta t$$

vf: velocidad final

vi: velocidad inicial

Δ : Aceleración constante, a

La aceleracion será calculada del brazo de la persona que este efectuando el ejercicio, esto se hará con una comparación entre cámara aplicando visión artificial y con el sensor resistivo.

2.5 Hipótesis

2.5.1 Hipótesis de investigación.

El error en la estimación de la fuerza obtenido por el sistema de visión artificial es inferior al 20%.

2.5.2 Hipótesis nula

El error en la estimación de la fuerza obtenido por el sistema de visión artificial no es inferior al 20%.

2.5.3 Hipótesis Alternativa

El error esta entre un 30 y 50%.

3. Metodología

3.1 Enfoque

En el enfoque se esta relacionando en lo cuantitativo, con datos cuantitativos o cuantificables, lo cual conlleva a obtener propiedades, características que posee el objeto de estudio. El trabajo se desarrolla bajo un enfoque cuantitativo y no cualitativo, en el que las variables que alimentan el sistema son de forma numérica por ende los resultados obtenidos determinan el error porcentual de este prototipo.

3.2 Paradigma

Esta investigación tiene un paradigma neopositivista debido a que la filosofía del racionalismo y la del empirismo se deducen teóricamente y se comprueban en la práctica.

Donde se comparará los valores de fuerza obtenidos con los reales, por medio de las evaluaciones físicas de la prueba *tapping test*.

3.3 Método

3.3.1 Método Científico.

Se utiliza el método científico como método descriptivo para el desarrollo de la recolección de información en la prueba *Tapping test* utilizando una cámara de alta velocidad.

- Observación: La recolección de los Resultados obtenidos en pruebas *Tapping Test* Basandose en el tiempo de respuesta de él sistema automatizado.
- Hipótesis: Se puede obtener datos de la fuerza ejercida por el brazo en la prueba de *tapping test*, con un bajo error utilizando un sistema a base de visión artificial.
- Experimentación: Se compara los datos de la fuerza ejercida por el brazo mediante sensores de presión. Serán comparados con la estimación hecha por un sistema de visión artificial.
- Teoría: La fuerza ejercida por el brazo en la prueba de *tapping test* depende de la masa del brazo y de la aceleración de los movimientos. Se puede estimar el movimiento del brazo en la prueba de *tapping test* usando visión artificial.

- Conclusiones: Se espera evaluar y concluir después de las pruebas de experimentación.

3.4 Tipo de investigación

El proyecto de investigación: Estimación de la fuerza ejercida por el brazo en la prueba *tapping test*, Mediante una cámara de alta velocidad; se considera de tipo investigación descriptiva porque se va a caracterizar el desempeño de un prototipo para estimar la fuerza a partir de un análisis de error en dicha estimación.

3.5 Diseño de la investigación

Tiene un enfoque cuasi-experimental. Ya que se manipulará variables de estudio. Las cuales no se controlan como es la velocidad y la aceleración que ejecuta la prueba, donde se pretende estimar la fuerza ejercida por el brazo en la prueba *tapping-test* por medio de visión artificial con una población, en este tipo de investigación se pretende encontrar el porcentaje de error de la prueba al momento de ser ejecutada.

X=RG1 x 01

Y=RG2 - 02

-X : Sistema de medida a base de visión artificial.

-Y : Medición con sensores de presión.

-RG1 : Población (Estudiantes).

-RG2 : Población (Estudiantes).

-01 : Valor de la fuerza obtenido bajo visión artificial (X).

-02 : Medición de fuerza manuel(sensores y pesos). (Y).

3.6 Universo

Conjunto formado por voluntarios del programa de ingeniería electrónica, para estimar la fuerza ejercida por el brazo en la prueba *tapping-test* utilizando visión artificial. Este universo es finito ya que es cuantificable y accesible.

3.7 Muestra

Es una muestra de tipo probabilístico ya que se trabajará con voluntarios del programa de ingeniería electrónica de manera aleatoria dentro de una determinada población, escogidos por medio del asesor encargado del proyecto, en este caso pueden ser un determinado curso del programa de ingeniería.

3.8 Técnicas de recolección de información

Se va a organizar los datos de acuerdo al orden de una caracterización previamnte realizada por el monitor de la prueba, se tendrán en cuenta variables tales como, aceleración, velocidad pesos, velocidades medias, tiempos totales y tiempos por toque.

3.8.1 Validez de la técnica.

La técnica de recolección de información es válida, ya que, el estudio del movimiento a través del procesamiento de imagen ha sido utilziado en problemáticas similares, tales como las cámaras de seguridad de transito, análisis de comportamiento humano en los aeropuertos, etc... Se contará con el respaldo de distintos softwares validos para este tipo de investigación, estos son: PYTHON, IDE Y MATLAB, DB BROSER SQLITE los cuales su función sera tomar, convertir datos capturados por una cámara que convertiran aquellos resultados en datos de contexto físico y cuantitativo los cuales se analizaràn, se guardaràn y se proyectaràn de una manera agradable y amable al usuario en un entorno de evaluación tal como es el de la fuerza ya que una implemetancion de visión artificial en la prueba tappin test mejorara de una forma prolija esta actividad fisica.

3.8.2 Confiabilidad técnica.

Se podrá estimar el posicionamineto de los brazos de manera confiable para la prueba tapping test cuando se llega a utilizar una cámara con la posibilidad de manejar velocidades iguales o superiores a los 240 fps.

Gracias a las tecnologías para la captación de lo visual a lo digital, las cámaras se han ido reinventado e incursionando en nuevas formas para implementaciones de nuevas técnicas, Teniendo en cuenta lo anterior se dispone de las siguientes características en la toma de video.

Vídeo
4K60, FOV amplio

Relación de aspecto de 16:9

4K	FOV Amplio FOV SuperView	60, 30, 24 fps 30, 24 fps
2,7K	FOV Amplio FOV SuperView FOV Lineal	120, 60, 30, 24 fps 60, 30, 24 fps 60, 30, 24 fps
1080p	FOV Amplio FOV SuperView FOV Lineal	240, 120, 60, 30, 24 fps 120, 60, 30, 24 fps 120, 60, 30, 24 fps
720p	FOV Amplio FOV Lineal	240, 60 fps 60 fps

Figura 5. Caracterización de fps por resolución de video.

3.9 Instrumentos de recolección de la información

La recolección de la información obtenida se alamcenara en bases de datos las cuales tendrán la operación de caracterizar al ejecutor y de almacenar datos de la prueba del mismo ejecturo.

Las tablas dispondrán de nombres, códigos, pesos, identificaciónes, velocidades y tiempos

3.9.1 Base de datos.

El programa de Python está conectado con visual studio donde se ejecuta la programación y al momento de correr el código como resultado arroja una dirección ip http://127.0.0.1:8000/admin/ donde entra a una interfaz a una interfaz que funciona como almacén de data sobre información personal como la información de los sensores obtenida antes y después de haber realizado cada prueba de tapping test, lo cual quiere decir, que va a guardar grandes cantidades de información de forma organizada para poder encontrarla y utilizarla de manera fácil y ordenada. Por el encargado de las pruebas TT.

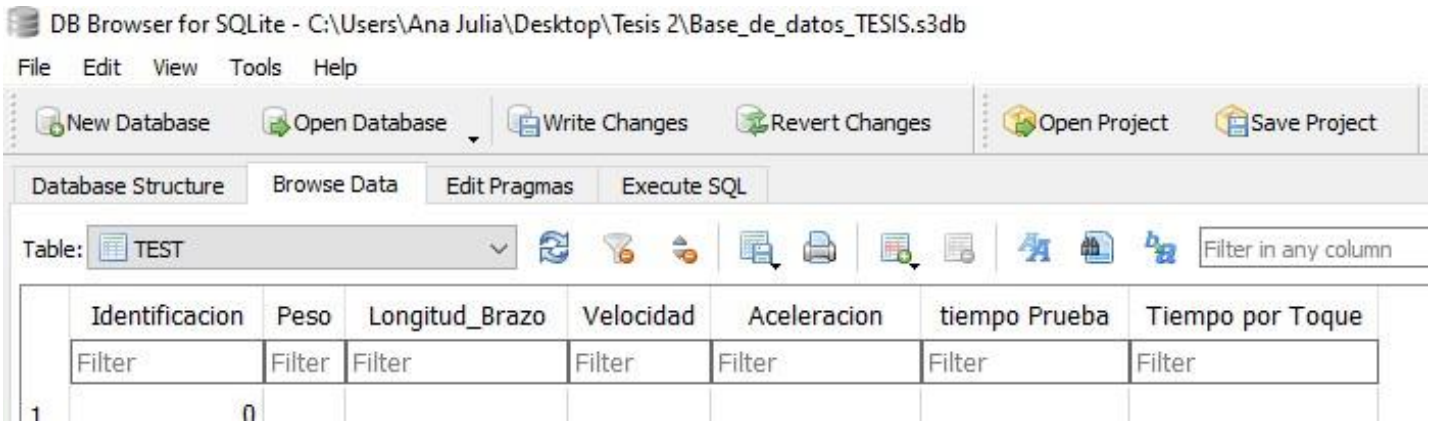


Figura 6. Base de datos.

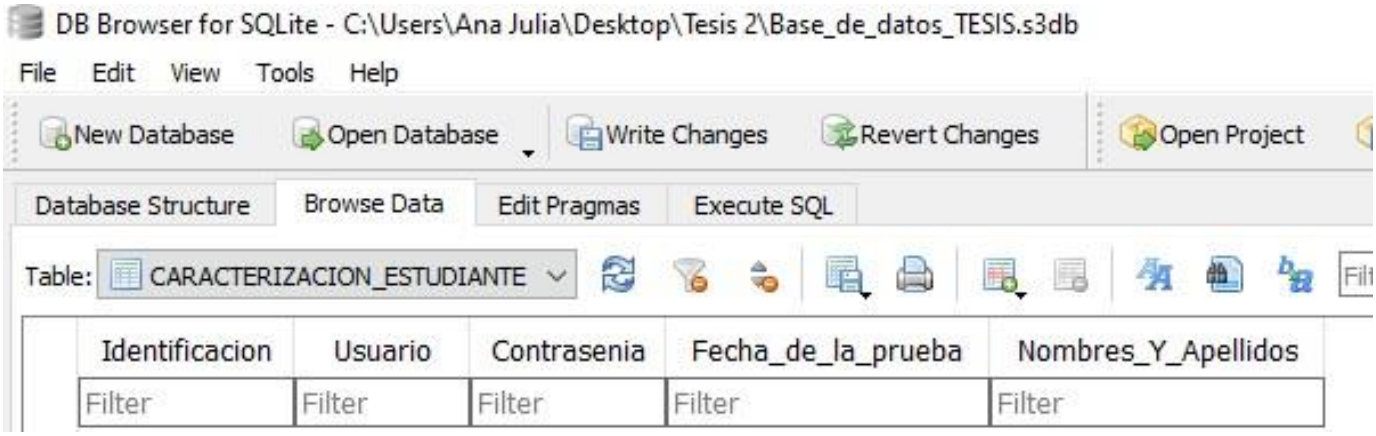


Figura 7. Base de datos2.

Se evidencian 2 bases de datos, la primera donde se aplicara los resultados de las pruebas tapping test y la segunda; donde se realiza la caracterización de los estudiantes o personas a realizar la prueba,

4. Resultados

A continuación, se detalla el diseño experimental para abordar los objetivos propuestos. De igual forma, en paralelo se muestra el diseño del sistema de monitoreo electrónico basado en visión artificial para el diagnóstico de la fuerza en las pruebas tapping test.

4.1 Establecer los requisitos que debe tener el sistema basado en visión artificial para la estimación de la fuerza ejercida en la prueba *tapping test*

4.1.1 Requerimientos de Hardware.

Para poder entablar un entorno el cual se confroma de una parte física y una parte virtual, lo primero que se debe tener en cuenta son los requerimientos que se quiere buscar para ambos términos los cuales tendrán la finalidad de ofrecer unos siertos servicios para unas necesidades definidas.A continuación se mostraran los requerimientos de hardware.

- Se necesitará un tablero el cual estará definido por una superficie que a su vez mostrara los círculos de contacto para las manos en la prueba. Debajo de cada circulo de contacto se instalara un sensor el cual nos brindara respuestas físicas variables.

- Se necesitará un dispositivo el cual sea capaz de recibir, procesar y mostrar datos variables los cuales serán obtenidos mediante dispositivos sensoriales.

- Se necesitará una estación de video la cual sea capaz de mostrar el entorno del tablero para así hacer captura del video mientras se este haciendo la prueba.

4.1.2 Requerimientos de Software.

Al igual que el apartado de hardware se dispondrá de herramientas virtuales las cuales ayudaran a un procesamiento de información, un procesamiento de un dispositivo electrónico y la capacida de mostrar la información de una manera simple y concreta al usuario, A continuación se mostraran los requerimientos de software.

- Realizar un firmware el cual nos ayude a obtener, procesar y guardar información obtenida mediante instrumentos sensoriales en la placa de desarrollo ESP32.

- Realizar Algoritmo el cual deberá proesar información videografica mediante la cual se podrá calcular variables físicas tales como la presión, la fuerza o incluso la velocidad,

- Realizar un entorno visual el cual ayudara a mostrar la información al usuario de una manera que se podrá entender y analizar lde una forma intuitiva..

4.2 Implementar un prototipo de sistema para estimar la fuerza ejercida en la prueba tapping test utilizando visión artificial.

Para poder implementar un prototipo el cual estime la fuerza ejercida en la prueba tapping test utilizando visión artificial se deberá designar los requisitos del sistema y después de ello se empesara con el diseño y la construcción de cada tópico para solventar cada reqerimientos. A continuación el diseño y la construcción del hardware.

4.2.1 Diseño de Hardware.

- **Diagrama de bloque.**

En la figura 6 se muestra como principal instrumento un Pc donde están varias aplicaciones instaladas como Python, arduino y visualstudio .Se pretende interconectar el módulo Esp32 con dos sensores de fuerza donde se obtiene los datos y guarde la información en una base de datos en *SQlite* sabiendo que el módulo Esp32 se utiliza de manera inalámbrica por medio de wi-fi donde el pc será una especie de herramienta *Wireless access point*, donde se utiliza una interfaz de usuario por medio de *Python* para desarrollar las programaciones correspondientes al prototipo de estimación de fuerza.

Sensores Piezoeléctrico: Es un pequeño sensor de presión resistivo con una superficie de contacto de 4 mm de diámetro. El sensor varía su resistencia según la presión o fuerza aplicada en el área circular. Cuanto más se presione, menor será su resistencia. Cuando no es presionado su resistencia es superior a 1MΩ. Puede detectar presión desde tan solo 2 gramos y aunque puede soportar hasta 50Kg de presión, es muy delicado

PC (Terminal Computacional): la necesidad de satisfacer con un dispositivo el cual soporte las diferentes inrtafaces para colocar en funcionamiento diferentes algortimos o programas que conllevan todos juntos a realizar un sitema el cual será ocupado como centro operacional de todo el sitema.

- Acepta y almacena la entrada de datos,
- Procesa la entrada de datos
- Genera la salida en un formato requerido

Python servido WI-FI; Se dispondrá de una red host local wi-fi para así contar con una red segura y dedicada solamente a la comunicación de datos a través del módulo WIFI utilizado para esto se creara una red IP para la óptima comunicación sin interferencia alguna sobre todos los datos que se estarían registrando en el sistema Tapping Test.

Se descarga e instala las librerías de wifi.h en IDE ARDUINO y se dispone a digitar los siguientes comandos:

```
#include wifi.h

const char* ssid = "Tapping test";

const char* pass = "1234567890";
```

Donde se deberá incluir el nombre de la red y su correspondiente contraseña.

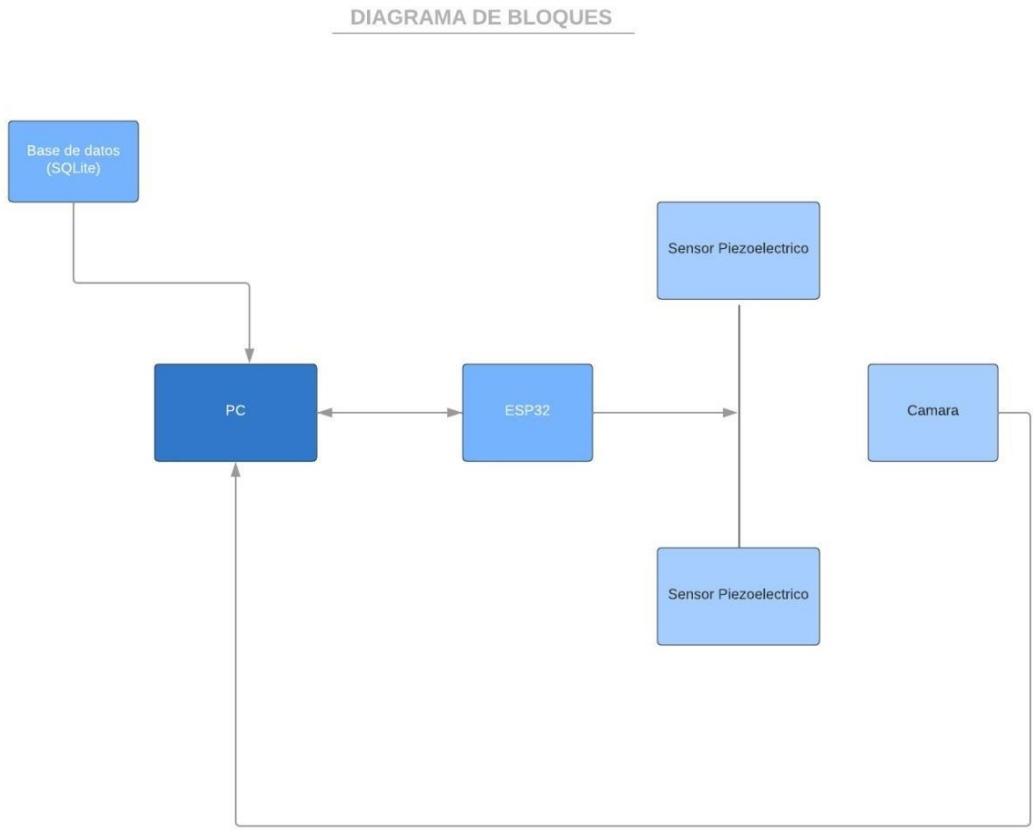


Figura 8. Diagrama de bloques

- **Diagrama de conexiones.**

El módulo Esp32 está conectado a una alimentación de 5v donde tiene una salida digital-análoga conectada a un divisor de voltaje con dos resistencias una de 680 k ohm y otra de 22 k ohm y se conecta a un amplificador operacional LM358N conectado a una resistencia y un capacitor cerámico de 0.01uF y una resistencia de 3.3k ohm con una entrada del sensor piezoeléctrico. Los pines a utilizar son A0, A1, 1 alimentacion de 5v, 2 alimnetaciones de 3.3v 4GNDS, SCL y SDA, las utlimas 2 daran coneccion a una pantalla lsd la cual dara comunicaciónal usuario sobre la inrteaz del programa cargado en ESP32.

En el principio se realiza pruebas con amplificador operaciónal para poder captar asi una señal mas tratada pero se llega a la conslusion que es mejor por medio de un divisor de voltaje a la salida de la señal del sensor, esto da como consecuencia una señal sin rudio y un poco mas amplificada.

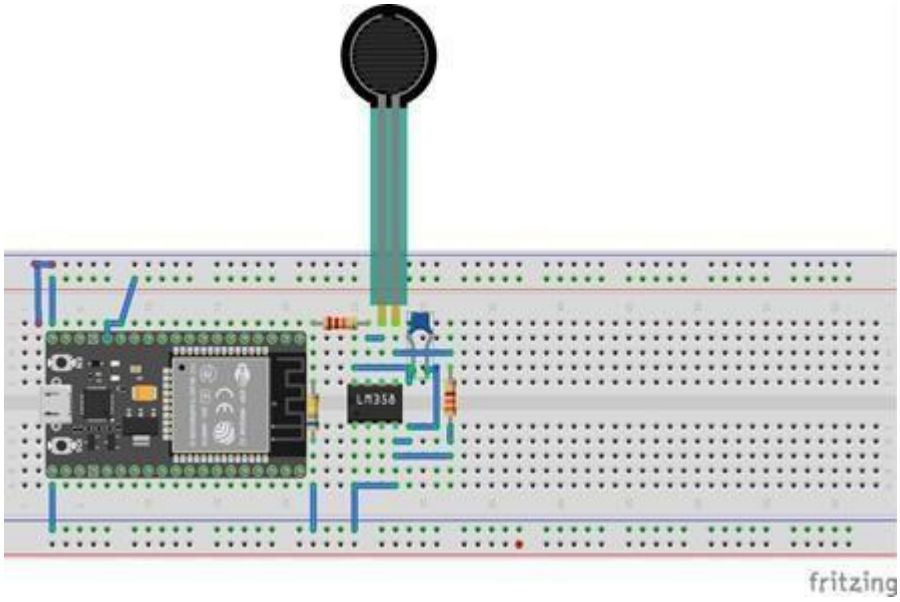


Figura 9. Diagrama de conexión.

- **Módulo Esp-32 Wi-fi.**

El **ESP-WROOM-32** es un módulo que integra Wi-Fi y Bluetooth, ideal para desarrollar e interconectar diferentes lenguajes de programación como es el arduino y lenguaje C++. La integración de Bluetooth, y Wi-Fi permite una amplia gama de aplicaciones, el uso de Wi-Fi permite una comunicación para conectarse a una red LAN y a través de un Modem Router conexión a Internet, mientras que el Bluetooth nos permite conectarse directamente a otro dispositivo como un celular, además de hacer uso de toda la información sobre proyectos y librerías disponibles en internet.

La desventaja de utilizar esta placa es que no se encuentra mucha información sobre la misma, también hay muchas variaciones y distribuidores los cuales presentan el producto con diferentes características las cuales podrían afectar el rendimiento de sensores, dispositivos, redes, información. Etc.

Figura 10. ESP-WROOM-32.

4.2.2 Construccion Hardware

Se diseñó un prototipo el cual contiene comunicación WIFI, comunicación USB en tiempo real lo cual facilitara supervisar el rendimiento de sensores y/o actualizar el mismo sistema. dispone de pantalla LSD el cual muestra la inicialización del sistema y notifica los valores de cada sensor. Cableado interno con 2 circuitos de potencia, interruptor de ON/OFF

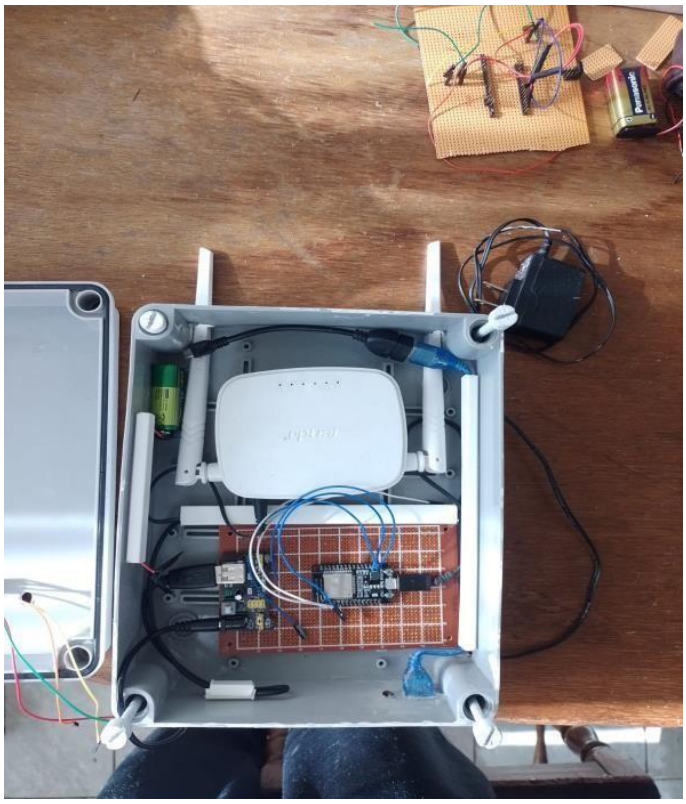


Figura 11. Cara interna del prototipo hardware1.

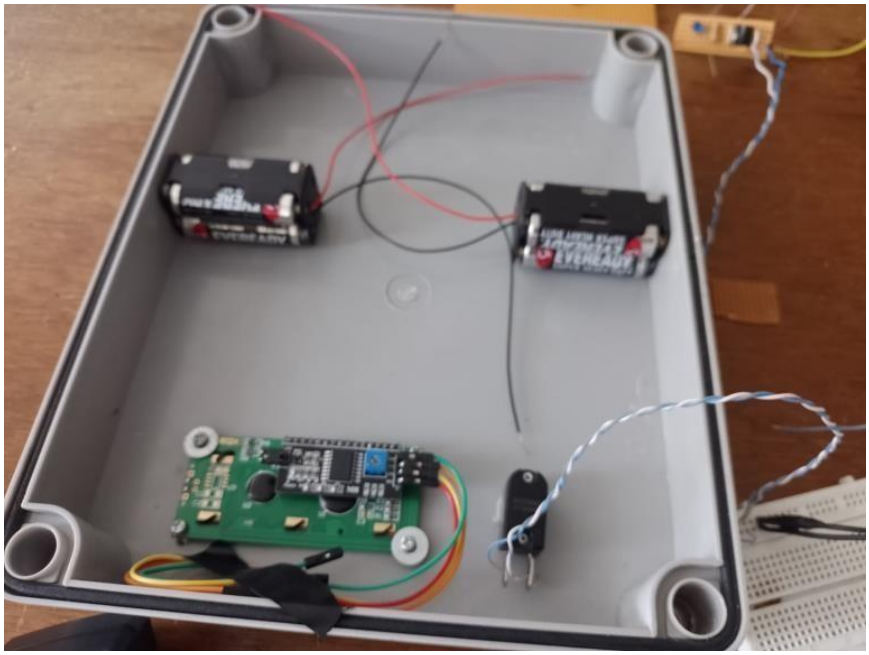


Figura 12. Cara interna del prototipo hardware2.

Por dentro del prototipo se encuentran las partes de alimentación, las partes de comunicación y las partes del procesamiento lógico, Entradas de sensores, USB y cuenta con un LED el cual indica cuando el sistema esté funcionando en conjunto.

Figura 13. Cara externa del prototipo hardware1.

El procesamiento se hace mediante la programación de la placa ESP32 junto al software de procesamiento de video utilizando las herramientas y tecnológicas de la visión artificial

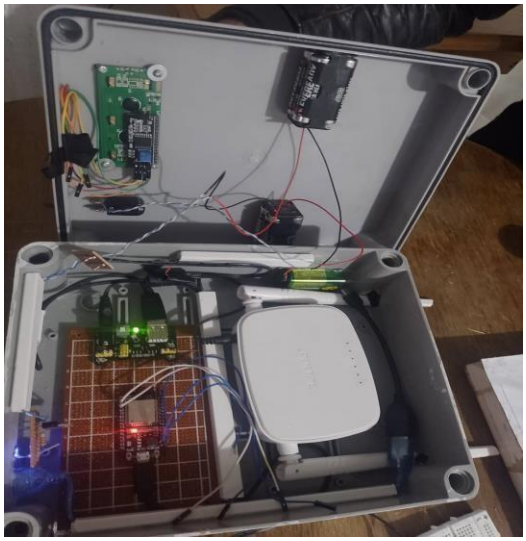


Figura 14. Cara interna del prototipo hardware3.

El prototipo es un boceto interactivo del producto final o de solo una parte, lo cual permitirá navegar por diversas secciones para poder experimentar con el contenido y las interfaces. El punto de enfoque puede ser «cajas grises» de baja fidelidad o de alta, la cual coincide a la última etapa de diseño.

Figura 15. Programa en LSD 1.

El sistema inicia con el nombre de la prueba a realizar, después de varios retardos y rutinas de procesamientos lógicos, saldrá la actualización del sistema conectándose a la red WIFI, la cual esta utiliza tecnología TCP para la placa de desarrollo ESP32 esto será única y exclusiva por el mismo prototipo, sin descartar la opción de poder conectar en otra red local, aparece IP de la red WIFI, por los protocolos de seguridad de Ethernet.

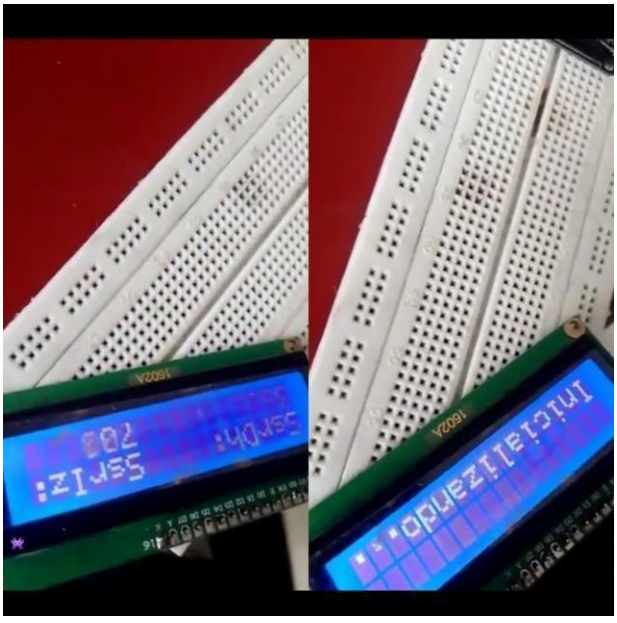


Figura 16. Programa en LSD 2.

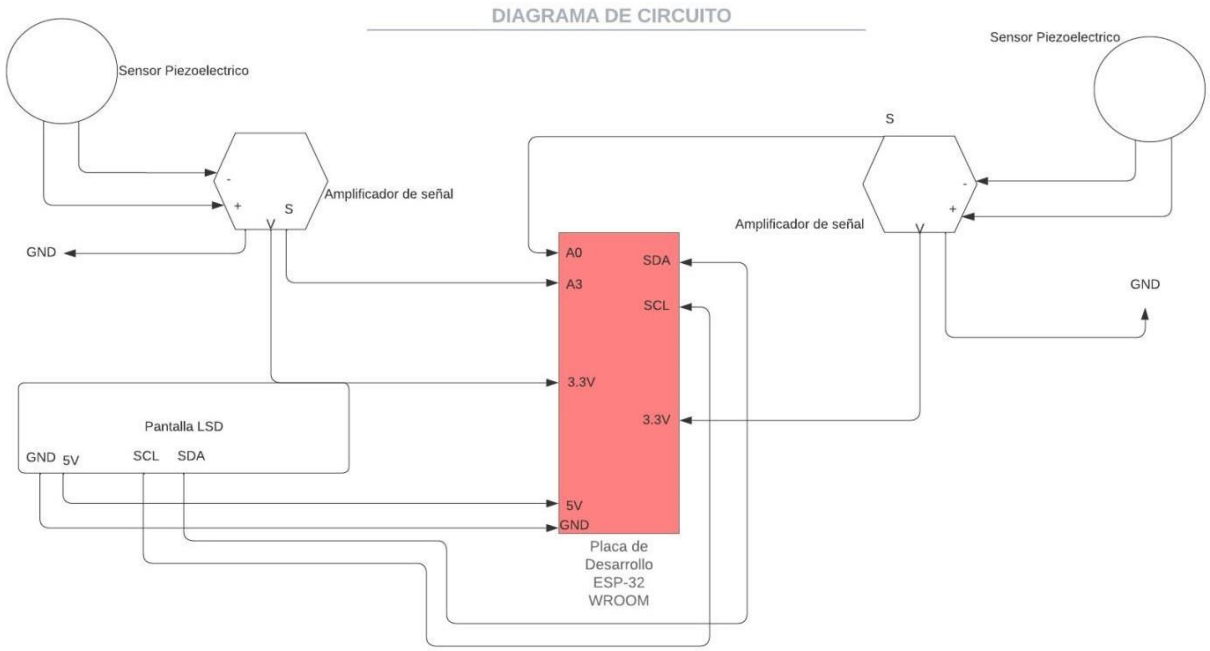


Figura 17. Etapa de procesamiento

Al término de unos segundos el sistema empezará a sincronizarse y empezará a mostrar los datos leídos previamente por los sensores.

- **Caracterización de sensores Piezoeléctricos.**

Para poder caracterizar los sensores se procede a escoger los pesos necesarios para poder realizar esta acción, con la ayuda de un asesor externo se llega a la decisión de proceder a sensar los valores de fuera con 4 objetos de peso y prueba; cada uno con diferente peso para así poder abarcar un rango un poco amplio con esto poder incluir los mayores valores posibles.

Para poder tener un poco más de fiabilidad con la exactitud del cuerpo a sensar se opta por cuerpos de porcelanicron.

Se lanzan desde una altura de 35cm y del sensor con un tiempo aproximado entre las 12 pruebas de cada peso de 17,0 segundos.

Figura 18. Pesos de caracterización de sensores piezoeléctricos.

Para poder apreciar la relación se expone la siguiente conversión

$$\text{Voltaje} = \frac{\text{Entrada Analógica ESP32} * 3.3\ \text{V}}{1024}$$

Se recogen los siguientes datos desde IDE:

En cada casilla se evidencia el resultado en milivoltios recogios por los sensores al percibir la presencia de las diferentes masas caracterizadas,

Tabla 1. Valores de caracterización pesos.

15 Gramos	**25 Gramos**	**40 Gramos**	**70 Gramos**
16.0mV	32.0 mV	50.0 mV	160.0 mV
26.0 mV	40.0 mV	80.0 mV	220.0 mV
18.0 mV	80.0 mV	250.0 mV	160.0 mV
13.0 mV	45.0 mV	80.0 mV	800.0 mV
15.0 mV	48.0 mV	160.0 mV	220.0 mV
15.0 mV	40.0 mV	80.0 mV	220.0 mV
30.0 mV	20.0 mV	160.0 mV	200.0 mV
10.0 mV	25.0 mV	80.0 mV	400.0 mV
24.0 mV	23.0 mV	160.0 mV	200.0 mV
11.0 mV	22.0 mV	160.0 mV	400.0 mV
24.0 mV	38.0 mV	80.0 mV	160.0 mV
15.0 mV	50.0 mV	95.0 mV	220.0 mV

Para esta sesión se llevaron a cabo 12 pruebas.

15 Gramos

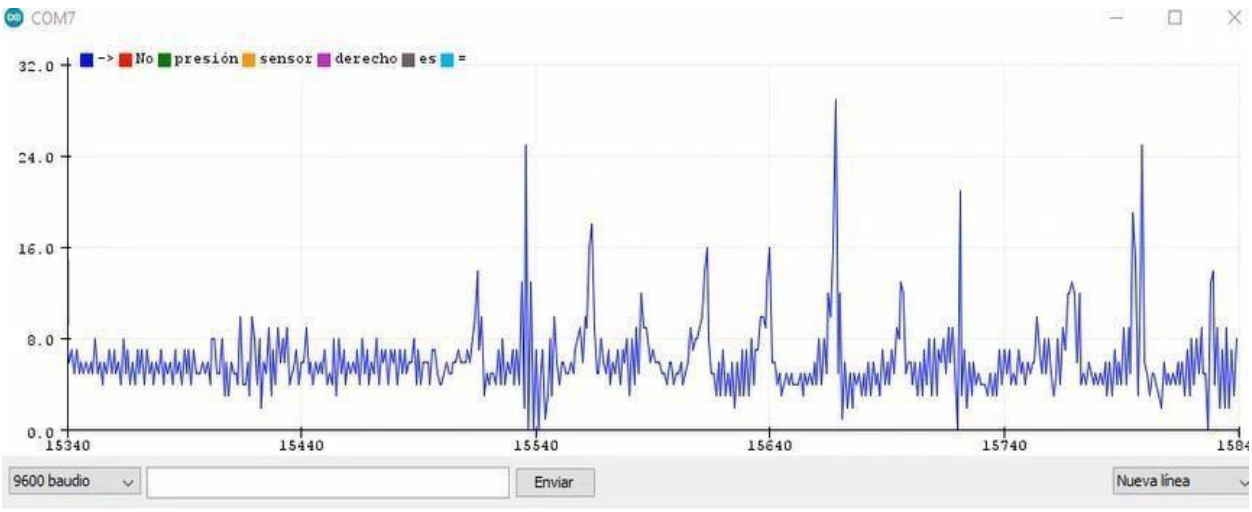


Figura 19. Plano de Caracterización para peso de 15 gramos.

En la figura 19 se puede llegar a observar las 12 veces en las que el sensor reacciona frene ala peso de 15 gramos.

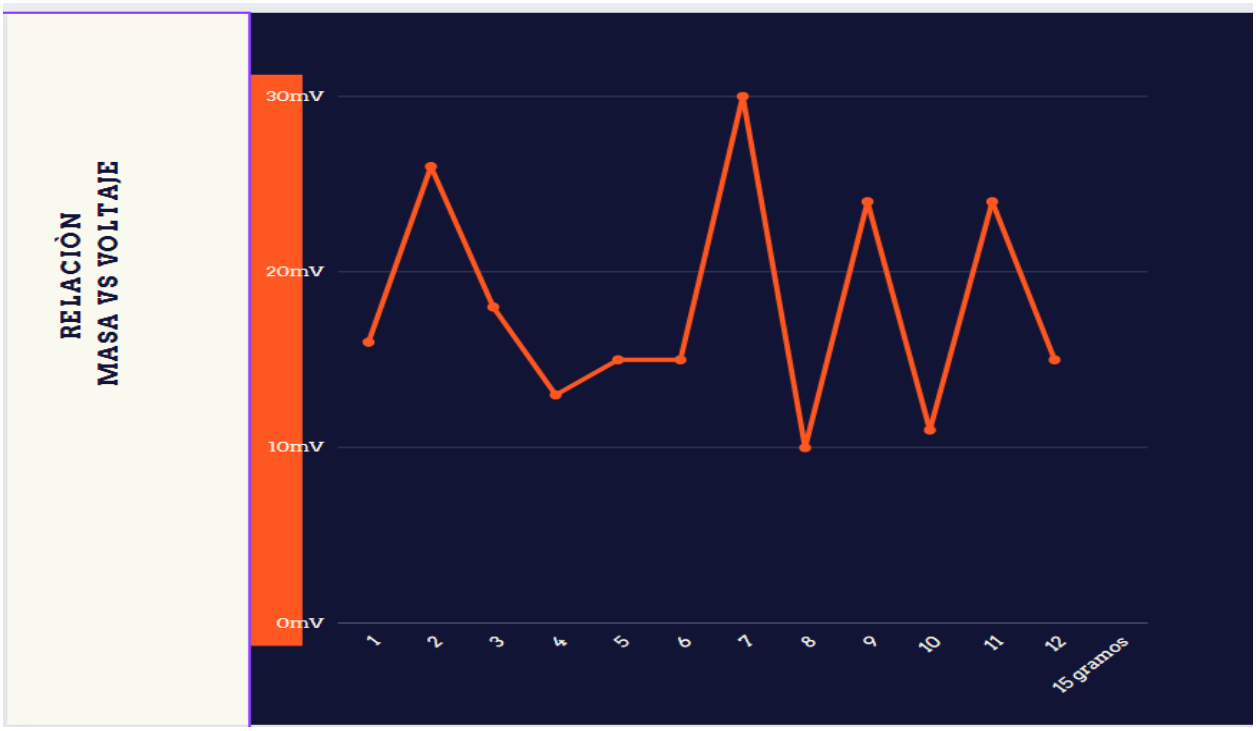


Figura 20. Comportamiento para masa de 15 gramos.

En la figura 20 se observa el comportamiento del sensor presentando los 12 golpes en los que presentan una correlacion lógico

En la figura 21 se observa como en la lectura del sensor se encuentra entre los 0mV y los 80mV.

25 Gramos

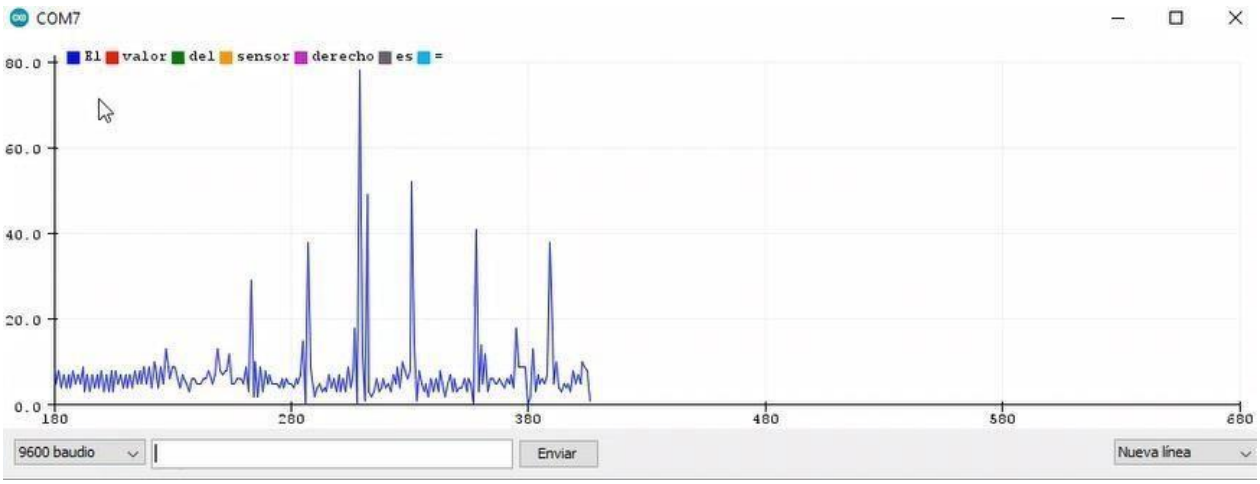


Figura 21. Plano de Caracterización para peso de 25 gramos.

En la figura 21 se puede observar que se tiene un pico que oscila entre los 0mV y los 80mV.

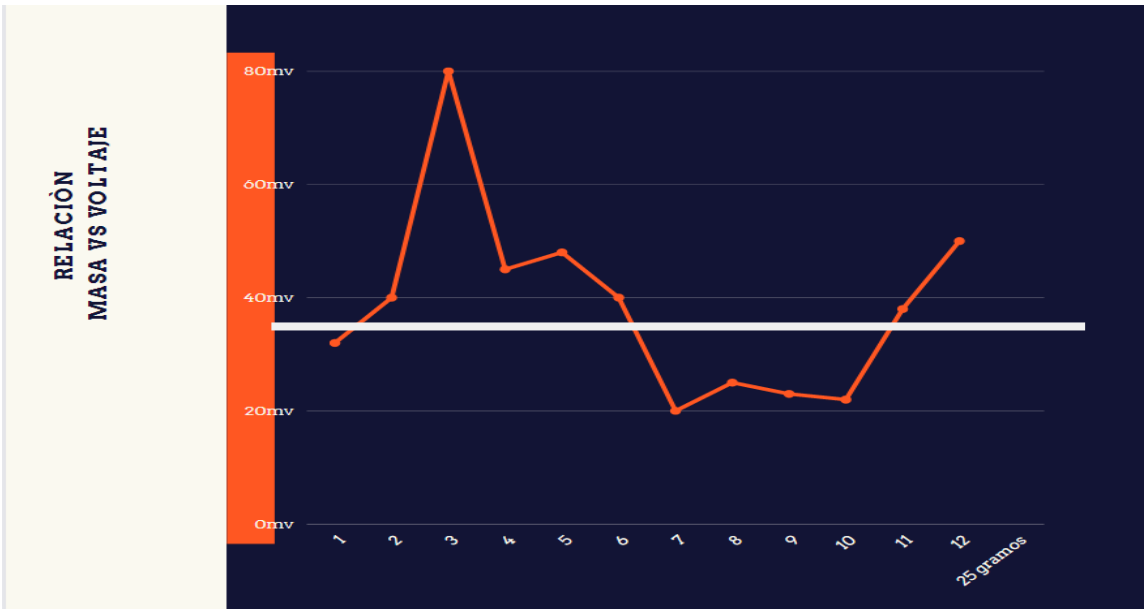


Figura 22. Comportamiento para masa de 25 gramos

En la figura 22 se puede evidenciar como el dato promedio ocsila entre los 20mV y los 50mV

40 Gramos

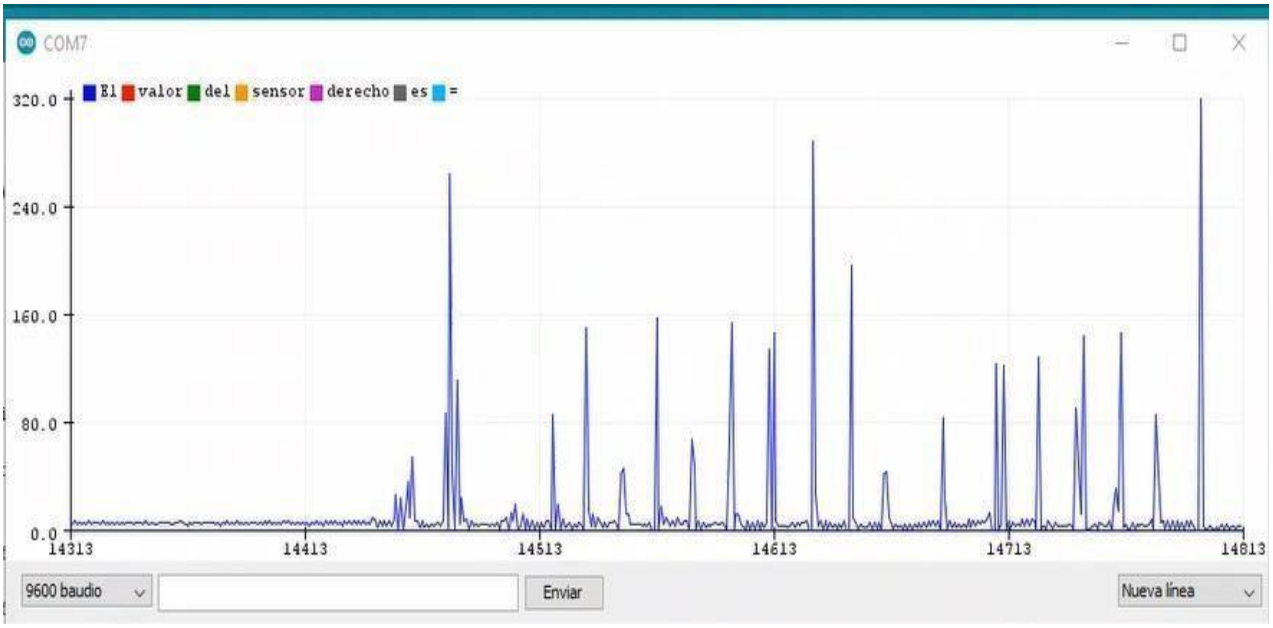


Figura 23. Plano de Caracterización para peso de 40 gramos.

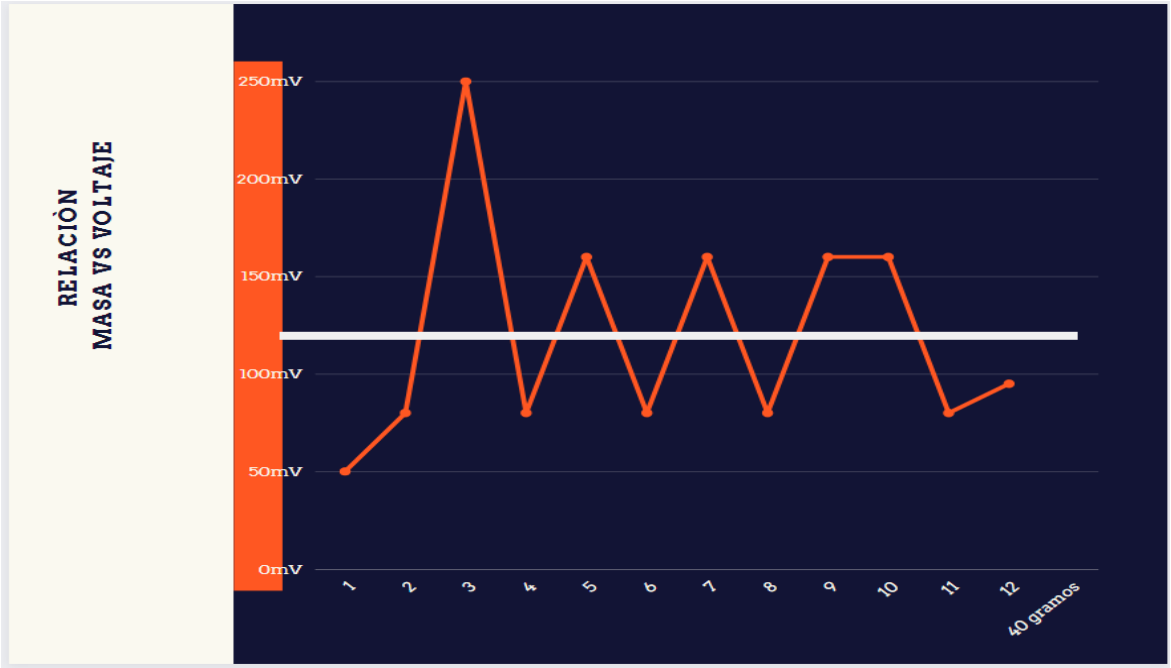


Figura 24. Comportamiento para masa de 40 gramos.

70 Gramos

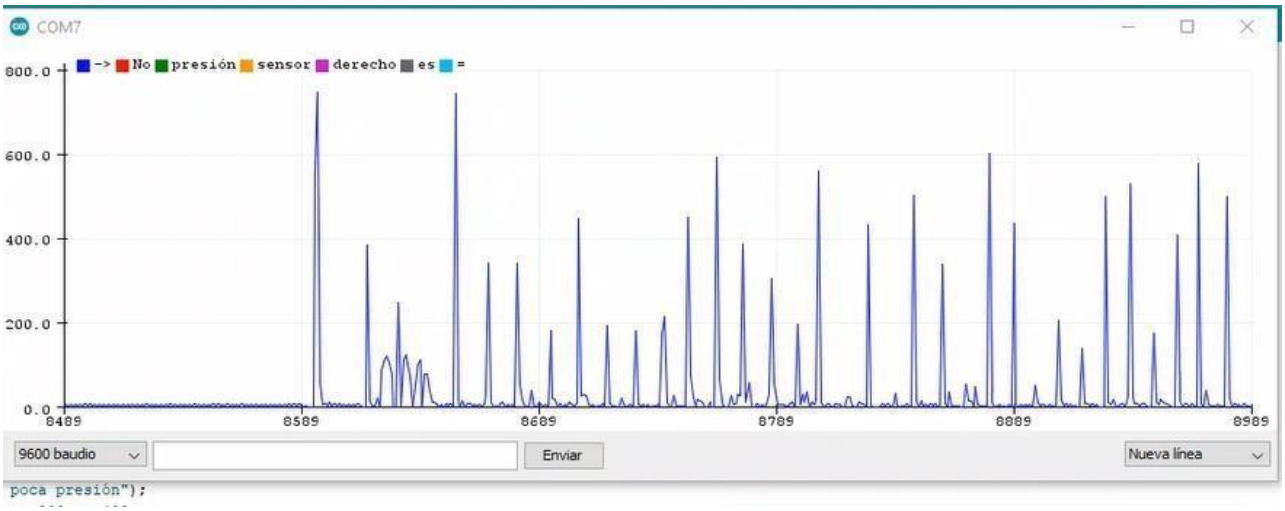


Figura 25. Plano de Caracterización para peso de 70 gramos.

En la figura 26 se puede observar que a mayor masa, menor es la influencia del ruido, gracias al acondicionamiento de la señal por medio del divisor de voltaje, si no la señal mostraría mucho ruido infiltrado y se perdería información valiosa.

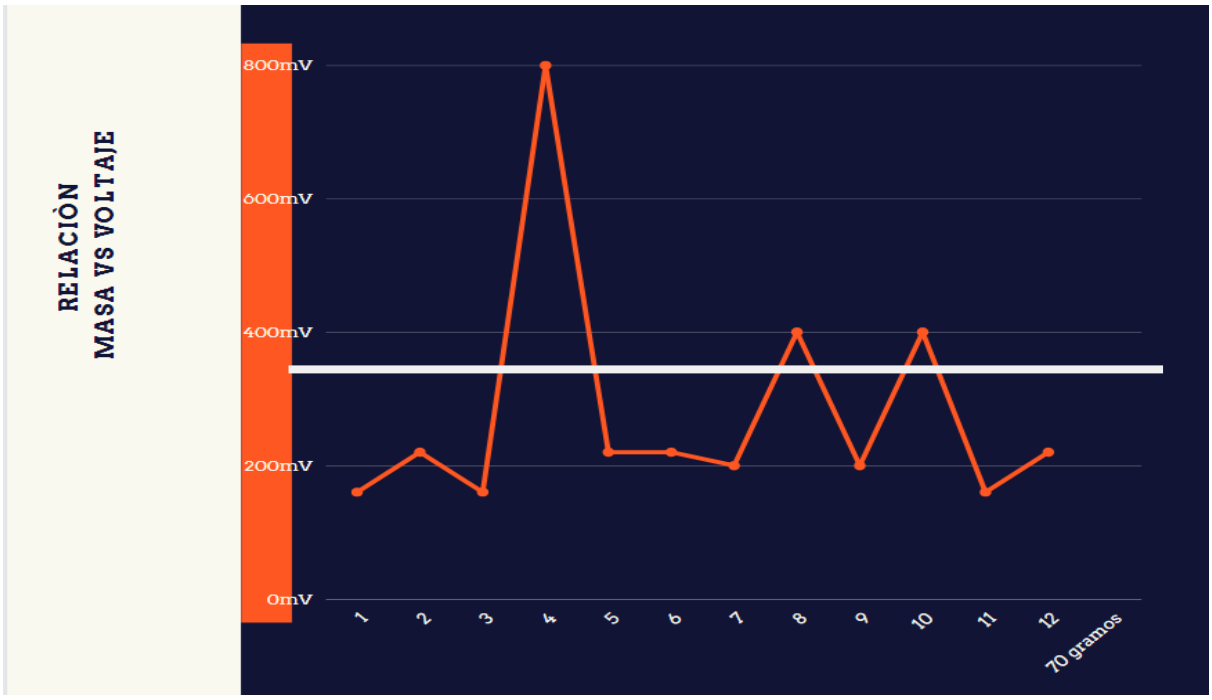


Figura 26. Comportamiento para masa de 70 gramos.

Se llega a lograr una caracterización de los sensores piezoeléctricos de acuerdo a unos pesos ya pre-establecidos, los resultados son los siguientes:

Tabla 2. Caracterización sensores.

15 gramos	18,08.0mV
25 gramos	38,583.0mV
40 gramos	119,58.0mV
70 gramos	280.0mV

De acuerdo a cada masa se realiza un promedio de entre todas las sesiones realizadas y los resultados son los que se mostraron previamente,

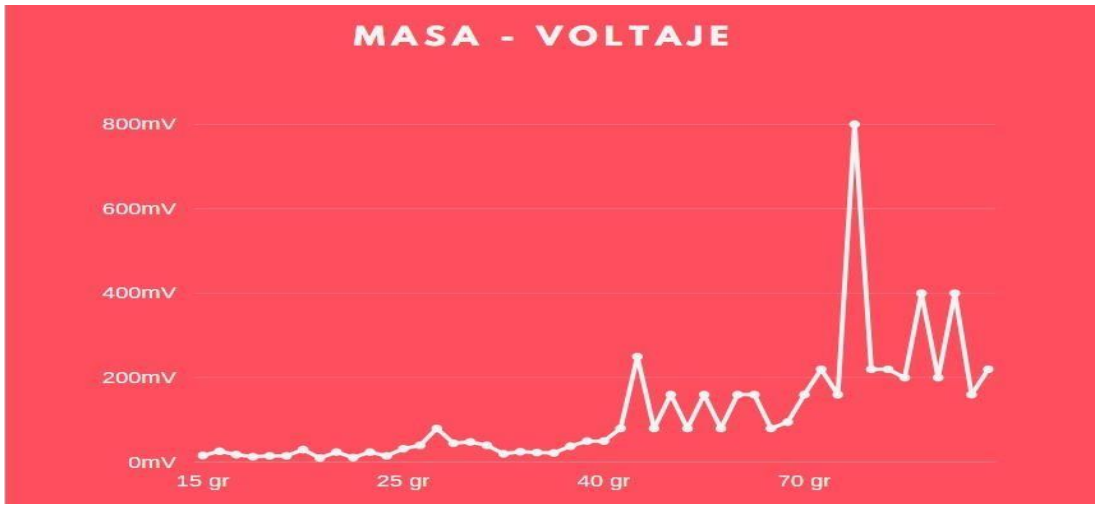


Figura 27. Comportamiento del sensor a través de las masas.

En la figura 27 se puede evidenciar que el comportamiento del sensor es un comportamiento expone

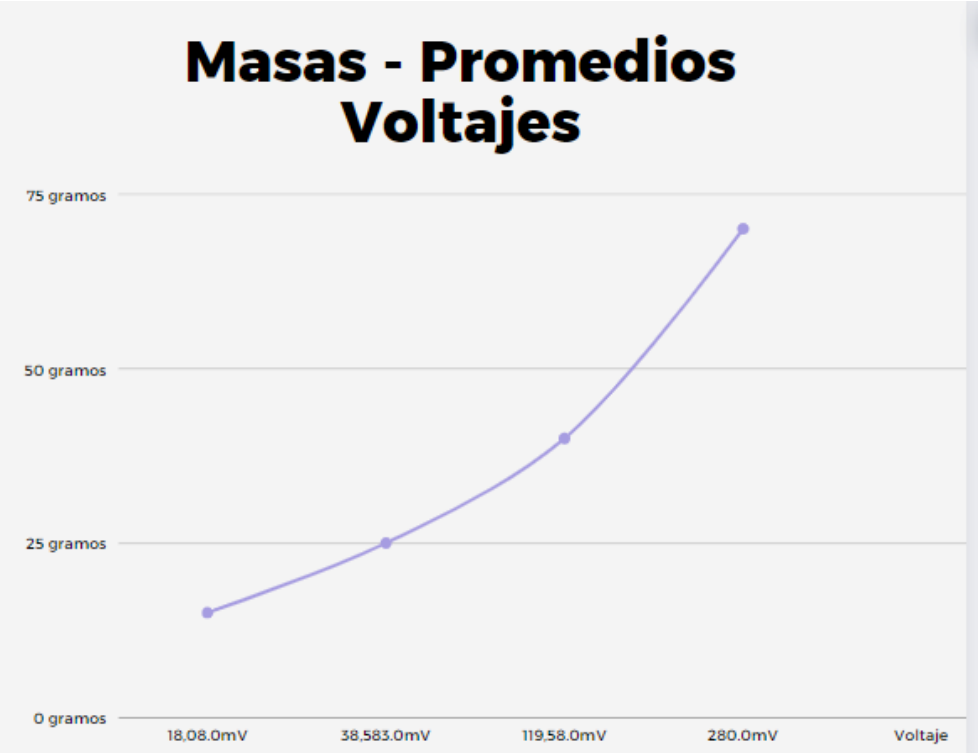


Figura 28. Comportamiento del sensor a través de las masas1.

Teniendo encuenta las pruebas se plantea un tiempo de prueba entre las diferentes pruebas, de acuerdo a esto se puede llegar a calcular una velocidad por toque, la cual se puede llegar a derivar, obeniendo así una acelecarion de cada prueba.

Tabla 3. Sistema 1.

Velocidad Aproximada	0,02m/s	**Aceleración Aproximada**	0,00117647m/s2
a-Masa 1	15gr	**a-Fuerza 1**	0,07271999999999999 N
b-Masa 2	25gr	**b-Fuerza 2**	0,07500000000000001 N
c-Masa 3	40gr	**c-Fuerza 3**	0,12 N
d-Masa 4	70gr	**d-Fuerza 4**	0,21000000000000002 N

Se realiza la estimación aproximada de la velocidad de los cuerpos teniendo en cuenta la distancia y el tiempo para cada caracterización de masas.

Se realiza para cálculo de la aceleración aproximada la respectiva derivada de la velocidad aproximada, se realizó un solo dato de velocidad y de aceleración para las 4 masas ya que los cambios en sus valores son insignificantes en los resultados finales.

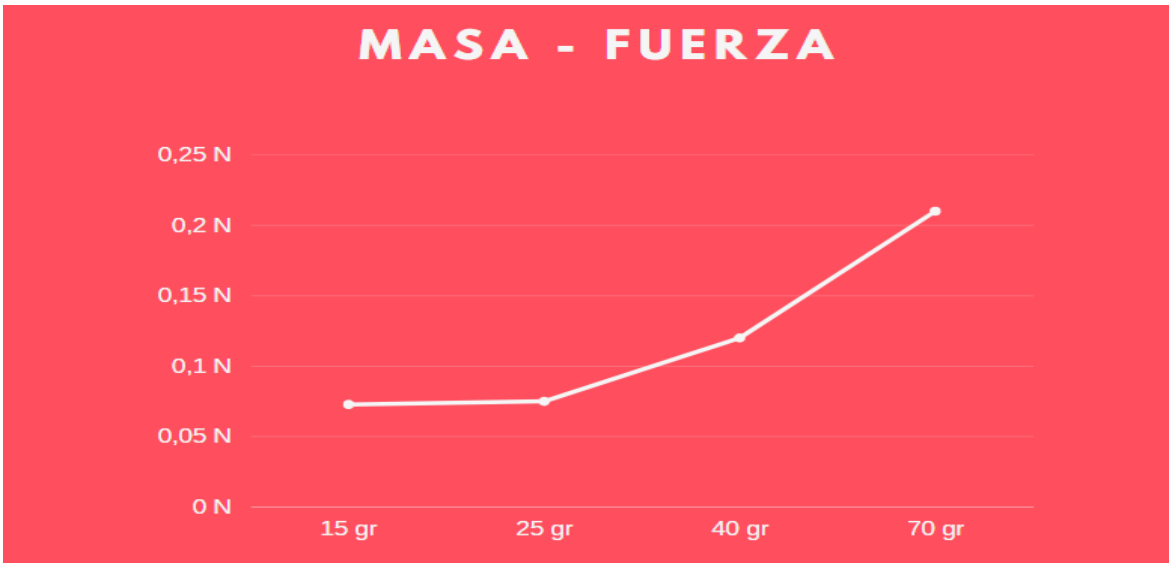


Figura 29. Relación Masa respecto a la fuerza.

En la figura 29 se puede evidenciar como la relación del sensor frente a las masas es de un aspecto exponencial ya que a mayor masa, mayor fuerza.

- **Modelo fisico de estimación de la fuerza a partir de datos de movimiento y características físicas del ejecutor.**

Para poder calcular y/o estimar la fuerza en las pruebas tapping test, se deberá construir un entorno el cual contenga las especificaciones. La actividad física constará de dos discos frente al ejecutor. Entendiendo el contexto anterior descrito, se procede a construir el ámbito estructural.

Figura 30. Tablero Tapping test 1.

Debajo de cada círculo se dispone de espuma la cual amortiguará cada golpe, sobre estas se instalan sensores conectados al sistema de cableado interno el cual está organizado por cada canaleta...

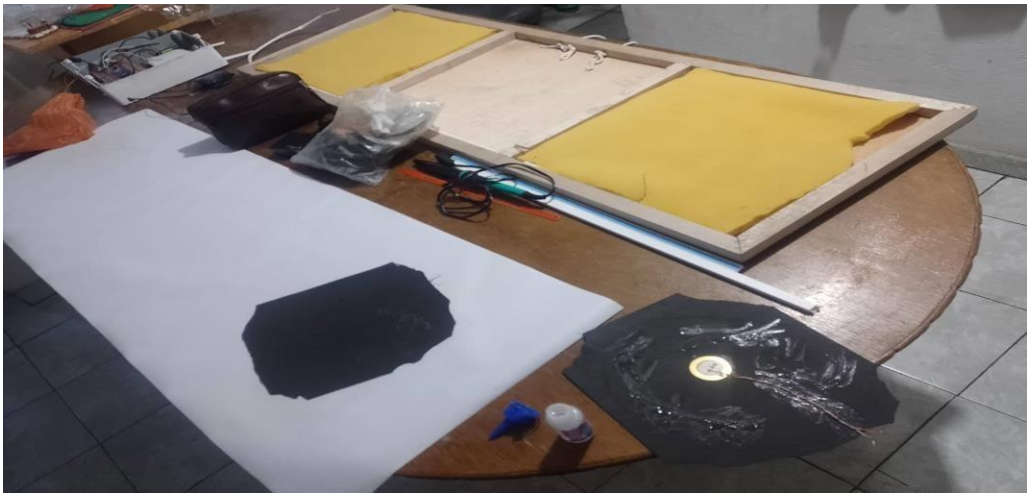


Figura 31. Tablero Tapping test 2

Al final se conecta el tablero con el dispositivo de recolección de información los cuales estarán conectadas por cableado interno previamente sellado en plástico aislante y protección...

Figura 32. Tablero Tappig test.

4.2.3 Diseño y construcción de Software

- **Análisis mediante visión artificial.**

En muchas aplicaciones basadas en la visión artificial, se utiliza la detección de movimiento. Por ejemplo, cuando se quiere contar las personas que pasan por un determinado

Lugar o cuántos coches han pasado por un peaje. En todos los casos, lo primero que setendría quehacer es extraer las personas o vehículos que hay en la escena.

Existen diferentes técnicas, métodos o algoritmos que posibilitan la detección de movimiento. Al igual que en otras materias, en la visión artificial no hay casos genéricos.Dependerá de cada situación usar uno u otro. Para hacer la detección de movimiento con OpenCV y Python.

La substracción de fondo consiste en tomar una imagen de la escena sin movimiento y restar los sucesivos fotogramas que se van obteniendo de un vídeo. A la imagen sin movimiento se le llama **fondo o segundo plano** (background en inglés). El fotograma que se va a analizar sería el **primer plano** (foreground en inglés). Por lo tanto, se tendrá un fondo al que se le va restando los diferentes fotogramas.

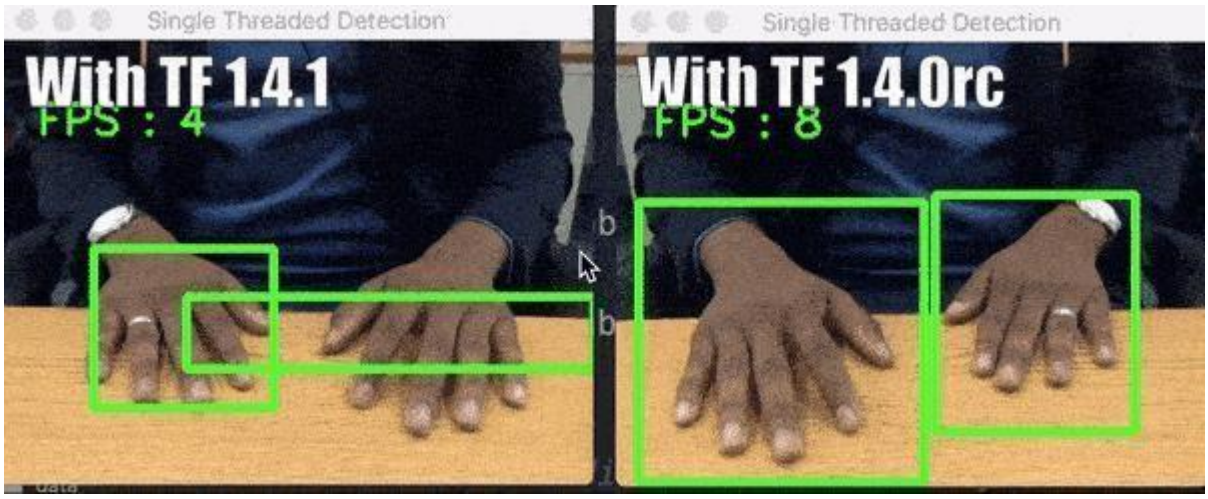


*Figura 33. Detección de manos, sustracción de objetos de un entorno1***.**

Es muy sensible a los cambios de luz. Imagínate que tomas la imagen de referencia en una habitación con luz natural. A las 10:00 de la mañana habrá unas condiciones de luz. Pero a las 18:00 de la tarde habrá otras. También es muy sensible a los movimientos de la cámara. Un movimiento muy pequeño puede hacer que se detecten falsos positivos en la escena. Por el contrario, este método funciona muy bien en entornos con iluminación controlada y detecta perfectamente la silueta de los objetos en movimiento.

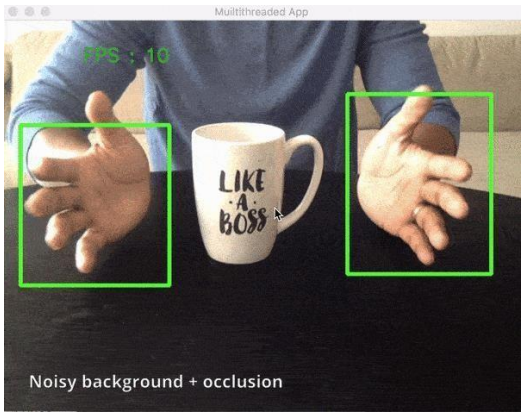


Figura 34. Detección de manos, sustracción de objetos de un entorno2

- **Faces de un proceso de detección de movimiento.**
-

Para crear un algoritmo que permita la detección de movimiento con OpenCV se tendrá que tener en cuenta unos factores, los cuales se mencionaran a continuación;

1. Conversión a escala de grises y eliminación de ruido.
2. Operación de substracción entre el segundo plano y el primer plano.
3. Aplicar un umbral a la imagen resultado de la resta.
4. Detección de contornos o blobs.

Algo muy común en las ciencias de la computación, en particular en la visión artificial, son los parámetros. Cada parámetro puede tener un rango de valores. El valor correcto dependerá de muchos factores. Está en nuestras manos adaptar cada valor a una situación concreta.

Figura 35. Detección de manos, sustracción de objetos de un entorno3.

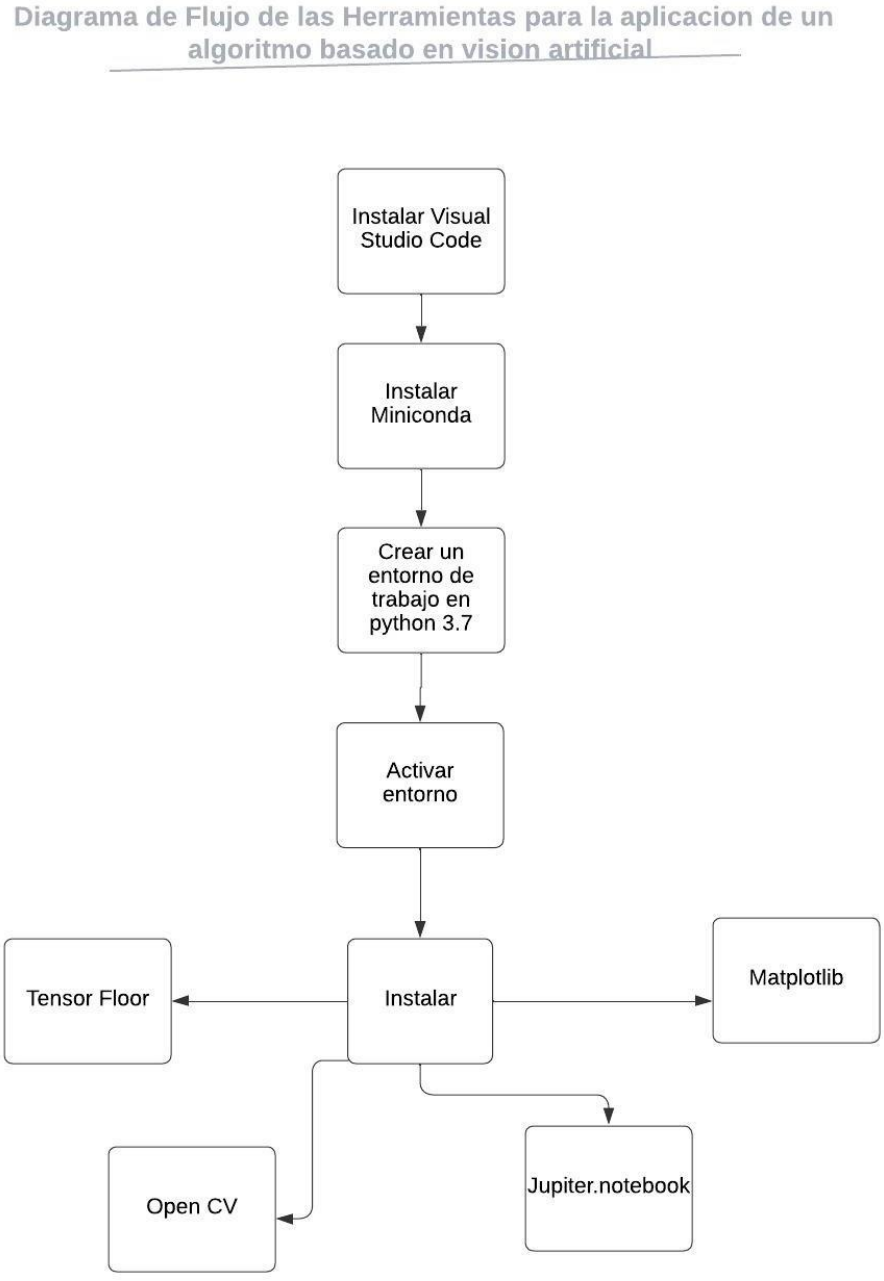


Figura 36. Diagrama de flujo de las herramientas parala aplicación de un algoritmo basadoen visión artificial.

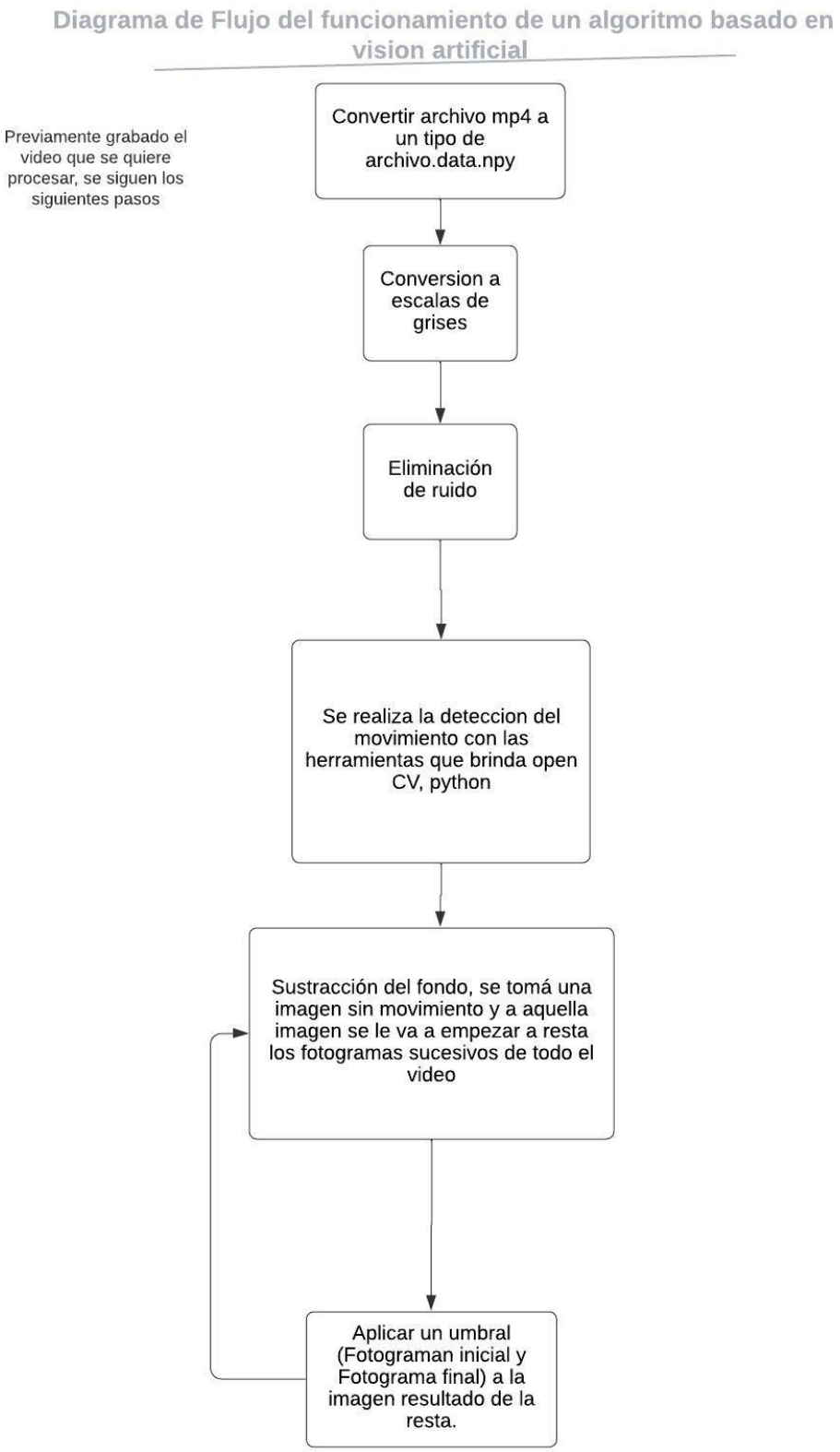


Figura 37. Diagrama del funcionamiento de un algoritmo basadoen visión artificial.

- **Requerimientos del sistema basado en visión artificial.**

Tabla 4. Características generales para la aplicación de un sistema

Tema	Descripción
Caracterización de los sensores	Se realiza la caracterización de fuerza y presión de acuerdo a los sensores, se tomaran la mayor cantidad de datos para poder así sensar con mucha fidelidad los valores
Caracterización de la cámara	Se seleccionaran los fps adecuados y la resolución optima para poder calcular los valores de cada video.
Reacción a condiciones especiales	Se requerirá de procedimientos para cualquier corrección Llegado el caso el entorno lo nocesatiase en donde se presentaran las pruebas
Características Generales	Se proveerá al ejecutor las características generales de Cada grupo a examinar.

- **Interfaz de usuario por medio de python y Matlab.**

Formado por imágenes y objetos gráficos, que representan la información y acciones que se encuentran en el sistema de prueba de tapping test. Su objetivo es el de crear un entorno visual fácil de usar para que fluya la comunicación con la persona usuario a cargo de las pruebas.

La comunicación de la interfaz hacia el usuario será de la manipulación directa de los sensores y de la fuerza ejercida sobre ellos, como también para simplificar la interacción y mejorarla se mostrara el comportamiento de la fuerza mostrado con la ayuda de un gráfico mostrado en la misma terminal.

Las GUI simplifican el funcionamiento de una computadora u otras máquinas permitiendo a los usuarios que no están familiarizados con la programación, ser capaces de utilizar la máquina en una variedad de maneras. Las GUI también están diseñadas para que sean intuitivas de usar, ya que permiten a los usuarios adquirir experiencia y conocimientos debido a que utilizan una interfaz gráfica de usuario.

De hecho la mayoría de GUI estan programadas para desplegar el resultado de una acción inmediata. Cuando un usuario hace clic en un icono de un programa ésta se abre. Cuando un usuario elimina un archivo desde una computadora de escritorio el icono del archivo desaparece. No obstante, detrás de cada interfaz gráfica de usuario, existe una interfaz de línea de comandos o CLICK que requiere comandos de texto introducidos mediante un teclado para funcionar así como el conocimiento de la orden de texto apropiado.

Se comeinza con la atmosfera de la introducción al software para esto se utilizara 2 ventanas las cuales darán acceso o lo negara y por ende habrá un reintento del mismo.

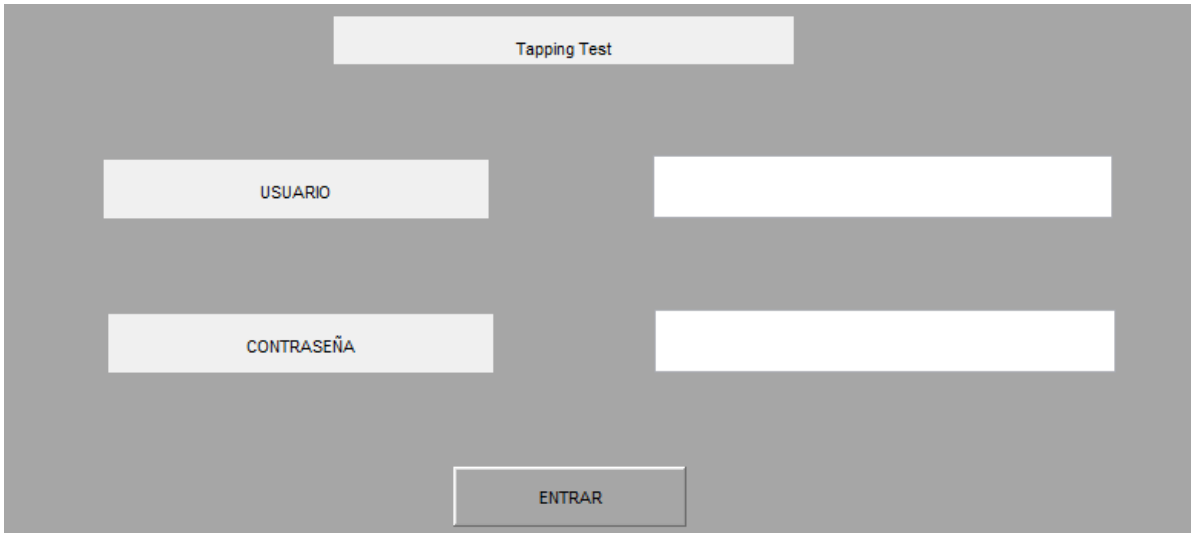


Figura 38. Ventana iniciar sesión.

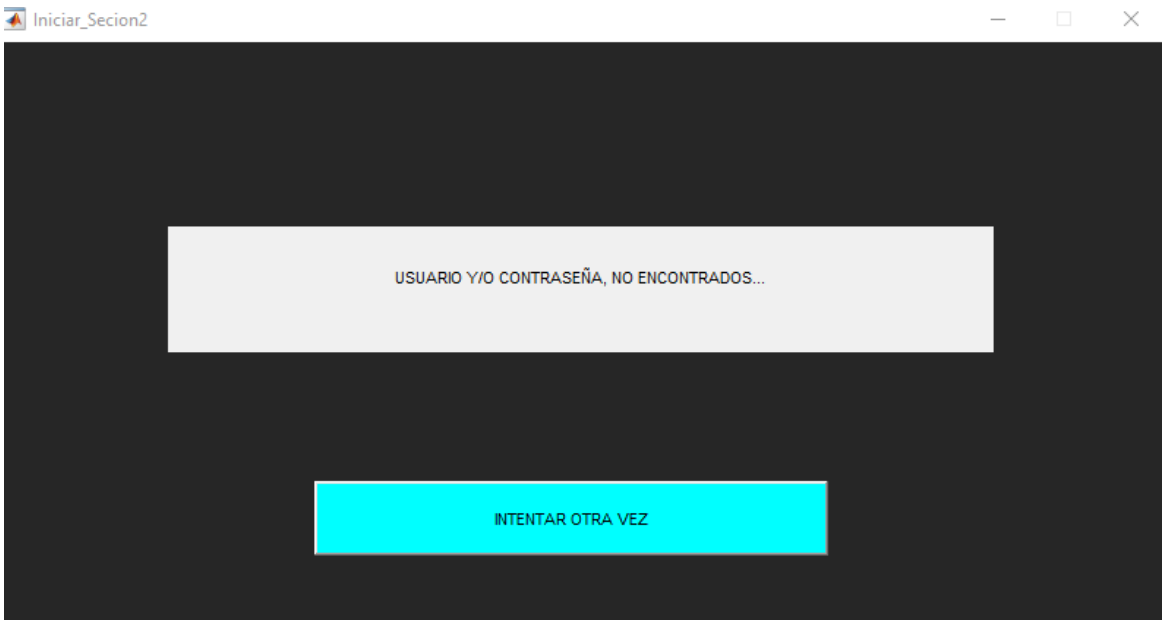


Figura 39. Ventana sesión denegada.

En la parte siguiente del programa se necesitará concretar los aspectos del funcionamiento del programa y se propuso 2 entornos los cuales estarán dirijidos a los datos de la prueba del tapping test y la otra dirigida al usuario y su interaccion con la misma.

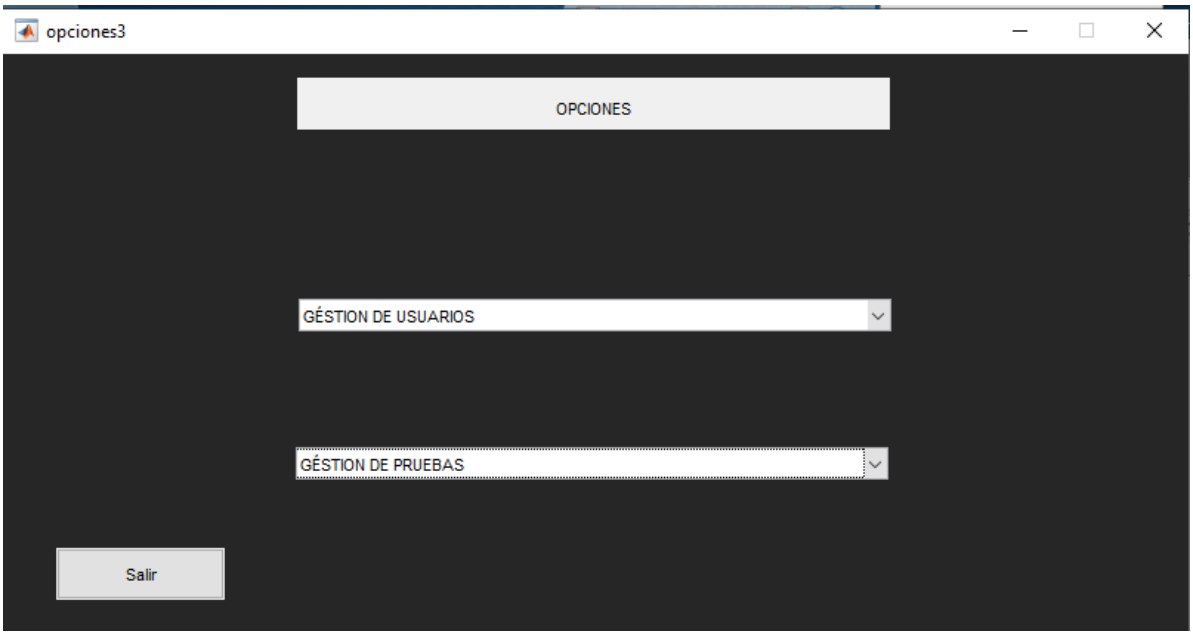


Figura 40. Opciones.

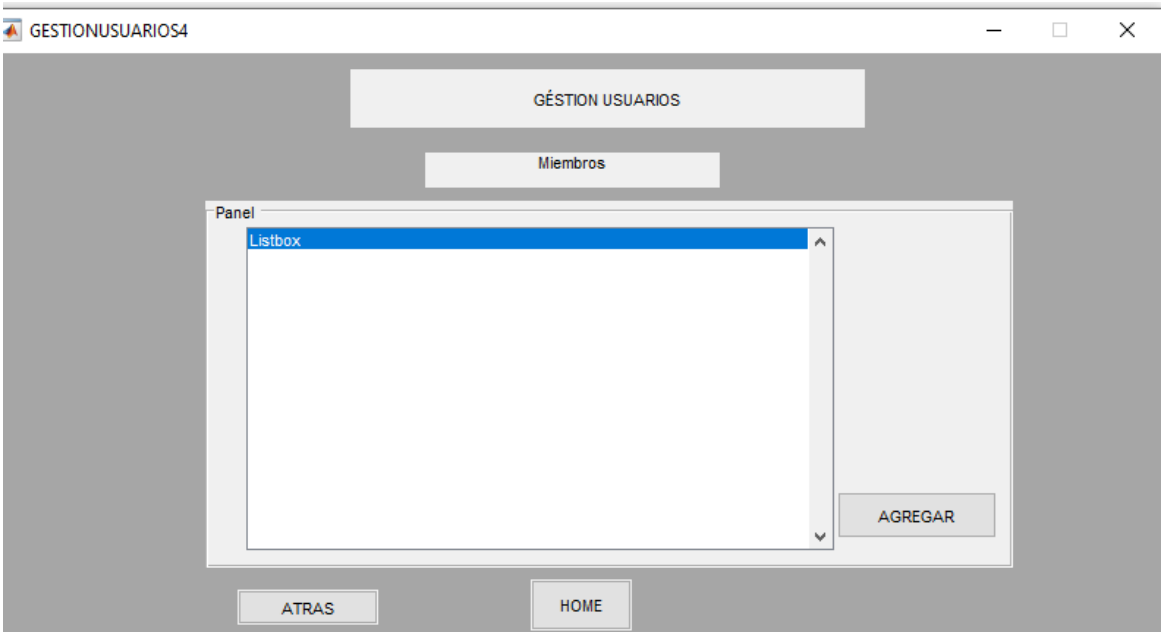


Figura 41. Ventana Gestion usuarios 1.

En esta atmosfera del software se quiere crear los grupos de prueba y asi mismo agregar a usuarios a los ya existentes grupos a su vez que se podrá crear y caracterizar una cuenta.

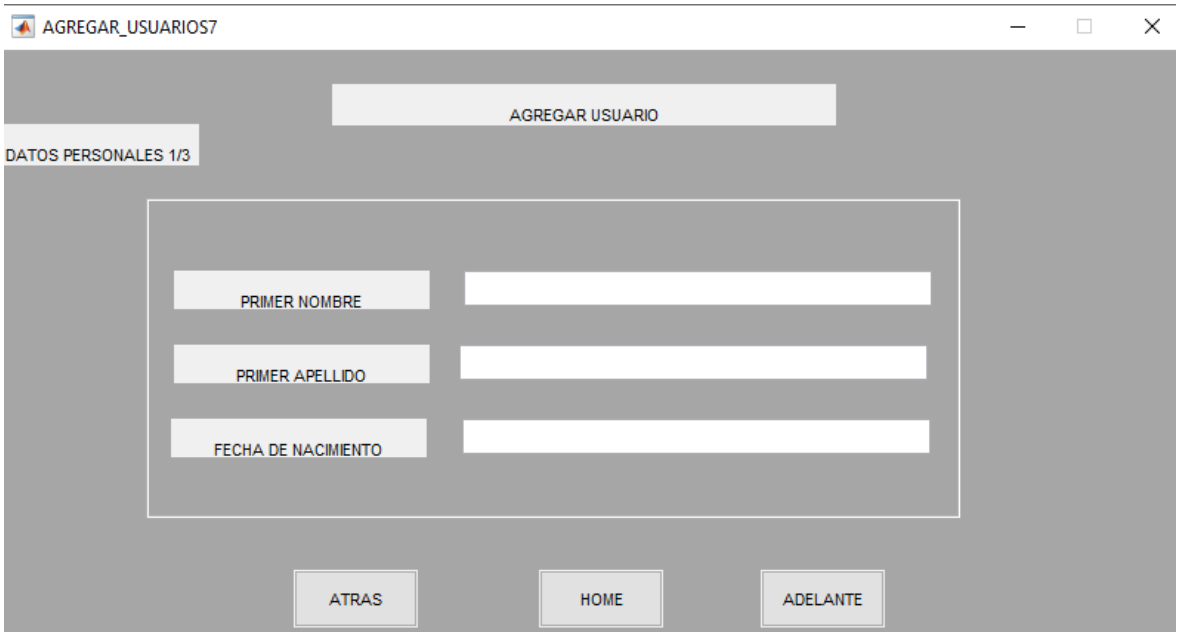


Figura 42. Ventana Gestión usuarios2

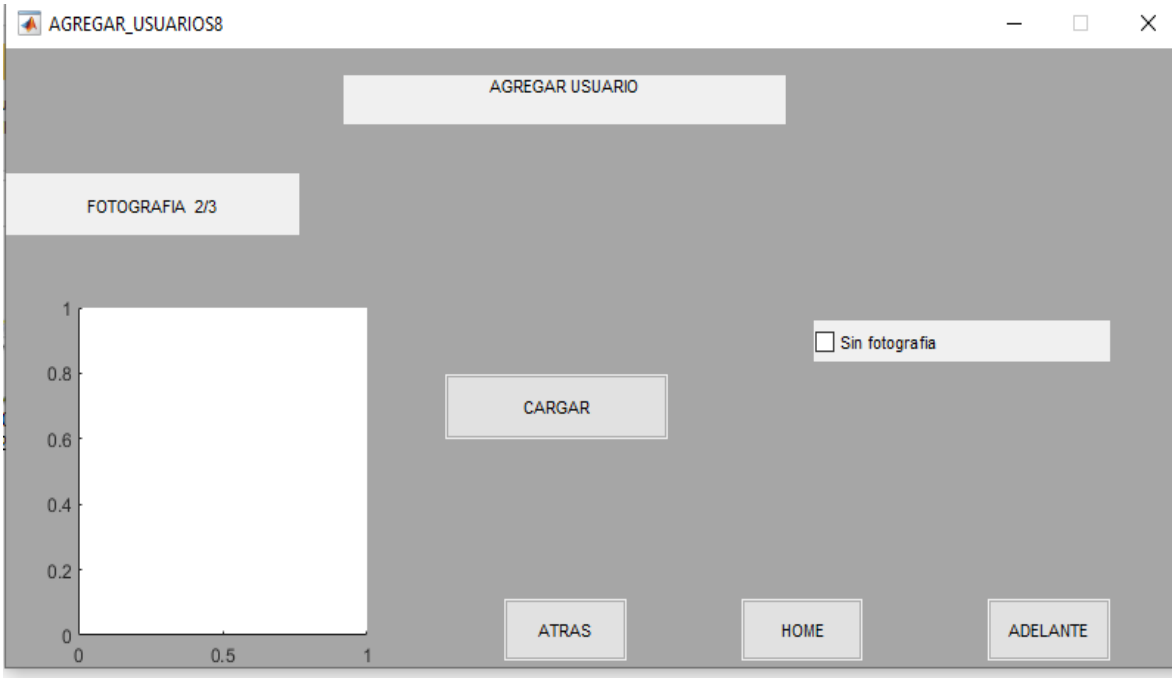


Figura 43. Ventana Gestión usuarios3.

En el siguiente aspecto se presenta la posibilidad de poder caracterizar al usuario con una fotografía.

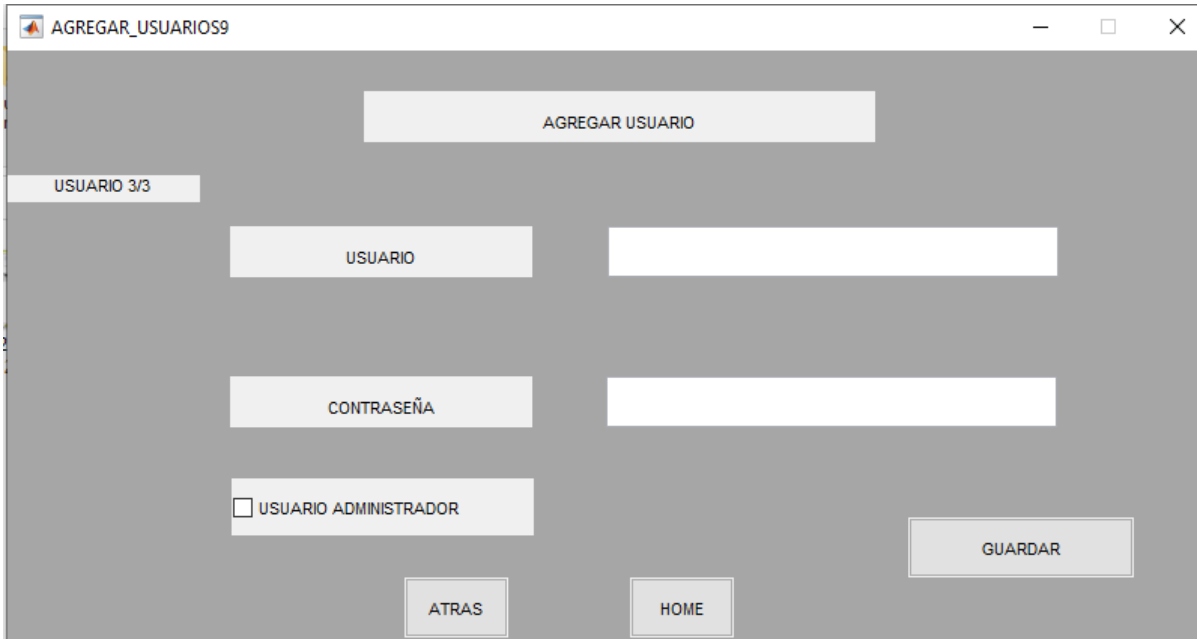


Figura 44. Ventana Gestión usuarios4.

En este aspecto se podrá crear un nombre de usuario y una contraseña, también se podrá seleccionar usuarios de administradores; los cuales tendrán accesos a la información de cada una de los test.

Figura 45. Ventana Gestión usuarios5.

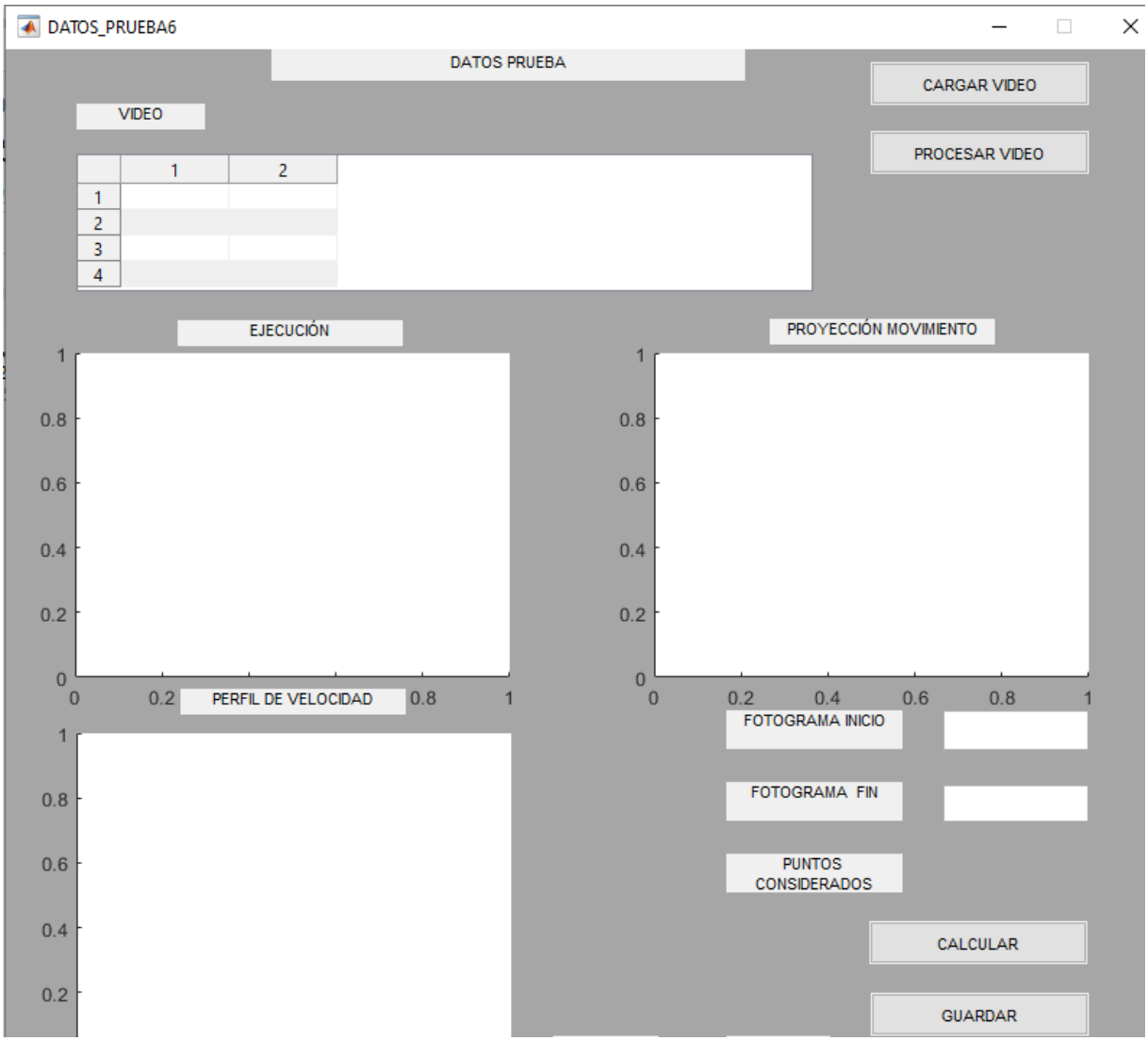


Figura 46. Ventana Gestión usuario6.

En la atmosfera de esta ventana se mostrara el perfil de velocidad en cada brazo del usuario, la proyección de su movimiento y la curva de fuerza en su ejecución, se mostrara el primer y el ultimo fotograma con cada de sus puntos seleccionados.

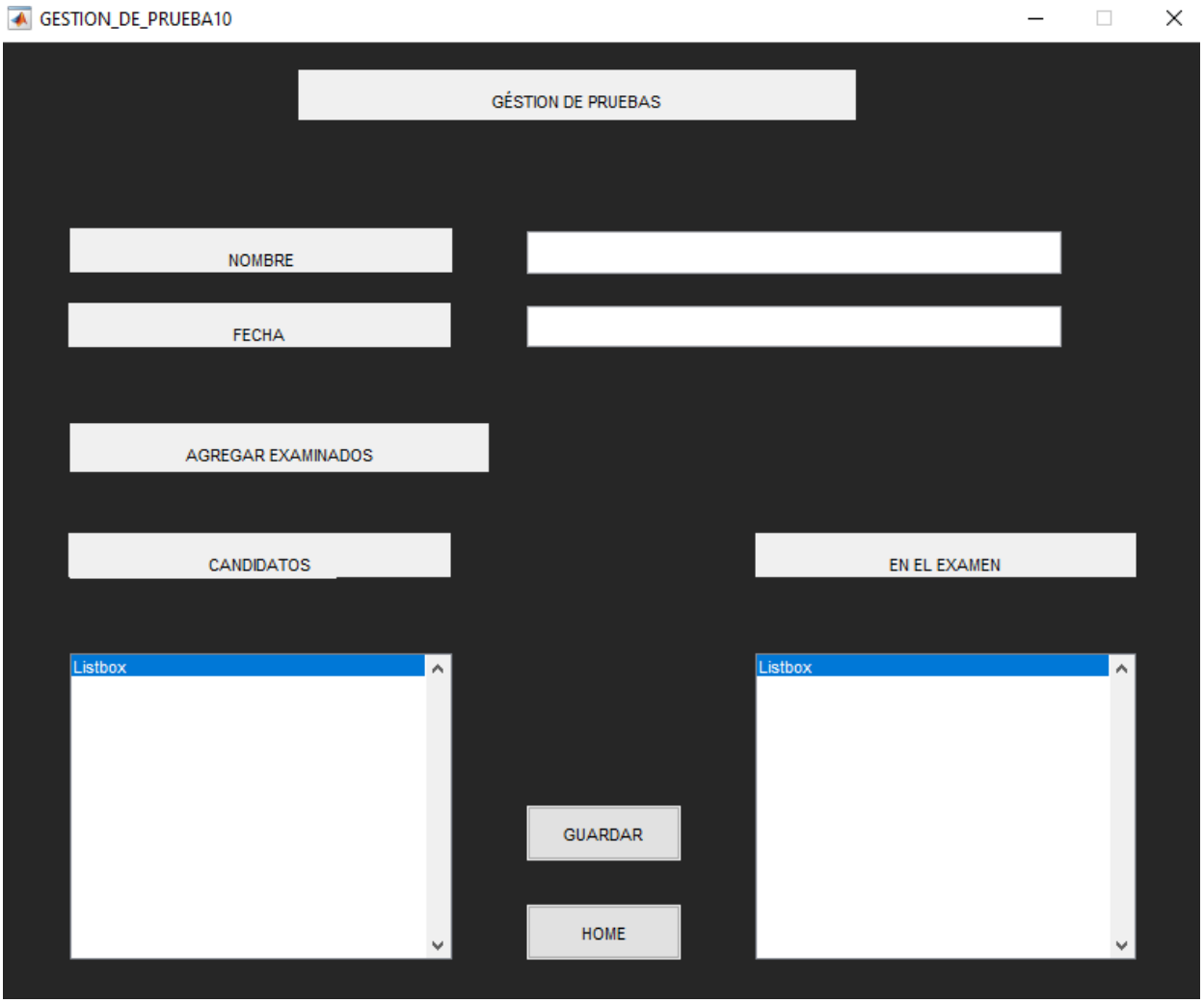


Figura 47. Ventana Gestión Prueba1.

La gestión del software se direcciona hacia los datos de la prueba, se podrá buscar seleccionar por nombre y por fecha, se mostrará también el listado de los candidatos y los ejecutores.

Figura 48. Ventana Gestión Prueba2.

En la siguiente ventana se podrá encontrar los grupos examinados. Se podrán eliminar grupos al igual que ver los datos de cada una de las pruebas seleccionados.

Para finalizar se mostrará los resultados de cada video de las pruebas seleccionados por usuario.

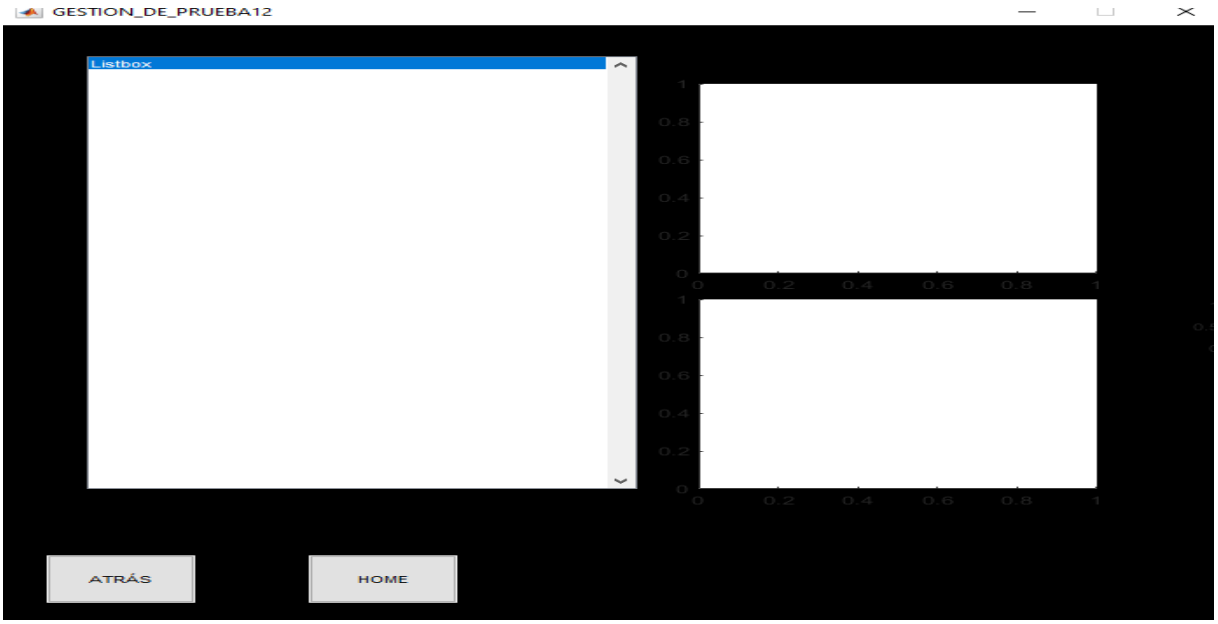


Figura 49. Ventana Gestión Prueba

4.3 Implementar un modelo de estimación de la fuerza a partir de datos de movimiento y características físicas del ejecutor.

Teniendo en cuenta los datos de movimiento de la caracaterizacion de los sensores, se plantea el siguiente modelo matemático exponencial con las herramientas de Matlab.

```
clc
clear
close

DM = [15 25 40 70];
V = [18.08 38.583 119.58 280];
plot(DM,V, 'ok-.')


n = length(DM);
y = log(V);
Sy = sum(y);
Sx = sum(DM);
Sx2 = sum(DM.^2);
Sxy = sum(DM.*y);
```

Figura 50. Modelo Matematico.

En la figura 50 Se introducen nuestros datos de masa y nuestros daos de voltaje. Con estos valores ya establecidos se empiezan a reemplazar en las formulas ya preestablecidas de un modelo exponencial.

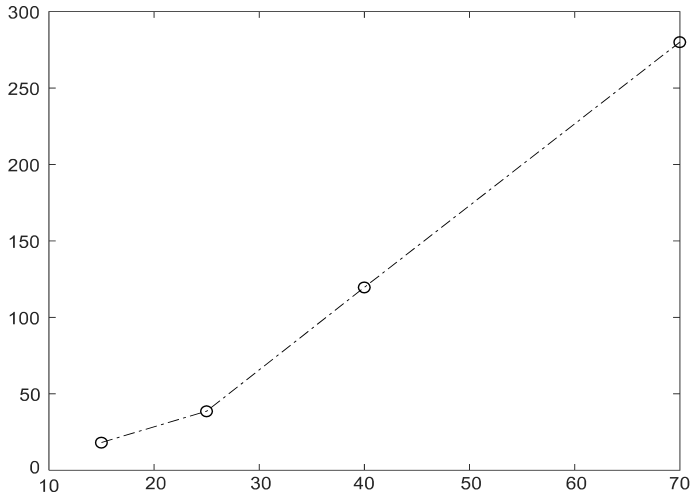


Figura 51. Datos para un Modelo Matematico.

En la figura 51 se puede evidenciar los datos del modelo, en este instante conociendo que son valores de masas y son valores de vontajes.

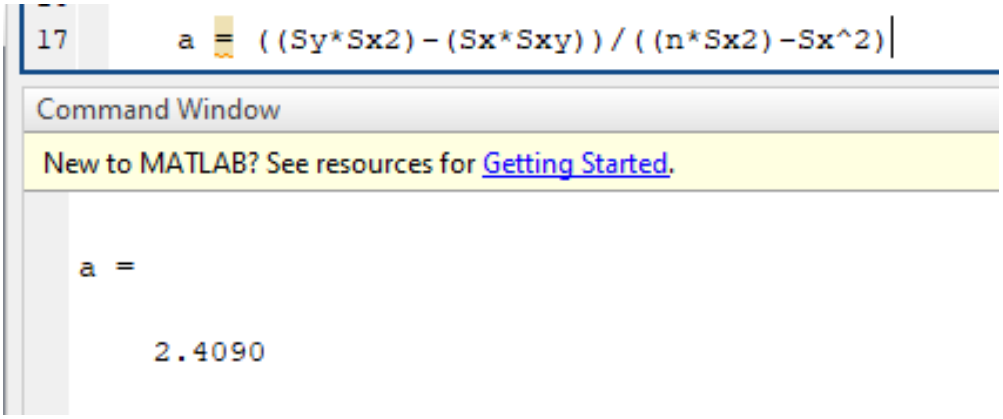


Figura 52. Valor de ‘’a’’ del Modelo Matematico.

En la figura 52 se puede evidenciar el valor de la variable ‘’a’’ para este modelo matemático, su valor es de 2.4090.

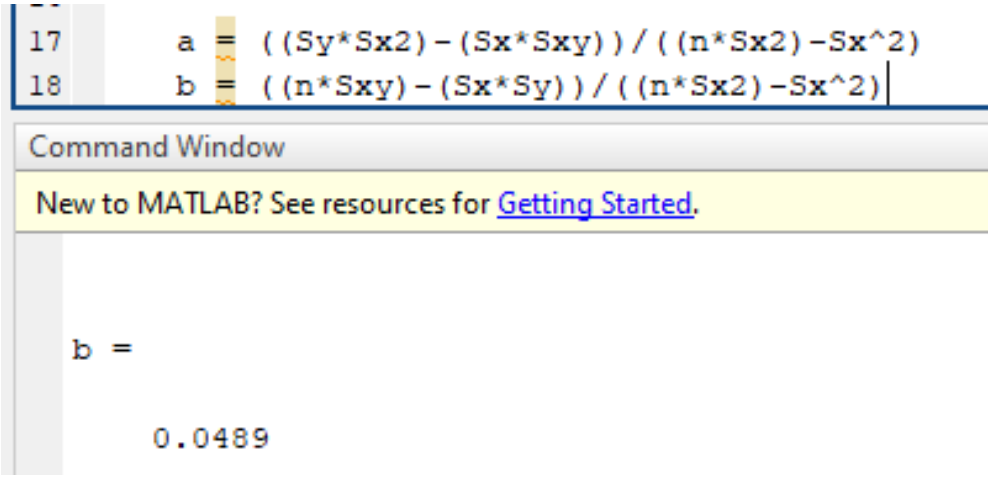


Figura 53.Valor de ‘’b’’ del Modelo Matematico.

En la figura 53 se puede evidenciar el valor de la variable ‘’b’’ para este modelo matemático, su valor es de 0.0489.

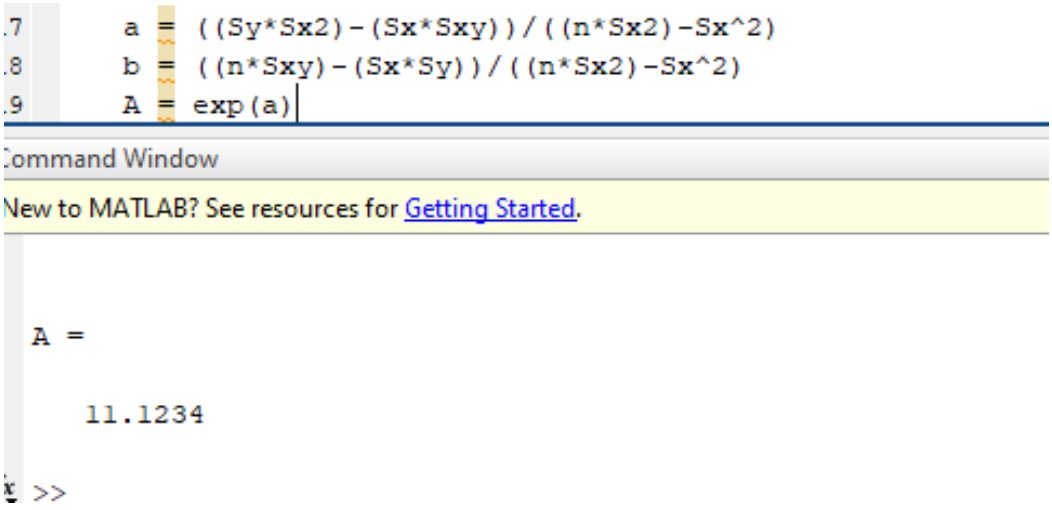


Figura 54. Valor de ‘’A’’ del Modelo Matematico.

En la figura 54 se puede evidenciar el valor de la variable ‘’A’’ para este modelo matemático, su valor es de 11.1234.

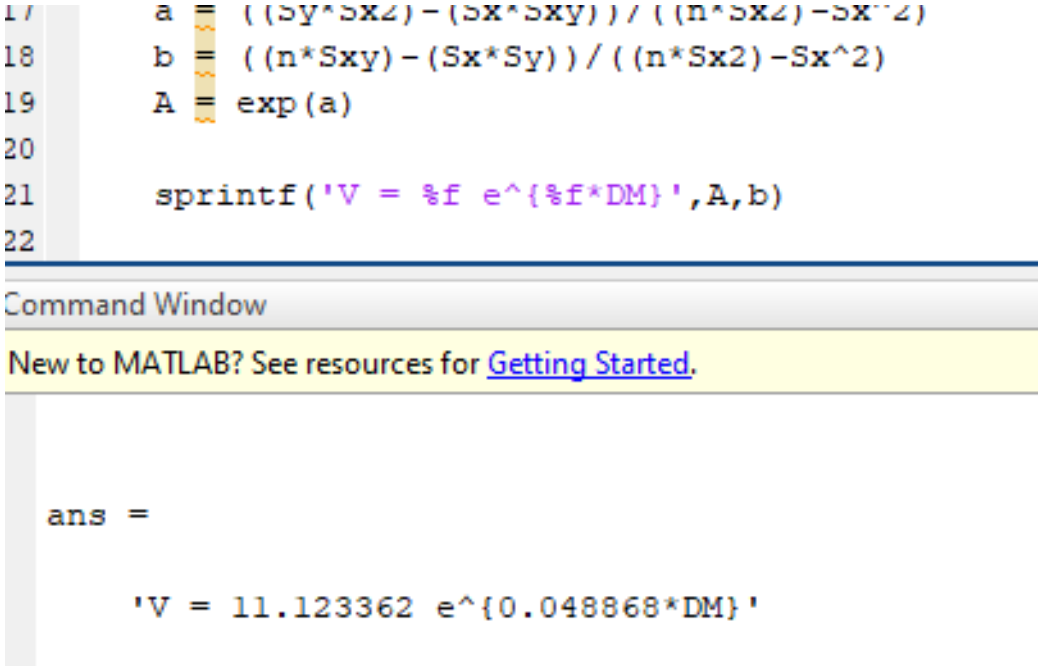


Figura 55.Modelo Matematico.

En la figura 55 se puede evidenciar la formula del modelo matemico exponencial donde DM es inerpretador por el valor de la masas.

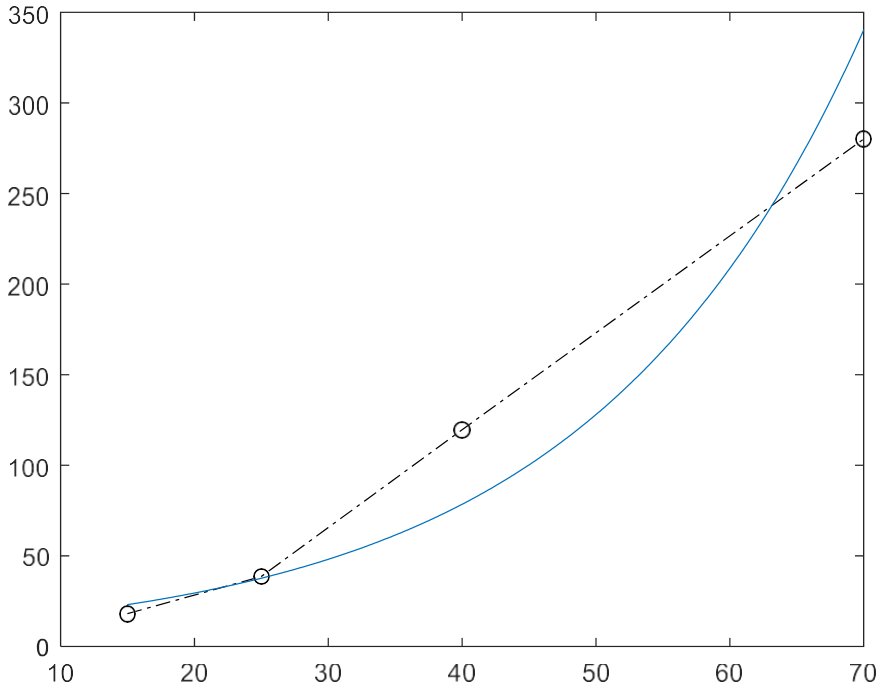


Figura 56. Simulacion de puntos en el Modelo Matematico.

En la figura 56 se puede evidenciar como se simulan valores de puntos de 0.1 en el modelo matemático ya establecido, se evidencia cierta aproximación en la información.

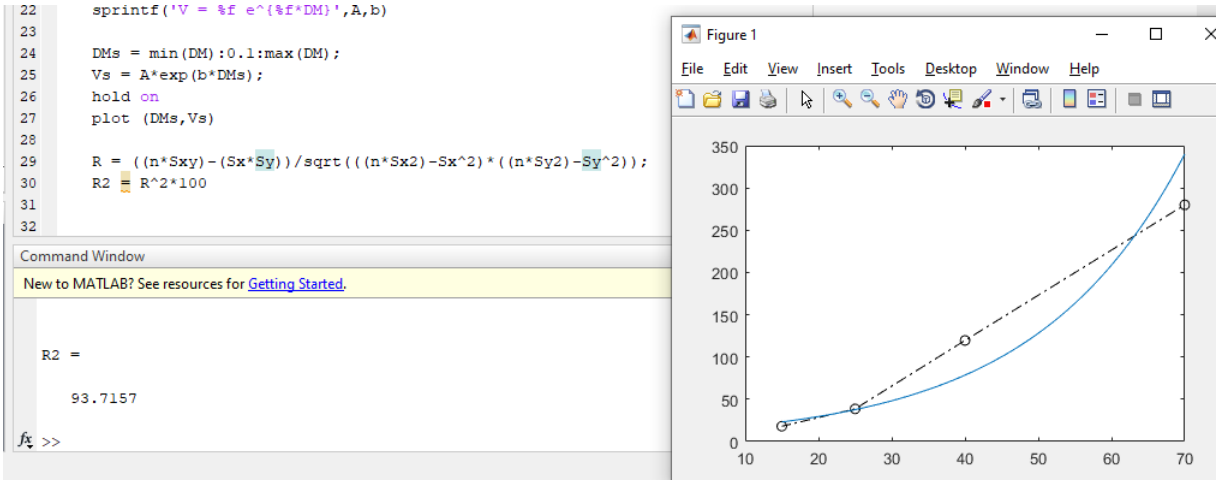


Figura 57.Simulacion de puntos en el Modelo Matematico.

En la figura 57 se puede evidenciar el valor de 93.7157 lo cual hace referencia que el modelo matemático funciona en la realidad bajo ese porcentaje.

4.4 Evaluar el desempeño del prototipo construido.

El día 11 de octubre se realizó las primeras pruebas con 8 estudiantes de Noveno semestre de ingeniería electrónica. Esto con el fin de poder evaluar el prototipo fisico

Los datos se tomaron en el laboratorio de electrónica Buenaventura de la facultad de ingeniería en el sexto piso del edificio Holanda.

Figura 58. Tappig test 11 de octubre 2022.

En la figura 58 se evidencia una voluntario realizando la prueba tapping test. Frente a ella se tienen las cámaras grabando el ejercicio y el prototipo que recolecta la información de los sensores ubicados en el tablero.

Figura 59. Tappig test2 11 de octubre 2022.

En la figura 59 se muestra al Aseor Mario Henao y una estudiante haciendo una demostración de la forma correcta de realizar el ejercicio.

De la primera sesión se pueden obtener los siguientes resultados, En la siguiente tabla se evidencia los resultados del tiempo organizados de acuerdo a los códigos de cada estudiante.

Tabla 5. Base de datos primera sesión.

Identificación	Código	Nombres y Apellidos	Tiempo en la Prueba
1006788810	0018218	Lucia Geraldina Gómez	17.09 Segundos
1193132462	0017218	Andrés Felipe Quiñonez Mesías	17.27 Segundos
1085337137	O005218	Esteban Alejandro Meneses Ortega	24.93 Segundos
1004673496	O021218	Euler Enrique López Almeida	14.40 Segundos
10853266088	O002218	Tatiana Paola Cabrera	16.80 Segundos
1086110205	O02521	Luis Castillo Mancajo	30.15 Segundos
1010018880	0010218	Juan Sebastián Rodríguez Mora	18.40 Segundos
	0015117	Kevin Steven Sánchez	16.46 Segundos

Se tapan las columnas de identificación u de nombres y apellidos ya que en el momento de realiar la prueba no se le pide permiso para otorgar esta información personal.

4.4.1 Base de datos de la sesión primera

Figura 60. Base de almacenamiento datos sensores (IDE).

En la figura 60 se muestra la carpeta de almacenamiento donde se guardan los resultados de cada prueba organizados de acuerdo al código de la prueba.

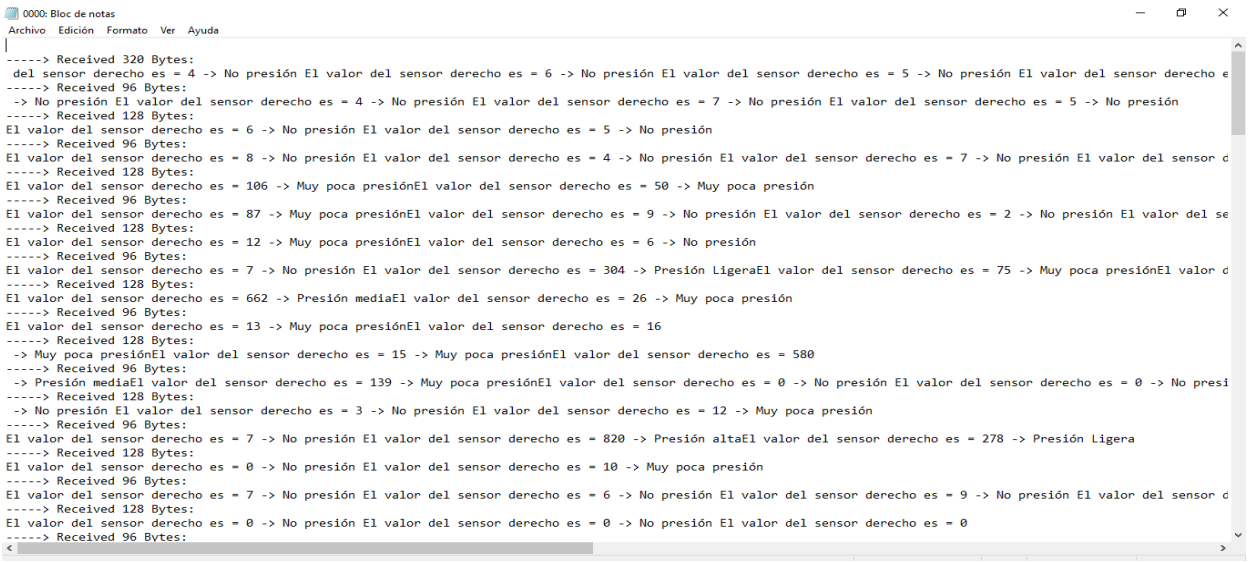


Figura 61. Archivo de base de datos de laprueba.

En la figura 61 se puede evidenciar la lectura de los sensores de acuerdo a las relaciones de golpe del ejecutor, esto es lo que contiinen los archivos de la figura xx.

En la siguiente tabla se evidencia los m¿nombres de la prueba asgnados a cada tiempo y a cada código del estudiante.

Tabla 6. Base de datos primera sesion1.

[redacted]	0018218	[redacted]	17.09 Segundos	Código Prueba 0000
	0017218		17.27 Segundos	Código Prueba 0001
	O005218		24.93 Segundos	Código Prueba 0002
	O021218		14.40 Segundos	Código Prueba 0003
	O002218		16.80 Segundos	Código Prueba 0004
	O02521		30.15 Segundos	Código Prueba 0005
	0010218		18.40 Segundos	Código Prueba 0006
	0015117		16.46 Segundos	Código Prueba 0007

Prueba 0000

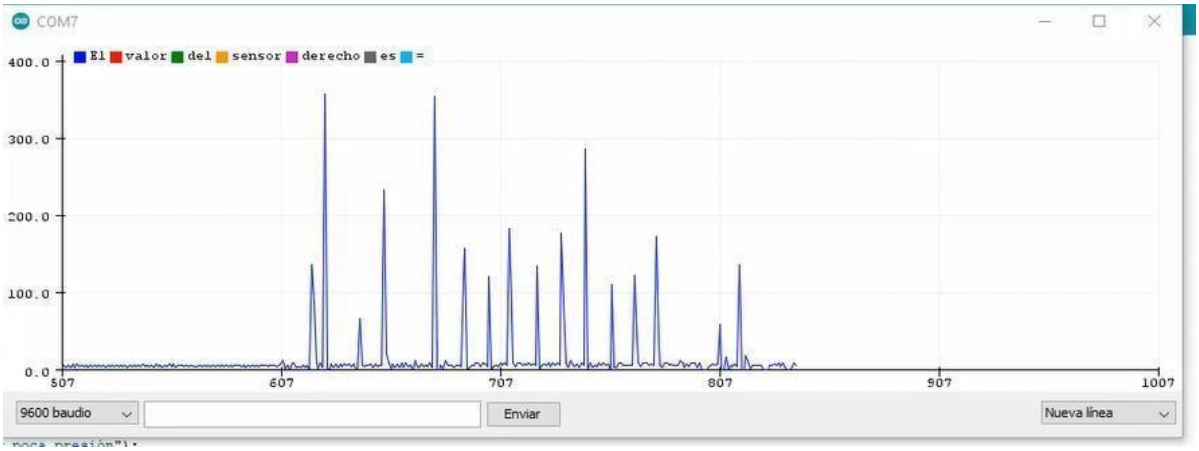


Figura 62. Diagrama de datos 0000.

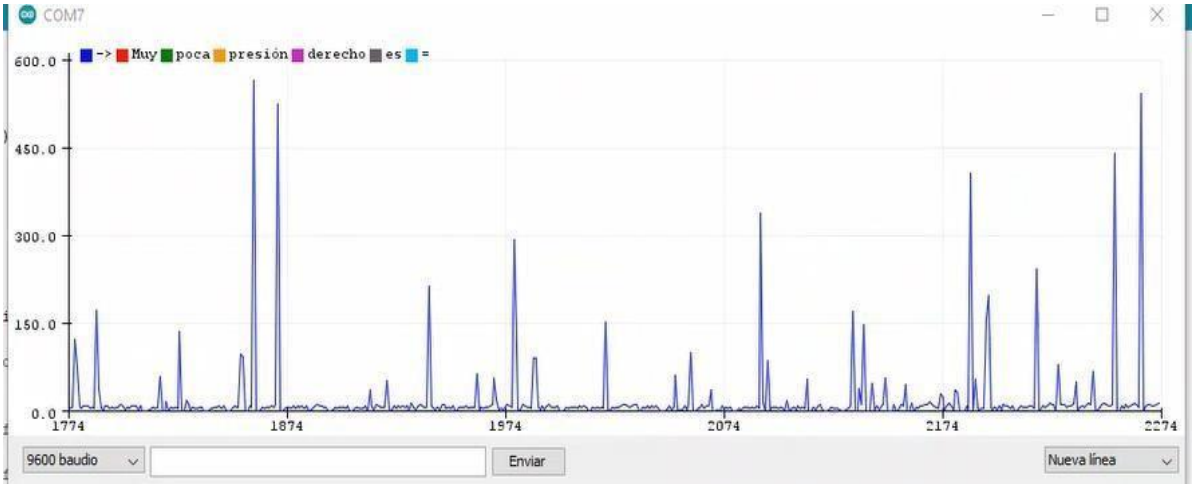


Figura 63. Diagrama 2 de datos 0000

En las figuras 62 y 63 se puede observar la intensidad y los picos de cada golpe en la primer pruena de tapping test con el prototipado físico. Los demás resultados se encuentran en el apartado Anexo2,

El día 14 de octubre se realiza las segundas pruebas con 18 estudiantes de Primer semestrede ingeniería electrónica.

Los datos se tomaron en el laboratorio de Física de la facultad de ingeniería en el sexto piso del edificio Holanda.

	Identificacion	Usuario	Contrasenia	Fecha_de_la_prueba	Nombres_Y_Apellidos
	Filter	Filter	Filter	Filter	Filter
1	1004640597	2221105031	2221105031	2022-10-14	Javier Pantoja
2	1004509228	2221105007	2221105007	2022-10-14	Firma
3	1004771247	2221105016	2221105016	2022-10-14	Firma
4	1014656374	2221105001	2221105001	2022-10-14	Danna Riaño
5	1019984559	2221105010	2221105010	2022-10-14	johan Cadena
6	1087114595	2221105034	2221105034	2022-10-14	Julian Andrade
7	1004216949	2221105021	2221105021	2022-10-14	Firma
8	1004437671	2221105027	2221105027	2022-10-14	Jose Quiroz
9	1004292373	2221105013	2221105013	2022-10-14	Andres Erazo
10	1004192568	2221105035	2221105035	2022-10-14	Sofia Castro
11	1004743499	2221105015	2221105015	2022-10-14	Javier Manso
12	1085340821	2221105024	2221105024	2022-10-14	Firma
13	1080690719	2221105012	2221105012	2022-10-14	Tian Tello
14	1193385742	2221105011	2221105011	2022-10-14	Anderson Figueroa
15	1089194094	2221105014	2221105014	2022-10-14	Cristian Gorra
16	1086498059	2221105025	2221105025	2022-10-14	Angie Estrada
17	1081053345	2221105032	2221105032	2022-10-14	Jefferson Enrique
18	1004622191	2221105018	2221105018	2022-10-14	Harold Foelagon

Figura 64. Base de datos participantes (SQL).

	Identificacion	Peso	Longitud_Brazo	Velocidad	Aceleracion	tiempo Prueba	Tiempo por Toque
	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1004640597	67 kg	0,55 m	0,04 m/s	0,00182732m/s2	21:89 Segundos	0,43 Segundos
2	1004509228	65 kg	0,55 m	0,04 m/s	0,00173611m/s2	23:04 Segundos	0,46 Segundos
3	1004771247	56 kg	0,55 m	0,04 m/s	0,00176134m/s2	22:71 Segundos	0,45 Segundos
4	1014656374	57 kg	0,50 m	0,04 m/s	0,00187882m/s2	21:29 Segundos	0,42 Segundos
5	1019984559	60 kg	0,55 m	0,05 m/s	0,00272628m/s2	18:34 Segundos	0,36 Segundos
6	1087714595	65 kg	0,55 m	0,04 m/s	0,00199203m/s2	20:08 Segundos	0,40 Segundos
7	1004216949	63 kg	0,55 m	0,04 m/s	0,00187793m/s2	21:30 Segundos	0,42 Segundos
8	1004437671	60 kg	0,55 m	0,04 m/s	0,00191113m/s2	20:91 Segundos	0,41 Segundos
9	1004192377	65 kg	0,55 m	0,04 m/s	0,00197726m/s2	20:23 Segundos	0,40 Segundos
10	1004192568	60 kg	0,50 m	0,04 m/s	0,00194458m/s2	20:57 Segundos	0,41 Segundos
11	1004743499	89 kg	0,55 m	0,04 m/s	0,00192678m/s2	20:76 Segundos	0,41 Segundos
12	1085340822	61 kg	0,55 m	0,08 m/s	0,00764818m/s2	10:46 Segundos	0,20 Segundos
13	108069019	71 kg	0,55 m	0,07 m/s	0,00611888m/s2	11:44 Segundos	0,22 Segundos
14	119338572	68 kg	0,55 m	0,07 m/s	0,00632911m/s2	11:06 Segundos	0,22 Segundos
15	1084194099	67 kg	0,55 m	0,07 m/s	0,00601892m/s2	11:63 Segundos	0,23 Segundos
16	1086498059	57 kg	0,50 m	0,06 m/s	0,00441826m/s2	13:58 Segundos	0,27 Segundos
17	1081053345	75 kg	0,55 m	0,05 m/s	0,00274725m/s2	18:20 Segundos	0,36 Segundos
18	1004629121	55 kg	0,55 m	0,04 m/s	0,00242718m/s2	16:48 Segundos	0,32 Segundos

Figura 65. Base de datos participantes1.1 (SQL).

En las figuras 64 y 65 se puede evidenciar las 18 pruebas de la segunda sesión, de acuerdo a los términos legales para el uso de la información personal se pide previa autorización para el uso de la misma.

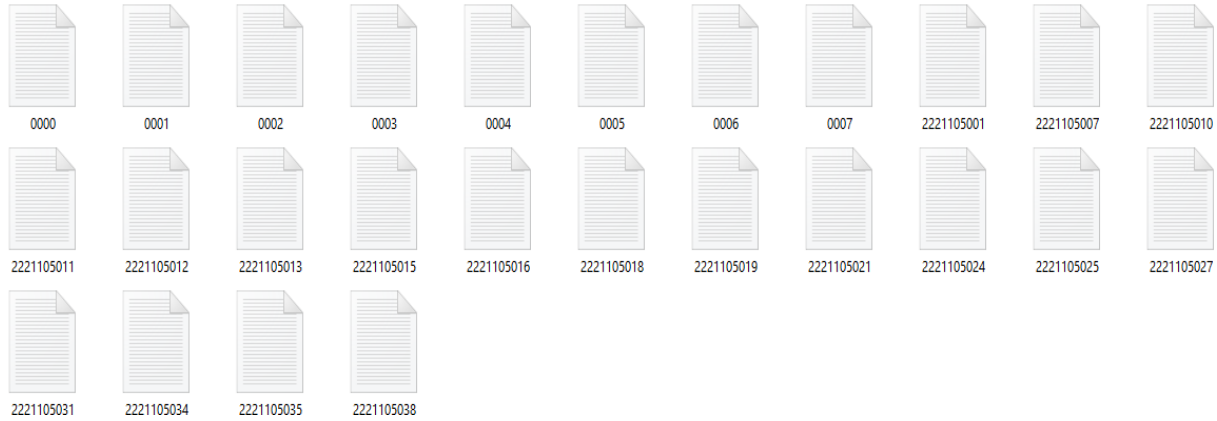


Figura 66. Base de almacenamiento datos sensores 1.1 (IDE).

En la figura 66 se puede evidenciar la base de almacenamiento del dispositivo encargado de almacenar los datos provenientes de las pruebas gaurdados de acuerdo al código de cada estudiante

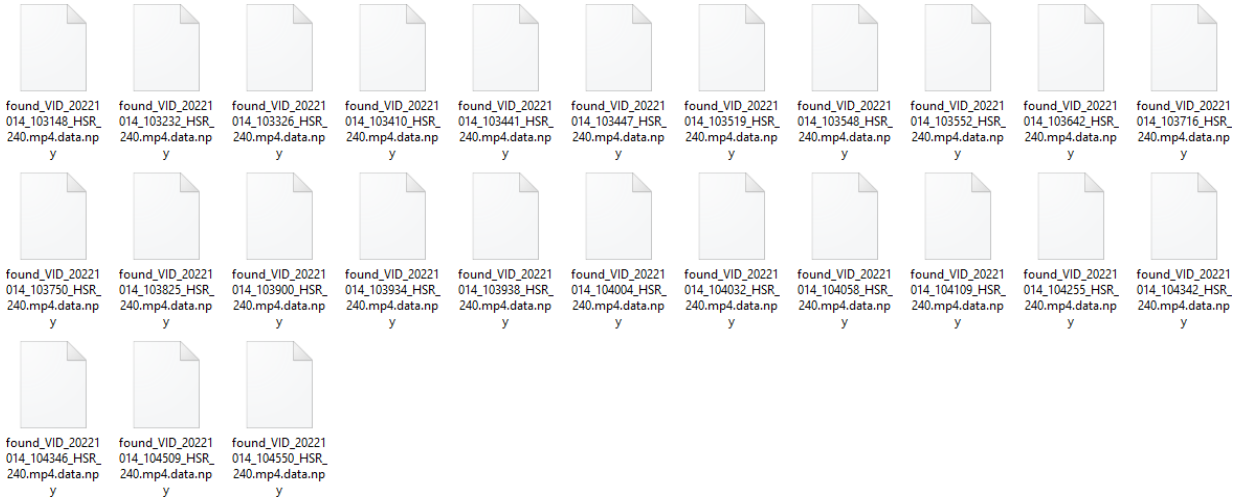
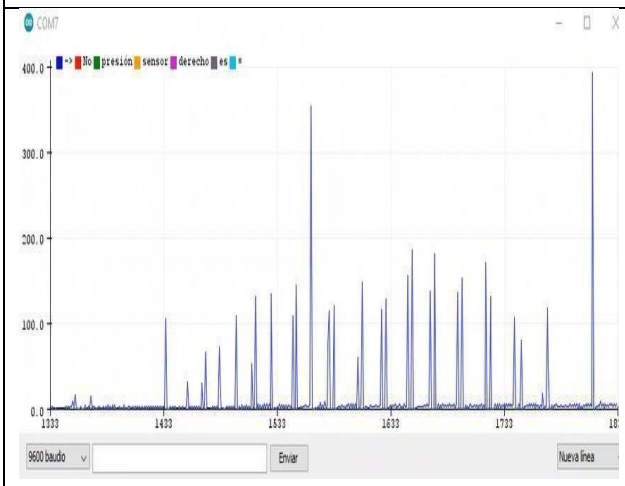
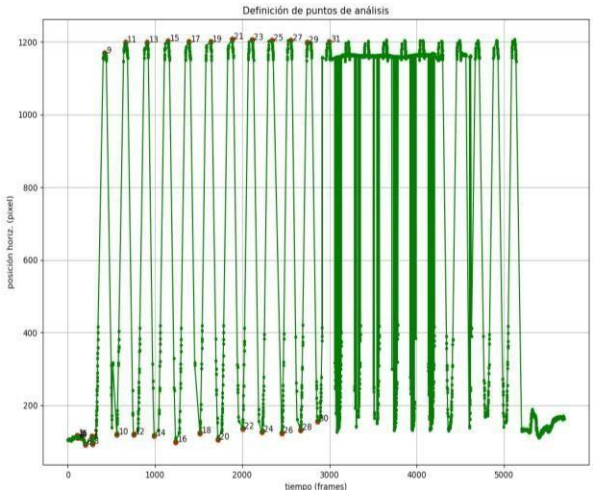


Figura 67. Base de almacenamiento datos de los videos preprocesador.

En la figura 67 se puede observar los archivos procesados a un tipo de archivo adecuado para ser analizado por el algoritmo procesador de videos.

Sensores	Visión artificial
	

Se puede evidenciar los toques registrados por las placas del tablero tapping test relacionada con los toques registrados por visión artificial, se registra una razonable similitud en los valores y datos obtenidos.Las demás pruebas se anexan en el apartado anexo3.

En las pruebas donde se evidencia que se están sobreponiendo puntos de toque, uno sobre otro, se tendría que verificar en el video original el posicionamientos del ejecutor frente al tablero de tapping test; ya que esto podría tener consecuencias en los resultados procesados de los mismos.

Esto produce que el algoritmo entre en conflicto, eso podría dar como resultado; tener un error alto en el procesamiento de imagen lo que conlleva a una velocidad no muy fiable.

Tabla.7 Masas brazos relación fuerza.

Código	Masa Brazo (kg)	Fuerza (N)
2221105031	3,9 kg	0,007126548N
2221105007	3,7 kg	0,006423607N
2221105016	2,8 kg	0,0040931752N
2221105001	2,9 kg	0,005448578N
2221105010	3,1 kg	0,0084568N
2221105034	3.7 kg	0,007370511N
2221105021	3.5 kg	0,006572755N
2221105027	3,1 kg	0,005924503N
2221105013	3.7 kg	0,007315862N
2221105035	3,1 kg	0,006028198N
2221105015	5,1 kg	0,009826578N
2221105024	3.2 kg	0,024474176N
2221105012	4,3 kg	0,026311184N
2221105011	4,6 kg	0,02531644N
2221105019	3,8 kg	0,022871896N
2221105025	2,9 kg	0,012812959N
2221105038	4,5 kg	0,012362625N
2221105018	2,6 kg	0,006310668N

En la anterior tabla se evidencian las casillas donde se organiza la información proveniente del código del estudiante, del peso del brazo de cada estudiante y la fuerza obtenida mediante el sistema sensorial prototipado,

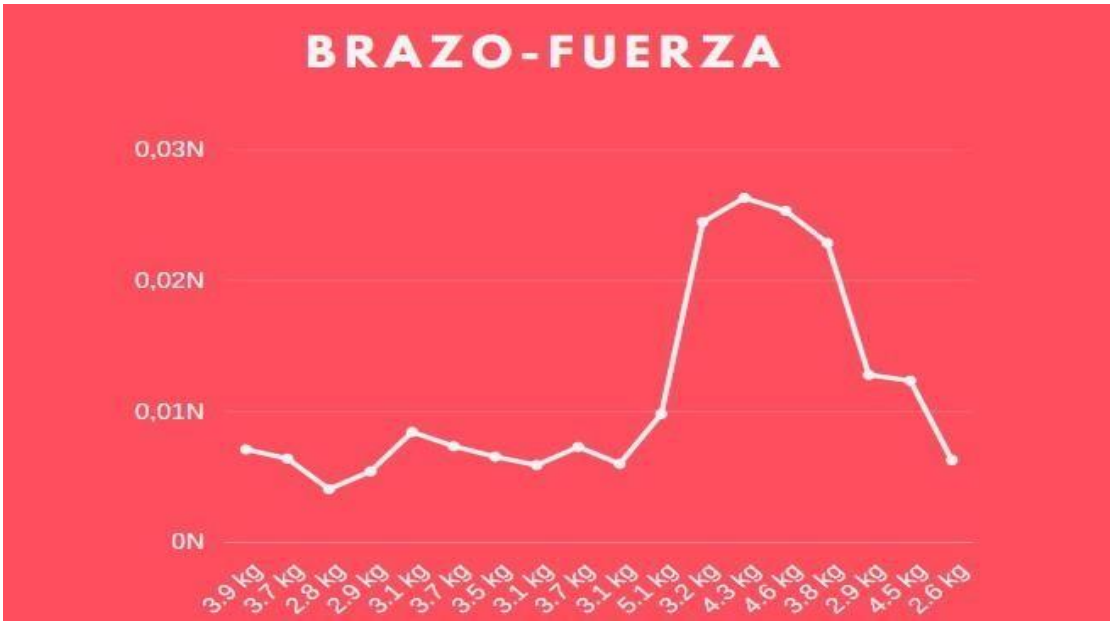


Figura 68. Comportamiento de fuerza entre las masas.

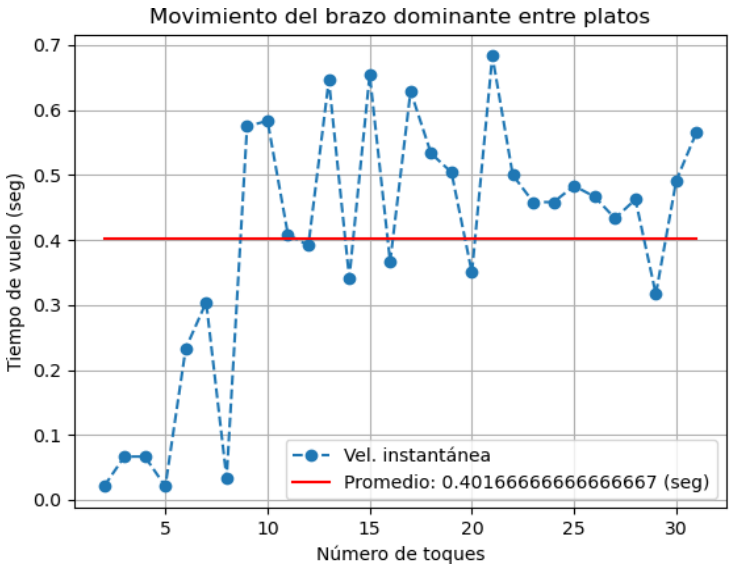
En la figura 68 se puede evidenciar el comporamiento del sistema frente a las 18 pruebas de tapping test.

Se puede observar que desde la prueba 11 a la prueba 16 se encuentran los datos mas altos de fuerza y presión obtenidas por la prieba tapping test.

Se puede analizar que en el rango del dato 11 y 16 se encuentran los valores mas alto en los pesos de los brazos del ejecutor.

4.4.2 Velocidad calculada mediante visión artificial

Tabla 7. Velocidad calculada a través de visión artificial.

Prueba 1: 2221105031	

4.4.3 Posicionamientos de la mano dominante con la respectiva proyección del movimiento

En las siguientes imágenes se podrá evidenciar la ilustración que provee el software gracias a la captura de cada toque en la prueba dispuesto por un frame del video, esto hace que se pueda obtener el posicionamiento y el comportamiento de cada mano dominante de los ejecutores.

Prueba 1:

2221105031

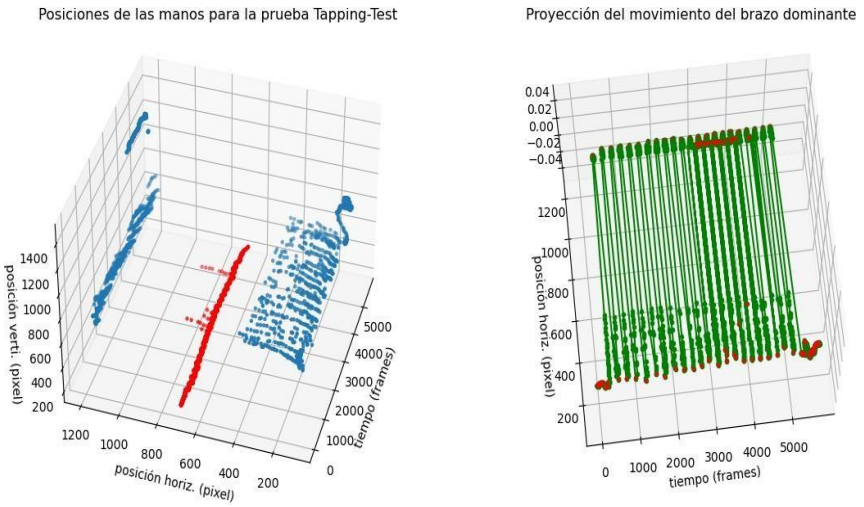


Figura 69. *Posicionamientos de la mano dominante con la respectiva proyección del movimiento1*.

En la figura 69 se puede observar el posicionamiento de la mano dominante del ejecutor tomada a través del video previamente procesado con la proyección del movimiento con respecto a los toques en los platos.

Tabla.8 Variables recolectadas por el sistema

Velocidad (m/s)Sistema	Aceleración (m/s2)sistema	Fuerza (N)sistema
0.43m/s	0.00182732 m/s2	0,007126548N
0.46m/s	0.00173611 m/s2	0,006423607N
0.45m/s	0.00176134 m/s2	0,0040931752N
0.42m/s	0.00187882 m/s2	0,005448578N
0.38m/s	0.00272628 m/s2	0,0084568N
0.40m/s	0.00199203 m/s2	0,007370511N
0.42m/s	0.00187793 m/s2	0,006572755N
0.41m/s	0.00191113 m/s2	0,005924503N
0.40m/s	0.00197726 m/s2	0,007315862N
0.41m/s	0.00194458 m/s2	0,006028198N
0.41m/s	0.00192678 m/s2	0,009826578N
0.20m/s	0.00764818 m/s2	0,024474176N
0.22m/s	0.00611888 m/s2	0,026311184N
0.22m/s	0.00632911 m/s2	0,02531644N
0.23m/s	0.00601892 m/s2	0,022871896N
0.27m/s	0.00441826 m/s2	0,012812959N
0.36m/s	0.00274725 m/s2	0,012362625N
0.32m/s	0.00242718 m/s2	0,006310668N

En la tabla anterior se evidencia los datos tales como la velocidad, aceleración y fuerza obtenidas por el portotipado físico.

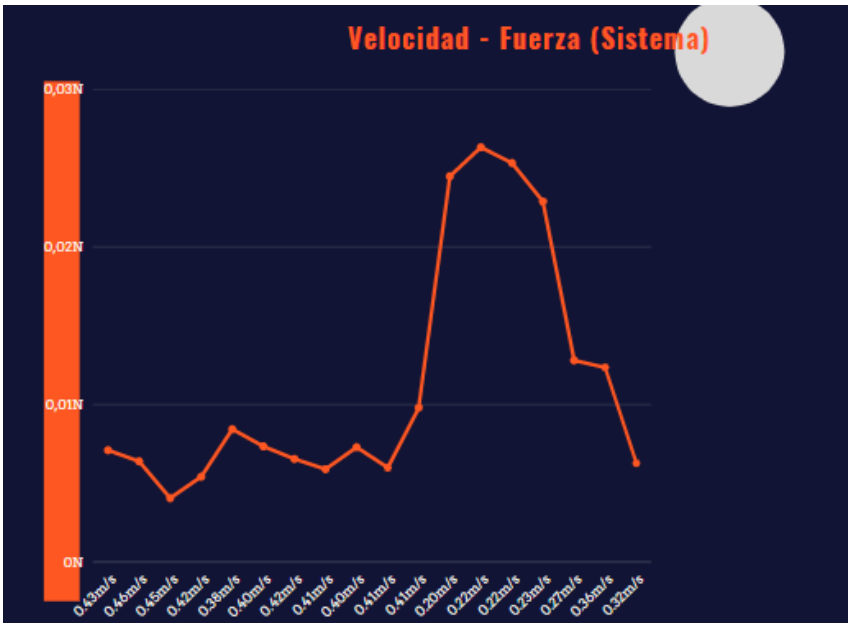


Figura 70. Velocidades con relación a la fuerza del sistema

En la figura 70 se evidendia el comportamiento de la velocidad con respecto a la fuerza del sistema.

Tabla 9. Variables recolectadas por Visión artificial

Velocidad (m/s)Visión Artificial	Aceleración (m/s2) Visión Artificial	Fuerza (N) Visión Artificial
0.40m/s	0.001 m/s2	0,0039N
0.38m/s	0.00105263 m/s2	0,003894731N
0.29m/s	0.00137932 m/s2	0,003862236N
0.64m/s	0.000625 m/s2	0,00188125N
0.41m/s	0.00097561 m/s2	0,003024391N
0.41m/s	0.00097561 m/s2	0,003609757N
0.41m/s	0.00187793 m/s2	0,003414635N
0.24m/s	0.00166667 m/s2	0,005166491N
0.35m/s	0.00114236 m/s2	0,004226732N
0.24m/s	0.00166667 m/s2	0,008500017N
0.20m/s	0.002 m/s2 m/s2	0,0064N
0.17m/s	0.00235294 m/s2	0,024426N
0.22m/s	0.00181818 m/s2	0,05818176N
0.24m/s	0.00166667 m/s2	0,007166681N
0.20m/s	0.002 m/s2 m/s2	0,0076N
0.22m/s	0.00181818 m/s2	0,05272722N
0.30m/s	0.00133333 m/s2	0,00599985N
0.25m/s	0.0016 m/s2	0.00416 N

En la tabla anterior se evidencia los datos obtenidos a través del procesamiento de imagen en visión artificial.

Se puede evidenciar datos tales como de velocidad, aceleración y fuerza.

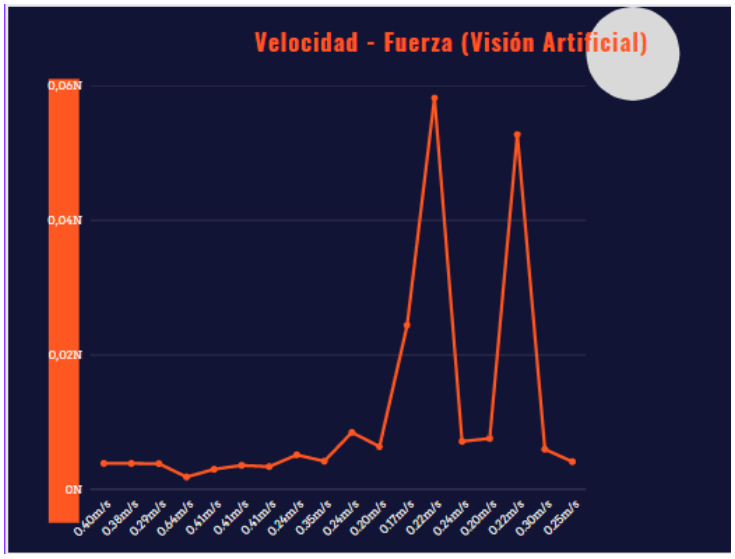


Figura 71. Velocidades con relación a la fuerza del sistema

En la figura 71 se puede evidenciar el comportamiento de la velocidad frente a la fuerza obtenida mediante el procesamiento de imagen con visión artificial, se evidencia similitud en cuanto a la del sistema físico.

Figura 72. Relaciones de la velocidad calculada a través de visión artificial y la fuerzacalculada a través del sistema.

En la figura 72 se puede observar la relación de la velocidad calculada a través de visión artificial y la fuerza calculada a través del sistema, convinando a si con efectividad ambos sistemas.

Tabla 10 comparación resultados

SUJETO	MASA BRAZO	VELOCIDAD (S)	FUERZA (S)	VELOCIDAD (V.A)	FUERZA (V.A)
2221105031	3,9 kg	0.43m/s	0,007126548N	0.40m/s	0,0039N
2221105007	3,7 kg	0.46m/s	0,006423607N	0.38m/s	0,003894731N
2221105016	2,8 kg	0.45m/s	0,0040931752N	0.29m/s	0,003862236N
2221105001	2,9 kg	0.42m/s	0,005448578N	0.64m/s	0,00188125N
2221105010	3,1 kg	0.38m/s	0,0084568N	0.41m/s	0,003024391N
2221105034	3.7 kg	0.40m/s	0,007370511N	0.41m/s	0,003609757N
2221105021	3.5 kg	0.42m/s	0,006572755N	0.41m/s	0,003414635N
2221105027	3,1 kg	0.41m/s	0,005924503N	0.24m/s	0,005166491N
2221105013	3.7 kg	0.40m/s	0,007315862N	0.35m/s	0,004226732N
2221105035	3,1 kg	0.41m/s	0,006028198N	0.24m/s	0,008500017N
2221105015	5,1 kg	0.41m/s	0,009826578N	0.20m/s	0,0064N
2221105024	3.2 kg	0.20m/s	0,024474176N	0.17m/s	0,024426N
2221105012	4,3 kg	0.22m/s	0,026311184N	0.22m/s	0,05818176N
2221105011	4,6 kg	0.22m/s	0,02531644N	0.24m/s	0,007166681N
2221105019	3,8 kg	0.23m/s	0,022871896N	0.20m/s	0,0076N
2221105025	2,9 kg	0.27m/s	0,012812959N	0.22m/s	0,05272722N
2221105038	4,5 kg	0.36m/s	0,012362625N	0.30m/s	0,00599985N
2221105018	2,6 kg	0.32m/s	0,006310668N	0.25m/s	0.416 N

En la anterior tabla se puede evidenciar los datos organizados tales como velocidades, aceleraciones, fuerzas y códigos.

5. Análisis de resultados

En el siguiente apartado se van analizar diferentes ítems los cuales conllevaron a que se produjera un error en la obtenicion de los resultaos. Tales como problemas técnicos, físicos, personales, etc.

- El posicionamiento de la cámara frente al objeto de estudio que en este caso era el estudiante con el tablero se recomienda hacer la captura desde la parte de arriba y no de frente ya que la captación de imagen para el procesamiento en visión artidicial se vuelve un poco mas sencilla y amigable con el entorno virtual.
- El sensor piezoeléctrico dentro de su documentación y la datasheet muestra un error +/- del 12 % de error de fabrica.
- La estación de video deberá tener correciones de postura en el tablero ya que al momento de procesar el video, las esquinas fuera de cotexto en las pruebas pueden llegar hacer interferencias de los datos que se quieren obtener,
- La espuma con la que se relleno el tablero con respecto a los discos puede llegar a ser un impedimento para que los valores recibidos por el sensor no sea del todo claro, se deberá investigar un material el cual sea capaz de resistencia de fuerza y presión que a su vez no interfiera con el sensor mismo.
-

Se recolectan los siguientes datos organizados en la tabla siguiente, se comparan los resultados obtenidos por el prototipo físico y el algoritmo en visión artificial.

Tabla 11 error porcentual

Sujeto	Fuerza Sistema	Fuerza Visión artificial	Error Porcentual
2221105031	0,007126548N	0,0039N	45.27 %
2221105007	0,006423607N	0,003894731N	39.36 %
2221105016	0,0040931752N	0,003862236N	5.64 %
2221105001	0,005448578N	0,00188125N	65.47 %
2221105010	0,0084568N	0,003024391N	64.23 %
2221105034	0,007370511N	0,003609757N	51.02 %
2221105021	0,006572755N	0,003414635N	48.04 %
2221105027	0,005924503N	0,005166491N	12.79 %
2221105013	0,007315862N	0,004226732N	42.22 %
2221105035	0,006028198N	0,008500017N	41.00 %
2221105015	0,009826578N	0,0064N	34.87 %
2221105024	0,024474176N	0,024426N	0.19 %
2221105012	0,026311184N	0,05818176N	54.77 %
2221105011	0,02531644N	0,007166681N	71.69 %
2221105019	0,022871896N	0,0076N	66.77 %
2221105025	0,012812959N	0,05272722N	75.69 %
2221105038	0,012362625N	0,00599985N	51.46 %
2221105018	0,006310668N	0.00416 N	34.07 %

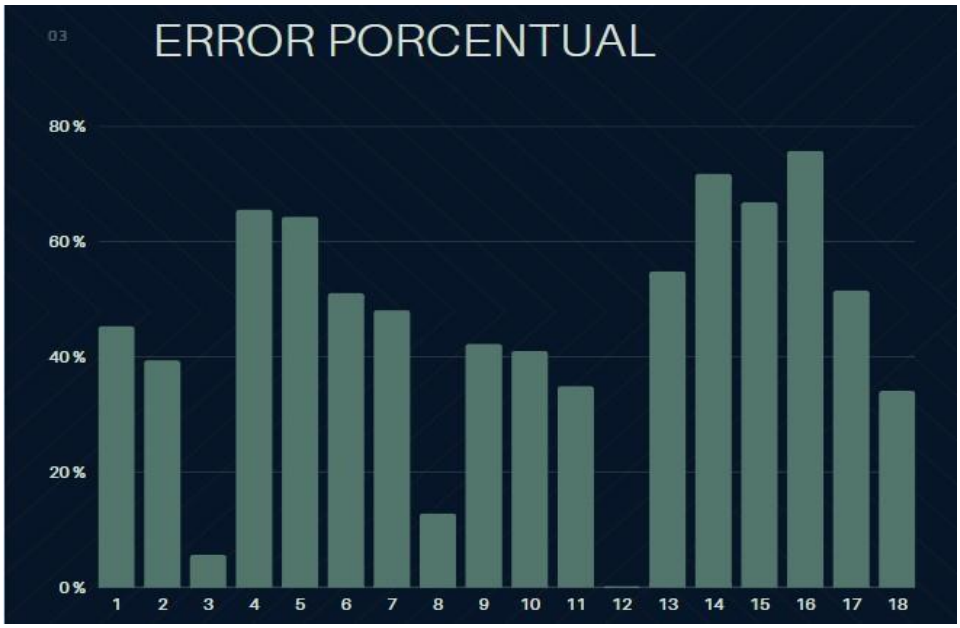


Figura 73. Estadística sobre el error porcentual relativo.

El error porcentual aproximado en el algoritmo de visión artificial aplicado en el sistema Tapping test es de 44.66%

Se considera que los Intervalos es un tema relevante y que se encuentra frecuentemente en los gráficos de error y comparaciones en los sitemas esto ayuda a la ubicación de las secciones a mejorar en el sistema implementado. Por ello mismo es uno de los temas bases para desarrollar los temas e ingestigacion venidera.

Recomendaciones

- En la elección de la placa de desarrollo se realiza pruebas con arduino UNO y con ESP8266 las respuestas son buenas aunque no satisfacen la necesidad por ende se escoge ESP32 las lecturas análogas y las transmisión de datos trabajando al mismo tiempo no interfieren entre sí, por ende esto conlleva a un trabajo fluido, tener en cuenta fabricantes, ya que al igual que con la elección de placas existen muchas referencias en este caso se hicieron pruebas con ESP WROOM 32, ESPRESSIF ESP32 y ESP32 wroom 32u, se realiza prototipo con la última referencia.
- El Angulo de la cámara en el posicionamiento frente al tablero y el ejecutante se llega a los términos sobre los cuales se tiene que tener una amplia referencia visual para que así el procesamiento de la imagen se comporte correcto con el programa.
- La caracterización de los parámetros para los sensores se ven variables las cuales deberán ser ajustados de acuerdo a las necesidades del ejercicio para así con esto poder automatizar un producto final.
- La altura en la caracterización de los sensores tendrá que ser menor para así tener respuestas más rápidas sin desperdiciar información en el resultado, minimizar resultados para la mejor posible apreciación y lectura.
- Tener en cuenta para producto final la instalación de lector de memoria SD en placa así con esto se tendría un almacenamiento de datos externos sin necesidad de utilizar el de la placa y así no interrumpir el flujo de los mismos, elimina el componente de tener que tener un PC conectado.
- Se recomienda recortar previamente el video en los momentos exactos del inicio de ejercicio así esto facilitara el poder encontrar los fotogramas adecuados para el buen uso del programa.

Conclusiones

- Este proyecto de grado tuvo como objetivo trabajar a lo largo de todo su abordaje con una corriente de pensamiento que supone diferentes además de amplios momentos de no respuesta positiva como respuesta de un pensamiento en los cuales repercuten o afectan múltiples factores tales como (Familiares, sociales, educativos, económicos, etc.). Además, esta tesis ha demostrado como encarecer el carácter sintomático de una determinada elección. Dicha elección tiene como condición adquirir una cualidad de madurez suficiente para que pueda ser llevada a cabo a favor de una decisión, y esto gracias a las diferentes formas de aplicación de métodos, los cuales nos ayudaran a llegar a ciertas conclusiones, tales como la que se está leyendo.
- En la figura 76 se ve como el comportamiento regular de la fuerza a la respuesta de ciertas masas se ve afectada cuando la velocidad de esta misma aumenta.: a mayor velocidad mayor va a ser la fuerza resultante.
- La etapa de saturación de la entrada de la respuesta del sensor se ve en estado de iniciación cuando la masa aumenta en los 42grf, esto afecta la señal y por ende llena los archivos de datosbasuras, tener en cuenta que se deberá tener un amplificador de voltaje al final de la salida del mismo, esto hará que aumente la respuesta y disminuya el ruido.
- El error en la estimación de la fuerza obtenida por el algoritmo de visión artificial esta entre un 30% y un 50%
- La investigación es una hipótesis alternativa.
- Para un buen análisis de imagen sobre el video a procesar tener en cuenta el primer fotograma y el ultimo fotograma, eso alterara de manera beneficiosa o no y dependerá de en qué punto se quiere tener el procesamiento de imagen.
- La espuma debajo de la superficie de la lamina de toques en el tablero puede llegar afectar el resultado de los sensores, tener encuenta un material el cual cumpla la función de la misma y no interfiera en la lectura de los sensores.

.

Referencias

Bačić, B., y Hume, P. A. (2018). Computational intelligence for qualitative coaching diagnostics: Automated assessment of tennis swings to improve performance and safety. *Big Data*, 6(4), 291–304. https://doi.org/10.1089/big.2018.0062

D. (2012). No Titleיוויקה ץנעת תנומת :בצמ .עטונה זולע, 66, 37–39.

Development and standardisation of the computerised finger tapping test: Comparison with other finger tapping instruments. *Wiley Oline Library*, 83 (1), 9-13. https://doi.org/10.1111/j.1600-0404.1991.tb03952.x

Energía, L. P. Y. (n.d.). *Universidad cesmag*.

Gary, P. H.; J. B. M.; S. C. T. et al. (1990). The New England Journal of Medicine Downloaded from nejm.org on April 1, 2015. For personal use only. No other uses without permission. Copyright © 1990 Massachusetts Medical Society. All rights reserved. *The New English Journal of Medicine*, 323(16), 1120–1123.

Jos, A. (2019). *Medición y evaluación de la condición física : batería de test Descripción : no dominante*. 1–7.

Lucchini, R., Selis, L., Folli, D., Apostoli, P., Mutti, A., Vanoni, O., Iregren, A., y Alessio, L. (1995). Neurobehavioral effects of manganese in workers from a ferroalloy plant after temporary cessation of exposure. *Scandinavian Journal of Work, Environment and Health*, 21(2), 143–149. https://doi.org/10.5271/sjweh.1369

Lv, D., Huang, Z., Sun, L., Yu, N., y Wu, J. (2017). Smart motion reconstruction system for golf swing: a DBN model based transportable, non-intrusive and inexpensive golf swing capture and reconstruction system. *Multimedia Tools and Applications*, 76(1), 1313–1330. https://doi.org/10.1007/s11042-015-3102-7

Malawski, F., & Kwolek, B. (2018). Recognition of action dynamics in fencing using multimodal cues. *Image and Vision Computing*, 75, 1–10. https://doi.org/10.1016/j.imavis.2018.04.005

Mariana Haro, D. (2014). Laboratorio de análisis de marcha y movimiento. *Revista Médica Clínica Las Condes*, 25(2), 237–247. https://doi.org/10.1016/s0716-8640(14)70034-3

Mendo, A. H., Sánchez, V. M., & Morales, V. G. (2011a). Finger taping test. Precisión del diseño de medidas entre muestras de deportistas de elite y no deportistas. *Cuadernos de Psicologia Del Deporte*, 11(1), 29–43.

Mendo, A. H., Sánchez, V. M., & Morales, V. G. (2011b). Finger tapping test. Precision of the measurement design between samples of elite athletes and not athletes. *Cuadernos de Psicologia Del Deporte*, 11(1), 29–43.

Molina Ruiz, H. D. (2010). Escuela Superior Tepeji del Río. *Escuela Superior de Tepeji*, 45–55.

Programa de Ingenieria Electronica. (2015). *Proyecto Educativo del Programa.*

Rojas Montes, J. (2012). Reseña: Procesamiento digital de imágenes con MatLAB y SIMULINK. *Sistemas y Telemática*, 10(21), 77. https://doi.org/10.18046/syt.v10i21.1198

Russell, EW, Neuringer, C. y Goldstein, G. (1970). Evaluación del daño cerebral: un enfoque neuropsicológico clave. *Wiley-Interscience*. Recuperado de: https://psycnet.apa.org/record/1971-04775-000

Sanabria S., John J.; Archila D., J. F. (2011). *Detección y análisis de movimiento usando visión artificial Motion Detection and Analysis Using Artificial Vision.*

Tamime, A. (2019). No TitleΕΛΕΝΗ. Αγαη, 8(5), 55. Recuperado de: https://www.tandfonline.com/doi/abs/10.1207/s15324826an1102_5

Anexos

Anexo 1. Tapping Test

https://mega.nz/file/HNonwQ5S#GgCCxDY4obWplT4ckkls3FRHrCkIs5O6uKyl92G8eNc

Anexo 2. Datos primeras pruebas del prototipo físico.

Prueba 0001

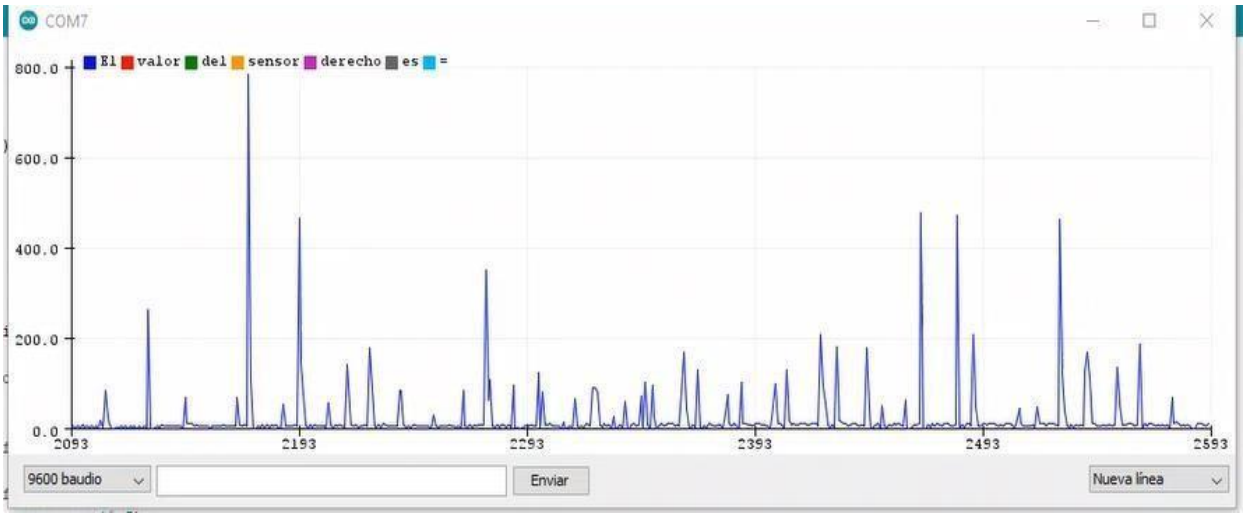


Figura 74. Diagrama de datos 0001

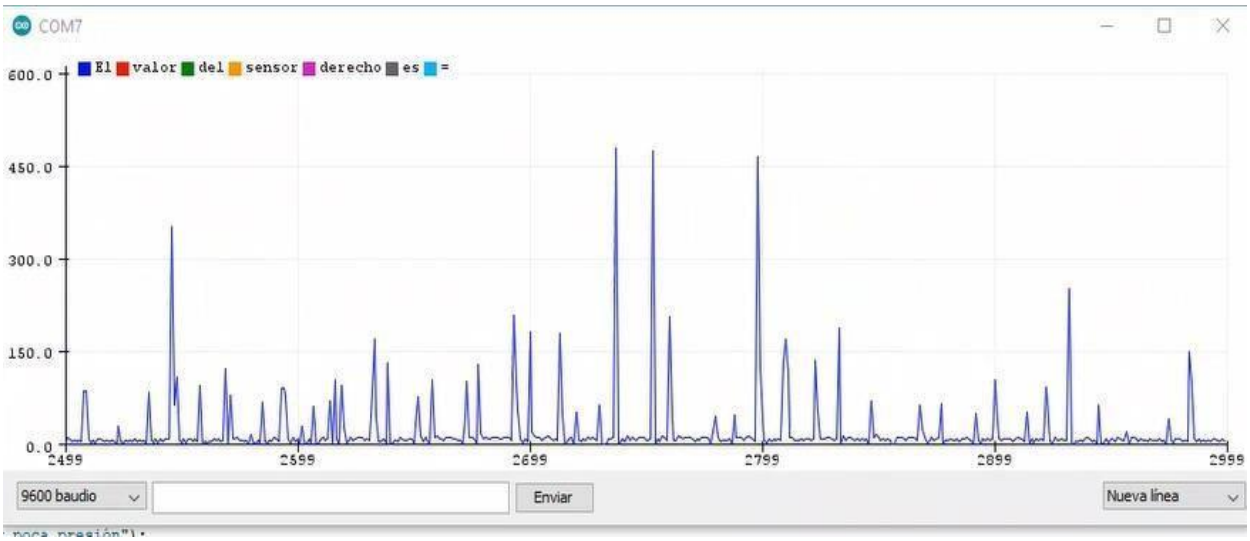


Figura 75. Diagrama 2 de datos 0001

Prueba 0002

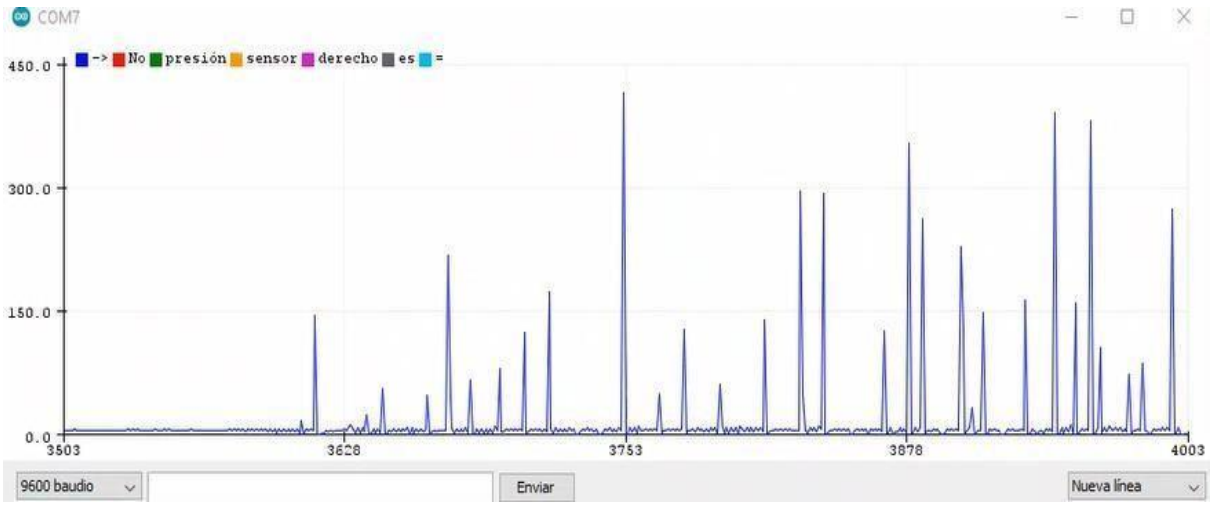


Figura 76. Diagrama de datos 0002.

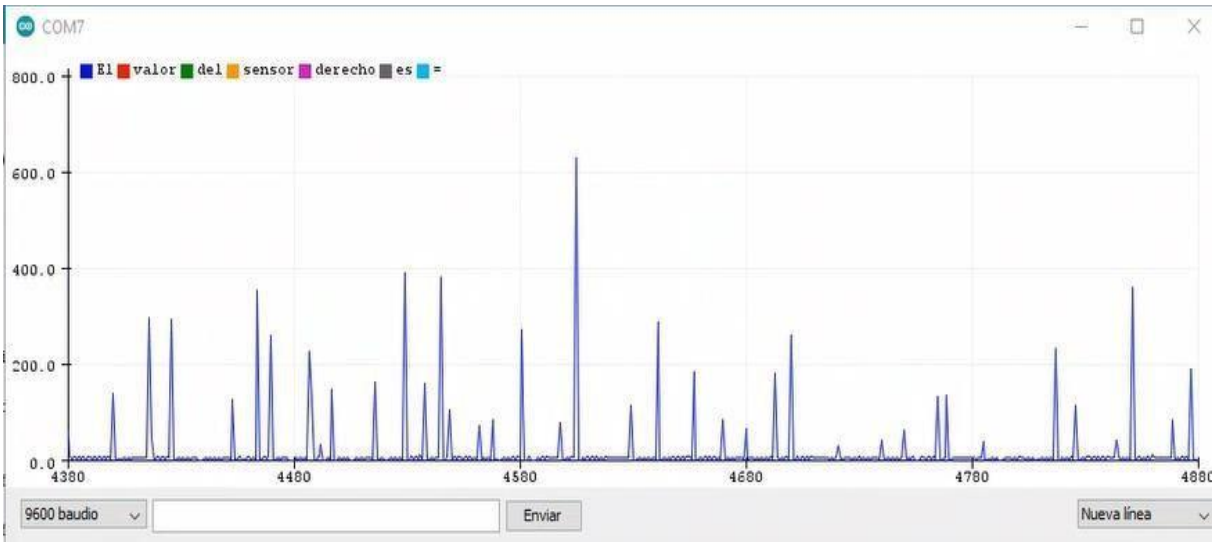


Figura 77. Diagrama 2 de datos 0002.

Prueba 0003

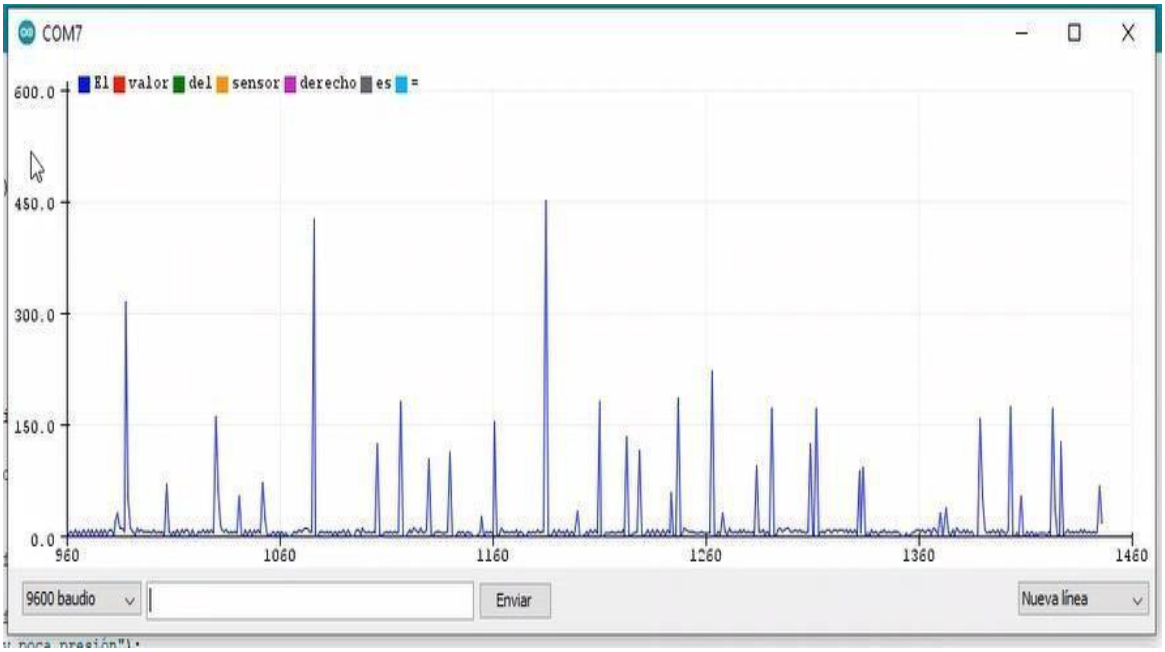


Figura 78. Diagrama de datos 0003.

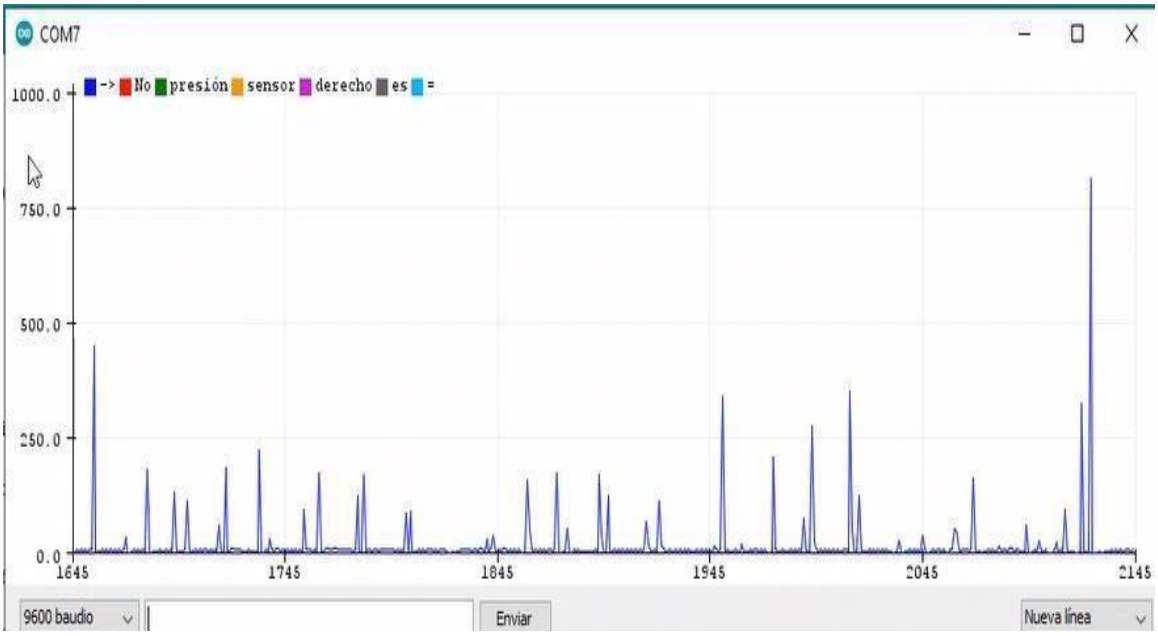


Figura 79. Diagrama 2 de datos 0003.

Prueba 0004

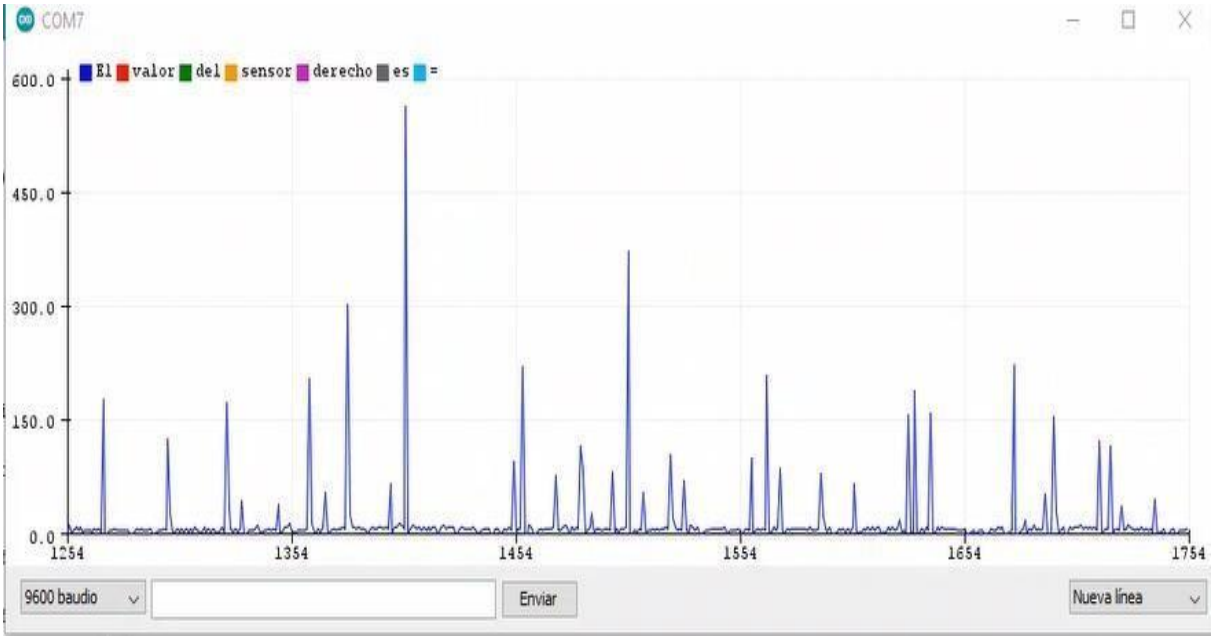


Figura 80. Diagrama de datos 0004.

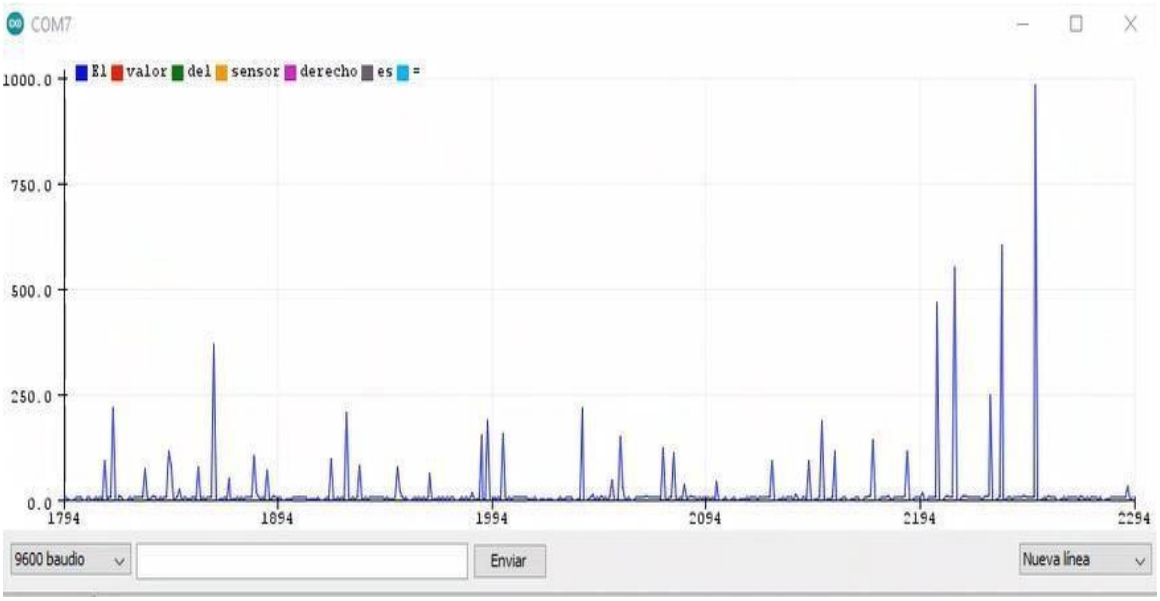


Figura 81. Diagrama 2 de datos 0004.

Prueba 0005

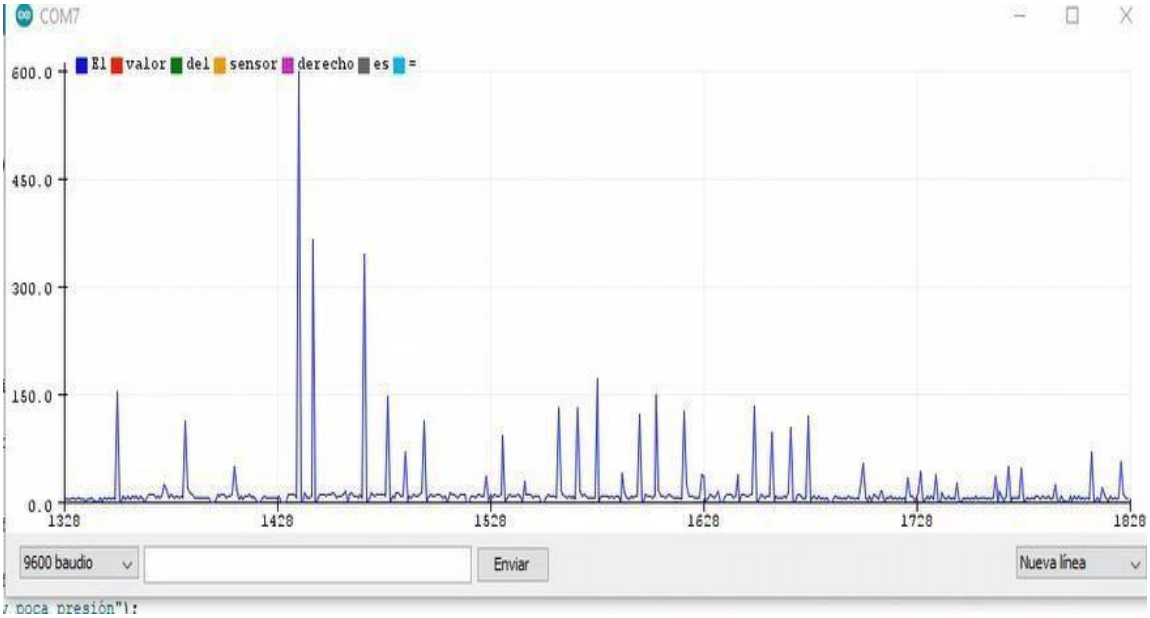


Figura 82. Diagrama de datos 0005.

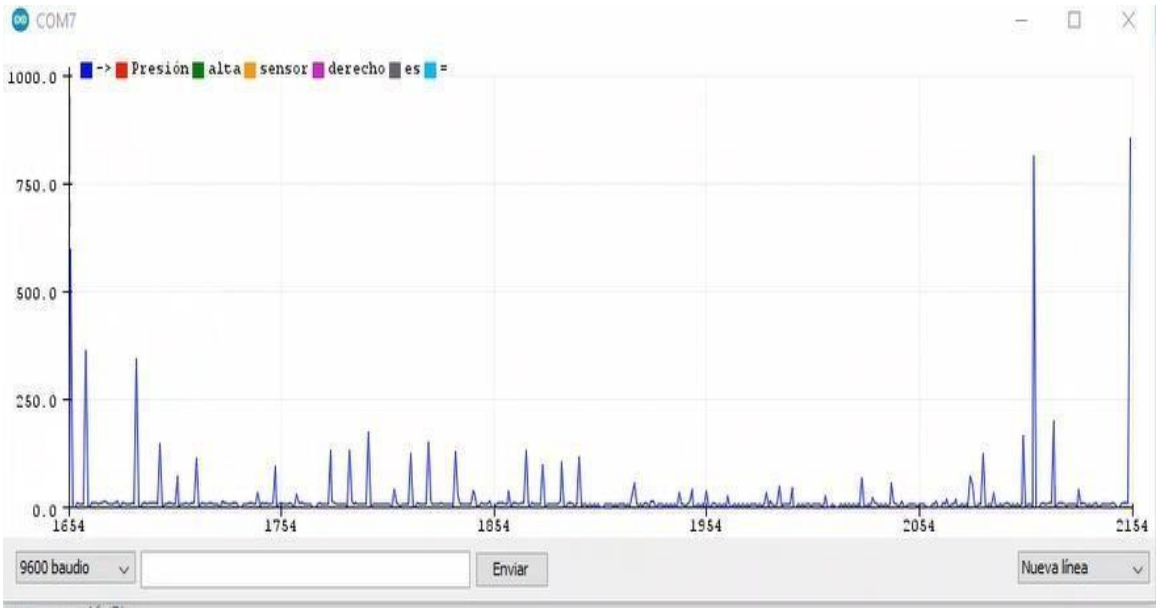


Figura 83. Diagrama 2 de datos 0005.

Prueba 0006

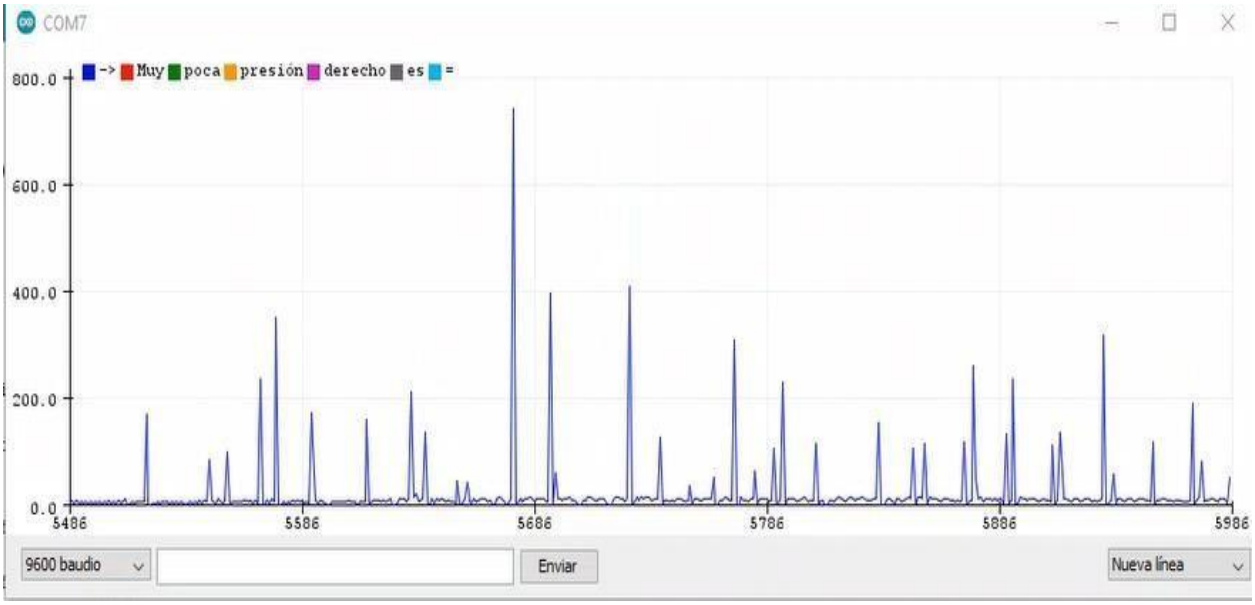


Figura 84. Diagrama de datos 0006.

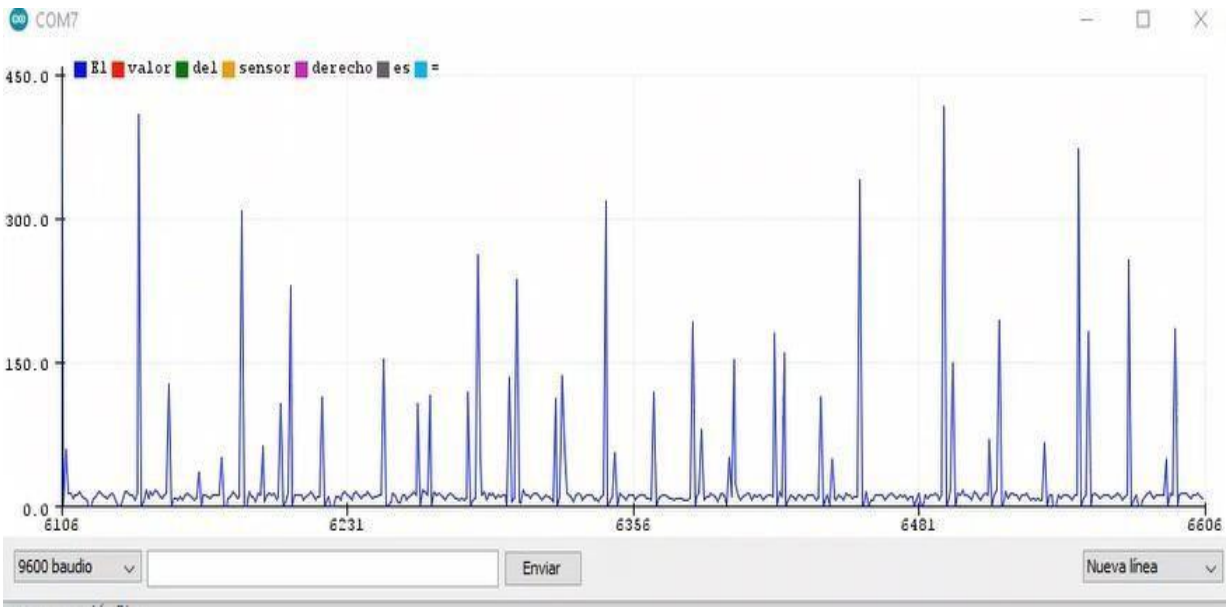


Figura 85. Diagrama 2 de datos 0006.

Prueba 0007

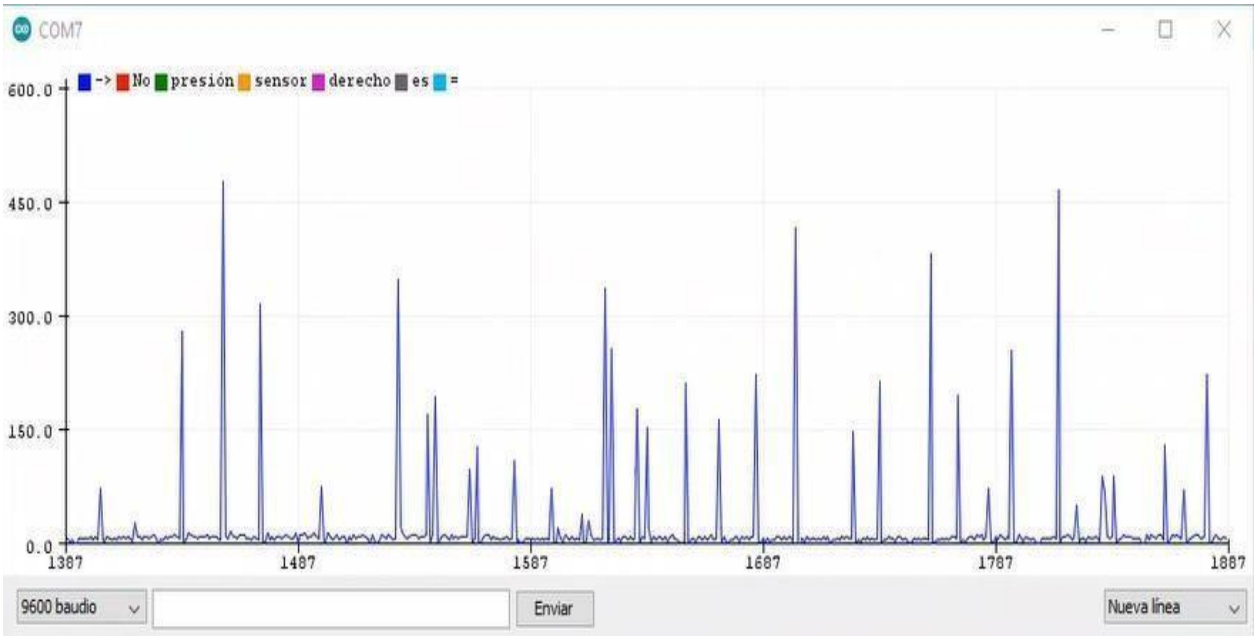


Figura 86. Diagrama de datos 0007.

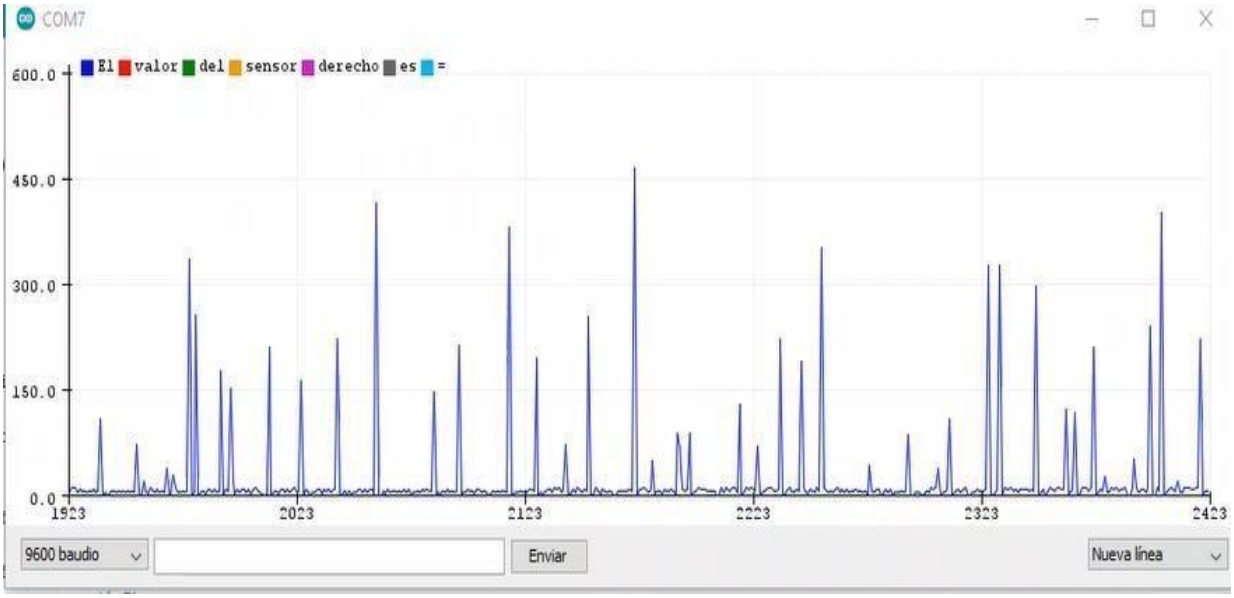
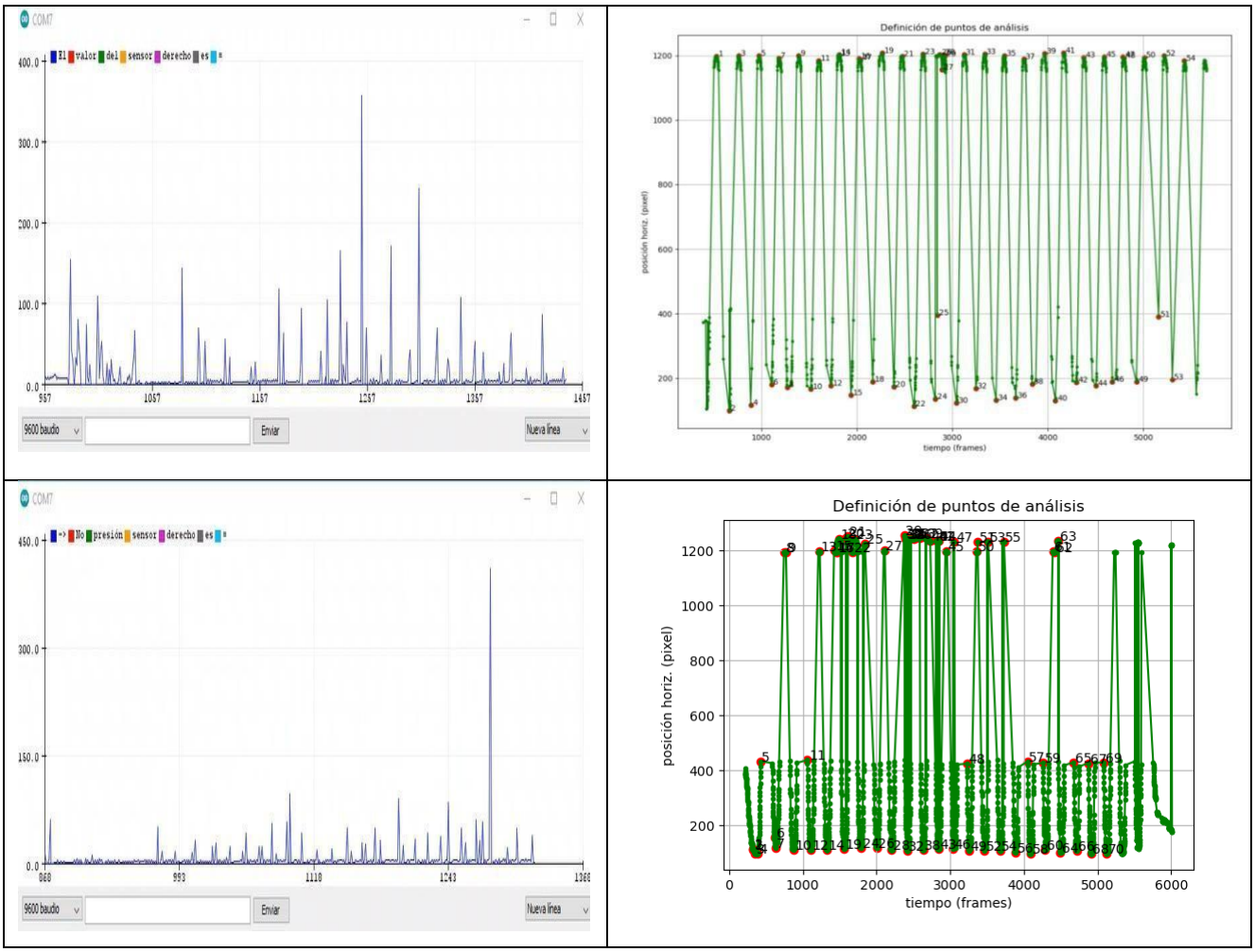
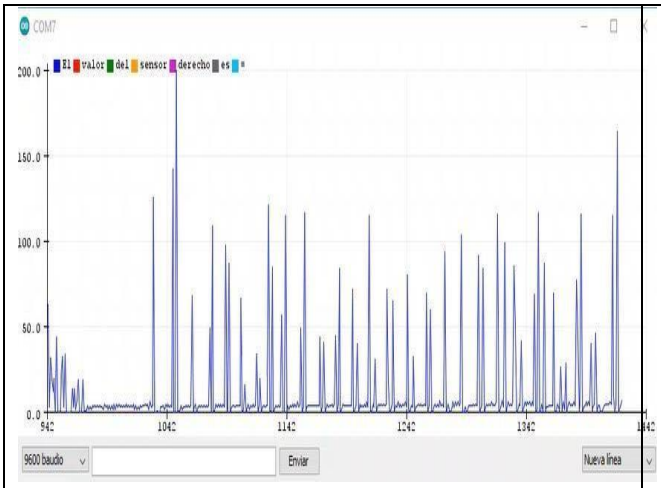
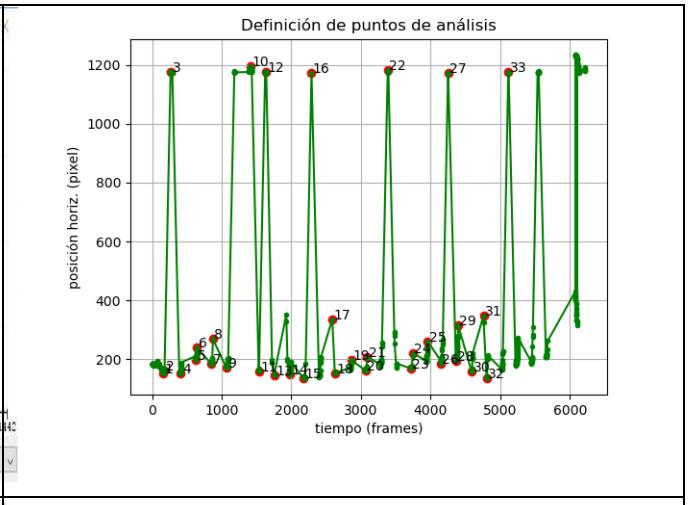


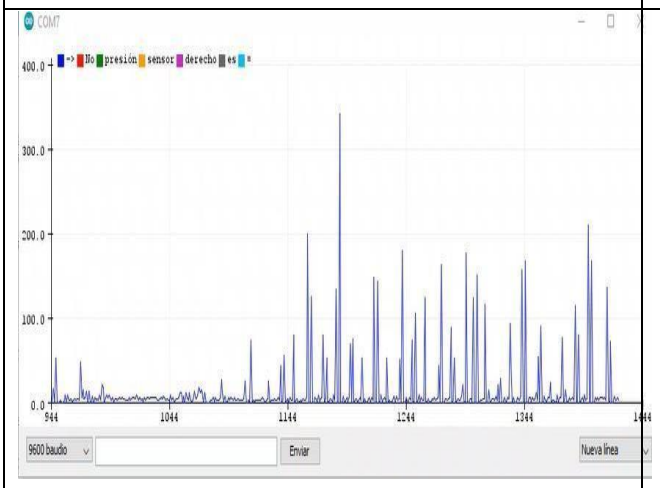
Figura 87. Diagrama 2 de datos 0007.

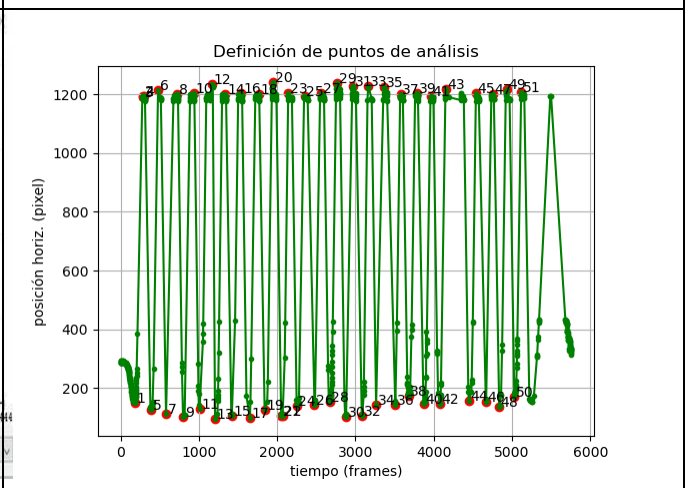
Anexo 3. Datos segunda pruebas con el prototipo fiico y cirtual.

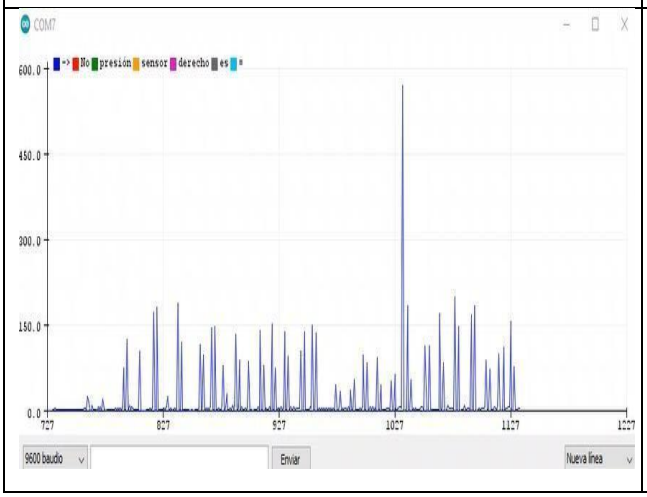


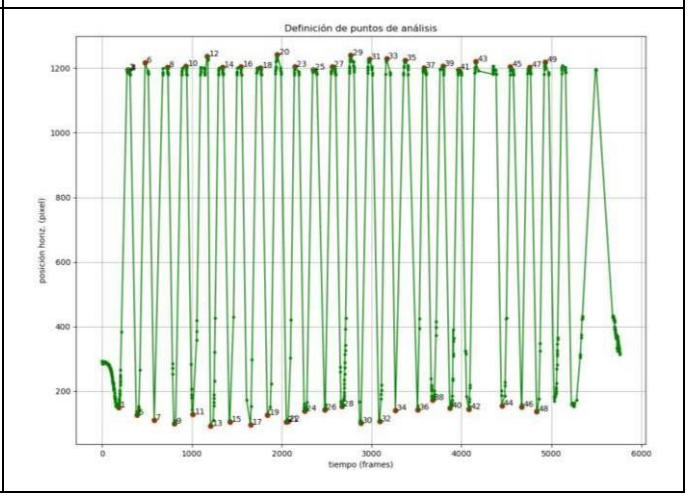

COM7
El valor del sensor derecho es
9600 baudio
Enviar
Nueva línea

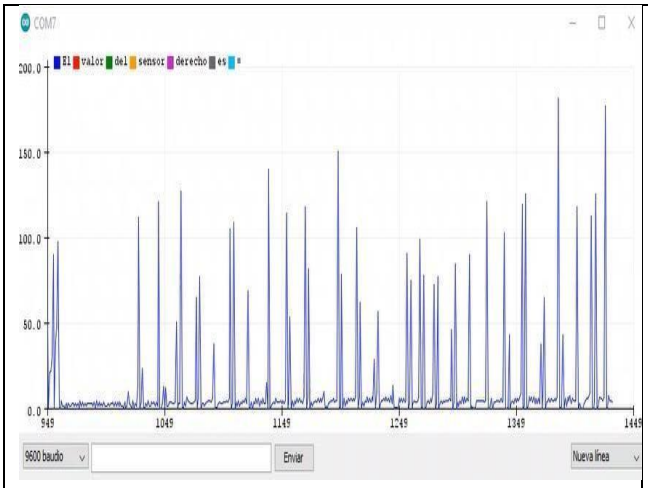

Definición de puntos de análisis
posición horiz. (pixel)
tiempo (frames)

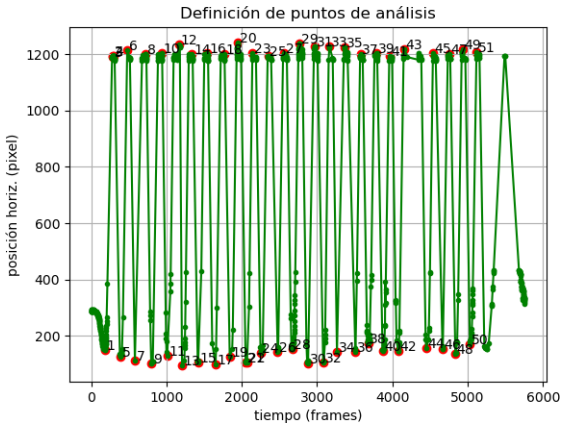

COM7
=> No presión sensor derecho es
9600 baudio
Enviar
Nueva línea

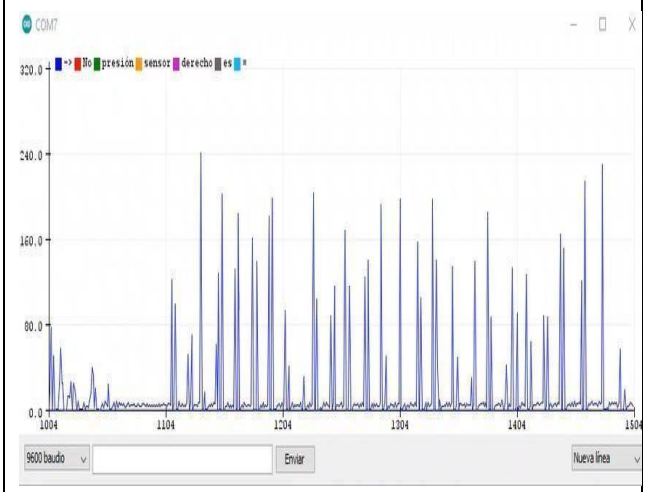

Definición de puntos de análisis
posición horiz. (pixel)
tiempo (frames)

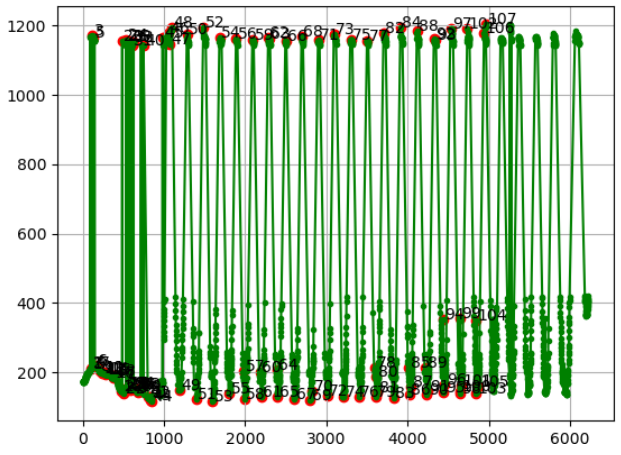

COM7
=> No presión sensor derecho es
9600 baudio
Enviar
Nueva línea

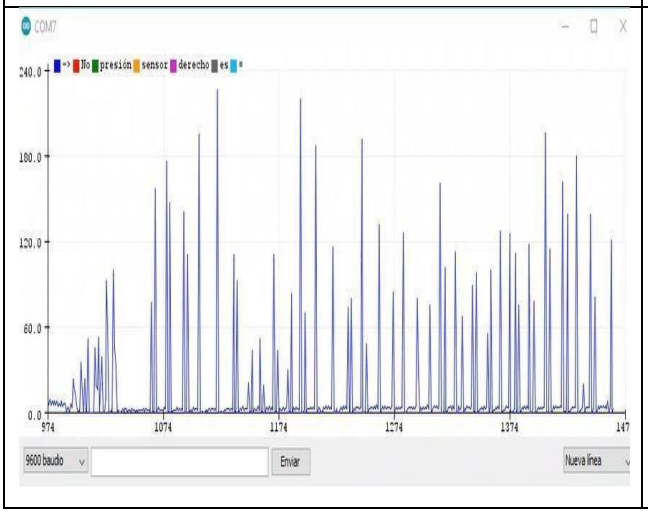

Definición de puntos de análisis
posición horiz. (pixel)
tiempo (frames)

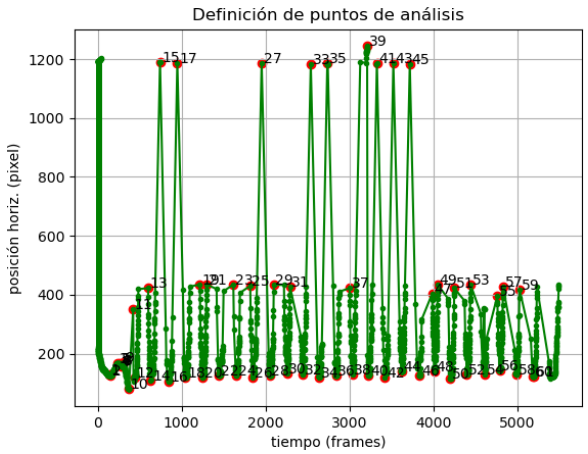

COM7
El valor del sensor derecho es
9600 baudio
Enviar
Nueva línea

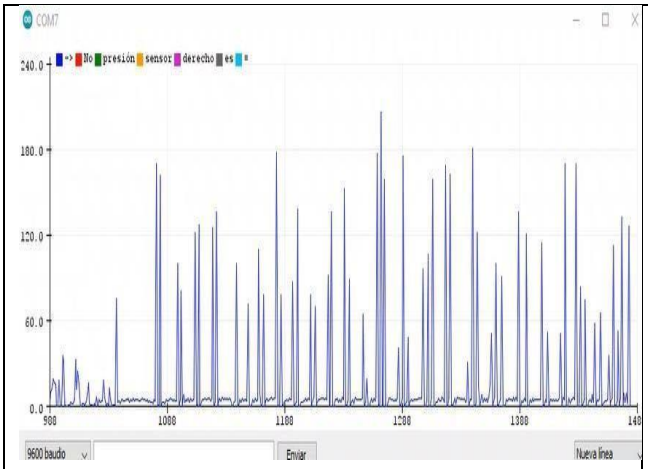

Definición de puntos de análisis
posición horiz. (pixel)
tiempo (frames)

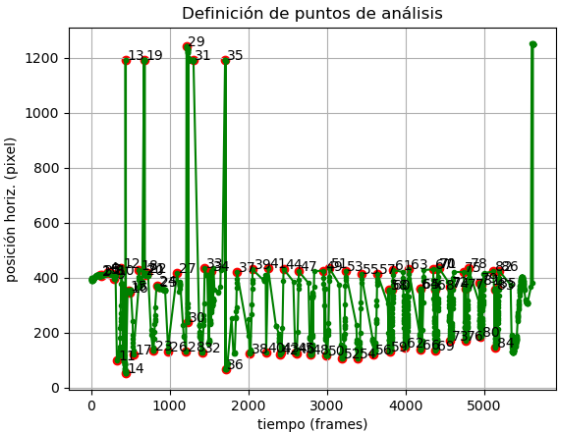

COM7
-> No presión sensor derecho es
9600 baudio
Enviar
Nueva línea

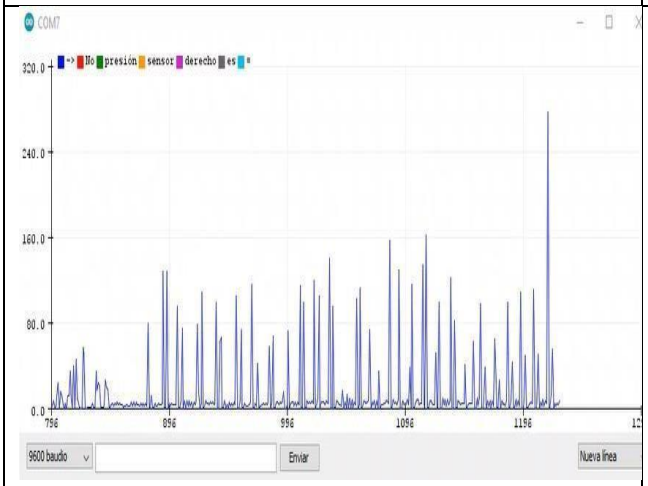


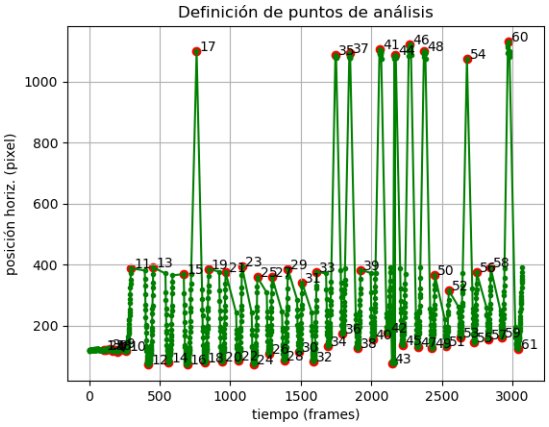

COM7
-> No presión sensor derecho es
9600 baudio
Enviar
Nueva línea

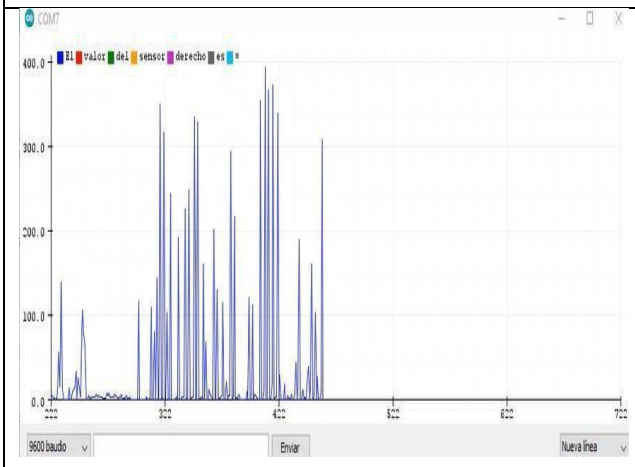

Definición de puntos de análisis
posición horiz. (pixel)
tiempo (frames)

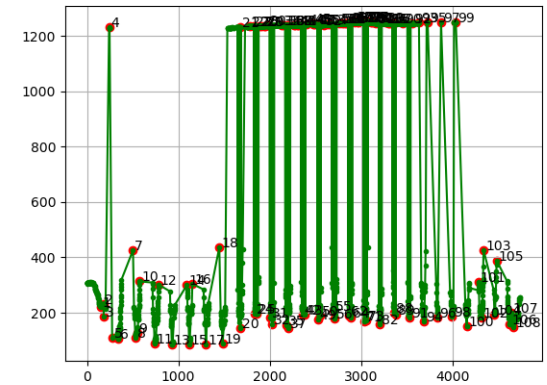

COM7
9600 baudo
Enviar
Nueva línea

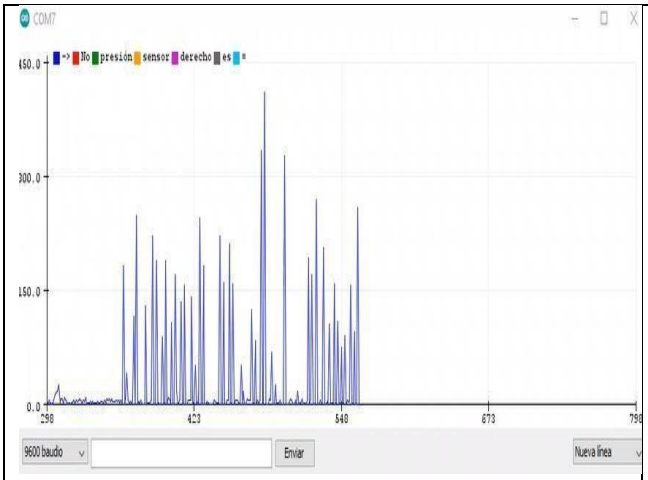

Definición de puntos de análisis
posición horiz. (pixel)
tiempo (frames)

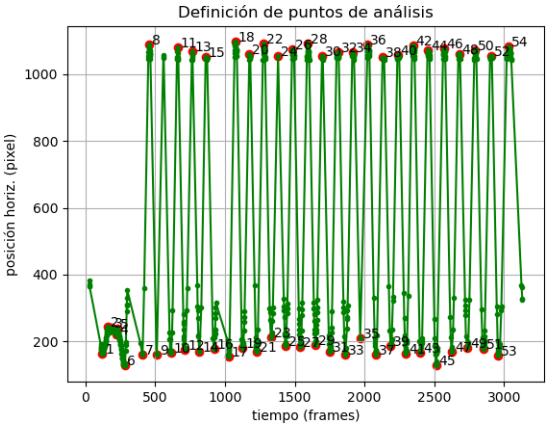

COM7
9600 baudo
Enviar
Nueva línea

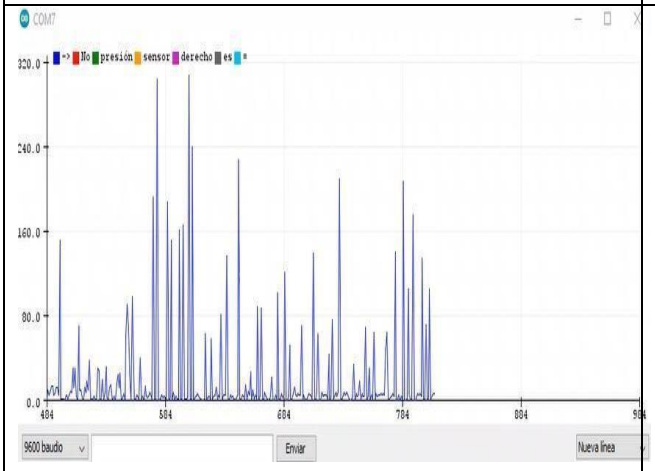

Definición de puntos de análisis
posición horiz. (pixel)
tiempo (frames)

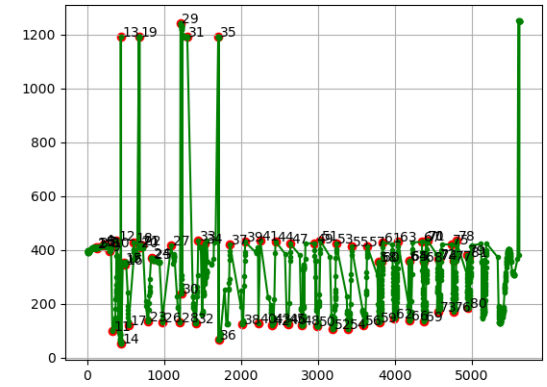

COM7
9600 baudo
Enviar
Nueva línea

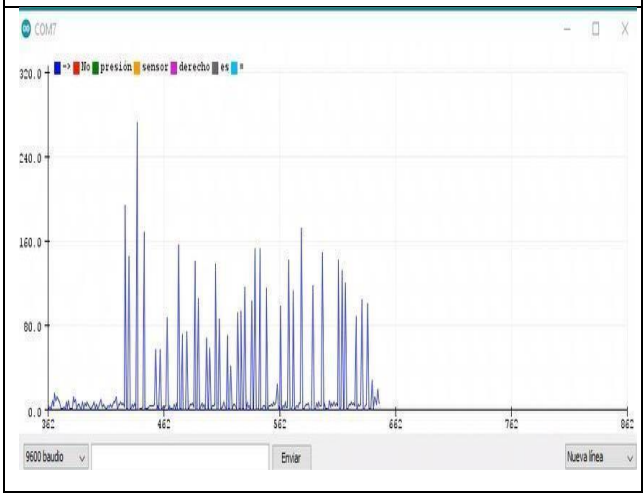


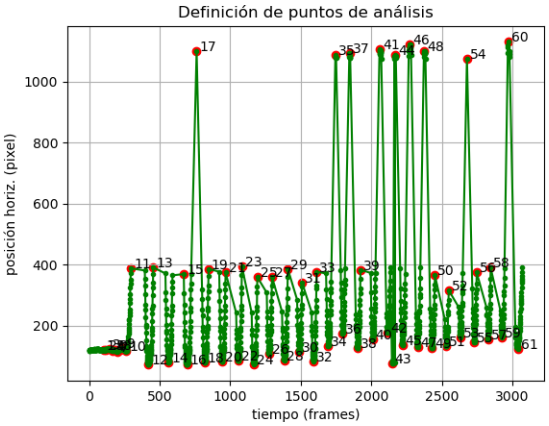

COM7
No presión sensor derecho es
9600 baudio
Enviar
Nueva línea

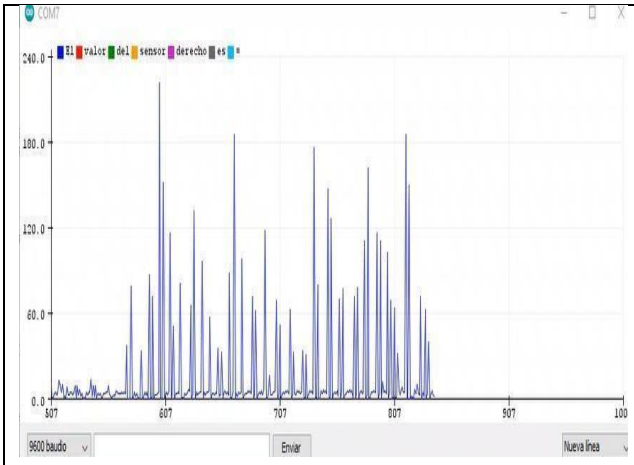

Definición de puntos de análisis
posición horiz. (pixel)
tiempo (frames)

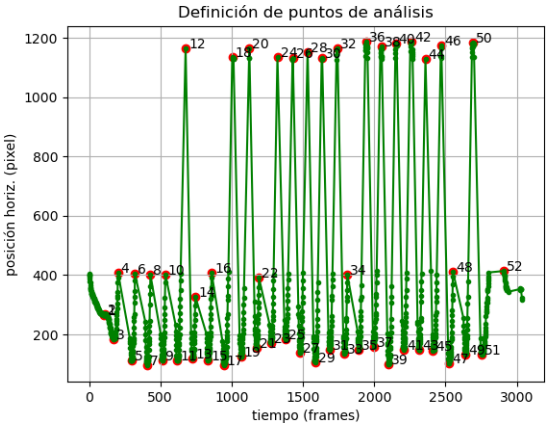

COM7
No presión sensor derecho es
9600 baudio
Enviar
Nueva línea

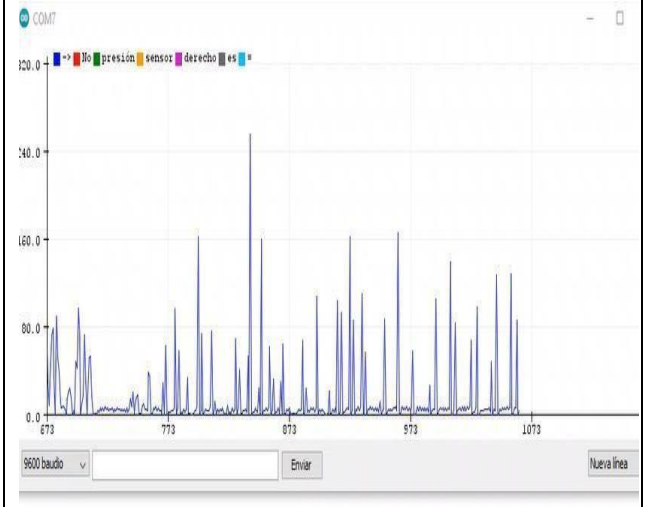


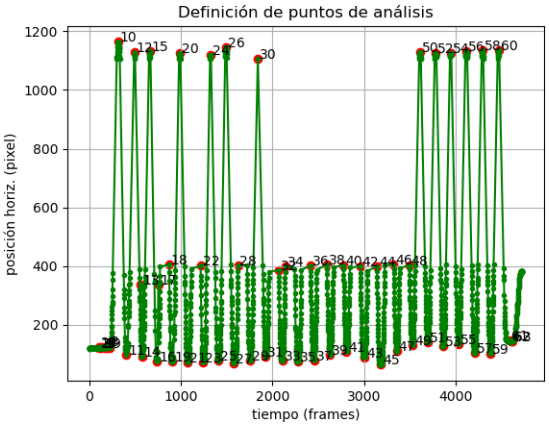

COM7
No presión sensor derecho es
9600 baudio
Enviar
Nueva línea

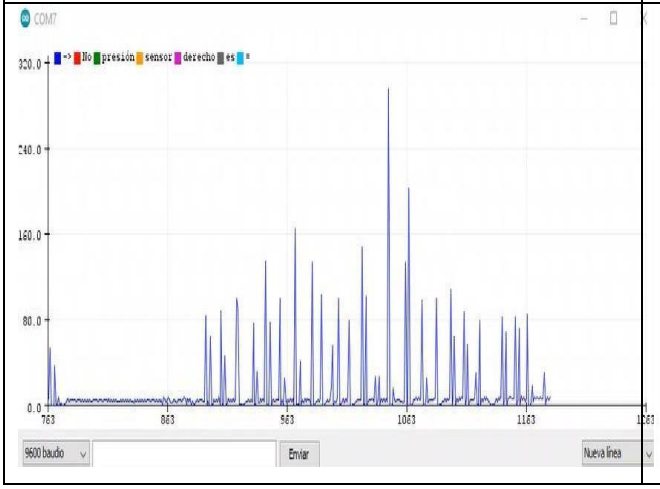

Definición de puntos de análisis
posición horiz. (pixel)
tiempo (frames)

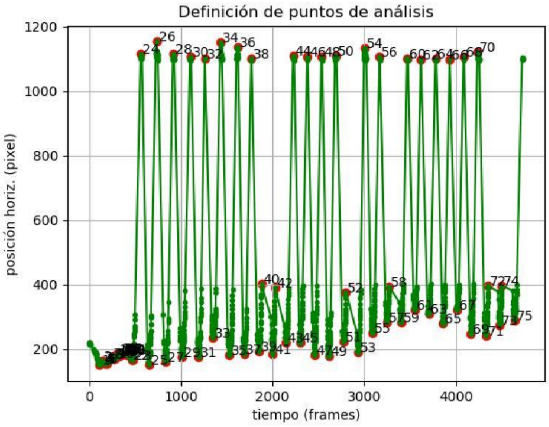

COM7
El valor del sensor derecho es
9600 baudio
Enviar
Nueva línea


Definición de puntos de análisis
posición horiz. (pixel)
tiempo (frames)

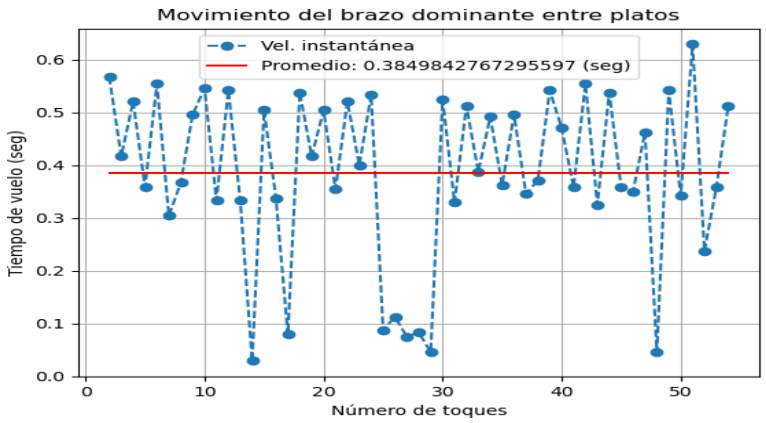
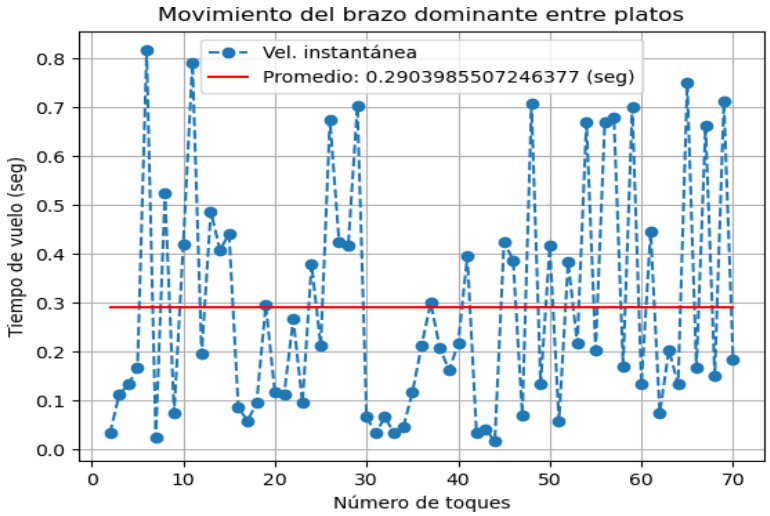

COM7
=> No presión sensor derecho es
9600 baudio
Enviar
Nueva línea

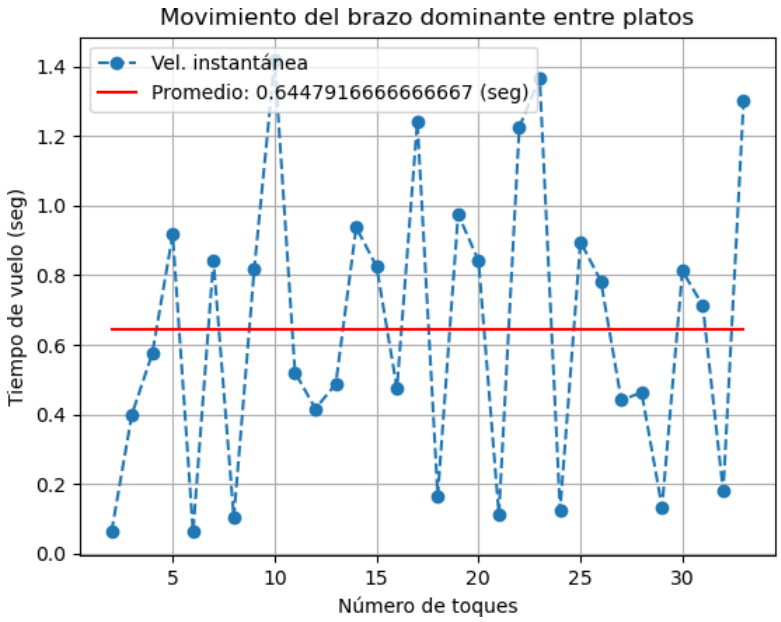
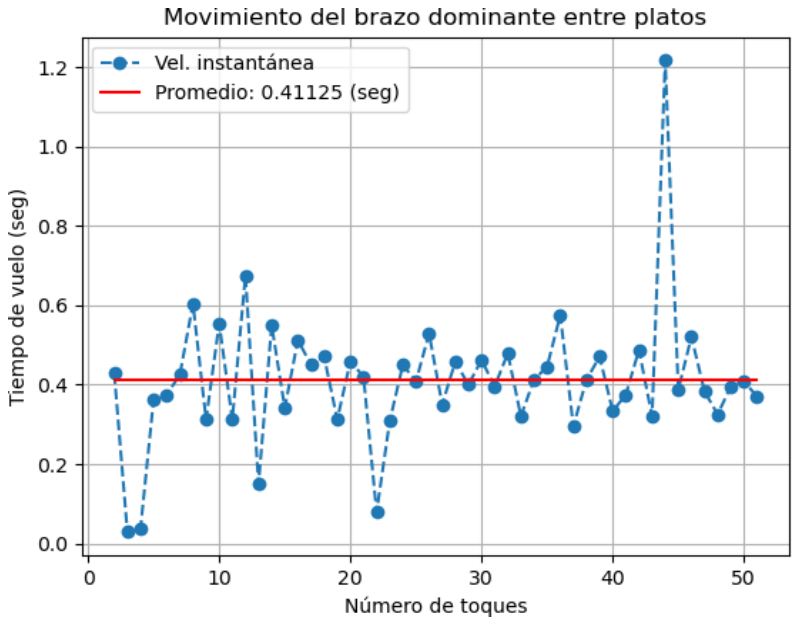

Definición de puntos de análisis
posición horiz. (pixel)
tiempo (frames)

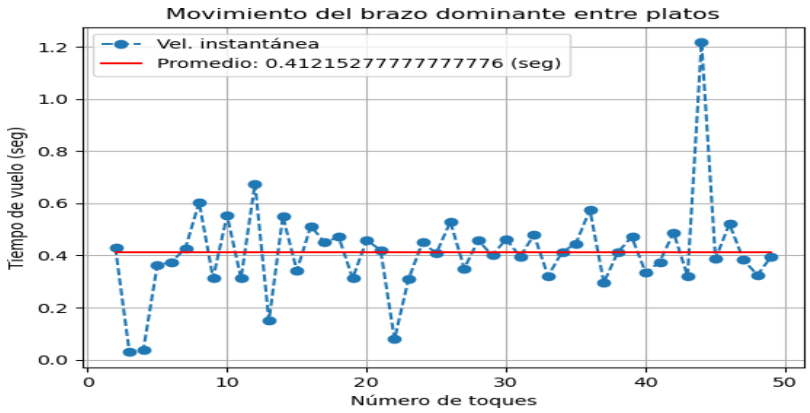
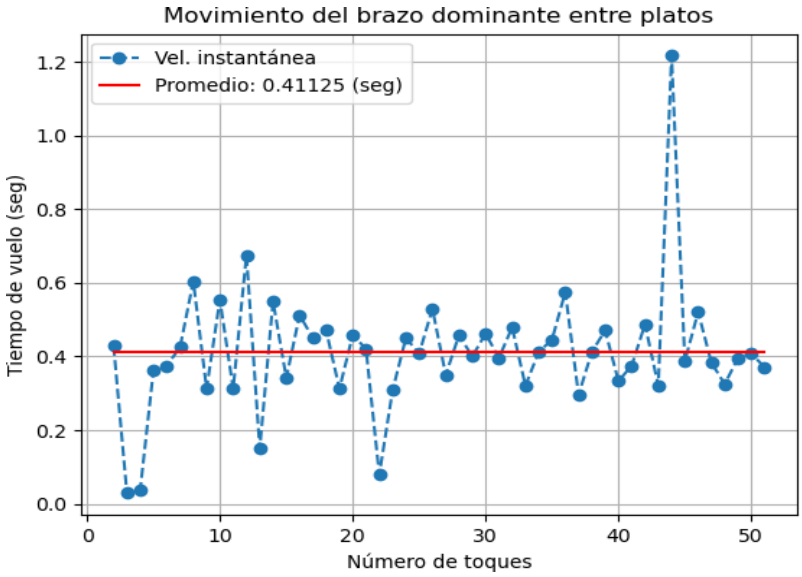

COM7
=> No presión sensor derecho es
9600 baudio
Enviar
Nueva línea

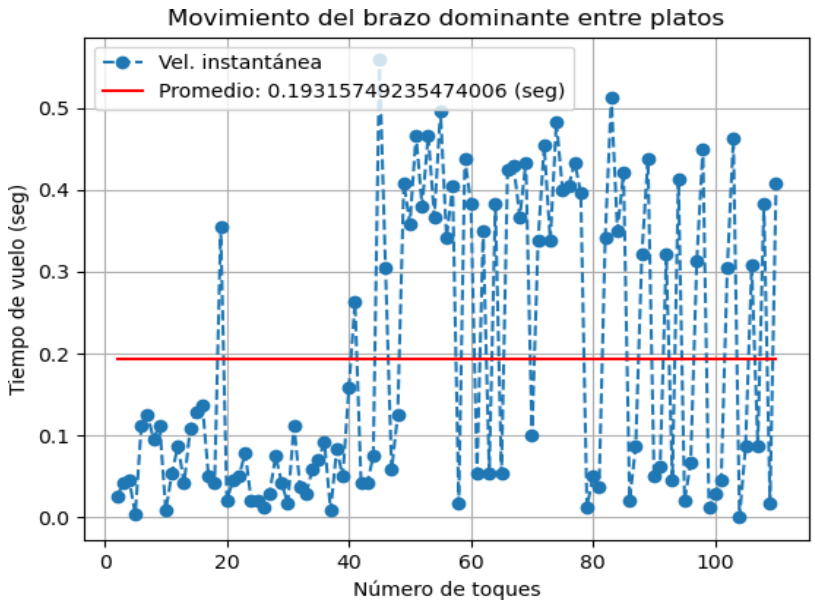
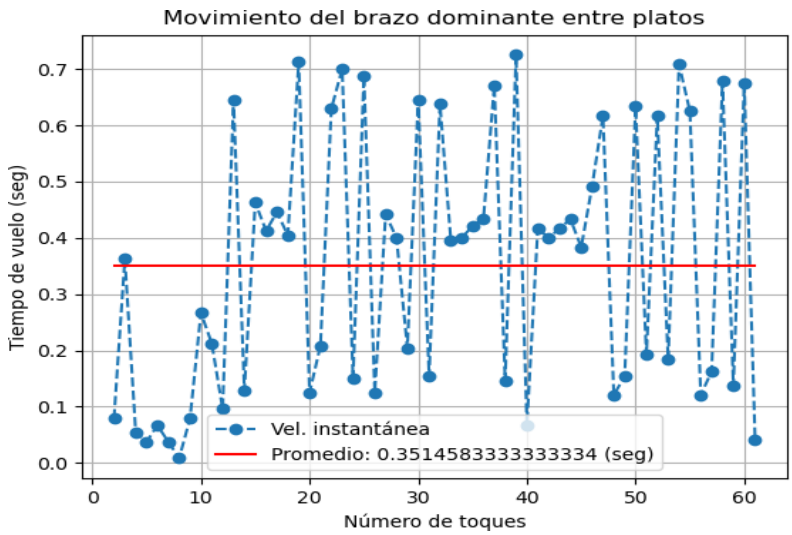

Definición de puntos de análisis
posición horiz. (pixel)
tiempo (frames)

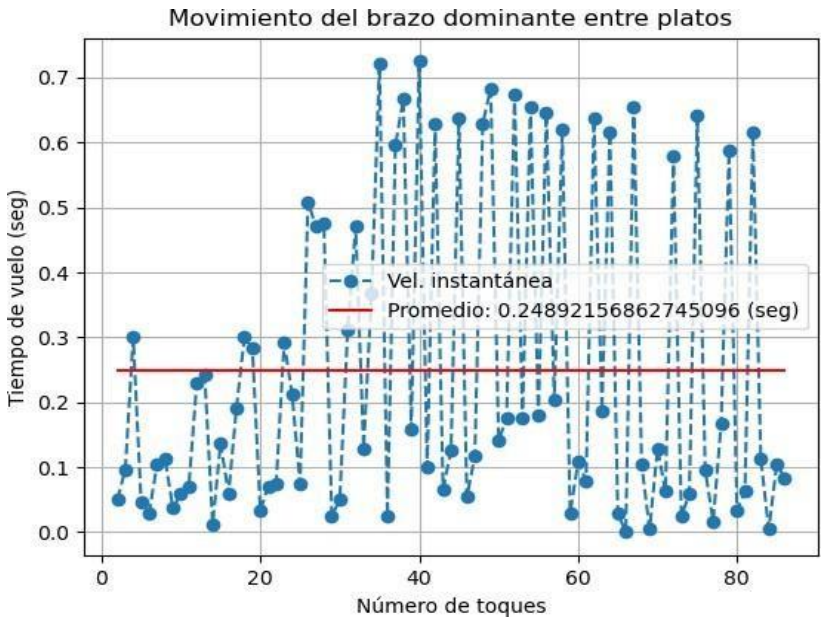
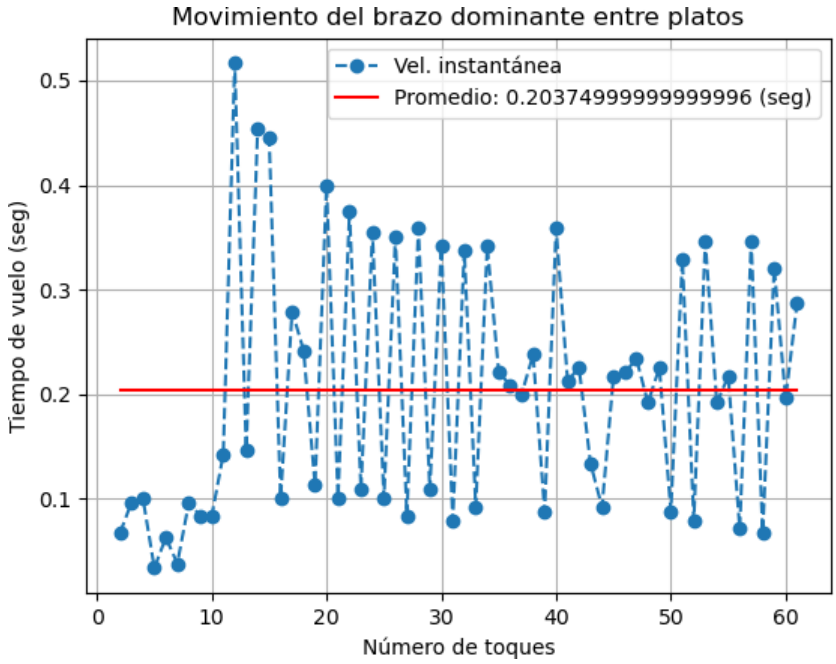
Anexo 4. Datos Velocidad obtenida mediante visión artificial..

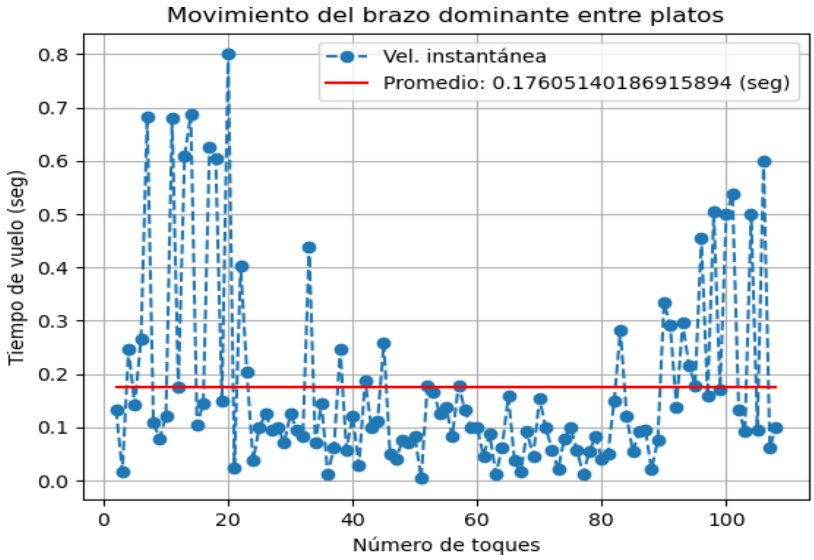
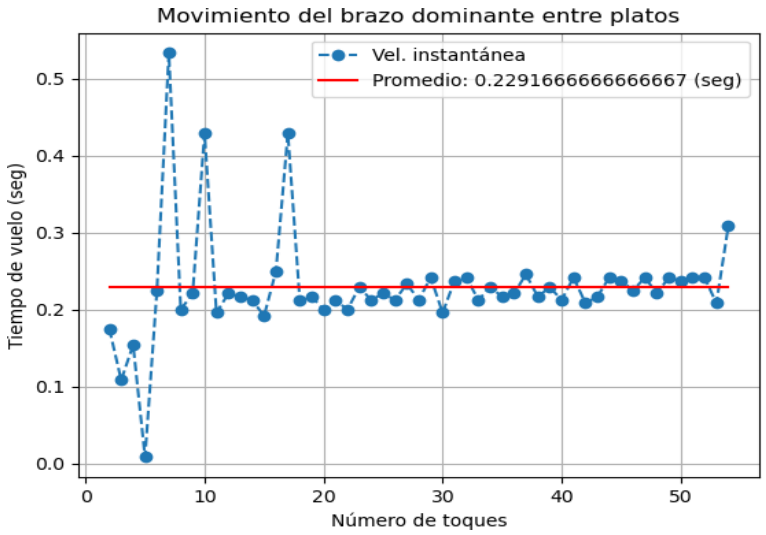
Prueba 2: 2221105007	
Prueba 3: 2221105016	

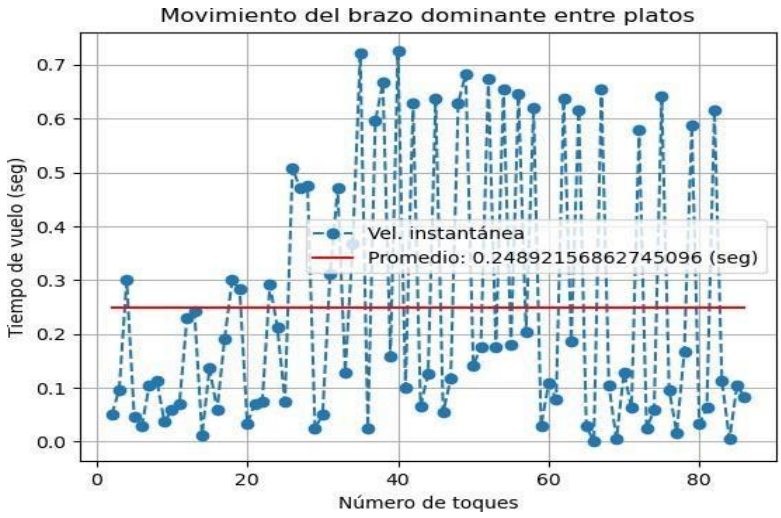
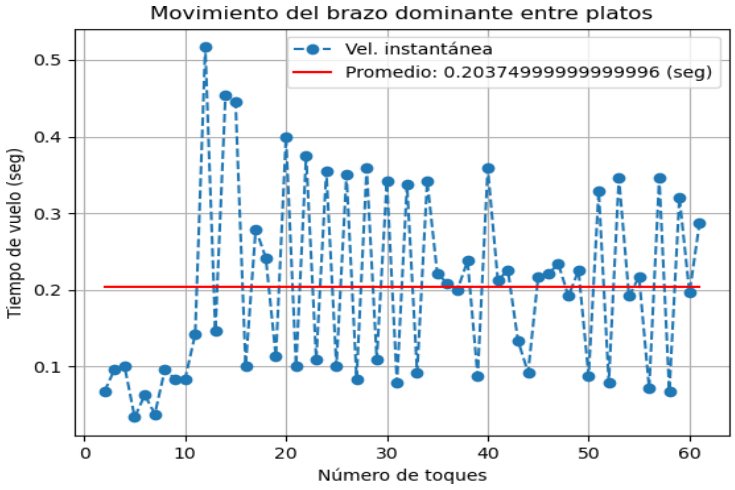
Prueba 4: 2221105003	
Prueba 5: 2221105010	

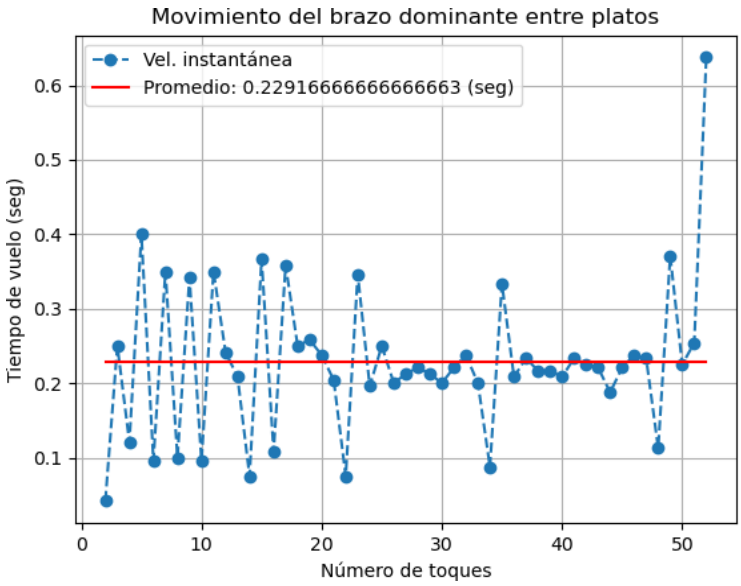
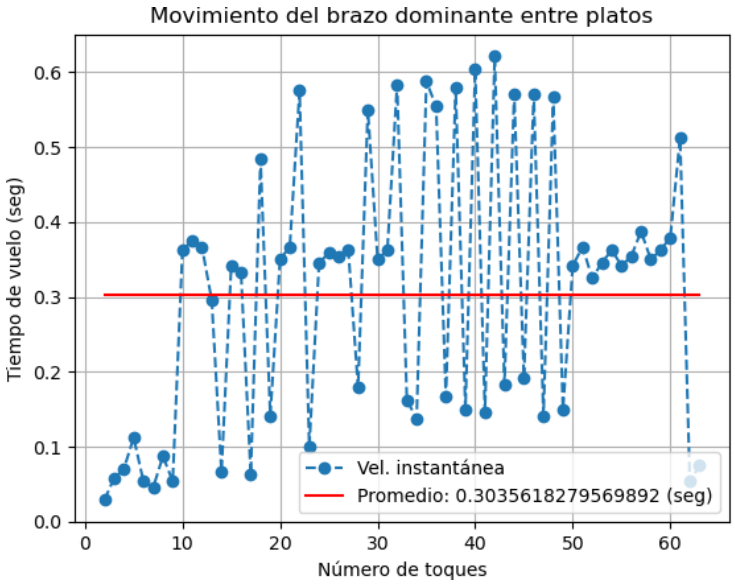
Prueba 6: 2221105034	
Prueba 7: 2221105021	

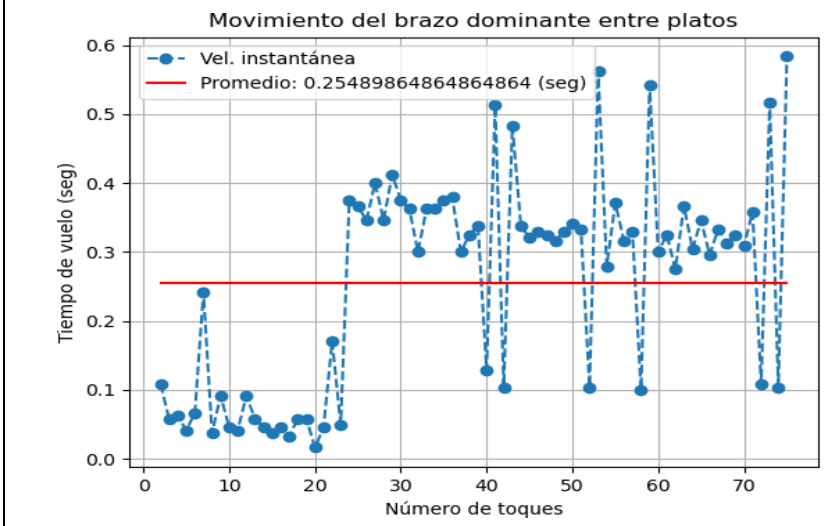
Prueba 8: 2221105027	
Prueba 9: 2221105013	

Prueba 10: 2221105035	
Prueba 11: 2221105015	

Prueba 12: 2221105024	
Prueba 13: 2221105012	

Prueba 14: 2221105011	
Prueba 15: 2221105014	

Prueba 16: 2221105025	
Prueba 17: 2221105032	
Prueba 18: 2221105018	



Anexo 5. Datos de la proyección de ka mano domintante frente al movimiento.

Prueba 2:

2221105007

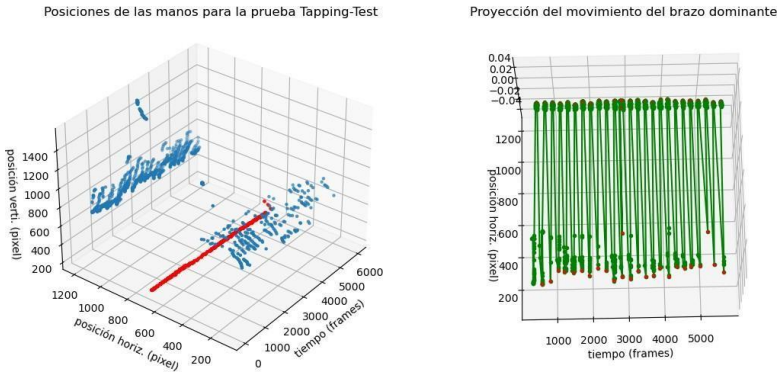


Figura 88. Posicionamientos de la mano dominante con la respectiva proyección del movimiento2.

Prueba 3:

2221105016

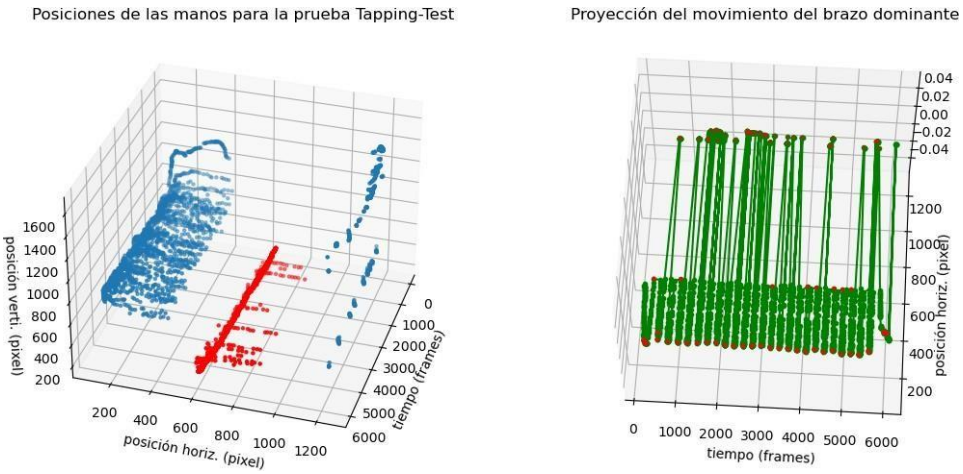


Figura 89. Posicionamientos de la mano dominante con la respectiva proyección del movimiento3.

Prueba 4:

2221105003

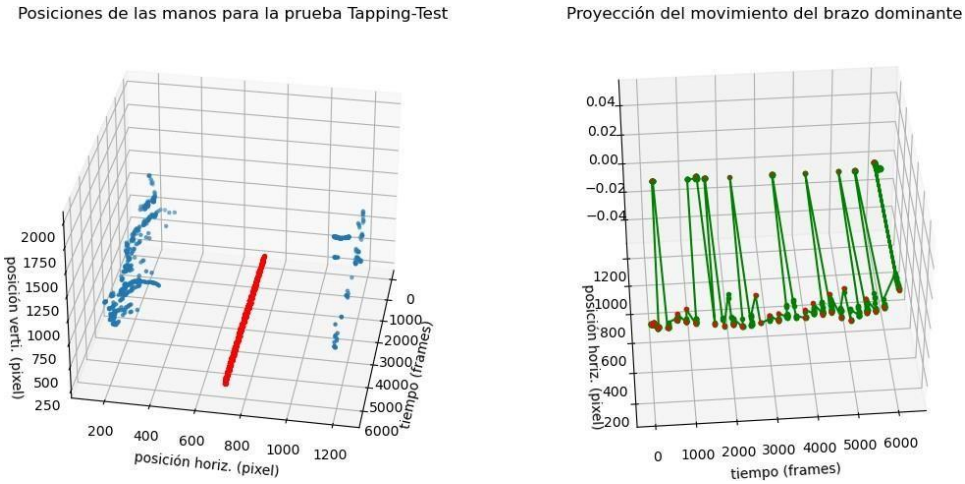


Figura 90. Posicionamientos de la mano dominante con la respectiva proyección del movimiento4.

Prueba 5:

2221105010

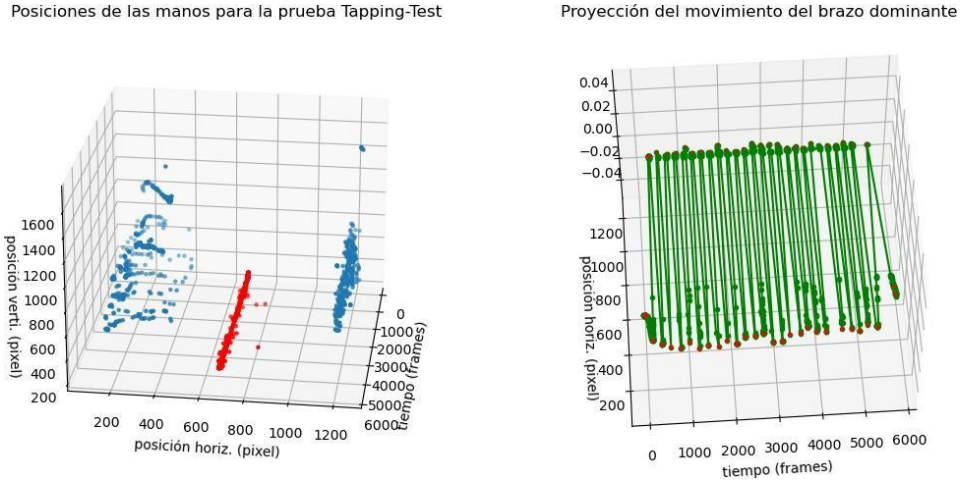


Figura 91. Posicionamientos de la mano dominante con la respectiva proyección del movimiento5

Prueba 6:

2221105034

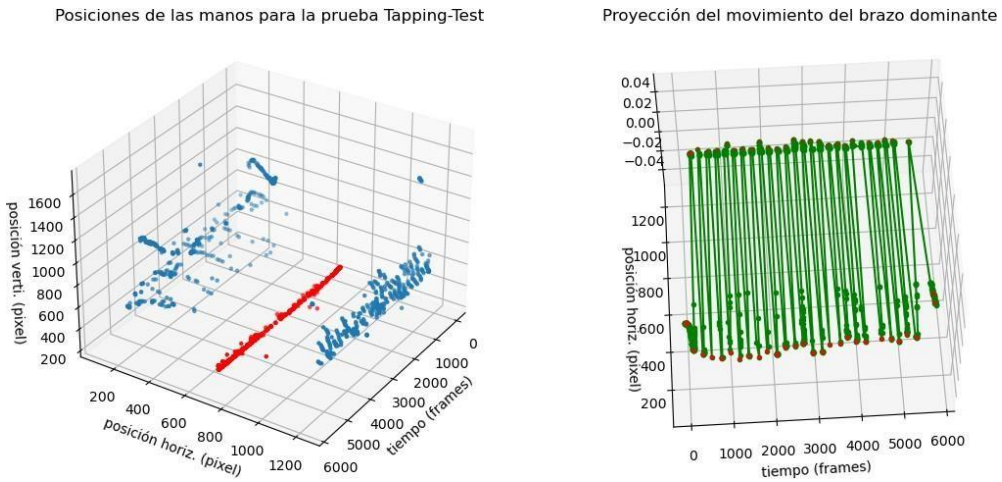


Figura 92. Posicionamientos de la mano dominante con la respectiva proyección del movimiento6.

Prueba 7:

2221105021

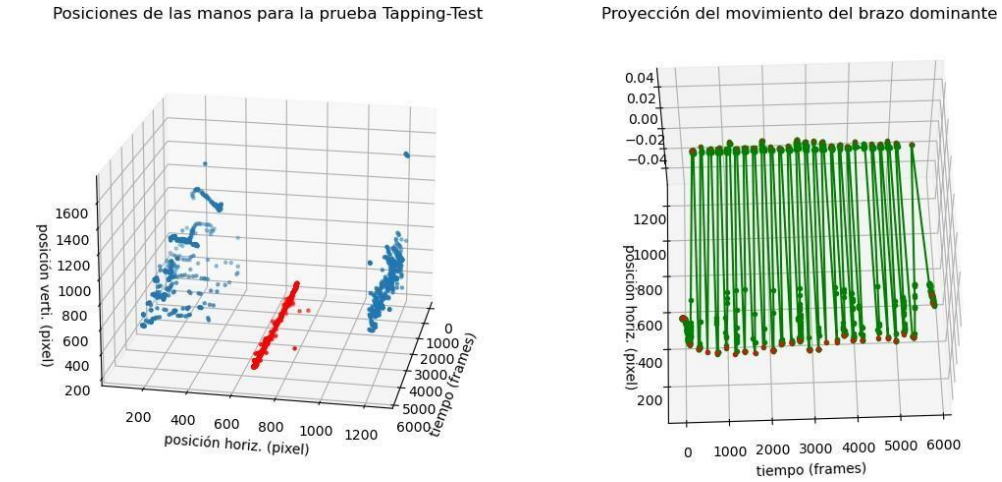


Figura 93. Posicionamientos de la mano dominante con la respectiva proyección del movimiento7.

Prueba 8:

2221105027

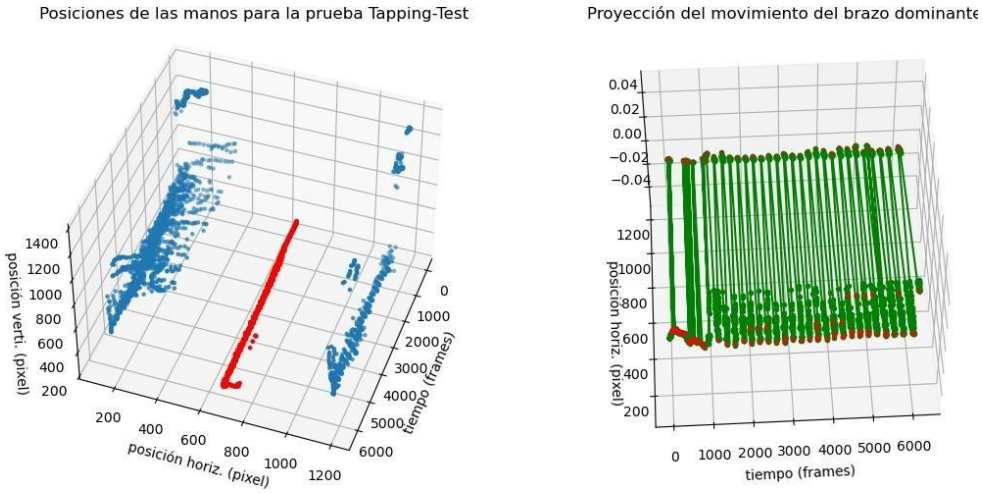


Figura 94. Posicionamientos de la mano dominante con la respectiva proyección del movimiento8.

Prueba 9:

2221105013

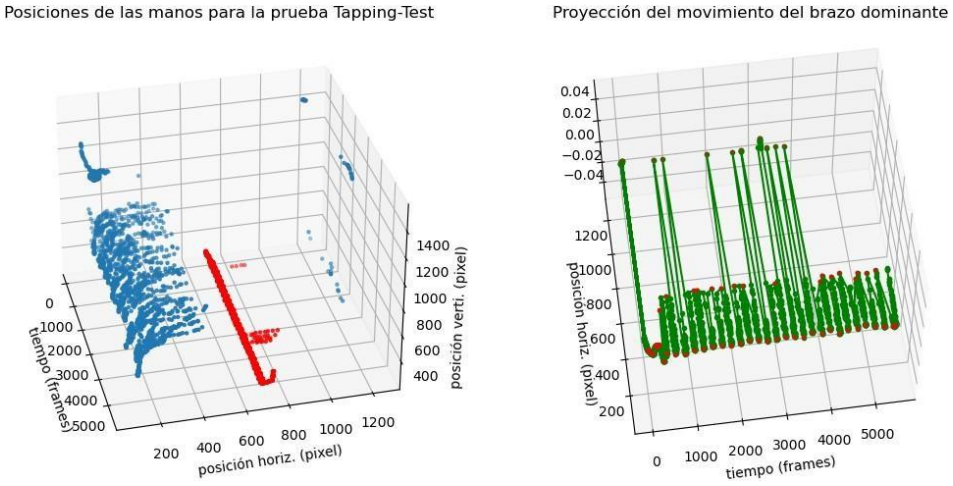


Figura 95. Posicionamientos de la mano dominante con la respectiva proyección del movimiento9.

Prueba 10:

2221105035

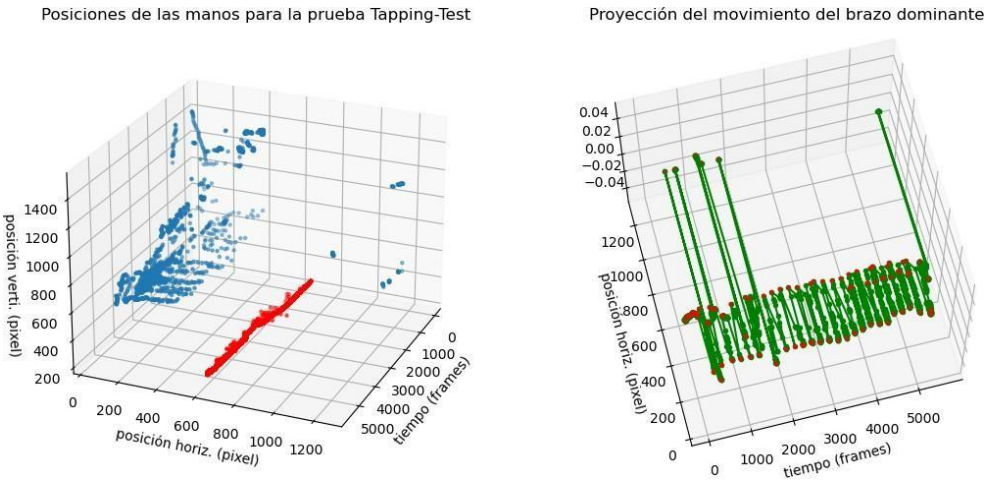


Figura 96. Posicionamientos de la mano dominante con la respectiva proyección del movimiento10.

Prueba 11:

2221105015

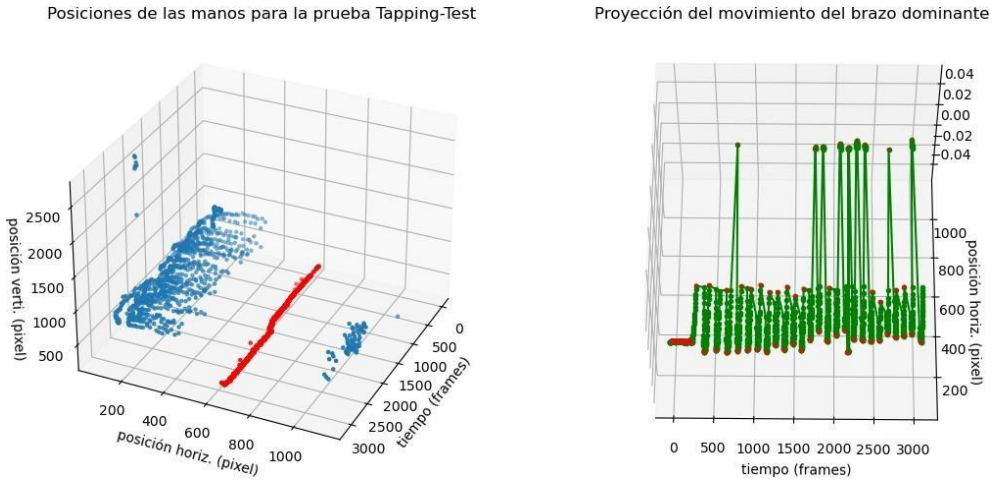


Figura 97. Posicionamientos de la mano dominante con la respectiva proyección del movimiento11.

Prueba 12:

2221105024

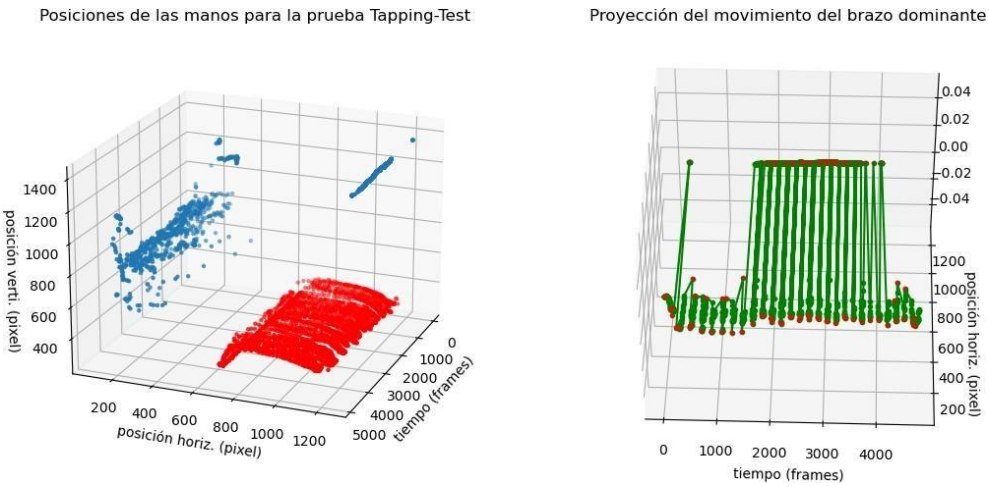


Figura 98 Posicionamientos de la mano dominante con la respectiva proyección del movimiento12.

Prueba 13:

2221105012

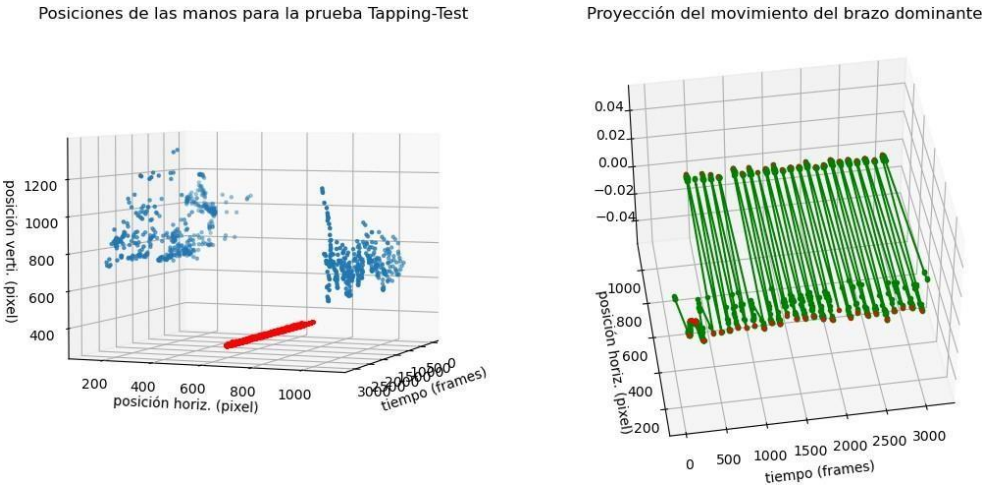


Figura 99. Posicionamientos de la mano dominante con la respectiva proyección del movimiento13.

Prueba 14:

2221105011

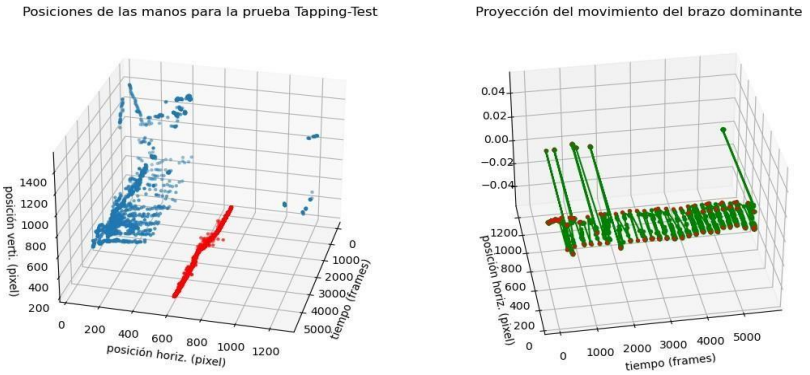


Figura 100. Posicionamientos de la mano dominante con la respectiva proyección delmovimiento14.

Prueba 15:

2221105014

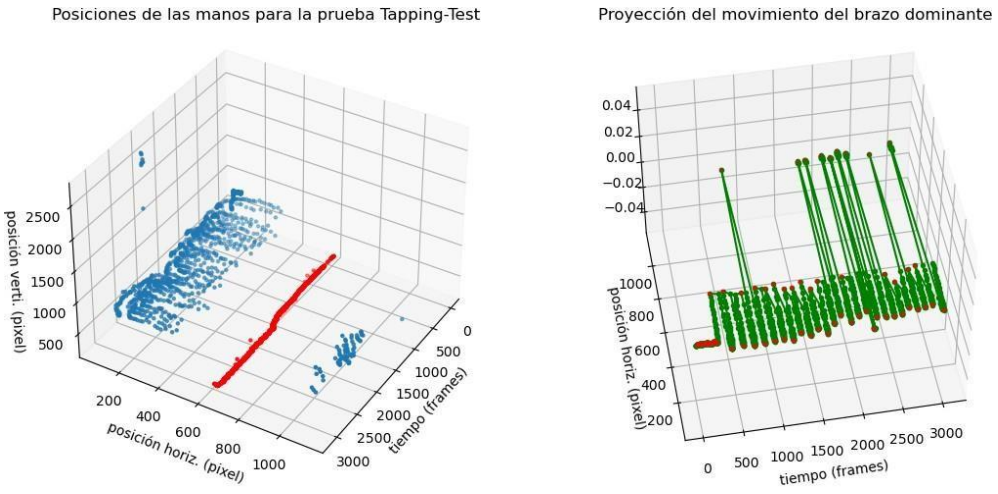


Figura 101. Posicionamientos de la mano dominante con la respectiva proyección delmovimiento15.

Prueba 16:

2221105025

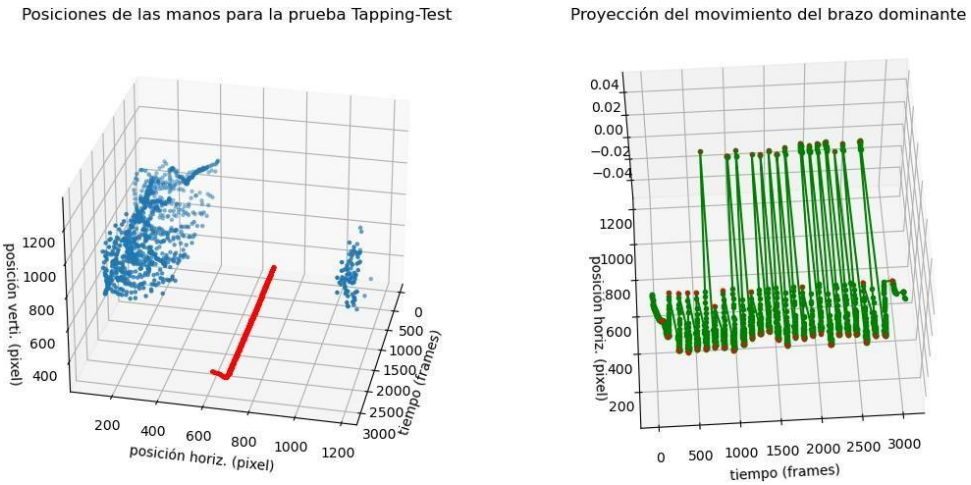


Figura 102. Posicionamientos de la mano dominante con la respectiva proyección delmovimiento16.

Prueba 17:

2221105032

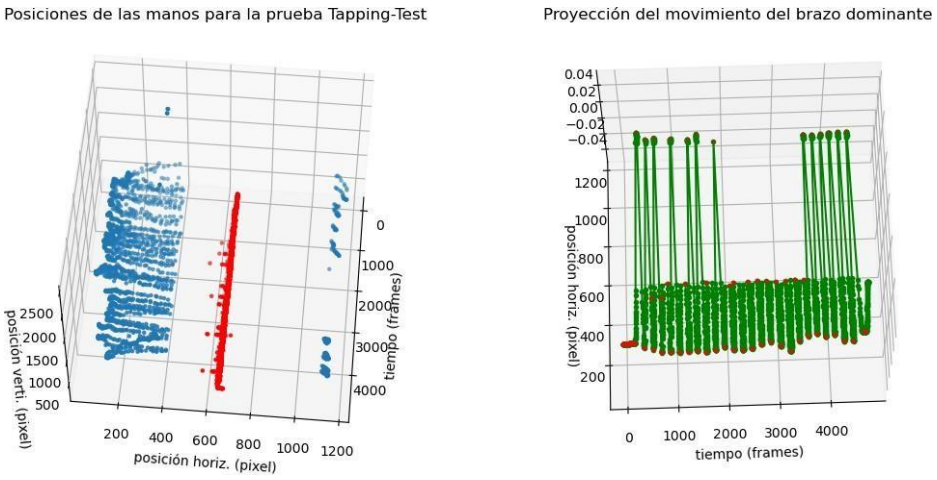


Figura 103. Posicionamientos de la mano dominante con la respectiva proyección delmovimiento17.

Prueba 18:

2221105018

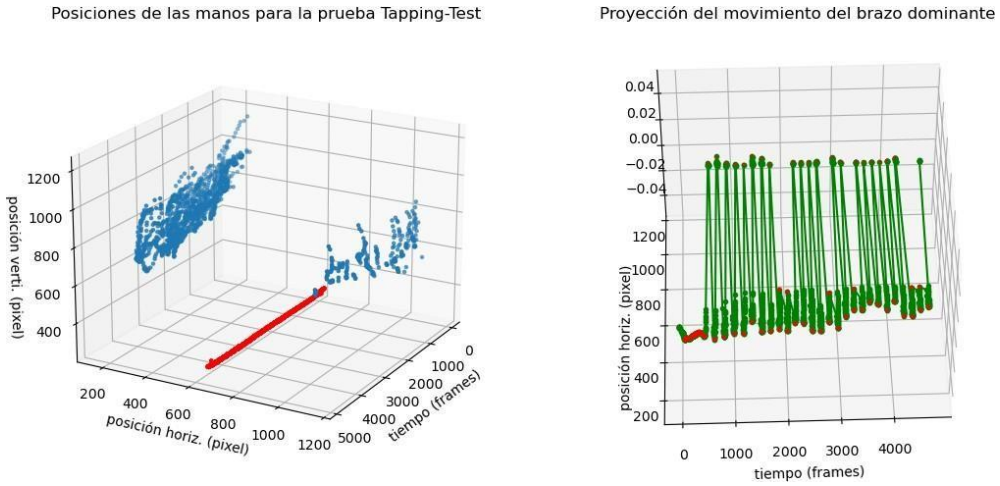


Figura 104. Posicionamientos de la mano dominante con la respectiva proyección delmovimiento18.

Anexo 6. Código interfaz graficas.

```
function varargout = Iniciar_Secion1(varargin)
% INICIAR_SECION1 MATLAB code for Iniciar_Secion1.fig
%      INICIAR_SECION1, by itself, creates a new INICIAR_SECION1 or raises the existing
%      singleton*.
%
%      H = INICIAR_SECION1 returns the handle to a new INICIAR_SECION1 or the handle to
%      the existing singleton*.
%
%      INICIAR_SECION1('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in INICIAR_SECION1.M with the given input arguments.
%
%      INICIAR_SECION1('Property','Value',...) creates a new INICIAR_SECION1 or raises the
%      existing singleton*. Starting from the left, property value pairs are
%      applied to the GUI before Iniciar_Secion1_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Iniciar_Secion1_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Iniciar_Secion1

% Last Modified by GUIDE v2.5 21-Aug-2022 16:00:43

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
```

```
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton', gui_Singleton, ...
                   'gui_OpeningFcn', @Iniciar_Secion1_OpeningFcn, ...
                   'gui_OutputFcn', @Iniciar_Secion1_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Iniciar_Secion1 is made visible.
function Iniciar_Secion1_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin  command line arguments to Iniciar_Secion1 (see VARARGIN)

% Choose default command line output for Iniciar_Secion1
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Iniciar_Secion1 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Iniciar_Secion1_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double
```

```
% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a double


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close(Iniciar_Secion1);
opciones3
```

```
function varargout = Iniciar_Secion2(varargin)
% INICIAR_SECION2 MATLAB code for Iniciar_Secion2.fig
%      INICIAR_SECION2, by itself, creates a new INICIAR_SECION2 or raises the existing
%      singleton*.
%
%      H = INICIAR_SECION2 returns the handle to a new INICIAR_SECION2 or the handle to
%      the existing singleton*.
%
%      INICIAR_SECION2('CALLBACK',hObject,eventData,handles,...)  calls  the  local
%      function named CALLBACK in INICIAR_SECION2.M with the given input arguments.
%
%      INICIAR_SECION2('Property','Value',...) creates a new INICIAR_SECION2 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Iniciar_Secion2_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Iniciar_Secion2_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Iniciar_Secion2

% Last Modified by GUIDE v2.5 21-Aug-2022 13:03:43

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Iniciar_Secion2_OpeningFcn, ...
                   'gui_OutputFcn', @Iniciar_Secion2_OutputFcn, ...
                   'gui_LayoutFcn', [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Iniciar_Secion2 is made visible.
function Iniciar_Secion2_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin  command line arguments to Iniciar_Secion2 (see VARARGIN)
```

```
% Choose default command line output for Iniciar_Secion2
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Iniciar_Secion2 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Iniciar_Secion2_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close(Iniciar_Secion2);
Iniciar_Secion1

function varargout = opciones3(varargin)
% OPCIONES3 MATLAB code for opciones3.fig
%      OPCIONES3, by itself, creates a new OPCIONES3 or raises the existing
%      singleton*.
%
%      H = OPCIONES3 returns the handle to a new OPCIONES3 or the handle to
%      the existing singleton*.
%
%      OPCIONES3('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in OPCIONES3.M with the given input arguments.
%
%      OPCIONES3('Property','Value',...) creates a new OPCIONES3 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before opciones3_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to opciones3_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help opciones3
```

```
% Last Modified by GUIDE v2.5 21-Aug-2022 18:21:07

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton', gui_Singleton, ...
                   'gui_OpeningFcn', @opciones3_OpeningFcn, ...
                   'gui_OutputFcn', @opciones3_OutputFcn, ...
                   'gui_LayoutFcn', [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before opciones3 is made visible.
function opciones3_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin  command line arguments to opciones3 (see VARARGIN)

% Choose default command line output for opciones3
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes opciones3 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = opciones3_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu1 contents as cell array
%        contents{get(hObject,'Value')} returns selected item from popupmenu1
value = get(handles.popupmenu1,'Value')
switch value
   case 2 %Agregar Usuario
      close(opciones3);
      GESTIONUSUARIOS4;
   case 3 %Buscar/Ver Usuario
      close(opciones3);
      GESTIONUSUARIOS5;
end

% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



% --- Executes on selection change in popupmenu2.
function popupmenu2_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu2 contents as cell array
%        contents{get(hObject,'Value')} returns selected item from popupmenu2
value = get(handles.popupmenu2,'Value')
switch value
   case 2 %Agregar Prueba
      close(opciones3);
      GESTION_DE_PRUEBA10;
   case 3 %Buscar/Ver Prueba
      close(opciones3);
      GESTION_DE_PRUEBA11;
end

% --- Executes during object creation, after setting all properties.
function popupmenu2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
```

```
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close(opciones3);
Iniciar_Secion1

function varargout = GESTIONUSUARIOS4(varargin)
% GESTIONUSUARIOS4 MATLAB code for GESTIONUSUARIOS4.fig
%      GESTIONUSUARIOS4, by itself, creates a new GESTIONUSUARIOS4 or raises the existing
%      singleton*.
%
%      H = GESTIONUSUARIOS4 returns the handle to a new GESTIONUSUARIOS4 or the handle to
%      the existing singleton*.
%
%      GESTIONUSUARIOS4('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in GESTIONUSUARIOS4.M with the given input arguments.
%
%      GESTIONUSUARIOS4('Property','Value',...) creates a new GESTIONUSUARIOS4 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before GESTIONUSUARIOS4_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to GESTIONUSUARIOS4_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help GESTIONUSUARIOS4

% Last Modified by GUIDE v2.5 21-Aug-2022 13:48:22

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn',  @GESTIONUSUARIOS4_OpeningFcn, ...
                   'gui_OutputFcn', @GESTIONUSUARIOS4_OutputFcn, ...
                   'gui_LayoutFcn', [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
```

```
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before GESTIONUSUARIOS4 is made visible.
function GESTIONUSUARIOS4_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin  command line arguments to GESTIONUSUARIOS4 (see VARARGIN)

% Choose default command line output for GESTIONUSUARIOS4
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes GESTIONUSUARIOS4 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = GESTIONUSUARIOS4_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close(GESTIONUSUARIOS4);
opciones3


% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close(GESTIONUSUARIOS4);
opciones3

% --- Executes on selection change in listbox1.
```

```
function listbox1_Callback(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns listbox1 contents as cell array
%        contents{get(hObject,'Value')} returns selected item from listbox1


% --- Executes during object creation, after setting all properties.
function listbox1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close(GESTIONUSUARIOS4);
AGREGAR_USUARIOS7

function varargout = GESTIONUSUARIOS5(varargin)
% GESTIONUSUARIOS5 MATLAB code for GESTIONUSUARIOS5.fig
%      GESTIONUSUARIOS5, by itself, creates a new GESTIONUSUARIOS5 or raises the existing
%      singleton*.
%
%      H = GESTIONUSUARIOS5 returns the handle to a new GESTIONUSUARIOS5 or the handle to
%      the existing singleton*.
%
%      GESTIONUSUARIOS5('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in GESTIONUSUARIOS5.M with the given input arguments.
%
%      GESTIONUSUARIOS5('Property','Value',...) creates a new GESTIONUSUARIOS5 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before GESTIONUSUARIOS5_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to GESTIONUSUARIOS5_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help GESTIONUSUARIOS5
```

```
% Last Modified by GUIDE v2.5 21-Aug-2022 14:06:14

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn',  @GESTIONUSUARIOS5_OpeningFcn, ...
                   'gui_OutputFcn', @GESTIONUSUARIOS5_OutputFcn, ...
                   'gui_LayoutFcn', [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before GESTIONUSUARIOS5 is made visible.
function GESTIONUSUARIOS5_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin  command line arguments to GESTIONUSUARIOS5 (see VARARGIN)

% Choose default command line output for GESTIONUSUARIOS5
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes GESTIONUSUARIOS5 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = GESTIONUSUARIOS5_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
```

```
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close(GESTIONUSUARIOS5);
AGREGAR_USUARIOS7

% --- Executes on selection change in listbox1.
function listbox1_Callback(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns listbox1 contents as cell array
%        contents{get(hObject,'Value')} returns selected item from listbox1


% --- Executes during object creation, after setting all properties.
function listbox1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close(GESTIONUSUARIOS5);
GESTIONUSUARIOS4


% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close(GESTIONUSUARIOS5);
opciones3

% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close(GESTIONUSUARIOS5);
```

DATOS_PRUEBA6

```
% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function varargout = DATOS_PRUEBA6(varargin)
% DATOS_PRUEBA6 MATLAB code for DATOS_PRUEBA6.fig
%      DATOS_PRUEBA6, by itself, creates a new DATOS_PRUEBA6 or raises the existing
%      singleton*.
%
%      H = DATOS_PRUEBA6 returns the handle to a new DATOS_PRUEBA6 or the handle to
%      the existing singleton*.
%
%      DATOS_PRUEBA6('CALLBACK',hObject,eventData,handles,...)  calls  the  local
%      function named CALLBACK in DATOS_PRUEBA6.M with the given input arguments.
%
%      DATOS_PRUEBA6('Property','Value',...) creates a new DATOS_PRUEBA6 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before DATOS_PRUEBA6_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to DATOS_PRUEBA6_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help DATOS_PRUEBA6

% Last Modified by GUIDE v2.5 21-Aug-2022 14:22:26

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @DATOS_PRUEBA6_OpeningFcn, ...
                   'gui_OutputFcn', @DATOS_PRUEBA6_OutputFcn, ...
                   'gui_LayoutFcn', [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
```

```
% --- Executes just before DATOS_PRUEBA6 is made visible.
function DATOS_PRUEBA6_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin  command line arguments to DATOS_PRUEBA6 (see VARARGIN)

% Choose default command line output for DATOS_PRUEBA6
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes DATOS_PRUEBA6 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = DATOS_PRUEBA6_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

[filename path]=uigetfile(('*.mov,*mp4'),'Abrir video');
if isequal(filename,0)
   return

else
   folder='\Users\Ana Julia\Desktop\Interfaz Grafica\vids_test';
   video=imread(strcat(path,filename));
   imwrite(video,fullfile(folder,'vidtest_tt.mp4'));


end

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close(DATOS_PRUEBA6);
GESTIONUSUARIOS5

% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close(DATOS_PRUEBA6);
opciones3


function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a double


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

function varargout = AGREGAR_USUARIOS7(varargin)
% AGREGAR_USUARIOS7 MATLAB code for AGREGAR_USUARIOS7.fig
%      AGREGAR_USUARIOS7, by itself, creates a new AGREGAR_USUARIOS7 or raises the existing
%      singleton*.
%
%      H = AGREGAR_USUARIOS7 returns the handle to a new AGREGAR_USUARIOS7 or the handle to
%      the existing singleton*.
%
%      AGREGAR_USUARIOS7('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in AGREGAR_USUARIOS7.M with the given input arguments.
%
%      AGREGAR_USUARIOS7('Property','Value',...) creates a new AGREGAR_USUARIOS7 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before AGREGAR_USUARIOS7_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to AGREGAR_USUARIOS7_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help AGREGAR_USUARIOS7

% Last Modified by GUIDE v2.5 21-Aug-2022 14:32:18

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
```

```
                'gui_Singleton',  gui_Singleton, ...
                'gui_OpeningFcn', @AGREGAR_USUARIOS7_OpeningFcn, ...
                'gui_OutputFcn', @AGREGAR_USUARIOS7_OutputFcn, ...
                'gui_LayoutFcn', [] , ...
                'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before AGREGAR_USUARIOS7 is made visible.
function AGREGAR_USUARIOS7_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin  command line arguments to AGREGAR_USUARIOS7 (see VARARGIN)

% Choose default command line output for AGREGAR_USUARIOS7
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes AGREGAR_USUARIOS7 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = AGREGAR_USUARIOS7_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close(AGREGAR_USUARIOS7);
opciones3
```

```
% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close(AGREGAR_USUARIOS7);
opciones3

% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close(AGREGAR_USUARIOS7);
AGREGAR_USUARIOS8



function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a double
```

```
% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a double


% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

function varargout = AGREGAR_USUARIOS8(varargin)
% AGREGAR_USUARIOS8 MATLAB code for AGREGAR_USUARIOS8.fig
%      AGREGAR_USUARIOS8, by itself, creates a new AGREGAR_USUARIOS8 or raises the existing
%      singleton*.
%
%      H = AGREGAR_USUARIOS8 returns the handle to a new AGREGAR_USUARIOS8 or the handle to
%      the existing singleton*.
%
%      AGREGAR_USUARIOS8('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in AGREGAR_USUARIOS8.M with the given input arguments.
%
%      AGREGAR_USUARIOS8('Property','Value',...) creates a new AGREGAR_USUARIOS8 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before AGREGAR_USUARIOS8_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to AGREGAR_USUARIOS8_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
```

```
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help AGREGAR_USUARIOS8

% Last Modified by GUIDE v2.5 21-Aug-2022 16:18:15

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @AGREGAR_USUARIOS8_OpeningFcn, ...
                   'gui_OutputFcn', @AGREGAR_USUARIOS8_OutputFcn, ...
                   'gui_LayoutFcn', [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before AGREGAR_USUARIOS8 is made visible.
function AGREGAR_USUARIOS8_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin  command line arguments to AGREGAR_USUARIOS8 (see VARARGIN)

% Choose default command line output for AGREGAR_USUARIOS8
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes AGREGAR_USUARIOS8 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = AGREGAR_USUARIOS8_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[filename path]=uigetfile(('*.jpg'),'Abrir Imagen');
if isequal(filename,0)
   return

else
   folder='\Users\Ana Julia\Desktop\Interfaz Grafica\Profiles_Pics';
   foto=imread(strcat(path,filename));
   imwrite(foto,fullfile(folder,'imagen_pp.jpg'));
   axes(handles.axes1);
   imshow(foto)
end

% --- Executes on button press in checkbox1.
function checkbox1_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox1


% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close(AGREGAR_USUARIOS8);
AGREGAR_USUARIOS7

% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close(AGREGAR_USUARIOS8);
opciones3


% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close(AGREGAR_USUARIOS8);
```

AGREGAR_USUARIOS9

```
% --- Executes when user attempts to close figure1.
function figure1_CloseRequestFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: delete(hObject) closes the figure
delete(hObject);
function varargout = AGREGAR_USUARIOS9(varargin)
% AGREGAR_USUARIOS9 MATLAB code for AGREGAR_USUARIOS9.fig
%      AGREGAR_USUARIOS9, by itself, creates a new AGREGAR_USUARIOS9 or raises the existing
%      singleton*.
%
%      H = AGREGAR_USUARIOS9 returns the handle to a new AGREGAR_USUARIOS9 or the handle to
%      the existing singleton*.
%
%      AGREGAR_USUARIOS9('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in AGREGAR_USUARIOS9.M with the given input arguments.
%
%      AGREGAR_USUARIOS9('Property','Value',...) creates a new AGREGAR_USUARIOS9 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before AGREGAR_USUARIOS9_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to AGREGAR_USUARIOS9_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help AGREGAR_USUARIOS9

% Last Modified by GUIDE v2.5 21-Aug-2022 15:38:53

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @AGREGAR_USUARIOS9_OpeningFcn, ...
                   'gui_OutputFcn', @AGREGAR_USUARIOS9_OutputFcn, ...
                   'gui_LayoutFcn', [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
```

```
% End initialization code - DO NOT EDIT


% --- Executes just before AGREGAR_USUARIOS9 is made visible.
function AGREGAR_USUARIOS9_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin  command line arguments to AGREGAR_USUARIOS9 (see VARARGIN)

% Choose default command line output for AGREGAR_USUARIOS9
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes AGREGAR_USUARIOS9 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = AGREGAR_USUARIOS9_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
```

```
end



function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a double


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



% --- Executes on button press in checkbox1.
function checkbox1_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox1



% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close(AGREGAR_USUARIOS9);
AGREGAR_USUARIOS8

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close(AGREGAR_USUARIOS9);
opciones3

% --- Executes on button press in pushbutton3.
```

```
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
function varargout = GESTION_DE_PRUEBA10(varargin)
% GESTION_DE_PRUEBA10 MATLAB code for GESTION_DE_PRUEBA10.fig
%      GESTION_DE_PRUEBA10, by itself, creates a new GESTION_DE_PRUEBA10 or raises the existing
%      singleton*.
%
%      H = GESTION_DE_PRUEBA10 returns the handle to a new GESTION_DE_PRUEBA10 or the handle to
%      the existing singleton*.
%
%      GESTION_DE_PRUEBA10('CALLBACK',hObject,eventData,handles,...)   calls   the   local
%      function named CALLBACK in GESTION_DE_PRUEBA10.M with the given input arguments.
%
%      GESTION_DE_PRUEBA10('Property','Value',...) creates a new GESTION_DE_PRUEBA10 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before GESTION_DE_PRUEBA10_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to GESTION_DE_PRUEBA10_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help GESTION_DE_PRUEBA10

% Last Modified by GUIDE v2.5 21-Aug-2022 15:37:45

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn',  @GESTION_DE_PRUEBA10_OpeningFcn, ...
                   'gui_OutputFcn', @GESTION_DE_PRUEBA10_OutputFcn, ...
                   'gui_LayoutFcn', [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before GESTION_DE_PRUEBA10 is made visible.
function GESTION_DE_PRUEBA10_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin  command line arguments to GESTION_DE_PRUEBA10 (see VARARGIN)

% Choose default command line output for GESTION_DE_PRUEBA10
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes GESTION_DE_PRUEBA10 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = GESTION_DE_PRUEBA10_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a double


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in listbox1.
function listbox1_Callback(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns listbox1 contents as cell array
%        contents{get(hObject,'Value')} returns selected item from listbox1


% --- Executes during object creation, after setting all properties.
function listbox1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in listbox2.
function listbox2_Callback(hObject, eventdata, handles)
% hObject    handle to listbox2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns listbox2 contents as cell array
%        contents{get(hObject,'Value')} returns selected item from listbox2
```

```
% --- Executes during object creation, after setting all properties.
function listbox2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close(GESTION_DE_PRUEBA10);
opciones3

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function varargout = GESTION_DE_PRUEBA11(varargin)
% GESTION_DE_PRUEBA11 MATLAB code for GESTION_DE_PRUEBA11.fig
%      GESTION_DE_PRUEBA11, by itself, creates a new GESTION_DE_PRUEBA11 or raises the existing
%      singleton*.
%
%      H = GESTION_DE_PRUEBA11 returns the handle to a new GESTION_DE_PRUEBA11 or the handle to
%      the existing singleton*.
%
%      GESTION_DE_PRUEBA11('CALLBACK',hObject,eventData,handles,...)   calls   the   local
%      function named CALLBACK in GESTION_DE_PRUEBA11.M with the given input arguments.
%
%      GESTION_DE_PRUEBA11('Property','Value',...) creates a new GESTION_DE_PRUEBA11 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before GESTION_DE_PRUEBA11_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to GESTION_DE_PRUEBA11_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help GESTION_DE_PRUEBA11

% Last Modified by GUIDE v2.5 21-Aug-2022 15:55:30

% Begin initialization code - DO NOT EDIT
```

```
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn',  @GESTION_DE_PRUEBA11_OpeningFcn, ...
                   'gui_OutputFcn', @GESTION_DE_PRUEBA11_OutputFcn, ...
                   'gui_LayoutFcn', [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before GESTION_DE_PRUEBA11 is made visible.
function GESTION_DE_PRUEBA11_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin  command line arguments to GESTION_DE_PRUEBA11 (see VARARGIN)

% Choose default command line output for GESTION_DE_PRUEBA11
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes GESTION_DE_PRUEBA11 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = GESTION_DE_PRUEBA11_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on selection change in listbox1.
function listbox1_Callback(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns listbox1 contents as cell array
%        contents{get(hObject,'Value')} returns selected item from listbox1


% --- Executes during object creation, after setting all properties.
function listbox1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close(GESTION_DE_PRUEBA11);
GESTION_DE_PRUEBA12

% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close(GESTION_DE_PRUEBA11);
```

opciones3

```
function varargout = GESTION_DE_PRUEBA12(varargin)
% GESTION_DE_PRUEBA12 MATLAB code for GESTION_DE_PRUEBA12.fig
%      GESTION_DE_PRUEBA12, by itself, creates a new GESTION_DE_PRUEBA12 or raises the existing
%      singleton*.
%
%      H = GESTION_DE_PRUEBA12 returns the handle to a new GESTION_DE_PRUEBA12 or the handle to
%      the existing singleton*.
%
%      GESTION_DE_PRUEBA12('CALLBACK',hObject,eventData,handles,...)  calls  the  local
%      function named CALLBACK in GESTION_DE_PRUEBA12.M with the given input arguments.
%
%      GESTION_DE_PRUEBA12('Property','Value',...) creates a new GESTION_DE_PRUEBA12 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before GESTION_DE_PRUEBA12_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to GESTION_DE_PRUEBA12_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help GESTION_DE_PRUEBA12

% Last Modified by GUIDE v2.5 21-Aug-2022 15:53:44

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn',  @GESTION_DE_PRUEBA12_OpeningFcn, ...
                   'gui_OutputFcn', @GESTION_DE_PRUEBA12_OutputFcn, ...
                   'gui_LayoutFcn', [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT



% --- Executes just before GESTION_DE_PRUEBA12 is made visible.
function GESTION_DE_PRUEBA12_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin  command line arguments to GESTION_DE_PRUEBA12 (see VARARGIN)
```

```
% Choose default command line output for GESTION_DE_PRUEBA12
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes GESTION_DE_PRUEBA12 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = GESTION_DE_PRUEBA12_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on selection change in listbox1.
function listbox1_Callback(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns listbox1 contents as cell array
%        contents{get(hObject,'Value')} returns selected item from listbox1


% --- Executes during object creation, after setting all properties.
function listbox1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close(GESTION_DE_PRUEBA12);
GESTION_DE_PRUEBA11
```

```
% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close(GESTION_DE_PRUEBA12);
opciones3
```

Anexo7. Código Procesamiento de imagen basado en visión artificial.

```
import matplotlib.pyplot as plt
import numpy as np

# save generated data for post-processing
vclip_name = "ttest3.mp4"
data_file = "found_" + vclip_name + ".data.npy"
foundarr = np.load(data_file)
foundarr = foundarr.astype(np.float64)
print("Points found: " + str(len(foundarr)))

# normalize
avy = np.average(foundarr[:,1])
print("avy=" + str(avy) + " - type: " + str(np.dtype(avy)))
avz = np.average(foundarr[:,2])
print("avz=" + str(avz))
dvy = np.std(foundarr[:,1])
print("dvy=" + str(dvy) + " - type: " + str(np.dtype(dvy)))
dvz = np.std(foundarr[:,2])
print("dvz=" + str(dvz))
print("type: " + str(foundarr[:,1].dtype))

foundarr[:,1] = (foundarr[:,1] - avy) / dvy
foundarr[:,2] = (foundarr[:,2]-avz) / dvz

# display
npoints = len(foundarr)
ndarm = []
darm = []

for i in range(npoints):
    x = foundarr[i, 0]
    y = foundarr[i, 1]
    z = foundarr[i, 2]
```

```
    if z < 0:
        ndarm.append((x, y, z))
    else:
        darm.append((x, y, z))

ndarm = np.array(ndarm)
darm = np.array(darm)

# print found hands along time
fig, (ax1, ax2) = plt.subplots(1, 2)
ax1.axis('off')
ax2.axis('off')
ax1 = fig.add_subplot(1, 2, 1, projection='3d')
ax1.scatter(ndarm[:, 0], ndarm[:, 1], ndarm[:, 2], marker='.', c='r', label='Brazo no dominante')
ax1.scatter(darm[:, 0], darm[:, 1], darm[:, 2], marker='.', label='Brazo dominante')
ax1.set_xlabel('tiempo (frames)')
ax1.set_ylabel('posición horiz. (pixel)')
ax1.set_zlabel('posición verti. (pixel)')
ax1.set_title('Posiciones de las manos para la prueba Tapping-Test')

# print hand projection to the horizontal plane
ax2 = fig.add_subplot(1, 2, 2, projection='3d')
#ax2.scatter(darm[:, 0], darm[:, 1], 0*darm[:, 2], marker='.', c='g')
ax2.plot(darm[:, 0], darm[:, 1], 0*darm[:, 2], marker='.', c='g')
ax2.set_xlabel('tiempo (frames)')
ax2.set_ylabel('posición horiz. (pixel)')
ax2.set_title('Proyección del movimiento del brazo dominante')

plt.legend()
plt.show()


#!/usr/bin/env python3
# -*- coding: utf-8 -*-

"""
Created on Fri Sept 10 17:16:00 2021
@author: mhenao, dini
"""

from utils import detector_utils as detector_utils
import cv2
import tensorflow.compat.v1 as tf
import matplotlib.pyplot as plt
```

```
import numpy as np

# global variables
vclip_name = "ttest5.mp4"
vclip_location = "../" + vclip_name
data_file = "found_" + vclip_name + ".data"
det_graph, sess = detector_utils.load_inference_graph()
det_threshold = 0.8
det_needs_vflip = True

# main method
if __name__ == "__main__":
    # load video
    vclip = cv2.VideoCapture(vclip_location)

    if not vclip.isOpened():
        print("Could not open video source")
        exit -1

    # get video dimensions
    width = vclip.get(cv2.CAP_PROP_FRAME_WIDTH)
    height = vclip.get(cv2.CAP_PROP_FRAME_HEIGHT)
    print("width = " + str(width))
    print("height = " + str(height))

    # get video frames per second
    fps = vclip.get(cv2.CAP_PROP_FPS)
    print("FPS = " + str(fps))

    # get total amount of frames in video
    framecount = vclip.get(cv2.CAP_PROP_FRAME_COUNT)
    print("Frame number = " + str(framecount))

    # data storage
    found = []

    # display read frame
    for i in range(int(framecount)):
        # get frame
        ret, frame = vclip.read()
        frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

        # perform a vertical flip if necessary
        if det_needs_vflip:
```

```
            frame = cv2.flip(frame, 0)

        # detect hands
        boxes, scores = detector_utils.detect_objects(frame, det_graph, sess)
        for j in range(len(scores)):
            if scores[j] > det_threshold:
                # calculate hand position
                (left, right, top, bottom) = (boxes[j][1] * width, boxes[j][3] * width, boxes[j][0] *
height, boxes[j][2] * height)
                x = int(left + 0.5*(right-left))
                y = int(bottom + 0.5*(top-bottom))

                # append found hand position
                found.append((i, x, y))

                # display found hand
                cv2.circle(frame, (x, y), 10, (0,0,255), cv2.FILLED)

        frame = cv2.flip(frame, 0)
        cv2.imshow('frame', frame)
        cv2.waitKey(20)
        print("Processing: " + str(i), end='\r')

    # release video
    vclip.release()
    cv2.destroyAllWindows()

    # save generated data for post-processing
    foundarr = np.array(found)
    np.save(data_file, foundarr)

    # print found hands along time
    fig = plt.figure(figsize=(20, 20))
    ax = fig.add_subplot(projection='3d')
    ax.scatter(foundarr[:, 0], foundarr[:,1], foundarr[:, 2], marker='.')
    ax.set_xlabel('tiempo (frames)')
    ax.set_ylabel('posición horiz. (pixel)')
    ax.set_zlabel('posición verti. (pixel)')
    plt.show()


import matplotlib.pyplot as plt
import numpy as np

# load generated data for post-processing
```

```
vclip_name = "ttest3.mp4"
data_file = "found_" + vclip_name + ".data.npy"
foundarr = np.load(data_file)
foundarr = foundarr.astype(np.float64)

print("\nFILE CHARACTERISTICS:")
print("Points found: " + str(len(foundarr)))

# configure calculations
fps = 240.0

# normalize
avx = np.average(foundarr[:,1])
avy = np.average(foundarr[:,2])
dvx = np.std(foundarr[:,1])
dvy = np.std(foundarr[:,2])

print("\nNORMALIZATION:")
print("avx=" + str(avx))
print("avy=" + str(avy))
print("dvx=" + str(dvx))
print("dvy=" + str(dvy))

foundarr[:,1] = (foundarr[:,1] - avx) / dvx
foundarr[:,2] = (foundarr[:,2]-avy) / dvy

# split points corresponding to dominant and non-dominant hand
npoints = len(foundarr)
ndarm = []
darm = []

for i in range(npoints):
    t = foundarr[i, 0]
    x = foundarr[i, 1]
    y = foundarr[i, 2]

    # split dominant / non-dominant hand
    if y < (-1.2)*(x**2)-0.2:
        ndarm.append((t, x, y))
    else:
        darm.append((t, x, y))

ndarm = np.array(ndarm)
ndarm[:,1] = ndarm[:,1]*dvx + avx
```

```
ndarm[:,2] = ndarm[:,2]*dvy + avy

darm =  np.array(darm)
darm[:,1] = darm[:,1]*dvx + avx
darm[:,2] = darm[:,2]*dvx + avx

# estimate dominant hand speed
speed = []
time_frame = []
for i in range(darm.shape[0]):
    if i>0:
        t2 = darm[i][0]
        t1 = darm[i-1][0]
        x2 = darm[i][1]
        x1 = darm[i-1][1]
        y2 = darm[i][2]
        y1 = darm[i-1][2]

        delta_x = x2 - x1
        delta_y = y2 - y1
        delta_t = t2 - t1

        dist = np.sqrt( (delta_y)**2 + (delta_x)**2 )
        ispeed = dist / delta_t

        speed.append(ispeed)
        time_frame.append(t2)


speed = np.array(speed)
time_frame = np.array(time_frame)

# estimate points where the arm changes direction
switchpoints = []
n_around = 10
inf_lim = n_around
sup_lim = darm.shape[0] - n_around

for i in range(darm.shape[0]):
    if i>inf_lim and i<sup_lim:
        curr_sl = (darm[i, 1] - darm[i-1, 1]) / (darm[i, 0] - darm[i-1, 0])
        curr_sl = np.sign(curr_sl)
        prev_sl = (darm[i-1, 1] - darm[i-2, 1]) / (darm[i-1, 0] - darm[i-2, 0])
        prev_sl = np.sign(prev_sl)
```

```
        if curr_sl != prev_sl:
            inf_ref = i - n_around
            sup_ref = i + n_around + 1

            #if curr_sl>0 and not np.any(np.less(darm[inf_ref:sup_ref, 1], darm[i-1, 1])):
            if curr_sl>0 and not np.any( np.less(darm[inf_ref:i-1, 1], darm[i-1, 1]) ) and not np.any(
np.less_equal(darm[i:sup_ref, 1], darm[i-1, 1]) ):
                switchpoints.append( (darm[i-1][0],darm[i-1][1])  )
            #if curr_sl<0 and not np.any(np.greater(darm[inf_ref:sup_ref, 1], darm[i-1, 1])):
            if curr_sl<0 and not np.any( np.greater(darm[inf_ref:i-1, 1], darm[i-1, 1]) ) and not
np.any( np.greater_equal(darm[i:sup_ref, 1], darm[i-1, 1]) ):
                switchpoints.append( (darm[i-1][0],darm[i-1][1])  )

switchpoints = np.array(switchpoints)
#print(switchpoints)

# smooth dominant hand movement plot
# smooth_darm = []
# window_size = 1
# inf_lim = window_size
# sup_lim = darm.shape[0]-window_size

# for i in range(darm.shape[0]):
#     if i>=inf_lim and i<sup_lim:
#         avg = np.average(darm[i-window_size:i+window_size+1, 1])
#         smooth_darm.append( (darm[i, 0], avg) )
# smooth_darm = np.array(smooth_darm)

# print found hands along time
fig, (ax1, ax2) = plt.subplots(1, 2)
ax1.axis('off')
ax2.axis('off')
ax1 = fig.add_subplot(1, 2, 1, projection='3d')
ax1.scatter(ndarm[:, 0], ndarm[:, 1], ndarm[:, 2], marker='.', c='r', label='Brazo no dominante')
ax1.scatter(darm[:, 0], darm[:, 1], darm[:, 2], marker='.', label='Brazo dominante')
ax1.set_xlabel('tiempo (frames)')
ax1.set_ylabel('posición horiz. (pixel)')
ax1.set_zlabel('posición verti. (pixel)')
ax1.set_title('Posiciones de las manos para la prueba Tapping-Test')

# print hand projection to the horizontal plane
ax2 = fig.add_subplot(1, 2, 2, projection='3d')
ax2.scatter(switchpoints[:, 0], switchpoints[:, 1], 0*switchpoints[:,0], marker='.', c='r')
ax2.plot(darm[:, 0], darm[:, 1], 0*darm[:, 2], marker='.', c='g')
```

```
ax2.set_xlabel('tiempo (frames)')
ax2.set_ylabel('posición horiz. (pixel)')
ax2.set_title('Proyección del movimiento del brazo dominante')

# print speed plot
fig2 = plt.figure(2)
ax3 = fig2.gca()
ax3.plot(darm[:,0], darm[:,2], marker='.')
ax4 = ax3.twinx()
ax4.plot(time_frame, speed, marker='.', c='r')
ax3.grid()

# print switchpoints plot
fig3 = plt.figure(3)
ax5 = fig3.gca()
ax5.scatter(switchpoints[:,0], switchpoints[:,1], marker='o', c='r')
ax5.plot(darm[:,0], darm[:,1], marker='.', c='g')
#ax5.plot(smooth_darm[:,0], smooth_darm[:,1], marker='.', c='b')
for (t,x) in switchpoints:
    point_txt = str(int(t))
    ax5.annotate(point_txt, (t,x))
ax5.grid()

# specify limits to take care when counting taps
inf_lim = 130
sup_lim = 2510

selected = np.logical_and(switchpoints[:,0]>inf_lim, switchpoints[:,0]<sup_lim)
fig4 = plt.figure(4)
ax6 = fig4.gca()
ax6.scatter(switchpoints[selected,0], switchpoints[selected,1], marker='o', c='r')
ax6.plot(darm[:,0], darm[:,1], marker='.', c='g')

i=1
for (t,x) in switchpoints[selected]:
    point_txt = str(i)
    ax6.annotate(point_txt, (t+20,x))
    i=i+1
ax6.grid()

plt.show()
```

Anexo8 . Código IDE.

```
#include <LiquidCrystal_I2C.h>
#include <WiFi.h>

const char* ssid = "Tapping test";
const char* pass = "1234567890";
int inputPin = A0;
int inputPin1 = A3;

LiquidCrystal_I2C lcd (0x27, 16,2); //

void  setup () {

  pinMode (inputPin, INPUT);
  pinMode (inputPin1, INPUT);
  lcd. init ();
  lcd. backlight ();
  lcd. print ( "Eather Tamer " );
  lcd.setCursor(0,1);
  lcd.print ("          COORP");
  delay(4700);
  lcd.clear();
  delay(5000);
  lcd. print ( "..... " );
  lcd.setCursor(0,1);
  delay(1000);
  lcd.print  ( "..... ");
  delay(5000);
```

```
lcd.clear();
lcd.print  ( "..... ");
delay(1000);
lcd.setCursor(0,1);
delay(1000);
lcd.clear();
delay(7000);
lcd. print ( "TAPPING TEST" );
delay(10000);
lcd.clear();
lcd. print ( " " );
delay(5000);
lcd.clear();
delay(3000);
lcd.setCursor(0,1);
lcd.print ("Inicializando.. ");
delay(5000);


Serial.begin(115200);

WiFi.begin(ssid, pass);
delay(2000);

Serial.print("Se está conectando a la red WiFi denominada  ");
Serial.println(ssid);
lcd.clear();
lcd.print("CONECTANDO A RED");
lcd.setCursor(0,1);
lcd.print("WIFI");
delay(3500);
```

```
 while (WiFi.status() != WL_CONNECTED) {
      delay(500);
      Serial.print(".");
      lcd.clear();
      lcd.setCursor(0,1);
      lcd.print("...");
   }

 Serial.println("");
 Serial.println("WiFi connected");
 Serial.println("IP address: ");

 Serial.println(WiFi.localIP());
 lcd.clear();
 lcd.print("CONECTANDO");
 lcd.setCursor(0,1);
 lcd.print(WiFi.localIP());
 delay(3500);
}

 void loop () {

  int anRg = analogRead(inputPin);

  Serial.println ("El valor del sensor derecho es = ");
  Serial.println (anRg);

 if (anRg < 10)       // from 0 to 9
   Serial.println(" -> No presión ");
 else if (anRg < 200) // from 10 to 199
```

```
  Serial.println(" -> Muy poca presión");
else if (anRg < 500) // from 200 to 499
Serial.println(" -> Presión Ligera");
else if (anRg < 800) // from 500 to 799
Serial.println(" -> Presión media");
else // from 800 to 1023
Serial.println(" -> Presión alta");

lcd.clear();
lcd.setCursor(0,0);
lcd.print("SsrDh: ");
lcd.setCursor(0,1);
lcd.print(anRg);

int anRg1 = analogRead(inputPin1);

 Serial.println ("El valor del sensor izquierdo es = ");
 Serial.println (anRg1);

if (anRg1 < 10)       // from 0 to 9
  Serial.println(" -> No presión ");
else if (anRg1 < 200) // from 10 to 199
Serial.println(" -> Muy poca presión");
else if (anRg1 < 500) // from 200 to 499
Serial.println(" -> Presión  Ligera");
else if (anRg1 < 800) // from 500 to 799
Serial.println(" -> Presión media");
else // from 800 to 1023
  Serial.println(" -> Presión alta");

lcd.clear();
```

```
 lcd.setCursor(10,0);
 lcd.print("SsrIz:  ");
 lcd.setCursor(10,1);
 lcd.print(anRg1);

  delay (10);


}
```

UNIVERSIDAD CESMAG NIT: 800.109.387-7 VIGILADA MINEDUCACIÓN	**CARTA DE ENTREGA TRABAJO DE GRADO O TRABAJO DE APLICACIÓN – ASESOR(A)**	**CÓDIGO:** AAC-BL-FR-032
		VERSIÓN: 1
		FECHA: 09/JUN/2022

San Juan de Pasto, __3/03/2023_

Biblioteca
REMIGIO FIORE FORTEZZA OFM. CAP.
Universidad CESMAG
Pasto

Saludo de paz y bien.

Por medio de la presente se hace entrega del Trabajo de Grado / Trabajo de Aplicación denominado **"Estimación de la fuerza ejercida por el brazo en la prueba tapping test utilizando visión ariticial_",** presentado por el (los) autor(es) Diego Nicolás Zambrano Velasco del Programa Académico INGENIERÍA ELECTRONICA al correo electrónico trabajosdegrado@unicesmag.edu.co. Manifiesto como asesor(a), que su contenido, resumen, anexos y formato PDF cumple con las especificaciones de calidad, guía de presentación de Trabajos de Grado o de Aplicación, establecidos por la Universidad CESMAG, por lo tanto, se solicita el paz y salvo respectivo.

Atentamente,

Mario Fernando Henao Rosero
C.C. **4.612719**
Programa: **Ingeniería Electrónica**
Cel. **3163455039**
C.E.

INFORMACIÓN DEL (LOS) AUTOR(ES)	
Nombres y apellidos del autor: Diego Nicolás Zambrano Velasco	**Documento de identidad:** 1233191532
Correo electrónico: diegonzambrano@gmail.com	**Número de contacto:** 317491248
Nombres y apellidos del autor:	**Documento de identidad:**
Correo electrónico:	**Número de contacto**
Título del trabajo de grado: **''Estimación de la fuerza ejercida por el brazo en la prueba tapping test utilizando visión artificial''**	
Facultad y Programa Académico: **Facultad de Ingenieria** **Ingenieria Electronica**	

En mi (nuestra) calidad de autor(es) y/o titular (es) del derecho de autor del Trabajo de Grado o de Aplicación señalado en el encabezado, confiero (conferimos) a la Universidad CESMAG una licencia no exclusiva, limitada y gratuita, para la inclusión del trabajo de grado en el repositorio institucional. Por consiguiente, el alcance de la licencia que se otorga a través del presente documento, abarca las siguientes características:

a) La autorización se otorga desde la fecha de suscripción del presente documento y durante todo el termino en el que el (los) firmante(s) del presente documento conserve(mos) la titularidad de los derechos patrimoniales de autor. En el evento en el que deje(mos) de tener la titularidad de los derechos patrimoniales sobre el Trabajo de Grado o de Aplicación, me (nos) comprometo (comprometemos) a informar de manera inmediata sobre dicha situación a la Universidad CESMAG. Por consiguiente, hasta que no exista comunicación escrita de mi(nuestra) parte informando sobre dicha situación, la Universidad CESMAG se encontrará debidamente habilitada para continuar con la publicación del Trabajo de Grado o de Aplicación dentro del repositorio institucional. Conozco(conocemos) que esta autorización podrá revocarse en cualquier momento, siempre y cuando se eleve la solicitud por escrito para dicho fin ante la Universidad CESMAG. En estos eventos, la Universidad CESMAG cuenta con el plazo de un mes después de recibida la petición, para desmarcar la visualización del Trabajo de Grado o de Aplicación del repositorio institucional.
b) Se autoriza a la Universidad CESMAG para publicar el Trabajo de Grado o de Aplicación en formato digital y teniendo en cuenta que uno de los medios de publicación del repositorio institucional es el internet, acepto(amos) que el Trabajo de Grado o de Aplicación circulará con un alcance mundial.
c) Acepto (aceptamos) que la autorización que se otorga a través del presente documento se realiza a título gratuito, por lo tanto, renuncio(amos) a recibir emolumento alguno por la publicación, distribución, comunicación pública y/o cualquier otro uso que se haga en los términos de la presente autorización y de la licencia o programa a través del cual sea publicado el Trabajo de grado o de Aplicación.

d) Manifiesto (manifestamos) que el Trabajo de Grado o de Aplicación es original realizado sin violar o usurpar derechos de autor de terceros y que ostento(amos) los derechos patrimoniales de autor sobre la misma. Por consiguiente, asumo(asumimos) toda la responsabilidad sobre su contenido ante la Universidad CESMAG y frente a terceros, manteniéndola indemne de cualquier reclamación que surja en virtud de la misma. En todo caso, la Universidad CESMAG se compromete a indicar siempre la autoría del escrito incluyendo nombre de(los) autor(es) y la fecha de publicación.
e) Autorizo(autorizamos) a la Universidad CESMAG para incluir el Trabajo de Grado o de Aplicación en los índices y buscadores que se estimen necesarios para promover su difusión. Así mismo autorizo (autorizamos) a la Universidad CESMAG para que pueda convertir el documento a cualquier medio o formato para propósitos de preservación digital.

NOTA: En los eventos en los que el trabajo de grado o de aplicación haya sido trabajado con el apoyo o patrocinio de una agencia, organización o cualquier otra entidad diferente a la Universidad CESMAG. Como autor(es) garantizo(amos) que he(hemos) cumplido con los derechos y obligaciones asumidos con dicha entidad y como consecuencia de ello dejo(dejamos) constancia que la autorización que se concede a través del presente escrito no interfiere ni transgrede derechos de terceros.

Como consecuencia de lo anterior, autorizo(autorizamos) la publicación, difusión, consulta y uso del Trabajo de Grado o de Aplicación por parte de la Universidad CESMAG y sus usuarios así:

- Permito(permitimos) que mi(nuestro) Trabajo de Grado o de Aplicación haga parte del catálogo de colección del repositorio digital de la Universidad CESMAG, por lo tanto, su contenido será de acceso abierto donde podrá ser consultado, descargado y compartido con otras personas, siempre que se reconozca su autoría o reconocimiento con fines no comerciales.

En señal de conformidad, se suscribe este documento en San Juan de Pasto a los__3__días del mes de _Marzo___del año __2023___

 Firma del autor	Firma del autor
Firma del autor	Firma del autor
Nombre del autor:	Nombre del autor:
 Firma del asesor	