Instituto Tecnológico de Costa Rica

Escuela de Ingeniería en Electrónica

Diseño de un Algoritmo mediante procesamiento de imágenes para la determinación y control de la posición del brazo Amatrol Pegasus I en el Instituto Tecnológico de Costa Rica Sede San Carlos

Informe de Proyecto de Graduación para optar por el título de Ingeniero en Electrónica con el Grado Académico de Licenciatura

Randald Durán Solano

San Carlos, diciembre de 2018

INSTITUTO TECNOLOGICO DE COSTA RICA
ESCUELA DE INGENIERIA ELECTRONICA
PROYECTO DE GRADUACIÓN
TRIBUNAL EVALUADOR
ACTA DE EVALUACIÓN

Proyecto de Graduación defendido ante el presente Tribunal Evaluador como requisito para optar por el título de Ingeniero en Electrónica con el grado académico de Licenciatura, del Instituto Tecnológico de Costa Rica.

Estudiante: DURÁN SOLANO RANDALD

Carné: 2013027110

Nombre del Proyecto: *"Diseño de un Algoritmo mediante procesamiento de imágenes para la determinación y control de la posición del brazo Amatrol Pegasus I en el Instituto Tecnológico de Costa Rica Sede San Carlos"*

Miembros del Tribunal

Ing. Luis Diego Gómez Rodríguez

Profesor lector

Ing. Luis Miguel Esquivel Sancho

Profesor lector

Ing. Pablo César Rodríguez Vargas

Profesor asesor

Los miembros de este Tribunal dan fe de que el presente trabajo de graduación ha sido aprobado y cumple con las normas establecidas por la Escuela de Ingeniería Electrónica

Nota Final del Proyecto de Graduación : 95

San Carlos, 29 de enero 2019

RESUMEN

El documento presenta la propuesta de diseño, evaluación e implementación de un algoritmo que permita dar movilidad a un equipo industrial. El proyecto fue planteado por la Escuela de Ingeniería en Electrónica en la sede regional de San Carlos, con el objetivo de reactivar el uso del brazo industrial Amatrol's Pegasus I que se encuentra en el laboratorio de sistemas digitales.

Se planteo el proyecto para la elaboración de un algoritmo que mediante procesamiento de imágenes permita determinar la posición del equipo y los ángulos de inclinación de sus extremidades para poder generar un control retroalimentado de los motores.

El proyecto se trabajó en tres modulos principales, el proceso de detección, seguimiento y procesado de las imágenes, el sistema de comunicación para trasiego de información y el algoritmo de procesado y activación de los motores del microcontrolador.

El objetivo de este informe es exponer como se desarrolló el algoritmo y el proceso de investigación llevado a cabo; para la implementación se realizó el análisis de distintas alternativas de solución correspondientes al hardware y software, se realizaron distintas pruebas de ejecución y el análisis de los factores a tomar en cuenta en el desarrollo de un algoritmo implementando visión por computadora.

Se desarrollaron los procesos de trabajo para cada entrada de visión independiente entre sí, y el diseño de software para la activación de las señales de control respectivas de los motores de corriente continua requeridos para el posicionamiento.

Palabras clave: Algoritmo, visión por computadora, módulos, Hardware, Software, Microcontrolador, Retroalimentación

ABSTRACT

The document presents the proposal of design, evaluation and implementation of an algorithm that allows to give mobility to an industrial gear. The project was raised by the School of Electronics Engineering at the regional headquarters of San Carlos, with the aim of reactivating the use of the Amatrol's Pegasus I industrial arm that is in the digital systems laboratory.

The project was proposed for the elaboration of an algorithm that through image processing allows to determine the position of the equipment and the angles of inclination of its extremities to be able to generate a feedback control of the motors.

The project was worked on three main modules, the process of detection, tracking and processing of the images, the communication system for information transfer and the algorithm of processing and activation of the microcontroller motors.

The objective of this report is to show how the algorithm and the research process carried out were developed, for the implementation was made the analysis of different solution alternatives corresponding to the hardware and software, different execution tests were carried out and the analysis of the factors to consider in the development of an algorithm implementing computer vision.

The work processes were developed for each independent vision input, and the software design for the activation of the respective control signals of the DC motors required for the positioning.

Keywords: Algorithm, computer vision, modules, Amatrol's Pegasus I, Hardware, Software, Microcontroller, Feedback, DC

DEDICATORIA

A mis padres, que me han dado su apoyo a lo largo de los años y me han enseñado el valor de trabajo, por guiarme y luchar día a día para permitirme alcanzar mis metas.

A mi hermano por su amistad y apoyo en todos nuestros años y a mi hermana por su tiempo.

AGRADECIMIENTO

A Julio Zamora, José Rosales, Liliana Boza, Katherine Herrera, Leninyer Pérez, Brayanth Rodríguez, Dayna Arroyo, Kevin Solano, Álvaro Sossa y todos los estudiantes de electrónica por ser un espacio de paz, apoyo y compañía a lo largo del camino que recorrimos juntos.

Al profesor Rogelio González por su ayuda y disponibilidad a lo largo del desarrollo del proyecto aún a la distancia.

A Cynthia Artavia por ayudarme incontables veces durante toda mi carrera, por su ayuda, su apoyo y su lucha para llegar al final de la meta, por siempre dar más de lo necesario por los estudiantes.

A Kevin Hernández por ayudarme más allá de los límites, por estar disponible cuando fue necesitado.

A Delia Porras por brindarme su ayuda en tiempos difíciles, ayudarme a continuar y darme una guía y una base para el proceso, por ser una profesora y una amiga.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPITULO 1: INTRODUCCIÓN

1.1. Problema existente e importancia de su solución

El Pegasus I pertenece a la compañía Amatrol®, la cual es una empresa dedicada a diseñar, desarrollar y fabricar sistemas de aprendizajes técnicos, eLearning interactivo y simuladores educativos para el desarrollo de proyectos o trabajos dentro de compañías o centros educativos.

El robot industrial Pegasus I es un equipo articulado de cinco ejes, esto significa que todos los ejes son de tipo rotatorio; para este tipo de robot los tres primeros ejes se denominan: cintura, codo y hombro, desde la pinza exterior hasta la base del brazo. En la Figura 1.1 se puede observar una imagen representativa del Pegasus I.

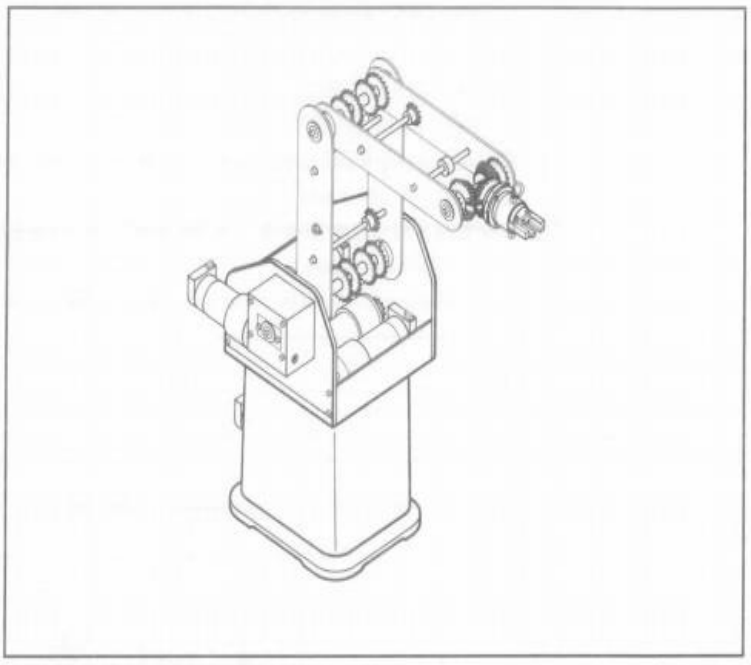


Figura 1.1.Modelo Pegasus I

Fuente: Amatrol. (12 de septiembre de 2018). *Moder control technology.* Obtenido de http://www.moderncontroltechnology.com/docs/Peggy_Robot/Vendor/Amatrol/RobotActivityModule.pdf

Cada uno de los ejes brinda un desplazamiento espacial al equipo en ambos sentidos de giro mediante la implementación de motores de corriente continua, cada motor requiere de una señal eléctrica que establezca el sentido de rotación y otra que brinde el valor de voltaje correspondiente a la velocidad de giro. En la Figura 1.2 se muestran los ejes de rotación del Pegasus I.

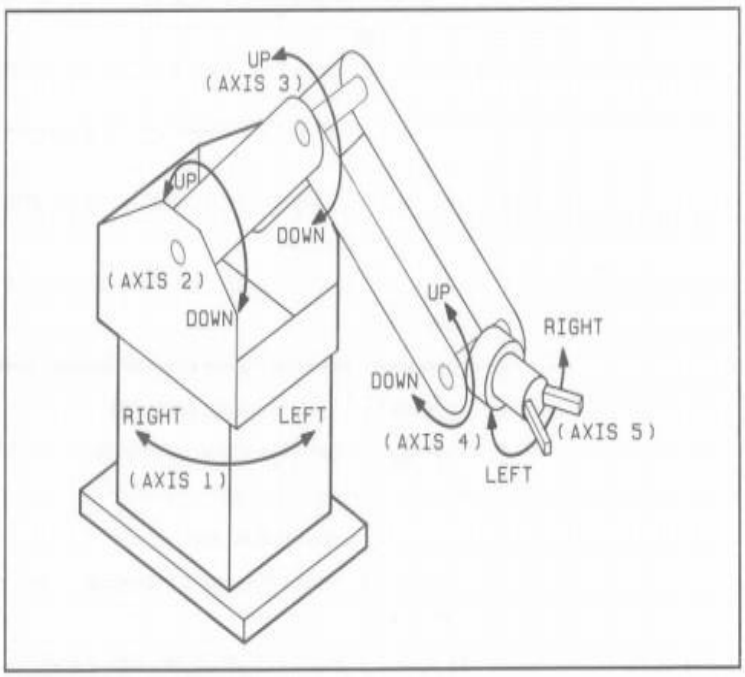


Figura 1.2. Ejes de rotación del Pegasus
Fuente: Amatrol. (12 de septiembre de 2018). *Moder control technology*. Obtenido de http://www.moderncontroltechnology.com/docs/Peggy_Robot/Vendor/Amatrol/RobotActivityModule.pdf

El Pegasus I cuenta con un sistema de posicionamiento y control compuesto por un conjunto de encoders rotatorios electromagnéticos encargados de transcribir la posición angular de un eje en un código digital y un software de procesamiento en su versión 1.45, este software está creado para procesar la información, también brinda instrucciones de movimiento o procesos consecutivos de desplazamientos mediante motores de corriente continua.

Actualmente el pegasus se encuentra en la Escuela de Ingeniería Electrónica y no cuenta con el software de programación, el cableado de conexión hacia las señales de control se encuentra incompleto y no posee las entradas físicas para brindar conexión a los equipos periféricos de funcionamiento, de igual forma los encoders de posicionamiento presentan problemas de calibración.

El pegasus presenta un sistema de alimentación y las conexiones hacia los motores, así como un conjunto de puentes H que permiten realizar un control para habilitar el movimiento de los motores en ambos sentidos de giro y una señal binaria para ejecutar un control de la velocidad.

Debido a la falta de un sistema fiable de medición de posición y a la ausencia de un método de control para el uso del equipo, es necesario desarrollar un sistema que brinde funcionalidad y sea independiente al sistema de medición con el que se cuenta actualmente, para evitar fallos por factores mecánicos.

Se plantea una solución digital que permita realizar el muestreo constante de los valores de posición del equipo y brindar un desplazamiento requerido mediante un controlador externo.

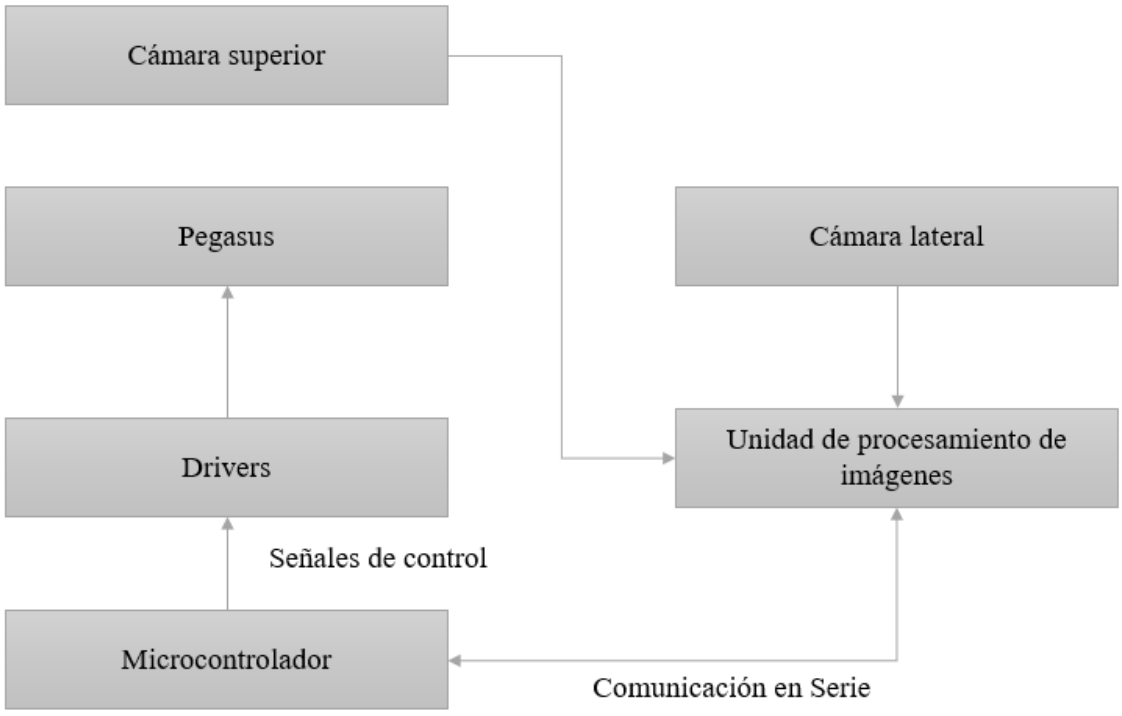


Figura 1.3. Diagrama de solución
Fuente: Elaboración propia

Se busca el diseño de un algoritmo que permita realizar captura de imagen mediante una cámara lateral y una superior, calculando en tiempo real el valor de los ángulos de inclinación

entre las extremidades y la posición de la base sobre su eje de movimiento, la información es retroalimentada hacia un microcontrolador que se encarga de ejecutar la activación de las señales de control determinando el sentido de giro necesario para cada motor.

La importancia de un algoritmo de procesamiento para el control del equipo radica en su independencia de los sistemas electromecánicos, brindando una alternativa que no representa desgaste físico por uso constante y que puede ser definida y modificada para dar trabajo de manera más sencilla y rápida.

La implementación de un algoritmo para el análisis y control representa beneficios sobre los sistemas físicos actuales, estos beneficios son:

- Definición de parámetros de uso adaptables a las condiciones propias de cada sistema mecánico sobre el que se va a implementar.
- Libertad de selección de diferentes puntos de detección para el proceso de información
- Incorporación de diferentes funciones matemáticas para el cálculo de múltiples datos.
- Exportable a distintos sistemas mecánicos.

1.2. Solución Seleccionada

En esta sección se describen los requerimientos establecidos para el desarrollo del proyecto y se plantea la solución a desarrollar.

La Tabla 1.1 delimita los requerimientos definidos para el proyecto, esto con el fin de hacerlos más específicos para su desarrollo y como puntos de partida y gestión del proceso de trabajo.

Tabla 1.1. Requerimientos de solución

Parámetro	Especificación
Limites operativos y mecánicos	▪ El microcontrolador dará movilidad mediante señales digitales a los módulos actuadores encargados de la activación de los motores. ▪ Las áreas límites de operación del brazo se definirán de manera práctica tomando en cuenta los desplazamientos que no presente riesgo de daño mecánico y los límites de detección visual. ▪ Se controlarán al menos 3 motores de corriente continua ▪ Se contará con no más de 2 puntos de visión ▪ El sistema contara con comunicación entre la unidad de procesamiento de imágenes y el microcontrolador
Tiempo de operación	El tiempo operativo depende de las posiciones seleccionadas por el usuario El tiempo de comunicación se determinará según la velocidad de operación de los motores
Interfase con el operador	El operador contara con la selección de los valores angulares de las extremidades y un valor coordenado al que desea desplazar el brazo.
Características estándar	El microcontrolador podrá leer y escribir información por medio de conexión en serie.

Fuente: Elaboración propia

El sistema empotrado al ser un dispositivo aparte del encargado de procesar las imágenes debe ejecutársele dos tipos de pruebas, una de ellas son las pruebas de velocidad de lectura de información, debido a que se trabaja con un procesador más lento y, por último, las pruebas de salida de señales para determinar su capacidad para controlar la velocidad de los pines habilitadores de señal en los actuadores.

1.2.1. Alcances y limitaciones

Esta sección pretende limitar el desarrollo del proyecto desde su inicio hasta su conclusión, el proyecto parte de un análisis y estudio bibliográfico sobre los temas enfocados al desarrollo y análisis en la rama del procesamiento de imágenes, se plantea el desarrollo metodológico a abordar, desde el desarrollo del código hasta la implementación de las señales sobre los actuadores, además, se estudian las diferentes alternativas de diseño y se evalúan bajo parámetros de selección definidos en el capítulo cuatro.

La propuesta se divide en módulos, cada módulo se analiza de manera independiente y las siguientes limitaciones se establecen según las capacidades de hardware, software y versatilidad en la mecánica del brazo para desempeñar las funciones:

- El rango angular de rotación de la base se determinará de manera práctica basándose en el ángulo de rotación para el cual la cámara lateral no detecta las marcas de color.
- En caso de ser necesario la activación de los motores se realizará de manera secuencial para evitar el riesgo de daño mecánico o falsas lecturas.
- La detección y rastreo de los puntos de medición se llevará a cabo a través de filtrado de color.
- Se trabajará bajo un ambiente controlado en términos de exposición a la luz y coloración de fondo para no afectar los filtros de selección de color.
- El desplazamiento horizontal del Pegasus I se definirá por el ángulo de apertura visual de la cámara superior y el punto en el que no se presenta la pérdida del color por la cámara lateral.

La solución planteada establece el desarrollo de un algoritmo que permita hacer una detección de posicionamiento del pegasus y darle movimiento mediante un microcontrolador.

Se busca implementar dos dispositivos de entradas de video que permitan hacer una toma de imagen constante de la parte lateral y superior del brazo, se colocan secciones de color sobre los puntos de pivote del Amatrol Pegasus y sobre una sección de referencia de la base; las tomas son procesadas mediante filtrado y morfologías para hacer seguimiento de los puntos, mediante bibliotecas matemáticas se realiza el cálculo de posición con respecto a la matriz de la cámara y se determinan los ángulos entre las extremidades, la unidad de procesamiento de imágenes se encarga de realizar la determinación constante de los valores adquiridos por

video, mantiene la comunicación constante de la lectura y escritura con el microcontrolador para permitir una transmisión constante de los datos, el microcontrolador procesa la información y se encarga de realizar la activación de los motores, desplaza el brazo a la posición solicitada y por último, corrobora que se alcanzó el punto deseado.

CAPITULO 2: META Y OBJETIVOS

2.1. Meta

Diseñar un algoritmo que permita determinar la posición del brazo pegasus I y mediante retroalimentación de señales de control dar movilidad a una posición deseada.

2.2. Objetivo general

Desarrollar un algoritmo que permita determinar la posición y controlar el desplazamiento del Pegasus I mediante un procesamiento de imágenes.

2.3. Objetivos específicos

- Diseñar un código de detección y seguimiento de color para el cálculo de ángulos y posicionamiento de la base por medio de un procesamiento de imágenes en tiempo real.
- Determinar el comportamiento en la detección de los ángulos de la cámara lateral dada una rotación de la base.
- Implementar el código de detección y retroalimentación para el control de los motores.

CAPITULO 3: MARCO TEÓRICO

En este capítulo se tratan los principales conceptos relacionados con la solución del problema, se presenta la descripción de los principales principios físicos y electrónicos presentes en el documento.

3.1 Descripción del Sistema o proceso para mejorar

En la

Figura 3.1 diagrama de flujo de la secuencia de etapas del proceso, el diagrama muestras las acciones que llevan a cabo para la ejecución de la solución desde la entrada hasta el desplazamiento final desde un nivel más abstracto.

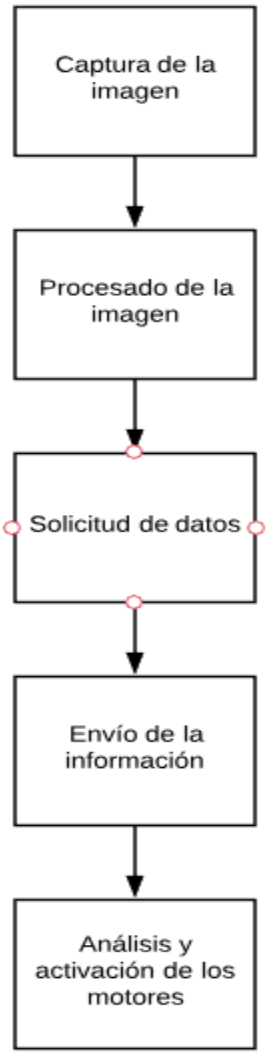


Figura 3.1 Diagrama del proceso de solución
Fuente: Elaboración propia

El Pegasus es un brazo creado para llevar a cabo procesos industriales diseñado bajo un sistema de motores y una estructura de piñones se logra dar movilidad al equipo.

El proceso de mejora que busca la implementación de la solución brinda un sistema alternativo digital que permita determinar posición y brinda movilidad al equipo, esto ante la presente de que el estado actual del Pegasus es un de inactividad ante los problemas de calibración en los sistemas de medición y la falta componentes del equipo que permitan usar el software de desarrollo.

3.2 Antecedentes Bibliográficos

En esta sección se definen las posibles tecnologías, métodos o procedimientos existentes a emplear en el desarrollo de la solución y las opciones presentes en el mercado para el área a trabajar.

3.2.1. Tensorflow

Tensorflow es una biblioteca de software libre que se utiliza para realizar cálculos números mediante diagramas de flujo de datos. Los nodos representan operaciones matemáticas de los tensores que están comunicados entre ellos. El sistema de aprendizaje automático es desarrollado por Google Brain, y puede correr sobre múltiples CPU y GPU.

Lo datos son seleccionas para determinar su selección bajo un conjunto de elementos que puede ser representados por un arreglo y a los cuales se les puede dar características específicas de detección, el modelo requiere un entrenamiento para la construcción de la red.

3.2.2. OpenCV

OpenCV representa una librería de programación principalmente dirigida a la visión por computadora en tiempo real, el sistema fue desarrollado como una propuesta de investigación de Intel para generar aplicaciones intensivas del CPU. OpenCV presenta más de 2500 algoritmos optimizados y algoritmos de aprendizaje.

3.3.3. YOLO

YOLO es un proyecto libre independiente diseñado para la detección de objetos en tiempo real el sistema aplica una imagen desde múltiples ubicaciones y escalas puntuando las regiones de la imagen para crear una base de datos que permita realizar su detección, cuenta con una red entrenada que realiza determinación de múltiples objetos de forma simultánea

utilizando sistemas de división por regiones, saltos de predicción y probabilidades por región para generar selección final. El sistema de YOLO se encuentra en el mercado y es libre uso, con una base de datos estándar predeterminada para la detección de los factores mas comunes presentes en una imagen, pero requiere entrenamiento adicional para detecciones específicas.

3.3 Principios Físicos y Electrónicos

En esta sección se definen todos los conceptos físicos y electrónicos presentes en el documento y que conciernen a la solución implementada

3.3.1. El color

El color es una cualidad perceptiva relacionada con la composición y descomposición espectral de la luz visible al incidir en el ojo humano. La luz se denomina acromática si su único parámetro visible es la intensidad, para establecer el valor de la intensidad se utilizan tres cantidades:

- radiancia: potencia de la fuente luminosa, medida en watts (W)
- luminancia: cantidad de energía percibida, medida en lúmenes (lm)
- brillo: descriptor subjetivo, relacionado con la noción acromática de la intensidad.

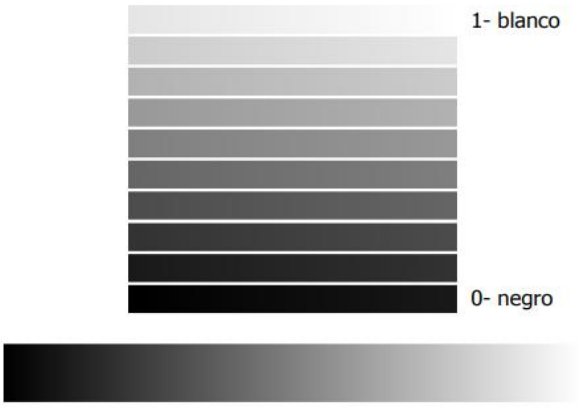


Figura 3.2 Diagrama de luz acromática
Fuente: Guerrero, R. (2015). *Teoría del color.* San Luis, Argentina: Universidad Nacional de Argentina.

En el ojo humano, la percepción del color es posible debido a tres, longitudes de onda, largas (rojo, 65%), medias (verde, 33%) y cortas (azul, 2%). (Guerrero, 2015)

La constitución tricromática de la estructura de la retina justifica la posibilidad de percibir los tres colores primarios rojo, verde y azul, la mezcla de los tres colores de manera aditiva resulta en luz blanca y de manera sustractiva en color negro.

Los colores secundarios cian, magenta, y amarillo se producen de la combinación de dos colores primarios de forma aditiva.

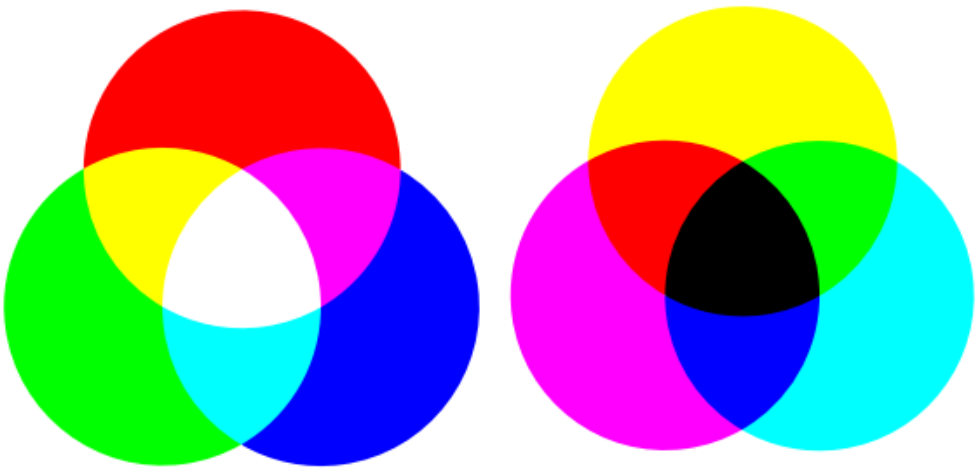


Figura 3.3 Combinación de colores aditivos y sustractivos
Fuente: Alvarado Moya, J. P. (2012). *Procesamiento y Análisis de Imágenes Digitales.* Cartago, Costa Rica: Instituto Tecnológico de Costa Rica.

Los tres componentes principales de color crean un espacio conocido como RGB. Estas definiciones se utilizan en sistema de despliegue de imagen como tubos catódicos, monitores de cristal líquido y plasma. Sin embargo, como son valores que se derivan de una curva espectral, no existe un único valor rojo, azul o verde. (Alvarado Moya, 2012)

3.3.2. Características del color

Para caracterizar las propiedades perceptivas del color por parte del sistema visual humano se utilizan tres parámetros:

- Brillo: corresponde a la noción acromática de la intensidad de luz
- Matriz: relacionado con la frecuencia dominante del espectro de luz incidente
- Croma o saturación: relacionado con la pureza de la frecuencia dominante.

La comisión internacional de iluminación (CIE) estandarizo las curvas de ponderación de la distribución espectral de potencia para calcular los valores x, y, z con los que se describe el color (Alvarado Moya, 2012)

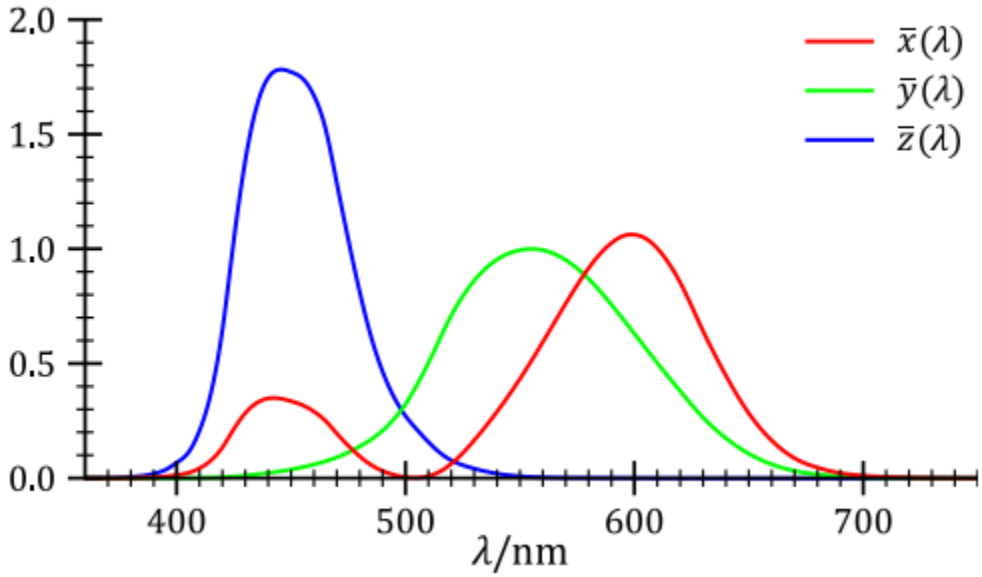


Figura 3.4 Longitud de onda de los colores primarios
Fuente: Alvarado Moya, J. P. (2012). *Procesamiento y Análisis de Imágenes Digitales.* Cartago, Costa Rica: Instituto Tecnológico de Costa Rica.

Con las densidades espectrales de potencia estándar y asumiendo que $I(\lambda)$ es la distribución espectral de potencia de luz incidente, se calculan los valores tri estimulo de x, y, z con la representación de las variables como:

$$X = \int_0^\infty I(\lambda)\bar{x}\ (\lambda)\ d(\lambda) \quad (3.1)$$

$$Y = \int_0^\infty I(\lambda)\bar{y}\ (\lambda)\ d(\lambda) \quad (3.2)$$

$$Z = \int_0^\infty I(\lambda)\bar{z}\ (\lambda)\ d(\lambda) \quad (3.3)$$

(Alvarado Moya, 2012)

3.3.3. HSV

Es un modelo basado en coordenadas polares y no en cartesianas. HSV no tiene una transformación lineal con el modelo RGB

$$H = \begin{cases} no\ definido, & si\ \ MAX = MIN \\ 60^o * \frac{G-B}{MAX-MIN} + 0, & si\ MAX = R\ y\ G \geq B \\ 60^o * \frac{G-B}{MAX-MIN} + 360, & si\ MAX = R\ y\ G < B \\ 60^o * \frac{G-B}{MAX-MIN} + 120, & si\ MAX = G \\ 60^o * \frac{G-B}{MAX-MIN} + 240, & si\ MAX = B \end{cases} \quad (3.4)$$

Hue (tono): Es un atributo asociado con la longitud de onda dominante en una mezcla de ondas luminosas.

Saturación: Se refiere a la pureza relativa o cantidad de luz blanca mezclada con un tono

Value (luminiscencia): está vinculado con la intensidad de luz. (Guerrero, 2015)

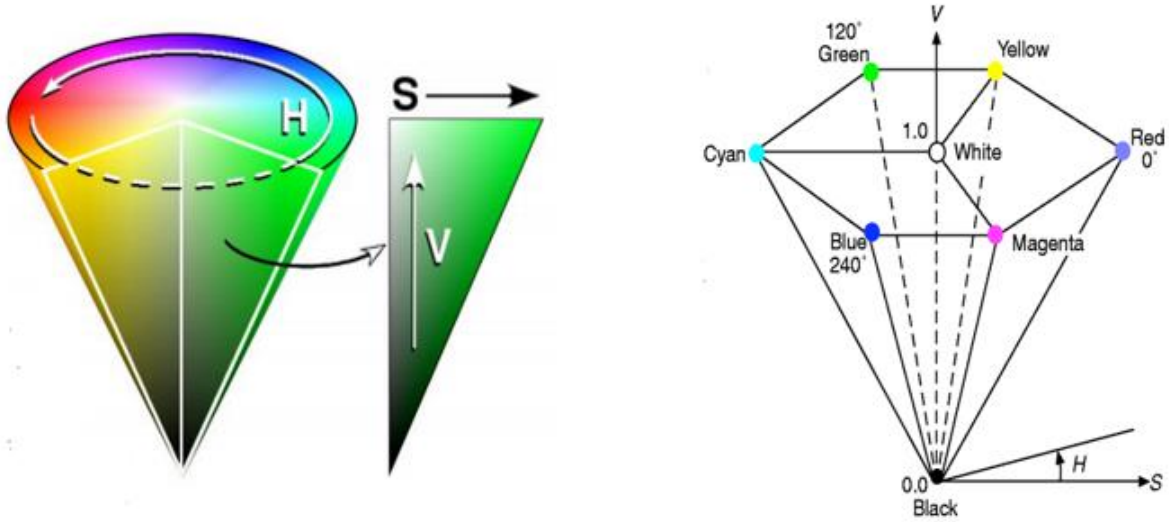


Figura 3.5 Modelo de color HSV
Fuente: Alvarado Moya, J. P. (2012). *Procesamiento y Análisis de Imágenes Digitales.* Cartago, Costa Rica: Instituto Tecnológico de Costa Rica.

Las cámaras utilizan el modelo RGB para determinar el color de un objeto, cuando la cámara lee los valores, estos se convierten a valores HSV. Los valores HSV se utilizan en el código para determinar la posición del objeto que se quiere rastrear. El modelo de color de HSV brinda mejor información para análisis computacional, establecer un rango de variación de color en HSV brinda la posibilidad de seleccionar una frecuencia de color y moverse sobre sus parámetros para cubrir las distintas tonalidades en diferente exposición. (Guerrero, 2015)

OpenCV trabaja con valores de 8 bits que permiten un valor de 180 para representa la frecuencia principal por lo que los valores angulares de HSV se dividen entre dos dando un rango de 0-180, los parámetros de valor y saturación aumentan de 0 a 255 en términos porcentuales

3.3.3. Morfologías

La morfología matemática se basa en operaciones de teoría de conjuntos en el caso de imágenes binarias. Las operaciones morfológicas simplifican imágenes y conservan las principales características de forma de los objetos, un sistema de operadores de este tipo y su composición permite que las formas subyacentes sean identificadas y reconstruidas. (Alvarado Moya, 2012)

3.3.3.1. Dilatación y erosión

La erosión de A por el elemento estructural B se denota $A \ominus B$ y se define como

$$A \ominus B = \left\{ \underline{x} \middle| (B)_{\underline{z}} \subseteq A \right\} \quad (3.5)$$

$$= \left\{ \underline{x} \middle| (B)_{\underline{z}} \subseteq A^C = \emptyset \right\} \quad (3.6)$$

El conjunto resultante de la erosión es aquel de todos los puntos $\underline{x}$ tales que si el elemento estructural B se desplaza en $\underline{x}$, todos sus elementos forman parte de A.

La dilatación de A con el elemento estructural B, denota como $A \otimes B$ se define como

$$A \otimes B = \left\{ \underline{z} \middle| (\hat{B})_{\underline{z}} \subseteq \emptyset \right\} \quad (3.7)$$

$$= \left\{ \underline{z} \middle| \left[(\hat{B})_{\underline{z}} \cap A \right] \subseteq A \right\} \quad (3.8)$$

En el proceso de dilatación el elemento estructural es reflejado con respecto a su origen antes de ser trasladado. La dilatación de A por B es el conjunto de todos los desplazamientos $\underline{x}$ tales que B y A se traslapan en al menos un elemento.

La dilatación y la erosión son operaciones duales entre sí, con respecto al complemento y reflexión de conjuntos.

$$(A \ominus B)^C = A^C \otimes \hat{B} \quad (3.9)$$

$$(A \otimes B)^C = A^C \ominus \hat{B} \quad (3.10)$$

Para elementos estructurales simétricos que cumplen $B = \hat{B}$, la erosion de una imagen con B se obtiene dilatando el fondo de la imagen A^C (Alvarado Moya, 2012)

3.3.3.2. Apertura y clausura

La apertura y la clausura son operadores compuestos de erosiones y dilataciones. La apertura usualmente suaviza el contorno de objetos, eliminando istmos delgados, y pequeñas penínsulas. La clausura fusiona pequeños "agujero" en los bordes o golfos delgados, así como cierra agujeros y rendijas.

La apertura del conjunto A por el elemento estructural B se denota como $A \circ B$ y se calcula con la siguiente fórmula:

$$A \circ B = (A \ominus B) \otimes B \quad (3.11)$$

La apertura se obtiene erosionando A con B y luego dilatando el resultado con B. La apertura tiene las siguientes propiedades:

$A \circ B$ es un subconjunto de A

Si C es un subconjunto de D, entonces $C \circ D$ es un subconjunto de $D \circ B$

$$(A \circ B) \circ B = A \circ B \quad (3.12)$$

La clausura del conjunto A con el elemento estructural B se denota $A \cdot B$ y está definida como

$$A \cdot B = (A \otimes B) \ominus B \quad (3.13)$$

Lo que indica que clausurar con B es equivalente a dilatar con B. y al resultado erosionarlo por el mismo elemento estructural. La clausura tiene las siguientes propiedades:

A es un subconjunto $A \cdot B$

Si C es un subconjunto de D, entonces $C \cdot D$ es un subconjunto de $D \subset B$

$$(A \cdot B) \cdot B = A \cdot B \quad (3.14)$$

La clausura y la apertura son duales entre sí, con respecto al complemento y reflexión

$$(A \cdot B)^C = (A^C \circ \hat{B}) \quad (3.15)$$

$$(A \circ B)^C = (A^C \cdot \hat{B}) \quad (3.16)$$

(Alvarado Moya, 2012)

3.3.4. Estereoscopía

Desde un punto de vista técnico, la estereoscopia se refiere al desarrollo, mediante el uso de una cámara especial de doble lente, de dos fotografías simultaneas de un mismo referente. (Crespo Armaiz, 2016)

La visión estereoscópica toma como modelo el sistema estereoscópico biológico donde el desplazamiento relativo de los ojos permite obtener la profundidad de los objetos o tercera dimensión mediante un simple proceso de triangulación a partir de las dos imágenes creadas por el mismo objeto de la escena 3D en cada ojo. (Gerrero Hernández, Pajares Martisanz, & Mata García, 2014)

En un sistema artificial de visión estereoscópica generalmente se utilizan dos cámaras separadas entre sí una cierta distancia relativa con las que se obtienen las respectivas imágenes del par estéreo El procedimiento consiste en captar dos imágenes de una misma escena, cada imagen es capturada desde una posición de las cámaras ligeramente diferente, por lo que las imágenes se presentan también ligeramente desplazadas entre sí, siendo éste el fundamento básico de la visión estereoscópica, ya que este hecho es el que va a permitir la obtención de la distancia a la que se encuentra un determinado objeto. (Gerrero Hernández, Pajares Martisanz, & Mata García, 2014)

3.3.5. Comunicación

- Comunicación en serie

Una comunicación en serie establece un sistema de transmisión de información que envía un bit después de otro de manera que solo un bit se transmite en un momento particular. (Cánepa, Ferrari Bihurriet, & Picard, 2014)

- Comunicación Síncrona

La información se transmite en secuencia bit tras bit con velocidad de transmisión fija, esto significa que el transmisor y el receptor están sincronizados entre ellos con la misma frecuencia de reloj. (Cánepa, Ferrari Bihurriet, & Picard, 2014)

- Comunicación Asíncrona

El transmisor y el receptor se abstienen de emitir largas secuencias de bits porque no hay una sincronización completa entre el transmisor que envía los datos y el receptor que recibe los datos. En este caso, la información se divide. en bloques, del tamaño de un byte. Cada uno de los bloques tiene un Bit de "inicio" y bit de "parada". El bit "Inicio" marca el comienzo de un nuevo bloque, el bit de "parada" marca el final. Los bloques de información no deben transmitirse necesariamente en el mismo espacio de tiempo, ya que son independientes del reloj. (Cánepa, Ferrari Bihurriet, & Picard, 2014)

3.3.6. Motores de corriente continua

Los motores eléctricos de corriente continua tienen mayor aplicación cuando se dispone de tensiones pequeñas y se busca una fuerza de desplazamiento elevada. Para su funcionamiento requieren la introducción de corriente continua en las bobinas del rotor con el fin de crear campos magnéticos que interactúen con los del estator para provocar un movimiento mecánico. (Martinez Rueda, 2006)

La fuerza electromotriz surge por el hecho de producirse una inducción en el rotor debido a la existencia de un movimiento dentro de un campo magnético esta variación del flujo magnético origina sobre los conductores del rotor una fuerza electromotriz que se opone a la tensión aplicada. (Martinez Rueda, 2006)

3.3.7. Encoder

El encoder es un dispositivo que provee pulsos eléctricos si su eje se encuentra rotando. El número de pulsos generados es proporcional a la posición angular del eje, es un dispositivo utilizado como transductor de posición. La señal de entrada del encoder es la posición angular de su eje con respecto a los ejes establecidos como referencia. Las señales de salida son dos pulsos desplazados por un cuarto de pulso. (Negrea , Szabó, & Imecs, 2010)

CAPITULO 4: PROCEDIMIENTO METODOLÓGICO

4.1 Obtención y análisis de información

El problema planteado es establecido por la Escuela de Ingeniería en Electrónica, dando como escenario un equipo que no se encuentra en uso y que no cuenta con un sistema que brinde la capacidad de dar funcionalidad al pegasus, durante una primera fase de trabajo se desarrolló, por parte de estudiantes de la escuela, la conexión de alimentación del pegasus y se habilitó un movimiento independiente de cada motor por medio de una modalidad de encendido y apagado, durante esta primera etapa se concluyeron que existen problemas de calibración en los sistemas de medición actuales con los que contaba el equipo y además, se plantea el desarrollo de un sistema que permita determinar posicionamiento y dar movilidad al equipo.

4.2 Evaluación de las alternativas y síntesis de solución

En esta sección se definen los parámetros de evaluación y las diferentes alternativas de solución que se analizaron para el desarrollo del proyecto.

4.2.1. Parámetros de Selección

Para la selección de las diferentes alternativas de solución se utilizan parámetros definidos en base a los requerimientos y desarrollo del proyecto, cada conjunto de alternativas se analizará para cada parámetro cuando así se necesite, los parámetros para los cuales no apliquen no serán tomados en cuenta y se utilizarán los de mayor relevancia con respecto a cada área para la toma de decisiones.

Dependiendo del área de trabajo donde se encuentren las alternativas y el proceso que conlleve su uso, ciertos parámetros tendrán más relevancia que otros a la hora de definir la selección, esto se modificará para cada análisis con el propósito de definir la alternativa en base a su parámetro de relevancia y su utilidad al momento de cumplir los requerimientos y objetivos.

- **Costo económico**

El parámetro de costo económico evalúa todas las alternativas que conllevan a una inversión para la adquisición tanto física como digital, todas las alternativas que sean de uso libre, que se encuentren a disposición para realizar el proyecto, o que presente un costo mínimo no

aplicarán para este parámetro de evaluación, se utilizará como primer parámetro de exclusión.

El propósito del parámetro de costo económico es reducir de manera más eficiente el presupuesto requerido para el desarrollo del proyecto, esto con el propósito de convertirlo más viable al ser exportado a otros equipos de trabajo similar.

- **Uso de recursos**

El uso de recursos aplica tanto para software como para hardware a implementar en el desarrollo del proyecto, al contar con dos equipos físicos separados como lo son el microcontrolador y la computadora, es necesario analizar el uso de recursos que se necesitan en ambos y el que requieren entre ellos para su compatibilidad, el objetivo es optar por las alternativas que brinda un brecha más amplia en el manejo del proyecto, permitiendo ejecutar el proyecto en equipos de menor capacidad o implementar más opciones para mejora es versiones posteriores

El uso de recursos es especialmente importante cuando se trabaja con comunicaciones y la apertura simultanea de dispositivos, debido a que se busca que la información se transmita y se trabaje de la manera más fluida posible.

- **Velocidad de procesamiento**

El parámetro de velocidad de procesamiento se aplicará principalmente a procesos de software y a las alternativas de hardware a implementar, para verificar si cumplen con las velocidades necesarias para realizar el trabajo. Se busca que el cálculo de los datos y el envío de la información brinde un tiempo de respuesta óptimo para realizar el control de los motores, entre mayor es la velocidad de los procesos involucrados se amplía la variación en los tiempos de control y se vuelve adaptable para diferente velocidades y frecuencias de trabajo.

- **Facilidad de implementación y desarrollo**

El parámetro de facilidad de implementación busca determinar el camino que brinde mayor compatibilidad entre todas las áreas y equipos involucrados en el desarrollo del proyecto, involucra el uso de herramientas y bibliotecas que complementen los procesos internos.

Otra característica importante es la interfaz que presentan las alternativas para su uso, se busca priorizar la facilidad de implementación y desarrollo para minimizar los problemas de conexión y los tiempos requeridos para llevar a cabo un proceso.

4.2.2. Microcontrolador

Para el microcontrolador encargado de las señales de control, se establecen ciertas características deseadas para su funcionalidad, estas se encuentran fundamentadas en los requerimientos de la Tabla 1.1.

El microcontrolador debe ser capaz de operar a una frecuencia de lectura de datos variable, y que permita analizar la información y procesarla a una tasa de tiempo aceptable debido a que los motores no cuentan con un sistema de alimentación que pueden dar voltaje en su rango máximo de operación, estos no pueden realizar un control de velocidad con el margen total de trabajo, por esta razón se debe determinar para que rango de valores binarios se puede realizar un control de velocidad y que la frecuencia de transmisión sea óptima para la mejor velocidad.

La variación en la velocidad de transmisión permite hacer al sistema menos dependiente del control de velocidad y maximizar el tiempo de ejecución del código, también brinda una brecha de trabajo en relación con el tamaño de los paquetes y la capacidad del canal.

Se requiere que el microcontrolador cuente con un presupuesto de entradas y salidas, las salidas deben ser digitales para la generación del voltaje de control. El conjunto de entradas es necesario si se utiliza un protocolo de comunicación que lo amerite, los protocolos de comunicación pueden presentar adaptadores externos que requieren el uso de puertos de entrada para la lectura de los datos.

Los motores de corriente continua son habilitados a través de módulos duales (L298N), cada módulo puede soportar dos motores y está directamente conectado en sus salidas con la fuente de alimentación. Existen tres señales de control para cada motor, dos de estas controlan la polaridad y el sentido de giro de los motores y la tercera señal se encarga de brindar la señal de salida para controlar la velocidad.

El proyecto está establecido para controlar cuatro ejes de movimiento, por lo tanto, el microcontrolador debe contar con 12 salidas digitales, es de interés que cuente con

comunicación en serie hacia el ordenador que procesa las imágenes para realizar la transmisión de datos de forma más sencilla.

Otro aspecto importante es que el microcontrolador cuente con una interfaz de programación sencilla y compatible con el protocolo de comunicación entre los dispositivos.

Se seleccionaron como alternativas de desarrollo los dispositivos arduino Mega y raspberry pi 3, dado que ambas alternativas se encuentran disponibles dentro de la Escuela de Electrónica. Para ambos se realizó una comparación de las especificaciones técnicas

Tabla 4.1 Comparacion de los parametros de microcontroladores

Dispositivo	Frecuencia	I/O	Interfaz de código	Conexión en serie	Conexión Ethernet
Arduino Mega	16MHz	54	Arduino IDLE	Si	No
Raspberry Pi	1.2GHz	40	Código libre	Si	Si

Fuente: Elaboración propia

- **Arduino Mega**

  Arduino mega es una placa basada en el microcontrolador Atmega2560 diseñado en una arquitectura ARM con instrucciones de 32 y 64 bits. Este microcontrolador de 8 bits trabaja en conjunto con una SRAM de 8KB, 4KB de EEPROM y 256KB de flash a un voltaje operativo de 5 V, el software de desarrollo es un entorno integrado basado en java, la placa cuenta con 54 pines I/O, 16 entrada analógica y 15 pueden ser utilizados como salidas PWM.

- **Raspberry Pi 3**

  Raspberry es un sistema embebido con un procesador Chipset Broadcom BCM2387 y un procesador Cortex-A53 de 1.2Ghz de cuatro núcleos basado en una arquitectura ARM, cuenta con unidad de proceso gráfico y un 1Gb de memoria RAM, proporciona 40 pines de 2.54mm 27 son pines GPIO, y un sistema operativo Raspbian para desarrollo de código libre.

Se evalúan las dos opciones de microprocesador y su cumplimiento con los parámetros establecidos con anterioridad. Usando el presupuesto de pines como requerimiento principal

de selección, ambas opciones presentan la cantidad necesaria de salida digitales para el control de todos los motores, y al estar disponibles dentro de la Escuela de Electrónica no aplican para una evaluación de costo económico.

Raspberry presenta un procesador con una mayor frecuencia de trabajo, pero ambos microcontroladores cuentan con la capacidad de ejecutar las instrucciones y la lectura de transmisión

Arduino presenta una interfaz de desarrollo de código más sencilla y accesible, con amplias fuentes de documentación y funciones que facilitan la lectura de los datos por comunicación en serie, al contar con una interfaz y lenguaje definido se establecen los tiempos de ejecución de las funciones dentro del código.

Raspberry trabaja sobre un sistema operativo Raspbian de desarrollo similar a Linux, por lo que el desempeño del código es más dependiente del lenguaje a utilizar y del hardware, esto plantea la necesidad de evaluar cual lenguaje de programación permite la transmisión de los datos de manera más sencilla y eficiente en el equipo.

Tomando en cuenta el código a desarrollar y que la diferencia en velocidades de procesamiento no va a afectar el desempeño de ejecución se define como primera alternativa de implementación la placa de arduino mega por la interfaz de desarrollo de código, la facilidad de implementación y el cumplimiento de los requerimientos.

Planificando la implementación de un protocolo de comunicación síncrono, arduino mega presenta la capacidad de contar con tres puertos de comunicación en serie previamente establecidos para facilitar la implementación de módulos de conversión USB a TTL, y permitir un sistema de comunicación simplex doble.

4.2.3. Lenguaje de programación

El proyecto está enfocado en el diseño de un algoritmo específicamente, el lenguaje de programación es la base primordial para el desarrollo del proyecto, el lenguaje debe cumplir con los parámetros de evaluación y con los requisitos de comunicación, debido a que el procesamiento se da a través de librerías computacionales y el lenguaje debe contar con soporte de estas librerías.

El código se encarga de aplicar las morfologías y el filtrado de los colores por lo que se necesita el menor consumo de hardware posible, dado que el ciclo de detección es constante y conlleva el uso de hilos para realizar protocolos de comunicación.

Otro factor de interés e importancia del lenguaje de programación es la facilidad en el desarrollo de código, considerando el uso de diferentes esquemas de computación visual, bibliotecas matemáticas para el cálculo de los datos, trasiego de datos con el microcontrolador y el uso de hilos se busca que el lenguaje presente una interfaz sencilla que permita incorporar todas las áreas del algoritmo de la mejor manera ayudando a la reducción de ciclos y tiempo de ejecución.

- Python

  Python es lenguaje de programación multiparadigma que trabaja como un lenguaje interpretado, tiene la capacidad de utilizar tipado dinámico y es multiplataforma. Presenta ventajas en proceso de escritura ya que hace hincapié en una sintaxis que favorezca el código legible, tiene soporte para OpenCV, librerías matemáticas para cálculos algebraicos y trabajo de matrices, cuenta con librerías para facilitar la comunicación en serie.

- Matlab

  Matlab es una herramienta de software matemático con entorno de desarrollo integrado (IDLE) que cuenta con su propio lenguaje de programación, presenta la capacidad de trabajar las imágenes como matrices multidimensionales y aplicar algoritmos de algebra lineal para realizar la ejecución del análisis con mayor rapidez. Actualmente presenta soporte para OpenCV, pero no para otras librerías como Tensorflow para las que requiere realizar una interfaz gráfica de usuario y transferir las entradas a un código de Python o C para realizar la ejecución.

- C

  C es un lenguaje de bajo nivel que está pensado para la creación de software de sistemas, aunque también se utiliza para el desarrollo y la creación de aplicaciones, al ser un lenguaje tan cercano al procesador presenta la ventaja de contar con tiempos de ejecución menores y estar débilmente tipificado además de contar con estructuras típicas de alto nivel.

  C no posee un intérprete o entorno de desarrollo que facilite el desarrollo de código y la, además de carecer de documentación con respecto a otros lenguajes de alto nivel por su complejidad de sintaxis.

Utilizando el costo económico de las alternativas como primer parámetro de exclusión se determinó que la herramienta de software Matlab es la única que no cumple con el parámetro de selección y adyacente a esto es la alternativa que requiere mayor capacidad de hardware para lograr su ejecución.

Las restantes dos alternativas no aplican para la evaluación del parámetro anterior por ser lenguajes de desarrollo libre y de baja utilización de recursos.

Tanto Python como C son capaces de cumplir los requerimientos de desarrollo del proyecto, la mayor diferencia entre los dos lenguajes es el nivel de dificultad y documentación que presentan para facilitar la implementación del código, Python cuenta con una sintaxis de desarrollo mucho más sencilla y una mayor cantidad de documentación establecida en todo tipo de áreas.

A pesar de ser un lenguaje de alto nivel para bibliotecas computacionales como OpenCV el código de OpenCV para Python se encuentra envuelto alrededor del código original de C, por lo tanto, al llamar una función de OpenCV desde Python se ejecuta adyacentemente el código de C los que presenta una diferencia en el desempeño entre estos dos lenguajes mínima.

4.2.4. Herramienta de visión por computadora

La herramienta de visión computacional establece el camino y conjunto de acciones a tomar para realizar el análisis de las imágenes y el cálculo de los datos, el primer requisito debe ser

que cuente con soporte en alguno de las alternativas de lenguaje para poder realizar su implementación.

Actualmente las dos herramientas o métodos más utilizados para el análisis de imágenes son el método directo como una matriz de pixeles y las redes de aprendizaje automático; la biblioteca requiere contar con funciones para filtrado y seguimiento que permitan detectar secciones específicas de una imágenes de manera continua mediante un entrada de video, sin hacer un uso extenso de recursos ya que requiere realizar el procesos desde dos entradas para puntos de seguimiento múltiples y distintos entre ellos.

Para las herramientas de visión se tomaron dos alternativas de bibliotecas, ambas de uso gratuito y con soporte en todas las alternativas de lenguaje.

- OpenCV

  Es una librería de visión por computadora con clases y métodos prediseñados para realizar análisis de imagen y video en tiempo real, posee análisis por selección y la capacidad de desarrollar aprendizaje automático, es una biblioteca multiplataforma eficiente en la variedad de funciones y procesos y diseñada para ser de fácil implementación a nivel de código, posee documentación propia e independiente de todas sus funciones.

- Tensorflow

  Tensorflow es una biblioteca de visión que es utilizada generalmente para procesos de aprendizaje automático y algoritmos de análisis profundo, es multiplataforma para lenguaje de uso libre y trabaja bajo un sistema de entrenamiento de red, utiliza un proceso de enseñanza bajo un conjunto de imágenes para mejorar la exactitud de detección.

Entre los diferentes parámetros de selección aplicables a las alternativas solo se seleccionaron la facilidad de implementación y el uso de recursos, para ambos la alternativa de tensorflow representa la opción más compleja, considerando que la biblioteca no requiere de muchos recursos para su ejecución pero si para el proceso de entrenamiento de red, el tiempo de este proceso depende de la cantidad de muestras y los recursos que se encuentren disponibles para hacerlo, por lo que generalmente se implementan módulos de GPU, este mismo proceso

representa una desventaja en términos de implementación, porque debe realizarse el proceso para todos los puntos a seleccionar en los dos ángulos de visión.

OpenCV establece una alternativa más optima por su facilidad de instalación y desarrollo y amplia cantidad de función que permiten formular un código escalable a mayor cantidad de punto de una manera más sencilla, el sistema de apertura de entrada le permite obtener imagen de manera múltiple en tiempo real y posee compatibilidad de desarrollo en los diferentes lenguajes, por lo que cumple con los requerimientos de desarrollo.

4.2.5. Equipo de visión

Para el equipo de visión se optó por realizar la evaluación de dos alternativas distintas entre sí, ya que se contempló la posibilidad de obtener una parte del análisis computacional a partir de un hardware y reducir el trabajo realizado en software. A partir de lo mencionado se contemplaron las alternativas de entradas mediante cámara web y la incorporación de un sensor de visión.

El objetivo del equipo de visión es realizar la captura de la imagen como entrada para su posterior evaluación y detección de puntos. Una característica importante es la percepción de la imagen, la detección de un color se da mediante un modelo RGB, el equipo de visión requiere la capacidad de mantener la mayor estabilidad del color ante exposiciones a la luz, y requerir la mínima cantidad de recursos para su operación.

- Cámara web

  Se definió un dispositivo digital con conexión USB para realizar la captura del video, dentro de las características se definen el tamaño de la matriz de pixeles, la velocidad de toma de los fotogramas, la compatibilidad con los sistemas operativos, la estabilización de luz y la compatibilidad de USB 2.0 y 3.0.

- Sensor de visión

  Es un dispositivo integrado para la detección y evaluación de objetos y escenas de manera independiente, posee la capacidad de comparar contornos, contar pixeles, registrar dimensiones y determinar ángulos de inclinación, cuenta con integración de luz segmentada y censado 3D.

Como los puntos de toma de imágenes son laterales y superiores se requieren dos unidades indiferentemente de la alternativa, esto hace que el parámetro determinante de exclusión sea el costo económico con respecto a sensor de visión, dado que presenta un costo superior con respecto a la primera alternativa.

La cámara web debe ser evaluada para su selección priorizando su compatibilidad con los puertos de conexión debido a que los puertos USB cuenta con un ancho de banda limitado para el procesamiento de datos, sin embargo, los puerto USB 2.0 y 3.0 cuentan con anchos de banda independientes por esto se requiere que una cámara y el microcontrolador se encuentren conectados a un ancho de banda distinto al de la segundo cámara, para permitir el trasiego de datos sin saturar los puertos de conexión.

4.2.6. Protocolo de comunicación

El protocolo de comunicación establece el tipo de comunicación a utilizar para enviar los datos obtenidos a través del análisis mediante software hasta el microcontrolador para realizar el control de los motores de cada eje; con respecto a los requerimiento de implementación y los parámetros de selección, el protocolo solo se evaluó sobre la velocidad de ejecución, y la facilidad de implementación, ya que no aplicaban para los demás parámetros y ambas alternativas cumplen con los requerimiento de desarrollo.

Debido al sistema de alimentación actual de los motores que controlan el desplazamiento de las articulaciones, no se puede definir de manera concreta la capacidad de realizar un control de velocidad a través de los puentes H, las articulaciones pueden encontrarse en una posición en la que presenten un mayor peso para el motor en un sentido de giro y un peso menor en el sentido contrario imposibilitando utilizar la misma velocidad en ambas direcciones.

El protocolo de comunicación debe estar defino por un canal que permita realizar él envió de los datos a la velocidad máxima de giro de los motores.

Debido a la limitante de puerto de conexión tanto en el equipo para procesar las imágenes como en el microcontrolador, se plantearon dos protocolos de fácil acceso para implementar el proyecto.

- Comunicación WiFi

  La comunicación por red wifi es una interconexión inalámbrica entre dispositivos mediante un punto de acceso a internet, la comunicación está establecida por un protocolo de empaquetado formado por un nivel de capas, la velocidad de envío es dependiente de la velocidad de conexión del proveedor de servicio a internet y el tamaño del paquete a transmitir.

- Comunicación en serie

  El modelo de comunicación en serie o por puerto en serie es el proceso que permite enviar datos un byte a la vez a través de un sistema encadenado donde cada byte se lee consecutivamente uno después del otro hasta completar el mensaje, posee la característica de ser de frecuencia variable ya que el mensaje no se envía como un paquete completo, generalmente la comunicación se da por conexión USB entre la computadora y el microcontrolador.

Como parte del proceso de diseño, se necesita una selección de alternativas que no conlleve a un conflicto entre ellas mismas a la hora de realizar la implementación, de las alternativas anteriores ambas cumplen con los requerimientos de desarrollo y son viables, sin embargo, implementar una comunicación en serie por puerto USB conlleva un aumento en el uso del ancho de banda de este y utilizar un protocolo wifi significa utilizar un módulo de wifi para el microcontrolador lo que implica reducir la cantidad de pines disponibles para la escritura de señales.

Debido a que el ancho de banda de USB es algo que se puede trabajar reduciendo el uso de recursos en los puertos y que se cuenta con ancho de banda independientes, se definió como alternativa de comunicación el protocolo en serie ya que los pines del microcontrolador son limitados y un módulo extensor de pines es algo de lo que no se dispone.

4.3. Implementación de la solución

En esta sección se definirá el procedimiento seguido para llevar a cabo la implementación de la solución. Al ser una implementación de software se presenta la instalación de bibliotecas

y paquetes requeridos para llevar a cabo el desarrollo del código, así como la conexión física de señales.

4.3.1. Implementación de Software

Para la implementación del software necesario se requiere realizar la instalación de la biblioteca de computación visual a utilizar y el lenguaje de programación, para este proceso se definen los pasos de implementación de las alternativas de soluciones.

Los pasos de instalación para la biblioteca de OpenCV y el lenguaje operativo de Python se presentarán para Linux el cual es el sistema operativo sobre el que se desarrolló el proceso, la instalación de la biblioteca.

Para realizar la instalación de la biblioteca de visión se proceda a la ejecución de los siguientes comandos en la ventada de comandos del equipo.

Primero se procede a realizar una actualización del sistema

- *sudo apt-get update*
- *sudo apt-get upgrade*

Se realiza la instalación de las dependencias necesarias para la ejecución de la biblioteca.

- *sudo apt-get -y install build-essential cmake cmake-qt-gui pkg-config libpng12-0 libpng12-dev libpng++-dev libpng3 libpnglite-dev zlib1g-dbg zlib1g zlib1g-dev pngtools libtiff4-dev libtiff4 libtiffxx0c2 libtiff-tools*
- *sudo apt-get -y install libjpeg8 libjpeg8-dev libjpeg8-dbg libjpeg-progs ffmpeg libavcodec-dev libavcodec53 libavformat53 libavformat-dev libgstreamer0.10-0-dbg libgstreamer0.10-0 libgstreamer0.10-dev libxine1-ffmpeg libxine-dev libxine1-bin libunicap2 libunicap2-dev libdc1394-22-dev libdc1394-22 libdc1394-utils swig libv4l-0 libv4l-dev python-numpy libpython2.6 python-dev python2.6-dev libgtk2.0-dev pkg-config*

Para realizar la descarga y descompilado de la librería se puede realizar de manera manual mediante la dirección oficial o a través de la ventana de comando.

- *Wget http://sourceforge.net/projects/opencvlibrary/files/opencv-unix/2.4.9/opencv-2.4.9.zip/download -O opencv.zip*

- *unzip opencv.zip*
- *rm opencv.zip*
- *cd OpenCV-2.4.9*

Se recomienda revisar la versión para posteriores actualizaciones.

Se crea un directorio para almacenar los ficheros compilados y se accede a la dirección

- *mkdir build*
- *cd build*

Se utiliza la herramienta multiplataforma Cmake para la instalación de los módulos requeridos, los módulos dependen de las necesidades de cada proyecto, una configuración estándar y versátil se establece con el comando

- *cmake -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/usr/local -D WITH_TBB=ON -D BUILD_NEW_PYTHON_SUPPORT=ON -D WITH_V4L=ON -D INSTALL_C_EXAMPLES=ON -D INSTALL_PYTHON_EXAMPLES=ON -D BUILD_EXAMPLES=ON -D WITH_QT=ON -D WITH_OPENGL=ON ..*

Una vez establecidos los modulos de manera correcta se procede a compilar la librería

- *make*
- *sudo make install*

Una vez instalada la biblioteca, se requiere de configuraciones para su correcto funcionamiento y se accede al archivo opencv.conf mediante el editor de texto predeterminado nano

- *sudo nano /etc/ld.so.conf.d/opencv.conf*

Al final del archivo de adjunta la siguiente línea de código, para dar dirección

- */usr/local/lib*

A continuación, se edita el archivo bashrc.

- *sudo nano /etc/bash.bashrc*

Y por último se añade la siguiente línea de código al final del archivo

- *PKG_CONFIG_PATH=$PKG_CONFIG_PATH:/usr/local/lib/pkgconfig export PKG_CONFIG_PATH*

Para la implementación del lenguaje de programación se establecen los pasos de la versión más estable y actual disponible. La instalación se puede realizar de manera manual mediante el sitio web oficial, de igual manera se establecen los pasos a realizar en la ventana de comandos.

Se realiza una instalación de los prerrequisitos para Python

- *sudo apt-get install build-essential checkinstall*
- *sudo apt-get install libreadline-gplv2-dev libncuesesw5-dev libssl-dev libsqlite3-dev tk-dev libgbm-dev libc6-dev libbz2-dev*

Se realiza la descarga del sitio oficial y se extrae el paquete

- *cd /usr/src*
- *sudo wget https://www.python.org/ftp/python/3.6.6/python-3.6.6.tgz*
- *sudo tar xzf Python-3.6.6.tgz*

Se compila el código fuente de Python

- *cd python-3.6.6*
- *sudo ./configure – enable-optimizations*
- *sudo make altinstall*

Para la utilización de librerías de comunicación en serie como PySerial la instalación se puede realizar a través de pip una aplicación para la instalación de complementos de manera más rápida.

Para el uso de herramientas matemáticas y de manejo de matrices se utiliza el paquete de funciones Spicy

- *sudo apt-get install python-numpy python-scipy python-matplotlib ipython ipython-notebook python-pandas python-sympy python-nose*

Para la instalación de IDE de arduino solo es necesario realizar la descarga de la plataforma oficial, extraer el paquete y ejecutar el archivo *install.sh*

4.3.2. Implementación de Hardware

Para la implementación de la sección de hardware se definen la conexión de las señales de control correspondientes de los motores y la conexión de los puertos de comunicación.

El puente H L298N cuente con capacidad de realizar conexión a dos motores de manera simultánea.

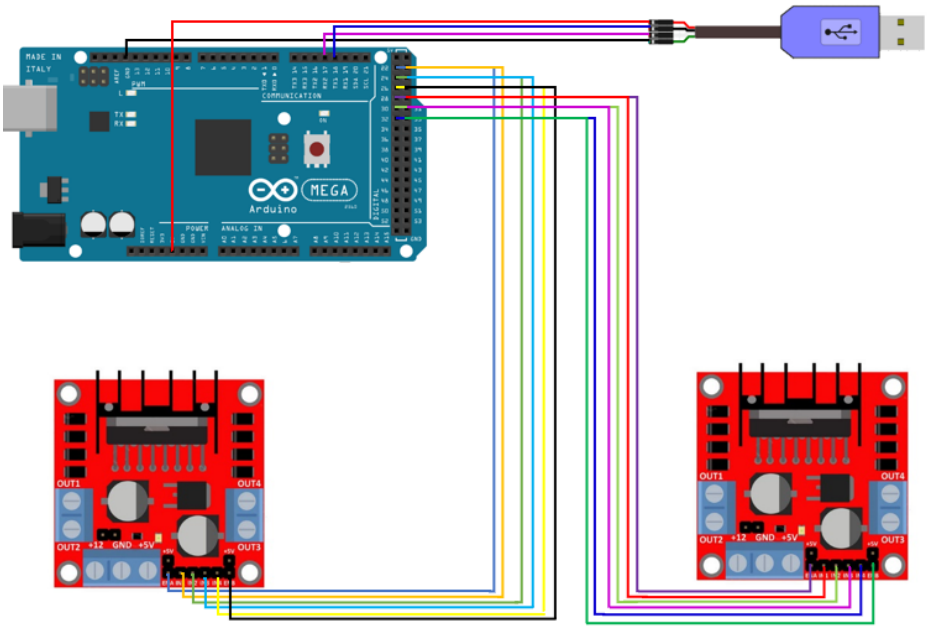


Figura 4.1. Diagrama de conexión de los elementos de hardware
Fuente: Elaboración propia

El módulo de conexión USB está conectado a la unidad de procesamiento de imágenes para establecer la conexión de escritura desde el arduino, adicionalmente el puerto COM de programación del arduino está conectado mediante USB a un puerto distinto para realizar la lectura de los datos retroalimentados mediante las cámaras.

Ambas cámaras, tanto la lateral como la superior, conllevan conexión USB a la unidad de procesamiento de imágenes para trabajar como unidades de entrada.

4.3.3. Mecanismo de obtención de información

Los mecanismos de obtención de la información se dividen en tomas de muestras físicas y digitales.

Dentro de los métodos de obtención de información se realizó una toma digital de información en la cual se hicieron diferentes mediciones, se realizó primeramente un análisis de morfologías, tomando diferentes morfologías para las muestras de color a diferentes distancias esto con el propósito de determinar cuál morfología era la adecuada para aplicar al filtro y obtener el mejor resultado para implementar en la solución, este proceso también permitió determinar parámetros utilizados en las funciones morfológicas que aumentaran la efectividad de detección.

Adicionalmente, se tomaron muestras de contornos para ejemplificar como la detección y determinación de contornos varía a diferentes distancias de la cámara y como se ve afectada cuando se encuentra expuesto a un cambio en la profundidad.

Se tomó un conjunto de mediciones físicas y digitales en las que se capturó el ángulo detectado por la cámara y se tomó un grupo de mediciones físicas del mismo ángulo, esto para tomar el porcentaje de error que presenta la imagen en el cálculo de los datos, este proceso se hizo para nueve posiciones con diferentes diez datos por posición, en las que se cambió la distancia a la que se encontraba el brazo de la cámara.

Se tomaron y se graficaron las variaciones de medición que presenta la cámara cuando se presenta una rotación en la base en sentido horario y antihorario para determinar los cambios en la medición que presenta las diferentes profundidades.

4.4. Reevaluación y rediseño

En esta sección se definen los métodos a aplicar para brindar mejoras y avances futuros a la solución, las alternativas se enfatizan en optimizar la ejecución del código.

4.4.1. Alternativas de mejora

- **Computación Paralelo**

  La computación paralela es un modelo que permite la ejecución de diferentes procesos de manera simultánea. Dentro de las clases de paralelismo se define para mejora del proyecto la computación multinúcleo, esta clase permite emitir múltiples instrucciones por ciclo de reloj desde múltiples flujos de instrucciones, esto para lograr distribuir las instrucciones de ejecución en diferente núcleos de procesamiento para separar el uso de recursos del hardware, adicionalmente a esto se pueden

implementar otros tipos de paralelismo, como el paralelismo de tarea y paralelismo a nivel de instrucción para definir de forma más concreta las instrucciones por ciclo de reloj y los grupos de información y memoria sobre los que se realizan la operaciones esto permite hacer una distribución más controlada del uso de recursos y la ejecución del código y aumenta la brecha de trabajo disponible en lo que respecta a hardware permitiendo incluir procesos más robustos.

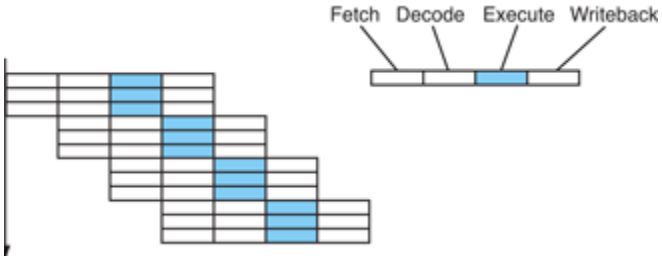


Figura 4.2 Paralelismo de instrucción
Fuente: *Quizlet*. (1 de Noviembre de 2018). Obtenido de https://quizlet.com/31893528/chapter-16-instruction-level-parallelism-and-superscalar-processors-flash-cards/

- **Controlador PID**

  El mecanismo de control simultáneo consiste en un algoritmo de parámetros proporcional, integral y derivativo, que permite hacer un análisis entre los errores pasados presentes y la predicción de los futuros para hacer una estabilización constante del valor deseado, el controlador PID se puede establecer de forma más eficiente si se restructura el sistema de alimentación actual de los motores para que alcancen el rango de voltaje máximo al que pueden trabajar y contar una variación en las velocidades de ejecución de las extremidades más amplia que la que poseen actualmente.

  La implementación de un sistema PID establece el muestreo de las señales de error, el modelado gráfico de la respuesta del sistema ante perturbaciones y la determinación y ajuste de las constantes para la aproximación del error en estado estacionario a cero.

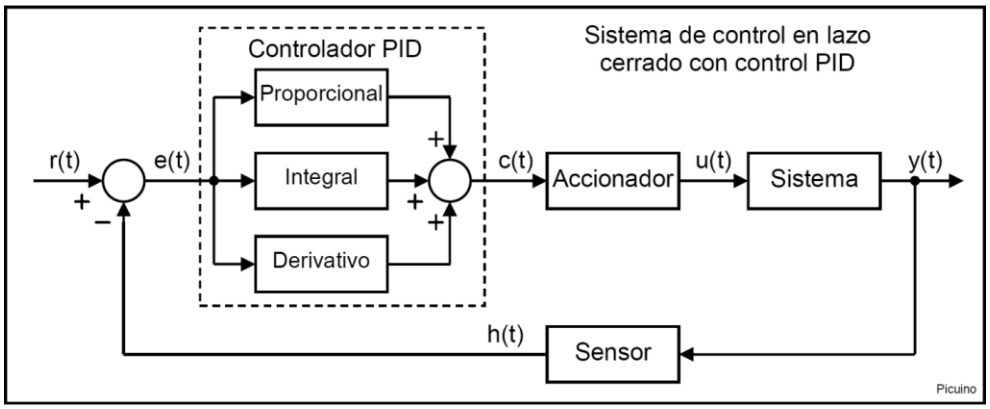


Figura 4.3. Diagrama de bloques de un controlador PID en lazo realimentado

Fuente: *Pincuico*. (1 de Diciembre de 2018). Obtenido de https://www.picuino.com/es/arduprog/control-pid.html

- **Programación GPU**

La implementación de programación de alto desempeño a través de módulos GPU permite paralelizar la computación hacia los núcleos de procesamiento de la tarjeta de video, esto permite hacer uso de un alto número de hilos de ejecución de manera simultánea que realizan tareas independientes, así como dedicar el procesamiento gráfico de las capturas y el despliegue a la unidad GPU, la memoria compartida se puede dividir entre los hilos y dependiendo del tamaño disponible puede utilizarse como un caché para agilizar el acceso a los datos, por medio de un patrón adaptador se dispone de adaptadores de interface para determinados lenguajes como PyCUDA para desarrollar código de Python utilizando tarjeta de video nVidea. Debe tomarse en cuenta los anchos de banda de los buses y sus latencias para no sufrir afectaciones por cuellos de botella al implementar este proceso.

CAPITULO 5: DESCRIPCIÓN DETALLADA DE LA SOLUCIÓN

5.1. Descripción del hardware

En esta sección se presentará la estructura, funcionalidad y descripción técnica del hardware utilizado para la implementación del proyecto, se dividirá en los tres componentes primordiales para la toma de información y control del equipo

5.1.1. Arduino Mega

El arduino mega está diseñado sobre el microcontrolador ATmega2560, para la implementación del proyecto se estableció como el dispositivo encargado de las señales de control para el sentido de giro y la velocidad de los motores. Las especificaciones técnicas relevantes para su selección se definen en la siguiente tabla.

Tabla 5.1. Parámetros internos del arduino mega

Parámetro	Componente o valor
Microcontrolador	ATmega2560
Voltaje de Operación	5 V
Voltaje de entrada recomendado	7-12 V
Voltajes limite	6-20
Pines digitales de entrada y salida	54
Pines de entrada analógica	16
Corriente CD por pin	40 mA
Memoria Flash	256 KB
SRAM	8 KB
EEPROM	4 KB
Velocidad de reloj	16 MHz

Fuente: Arduino. (20 de Octubre de 2018). Obtenido de https://arduino.cl/arduino-mega-2560/

El microprocesador ATmega2560 trabaja a una velocidad de reloj superior a los demás modelos de arduino y la cantidad de pines digitales disponibles presentan los dos motivos principales de selección, adicionalmente a esto, el ATmega 2560 cuenta con los conjuntos de

memoria y pines de tipo receptor transmisor síncrono asíncrono universal presentes en la tabla 5.2

Tabla 5.2.Comparativa de los microcontroladores atmega

Dispositivo	Flash	EEPROM	RAM	Serial USART
ATmega640	64 KB	4 KB	8 KB	4
ATmega1280	128 KB	4 KB	8 KB	4
ATmega1281	128 KB	4 KB	8 KB	2
ATmega2560	256 KB	4 KB	8 KB	4
ATmega2561	256 KB	4 KB	8 KB	2

Fuente: Atmel. (11 de Noviembre de 2018). *Microchip.* Obtenido de http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-2549-8-bit-AVR-Microcontroller-ATmega640-1280-1281-2560-2561_datasheet.pdf

Las funciones de ejecución del arduino correspondientes al hardware se dividen en dos secciones específicas.

- Comunicación en serie

  El arduino mega cuenta con dos sistemas de comunicación en serie, un canal se encarga de hacer la lectura de los datos provenientes de la unidad de procesamiento de imágenes, en este caso los datos corresponden a los valores numéricos deseados por el usuario a los que se desplazara el equipo y los valores de retroalimentación que cambian constantemente según las muestras que toma la cámara, esta información se envía a través del canal cada 50 ms y este canal corresponde al puerto COM0 de alimentación del arduino que se conecta de manera física con un puerto USB de la unidad de procesamiento de imágenes.

  El segundo canal es un canal de comunicación en serie que se encuentra establecido a través de los pines de transmisión y recepción, el arduino mega cuenta con tres pares de pines para comunicación en serie definidos como tx1 rx1, tx2 rx2 y tx3 rx3 de los cuales se utilizaron el primer grupo, la conexión se realizó hacia un segundo puerto USB de la unidad de procesamiento de imágenes mediante un adaptados de TTL a USB, el adaptador requiere de una conexión cruzada entre los pines tx y rx del arduino por lo que el pin rx del adaptador se conecta al pin tx1 del arduino y el tx al rx1,

adicionalmente el adaptador requiere conexión de alimentación a 5 V y conexión de tierra con hacia el arduino.

- Señales de control

  El arduino asigna 12 señales de control establecidas como salidas digitales, del pin 22 hasta el pin 33, de las 12 señales cuatro corresponden a controles de velocidad y ocho de ellas a señales de estado encargadas de establecer en alto o bajo los pines encargados del sentido de giro de los motores.

  Cuando el arduino procesa la información obtenida a través de la lectura de las cámaras contra la información de la posición que el usuario solicita, se determina la salida digital que debe establecer en estado de alto voltaje y la que debe establecerse en bajo, con el establecimiento de las salidas se designa una escritura analógica correspondiente al valor que define la velocidad de giro del motor, este valor varía entre 0 y 255, sin embargo, se requiere un valor mínimo de 128 para realizar la activación del motor, a partir de ese valor se ajusta la velocidad. De las cuatro señales de control de velocidad solo una trabaja con el valor mínimo de 128 debido a problemas mecánicos presentes en la cadena, las demás trabajan en rangos superiores a 250.

5.1.2. Modulo Controlador L298N

El módulo controlador consiste en dos configuraciones H establecidas por cuatro transistores que permiten controlar la velocidad y la dirección de los motores en corriente continua, el rango de operación del módulo va desde 3 V hasta 35 V y una corriente de 2 A, se debe tomar en cuenta que el módulo requiere un voltaje de operación de 3 V, por lo que, los motores reciben 3 V menos del valor establecido por la fuente de alimentación. En la siguiente figura se presenta el modelo de controlador utilizado para la solución.

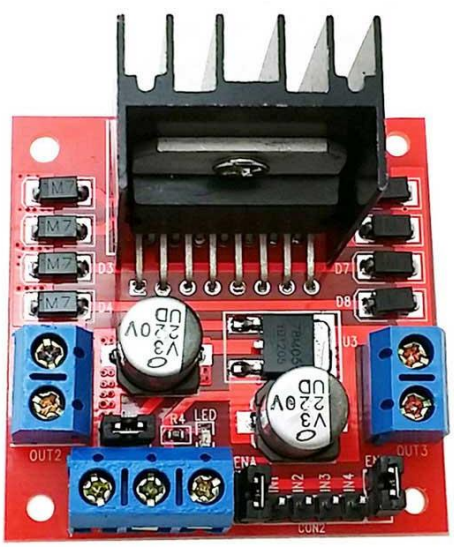
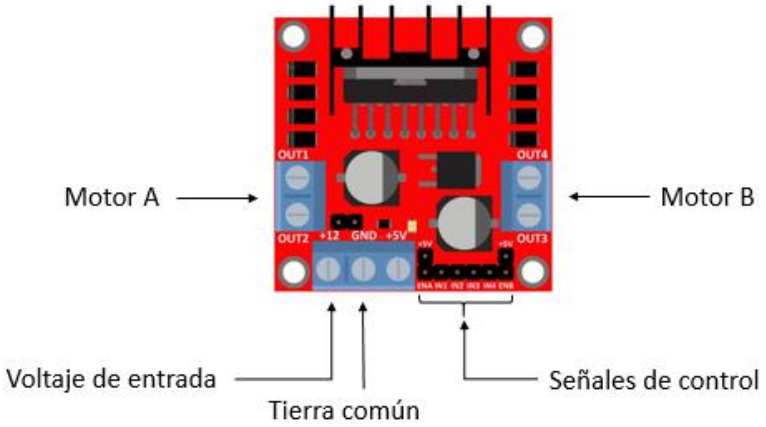


Figura 5.1.Puente H L298N
Fuente: Nylampmechatronics. (14 de noviembre de 2018). Obtenido de https://naylampmechatronics.com/drivers/11-driver-puente-h-l298n.html

El módulo cuenta con cuatro conexiones, las conexiones laterales de salida hacia los motores, la señales de control provenientes del arduino y las señales de control, este módulo cuenta con seis señales de control, tres para cada motor, la primera representa el valor de la velocidad del motor, las otras dos reciben un valor de 5 V o 0 V dependiendo de cuál sea el sentido de giro que se desee brindar al motor, la última conexión del módulo es el sistema de alimentación correspondiente de los motores, este sistema de alimentación se conecta al módulo controlador y a su vez su terminal de tierra se conecta con la tierra respectiva del arduino, en la siguiente figura se presenta la conexión de las terminales.

Figura 5.2. Parámetros de conexión del L298N



Fuente: Elaboración propia

Las señales de control están catalogadas de izquierda a derecha como ENA, IN1, IN2, IN3, IN4, ENB, de estas las primeras tres corresponden al motor A y las restantes al motor B.

Es importante tomar en cuenta que los pines ENA y ENB encargados del control de la velocidad se encuentran preestablecidos para una conexión de 5V proporcionados por el módulo, esto habilita los motores a la máxima velocidad de trabajo, sin embargo, si se desea definir este valor de velocidad se debe remover el conector y trabajar el pin como una señal de control desde el arduino.

En la siguiente tabla se presentan los rangos máximos de operación y en la Figura 5.3 el diagrama de bloques del módulo.

Tabla 5.3. Parámetros de operación del L298N

Símbolo	Parámetro	Valor	Unidad
Vs	Fuente de alimentación	50	V
Vss	Voltaje lógico de alimentación	7	V
Vi, Ven	Voltaje de entrada y habilitación	-0.3 a 7	V
Io	Pico de corriente en la salida (Operando en corriente continua)	2	A
Vsens	Voltaje de detección	-1 a 2.3	V
Ptot	Disipación de potencia total	25	W
Top	Temperatura de operación de Unión	-25 a 130	^{0}C

Fuente: Sparfunk Electronics. (15 de Noviembre de 2018). Obtenido de https://www.sparkfun.com/datasheets/Robotics/L298_H_Bridge.pdf

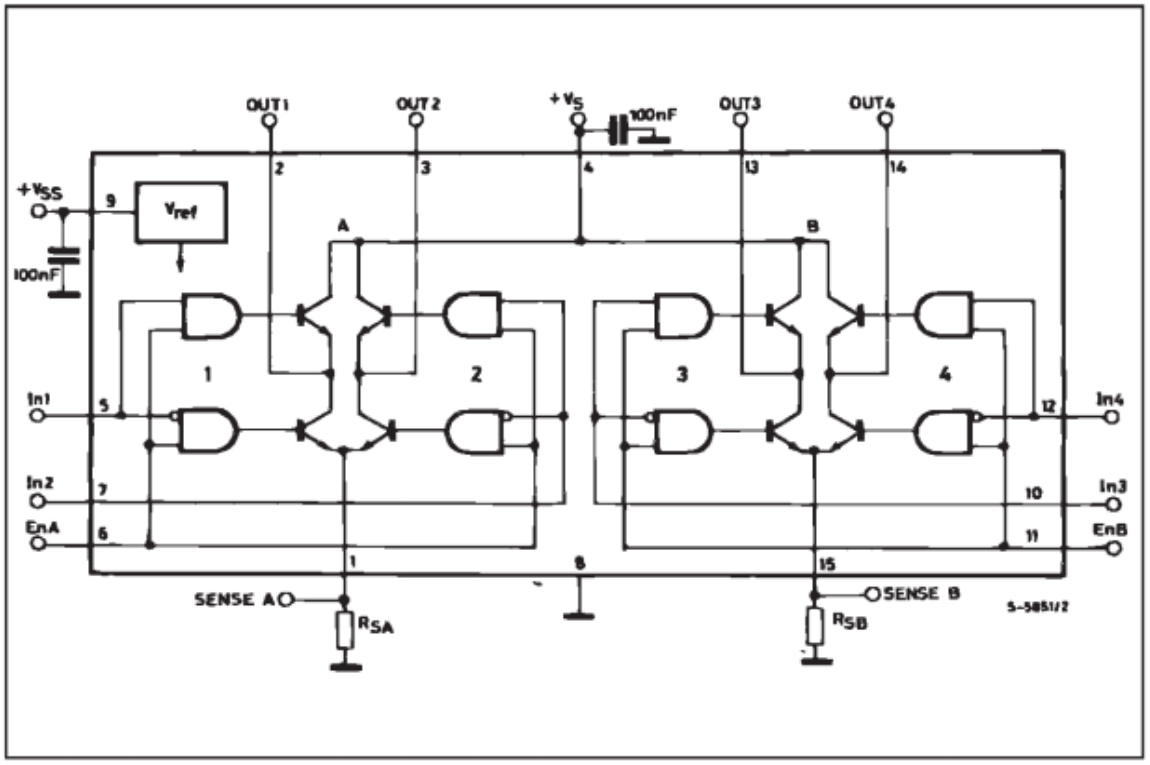


Figura 5.3. Diagrama de bloques del L298N
Fuente: Sparfunk Electronics. (15 de Noviembre de 2018). Obtenido de https://www.sparkfun.com/datasheets/Robotics/L298_H_Bridge.pdf

5.1.3 Unidad de procesamiento de imágenes

Debido a que la unidad de procesamiento de imágenes es una computadora, en esta sección se describirán las especificaciones técnicas con las que se contó para desarrollar el proyecto, adicionalmente se definirán las características técnicas de las cámaras con las que se hizo la captura de las imágenes.

En la siguiente tabla se muestran las especificaciones técnicas de la unidad de procesamiento de imágenes

Tabla 5.4.Parametros de la unidad de procesamiento de imágenes

Parámetro	Cantidad o valor
Velocidad base del reloj	2.5 GHz
Máximo incremento de velocidad de reloj	3.1 GHz
Resolución	1920 x 1080
Puertos	2x USB 3.0 Generación 1 Tipo-A 2x USB 2.0 Tipo-A
GPU	2 GB GDDR3 VRAM

Fuente: Elaboración propia

La unidad de procesamiento de imágenes requiere de cuatro conexiones USB, dos provenientes de las cámaras que funcionan como dispositivos de entrada para la toma de imágenes y dos más para establecer la comunicación con el microcontrolador, de estos últimos un puerto se encarga de realizar la escritura hacia el arduino y otro de realizar la lectura desde el mismo.

Es importante recalcar la disponibilidad de puertos USB 2.0 y 3.0, los puertos USB cuentan con una tasa de transmisión, por lo que pueden presentar saturación de datos si se excede la capacidad del canal disponible. Para el puerto USB 2.0 la tasa practica máxima es de 35 Mb/s, mientras que para el USB 3.0 es de 600 Mb/s, ambos anchos de banda son independientes uno del otro por lo que contar con dispositivos de conexión en 3.0 implica que no se utiliza el ancho de banda disponible en el 2.0, esto es importante, dado que, al procesar imágenes y trabajar con una resolución mayor implica que es necesario procesar más información a través del puerto y esto presenta una reducción en el tiempo de ejecución

del algoritmo, debido a que los cuatro puertos USB se mantienen en uso durante la ejecución del código y que la información debe transmitirse cada 50 ms se estableció una resolución de (426x240) pixeles para obtener la mayor fluidez posible entre cada toma de imágenes.

Los dispositivos de entrada son cámaras web modelo Logitech c920 que se presentan en la siguiente figura

Figura 5.4. Logitech c920
Fuente: Logitech. (10 de Noviembre de 2018). Obtenido de https://www.logitech.com/es-es/product/hd-pro-webcam-c920

Se utilizaron dos para realizar la captura de las marcas de color colocadas en el equipo, una en una base estática a un costado del equipo desde una posición frontal a las tres marcas laterales, la segunda cámara se colocó en la parte superior de forma estática y se encarga de capturar dos marcas de color, estas dos últimas marcas están distribuidas para hacer el cálculo de posición y rotación de la base, una de estas marcas se encuentra en una saliente a un costado de la base del brazo, esta saliente rota con la base del brazo, la segunda se coloca como un punto guía mediante una pieza móvil que se ubica en una sección de la base, que se desplaza con el brazo pero que no rota con este, esto permite determinar el centro del brazo desde la cámara superior con un análisis de matrices, contar con un punto de referencia que no rota y un punto de rotación para determinar el ángulo de giro de la base, la determinación de la posición en la matriz y el análisis se define en el desarrollo del algoritmo.

En la siguiente tabla de se definen las especificaciones de las cámaras utilizadas para implementar la solución

Tabla 5.5. Características de la logitech c920

Parámetro	Tipo de Conexión o valor
Tipo de conexión	USB
Protocolo USB	USB 2.0 compatible con 3.0
Tipo de sensor	Cristal
Resolución óptica	3 MP
Campo visual diagonal	78º
Longitud focal	3.67 mm
Frecuencia de recuadro máxima	1080p a 30fps

Fuente: Elaboración propia

5.2. Descripción del software

En esta sección se describe el desarrollo del algoritmo y el proceso de este para llevar a cabo la implementación de la solución, la descripción del algoritmo se separa en módulos, cada uno correspondiente a una función específica, de igual manera se describirá el código implementado en el arduino para realizar la activación de los motores.

A continuación, se presenta un diagrama de flujo del código de la unidad de procesamiento de imágenes.

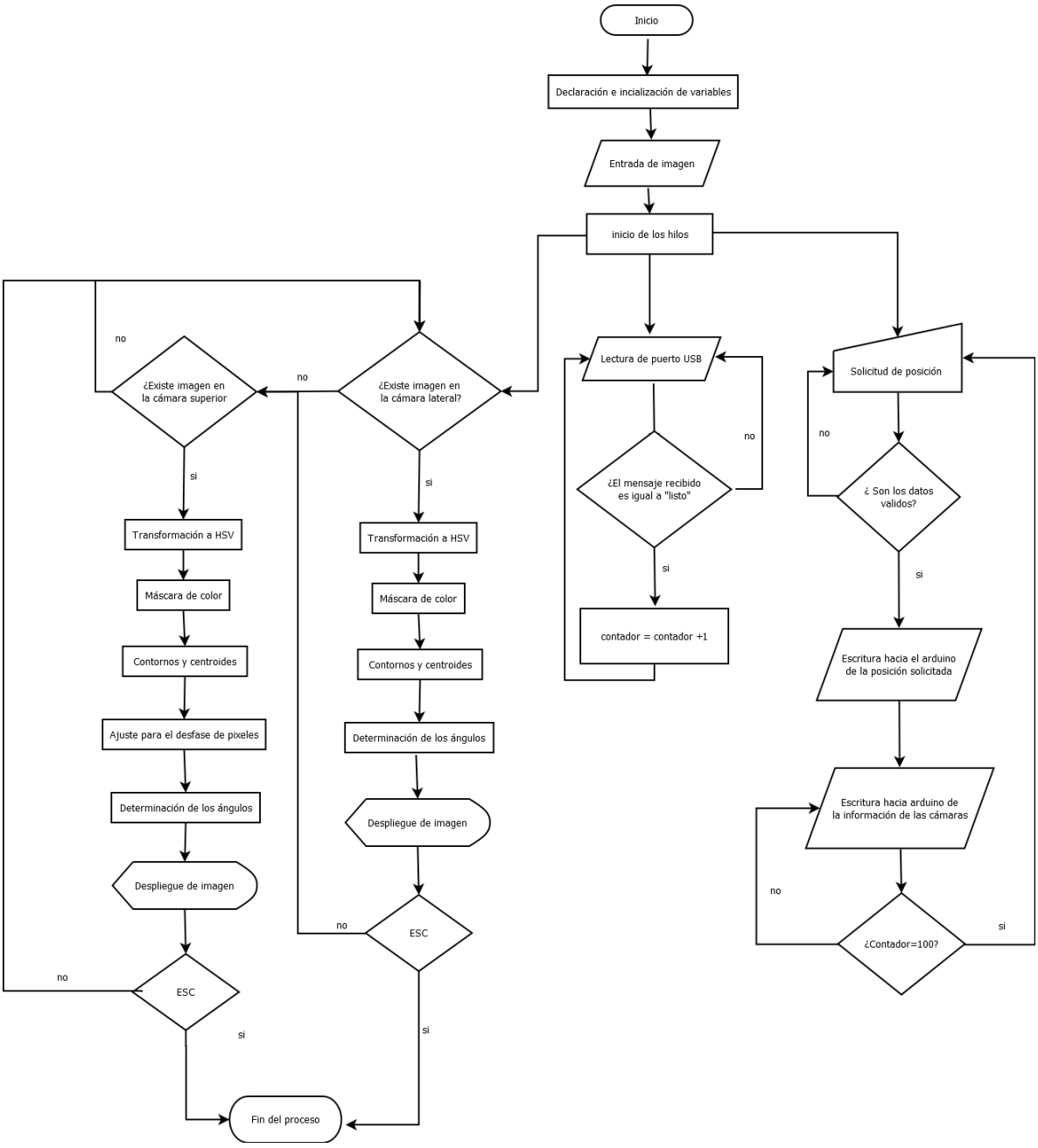


Figura 5.5 Diagrama de flujo del código de solución

Fuente: Elaboración propia

El encabezado del código emplea módulos para acceder a funciones por el intérprete, uso de tiempos, comunicación en serie, hilos de ejecución, arreglos, librerías matemáticas y funciones del sistema.

Se establecen los puertos que se van a utilizar para la comunicación en serie de la siguiente manera.

- puerto = '/dev/ttyACM0'

Para el puerto ACM0 de escritura hacia el arduino se establece con un valor en baudios de 230400

- puerto2= '/dev/ttyUSB0'

Para la lectura del USB que recibe la información proveniente del arduino se establece un valor en baudios de 115200.

Se definen las funciones correspondientes a las diferentes resoluciones, para que el usuario habilite la resolución a la que desea trabajar antes de ejecutar el código, si el hardware de uso le permite hacerlo, finalmente se inicializan todas las variables del código en cero.

5.2.1. Módulo de análisis de imágenes

El módulo de análisis de imágenes es el encargado realizar la toma, procesamiento, y cálculos correspondientes de las capturas realizadas a través de las cámaras.

El módulo se ejecuta como un hilo esto implica que el proceso se llama a ejecutarse una vez y se mantiene trabajando de manera independiente al resto del código, las variables que se determinan se establecen como globales para actualizarse a lo largo de las demás funciones e hilos del código. El módulo es un ciclo constante, esto permite realizar el análisis de manera constante y actualizar la información con cada captura realizada. El código parte de la definición de la captura de las cámaras, esto quiere decir que cada cámara una vez abierta realiza una captura de video, donde este video se procesa como una captura de imágenes, estas capturas se asignan a variables, una variable para la cámara superior y otra para la cámara lateral, esta asignación de variables tiene la característica de retornar un valor booleano, esto implica que si se tiene una captura de la cámara el valor es verdadero, en caso de que no se obtenga una captura el valor es falso, esto se utiliza para hacer una separación del procesado de las tomas utilizando condicionales, esto significa que si existe una toma de la cámara lateral se realiza el proceso de análisis y si no existe se continúa con la misma consulta para la cámara superior, al completar el procesado de las dos cámaras el ciclo se reinicia y se repite el proceso.

En la Figura 5.6 se muestra una toma de la cámara lateral.

Figura 5.6 Captura de la cámara Lateral
Fuente: Elaboración propia

Una vez determinado si existe la captura por parte de la cámara se realiza el procesado de la imagen, si la cámara lateral retorna un valor verdadero se accede al subproceso, dentro de este se realiza una conversión de la imagen del modelo de color RGB al modelo HSV esto para facilitar el manejo de los rangos de color, con la captura en HSV se realiza una máscara que funciona como filtro, esta máscara retorna una imagen binaria del color filtrado, es significa que la imagen resultante es una imagen en blanco y negro, donde los puntos blancos representan unos y los negros ceros, los uno son los pixeles del color filtrado.

El resultado obtenido luego de aplicar el filtro mediante la máscara es uno que presenta ruido a lo largo de la imagen, esto quiere decir que se filtran puntos que corresponden a falsos positivos a lo largo de la imagen, en la Figura 5.7 se presenta el resultado obtenido de la aplicación de la máscara a la captura de imagen.

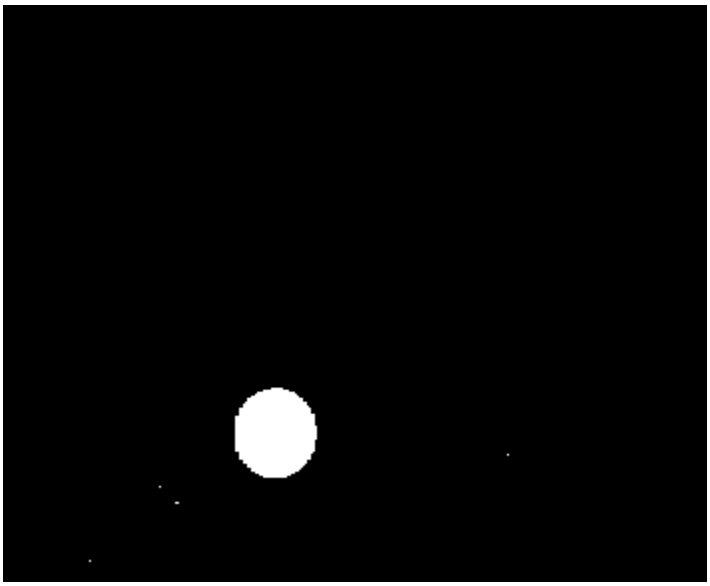


Figura 5.7 Mascara de filtrado de color azul
Fuente: Elaboración propia

Debido a estos ceros que se filtran como unos se aplican transformaciones morfológicas a la máscara, estas trasformaciones son operaciones basadas en la forma de la imagen, las principales morfologías se definen en el Capítulo 3 y el análisis de las muestras morfológicas tomadas se presentan en el Capítulo 6.

La morfología seleccionada para el proceso es la de apertura, la cual es una morfología de erosión seguida de una morfología de dilatación, la operación de morfología recibe dos entradas para su ejecución, una es la imagen sobre la cual se va a realizar el proceso que corresponde a la máscara y la segunda entrada es un kernel. El kernel es un arreglo de coeficientes numéricos de tamaño establecido impar con un punto de ancla colocado en el centro del arreglo, en la Figura 5.8 se presenta un ejemplo de un kernel de 3x3 y su punto de ancla.

1	-2	1
2	-4	2
1	-2	1

Figura 5.8 Representación de la estructura de un kernel
Fuente: Elaboración propia

El proceso morfológico realiza la operación de convolución, se coloca el ancla del kernel sobre el píxel que se desea determinar, con el resto del kernel sobreponiéndose sobre los pixeles adyacentes al ancla, se multiplica cada coeficiente del arreglo del kernel por el correspondiente valor del pixel en la imagen y se suma el resultado, este resultado se sustituye en la posición del pixel donde se encuentra el ancla.

Para el proceso de apertura el kernel define un proceso particular, debido a que la apertura consiste de una erosión seguida de una dilatación, en la erosión el ancla del kernel se coloca sobre todos los pixeles binarios que no son cero y ese especifico pixel se seguirá considerando como uno solo, si todos los pixeles que se encuentren bajo el arreglo del kernel también son uno, de otra forma el píxel es erosionado y se convierte en un cero en la imagen y se descarta. Este proceso sirve para eliminar falsos positivos, ya que, un píxel considerado como un uno que no cuente con otros pixeles positivos en los alrededores, no forman parte del color que se desea filtrar, debido a que el color a filtrar es un conjunto constante de unos.

Luego de aplicar la erosión se aplica la dilatación para hacer recuperación de la sección de la máscara que se reduce después de ser erosionada, la dilatación aplica el kernel sobre los pixeles de la misma manera que la erosión, sin embargo, para la dilatación solo se necesita que uno de los pixeles bajo el arreglo del kernel sea un uno para que el resultado se defina como un uno, este proceso aumenta el área blanca de la máscara sin embargo al haber eliminado previamente los falsos positivos mediante la erosión no se incrementan las áreas que no corresponden al color filtrado. La Figura 5.9 muestra el resultado de aplicar la morfología de apertura.

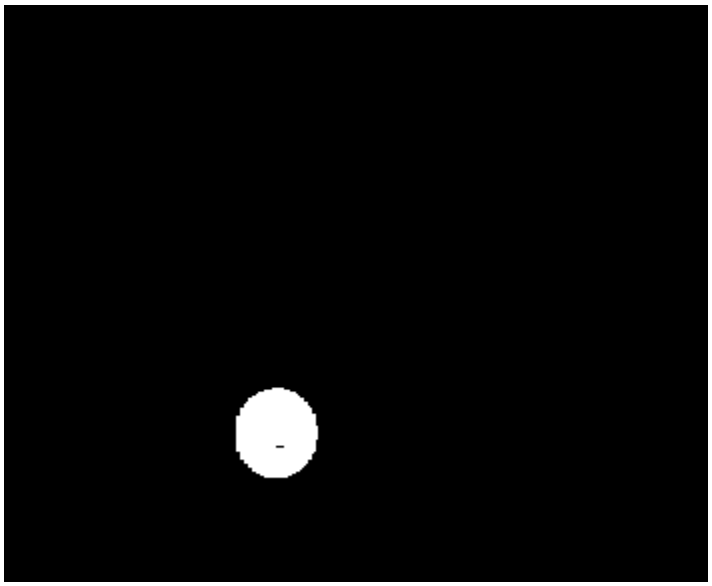


Figura 5.9 Morfología de apertura aplicada a la máscara azul
Fuente: Elaboración propia

Luego de eliminar falsos positivos y obtener una máscara más limpia se realiza un proceso de suavizado de la máscara, con el propósito de llenar posibles aperturas entre el área filtrada y para definir un área circular más notable, para este proceso se recorre cada pixel de la imagen y se reemplaza este pixel por la mediana de sus pixeles vecinos, esto con una vecindad cuadrada alrededor de cada pixel evaluado. En la Figura 5.10 se muestra el resultado de suavizado de la máscara.

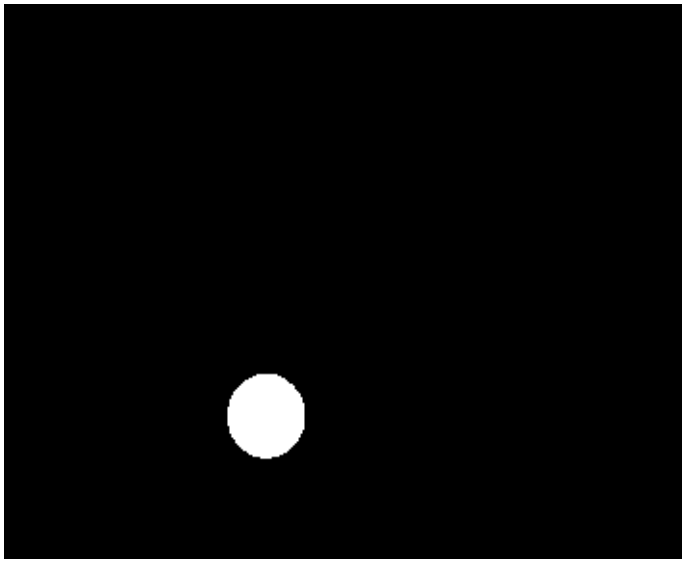


Figura 5.10 Suavizado aplicado a la morfología de apertura
Fuente: Elaboración propia

Cuando se obtiene el resultado de la Figura 5.10 se cuenta con una máscara para color más estable, sobre la misma máscara ya transformada y suavizada se realiza el cálculo del contorno.

La detección de contornos determina los puntos de la circunferencia en la máscara que son colindantes con sus vecinos definidos como ceros, cada contorno es almacenado como un vector de puntos x, y dentro de la matriz. Para la recuperación de los valores de los contornos se utilizó un sistema sin jerarquía esto debido a que cada máscara de cada color no presenta múltiples contornos a detectar. El método de aproximación utilizado fue una cadena de aproximación sencilla cuyo objetivo es determinar los contornos mediante cuatro puntos externos sobre la máscara, esto permite eliminar los puntos redundantes y realizar ahorro de memoria.

Una vez que se obtiene el contorno de la imagen se calcula el momento, el momento de una imagen se determina por el teorema de Green y se define como un peso promedio de la intensidad del píxel, la función para la determinación de los momentos retorna un diccionario con el valor de todos los momentos calculados, un diccionario permite acceder a un valor a través de una clave. De los momentos calculados se requieren los momentos centrales los cuales están definidos como M_{00}, M_{01}, M_{10} estos momentos se utilizan para determinar las coordenadas $x\ y$ que representa el centroide del contorno, establecido como el centro geométrico de una figura plana.

Una vez determinado el momento $x\ y$ se logra dar seguimiento del color al desplazarse el brazo, ya que, en cada toma se hace el filtrado, la determinación de contornos y el cálculo de momentos, a estos momentos $x\ y$, se les asigna una variable para su almacenamiento.

Todo el proceso desde la definición de la máscara se repite para las otras dos marcas de color, la amarilla y la verde, en la cámara lateral con esto se puede obtener los centroides $x\ y$ de los tres colores.

Las seis variables $x\ y$ obtenidas a través de los momentos de las tres marcas, se utilizan para los cálculos de los ángulos entre los centroides de los colores, empleando una librería matemática la cual permite realizar operaciones trigonométricas, además, se deben convertir las variables a punto flotante para que puedan desplazarse entre los subpíxeles de la matriz.

Para el ángulo entre la marca azul y la amarilla se define las variables, x_1y y_1 para la azul y x_2y y_2 para la amarilla, el ángulo se determina, tal y como se muestra en la fórmula 5.1.

$$\arctan\left(\frac{y_1-y_2}{x-x_2}\right) * \frac{180}{\pi} \quad (5.1)$$

El cálculo del arco tangente se realizó para determinar el ángulo entre el punto azul y el amarillo tomando el azul como referencia y el ángulo entre el punto amarillo y el punto verde tomando el amarillo como referencia.

Para la cámara superior, el proceso de filtrado y cálculo de los momentos se realiza de la misma manera que para la cámara lateral, sin embargo, el proceso para calcular el ángulo de rotación es diferente, debido a que, las marcas de color se colocan es posiciones que no permiten calcular el ángulo de rotación de forma directa. Para el cálculo del ángulo de rotación se tomó la marca de color magenta y el punto $x\ y$ se desplazó sobre la matriz de la imagen para ubicarlo en el centro de la base, dado que la marca azul se encuentra a un costado del brazo, era necesario tener un punto en el centro de giro del brazo que se mantuviera estático para medir el ángulo de giro. En la Figura 5.11 se muestra una imagen de la cámara superior donde se observa el color magenta de guía en la base del brazo, el color azul a un costado del brazo y el punto desplazado de color azul el cual se encuentra en el centro de la base.

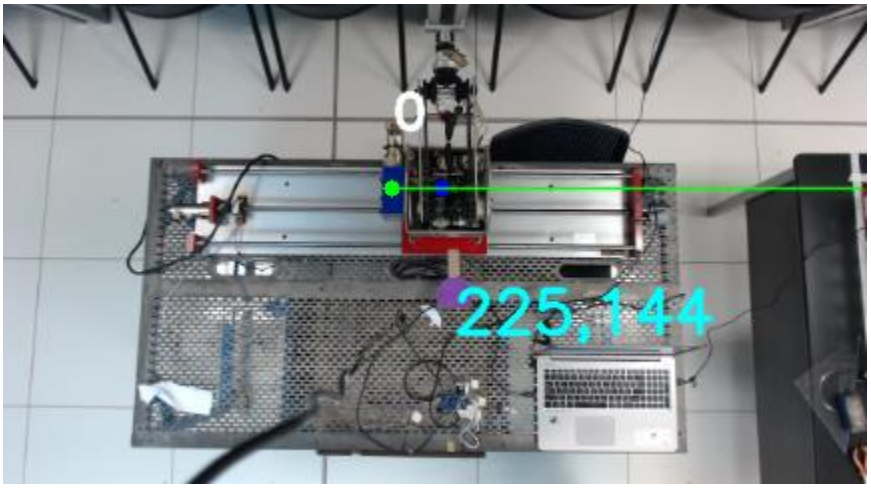


Figura 5.11 Captura de la cámara superior
Fuente: Elaboración propia

Cuando la base ejecuta un giro sin importar el sentido la marca azul giro con la base, sin embargo, la marca magenta se encuentra en una posición en la que no gira con la base, pero si se desplaza con ella esto permite tener un punto de guía para determinar el ángulo de giro de la base

El punto azul de la Figura 5.11 presenta una desviación en la matriz, cuando el brazo se desplaza hacia la izquierda o derecha, el punto no se mantiene en el centro del brazo, esto se debe a que la cámara se encuentra estática y aunque percibe la marca magenta desde una posición diagonal a ella, el punto debe desplegarse en una matriz plana, este corrimiento no presenta errores en la medición del ángulo de rotación, considerando que, solo se desplaza del centro de la base hacia la izquierda o derecha pero no se superpone sobre la marca azul, sin embargo, se planteó una corrección para el desplazamiento de pixeles, se tomó la posición del punto azul en una medición central, la cual se ubicó en la matriz en el pixel 205, luego se desplazó hacia la izquierda y se tomó la posición en pixeles en la que se encontraba el punto azul y la posición en la que debería estar, a partir de este análisis se determinó que la base se desplaza hacia la izquierda desde el pixel 205 hasta el 132, pero debería llegar hasta el pixel 115, con estos valores se establece que en los 73 pixeles que se desplaza el punto hay que reajustar 17 esto implica realizar una corrección de 0.23287671 por cada pixel desplazado, realizando el ajuste para el desplazamiento hacia la izquierda y hacia la derecha se logró mantener en punto de guía en el centro de la base a lo largo de todo el riel de movimiento.

Adicionalmente se llevó a cabo la relación entre el desplazamiento en pixeles en centímetros para determinar de manera experimental la proporción de distancia a la que equivale un píxel.

El hilo de las cámaras una vez llamado se mantendrá en un proceso constante de análisis de las imágenes, si se desea cerrar el ciclo se declaró la tecla ESC como salida, una vez ejecutada la salida se da una liberación de las cámaras para que no se mantengan encendidas y se establece la destrucción de las ventanas desplegadas para borrarlas del monitor, de igual manera se hace un cambio de variable para globalizar la salida en los demás hilos.

5.2.2. Módulo de lectura

El módulo de lectura corresponde al segundo hilo de ejecución del código, se ejecuta como un hilo porque se busca que la lectura sea independiente del proceso general, su única función es leer a través del puerto en serie definido como USB1, este mismo puerto es por el cual el

arduino escribe cuando los motores alcanzan la posición final. La lectura se define como, lectura hasta el carácter ‘o’, esto debido a que, el arduino envía un mensaje con la palabra ‘listo’ y no se busca que ingrese información adicional a este mensaje, una vez leído el puerto se utiliza un condicional y se consulta si el mensaje leído en el puerto en serie es igual al texto ‘listo’ con esto se corrobora que el mensaje que lee es igual al que el arduino envía y no se ha dado pérdida o daño de información, cada vez que se da una lectura correcta del dato se produce un aumento de uno a un contador.

5.2.3. Módulo de toma de datos

El módulo de ingreso de datos es un proceso que no trabaja como hilo, ya que la toma de datos debe realizarse solo al inicio de la ejecución del código y deben volver a solicitarse los valores cuando el arduino lo indique. La toma de valores se realiza uno por uno y los valores entran al sistema como enteros. El proceso se encuentra dentro de un ciclo infinito, sin embargo, una vez que se solicitan los datos se llama a la función de escritura de manera subsecuente y esta función únicamente retorna para reiniciar la toma de datos cuando el hilo de lectura recibe el mensaje del arduino.

5.2.4. Módulo de escritura

El módulo de escritura se llama como una función, pero el llamado se realiza internamente desde la función de ingreso de datos, esto debido a como se ejecuta la escritura, una vez que el usuario ingresa los datos de posición a los que desea llegar se hace el llamado a la función de escritura, esta función se encarga de escribir a través del puerto COM0 hacia el arduino la posición que el usuario desea alcanzar en un primer mensaje, luego entra en un ciclo constate en el que escribe las variables calculadas en el módulo de análisis de imágenes, esto permite dar retroalimentación de información al arduino, debido a que cada vez que un valor angular o de posición cambia globalmente en el código también cambia en la función de escritura. Los datos angulares y de posición que el usuario solicita y los que se calculan de la toma de imágenes se agrupan como un solo paquete y se coloca un símbolo de separación entre cada uno para ser enviados, el arduino al recibir el mensaje se encarga de separar y asignar cada valor a una variable para posteriormente hacer el proceso de comparación de datos y activación de los motores. El proceso de escritura es cancelado por la variable contador que se encuentra en el hilo de lectura si esta alcanza un valor de 100, esto quiere decir que se ha recibido por parte del arduino el mensaje ‘listo’ 100 veces el sistema de lectura cambia de

estado una bandera global para que la función de escritura retorne a la función de toma de datos, el sistema deja de escribir vuelve a pedir un nuevo conjunto de datos al cual el usuario quiera desplazar el equipo y vuelve a escribir, envía los datos deseados de posición y vuelve al ciclo donde continua enviando la información determinada por las cámaras, este ciclo se repite cada vez que el brazo alcanza la posición solicitada.

5.2.5. Código del microcontrolador

El arduino al corresponder a un hardware independiente a la unidad de procesamiento de imágenes, requiere de un código para recibir y procesar la información para poder dar un accionar a los motores.

El encabezado del código conlleva todas las variables a utilizar y las asignaciones de los números de pines, luego en la sección de inicialización de variables se inician los pines de comunicación en serie a utilizar y los baudios de operación, para la implementación se utilizaron Serial y Serial1 de los tres pines de salida en serie con los que cuenta el arduino mega, en esta sección se definen los pines digitales como salidas para que envíen los voltajes que operaran como señales de control.

El código parte de una verificación de información disponible en el puerto en serie, mientras el puerto en serie cuente con información se inicia el ciclo, en el interior del ciclo existen dos procesos, el de los datos enviados por el usuario y el de los datos actualizados mediante la unidad de procesamiento de imágenes, el recibirse la información del puerto en serie esta se almacena y posteriormente se lee el primer carácter del mensaje, este carácter varía entre el mensaje con los datos del usuario y el mensaje con los datos determinados de las imágenes, de esta manera se termina a que grupos de datos pertenece el mensaje, una vez identificado el grupo de datos se procede a su separación, debido a que los datos con información están divididos en el mensaje por un carácter específico se puede leer el mensaje en orden y determinar donde inicia y donde termina un dato sin importar su longitud y así extraerlo del mensaje y guardarlo en una variable, una vez que los datos han sido extraídos y almacenados en variables se cuenta tanto con los datos solicitados por el usuario como con los datos determinados en la unidad de procesamiento de imágenes, posteriormente se procede a realizar una comparación en un grupo de condicionales, si la posición que mide la cámara es menor que la deseada por el usuario se produce la activación del motor para alcanzar la

posición, en el caso contrario se activa el motor con la señal contraria, se cuenta con un rango de variación de cinco unidades en ambos sentidos desde el valor esperado, si el ángulo medido por la cámara no es mayor ni menor a este rango significa que la posición se encuentra dentro del rango permitido y se procede a apagar los motores, este análisis se reproduce para todas las variables de todos los motores, este proceso se repite de manera constante por lo que existe una retroalimentación de movimiento para alcanzar el rango

Cuando se accede al condicional de apagado de un motor se activa una variable booleana, una vez que todos los motores activan su respectivas variables booleanas indican que han alcanzado la posición deseada, al alcanzar la posición deseada el arduino escribe mediante el puerto Serial1 un mensaje con la palabra ‘listo’ para indicar a la unidad de procesamiento de imágenes que se ha alcanzado la posición y se puede solicitar un nuevo conjunto de datos, sin embargo, este proceso se corrobora mediante un contador para asegurar la posición.

En la siguiente figura se presenta el diagrama de flujo del microcontrolador para un motor

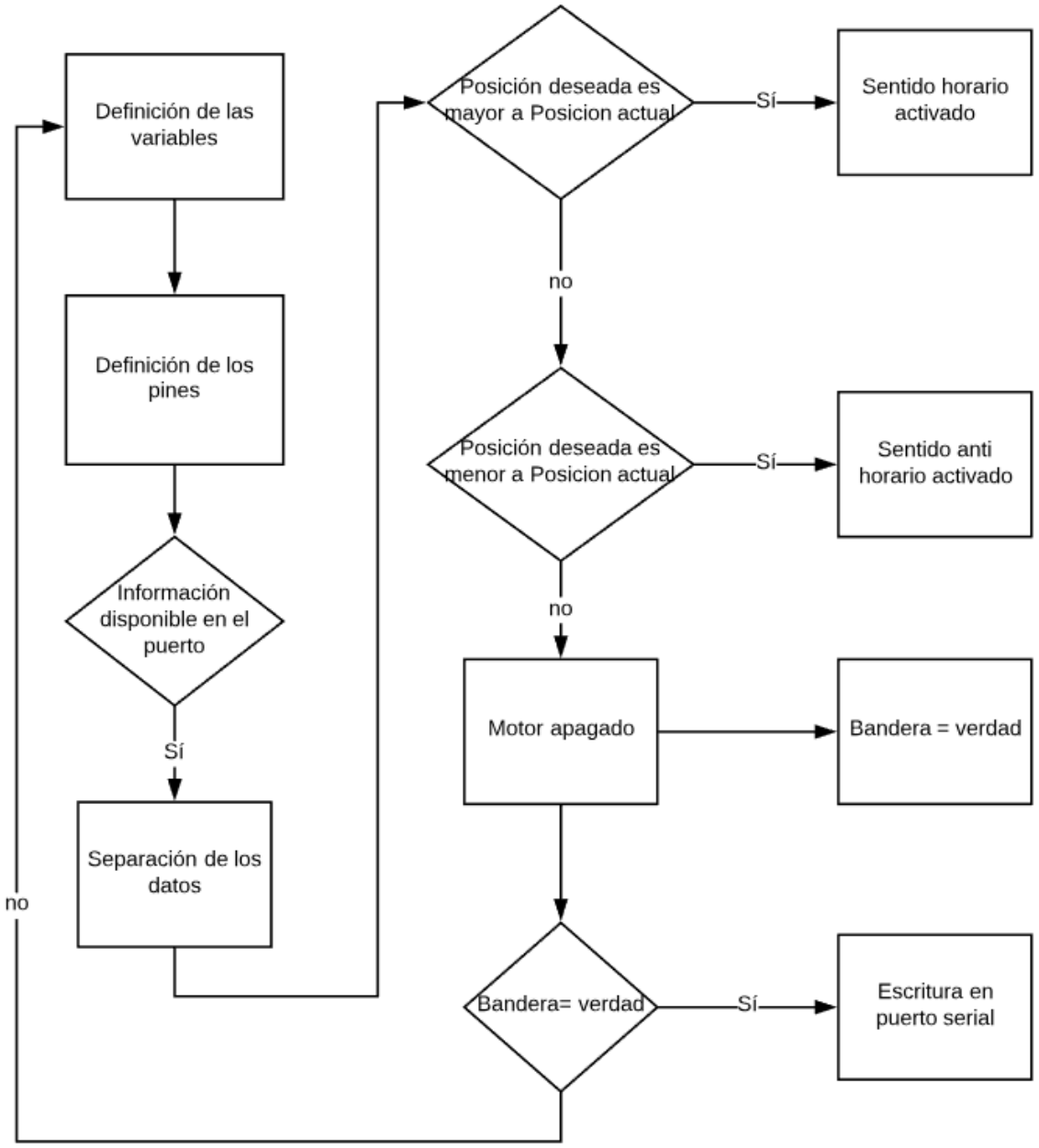


Figura 5.12 Diagrama de flujo del microcontrolador
Fuente: Elaboración propia

CAPITULO 6: ANÁLISIS DE RESULTADOS

En esta sección se presenta la tabulación y representación de los resultados obtenidos a través de las pruebas realizadas y el análisis correspondiente al funcionamiento del sistema en los diferentes puntos a tratar.

El análisis de cada resultado se realizará conforme se vaya presentando y se dividirá en las diferentes muestras tomadas. Para efectos del análisis de resultados los ángulos correspondientes al brazo serán los ángulos en la parte inferior de la captura formados entre el color azul y el amarillo y los correspondientes al antebrazo en la parte superior entre el color amarillo y el verde.

6.1. Análisis de morfologías

Para el análisis de morfologías se tomaron muestras de las distintas morfologías aplicables para los diferentes puntos de color a diferentes distancias de la cámara, para determinar cuál era la mejor combinación de parámetros para la implementación de la solución

Para las diferentes pruebas realizadas se contó con tres posiciones para la base del motor, una primera posición cercana a la cámara de 76 cm, una segunda a una distancia media de 118.8 cm y una tercera a una distancia lejana de 160 cm esto para simular y determinar el comportamiento de las pruebas en desplazamientos laterales.

Para las pruebas morfológicas se tomaron siete morfologías diferentes sobre las mismas máscaras para ver cual representa un mejor resultado.

En la siguiente figura se presenta una captura de la cámara lateral sobre la que se aplicaron las morfologías.

Figura 6.1 Toma de la cámara lateral del Pegasus
Fuente: Elaboración propia

Se presentarán todas las pruebas morfológicas para el color azul y a partir de la selección de morfología se presentarán las pruebas para los demás colores y las demás distancias.

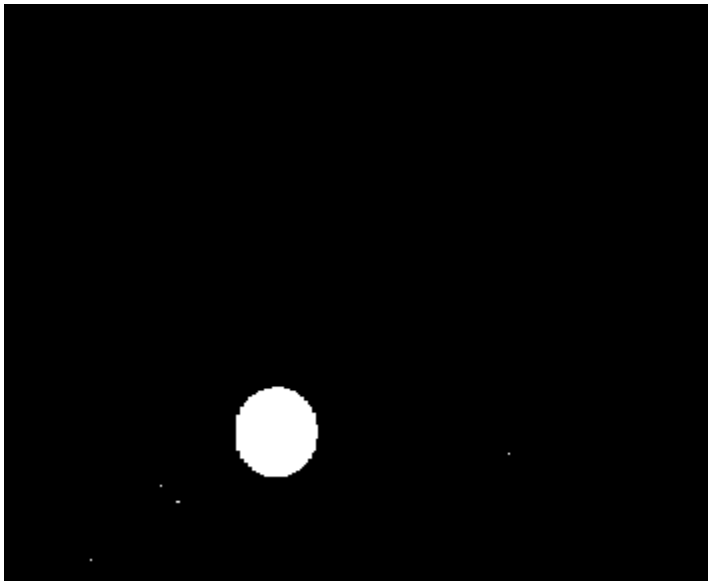


Figura 6.2 Mascara original para filtrado azul
Fuente: Elaboración propia

Figura 6.3 Morfología blackhat
Fuente: Elaboración propia

Figura 6.4 Morfología tophat
Fuente: Elaboración propia

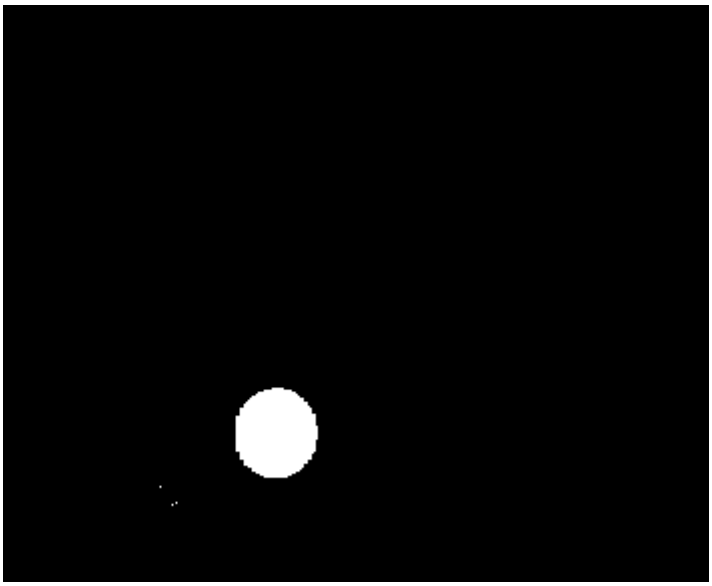


Figura 6.5 Morfología de clausura
Fuente: Elaboración propia

Figura 6.6 Morfología de dilatación
Fuente: Elaboración propia

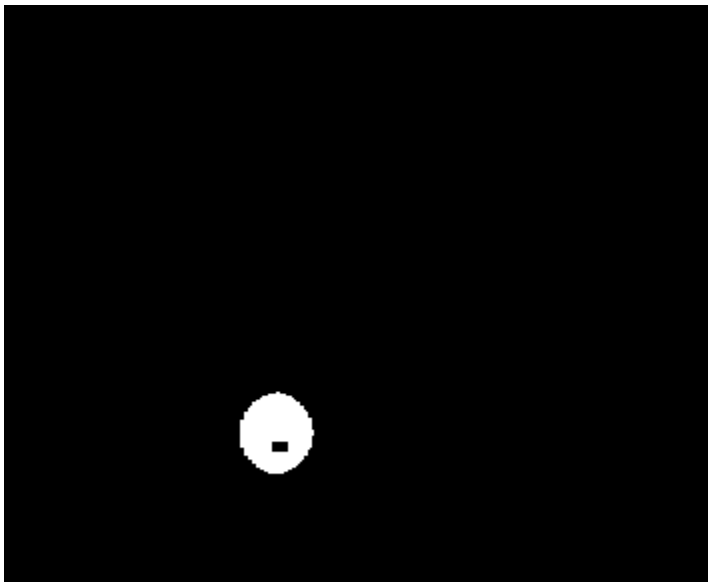


Figura 6.7 Morfología de erosión
Fuente: Elaboración propia

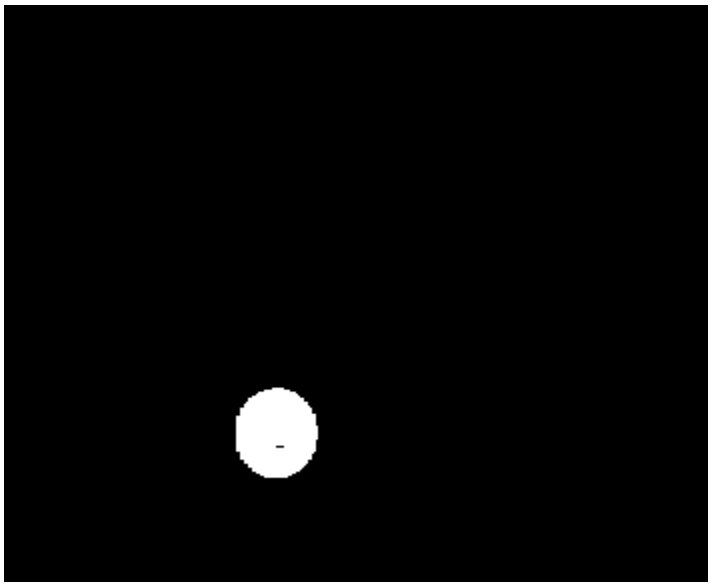


Figura 6.8 Morfología de Apertura
Fuente: Elaboración propia

Figura 6.9 Gradiente morfológico
Fuente: Elaboración propia

Una vez realizadas las pruebas morfológicas se procede a determinar cuál de las aplicadas representa una mejor detección y eliminación del ruido. Las morfologías de tophat y blackhat se descartan debido a que no presenta detección alguna del color, esto se debe a que el resultado de estas morfologías es la diferencia entre las mascara original y las morfologías de apertura para el caso de tophat y clausura para blackhat, dado que la máscara original se asemeja mucho a las morfologías de apertura y clausura, el resultado de tophat y blackhat se vuelve irrelevante y se descartan del proceso.

El gradiente morfológico realiza la diferencia entre la morfología de dilatación y erosión por lo que da como resultado la línea exterior de la máscara de detección para el color azul, debido a que se trabaja con contornos se desea que el color sea constante en la máscara y sea una superficie rellena por lo que el gradiente morfológico no es aplicable para el resultado deseado.

La morfología de dilatación presenta un incremento en el área de los falsos positivos de la imagen por lo que no funciona para la detección del color, mientras que, la máscara de erosión produce una reducción de la máscara por lo que elimina los falsos negativos, pero también reduce el área presente en la máscara original.

De las siete morfologías aplicadas las dos con mejor respuesta para procesamiento de la máscara fueron la de apertura y la de clausura ambas son una combinación de la erosión seguida de una dilatación y dilatación seguida de una erosión respectivamente.

Se determinó que la morfología de apertura era la más indicada para procesar la máscara debido al orden con que se aplican la erosión y la dilatación, si se utilizara la clausura se estaría aplicando primero la dilatación y si se presentara un falso negativo con un área más grande que la del kernel, dilatarlo y luego erosionarlo no eliminara por completo el falso positivo, por eso se decide utilizar la apertura para realizar primero la erosión de los falsos negativos y luego la dilatación del área final permitiendo recuperar la sección que se reduce de la máscara.

Una vez aplicada la morfología se realiza un suavizado en base a un arreglo de pixeles impar, el suavizado permite que el resultado final sea una imagen con un contorno más constante a lo largo de la circunferencia y como el proceso que aplica el dar a un pixel el valor de la media de los pixeles vecinos también cumple la función de eliminar falsos negativos mínimos en área que se filtren en la morfología, para el color azul el arreglo se tomó de 15x15 ya que para arreglos menores se presentaban falsos negativos.

En la siguiente imagen se presenta la morfología de apertura suavizada.

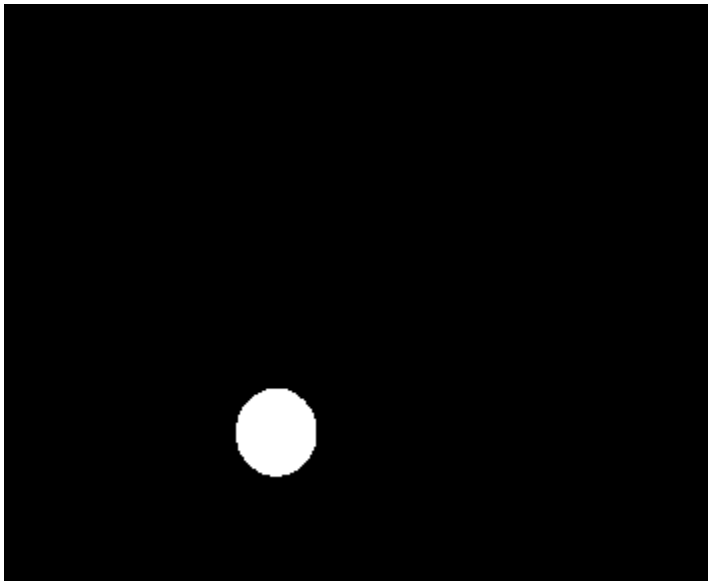


Figura 6.10 Máscara final suavizada para color azul a 76 cm
Fuente: Elaboración propia

Una vez determinada la morfología de trabajo se procedió a realizar las tomas para los diferentes colores y a diferentes distancias.

El color azul no presentó problemas de detección a las distancias de 118.8 cm y 160 cm como se muestra a continuación.

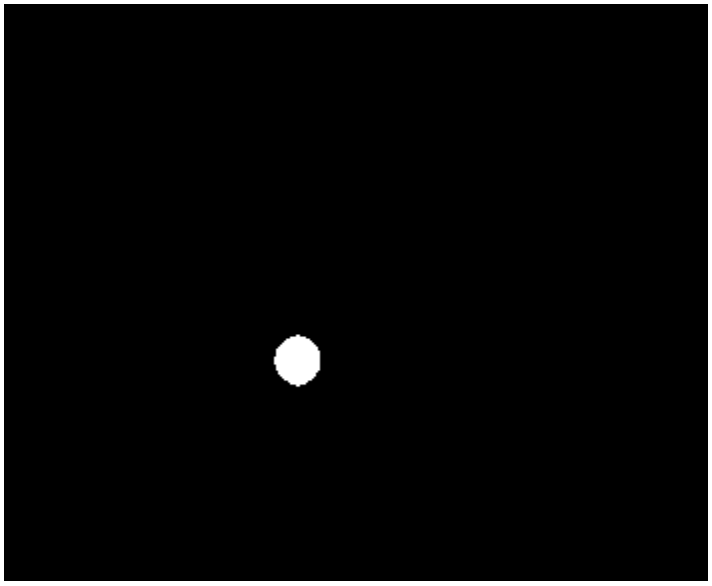


Figura 6.11 Máscara final suavizada para color azul a 118.8cm
Fuente: Elaboración propia

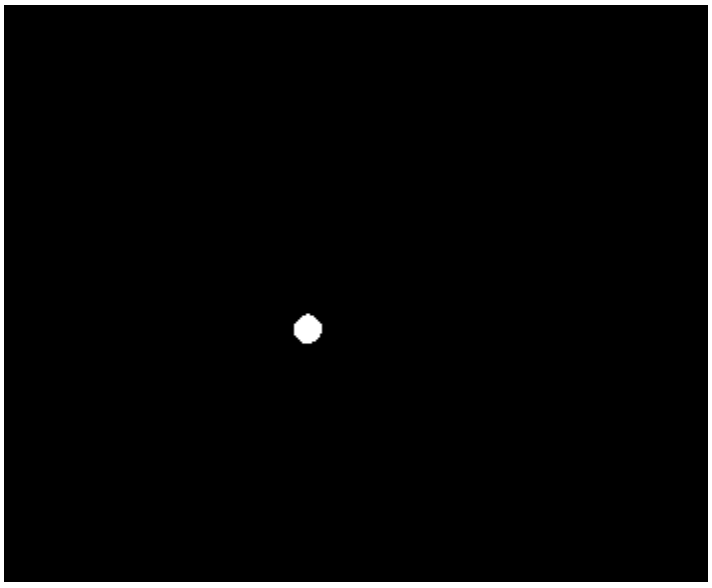


Figura 6.12 Máscara final suavizada para color azul a 118.8cm
Fuente: Elaboración propia

El color amarillo y el color verde presentan menores áreas a la hora de realizar la detección, debido a esto fue necesario aplicar las morfologías de apertura y reducir el tamaño del arreglo del suavizado ya que a la distancia más lejana de 160 cm, el área resultante de la morfología era muy pequeña y el suavizado producía una reducción de área, ya que, se contaba con mayoría de pixeles vecinos establecidos como ceros, dando como resultado un área que podía presentar perdidas de detección por lo que se ajustó el tamaño del arreglo del suavizado a un valor distinto.

Para el color amarillo el arreglo del suavizado se estableció de 11x11 y el del color verde de 9x9 esto con el fin de no perder color a la mayor distancia, pero tampoco permitir falsos positivos en los puntos más cercanos.

En las siguientes figuras se presentan las máscaras finales para el color amarillo y verde respectivamente y a las diferentes distancias.

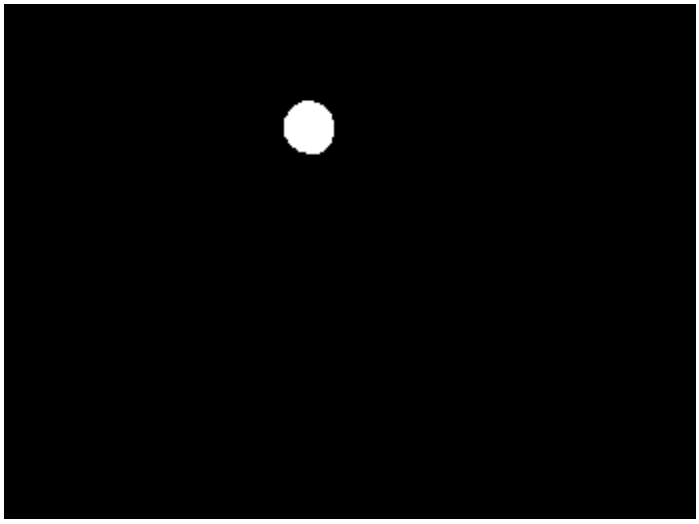


Figura 6.13 Máscara final suavizada para color amarillo a 76 cm
Fuente: Elaboración propia

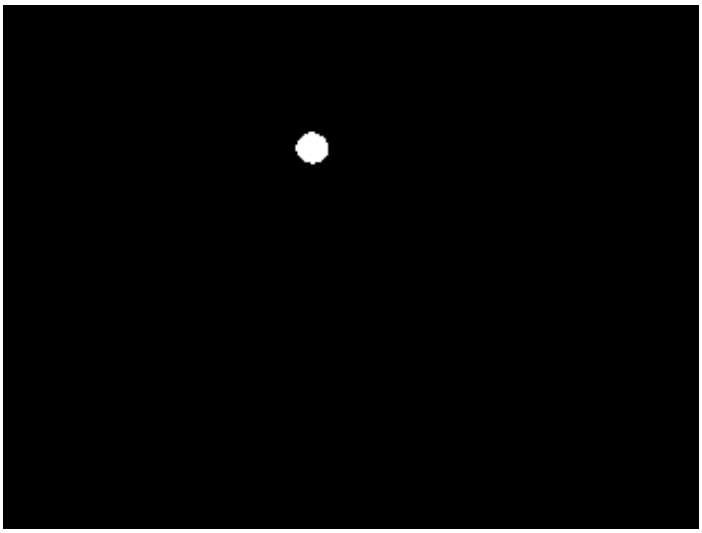


Figura 6.14 Máscara final suavizada para color amarillo a 118.8 cm
Fuente: Elaboración propia

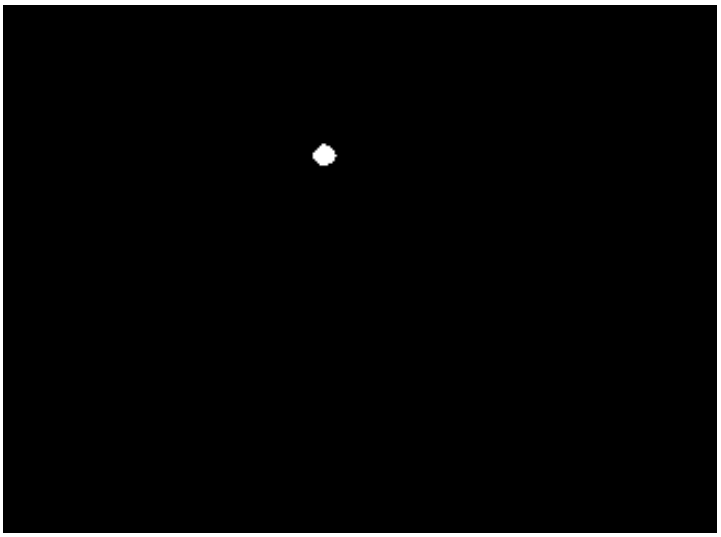


Figura 6.15 Máscara final suavizada para color amarillo a 160 cm
Fuente: Elaboración propia

Figura 6.16 Máscara final suavizada para color verde a 76 cm
Fuente: Elaboración propia

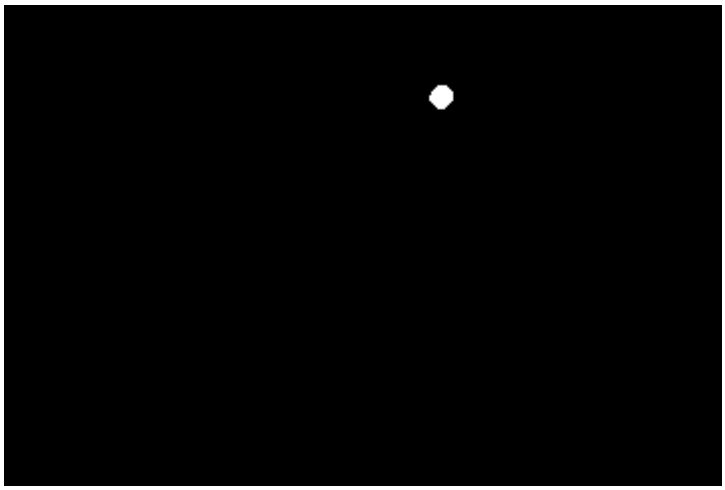


Figura 6.17 Máscara final suavizada para color verde a 118.8 cm
Fuente: Elaboración propia

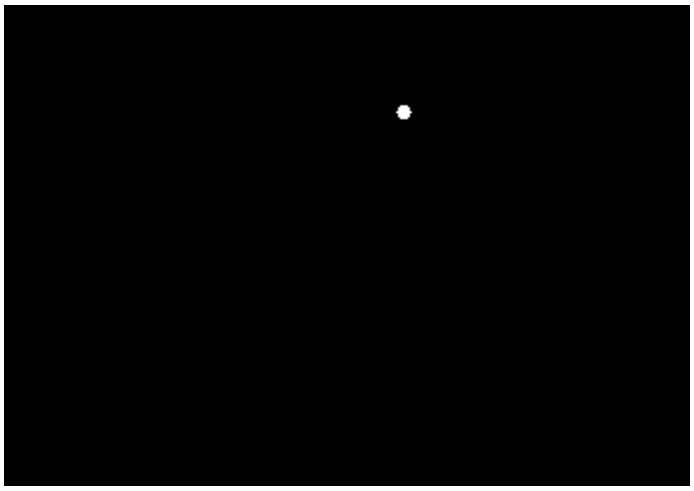


Figura 6.18 Máscara final suavizada para color verde a 160 cm
Fuente: Elaboración propia

La aplicación de las morfologías se ejecutó de igual manera para las marcas de color azul y magenta de la cámara superior.

En las siguientes figuras se presentan las máscaras correspondientes a la cámara superior.

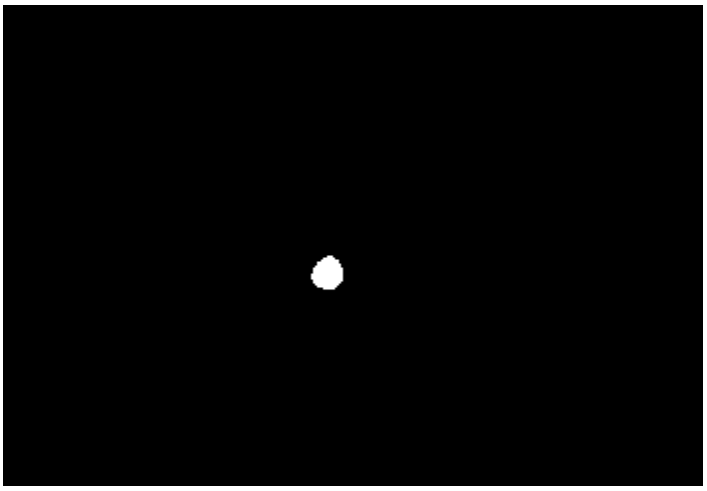


Figura 6.19 Mascara de la cámara superior para color magenta
Fuente: Elaboración propia

Figura 6.20 Mascara de la cámara superior para color azul
Fuente: Elaboración propia

6.2. Mediciones físicas de ángulos

Para determinar el funcionamiento en la determinación de los ángulos por medio del código se tomaron diferentes muestras a las tres distancias de prueba, luego se realizó la toma de diez mediciones físicas para esos ángulos con el propósito de establecer la exactitud.

En las siguientes figuras se presentan los ángulos determinados por el código a las diferentes distancias.

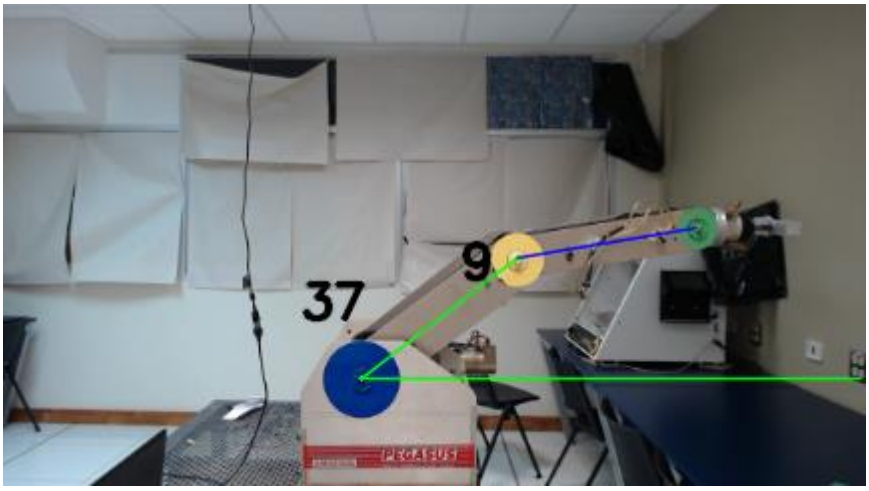


Figura 6.21 Captura de ángulos a 76 cm
Fuente: Elaboración propia

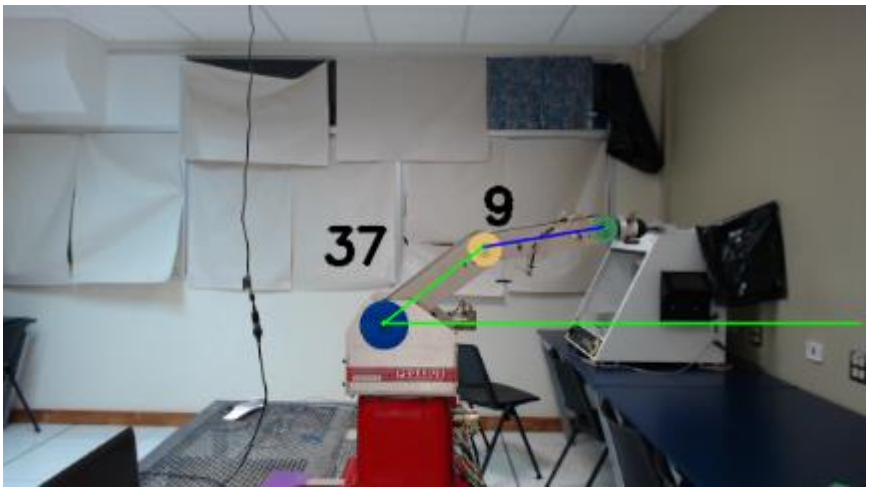


Figura 6.22 Captura de ángulos a 118.8 cm
Fuente: Elaboración propia

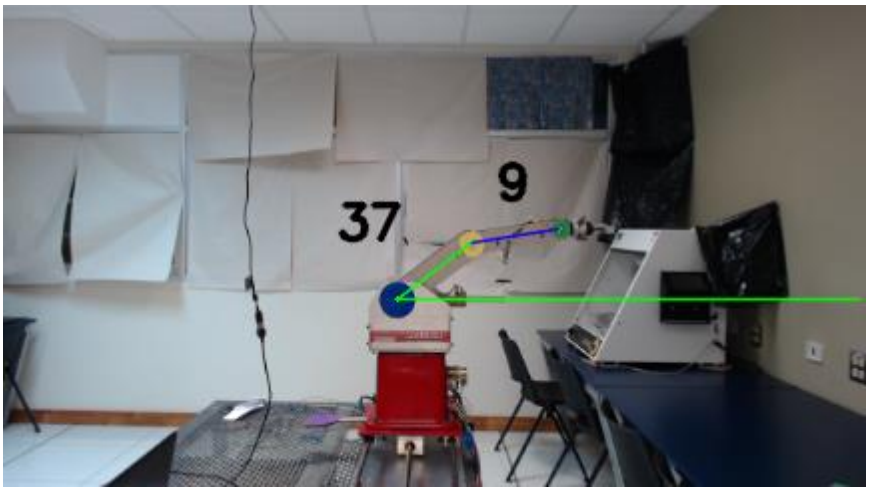


Figura 6.23 Captura de ángulos a 160 cm
Fuente: Elaboración propia

Para las tres distancias no hubo variación en la primera muestra de ángulos determinados por la cámara, se tabularon las 10 mediciones físicas.

Tabla 6.1 Primera muestra de ángulos físicos y ángulos determinados mediante la cámara para las tres distancias de muestreo

Primera Muestra											
76 (±0.05 cm)				118.8 (±0.05) cm				160 (±0.05) cm			
Ángulo de cámara		Ángulo Físico (° ± 0.5)		Ángulo de cámara		Ángulo Físico (° ± 0.5)		Ángulo de cámara		Ángulo Físico (° ± 0.5)	
37	9	37.5	9	37	9	37	8	37	9	37.5	9.5
37	9	36.5	9.5	37	9	36.5	8.5	37	9	37	9
37	9	36.5	9.5	37	9	37.5	9.5	37	9	36.5	9.5
37	9	37	9.5	37	9	37	9.5	37	9	37	9
37	9	36	9	37	9	37	9	37	9	37	9
37	9	37	8.5	37	9	36	9	37	9	37.5	9.5
37	9	36.5	8	37	9	36.5	9.5	37	9	36	8.5
37	9	37	8	37	9	37.5	8.5	37	9	36.5	8.5
37	9	37	9	37	9	37	8.5	37	9	37	9.5
37	9	36.5	9.5	37	9	37	9.5	37	9	37.5	8.5

Fuente: Elaboración propia

Tabla 6.2 Segunda muestra de ángulos físicos y ángulos determinados mediante la cámara para las tres distancias de muestreo

Segunda Muestra											
76 (±0.05 cm)				118.8 (±0.05) cm				160 (±0.05) cm			
Ángulo de cámara		Ángulo Físico (° ± 0.5)		Ángulo de cámara		Ángulo Físico (° ± 0.5)		Ángulo de cámara		Ángulo Físico (° ± 0.5)	
16	61	16.5	61.5	16	61	16	61.5	16	60	15.5	61.5
16	61	15	61.5	16	61	16.5	61.5	16	60	15.5	61
16	61	15.5	61	16	61	15	62	16	60	16	62
16	61	15.5	61.5	16	61	15.5	61	16	60	16	62
16	61	15	62	16	61	16	61.5	16	60	16	61.5
16	61	16	61.5	16	61	17	60.5	16	60	16.5	61
16	61	16.5	61.5	16	61	16.5	61	16	60	16	60.5
16	61	14.5	61.5	16	61	16.5	61	16	60	15	60.5
16	61	16.5	61	16	61	15.5	61.5	16	60	16.5	61
16	61	16.5	60.5	16	61	16	62	16	60	15.5	61

Fuente: Elaboración propia

Tabla 6.3 Tercera muestra de ángulos físicos y ángulos determinados mediante la cámara para las tres distancias de muestreo

Tercera Muestra											
76 (±0.05 cm)				118.8(±0.05 cm)				160(±0.05 cm)			
Ángulo de cámara		Ángulo Físico (° ± 0.5)		Ángulo de cámara		Ángulo Físico (° ± 0.5)		Ángulo de cámara		Ángulo Físico (° ± 0.5)	
103	3	130	3.5	102	2	103	3	104	3	103.5	3
103	3	103.5	3	102	2	103	3	104	3	103.5	2.5
103	3	103	3.5	102	2	103.5	3	104	3	103	3
103	3	103	3.5	102	2	103	3	104	3	102.5	3.5
103	3	103	3.5	102	2	103.5	3	104	3	103	3
103	3	103.5	3	102	2	102.5	3.5	104	3	103.5	3.5
103	3	102.5	3	102	2	103	3.5	104	3	103	3.5
103	3	103	3.5	102	2	102.5	2.5	104	3	103	3.5
103	3	103	3	102	2	103.5	3.5	104	3	103.5	3
103	3	103	3	102	2	103	2.5	104	3	103.5	3

Fuente: Elaboración propia

El propósito del análisis entre los ángulos determinados por el código y las mediciones realizadas físicamente es observar la variación del ángulo que determina la cámara cuando se produce un desplazamiento de los puntos de detección, el alejar la base dificulta la detección de las marcas de color y produce una reducción de los contornos, esto puede incurrir en cambios del ángulo detectado por la cámara, para las tres pruebas realizadas solo se da una variación de ángulo en la segunda prueba en la máxima distancia, el ángulo del antebrazo varia en un grado, y para la tercera prueba el ángulo del antebrazo y el del brazo varían en un grado a una distancia central, a una distancia lejana el ángulo del brazo varía en dos grados, estas variaciones se presentan debido a que el desplazamiento del brazo implica una variación del contorno de detección y a diferentes posiciones de giro y desplazamiento cada marca de color recibe una exposición a la luz diferente lo significa que la composición RGB que detecta la cámara varia de forma distinta para cada marca.

Es importante establecer que entre más se acerque una marca de color a los extremos del ángulo de visión de la cámara existe una mayor posibilidad de que la detección del ángulo varíe, esto debido a que la marca presenta inclinación su área varía respecto a un área que se encuentre frontal a la cámara, ya que la toma es una representación en dos dimensiones.

Otro punto para considerar es que los ángulos se determinan a partir de dos puntos, el ángulo del brazo se calculó entre el punto azul y el amarillo, mientras que el ángulo del antebrazo se calculó entre el amarillo y el verde, por lo que una variación del contorno amarillo que desplace el centro puede afectar tanto el ángulo del brazo como el del antebrazo.

La variación determinada por la cámara se encuentra dentro del rango de operación establecido para la implementación de la solución, pero debe tenerse presente, que puede verse afectada por factores del entorno.

Otro factor para el análisis de las muestras físicas es ver la exactitud en el cálculo de los ángulos por parte del código, para las muestras a 76 cm que representan las tomas más exactas por la cámara, los ángulos físicos varían en ±1.5 grados parte de estas variaciones pueden presentarse por errores de medición humanos. Las mediciones presentan un porcentaje de error inferior al 10% por lo que están dentro de un margen de trabajo tolerable.

Debido a que no se cuenta con una manera fiable de medir físicamente los ángulos determinados por la cámara superior por la estructura del equipo se toma como guía el análisis de la cámara lateral debido a que ambas cámaras implementan el mismo análisis para la determinación de ángulos.

6.3 Análisis de rotación

En esta sección se analiza la variación de la detección del ángulo que captura la cámara debido a la rotación de la base, esto con el fin de caracterizar la desviación para rotaciones horarias y anti horarias y determinar qué sentido de giro presenta la mayor desviación, este análisis parte del fundamente de la estereoscopia y de que no se cuenta con una cámara de doble lente que ayude a determinar los cambios de profundidad de las marcas de color cuando la base gira.

Se tabularon los datos determinados a través de las capturas de la cámara para rotación desde -25 grados en la base hasta 25 grados, estos límites fueron seleccionados basándose en los rangos máximo de detección de la cámara en los que aún no se presenta perdida de las marcas de color.

Tabla 6.4 Variación de los ángulos del brazo determinados mediante código para diferentes rotaciones de base

Rotación (°)	Posición 1		Posición 2		Posición 3	
	Valor real	Brazo	Valor real	Brazo	Valor real	Brazo
25	30	44	12	22	1	10
20	30	38	12	19	1	7
15	30	35	12	18	1	5
10	30	33	12	16	1	4
5	30	31	12	14	1	3
0	30	30	12	12	1	1
-5	30	29	12	11	1	1
-10	30	29	12	11	1	0
-15	30	28	12	11	1	0
-20	30	29	12	11	1	-2
-25	30	29	12	11	1	-3

Fuente: Elaboración propia

La desviación angular se tabula para el rango de rotaciones medidas.

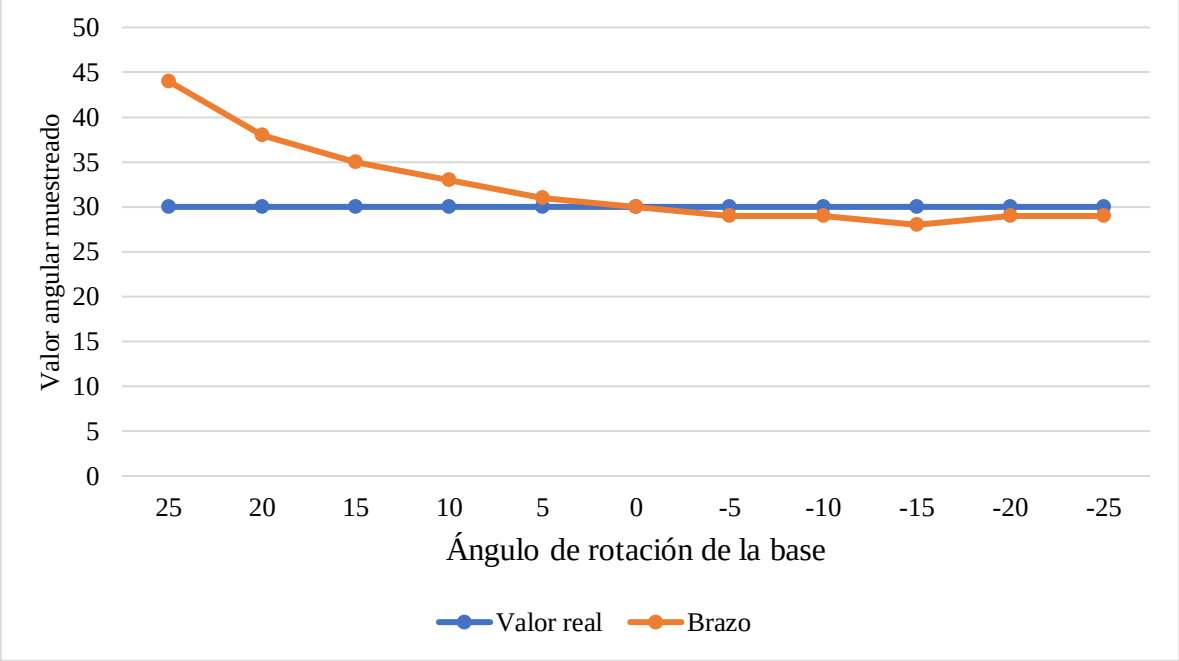


Figura 6.24. Variación de la determinación angular del brazo en la primera posición
Fuente: Elaboración propia

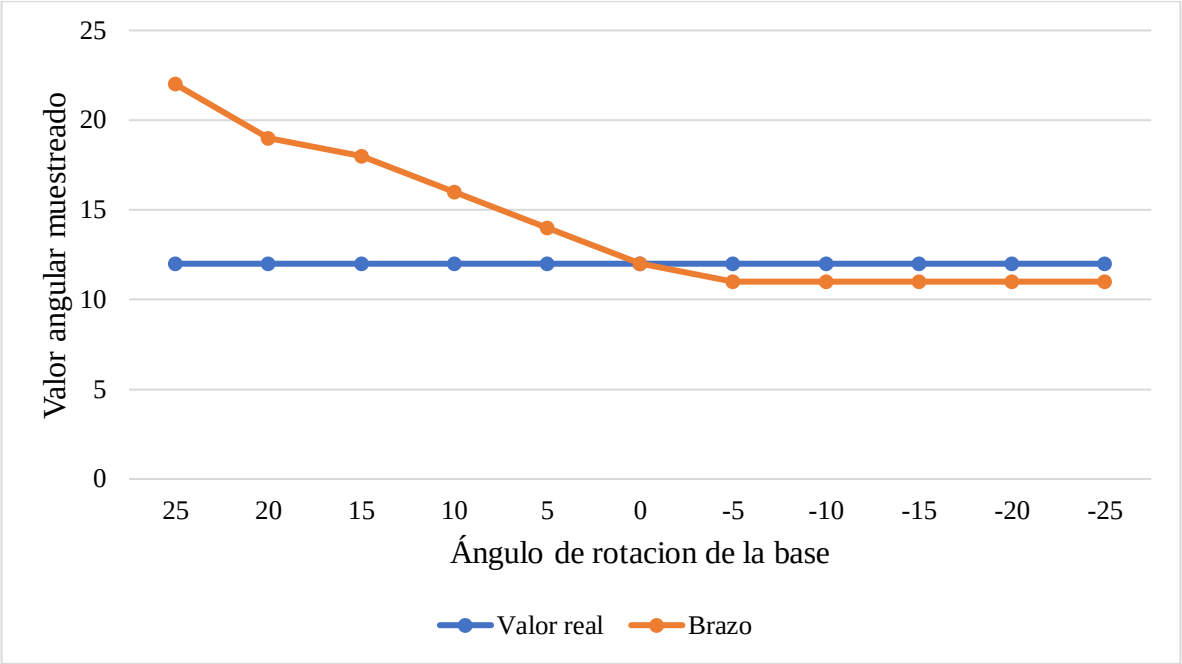


Figura 6.25 Variación de la determinación angular del brazo en la segunda posición
Fuente: Elaboración propia

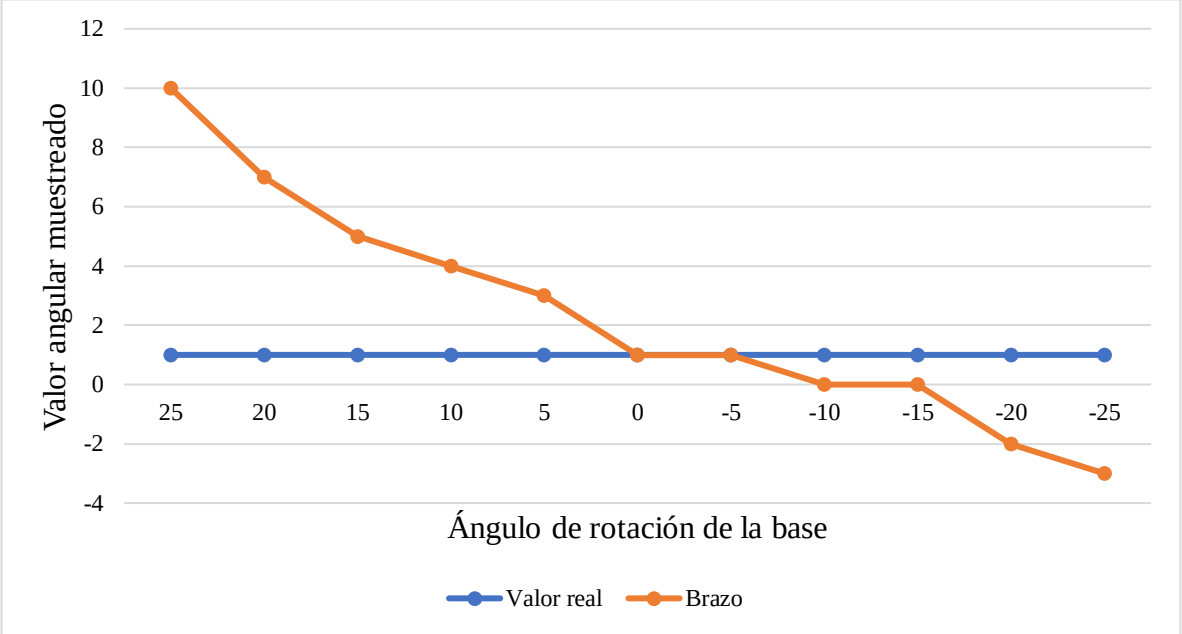


Figura 6.26 Variación de la determinación angular del brazo en la tercera posición
Fuente: Elaboración propia

En las figuras 6.24, 6.25 y 6.26 se muestra la desviación de los valores medidos por la cámara en color naranja conforme se varia la rotación de la base en ambos sentidos, el color azul representa el ángulo que debería de medirse y mantenerse constante para todas las rotaciones. La rotación positiva presenta una mayor desviación que la negativa, esto se debe a que la

rotación positiva hace que las marcas de color se alejen de la base de la cámara mientras que la negativa acerca las marcas de color hacia la cámara, permitiendo que la marca de color mantenga un contorno más constante.

En la Figura 6.27 y la Figura 6.28 se presenta la afectación de los contornos de detección para una rotación de la base positiva y negativa respectivamente.

Figura 6.27 Contornos de detección para una rotación positiva de la base
Fuente: Elaboración propia

Figura 6.28 Contornos de detección para una rotación negativa de la base
Fuente: Elaboración propia

Otro factor importante es que la desviación de los valores del punto real se vuelve más pronunciada conforme menor sea el valor del ángulo, esto se aprecia principalmente en la Figura 6.26 donde la desviación tanto de rotación positiva como negativa es mayor que en la

Figura 6.24 y la Figura 6.25 esto se debe a que la matriz toma el punto de origen para medir el ángulo y establece ese origen de manera horizontal sobre la matriz de dos dimensiones, cuando las marcas de color se encuentran al mismo nivel que la línea de origen y se ven expuestas a una rotación, la variación con respecto al punto de origen es más marcada ya que los colores están horizontalmente iguales y es en este punto donde las diferencias de profundidad dan mayor variación, a diferencia de un ángulo recto donde una marca de color esta sobre otra y la rotación presenta un cambio en la inclinación de la marca de color más que un cambio de profundidad.

En las siguientes figuras se ejemplifica la variación en la detección del ángulo para valores cercanos al punto de referencia, los ángulos establecidos fueron de un grado para el brazo y el antebrazo.

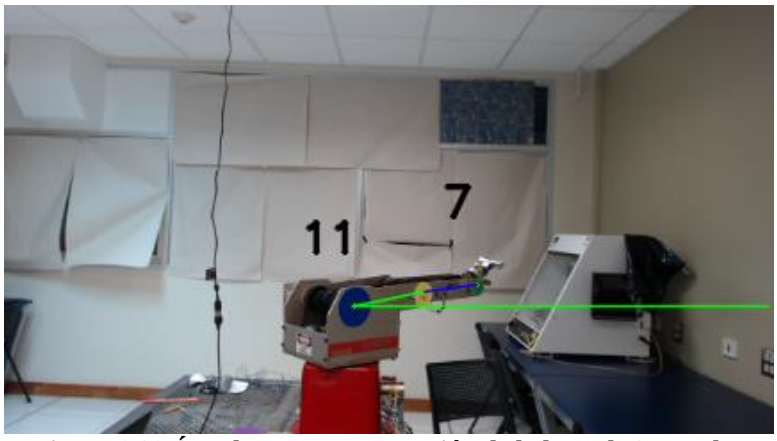


Figura 6.29 Ángulos para una rotación de la base de 25 grados
Fuente: Elaboración propia

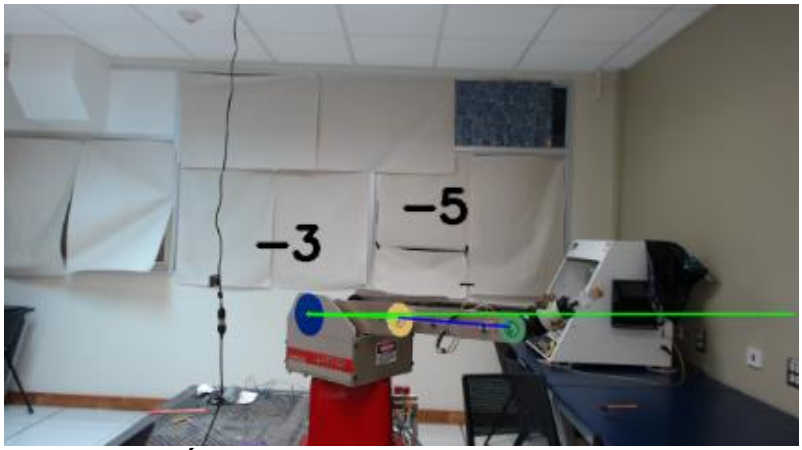


Figura 6.30 Ángulos para una rotación de la base de -25 grados
Fuente: Elaboración propia

Los Gráficos #1, #2 y #3 presentan un coeficiente de correlación lineal de 0.74, 0.84 y 0.94 respectivamente, esto indica que conforme menor se vuelve el ángulo de la articulación la variación que presenta a lo largo de las rotaciones se vuelve más lineal.

El análisis de la variación angular para rotaciones de base se realizó también para la sección del antebrazo.

Tabla 6.5 Variación de los ángulos del antebrazo determinados mediante código para diferentes rotaciones de base

Rotación (°)	Posición 1		Posición 2		Posición 3	
	Valor real	Antebrazo	Valor real	Antebrazo	Valor real	Antebrazo
25	82	86	90	92	1	10
20	82	85	90	92	1	7
15	82	83	90	92	1	5
10	82	83	90	92	1	4
5	82	83	90	91	1	3
0	82	82	90	90	1	1
-5	82	82	90	90	1	1
-10	82	82	90	89	1	0
-15	82	81	90	89	1	0
-20	82	81	90	89	1	-2
-25	82	81	90	88	1	-3

Fuente: Elaboración propia

Los datos de la posición uno y dos poseen menor variación que los de la posición tres debido a que se acercan más a ángulos rectos la posición de las marcas que percibe la cámara se ve menos expuesta a la variación de la profundidad en la rotación.

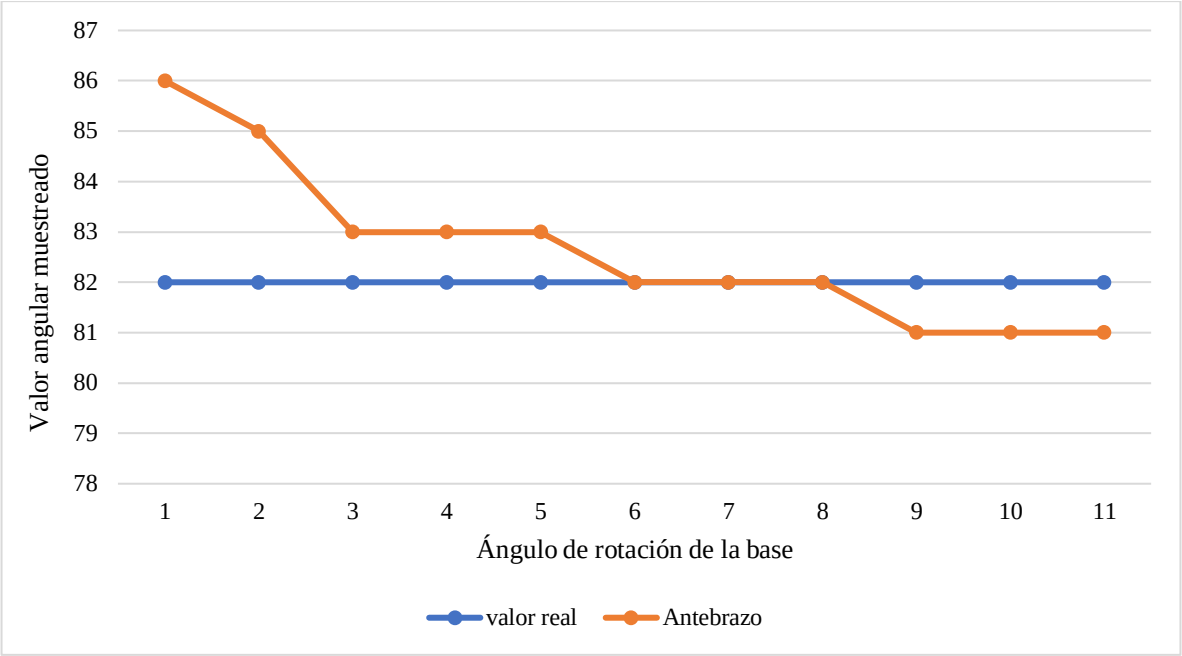


Figura 6.31 Variación de la determinación angular del antebrazo para la primera posición
Fuente: Elaboración propia

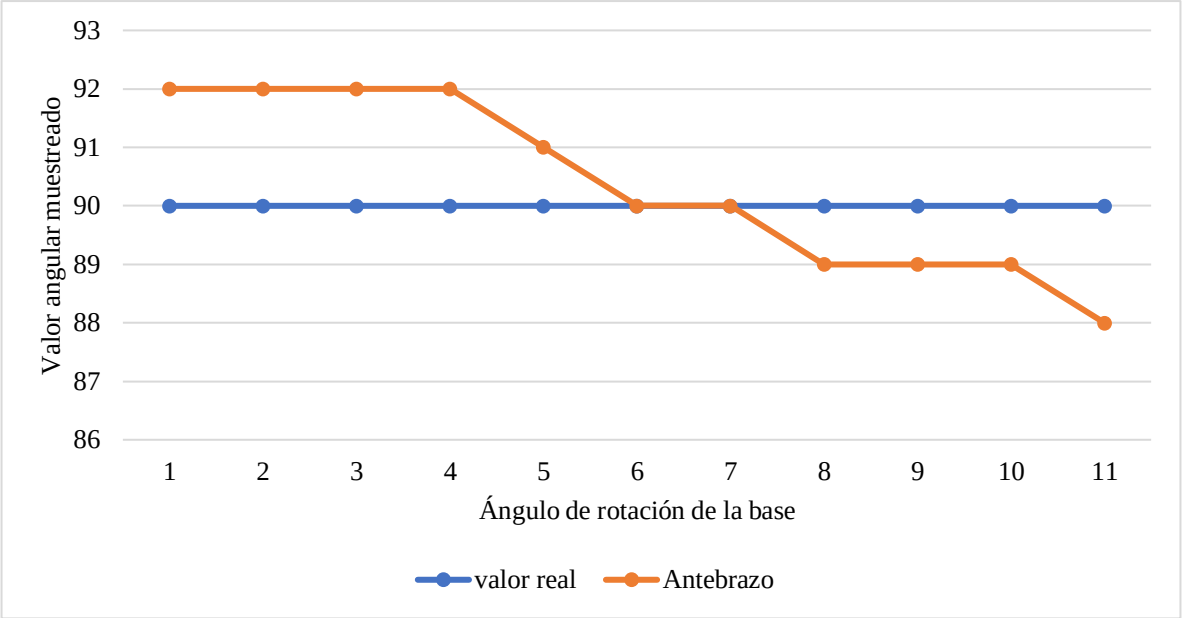


Figura 6.32 Variación de la determinación angular del antebrazo para la segunda posición
Fuente: Elaboración propia

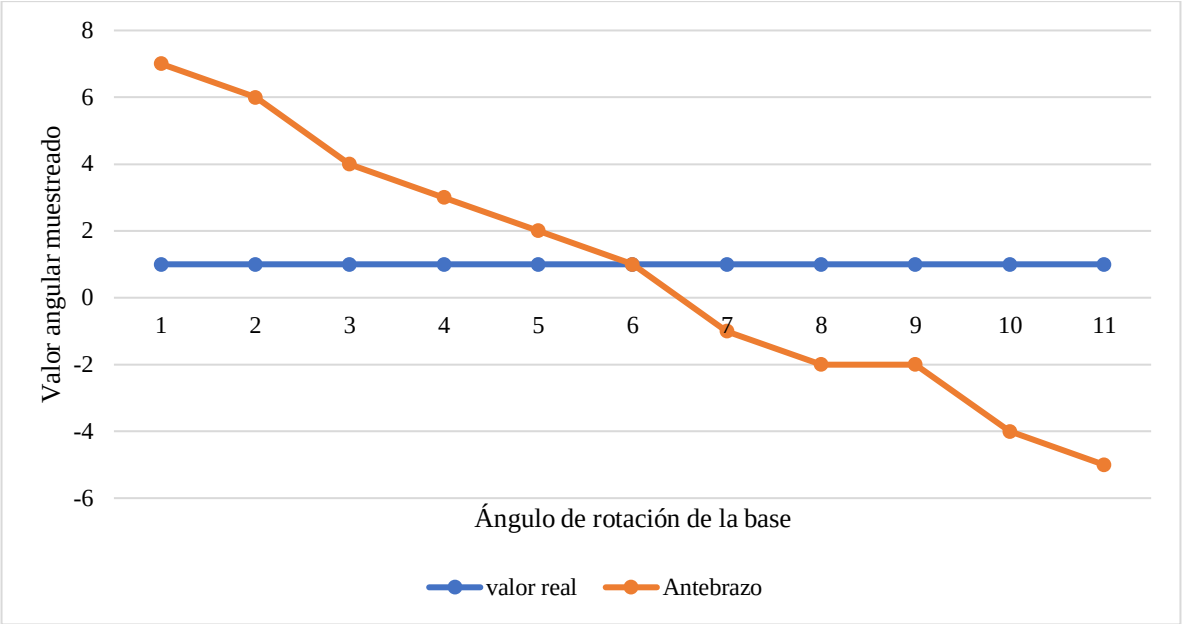


Figura 6.33 Variación de la determinación angular del antebrazo para la tercera posición
Fuente: Elaboración propia

Para las posiciones del antebrazo se puede observar que los ángulos de inclinación de articulaciones cercanos a ángulos rectos presentan menos variaciones frente a las distintas rotaciones, esto permite que la diferencia a máximas rotaciones de la base no sea tan alta, entre los ángulos de 80 y 92 se aprecia la estabilidad de las mediciones al aplicar las rotaciones, sin embargo, no se encuentra ausente de cambios.

Los gráficos cuatro, cinco y seis presenta un coeficiente de correlación lineal de 0.85, 0.92 y 0.94 respectivamente manteniendo la relación lineal entre el ángulo muestreado y el de rotación de la base.

Adicionalmente se grafica la posición del brazo y el antebrazo tomando una rotación de la base de 25 grados, con el objetivo de ver el comportamiento presente a los en todas las posiciones, se grafica el ángulo a cero grados contra su valor a 25 grados, esto para las tres posiciones presentadas en la Tabla 6.4 y Tabla 6.5

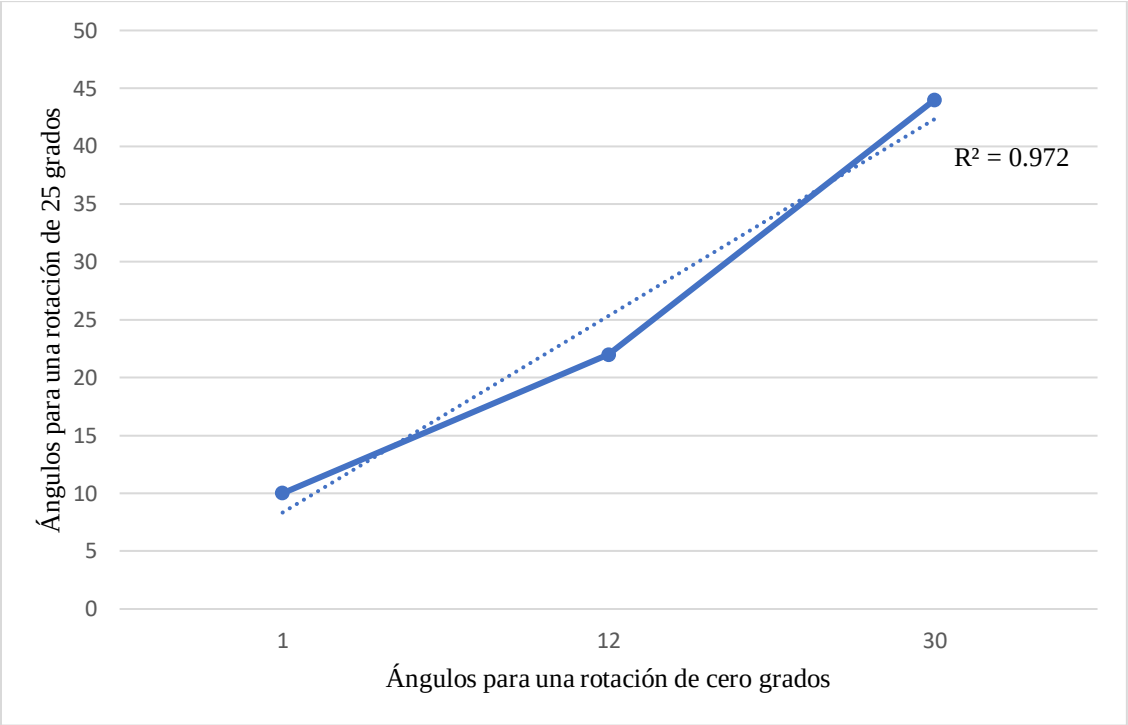


Figura 6.34 Relación de linealidad para posiciones del brazo
Fuente: Elaboración propia

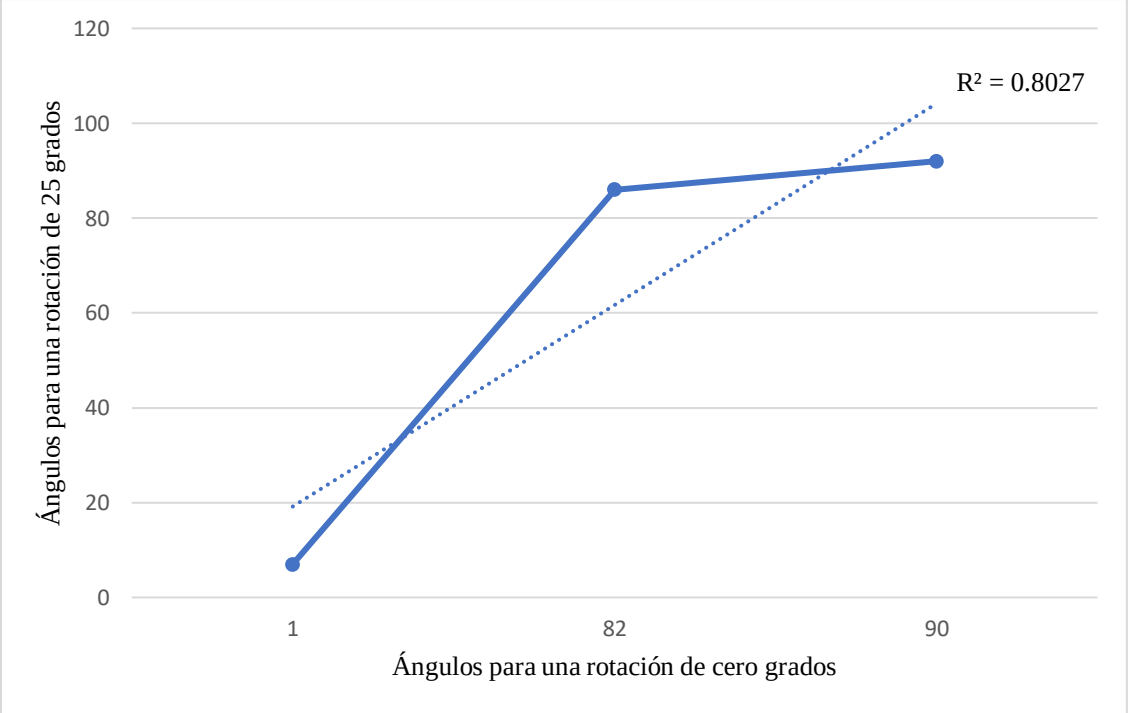


Figura 6.35 Relación de linealidad para las posiciones del antebrazo
Fuente: Elaboración propia

La Figura 6.35 presenta un menor coeficiente de relación debido a la diferencia entre los ángulos de las posiciones, que van des un grado a 82 y 90 donde el grado de 1 es el que presentaba mayor diferencia con respecto en su medición a 25 grados de rotación de base.

6.4. Análisis de estabilidad de detección

Adicionalmente a las mediciones angulares se implementó en el código la opción de calcular las áreas que detectan los contornos de las diferentes marcas de color, estas áreas se determinan en pixeles y sirven como una segunda fase de filtrado de color. Es posible calcular el área de los contornos en el punto más lejano y en el más cercano e implementar un rango de áreas de detección para que cada color, asi un área fuera de este rango, aunque este dentro del filtro de color será excluido.

Para esta prueba se tomó una gran cantidad de datos del área detectada a lo largo del tiempo, para las tres marcas de color y a las tres distancias de prueba establecidas, esto con el fin de representar gráficamente la estabilidad en la detección de los contornos en el tiempo.

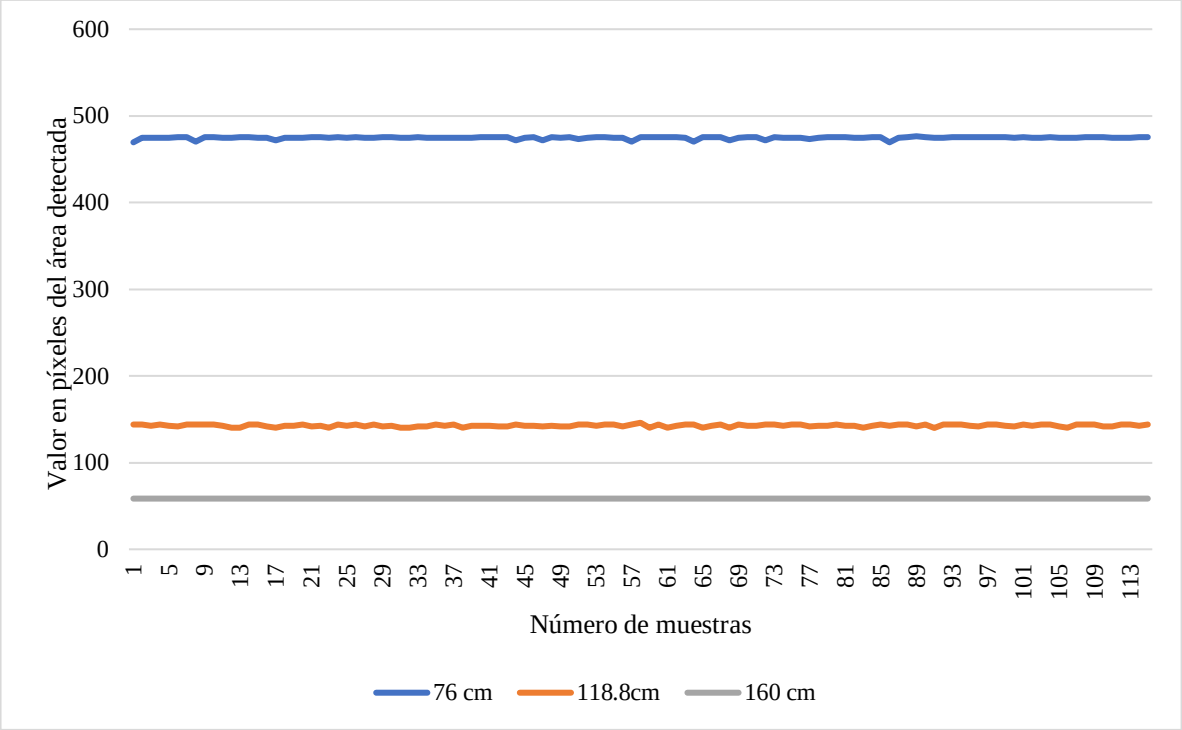


Figura 6.36 Variación del área amarilla
Fuente: Elaboración propia

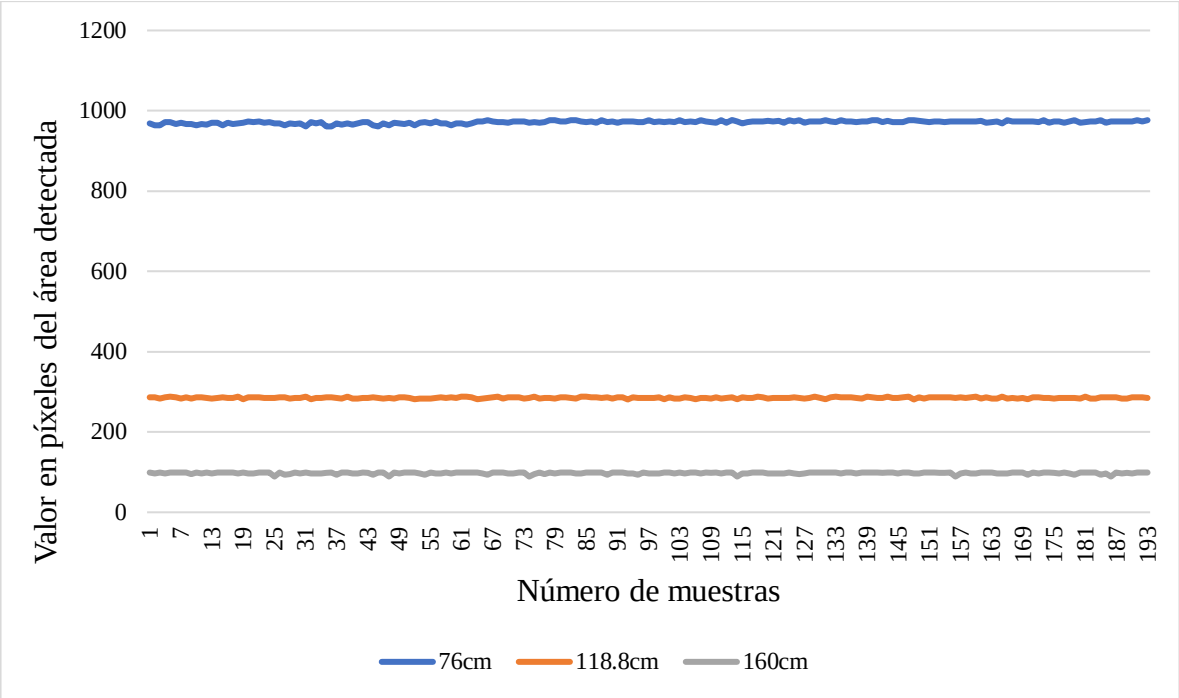


Figura 6.37 Variación del área azul
Fuente: Elaboración propia

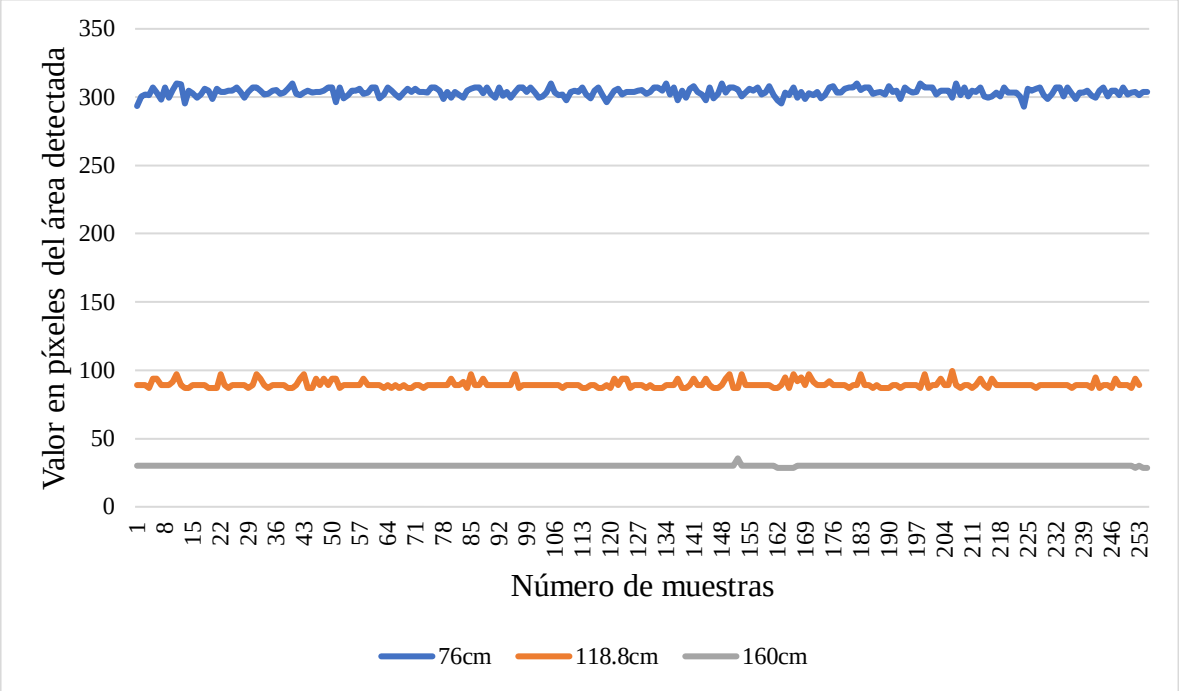


Figura 6.38 Variación del área verde
Fuente: Elaboración propia

Las figuras 6.36, 6.37 y 6.38 se puede observar la variación de las áreas amarillas, azul y verde respectivamente, y se nota que no existen cambios marcados a lo largo de las tomas,

las áreas que muestras una fluctuación un poco más visible son las áreas de la marca verde, sin embargo, la variación es mínima ya que se está trabajando con un área en pixeles y presenta cambios de entre 15 y 20 pixeles en un área de 300 o un máximos de diferencia de 10 pixeles para un área promediada en los 90 pixeles, además los pixeles no varían todos de forma conjunta en un solo sector, pueden variar esporádicamente en toda el área de detección por lo que se requiere una variación de un número significativo de pixeles para producir una deformación del contorno y que logre variar el centro de la marca y por ende el ángulo que se detecta.

Es importante establecer que estas mediciones representan la estabilidad del contorno y una variación muy significativa del área no implica estrictamente que el contorno se deformo puede significar que se filtró en la máscara una sección del mismo color con un área distinta y la cámara proceso esa área, sin embargo, estas variaciones no deben depreciarse, las variaciones en la detección del contorno combinadas con un desplazamiento del brazo durante la ejecución y con esto, una variación de la exposición a la luz pueden presentar que la detección del contorno pase de estar constante a mantenerse en una zona de variación donde el momento central varía entre dos pixeles, esto puede hacer que el ángulo detectado se encuentre variando entre 1 o 2 grados, la variación no sale del límite de trabajo pero debe tenerse presente este factor.

6.5. Ajuste de cámara superior

Durante la toma de pruebas con la cámara superior se determinó que el punto que se colocó en el centro del base para medir la rotación de la misma presentaba un desplazamiento cuando la base se desplaza de forma horizontal sobre el riel de trabajo, esto se debe a que la cámara superior esta fija y el punto colocado en el centro de la base se determinar a partir de una marca de color magenta utilizada como guía, a partir de esa marca el punto se desplaza hasta el centro de la base, sin embargo, cuando el brazo se mueve sobre el riel de trabajo la base se desplaza cerca de los límites del ángulo de visión de la cámara, y el punto central sigue determinándose por la marca magenta, la cámara desplaza en punto en una matriz de dos dimensiones y no es capaz de ajusta la inclinación que se da al desplazar el equipo

En la siguiente figura se muestra el punto azul en el centro de la base que se determina a partir de la marca de color magenta.

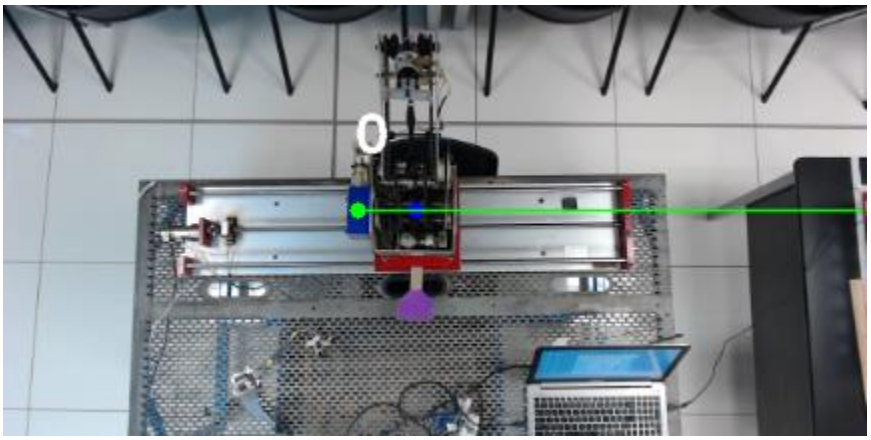


Figura 6.39 Captura de la cámara superior en la posición central del riel
Fuente: Elaboración propia

Al momento de desplazar la base hacia la izquierda el punto azul sufre un desplazamiento hacia la derecha, y al desplazar la base hacia a la derecha el punto azul sufre un desplazamiento hacia la izquierda.

En las siguientes figuras se muestra el desplazamiento del punto central al mover la base.

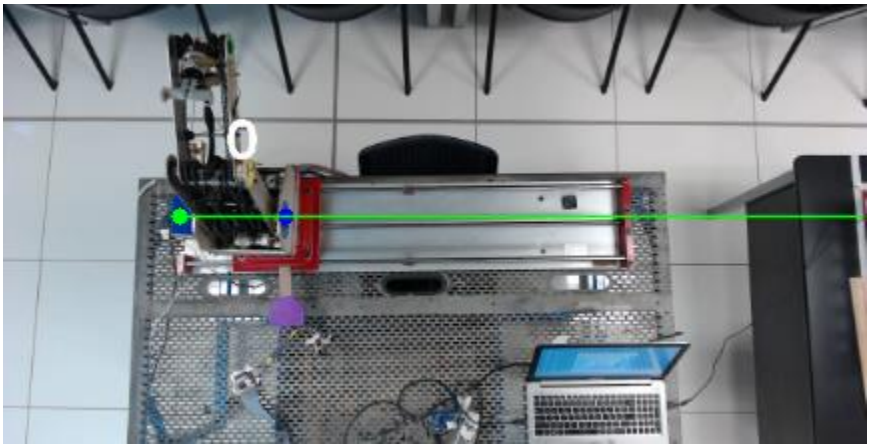


Figura 6.40 Captura de la cámara superior en la posición izquierda del riel
Fuente: Elaboración propia

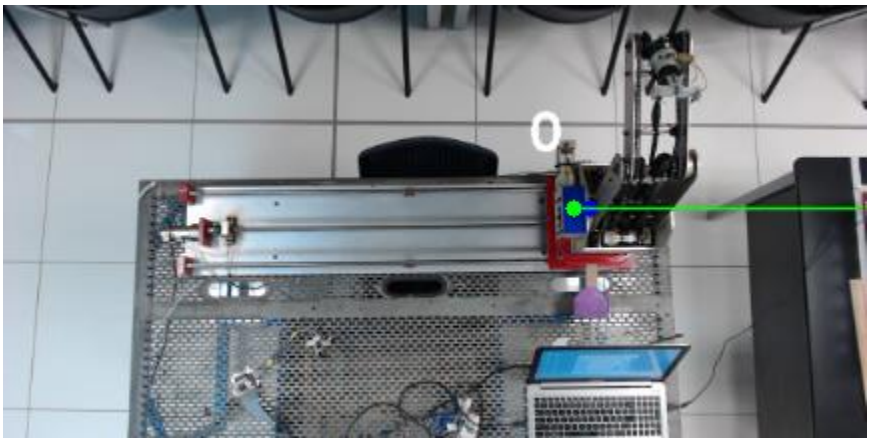


Figura 6.41 Captura de la cámara superior en la posición derecha del riel
Fuente: Elaboración propia

Debido a que la cámara superior no se encontraba directamente sobre el centro del riel del equipo el desplazamiento que sufre en pixeles el punto azul es diferente hacia la derecha y hacia la izquierda, por lo que, se realizó un ajuste para los pixeles de desplazamiento en ambas direcciones, para el movimiento hacia la izquierda el desplazamiento es de 73 pixeles, y la desviación del punto azul es de 17 pixeles y para la derecha el desplazamiento de es 95 pixeles y la desviación es de 16 pixeles por lo que se realiza mediante código un ajustes de 0.234 por pixel hacia la izquierda y 0.168 por pixel hacia la derecha.

En las siguientes figuras se presentan las capturas de la cámara superior en las diferentes posiciones luego de realizar el ajuste.

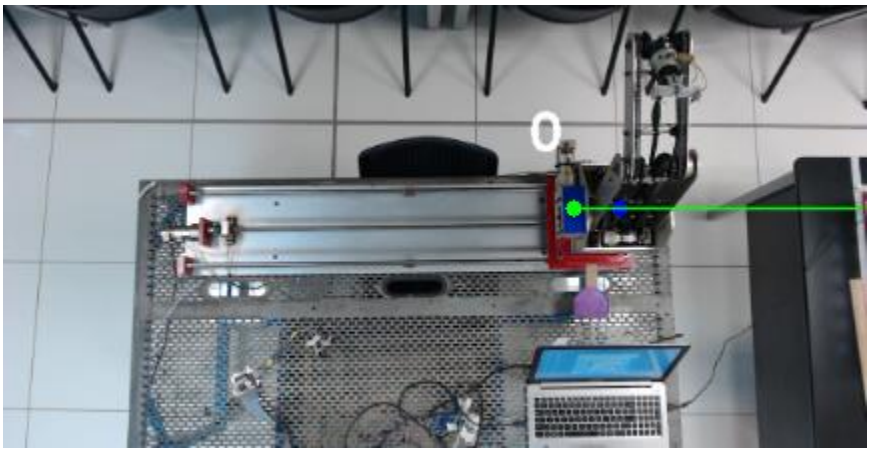


Figura 6.42 Captura de la cámara superior en la posición derecha luego del ajuste
Fuente: Elaboración propia

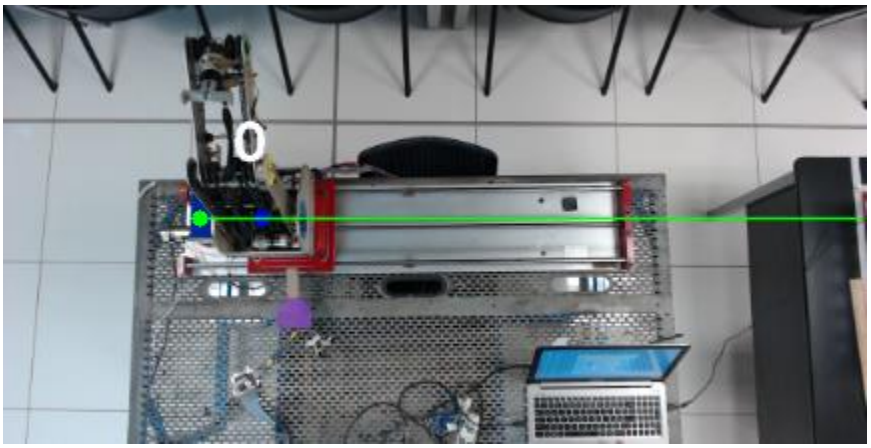


Figura 6.43 Captura de la cámara superior en la posición izquierda luego del ajuste
Fuente: Elaboración propia

6.6. Uso de recursos

Adicionalmente a los procesos de análisis correspondiente a las tomas de las cámaras se realizaron tomas del uso de recursos durante la ejecución del código.

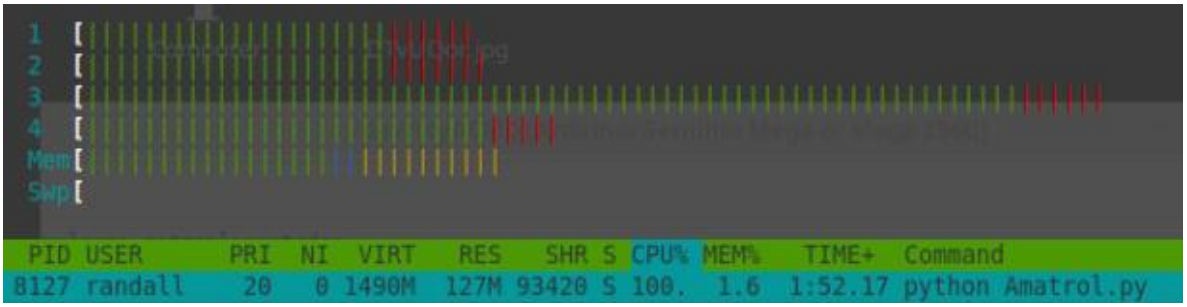


Figura 6.44 Recursos utilizados durante la ejecución del código
Fuente: Elaboración propia

En la Figura 6.44 se muestra el uso de los núcleos, memoria y porcentajes de uso del CPU durante la ejecución del código, la información se tomó nuevamente para otras dimensiones de matriz para determinar el aumento en el consumo de recursos.

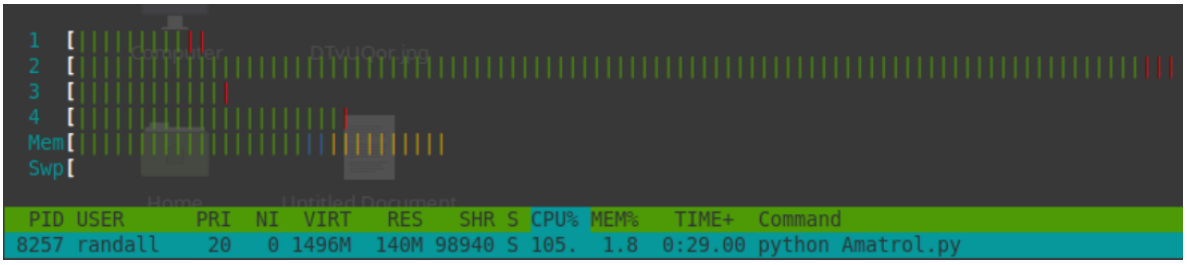


Figura 6.45 Recursos utilizados durante la ejecución del código para una matriz de 640x480 píxeles
Fuente: Elaboración propia

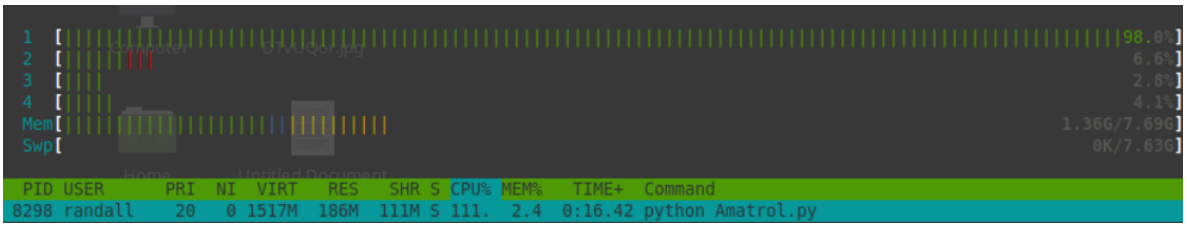


Figura 6.46 Recursos utilizados durante la ejecución del código para una matriz de 1280x720 píxeles
Fuente: Elaboración propia

El aumento de las matrices sobre las que se despliegan las capturas de las cámaras conlleva un aumento en el trabajo asignado al núcleo de operación que se puede apreciar en el porcentaje de trabajo del CPU, este incremente a 105% para la matriz de 640x480 y a 111% para la de 1280x720, el aumento por encima del 100% del porcentaje de uso del CPU se debe el porcentaje que se despliega para una sola unidad central de proceso, en sistema multinúcleo se puede tener porcentaje de CPU mayores al 100% lo que implica un saturación de un núcleo de trabajo.

6.7. Relación entre distancia y pixeles

Para la determinación entre la distancia física medida en centímetros y el desplazamiento representado en pixeles por la cámara se tomó un conjunto de muestras para dos posiciones de pixeles y se midió físicamente la distancia entre estos pixeles, de esta manera se estableció el valor aproximado en centímetros por píxel es de 0.39 centímetros por píxel para el desplazamiento horizontal, lo que estable que la matriz de trabajo cubre un área de 170.4 cm. Esta relación permite establecer la posición horizontal de desplazamiento del equipo mediante una unidad física, estableciendo previamente un punto de origen y una región de trabajo.

6.8. Pruebas de ejecución

Se tabularon prueba de ejecución correspondientes al código en operando, se tomaron los datos seleccionados por el usuario, los datos cálculos a través del código al final del desplazamiento y los datos físicos medidos para esos ángulos.

En la siguiente tabla se presentan los datos para la prueba de ejecución.

Tabla 6.6 Datos obtenidos de las pruebas de ejecución

Parámetro	Posición Seleccionada	Posición determinada	Medición física (° ± 0.5)
Posición en pixeles	130	131	N/A
Angulo de rotación de la base	0	0	N/A
Ángulo del brazo	30	32	32.5
Ángulo del antebrazo	30	32	32
Posición en pixeles	110	112	N/A
Angulo de rotación de la base	-20	-19	N/A
Ángulo del brazo	20	21	23
Ángulo del antebrazo	20	21	23
Posición en pixeles	150	148	N/A
Angulo de rotación de la base	0	0	N/A
Ángulo del brazo	40	40	40.5
Ángulo del antebrazo	20	21	20
Posición en pixeles	110	109	N/A
Angulo de rotación de la base	10	9	N/A
Ángulo del brazo	40	43	41
Ángulo del antebrazo	20	20	20

Fuente: Elaboración propia

En la Tabla 6.6 se presentan tres grupos de datos obtenidos de las pruebas de ejecución donde se aprecia la variación entre los ángulos determinados y los físicos para distintas rotaciones, es importante establecer que para estas pruebas se da el movimiento de todos los motores de manera simultanea para alcanzar la posición, esto implica que dependiendo de movimiento a realizar y la posición final puede darse menor o mayor variación entre el valor determinado por código y el medido físicamente, debido a que se puede dar la rotación antes o después de llevar las extremidades a los ángulos solicitados.

CAPITULO 7: CONCLUSIONES Y RECOMENDACIONES

7.1. Conclusiones

- Se desarrolló un algoritmo para determinación de posición y ángulos de inclinación de las extremidades del Pegasus I.
- Se determinó la distancia física por píxel para el desplazamiento horizontal de la base.
- Se implementó un algoritmo de comunicación y control retroalimentado para brindar control a las señales de los motores
- El procesamiento de imágenes mediante filtrado de color es afectado principalmente por las diferencias en exposición a la luz de cada color.
- Las dimensiones de la imagen procesada se ven limitadas por la tasa de trasmisión de datos del puerto de comunicación.
- La implementación de dos puntos de visión no permite realizar un control de todos los puntos de desplazamiento del Pegasus I.
- Los cambios de profundidad entre de las marcas de detección incurren en variaciones en la determinación del ángulo.

7.2. Recomendaciones

- Implementar un puerto de comunicación USB 3.1 o superior para mejor la tasa de transmisión y las resoluciones de trabajo.
- Utilizar el equipo en un entorno controlado en términos de exposición a la luz y colores del entorno.
- Implementar cámaras de doble lente para realizar el ajuste de profundidad o un sistema que permita la cámara lateral y superior rotar y desplazarse con la base.

CAPITULO 8: BIBLIOGRAFÍA

Alvarado Moya, J. P. (2012). *Procesamiento y Análisis de Imágenes Digitales.* Cartago, Costa Rica: Instituto Tecnológico de Costa Rica.

Amatrol. (12 de septiembre de 2018). *Moder control technology*. Obtenido de http://www.moderncontroltechnology.com/docs/Peggy_Robot/Vendor/Amatrol/RobotActivityModule.pdf

Arduino. (20 de Octubre de 2018). Obtenido de https://arduino.cl/arduino-mega-2560/

Atmel Corporation. (11 de Noviembre de 2018). *Microchip.* Obtenido de http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-2549-8-bit-AVR-Microcontroller-ATmega640-1280-1281-2560-2561_datasheet.pdf

Cánepa, V. M., Ferrari Bihurriet, F., & Picard, A. (2014). *Comunicación Serie.* Buenos Aires: Universidad de Buenos Aires.

Crespo Armaiz, J. (2016). *La primera sociedad mediática fotografía, estereoscopía y el nuevo orden visual.* Ciudad de Mexico : Lulu.

Gerrero Hernández, J. M., Pajares Martisanz, G., & Mata García, M. G. (2014). *Técnicas de procesamiento de imágenes estereoscópicas.* Madrid: Universidad Complutense de Madrid.

Guerrero, R. (2015). *Teoría del color.* San Luis, Argentina: Universidad Nacional de Argentina.

Logitech. (10 de Noviembre de 2018). Obtenido de https://www.logitech.com/es-es/product/hd-pro-webcam-c920

Martinez Rueda, J. (2006). *Sistemas Eléctricos y electrónicos de la aeronaves.* Madrid: Paraninfo.

Negrea , A., Szabó, C., & Imecs, M. (2010). *Incremental Encoder based position and speed identification: Modeling and simulation.* Cluj-Napoca.

Nylampmechatronics. (14 de noviembre de 2018). Obtenido de https://naylampmechatronics.com/drivers/11-driver-puente-h-l298n.html

Pincuico. (1 de Diciembre de 2018). Obtenido de https://www.picuino.com/es/arduprog/control-pid.html

Quizlet. (1 de Noviembre de 2018). Obtenido de https://quizlet.com/31893528/chapter-16-instruction-level-parallelism-and-superscalar-processors-flash-cards/

Sparfunk Electronics. (15 de Noviembre de 2018). Obtenido de https://www.sparkfun.com/datasheets/Robotics/L298_H_Bridge.pdf

Tensorflow. (1 de 12 de 2018). Obtenido de https://www.tensorflow.org/?hl=es

Apéndice 1. Algoritmo

```
#!/usr/bin/env/python3.5
#OpenCV 3.4.1
#Date 20th January, 2018
#Discription : Measuring angle
import os
import serial
import syslog
import time
from threading import Thread
import cv2
import numpy as np
import math
import sys

status = os.system('/bin/true') #
port = '/dev/ttyACM0' # puerto de arduino
ard = serial.Serial(port,230400,timeout=1) # parametros del puerto arduino
nombre,velocidad y tiempo de espera
port2= '/dev/ttyUSB0'# parametros del puerto usb nombre,velocidad y tiempo de espera
ser = serial.Serial(port2,115200,timeout=2)
time.sleep(3)


##////////////////////////////////////////////Definicion de diferentes resoluciones de
uso//////////////////////////////////////////////////////
#def make_1080p():
```

```
#       cap.set(3, 1920)
#       cap.set(4, 1080)
#def make_720p():
#       cap.set(3, 1280)
#       cap.set(4, 720)
##////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
///////////////
cap = cv2.VideoCapture(1) # puerto usb 1 para la camara lateral
cap2 = cv2.VideoCapture(2) # puerto usb 2 para la camara lateral
# ///////////////// los puerto usb se definen en el orden en que se conecten las
camaras//////////////////////////////////


#////////////////////////////resolución de trabajo utilizada/////////////////////////////////////////////////////////
def make_426p():
        global cap
        global cap2


        cap.set(3, 426)
        cap.set(4, 240)
        cap2.set(3, 426)
        cap2.set(4, 240)
make_426p()
#////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/////////////////


###////////////////////////// PARAMETROS INICIALES//////////////////////////////////
```

```
font = cv2.FONT_HERSHEY_SIMPLEX
height = 640
width  = 360
j=""
salida=0
M=0
des=0
ab=0
angle=0
angle2=0
b1=0
b2=0
b3=0
b4=0
salido=0
mov=0
base=0
ante=0
brazo=0
dead= False
nuevo = 0
begin =0
fin=0
empezar_comunicacion=1
lp=0
lr=0
lb=0
la=0
```

```
#////////////////////////////////////////////////////////Hilo de ejecución de las
cámaras//////////////////////////////////////////
def camaras():
#////////////////////////////////////////////////variables globales//////////////////////////////////////////////////
        global fin
        global b1
        global b2
        global b3
        global b4

        global des
        global ab
        global angle
        global angle2
        global salida
        global mov
        global base
        global ante
        global brazo
        global j
        global dead
        global cap
        global cap2
        global ret0
        global ret1
        global frame
        global frame2
```

```
        global c
        global nuevo
#////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
        while True:
                ret0, frame = cap.read()
                ret1, frame2 = cap2.read()
                # lectura de las capturas


                #proceso para la camara lateral
                if(ret0):
        #////////////////////////////////////////////Marca Azul/////////////////////////////////////
                        # conversion a espacio de color HSV
                        hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
                        mask = cv2.inRange(hsv, np.array((107,40,40 )), np.array((125,255,
255))) #mascara con rango de colo azul
                        kernel1 = np.ones((5,5), np.uint8)#definicion del kernel utilizado en
morfologias
                        opening= cv2.morphologyEx(mask, cv2.MORPH_OPEN,kernel1)
        #morfologia de apertura
                        median = cv2.medianBlur(opening,15)#suavizado


                        # encontrar los contornos
                        _, contours, hierarchy = cv2.findContours(median, cv2.RETR_LIST,
cv2.CHAIN_APPROX_SIMPLE)


                        # los contornos se almacenan en best_cnt por que corresponden a
arreglos de pixeles
                        #max_area = 0    , la variable max_area se puede utilizar para tener
un area maximasobre la cual definir el contorno
                        for cnt in contours:
```

```
                area = cv2.contourArea(cnt)# retorna el area en pixeles dentro del contorno, se debe imprimir el area para determinar los paremetros de area_minima y area_maxima

                if  0 < area:   # la condicion del if se dejo abierta para cualquier area detectada, para definir el filtro por area se sustituye la condicion del if:  if area_minima < area < area_maxima

                        best_cnt=cnt


        #si no se obtiene un contorno se hace un cierre de las ventanas y salida del codigo

        if 'best_cnt' in dir():

                salida=0

        else:

                print('no hay deteccion azul para la camara lateral')

                cv2.destroyAllWindows()

                cap.release()

                cap2.release()

                salida=1;

                raise SystemExit

                exit()


        # se encuentran los centroides de best_cnt y se dibuja un circulo en este centroide

        M = cv2.moments(best_cnt)

        cx, cy = int(M['m10']/M['m00']), int(M['m01']/M['m00'])

        #cv2.putText(frame, str(cx) + "," + str(cy), (cx, cy + 20), font, 1,(255,255,0), 2, cv2.LINE_AA) #Esta linea escribe el texto de los pixeles x y de posicion del centroide

        cv2.circle(frame,(cx, cy), 1,(0, 255, 0), 1) # dibuja el circulo

        cv2.line(frame,(cx, cy),((height, width)[0], cy),(0, 255, 0), 1, cv2.LINE_AA)# dibuja la linea horizontal de referencia correspondiente al centroide
```

```
                    #////////////////////////////////////////////Marca
Amarilla////////////////////////////////////////////////////////////////////////////
                    mask2= cv2.inRange(hsv, np.array((15, 58,58)), np.array((34, 255,
255)))
                    opening2= cv2.morphologyEx(mask2, cv2.MORPH_OPEN,kernel1)
                    median2 = cv2.medianBlur(opening2,11)


                    # se encuentran los contornos en el umbral amarillo
                    _, contours4,hierarchy = cv2.findContours(median2,
cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)


                    # se encuentran los contornos para el rango de areas
                    max_area2 = 0
                    #best_cnt2 = 0
                    for cnt in contours4:
                        area2 = cv2.contourArea(cnt)
                        if area2 > 0:


                            best_cnt2 = cnt
                    if 'best_cnt2' in dir():
                        salida=0
                    else:
                        print('no hay deteccion amarila para la camara lateral')
                        cv2.destroyAllWindows()
                        cap.release()
                        cap2.release()
```

```
                    salida=1;
                    raise SystemExit



                # se enceuntra los corntornos y se dibuja el circulo
                M2 = cv2.moments(best_cnt2)
                cx2, cy2 = int(M2['m10'] / M2['m00']), int(M2['m01'] / M2['m00'])
                #cv2.putText(frame, str(cx2) + "," + str(cy2), (cx2, cy2 + 20), font,
1,(255, 255,0), 2, cv2.LINE_AA)
                cv2.circle(frame,(cx2, cy2), 1, (0, 255, 0), 1)
                cv2.line(frame,(cx, cy), (cx2, cy2),(0, 255, 0), 1, cv2.LINE_AA) #
esta linea conecta los centroides de la marca azul con los del la marca amarilla



                #////////////////////////////////////////////////Marca
Verde////////////////////////////////////////////////////////////////////////////


                mask3 = cv2.inRange(hsv, np.array((40,5,5 )), np.array((80,255,
255)))


                opening3= cv2.morphologyEx(mask3, cv2.MORPH_OPEN,kernel1)
                median3 = cv2.medianBlur(opening3,9)


                # se encuentran los contornos en el umnbral verde
                _, contours6, hierarchy = cv2.findContours(median3,
cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)


                # se definen las ares de los contornos y se almacenan en best_cnt3
                max_area3 = 0
```

```
for cnt in contours6:
        area3 = cv2.contourArea(cnt)
        if area3 > max_area3:
            max_area3 = area3
            best_cnt3 = cnt



if 'best_cnt3' in dir():
        salida=0
else:
        print('no hay deteccion verde para la camara lateral')
        cv2.destroyAllWindows()
        cap.release()
        cap2.release()
        salida=1;
        raise SystemExit



        # se encuentran los contornos y se dibuja en circulo
        M3 = cv2.moments(best_cnt3)
        cx3, cy3 = int(M3['m10'] / M3['m00']), int(M3['m01'] / M3['m00'])
        #cv2.putText(frame, str(cx3) + "," + str(cy3), (cx3, cy3 + 20), font,
1,(255,255,0), 2, cv2.LINE_AA)
        cv2.circle(frame,(cx3, cy3), 1,(0, 255, 0), 1)
        cv2.line(frame,(cx2, cy2), (cx3, cy3),(255, 0, 0), 1, cv2.LINE_AA)#
esta linea conecta los centroides de la marca amarilla con los del la marca verde
        #/////
```

```
                    #los centroides se convierten a valores de punto flotante
                    cx = float(cx)
                    cy = float(cy)
                    cx2 = float(cx2)
                    cy2 = float(cy2)
                    cx3 = float(cx3)
                    cy3 = float(cy3)
                    angle = int(math.atan2((cy - cy2), (cx2 - cx)) * 180 // math.pi) #se
calcula el angulo entre azul y amarillo
                    angle2 = int(math.atan2((cy2 - cy3), (cx3 - cx2)) * 180 // math.pi) #se
calcula el angulo entre amarillo y verde



                    #cv2.putText(frame, str(angle),(int(cx2) - 10, (int(cy2) + int(cy) +
50) // 2), font, 1, (255, 255, 0), 2,  cv2.LINE_AA)
                    cv2.putText(frame, str(angle), (int(cx) - 30, int(cy) - 30 ), font, 0.8,
(0, 0, 0), 2, cv2.LINE_AA) # se imprime el valor del angulo del brazo en pantalla
                    cv2.putText(frame, str(angle2), (int(cx) + 50, int(cy) - 30 ), font, 0.8,
(0, 0, 0), 2, cv2.LINE_AA) # se imprime el valor del angulo del antebrazo en pantalla



                    # se despliega la variable frame que corresponde a la camara lateral
                    cv2.imshow('Measuring Angle', frame)
                    #si desea desplazar cualquiera de las morfologias o suavizados se
utilizan esta funciones, simplementa se cambia la variable a desplegar
                    # cv2.imshow('median ', median)
                    #cv2.imshow('median2 ', median2)
```

```
                    #cv2.imshow('median3 ', median3)


                    # si se presiona ESC se sale del ciclo
                    c = cv2.waitKey(1) % 0x100
                    if c == 27 or c == 10:
                        break
                    #proceso para la camara superior
                if(ret1):
                    # se convierte la imagen a HSV
                    hsv=cv2.cvtColor(frame2, cv2.COLOR_BGR2HSV)
#////////////////////////////////////////////Marca Azul
superior/////////////////////////////////////////////////////////////////////////////
                    mask4 = cv2.inRange(hsv, np.array((107,30,40 )), np.array((125,255,
255)))
                    kernel1 = np.ones((5,5), np.uint8)
                    opening4= cv2.morphologyEx(mask4, cv2.MORPH_OPEN,kernel1)
                    median4 = cv2.medianBlur(opening4,15)


                    # contornos para el umbral azul
                    _, contours8, hierarchy = cv2.findContours(median4,
cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)


                    # se almacenan contornos en best_cnt4
                    max_area4 = 0


                    for cnt in contours8:
                        area4 = cv2.contourArea(cnt)
                        if area4 > max_area4:
                           max_area4 = area4
```

```
                    best_cnt4 = cnt

# se cierra el proceso si no se detecta el color
                if 'best_cnt4' in dir():
                    salida=0
                else:
                    print('no hay deteccion azul para la camara superior')
                    cv2.destroyAllWindows()
                    cap2.release()
                    cap.release()
                    salida=1;
                    raise SystemExit

                M4 = cv2.moments(best_cnt4)
                cx4, cy4 = int(M4['m10'] / M4['m00']), int(M4['m01'] / M4['m00'])

                cv2.circle(frame2,(cx4, cy4), 1,(0, 255, 0), 5)


                #////////////////////////////////////////////////Marca magenta
superior////////////////////////////////////////////////////////////////////////////////

                mask5 = cv2.inRange(hsv, np.array((125,5,5 )), np.array((170,255,
255)))
                opening5= cv2.morphologyEx(mask5, cv2.MORPH_OPEN,kernel1)
                median5 = cv2.medianBlur(opening5,15)

                # contornos para el umbral magenta
```

```
                    _, contours10, hierarchy = cv2.findContours(median5,
cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)

                    # se definen los contornos en best_cnt5
                    max_area5 = 0

                    for cnt in contours10:
                        area5 = cv2.contourArea(cnt)
                        if area5 > max_area5:
                            max_area5 = area5
                            best_cnt5 = cnt



                    if 'best_cnt5' in dir():
                        salida=0
                    else:
                        print('no')
                        cv2.destroyAllWindows()
                        cap2.release()
                        cap.release()
                        salida=1;
                        raise SystemExit
                    #se determina el centroide
                    M5 = cv2.moments(best_cnt5)
                    cx5, cy5 = int(M5['m10'] / M5['m00']), int(M5['m01'] / M5['m00'])
                    cv2.putText(frame2, str(cx5-115) + "," + str(cy5), (cx5, cy5 + 20),
font, 1,(255,255,0), 2, cv2.LINE_AA)
                    # se realiza el ajuste del centroide de la marca magenta
                    des=cx5-115
```

```
                cyc=cy5-47
                if(cx5<205):
                        cxa=int(round(205-(205-cx5)*1.23287671))
                if(cx5==205):
                        cxa=cx5
                if(cx5>205):
                        cxa=int(round(205+(-(205-cx5)*1.16842105)))
                #se pinta el circulo y la linea entre la marca azul y la magenta
                cv2.circle(frame2,(cxa, cyc), 1,(255, 0, 0), 5)
                cv2.line(frame2,(cxa, cyc),((height, width)[0], cyc),(0, 255, 1), 1,
cv2.LINE_AA)
                cv2.line(frame2,(cx4, cy4), (cxa, cyc),(0, 255, 0), 1, cv2.LINE_AA)
        # los centroides se convierten a punto flotante

                cx4 = float(cx4)
                cy4 = float(cy4)
                cx5 = float(cx5)
                cy5 = float(cy5)
                cxa = float(cxa)
                ab = int(math.atan2((cy4 - cyc), (cxa - cx4)) * 180 // math.pi)
                cv2.putText(frame2, str(ab), (int(cx5) - 30, int(cyc) - 30 ), font, 0.8,
(255, 255, 255), 2, cv2.LINE_AA)
                # se despliega la imagen de la camara
                cv2.imshow('Measuring Angle2', frame2)
                #se cierra el ciclo si se presiona ESC
                c = cv2.waitKey(1) % 0x100
                if c == 27 or c == 10:
                        break
```

```
        # Se limpian las ventanas y se liberan las camaras si se cierra el ciclo
        cv2.destroyAllWindows()
        cap.release()
        cap2.release()
        salida=1
# se inicia el hilo de las camaras
tc = Thread(target=camaras, args=())
tc.start()
#////////////////////////////////////////////////////////////////////////////Hilo de
lectura//////////////////////////////////////////
def leer():
        #//////////////////////////////////////////variables
globales////////////////////////////////////////////////////////////////
        global fin
        global b1
        global b2
        global b3
        global b4
        global begin
        global des
        global ab
        global angle
        global angle2
        global salida
        global mov
        global base
        global ante
```

```
        global brazo
        global j
        global dead
        global cap
        global cap2
        global ret0
        global ret1
        global frame
        global frame2
        global c
        global nuevo

        # si existe salida en alguno de los otros hilos se cierra el proceso
        if(salida==1):
                raise SystemExit
                exit()
        time.sleep(20)
        #se inicia el cilo de lectura de constante del mensaje "listo" enviado por el arduino
        while True:
                ser_line=ser.read_until("o")


                time.sleep(0.1)
                ser.flush()
                #si se recibe el mensaje listo correctamente se suma al contador de
verificación
                if(ser_line=="listo"):


                        fin=fin+1
                        #main()
```

```
# se inicia el hilo de lectura
tl = Thread(target=leer, args=())
tl.start()
#////////////////////////////////////////// Hilo de escritura de informacion ////////////////////////////////
def arduino():
        global fin
        global nuevo
        global dead
        global b1
        global b2
        global b3
        global b4

        global des
        global ab
        global angle
        global angle2
        global salida
        global mov
        global base
        global ante
        global brazo
        global j
        global cap
        global cap2
        global ret0
```

```
global ret1
global frame
global frame2
global c


M=0
#si hay una salida en algun hilo externo se cierra en proceso
if(salida==1):
        raise SystemExit
        exit()
#se crea un buffer donde se almacenan los datos de posicion deseada como un solo mensaje de texto se parados por *
buff = '+'+str(mov)+'*'+str(base)+'*'+str(ante)+'*'+str(brazo)+'*'+'/'
#se escribe a traves del puerto de arduino los datos deseados una vez
ard.write(buff)
ard.flush()
time.sleep(1)
#se inicia el ciclo de escritura de las camaras
while True:
    #la infromacion de las camaras se agrupa como un solo mensaje de texto y se escribe hacia el arduino
    buff1 =str(des)+'*'+str(ab)+'*'+str(angle)+'*'+str(angle2)+'*'+'/'
    ard.write(buff1)
    time.sleep(0.05)
    ard.flush()


    #si presenta una salida durante el ciclo de escritura se cierra el proceso
    if(salida==1):
        raise SystemExit
```

```
                exit()
            #si durante el envio de datos de las camaras el arduino alcanza la posicion y el contador de verificacion se completa se retorna a pedir un grupo de datos iniciales
            if (fin==100):
                fin=0
                ser.flush()
                return()



    exit()
# se inicia el hilo de arduino
ta = Thread(target=arduino, args=())
ta.start()
#//////////////////////////////////////Funcion para solicitar los datos/////////////////////////////////////////////
def main():
    #la solicitud de datos se encicla
    while True:
        #variables globales
        global fin
        global nuevo
        global ret0
        global ret1
        global frame
        global frame2
        global b1
        global b2
        global b3
        global b4
```

```
global des
global ab
global angle
global angle2
global salida
global mov
global base
global ante
global brazo
global j
global cap
global cap2
global c
global empezar_comunicacion
global begin
global lp
global lr
global lb
global la
# si se da una salida en un hilo externo se cierra el sistema
if(salida==1):
        raise SystemExit
        exit()


while lp=0:
        mov= input('\n Indique el desplazamiento de la base (rango de
operacion 0-145): ')
        if (0 < mov < 145):
```

```
                    lp=1
                else:
                    print('rango de operacion invalido')
            while lr=0:
                base= input('Indique el angulo de rotacion de la base (rango de
operacion -25 hasta 25 grados): ')
                if(-25 < base < 25):
                    lr=1
                else:
                    print('rango de operacion invalido')
            while lb =0:
                ante= input('Indique el angulo de rotacion del brazo (rango de 0 a 90
grados): ')
                if( 0 < ante < 90):
                    lb=1
                else:
                    print('rango de operacion invalido')
            while la=0:
                brazo= input('Indique el angulo de rotacion del antebrazo (rango de 0
a 90 grados): ')
                if (0 < brazo < 90):
                    la=1
                else:
                    print('rango de operacion invalido')
            fin=0 # cada vez que se da una posicion se inicia el contador de verificaciond
e lectura en 0
            arduino() # cuando se toman los datos se llama a la funcion de escritura para
iniciar el envio


            if(salida==1):
```

```
                raise SystemExit
                exit()



main()
```

Código de Arduino

```
int setpoint = 55;
String readString;
String des;
String ab;
String a1;
String a2;
String angle;
String angle2;
String maximo;
char buff[30]={};
int a=0;
int b=0;
int c=0;
int d=0;
int q=0;
int w=0;
```

```
int e=0;
int r=0;
int z=0;
int x=0;
int n=0;
int v=0;
const int motorAH =31 ;
const int motorAA = 32;
int ENA=33;
const int motorBH = 44;
const int motorBA = 48;
int ENB=42;
const int motorEH = 23;
const int motorEA = 24;
int ENE=22;
const int motorFD= 26;
const int motorFI = 25;
int ENF=27;
int maxi=0;
int mini=0;
int rmax=0;
int rmin=0;
int bb= 0;
volatile boolean b1= false;
volatile boolean b2 = false;
volatile boolean b3= false;
volatile boolean b4= false;
int be= 0;
```

```
int bf=0;
int ba= 0;
int listo=0;
int o=1;

void setup()
{//iniciar la comunicacion en serie
  Serial.begin(230400);
  Serial.setTimeout(50);
  Serial1.begin(115200);
  Serial1.setTimeout(100);

  // se definene los pines de salida de los motores

   pinMode(motorAH , OUTPUT);
   pinMode(motorAA , OUTPUT);
   pinMode(motorBH , OUTPUT);
   pinMode(motorBA , OUTPUT);
   pinMode(motorAH , OUTPUT);
   pinMode(motorFD , OUTPUT);
   pinMode(motorFI , OUTPUT);
   pinMode(motorEA , OUTPUT);
   pinMode(ENA , OUTPUT);
   pinMode(ENB , OUTPUT);
   pinMode(ENE , OUTPUT);
   pinMode(ENF , OUTPUT);
   analogWrite(ENB,0);
   analogWrite(ENE,0);
```

```
  analogWrite(ENA,0);
  analogWrite(ENF,0);


}

void loop()
{


   //lee y separa los valores solicitados por el usuario
   while (0< Serial.available())
   {

    Serial.readBytesUntil('/',buff,30);



    if (buff[0]=='+'){
       for (a=1;buff[a]!='*';a++){
          des+=buff[a];
        }
       for (b=a+1;buff[b]!='*';b++){
          ab+=buff[b];
        }
       for (c=b+1;buff[c]!='*';c++){
          a1+=buff[c];
        }
```

```
    for (d=c+1;buff[d]!='*';d++){
       a2+=buff[d];
     }
    // convierte los valores a enteros y los imprime en el monitor serial
    q = des.toInt();
    w = ab.toInt();
    e = a1.toInt();
    r = a2.toInt();
    Serial.print("valores iniciales \n");
    Serial.print(q); //see what was received
    Serial.print(" \n");
    Serial.print(w); //see what was received
    Serial.print(" \n");
    Serial.print(e); //see what was received
    Serial.print(" \n");
    Serial.print(r); //see what was received
    Serial.print(" \n");

    des="";
    ab="";
    a1="";
    a2="";
    char buff[30]={};


  }

/// separa los valores de las camaras y los convierte a enteros
```

```
  for (a=0;buff[a]!='*';a++){
  des+=buff[a];
  }

for (b=a+1;buff[b]!='*';b++){
   ab+=buff[b];
  }

for (c=b+1;buff[c]!='*';c++){
   a1+=buff[c];
  }

for (d=c+1;buff[d]!='*';d++){
   a2+=buff[d];
  }



  z = des.toInt();
  x = ab.toInt();
  n = a1.toInt();
  v = a2.toInt();


  //compara la posicion de las camaras con la posicion solicitada y activa las señales de control de cada motor motor B es brazo, motor E antebrazo, motor A es rotacion de la base y motor F es desplazamiento
  if(v>r+5){
    digitalWrite(motorBH, LOW);
```

```
  digitalWrite(motorBA, HIGH);
  analogWrite(ENB,128);

  }
else if(v<r-5){
  digitalWrite(motorBH, HIGH);
  digitalWrite(motorBA, LOW);
  analogWrite(ENB,128);

  }
else{
  digitalWrite(motorBH, LOW);
  digitalWrite(motorBA, LOW);
  analogWrite(ENB,50);
    b1=true;

  }
 if(n>e+5){
  digitalWrite(motorEH, HIGH);
  digitalWrite(motorEA, LOW);
  analogWrite(ENE,200);

  }
else if(n<e-5){
  digitalWrite(motorEH, LOW);
  digitalWrite(motorEA, HIGH);
  analogWrite(ENE,200);
```

```
  }
else {
  digitalWrite(motorEH, LOW);
  digitalWrite(motorEA, LOW);
  analogWrite(ENE,50);
    b2=true;



  }
if(x>w+4){
  digitalWrite(motorAH, HIGH);
  digitalWrite(motorAA, LOW);
  analogWrite(ENA,250);

  }
else if(x<w-4){
  digitalWrite(motorAH, LOW);
  digitalWrite(motorAA, HIGH);
  analogWrite(ENA,250);

  }
else{
  digitalWrite(motorAH, LOW);
  digitalWrite(motorAA, LOW);
  analogWrite(ENA,50);
    b3=true;

  }
```

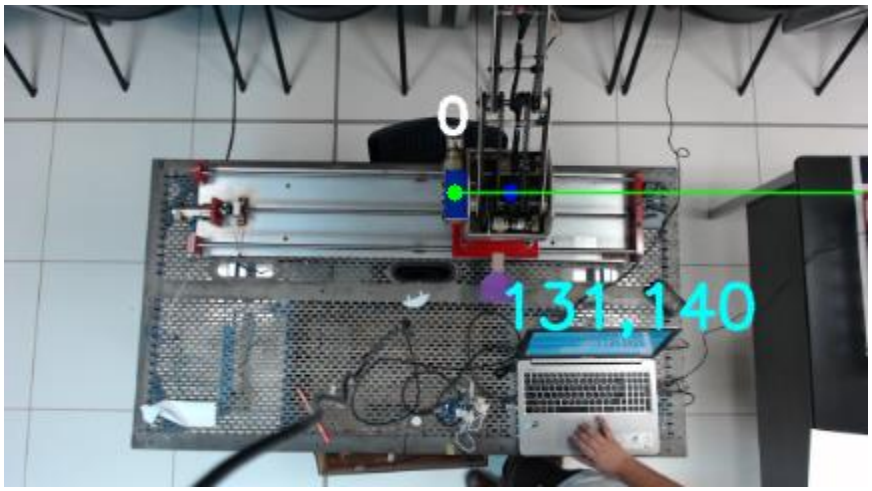
```
if(z>q+4){
  digitalWrite(motorFD, HIGH);
  digitalWrite(motorFI, LOW);
  analogWrite(ENF,128);

  }
else if(z<q-4){
  digitalWrite(motorFD, LOW);
  digitalWrite(motorFI, HIGH);
  analogWrite(ENF,128);

  }
else{
  digitalWrite(motorFD, LOW);
  digitalWrite(motorFI, LOW);
  analogWrite(ENF,50);

    b4=true;

  }///// imprime todos los valores
Serial.print("valores iniciales \n");
Serial.print(q); //see what was received
Serial.print(" \n");
Serial.print(w); //see what was received
Serial.print(" \n");
Serial.print(e); //see what was received
Serial.print(" \n");
Serial.print(r); //see what was received
```

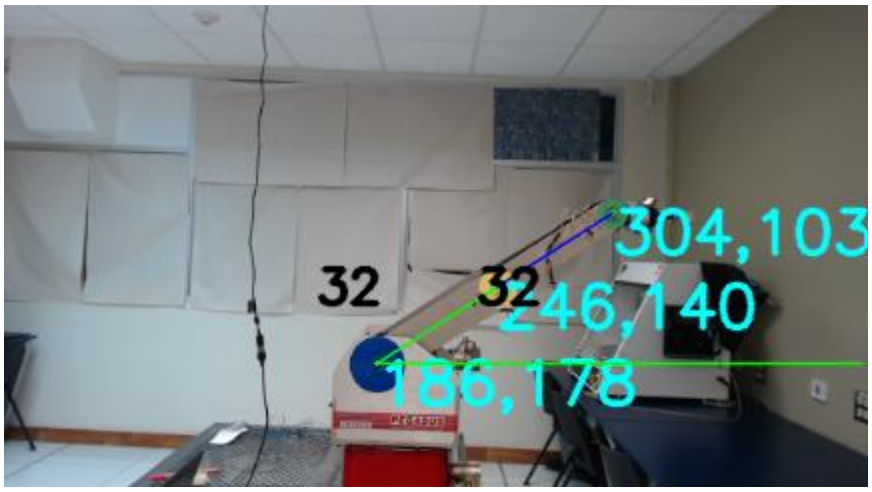
```
    Serial.print(" \n");
    Serial.print("valores retroalimentados \n");
    Serial.print(z); //see what was received
    Serial.print(" \n");
    Serial.print(x); //see what was received
    Serial.print(" \n");
    Serial.print(n); //see what was received
    Serial.print(" \n");
    Serial.print(v); //see what was received
    Serial.print(" \n");
    Serial.flush();
    ///// limpia las variables y el buffer
    des="";
    ab="";
    a1="";
    a2="";
    char buff[30]={};
    z=0;
    x=0;
    n=0;
    v=0;
    //////// si las banderas de verificacion en la seccion de activacion de motores son verdaderas se envia el mensaje "listo" indicando que se alcanzo la posicion
   if(b3==true ){
    if(b2==true){
       if(b1==true){
         if(b4==true){
```

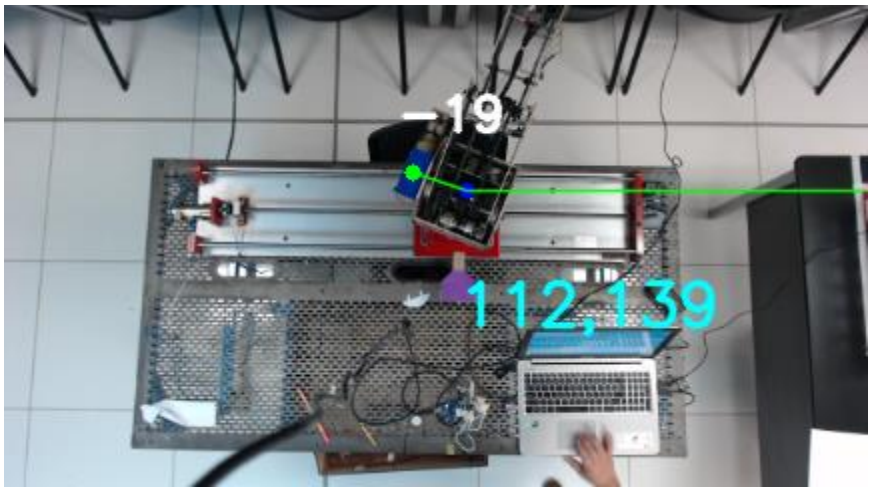
```
            Serial1.print("listo");
            b1=false;
            b2=false;
            b3=false;
            b4=false;
            Serial1.flush();


          }


        }

      }

    }


    }


}
```

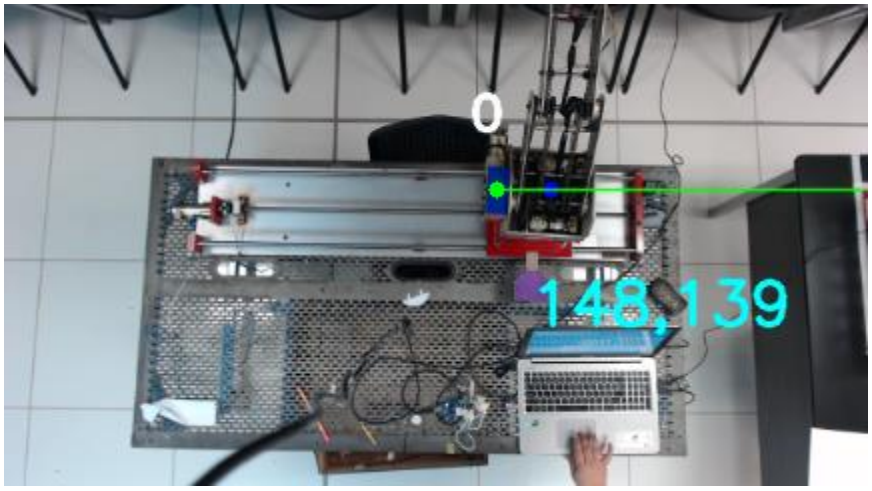
Apéndice 2. Pruebas de ejecución

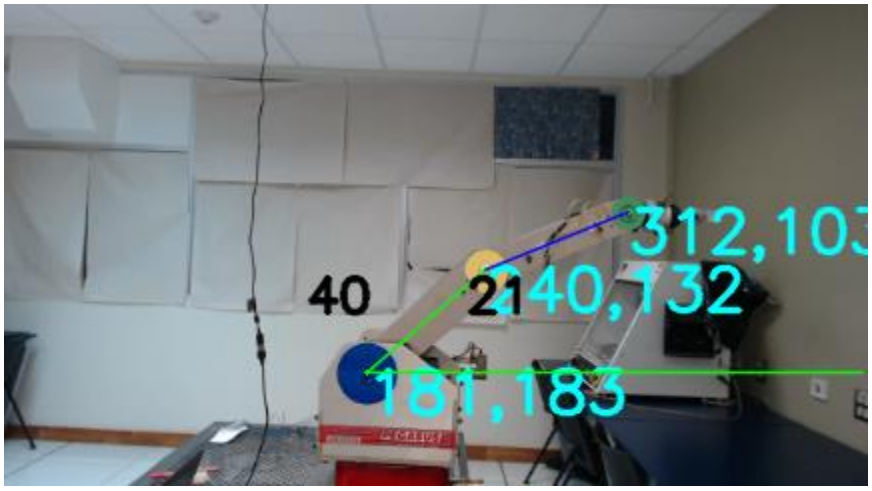
Tomas correspondientes a las pruebas de ejecución de la sección

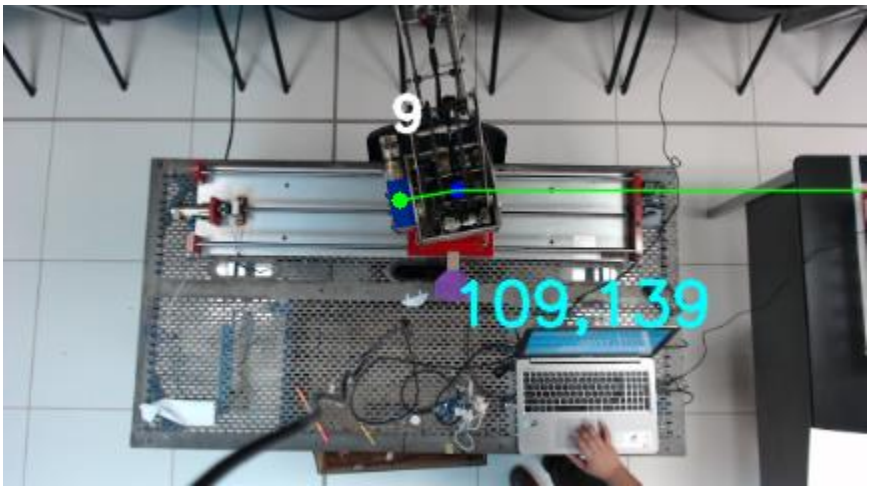





−19
112,139


0
148,139


312,103
40
21
140,132
181,183


9
109,139


43
23
290,108
239,130
195,172

Apéndice 3. Diagrama de conexión