**UNIVERSIDAD TECNOLÓGICA ISRAEL**

**TRABAJO DE TITULACIÓN EN OPCIÓN AL GRADO DE:**

**INGENIERO EN ELECTRÓNICA DIGITAL Y TELECOMUNICACIONES**

# TEMA: PROTOTIPO PARA CLASIFICACIÓN DE ROPA HOSPITALARIA A TRAVÉS DE ETIQUETAS DE COLOR UTILIZANDO VISIÓN ARTIFICIAL MEDIANTE ARDUINO

**AUTOR: JORGE LUIS GARCÉS ZAMORA**

**TUTORA: Mg. SILVIA DIANA MARTÍNEZ MOSQUERA**

**QUITO- ECUADOR**

**AÑO: 2018**

## DECLARACION DE AUTOR

Yo, JORGE LUIS GARCÉS ZAMORA, declaro que el trabajo aquí escrito es de mi autoría y que no ha sido presentando por ningún grado o alguna calificación profesional, y, que se ha utilizado referencias bibliográficas que están incluidas en este documento.

La Universidad Tecnológica Israel del Ecuador, puede hacer uso de los derechos correspondientes de este trabajo, de acuerdo lo establecido por la Ley de Educación Superior.

…….

**JORGE LUIS GARCÉS ZAMORA**

## CERTIFICACIÓN DEL TUTOR.

La suscrita, MSc. SILVIA MARTÍNEZ, en calidad de docente de la Universidad Tecnológica Israel del Ecuador, certifica que el estudiante JORGE LUIS GARCÉS ZAMORA, realizó el trabajo de titulación para la obtención del título de Ingeniería en Electrónica Digital y Telecomunicaciones con el tema "**PROTOTIPO PARA CLASIFICACIÓN DE ROPA HOSPITALARIA A TRAVÉS DE ETIQUETAS DE COLOR UTILIZANDO VISIÓN ARTIFICIAL MEDIANTE ARDUINO**", bajo mi dirección, habiendo cumplido satisfactoriamente con la disposición reglamentarias establecidas para el efecto del mismo.

---

MSc. SILVIA MARTÍNEZ
**DIRECTORA DE TESIS**

## AGRADECIMIENTO

Agradezco en primer lugar a Dios por la oportunidad de existir y sobre todo por permitirme dar un paso más para hoy en día poder culminar mi carrera profesional. Más sin embargo en este diario vivir agradezco a la persona que ha estado junto a mi desde el primer paso de vida, quien de forma incondicional y con el profundo amor que le caracteriza me ha brindado su apoyo y confianza absoluta, mi amada y abnegada mamita.

Agradezco también a todos quienes me brindaron de forma incondicional su sano consejo para poder continuar y no desfallecer.

A la Universidad "Tecnológica Israel", agradezco por los conocimientos transmitidos por medio de los docentes en lo largo de mi carrera estudiantil, sabiendo fomentar los conocimientos inherentes a la profesión.

A la MSc. Silvia Martínez quien de forma puntual dirigió mi tema de tesis, entregando consejos acertados para salir avante en los problemas normales a los del desarrollo de la tesis, por lo que ha sido fundamental su apoyo y dirección.

**JORGE LUIS GARCÉS ZAMORA**

## DEDICATORIA

El esfuerzo y trabajo realizado se lo dedico en primer lugar a Dios, quien me dio coraje y fuerza para continuar pese a cualquier circunstancia logrando así llegar a culminar uno de mis mayores objetivos de vida.

Dedico el esfuerzo realizado a mi familia, quienes me brindaron su apoyo de manera incondicional, sabiendo darme palabras de aliento y sobre todo por estar pendientes en los momentos más difíciles de mi vida dándome su confianza y principalmente su amor.

**JORGE LUIS GARCÉS ZAMORA**

# ÍNDICE

## Índice de Figuras

# Índice de Tablas

# Índice de Códigos

## RESUMEN

Los sistemas de visión artificial han ido en constante evolución, de la mano del desarrollo de herramientas computacionales que soportan este tipo de sistemas. Este proyecto desarrolló un sistema de clasificación de prendas hospitalarias basadas en la detección de color usando visión artificial. Las distintas teorías y conceptos se analizan para poder llevar a cabo el desarrollo de la programación del algoritmo para la detección de color y su subsiguiente clasificación.

El algoritmo de Marr-Palmer incluye un modelo general de visión artificial con el cual este proyecto ha sido desarrollado. El proceso de detección de color ha sido programado en Matlab, aquí se realiza la explicación de los códigos y funciones necesarias para que el proceso de visión artificial se realice. Los diagramas de flujo para el proceso de visión artificial están detallados y explicados con minuciosidad.

El proyecto está controlado por 3 Arduinos con 3 fases de acción, para los diferentes escenarios que se presenten los Arduinos tendrán lo necesario para actuar. La lógica de programación junto con algunos comandos se explica en este proyecto.

La interfaz gráfica desarrollada en LabVIEW permite la visualización y control de los distintos procesos que se llevan a cabo dentro de la clasificación. Esta interfaz es amigable con el operador y contempla la visualización de todos los procesos que se llevan a cabo en el proceso de clasificación.

Los resultados obtenidos después de toda la programación son mostrados al final de este proyecto.



**Palabras clave:** Arduino, artificial, Labview, segmentación, visión

## ABSTRACT

The systems of artificial vision have gone in constant evolution, of the hand of the development of computational tools that support this type of systems. This project developed a classification system for hospital garments based on color detection using artificial vision. The different theories and concepts are analyzed to be able to carry out the development of the programming of the algorithm for the detection of color and its subsequent classification.

The Marr-Palmer algorithm includes a general model of artificial vision with which this project has been developed. The process of color detection has been programmed in Matlab, here the explanation of the codes and functions necessary for the process of artificial vision is carried out. The flow diagrams for the artificial vision process are detailed and explained with thoroughness.

The project is controlled by 3 Arduinos with 3 action phases, for the different scenarios presented the Arduinos will have what it takes to act. The programming logic together with some commands is explained in this project.

The graphical interface developed in LabVIEW allows the visualization and control of the different processes that are carried out within the classification. This interface is friendly with the operator and contemplates the visualization of all the processes that are carried out in the classification process.

The results obtained after all the programming are shown at the end of this project.



**Keywords:** Arduino, artificial, Labview, segmentation, vision

# INTRODUCCIÓN

## Antecedentes

La visión artificial incluye varios métodos para adquirir, procesar y analizar imágenes del mundo real y de esta manera tratar los datos recolectados. La visión artificial es aplicada en los sectores industriales, científicos y en el área de seguridad. El uso de la visión artificial en las diversas áreas ha aumentado los índices de productividad y disminuido la pérdida de materiales, de esta manera se tiene un producto final de menor costo; pero de mayor calidad.

En el sector industrial se ha desarrollado varias aplicaciones para el control de calidad, el conteo de artículos, control de rotación de maquinaria y clasificaciones de materiales utilizando la visión artificial. En la industria agrícola se utiliza la clasificación de frutas o verduras en diferentes categorías a través de bandas transportadoras para su debido empaque y distribución. En el área de la medicina se utiliza la visión artificial para el reconocimiento automático de vasos sanguíneos para lo cual es necesario la identificación de un objeto mediante la comparación de sus características visuales con las de un patrón almacenado

## Planteamiento y justificación del problema

Las entidades médicas como hospitales o clínicas generan decenas de prendas hospitalarias diariamente provenientes de los pacientes y usuarios los cuales se atienden. Algunas de estas prendas han tenido contacto con pacientes críticos, los cuales se los clasifica como aislados, toda prenda proveniente de este tipo de pacientes es considerada como ropa contaminada y su tratamiento para el procesamiento correspondiente de lavado debe ser diferenciado al de una prenda estimada como sucia, la cual no proviene de pacientes aislados.

La clasificación de prendas contaminadas y no contaminadas de una entidad médica es realizada por el personal de asistencia de los hospitales, es decir por parte de los auxiliares de lavandería hospitalaria. Estas prácticas y procesos ponen en riesgo la salud de las personas que hacen este tipo de clasificación, por lo que se requiere un proceso que automatice esta clasificación siendo necesario para las entidades médicas.

Este proyecto creó un prototipo que permite la clasificación de tres tipos de prendas hospitalarias mediante técnicas de color usando visión artificial. El prototipo busca que las decisiones de clasificación de las prendas se basen en la detección de los colores que proporcione el algoritmo de visión artificial junto con una cámara web la cual simularía la acción del ojo humano.

El algoritmo de visión artificial debe contener la capacidad suficiente para la detección de al menos tres colores para los tres tipos de prendas que serán clasificados. En el prototipo para controlar su funcionamiento es necesaria la programación de un algoritmo de visión artificial, lo cual se desarrollará dentro de este proyecto. El algoritmo de visión artificial que se desea programar en este proyecto es el modelo de visión artificial de Marr-Palmer. Este modelo de visión define 4 etapas como son: adquisición de datos, reproceso, segmentación y detección de color. Si se cumplen estas 4 etapas podremos completar exitosamente el proceso de detección de colores mediante visión artificial

El desarrollo de un prototipo controlado mediante visión artificial presenta ventajas sobre sensores de color los cuales se encuentran de manera comercial. Los sensores presentan problemas cuando la ubicación del elemento a detectar varia es decir se debe priorizar la precisión en la ubicación del elemento en este caso ropa hospitalaria. Mientras que cuando se utilizan técnicas de color mediante visión artificial, el algoritmo a desarrollar busca que siendo cumplidas ciertas condiciones de ubicación del elemento el programa sepa encontrar el color de la prenda a clasificar y poder tomar las decisiones necesarias para su clasificación.

Este algoritmo de visión artificial será desarrollado en lenguaje de programación C siendo necesaria una investigación sobre procesos, comandos y algoritmos que ofrece la herramienta Matlab para poder desarrollar el código de visión artificial.

El proceso de investigación también involucra aspectos necesarios como procesamiento de imágenes, reconocimiento, segmentación y filtros aplicados a la información que proporciona la cámara con el objetivo de reconocer el color y aplicarlos en la decisión que debe realizar el prototipo para clasificar.

Este prototipo involucra una fase de diseño electrónico ya que una vez hecha la respectiva detección del color se debe tomar acciones donde se involucran procesos electrónicos. La programación de estos procesos será realizada en un microcontrolador

embebido en la plataforma de software libre Arduino. El control de motores, servomotores y la recepción de información para la acción de estos serán desarrollados en esta plataforma. Se deberá investigar y seleccionar adecuadamente los elementos a utilizar en el prototipo para evitar fallos en el funcionamiento de este. Las características eléctricas, físicas y mecánicas de estos deben ser cuidadosamente analizadas y seleccionadas.

Todos estos procesos deben estar bien controlados y perfectamente sincronizados porque se emula un proceso industrial como tal, donde el funcionamiento continuo y la automatización de este es imprescindible.

Para que todos estos procesos funcionen de manera sincronizada, la programación junto con los elementos y dispositivos a utilizar deben estar perfectamente alineados a una correcta estructura. El movimiento de las prendas se realizará mediante bandas transportadoras y la ubicación de estas en las respectivas lavadoras se lo hará por ductos de conducción. La selección de los materiales para la estructura del prototipo es importante ya que cada fase debe tener niveles de confianza en su funcionamiento para que la continuidad del proceso se lleve a cabo con normalidad. La fase de investigación con respecto a la parte mecánica de este prototipo también será explicada dentro de este proyecto.

## Objetivo General

Implementar un prototipo para clasificación de ropa hospitalaria a través de una banda transportadora y algoritmo de visión artificial desarrollados en el lenguaje de programación Matlab y la plataforma Arduino junto con una interfaz remota.

## Objetivos Específicos

Crear una maqueta con banda transportadora y ductos de conducción que permitan emular el proceso automatizado que seguirían las prendas hospitalarias dentro de su respectivo lavado.

Desarrollar un sistema de visión artificial con adquisición de imágenes mediante cámara web y programación en el lenguaje Matlab, que permita la detección de etiquetas de colores sobre prendas hospitalarias,

Programar un sistema de control que permita tomar las acciones pertinentes a los elementos correspondientes como motores o servo motores para la clasificación de las prendas hospitalarias, tomando la información tanto de los sensores instalados como del sistema de visión artificial para su correcto funcionamiento.

Implementar una interfaz gráfica que permita de manera remota controlar el proceso realizado por la maqueta, para que un operador pueda tomar las medidas necesarias si existe inconvenientes dentro del proceso automatizado de clasificación.

Presentar las pruebas y resultados tanto del proceso de clasificación como del proceso de detección del color mediante la visión artificial, para la comprobación del correcto funcionamiento de los algoritmos y programas implementados dentro de este proyecto.

**Alcance**

El desarrollo del prototipo contará con una banda transportadora principal para trasladar la ropa hospitalaria.

- Bolsas de color rojo destinadas para ropa contaminada (Áreas críticas)
- Bolsas de color azul para la ropa no contaminada (Hospitalización)
- Bolsas de color verde para la ropa semi sucia (Consulta externa)

La cámara es un elemento importante del proceso ya que enviará una imagen al algoritmo realizado en MATLAB para detectar el color de las bolsas de ropa, la banda transportadora contará con su respectivo motor el cual girará constantemente hasta que se interrumpa de manera remota por la interfaz gráfica o de manera local cortando el paso de energía eléctrica.

Otra opción que existe para la finalización de proceso del traslado de la ropa en la banda transportadora será cuando la capacidad de las lavadoras esté completa.

Una vez identificado el tipo de prenda, ya sea como contaminada, no contaminada o semi sucia, un elemento al final de la banda transportadora direccionara la ropa por dos caminos hacia la emulación de un proceso de lavado. Estos dos caminos contarán con su respectiva inclinación para que directamente lleguen a los coches correspondientes.

El proyecto emulará tres lavadoras con diferente capacidad. Un contador verificara que la lavadora está llena y cambiara la dirección del camino para que comience a llenar la siguiente lavadora. Cuando una de las lavadoras esté completa, un indicador se activará en la interfaz gráfica de usuario para que el operador no siga enviando ropa. Si los dos caminos están llenos y las capacidades de sus lavadoras también, la banda automáticamente se detendrá.

La interfaz gráfica tendrá los elementos necesarios para iniciar, pausar y parar el proceso. En medio de los procesos se tendrán alarmas que serán mostradas en la interfaz de usuario por posibles complicaciones del proceso.

Todo el proceso de control y operación estará diseñado en base a funcionalidades de un Arduino, esto gracias a sus múltiples funciones y fácil manejo. El proyecto contará con las herramientas necesarias para el correcto funcionamiento del mismo, verificando las condiciones y eventos que puedan suscitarse y podrá funcionar de manera autónoma o de manera manual.

**Descripción de los capítulos**

Debido a las premisas presentadas en el capítulo uno de este proyecto se expone las temáticas y conceptos necesarios para poder desarrollar todos los lineamientos necesarios para la implementación de este proyecto. En este capítulo se incluyen los principales conceptos para entender los modelos de visión artificial y como ser implementados. También se incluyen características de los diferentes lenguajes de programación entre ellos: Matlab, Arduino, LabVIEW.

En el capítulo dos se toman los conceptos descritos por el capítulo uno y se plantea las soluciones que deben llevarse a cabo para el desarrollo del proyecto. Se explica con mayor profundidad los procedimientos que se deben implementar para el correcto

funcionamiento del proyecto. Esto incluye la propuesta de solución al algoritmo de visión artificial, la interfaz gráfica en LabVIEW y el accionar de los actuadores.

En el capítulo tres se describen los procedimientos y los comandos necesarios para poder implementar lo planteado en el capítulo dos. Se explica a detalle los comandos utilizados para la implementación del algoritmo de visión artificial. Además, se describe los procesos que seguirán tanto los Arduino como la interfaz gráfica en LabVIEW, las soluciones descritas en el capítulo dos serán implementadas y probadas para que su funcionamiento sea acorde. La sinergia que debe tener cada solución dentro de la maqueta se verificara en las pruebas y resultados.

Al final se expondrán las conclusiones de este proyecto respaldando los objetivos planteados, además se indicarán las recomendaciones para proyectos futuros.

# CAPÍTULO I

# FUNDAMENTACIÓN TEÓRICA

Este capítulo presenta la información necesaria para que los cálculos, algoritmos y procesos del proyecto sean entendidos, entre estas definiciones básicas están la visión artificial, características de los lenguajes de programación y plataformas a utilizar.

## 1.1 VISIÓN POR COMPUTADOR

La visión por computador también llamada visión artificial tiene como objetivo capturar y procesar características de una imagen, lo cual se realiza por elementos artificiales como cámaras o sensores acústicos, térmicos, táctiles entre otros. La visión artificial también busca interpretar información de una determinada imagen ya sea del tipo espacial o temporal.

Algunas de las aplicaciones en las cuales la visión artificial tiene uso están el reconocimiento de caracteres, la inspección visual, verificación de productos, construcción y reconstrucción de modelos, aplicaciones médicas, seguridad para conductores en las vías, captura de movimiento, sistemas de monitoreo y vigilancia, sistemas biométricos, entre otros.

### 1.1.1 Luz visible

El espectro visible lo constituyen aquellas ondas luminosas que pueden ser apreciadas por el ojo humano, este va desde los 400 a 700 nm, cómo se observa en la Figura 1.1. Las ondas recibidas en el ojo humano se transforman en impulsos eléctricos que son enviados al cerebro. El cerebro procesa esta información y lo transforma en sensaciones de brillo y color de los diferentes objetos (Bertozzi et al., 2002).

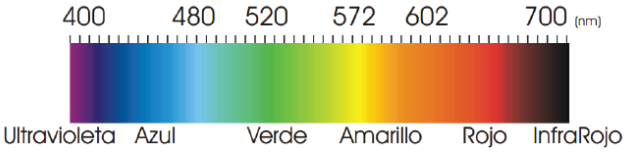



**Figura 1.1 Espectro de luz visible**

**Fuente:** (Bertozzi et al., 2002)

### 1.1.2 Intensidad luminosa

Es la cantidad de flujo luminoso emitido por unidad de ángulo sólido, la cantidad de intensidad luminosa de una fuente de luz se mide en candelas. Esta no depende de la distancia a la cual se encuentre la fuente de luz ya que varía según la orientación de la medición realizada hacia la fuente de luz. El flujo luminoso es la cantidad de flujo o de haces luminosos percibidos por el ojo humano a diferentes longitudes de onda.

### 1.1.3 Bastones

Son las células presentes en la retina que permiten la detección del brillo de haces luminosos emitidos por un objeto. Son células fotorreceptoras que pertenecen a la retina y son responsables de la visión en una baja condición de luminosidad.

### 1.1.4 Conos

Son las células presentes en la retina que permiten detectar las variaciones de longitud de onda de un haz luminoso emitido por un objeto, en cantidad son mucho menor que los bastones y por lo tanto su sensibilidad es menor.

Existen diferentes tipos de conos los cuales son sensibles a diferentes longitudes de onda, por ejemplo, los conos tipo S (Short) son sensibles a longitudes de onda corta pertenecientes a los tonos de azul, los conos tipo M (Medium) son sensibles a longitudes de onda media pertenecientes a los tonos verdes, dentro de la retina estos se encuentran en mayor cantidad indicando que los tonos verdes son los más percibidos por el ojo humano y

los conos tipo L (*Large*) son sensibles a longitudes de onda larga pertenecientes a los tonos de rojo (Jácome, Teresa, Andrade, & Javier, 2014).

### 1.1.5 Brillo

Es la cantidad de flujo luminoso recibido o emitido por unidad de superficie. En este sentido, el flujo luminoso está sujeto a cuántas partículas que se hallan en determinada cantidad de espacio (superficie) y, en relación con el tiempo, son capaces de emitir luz.

### 1.1.6 Tono o Matiz

Cuando los conos perciben las distintas longitudes de onda estos se activan en mayor o menor cantidad dependiendo del color dominante en el objeto, el matiz es la relación entre las diferentes activaciones de estos conos.

### 1.1.7 Contraste

La luminosidad y matiz de una imagen que contiene zonas claras y oscuras varia, el contraste se define como la diferencia entre estos parámetros.

### 1.1.8 Saturación

También conocida como viveza o palidez del color de una imagen, la saturación se define como la diferencia entre la longitud de onda dominante y las demás longitudes de onda. Si la diferencia no es grande, la imagen dará una sensación de palidez, mientras que con una diferencia mayor la viveza de la imagen aumentará.

### 1.1.9 Píxel

Es el elemento más pequeño y de menor información en una imagen. Contiene información de la posición dentro de una imagen y la información de características mínimas como color, textura, valor de gris, entre otros.

### 1.1.10 Imagen

La imagen como conjunto se define como el conjunto finito, no nulo de pixeles con posiciones diferentes dentro de la rejilla de una imagen, los pixeles pueden ser iguales o no pero el número de pixeles de una imagen debe ser igual al número de rejillas que presenta la imagen (Lueiza Valenzuela, Bro, Meza Montoya, & Reyes, 2007).

La imagen puede ser vista como función vectorial del mapeo de las rejillas, esta función vectorial asume para cada posición un vector de valores, donde cada posición con su vector de valores corresponde a un canal distinto de la imagen.

### 1.1.11 Región

La región es un subconjunto no nulo de una imagen, dos partes independientes de una imagen pueden corresponder a una misma región o dos partes conexas o dependientes pueden ser parte de una misma región.

### 1.1.12 Fondo

Fondo es todo aquello que no forma parte de la imagen a reconocer, dentro de una imagen pueden existir regiones donde se encuentran objetos a reconocer, las regiones que no correspondan o no contengan objetos de interés para ser reconocidos se consideraran como fondo.

### 1.1.13 Objeto

Objeto es aquella entidad perteneciente al mundo real con un significado contextual en una u otra aplicación de interés.

### 1.1.14 Histograma

Herramienta para representar ya sea de manera cuantitativa o cualitativa una imagen, en el (eje x) representa la distribución de intensidades de gris mientras que en el (eje y) estará la cantidad de pixeles pertenecientes a cada distribución de gris, la Figura 1.2 muestra el ejemplo de un histograma aplicado a una imagen (González Benítez, 2010).

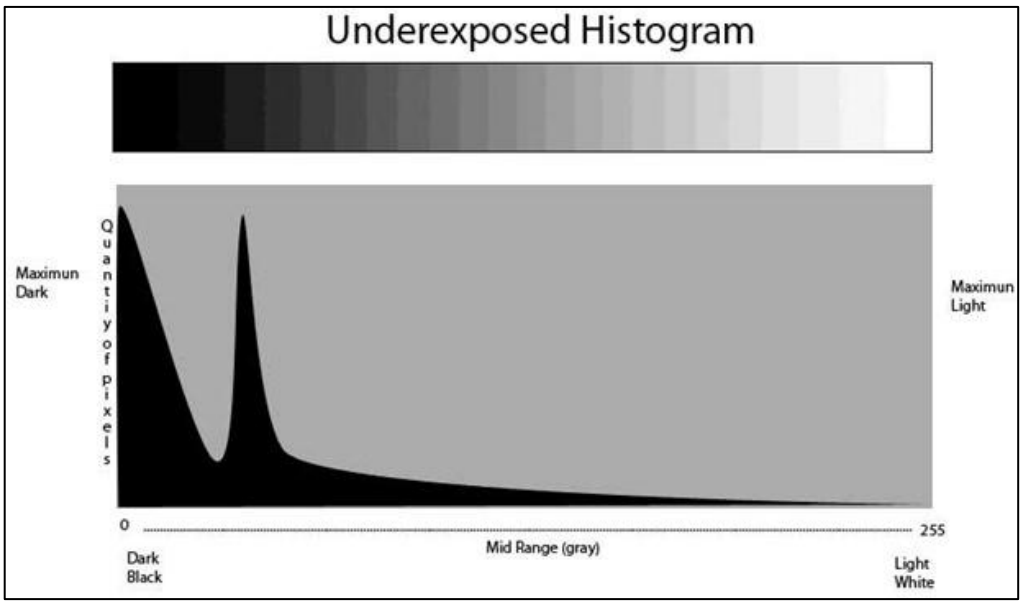



**Figura 1.2 Histograma de una imagen.**

Fuente: (Lueiza Valenzuela, Bro, Meza Montoya, & Reyes, 2007).

### 1.1.15 Sistema fisiológico visual

El ojo humano principal actuador del sistema de visión el cual transforma la luz capturada en impulsos que son procesados en el cerebro, el procesamiento de los impulsos eléctricos en el cerebro da como resultado la percepción visual.

### 1.1.16 Percepción acromática

Son las sensaciones producidas por los bastones y se relacionan con los siguientes fenómenos:

- Sensibilidad a la intensidad, brinda la capacidad de distinguir entre dos niveles de intensidad, la mayoría de estas intensidades son perdidas cuando el número de sensaciones dadas superan los 24 tonos. Esta intensidad percibida no responde a una función lineal como indica la Figura 1.3, dando al ser humano la capacidad de distinguir pigmentos de intensidades similares siempre y cuando no exista un contraste adecuado cambiando su apreciación. (Abid K, 2017).

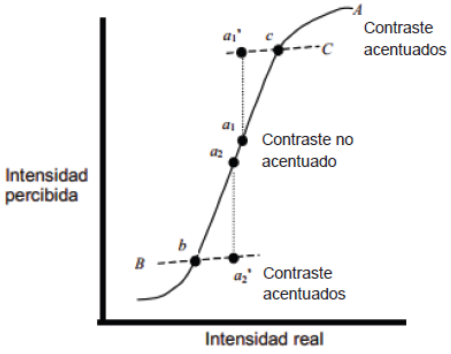



Figura 1.3 Intensidad recibida vs intensidad real

**Fuente:** (Abid K, 2017).

La Figura 1.4 muestra que junto a mayores niveles de intensidad la precisión de detección disminuye, necesitándose mayores periodos de adaptación. El cuadro gris del cuadrado interior de la Figura 1.4 de la derecha parece más oscuro que el cuadrado interior de la figura izquierda, a pesar de que ambos están tintados con el mismo gris.

**Figura 1.4 Comparativa entre niveles de intensidad y la precisión de detección**

Fuente: (Abid K, 2017).

### 1.1.17 Modelo de visión artificial

La visión artificial busca emular lo realizado por el cerebro cuando recibe las señales luminosas por parte del ojo humano, para esto se definen básicamente cuatro etapas como lo indica la Figura 1.5 (Fernandes & Bala, 2015).

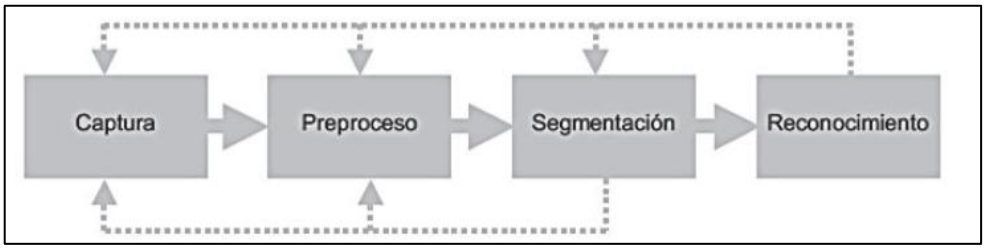



**Figura 1.5** Etapas de una visión artificial

**Fuente:** (Fernandes & Bala, 2015)

- **Captura:** Fase sensorial, necesaria una adquisición de imágenes por parte de algún dispositivo o sensor para ser digitalizada.
- **Reproceso:** Filtrado de imagen necesario para el resalte de los elementos o características más importantes de la imagen capturada. En este proceso también se busca eliminar aquellas características imperceptibles o innecesarias para el procesamiento.
- **Segmentación:** De toda la imagen se toman los elementos de interés para poder comprenderla y procesarla.
- **Reconocimiento:** Una vez establecidas las características a analizarse el reconocimiento busca en la imagen las características separadas en la segmentación para ser distinguidas.

Este proceso no es lineal ni secuencial siendo necesarias regresiones dentro del proceso por errores dentro de los mismos.

Los problemas a nivel computacional son analizados basando su principio en el modelo de visión general de Marr-Palmer. El modelo de visión de Marr-Palmer esta descrito en la Figura 1.6.

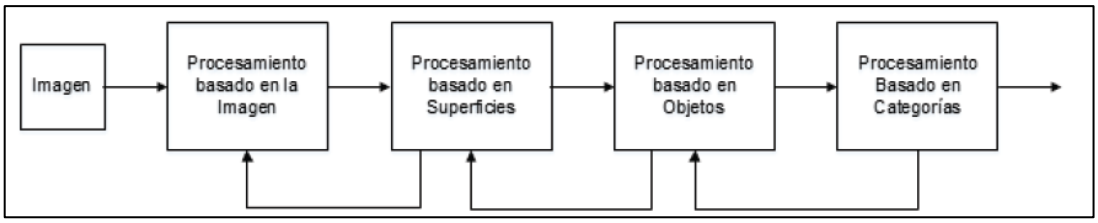



**Figura 1.6 Modelo de visión general de Marr-Palmer**

Fuente: (Fernandes & Bala, 2015)

Este modelo permite observar, analizar y corregir errores a diferentes niveles en la imagen.

- Basado en la imagen muestra el contenido por pixel.
- Basado en superficies otorga información adicional con el objetivo de regularizar problemas como configuración de luz y color.
- Basado en objetos muy similar al ítem anterior, pero con la diferencia que en las superficies con ciertas características pueden llegar a formar una imagen.
- Basado en categorías busca cualquier relación entre los objetos e imágenes.

### 1.1.18 SEGMENTACIÓN

La segmentación de una imagen tiene como objetivo separar a una figura en regiones con un fin específico para cierta aplicación (Abaya, Basa, Sy, Abad, & Dadios, 2014). Presentándose en una visión artificial una paradoja singular, dice:

"Para segmentar objetos es necesario reconocerlos y para esto se requiere segmentarlos".

#### 1.1.18.1 Segmentación supervisada

En este tipo de segmentación el factor humano se ve involucrado, verificando características para poder determinar el tipo de segmentación que tendrá una imagen.

#### 1.1.18.2 Segmentación no supervisada

La segmentación la realiza la máquina de manera autónoma, sin la necesidad de ayuda por parte del ser humano.

#### 1.1.18.3 Segmentación a nivel de objeto

La segmentación a nivel de objeto asegura que cada parte segmentada de una imagen contenga únicamente un objeto de interés, asegurándose que no exista varios objetos dentro de una misma región segmentada. El número de segmentos debe llegar a un número optimo el cual es definido por el operador para que no exista sobre-segmentación (Mukundan & Ramakrishnan, 1998).

#### 1.1.18.4 Segmentación a nivel de imagen

Para la segmentación a nivel de imagen, se toman regiones conformadas por una cierta cantidad de pixeles en los cuales se mantienen ciertos niveles de uniformidad y homogeneidad, cada método de segmentación definirá el valor de homogeneidad con la que las regiones segmentadas contaran para su identificación.

#### 1.1.18.5 Segmentación a nivel de imagen: Umbralización

Se utilizan básicamente dos regiones una región objeto y una región fondo. El operador definirá el parámetro de uniformidad o umbral que responde a cierta característica con la cuales contará la imagen y toda imagen que no supere este umbral será definido en la región fondo y las imágenes que logren superar este umbral se las colocara en la región objeto.

Cuando el parámetro de uniformidad no lo determina un operador, este puede ser previsto de manera automática tomando en cuenta las condiciones en las que las imágenes

van a ser segmentadas, este algoritmo de determinación de umbral puede ser dado por el algoritmo de Otsu (Fernández & Javier, 2009).

#### 1.1.18.6 Segmentación a nivel de imagen: Histogramas

En un análisis basándose en los histogramas de las imágenes se pueden determinar diferentes regiones separando y tomando como parámetro de uniformidad los máximos relativos en un histograma. Las segmentaciones de imágenes en tonos de grises utilizan mucho este método debido a que se puede separar las zonas más claras de la misma.

### 1.1.19 ESPACIOS DE COLOR

#### 1.1.19.1 Espacio *Red Green Blue* (RGB)

Espacio de color formado por tres ejes y tres planos, cada uno de estos planos se relaciona con un color primario rojo (RED), verde (GREEN), azul (BLUE), los valores que pueden tomar los diferentes ejes o colores van entre 0 y 1. Estos planos pueden tomar una representación por canales de pixeles asociando un byte a cada canal, dando un total de 256 tonalidades diferentes de cada eje o color, estos valores irían entre 0 y 255. Al juntar los tres ejes cada pixel formaría parte de un punto dentro de este plano en tres dimensiones, además teniendo en cuenta las 256 diferentes combinaciones de cada uno de los pixeles en cada eje, el total de colores con los cuales se puede representar a un pixel seria 16777216.

#### 1.1.19.2 Espacio *Hue Saturation Value (*HSV)

En este espacio se dejan atrás las características de los tres colores primarios de una imagen y se buscan otras características como: Tono (HUE), saturación (SATURATION), brillo (VALUE). La Figura 1.7 muestra la representación del espacio HSV.

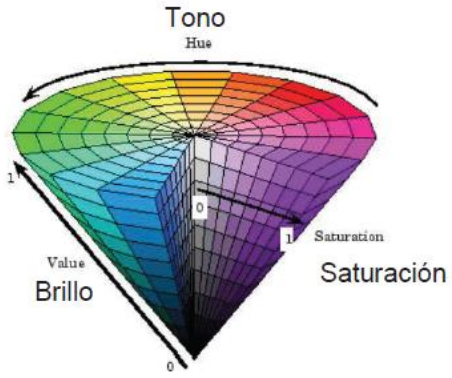



**Figura 1.7** Representación espacio HSV

Fuente: (Koivo & Houshangi, 1991).

La representación de un píxel en los diferentes espacios de color pueden alternarse dependiendo de la aplicación que se busque. Las características que se representan en cada uno de los espacios de color serán primordiales al escoger o seleccionar los parámetros necesarios en la segmentación de una imagen (Koivo & Houshangi, 1991).

## 1.2 SOFTWARE

Los Software utilizados para el desarrollo del sistema fueron Matlab, en especial el Toolbox denominado *Image Processing* para el proceso y análisis de imágenes. También se utilizó LabVIEW para el desarrollo de sistema en base a un lenguaje gráfico y visual. A continuación, se describe cada uno de ellos.

### 1.2.1 Matlab

Matlab es un lenguaje de cálculo técnico utilizado para miles de aplicaciones, lo conforman un entorno de desarrollo integrado IDE con su lenguaje propio llamado M. Matlab ha permitido el desarrollo de aprendizaje automático procesamiento de señales, procesamiento de imágenes, visión artificial, comunicaciones, finanzas computacionales, diseño de control, robótica entre otros campos (Sabas, 2011).

El lenguaje de programación de Matlab esta optimizado para la resolución de problemas científicos y de ingeniería basándose en matrices, es ideal para las matemáticas computacionales. Los miles de herramientas, códigos, comandos y toolbox han permitido el desarrollo de cientos de miles de algoritmos utilizados en diferentes facetas del ámbito científico-técnico.

La última versión disponible es la R2017A, cabe recalcar que Matlab es un programa licenciado y con un costo elevado.

Los toolbox o cajas de herramientas permiten implementar funciones adicionales a las propias del lenguaje de Matlab, estas cajas de herramientas son desarrolladas dependiendo de la necesidad con la que las fases de investigación de ciertos procesos se ejecuten. Para el proceso de imágenes con el cual este proyecto se desarrollará utiliza el toolbox *Image Processing.*

#### 1.2.1.1 Image Processing

Este Toolbox como dice su nombre permite el procesamiento de las imágenes. Matlab junto con esta herramienta permite analizar y procesar imágenes con los formatos: TIFF, JPEG, GIF, BMP, PNG, XWD.

Como ya se había mencionado anteriormente los espacios de color como RGB contienen tres planos para definir a una imagen, Matlab facilita el proceso por su naturaleza de manejo de matrices y vectores, asignando un vector a cada plano RGB. Además, el tipo de dato podría variar dependiendo de la aplicación que se busque con los vectores y matrices correspondientes a una imagen.

El cambio de tipo de dato original a otro es rápido y automático, con eso una imagen puede ser binarizada, cambiada a escala de grises, cambiar a espacio HSV o crear una imagen indexada ente otros.

Además, esta herramienta permite el manejo de transformaciones de intensidad como la gamma, logarítmica, histogramas de una imagen, entre otros.

También permite realizar la segmentación de las imágenes, manejando diferentes tipos de umbrales y diferentes técnicas como: Sobel, Prewitt, Roberts, LoG, Canny. De la

misma manera permite realizar la umbralización de las imágenes y manejo de otros conceptos necesarios para la implementación de la visión artificial (Flores, 2017).

### 1.2.2 LabVIEW

Acrónimo de Laboratory Virtual Instrument Engineering Workbench, es una plataforma que permite el desarrollo de sistemas de ingeniería. El desarrollo de estos sistemas se programa en base a lenguaje gráfico y visual. Es una plataforma con gran uso en la simulación de hardware y software de control ya que permite una emulación de procesos de alto nivel.

El lenguaje G, utilizado en LabVIEW es de alto nivel y muy robusto, cuenta con un gran número de herramientas, condiciones, funciones y elementos para la simulación cientos de procesos principalmente industriales.

El desarrollo de Interfaces graficas de usuario, tiene ventajas sobre el desarrollo de interfaces graficas de usuario en Matlab. La principal ventaja es los elementos adicionales que presenta entre ellos contadores, botones, simuladores de válvulas entre otros. Además, la comunicación es sencilla con respecto a Matlab, ya que presenta los elementos y conectores necesarios para realizar las funciones u operaciones que se desean dentro del proceso (Sabas, 2011).

## 1.3 ARDUINO

Arduino es una placa de desarrollo siendo una plataforma de código abierto destinada al uso en proyectos de electrónica. El elemento principal dentro de la placa de desarrollo es el microcontrolador la cual realiza el procesamiento de cada acción programada dentro de la Interfaz de Desarrollo Integrado (IDE) propia de Arduino. La IDE de Arduino permite escribir y cargar el código programado en la computadora hacia la placa Arduino.

Una de las razones por las cuales el desarrollo y uso de la plataforma Arduino ha sido la fácil programación y carga del programa hacia la placa, a diferencia de otros microcontroladores que necesitan un programador externo, Arduino permite que la carga de sus programas se realice mediante la interfaz USB (*Universal Serial Bus*). Además, su

lenguaje simplificado basado en C++ con funciones y librerías ha dado paso al desarrollo de aplicaciones en el ámbito electrónico.

La interaccion con LED's, motores, altavoces, unidades GPS (*Global Positioning System*), cámaras, Internet, teléfonos móviles entre otros le dan una gran flexibilidad a esta plataforma, permitiendo que miles de usuarios desarrollen sus proyectos creando una gran comunidad de usuarios interesados en el desarrollo de este tipo de proyectos.

### 1.3.1 Elementos Placa Arduino

A pesar de que existen algunos modelos de placas de desarrollo, las cuales se explicaran más adelante, todas tienen en común ciertos elementos, la Figura 1.8 muestra la placa Arduino.

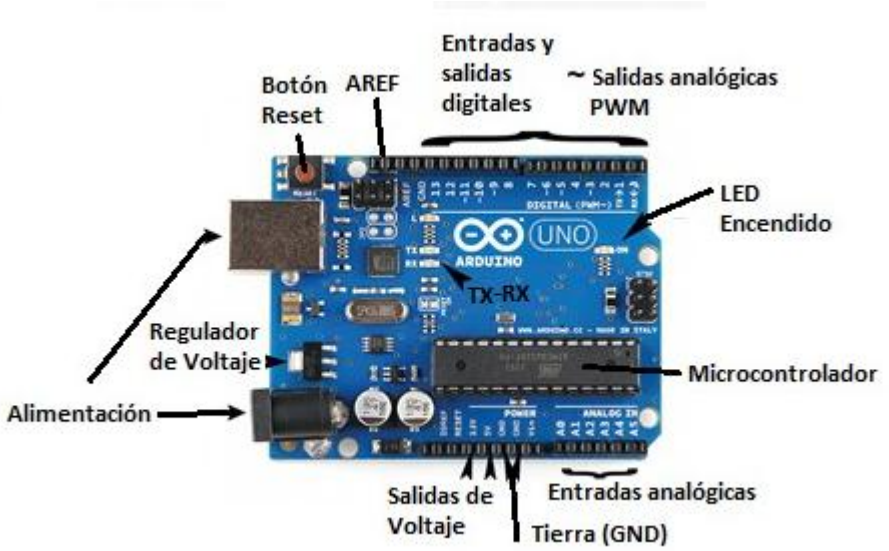



**Figura 1.8 Placa de Desarrollo Arduino**

Fuente: Elaborado por el autor

De acuerdo a la figura 1.8 los elementos que conforman el dispositivo Arduino Uno son:

- **Alimentación:** La conexión indicada corresponde generalmente a un Jack USB, pero en algunas variaciones de Arduino utilizan otro tipo de interfaz, la misma que servirá como alimentación y para cargar los programas de la placa. La placa puede tener una conexión o alimentación externa independiente de la conexión USB de la

computadora como lo indica el conector de Barrel donde la conexión óptima indica es de 6 a 12 Voltios en la mayoría de las placas.

- **Ground:** GND o tierra, sirve para aterrizar a tierra los circuitos, si se conecta fuentes de alimentación externa, se deberá cortocircuitar las tierras.

- **Salidas de voltaje:** Salidas de voltaje continuo corresponde a 5 voltios y a 3.3 voltios. Servirá para alimentar otros dispositivos tomando en cuenta la limitación de corriente que presenta la placa Arduino.

- **Entradas Analógicas:** Se las denota con la Letra A y el número de entrada que corresponde, para la Placa de la Figura 2.1 contiene 6 entradas de la A0 a A5. Estas entradas admiten variaciones de voltaje de 0 a 5 voltios los cuales pueden provenir de sensores analógicos u otro tipo de entrada de tensión variante.

- **Entradas y Salidas digitales:** Los pines de la placa podrán ser utilizados de ambas formas ya sea como salidas o entradas digitales el número de pines dependerá del tipo de placa que estemos manejando. La configuración le dará la característica al pin de ser entrada o salida.

- **Salidas Analógicas:** También denominadas salidas PWM (*pulse with modulation*), ya que permiten emitir señales moduladas por ancho de pulso o PWM. Se las identifica con la tilde en el número de pin (~). Esto no impide que estas salidas sean usadas como digitales, pero si añaden una característica especial a esos pines. El número de salidas analógicas dependerá de la placa de desarrollo que se esté manejando.

- **AREF:** Pin que se utiliza como soporte para la referencia analógica, usualmente dan la tensión o voltaje de referencia máximo que puede aceptarse en una entrada analógica.

- **Reset Button:** Pone temporalmente el botón de reinicio a tierra, reiniciando cualquier código que tenga el Arduino cargado.

- **Led Encendido:** Indica que el Arduino está funcionando y conectado a la fuente de alimentación correcta, si este led no enciende existe una gran probabilidad de que el circuito armado en base a Arduino este dañado.

- **Led Tx-Rx:** Permite saber si la placa Arduino está recibiendo o mandando datos para su funcionamiento, cada que recibe un paquete de información el led de Rx se enciende, del mismo modo ocurre en el led de Tx cuando el Arduino envía.

- **Microcontrolador:** Es el circuito integrado principal de la placa y funciona como el cerebro principal que realiza el procesamiento de toda la información de entrada y salida de la placa. Estos microcontroladores son de marca Atmel, y el tipo de microcontrolador dependerá de la placa que estemos utilizando, donde su principal característica la darán la velocidad y numero de bits que utiliza el microcontrolador.

- **Regulador de voltaje:** Es una pieza con la cual realmente no se interactúa en el proceso de programación e instalación de proyectos basados en Arduino, pero corresponde a una parte importante para el funcionamiento correcto de la placa. Permite regular el voltaje que ingresa a la placa Arduino, si no existiera este elemento probablemente muchos de los circuitos no funcionarían.

### 1.3.2 Tipos de Placa Arduino

#### 1.3.2.1 Arduino UNO

La Figura 1.9 muestra un Arduino uno, con su forma y principales elementos como se indicó en el literal anterior. La placa Arduino UNO es una de las más comunes y de fácil adquisición en el mercado (Brahmbhatt, 2013).


**Figura 1.9 Placa de Desarrollo Arduino UNO**

Fuente: (Brahmbhatt, 2013)

Arduino UNO basa su funcionamiento en el microcontrolador Atmel ATmega328p, contiene 14 pines de entradas o salidas digitales numerados del 0 al 13 con salida máxima de 5 voltios y 40 mA de corriente, de estas 14 salidas 6 pueden ser utilizadas como salidas PWM, los pines que aceptan salidas analógicas son los pines: 3,5,6,9,10,11. Cuenta con 6 entradas analógicas en la parte inferior numeradas de la A0 a la A5. El microcontrolador contiene un cristal de 16 Mhz. El microcontrolador soporta 3 timers o tipos de reloj controlados por las 6 salidas analógicas. Cuenta con 32k de almacenamiento en la memoria flash, esto indica que la cantidad máxima de código que cabe dentro del microcontrolador y por ende en la placa son 32Kbytes.

La entrada de alimentación es una interfaz USB tipo B hembra, el cable con el cual se alimentará a la placa y se cargaran los programas está conformado por un USB tipo A macho en un lado y un USB tipo B macho en el lado a conectar en la placa.

#### 1.3.2.2 Arduino NANO

El Arduino NANO no es muy diferente a lo ofrecido por el Arduino UNO, un Arduino NANO es mostrado en la Figura 1.10.

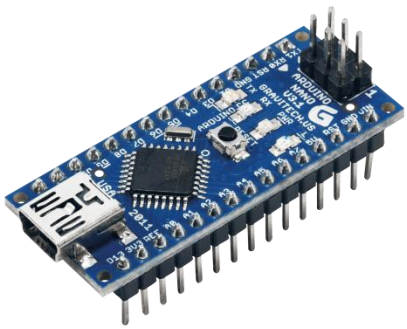


**Figura 1.10 Placa de Desarrollo Arduino NANO**

Fuente: (Brahmbhatt, 2013)

El Arduino NANO es considerado una versión miniatura del Arduino UNO, es por esto por lo que las principales diferencias se presentan en las dimensiones y puertos de alimentación de la placa. Las dimensiones de esta placa son 4.5 x 1.8 cm. Mientras que usan un puerto mini USB para la carga de programas. No contienen puerto de alimentación externa. En las demás características el Arduino NANO es exactamente igual al Arduino UNO.

### 1.3.2.3 Arduino MEGA

La Figura 1.11 muestra un Arduino MEGA, con su forma y principales elementos.

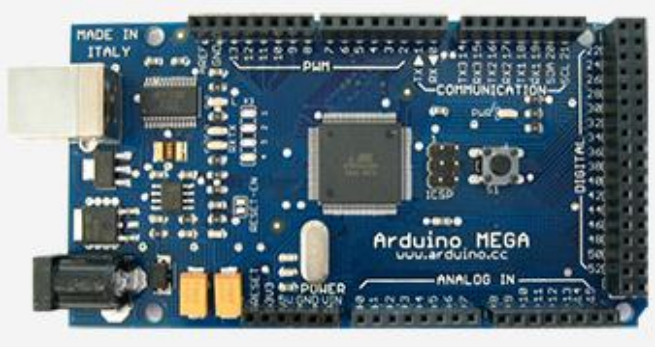



**Figura 1.11 Placa de Desarrollo Arduino MEGA**

Fuente: (Brahmbhatt, 2013)

Arduino MEGA basa su funcionamiento en el microcontrolador Atmel ATmega1280, contiene 54 pines de entradas o salidas digitales numerados del 0 al 53 con salida máxima de 5 voltios y 40 mA de corriente, de estas 54 salidas 14 pueden ser utilizadas como salidas PWM, los pines que aceptan salidas analógicas son los pines del 2 al 13 y 44 a 46. Cuenta con 16 entradas analógicas en la parte inferior numeradas de la A0 a la A15. El microcontrolador contiene un cristal de 16 Mhz. El microcontrolador soporta 6 timers o tipos de reloj controlados por las 14 salidas analógicas. Cuenta con 128k de almacenamiento en la memoria flash, esto indica que la cantidad máxima de código que cabe dentro del microcontrolador y por ende en la placa son 128Kbytes. Adicionalmente cuenta con 4 pares de pines para comunicación serial.

La entrada de alimentación es una interfaz USB tipo B hembra, el cable con el cual se alimentará a la placa y se cargaran los programas está conformado por un USB tipo A macho en un lado y un USB tipo B macho en el lado a conectar en la placa.

La programación dentro de un Arduino MEGA es similar a los anteriores, pero se debe seleccionar el tipo de Arduino con la cual se está trabajando debido a las funciones y características propias que tiene cada uno de los diferentes Arduinos.

## 1.4 ELEMENTOS MAQUETA

### 1.4.1 Banda transportadora

La banda o cinta transportadora es uno de los métodos de transportación continua de materiales o elementos más utilizados en la industria. La idea principal de una banda transportadora es que otorgue a un sistema de transportación continuidad y dinamismo para ello debe contener una banda sin fin. Para el correcto funcionamiento de una banda transportadora se consideran dos tipos de estaciones, una estación accionadora la cual pone en movimiento motor con su correspondiente tambor y una estación tensora que contiene el dispositivo tensor donde se sujetara la banda sin fin. En cada estación se contempla rodillos por donde realizara el giro la banda transportadora.

Las bandas transportadoras pueden tener cierto grado de inclinación, pero generalmente en la industria se colocan de forma horizontal. Los materiales con los cuales las bandas se fabrican son variados, pero deben cumplir con ciertos requisitos mínimos como:

- Alta resistencia mecánica longitudinal, evitando que el continuo movimiento y giro en los tambores afecte el correcto funcionamiento de la banda con posibles daños a futuro.
- Flexibilidad en direcciones longitudinal, evitando cortes o rompeduras en la zona de los rodillos de la banda.
- Elevada resistencia al desgaste y a la desestratificación por reiterados dobleces.

- Alta resistencia a la humedad, este factor es más crítico cuando se transportan elementos con cierto grado de humedad como bebidas, alimentos, entre otros.

Los materiales con lo cual se fabriquen las bandas también dependerán mucho del tipo de trabajo que se va a realizar o qué tipo de materiales va a llevar la banda. Algunos de los tipos son:

- Lisa: para instalaciones con ángulo de inclinación 0 o cercano a cero, es decir para aplicaciones horizontales.
- Superficie rugosa: consiguiendo evitar la caída o desplazamiento de los elementos llevados en la banda.
- Con pestañas onduladas y salientes o nervios en V: para bandas con un gran ángulo de inclinación evitando el movimiento de los materiales transportados.

La Figura 1.12 muestra una banda transportadora con 0 grados de inclinación y material rugoso. Se puede apreciar las dos estaciones, el motor que hace girar a la banda y los rodillos por los cuales la banda realiza el giro.

**Figura 1.12 Banda de transportadora con material rugoso.**

Fuente: (Bertozzi et al., 2002)

### 1.4.2 Motores de corriente continua

Los motores de corriente continua o DC basan su funcionamiento principalmente en la repulsión que ejercen los polos magnéticos de un imán ubicado de forma permanente, con un electroimán ubicado o montado en el eje de la estructura del motor. Este electroimán comienza a actuar cuando es excitado con una corriente DC, de ahí su nombre. El torque de un motor deberá estar acorde al peso de la carga transportada por la banda.

Los motores DC generan una velocidad nominal sin carga, la cual variará y dependerá directamente de la carga colocada en la banda transportadora que hace girar este motor.

Los motores DC tienen en su funcionamiento mecánico una relación de engranaje, mientras mayor relación de engrane menor velocidad nominal sin carga, pero mayor torque máximo. Los motores para aplicaciones electrónicas pequeñas pueden llegar a consumir 800 mA de corriente, las placas de desarrollo y los microcontroladores no soportan tal cantidad de corriente, siendo necesario una fuente de alimentación externa que genera la corriente necesaria para la alimentación de estos motores (Bertozzi et al., 2002). La Figura 1.13 muestra un motor DC de marca Pololu.


**Figura 1.13 Motor DC para circuitos pequeños marca Pololu**

Fuente: (Bertozzi et al., 2002)

### 1.4.3 Servo Motor

Un servo motor es un tipo especial de motor que tiene la capacidad de girar ciertos grados y mantener su posición hasta una nueva instrucción. Con un buen control se puede lograr determinar la posición del eje del motor en cualquier instante de tiempo.

A diferencia de los motores DC se mantienen girando mientras es excitado por una corriente, el servomotor recibe una señal PWM con el objetivo que gire cierta cantidad de grados y se mantenga en esa posición. Cabe recalcar que un servo motor tiene como componente principal un motor DC, al cual se han aplicado y unido una caja de engranes que controlan la posición junto con la señal PWM enviada. Generalmente el giro de un servo motor comprende de los 0 a los 180 grados sexagesimales.

Las señales PWM son señales moduladas por ancho de pulso, el valor de voltaje continuo que emiten estas señales PWM depende del ancho de pulso que tiene una señal de este tipo. Mientras mayor ancho el pulso, mayor giro tendrá el servomotor. La Figura 1.14 muestra un servomotor para aplicaciones relativamente pequeñas.

**Figura 1.14 Servomotor**

Fuente: (Bertozzi et al., 2002)

### 1.4.4 Control PWM

El control de servo motor es comúnmente controlado a través de señales con modulación de ancho de pulso o por sus siglas en ingles PWM. El objetivo de esta señal es controlar tanto el tiempo en alto como el tiempo en bajo del pulso de una señal cuadrada, pero manteniendo fijo el periodo o la frecuencia de la señal, todo esto con el objetivo de controlar la posición del servo motor.

La duración en alto de una serie de pulsos indicará el ángulo donde se situará el servo motor, el servo entiende una duración máxima y mínima de duración de pulso, estos valores indicaran los extremos de ubicación 0 o 180 grados. Los valores más generales de estos pulsos se encuentran entre 1 a 2 ms, siendo el valor de 1.5 ms el valor central de posición es decir 90 grados sexagesimales. La Figura 1.15 muestra la relación de altos y bajo que tiene una señal PWM para el control de posición de un servo motor.

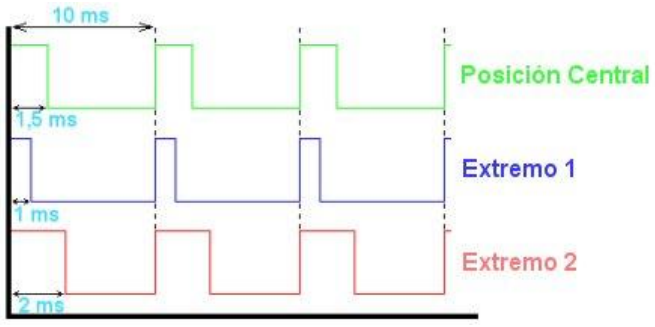



**Figura 1.15 Señales PWM para el control de posición de un servo motor**

Fuente: (Bertozzi et al., 2002)

### 1.4.5 Sensores Infrarrojos

Los sensores infrarrojos son sensores que basan su funcionamiento en la detección de presencia. Estos sensores emiten y receptan una señal lumínica no visible que se encuentra en la zona de infrarrojos. Estas señales serán analizadas por el tiempo en que se demoran en ser rebotadas por el objeto que se busca detectar. Cuando un objeto no pasa cerca del sensor infrarrojo la señal podrá demorarse considerablemente e incluso no ser recibida, pero cuando un objeto pasa cerca del sensor infrarrojo la señal será directamente recibida nuevamente dando a indicar que se ha detectado la presencia de un objeto.

Los sensores infrarrojos utilizados son digitales como los mostrados en la Figura 1.16, lo que indica que son previamente acondicionados para el uso directo en la placa de desarrollo Arduino

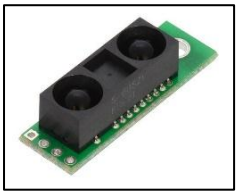


**Figura 1.16 Sensor infrarrojo digital**

Fuente: (Koivo, 1991)

### 1.4.6 Conversor DC-DC

Son dispositivos que convierten voltaje continuo a voltaje continuo de diferente valor, estas conversiones pueden ser elevadores o reductores, además permiten regular ese voltaje convertido.

Al tener un voltaje regulado a la salida del convertidor permite una salida de voltaje continuo y constante independiente de las perturbaciones o alteraciones que pueden presentarse en la entrada del convertidor. Este convertidor también permite mantener la corriente sin afectar a la fuente de entrada, útil en aplicaciones con transporte de carga como en las bandas transportadoras.

### 1.4.7 Driver de Motor

Un controlador de motor básicamente está constituido por una serie de puentes H dentro de un circuito integrado. Un puente H es aquel circuito electrónico que permite el control de giro y velocidad de un motor DC a través de entradas PWM.

Se le denomina puente H por la forma de su diagrama circuital el cual está constituido de 4 interruptores ya sean mecánicos o utilizando transistores donde la combinación de apertura y cierre de estos 4 interruptores generaran una tensión positiva o negativa al motor brindando así el sentido de giro del motor. No se puede tener a todos los interruptores cerrados debido a que esto provocaría un cortocircuito afectando al funcionamiento del motor.

En un driver de motor está controlado y únicamente permitirá los estados donde el motor gira, si se encuentra esos estados donde el puente H generaría un cortocircuito, el driver no da paso a ese estado y el motor no se accionará. La Figura 1.17 muestra el diagrama circuital principal de un puente H.

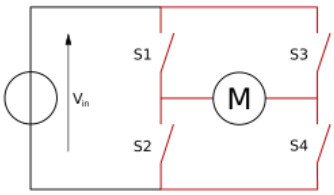



**Figura 1.17** Diagrama circuital puente H

Fuente: (Abaya, 2014)

Si S1 junto con S4 se encuentran conectados y S2 con S3 abiertos el motor girara en avance. Si se invierten estos estados el motor retrocederá.

# CAPITULO 2
# PROPUESTA

## 2.1 REQUERIMIENTOS FUNCIONALES

El sistema clasificará ropa hospitalaria mediante reconocimiento de color. Para la identificación del color el sistema utilizará una cámara. Además, el proyecto contará con una banda transportadora principal donde estará ubicada la cámara.

El sistema tendrá una interfaz gráfica amigable para el usuario. La banda transportadora contará con su respectivo motor el cual girará constantemente hasta que se interrumpa de manera remota por la interfaz gráfica o de manera local con botones pertenecientes al prototipo.

Una vez identificado el tipo de prenda, al final de la banda transportadora se direccionará la ropa por dos caminos hacia la emulación de un proceso de lavado. Estos dos caminos contarán con su respectiva inclinación para que directamente lleguen a los coches correspondientes.

La interfaz gráfica tendrá los elementos necesarios para iniciar, pausar y parar el proceso. Además, posee indicadores de la detección de colores. En medio de los procesos se tendrán alarmas que serán mostradas en la interfaz de usuario por posibles complicaciones del proceso. El proyecto podrá funcionar de manera autónoma o de manera manual.

## 2.2 REQUERIMIENTOS NO FUNCIONALES

El algoritmo de reconocimiento de color será realizado en MATLAB. Todo el proceso de control y operación estará diseñado en base a funcionalidades de un Arduino.

## 2.3 DIAGRAMA DE REQUERIMIENTOS

En base a los requerimientos identificados anteriormente se describen los actores y sus funciones mediante el uso del diagrama de casos de uso mostrado en la Figura 2.1.

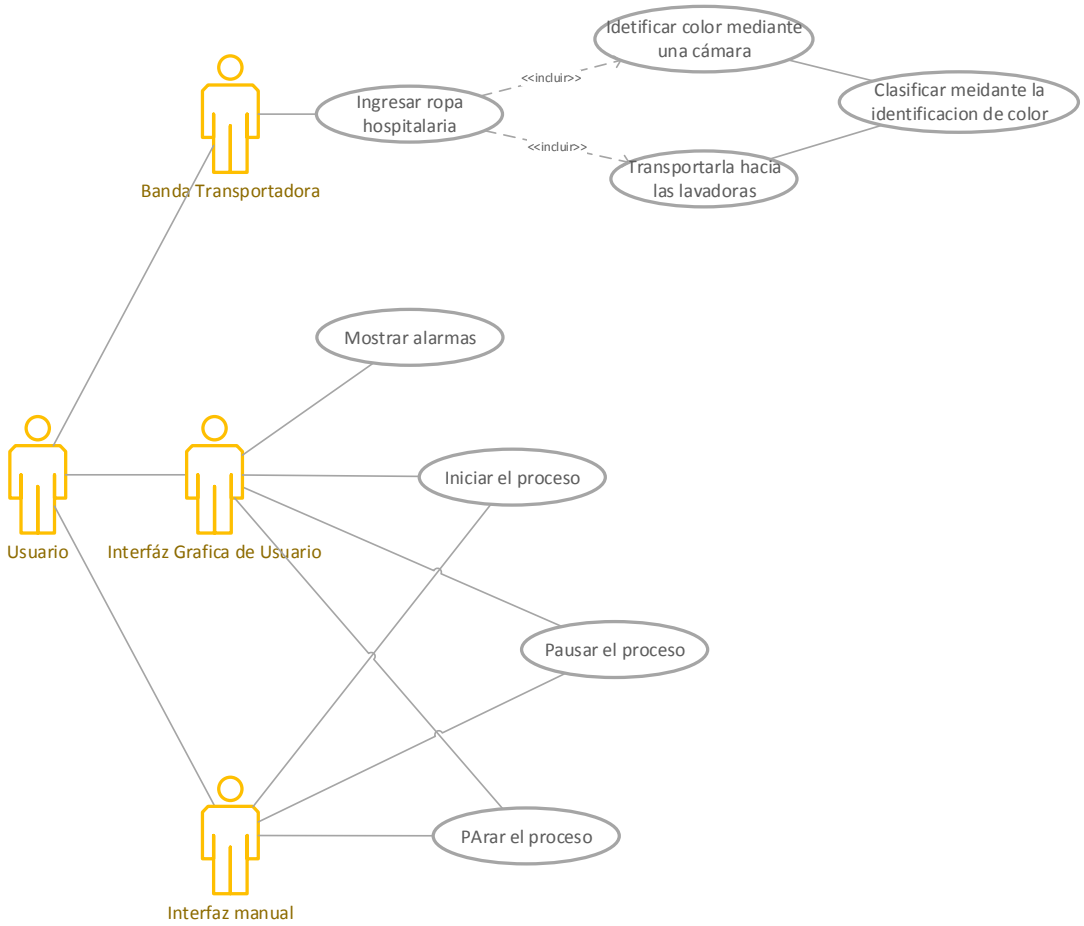



**Figura 2.1**: Diagrama de requerimientos

Fuente: Elaborado por el autor

En la Figura 2.1 se muestra el diagrama de requerimientos donde se indican los actores que intervienen en el prototipo, los cuales son representados por la imagen de una persona unido mediante líneas a las acciones que cada actor puede realizar, existe funciones que se ejecutan siempre y cuando una acción anterior haya sido ejecutada dicha función la relaciona con la acción necesaria anterior mediante la etiqueta <<incluir>>.

## 2.4 DISEÑO DE HARDWARE

El material que va a ser transportado es el parámetro más importante para el diseño y selección de componentes de una banda transportadora. Las características principales que se deben considerar para los cálculos son los presentados a continuación.

Para poder realizar su función, la banda transportadora requiere una potencia que será suministrada por el motor que acciona el tambor conductor o motriz. El momento del motor se transmite en forma de fuerza tangencial sobre la superficie del tambor motriz creando el empuje o fuerza requerida para mover el material desde el punto de carga hasta el de descarga. Esta fuerza es la resultante de la sumatoria de las siguientes fuerzas:

- Fuerza necesaria para mover la banda sin carga, vacía.
- Fuerza necesaria para desplazar la carga horizontalmente.
- Fuerza necesaria para elevar o descender la carga.

Estas son las fuerzas producidas únicamente para desplazar el material desde el punto de alimentación hasta el de descarga. Para calcular las tensiones en la banda y la potencia requerida, se deben considerar las perdidas mecánicas por fricción del sistema completo de accionamiento, las cuales son debidas a las guías de carga, rascadores y cualquier tipo de desalineamiento de los tambores o rodillos.

### 2.4.1 Factores por fricción y por longitud

Las fuerzas mencionadas anteriormente se deben básicamente a la fricción generada por el peso de la banda, peso de las partes móviles de los rodillos y tambores, arrastres debidos a las guías de carga, rascadores y desalineamientos. Adicionalmente, el peso del material sobre la banda y la fricción interna que el material genera a medida que pasa una y otra vez sobre los rodillos incrementan la fricción en el sistema. El cálculo de estas fuerzas de fricción depende de un factor llamado coeficiente de fricción de las partes móviles f, el cual varía en función del tipo de rodillos empleados, la estructura de la banda y el mantenimiento del sistema.

### 2.4.2 Selección de Motor DC

Analizando estos factores se determinó que un motor ideal para este proyecto es aquel que cuente con un torque considerable, ya que la velocidad no es un factor primordial en el desarrollo del presente proyecto.

El motor que cumple con estas condiciones es: 100:1 Micro Metal Gearmotor HPCB 6V de marca Pololu.

Este es un mini motor de 6 vatios de alta potencia que cuenta con cepillos de carbón de larga duración que dentro cuenta con una caja de engranes que entrega una relación torque velocidad de 100 a 1.

Cuenta con una sección trasversal de 10 x 12 mm y el eje de salida en forma de D tiene dimensiones de 9 mm de largo x 3 mm de ancho. El motor tiene una velocidad angular máxima de 320 RPM y consume 120 mA de corriente sin carga, cuenta con un torque máximo de 2.2 Kg-cm. La Figura 2.2 muestra el motor seleccionado para este proyecto.

**Figura 2.2 Motor Micro Metal Gearmotor H HPCB 6V de marca Pololu**

Fuente: (Jácome, 2014)

### 2.4.3 Selección de servo motor

Los requerimientos para la selección del servo motor son menores debido a que el torque necesario para realizar la acción no contempla la distancia que recorre el objeto a través de la banda transportadora. Tomando estas consideraciones se ha seleccionado el servo: Tower Pro Micro Servo SG90 9g. Es un micro servo debido a sus dimensiones y peso, pero las funcionalidades se mantienen con respecto a los servos más grandes.

Las dimensiones de este micro servo son 22.2 x 11.8 x 31 mm aproximadamente, tiene un peso de 9g, soporta un torque de 1.8 kgf-cm, se mueve en 0.1 segundos por cada 60 grados de rotación siendo su giro total de 180 grados, soporta apertura del pulso de 1 a 2 ms. Cuenta con tres entradas, alimentación, tierra y control, en este último se mandará la señales PWM, que realizan el control del movimiento del servo. La Figura 2.3 muestra el servo motor seleccionado.


**Figura 2.3** Servo Tower Pro Micro Servo SG90 9g

Fuente: (Jácome, 2014)

### 2.4.4 Selección Sensor infrarrojo

El sensor infrarrojo seleccionado es el módulo TCRT5000. Este módulo cuenta con todas las características de detección de obstáculos mencionadas en la primera sección. Una de las principales características es que este módulo ya se encuentra acondicionado y optimizado para ser utilizado en la placa Arduino.

Tiene un rango de detección de 1 a 8 mm, cuenta con una salida digital y una salida analógica, las dos permiten la detección de objetos para la cual se presenta un led indicador de estado para realizar el respectivo aviso cuando se detecte un objeto, en su parte posterior tiene un potenciómetro que permite ajustar la sensibilidad del sensor, todo esto sumado a un amplificador LM393 que le brinda estabilidad al circuito. La figura 2.4 presenta el módulo sensor infrarrojo utilizado para este proyecto.

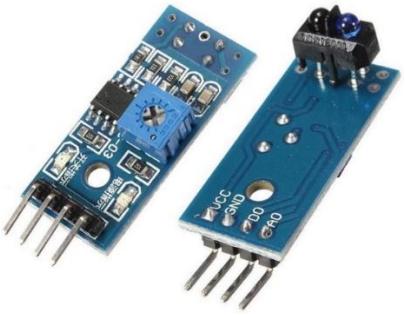



**Figura 2.4** Modulo sensor infrarrojo TCRT5000

### 2.4.5 Selección de placa de desarrollo Arduino

Este proyecto cuenta con tres fases de procesamiento los cuales son:

- Procesamiento de los datos generados en Matlab, pertenecientes al proceso de visión artificial (Fase 1)
- Accionamiento de los actuadores como servo y motor DC (Fase 2).
- Sistemas de comunicación, reconocimiento de los datos de sensores y muestra datos de visualización en el panel (Fase 3).

Se ha seleccionado 3 microcontroladores por las limitaciones que presenta la programación en Arduino. Estos 3 Arduinos están encargados de las distintas fases del proyecto los cuales se dividen en las siguientes fases.

- Arduino Uno: Fase 1
- Arduino Nano: Fase 2
- Arduino Mega: Fase 3

La explicación de cada fase se detallará en el desarrollo e implementación en secciones subsiguientes.

### 2.4.6 Selección conversor DC-DC

El conversor seleccionado es el módulo LM256, es un convertidor DC-DC Buck o reductor. Este módulo permite regular voltajes de entrada 12 voltios a 5, 3.3 o 2.2 voltios, usado ampliamente en aplicaciones para Arduino, PICS, Rasberry Pi, fuentes, entre otros.

Este módulo basa su funcionamiento en el circuito integrado Lm2596, regulador DC-DC Step Down, útil para fuentes de conmutación tipo buck. Permite una corriente de hasta 3A, manejando de buena manera sobre voltajes y voltajes de rizado.

Admite voltajes de entrada de 4.5 a 40 voltios con salidas ajustables de 1.5 a 35 voltios, frecuencia de conmutación de 150 Khz, con dimensiones de 43 x 20 x 14 mm aproximadamente. La figura 2.5 muestra el conversor DC-DC seleccionado.


**Figura 2.5 Módulo conversor DC-DC LM2596**

Fuente: (Jácome, 2014)

### 2.4.7 Selección driver de motor

El circuito integrado seleccionado es el driver L293D, de Texas Instruments, este circuito integrado cuenta con 4 puentes H dentro del circuito integrado, lo que permite un control de hasta 4 motores DC.

Permite conducir una corriente de hasta 600 mA, y manejar voltajes de 4.5 a 36 voltios. Las señales de entrada corresponden a PWM activadas en parejas de pines, generando las secuencias antes mencionadas. Este driver tiene inconvenientes con corrientes parasitas, siendo necesario diodos para evitar cualquier tipo de inconveniente. En la figura 2.6 e muestra el diagrama de pines del circuito integrado L293D.

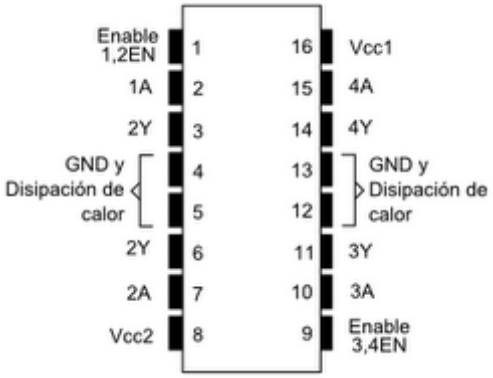



**Figura 2.6** Circuito Integrado LM293D

Fuente: (Jácome, 2014)

Hay que considerar que el voltaje de alimentación de las entradas de los puentes H varía con respecto al voltaje de alimentación del integrado, siendo necesario un doble voltaje de alimentación.

### 2.4.8 Selección Cámara Web

Dentro del proceso de visión artificial, el elemento que realiza la emulación del sistema visual humano es la cámara web, esta se encarga de enviar las imágenes para ser procesadas a través de Matlab siendo parte importante del proceso de este proyecto. La cámara seleccionada para realizar esta actividad fue la cámara web Altek @one 8Mp. Esta cámara cuenta con 8 megapíxeles y 6 leds que iluminan la zona donde se realiza la captura. La Figura 2.7 muestra la cámara web seleccionada.

**Figura 2.7 Cámara web Altek @one 8Mp**

Fuente: (Sabas, 2014)

### 2.4.9 Selección Elementos maqueta

Para que exista un soporte correcto a toda la estructura la maqueta se compone en un 80% de madera sobre la cual irán algunos de los elementos detallados con anterioridad, la banda transportadora y el panel de visualización de estados. Además, el diseño sobrio y elegante de la maqueta es un factor adicional, si bien no afecta al funcionamiento como tal del proyecto, permite tener una presentación estética del prototipo.

Los únicos elementos que no son realizados o diseñados en madera son la banda transportadora los tambores y las resbaladeras.

La banda transportadora está diseñada en caucho elástico no lisa, para poder transportar la carga en este caso la simulación de prendas hospitalarias. La longitud de la banda transportadora es de 50 cm, las prendas recorrerán esa distancia hasta el clasificador. Los tambores que permiten el giro de la banda transportadora están recubiertos de material rugoso (lija), para un correcto agarre y circulación de la banda transportadora. Los ejes de los tambores están constituidos de material plástico y de metal para que la circulación de la banda no se afecte. A lo largo de la banda transportadora se encuentran ejes que evitan que la carga no se salga del camino con facilidad y llegue al punto de clasificación correctamente.

Las resbaladeras son de material liso y resbaloso como aluminio, esto permite que los productos clasificados recorran sin dificultad hasta su punto de entrega. Las resbaladeras están ubicadas con 50 grados de inclinación para que la gravedad actúe y entregue las prendas clasificadas a su punto de recolección.

La cámara web estará ubicada a 15 cm del inicio del recorrido, dando una distancia de 35 cm a las prendas para poder ser clasificadas. el recorrido de 35 cm da el tiempo suficiente a los algoritmos tanto de activación de los actuadores como del sistema de visión artificial y la comunicación entre los sistemas para que realicen las acciones pertinentes tanto para la clasificación como para la verificación de datos en la interfaz gráfica.

El panel de visualización cuenta con 8 leds que indicaran diferentes estados entre los cuales se encuentran, activo, paro, estados de completado de proceso y llenado de lavadoras, pasando por la activación de leds para la indicación de detección de los tres

colores. La figura 2.8 muestra la estructura que se utilizó para la representación del prototipo.

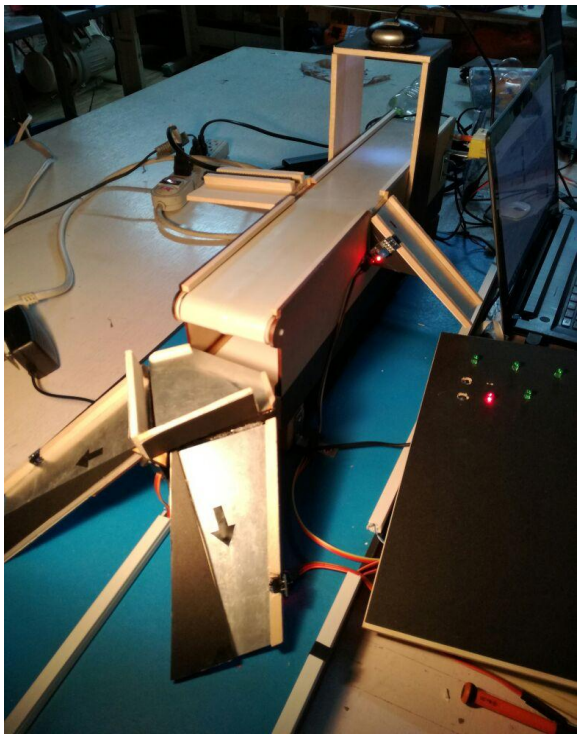


**Figura 2.8 Estructura Maqueta**

Fuente: Elaborada por el autor

## 2.5 DISEÑO DEL SOFTWARE

### 2.5.1 VISION ARTIFICIAL

El proceso de visión artificial realizado en este proyecto basa su funcionamiento en el modelo general de Marr-Palmer. Este modelo indica que existe una fase de captura o fase sensorial, el reproceso de la imagen, la segmentación y el reconocimiento.

La fase sensorial la realizará la cámara web, enviando imágenes hacia el Matlab para que dentro del código sea procesada, las imágenes serán enviadas en tiempo real; eso quiere decir, que una vez activada la cámara enviara las imágenes necesarias para que el código realice la detección.

Para la activación de la cámara y del envío de datos se debe seleccionar el puerto al cual está conectado la cámara web, cuando se selecciona el puerto conectado, se procede a encontrar el formato con el cual la cámara web envía los datos, esto se puede regular o

encontrar usando el formato por defecto con el cual cuenta la cámara. De la entrada de video se selecciona la resolución con la cual la cámara envíe los datos correspondientes. Se debe seleccionar una resolución adecuada, al escoger una resolución de imagen alta será más sencillo el proceso de detección de los colores, porque habrá más información para realizar la detección, pero al contar con una alta resolución el procesamiento de las imágenes será mucho más complejo llegando a consumir gran cantidad de recursos computacionales. Pero si se escoge una resolución demasiado baja, los colores pueden llegar a no distinguirse o confundirse entre sí.

Dentro de la adquisición se debe configurar una toma "infinita" de cuadros debido a que se debe simular lo realizado por el ojo humano, una toma infinita de cuadros emula un video en tiempo real, el proceso y el número cuadros tomados para el reproceso puede ser controlado, pero es importante que el video se mantenga activo durante toda la sesión de visión artificial.

Para la fase sensorial es importante también elegir el espacio de color con el cual se va a trabajar, para este proyecto se ha seleccionado un espacio de color RGB debido a su facilidad de trabajo y mayor familiarización con este espacio, cabe recalcar que si se utiliza un diferente espacio de color únicamente variaría el reproceso de la imagen, existiendo dificultad en el reconocimiento de cada uno de los colores. La detección de las características de los objetos influirá directamente en el espacio de color óptimo para el trabajo, en este caso las características que se busca analizar son los colores de los objetos, siendo de mejor utilidad el trabajo en un espacio de color RGB.

Para el reproceso se debe colocar una cantidad limitada de cuadros que tomará de todos los que entrega la cámara dentro de la fase sensorial. Los cuadros tomados serán analizados uno a uno hasta encontrar el cuadro útil para su reconocimiento. Es importante limitar el número de cuadros ya que puede llegar a dar lazos infinitos dentro del código impidiendo su cierre y finalización normal, obligando a un reinicio del programa Matlab, evitando la continuidad y dinamismo que piden estos procesos de clasificación con bandas transportadoras.

Dentro de esta fase es importante resaltar las características de los elementos que se van a analizar, de los miles de cuadros que se toman se deben verificar cuales son de interés y cuáles no. Dentro de una imagen pueden existir más detalles finos de lo que la fase de reproceso necesita para el reconocimiento, esta fase omite la búsqueda de

elementos pequeños o finos de las imágenes capturadas, analizando únicamente lo importante como lo es el color de las prendas hospitalarias.

El filtro realizado por la fase de reproceso también busca resaltar el centroide de las imágenes detectadas ya con un color específico. El centroide centrara y enmarcara la zona en la cual se encuentra detectado el color, esto posibilita la utilización de diferentes prendas con diferentes indicativos y diferentes tamaños brindando versatilidad en el reconocimiento y la utilización de la visión artificial.

La fase de segmentación aparta la imagen tomada en el reproceso separando cada fotograma restando sus diferentes colores principales RGB, para ser comparados en cada vector de su espacio de color. Esta segmentación se la realiza a nivel de imagen ya que se van tomando características principales de cada color para después ser analizados y comparados por separados. Debido a que existe retroalimentación constante en cada una de las fases, mientras se va realizando la segmentación de la imagen es necesario ir eliminando características innecesarias para el análisis de las imágenes como fotogramas con cierta cantidad inferior de pixeles, o filtrar el ruido presente en todas las imágenes.

La segmentación de la imagen también involucra una fase de binarización para realizar un análisis de una imagen en escala de grises.

Una vez segmentada y binarizada la imagen se realiza la comparación y reconocimiento de los colores, el reconocimiento también involucra una asignación de valores a los pines para realizar la comunicación entre Matlab y el microcontrolador que controle esa fase del diseño del proyecto.

La comparación para la finalización del proceso de visión artificial busca las características propias de cada imagen. Si no ha detectado nada el proceso se realimenta hasta llegar a la fase de adquisición de datos o fase sensorial y comenzar nuevamente.

En la figura 2.9 muestra el diagrama de flujo que debe seguir la imagen hasta llegar a un reconocimiento del color.

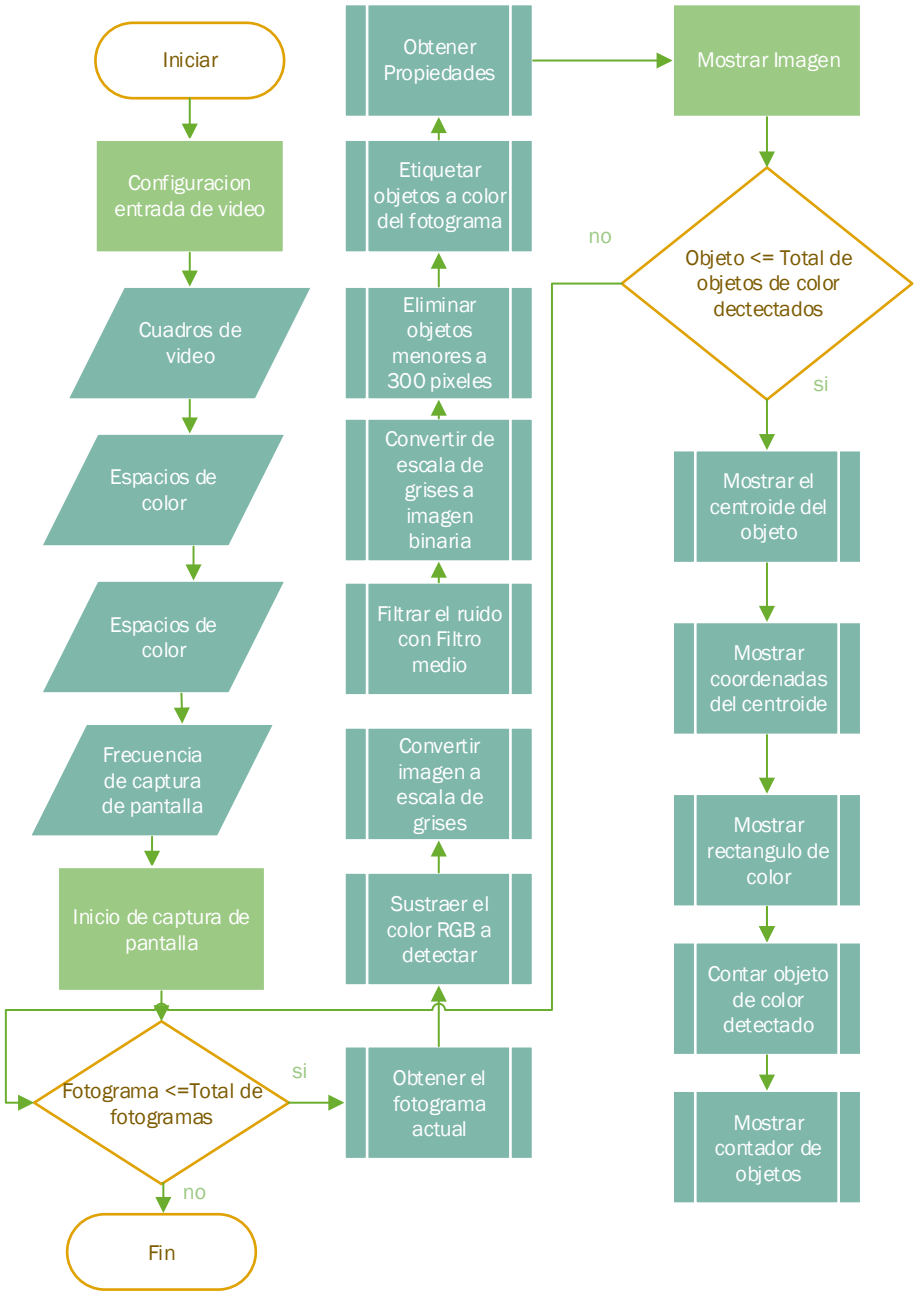



**Figura 2.9 Diagrama de flujo, proceso visión artificial**

**Fuente:** Elaborado por el autor

### 2.5.2 FASES PROGRAMACIÓN ARDUINO

Para la programación del algoritmo de reconocimiento de color utilizaremos MATLAB porque tiene todas las herramientas necesarias para el procesamiento de

imágenes y comunicación directa con Arduino lo cual permite integrar hardware y software directamente.

#### 2.5.2.1 Fase 1: Procesamiento de los datos generados en Matlab

Para la programación de esta fase no es necesaria la utilización del programa de Arduino ya que la programación se la realizará directamente en Matlab y será enviada al Arduino Uno. Después de todo el proceso de visión artificial realizado basado en el algoritmo de Marr-Palmer se obtiene una identificación del color perteneciente a un tipo de prenda hospitalaria, esta información debe ser interpretada por parte del Arduino Uno.

Algunas de las acciones que deberá tomar este Arduino son: encender los leds de reconocimiento de color y enviar información digital al segundo Arduino con el objetivo de realizar la comunicación con los demás procesos.

El hecho de realizar el proceso de visión artificial dentro de lenguaje Matlab limita las funcionalidades del Arduino. La complejidad del proceso que realiza la visión artificial no permite que el Arduino reaccione a otros eventos. Se puede decir que el Arduino Uno queda prácticamente bloqueado mientras Matlab realiza la detección del color. Los procesos sencillos como colocar salidas digitales en alto o en bajo una vez finalizada la detección de los colores no limita el accionar de los demás procesos como apuesta en marcha de los diferentes motores.

Para poder realizar la comunicación entre Arduino y Matlab se utiliza el programa adioes.pde, este programa es necesario para que la placa Arduino sea capaz de reconocer los comandos que se envían desde Matlab ya que contiene funciones y librerías necesarias para la interacción entre estas dos tecnologías. La principal ventaja que presenta este programa es que permite realizar programas o configuraciones dentro de Matlab que podrán ser interpretadas y permitirán el accionar de Arduino. Tomando en cuenta que siempre existirán funciones de Matlab o accionares del mismo que no podrán ser interpretadas por Arduino. Para el caso de este proyecto solo se utilizarán funciones para colocar salidas en alto o bajo.

#### 2.5.2.2 Fase 2: Procesamiento de los datos generados en Matlab

Esta segunda fase se centra en el accionamiento del servo motor, el cual clasifica las prendas dependiendo del tipo de prenda, estos dos caminos de selección básicamente determinan si la ropa a clasificar está contaminada o no. Después de la detección de color se debe tomar un tiempo de espera hasta que llegue la carga a la zona de clasificación. Este tiempo de retardo limita el funcionamiento del Arduino.

Estos tiempos son sincronizados y calibrados para que la carga que llegue sea clasificada de manera correcta, durante este tiempo de espera entre la detección del color y la clasificación de la prenda no se pueden realizar acciones por parte del Arduino.

Al ser un Arduino específico para el accionar del servo motor no debe realizar ninguna comunicación ni con Matlab ni con LabVIEW, pero si recibe la información de la detección por parte del Arduino Uno.

#### 2.5.2.3 Fase 3: Sistemas de comunicación, reconocimiento de los datos de sensores, muestra datos en el panel de visualización y accionar motor DC

Este Arduino contempla la mayoría de las acciones presentes en el prototipo. Recibe la información por parte del Arduino Uno con respecto a la detección de color que realiza el sistema de visión artificial, esta información la transmite hacia la interfaz gráfica de Laview para poder realizar la visualización del proceso que se está llevando a cabo. La interfaz gráfica toma la información que le entrega el Arduino Mega y realiza las operaciones en su interfaz, pero no solamente recibe información, LabVIEW también emite información útil para el accionar de los diferentes elementos.

La información que envía LabVIEW es de importancia ya que principalmente emite la señal PWM para accionar el motor DC y por consiguiente los tambores que mueven la banda transportadora. Esta interfaz también permite parar el proceso cuando el operador considere necesario.

La interfaz LabVIEW recibe los datos de los sensores infrarrojos que realizan el conteo de las prendas que han sido ubicadas en cada uno de los estantes clasificados. Esta información permite accionar indicadores dentro de la interfaz permitiendo saber al

operador si se encuentra lleno o no el estante. Si se encuentra lleno se activarán procesos que simulan el lavado de las prendas.

Además, este Arduino permitirá activar las luces de aviso de los diferentes procesos, incluyendo los de aviso de detección de color, tomando información que viene dada por la fase 1.

### 2.5.2.4 Interacción de las 3 fases

La figura 2.10 muestra en resumen la interacción entre los diferentes Arduinos y sus respectivas fases.

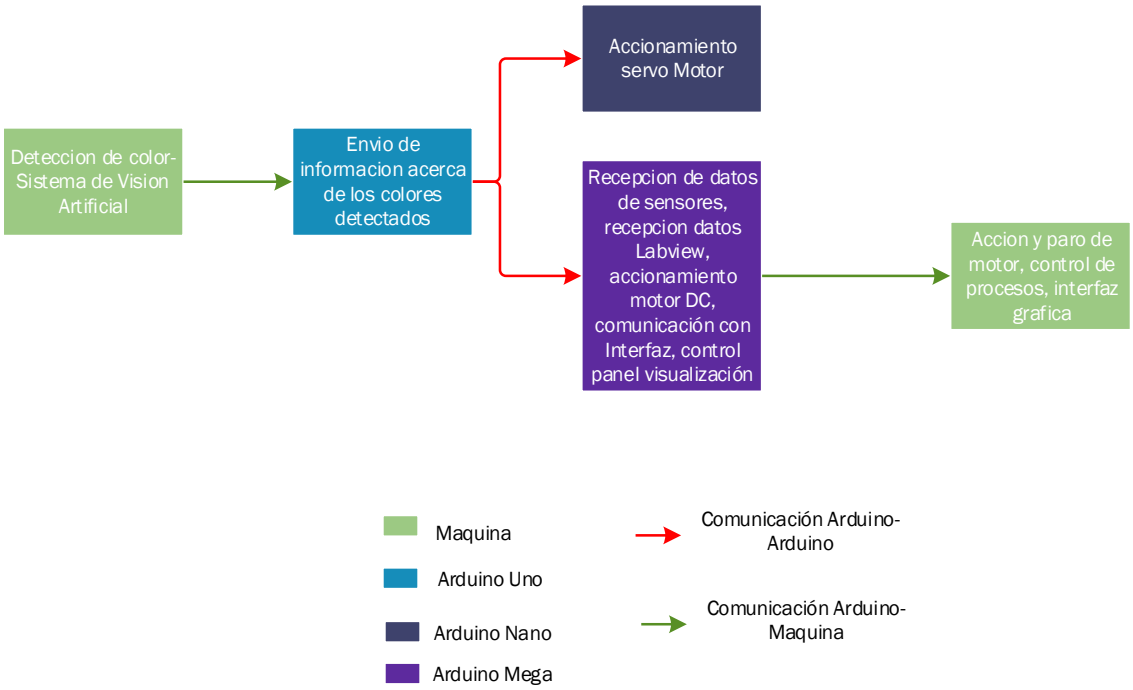



**Figura 2.10 Interacción entre Fases de procesamiento y sus respectivos Arduino**

Fuente: Elaborado por el autor

### 2.5.3 INTERFAZ GRAFICA DE USUARIO

La interfaz gráfica de usuario se la desarrollará en LabVIEW y deberá contener los siguientes elementos:

- Botón de marcha del proceso.
- Botón de parar el proceso.
- Contador de prendas no contaminadas de adultos.
- Contador de prendas no contaminadas de niños.
- Contador de capacidad de lavadora.
- Indicador de errores de comunicación.
- Selección del puerto de comunicación.
- Indicador de detección de color verde
- Indicador de detección de color azul.
- Indicador de detección de color rojo.
- Temporizador del tiempo de lavado.

La interfaz deberá contener todos estos elementos y adicionalmente debe ser agradable y de fácil utilización para el usuario. Los operadores del proceso de lavado en las entidades médicas no suelen ser personas que manejan con gran capacidad el área técnica, por esto la interfaz debe ser de fácil utilización para este tipo de personas.

También es importante controlar que la comunicación entre LabVIEW y Arduino se realice de manera correcta. Arduino como se explicó en la fase 1 y 3 envía los datos y la información de los colores detectados a través de una comunicación Arduino Arduino, pero la visualización de los colores detectados por parte de los operadores se realizará mediante esta interfaz, si no existiera una correcta comunicación Arduino LabVIEW, no se pudiera apreciar ninguno de los procesos contenidos en la clasificación de prendas.

A pesar de que no existe una comunicación directa entre lo realizado en Matlab y la interfaz en LabVIEW, si es importante que ambos procesos estén en correcto funcionamiento, si alguno de los dos falla el funcionamiento correcto del otro se verá afectado, por ende el proceso de clasificación no se realizara.

# CAPÍTULO III
# IMPLEMENTACIÓN

## 3.1 DESARROLLO DEL SISTEMA

Los elementos utilizados para el desarrollo del hardware del sistema son los siguientes:

- Motor DC pololu
- Sensores infrarrojos
- Cámara Web
- Leds indicadores
- Arduino uno
- Convertidor DC-DC
- Servo motor

### 3.1.1 Diagrama Circuital

La Figura 3.1 muestra el diagrama circuital de los elementos descritos en la sección anterior. Este diagrama permitió realizar las incorporaciones dentro de la maqueta y las respectivas conexiones para su funcionamiento.

El diagrama responde básicamente a facilidad y ubicación de los pines, considerando tanto entradas analógicas o digitales, de igual manera salidas analógicas y digitales.

Para el funcionamiento de los motores tanto servo motor como motor DC, se utilizan salidas PWM o analógicas, el número de pines utilizados responden a las características descritas en la sección 1 donde se encuentran detallados que pines pueden ser utilizados tanto en el Arduino Mega como en el Arduino Nano. Se seleccionó una de estas salidas para el accionar del motor DC en el Arduino Mega y para accionar el servo motor uno de los pines del Arduino Nano.

Los sensores están especificados en el diagrama circuital y estarán conectados a entradas digitales, porque los sensores entregan ese tipo de información.

La información que recibe el Arduino Uno donde se entrega los datos pertenecientes a la detección de los colores la computadora la entrega a través del cable

serial USB. Como se describió en la sección uno, este cable no solo sirve para alimentar al Arduino, sino también para realizar la comunicación.

Arduino Uno tendrá la información de los colores detectados de forma digital en los pines 5, 6 y 7. Esta información se envía a los Arduinos Nano y Mega. El Arduino Nano recibe esta información en los pines 2, 3 y 4 respectivamente, mientras que el Arduino Mega recibe esa información en lo pines 25,27 y 29.

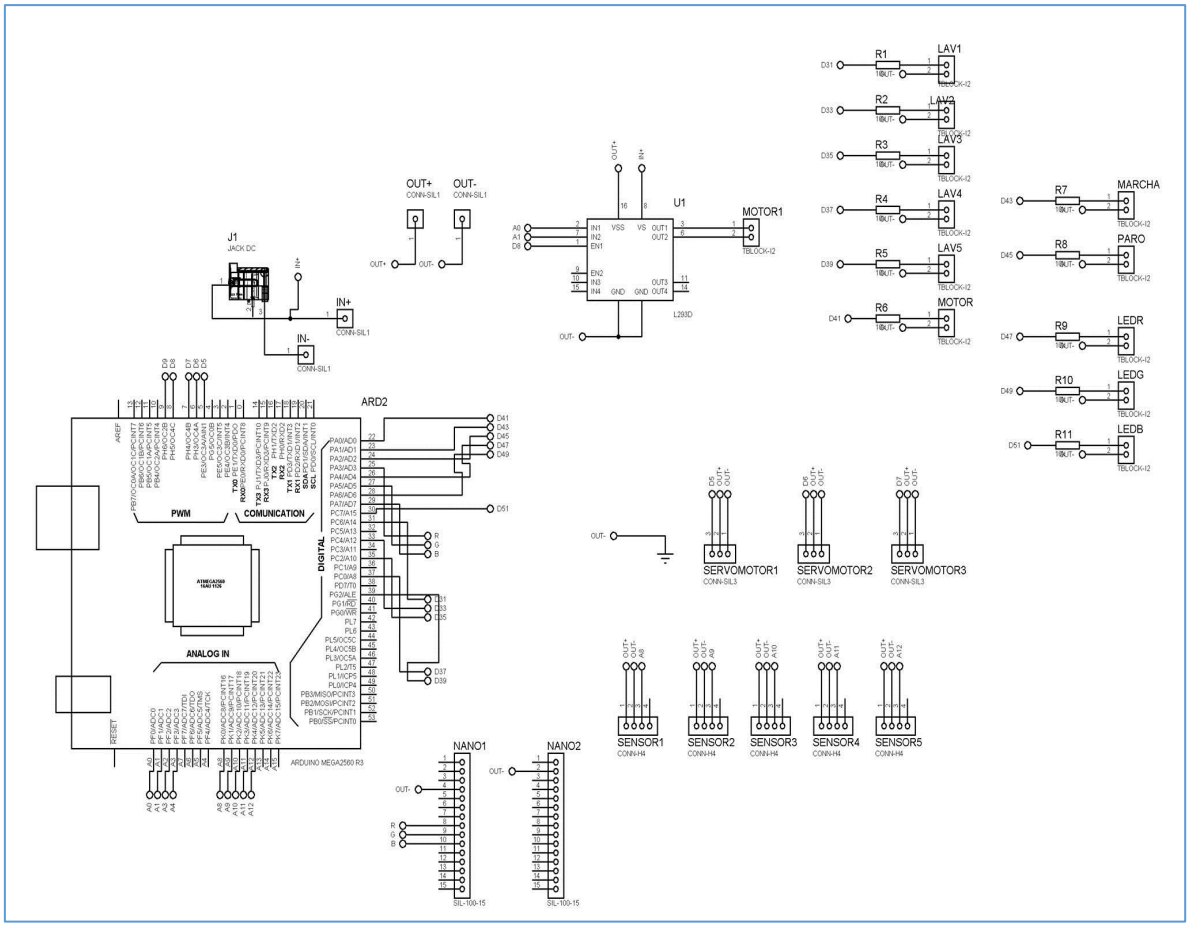



**Figura 3.1 Diagrama circuital, distribución de pines**

Fuente: Elaborado por el autor

## 3.2 DESARROLLO DEL SOFTWARE

### 3.2.1 Sistema de Visión artificial

La Figura 2.7 muestra el proceso que debe seguir el sistema de visión artificial basado en el algoritmo de Marr-Palmer, esta sección explicara cómo se desarrolló este algoritmo dentro de Matlab.

Se inicia eliminando todos los puertos válidos para asignar un nuevo puerto de comunicación a la cámara y al Arduino, este proceso se lo muestra en la línea de Código 3.1:

```
delete(instrfind({'Port'},{'COM6'}))
```

**Código 3. 1** Inicialización de puertos.

Fuente: (Mathworks, 2018)

Una vez eliminado todos los puertos el propio sistema asigna un nuevo puerto a la cámara para proseguir a la toma del formato de imagen que entrega la cámara web como se muestra en la línea de Código 3.2.

```
a = imaqhwinfo;
```

**Código 3. 2** Guardado del formato.

Fuente: (Mathworks, 2018)

La variable “a” guardara este formato y todas las salidas o acciones a realizar se darán en base a este formato junto con esta variable.

Como ya se habían eliminado los puertos previamente se selecciona una variable para que sea asignada al puerto de comunicación donde estará ubicado el Arduino Uno. Esta asignación se realiza con el siguiente comando mostrado en la línea de Código 3.3:

```
b= arduino('COM6');
```

**Código 3. 3** Puerto de comunicación.

Fuente: (Mathworks, 2018)

La variable “b” contendrá el puerto de comunicación, en este caso el 6. Para poder inicializar la cámara y consecuentemente la captura de fotogramas, se debe activar la misma y colocar la resolución de la misma, esto se lo ejecuta como se indica en la línea de Código 3.4 mostrado a continuación.

```
vid = videoinput('winvideo', 1, 'YUY2_160x120');
```

**Código 3. 4** Captura de fotogramas.

Fuente: (Mathworks, 2018)

Como se había mencionado al activar el video, se deben configurar las propiedades de los objetos captados por la cámara, entre estas propiedades se encuentra la captura infinita teniendo un video continuo. Esto se realiza con el comando mostrado en la línea de Código 3.5.

```
set(vid, 'FramesPerTrigger', Inf);
```

**Código 3. 5** Captura de propiedades.

Fuente: (Mathworks, 2018)

Las otras propiedades de los objetos previos al proceso que se deben configurar son el espacio de color y la propiedad de la frecuencia de fotograma. Estos dos procesos se los realiza utilizando en las líneas de Código 3.6.

```
set(vid, 'ReturnedColorspace', 'rgb')
vid.FrameGrabInterval = 2;
```

**Código 3. 6** Captura de frecuencia y espacio de color.

Fuente: (Mathworks, 2018)

Cabe recalcar que en el proceso de inicialización del video se colocó el nombre del objeto como “vid”, para poder modificar las propiedades existen los dos comandos presentados.

A continuación, se inicia el video, el inicializar las propiedades y parámetros no garantiza la inicialización del video como tal, para esto debemos realizarlo con el siguiente comando que se muestra en la línea de Código 3.7.

```
start(vid)
```

**Código 3. 7** Inicialización del video.

Fuente: (Mathworks, 2018)

En este punto finaliza la primera etapa que es la fase sensorial donde se encuentra el primer condicional. Este condicional fija un valor de imágenes capturadas, esto dará un tiempo de utilización al algoritmo e indicara el máximo número de imágenes que se procesaran. Este condicional se muestra en la línea de Código 3.8

```
while(vid.FramesAcquired<=2000)
```

**Código 3. 8** Condicional.
Fuente: (Mathworks, 2018)

El parámetro vid.FramesAcquired es un contador que guarda la cantidad de fotogramas que emite la cámara, si tal cantidad de cuadros supera al número indicado en el condicional, en este caso 2000, el proceso finalizara.

Mientras en el condicional no acabe el proceso no podrá finalizar por lo que seguirá analizando, segmentando y reconociendo colores.

Si el condicional anterior se cumple, comenzara el siguiente punto del algoritmo de visión artificial que es el reproceso y segmentación. Se toma primeramente la captura de uno de los fotogramas y se la almacena para su próximo análisis como se muestra en la línea de Código 3.9.

```
data = getsnapshot(vid);
```

**Código 3. 9** Reproceso y segmentación.

Fuente: (Mathworks, 2018)

Una vez tomada la captura se deben separar en tres nuevos elementos, extrayendo el color rojo, azul y verde de la imagen original. Esta separación es importante para poder analizar las propiedades por separado de cada fotograma.

Se realizará el ejemplo de segmentación para el color rojo, siendo similar al proceso que se deberá seguir para los demás colores.

Para obtener un vector de elementos color rojo, se resta de la imagen original el componente rojo, pero de la imagen transformada a escala de grises. Esto se realiza en la línea de Código 3.10.

```
red_im = imsubtract(data(:,:,1), rgb2gray(data));
```

**Código 3. 10** Vector de elementos color rojo.

Fuente: (Mathworks, 2018)

Los elementos color rojo serán extraídos de la imagen en escala de grises que se guarda como vector de valores. Mediante un filtro se elimina el ruido tratando de mejorar la imagen obtenida.

La imagen en escala de grises quitada el color rojo, se binariza para ser analizada bit a bit, el formato de una imagen binaria es bw, la binarización de la imagen se realiza en la línea de Código 3.11.

```
red_im = im2bw(red_im,0.18);
```

**Código 3. 11** Vector binario de elementos color rojo.

Fuente: (Mathworks, 2018)

Las imágenes contienen pixeles no significativos, así que se eliminan todas las imágenes que contengan menos de 300, para que la detección de color no se vea afectado por pixeles no significativos. Esto se realiza en la línea de Código 3.12.

```
red_im = bwareaopen(red_im,300);
```

**Código 3. 12** Pixeles no significativos.

Fuente: (Mathworks, 2018)

En la línea de Código 3.13 se crea una matriz de etiquetas de las mismas dimensiones, pero agrupados en elementos de 8, verificando que para ser agrupados 8 elementos deben estar conectados entre sí, la conectividad predeterminada es 8.

```
bw_red = bwlabel(red_im, 8);
```

**Código 3. 13** Matriz de etiquetas.

Fuente: (Mathworks, 2018)

Después de todo este proceso el análisis para la detección no va a ser realizada bit a bit de los vectores, se realizará la comparación con las propiedades que se extraen de estas matrices. Para extraer las propiedades se utiliza el comando mostrado en la línea de Código 3.14.

```
stats_red = regionprops(logical(bw_red), 'BoundingBox', 'Centroid');
```

**Código 3. 14** Comparación de propiedades.

Fuente: (Mathworks, 2018)

Este comando busca generar las propiedades, pero dentro de un centroide de la imagen permitirá enmarcar si es que se ha detectado un color, como se explicara más adelante.

El vector *stats_red* contendrá la información del color seleccionado a analizar, si dentro de todos los procesos, agrupaciones, filtros y eliminación de valores no significativos no se encuentran valores dentro de este vector significara que no se ha detectado ningún pixel con ese respectivo color.

Para los demás colores es exactamente el mismo proceso, variando únicamente en la resta del color, el cual toma diferentes planos del espacio RGB. Los vectores resultantes para los dos colores faltantes (azul, verde).

Una vez almacenado la información, llega el segundo condicional este condicional se aplicará a cada color. Este condicional verifica si existen elementos dentro del vector de propiedades de cada color. Si el vector está vacío significara como se había mencionado antes que no se ha detectado ningún elemento del color respectivo. La verificación la realizara para todos los colores y se detendrá cuando halle elementos dentro del vector de propiedades. El condicional se muestra en la línea de Código 3.15.

```
for object_red = 1:length(stats_red)
```

**Código 3. 15** Segundo condicional.

Fuente: (Mathworks, 2018)

Basta que la longitud del vector sea uno para que entre al lazo y realice las acciones, pero es poco improbable que después de todos los procesos previos se realicen reconocimientos tan pequeños de color.

Los comandos mostrados en las líneas de Código 3.16 se ejecutan para enmarcar e imprimir la imagen enmarcada en la zona donde se encontró el color.

```
bb_r = stats_red(object_red).BoundingBox;
bc_r = stats_red(object_red).Centroid;
rectangle('Position',bb_r,'EdgeColor','r','LineWidth',2)
plot(bc_r(1),bc_r(2), '-m+')
```

**Código 3. 16** Enmarcación e impresión de imágenes.

Fuente: (Mathworks, 2018)

El proceso detecta el centroide ubicado en los procesos anteriores, y dibuja un rectángulo en la posición donde el vector de propiedades detecto color, en este caso rojo. La impresión de la imagen en un cuadro de imagen se realizará del fotograma original capturado en el primer proceso junto con el rectángulo creado en esta etapa. El rectángulo se dibujará del color detectado, si es rojo el color detectado el rectángulo se ubicará con centro en el centroide, dimensiones dadas por el vector de propiedades del color detectado y con un color rojo en los bordes. Ubicando específicamente la zona donde se detectó el color.

Cabe aclara que la detección se puede realizar de varios colores, es decir si un elemento contiene dos colores (rojo y verde), se dibujará la zona y el rectángulo de cada color, pero para las acciones correspondientes por parte de los actuadores como el servo motor, la clasificación se realizará en base al color más significativo el cual será detectado primero por las dimensiones de su vector. Se procurará que los colores sean únicos para que no existan conflictos en las decisiones del actuador.

Si en el proceso se detectó el color, entrará como ya se mencionó a un lazo para dibujar el rectángulo, dentro de este lazo también se activarán los pines del Arduino, activando los correspondientes al color detectado y apagando los demás pines, esta función se realiza en las líneas de Código 3.17.

```
b.pinMode(5,'output');
%while(true)
b.digitalWrite(5,1);
b.pinMode(13,'output');
b.digitalWrite(13,1);
pause(1);
b.digitalWrite(5,0);
b.digitalWrite(13,0);
pause(1);
```

**Código 3. 17** Activación pines Arduino.

Fuente: (Mathworks, 2018)

Se consideran pausas entre el encendido y apagado de los pines para que no exista conflicto en la comunicación Arduino-Arduino.

Una vez detectado el color y enviada la información hacia los pines el proceso se repetirá hasta que la cantidad de fotogramas cierre el primer condicional. Se puede subir y modificar el valor, pero para las pruebas correspondientes dentro de este proyecto se ha tomado ese valor referencial.

### 3.2.2 Programación en Arduino

La programación de la primera fase es decir Arduino se la realizo mientras el sistema de visión artificial estaba en funcionamiento.

Para la fase dos es decir el accionar del servo motor se sigue el siguiente diagrama mostrado en la figura 3.2

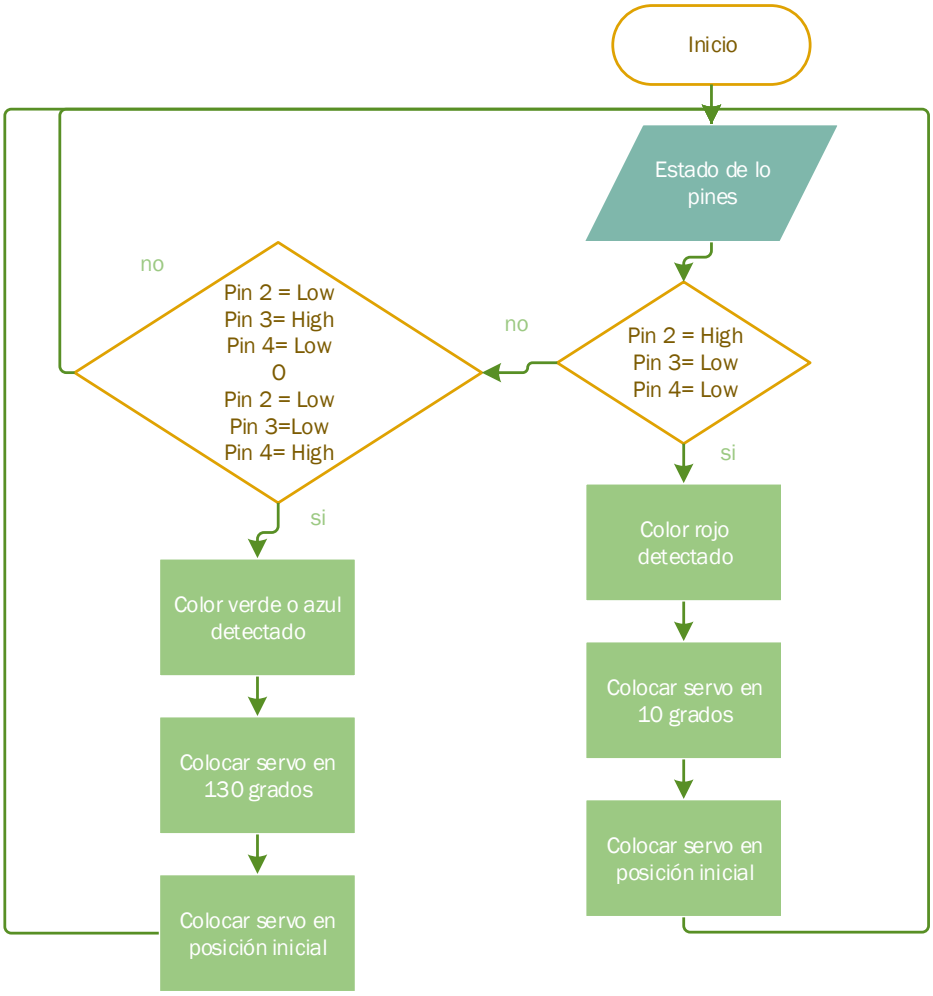



**Figura 3.2** Programación Arduino Nano

**Fuente:** Elaborado por el autor

Los principales comandos para realizar el proceso mostrado en la Figura 3.2 son los mostrados en las líneas de Código 3.18.

Para realizar el movimiento del servo motor se utiliza en primer lugar la librearía servo.h, una vez inicializado el servo motor se prosigue a realizar el movimiento, para luego realizar la lectura del estado de los pines.

```
#include <Servo.h>
Servo servo1;
servo1.attach(10);
botonStado1 = digitalRead(rojo);
 botonStado2 = digitalRead(verde);
 botonStado3 = digitalRead(azul);
```

**Código 3. 18** Activación del servo motor.

Fuente: (Mathworks, 2018)

Leyendo el estado de los pines y almacenándolos en variables del tipo booleana e pueden construir los diferentes condicionales el comando para realizar las condiciones dentro de Arduino está dado por la línea de Código 3.19.

```
if (val == LOW) {}
```

**Código 3. 19** Lectura del estado de los pines.

Fuente: (Mathworks, 2018)

El nombre de la variable puede ser cualquiera, pero en este caso debe ser de tipo booleana para que admita la condición de HIGH o LOW por lo que se deben anidar condicionales para que se cumplan las condiciones, el comando de los condicionales anidados es el mismo. Hay que considerar que existe un tiempo de retardo entre el reconocimiento del pin de entrada y el accionar del servo motor, como ya se había mencionado, pero la regresión a la posición inicial se realizara de manera inmediata, debido a que el servo debe estar listo para recibir una nueva carga y realizar la respectiva clasificación.

El Arduino Mega utilizara los mismos comandos para la lectura de datos que envía el Arduino Uno y de los estados de los sensores infrarrojos, así como del seteo de valores en los pines donde están conectados las luces LED.

Un comando adicional que utiliza Arduino Mega es la comunicación serial necesaria para la comunicación entre Arduino y LabVIEW. El comando utilizado para emitir la información necesaria ya sea de los colores detectados como de los valores entregados por los sensores es el mostrado en la línea de Código 3.20.

```
Serial.print(cuenta2);
```

**Código 3. 20** Lectura del estado de los pines.

Fuente: (Mathworks, 2018)

Enviando el valor contenido de la variable de manera serial, como se explicará a continuación LabVIEW receptará esos datos y realizará los procesos pertinentes dentro de la interfaz devolviendo los valores necesarios para el accionar del Arduino Mega.

### 3.2.3 Programación en LabVIEW

Para la realización de la interface gráfica de usuario se utilizó el programa LabVIEW (ver Figura 3.3 y 3.4) ya que tiene la versatilidad de crear pantallas que son muy amigables con el usuario, permitiendo que sea fácil de entender para cualquier persona que haga uso del sistema implementado. Para la programación de la interface se empezó estableciendo la comunicación entre dispositivos, definiendo parámetros como la velocidad de la comunicación, el tamaño del dato y el tipo de paridad. En este proceso de configuración se debe definir cuantos datos se van a recibir para poder separar el arreglo de datos y enviarlos a cada uno de los indicadores. En la interface se muestra el estado de cada una de las partes del proceso que se está desarrollando, además se creó un pequeño sub programa que realiza un tiempo para el proceso de lavado una vez que se hayan llenado las lavadoras. De la misma manera el programa envía comandos a la tarjeta Arduino para detener o activar el proceso y comandos una vez que termina con el proceso de lavado para reiniciar el programa.

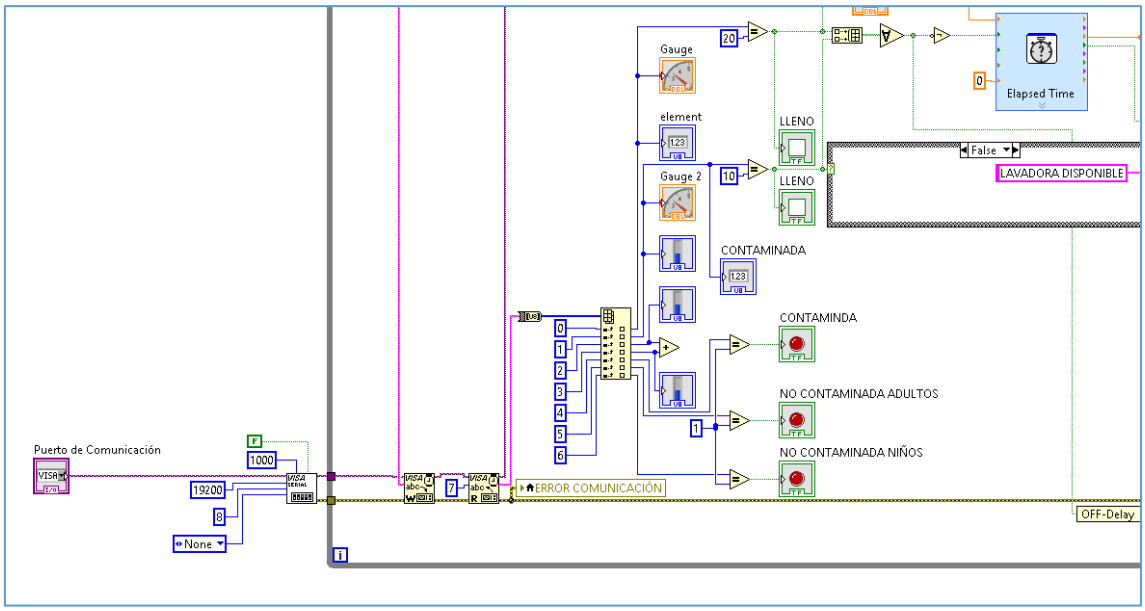



**Figura 3.3** Programación LabVIEW parte 1

Fuente: Elaborada por el autor

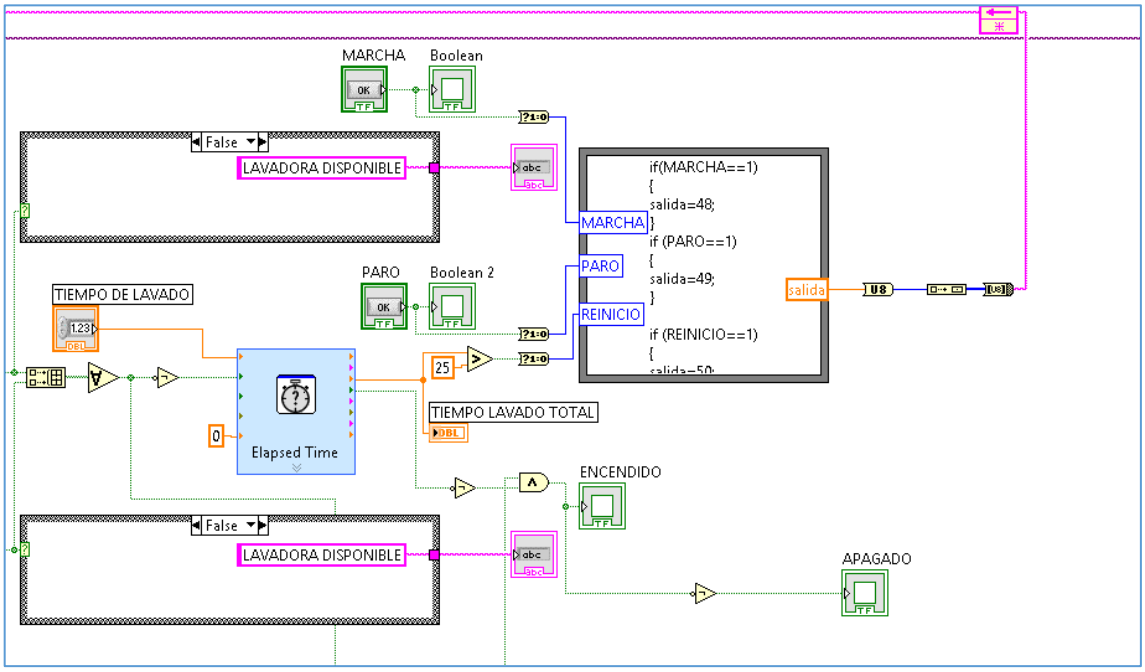



**Figura 3.4** Programación LabVIEW parte 2

Fuente: Elaborada por el autor

Para la realización del HMI a través de LabVIEW, se trabaja en el panel frontal donde se encuentra diferentes visualizadores permitiendo que la interface sea atractiva para el usuario y sobre todo que pueda entender lo que está pasando durante el desarrollo de la clasificación. La interface consta de botones de marcha y paro para controlar el proceso,

luces que indican que tipo de prenda de vestir y el inicio de lavado como se observa en la Figura 3.5.

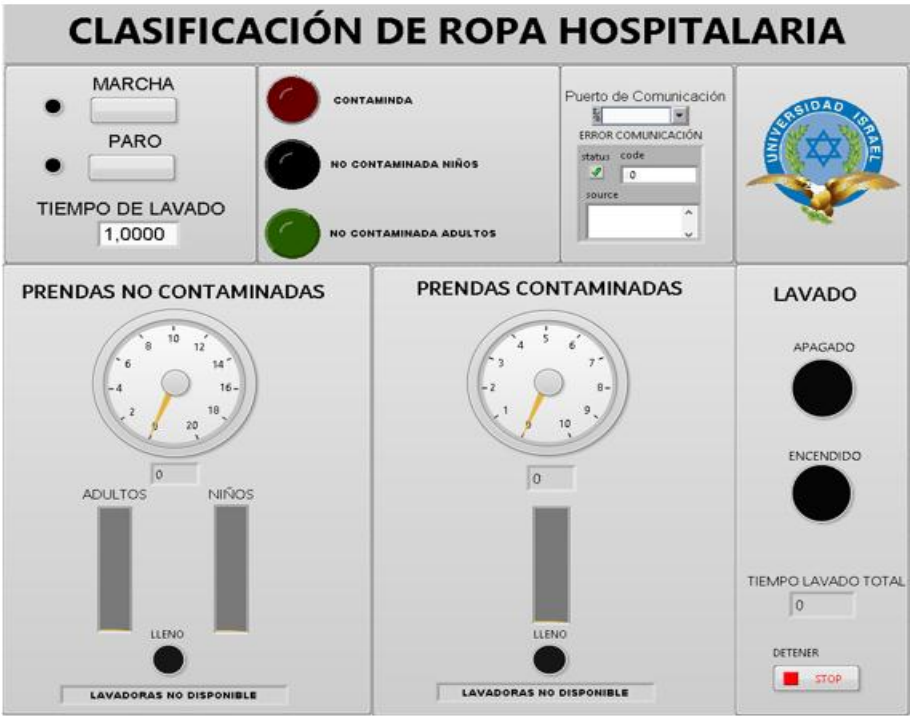



**Figura 3.5** Interfaz Gráfica de Usuario

Fuente: Elaborada por el autor

## 3.3 IMPLEMENTACIÓN DEL SISTEMA

Para el desarrollo del sistema después de realizar los códigos respectivos, se procedió a implementar el hardware del prototipo. En la Figura 3.6 muestra el motor DC pololu instalado en la maqueta.

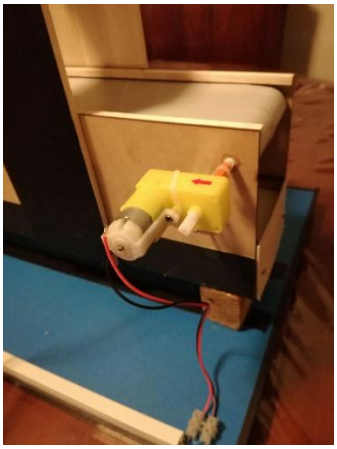


**Figura 3.6** Instalación motor DC

Fuente: Elaborada por el autor

Como se puede observar el giro está condicionado en el sentido de la flecha roja que se observa en la Figura 3.6. El controlador del motor está programado para que el sentido de giro sea único y con velocidad fija. Fue necesario poner estas condiciones debido a las acciones futuras que se deben tomar, estas acciones deben estar plenamente sincronizadas y si la velocidad o el giro se vieran afectados el proceso no se cumpliría con normalidad.

La Figura 3.7 muestra la instalación de uno de los sensores que están ubicados en cada una de las bandas alternativa de la maqueta, estos sensores realizaran el conteo del número de prendas con las que el proceso de lavado empezará.

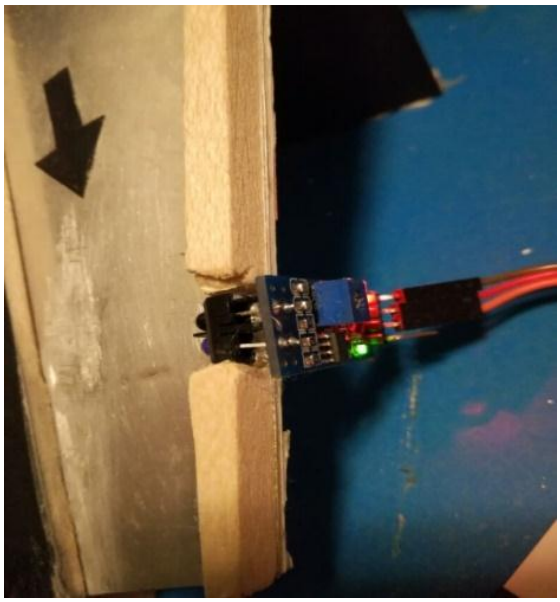


**Figura 3.7** Instalación sensores infrarrojos

Fuente: Elaborada por el autor

Las figuras 3.8 y 3.9 muestran la instalación de la cámara web del sistema de visión artificial.


**Figura 3.8** Instalación cámara web, parte inferior

Fuente: Elaborada por el autor

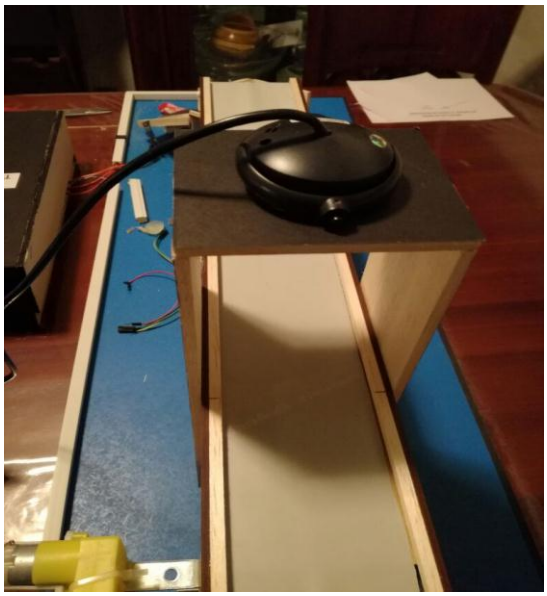


**Figura 3.9** Instalación cámara web, parte superior

Fuente: Elaborada por el autor

La figura 3.10 muestra el panel de notificaciones, este panel contiene las luces correspondientes a los siguientes procesos:

- Luz para indicar que está en marcha la Banda.
- Luz que indica que el proceso de lavado se está llevando a cabo.
- Luz para indicar que la lavadora uno está llena.
- Luz para indicar que la lavadora dos está llena.
- Luz que indica que el proceso de lavado se completó y se debe accionar la interfaz para comenzar la detección.

- Luz de color rojo detectado.
- Luz de color azul detectado.
- Luz de color verde detectado.


**Figura 3.10** Panel de notificaciones

Fuente: Elaborada por el autor

La figura 3.11 muestra la instalación del servo motor con dos paletas que empujaran la carga a los dos caminos correspondientes, en esta figura también se puede apreciar los materiales de las resbaladeras, que le dan la dirección necesaria a los elementos hacia los sensores.

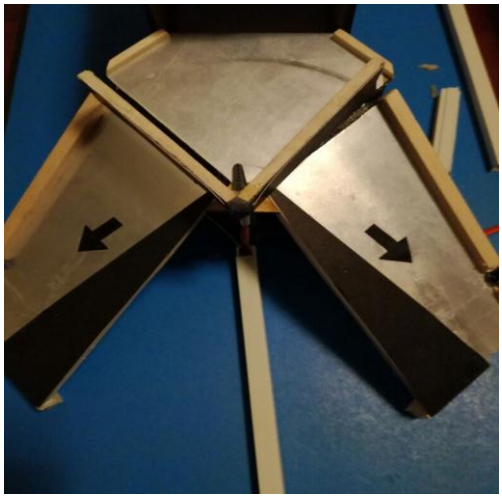


**Figura 3.11** Instalación servo motor

Fuente: Elaborada por el autor

En la Figura 3.12, se puede observar cómo quedó ya todo el sistema implementado

**Figura 3.12** Implementación final del prototipo

Fuente: Elaborada por el autor

## 3.4 PRUEBAS DE FUNCIONAMIENTO

En este proyecto se ha realizado el procesamiento de imagen por color, en el cual se ha obtenido diferentes resultados que se detallarán en este capítulo.

### 3.4.1 Resolución de Imagen

Existen diferentes factores que afecta en la calidad de la imagen, entre ellas se tiene el brillo, enfoque, el contraste o en este caso la resolución. Cuando existe mayor resolución de la imagen capturada mayor calidad tendrá, por lo tanto, el tiempo de procesamiento para realizar la detección del color aumenta. Por este motivo se escogió una resolución de

160x120, la cual es la resolución mínima que admite la cámara que realiza la toma de imágenes, la visualización de estos factores se muestra en las Figuras 3.13 y 3.14

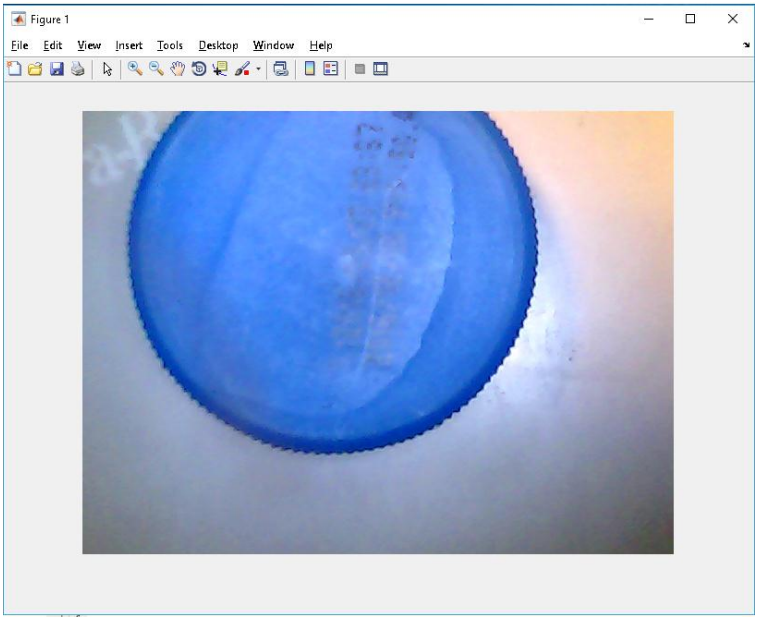



**Figura 3.13** Resolución de imagen capturada ejemplo

Fuente: Elaborada por el autor

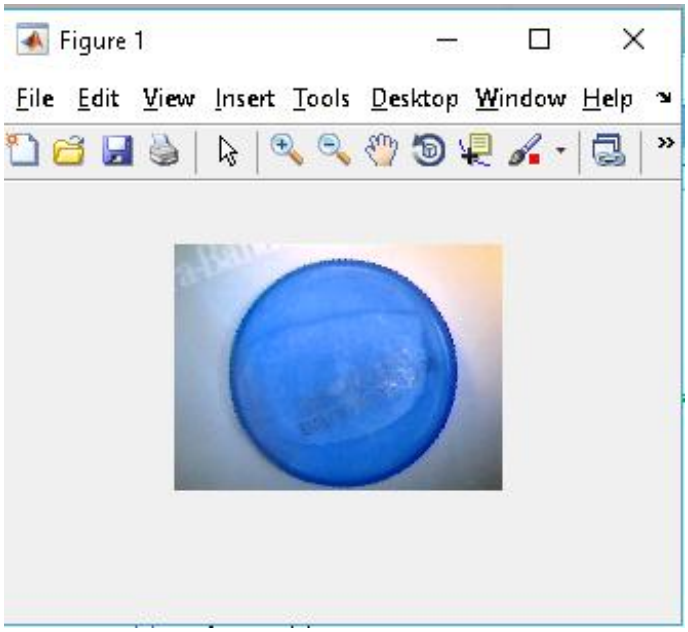



**Figura 3.14** Resolución de imagen captura da ejemplo

Fuente: Elaborada por el autor

### 3.4.2 Filtrado de la imagen

Existen diferentes tipos de filtros para reducir el ruido que tienen las imágenes digitales obtenidas por procesos previos como captura, digitalización y transmisión. El problema de utilizar filtros para la reducción del ruido es provocar difuminación en los bordes de la imagen procesada. La imagen que se muestra a continuación se observa la importancia de definir el tamaño de la máscara, mientras mayor sea se obtiene mayor reducción del ruido, por consiguiente, la imagen se torna difuminada. Para el filtrado se utilizó un kernel de 3x3, como se muestra en la Figura 3.15.

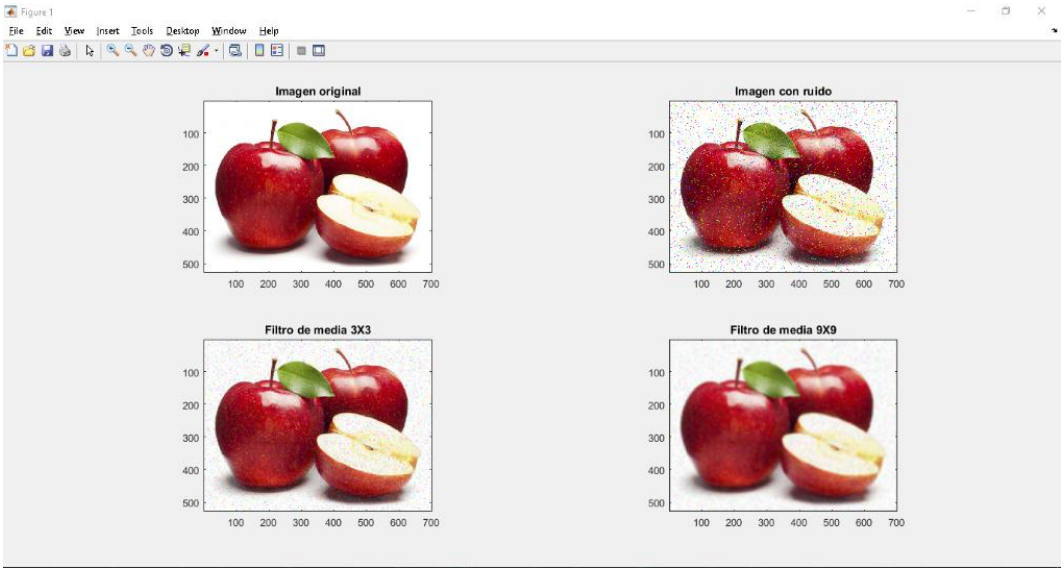



**Figura 3.15** Filtrado de Imagen

Fuente: Elaborada por el autor

### 3.4.3 Segmentación de color

Cuando se lee una imagen en el programa Matlab, es interpretada como tres matrices, cada una corresponde a los planos R, G y B. Para la segmentación se crea una nueva imagen con uno de los colores extraídos. El color extraído es el que posteriormente va a ser reconocido, como se muestra en la Figura 3.16.

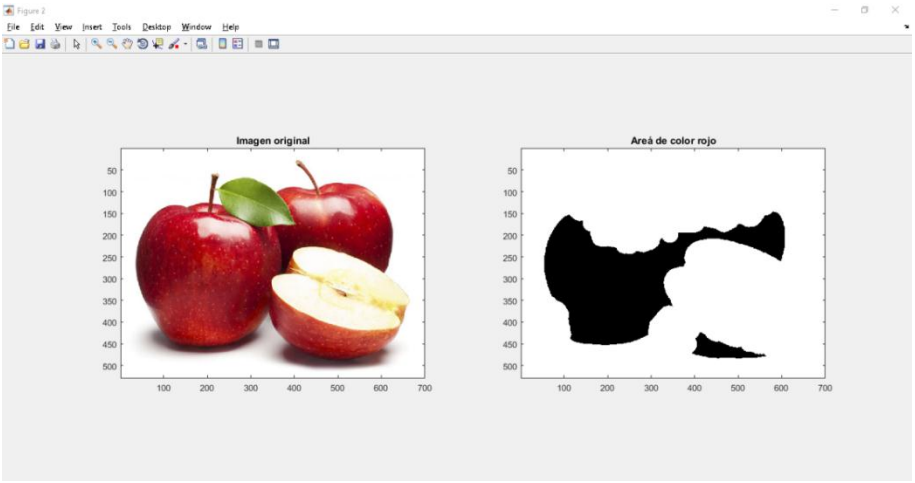



**Figura 3.16** Segmentación de color Rojo

Fuente: Elaborada por el autor

### 3.4.4 Seguimiento de colores

Para el seguimiento de colores se analiza en cada uno de los *frames* que toma durante la ejecución del programa, en cada una de las pasadas del lazo captura una imagen y realiza el procedimiento mostrado anteriormente filtrado, segmentación. Cuando el programa se ejecuta captura una imagen del video en tiempo real, el color es sustraído dependiendo el color que se estableció para la detección. La imagen capturada es convertida a escala de grises. Se descarta objetos en la imagen menores a 300 pixeles utilizando el método *bwareopen*. Un ejemplo del seguimiento de color se visualiza en las Figuras 3.17, 3.18, y 3.19.

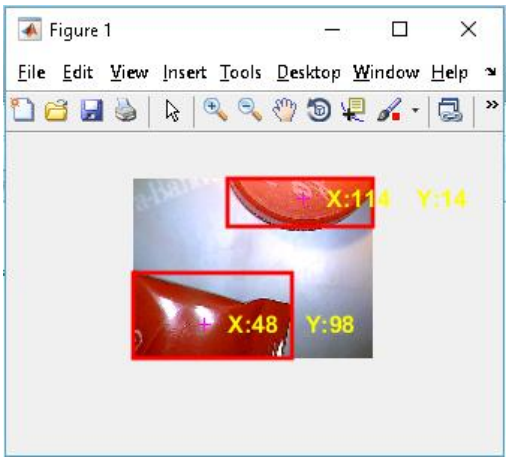



**Figura 3.17** Seguimiento de color Rojo

Fuente: Elaborada por el autor

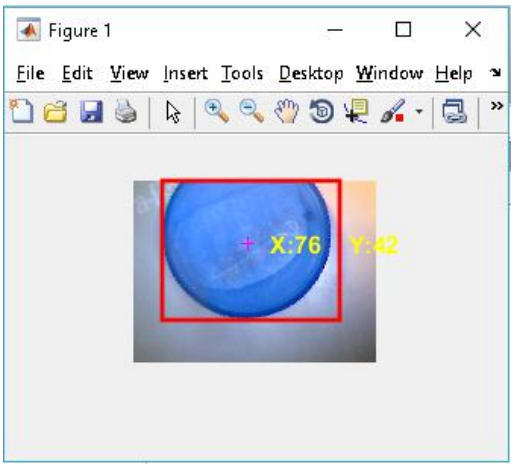



**Figura 3.18** Seguimiento de color Azul

Fuente: Elaborada por el autor

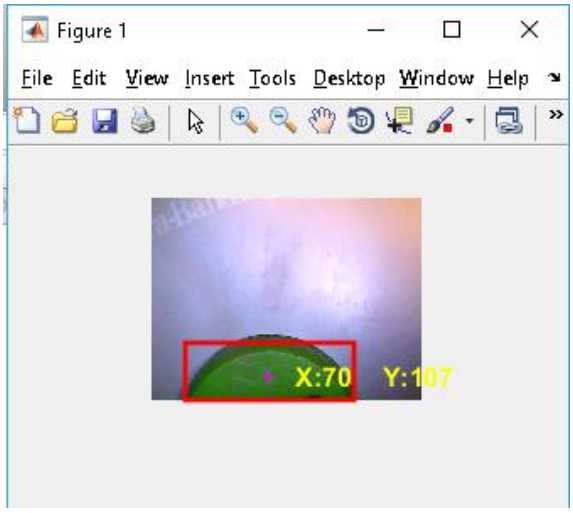



**Figura 3.19** Seguimiento de color Verde

Fuente: Elaborada por el autor

### 3.4.5 Comunicación

La comunicación que se estableció entre la computadora y el sistema de clasificación de objetos como se muestra en la Figura 3.20 es una comunicación serial, la cual es punto a punto. La velocidad de transmisión de los datos es alrededor de 19200 baudios permitiendo que no se pierda la comunicación entre dispositivos ya que se envían alrededor de 7 datos.

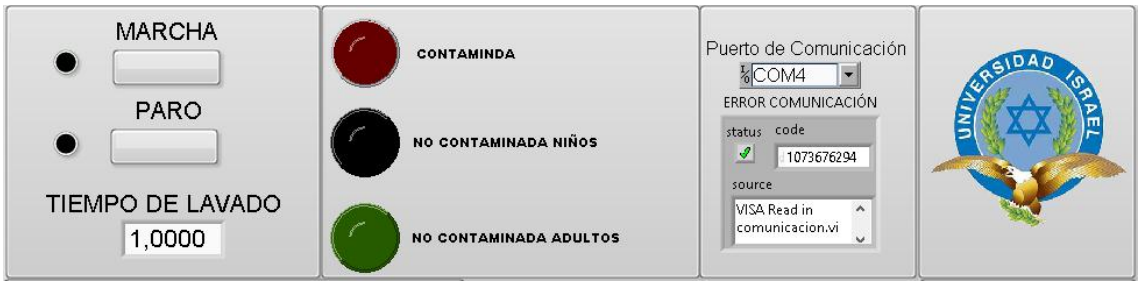



**Figura 3.20** Puerto y estado de comunicación

Fuente: Elaborada por el autor

Las Figuras 3.21, 3.22 y 3.23 Muestran el reconocimiento en el prototipo de clasificación

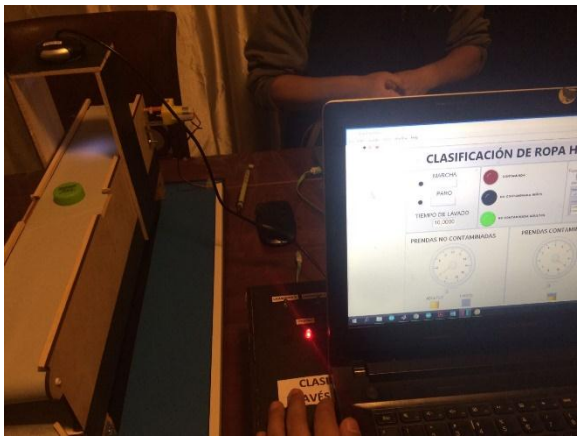



**Figura 3.21** Datos Obtenidos al detectar color verde

Fuente: Elaborada por el autor

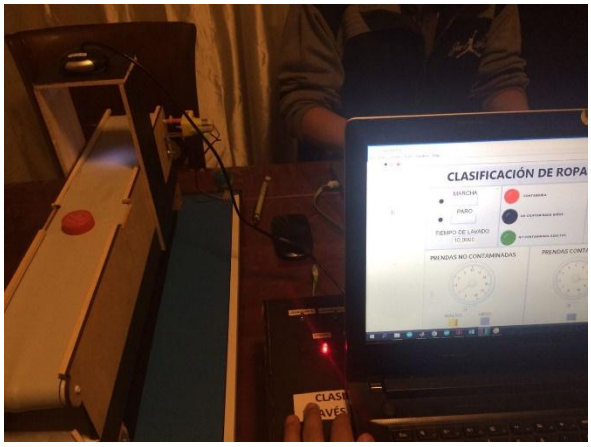



**Figura 3.22** Datos Obtenidos al detectar color rojo

Fuente: Elaborada por el autor

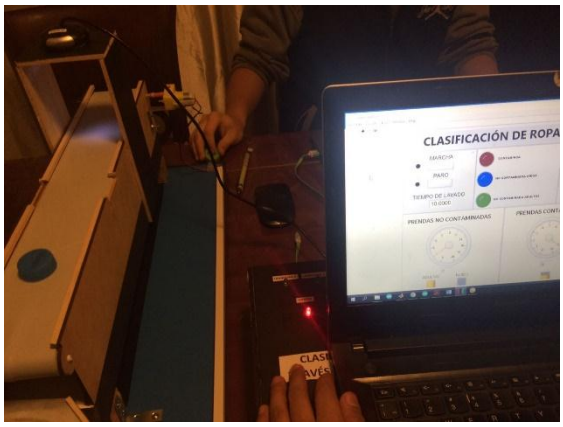



**Figura 3.23** Datos Obtenidos al detectar

Fuente: Elaborada por el autor

### 3.4.6 Visualización de Datos

Una vez que se ejecuta el proceso se obtiene datos de cada uno de los sensores infrarrojos y los datos del proceso de visión artificial. La tarjeta Arduino interpreta los datos que obtiene y los estados del proceso son visualizados a través de leds. En el panel de visualización cuenta con luces de aviso que indican el estado de las lavadoras, el color detectado, si el proceso está detenido o ejecutándose. En las Figuras 3.24, 3.25 y 3.26 se observa la detección del color azul, verde y rojo respectivamente, mostrado en el panel de visualización

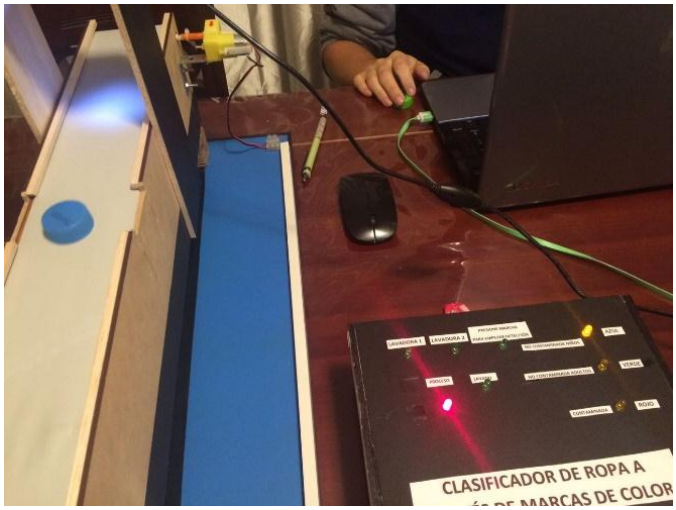



**Figura 3.24** Detección del color azul mostrado en el panel de visualización.

Fuente: Elaborada por el autor


**Figura 3.25** Detección del color verde mostrado en el panel de visualización

Fuente: Elaborada por el autor

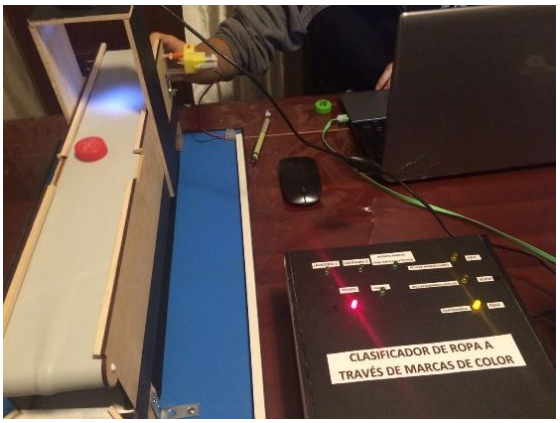



**Figura 3.26** Detección del color rojo mostrado en el panel de visualización

Fuente: Elaborada por el autor

## 3.5 ANÁLISIS DE RESULTADOS

### 3.5.1 Determinación de la repetibilidad de la planta para clasificar ropa

Para determinar el desempeño de la planta para clasificar ropa a través de marcas de color, se muestra la repetibilidad para determinar el error. Esta prueba consiste en que cada una de las vestimentas sea detectada correctamente las veces que se requiera. Este valor representa un grado de exactitud durante la repetición de la detección. La desviación estándar muestra la dispersión de las muestras en cada experimento:

: Desviación estándar

: Valor del conjunto de datos usados para la obtención de su desviación.

: Valor promedio entre todos los datos

: Número de datos por punto muestreado

**Tabla 3.1 Pruebas de repetibilidad No Contaminada Niños**

| **Pruebas de repetitivilidad Contaminada Niños** | | | | |
|---|---|---|---|---|
| Número de pruebas | Número de experimentos | Promedio de ensayos realizados | Límite Máximo de ensayos realizados | Límite Mínimo de ensayos realizados |
| 1 | 9 | 9,5 | 10,17 | 8,83 |
| 2 | 10 | 9,5 | 10,17 | 8,83 |
| 3 | 10 | 9,5 | 10,17 | 8,83 |
| 4 | 8 | 9,5 | 10,17 | 8,83 |
| 5 | 10 | 9,5 | 10,17 | 8,83 |
| 6 | 8 | 9,5 | 10,17 | 8,83 |
| 7 | 9 | 9,5 | 10,17 | 8,83 |
| 8 | 9 | 9,5 | 10,17 | 8,83 |
| 9 | 10 | 9,5 | 10,17 | 8,83 |
| 10 | 10 | 9,5 | 10,17 | 8,83 |
| 11 | 9 | 9,5 | 10,17 | 8,83 |
| 12 | 10 | 9,5 | 10,17 | 8,83 |
| 13 | 9 | 9,5 | 10,17 | 8,83 |
| 14 | 10 | 9,5 | 10,17 | 8,83 |
| 15 | 10 | 9,5 | 10,17 | 8,83 |
| 16 | 10 | 9,5 | 10,17 | 8,83 |
| 17 | 9 | 9,5 | 10,17 | 8,83 |
| 18 | 10 | 9,5 | 10,17 | 8,83 |
| 19 | 10 | 9,5 | 10,17 | 8,83 |
| 20 | 10 | 9,5 | 10,17 | 8,83 |
| **Promedio** | | 9,50 | | |
| **Desviación estándar** | | 0,67 | | |
| **Límite Máximo** | | 10,17 | | |
| **Límite Mínimo** | | 8,83 | | |

Fuente: Elaborada por el autor

**Tabla 3.2 Pruebas de repetibilidad No Contaminada Adultos**

| **Pruebas de repetitivilidad No Contaminada Adultos** | | | | |
|---|---|---|---|---|
| Número de pruebas | Número de experimentos | Promedio de ensayos realizados | Límite Máximo de ensayos realizados | Límite Mínimo de ensayos realizados |
| 1 | 10 | 9,45 | 10,04 | 8,86 |
| 2 | 10 | 9,45 | 10,04 | 8,86 |
| 3 | 9 | 9,45 | 10,04 | 8,86 |
| 4 | 10 | 9,45 | 10,04 | 8,86 |
| 5 | 9 | 9,45 | 10,04 | 8,86 |
| 6 | 9 | 9,45 | 10,04 | 8,86 |
| 7 | 10 | 9,45 | 10,04 | 8,86 |
| 8 | 10 | 9,45 | 10,04 | 8,86 |
| 9 | 9 | 9,45 | 10,04 | 8,86 |
| 10 | 10 | 9,45 | 10,04 | 8,86 |
| 11 | 9 | 9,45 | 10,04 | 8,86 |
| 12 | 8 | 9,45 | 10,04 | 8,86 |
| 13 | 9 | 9,45 | 10,04 | 8,86 |
| 14 | 9 | 9,45 | 10,04 | 8,86 |
| 15 | 10 | 9,45 | 10,04 | 8,86 |
| 16 | 10 | 9,45 | 10,04 | 8,86 |
| 17 | 10 | 9,45 | 10,04 | 8,86 |
| 18 | 10 | 9,45 | 10,04 | 8,86 |
| 19 | 9 | 9,45 | 10,04 | 8,86 |
| 20 | 9 | 9,45 | 10,04 | 8,86 |
| **Promedio** | | 9,45 | | |
| **Desviación estándar** | | 0,59 | | |
| **Límite Máximo** | | 10,04 | | |
| **Límite Mínimo** | | 8,86 | | |

Fuente: Elaborada por el autor

**Tabla 3.3 Pruebas de repetibilidad Contaminada**

| **Pruebas de repetibilidad Contaminada** | | | | |
|---|---|---|---|---|
| Número de pruebas | Número de Experimentos | Promedio de ensayos realizados | Límite Máximo de ensayos realizados | Límite Mínimo de ensayos realizados |
| 1 | 10 | 9,6 | 10,18 | 9,02 |
| 2 | 10 | 9,6 | 10,18 | 9,02 |
| 3 | 10 | 9,6 | 10,18 | 9,02 |
| 4 | 10 | 9,6 | 10,18 | 9,02 |

| | | | | |
|---|---|---|---|---|
| 5 | 9 | 9,6 | 10,18 | 9,02 |
| 6 | 9 | 9,6 | 10,18 | 9,02 |
| 7 | 8 | 9,6 | 10,18 | 9,02 |
| 8 | 10 | 9,6 | 10,18 | 9,02 |
| 9 | 10 | 9,6 | 10,18 | 9,02 |
| 10 | 10 | 9,6 | 10,18 | 9,02 |
| 11 | 10 | 9,6 | 10,18 | 9,02 |
| 12 | 10 | 9,6 | 10,18 | 9,02 |
| 13 | 9 | 9,6 | 10,18 | 9,02 |
| 14 | 9 | 9,6 | 10,18 | 9,02 |
| 15 | 10 | 9,6 | 10,18 | 9,02 |
| 16 | 10 | 9,6 | 10,18 | 9,02 |
| 17 | 10 | 9,6 | 10,18 | 9,02 |
| 18 | 10 | 9,6 | 10,18 | 9,02 |
| 19 | 9 | 9,6 | 10,18 | 9,02 |
| 20 | 9 | 9,6 | 10,18 | 9,02 |
| **Promedio** | | 9,60 | | |
| **Desviación estándar** | | 0,58 | | |
| **Límite Máximo** | | 10,18 | | |
| **Límite Mínimo** | | 9,02 | | |

Fuente: Elaborada por el autor

### 3.5.2 Cálculo de errores en las pruebas

Las pruebas ejecutadas fueron realizadas en una sala cerrada iluminada con luz artificial. El análisis de cada prueba fue realizado bajo las mismas condiciones para realizar un estudio comparativo del sistema propuesto a través del cálculo del error relativo, el cual es el cociente entre la diferencia del valor de detecciones experimental y el valor real, que en este caso es el valor de referencia que brinda la capacidad de las lavadoras.

El Error Relativo Porcentual es el siguiente:

$$e_{relativo} = \left(\frac{V_{sistema} - V_{real}}{V_{real}}\right) x\ 100\ \%$$

**Tabla 3.4. Cálculo de errores en las pruebas de repetitividad No Contaminada Niños**

| **Pruebas de repetitivilidad No Contaminada Niños** | | | |
|---|---|---|---|
| Número de pruebas | Número de experimentos | Referencia de ensayos realizados | Error Relativo Porcentual |
| 1 | 9 | 10 | 10 |
| 2 | 10 | 10 | 0 |
| 3 | 10 | 10 | 0 |
| 4 | 8 | 10 | 20 |
| 5 | 10 | 10 | 0 |
| 6 | 8 | 10 | 20 |
| 7 | 9 | 10 | 10 |
| 8 | 9 | 10 | 10 |
| 9 | 10 | 10 | 0 |
| 10 | 10 | 10 | 0 |
| 11 | 9 | 10 | 10 |
| 12 | 10 | 10 | 0 |
| 13 | 9 | 10 | 10 |
| 14 | 10 | 10 | 0 |
| 15 | 10 | 10 | 0 |
| 16 | 10 | 10 | 0 |
| 17 | 9 | 10 | 10 |
| 18 | 10 | 10 | 0 |
| 19 | 10 | 10 | 0 |
| 20 | 10 | 10 | 0 |

Fuente: Elaborada por el autor

En la figura 3.27 se evidencia el error relativo porcentual en la Prueba de repetitividad de prendas de vestir denominada “No Contaminada Niños”, donde podemos observar que los errores son relativamente bajos dentro de las 20 pruebas realizadas. Existen 2 pruebas donde se genera un error del 20% y esto se debe a las variaciones de luminosidad presentadas en la sala de prueba donde se encontraba el prototipo. Por esta razón se justifica que el ambiente de trabajo donde se encuentre este sistema propuesto debe ser cerrado y con una iluminación constante, ya que la luz de ambiente indirectamente afecta a los sensores infrarrojos.

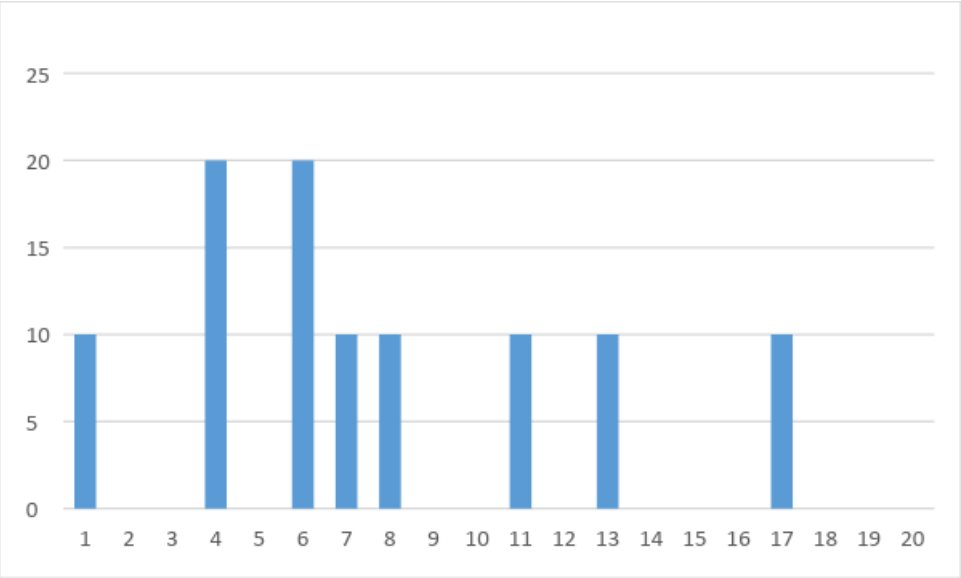



**Figura 3.27 Cálculo de errores en las pruebas de repetitividad No Contaminada Niños**

Fuente: Elaborada por el autor

**Tabla 3.5 Pruebas de repetitividad No Contaminada Adultos**

| **Pruebas de repetitividad No Contaminada Adultos** | | | |
|---|---|---|---|
| Número de pruebas | Número de experimentos | Referencia de ensayos realizados | Error Relativo Porcentual |
| 1 | 10 | 10 | 0 |
| 2 | 10 | 10 | 0 |
| 3 | 9 | 10 | 10 |
| 4 | 10 | 10 | 0 |
| 5 | 9 | 10 | 10 |
| 6 | 9 | 10 | 10 |
| 7 | 10 | 10 | 0 |
| 8 | 10 | 10 | 0 |
| 9 | 9 | 10 | 10 |
| 10 | 10 | 10 | 0 |
| 11 | 9 | 10 | 10 |
| 12 | 8 | 10 | 20 |
| 13 | 9 | 10 | 10 |
| 14 | 9 | 10 | 10 |
| 15 | 10 | 10 | 0 |
| 16 | 10 | 10 | 0 |
| 17 | 10 | 10 | 0 |
| 18 | 10 | 10 | 0 |
| 19 | 9 | 10 | 10 |
| 20 | 9 | 10 | 10 |

Fuente: Elaborada por el autor

En la figura 3.28 se muestra el error relativo porcentual en la Prueba de repetitividad de prendas de vestir denominada “No Contaminada Adultos”, donde podemos observar que los errores son relativamente bajos dentro de las 20 pruebas realizadas. Existen 2 pruebas donde se genera un error del 20% y esto se debe a las variaciones de luminosidad presentadas en la sala de prueba donde se encontraba el prototipo. Por esta razón se justifica que el ambiente de trabajo donde se encuentre este sistema propuesto debe ser cerrado y con una iluminación constante, ya que la luz de ambiente indirectamente afecta a los sensores infrarrojos.

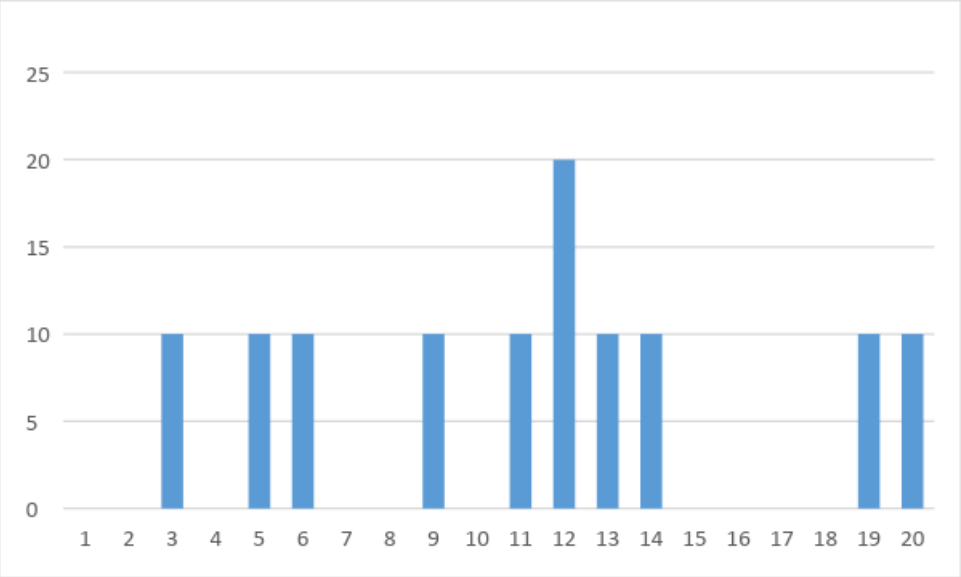



**Figura 3.28 Cálculo de errores en las pruebas de repetitividad No Contaminada Niños**

Fuente: Elaborada por el autor

**Tabla 3.6 Pruebas de repetibilidad Contaminada**

| **Pruebas de repetibilidad Contaminada** | | | |
|---|---|---|---|
| Número de pruebas | Número de experimentos | Referencia de ensayos realizados | Error Relativo Porcentual |
| 1 | 10 | 10 | 0 |
| 2 | 10 | 10 | 0 |
| 3 | 10 | 10 | 0 |
| 4 | 10 | 10 | 0 |
| 5 | 9 | 10 | 10 |
| 6 | 9 | 10 | 10 |
| 7 | 8 | 10 | 20 |
| 8 | 10 | 10 | 0 |

| | | | |
|---|---|---|---|
| 9 | 10 | 10 | 0 |
| 10 | 10 | 10 | 0 |
| 11 | 10 | 10 | 0 |
| 12 | 10 | 10 | 0 |
| 13 | 9 | 10 | 10 |
| 14 | 9 | 10 | 10 |
| 15 | 10 | 10 | 0 |
| 16 | 10 | 10 | 0 |
| 17 | 10 | 10 | 0 |
| 18 | 10 | 10 | 0 |
| 19 | 9 | 10 | 10 |
| 20 | 9 | 10 | 10 |

Fuente: Elaborada por el autor

En la figura 3.29 se indica el error relativo porcentual en la Prueba de repetitividad de prendas de vestir denominada “Contaminada”, donde podemos observar que los errores son relativamente bajos dentro de las 20 pruebas realizadas. Existen 2 pruebas donde se genera un error del 20% y esto se debe a las variaciones de luminosidad presentadas en la sala de prueba donde se encontraba el prototipo. Por esta razón se justifica que el ambiente de trabajo donde se encuentre este sistema propuesto debe ser cerrado y con una iluminación constante, ya que la luz de ambiente indirectamente afecta a los sensores infrarrojos.

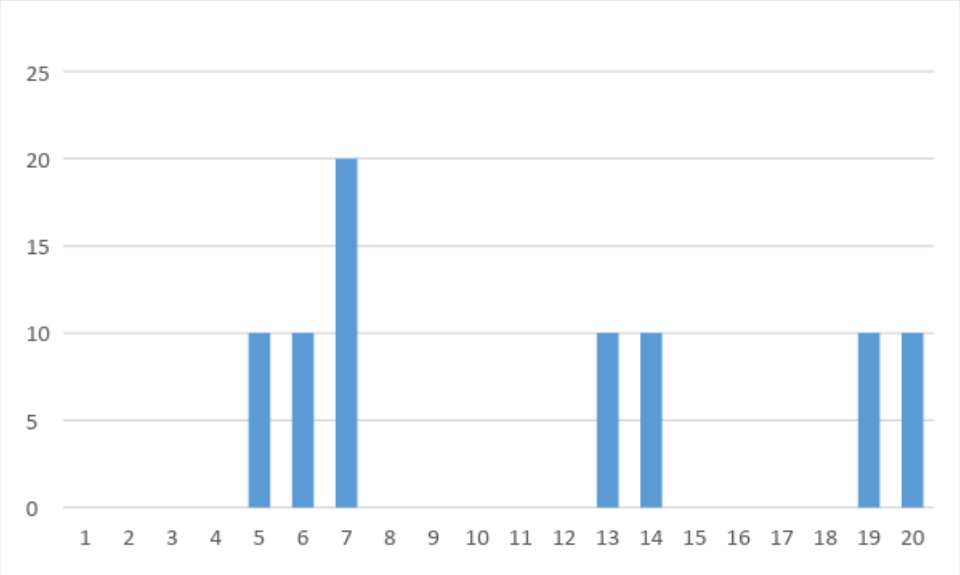



**Figura 3.29 Pruebas de repetibilidad Contaminada**

Fuente: Elaborada por el autor

# CONCLUSIONES

Se elaboró una maqueta con una banda transportadora y ductos de conducción que permiten emular el sistema de clasificación de las prendas hospitalarias para su tratamiento adecuado ya que el implementarlo implicaría un gasto excesivo para la demostración del mismo.

A través del uso de una cámara web y la programación en Matlab desarrollada para el sistema de visión artificial se logró que a través de la adquisición de imágenes se detecte colores para clasificar las prendas hospitalarias.

Se programó un sistema de control que con la ayuda de los elementos como motor, servo motor, sensores instalados y la información recolectada permitió la correcta clasificación de las prendas hospitalarias, a través del sistema de visión artificial

La interfaz gráfica permite controlar de manera remota el proceso realizado por la maqueta, la cual permite que un operador tome acciones de acuerdo a su requerimiento o de existir algún inconveniente durante el funcionamiento.

Se realizó pruebas del proceso de clasificación para comprobar el correcto funcionamiento de los programas desarrollados y de la implementación integral del proyecto.

# RECOMENDACIONES

El algoritmo desarrollado en este trabajo no es estándar porque no funciona para todos los ambientes, por esta razón, en el momento de implementarlo en otro lugar es necesario realizar una calibración del algoritmo para que se ajuste a las condiciones del ambiente al cual va ser sometido.

Si se requiere la implementación de este prototipo a una escala más grande es necesario que se realice un ajuste del algoritmo para mejorar la detección e incorporar el uso de sensores infrarrojos con mayor velocidad de respuesta con el fin de obtener un resultado con mayor confiabilidad.

Es importante considerar los tiempos entre la detección y el movimiento de la banda transportadora de tal manera que permita ajustar la tasa de salida de las prendas de clasificación con el objetivo de no generar cuellos de botella en el proceso de detección, selección y lavado de la ropa.

Debido a que el sistema fue diseñado para que sea sujeto a cambios ya que el controlador posee puertos libres, es recomendable leer el manual de usuario y los esquemas de conexión para utilizarlos de manera correcta y que no se produzcan errores a la hora de implementar un nuevo sensor o actuador.

Verificar que todos los dispositivos se encuentren conectados al computador principal antes de ejecutar el proceso y sobre todo tener en cuenta que en el caso de que exista un cambio de computador, este debe poseer altas prestaciones de memoria RAM y velocidad de procesamiento debido a que la visión artificial genera una alta carga computacional.

# REFERENCIAS BIBLIOGRÁFICAS

Abaya, W. F., Basa, J., Sy, M., Abad, A. C., & Dadios, E. P. (2014). Low cost smart security camera with night vision capability using Raspberry Pi and OpenCV. In *2014 International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM)* (pp. 1–6). https://doi.org/10.1109/HNICEM.2014.7016253

Abid K, A. M. (2017, July 3). OpenCV-Python Tutorials Documentation. Retrieved from https://media.readthedocs.org/pdf/opencv-python-tutroals/latest/opencv-python-tutroals.pdf

Bertozzi, M., Broggi, A., Cellario, M., Fascioli, A., Lombardi, P., & Porta, M. (2002). Artificial vision in road vehicles. *Proceedings of the IEEE*, *90*(7), 1258–1271. https://doi.org/10.1109/JPROC.2002.801444

Brahmbhatt, S. (2013). Embedded Computer Vision: Running OpenCV Programs on the Raspberry Pi. In *Practical OpenCV* (pp. 201–218). Apress. https://doi.org/10.1007/978-1-4302-6080-6_11

Fernandes, S. L., & Bala, J. G. (2015). Low Power Affordable and Efficient Face Detection in the Presence of Various Noises and Blurring Effects on a Single-Board Computer. In *Emerging ICT for Bridging the Future - Proceedings of the 49th Annual Convention of the Computer Society of India (CSI) Volume 1* (pp. 119–127). Springer, Cham. https://doi.org/10.1007/978-3-319-13728-5_13

Fernández, G., & Javier, F. (2009). Reconocimiento de objetos en una cocina con una webcam. Retrieved from https://e-archivo.uc3m.es/handle/10016/10007

González Benítez, U. (2010). Detección de personas a partir de visión artificial. Retrieved from https://e-archivo.uc3m.es/handle/10016/10115

Jácome, C., Teresa, M., Andrade, M., & Javier, D. (2014). Control por visión de un cuadricóptero utilizando ROS. Retrieved from http://bibdigital.epn.edu.ec/handle/15000/8723

Koivo, A. J., & Houshangi, N. (1991). Real-time vision feedback for servoing robotic manipulator with self-tuning controller. *IEEE Transactions on Systems, Man, and Cybernetics*, *21*(1), 134–142. https://doi.org/10.1109/21.101144

Lueiza Valenzuela, J. M., Bro, P. B. (Prof G., Meza Montoya, F. (Prof I., & Reyes, F. (Prof I. (2007). *Comparacion entre contornos activos y transformada de Hough, para la deteccion de circlos en imagenes digitales* (Thesis). Universidad de Talca (Chile). Escuela de Ingenieria Civil en Computacion. Retrieved from http://dspace.utalca.cl/handle/1950/7980

Mukundan, R., & Ramakrishnan, K. R. (1998). *Moment Functions in Image Analysis: Theory and Applications*. World Scientific.

Flores, M., Medina, L. , & Mayorquin J., Revista_de_Prototipos_Tecnologicos_V3_N7_1.pdf. (n.d.). Retrieved from http://www.ecorfan.org/spain/researchjournals/Prototipos_Tecnologicos/vol3num7/Revista_de_Prototipos_Tecnologicos_V3_N7_1.pdf

Sabas. (2011, June 26). Reconocimiento de los colores básicos RGB por medio de WebCam con MatLab. Retrieved from http://tecnologicobj12.blogspot.com/2011/06/reconocimiento-de-los-colores-basicos.html

Sensor de infrarrojos | Wiki de Robótica. (n.d.). Retrieved August 14, 2017, from http://wiki.robotica.webs.upv.es/wiki-de-robotica/sensores/sensores-proximidad/sensor-infrarrojos/

TUTORIAL DE VISIÓN ARTIFICIAL USANDO MATLAB ORIENTADO A ZIGBEE. (n.d.). Retrieved August 14, 2017, from http://plataformaszigbee.blogspot.com/2012/08/tutorial-de-vision-artificial-usando.html

Yolanda González Cid. (n.d.). Aplicaciones de la Ingeniería Electrónica II. Retrieved from http://dmi.uib.es/~ygonzalez/Master_10212/Espacios_color_10210.pdf

# ANEXO I: MANUAL TÉCNICO

## INTRODUCCIÓN

El presente manual va dirigido al administrador del sistema, otros desarrolladores o puede ser usado para auditorias de calidad. Constituye una guía para la realización de las operaciones de mantenimiento del sistema indicando las especificaciones técnicas más importantes del sistema.

## OBJETIVO GENERAL DEL PROYECTO

Implementar un prototipo para clasificación de ropa hospitalaria a través de una banda transportadora y algoritmos de visión artificial desarrollados en el lenguaje de programación Matlab y la plataforma Arduino junto con una interfaz remota.

## OBJETIVO ESPECÍFICOS DEL PROYECTO

Crear una maqueta con banda transportadora y ductos de conducción que permitan emular el proceso automatizado que seguirían las prendas hospitalarias dentro de su respectivo lavado.

Desarrollar un sistema de visión artificial con adquisición de imágenes mediante cámara web y programación en el lenguaje Matlab, que permita la detección de etiquetas de colores sobre prendas hospitalarias,

Programar un sistema de control que permita tomar las acciones pertinentes a los elementos correspondientes como motores o servo motores para la clasificación de las prendas hospitalarias, tomando la información tanto de los sensores instalados como del sistema de visión artificial para su correcto funcionamiento.

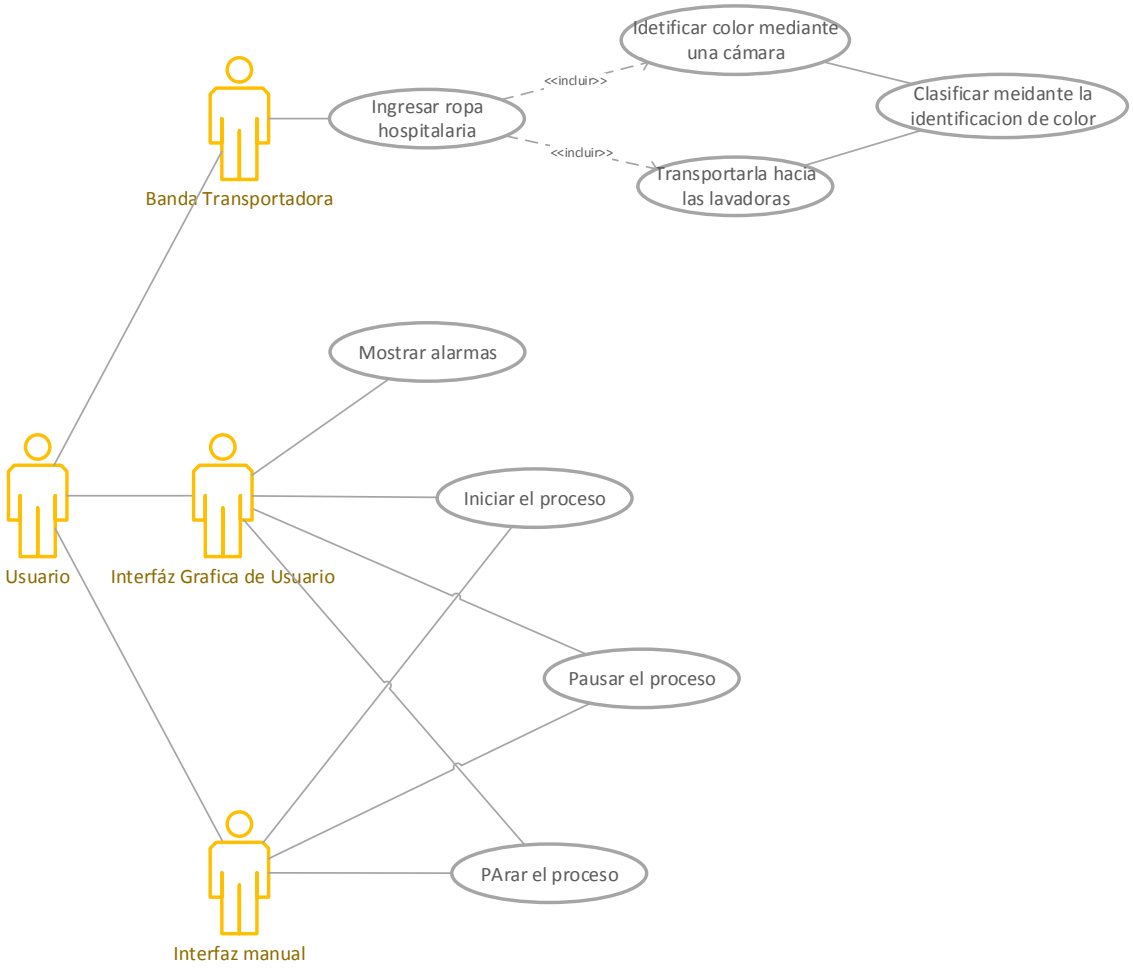
Implementar una interfaz gráfica que permita de manera remota controlar el proceso realizado por la maqueta, para que un operador pueda tomar las medidas necesarias si existe inconveniente dentro del proceso automatizado de clasificación.

Presentar las pruebas y resultados tanto del proceso de clasificación como del proceso de detección del color mediante la visión artificial, para la comprobación del correcto funcionamiento de los algoritmos y programas implementados dentro de este proyecto.

## CONTENIDO

### Requerimientos

Los requerimientos del sistema serán descritos a través del siguiente diagrama de casos de uso para el sistema.




En el diagrama de casos de uso se pudo identificar los actores del sistema los cuales se detalla a continuación:

Banda transportadora

Sera la encargada de iniciar el proceso de transporte, detección y clasificación de la ropa usando el algoritmo de detección de color. A continuación, se detalla los elementos utilizados para realizar dichas funciones.

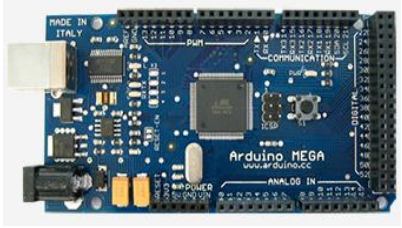
Motor DC.

Sensores infrarrojos.

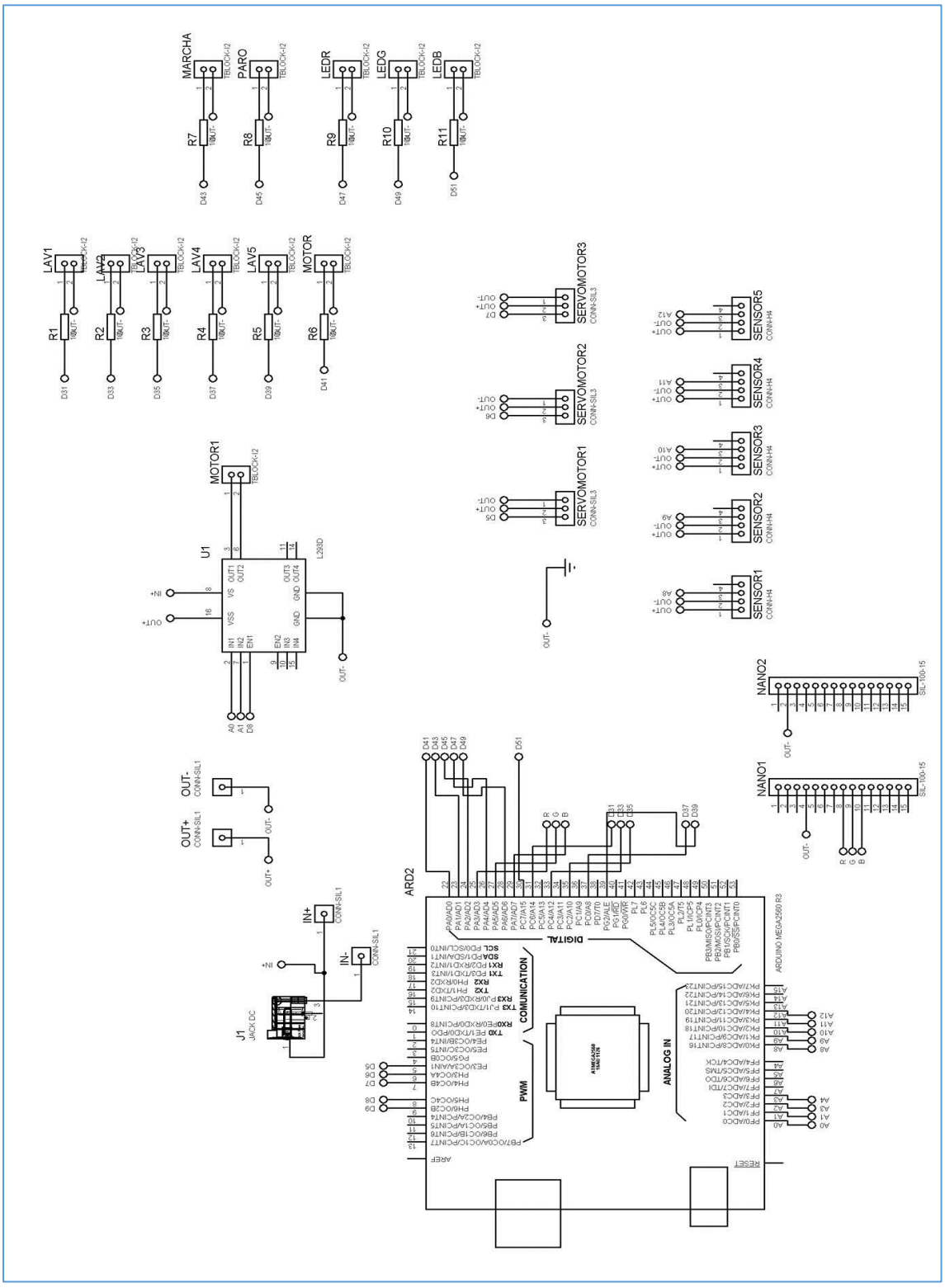
Cámara web.


Servo motor.

Arduino

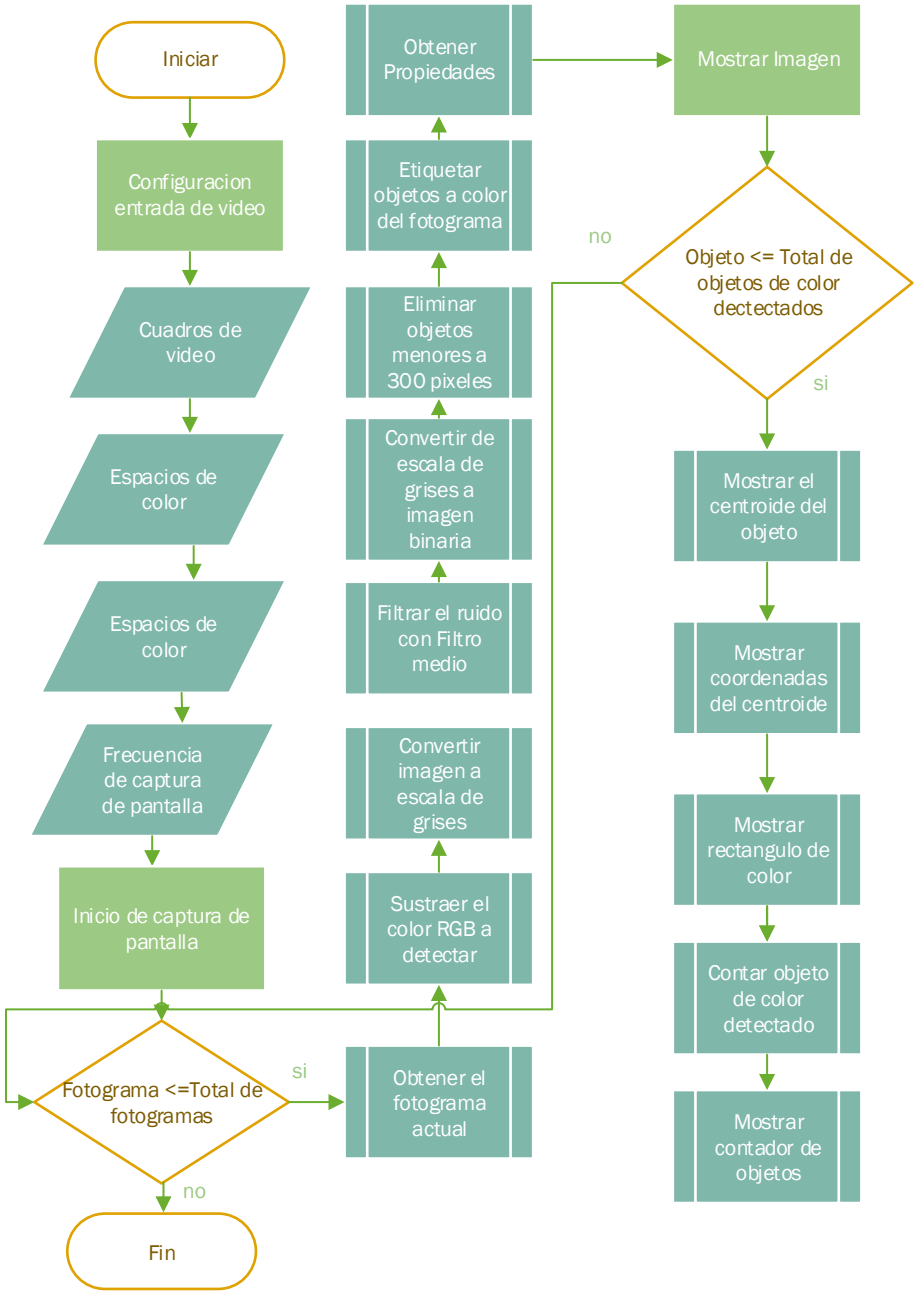


La conexión de los elementos indicados anteriormente se muestra la siguiente figura.



**ALGORITMO DE VISIÓN ARTIFICIAL.**

El proceso de visión artificial realizado en este proyecto basa su funcionamiento en el modelo general de Marr-Palmer como se muestra en el siguiente diagrama.




Este modelo indica que existe una fase de captura o fase sensorial, el reproceso de la imagen, la segmentación y el reconocimiento.

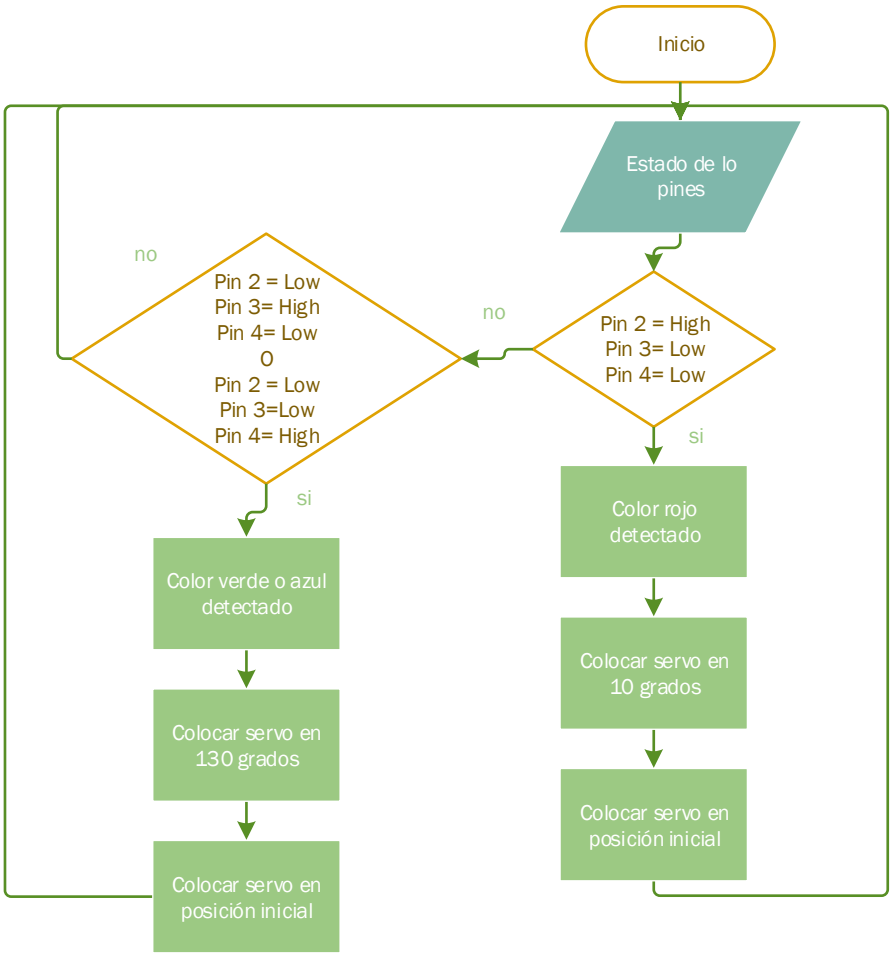
La fase sensorial la realizará la cámara web, enviando imágenes hacia el Matlab para que dentro del código sea procesada, las imágenes serán enviadas en tiempo real, eso quiere

decir que una vez activada la cámara enviara las imágenes necesarias para que el código realice la detección.

**Programación de Arduino**

La programación de la primera fase es decir Arduino se la realizo mientras el sistema de visión artificial estaba en funcionamiento.

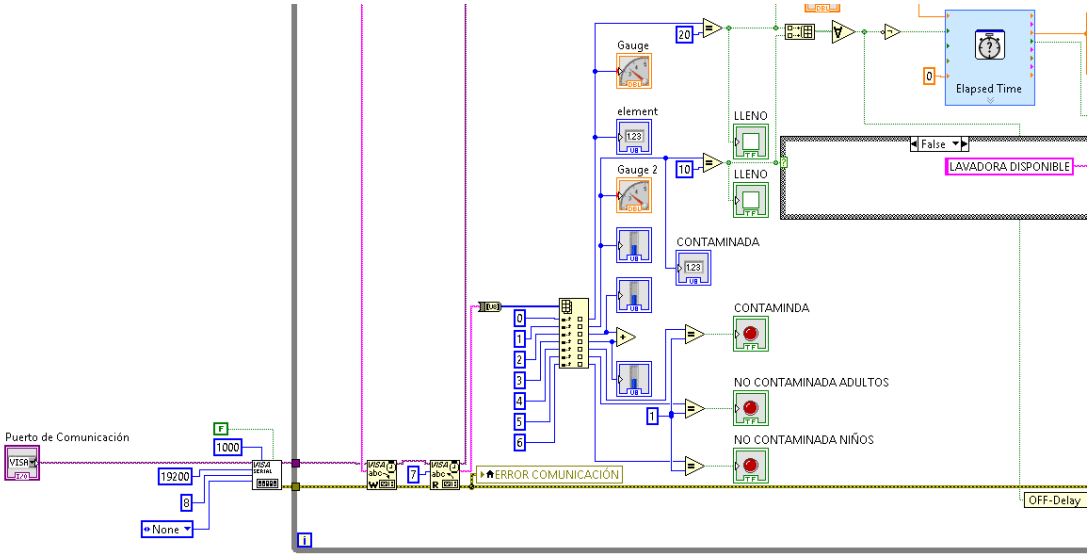
Para la fase dos es decir el accionar del servo motor se sigue el siguiente diagrama mostrado en la figura.



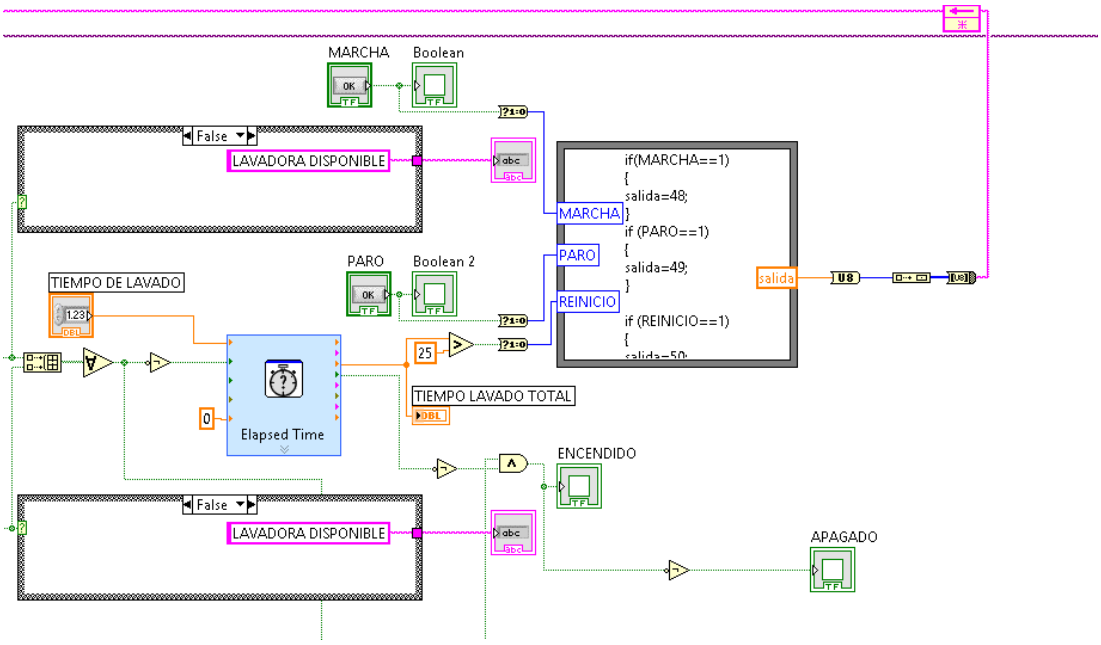

**INTERFAZ GRÁFICA**

Para la realización de la interface gráfica de usuario se utilizó el programa LabVIEW ya que tiene la versatilidad de crear pantallas que son muy amigables con el usuario, permitiendo que sea fácil de entender para cualquier persona que haga uso del sistema

implementado. Para la programación de la interface se empezó estableciendo la comunicación entre dispositivos, definiendo parámetros como la velocidad de la comunicación, el tamaño del dato, y el tipo de paridad. En este proceso de configuración se debe definir cuantos datos se van a recibir para poder separar el arreglo de datos y enviarlos a cada uno de los indicadores. En la interface se muestra el estado de cada una de las partes del proceso que se está desarrollando, además se creó un pequeño subprograma que realiza un tiempo para el proceso de lavado una vez que se hayan llenado las lavadoras. De la misma manera el programa envía comandos a la tarjeta Arduino para detener o activar el proceso y comandos una vez que termina con el proceso de lavado para reiniciar el programa.

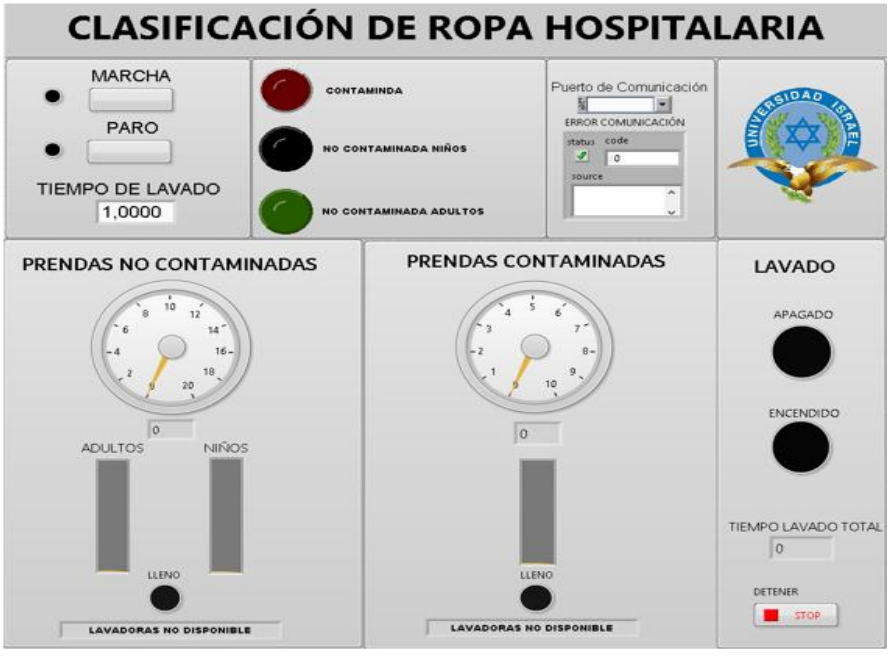



Programación LabVIEW parte 1




Programación LabVIEW parte 2

Para la realización del HMI a través de LabVIEW, se trabaja en el panel frontal donde se encuentra diferentes visualizadores permitiendo que la interface sea atractiva para el usuario y sobre todo que pueda entender lo que está pasando durante el desarrollo de la clasificación. La interface consta de botones de marcha y paro para controlar el proceso, luces que indican que tipo de prenda de vestir y el inicio de lavado.

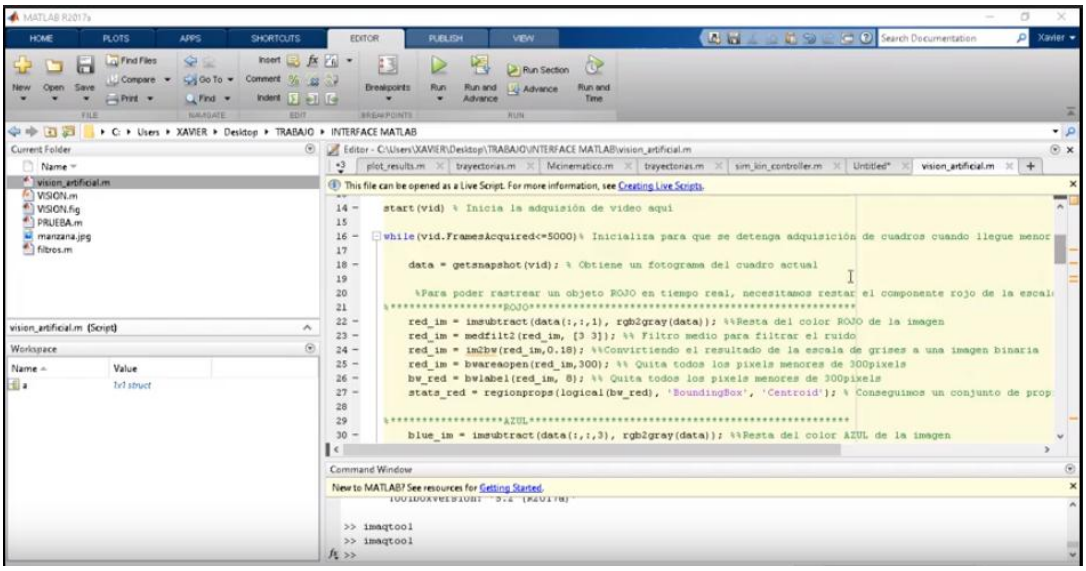


# ANEXO II: MANUAL DE USUARIO

## INTRODUCCIÓN

El presente manual va dirigido a los usuarios finales del sistema para el correcto uso del sistema. Se detallará paso a paso como

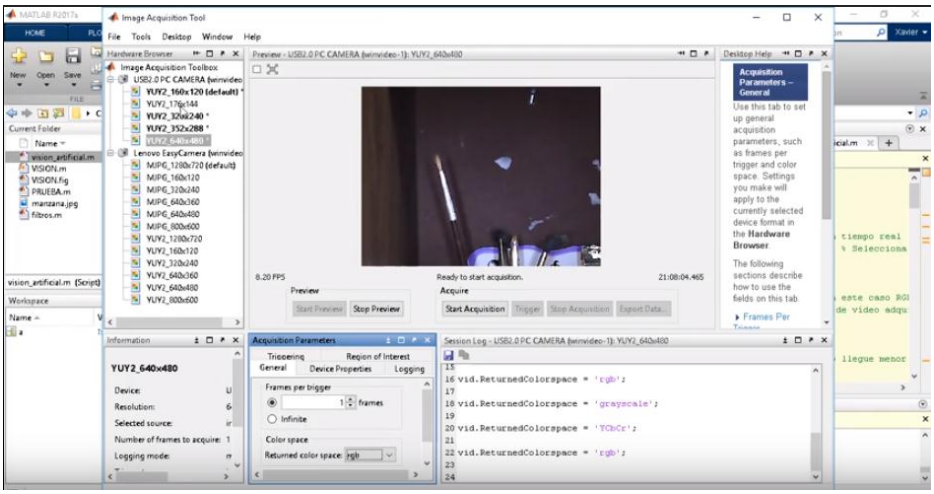
1. El primer paso en Matlab para manejar una cámara web es la detección de objetos en tiempo real



Para lo cual es necesario asignar un formato a la resolución

```
%vid = videoinput('winvideo', 1, 'YUY2_320x240');% Captura los cuadros del video que corre en tiempo real
vid = videoinput('winvideo', 1, 'YUY2_160x120');                                    % Selecciona
```

Toda esta información se obtiene a través de comando Image Tool



Es importante inicializar correctamente las propiedades el objeto

```
% Inicialización de las propiedades del objeto
set(vid, 'FramesPerTrigger', Inf); %%El objeto se mantiene adquiriendo cuadros
set(vid, 'ReturnedColorspace', 'rgb') %%Especifica el espacio del color que quiero revisar en este caso RGB
vid.FrameGrabInterval = 2; %%Propiedad que especifica la frecuencia con el objeto de entrada de video adqu
```

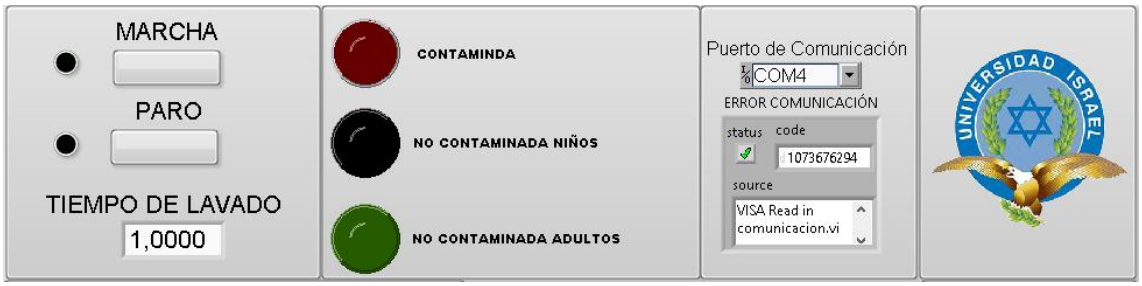
Rastreo del color rojo

```
%Para poder rastrear un objeto ROJO en tiempo real, necesitamos restar el componente rojo de la escal
%*****************ROJO*****************************************************
red_im = imsubtract(data(:,:,1), rgb2gray(data)); %%Resta del color ROJO de la imagen
red_im = medfilt2(red_im, [3 3]); %% Filtro medio para filtrar el ruido
red_im = im2bw(red_im,0.18); %%Convirtiendo el resultado de la escala de grises a una imagen binaria
red_im = bwareaopen(red_im,300); %% Quita todos los pixels menores de 300pixels
bw_red = bwlabel(red_im, 8); %% Marcar todos los componentes en la imagen
stats_red = regionprops(logical(bw_red), 'BoundingBox', 'Centroid'); % Conseguimos un conjunto de prop
```

Rastreo del color azul

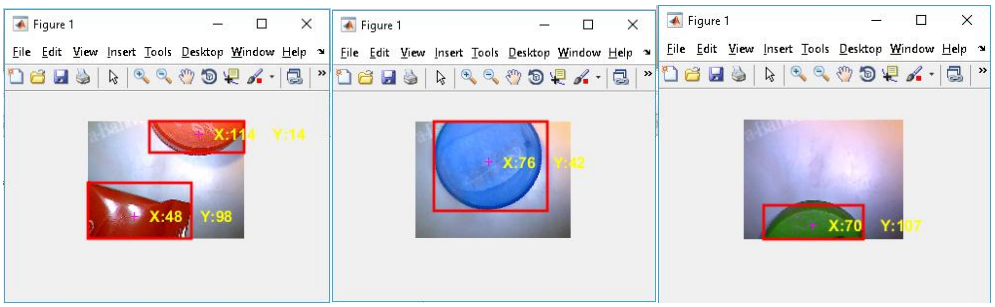
```
%*****************AZUL*****************************************************
blue_im = imsubtract(data(:,:,3), rgb2gray(data)); %%Resta del color AZUL de la imagen
blue_im = medfilt2(blue_im, [3 3]); %% Filtro medio para filtrar el ruido
blue_im = im2bw(blue_im,0.18); %%Convirtiendo el resultado de la escala de grises a una imagen binaria
blue_im = bwareaopen(blue_im,300); %% Quita todos los pixels menores de 300pixels
bw_blue = bwlabel(blue_im, 8); %% Marcar todos los componentes en la imagen
stats_blue = regionprops(logical(bw_blue), 'BoundingBox', 'Centroid'); %% Conseguimos un conjunto de p
```

2. Se debe configurar el puerto y estado de comunicación

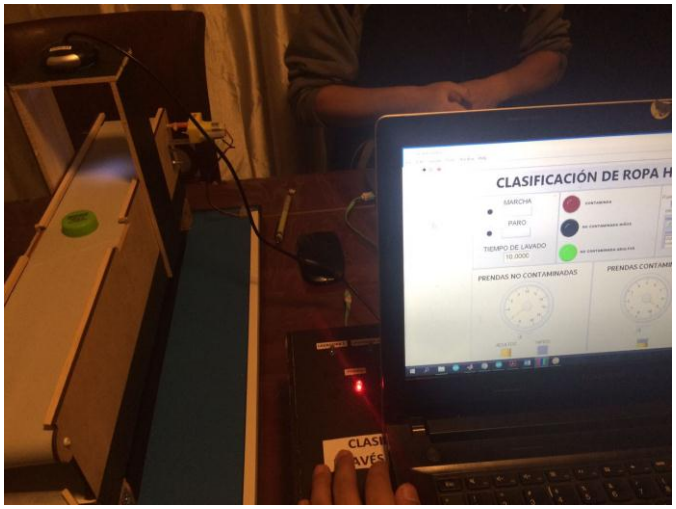


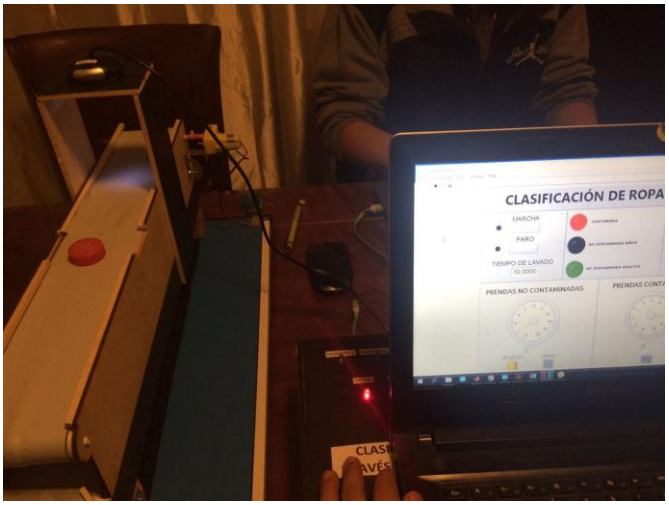

3. Seguimiento de los colores

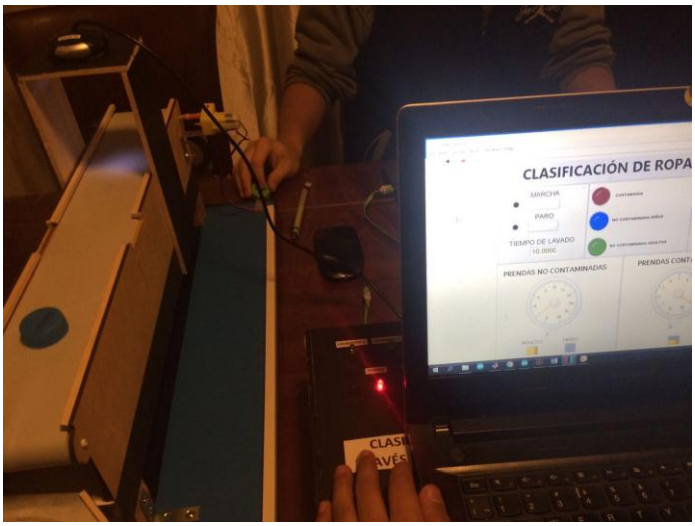
Para el seguimiento de colores se analiza en cada uno de los *frames* que toma durante la ejecución del programa, en cada una de las pasadas del lazo captura una imagen y realiza el procedimiento mostrado anteriormente filtrado, segmentación. Cuando el programa se ejecuta captura una imagen del video en tiempo real, el color es sustraído dependiendo el color que se estableció para la detección. La imagen captura es convertida a escala de grises. Se descarta objetos en la imagen menores a 300 pixeles utilizando el método *bwareopen*. Un ejemplo del seguimiento de color se visualiza en las siguientes figuras




En las siguientes figuras se muestra el reconocimiento en el prototipo de clasificación cómo va clasificando según el color.




CLASIFICACIÓN DE ROPA
MARCHA
PARO
TIEMPO DE LAVADO
PRENDAS NO CONTAMINADAS

# ANEXO III: CÓDIGO DE MATLAB

```
1. delete(instrfind({'Port'},{'COM4'}))

2. a = imaqhwinfo; % Formato de la cámara previamente seleccionado
3. b= arduino('COM4');

4. %vid = videoinput('winvideo', 1, 'YUY2_320x240');% Captura los cuadros del video
que corre en tiempo real
5. vid = videoinput('winvideo', 1, 'YUY2_160x120');

% Selecciona también la resolución de la cámara

% Inicializacióm de las propiedades del objeto
6.    set(vid, 'FramesPerTrigger', Inf); %%El objeto se mantiene adquiriendo cuadros
7.    set(vid, 'ReturnedColorspace', 'rgb') %%Especifica el espacio del color que quiero
revisar en este caso RGB
8.    vid.FrameGrabInterval = 2; %%Propiedad que especifica la frecuencia con el objeto
de entrada de vídeo adquiere un fotograma de la secuencia de vídeo

9.   start(vid) % Inicia la adquisión de video aquí

10.    while(vid.FramesAcquired<=7000)% Inicializa para que se detenga adquisición de
cuadros cuando llegue menor o igual a 200000

11.    data = getsnapshot(vid); % Obtiene un fotograma del cuadro actual

    %Para poder rastrear un objeto ROJO en tiempo real, se requiere restar el componente
rojo de la escala de grises de la imagen
%******************ROJO**************************************************
****
12.   red_im = imsubtract(data(:,:,1), rgb2gray(data)); %%Resta del color ROJO de la
imagen
13.   red_im = medfilt2(red_im, [3 3]); %% Filtro medio para filtrar el ruido
14.   red_im = im2bw(red_im,0.18); %%Convirtiendo el resultado de la escala de grises a
una imagen binaria
15.   red_im = bwareaopen(red_im,300); %% Quita todos los pixels menores de 300pixels
16.   bw_red = bwlabel(red_im, 8); %% Quita todos los pixels menores de 300pixels
17.   stats_red = regionprops(logical(bw_red), 'BoundingBox', 'Centroid'); % Conseguimos
un conjunto de propiedades para cada región marcada

%******************AZUL**************************************************
***
18.    blue_im = imsubtract(data(:,:,3), rgb2gray(data)); %%Resta del color AZUL de la
imagen
19.    blue_im = medfilt2(blue_im, [3 3]); %% Filtro medio para filtrar el ruido
20.    blue_im = im2bw(blue_im,0.18); %%Convirtiendo el resultado de la escala de grises
a una imagen binaria
```

```
21.    blue_im = bwareaopen(blue_im,300); %% Quita todos los pixels menores de
300pixels
22.    bw_blue = bwlabel(blue_im, 8); %% Marcar todos los componentes en la imagen
23.    stats_blue = regionprops(logical(bw_blue), 'BoundingBox', 'Centroid'); %%
Conseguimos un conjunto de propiedades para cada región marcada

%******************VERDE*********************************************
****
24.    green_im = imsubtract(data(:,:,2), rgb2gray(data)); %%Resta del color VERDE de la
imagen
25.    green_im = medfilt2(green_im, [3 3]); %% Filtro medio para filtrar el ruido
26.    green_im = im2bw(green_im,0.05); %%Convirtiendo el resultado de la escala de
grises a una imagen binaria
27.    green_im = bwareaopen(green_im,300);  %% Quita todos los pixels menores de
300pixels
28.    bw_green = bwlabel(green_im, 8); %% Marcar todos los componentes en la imagen
29.    stats_green = regionprops(bw_green, 'BoundingBox', 'Centroid'); %% Conseguimos
un conjunto de propiedades para cada región marcada


30. imshow(data)%% Mostrar imagen

31. hold on %% Mantener encendido

   % Este bucle encierra en rectangulo ROJO del objeto
32. for object_red = 1:length(stats_red) %%Si el objeto a asignar es ROJO
33. bb_r = stats_red(object_red).BoundingBox; %%Una vez asignado ROJO caja con
bordes
34. bc_r = stats_red(object_red).Centroid; %%Ubicar el centroide de del rectángulo
35. rectangle('Position',bb_r,'EdgeColor','r','LineWidth',2) %%Selección del color y grosor
de líneas del rectángulo
36. plot(bc_r(1),bc_r(2), '-m+') %%Grafico de los referenciado previamente

      %%Arduino Rojo
37.       b.pinMode(7,'output'); %Configura el pin 07 como salida
38.       b.pinMode(13,'output'); %Configura el pin 07 como salida
39.       b.digitalWrite(13,1); %El pin 07 se pone en alto
40.       b.digitalWrite(7,1); %El pin 07 se pone en alto
41.       pause(1); % 1 segundo de espera
42.       b.digitalWrite(7,0); %El pin 07 se pone en
43.       b.digitalWrite(13,0);
44.       pause(1); % 1 segundo de espera

45.    end %% Fin de bucle de rectángulo ROJO del objeto

46.    hold on %Mantener encendido


   % Este bucle encierra en rectángulo VERDE del objeto
47.    for object_green= 1:length(stats_green) %%Si el objeto a asignar es VERDE
```

```
48.         bb_g = stats_green(object_green).BoundingBox; %%Una vez asignado ROJO
caja con bordes
49.         bc_g = stats_green(object_green).Centroid; %%Ubicar el centroide de del
rectángulo
50.         rectangle('Position',bb_g,'EdgeColor','g','LineWidth',2) %%Selección del color y
grosor de líneas del rectángulo
51.         plot(bc_g(1),bc_g(2), '-m+')  %%Grafico de los referenciado previamente

       %% Ardiuno Verde
52.         b.pinMode(6,'output'); %Configura el pin 06 como salida
53.         b.digitalWrite(6,1); %El pin 06 se pone en alto
54.         b.pinMode(13,'output'); %Configura el pin 07 como salida
55.         b.digitalWrite(13,1); %El pin 07 se pone en alto
56.         pause(1); % 1 segundo de espera
57.         b.digitalWrite(6,0); %El pin 06 se pone en bajo
58.         b.digitalWrite(13,0); %El pin 07 se pone en alto
59.         pause(1); % 1 segundo de espera

60.     end %% Fin de bucle de rectángulo VERDE del objeto

61.     hold on %Mantener encendido


    % Este bucle encierra en rectángulo AZUL del objeto
62.     for object_blue = 1:length(stats_blue) %%Si el objeto a asignar es AZUL
63.         bb_b = stats_blue(object_blue).BoundingBox; %%Una vez asignado ROJO caja
con bordes
64.         bc_b = stats_blue(object_blue).Centroid;  %%Ubicar el centroide de del
rectángulo
65.         rectangle('Position',bb_b,'EdgeColor','b','LineWidth',2) %%Selección del color y
grosor de líneas del rectángulo
66.         plot(bc_b(1),bc_b(2), '-m+') %%Grafico de los referenciado previamente

       %% Arduino Azul
67.         b.pinMode(5,'output'); %Configura el pin 05 como salida
       %while(true) %Condición infinita, true nunca dejará de ser true
68.         b.digitalWrite(5,1); %El pin 05 se pone en alto
69.          b.pinMode(13,'output'); %Configura el pin 07 como salida
70.         b.digitalWrite(13,1); %El pin 07 se pone en alto
71.         pause(1); % 1 segundo de espera
72.         b.digitalWrite(5,0); %El pin 05 se pone en bajo

73.         b.digitalWrite(13,0); %El pin 07 se pone en alto
74.         pause(1); % 1 segundo de espera


75.     end %% Fin de bucle de rectangulo AZUL del objeto

76. hold off %Mantener apagado
```

77. end %% Finaliza bucles

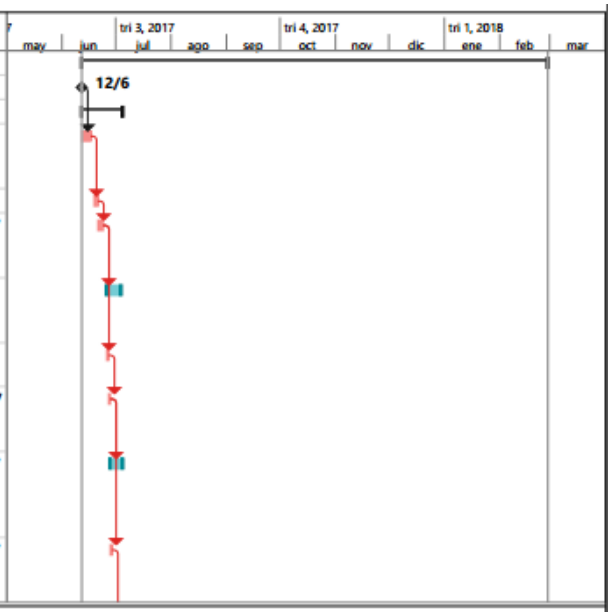
78. stop(vid); %% Parar adquisición de video

79. flushdata(vid); %% Limpiar la información de la memoria buffer

80. clear all %% Limpiar todas las variables del programa
81. sprintf('%s','FIN DEL RECONOCIMIENTO DE COLOR ')

# ANEXO IV: CRONOGRAMA

| Id | Modo de tarea | Nombre de tarea | Duración | Comienzo |
|---|---|---|---|---|
| 1 | | **PROYECTO CLASIFICADOR** | **187 días** | **lun 12/6/17** |
| 2 | | INICIO | 0 días | lun 12/6/17 |
| 3 | | **FASE 1** | **17 días** | **lun 12/6/17** |
| 4 | | Act. 1: Búsqueda de un problema que aqueje en un hospital | 5 días | lun 12/6/17 |
| 5 | | Act. 2: Visita al Hospital IESS | 2 días | lun 19/6/17 |
| 6 | | Act. 3: Investigación de los procedimientos para lavado de ropa hospitalaria | 3 días | mié 21/6/17 |
| 7 | | Act. 4: Estudio de los métodos para detección y seguimiento de objetos | 6 días | lun 26/6/17 |
| 8 | | Act. 5 Selección de algoritmo para detectar | 1 día | lun 26/6/17 |
| 9 | | Act. 6: Instalación del toolbox Imagen Adquisition en Matlab | 1 día | mar 27/6/17 |
| 10 | | Act. 7: Revisión de las funciones para realizar procesamientos de imágenes | 5 días | mié 28/6/17 |
| 11 | | Act. 8: Busqueda de sistemas embedidos para conectar con el software | 1 día | mié 28/6/17 |

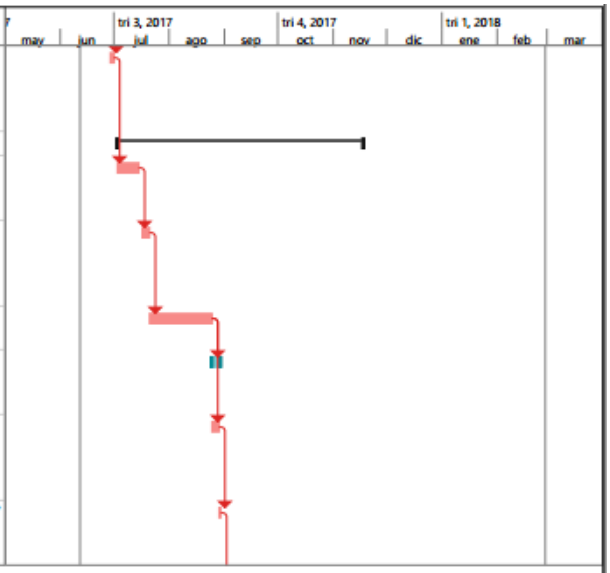



Proyecto: ProyectoClasificador
Fecha: mar 4/9/18

Tarea
División
Hito
Resumen
Resumen del proyecto
Tarea inactiva
Hito inactivo
Resumen inactivo
Tarea manual
solo duración
Informe de resumen manual
Resumen manual
solo el comienzo
solo fin
Tareas externas
Hito externo
Fecha límite
Tareas críticas
División crítica
Progreso
Progreso manual



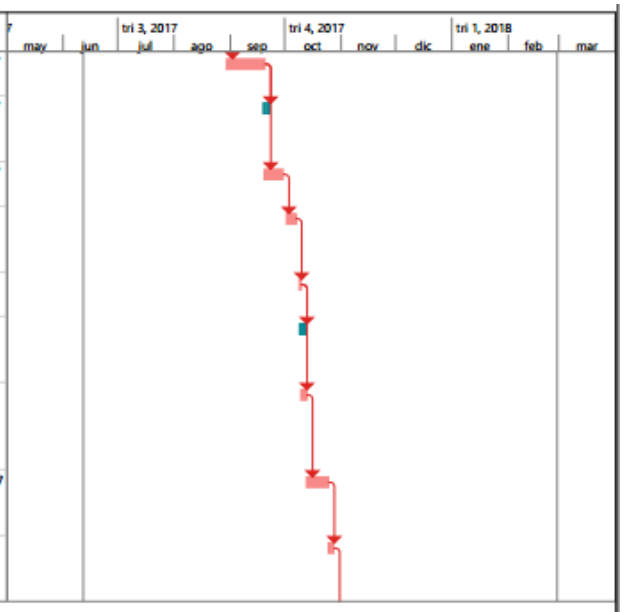
| Id | Modo de tarea | Nombre de tarea | Duración | Comienzo |
|---|---|---|---|---|
| 12 | | Act. 9: Ajustes necesarios de placa Arduino para ser controlado desde el programa Matlab | 2 días | jue 29/6/17 |
| 13 | | **FASE 2** | **100 días** | **lun 3/7/17** |
| 14 | | Act. 1: Implementación del algoritmo de Marr-Palmer en Matlab | 10 días | lun 3/7/17 |
| 15 | | Act. 2: Ajuste de parámetros para el ambiente donde se realizará la detección de colores | 4 días | lun 17/7/17 |
| 16 | | Act. 3: Diseño del prototipo (Parte mecánica) | 25 días | vie 21/7/17 |
| 17 | | Act. 4: Investigación sobre los sensores para detectar objetos | 3 días | vie 25/8/17 |
| 18 | | Act. 5: Compra de materiales para la realización del prototipo y los sensores infrarrojos | 2 días | vie 25/8/17 |
| 19 | | Act. 6: Servicio de corte laser para elaborar la banda transportadora | 1 día | mar 29/8/17 |




Proyecto: ProyectoClasificador
Fecha: mar 4/9/18

Tarea
División
Hito
Resumen
Resumen del proyecto
Tarea inactiva
Hito inactivo
Resumen inactivo
Tarea manual
solo duración
Informe de resumen manual
Resumen manual
solo el comienzo
solo fin
Tareas externas
Hito externo
Fecha límite
Tareas críticas
División crítica
Progreso
Progreso manual

| Id | Modo de tarea | Nombre de tarea | Duración | Comienzo |
|---|---|---|---|---|
| 20 | | Act. 7: Construcción de la banda transportadora | 15 días | mié 30/8/17 |
| 21 | | Act 8: Pruebas de la banda transportadora con los objetos a movilizar | 2 días | mié 20/9/17 |
| 22 | | Act. 9: Construcción de las resbaladeras | 8 días | mié 20/9/17 |
| 23 | | Act. 10: Implementación del servo motor para la clasificación | 5 días | lun 2/10/17 |
| 24 | | Act. 11: Acoplamiento de los sensores a las resbaladeras | 1 día | lun 9/10/17 |
| 25 | | Act. 12: Acondicionamiento de los sensores infrarrojos para detectar objetos | 2 días | mar 10/10/17 |
| 26 | | Act. 13: Ajuste de parámetros en el código de visión artificial para el prototipo desarrollado | 3 días | mar 10/10/17 |
| 27 | | Act. 14: Desarrollo del programa de control en el Arduino MEGA | 8 días | vie 13/10/17 |
| 28 | | Act. 15: Desarrollo del programa de control en el Arduino Nano | 3 días | mié 25/10/17 |

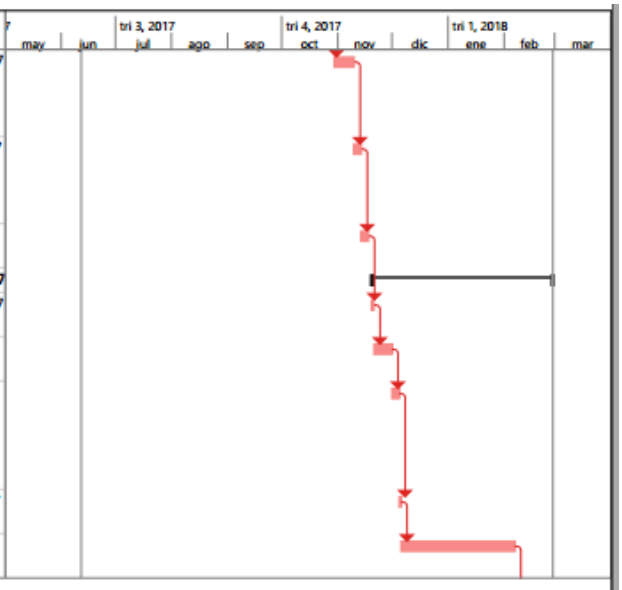



Proyecto: ProyectoClasificador
Fecha: mar 4/9/18

Tarea
División
Hito
Resumen
Resumen del proyecto
Tarea inactiva
Hito inactivo
Resumen inactivo
Tarea manual
solo duración
Informe de resumen manual
Resumen manual
solo el comienzo
solo fin
Tareas externas
Hito externo
Fecha límite
Tareas críticas
División crítica
Progreso
Progreso manual



| Id | Modo de tarea | Nombre de tarea | Duración | Comienzo |
|---|---|---|---|---|
| 29 | | Act. 16: Realización de las placa para integrar el sistema de visualización del proceso | 9 días | lun 30/10/17 |
| 30 | | Act. 17: Configuración de la velocidad de la banda transportadora y los grados del servomotor | 2 días | vie 10/11/17 |
| 31 | | Act. 18: Realización de las pruebas de clasificación | 4 días | mar 14/11/17 |
| 32 | | **FASE 3** | **72 días** | **lun 20/11/17** |
| 33 | | Act. 1: Calibración del sistema implementado | 1 día | lun 20/11/17 |
| 34 | | Act. 2: Realización de una interface gráfica de usuario | 8 días | mar 21/11/17 |
| 35 | | Act. 3: Modificación del programa para realizar comunicación entre la computadora y la tarjeta embebida. | 2 días | vie 1/12/17 |
| 36 | | Act. 4: Pruebas finales del prototipo | 1 día | mar 5/12/17 |
| 37 | | Act. 5: Realización del escrito | 45 días | mié 6/12/17 |




Proyecto: ProyectoClasificador
Fecha: mar 4/9/18

Tarea
División
Hito
Resumen
Resumen del proyecto
Tarea inactiva
Hito inactivo
Resumen inactivo
Tarea manual
solo duración
Informe de resumen manual
Resumen manual
solo el comienzo
solo fin
Tareas externas
Hito externo
Fecha límite
Tareas críticas
División crítica
Progreso
Progreso manual


Id
Modo de tarea
Nombre de tarea
Duración
Comienzo
7
may
jun
tri 3, 2017
jul
ago
sep
tri 4, 2017
oct
nov
dic
tri 1, 2018
ene
feb
mar
38
Act. 6 Modificaciones del prototipo
15 días
mié 7/2/18
39
FIN
0 días
mar 27/2/18
27/2
40
Proyecto: ProyectoClasificador
Fecha: mar 4/9/18
Tarea
División
Hito
Resumen
Resumen del proyecto
Tarea inactiva
Hito inactivo
Resumen inactivo
Tarea manual
solo duración
Informe de resumen manual
Resumen manual
solo el comienzo
solo fin
Tareas externas
Hito externo
Fecha límite
Tareas críticas
División crítica
Progreso
Progreso manual
Página 5

## DECLARACIÓN Y AUTORIZACIÓN

Yo, Jorge Luis Garcés Zamora, CI 1710964337 autor del trabajo de graduación:

**Prototipo para clasificación de ropa hospitalaria a través de etiquetas de color utilizando visión artificial mediante Arduino**, previo a la obtención del título de **Ingeniería en Electrónica Digital y Telecomunicaciones** en la UNIVERSIDAD TECNOLÓGICA ISRAEL.

1.- Declaro tener pleno conocimiento de la obligación que tienen las instituciones de educación superior, de conformidad con el Articulo 144 de la Ley Orgánica de Educación Superior, de difundir el respectivo trabajo de graduación para que sea integrado al Sistema Nacional de Información de la Educación Superior del Ecuador para su difusión pública respetando los derechos de autor.

2.- Autorizo a la SENESCYT a tener una copia del referido trabajo de graduación, con el propósito de generar un repositorio que democratice la información, respetando las políticas de propiedad intelectual vigentes.

Quito, Septiembre del 2018

**Atentamente.**

**Jorge Luis Garcés Zamora**
**C.I.** 1710964337