TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE NUEVO LEÓN
División de Estudios Profesionales

Trabajo de Titulación.

Opción 1: TESIS.

Proyecto: “IMPLEMENTACIÓN DE UN SISTEMA DE VISIÓN PARA UNA PLATAFORMA ROBÓTICA CON APLICACIONES DE BÚSQUEDA Y RESCATE.”

ALUMNO(S):	Francisco Javier Pérez Encina
No. CONTROL:	15480429
CARRERA:	Ingeniería en Mecatrónica.
ASESOR DE TESIS:	M.I.M Juan Ángel Rodríguez Salinas.
REVISORES:	

Guadalupe, N.L. Diciembre, 2020

RESUMEN:

Se realizó un sistema de visión basado en Python y Opencv, para su implementación en una plataforma móvil con fines de búsqueda y rescate.

Primero se evaluaron las capacidades de procesamiento que tendría el robot para poder ejecutar las tareas de control y desde luego las de visión, ya que el sistema de visión debe cumplir con tres tareas primordiales las cuales son detectar códigos QR, detectar patrones (danger labels), y finalmente tener visión térmica.

Para ello se realizaron pruebas en mini ordenadores de bajo consumo y muy económicos como las tarjetas Jetson nano de la marca Nvidia y Raspberry pi, siendo esta ultima la que mejor se pudo adaptar a la compatibilidad de los sensores y librerías que se emplearon en el algoritmo final.

El algoritmo consta de librerías e instrucciones en Python, el cual es un lenguaje de programación muy flexible al momento de usar todo tipo de librerías. Por otro lado tenemos Opencv, el cual es una plataforma con muchas librerías basadas en el desarrollo de la visión artificial, y fue de mucha ayuda al momento de hacer las instrucciones de detección de patrones y QR. Respecto a este último, su respectiva librería fue la llamada Zbar, la cual es especializada en la detección y escaneo de códigos QR, que en conjunto con Opencv, se mejoró la interacción que tiene al momento de detectar un QR.

Posteriormente la cámara térmica se programó con base en sus librerías de la propia marca, adjuntando otras como la llamada Scipy la cual fue de gran apoyo para la interpolación de los datos mandados por el sensor de temperatura.

Finalmente todas las partes del algoritmo que constituyeron a los códigos QR, patrones y visión térmica, se juntaron en un solo algoritmo que es capaz de procesar la información de estos tres aspectos clave para la visión del robot, esto ahorrando el uso de cámaras, espacio en la pantalla del operador y por lo tanto mayor comodidad y facilidad al manejo de esta interfaz.

ABSTRACT.

A vision system was made, based on Python and opencv was developed for implementation on a mobile platform for search and rescue purposes.

First, the processing capabilities that the robot would have were evaluated in order to execute the control tasks and, of course, the vision tasks, since the vision system must fulfill three main tasks, which are to detect QR codes, to detect patterns (danger labels), and finally to have thermal vision.

For this purpose, tests were carried out on low-consumption and very economical mini-computers such as Nvidia's Jetson nano and Raspberry pi cards, the latter being the one that could best adapt to the compatibility of the sensors and libraries used in the final algorithm.

The algorithm consists of libraries and instructions in Python, which is a very flexible programming language when using all kinds of libraries. On the other hand we have Opencv, which is a platform with many libraries based on the development of machine vision, and was very helpful when making the pattern detection and QR instructions. Regarding the last one, its respective library was the one called Zbar, which is specialized in the detection and scanning of QR codes, which together with Opencv, improved the interaction it has when detecting a QR.

Later, the thermal camera was programmed based on its own brand libraries, adding others like Scipy which was of great support for the interpolation of data sent by the temperature sensor.

Finally all the parts of the algorithm that constituted the QR codes, patterns and thermal vision, were joined in a single algorithm that is capable of processing the information of these three key aspects for the vision of the robot, this saving the use of cameras, space on the screen of the operator and therefore greater comfort and ease of use of this interface.

Dedicatorias.

Para mi familia, mi padre y mi madre primordialmente. Para mis preciados compañeros de equipo y maestros asesores que me acompañaron a lo largo de estos años de investigación referente a este proyecto.

AGRADECIMIENTOS.

En agradecimiento principalmente con mi familia, mi mama y mi papa por el apoyo que me dieron a lo largo del desarrollo del proyecto tanto en el aspecto económico como en el motivacional. Seguidamente a mis compañeros y maestros asesores del tecnológico de Nuevo León, que me permitieron colaborar y compartir ideas para llevar a cabo mi aportación al proyecto.

ÍNDICE.

LISTA DE FIGURAS.

LISTA DE TABLAS.

CAPITULO 1.

INTRODUCCIÓN.

1.1 Conceptos básicos.

México, un país construido por las riquezas y costumbres de su gente, un país que, a pesar de no tener el nivel de una potencia de primer nivel, y que, si es verdad que en ocasiones entra en conflicto con su propia gente por problemas de pobreza, corrupción, desigualdad, el no tener una calidad de educación apropiada, robos, discriminación etc. En momentos críticos ocurre algo que nos diferencia entre muchos otros países, y es el hecho de unirse para ayudar a los demás, gente que no tendrá la preparación de un rescatista, médico o soldado, sea capaz de meterse en zonas de riesgo con el fin de ayudar a los demás sin esperar nada a cambio, ese es el mejor punto a favor de nuestro país, por ejemplo, el 19 de septiembre de 1985 en la ciudad de México, un terremoto de 8.1 grados Richter donde 2800 edificios sufrieron daños críticos y 800 fueron derrumbados [1]. Además de contar con aproximadamente 12,000 muertos y cientos de desaparecidos, esto sin mencionar la crisis económica que ya sufría nuestro país y que posteriormente ocurrió debido a esta catástrofe que incluso hasta en la década de los noventa aún había áreas sin reconstrucción. Recientemente el 19 de septiembre de 2019 como por arte de coincidencia, se repite la historia con otro terremoto de 7.1 grados Richter en el centro de la ciudad de México, dejando un saldo de aproximadamente 369 muertes y más de cien desaparecidos, nos dejan con la reflexión si realmente aprendimos una lección acerca de cómo actuar ante estos eventos [2].

Con base en estos datos, nace una necesidad de ayudar o al menos dar una aportación en el área tecnológica para actuar ante estos escenarios, y, en el Instituto Tecnológico de Nuevo León, surgió la idea de formar un equipo conformado por maestros y alumnos, para elaborar un prototipo de un robot de búsqueda y rescate, midiéndose en competencias que adaptan escenarios simulados de desastres naturales como lo son los terremotos, a nivel regional, nacional e internacional las

cuales cuenta mucho que dicho robot tenga una buena interfaz de comunicación, destreza y lo más importante, un sistema de visión.

Un sistema de visión artificial, es aquel que, mediante el uso de cámaras, se puede obtener información acerca del entorno en el que ellas estén y con base en esa información poder tomar alguna acción o análisis del mismo, esto permitirá al operador observar el escenario frente al robot sin ponerse en riesgo o a su equipo de rescate. En resumen, el siguiente documento tendrá como objetivo mostrar el desarrollo de una aplicación de visión artificial enfocada en tareas de un robot de rescate y sus algoritmos. Además de hacer énfasis en los detalles técnicos de los equipos de cómputo (hardware), ya que de estos depende la eficiencia del procesamiento de los algoritmos de visión.

1.1.-Antecedentes:

Existen diversos sistemas de visión que a lo largo de la historia han evolucionado, antes se pensaba que una cámara solo servía para capturar fotos y recuerdos, sin embargo una cámara es un poderoso sensor que brinda mucho más que eso y se le puede aprovechar para aplicarlo en sistemas inteligentes, algo similar a esta propuesta es la que desarrollaron Sharma y D. J. Shah, donde se plantea un diseño de un sistema de visión por computadora para la detección de animales de carretera para evitar accidentes tanto por atropellar al animal como chocar con otro vehículo al intentar esquivarlos [3]. En dicho sistema se plantea el uso de *Opencv*, el cual es un conjunto de librerías de visión artificial y *machine learning* creada originalmente por Intel desde 1999 y que ahora es de código abierto lo que la hace compatible con lenguajes de programación como Java, Python, c++, Matlab entre otros. Además de emplear la técnica *Cascade*, lo que significa que tiene una base de datos con una cantidad considerable de fotos, en este caso de animales de granja. La cámara al detectar a estos animales en el camino, pasa a alertar al conductor el cual llevará la cámara en el vehículo. Consideraron usar una cámara de 30 fps cerca del espejo retrovisor mirando en línea recta a la carretera y una computadora Intel i5-2430M con 4GB de RAM y un CPU de 2.40 GHz, el cual es un procesador de gama media-

alta con un costo aproximado de $285 dólares en línea. Además de emplear Opencv y Visual Studio 10 en Windows 7, esto se considera bastante bien para correr algoritmos de visión que no requieren de tanta complejidad, además de ser fácil de implementar para nuestra propuesta.

M. Ben Ayed y sus colaboradores Elkosantini, S. A. Alshaya M. y Abid plantearón un sistema similar con las técnicas de *Cascade* y Opencv, esta vez con el propósito de detectar rostros de personas con el fin de ver actividades sospechosas, en pocas palabras, un sistema de visión de seguridad para evitar robos a bancos, aeropuertos, tiendas comerciales etc [4].

En este proceso una Intel i5-5200 CPU 2.2 GHz. evaluado en 65 dólares aproximadamente. Además de una placa *raspberry pi 3*, la cual es un mini ordenador con una velocidad de 1.4GHz. y 1 GB de RAM, evaluada en 60 dólares, con un sistema operativo raspbian Linux, usada para los métodos del modelo matemático y una cámara HP. Este método es factible ya que se presenta la colaboración entre dos equipos (La Intel y la raspberry), que a pesar de que es más espacio y hardware, mientras más económico y factible mucho mejor.

De momento podemos ver que Opencv es una pieza fundamental y que es muy importante, saber escoger el equipo en donde los algoritmos se ejecutarán, un buen sistema de visión depende del nivel de procesamiento de su hardware y la calidad de sus cámaras, esto también implica a que los precios se elevarán, así que el reto es hacerlo con el menor gasto posible. Otra propuesta la comparten V. Mazzia y sus colaboradores A. Khaliq, F. Salvetti y M. Chiaberge, en su sistema de detección de manzanas en tiempo real, la novedad en su investigación, fue el empleo de un sistema de redes neuronales llamada *YOLO* [5]. Este sistema creado por Joseph Redmon de la Universidad de Washington es capaz de procesar y clasificar imágenes en una potente red neuronal. Entonces aprovechando esta tecnología, se empleó un sistema para la detección de manzanas en huertos para estimar su estado y saberlas aprovechar más en las cosechas. En una base de datos con imágenes de manzanas se realizó el entrenamiento de la red neuronal mejorando el rango de las escalas, se logró más precisión al momento de hacer la detección

que antes cuando solo se usaba la configuración preestablecida de YOLO v3. Además, se usaron diferentes plataformas como *Raspberry pi 3*+ con una *Intel NCS2* evaluada en 160 dólares en línea y una *Jetson nano de la marca Nvidia*, la cual es otro mini ordenador similar a raspberry, pero de una velocidad de 1,43 GHz y 4GB de RAM implementada para el desarrollo de inteligencia artificial evaluada en 140 dólares aproximadamente.

Una parte importante del desarrollo del sistema de visión empleado en esta tesis, es que sea capaz de realizar lecturas de códigos, más en específico, de códigos QR los cuales Xiang Zhang en colaboración con Hangzai Luo, Jinye Peng, Jianping Fan y Long Chen propusieron analizar *Zbar* y *Zxing* los cuales son algoritmos de código abierto para la detección de códigos QR empleando el algoritmo de Canny, el cual fue desarrollado por John F. Canny en 1986 y utiliza un algoritmo de múltiples etapas para detectar una amplia gama de bordes en imágenes [6].

En su investigación realizan la detección de códigos QR procesando la imagen, para posteriormente aplicar el método Canny (detectar los bordes del QR). *Zbar* es una herramienta que facilitara la parte de la detección de los códigos QR.

Finalmente, otro apartado que el sistema de visión propuesto en esta tesis debe cumplir y que es muy importante sobre todo para la búsqueda y exploración es la visión térmica. Es importante considerar que una cámara térmica profesional tiene un precio muy elevado, por ejemplo, una cámara térmica de tipo pistola como la E4 e FLIR, tienen un costo de hasta $35, 000 MXN en línea, por lo tanto, se considera un módulo muy económico de la marca *Adafruit* compatible con raspberry, el cual es el *Amg8833* que tiene un costo de apenas $1,500 MXN en línea.

DPS Setyohadi con sus colaboradores V Rahmania, H Y Riskiawan, S T Sarena, S Arifin, DE Putra y Edy Setiawan usaron este módulo para el procesamiento de imágenes térmicas usando redes neuronales y de esa manera tener el sistema de control de posición de un horno [7], además de analizar los métodos de extracción de información de las imágenes mediante RGB y HSV, los cuales son modelos de colores que nos facilitan la detección de objetos en imágenes, y que también tenemos contemplado de usar en esta propuesta.

1.2.- Planteamiento del problema:

Se tiene una plataforma móvil la cual tiene como objetivo desenvolver el papel de búsqueda, exploración y rescate en situaciones de alto riesgo. Se requiere de un sistema de visión artificial que sea capaz de facilitar al operador del robot dichos papeles sin la necesidad de ponerse en riesgo así mismo o a su equipo o demás rescatistas.

1.3.- Propuesta de solución.

Con base a lo anterior mencionado, en esta propuesta se utilizará la herramienta Zbar para la detección de QR en conjunto con las librerías de Opencv, el cual mediante el algoritmo de Canny, se detectarán los patrones (imágenes de símbolos de seguridad y precaución también llamados *hazard labels*), que el robot debe detectar además de los QR. Finalmente, para la visión térmica se empleará un módulo amg8833, utilizando una jetson nano (aunque raspberry también está contemplada), con web cams y módulos de cámaras de 5mpx.

1.4.- Objetivo general.

Implementar un sistema de visión que realice la detección de códigos QR, detectar patrones de seguridad y tener visión térmica que pueda facilitar el trabajo de los rescatistas evaluando la situación de riesgo para evitar accidentes hacia ellos o su equipo de rescate.

1.5.- Objetivos específicos.

- Implementar un modelo matemático ya propuesto para la detección de imágenes.
- Instalar y preparar el hardware y software para la implementación de las cámaras y las librerías a usar (Opencv, Zbar, Python, etc.)
- Implementar un algoritmo de detección de los códigos QR y Patrones.
- Realizar pruebas en fotos y en tiempo real de la detección de QR y Patrones.

- Realizar pruebas de visión térmica con la amg8833.

1.6.- Justificación.

Muchas regiones de México están propensas a sufrir por desastres naturales ya que no contamos con la infraestructura adecuada para soportar estos eventos, por lo cual cada que ocurre un evento siempre hay pérdidas humanas incontables, y de ahí la propuesta de esta tesis va más allá que la de simplemente obtener puntos extra en una competencia, sino que se espera ser un escalón que impulse más adelante a que surja un sistema de rescate eficiente y práctico para los rescatistas y esto derive a menos pérdidas humanas cada vez que surjan desastres naturales o terroristas.

CAPITULO 2.

PROCESAMIENTO DE IMÁGENES.

En el siguiente capítulo se describe el procedimiento a detalle de todas las variables y aspectos teóricos que lleva a cabo el algoritmo para el procesamiento de las imágenes, explorando métodos y funciones del software que facilitaran la detección de patrones o señales de peligro (Hazard labels) y códigos QR.

2.1.- Visión por computadora.

La visión por computadora es el proceso mediante el cual se toma una imagen y se produce una nueva versión modificada de esta imagen. Con el fin de obtener información de ella para sustentar una necesidad.
También la podemos definir como un análisis mediante el cual a partir de una imagen se obtiene una medición, interpretación o decisión. [10]

El principal objetivo de este estudio es reconocer y localizar objetos en el ambiente mediante el procesamiento de las imágenes. Aunque ambos campos tienen mucho en común, el objetivo final de procesamiento de imágenes es mejorar la calidad de las imágenes para su posterior utilización o interpretación.
Para la visión por computadora hay que tener en claro tres aspectos importantes. Primero que visión como ya se ha mencionado es un proceso computacional, la descripción de las imágenes a obtener depende del observador y es necesario eliminar la información que no sea útil (reducción de información). [11]

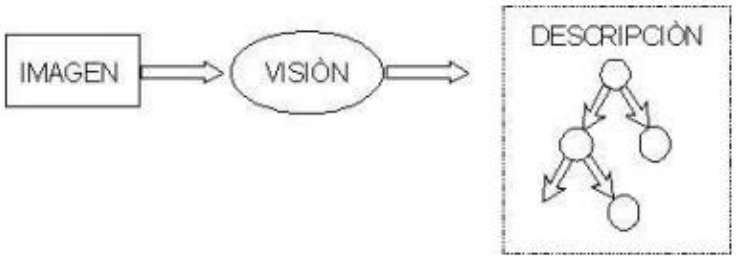

Figura 1.- Esquema general de la visión por computadora. [11]

La visión artificial y el procesamiento de imágenes, está estrictamente ligado ya que se requieren de procedimientos como filtros para poder procesar una imagen, de ahí la información obtenida puede ser utilizada para algún objetivo en específico.

Procesamiento: con el fin de mejorar la calidad de la imagen obtenida se emplean ciertos filtros digitales que eliminan el ruido en la imagen o bien aumentan el contraste.

Adquisición de la imagen: se obtiene la imagen adecuada del objeto de estudio. Dependiendo de la aplicación la imagen puede ser una fotografía, radiografía, termografía, etc.

Segmentación: Proceso por el cual se pretende separar el objeto de estudio del fondo de la imagen.

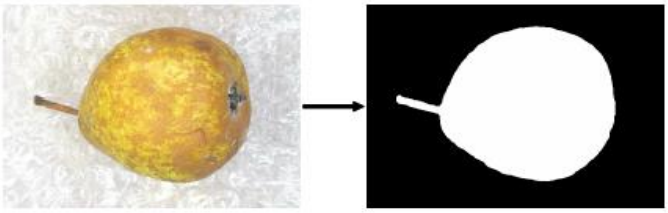

Figura 2.- Segmentación de una imagen. [10]

Medición (extracción de características): se realiza una medición objetiva de ciertos atributos de interés del objeto de estudio.

Interpretación (clasificación): de acuerdo a los valores obtenidos en las mediciones se lleva a cabo una interpretación del objeto. [10]

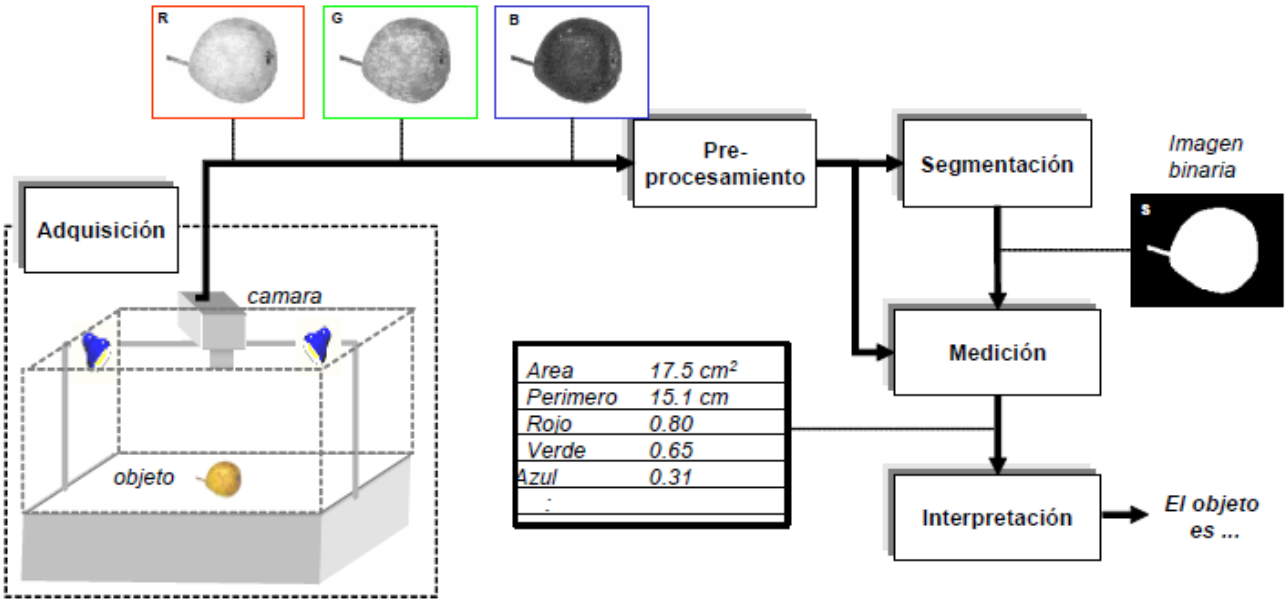

Figura 3.- Esquema detallado del procesamiento de imágenes. [10]

2.2.-Pixeles e Histogramas.

Un pixel como el elemento más básico en una imagen. Estos contienes diferentes valores de intensidad luminosa.

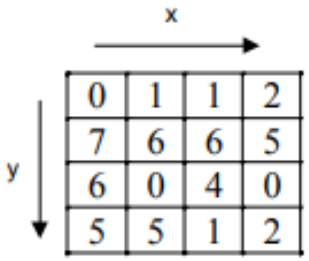

Fig. 4 Imagen de 16 pixeles. [12]

Matemáticamente, una imagen se representa por r = f (x, y), donde r es la intensidad luminosa del píxel cuyas coordenadas son (x, y). Un sistema para procesar imágenes se representa como g (x, y) = T [ ] f (x, y).

Además de definir un color como aquel que se forma mediante la combinación de los tres colores básicos rojo, azul y verde (RGB por sus siglas en ingles).

Una de las herramientas más empleadas para saber la distribución de los pixeles, es el histograma.

El histograma es una representación gráfica de la distribución del color en una imagen. [12]

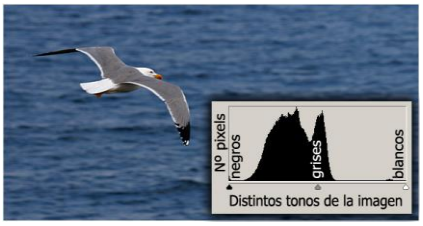

Fig. 5 Ejemplo de un histograma con distintos valores de pixeles (blanco y negro). [12]

Para la detección de imágenes se deberá adquirir la información de importancia en las imágenes, es decir la información que los bordes de estas mismas nos brindan, para posteriormente procesar y clasificar. Se empleará para ello el algoritmo de Canny.

2.3.- Algoritmo de Canny.

En 1986, Canny propuso un método para la detección de bordes, el cual se basa en tres criterios:

- **La detección.** Evita la eliminación de bordes importantes y no detectar bordes falsos o no deseados.
- **Localización.** Establece que la distancia entre la posición real y la localizada del borde se debe minimizar.
- **Respuesta.** Estima múltiples respuestas correspondientes a un único borde.

Uno de los métodos relacionados con la detección de bordes es el uso de la primera derivada, la cual se emplea para tomar el valor de cero en todas las regiones donde no varía la intensidad y tiene un valor constante en toda la transición de intensidad. Por lo tanto, un cambio de intensidad en una imagen, se considera como un cambio brusco en la primera derivada [9], característica dada para la aplicación del algoritmo de Canny.

El algoritmo de Canny consiste en tres pasos.

1. Obtención de la gradiente: Calculamos la magnitud y orientación del vector gradiente de cada pixel.
2. Supresión: en este punto adelgazaremos el ancho de los bordes, obtenidos en el gradiente, hasta lograr bordes de un pixel de ancho.
3. Histéresis de umbral: Este paso se pretende eliminar bordes falsos, aplicando histéresis basado en el nivel de umbral. [10]

2.3.1.- Obtención de la gradiente.

Antes de obtener la gradiente, debemos tener claro que existe ruido en las imágenes las cuales nos pueden dar resultados no deseados provocando errores en las detecciones, por lo cual existen muchas técnicas de filtrado para suavizar la imagen original y eliminar el mayor ruido posible, una de esas técnicas para suavizar es empleando el filtro Gaussiano.

El filtro Gaussiano, es un filtro con características de filtro pasa-bajo, cuyos coeficientes están determinados por los valores de una función de distribución Gaussiana [13].

$$f(x) = ae\frac{-(x-u)^2}{2\sigma^2} \qquad (1)$$

Donde

- a: constantes (a>0).
- u: Es la media.
- σ^2: Es la varianza.

La máscara más empleada en el filtro Gaussiano basada en una distribución Gaussiana es:

$$W = \frac{1}{16} * \begin{matrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{matrix} \qquad (2)$$

Esta máscara, más conocida como kernel, es una matriz de 3x3 que se caracteriza por asignar un mayor peso al píxel central y a los píxeles que se encuentran cercano a este, y menor peso a los píxeles alejados, provocando así un suavizado en la imagen [11].

Jorge Valverde Rebaza realiza el siguiente análisis empleando ya un ejemplo de imágenes con el filtro gaussiano para aplicar el algoritmo de Canny [10]. Una vez suavizada la imagen, para cada píxel se obtiene la magnitud y módulo (orientación) del gradiente, obteniendo así dos imágenes.

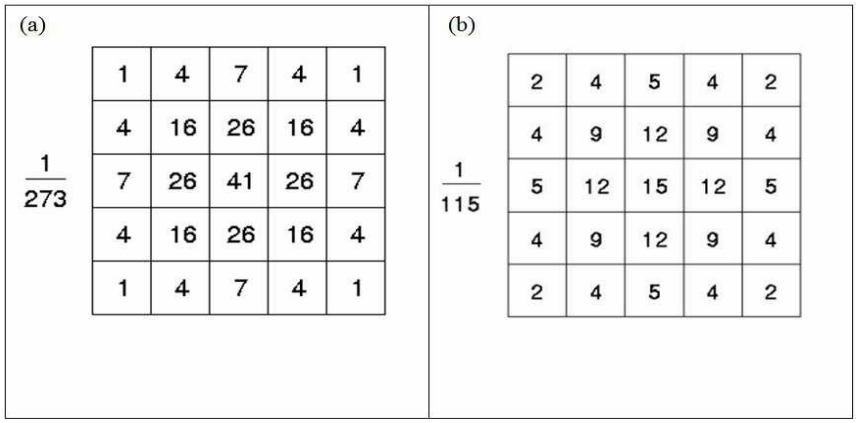

Fig. 6 Ejemplo de imágenes con filtro gaussiano, obtenidas de [13].

2.3.2.- Supresión no máxima al resultado del gradiente.

Las dos imágenes generadas en el paso anterior sirven de entrada para generar una imagen con los bordes más delgados. Se consideran cuatro direcciones identificadas por las orientaciones de 0º, 45º, 90º y 135º con respecto al eje horizontal. Para cada píxel se encuentra la dirección que mejor se aproxime a la dirección del ángulo de gradiente.

Entrada	Salida
Imagen (i) Con máscara de convolución H, con media cero y desviación estándar σ .	Imagen E_m de la magnitud del gradiente.
	Imagen Eo de la orientación del gradiente.

Tabla 1.- Representación del algoritmo para la obtención de la gradiente. [13]

1. Suavizar la imagen (i) con H mediante un filtro gaussiano y obtener J como imagen de salida.
2. Para cada píxel (i, j) en J, obtener la magnitud y orientación del gradiente basándose en las siguientes expresiones:
   El gradiente de una imagen f (x, y) en un punto (x, y) se define como un vector bidimensional dado por la ecuación:

$$G[(f(x,y)] = \begin{matrix} Gx \\ Gy \end{matrix} = \begin{matrix} \frac{\partial}{\partial x} f(x,y) \\ \frac{\partial}{\partial y} f(x,y) \end{matrix} \qquad (3)$$

Siendo un vector perpendicular al borde, donde el vector G apunta en la dirección de variación máxima de f en el punto (x, y) por unidad de distancia, con la magnitud y dirección dadas por:

$$|G| = \sqrt{Gx^2 + Gy^2} = |Gx| + |Gy| \qquad (4)$$

$$\varphi(x,y) = \tan^{-1}\frac{Gy}{Gx} \qquad (5)$$

3. Obtener E_m a partir de la magnitud de gradiente y E_o a partir de la orientación, de acuerdo a las expresiones anteriores.

Posteriormente se observa si el valor de la magnitud de gradiente es más pequeño que al menos uno de sus dos vecinos en la dirección del ángulo obtenida en el paso anterior. De ser así se asigna el valor 0 a dicho píxel, en caso contrario se asigna el valor que tenga la magnitud del gradiente.

La salida de este segundo paso es la imagen I_n con los bordes adelgazados, es decir, E_m (i j), después de la supresión no máxima de puntos de borde. [13]

2.3.3.-Histéresis de umbral a la supresión no máxima.

Los resultados obtenidos en el paso anterior suele contener máximos locales creados por el ruido. Una solución para eliminar dicho ruido es la histéresis del umbral. El proceso consiste en tomar la imagen obtenida del paso anterior, tomar la orientación de los puntos de borde de la imagen y tomar dos umbrales, el primero más pequeño que el segundo. Para cada punto de la imagen se debe localizar el siguiente punto de borde no explorado que sea mayor al segundo umbral. A partir de dicho punto seguir las cadenas de máximos locales conectados en ambas direcciones perpendiculares a la normal del borde siempre que sean mayores al primer umbral. Así se marcan todos los puntos explorados y se almacena la lista de todos los puntos en el contorno conectado. Es así como en este paso se logra eliminar las uniones en forma de Y de los segmentos que confluyan en un punto. [13]

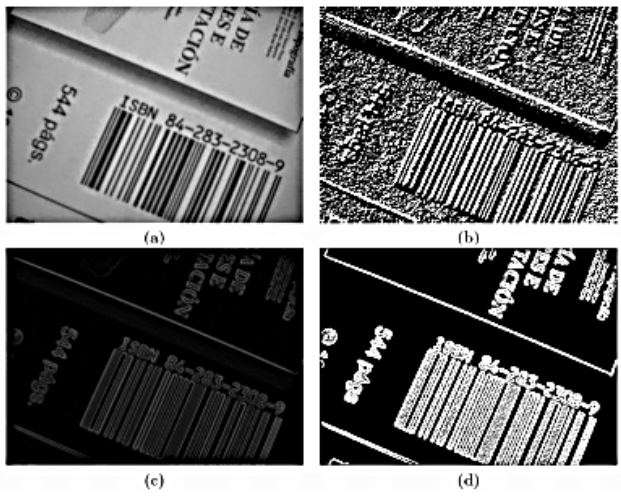

Figura 7.- Representación de los pasos anteriores para la detección de bordes mediante canny. A) Imagen original, B) Orientación, C) Supresión no máxima, D) Histéresis de umbral. [13]

Frecuentemente, es común que un cuarto y último paso se realice en el algoritmo de Canny, este paso consiste en cerrar los contornos que pudiesen haber quedado abiertos por problemas de ruido.

Un método muy utilizado es el algoritmo de *Deriche* y *Cocquerez*. Este algoritmo utiliza como entrada una imagen binarizada de contornos de un píxel de ancho. El algoritmo busca los extremos de los contornos abiertos y sigue la dirección del máximo gradiente hasta cerrarlos con otro extremo abierto.

El procedimiento consiste en buscar para cada píxel uno de los ocho patrones posibles que delimitan la continuación del contorno en tres direcciones posibles. Esto se logra con la convolución de cada píxel con una máscara específica. Cuando alguno de los tres puntos es ya un píxel de borde se entiende que el borde se ha cerrado, de lo contrario se elige el píxel con el valor máximo de gradiente y se marca como nuevo píxel de borde y se aplica nuevamente la convolución. Estos pasos se repiten para todo extremo abierto hasta encontrar su cierre o hasta llegar a cierto número de iteraciones determinado [13]. Los resultados obtenidos de este proceso se presentan a continuación en las figuras 8, 9 y 10.

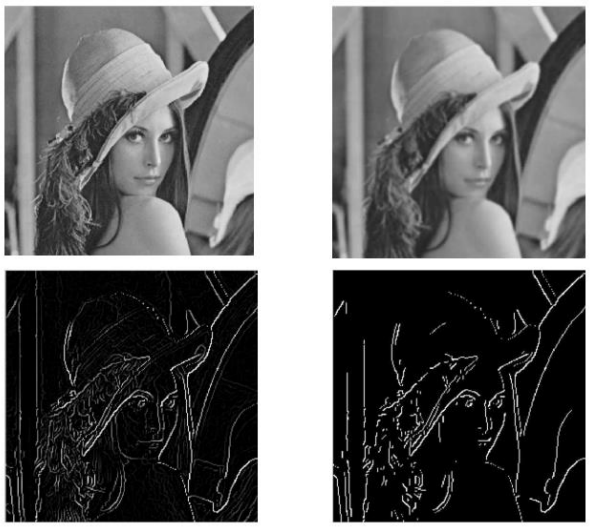

Figura 8.- Muestra general del proceso de detección de bordes mediante el algoritmo de Canny. [13]

Figura 9. Se muestran estas dos figuras en las que se aplicó el algoritmo de detección de bordes de Canny con umbrales diferentes al aplicar la histéresis. En la figura izquierda se usó umbrales con los valores 110 y 150. En la figura de la derecha se usó umbrales con los valores 120 y 180. [13]

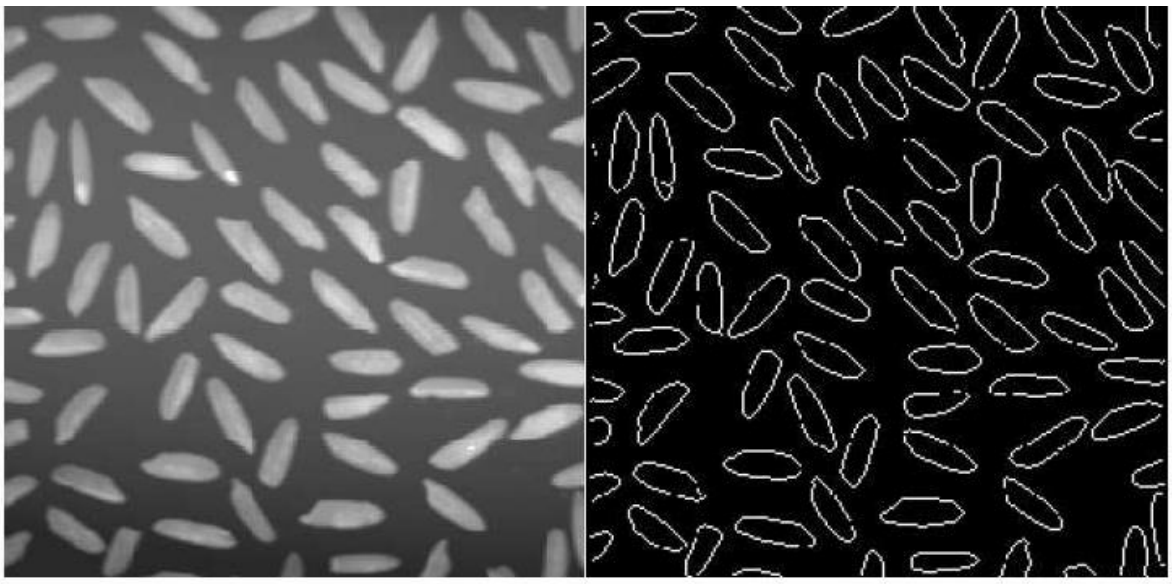

Figura 10. Se muestran otras dos figuras con más elementos en las que se aplicó el algoritmo de detección de bordes de Canny, donde la de la izquierda es la original y la de la derecha el resultado final. [13]

2.4.- Python.

Los lenguajes de programación son la herramienta básica de construcción de programas, como lo son el machete y el azadón para un campesino, el pico y la pala para un constructor.

Durante los años 90 ocurre una serie de eventos que marcan ciertos pautas para el futuro desarrollo del software libre, como es el lanzamiento de la primera versión del núcleo Linux por Linus Torvalds en 1991, y en ese mismo año Guido van Rossum libera la primera versión del lenguaje de programación Python.

Python cuenta con facilidades para la programación orientada a objetos, imperativa y funcional, por lo que se considera un lenguaje multi-paradigmas. Fue basado en el lenguaje ABC y se dice que fue influenciado por otros como C, Algol 60, Modula-3 e Icon según su propio autor.

Es un lenguaje de alto nivel ya que contiene implícitas algunas estructuras de datos como listas, diccionarios, conjuntos y tuplas, que permiten realizar algunas tareas complejas en pocas líneas de código y de manera legible.

La sintaxis de Python es muy sencilla, tanto que en algunas ocasiones parece pseudocódigo. Es muy interesante observar las diferencias que existen entre el programa Hola Mundo de Python y el de otro lenguaje de alto nivel como C++:

```
#include <iostream>
using namespace std;
int main()
{
cout << "Hello World" <<
endl;
return 0;
}
```

```
print "Hello World"
```

Figura 11.-Comparacion de sintaxis de un “Hola mundo”, entre C++ (izquierda) y Python (derecha). [14]

Una de las fortalezas de Python, y quizás la mayor, es la librería estándar con que cuenta. Con decenas de módulos cubre la mayoría de las necesidades básicas de un programador y mucho más. En esta se le da cobertura de forma muy intuitiva a tópicos como:

• Cadenas

• Estructura de datos

• Funciones numéricas y matemáticas

• Compresión de datos

• Formatos de archivo

• Criptografía

• Servicios de los Sistemas Operativos

• Comunicación entre Procesos

• Manejo de datos de Internet

• Servicios multimedia

• Manejo de excepciones.

Existen más de 40 tópicos en la referencia de la librería de Python4 por lo que se puede afirmar que es una de las más completas con que se cuenta en la actualidad, comparable con la de Java y .NET.

De todas las características que posee Python, una de las más importantes es su capacidad de reutilizar código escrito en los lenguajes C y C++. Existen mecanismos que hacen muy sencilla la tarea de envolver funciones y clases hechas en estos lenguajes, entre los que se encuentran Boost. Python, Sip y Shiboken. La importancia de esta integración es relevante, ya que las bases de código en lenguajes como C y C++ son las más grandes disponibles por el software libre hoy en día, y permiten no tener que duplicar código ya existente.

Uno de los grandes mitos alrededor de Python es acerca de su pobre rendimiento. Esto no es del todo correcto, ya que aunque es un lenguaje interpretado y estos tienden a ser más lentos que los lenguajes compilados, Python, a diferencia de otros lenguajes interpretados, ha implementado toda su librería estándar en el lenguaje C, lo que hace que sus funciones primitivas sean bastante eficientes. Además, puede compilarse su código a bytecodes, similar al que usan Java y .NET, lo que optimiza aún más el proceso de interpretación [14].

Las siguientes librerías son de gran relevancia para fines de este proyecto:

2.4.1.- NumPy.

NumPy es un paquete de Python que significa "*Numerical Python*", es la librería principal para la informática científica, proporciona potentes estructuras de datos, implementando matrices y matrices multidimensionales. Estas estructuras de datos garantizan cálculos eficientes con matrices.

Usando NumPy en Python proporciona una funcionalidad similar a MATLAB ya que ambas interpretadas, y ambos permiten al usuario escribir programas rápidos, siempre y cuando la mayoría de las operaciones funcionan en arrays o matrices en lugar de escalares . En comparación, MATLAB cuenta con un gran número de cajas de herramientas adicionales, en particular Simulink , mientras que NumPy se integra intrínsecamente con Python, un lenguaje de programación más moderno y completo. Por otra parte, los paquetes de Python complementarios están disponibles; SciPy es una biblioteca que añade más MATLAB-como funcionalidad y Matplotlib es un paquete de trazado que proporciona MATLAB-como el trazado de funcionalidad. Internamente, tanto MATLAB y NumPy confían en BLAS y LAPACK para los cálculos de álgebra lineal eficientes.

Python fijaciones de la ampliamente usada visión por ordenador de la biblioteca OpenCV utilizan matrices NumPy para almacenar y operan en los datos. Dado que

las imágenes con múltiples canales simplemente se representan como matrices tridimensionales, la indexación, rebanar o enmascaramiento con otros arreglos son formas muy eficientes a los píxeles de acceso específicas de una imagen. La matriz NumPy como estructura universal de datos en OpenCV para imágenes, extraídas puntos de función, los núcleos de filtro y muchos simplifica enormemente el flujo de trabajo más programación y depuración.

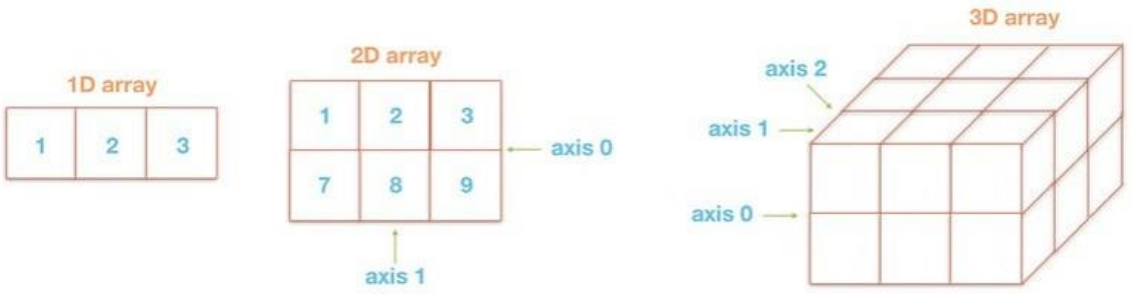

Fig. 12.- Representación de una matriz unidimensional o 1D. Una matriz bidimensional o 2D, en donde las filas se indican como el eje 0, mientras que las columnas son el eje 1. Y una en 3D. [15]

2.4.2.-SciPY.

Scipy es una biblioteca de código abierto de herramientas y algoritmos matemáticos que nació a partir de la colección original de Travis Oliphant y que consistía en módulos de extensión para Python. Scipy contiene módulos para optimización, álgebra lineal, integración, interpolación, funciones especiales, FFT, procesamiento de señales e imagen, resolución de EDOs y otras tareas relacionadas con la ciencia e ingeniería. Está dirigida al mismo tipo de usuarios que los de aplicaciones como MATLAB, GNU Octave, y Scilab.

Esta librería esta organizada por subpaquetes donde cada una, está enfocado a un tema de cálculos específicos:

- Algebra lineal -> linalg
- Procesamiento de señales -> signal

- Funciones estadísticas -> stats
- Funciones especiales -> special
- Integración -> integrate
- Herramientas de interpolación -> interpolate
- Herramientas de optimización -> optimize
- Algoritmos de transformada de Fourier -> fftpack
- Entrada y salida de datos -> io
- Wrappers a la librería LAPACK -> lib.lapack
- Wrappers a la librería BLAS -> lib.blas
- Wrappers a librerías externas -> lib
- Matrices sparse -> sparse
- otras utilidades -> misc
- Vector Quantization / Kmeans -> cluster
- Ajuste a modelos con máxima entropía -> maxentropy.

2.4.3.-Matplotlib.

Matplotlib es una librería de trazado utilizada para gráficos 2D en lenguaje de programación Python, es muy flexible y tiene muchos valores predeterminados incorporados que facilitaran muchísimo el trabajo de algunos datos y con esto puedes comenzar a trazar una función.

Los datos de Machine Learning para graficar en matplotlib deberán estar estructurados bajo la librería de NumPy, por lo que es recomendable revisar esta librería, primeramente. [17]

2.4.4.- Pip.

PIP es un acrónimo que significa "Paquetes de instalación PIP" o "Programa de instalación preferida". Es una utilidad de línea de comandos que le permite instalar, reinstalar o desinstalar paquetes PyPI con un comando simple y directo: "pip". [18]

2.5.-Opencv.

Las siglas Opencv provienen de los términos anglosajones "*Open Source Computer Vision Library*". Por lo tanto, Opencv es una librería de tratamiento de imágenes, destinada principalmente a aplicaciones de visión por computador en tiempo real.

La librería tiene más de 2500 algoritmos, que incluye algoritmos de *machine learning* y de visión artificial para usar.

Estos algoritmos permiten identificar objetos, caras, clasificar acciones humanas en vídeo, hacer tracking de movimientos de objetos, extraer modelos 3D, encontrar imágenes similares, eliminar ojos rojos, seguir el movimiento de los ojos, reconocer escenarios. Se usa en aplicaciones como la detección de intrusos en vídeos, monitorización de equipamientos, ayuda a navegación de robots, inspeccionar etiquetas en productos, etc.

OpenCV está escrito en C++, tiene interfaces en C++, C, Python, Java, MATLAB y funciona en Windows, Linux, Android y Mac. [19]

Usando la biblioteca OpenCV podemos acceder a la cámara web o cualquier otro dispositivo de captura que tengamos instalado en nuestro sistema, cada una de las imágenes capturadas podrán almacenarse para su análisis o procesamiento en tiempo real si así lo deseamos, tenemos disponible una clase que nos servirá para guardar los videos previamente capturados y procesados, el formato de almacenamiento depende de las características habilitadas, pero puede ser MP4, AVI, WMV, etc., y otros si tenemos los códec.

Para acceder a la webcam lo hacemos de manera parecida a mostrar un video, solo que en lugar de especificar un archivo de video a la clase *"VideoCapture"* indicaremos el número de dispositivo o índice de cámara que deseamos usar, cero si tenemos solo una cámara.

```
from cv2 import *

namedWindow("webcam")
vc = VideoCapture(0);

while True:
    next, frame = vc.read()
    imshow("webcam", frame)
    if waitKey(50) >= 0:
        break;
```

Figura 13.- Código básico para acceder a una web cam mediante Opencv y Python. [20]

Por otro lado si se aplican varias funciones que permiten modificar la imagen original, primero *"cvtColor*", convierte la imagen a escala de grises, "*GaussianBlur"*, aplicar un filtro gaussiano, y finalmente aplicamos *Canny* el cual permite detectar los bordes de la imagen, con esto se podrá experimentar con el tratamiento de imágenes de una manera más sencilla. [20]

```
while True:
    next, frame = vc.read()

    gray = cvtColor(frame, COLOR_BGR2GRAY)
    gauss = GaussianBlur(gray, (7,7), 1.5, 1.5)
    can = Canny(gauss, 0, 30, 3)

    imshow("webcam", can)
    if waitKey(50) >= 0:
        break;
```

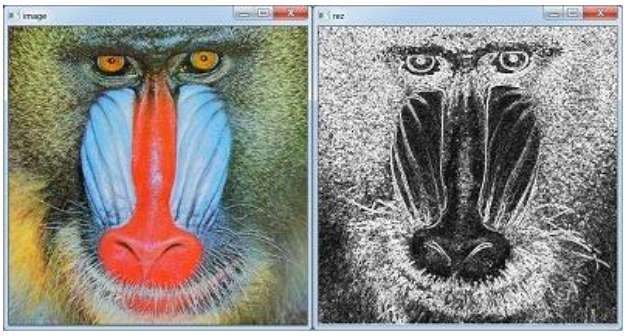

Figura 14.- Resultado de la detección de bordes de una imagen, mediante Opencv y Python, con su respectivo código. [20]

2.6.- Zbar.

Existen muchos módulos y paquetes que permiten hacer la detección de QR, sin embargo referente al tema de este documento se implementó el paquete Zbar.

Zbar.-es un paquete de software de código abierto para leer códigos de barras de varias fuentes, como transmisiones de video, archivos de imagen y sensores de intensidad sin procesar. Admite muchas simbologías populares (tipos de códigos de barras), incluidos EAN-13 / UPC-A, UPC-E, EAN-8, Código 128, Código 39, Intercalado 2 de 5 y Código QR.

La implementación flexible en capas facilita el escaneo y la decodificación de códigos de barras para cualquier aplicación: úsela de forma independiente con la GUI incluida y los programas de línea de comandos, integre fácilmente un widget de escaneo de códigos de barras en su aplicación Qt, GTK + o PyGTK GUI, aproveche una de las script o interfaces de programación (Python, Perl, C ++).Hasta una biblioteca C optimizada para uso incrustado. [21]

Para la detección de patrones se ha aplicado el método de detección de bordes mediante el algoritmo de Canny, previamente explicado, junto con los siguientes paquetes.

Math. Este módulo proporciona acceso a las funciones matemáticas definidas por el estándar C.

Imutils. Módulo que permitirá manipular el tamaño de las ventanas de las tomas de la cámara térmica y de la cámara RGB. [22]

Zbar realiza la detección de los QR de manera que primero, obtiene “frames” mediante una cámara, para posteriormente transformarla en blanco y negro de manera muy similar a los principios del algoritmo de Canny para detectar los bordes

de los QR. Pero en lugar de eso, este paquete aplica una función llamada "*zbar.Image*" la cual convierte la imagen en blanco y negro de la cámara a un formato que la librería de Zbar puede leer. Para posteriormente escanear su código y poderlo imprimir.

Nombre	Función.
Cv2.VideoCapture	Selecciona una cámara web conectada.
Capture.Read	Función para capturar frames.
Zbar.Image	Función para convertir una imagen obtenida de la cámara a una que Zbar pueda entender.
Zbar.ImageScanner	Activa el escáner de Zbar.
Codigo_qr.data	Guarda el mensaje que contiene el código QR.
cv.imshow	Muestra en pantalla la cámara con el escáner.

Tabla 2.- Instrucciones más relevantes para el algoritmo de la detección de QR.

A continuación se presenta un ejemplo de esto, con un código en Python, opencv y Zbar para detectar códigos QR.

```
import zbar
import numpy as np
import cv2

#Inicializar la camara
capture = cv2.VideoCapture(0)

while 1:
    #Capturar un frame
    val, frame = capture.read()

    if val:
        #Capturar un frame con la camara y guardar sus dimensiones
        frame_gris = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        dimensiones = frame_gris.shape #'dimensiones' sera un array que contendra el alto,  el ancho y los canales de la imagen en
este orden.

        #Convertir la imagen de OpenCV a una imagen que la libreria ZBAR pueda entender
        imagen_zbar = zbar.Image(dimensiones[1], dimensiones[0], 'Y800', frame_gris.tobytes())

        #Construir un objeto de tipo scaner, que permitira escanear la imagen en busca de codigos QR
        escaner = zbar.ImageScanner()

        #Escanear la imagen y guardar todos los codigos QR que se encuentren
        escaner.scan(imagen_zbar)

        for codigo_qr in imagen_zbar:
            dat = codigo_qr.data[:-2] #Guardar el mensaje del código QR. Los últimos dos caracteres son saltos de línea que hay que
eliminar
            print(dat)

        #Mostrar la imagen
        cv2.imshow('Imagen', frame)

    #Salir con 'ESC'
    k = cv2.waitKey(5) & 0xFF
    if k == 27:
        break



cv2.destroyAllWindows()
```

Figura 15.- Algoritmo para la detección de códigos QR, en Python, Opencv y Zbar [23].

CAPITULO 3. HARDWARE Y SU CONFIGURACIÓN.

En el capítulo anterior se vieron todos los aspectos referentes a las teorías de los algoritmos de detección a utilizar y el software donde se implementaran esas funciones. A continuación se mostrarán los componentes electrónicos, cámaras y sensores para la visión térmica. Además de su configuración y relación con el software previamente presentado en el anterior capitulo.

3.1.- Microcontroladores.

Más adelante se analizarán los factores del porqué se utilizó una raspberry pi 3, la cual es un ordenador con un microprocesador. Es importante analizar las diferencias entre un microcontrolador y un microprocesador, y de ahí por qué el microprocesador es la mejor opción para el procesamiento de imágenes.

Un microcontrolador es un dispositivo electrónico capaz de llevar a cabo procesos lógicos. Estos procesos o acciones son programados en lenguaje ensamblador por el usuario, y son introducidos en este a través de un programador.

Un microcontrolador es un solo circuito integrado que contiene todos los elementos electrónicos que se utilizaban para hacer funcionar un sistema basado con un microprocesador; es decir contiene en un solo integrado la Unidad de Proceso, la memoria RAM, memoria ROM , puertos de entrada, salidas y otros periféricos, con la consiguiente reducción de espacio.

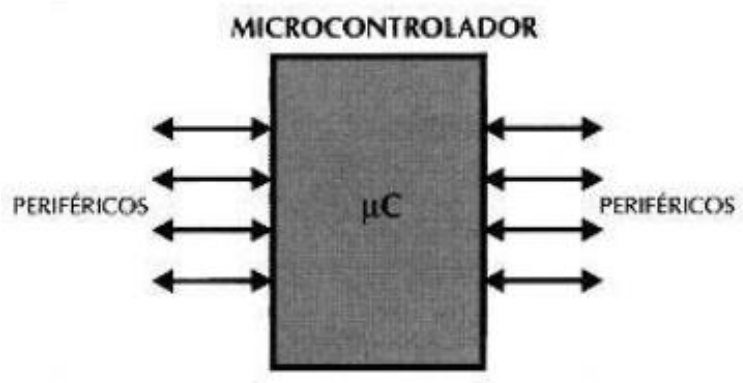

Figura 16.- Estructura general de un microcontrolador [24].

3.1.2.- Estructura de un microcontrolador.

El procesador.

Es el elemento más importante del microcontrolador y determina sus principales características, tanto a nivel hardware como software. Se encarga de direccionar la memoria de instrucciones, recibir el código OP de la instrucción en curso, su decodificación y la ejecución de la operación que implica la instrucción, así como la búsqueda de los operandos y el almacenamiento del resultado. Existen tres orientaciones en cuanto a la arquitectura y funcionalidad de los procesadores actuales.

Memoria.

En los microcontroladores la memoria de instrucciones y datos está integrada en el propio chip. Una parte debe ser no volátil, tipo ROM, y se destina a contener el programa de instrucciones que gobierna la aplicación. Otra parte de memoria será tipo RAM, volátil, y se destina a guardar las variables y los datos.

Hay dos peculiaridades que diferencian a los microcontroladores de los computadores personales:

No existen sistemas de almacenamiento masivo como disco duro o disquetes. Como el microcontrolador solo se destina a una tarea en la memoria ROM, solo hay que almacenar un único programa de trabajo.

La RAM en estos dispositivos es de poca capacidad pues solo debe contener las variables y los cambios de información que se produzcan en el transcurso del programa. Por otra parte, como solo existe un programa activo, no se requiere guardar una copia del mismo en la RAM pues se ejecuta directamente desde la ROM.

Los usuarios de computadoras personales están habituados a manejar Megabytes de memoria, pero, los diseñadores con microcontroladores trabajan con

capacidades de ROM comprendidas entre 512 bytes y 8 kilobytes y de RAM comprendidas entre 20 y 512 bytes.

Según el tipo de memoria ROM que dispongan los microcontroladores, la aplicación y utilización de los mismos es diferente. Se describen las cinco versiones de memoria no volátil que se pueden encontrar en los microcontroladores del mercado.

Memoria FLASH.

Se trata de una memoria no volátil, de bajo consumo, que se puede escribir y borrar. Funciona como una ROM y una RAM pero consume menos y es más pequeña. A diferencia de la ROM, la memoria FLASH es programable en el circuito. Es más rápida y de mayor densidad que la EEPROM. La alternativa FLASH está recomendada frente a la EEPROM cuando se precisa gran cantidad de memoria de programa no volátil. Es más veloz y tolera más ciclos de escritura/borrado. Las memorias EEPROM y FLASH son muy útiles al permitir que los microcontroladores que las incorporan puedan ser reprogramados en circuito, es decir, sin tener que sacar el circuito integrado de la tarjeta. Así, un dispositivo con este tipo de memoria incorporado al control del motor de un automóvil permite que pueda modificarse el programa durante la rutina de mantenimiento periódico, compensando los desgastes y otros factores tales como la compresión, la instalación de nuevas piezas, etc. La reprogramación del microcontrolador puede convertirse en una labor rutinaria dentro de la puesta a punto.

Puertas de Entrada y Salida.

Las puertas de Entrada y Salida (E/S) permiten comunicar al procesador con el mundo exterior, a través de interfaces, o con otros dispositivos. Estas puertas, también llamadas puertos, son la principal utilidad de las patas o pines de un microprocesador. Según los controladores de periféricos que posea cada modelo de microcontrolador, las líneas de E/S se destinan a proporcionar el soporte a las señales de entrada, salida y control.

Reloj principal.

Todos los microcontroladores disponen de un circuito oscilador que genera una onda cuadrada de alta frecuencia, que configura los impulsos de reloj usados en la sincronización de todas las operaciones del sistema. Esta señal del reloj es el motor del sistema y la que hace que el programa y los contadores avancen.

Generalmente, el circuito de reloj está incorporado en el microcontrolador y solo se necesitan unos pocos componentes exteriores para seleccionar y estabilizar la frecuencia de trabajo. Dichos componentes suelen consistir en un cristal de cuarzo junto a elementos pasivos o bien un resonador cerámico o una red R-C.

Aumentar la frecuencia de reloj supone disminuir el tiempo en que se ejecutan las instrucciones pero lleva aparejado un incremento del consumo de energía y de calor generado. [24]

3.1.3.- Arduino.

Arduino es una plataforma de prototipos electrónica de código abierto (open-source) basada en hardware y software flexibles y fáciles de usar. Está pensado para artistas, diseñadores, como hobby y para cualquiera interesado en crear objetos o entornos interactivos.

Arduino puede “sentir” el entorno mediante la recepción de entradas desde una variedad de sensores y puede afectar a su alrededor mediante el control de luces, motores y otros artefactos. El microcontrolador de la placa se programa usando el “*Arduino Programming Language*” (basado en Wiring1) y el “*Arduino Development Environment*” (basado en Processing2). Los proyectos de Arduino pueden ser autonomos o se pueden comunicar con software en ejecución en un ordenador (por ejemplo con Flash, Processing, MaxMSP, etc.).

Las placas se pueden ensamblar a mano o encargarlas pre ensambladas; el software se puede descargar gratuitamente. Los diseños de referencia del hardware (archivos CAD) están disponibles bajo licencia open-source, por lo que es libre de

adaptarlas a cualquier necesidad. Por lo tanto presenta ciertas ventajas como uno de los mejores microcontroladores para iniciar en el mundo de la automatización o programación.

- ***Barato:*** Las placas Arduino son relativamente baratas comparadas con otras plataformas microcontroladoras. La versión menos cara del módulo Arduino puede ser ensamblada a mano.
- ***Multiplataforma***: El software de Arduino se ejecuta en sistemas operativos Windows, Macintosh OSX y GNU/Linux. La mayoría de los sistemas microcontroladores están limitados a Windows.
- ***Entorno de programación simple y claro***: El entorno de programación de Arduino es fácil de usar para principiantes, pero su eficientemente flexible para que usuarios avanzados puedan aprovecharlo también. Para profesores, está convenientemente basado en el entorno de programación Processing, de manera que estudiantes aprendiendo a programar en ese entorno estarán familiarizados con el aspecto y la imagen de Arduino.
- ***Código abierto y software extensible:*** El software Arduino está publicado como herramientas de código abierto, disponible para extensión por programadores experimentados. El lenguaje puede ser expandido mediante librerías C++, y la gente que quiera entender los detalles técnicos pueden hacer el salto desde Arduino a la programación en lenguaje AVR C en el cual está basado. De forma similar, puedes añadir código AVR-C directamente en tus programas Arduino si quieres.
- ***Código abierto y hardware extensible***: El Arduino está basado en microcontroladores ATMEGA8 y ATMEGA168 de Atmel. Los planos para los módulos están publicados bajo licencia *Creative Commons*, por lo que diseñadores experimentados de circuitos pueden hacer su propia versión del módulo, extendiéndolo y mejorándolo. Incluso usuarios relativamente inexpertos pueden construir la versión de la placa del módulo para entender cómo funciona y ahorrar dinero.

Característica	Descripción
Microcontrolador	ATmega168
Voltaje de operación	5 V
Tensión de entrada (recomendada)	7 - 12 V
Tensión de entrada (límite)	6 - 20 V
Pines digitales de E/S	14 (de los cuales 6 proveen salidas PWM)
Pines de entrada analógicos	6
Corriente DC por pin E/S	40 mA
Corriente DC para pin 3.3 V	50 mA
Memoria Flash	16 KB (de los cuales 2 KB usados para bootloader)
SRAM	1 KB
EEPROM	512 bytes
Frecuencia de reloj	16 MHz

Tabla 3.- Características técnicas del arduino. Modelo (UNO).

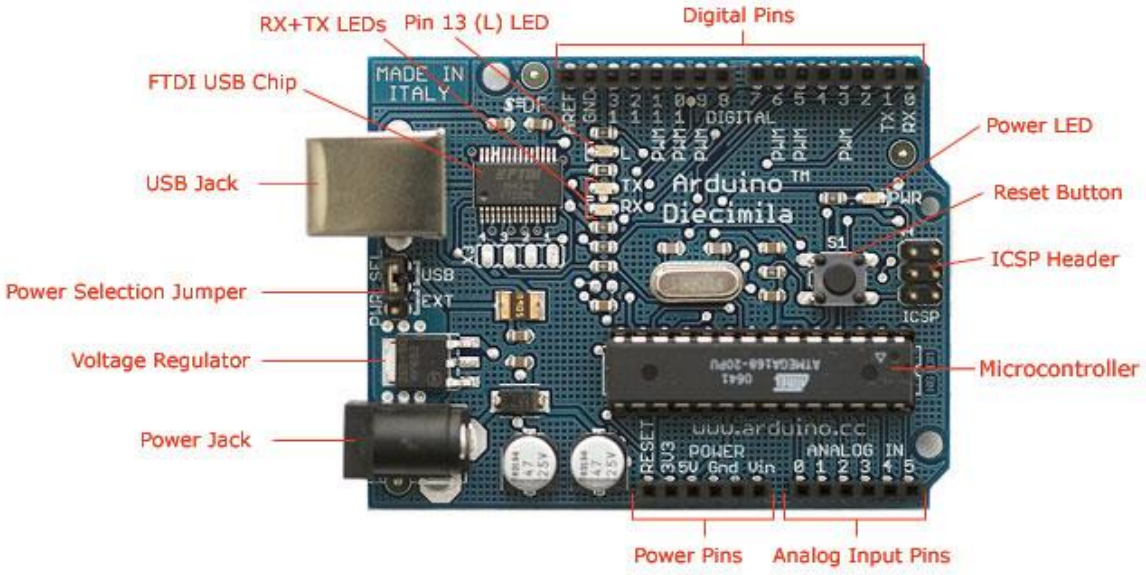

Figura 17.- Partes del microcontrolador Arduino modelo UNO.

3.2.-Microprocesador.

Arduino al ser un excelente ejemplo de un microcontrolador de fácil uso y especificaciones excepcionales para el trabajo, su nivel de procesamiento no es el adecuado para efectuar el procesamiento de imágenes, por lo cual se requerirá de algo aún más potente, los microprocesadores. También conocidos como CPU o unidad central de procesamiento, el microprocesador es un motor de cálculo completo que se fabrica en un solo chip de silicio. También se conoce como el corazón de cualquier ordenador normal, ya sea una máquina de escritorio, un servidor o un ordenador portátil.

Un microprocesador es un componente que realiza las instrucciones y tareas involucradas en el procesamiento informático y es el “motor” que se pone en marcha cuando enciendes el equipo. En un sistema informático, el microprocesador es la unidad central que ejecuta y gestiona las instrucciones lógicas que se le transmiten.

Un microprocesador está diseñado para ejecutar operaciones típicas tales como adición, sustracción, división, multiplicación, comunicación de dispositivos e interprocesos, administración de entradas, salidas, y más.

Está compuesto por circuitos integrados que contienen miles de transistores, dependiendo de la potencia del equipo.

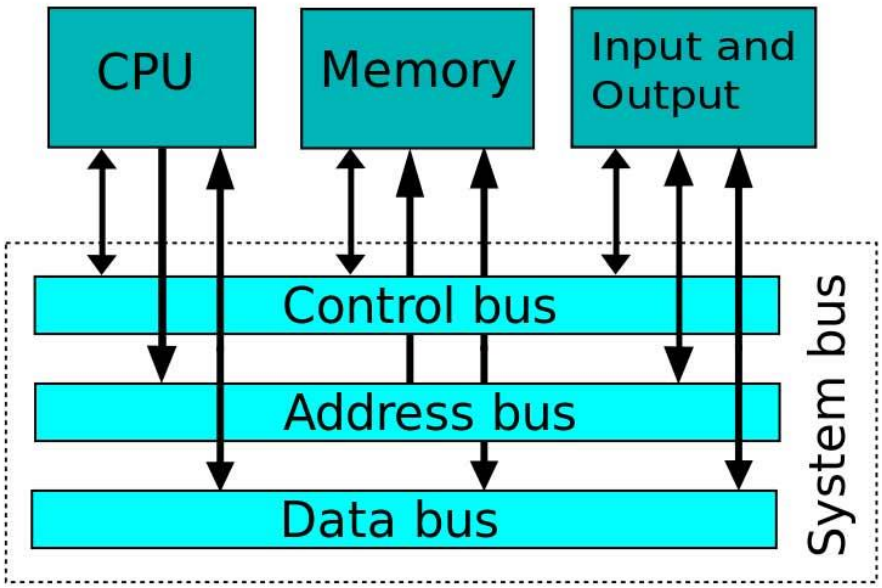

Figura 18.- Esquema de los componentes de un microprocesador.

Un procesador es el cerebro de un ordenador que consiste básicamente en una unidad aritmética y lógica (ALU), una unidad de control y una matriz de registro.

Como su nombre indica, ALU realiza todas las operaciones aritméticas y lógicas sobre los datos recibidos de los dispositivos de entrada o memoria.

Por lo tanto presentan también ventajas muy similares a los de un microcontrolador.

- ***Bajo coste:*** los microprocesadores están disponibles a bajo costo gracias a la tecnología de circuitos integrados. Lo que reduce el coste de un sistema informático.
- ***Alta velocidad:*** los chips de microprocesador pueden trabajar a muy alta velocidad gracias a la tecnología que se utiliza en ellos. Es capaz de ejecutar millones de instrucciones por segundo.

- ***Tamaño pequeño:*** debido a la tecnología de integración a gran escala y ultra gran escala, un microprocesador se fabrica con un tamaño de superficie muy reducido. Esto reducirá el tamaño de todo el sistema informático.
- ***Versátil:*** los microprocesadores son muy versátiles, el mismo chip se puede utilizar para una serie de aplicaciones simplemente cambiando el programa (instrucciones almacenadas en la memoria).
- ***Bajo consumo de energía:*** los microprocesadores se fabrican generalmente utilizando la tecnología de semiconductores complementario de óxido metálico (CMOS), en la que los MOSFETs (transistores de efecto de campo de metal-óxido-semiconductor) trabajan en los modos de saturación y corte. Por lo tanto, el consumo de energía es muy bajo.
- ***Menos generación de calor:*** comparados con los dispositivos de tubos de vacío (válvula termo-iónica), los dispositivos de semiconductores no emiten tanto calor.
- ***Fiable***: los microprocesadores son muy fiables, y la tasa de fallos es muy inferior a medida que se utiliza la tecnología de semiconductores.
- ***Portátil:*** los dispositivos o sistemas informáticos fabricados con microprocesadores pueden hacerse portátiles debido al reducido tamaño y bajo consumo de energía.

3.2.1.- Raspberry pi.

La Raspberry Pi es un ejemplo de una mini computadora con un nivel de microprocesador decente para efectuar el procesamiento de imágenes, esta es un ordenador de bajo costo y tamaño reducido, al cual se le puede conectar un televisor y un teclado para interactuar con ella exactamente igual que cualquier otra computadora.

Su origen remonta en Reino Unido, allí es donde nació como una organización caritativa la Fundación Raspberry Pi en 2009. Con el objetivo de animar a los niños a aprender informática en las escuelas. La Raspberry Pi se puede usar en proyectos de electrónica y para tareas básicas que haría cualquier ordenador de sobremesa

como navegar por internet, hojas de cálculo, procesador de textos, reproducir vídeo en alta definición e incluso jugar a ciertos juegos.

La Raspberry Pi es la placa de un ordenador simple compuesto por un CPU, memoria RAM, puertos de entrada y salida de audio y vídeo, conectividad de red, ranura SD para almacenamiento, reloj, una toma para la alimentación, conexiones para periféricos, etc. Además la cantidad de usos que se le pueden dar a esta placa en la actualidad es impresionante, solo por mencionar varios tenemos:

- Como media center, o lo que es lo mismo, para convertir una televisión en una smart TV, con software LIBRELEC u OSMC.
- Para emular una videoconsola retro jugando a grandes clásicos con RetroPie instalado.
- Como ordenador con sistema Linux, a través de distribuciones como Ubuntu, Raspbian (Debian) o Pidora (Fedora).
- Domótica, con Windows 10 IOT Core, lo que permite hacer de nuestra casa un espacio un poco más inteligente con proyectos como estaciones meteorológicas o hubs inteligentes.

Figura 19.- Raspberry pi, junto con los periféricos mínimos para su funcionamiento [27].

La Raspberry más actual es el modelo 4 hasta la fecha de este documento, pero aún no se ha comercializado tanto como sus predecesores. Las diferencias no es mucha, pero si es importante ver la capacidad de procesamiento del microprocesador para efectuar el procesamiento de imágenes. [27]

	SOC	FRECUENCIA DE RELOJ	RAM	PUERTOS USB	ETHERNET	WIRELESS/ BLUETOOTH
Raspberry Pi Modelo A+	BCM2835	700Mhz	512MB	1	No	No
Raspberry Pi Modelo B+	BCM2835	700Mhz	512MB	4	Si	No
Raspberry Pi 2 Modelo B	BCM2836 o BCM2837	900Mhz	1 GB	4	Si	No
Raspberry Pi 3 Modelo B	BCM2837	1200Mhz	1 GB	4	Si	si
Raspberry Pi 3 Modelo B+	BCM2837B0	1500Mhz	1 GB	4	Gigabit Ethernet over USB 2.0	Si
Raspberry Pi Zero	BCM2835	1000Mhz	512MB	1	No	No
Raspberry Pi Zero W	BCM2835	1000Mhz	512MB	1	No	Si

Tabla 4.- Comparativa entre varios modelos del rasperry pi. [27]

Es importante mencionar que para efectuar las conexiones de varios componentes, la raspberry pi a diferencia de arduino que maneja entradas y salidas analógicas por separado. La raspberry pi las tiene a ambas en puertos llamados “GPIO”, la cual se refiere a “General Inputs and Outputs Purpose”. Es decir que uno como usuario en el código decide si quiere su pin como entrada o como salida.

Estos pines son la conexión entre el mundo virtual del código informático y el mundo real. Con los GPIO, puedes conectar componentes del circuito a tu Raspberry Pi.

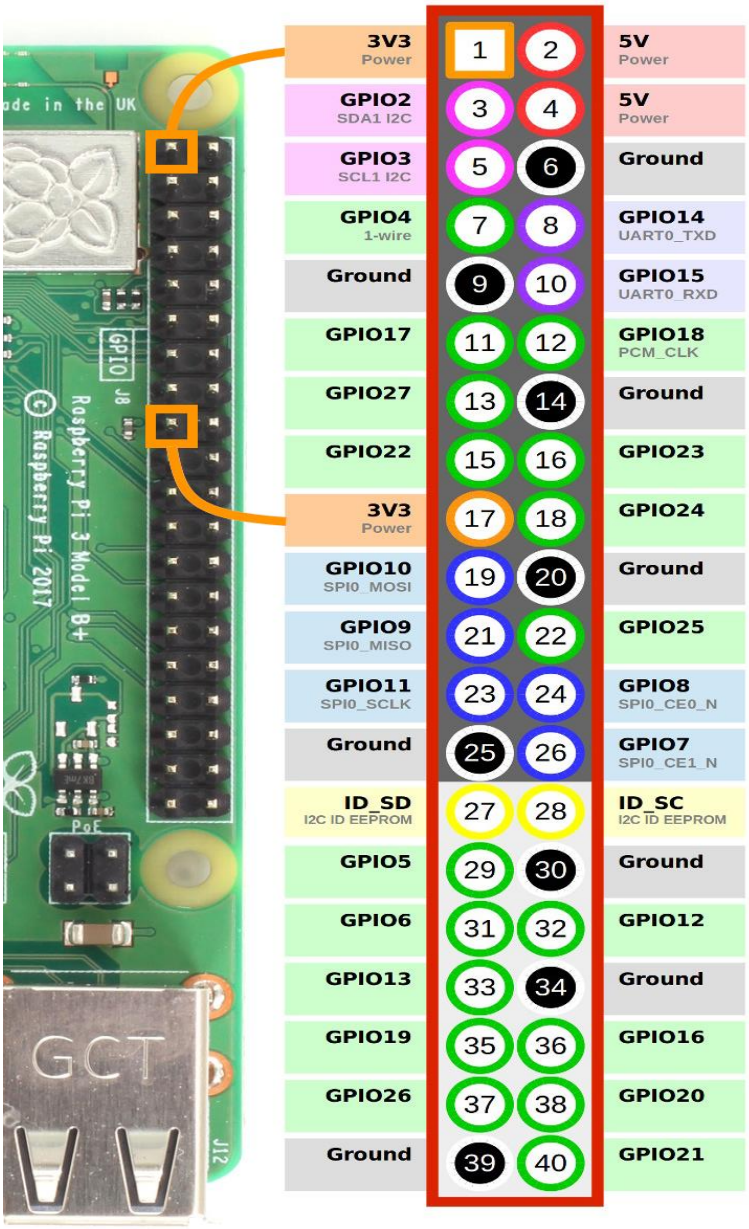

Figura 20.- Puertos GPIO de la raspberry pi. [27]

3.2.2.- Jetson Nano.

Raspberry pi, no es la única computadora a considerar en el mercado. Existe actualmente mucha competencia con características muy similares a la raspberry, aunque un punto muy importante que hay que reconocer de la raspberry, es que es mucha más comercial por lo cual, es más fácil de usar que muchas de las tarjetas de su competencia, hasta el momento.

Los recientes avances en inteligencia artificial han dado como resultado una serie de algoritmos utilizados para aplicaciones tales como reconocimiento facial, reconocimiento de voz, traducción instantánea y más.

Y es que durante la *GPU Technology Conference*, un evento anual organizado por Nvidia fue presentada la Jetson Nano la cual es una computadora de una sola tarjeta en forma de módulo de 70 x 45 mm que tendrá un costo de alrededor de unos $ 99 USD.

Esta nueva computadora de una sola tarjeta la Jetson Nano está pensada para ser implementada en robots y otros dispositivos impulsados por AI.

Las especificaciones de esta tarjeta son:

- GPU Maxwell de 128 núcleos
- Procesador Quad-core Arm A57 a 1,43 GHz
- Memoria del sistema – 4GB 64-bit LPDDR4 @ 25.6 GB / s
- Almacenamiento: ranura para tarjeta microSD (devkit) o flash de 16 GB eMMC (producción)
- Codificación de video – 4K @ 30 | 4x 1080p @ 30 | 9x 720p @ 30 (H.264 / H.265)
- Decodificación de video – 4K @ 60 | 2x 4K @ 30 | 8x 1080p @ 30 | 18x 720p @ 30 (H.264 / H.265)
- Dimensiones – 70 x 45 mm.
- Conector SO-DIMM de 260 pines para el módulo Jetson Nano.
- Salida de video – HDMI 2.0 y eDP 1.4 (solo video)

- Conectividad – Gigabit Ethernet (RJ45) + encabezado PoE de 4 pines
- USB: 4x puertos USB 3.0, 1x puerto USB 2.0 Micro-B para modo de alimentación o dispositivo
- Cámara I / F – 1x carriles MIPI CSI-2 DPHY compatibles con Leopard Imaging LI-IMX219-MIPI-FF-NANO y el módulo de cámara Raspberry Pi V2
- Expansión
- M.2 Socket Key E (PCIe x1, USB 2.0, UART, I2S e I2C) para tarjetas de red inalámbrica
- Cabezal de expansión de 40 pines con señales GPIO, I2C, I2S, SPI, UART
- Cabezal de botón de 8 pines con señales relacionadas con la alimentación del sistema, el restablecimiento y la recuperación forzada
- Misc – Power LED, cabezal de ventilador de 4 pines
- Fuente de alimentación: 5V / 4A a través del conector tipo barril de potencia o 5V / 2A a través del puerto micro USB; soporte opcional de PoE
- Dimensiones – 100 x 80 x 29 mm.

Puede ejecutar Ubuntu y otros sistemas operativos Linux el Jetson Nano. En total, los paquetes de Nvidia Jetson en un impresionante 472 GFLOPS de rendimiento informático. Además, hay soporte para sensores de alta resolución. El Jetson Nano puede ejecutar toneladas de sensores en paralelo y maneja varias redes neuronales por sensor.

Nvidia enfatiza que este último es compatible con una gama de marcos de inteligencia artificial dedicados en los que encontramos TensorFlow, PyTorch, Caffe, Keras y MXNet.

Figura 21.- Nvidia Jetson Nano. [28]

Al igual que raspberry, Jetson nano también cuenta con puertos GPIO.

Pi GPIO#	Sysfs GPIO	Name	Pin	Pin	Name	Sysfs GPIO	Pi GPIO#
		3.3VDC *Power*	1	2	5.0VDC *Power*		
2		SDA1 *I2C Bus 1*	3	4	5.0VDC *Power*		
3		SCL1 *I2C Bus 1*	5	6	GND		
4	gpio216	AUDIO_MCLK	7	8	TXD0		14
		GND	9	10	RXD0		15
17	gpio50	UART2_RTS	11	12	DAP4_SCLK	gpio79	18
27	gpio14	SPI2_SCK	13	14	GND		
22	gpio194	LCD_TE	15	16	SPI2_CS1	gpio232	23
		3.3VDC *Power*	17	18	SPI2_CS0	gpio15	24
10	gpio16	SPI_MOSI	19	20	GND		
9	gpio17	SPI_MISO	21	22	SPI2_MISO	gpio13	25
11	gpio18	SPI1_SCK	23	24	SPI1_CS0	gpio19	8
		GND	25	26	SPI1_CS1	gpio20	7
(0)		ID_SDA *I2C Bus 0*	27	28	ID_SCL *I2C Bus 0*		(1)
5	gpio149	CAM_AF_EN	29	30	GND		
6	gpio200	GPIO_PZ0	31	32	LCD_BL_PWM	gpio168	12
13	gpio38	GPIO_PE6	33	34	GND		
19	gpio76	DAP4_FS	35	36	UART2_CTS	gpio51	16
26	gpio12	SPI2_MOSI	37	38	DAP4_DIN	gpio77	20
		GND	39	40	DAP4_DOUT	gpio78	21

Tabla 5.- Distribución de los puertos GPIO´s de la Jetson nano.

En la superficie, las dos CPU parecen casi idénticas. De hecho son muy similares. El Cortex-A72 en la Raspberry Pi 4 es solo una generación más nueva que el Cortex-A57 en el NVIDIA Jetson Nano. El nuevo A72 tiene una velocidad de reloj ligeramente más alta, y fue diseñado para usar un 20% menos de potencia u ofrecer un 90% más de rendimiento que el A57.

En cuanto al rendimiento, vale la pena señalar que Raspberry Pi afirma que el Raspberry Pi 4 Cortex-A72 es aproximadamente un 50% más rápido que el procesador Raspberry Pi 3 B + Cortex-A53. Dado que el NVIDIA Jetson Nano usa un Cortex-A57 que se encuentra en el medio entre las dos generaciones de CPU Raspberry Pi, el rendimiento diferente entre el Raspberry Pi 4 y el NVIDIA Jetson Nano puede no ser tan significativo.

La nueva CPU Cortex-A72 en la Raspberry Pi 4 parece indicar más potencia de procesamiento de la CPU, en comparación con la Cortex-A57 en el NVIDIA Jetson Nano. Esto puede significar que el Raspberry Pi 4 es mejor para usarlo como una computadora de propósito general o como un reemplazo de mini-escritorio de bajo costo. Sin embargo, la GPU más potente del NVIDIA Jetson Nano le brinda capacidades, mucho mayores para el trabajo de gráficos e incluso para los usos de Inteligencia Artificial (AI) y Aprendizaje automático (ML). [29]

En conclusión, la raspberry pi, sirve más para propósitos generales y la Jetson nano sirve más para enfoques de inteligencia artificial. Para este proyecto se consideraron ambas al principio de las pruebas, sin embargo raspberry pi al ser más comercial y tener las librerías de visión con un acceso más fácil, se decidió usarla como parte del prototipo final.

	Cortex-A72	Cortex-A57
Junta utilizada	Raspberry Pi 4	NVIDIA Jetson Nano
Microarquitectura	ARMv8-A de 64 bits	ARMv8-A de 64 bits
Núcleos	4 4	4 4
Velocidad de reloj	1.5 Ghz	1,42 Ghz
Proceso de manufactura	28 nm	20 nm?
Caché L1	80 KiB (48 KiB I-cache con paridad, 32 KiB D-cache con ECC) *por núcleo*	80 KiB (48 KiB I-cache con paridad, 32 KiB D-cache con ECC) *por núcleo*
Caché L2	512 KiB a 4 MiB	512 KiB a 2 MiB
Caché L3	Ninguna	Ninguna
Año de lanzamiento de la CPU	2015	2013

Tabla 6.- Comparación entre procesadores de raspberry pi 4 (última versión) y Jetson nano.

3.3.- AMG8833.

Raspberry pi tiene un subconjunto de librerías para una cantidad considerable de sensores, por eso de momento se eligió al final ante la Jetson nano.

Uno de los objetivos principales de este proyecto es que se tenga implementado un apartado para la visión térmica.

La luz blanca (como la que proviene del sol) es una mezcla de luces de diferentes colores que interacciona con los objetos en función de la composición de éstos. Simplificándolo, se puede decir que los objetos absorben cierta parte de la luz y reflejan otra parte. El color que nosotros percibimos no es más que la reacción que se produce en el cerebro a la luz que llega a nuestros ojos reflejada por estos objetos. Por poner un ejemplo, las hojas de las plantas son verdes porque contienen clorofila, un pigmento que absorbe la luz azul y roja mientras que refleja la verde. Las cosas negras lo son porque absorben la luz de diferentes colores y las blancas porque reflejan toda la luz visible. Por lo tanto, la luz que nosotros podemos ver solo es una pequeñísima parte de una radiación conocida como radiación electromagnética.

Los seres humanos solo podemos ver la radiación con una longitud de onda entre los 400 y los 700 nm (esta unidad es la millonésima parte de un milímetro). El resto de la radiación, que ocupa la mayor parte del espectro, es invisible a nuestro ojo, bien porque tiene una longitud de onda mayor como las ondas de radio o porque tienen una longitud de onda menor, como los rayos gamma.

Ahora bien, que nosotros no podamos ver esa luz no quiere decir que los objetos no interactúen con ella. De hecho los objetos o, mejor dicho, las moléculas que forman estos objetos también interactúan con la luz que es ligeramente menos energética que la luz visible: la luz infrarroja.

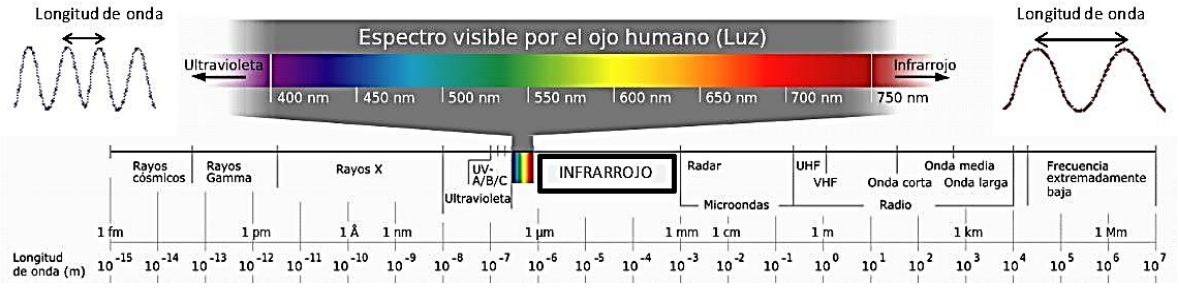

Figura 22.- Espectro electromagnético donde se detalla la composición de la luz visible.

La cámara infrarroja es una tecnología que nos permite observar y obtener imágenes, a partir de las emisiones de infrarrojos que emite un cuerpo. Estando este situado dentro de un espectro electromagnético.

Las cámaras de visión térmica funcionan captando la radiación infrarroja de un cuerpo u objeto. Acto seguido convierte esa emisión de energía calorífica en una imagen visible para el ojo humano. Los rayos infrarrojos se sitúan después del extremo rojo del espectro de colores visibles.

La conocida como “región infrarroja”, situada entre la radiación de microondas del espectro electromagnético y la radiación visible recibe el calor que irradian los cuerpos. Esta energía es utilizada para crear imágenes visibles al ojo humano.

Todos los cuerpos y objetos desprenden e irradian energía. Independientemente, de si están a temperatura ambiente o, incluso, fría.

Así, un objeto congelado, o a temperatura ambiente, también sería susceptible de ser percibido por una cámara de visión térmica.

Las ventajas de una cámara térmica son:

- La búsqueda de diagnósticos acertados y concluyentes, ahorra costes económicos. Adelantarse a un fallo en la aplicación industrial puede ser vital para la supervivencia de una industria. Así como para su desarrollo tecnológico.
- Evitar males mucho mayores, como incendios, es otro de los principales motivos por el cual instalar una cámara de visión térmica. El control exhaustivo de las aplicaciones industriales es útil para adelantarnos a las tareas de mantenimiento, así como a posibles accidentes.
- Aumentar la rapidez de la producción y que ésta sea eficiente, reduciendo los costes, es necesario para una industria que pretende ser competitiva. Al reconocer y detectar los fallos, podremos poner en funcionamiento estrategias de desarrollo.
- Mantenerse a la vanguardia de la tecnología nos permitirá adelantarnos a la competencia. Las cámaras de visión térmica nos ayudan a ver aquello que

no ven nuestros ojos. Cuando analizamos e interpretamos la radiación de los objetos reconocemos problemas hasta entonces invisibles. [30]

Por lo tato con la idea ya planteada de lo que es visión térmica. Una cámara profesional en relación a sus costos, puede salir a un precio considerable. Por lo cual en esta apartado se plantea el uso de una cámara de baja resolución pero que sirve para realizar las pruebas de detección. En este caso se implementará el módulo de la marca Adafruit Amg8833.

Esta cámara térmica de Adafruit basada en el sensor de Panasonic AMG8833 es un array de sensores térmicos de 8x8 que permite medir temperaturas entre 0ºC y 80ºC y conectado por I2C a un microcontrolador o similar, mostrar la huella térmica captada mediante interpolación.

Puede detectar a una persona a una distancia de hasta 7 metros y tiene una tasa de refresco de 10Hz, es compatible con Arduino y con Raspberry Pi. Cuenta con un pin que puede actuar como interruptor cuando un pixel se sale de una escala de valores previamente definida.

Puede funcionar tanto a 3V como a 5V pues cuenta con un regulador de voltaje.

Figura 23.- Cámara térmica AMG8833.

3.3.1.-Interfaz I2C.

Para la comunicación de la cámara térmica se empleará la comunicación I2C con las raspberry.

Esta interfaz es un tipo de bus diseñado por *Philips Semiconductors* a principios de los 80s, que se utiliza para conectar circuitos integrados (ICs). El I2C es un bus con múltiples maestros, lo que significa que se pueden conectar varios chips al mismo bus y que todos ellos pueden actuar como maestro, solo con iniciar la transferencia de datos. Este bus se utiliza dentro de una misma placa de un dispositivo.

El bus I2C, un estándar que facilita la comunicación entre microcontroladores, memorias y otros dispositivos con cierto nivel de "inteligencia", solo requiere de dos líneas de señal y un común o masa.

Las señales principales se describen en:

- SCL (System Clock) es la línea de los pulsos de reloj que sincronizan el sistema.
- SDA (System Data) es la línea por la que se mueven los datos entre los dispositivos.
- GND (Masa) común de la interconexión entre todos los dispositivos "enganchados" al bus.

Las líneas SDA y SCL son del tipo drenaje abierto, es decir, un estado similar al de colector abierto, pero asociadas a un transistor de efecto de campo (o FET). Se deben polarizar en estado alto (conectando a la alimentación por medio de resistores "pull-up") lo que define una estructura de bus que permite conectar en paralelo múltiples entradas y salidas.

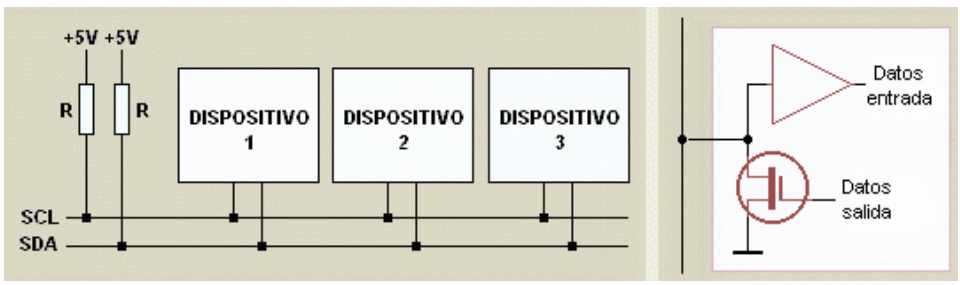

Figura 24.- Esquema de representación del bus I2C. [31]

Habiendo varios dispositivos conectados sobre el bus, es lógico que para establecer una comunicación a través de él se deba respetar un protocolo. Digamos, en primer lugar, lo más importante: existen dispositivos maestros y dispositivos esclavos. Solo los dispositivos maestros pueden iniciar una comunicación.

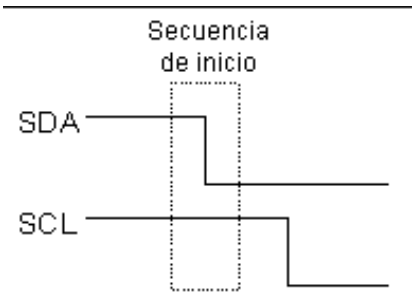

Figura 25.- Comparación de señales SDA y SCL de tipo escalón. [31]

La condición inicial, de bus libre, es cuando ambas señales están en estado lógico alto. En este estado cualquier dispositivo maestro puede ocuparlo, estableciendo la condición de inicio (start). Esta condición se presenta cuando un dispositivo maestro pone en estado bajo la línea de datos (SDA), pero dejando en alto la línea de reloj (SCL).

El primer byte que se transmite luego de la condición de inicio contiene siete bits que componen la dirección del dispositivo que se desea seleccionar, y un octavo bit que corresponde a la operación que se quiere realizar con él (lectura o escritura).

Si el dispositivo cuya dirección corresponde a la que se indica en los siete bits (A0-A6) está presente en el bus, este contesta con un bit en bajo, ubicado inmediatamente luego del octavo bit que ha enviado el dispositivo maestro. Este bit de reconocimiento (ACK) en bajo le indica al dispositivo maestro que el esclavo reconoce la solicitud y está en condiciones de comunicarse. Aquí la comunicación se establece en firme y comienza el intercambio de información entre los dispositivos.

Los términos empleados en resumen fueron los siguientes:

- Maestro (Master): Dispositivo que determina los tiempos y la dirección del tráfico en el bus. Es el único que aplica los pulsos de reloj en la línea SCL. Cuando se conectan varios dispositivos maestros a un mismo bus la configuración obtenida se denomina “multi-maestro”.
- Esclavo (Slave): Todo dispositivo conectado al bus que no tiene la capacidad de generar pulsos de reloj. Los dispositivos esclavos reciben señales de comando y de reloj generados desde el maestro.
- Bus libre (Bus Free): Estado en el que ambas líneas (SDA y SCL) están inactivas, presentando un estado lógico alto. Es el único momento en que un dispositivo maestro puede comenzar a hacer uso del bus.
- Comienzo (Start): Se produce cuando un dispositivo maestro ocupa el bus, generando la condición. La línea de datos (SDA) toma un estado bajo mientras que la línea de reloj (SCL) permanece alta.
- Parada (Stop): Un dispositivo maestro puede generar esta condición, dejando libre el bus. La línea de datos y la de reloj toman un estado lógico alto.
- Dato válido (Valid Data): Situación presente cuando un dato presente en la línea SDA es estable al tiempo que la línea SCL está a nivel lógico alto.
- Formato de Datos (Data Format): La transmisión de un dato a través de este bus consiste de 8 bits de dato (1 byte). A cada byte transmitido al bus le sigue un noveno pulso de reloj durante el cual el dispositivo receptor del byte debe generar un pulso de reconocimiento.
- Reconocimiento (Acknowledge): El pulso de reconocimiento, conocido como ACK (del inglés Acknowledge), se logra colocando la línea de datos a un nivel lógico bajo durante el transcurso del noveno pulso de reloj.
- Dirección (Address): Todo dispositivo diseñado para funcionar en este bus posee su propia y única dirección de acceso, preestablecida por el fabricante. Hay dispositivos que permiten definir externamente parte de la dirección de acceso, lo que habilita que se pueda conectar en un mismo bus un conjunto de dispositivos del mismo tipo, sin problemas de identificación. La dirección

00 es la denominada “de acceso general”; a esta responden todos los dispositivos conectados al bus.

- Lectura/Escritura (Bit R/W): Cada dispositivo tiene una dirección de 7 bits. El octavo bit (el menos significativo) que se envía durante la operación de direccionamiento, completando el byte, indica el tipo de operación a realizar. Si este bit es alto el dispositivo maestro lee información proveniente de un dispositivo esclavo. Si este bit es bajo, el dispositivo maestro escribe información en un dispositivo esclavo.

La Raspberry Pi también tiene una configuración para este tipo de dispositivos que emplean la interfaz i2c, y aún mejor tiene capacidad de procesamiento para interpolar y filtrar la salida del sensor. Al agregar potencia de procesamiento, puede 'convertir' la salida de 8x8 en lo que parece ser una pantalla de mayor resolución.

A continuación se muestra el diagrama de conexión para la raspberry y los pines SCL, SDA, GND y 3.3V de la cámara térmica, conectados a los puertos GPIO de la raspberry.

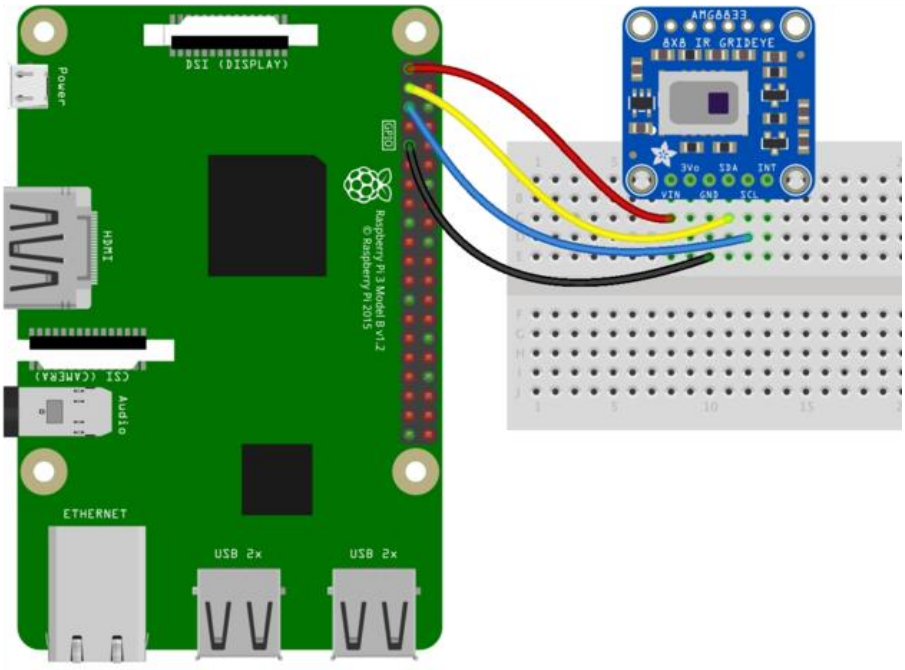

Fig. 26.- Cableado de la raspberry pi con la cámara térmica amg8833.

Para el caso de las librerías a emplear para el módulo de la amg8833 se requerirán las siguientes.

- *Adafruit_AMG88xx*. Esta librería permite el uso del módulo amg8833 en Python.

- *PyGame*. Es un módulo del lenguaje de programación Python que permiten la creación de videojuegos en dos dimensiones de una manera sencilla. Mediante PyGame podemos utilizar sprites (objetos), cargar y mostrar imágenes en diferentes formatos, sonidos, etc. Además, al ser un módulo destinado a la programación de videojuegos se puede monitorizar el teclado o joystick de una manera bastante sencilla. Para este caso, será usada para dibujar la pantalla donde estará la toma de la cámara térmica.

- *SciPy.* Como ya se había mencionado, es una biblioteca de código abierto que contiene herramientas y algoritmos matemáticos para Python. Convertirá la resolución 8x8 de la cámara a una de 32x32 para obtener 1024 valores para colores.

- *Colour*. Este módulo permite el manejo de los colores de la detección de la energía espectral para la cámara térmica. [32]

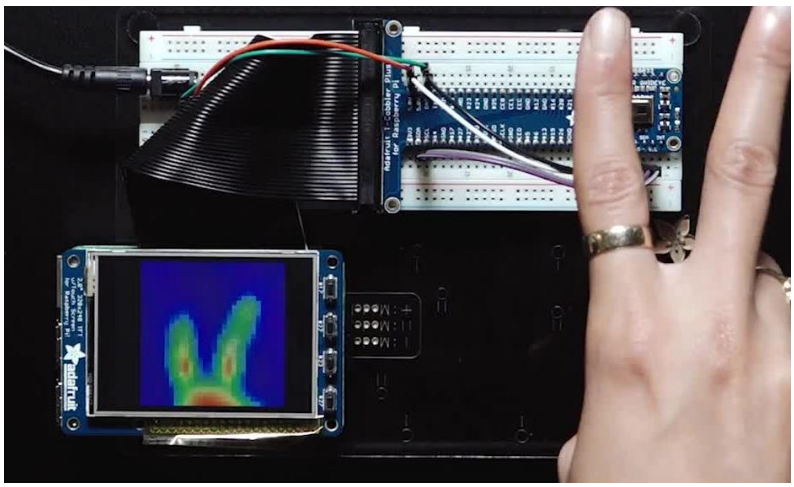

Fig. 27.- Funcionamiento de la amg8833 en la raspberry p. [32]

CAPITULO 4.-RESULTADOS.

En el capítulo 2 se estableció todo lo relacionado al software y a la teoría del algoritmo con su funcionamiento. En el capítulo 3 se estableció todo lo relacionado al hardware donde correrán dichos elementos mencionados en el 2. Ahora en este capítulo se reunirán todas las conclusiones hechas anteriormente para ponerlas a prueba y mostrar sus respectivos resultados.

4.1.- Instalación del sistema operativo raspbian en la rasperry pi 3.

Primeramente se debe cargar el sistema operativo en la tarjeta raspberry pi. El cual es raspbian. Rapbian es un sistema operativo basado en Debian de Linux el cual se encuentra en un paquete con varios instaladores llamado Noobs.

Para ello se utilizó una memoria micro SD de mínimo 32 GB, ya que posteriormente se instalara los paquetes de Opencv.

La micro sd que se utilizó es una sanDisk de clase 10 y 32 GB.

En el siguiente link, se descargaron los paquetes para la instalación del NOOBS en la micro sd.

https://www.raspberrypi.org/downloads/

Y seleccionar la siguiente opción:

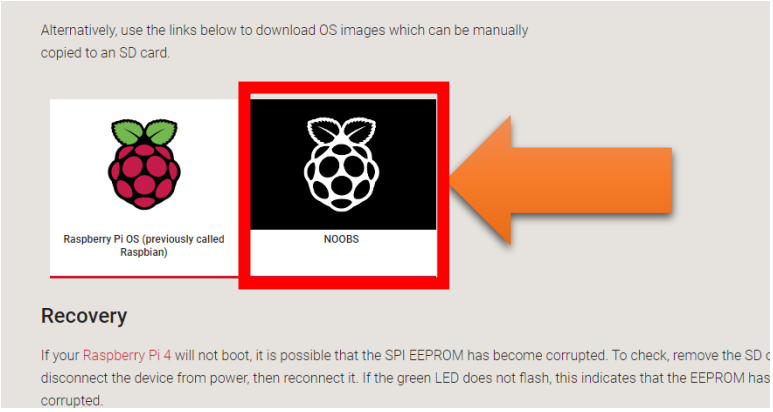

Fig. 28.- Selección del paquete NOOBS en la página de raspberry.

Esto re direccionará a otra pestaña donde se debe seleccionar la versión de noobs a utilizar.

Fig. 29.- Selección del NOOBS Full versión.

En lo que se descarga el Noobs, se formateo el micro SD el cual, se puede hacer desde las opciones básicas de nuestro sistema. Es decir, clic derecho en la unidad del micro SD, y presionar en formatear como se muestra en la siguiente figura 30 o utilizar un software, como en la figura 31.

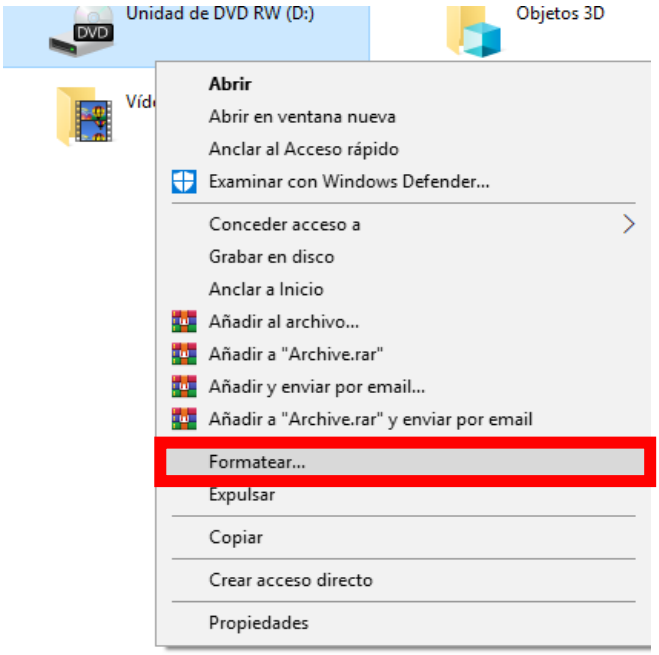

Fig. 30.-Opcion A para formatear, desde el sistema.

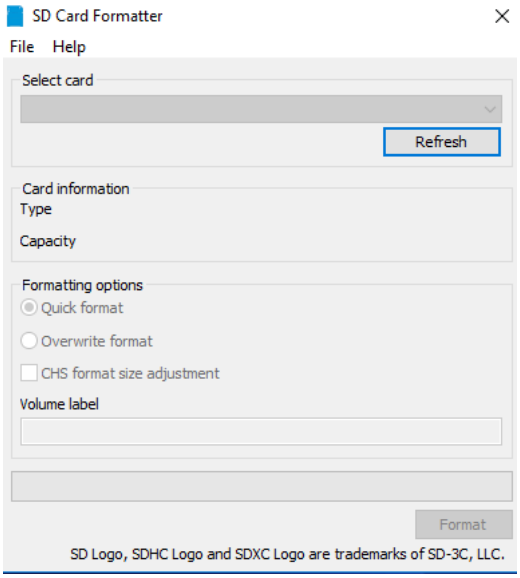

Fig. 31.-Opcion B para formatear, desde el software.

Una vez descargado el Zip con el Noobs, se procedio a descomprimirlo en el escritorio.

Los archivos descomprimidos de la fig. 32, se extrajeron, se copiaron y se pegaron e la micro SD.

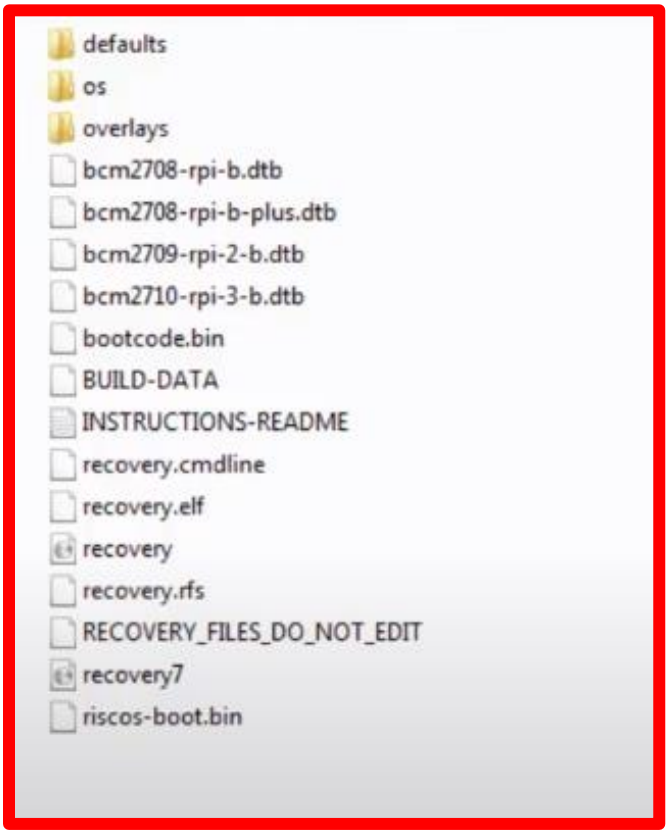

Fig. 32.- Archivos descomprimidos dentro de la carpeta NOOBS.

4.1.1- Configuración del NOOBS.

Una vez teniendo la Raspberry alimentada a 5V con un mínimo de 2A con mouse, teclado y monitor conectado. Se procedio a hacer lo siguiente:

Se selecciona la pestaña de *wiffi networks* (Fig. 33) para configurar el wiffi con un router, esto fue opcional, aunque ayuda a desplegar todas las opciones que ofrece NOOBS para tu instalación.

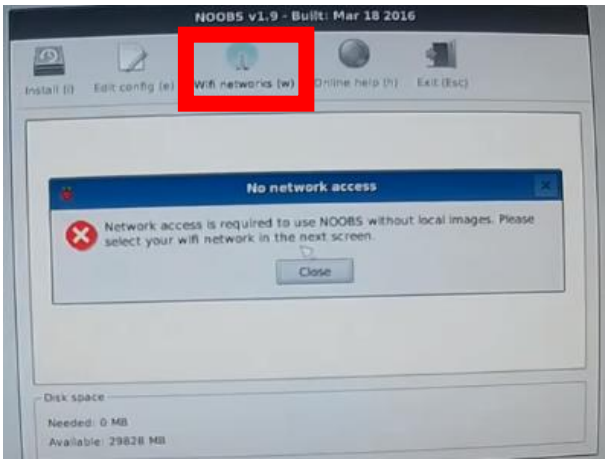

Fig. 33.-Configuracion de una red wiffi para la raspberry.

Posteriormente se seleccionó la opción “Raspbian” como se muestra en la figura 34. Para posteriormente seguir con la instalación de paquetes.

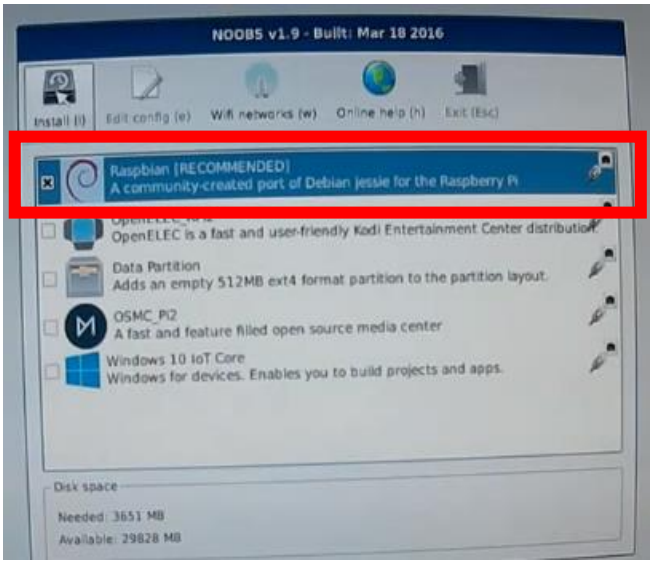

Fig. 34.- Selección del sistema operativo raspbian.

Finalmente después de 10 a 15 minutos en lo que carga al 100% (Fig. 35), se accede al escritorio de la raspberry, la cual fue el resultado correcto de la instalación de su sistema operativo (Fig. 36).

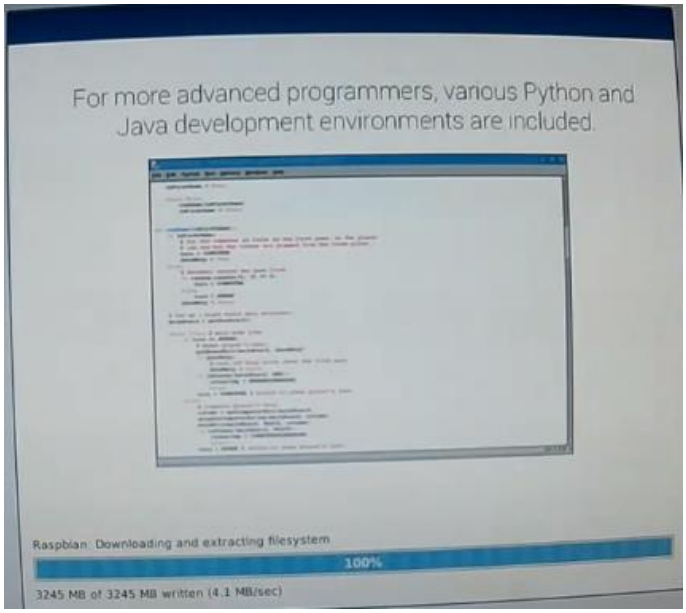

Fig. 35.- Pantalla final para la instalación del sistema operativo.

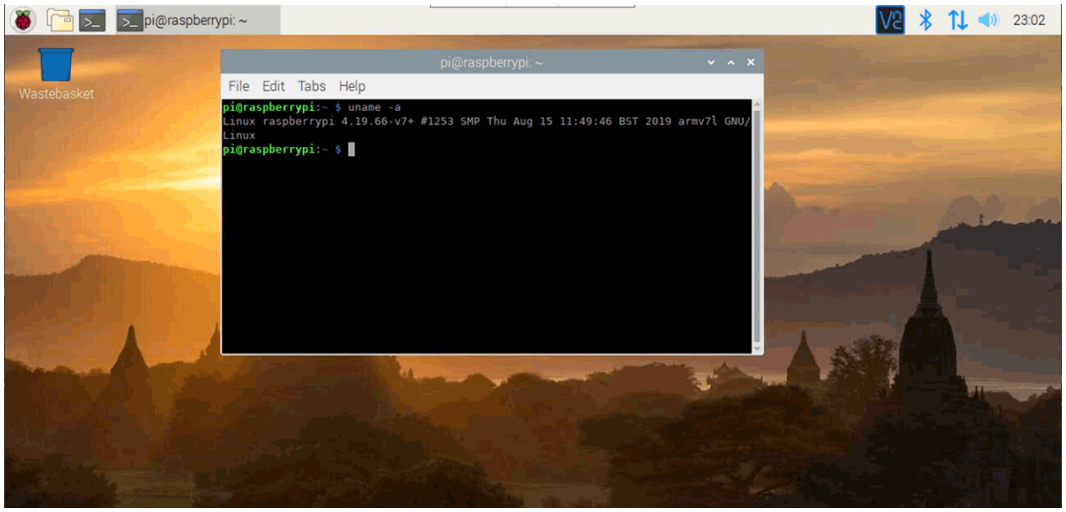

Fig. 36.- Pantalla del escritorio de la rasperry pi.

Finalmente se escribieron las siguientes dos líneas en la terminal para terminar de actualizar los paquetes base de la raspberry junto con sus aplicaciones que vienen de fábrica.

1. *sudo apt-get update*
2. *sudo apt-get upgrade*

El proceso tardó aproximadamente 15 minutos más, pero se actualizaron y se configuraron los paquetes en la Raspberry. Esto solo es la primera vez, las demás veces que se volvieron a ejecutar estas líneas de comando, solo tardaron 2 o 3 minutos a lo mucho.

4.2.- Configuración de Opencv.

Aprovechando una de las ventajas de raspberry, es que ya viene con Python 3 instalado, de acuerdo a los pasos anteriores, por lo cual a partir de este punto se procedió a instalar las librerías que se usaran en los códigos de programación del sistema de visión.

Cabe mencionar que se siguió el procedimiento para la instalación de Opencv 4 en una Raspberry 4, las cuales son las versiones más recientes hasta el momento, sin embargo el resultado final no dio inconvenientes para la instalación de Opencv 3 en una raspberry 3, ya que son muy similares.

Se inició ejecutando las siguientes instrucciones en una terminal.

1. sudo apt-get update

2. sudo apt-get upgrade

Esto actualizó los paquetes y dejó a la Raspberry mejor preparada para la instalación de uno nuevo, en este caso el paquete de Opencv 4.

Se procede a instalar los siguientes paquetes de Linux, Python y varias librerías que servirán durante el proceso de instalación en una terminal.

3. sudo apt-get install build-essential cmake unzip pkg-config

4. sudo apt-get install libjpeg-dev libpng-dev libtiff-dev

5. sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev

6. sudo apt-get install libxvidcore-dev libx264-dev

7. sudo apt-get install libgtk-3-dev

8. sudo apt-get install libcanberra-gtk*

9. sudo apt-get install libatlas-base-dev gfortran

10. sudo apt-get install python3-dev

Se descarga Opencv.

11. cd ~

12. wget -O opencv.zip https://github.com/opencv/opencv/archive/4.0.0.zip

13. wget-Oopencv_contrib.zip
https://github.com/opencv/opencv_contrib/archive/4.0.0.zip

Se descomprimen los archivos.

14. unzip opencv.zip

15. unzip opencv_contrib.zip

2.1.- Configuración e instalación de Opencv.

Se configuró un entorno virtual de Python 3 para Opencv, esto se realizó instalando Python Pip, un tipo de Python Package Manager.

1. wget https://bootstrap.pypa.io/get-pip.py

2. sudo python3 get-pip.py

Los entornos virtuales permitirán ejecutar diferentes versiones del software Python de forma aislada en el sistema. Las siguientes instrucciones son para configurar un solo entorno, pero fácilmente se podría tener un entorno para cada proyecto.

3. sudo pip install virtualenv virtualenvwrapper

4. sudo rm -rf ~ / get-pip.py ~ / .cache / pip

Se edita el archivo .bashrc, se coloca con el comando sudo nano .bashrc y se pega lo siguiente.

virtualenv y virtualenvwrapper

export WORKON_HOME = $ HOME /.virtualenvs

export VIRTUALENVWRAPPER_PYTHON = / usr / bin / python3

fuente /usr/local/bin/virtualenvwrapper.sh

Se guarda con Ctrl+ X y se presiona la tecla enter.

Posteriormente se ejecuta lo siguiente:

5. source ~ / .profile

Para que el sistema identifique el archivo previamente modificado con sus cambios.

Se crea el entorno virtual con la siguiente instrucción desde el terminal.

6. mkvirtualenv cv -p python3

Posteriormente a esto se ejecuta la siguiente instrucción:

7. workon cv

Se observará un CV a un lado del nombre de usuario en la terminal.

Ya dentro del espacio virtual. Se instala Numpy.

8. pip install numpy

A partir de aquí, empieza la instalación de Opencv 4.

Se crea la carpeta build dentro de la carpeta Opencv que se descargó previamente y se accede a ella.

9. cd ~ / opencv

10. mkdir build

11. cd build

Con mucha precaución y en la terminal (muy importante) se pegan los siguientes archivos, los cuales son los compiladores del opencv 4.

cmake -D CMAKE_BUILD_TYPE = LIBERACIÓN \

-D CMAKE_INSTALL_PREFIX = / usr / local \

-D OPENCV_EXTRA_MODULES_PATH = ~ / opencv_contrib / modules \

-D ENABLE_NEON = ON \

-D ENABLE_VFPV3 = ON \

-D BUILD_TESTS = OFF \

-D OPENCV_ENABLE_NONFREE = ON \

-D INSTALL_PYTHON_EXAMPLES = OFF \

-D BUILD_EXAMPLES = OFF ..

Salir de la carpeta build.

12. cd .. build

Abrir el archivo / etc / dphys-swapfile, y verificar que esté el CONF_SWAPSIZE = 2048

Salir con Ctrl+X y reiniciar con.

13. sudo /etc/init.d/dphys-swapfile stop

14. sudo /etc/init.d/dphys-swapfile start

Ejecutar la instalación de Opencv 4 con la siguiente instrucción.

15. make -j4

A partir de aquí, este proceso ira instalando todos los paquetes de opencv, mas no configurarlos, esto demoró aproximadamente 1 hora y media teniendo una fuente de ventilación a la Raspberry, aunque esto también depende de la micro SD y otros factores como la velocidad del internet que para este caso era de 10 Megas. Finalmente una vez al 100%, se ejecutaron las siguientes líneas de comando en la terminal para terminar de configurar Opencv 4.

16. sudo make install

17. sudo ldconfig

18. regresa el CONF_SWAPSIZE = 100, como era originalmente.

Cerrar la terminal y verificar con las siguientes instrucciones si se instaló bien Opencv, si sale esto quiere decir que fue un éxito.

19. python

20. >>> import cv2

21. >>> cv2. __versión__

22. '4.0.0'

4.3.- Instalación de las librerías de Zbar para la detección de códigos QR.

Una vez instalado Opencv, se procedió a instalar las librerías de Zbar, las cuales a diferencia de Opencv, su instalación es mucho más sencilla con diferencia y más rápida, así que, desde un terminal se ejecutó lo siguiente:

Preparar las librerías para usar Zbar.

1. sudo apt-get install libzbar0
2. workon barcode

Instalar Zbar.

3. pip install pyzbar
4. sudo apt-get update
5. sudo apt-get install zbar-tools

Y después en una nueva terminal como medio de comprobación, se escribió zbarcam y se pudo observar una interfaz de prueba donde se pueden rectificar varios QR dando la idea de que la instalación había sido un éxito. Pero esto es solo una prueba y es muy lento ya que carga con demasiados recursos a la raspberry por lo cual no fue considerado más allá de una práctica y no como parte del resultado final.

Ahora se pueden agregar las siguientes librerías de Zbar y empezar a trabajar con ellas, entre otras.

```
from pyzbar import pyzbar
import argparse
import cv2
```

4.4.-. Instalación de las librerías de la cámara térmica amg8833.

Se inició instalando las librerías para la configuración de los colores para la cámara térmica y la librería pygame.

1. sudo apt-get install -y python-scipy python-pygame
2. sudo pip3 install color

Posteriormente la librería de la cámara térmica.

3. sudo pip3 install adafruit-circuitpython-amg88xx

Se activó la interfaz I2C de la Raspberry, para en la opción de Raspberry config y en la sección de interfaces se marcó como "*Eneable*". Posteriormente la misma raspberry pi manda mensaje que debe reiniciarse. Para ya se debe tener conectada la cámara térmica como en la figura 27.

Una vez aclarado esto, se procedio a ejecutar lo siguiente en una nueva terminal.

4. sudo i2cdetect -y 1

Como resultado esto apareció en la pantalla, todo salió muy bien ya que esto nos dice que la Raspberry detectó la cámara con éxito.

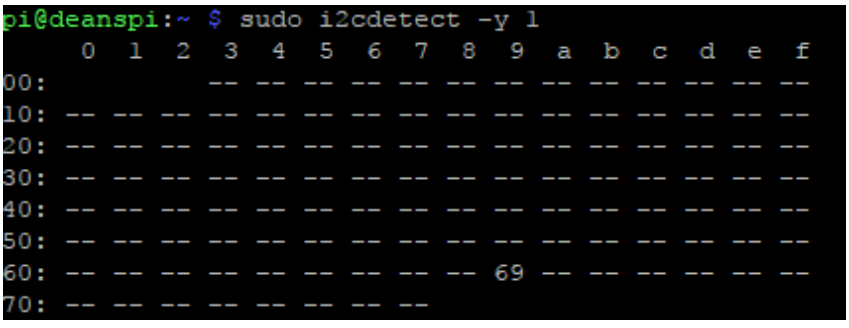

Fig. 37.- Detección de la cámara térmica con la comunicación i2c desde la raspberry pi.

4.5.-. Resultado final.

Con todo lo anterior mencionado y dejando las librerías correctamente instaladas, se procedio a escribir el código final el cual estará en la parte de anexos de este documento, con todas las librerías de Python, opencv, Zbar y el de la propia amg8833. Todo en uno solo, dando como resultado al ejecutar el fichero la figura 38, donde observamos como abre 3 pantallas.

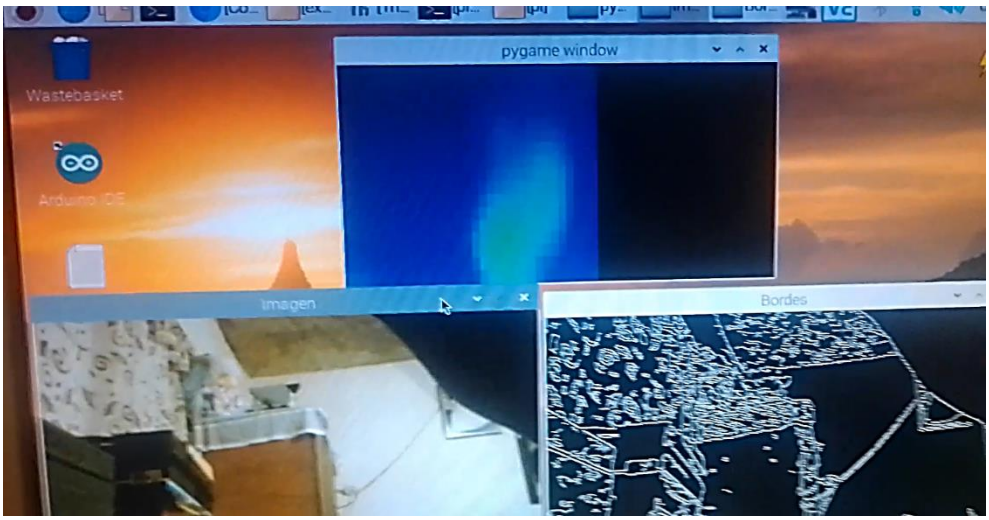

Fig. 38.- Algoritmo de visión ejecutándose correctamente.

A continuación observamos la detección de una etiqueta de peligro como muestra del resultado final de la detección de patrones.

Fig. 39.- Detección de patrones mediante Canny, resultado final.

A continuación se muestra la detección de QR, al momento de pasar uno enfrente de la cámara web, empleando zbar.

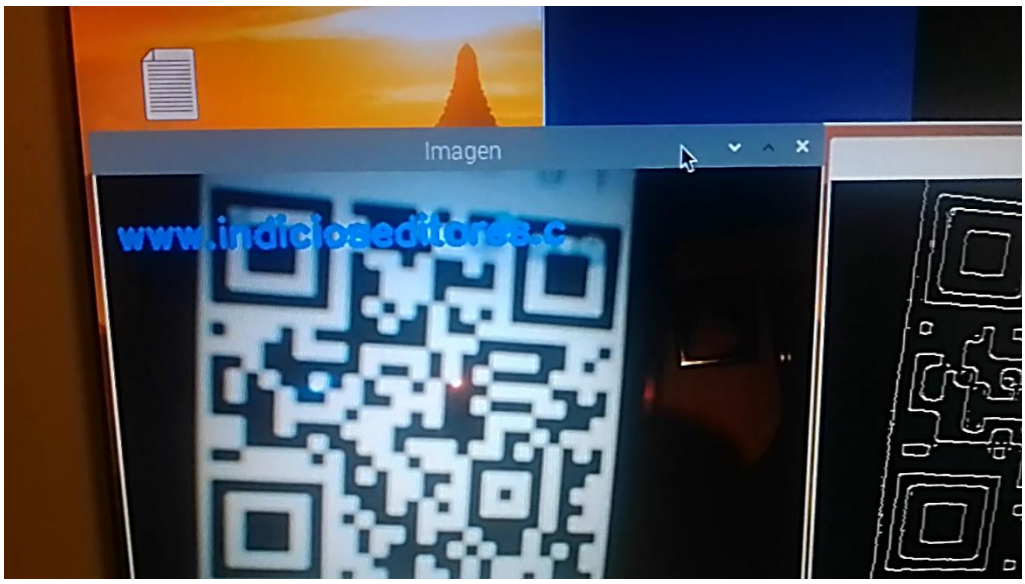

Fig. 40.- Detección QR mediante Zbar, resultado final.

Finalmente la prueba de la cámara térmica con un fosforo.

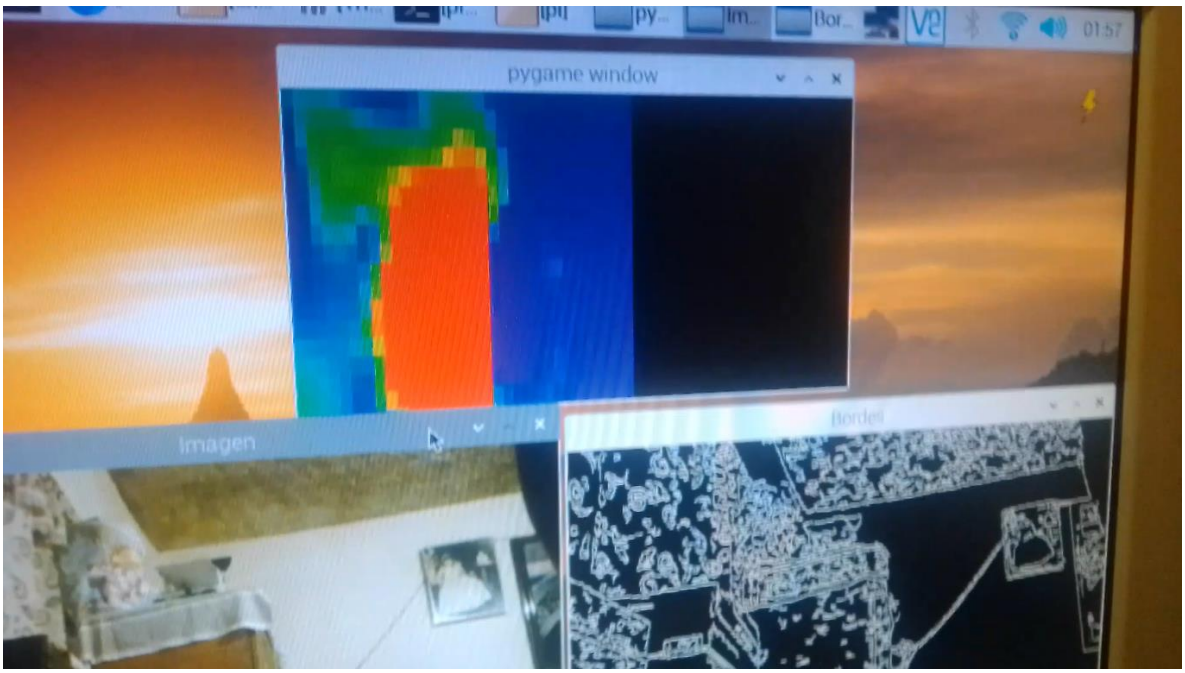

Fig. 41.- Detección del calor de un fosforo de la cámara térmica, resultado final.

CAPITULO 5.- CONCLUSIONES.

Analizando los resultados obtenidos durante las pruebas de la cámara con el algoritmo de visión, se podría concluir que la propuesta de solución que se planteó al inicio, dio un resultado esperado y funcional.

Cabe mencionar que en primera instancia, se comparó el rendimiento de las tarjetas Jetson nano y raspberry, para seleccionar cual sería la ideal para hacer el algoritmo de visión. El resultado concluye, en que, a pesar de que la tarjeta Jetson nano fue más rápida que la raspberry, esta última tiene muchas más fuentes de información de las librerías necesarias para llevar a cabo el algoritmo, que la Jetson nano, por lo cual al final se terminó por utilizar la raspberry, aunque como era de esperarse, el equipo en ocasiones presentaba pequeños atrasos en los fotogramas, debido al procesamiento que tenían los tres algoritmos corriendo al mismo tiempo en la raspberry, sin embargo todo estaba dentro del margen de error esperado.

Por lo tanto los objetivos también fueron concluidos de manera acertada, como se mencionaba antes, esto convierte a este sistema de visión en algo completamente funcional que implementado en su base móvil, el operador podrá acceder cómodamente a las cámaras para revisar el entorno. Todo esto a pesar de estar utilizando elementos de bajo costo, no habría problema que le impida realizar su trabajo. Aunque esto abre puertas, a que se pueda mejorar ya que también presenta muchas áreas de oportunidad, como lo son la resolución de las cámaras y la de la cámara térmica, que es el aspecto que puede llevar a una mayor inversión y por supuesto la computadora central. Por lo anteriormente mencionado sí que se recomendaría utilizar cámaras de mejor resolución o la adquisición de una cámara térmica más profesional.

CAPÍTULO 6.-REFERENCIAS.

[1] Sonia Corona, “El precio de un terremoto”, nota del periódico el País, 18 de septiembre del 2015. Recuperado de https://bit.ly/2xqwNZi

[2] Alberto Nájar, “Sismo 2017 en México: las lecciones no aprendidas que dejó el terremoto del 19 de septiembre” nota de la BBC news, 19 de septiembre del 2018. Recuperado de https://bbc.in/2xifNod

[3] S. U. Sharma and D. J. Shah, "A Practical Animal Detection and Collision Avoidance System Using Computer Vision Technique," in IEEE Access, vol. 5, pp. 347-358, 2017.

[4] M. Ben Ayed, S. Elkosantini, S. A. Alshaya and M. Abid, "Suspicious Behavior Recognition Based on Face Features," in IEEE Access, vol. 7, pp. 149952-149958, 2019.

[5] V. Mazzia, A. Khaliq, F. Salvetti and M. Chiaberge, "Real-Time Apple Detection System Using Embedded Systems with Hardware Accelerators: An Edge AI Application," in IEEE Access, vol. 8, pp. 9102-9114, 2020.

[6] Xiang Zhang, Hangzai Luo, Jinye Peng, Jianping Fan, Long Chen,” Fast QR code detection” in College of Information and Technology, Northwest University of China 2017.

[7] DPS Setyohadi, V Rahmania, H Y Riskiawan, S T Sarena, S Arifin, DE Putra and Edy Setiawan, “Thermal Image Processing Using Artificial Neural Network for Boiler TV-Furnace (Thermal CCTV) Position Control System” in Information Technology Department, Politeknik Negeri Jember, Indonesia 2019.

[8] M. C. José Jaime Esqueda Elizondo, “Fundamentos de Procesamiento de Imágenes” in Instituto Tecnológico de ciudad Madero. CONATEC, Noviembre de 2002.

[9] Gonzalo Pajares, Jesús M. de la Cruz. Visión por Computador. Imágenes digitales y aplicaciones. RaMa, 2001.

[10] Domingo Mery, “Visión por computador”. Departamento de Ciencia de la Computación en la Universidad Católica de Chile, Santiago de Chile 17 de agosto de 2004.

[11] Enrique Sucar, “Visión computacional” Instituto Nacional de Astrofísica, Óptica y Electrónica, Puebla, México.

[12] Quintana Vivanco, Leaned & Hubert, Fredes & Marañón, Enrique. (2013). OPTIMIZACIÓN DE LOS FILTROS MEDIANA-GAUSSIANO PARA UNA MEJOR CONVERGENCIA DEL SNAKE EN LA SEGMENTACIÓN DE IMÁGENES MÉDICAS.

[13] Valverde-Rebaza, Jorge. (2007). Detección de bordes mediante el algoritmo de Canny.

[14] Ivet Challenger Pérez, Yanet Díaz Ricardo y Roberto Antonio Becerra García " El lenguaje de programación Python," in IEEE Access, vol. 5, pp. 347-358, 2017. Universidad de Holguín Oscar Lucero Moya, Cuba 2014.

[15] González (2018, 21 Sept). “Introducción a la librería Numpy de python”. Recuperado de: https://ligdigonzalez.com/introduccion-a-numpy-python-1/

[16] Página web UniPython (2018, 27 Abril). “Scipy funciones principales”. Recuperado de: https://unipython.com/scipy-funciones-principales/

[17] Gonzalez (2028, 19 de octubre). “Introduccion a la librería matplotlib”. Recuperado de :https://ligdigonzalez.com/libreria-pandas-de-matplotlib-tutorial/

[18] Fernandez, D. P. (2020, 15 junio). ¿Cómo instalar PIP para Python en Windows, Mac y Linux? 【PASO A PASO】 Tecnonucleous. https://tecnonucleous.com/2018/01/28/como-instalar-pip-para-python-en-windows-mac-y-linux/

[19] Luismi Gracia, "¿Qué es Opencv?". Recuperado de: https://cutt.ly/4yZoUWT

[20] Página Tutor de programación, segmento llamado "Acceso a la webcam con OpenCV", 19 de agosto del 2017. Recuperado de: https://cutt.ly/byVDwZ9

[21] Jeff Brown. "ZBar bar code reader", 15 de Julio del 2015. Recuperado de: https://cutt.ly/KyVDiqU

[22] Python Software Foundation, "Mathematical functions". Enero del 2020. Recuperado de: https://cutt.ly/iyVDaob

[23] Robologs, "Detección de códigos QR en Python con OpenCV y ZBar", posteado el 17 de Julio del 2017. Recuperado de: https://cutt.ly/vyVDsEI

[24] Aguayo., P. (2004, 10 de Noviembre). Introducción al microcontrolador.

[25] Eriquez, R. (2009, 13 noviembre). Guia de usuario de Arduino (Revisado ed.). Creative Commons.

[26] Navas, M. Á. (2018, 22 febrero). Qué es y para qué sirve el microprocesador o CPU. Profesional Review. https://www.profesionalreview.com/2018/02/28/que-es-para-que-sirve-microprocesador/#Para_que_sirve_un_microprocesador

[27] Luis, E. R. (2018, 18 septiembre). De cero a maker: todo lo necesario para empezar con Raspberry Pi. Xataka. https://www.xataka.com/makers/cero-maker-todo-necesario-para-empezar-raspberry-pi

[28] Naranjo., D. “Nvidia Jetson Nano: una computadora para la implementación de aplicaciones AI”. Recuperado de: https://cutt.ly/xiHvgtK

[29] Pietschmann, C. (2020, 14 junio). Raspberry Pi 4 Vs NVIDIA Jetson Nano Developer Kit. Build5Nines. https://build5nines.com/raspberry-pi-4-vs-nvidia-jetson-nano-developer-kit/

[30] González., O. (2015 21 diciembre). “Vision Infrarroja”. Recuperado de: https://culturacientifica.com/2015/12/21/vision-infrarroja/

[31] E. (2018, 7 diciembre). Descripción y funcionamiento del Bus I2C | Robots Didácticos. Descripción y funcionamiento del I2C. http://robots-argentina.com.ar/didactica/descripcion-y-funcionamiento-del-bus-i2c/

[32] Dean Miller, “Raspberry Pi Thermal Camera”, actualizado el 04 de enero del 2020. Recuperado de: https://cutt.ly/hyVDhNG

ANEXOS.

Anexo A.- Código de verificación del funcionamiento de la cámara térmica amg8833.

```
1.  import os
2.  import math
3.  import time
4.
5.  import busio
6.  import board
7.
8.  import numpy as np
9.  import pygame
10. from scipy.interpolate import griddata
11.
12. from colour import Color
13.
14. import adafruit_amg88xx
15.
16. i2c_bus = busio.I2C(board.SCL, board.SDA)
17.
18. #low range of the sensor (this will be blue on the screen)
19. MINTEMP = 26.
20.
```

```
21.  #high range of the sensor (this will be red on the screen)
22.  MAXTEMP = 32.
23.
24.  #how many color values we can have
25.  COLORDEPTH = 1024
26.
27.  os.putenv('SDL_FBDEV', '/dev/fb1')
28.  pygame.init()
29.
30.  #initialize the sensor
31.  sensor = adafruit_amg88xx.AMG88XX(i2c_bus)
32.
33.  # pylint: disable=invalid-slice-index
34.  points = [(math.floor(ix / 8), (ix % 8)) for ix in range(0, 64)]
35.  grid_x, grid_y = np.mgrid[0:7:32j, 0:7:32j]
36.  # pylint: enable=invalid-slice-index
37.
38.  #sensor is an 8x8 grid so lets do a square
39.  height = 240
40.  width = 240
41.
42.  #the list of colors we can choose from
43.  blue = Color("indigo")
```

```
44.  colors = list(blue.range_to(Color("red"), COLORDEPTH))
45.
46.  #create the array of colors
47.  colors = [(int(c.red * 255), int(c.green * 255), int(c.blue * 255)) for c in colors]
48.
49.  displayPixelWidth = width / 30
50.  displayPixelHeight = height / 30
51.
52.  lcd = pygame.display.set_mode((width, height))
53.
54.  lcd.fill((255, 0, 0))
55.
56.  pygame.display.update()
57.  pygame.mouse.set_visible(False)
58.
59.  lcd.fill((0, 0, 0))
60.  pygame.display.update()
61.
62.  #some utility functions
63.  def constrain(val, min_val, max_val):
64.  return min(max_val, max(min_val, val))
65.
66.  def map_value(x, in_min, in_max, out_min, out_max):
```

```
67.     return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min
68.
69.     #let the sensor initialize
70.     time.sleep(.1)
71.
72.     while True:
73.
74.     #read the pixels
75.     pixels = []
76.     for row in sensor.pixels:
77.     pixels = pixels + row
78.     pixels = [map_value(p, MINTEMP, MAXTEMP, 0, COLORDEPTH - 1) for p
in pixels]
79.
80.     #perform interpolation
81.     bicubic = griddata(points, pixels, (grid_x, grid_y), method='cubic')
82.     #draw everything
83.     for ix, row in enumerate(bicubic):
84.     for jx, pixel in enumerate(row):
85.     pygame.draw.rect(lcd, colors[constrain(int(pixel), 0, COLORDEPTH- 1)],
86.     (displayPixelHeight * ix, displayPixelWidth * jx,
87.     displayPixelHeight, displayPixelWidth))
88.     pygame.display.update()
```

Anexo B.- código final en Python, con todas las librerías de Opencv para la detección de QR (Zbar), detección de patrones (Canny), y funciones de la cámara térmica amg8833, para una raspberry pi modelo 3.

```
################################ INICIO ################################
from Adafruit_AMG88xx import Adafruit_AMG88xx
import pygame
import os
import math
import time
from time import sleep
import zbar
import numpy as np
import cv2
from scipy.interpolate import griddata
from colour import Color
import imutils
##########################CAMARA  TERMICA#####################


#Rango minimo de deteccion de temperatura.
MINTEMP = 32


#Rango maximo de deteccion de temperatura.
```

```
MAXTEMP = 38

#Cantidad de valores que podemos obtener de los colores de la camara.
COLORDEPTH = 1024
os.putenv('SDL_FBDEV', '/dev/fb1')
pygame.init()

#Activa la camara termica
sensor = Adafruit_AMG88xx()

points = [(math.floor(ix / 8), (ix % 8)) for ix in range(0, 64)]
grid_x, grid_y = np.mgrid[0:7:32j, 0:7:32j]

#Tamaño de la ventana
height = 240 #240
width = 440 #240

#Lista de colores que se pueden elegir
blue = Color("indigo")
colors = list(blue.range_to(Color("red"), COLORDEPTH))

#configuracion de los array con los colores.
colors = [(int(c.red * 255), int(c.green * 255), int(c.blue * 255)) for c in colors]
```

```
displayPixelWidth = width / 30

displayPixelHeight = height / 30


lcd = pygame.display.set_mode((width, height))


lcd.fill((255,0,0))


pygame.display.update()

pygame.mouse.set_visible(False)


lcd.fill((0,0,0))

pygame.display.update()


#declaracion de variables que se utilizaran

def constrain(val, min_val, max_val):

    return min(max_val, max(min_val, val))


def map(x, in_min, in_max, out_min, out_max):

  return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min


#Tiempo de iniciacion del sensor de la camara termica.

time.sleep(.1)
```

```
#Activacion de la camara de PS3 para los QR y los patrones.
capture = cv2.VideoCapture(0)

while True:
############################## PATRONES ##############################
    val, frame = capture.read()# VAL es para los codigos QR y frame para los patrones
    frame = imutils.resize(frame, width=500) #Tamaño de la pantalla 500x500 pixeles
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)#Convierto la imagen a color en blanco y negro.
    edges = cv2.Canny(frame,100,200) #Hago la deteccion de los bordes mediante Canny.
    _,cnts,_=cv2.findContours(edges,cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)#En una variable llamada cnts,busco los contornos que se encuentren en el filtro de canny que salio del paso anterior.
    for c in cnts: #En un for declaro una nueva variable llamada C para tener los valores de las areas.
        area=cv2.contourArea(c)
        if area>8000: #Me delimito a esta área para que no me detecte el fondo.
            cv2.drawContours(frame, [c], 0, (0,255,0),3) #B G R en color verde
            texto='Label encontrada'

cv2.putText(frame,texto,(10,50),cv2.FONT_HERSHEY_SIMPLEX,2,(255,0,0),4) #B G R
```

```
########################## QR ########################################
    if val:
        #Capturar un frame con la camara y guardar sus dimensiones
        frame_gris = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        dimensiones = frame_gris.shape #'dimensiones' sera un array que contendra el alto, el ancho y los canales de la imagen en este orden.
        #Convertir la imagen de OpenCV a una imagen que la libreria ZBAR pueda entender
        imagen_zbar = zbar.Image(dimensiones[1], dimensiones[0], 'Y800', frame_gris.tobytes())

        #Construir un objeto de tipo scaner, que permitira escanear la imagen en busca de codigos QR
        escaner = zbar.ImageScanner()
        #Escanear la imagen y guardar todos los codigos QR que se encuentren
        escaner.scan(imagen_zbar)

        for codigo_qr in imagen_zbar:
            dat = codigo_qr.data[:-2] #Guardar el mensaje del código QR. Los últimos dos caracteres son saltos de línea que hay que eliminar

cv2.putText(frame,dat,(10,50),cv2.FONT_HERSHEY_SIMPLEX,0.9,(255,0,0),3) #Imprime el valor de los datos del QR en la pantalla
```

```
#######REGRESAMOS A LA CAMARA TERMICA Y EL FINAL################
    #Lectura de pixeles de la camara termica
    pixels = sensor.readPixels()
    pixels = [map(p, MINTEMP, MAXTEMP, 0, COLORDEPTH - 1) for p in pixels]

    #Realiza una interpolacion de los datos de la camara termica
    bicubic = griddata(points, pixels, (grid_x, grid_y), method='cubic')
    #print('Thermistor Temp = {0:0.2f} *C'.format(sensor.readThermistor()))

    #En un for se dibuja la pantalla para mostrar los colores de la camara termica.
    for ix, row in enumerate(bicubic):
        for jx, pixel in enumerate(row):
            pygame.draw.rect(lcd, colors[constrain(int(pixel), 0, COLORDEPTH- 1)],
(displayPixelHeight * ix, displayPixelWidth * jx, displayPixelHeight,
displayPixelWidth))

    pygame.display.update()
    #Mostrar la imagen de los patrones y los QR
    cv2.imshow('Imagen', frame)
    cv2.imshow('Bordes',edges)
    k = cv2.waitKey(5) & 0xFF
    if k == 27:
        break
```

```
# CERRAR TODO

cap.release()

cv2.destroyAllWindows()

################################## FIN ##################################
```