DISEÑO DE UN PROTOTIPO FUNCIONAL DE VEHÍCULO AUTÓNOMO POR MEDIO DE VISIÓN ARTIFICIAL Y SENSORES DE PROXIMIDAD, IMPLEMENTADO EN PYTHON SOBRE ARQUITECTURA DE RASPBERRY PI.

HÉCTOR ANDRÉS RODRÍGUEZ MARTÍNEZ
MIGUEL LEONARDO OCAMPO GÓMEZ
NEIRO CULMA YATE

UNIVERSIDAD ECCI
Facultad de Ingeniería
Programa de Ingeniería de Sistemas
Bogotá, D.C.
Febrero 2018

DISEÑO DE UN PROTOTIPO FUNCIONAL DE VEHÍCULO AUTÓNOMO POR MEDIO DE VISIÓN ARTIFICIAL Y SENSORES DE PROXIMIDAD, IMPLEMENTADO EN PYTHON SOBRE ARQUITECTURA DE RASPBERRY PI.

Presentado por:
HÉCTOR ANDRÉS RODRÍGUEZ MARTÍNEZ
MIGUEL LEONARDO OCAMPO GÓMEZ
NEIRO CULMA YATE

Presentado a:
Ing. Msc. LUIS EFRAÍN RUIZ SUAREZ (Director)
Ing. M.Ed. OSCAR ALBERTO ZAMBRANO OSPINA (Asesor Metodológico)

UNIVERSIDAD ECCI
Facultad de Ingeniería
Programa de Ingeniería de Sistemas
Bogotá, D.C.
Febrero 2018

Dedicatoria

Dedicamos este trabajo principalmente a nuestras familias, en especial a nuestros padres Stella Gómez, Miguel Ocampo, Olga Martínez y Héctor Rodríguez, María Eugenia Yate, quienes han velado por nuestro bienestar y educación, quienes con su apoyo incondicional nos motivaron a finalizar con éxito este gran proyecto.

Agradecimientos

Agradecemos al ingeniero Abdul Ruiz por su apoyo incondicional y orientación durante todo este proceso.

Al Ing. Efraín Ruiz, director para este trabajo de grado, por su valiosa guía y asesoramiento a la realización de la misma.

A la Universidad ECCI por la formación brindada, la cual nos permitió adquirir las bases necesarias para desarrollar este trabajo.

Por último, cada uno de los miembros de este equipo de trabajo agradece al resto de compañeros de proyecto, al considerar que sin el compromiso, armonía, arduo trabajo, esfuerzo y dedicación fue posible conseguir los resultados propuestos.

Abstract

There are currently a large number of traffic accidents due to the recklessness and experience of drivers of vehicles transiting the city. The statistics show that when bogotan people move from one place to another, the time spent can be extended even to the double by the bottlenecks, generated for various reasons.

From the travails to arrive opportunely to the site of destination, a person becomes incurred in the failure to comply with the rules of transit, carrying economic fines avoidable by computer solutions that employ artificial intelligence algorithms (AI), generating value for society on reducing rates of accidents as well as in tickets in terms of penalties.

Through this project, it is intended to identify and study a solution that can be considered feasible against the forementioned problems. For the foregoing, we will evaluate the use of a combination of computer technologies, understanding that similar searches are revolutionizing the automotive industry today, defining a potentially interesting market for them in the future years.

In order to carry out the proposed evaluation, a prototype of a scaled vehicle is used which was equipped with a light computing device, in which the algorithms required to make "intelligent" use of the information got from the sensors, allowing to generate answers from the actuators that are appropriate to the situations whose management has been defined as fundamental.

In this order of ideas, by making use of the advantages provided by a computer system, as well as a configurable model that allows to generate different random scenarios, with obstacles of diverse nature, it is possible to ensure that the algorithms in check are tested at such a level of rigor

that the appropriate selection and calibration of the most appropriate for the final proposal is reached.

Prefacio

En la actualidad existe un gran número de accidentes de tránsito debido a la imprudencia y experiencia de los conductores de vehículos que transitan por la ciudad. Las estadísticas muestran que cuando los bogotanos se movilizan de un sitio a otro, el tiempo empleado se puede prolongar inclusive hasta el doble por los embotellamientos, generados por diversos motivos.

A partir de los afanes para llegar oportunamente al sitio de destino, se llega a incurrir en el incumplimiento de las normas de tránsito, acarreando multas económicas evitables con soluciones informáticas que empleen algoritmos de inteligencia artificial (IA) generando valor a la sociedad en la disminución de las tasas de accidentalidad y a los contribuyentes en términos de penalizaciones.

Por medio del presente proyecto, se pretende identificar y estudiar una solución que pueda ser considerada factible ante las problemáticas mencionadas anteriormente. Para lo anterior, se evaluará el uso de una combinación de tecnologías informáticas, entendiéndose que búsquedas similares vienen revolucionando la industria automovilística en la actualidad, vislumbrándose un mercado potencialmente interesante para las mismas en los años futuros.

En función de poder efectuar la evaluación propuesta, se utiliza un prototipo de vehículo a escala el cual fue equipado con un dispositivo de computación ligero, en el cual se implementan los algoritmos requeridos para hacer uso “inteligente” de la información proveniente de los sensores y de esta forma generar respuestas sobre los actuadores que sean apropiadas ante las situaciones cuyo manejo ha sido definido como fundamental.

En este orden de las ideas, al hacer uso de las ventajas que brinda un sistema informático, así como de una maqueta configurable que permita generar diferentes escenarios aleatorios, con

obstáculos de diversa índole, se vela por que los algoritmos en comprobación sean puestos a prueba a tal nivel de rigurosidad que se llegue a una adecuada selección y calibración de los más apropiados para la propuesta final.

Tabla de contenido

Tabla ilustraciones

Ilustración 29 Diagrama de Gantt*72*

Tabla de tablas

Problema de investigación

1.1. Descripción del problema

Según estudios recientes, en las últimas décadas los accidentes de tránsito han dejado un alto índice de mortalidad. A la fecha, los siniestros viales son una de las primeras causas de muerte, principalmente en un rango de edades entre 5 a 44 años.

La Organización Mundial de la Salud OMS, es el organismo más importante para la regulación y seguimiento de esta problemática. Los accidentes de este tipo en varios países se han convertido en una epidemia difícil de controlar, por esta razón la OMS ha decidido implementar acciones a través de la iniciativa “Década de Acción para la seguridad vial 2011-2020”. **Fuente especificada no válida.**

Según cifras arrojadas por el estudio antes mencionado, todos los años fallecen cerca de 1.2 millones de personas en el mundo y entre 20 y 50 millones de personas sufren heridas, además, informa que el 62% de la población que hace parte de las víctimas mortales se presentan en países como: India, China, Estados Unidos, Rusia, Brasil, Irán, México, Indonesia, Sudáfrica y Egipto. **Fuente especificada no válida.**

1.2 Formulación del problema

¿Cuál es la configuración adecuada en el marco de la visión artificial y sensores básicos para conseguir un prototipo de conducción autónomo exitoso dentro de un ambiente controlado, dinámico y reducido?

Objetivos de la investigación

2.1. Objetivo general

Hallar un algoritmo de visión e inteligencia artificial que le permita a un vehículo detectar de manera autónoma la señal de pare que genera un semáforo, posibilitando, por medio de sensores de proximidad, la actuación ante otros obstáculos que encuentre durante su recorrido.

2.2 Objetivos específicos

- ✓ Construir un prototipo de un vehículo autónomo que contenga una cámara y sensores de proximidad como mecanismos de adquisición de datos del entorno.
- ✓ Construir un ambiente controlado configurable que sirva de escenario de evaluación del desempeño de loa algoritmos con los que será equipado el prototipo.
- ✓ Afinar un conjunto de algoritmos utilizables para la consecución de autonomía en la conducción del prototipo.
- ✓ Establecer una métrica de comparación de la eficacia exhibida en los refinamientos sucesivos de los algoritmos puestos a prueba.
- ✓ Seleccionar el ajuste de algoritmos que presente un mejor desempeño frente a las exigencias colocadas en la fase de su comprobación y revisión

Justificación y delimitación.

3.1. Justificación

Este proyecto se realiza con el ánimo de mostrar una alternativa para la solución de la problemática del tránsito que hay en las ciudades contemporáneas, además, para mostrar de una

manera más aplicada a la vida cotidiana las ventajas que posee la visión artificial en el diario vivir y cómo por medio de ella se pueden implementar diferentes aplicaciones: en este caso será un vehículo que reconozca la señal de pare de un semáforo.

Además, este trabajo permite implementar de una manera innovadora todos los conocimientos que fueron adquiridos por nosotros en el proceso de formación como ingenieros de sistemas, involucrándose la combinación de diferentes asignaturas tales como programación, electrónica digital, lógica computacional e inteligencia artificial, sin los cuales sería imposible llevar a cabo la ejecución del prototipo.

3.2. Delimitación

Este proyecto se basó en información y estudios realizados en la ciudad de Bogotá durante el año 2016, documentación relacionada con la visión artificial y la evasión de obstáculos por medio de sensores de proximidad. Los anteriores fueron integrados apropiadamente en un auto a escala como prototipo de vehículo donde se instalaron los componentes físicos, como sensores, cámara web, Raspberry Pi y demás elementos electrónicos que conllevaron al funcionamiento autónomo del mismo.

Las pruebas se realizaron en un ambiente controlado, el cual contó con un semáforo replicado a partir de los comúnmente utilizados en la ciudad de Bogotá, sin la lógica completa de un ciclo debido a las características de las pruebas (por lo cual su operación es manual, en el entorno controlado), además de ocho obstáculos compuestos por cajas de aproximadamente 25

centímetros de largo, 16 centímetros de ancho y 10 centímetros de alto ubicadas a lo largo del entorno controlado.

Marco de referencia de la investigación

4.1. Marco teórico

4.1.1. La Visión artificial.

"Se puede definir como un campo de la "Inteligencia Artificial" que, mediante la utilización de las técnicas adecuadas, permite la obtención, procesamiento y análisis de cualquier tipo de información espacial obtenida a través de imágenes digitales." (FSE, 2012)

Utiliza una serie de procesos los cuales son:

- ✓ Captación de imágenes.
- ✓ Memorización de la información.
- ✓ Procesado e interpretación de los resultados.

Las funcionalidades de la visión artificial son:

- ✓ Automatizar tareas repetitivas de inspección realizadas por operadores.
- ✓ Realizar controles de calidad de productos que no era posible verificar por métodos tradicionales.
- ✓ Realizar inspecciones de objetos sin contacto físico.
- ✓ Realizar la inspección del 100% de la producción (calidad total) a gran velocidad.
- ✓ Reducir el tiempo de ciclo en procesos automatizados.

- ✓ Realizar inspecciones en procesos donde existe diversidad de piezas con cambios frecuentes de producción.

Las principales aplicaciones de la visión artificial son:

- ✓ Identificación e inspección de objetos.
- ✓ Determinación de la posición de los objetos en el espacio.
- ✓ Establecimiento de relaciones espaciales entre varios objetos (guiado de robots).
- ✓ Determinación de las coordenadas importantes de un objeto.
- ✓ Realización de mediciones angulares.
- ✓ Mediciones tridimensionales.
- ✓ Control de procesos.
- ✓ Control de calidad.
- ✓ Aplicaciones no industriales (por ejemplo, control del tráfico) (FSE, 2012)

4.1.2. Elementos importantes para la captación de imágenes.

Muestreo digital: la función obtenida tras el resultado de la medida o muestreos realizados a intervalos de tiempo espaciados regularmente, siendo el valor de dicha función un número positivo y entero. Los valores que esta función toma en cada punto dependen del brillo que presenta en esos puntos la imagen original.

Píxel: una imagen digital se considera como una cuadrícula, así cada elemento de la misma se llama “píxel” (*Picture element*). La resolución estándar de una imagen digital se puede considerar de 512 x 484 pixeles.

Nivel de grises: cuando una imagen es digitalizada, la intensidad del brillo en la escena original correspondiente a cada punto es cuantificada, dando lugar a un número denominado "nivel de gris".

Imagen binaria: es aquella que sólo tiene dos niveles de gris: negro y blanco, de esta manera, cada píxel se convierte en negro o blanco en función del llamado nivel binario o UMBRAL.

Escena: es un área de memoria donde se guardan todos los parámetros referentes a la inspección de un objeto en particular. Hacen parte de la misma la cámara utilizada, las imágenes patrón memorizadas, las tolerancias, los datos a visualizar, las entradas y salidas de control, etc.

Window (ventana de medida): es el área específica de la imagen recogida que se quiere inspeccionar.

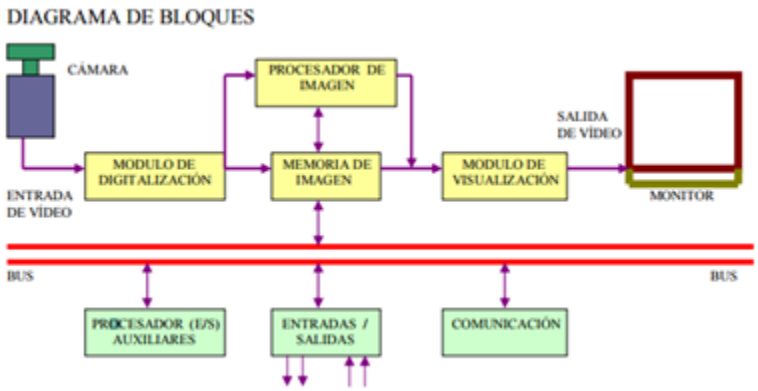


Ilustración *1* Diagrama de bloques imagen
Fuente:(V. M. Arévalo, 2004)

Módulo de digitalización: componente que transforma la señal analógica de la cámara a una señal digital.

Memoria de imagen: mecanismo de almacenamiento de la señal del módulo de digitalización.

Módulo de visualización: componente que convierte la señal digital almacenada en memoria, en señal de vídeo analógica para poder ser visualizada.

Procesador de imagen: algoritmo que procesa e interpreta las imágenes.

Módulo de entradas/salidas: componente que gestiona la entrada de captación de imagen y las salidas de control que actúan sobre dispositivos.

Comunicaciones: protocolos y circuitos empleados para la interoperabilidad de otros componentes como, por ejemplo: Vía I/O, Ethernet y RS232. (FSE, 2012)

La secuencia lógica del proceso es:

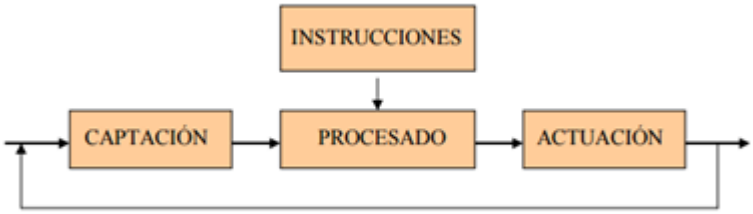


Ilustración *2* Secuencia lógica del proceso I/O
Fuente: (V. M. Arévalo, 2004)

En donde se define que:

Captación: es la obtención de la imagen visual del objeto a inspeccionar.

Instrucciones: definen el conjunto de operaciones a realizar para resolver el problema.

Procesado: se refiere al tratamiento de la imagen mediante las instrucciones aplicadas.

Actuación: implica la respuesta otorgada ante las condiciones del entorno (aparato, pieza, elemento) en función del resultado obtenido.

4.1.3 OpenCV

La librería OpenCV proporciona un marco de trabajo de alto nivel para el desarrollo de aplicaciones de visión por computador en tiempo real: estructuras de datos, procesamiento y análisis de imágenes, análisis estructural, etc.

Este marco de trabajo facilita en gran manera el aprendizaje e implementación de distintas técnicas de visión por computador, tanto a nivel docente como investigador, aislando al desarrollador de las peculiaridades de los distintos sistemas de visión.

"La librería OpenCV está dirigida fundamentalmente a la visión por computador en tiempo real. Entre sus muchas áreas de aplicación destacaría: interacción hombre-máquina (HCI 4); segmentación y reconocimiento de objetos; reconocimiento de gestos; seguimiento del movimiento; estructura del movimiento (SFM 5); y robots móviles." (V. M. Arévalo, 2004)

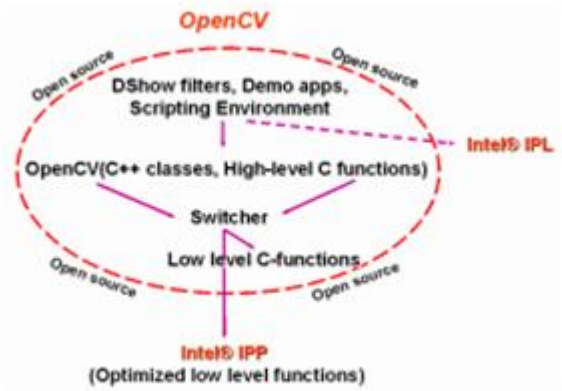


Ilustración *3* Estructura de la librería OpenCV

Fuente: (V. M. Arévalo, 2004)

"Todas estas herramientas de alto nivel hacen uso de un paquete de clases C++ y funciones C de alto nivel que utilizan a su vez funciones muy eficientes escritas en C. Concretamente, el conjunto de funciones suministradas por la librería OpenCV se agrupan en los siguientes bloques:" (V. M. Arévalo, 2004)

- ✓ Estructuras y operaciones básicas: matrices, grafos, árboles.
- ✓ Procesamiento y análisis de imágenes: filtros, momentos, histogramas.
- ✓ Análisis estructural: geometría, procesamiento del contorno, etc.

- ✓ Análisis del movimiento y seguimiento de objetos: plantillas de movimiento, seguidores (i.e. Lucas-Kanade), flujo óptico.
- ✓ Reconocimiento de objetos: objetos propios (eigen objects), modelos HMM
- ✓ Calibración de la cámara: morphing, geometría epipolar, estimación de la pose (i.e. POSIT).
- ✓ Reconstrucción tridimensional (funcionalidad experimental): detección de objetos, seguimiento de objetos tridimensionales.
- ✓ Interfaces gráficos de usuarios y adquisición de video.

La librería OpenCV proporciona varios paquetes para el desarrollo de aplicaciones de visión. Todos ellos se agrupan en librerías de C/C++ y en herramientas de scripting. Para usuarios avanzados existen herramientas como: HighGUI y CvCam, para usuarios nivel medio se encuentran herramientas como: Hawk y OpenCV Toolbox para Matlab®. HighGUI permite la escritura/lectura de imágenes en numerosos formatos (BMP, JPEG, TIFF, PxM, Sun Raster, etc.) y la captura de stream de video de capturadoras Matrox® y cámaras con drivers VFW/WDM; las ventanas HighGUI recuerdan su contenido (V. M. Arévalo, 2004), además, nos proporciona mecanismos muy fáciles de interaccionar con ellas:

Trackbars: captura la entrada del teclado y el ratón.

CvCam: proporciona una única interfaz de captura y reproducción bajo Linux y Win32.

Callbacks: para la gestión de stream de vídeo o ficheros AVI y un mecanismo fácil para implementar visión estéreo con dos cámaras USB o un estéreo-cámara.

Hawk: es un entorno visual con el intérprete ANSI C EiC como núcleo; soporta plugins; proporciona soporte para OpenCV, IPL y HighGUI vía plugin; y soporte de video.

La librería OpenCV proporciona un toolbox para Matlab® que se caracteriza por lo siguiente:

- ✓ Utiliza tipos nativos de Matlab® (matrices, estructuras).
- ✓ Compatibilidad con la Image Processing Toolbox.

Detectar un objeto basándonos en su color:

Cambiar el color space de BGR a HSV, de este modo se puede establecer un filtro para obtener sólo aquellas secciones de la imagen cuyo color se encuentre en el rango de color indicado, en HSV es más fácil establecer este rango.

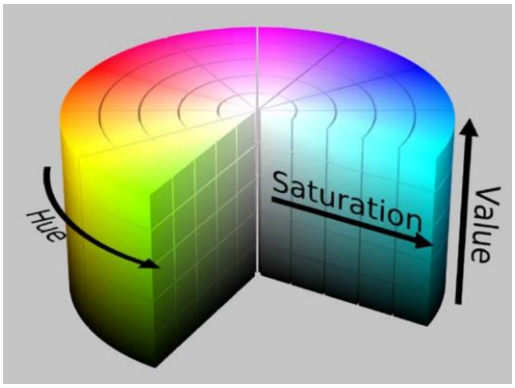


Ilustración *4* Rango de color HSV

Fuente: (Marin, 2016)

Para realizar la conversión se usa la función cvtColor (src, dst, CV_BGR2HSV), en la cual se indica la imagen fuente, destino y el tipo de conversión que puede ser: (Marin, 2016)

COLOR_BGR2HSV, COLOR_RGB2HSV, COLOR_HSV2BGR, COLOR_HSV2RGB.

Para filtrar sólo los colores que se encuentren en el rango indicado se utiliza la función inRange, por ejemplo filtrar el color azul, por lo que los rangos superior e inferior se definen de la siguiente manera: inferior {Scalar (110, 50, 50)}, superior {Scalar (130, 255, 255)} en HSV color space. (Marin, 2016)

Código:

```
"// obtener frame de la webcam
cap >> img;
// convertir imagen RGB a HSV
cvtColor(img, hsv, CV_BGR2HSV);

// aplicar filtro para color deseado
inRange(hsv, Scalar(110, 50, 50), Scalar(130, 255, 255), binary);
// aplicar transformaciones morfológicas (extrae la región de interés)
Mat element = getStructuringElement(MORPH_RECT, Size(15, 15));

   erode(binary, binary, element);
dilate(binary, binary, element);"
```

(Marin, 2016)

Este es el resultado de filtrar el color azul, se obtiene una imagen binaria donde el color blanco representa las áreas que contienen el color filtrado. (Marin, 2016)

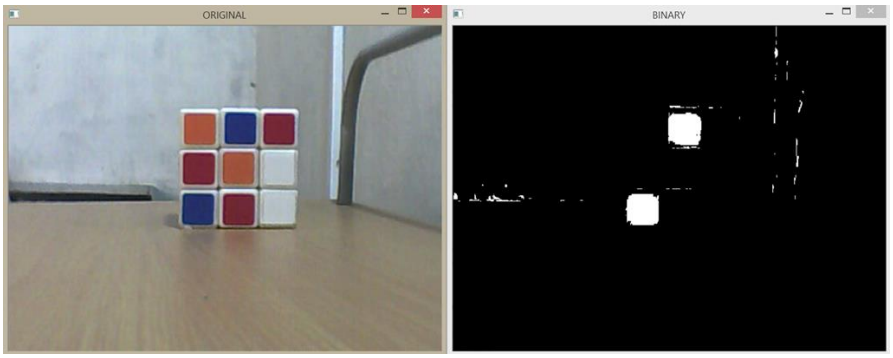


Ilustración *5* Resultado Filtro

Fuente: (Marin, 2016)

Para descartar lo demás se debe usar las transformaciones erode y dilate, configurados para buscar elementos rectangulares (MORPH_RECT) en la imagen binaria, MORPH_ELLIPSE para figuras circulares. (Marin, 2016)

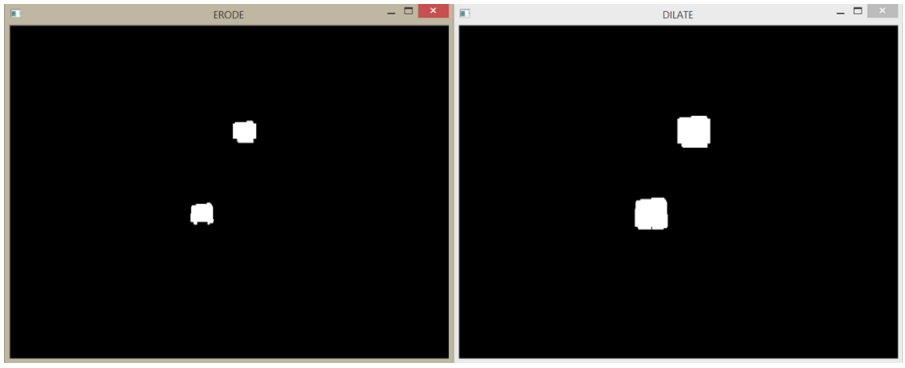


Ilustración *6* Identificación imagen binaria

Fuente: (Marin, 2016)

Lo siguiente es encontrar las coordenadas donde se encuentra la región de interés, con búsqueda de contornos, “findContours (binary, contours, CV_RETR_EXTERNAL, CV_CHAIN_APPROX_SIMPLE), binary es la imagen binaria que se creó, contours es la

variable tipo vector<vector<Point>> donde se almacenan los contornos encontrados, CV_RETR_EXTERNAL indica que solo se necesitan los contornos externos.” (Marin, 2016)

“Código:

```
   // Buscar contornos en la imagen binaria
vector< vector<Point> > contours;
findContours (binary, contours, CV_RETR_EXTERNAL, CV_CHAIN_APPROX_SIMPLE);

// dibujar todos los contornos encontrados
drawContours (binary, contours, -1, Scalar (255), CV_FILLED);”
```

(Marin, 2016)

Para finalizar se utiliza la información de los contornos para dibujar un rectángulo sobre la imagen original y marcar el color seleccionado, agregar las coordenadas correspondientes, estas se obtienen con la función “Rect r = boundingRect (contour); la función rectangle y putText dibujan el rectángulo y el texto.” (Marin, 2016)

“Código:

```
   // dibujar rectángulo y texto con coordenadas (x, y)
for (vector<Point> contour : contours) {
    Rect r = boundingRect(contour);
    rectangle(img, r.tl(), r.br(), CV_RGB(255, 0, 0), 2, CV_AA, 0);

    Point center(r.x + (r.width / 2), r.y + (r.height / 2));

    ostringstream str;
    str << center.x << "," << center.y;

    putText(img, str.str(), center, FONT_HERSHEY_COMPLEX_SMALL, 0.60, CV_RGB(0, 255, 0),
1, CV_AA);
}”
```

(Marin, 2016)

El resultado final es:

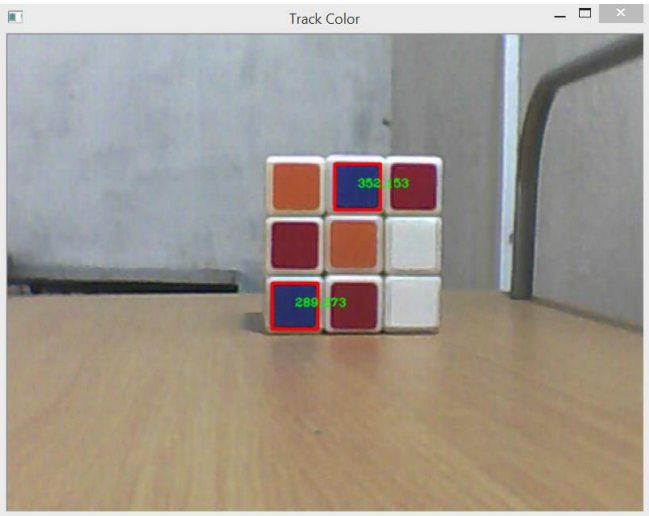


Ilustración *7* Coordenadas región de interés

Fuente: (Marin, 2016)

Acentuar color:

"Se usa la siguiente ecuación para calcular los valores mínimos y máximos aceptados, los valores que se encuentren dentro de este rango son procesados como un color válido (no cambian su color), los demás son convertidos a escala de grises". (Marin, 2016)

$$h_1 = (h - \frac{r}{2} + 360) mod 360$$

$$h_2 = (h + \frac{r}{2} + 360) mod 360$$

Ilustración *8* Formula calculo valores

Fuente: (Marin, 2016)

La variable h establece el color deseado a acentuar y r establece en rango de valores aceptable.

Lo primero que se hace es convertir la imagen a HSV, crear el cv: Mat para almacenar el resultado, debe ser del mismo tamaño y tipo que la imagen original, y finalmente calcular los valores h1 y h2. (Marin, 2016)

Código:

```
"cvtColor (src, hsv, CV_BGR2HSV);

int channels = src.channels ();
int nRows = src.rows;
int nCols = src.cols * channels;

dst.create (src.rows, src.cols, src.type ());

uchar h1 = (h - (r / 2) + 360) % 360;
uchar h2 = (h + (r / 2) + 360) % 360;"
```

(Marin, 2016)

Luego se recorre cada uno de los píxeles de la imagen, se toma el valor de la componente H y se compara con el rango calculado.

Código:

```
"for (int i = 0; i < nRows; i++) {
   ptr_dst = dst.ptr<uchar> (i);
   ptr_src = src.ptr<uchar> (i);
   ptr_hsv = hsv.ptr<uchar> (i);
```

```
    for (int j = 0; j < nCols; j += 3) {
        in_range_color = false;

        // obtener el valor hue (H)
        uchar H = ptr_hsv[j];

   // verificar si se encuentra en el rango indicado
        if (h1 <= h2) {
            if (H >= h1 && H <= h2)
                in_range_color = true;
        }
        else if (H >= h1 || H <= h2)
            in_range_color = true;

        // si está en el rango conservar el color
        // en caso contrario convertir a grises
        if (in_range_color == true) {
            ptr_dst[j + 0] = ptr_src[j + 0];
            ptr_dst[j + 1] = ptr_src[j + 1];
            ptr_dst[j + 2] = ptr_src[j + 2];
        }
        else {
            // conversión a grises usando en método de promedio
            uchar gray = (ptr_src[j] + ptr_src[j + 1] + ptr_src[j + 2]) / 3;
            ptr_dst[j + 2] = ptr_dst[j + 1] = ptr_dst[j] = gray;
        }
    }
}”
```

(Marin, 2016)

Para terminar la aplicación se crea un par de trackbar, el primero para cambiar el valor h y el segundo para el valor r, para actualiza la imagen. (Marin, 2016)

4.1.3 Elementos que componen un vehículo auto-conducido (*Self-driving vehicle*)

A parte de los sistemas de software que llevan meses mejorando desde la compañía norteamericana, los componentes hardware que hacen que sea posible el vehículo sin conductor son, entre otros, los siguientes:

Sensores LIDAR 6 de rotación en el techo, que escanean a más de 200 metros en todas las direcciones para generar un mapa tridimensional preciso del entorno del coche. (Gabinete Técnico de la Guardia, 2014)

Sensor de estimación de posición montado sobre la rueda trasera izquierda, mide los pequeños movimientos realizados por el coche y ayuda a localizar con exactitud su posición en el mapa.

Detector de rayos láser que averiguan cuál es el tráfico y lo descifran para poder moldear su forma de conducir.

Cuatro sensores de radar de Standard, tres en el frente y uno en la parte trasera, ayudan a determinar las posiciones de los objetos distantes.

Cámara montada cerca del retrovisor para detectar las señales de tránsito como semáforos, límites de velocidad y cuenta a la vez con ordenadores a bordo del vehículo para reconocer obstáculos como peatones y ciclistas, etc.

4.1.4. Antecedentes.

- ✓ **Sistemas de transporte inteligentes**

Surgen en la década de los noventa, con el objetivo principal de aumentar la eficacia y seguridad del transporte. Se presenta como una combinación de distintos sistemas avanzados de información, comunicación y control aplicados a vehículos e infraestructuras.

A inicio de los años 60 iniciaron los desarrollos de automática en vehículos, conocidos como sistemas avanzados para control de vehículos AVCS, paralelamente a los avances en tecnologías de telecomunicaciones, informática hasta llegar a iniciar a buscar tomar el control de los vehículos. Se basa en la utilización de la sinergia en diferentes entornos virtuales, informática, control con el fin de mejorar la seguridad en la conducción, además, pretende utilizar de forma eficiente los recursos de los vehículos, disminuir la congestión en las grandes ciudades, contaminación, y tiempos de transporte.

La conducción autónoma de vehículos se encuentra aún en fase de investigación y desarrollo, actualmente no se encuentran vehículos que circulen por las vías de una ciudad de una forma totalmente autónoma.

- ✓ **El proyecto GOOGLE DRIVERLESS CAR**

Empresas como google están apostando fuertemente a la creación de vehículos autónomos, el proyecto google driverless car, se ha realizado conjuntamente con el laboratorio de inteligencia artificial de Stanford. El proyecto cuenta con siete vehículos. Google ha sumado más de 1600 Km de recorridos en condiciones reales.

El automóvil sin conductor de Google (Google driverless car) es un proyecto que comenzó en 2009 con el objetivo de desarrollar la tecnología necesaria para crear coches sin conductor, que circulen de forma autónoma. Actualmente el líder del proyecto es el ingeniero alemán de Google Sebastian Thrun 1, director del Stanford Artificial Intelligence Laboratory 2 y co-inventor de Google Street View 3. (Centro de Análisis y Prospectiva, 2014)

El equipo de Thrun en Stanford creó el vehículo robótico Stanley, que fue el ganador del DARPA 4 Grand Challenge 5 en 2005, un galardón otorgado por el Departamento de Defensa de los Estados Unidos y dotado con un premio de dos millones de dólares. El equipo encargado del proyecto estaba formado por 15 ingenieros de Google, entre los que se encontraban Chris Urmson, Mike Montemerlo y Anthony Levandowski, quienes habían trabajado en el DARPA Grand and Urban Challenges. Este coche es capaz de conducir autónomamente por ciudad y por carretera, detectando a otros vehículos, señales de tráfico, peatones, etc. (Centro de Análisis y Prospectiva, 2014)

Google no solo ha dotado al Toyota Prius de su tecnología, también lo está probando desde hace años en el Audi TT y en el Lexus RX 450h obteniendo muy buenos resultados. Desde que en el Estado de Nevada se aprobara la ley sobre vehículos "self-drive" el gigante de la tecnología ha estado mejorando durante estos años el software en sus calles. Desde hace aproximadamente un año, los 24 Lexus RX450h equipados con sensores de Google han estado circulando también por las calles de Mountain View (California, EE.UU.) donde el gigante tecnológico tiene su sede. Han mejorado el software para que pueda detectar cientos de objetos distintos de forma simultánea como peatones, autobuses, una señal de alto sostenida por una de cruce o un ciclista haciendo gestos que indican un posible giro. Y es que consideran que un vehículo "self-drive" debe y puede prestar atención a todos estos factores mucho mejor que un humano, puesto que las personas pueden estar cansadas o distraídas. (Centro de Análisis y Prospectiva, 2014)

- ✓ **Proyecto AUTOPIA 7.**

El Consejo Superior de Investigaciones Científicas (CSIC) y la Universidad Politécnica de Madrid llevan desarrollando desde hace ya más de 15 años la tecnología para hacer realidad los vehículos „self-drive", dentro del programa Autopia 7**.** (Gabinete Técnico de la Guardia, 2014)

Ya se ha probado por las carreteras de Madrid uno de los prototipos llamado Platero, un Citroën C3. Platero ha circulado por sí mismo a través de carreteras abiertas al tráfico desde San Lorenzo de El Escorial hasta Arganda del Rey, primero por ciudad y luego por autovía. Eso sí, lo ha hecho detrás de un coche guía y escoltado por la Agrupación de Tráfico de la Guardia Civil. (https://www.motorpasion.com, 2012)

Platero recorrió en total unos 100 km de distancia a una velocidad media resultante de unos 60 km/h. La velocidad máxima a la que puede circular, comprobada en circuito sin tener incidentes, ha sido de 109 km/h. (https://www.motorpasion.com, 2012)

El computador que va en el maletero del coche es la base del sistema de control del sistema de conducción autónoma y utiliza una estrategia de mando basada en la lógica borrosa que permite simular el comportamiento de un conductor humano. (Gabinete Técnico de la Guardia, 2014)

Es fundamental el método de navegación y posicionamiento DGPS, GPS diferencial, con una precisión de 0,5 m. Éste se tiene que complementar para mayor precisión con un sistema de radio con el que recibir correcciones diferenciales, o bien con un sistema de comunicación inalámbrica local Ethernet, con la misma función.

Se consigue una precisión de centímetro. Este sistema inalámbrico de comunicación entre vehículos también permite conocer la posición de los coches que están en las cercanías, y que estén equipados con el sistema, CarTo Car 8 o SARTRE 9. Ésta fue la función desempeñada por el coche guía que iba delante y que sí llevaba conductor. (https://www.motorpasion.com, 2012)

- ✓ **Proyecto AUTONOMOUS LAND VEHICLE (ALV)**

En Estados Unidos el Departamento de Defensa realiza sus propios experimentos en el llamado Autonomous Land Vehicle (ALV) y lleva a cabo la primera demostración de vehículos capaces de seguir un camino utilizando LIDAR, visión artificial y control robótico autónomo para controlar un vehículo robótico hasta los 30 km/h. En 1987, consiguen que el ALV sea capaz de viajar autónomamente fuera de carretera mediante un sistema de navegación autónomo basado en sensores. El nuevo modelo es capaz de viajar 600m a 3 km/h en terrenos complejos con fuertes pendientes, barrancos, grandes rocas y vegetación. (Automóvil, 2015)

- ✓ **Proyecto ARGO**

El proyecto ARGO consigue completar una ruta de 2000 km a lo largo de autopistas del norte de Italia durante 6 días y con una velocidad media de 90 km/h. Lo sorprendente del proyecto fue que el vehículo consiguió funcionar en modo totalmente automático el 94% del tiempo. El automóvil contaba sólo con dos cámaras detectoras de blancos y negros de bajo costo y utilizaban algoritmos de visión estereoscópica para entender el entorno (se guiaba gracias a que la carretera es negra y las líneas delimitantes son blancas) en contraposición a la aproximación laser/radar de los proyectos anteriores en el mismo campo. (Automóvil, 2015)

- ✓ **Proyecto PARKSHUTTLE**

ParkShuttle en Rotterdam, un sistema de minibuses que funcionan sin conductor.

- ✓ **Proyecto de la UC3M**

La Universidad Carlos III de Madrid cuenta con un proyecto propio en la carrera por el vehículo autónomo. (https://www.tecnocarreteras.es, 2011)

El proyecto cuenta con 3 modelos de vehículos: el IVVI, el iCab y el IVVI 2.0.

- o **Proyecto IVVI**

Universidad Carlos III de Madrid creo es el homólogo al coche desarrollado por Google.

Cuenta con un sistema estéreo blanco y negro, con cámaras de barrido progresivo para poder captar imágenes en movimiento y evitar así los problemas inherentes al vídeo entrelazado, además de una cámara a color para otros temas como la detección de señales de tráfico. Para detectar obstáculos como peatones u otros vehículos en condiciones de visibilidad adversas, cuenta con una cámara de infrarrojo lejano capaz de distinguir objetos por el calor desprendido. En cuanto al interior, el sistema cuenta con tecnología de monitorización del estado del conductor, empleando para ello una cámara infrarroja. El cerebro del coche viaja en el maletero, en donde se hallan dos PC encargados del procesamiento de los sistemas de visión por computador. (https://www.tecnocarreteras.es, 2011)

Para conocer el estado del vehículo, éste incorpora además un sistema GPS. Entre las tecnologías que se incorporan en este proyecto están:

- **Sistema de alerta ante el alejamiento involuntario del carril:** trabaja con imágenes que simulan una vista aérea de la vía y se acompañan de algoritmos de calibración automática.

Se distinguen los carriles por la discontinuidad de los niveles de gris de la imagen. (https://www.tecnocarreteras.es, 2011)

- **Sistema de detección de peatones:** dada la enorme diversidad de apariencia de los peatones y los cambios en la forma de estos entre imágenes sucesivas, se emplea un algoritmo basado en los Contornos Activos (técnica empleada para delinear el contorno de un objeto en una imagen 2D ruidosa), inicializados con los resultados de un sistema estéreo y teniendo en cuenta la simetría de las formas de los peatones. (https://www.tecnocarreteras.es, 2011)
- **Sistema de control del conductor:** midiendo parpadeo de los ojos, movimiento de la cabeza y apertura de la boca para detectar el grado de atención.

❖ **Sistema de reconocimiento de señales de tráfico:** emplea algoritmos genéticos y templados simulados como algoritmos de búsqueda para encontrar las señales presentes en las imágenes captadas por una cámara a color. (Iternova, 2011)

- **Sistema de control de velocidad variable:** el reconocimiento del vehículo delantero se realiza mediante análisis de imágenes en lugar de por sensores radar. El sistema se fusiona con el módulo de detección de carriles para delimitar la búsqueda. (Iternova, 2011)

o **Proyecto IVVI 2.0**

Mejora el diseño del vehículo e incorpora como novedad que los sistemas de ayuda a la conducción sólo informan al conductor cuando éste esté en una situación real de peligro. Además, se incorpora una sonda CANbus (que obtiene información del funcionamiento del vehículo) así como un sistema GPS-IMU (que informa sobre posición y velocidad del vehículo) para determinar el estado del vehículo. (Fuentes, 2012)

El tercer vehículo, el iCab, es un pequeño vehículo eléctrico de golf que es controlado por un ordenador embarcado. La función de estos vehículos será llevar de forma autónoma a los visitantes del Campus, por lo que la autonomía estará limitada al entorno del Campus y a entornos previamente conocidos. (Fuentes, 2012)

Mercedes es uno de los fabricantes que cuentan con una tecnología más avanzada en materia de conducción autónoma y de hecho el actual Clase S, el vehículo más avanzado del mercado, dispone de muchos sistemas para hacer posible la conducción sin conductor. Con esta licencia para probar sus coches en California la marca alemana da un importante paso adelante hacia la producción en serie de coches de conducción autónoma, al menos en ciertas condiciones. Lo más importante es que además de en Alemania, donde ya se prueban este tipo de coches en unas condiciones de carreteras más estrechas y con los semáforos antes del cruce, en Estados Unidos las circunstancias son completamente distintas, con carreteras mucho más anchas y con semáforos instalados en el otro lado del cruce. Y los coches de conducción autónoma deben funcionar en todos los países con independencia de las normas vigentes. (https://www.elconfidencial.com, 2014)

En 2010, con motivo de la Exposición Universal de Shanghai, cuatro furgonetas eléctricas realizaron el primer viaje intercontinental sin conductor, saliendo desde Italia y alcanzando China.

Según las propuestas de los grandes automovilísticos, que varían entre ellos, pero en algo sí coinciden: antes del 2020 circularán los primeros coches autónomos comerciales.

Nissan ha anunciado que en 2020 comercializará su primer modelo de coche autónomo. El viaje será progresivo y por etapas. De hecho, algunos sistemas de pilotaje asistido, como el control electrónico de estabilidad, comercializado bajo las siglas ESP, considerado uno de los

avances tecnológicos en materia de seguridad más relevantes, se ha incorporado con absoluta normalidad. Otros sistemas como el aparcamiento dirigido, el sistema de aviso de ángulos muertos o de frenado automático se están incorporando.

En el CES (Consumer Electronics Show), Mercedes paseó su prototipo FO 15, que además de ser un híbrido basado en hidrógeno y batería eléctrica, usa sensores 3D para liberar al conductor. En un alarde de altiva suficiencia, el vehículo puede rotar sus asientos delanteros como si se tratara de un vagón de tren. (https://www.elconfidencial.com, 2014)

También Ford ha anunciado que está probando coches autónomos en carretera, mientras que el coche de BMW viajó desde San Francisco sin conductor a bordo. (Bejerano, 2015) Quien más quien menos, los fabricantes esperan que el intercambio de datos sobre localización, velocidad y destino podrá canalizar el tráfico vial, pero les falta sortear, entre otros obstáculos, la inquietud del usuario por su privacidad.

4.2. Marco conceptual

Durante la investigación se requiere conocer e identificar diferentes conceptos, elementos y variables que ayudaran al buen desarrollo del prototipo del vehículo autónomo como son:

4.2.1. Acciones básicas.

- ✓ **Girar:** movimiento que realiza el vehículo haciéndole dar vueltas sobre sí mismo o alrededor del obstáculo.
- ✓ **Marcha**: posición de un vehículo que determina la velocidad y dirección del movimiento
- ✓ **Esquivar:** movimiento que realiza el vehículo para evitar un golpe o esquivar un obstáculo.

- ✓ **Girar a la Izquierda:** dirección que toma el automóvil en el sentido contrario a las manecillas del reloj.
- ✓ **Girar a la derecha:** dirección que toma el automóvil en el sentido de las manecillas del reloj.
- ✓ **Marcha Adelante:** es dirección que toma el vehículo respecto a un lugar que se encuentra delante de él cual toma como referencia.
- ✓ **Marcha Atrás:** es dirección que toma el vehículo respecto a un lugar que se encuentra detrás de él cual toma como referencia.

4.2.2. Conceptos de las acciones a realizar.

- ✓ **Como girar:** por medio del algoritmo o código de programación y según los sensores de proximidad, detecta si delante del vehículo se encuentra algún obstáculo, evalúa si al girar a la izquierda o derecha se encuentra también algún obstáculo para realizar el giro, el cual por medio de una señal eléctrica hace mover los motores para realizar el cambio de dirección.
- ✓ **Como reducir o aumentar marcha:** por medio de la cámara web y el algoritmo programado, el vehículo detectara el color que el semáforo está mostrando, si se encuentra en verde el vehículo se pondrá en marcha, si se encuentra en rojo o en amarillo el automóvil bajara la marcha hasta detenerse.
- ✓ **Avanzar o retroceder:** por medio del código de programación, los sensores de proximidad, y la cámara Web el automóvil tendrá una marcha hacia delante siempre y cuando no se encuentre con algún obstáculo de lo contrario, si su lado izquierdo y lado derecho se

encuentran ocupados por algún objeto, dará marcha hacia atrás y una vez libre la vía volverá a tomar su marcha hacia delante al punto designado.

4.2.3. Definiciones generales.

- ✓ **Motor:** máquina capaz de hacer funcionar el sistema el vehículo haciéndolo girar de derecha a izquierda y dirigirse hacia adelante o hacia atrás, el cual funciona a partir de una corriente eléctrica.
- ✓ **Sensor de proximidad:** elemento que detecta los obstáculos próximos al vehículo.
- ✓ **Puente en H:** circuito electrónico que permite a un motor eléctrico DC gire en ambos sentidos, además, de ir hacia delante y hacia atrás.
- ✓ **Raspberry pi:** es un ordenador de placa simple.
- ✓ **Cable:** hilo metálico o conjunto de hilos que sirve como conductor; puede tener una envoltura aislante.
- ✓ **Cámara:** cámara de video que se conecta a una computadora y permite que los videos o fotos pueden verse instantáneamente.
- ✓ **Batería:** aparato electromagnético capaz de acumular energía eléctrica y suministrar.
- ✓ **Memoria SD:** es un dispositivo en formato de tarjeta de memoria para dispositivos portátiles, el cual cumple la función de almacenar información.
- ✓ **Sistema Operativo:** conjunto de directrices y software que controlan los procesos de una computadora.
- ✓ **Obstáculo:** elemento, o situación que impide que el vehículo pueda continuar con su camino fácilmente sin necesidad de cambiar su marcha.

- ✓ **Semáforo:** dispositivo de señalización los cuales indican a vehículos y personas cuando pueden transitar y cuándo deben detenerse según indicadores de color rojo, amarillo y verde.
- ✓ **Camino:** terreno utilizado para ir de un lugar a otro. Puede encontrarse señalizada y delimitada o no.
- ✓ **Puertos (USB, HDMI):** medio por el cual podemos conectar dispositivos externos al sistema.
- ✓ **Llantas:** es una pieza circular, generalmente de metal, situada en el centro de una rueda y sobre la que se coloca un neumático y que va unida al eje del vehículo.

4.3. Marco legal

Debido a que los avances sobre este tema son recientes no se cuenta con una normativa completa acerca de la movilidad de los vehículos autónomos para poder circular en las calles, sin embargo, en varios países se han venido implementando diferentes controles para la evaluación de seguridad de estos vehículos entre los cuales están:

En EE.UU. se encuentra una política llamada HAV la cual consta de 15 puntos que se aplican al diseño, desarrollo, pruebas e implementación del vehículo. Con esta guía se pretende generar un diseño robusto y seguro; permitiendo diferentes metodologías para la detección de elementos y la configuración de los posibles eventos que ocurran. (Unidad Editorial Información Económica S.L., s.f.)

En España, el único movimiento regulatorio es el generado en la Dirección General de Tráfico (DGT) al aprobar la realización de pruebas con vehículos de conducción automatizada en vías

abiertas, con lo cual se ha facilitado transitar entre Vigo y Madrid con un vehículo autónomo. (Unidad Editorial Información Económica S.L., s.f.)

En Alemania, el parlamento alemán ha aprobado una ley que autoriza a los fabricantes probar sus sistemas de conducción autónoma en su país. Por el momento esta ley tiene unas limitaciones, como la presencia de una persona detrás del volante por si surgiese cualquier imprevisto, la inserción de una caja negra que se encargue de registrar todos los detalles del trayecto realizado por el vehículo autónomo. (Grupo Luike, 2017)

5. Tipo de investigación

Dada la naturaleza del problema planteado, de define esta investigación como experimental ya que es necesario la evaluación e identificación de los obstáculos y semáforos que el vehículo a desarrollar debe interpretar para realizar la acción según las reglas de conocimiento aplicadas, las cuales se deben probar para conocer su posible resultado.

Por esto se elabora una maqueta donde se simula un cruce vehicular controlado para la experimentación del prototipo.

6. Diseño metodológico

Se inicia con la investigación de los posibles algoritmos y proyectos que se están usando en el mundo y de esta forma poder tener un panorama más claro en la implementación del prototipo.

Posteriormente es importante que realizar una evaluación detallada de los componentes que se usan en el diseño y elaboración de vehículo autónomo y de esta manera garantizar el correcto funcionamiento de dichos elementos.

Una vez elegidos dichos componentes es necesario realizar la configuración de estos para la iniciar prueba de funcionalidad del vehículo en una prueba individual.

Cuando se verifique que las partes que se instalaron en el prototipo funcionan correctamente, se generaran de los algoritmos de visión artificial y proximidad para la detección todos los obstáculos, una vez realizado cada algoritmo se hace una prueba de cada uno por separado.

Luego de realizar las pruebas de los algoritmos, se inicia con la construcción del prototipo con los componentes electrónicos, de hardware y de software, probando su funcionalidad en conjunto.

Para realizar la prueba del prototipo final se debe construir una maqueta del ambiente controlado para realizar los experimentos con el vehículo autónomo ya construido, en cual se prueba y se busca que el prototipo cumpla los objetivos previstos.

Para finalizar se genera una evaluación del prototipo, y así, realizar los ajustes del vehículo autónomo hasta que supere los objetivos que se establecieron inicialmente.

7. Desarrollo de la Investigación

El proyecto “Diseño de un prototipo funcional de vehículo autónomo por medio de visión artificial y sensores de proximidad, implementado en PYTHON sobre arquitectura de RASPBERRY PI”, se desarrolla en el ámbito educativo durante el progreso de la carrera de ingeniería de sistemas en la Universidad ECCI.

Se da inicio a este proyecto con la siguiente pregunta:

¿Cuál es la configuración adecuada en el marco de la visión artificial y sensores básicos para conseguir un prototipo de conducción autónomo exitoso dentro de un ambiente controlado, dinámico y reducido?

Se realiza un estudio e investigación previa sobre la temática de vehículos autónomos en el mundo para conocer y tener referencias base para dar inicio a la estructura y configuración que se debía utilizar para generar el prototipo.

Se estableció que era necesario realizar y probar diferentes algoritmos para la identificación y cálculos de distancias de obstáculos, además, de un componente visual que permitiese la identificación de los colores regulares que utiliza una señal de transito como lo es un semáforo vehicular.

Se toman un código base inicial el cual fue elaborado de forma básica durante la clase de inteligencia artificial tomada en la universidad UECCI, y con este se generan tres etapas en donde se comienza a la actualización y modificación del código, componentes y características iniciales del vehículo autónomo para poder cumplir con los objetivos del proyecto planteado.

El proyecto se llevó a cabo en 3 etapas:

7.1. Etapa uno

- ✓ Se llevaron a cabo en el año 2016 una serie de reuniones con el grupo de trabajo para coordinar los detalles y las funciones que cada uno tendría dentro el proyecto; dichas reuniones se realizan en la sede principal de la Universidad ECCI
- ✓ Se instaló el sistema operativo que utiliza la Raspberry Pi y así dar inicio a las pruebas de algoritmos guía para verificar los elementos como sensores y cámara web.
- ✓ Se establecieron las conexiones de los componentes tanto en la Raspberry Pi y en los motores de la siguiente manera:

- o Conectar Puente H:

  Conectores derechos:
  Cable Naranja
  Cable Azul
  Conectores izquierdos:
  Cable Amarillo
  Cable Verde

  Conectores Raspberry:
  Izq- Der:
  Gris Blanco Cafe Rojo

- o GND Pin 6 (verde)

- o Sensor delantero:
  Izq a Der:
  Amarillo Pin 9 - Azul-Blanca Pin 10 - Naranja Pin 12 - Verde Pin 4
  Gnd Echo GIO 15 Trigger GIO 18 5V

- o Sensor tracero:
  Izq a Der:
  Cafe Pin14 - Morado Pin 11 - Azul Pin 13 - Rojo Pin 2
  Gnd Echo GIO 17 Trigger GIO 27 5V

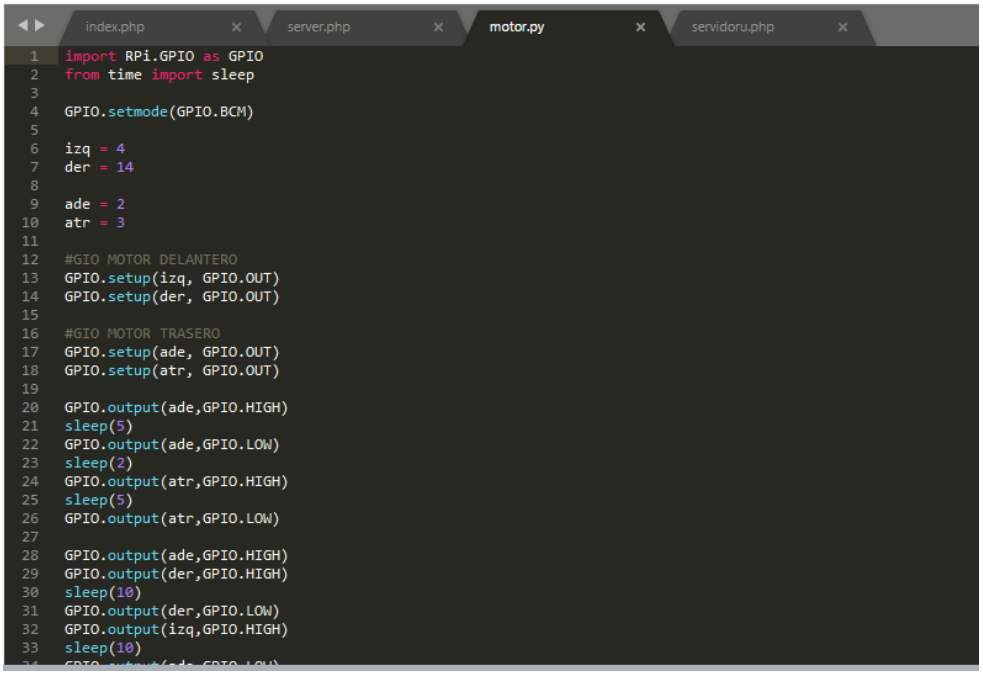


Ilustración *9* Código preliminar motores vehículo

Fuente: (Alumnos, Imagen, 2017)

✓ Se verificó la conexión y funcionamiento de los motores utilizados en el primer prototipo:

“import RPi.GPIO as GPIO
from time import sleep

GPIO.setmode(GPIO.BCM)

izq = 4
der = 14

ade = 2
atr = 3

#GIO MOTOR DELANTERO
GPIO.setup(izq, GPIO.OUT)
GPIO.setup(der, GPIO.OUT)

#GIO MOTOR TRASERO
GPIO.setup(ade, GPIO.OUT)

```
GPIO.setup(atr, GPIO.OUT)

GPIO.output(ade,GPIO.HIGH)
sleep(5)
GPIO.output(ade,GPIO.LOW)
sleep(2)
GPIO.output(atr,GPIO.HIGH)
sleep(5)
GPIO.output(atr,GPIO.LOW)

GPIO.output(ade,GPIO.HIGH)
GPIO.output(der,GPIO.HIGH)
sleep(10)
GPIO.output(der,GPIO.LOW)
GPIO.output(izq,GPIO.HIGH)
sleep(10)
GPIO.output(ade,GPIO.LOW)
GPIO.output(izq,GPIO.LOW)
```

GPIO.cleanup()" (Alumnos, Codigo, 2016)

- ✓ Ensamblado del primer prototipo del vehículo autónomo con los diferentes componentes que los integran.

Ilustración *10* Diagrama primer prototipo

Fuente: (Alumnos, Imagen, 2017)

- ✓ Debido a la cantidad de componentes (sensores, motores, Raspberry Pi, batería, cableado), fue necesario migrar dichos componentes a una estructura más grande y robusta que permitiera el movimiento del prototipo.

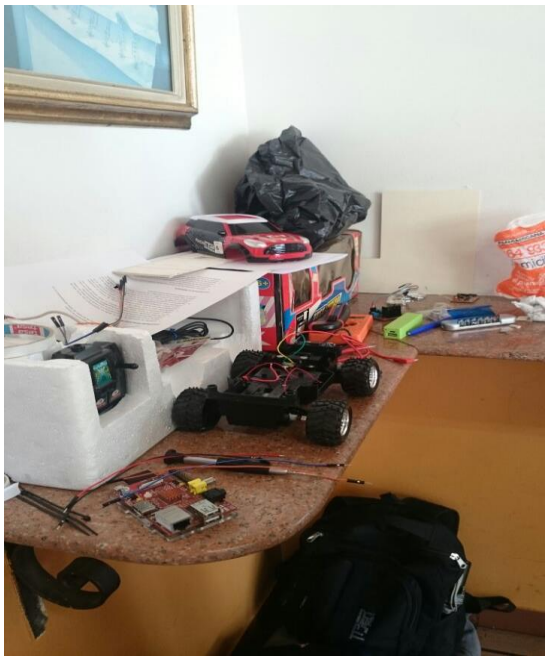


Ilustración *11* Instalación de componentes I

Fuente: (Alumnos, Imagen, 2017)

Ilustración *12* Instalación de componentes II

Fuente: (Alumnos, Imagen, 2017)

7.2. Etapa dos

- ✓ Se evaluó el código que se estaba utilizando para realizar la prueba inicial de los sensores por separado:

```
“import RPi.GPIO as GPIO
     import time
     import threading

     GPIO.setmode(GPIO.BCM)
     GPIO.setwarnings(False)

     class Sensor:
          __TRIG = 0
          __ECHO = 0

          def __init__(self, trig, echo):
               self.__TRIG = trig
               self.__ECHO = echo
               GPIO.setup(self.__TRIG,GPIO.OUT)
               GPIO.setup(self.__ECHO,GPIO.IN)

          def run(self):
               try:
                    while True:
                         GPIO.output(self.__TRIG, False)
                         print   "Waitng   For   Sensor   To
Settle"

                         time.sleep(2)
                         GPIO.output(self.__TRIG, True)
                         time.sleep(0.0001)
                         GPIO.output(self.__TRIG, False)

                         while GPIO.input(self.__ECHO)==0:
                              pulse_start = time.time()

                         while GPIO.input(self.__ECHO)==1:
                              pulse_end = time.time()
```

```
                              pulse_duration   =   pulse_end   -
 pulse_start
                              distance = pulse_duration * 17150
                              distance = round(distance, 2)

      print              "Distance:              {distance}cm
      {thr}".format(distance=(distance        -        0.5),
      thr=threading.currentThread().getName())
                 except KeyboardInterrupt:
                      print "  Quit"
                      GPIO.cleanup()

      sensor_1 = Sensor(18,15)
      sensor_2 = Sensor(27,17)

 s2   =   threading.Thread(target=sensor_2.run,   name="Sensor
      trasero")

      s1.start()
      s2.start()”
```

(Alumnos, Codigo, 2016)

- ✓ Se evaluó el código que se estaba utilizando para realizar la prueba inicial de la cámara para la detección de imágenes:

```
“import numpy as np
import cv2
import time

# Cargamos el vdeo
camara = cv2.VideoCapture(0)

# Inicializamos el primer frame a vacio.
# Nos servira para obtener el fondo
fondo = None

# Recorremos todos los frames
while True:
```

```
        # Obtenemos el frame
        (grabbed, frame) = camara.read()

        # Si shemos llegado al final del video salimos
        if not grabbed:
            break

        # Convertimos a escala de grises
        gris = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

        # Aplicamos suavizado para eliminar ruido
        gris = cv2.GaussianBlur(gris, (21, 21), 0)

        # Si todavia no hemos obtenido el fondo, lo obtenemos
        # Sera el primer frame que obtengamos
        if fondo is None:
            fondo = gris
            continue

        # Calculo de la diferencia entre el fondo y el frame actual
        resta = cv2.absdiff(fondo, gris)

        # Aplicamos un umbral
        umbral      =      cv2.threshold(resta,      25,      255,
cv2.THRESH_BINARY)[1]

        # Dilatamos el umbral para tapar agujeros
        umbral = cv2.dilate(umbral, None, iterations=2)

        # Copiamos el umbral para detectar los contornos
        contornosimg = umbral.copy()

        # Buscamos contorno en la imagen
        contornos,                  hierarchy                  =
cv2.findContours(contornosimg,cv2.RETR_TREE,cv2.CHAIN_APPROX_SI
MPLE)

        # Recorremos todos los contornos encontrados
        for c in contornos:
            # Eliminamos los contornos mas pequeños
            if cv2.contourArea(c) < 500:
                continue
```

```
      # Obtenemos el bounds del contorno, el rectángulo mayor que engloba
al contorno

               (x, y, w, h) = cv2.boundingRect(c)
               # Dibujamos el rectangulo del bounds
              cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0),
       2)

              # Mostramos las imagenes de la camara, el umbral y la resta
              cv2.imshow("Camara", frame)
              cv2.imshow("Umbral", umbral)
              cv2.imshow("Resta", resta)
              cv2.imshow("Contorno", contornosimg)

              # Capturamos una tecla para salir
              key = cv2.waitKey(1) & 0xFF

              # Tiempo de espera para que se vea bien
              time.sleep(0.015)

              # Si ha pulsado la letra s, salimos
              if key == ord("s"):
                  break

          # Liberamos la cmara y cerramos todas las ventanas
          camara.release()
          cv2.destroyAllWindows()”
```

(Alumnos, Codigo, 2016)

- ✓ Se realizó la instalación de los sensores de proximidad, inicialmente se pensó en uno delantero y otro trasero, además, se instaló la cámara web. Se realizó la prueba y verificación de que los componentes funcionaran apropiadamente.
- ✓ Se realizó por medio de un diagrama de flujo las decisiones que debía tomar el vehículo autónomo al calcular las distancias de los obstáculos; cuando girar a la izquierda, cuando a la derecha, cuando seguir el camino de forma recta y cuando detenerse.

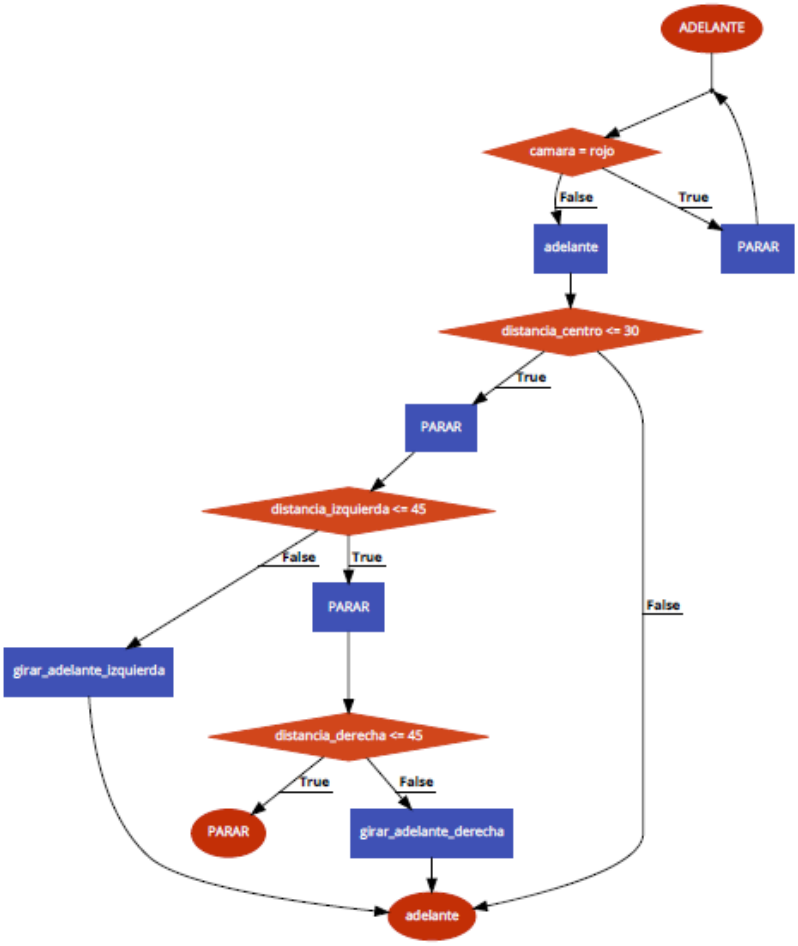


Ilustración *13* Diagrama de flujo movimiento del vehículo

Fuente: (Alumnos, Imagen, 2017)

✓ Se re-establecieron las conexiones de los componentes en el nuevo prototipo tanto en la Raspberry Pi y en los motores de la siguiente manera:

- Conectar Puente H:

  Conectores derechos:
  Cable Naranja
  Cable Azul

  Conectores izquierdos:
  Cable Amarillo

Cable Verde

Conectores Raspberry:
Izq- Der:
Gris Blanco Cafe Rojo

- GND Pin 6 (verde)

- Sensor Central:
  Izq a Der:
  Amarillo Pin 9 - Azul-Blanca Pin 10 - Naranja Pin 12 - Verde Pin 4
  Gnd Echo GIO 15 Trigger GIO 18 5V

  - Sensor Izquierda:
    Izq a Der:
    Cafe Pin14 - Morado Pin 11 - Azul Pin 13 - Rojo Pin 2
    Gnd Echo GIO 17 Trigger GIO 27 5V

  - Sensor Derecha:
    Izq a Der:
    Naranja Pin14 - Amarillo Pin 15 - Negro Pin 16 - Blanco Pin 2
    Gnd Echo GIO 22 Trigger GIO 23 5V

Ilustración *14* Instalación sensores I.

Fuente: (Alumnos, Imagen, 2017)

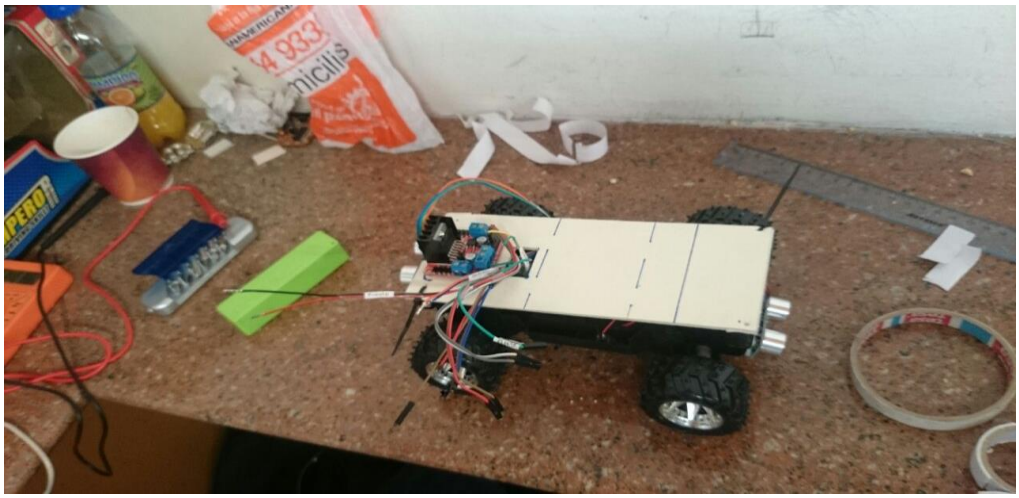


Ilustración *15* Instalación sensores II

Fuente: (Alumnos, Imagen, 2017)

- ✓ Se compiló e integró en Python el código para el cálculo realizado desde la información de los sensores, la identificación de colores con la cámara web y la activación de los motores.

```
index.php | server.php | motor.py | code_auto.py | servid

#!/user/bin/env python

import RPi.GPIO as GPIO
import time
import threading
import sys
from pyvirtualdisplay import Display
from time import sleep

sys.path.append('common')

from motor import Motor
from sensor import Sensor
from camara import Camara

#=================================================================
# Class:
# Autor:
# Date:
# Description:
#=================================================================
class Auto():
    __ACCIONES = ['FRENAR','ADELANTE','ATRAS','G-A-D','G-A-I'
                  ,'G-T-D','G-T-I']
    __ACTIVAR = True
    __DISPLAY = None

    s_central = None
    s_izq     = None
    s_der     = None
    camara    = None
    motor     = None
```

Ilustración *16* Integracion de código I

Fuente: (Alumnos, Imagen, 2017)

Ilustración *17* Integración código II

Fuente: (Alumnos, Imagen, 2017)

Ilustración *18* Integración código III

Fuente: (Alumnos, Imagen, 2017)

Ilustración *19* Integración código VI

Fuente: (Alumnos, Imagen, 2017)

7.3. Etapa tres

- ✓ Se realizó la organización del cableado y componente en el prototipo final garantizando que todos estos quedaran sujetos de una manera compacta.
- ✓ Además, se verificó el funcionamiento de cada componente de código por separado para luego entrar a realizar las pruebas de todos los componentes en conjunto.
- ✓ Para finalizar se organizó y se realizaron los comentarios a cada parte del código final.

7.3.1. Código cámara web.

```
index.php | server.php | motor.py | code_auto.py | camara.py
12  # Description:
13  #===============================================================
14  class Camara():
15      captura     = None
16      rojo_bajos = None
17      rojo_altos = None
18
19      is_color  = False
20
21      def __init__(self,_id):
22          self.captura = cv2.VideoCapture(_id)
23
24          #Establecemos el rango de colores que vamos a detectar
25          #En este caso de rojo oscuro6
26          self.rojo_bajos = np.array([170, 50, 50], dtype=np.uint8)
27          self.rojo_altos = np.array([180, 255, 255], dtype=np.uint8)
28
29      def close(self):
30          self.captura.release()
31          cv2.destroyAllWindows()
32
33      def detectar_color(self, mask, imagen):
34          #Asumo que el color no esta
35          self.is_color = False
36
37          #Encontrar el area de los objetos que detecta la camara
38          moments = cv2.moments(mask)#mask_Rojo)
39          area = moments['m00']
40
41          #Descomentar para ver el area por pantalla
42          if(area > 2000000):
43              #Buscamos el centro x, y del objeto
44              x = int(moments['m10']/moments['m00'])
45              y = int(moments['m01']/moments['m00'])
```

Ilustración *20* Código Cámara Web I

Fuente: (Alumnos, Imagen, 2017)

```
index.php | server.php | motor.py | code_auto.py | camara.py
50
51              self.is_color = True
52
53              #Dibujamos una marca en el centro del objeto
54              ## TO-DO: Comentariar el codigo pq no vamos a usar
55              ## ambiente grafico
56              cv2.rectangle(imagen, (x-5, y-5), (x+5, y+5),(0,0,255), 2)
57              cv2.putText(imagen, "pos :"+ str(x)+","+str(y), (x+10,y+10), cv2.FONT_HERSHEY_SIMPLEX, 1, (255,255,255), 2)
58
59          #Nostramos la imagen original con la marca del centro y
60          #la mascara
61          ## TO-DO: Comentariar el codigo pq no vamos a usar
62          ## ambiente grafico
63          cv2.imshow('mask', mask)
64          cv2.imshow('Camara', imagen)
65
66      def run(self):
67          while(True):
68              try:
69                  _, imagen = self.captura.read()
70                  hsv = cv2.cvtColor(imagen, cv2.COLOR_BGR2HSV)
71                  mask_Rojo = cv2.inRange(hsv, self.rojo_bajos, self.rojo_altos)
72
73                  self.detectar_color(mask_Rojo,imagen)
74
75                  ## Es codigo pararealizar las pruebas
76                  if cv2.waitKey(1) & 0xFF == ord('q'):
77                      self.close()
78                      return
79              except:
80                  print sys.exc_info()
81                  self.close()
```

Ilustración *21* Código Cámara web II

Fuente: (Alumnos, Imagen, 2017)

7.3.2. Código motor.

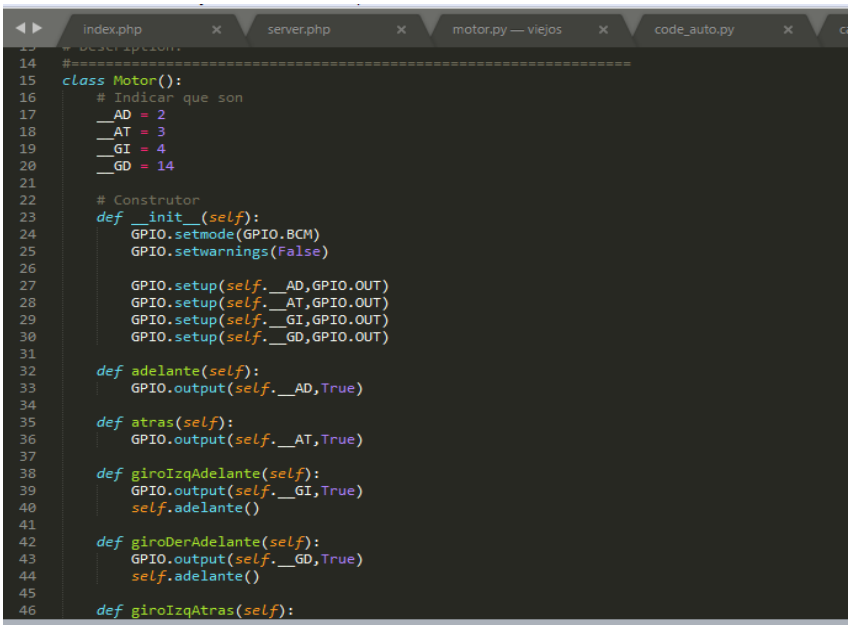


Ilustración *22* Código motor I

Fuente: (Alumnos, Imagen, 2017)

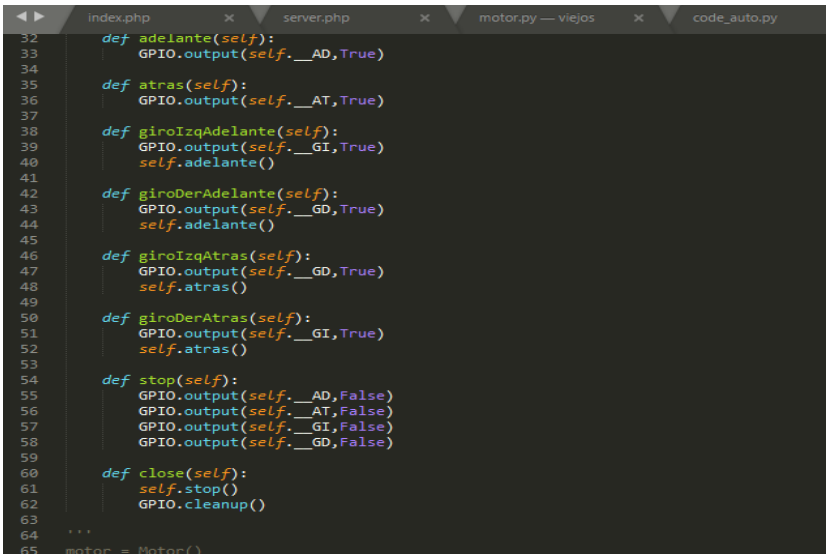
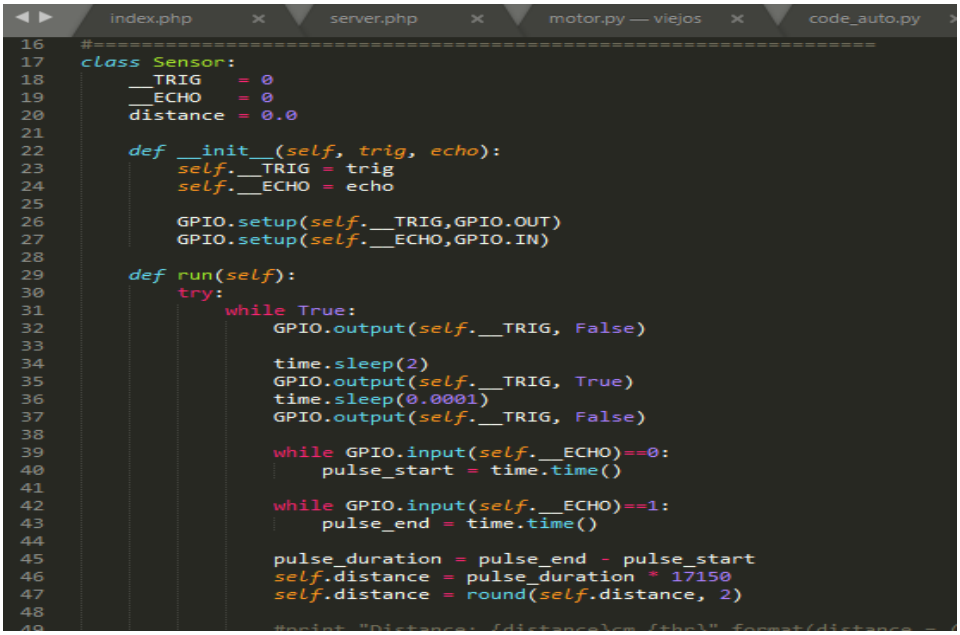


Ilustración *23* Código motor II

Fuente: (Alumnos, Imagen, 2017)

7.3.3. Código sensores.




```
#=================================================================
class Sensor:
    __TRIG    = 0
    __ECHO    = 0
    distance = 0.0

    def __init__(self, trig, echo):
        self.__TRIG = trig
        self.__ECHO = echo

        GPIO.setup(self.__TRIG,GPIO.OUT)
        GPIO.setup(self.__ECHO,GPIO.IN)

    def run(self):
        try:
            while True:
                GPIO.output(self.__TRIG, False)

                time.sleep(2)
                GPIO.output(self.__TRIG, True)
                time.sleep(0.0001)
                GPIO.output(self.__TRIG, False)

                while GPIO.input(self.__ECHO)==0:
                    pulse_start = time.time()

                while GPIO.input(self.__ECHO)==1:
                    pulse_end = time.time()

                pulse_duration = pulse_end - pulse_start
                self.distance = pulse_duration * 17150
                self.distance = round(self.distance, 2)
```

Ilustración *24* Código sensores I

Fuente: (Alumnos, Imagen, 2017)


```
        self.__TRIG = trig
        self.__ECHO = echo

        GPIO.setup(self.__TRIG,GPIO.OUT)
        GPIO.setup(self.__ECHO,GPIO.IN)

    def run(self):
        try:
            while True:
                GPIO.output(self.__TRIG, False)

                time.sleep(2)
                GPIO.output(self.__TRIG, True)
                time.sleep(0.0001)
                GPIO.output(self.__TRIG, False)

                while GPIO.input(self.__ECHO)==0:
                    pulse_start = time.time()

                while GPIO.input(self.__ECHO)==1:
                    pulse_end = time.time()

                pulse_duration = pulse_end - pulse_start
                self.distance = pulse_duration * 17150
                self.distance = round(self.distance, 2)

                #print "Distance: {distance}cm {thr}".format(distance =
            except RuntimeError as e:
                print e
                print "  Quit"
                GPIO.cleanup()
```

Ilustración *25* Código sensores II

Fuente: (Alumnos, Imagen, 2017)

- ✓ Como resultado final quedó construido en su totalidad el prototipo para realizar las pruebas en el ambiente controlado generado para tal fin.

Ilustración *26* Prototipo Final I

Fuente: (Alumnos, Imagen, 2017)

Ilustración *27* Prototipo final II

Fuente: (Alumnos, Imagen, 2017)

Ilustración *28* Prototipo final III

Fuente: (Alumnos, Imagen, 2017)

a. El vehículo al finalizar su construcción y para poder realizar una buena inicialización de todos los componentes tiene un tiempo de arranque aproximado de 45 segundos.

b. Se puede evidenciar un video soporte de la prueba en acción del prototipo construido.

7.3.4. Videos funcionamiento.

1. https://www.youtube.com/watch?v=sv7u-WOfK3k
2. https://www.youtube.com/watch?v=ojAxNU753zE

8. Fuentes de información

8.1. Fuentes primarias

- Ingeniero Abdul Ruiz Saldaña. (2016). ENTREVISTA. Coordinación de Ingeniería de Sistemas. Universidad ECCI. Bogotá.

8.2. Fuentes secundarias

- V. M. Arévalo, J. González, G. Ambrosi. (2004). LA LIBRERÍA DE VISIÓN ARTIFICIAL OPENCV APLICACIÓN A LA DOCENCIA E INVESTIGACIÓN. Dpto. De Ingeniería de Sistemas y Automática, Universidad de Málaga. España. Extraído de: http://mapir.isa.uma.es/varevalo/drafts/arevalo2004lva1.pdf
- Autor. (2012). APLICACIÓN PRÁCTICA DE LA VISIÓN ARTIFICIAL EN EL CONTROL DE PROCESOS INDUSTRIALES. Gobierno de España, Ministerio de Educación Español y Fondo Nacional Europeo. Extraído de: http://visionartificial.fpcat.cat/wp-ontent/uploads/UD_1_didac_Conceptos_previos.pdf
- Dr. Luis Salgado. (2007). VISIÓN ARTIFICIAL: FUNDAMENTOS Y APLICACIONES. Universidad Politécnica de Madrid. Extraído de: http://webcache.googleusercontent.com/search?q=cache:http://arantxa.ii.uam.es/~jms/seminarios_doctorado/abstracts2006-2007/20070503LSalagado.pdf
- Fernando Arboledas Cique y Jesús de Luis Serrano. (2014). INTELIGENCIA ARTIFICIAL EN MEDIOS DE TRANSPORTE. Universidad Carlos III de Madrid. Extraído de: http://www.it.uc3m.es/jvillena/irc/practicas/13-14/04.pdf

- Daniel Asegurado Turón. (2011). EL COCHE INTELIGENTE. Universidad Carlos III de Madrid. Extraído de: http://portal.uc3m.es/portal/page/portal/actualidad_cientifica/publi/feria_ciencia08/coche_intelig
- Joshué Manuel Pérez Rastelli. (2012). AGENTES DE CONTROL DE VEHÍCULOS AUTÓNOMOS EN ENTORNOS URBANOS Y AUTOVÍAS. Universidad Complutense de Madrid. Extraído de: https://www.google.com.co/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0ahUKEwiDoJDV09jXAhXCmuAKHQbcBM4QFggkMAA&url=http%3A%2F%2Fdigital.csic.es%2Fbitstream%2F10261%2F47804%2F1%2FTesis_Joshue_Perez.pdf&usg=AOvVaw37c1fvkPXWu4HBUDcXk4Ta
- Daniel Asegurado Turón. (2012).EL COCHE INTELIGENTE, EN BUSCA DE MAYOR SEGURIDAD, SOSTENIBILIDAD Y CONFORT. Universidad Carlos III de Madrid Leganés, Madrid, España. Extraído de:

  http://www.it.uc3m.es/jvillena/irc/practicas/11-12/01pres.pdf
- Dayana H. Bailon Delgado. (2015). AVANCE SIGNIFICATIVO DE INTELIGENCIA ARTIFICIAL: VEHÍCULOS AUTÓNOMOS. Escuela Superior Politécnica Agropecuaria de Manabí Manuel Félix López, Calceta – Ecuador. Extraído de:

https://dayuia6toinfor.files.wordpress.com.

- El Tiempo. (2016). LO DIFÍCIL QUE ES SALIR ILESO DE UN ACCIDENTE CON MOTO EN BOGOTÁ. Bogotá. Extraído de:
- Carmelo Marin. (2015). SEGUIMIENTO DE OBJETOS POR COLOR. Extraído de:
    - http://acodigo.blogspot.com.co/2016/04/seguimiento-de-objetos-por-color.htmlCarmelo Marin. (2015). ACENTUAR COLOR. Extraído de: http://acodigo.blogspot.com.co/
- Centro de Análisis y Prospectiva Gabinete Técnico de la Guardia Civil. (2014). Extraído de: http://intranet.gccap.bage.es/
- Robert Laganiere. (2011). OPENCV 2 COMPUTER VISION APPLICATION PROGRAMMING COOKBOOK. Extraído de: 9781849513241-OPENCV_2_COMPUTER_VISION_APPLICATION_PROGRAMMING.pdf.

9 Recursos

Durante el proyecto se tienen en cuenta los diferentes tipos de recursos que se utiliza durante la ejecución de este; en las siguientes tablas se observan los recursos humanos y de materiales que se tuvieron en cuenta.

Tabla *1* Tabla Recurso Humano

Recurso Humano			
Profesional	**Duración/Horas**	**Valor/Hora**	**Vr. Total**
Estudiante Ingeniería de Sistemas	144	$ 10.415,00	$ 1.499.760,00
Investigación, documentación del proyecto. Desarrollo del prototipo.			
Estudiante Ingeniería de Sistemas	144	$ 10.415,00	$ 1.499.760,00
Programación de los algoritmos. Desarrollo del prototipo.			
Estudiante Ingeniería de Sistemas	144	$ 10.415,00	$ 1.499.760,00
Conexión circuitos, accesorios para la funcionalidad del prototipo. Documentación del proyecto.			
Tutor	15	$ 50.000,00	$ 750.000,00
Asesor externo	15	$ 50.000,00	$ 750.000,00
Total recursos humanos	**$5.999.280,00**		

Fuente: Los Autores. 2018.

Tabla *2* Tabla recurso material

Recurso físico			
Elemento/Descripción	**Cantidad**	**Valor unitario**	**Vr. Total**
Sensores de proximidad	6	$ 2.500,00	$ 15.000,00
Identificación de obstáculos por proximidad			
Vehículo control remoto	1	$ 200.000,00	$ 200.000,00
Conexión entre de los diferentes dispositivos			
Puente H	1	$ 20.000,00	$ 20.000,00
Conexión entre de los diferentes dispositivos			

Raspberry pi	1	$ 360.000,00	$ 360.000,00
Conexión y configuración de los diferentes dispositivos.			
Cámara web	1	$ 30.000,00	$ 30.000,00
Identificación por medio de video del semáforo a escala			
Semáforo a escala	1	$ 60.000,00	$ 60.000,00
Realizar pruebas de identificación de los colores por medio de video.			
Computador portátil	2	$ 1.850.000,00	$ 3.700.000,00
Programación, configuración, generación de algoritmos para funcionalidad del prototipo.			
Total recursos físicos	**$4.385.000,00**		

Fuente: Los Autores. 2018.

Así, la inversión final del Proyecto fue de **$ 10.384.280,oo**

10 Cronograma

Para una buena planeación durante la ejecución del proyecto se tuvo en cuenta una serie de actividades para contar con una correcta organización y establecer un tiempo estimado para la finalización de este. A continuación, se observa el diagrama del cronograma de actividades y el diagramaba de Gantt perteneciente al proyecto.

	Nombre	Duración	Inicio	Fin	Predecesoras	Recursos
1	Trabajo de investigacion sobre vehículos autonomos	20d?	16/01/2017	10/02/2017		Hector Andres Rodriguez Martinez,Neiro Culma Yate,M
2	Generación del documento de grado.	30d?	13/02/2017	24/03/2017	1II+10d	Hector Andres Rodriguez Martinez,Neiro Culma Yate,M
3	Instalacion y configuracion del Sistema operativo en la raspberry Pi	10d?	27/03/2017	07/04/2017		Hector Andres Rodriguez Martinez[50%],Neiro Culma
4	**Sensores de proximidad**	15d?	**10/04/2017**	**28/04/2017**		
5	Configuración de los sensores de proximidad en el vehículo.	10d?	10/04/2017	21/04/2017	3	Hector Andres Rodriguez Martinez[10%],Neiro Culma
6	Prueba de los sensores de proximidad	5d?	24/04/2017	28/04/2017	5	Miguel Leonardo Ocampo Gomez,Raspberry pi[1],Sen
7	**Visión**	55d?	**01/05/2017**	**14/07/2017**		
8	Configuración de la cámara en la Raspberry PI	10d?	01/05/2017	12/05/2017	6	Hector Andres Rodriguez Martinez[40%],Neiro Culma
9	Generación de los objetos a identificar	10d?	05/05/2017	18/05/2017	8II+4d	Hector Andres Rodriguez Martinez[50%],Neiro Culma
10	Generación del algoritmo de visión artificial	30d?	15/05/2017	23/06/2017	9II+6d	Hector Andres Rodriguez Martinez,Neiro Culma Yate,M
11	Prueba los algoritmos visión artificial en el vehículo.	15d?	26/06/2017	14/07/2017	8,9,10	Hector Andres Rodriguez Martinez,Neiro Culma Yate,M
12	Generación del algoritmo para el movimiento autónomo del vehículo.	15d?	17/07/2017	04/08/2017	11	Hector Andres Rodriguez Martinez[50%],Neiro Culma
13	Prueba del vehículo para verificar su movimiento (girar izq, girar der, adelante, atras)	5d?	07/08/2017	11/08/2017	12	Hector Andres Rodriguez Martinez[80%],Neiro Culma
14	Conexión de todos los componentes físicos del vehículo.	20d?	14/08/2017	08/09/2017	4,7,13	Hector Andres Rodriguez Martinez,Miguel Leonardo C
15	Pruebas del prototipo con todos los componentes instalados y configurados	15d?	11/09/2017	29/09/2017	14	Hector Andres Rodriguez Martinez,Neiro Culma Yate,M

Ilustración *29* Cronograma Actividades
Fuente: Los Autores. 2018.

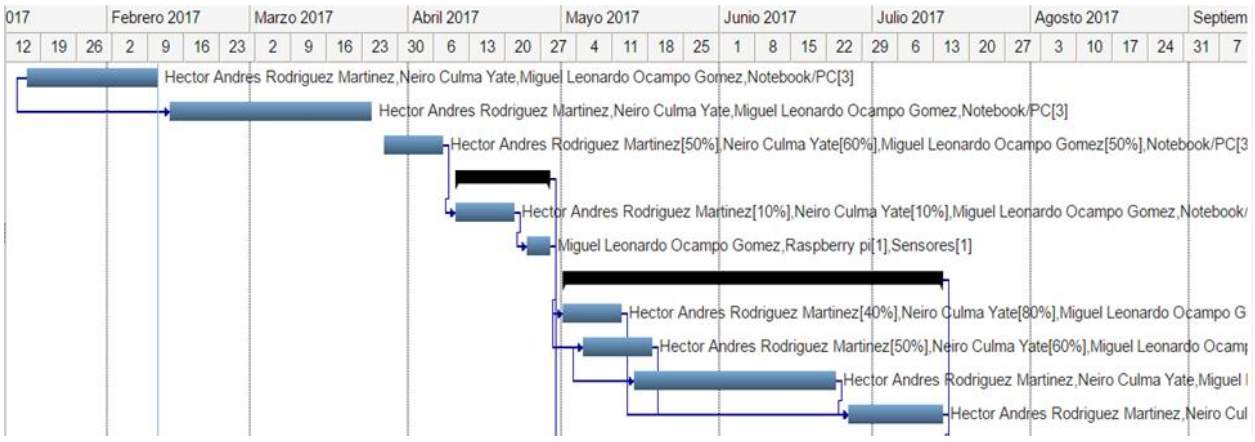


Ilustración *30* Diagrama de Gantt
Fuente: Los Autores. 2018.

11 Conclusiones

- ✓ Los cambios en el entorno, como, por ejemplo, el contraste y el brillo, afecta la identificación del color rojo que se estableció para la activación del algoritmo de identificación de imágenes.
- ✓ Se utilizaron baterías recargables para el funcionamiento del vehículo autónomo con capacidad de 2000 mAh cada una, las cuales poseen una autonomía tanto para la Raspberry Pi, como para el funcionamiento de los motores del prototipo de dos horas y media aproximadamente.
- ✓ Al realizar la programación de los sensores, se evidenció que si estos inician a captar la información sobre las distancias fuera del bucle el tiempo de captación de la información es mayor que si se realizar dentro del mismo, es decir que calcula luego de cumplir la condición inicial la identificación de los colores con la cámara.
- ✓ El puente H es el medio conductor que organiza los circuitos de los motores para iniciar la comunicación con la Raspberry Pi por medio de los pines GPIO los cuales deben identificarse previamente para ingresarlos en algoritmo que activa dichos motores. Los pines de la Raspberry Pi no cuentan con la misma numeración de los GPIO, es decir, por ejemplo, el pin 11 físico de la Raspberry Pi es el GPIO lógico 17.
- ✓ El arranque de cada componente y sistema operativo se tarda alrededor de 15 segundos en cada caso.
- ✓ Para que este prototipo lograra evitar un obstáculo exitosamente debió realizar los cálculos e identificar el primer objeto por lo menos a 30 centímetros de distancia del mismo teniendo en cuenta que este va a una velocidad constante de 3 metros por segundo (3 m/s), si llevamos

estos resultados a un escenario real a escala natural, tenemos que un vehículo viajando a una velocidad constante de 11 kilómetros por hora debe identificar el primer obstáculo a por lo menos 48 metros de distancia para poder esquivarlo exitosamente.

- ✓ La implementación del prototipo de vehículo autónomo con reconocimiento de semáforos, obstáculos, el cual cuenta con sensores de proximidad puede llegar a ser una herramienta útil que permita reducir las cifras de accidentalidad de manera global.
- ✓ El algoritmo fue desarrollado en Python, ya que este lenguaje es uno de los más usados en inteligencia artificial debido a la simplicidad de su sintaxis y su flexibilidad, además, con un ambiente de programación amigable para el usuario.
- ✓ No debemos centrarnos en una sola área o herramienta tecnológica, ya que hoy en día es necesario conocer diversas áreas, como en éste proyecto, en el cual se combinan tanto la parte de electrónica, robótica, programación e inteligencia artificial, todo esto para en un futuro ponerlas al uso de la comunidad sobre la prevención de accidentes automovilísticos por medio de éstas tecnologías modernas.
- ✓ Se combinan diversas áreas de las tecnologías, como es la robótica, programación, inteligencia artificial y electrónica, las cuales aportan valor agregado al diseño del prototipo del automóvil autónomo.
- ✓ La visión artificial es fundamental para realizar la captura de imágenes por medio de colores, se utiliza una cámara que apunta hacia el semáforo lo que nos permite identificar uno de los colores (verde - rojo), para seguido a esto realizar la acción programada de acuerdo a los resultados obtenidos (parar - seguir).

12 Recomendaciones

- ✓ Para futuras actualizaciones es necesario contemplar la implementación de 3 sensores traseros y dos sensores laterales.
- ✓ Para optimizar la visión del vehículo se recomienda implementar una cámara web trasera.
- ✓ Debido a que estas mejoras incrementan su peso es necesario migrar a un prototipo de mayor envergadura para su correcto funcionamiento.

Vita

Miguel Leonardo Ocampo Gómez, Héctor Andrés Rodríguez Martínez y Neiro Culma Yate, estudiantes en la Universidad ECCI en la facultad de ingeniería de sistemas en el año 2017, residentes en la ciudad de Bogotá, con pasiones muy similares ya que tienen como base el mundo de la informática, se encuentran en proceso de graduación y por medio de éste proyecto concluyen uno de los requisitos para obtener el diploma de Ingenieros de Sistemas para poder seguir ejerciendo y creciendo en ésta hermosa área de las TIC con profesionalismo y ética, heredada en cada una de las materias vistas durante la carrera.