UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

RED DE SEMÁFOROS INTELIGENTES

TESIS

Que para obtener el título de

Ingeniero en Computación

P R E S E N T A N

De Reza Trujillo Juan Francisco

Rivera Landeros Ángel De Jesús

DIRECTOR DE TESIS

Ing. Laura Sandoval Montaño

Ciudad Universitaria, Cd. Mx., 2016

AGRADECIMIENTOS

Juan F. de Reza T.

Agradezco a mi familia por el apoyo incondicional que me ha brindado, por estar siempre a mi lado escuchándome, aconsejándome y ayudándome. Sin ellos no habría llegado a la etapa a la que he llegado.

Retribuyo a mi esposa por todo lo que ha hecho por mí, por siempre bridarme su amor y cariño. Por siempre impulsarme a llegar a cada meta.

Agradezco a la profesora Laura por todas las facilidades y apoyo que me ha brindado. Así mismo, agradezco a todos mis profesores ya que han sido parte de esto por la enseñanza que me han dado.

Ángel de Jesús Rivera Landeros

Dedico esta tesis a mis amigos Juan, Melanie, Itzel, Sofía, quienes fueron un gran apoyo emocional durante el tiempo en que escribía esta tesis.

A mis padres quienes me apoyaron todo el tiempo con su trabajo y esfuerzo.

A mis maestros quienes nunca desistieron al enseñarme, aun sin importar que muchas veces no ponía atención en clase.

A todos los que me apoyaron para escribir y concluir esta tesis.

Dedico esta tesis a Dios, y a la Virgen María, quienes inspiraron mi espíritu para la conclusión de esta tesis profesional, en ingeniería.

CONTENIDO

1. INTRODUCCIÓN

Problemas de tránsito en la Ciudad de México

La sobrepoblación es un problema mundial que afecta a diversas urbes, una de ellas la Ciudad de México, disminuye la calidad de vida de los habitantes y provoca daños al medio ambiente.

Actualmente en la Ciudad de México hay graves problemas de tránsito que afectan a la ciudadanía, algunas de las causas son:

- El gran crecimiento de la población en los últimos años

  Según datos del sitio del Fideicomiso para el Mejoramiento de las Vías de Comunicación del Distrito Federal (2015). En las últimas décadas la población de la Ciudad de México ha crecido en gran medida, para el 2014 se tenían 8 millones 843 mil habitantes. El crecimiento de la población ha causado que cada vez existan más personas en movimiento sobre las vías de transporte. La facilidad actual que existe para obtener un crédito, ha favorecido a que un gran número de personas cuente con un automóvil particular como medio de transporte.

  En los últimos años la Ciudad de México ha sufrido una despoblación en las delegaciones centrales, y un mayor crecimiento en las áreas externas de la ciudad, así como de la población en los municipios del Estado de México aledaños. Esto provoca que los ciudadanos ahora realicen viajes más largos que antes, empeorando los problemas viales.

  En el libro la ética ambiental frente a la sobrepoblación Sagols L. (2010) menciona que:

  > La sobrepoblación es la principal causa del deterioro ambiental y de la deshumanización actual. Sin embargo, hay resistencia a asumir este dato debido a los prejuicios contra la crítica a la sobrepoblación (p.1).

- Infraestructura vial deficiente

  El no contar con una infraestructura vial adecuada en la ciudad afecta al flujo de tráfico. Si se contara con vialidades en buenas condiciones y se mejorara el sistema de funcionamiento de los semáforos, los índices de tráfico bajarían, los viajes tomarían menor tiempo disminuyendo el tiempo muerto que los ciudadanos gastan debido al tráfico.

- Ineficiencia del transporte público

  Otro factor importante que provoca problemas viales es la baja calidad del transporte público. Actualmente es mayor la población que viaja en vehículos de baja y mediana capacidad como son autos particulares, furgonetas y taxis, lo que provoca mayor cantidad de vehículos en vialidades teniendo problemas de tránsito. La mayoría de ciudadanos no quiere utilizar el transporte público debido al mal servicio que ofrece.

La gran flota vehicular en la ciudad genera altos índices de contaminación generando problemas como los siguientes:

- Daños en la salud

  La contaminación del aire afecta a la salud humana provocando enfermedades respiratorias crónicas y disminución de las capacidades respiratorias. Se ha registrado que la tasa de enfermedades respiratorias está incrementando y se adjudica a la contaminación ambiental en la ciudad; se han registrado una gran cantidad de muertes debido a este hecho. Lo anterior se deriva del gran parque vehicular en la ciudad, donde las concentraciones de gas emitidas son mayores a las permisibles.

  Algunos estudios señalan daño cardiovascular debido a la exposición de partículas finas como lo son los humos de diésel. Esto ha generado un mayor índice de mortalidad.

  Las personas actualmente pueden llegar a perder de dos a cuatro horas en promedio en el transporte, lo cual aumenta los niveles de estrés y en épocas de calor provoca daños a la salud debido a la deshidratación.

- Económicos y de tiempo

  Según la Asociación Mexicana de transporte y movilidad (2013) los problemas de tráfico causan que al año cada persona pierda 25 mil 677 pesos. Algunos de los ciudadanos invierten casi el 43% de su salario en el transporte público. En la Ciudad de México un trayecto promedio que hace tres años duraba cincuenta minutos, ahora dura una hora con veinte minutos, así mismo, la velocidad promedio de viaje ha bajado y esto representa un motivo de alarma.

- Daños en el medio ambiente

  Según la Asociación Mexicana de transporte y movilidad (2013). El transporte representa un 60% de la contaminación, además de los problemas viales, tiene repercusión monetaria en la población de la Ciudad de México. El tener mayor cantidad de vehículos en las vialidades y el que se realicen viajes más largos, provoca mayores niveles de contaminantes en el aire.

  Un automóvil causa una seria aportación al cambio climático, debido a los grandes tiempos de viaje generados por el tráfico, los depósitos contaminantes sobre el suelo (pérdida de aceite, derrame de gasolina, etc.) son mayores, esto afecta la vegetación produciendo su pérdida, perjudicando a la fauna y a los ciclos naturales.

Estos problemas nos afectan directa e indirectamente, por ello, al lograr que el tiempo de viaje de un vehículo disminuya, mejoraría la economía del propietario; el tiempo y la salud del conductor y sus pasajeros, así como la disminución de contaminantes. Todo ello se puede alcanzar haciendo una mejora a

la infraestructura vial, que se compone de semáforos que organizan la masa de automóviles para que los usuarios lleguen a su destino de manera rápida y eficiente.

Propuesta de solución para mejorar el flujo vial en la Ciudad de México

Se propone optimizar el tráfico, es decir, mejorar el flujo vial y el nivel de tráfico en las vialidades de la Ciudad de México a través de una mejora en los tiempos de operación de los semáforos que tienen como función organizar la flota vehicular.

Se pretende desarrollar un sistema de semáforos que operen de manera inteligente de acuerdo a su entorno. Cada uno de los semáforos debe reconocer el nivel de tráfico existente en la vialidad que controla, y con base en esta información sus tiempos de operación se ajusten permitiendo mejorar el nivel de tráfico reconocido, es decir, que haya menor densidad automovilística en las vialidades.

Al ajustar los tiempos de operación en cada semáforo de acuerdo al nivel de tráfico que se identifica cada cierto instante, es posible aumentar el tiempo de operación en luz verde para aquellos semáforos cuyas vialidades cuenten con un alto nivel de tráfico, usando parte del tiempo en luz verde de los semáforos en vialidades con bajo nivel de tráfico. El incrementar el tiempo de operación para vialidades con alto nivel de trafico supone liberar el flujo de automóviles al encontrar menor obstaculización.

Para el desarrollo de la red de semáforos inteligentes que permitan mejorar el flujo vial se propone un sistema que conste de tres fases, la primera se encargará del reconocimiento del nivel de tráfico en las vialidades que controla cada uno de los semáforos, la segunda realizará la optimización del tiempo de operación de cada semáforo con base en el nivel de tráfico identificado en la vialidad que controlan, y la tercera usará estos tiempos para controlar las luces de los semáforos.

Para la primera fase se propone un sistema de reconocimiento de patrones en imágenes, por este motivo cada uno de los semáforos deberá contar con un sensor de imagen tomando una captura de la vialidad y a partir de ella reconocer el nivel de tráfico existente.

Para la optimización de tráfico se debe tomar en cuenta las relaciones que existen entre semáforos de una misma intersección, cada intersección debe estructurar la información de niveles de tráfico reconocidos en cada semáforo para ser procesados por la segunda fase.

En la segunda fase se recibirán los niveles de tráfico identificados en los semáforos de una intersección y con base en sus relaciones se optimizarán los tiempos de operación de los semáforos en la intersección a través de algoritmos especializados en esta tarea. Los tiempos optimizados son enviados a la siguiente fase.

La tercera y última fase se encarga de utilizar los tiempos optimizados para controlar las luces de los semáforos, el tiempo optimizado es el tiempo de operación en luz verde.

Las tres fases mencionadas se ilustran en la figura 1.1.

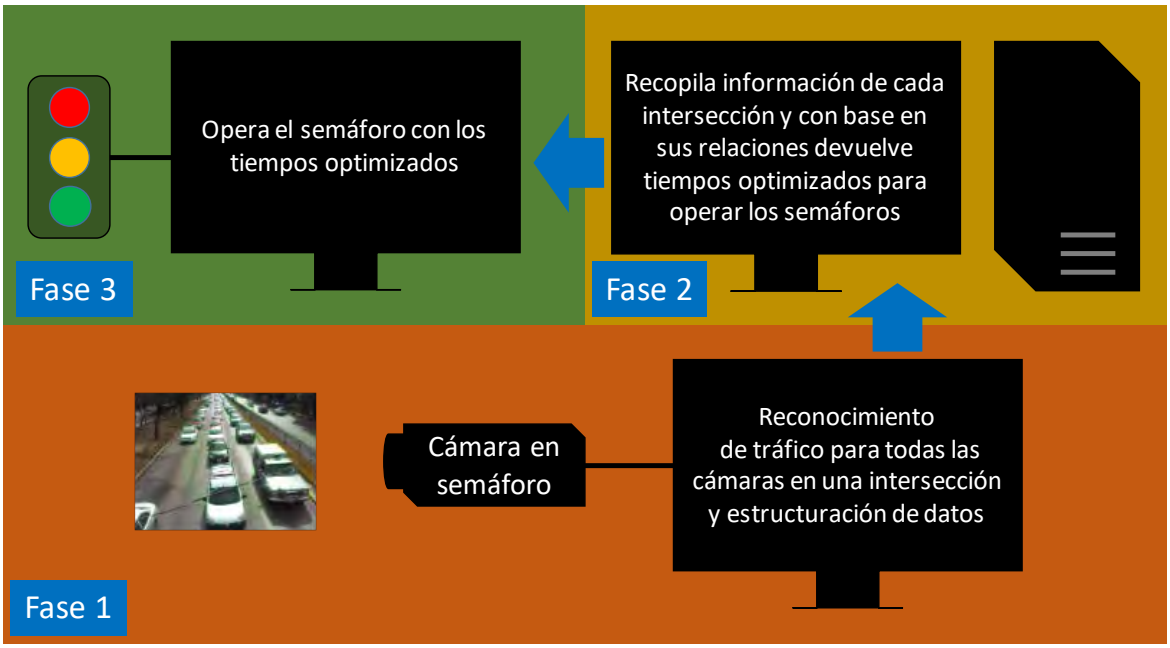


Figura 1.1. Las tres fases que componen a la red de semáforos inteligentes

A lo largo de este trabajo se describirán a detalle cada una de las fases que componen el desarrollo de la red de semáforos inteligentes. En el Tema 2 (Antecedentes) se identifican los problemas y su posible solución buscando los fundamentos teóricos que se necesitan para alcanzarla. Una vez realizada la investigación se procede a dar forma a cada una de las fases que componen a la red de semáforos inteligentes, en el Tema 3 (Análisis, desarrollo e implementación de la red de semáforos inteligentes) se describe a detalle el proceso de reconocimiento de nivel de tráfico a través de sensores de imagen (cámaras) y posteriormente se describe a detalle la construcción y funcionamiento de cada uno de los algoritmos de optimización de tráfico que permiten alcanzar el objetivo de la tesis. Finalmente, en el Tema 4 (Etapa final) se realizan pruebas en donde se verifique que los tiempos de operación en los semáforos de una intersección sean optimizados según los niveles de tráfico identificados y que pretenden optimizar el nivel de tráfico en las vialidades.

Existen proyectos similares en donde se propone crear redes de semáforos auto organizables y que demuestran ser una idea innovadora para mejorar el flujo vial en ciudades con un alto nivel de población como es el caso de la Ciudad de México, un caso de este tipo de proyecto es el que se lleva a cabo en el Instituto de Investigaciones en Matemáticas Aplicadas y Sistemas por el Dr. Carlos Gershenson García. En este lugar se está desarrollando un proyecto de semáforos auto - organizables que tiene como objetivo optimizar el nivel de tráfico a través de semáforos que trabajan de manera colectiva y reconozcan el nivel de tráfico en cada vialidad haciendo uso de cámaras. Ellos le llaman semáforos auto -organizables debido a que no requieren ningún control central, sino que ellos mismos toman decisiones de acuerdo a condiciones locales.

En este trabajo de tesis se toma la misma idea de crear semáforos capaces de variar sus tiempos de operación según las condiciones de su entorno local y hacer uso de sensores de imagen para el reconocimiento de nivel de tráfico, pero en este caso se propone contar con un control central encargado de ordenar a cada uno de los semáforos variar sus tiempos de operación, de acuerdo a la información que obtenga por parte de cada uno de ellos.

2. ANTECEDENTES

Propuesta para solucionar el problema

Dada la amplitud del problema a resolver este puede abordarse en diferentes etapas, las cuales se identifican en la tabla 2.1.

Problema identificado	Propuesta de solución
Sensor del nivel de tráfico.	Una cámara que realiza capturas cada cierto intervalo de tiempo; la salida del sensor servirá para identificar el nivel de tráfico.
Procesamiento de imagen.	Algoritmo de análisis de imágenes para obtener una aproximación al nivel de tráfico existente en el área. Una computadora ubicada en cada intersección se encargará de dicho trabajo. Dada la complejidad del problema, se pretende que funcione sólo bajo condiciones específicas, por ejemplo, en días soleados y despejados.
Comunicación entre semáforos dentro de una misma intersección.	Instalar un equipo de cómputo en cada intersección, para recabar la información obtenida de las cámaras y controlar el cambio de luces de los semáforos.
Comunicar las intersecciones.	Recabar la información localizada dentro de cada equipo de cómputo hacia una estación central, por medio de una red de datos.
Procesamiento de la información obtenida para lograr reducir el tráfico.	En la estación central los algoritmos de optimización de tráfico que permitirán mejorar los tiempos de operación de los semáforos en una intersección.
Representar la red de semáforos computacionalmente en la estación central.	Las características de una intersección vial y sus semáforos serán almacenadas en una base de datos.
Controlador que debe construirse para controlar las luces de los semáforos.	Por medio de un análisis del funcionamiento actual, derivar un implemento de hardware con funcionalidad similar para el control de luces, pero que opere en base a los tiempos de operación optimizados comunicados por el equipo de cómputo en cada intersección.

Tabla 2.1. Relación de problemas que se necesitan solucionar para cumplir con el objetivo.

En este tema se analizarán cada uno de los elementos mencionados en la tabla anterior, para abordar posibles soluciones a los problemas detectados, el tema siguiente consiste en mostrar la metodología para poder realizar la implementación.

En la figura 2.1 se presentan los elementos que componen al sistema de optimización de tráfico propuesto y cómo se relacionan.

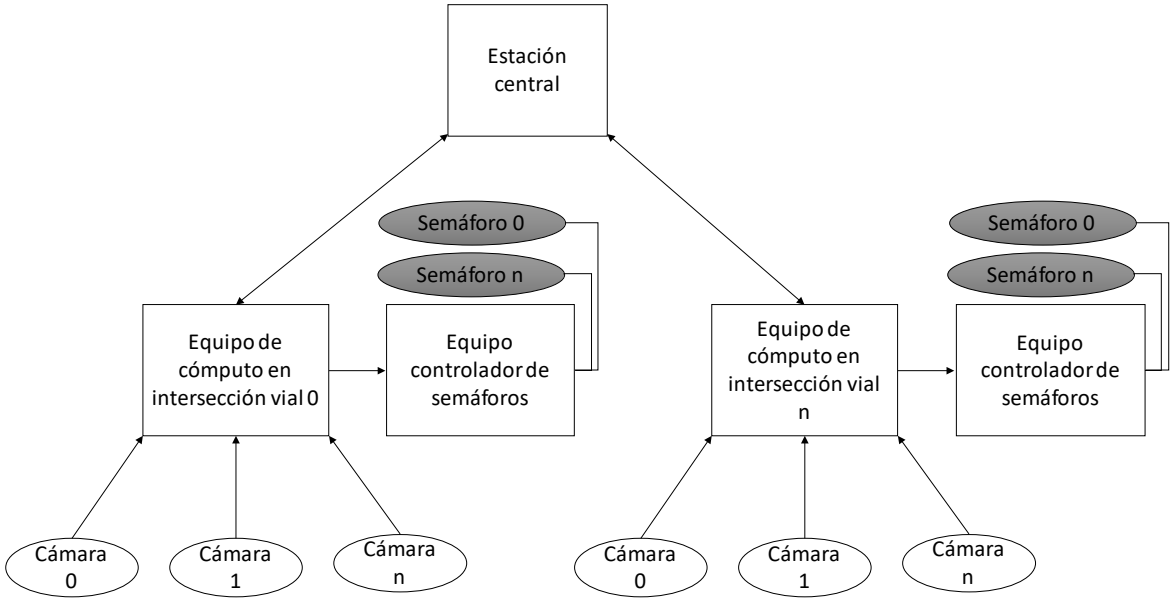


Figura 2.1. Elementos que conforman al sistema de semáforos inteligentes. El flujo de información es representado por flechas, se muestra en forma de bloques grises la infraestructura existente y en bloques blancos la infraestructura adicional que se propone.

Como se puede observar, los bloques involucran la solución a los problemas identificados y se muestra en la tabla 2.2.

Problema identificado	Bloque que responde o donde se implementa la solución
Sensor del nivel de tráfico.	Cámara n.
Procesamiento de imagen.	Equipo de cómputo en intersección vial n.
Comunicación entre semáforos dentro de una misma intersección.	Equipo de cómputo en intersección vial n.
Comunicar las intersecciones a un punto central.	Estación central y las flechas entre ella y equipo de cómputo en intersección vial n.
Procesamiento de la información obtenida para lograr reducir el tráfico.	Estación central.
Representar la red de semáforos computacionalmente en la estación.	Estación central.
El equipo que debe construirse para controlar las luces de los semáforos.	Equipo controlador de semáforos.

Tabla 2.2. Relación de problemas que se necesitan solucionar y el bloque del sistema que los solucionará.

A partir del análisis anterior, se definirá aquellos conceptos que se requieren conocer para formular la solución.

Cámaras

En este subtema tiene por objetivo describir a detalle los principales elementos que componen a la captura y procesamiento de la imagen.

Sensor de imagen

Un sensor de imagen es el elemento más importante de una cámara fotográfica, es el responsable de generar imágenes. Este sensor generalmente tiene una forma rectangular con una proporción 16:9, está compuesto por una matriz de elementos fotosensibles capaces de captar la intensidad de la luz y el color para después transformar esta intensidad luminosa en cargas eléctricas. Este sensor es por tanto un transductor capaz de transformar energía luminosa en energía eléctrica.

El tamaño real de la imagen que genera el sensor se mide en megapíxeles, a mayor cantidad de megapíxeles, mejor calidad de imagen, siempre y cuando sean pixeles reales. Cada uno de los elementos fotosensibles con los que cuenta el sensor es denominado pixel, un megapíxel contiene un millón de pixeles.

Cabe mencionar que mientras mayor sea el número de pixeles del sensor, éste tendrá un mayor tamaño, afectando al ángulo del campo de visión; a mayor tamaño del sensor, mayor será el campo de visión, el tamaño del sensor también afectará al tamaño total de la cámara fotográfica y el precio de ésta.

Tipos de sensores de imagen

Actualmente se cuentan con dos tipos de sensores de imagen que son CCD y CMOS, siendo los primeros los que más comúnmente son usados para aplicaciones profesionales. A continuación, se describe cada uno de ellos.

- **CCD**: Son dispositivos de acoplamiento de carga. Están conformados por una serie de condensadores que, a partir de una señal analógica, transfieren sus cargas hacia los condensadores cercanos. Debido a los elementos fotoeléctricos que contiene el sensor, al estar en exposición a la luz, generan pequeños pozos de carga eléctrica que varían de tamaño, dependiendo de la intensidad de luz y del tiempo que esté expuesto el elemento fotoeléctrico. Los pozos de carga funcionan como salida del sensor, y se requiere de un convertidor analógico digital para poder trabajar con ellos en aplicaciones digitales. Este tipo de sensores implican un costo muy elevado para su fabricación además de que consumen grandes cantidades de energía, por lo que las cámaras digitales que cuentan con este sensor, generan imágenes de alta calidad a un costo elevado.
- **CMOS**: Los sensores CMOS se componen de varias celdas fotosensibles que se encargan de captar la información luminosa y a diferencia de los sensores CCD no trasfieren carga, la conservan. Esta información se almacena en un formato digital para ser procesada por la cámara y generar la imagen. Para saber el color que debe tomar cada pixel, estos tienen encima una malla de filtros de colores rojo, verde y azul para captar la cantidad de luz roja, verde y azul de cada pixel y generar el color correcto. Este tipo de sensores son más baratos de fabricar que los sensores CCD, ya que requiere menos componentes electrónicos y el propio chip puede integrar funciones como compresión de fotografías o un convertidor analógico digital. Otro aspecto importante es que consumen menos energía que los sensores CCD teniendo mayor tiempo de vida. A pesar de que los sensores CCD tienen mejor calidad de imagen, los sensores CMOS, gracias al post-procesamiento de imágenes, pueden obtener imágenes de calidad similar a los CCD.

Los sensores de imágenes CMOS permiten procesar la imagen a una mayor velocidad ya que de esto se encarga el propio sensor mientras que en los CCD el procesamiento se debe realizar fuera del él. Las cámaras integradas con sensores CMOS tienen mayor velocidad en la captura de imágenes.

Objetivo de la cámara

Este es otro elemento muy importante en las cámaras fotográficas, porque en conjunto con el sensor de imagen, determinan el campo de visión y el nivel de detalle que será capaz de capturar la cámara fotográfica.

El campo de visión, es un elemento de suma importancia en una cámara fotográfica, indica la cobertura que visualizará. La longitud focal, determina el campo de visión y está dado por la distancia entre el objetivo de la cámara y el sensor de imagen, a mayor distancia menor será el campo de visión. Los campos de visión se pueden clasificar en tres tipos:

- **Vista normal**: Capaz de capturar una escena como si se tratara del ojo humano.
- **Telefoto**: Tiene una cobertura menor que la vista normal, pero tiene una mayor cantidad de detalle en la imagen, además tienen menos capacidad para recoger la luz.
- **Gran angular**: Es el campo de visión con mayor cobertura, pero a la vez, el que cuenta con una menor cantidad de detalle.

En el mercado se pueden encontrar una gran variedad de cámaras fotográficas con diferentes ángulos de visión, por lo que se debe conocer bien cuál es el campo de visión más favorecedor para satisfacer nuestras necesidades.

Los objetivos también son capaces de permitirnos ajustar el enfoque de la cámara, moviendo la distancia entre el objetivo y el sensor de imagen (longitud focal). Existen objetivos fijos en los cuales no es posible modificar esta distancia y solamente se cuenta con un tipo de campo de visión. Existen objetivos de óptica variable o con zoom, que permiten modificar la longitud focal y de esta manera tener diferentes campos de visión con un mismo objetivo.

Cabe mencionar que debe existir una relación entre el objetivo y el tamaño del sensor de imagen, si el objetivo de la cámara es para un sensor de imagen de menor tamaño, la imagen mostrará esquinas de color negro. Si el objetivo es para un sensor de imagen de mayor tamaño, el campo de visión será menor del que este es capaz de mostrar, creando un efecto de telefoto.

Un objetivo también es capaz de regular la cantidad de luz que incide sobre el sensor de imagen, lo que permitirá que se tengan mejores o peores imágenes en situaciones de poca luz.

Otro aspecto que debe ser tomado en cuenta para aplicaciones con cámaras, es la profundidad del campo de visión, que hace referencia a la distancia más allá del punto de enfoque, en donde los objetos de la imagen aún son nítidos. En la figura 2.2 se puede ver ilustrado este concepto.

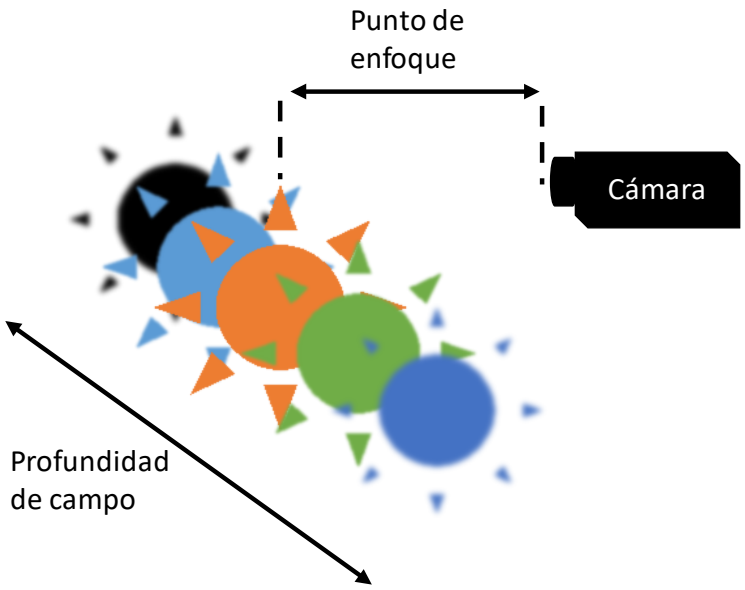


Figura 2.2. Enfoque de una cámara en un campo con profundidad.

Reconocimiento y procesamiento de imágenes

Para poder hacer el reconocimiento de una imagen lo que se debe hacer es analizar toda la imagen pixel a pixel, lo que involucra el uso de gran cantidad de recursos computacionales. Para poder saber si un conjunto de imágenes pertenece a la misma clase, por ejemplo, para poder identificar rostros en sus diferentes expresiones, es decir, tristeza, felicidad, enojo, etcétera, se debe extraer patrones que no presenten demasiado cambio.

Los sistemas de reconocimiento de imágenes deben ser suficientemente robustos para identificar todas las variaciones posibles del objeto que se está analizando.

A partir de características locales del objeto a identificar, es decir, aquellos patrones que no presentan demasiado cambio en una imagen base, se realiza una extracción de las características locales para compararlas con las características locales de las imágenes a reconocer, se analiza pixel a pixel y dependiendo de la varianza que exista entre ellas es como se decide si se trata del mismo objeto o no.

Una de las técnicas más utilizadas en la actualidad para el reconocimiento de imágenes son las redes neuronales, sobre todo cuando no se sabe cuáles son las características locales del objeto a reconocer en las imágenes.

Para hacer un reconocimiento de imágenes a través de una red neuronal, de manera general, lo que se debe hacer, en primera instancia, es tener un conjunto de imágenes del objeto a reconocer para que de ellas se extraigan las características locales comunes entre las imágenes, estos serán nuestros valores de prueba y a partir de ellos es como la red neuronal entrará en una etapa de aprendizaje con la finalidad de que

reconozca que todo ese conjunto de imágenes pertenece a un mismo objeto. Una vez que se termina esta etapa al introducir las características locales de una nueva imagen, estas serán entrada para la red neuronal para que se compare el conocimiento adquirido (imágenes de aprendizaje), con la imagen que contiene diferentes características y sea capaz de identificar si se trata del mismo objeto o no, a esta etapa se le llama reconocimiento. A mayor cantidad de imágenes de muestra con diferentes variaciones para la fase de aprendizaje, la red neuronal identificará mejor los objetos de las imágenes, aunque presenten grandes variaciones.

Redes Neuronales Artificiales

El cerebro humano tiene un mecanismo de funcionamiento muy complejo, ya que puede realizar una diversidad de tareas como son pensar, recordar y resolver problemas, por lo que el hombre y las computadoras realizan tareas muy diferentes, la computadora funciona básicamente para realizar procedimientos repetitivos, por este motivo es que los científicos han intentado reflejar el comportamiento del cerebro humano en las computadoras, lo que dio nacimiento a las redes neuronales artificiales.

Las redes neuronales artificiales (RNA) basan su funcionamiento y tienen elementos análogos a las redes neuronales biológicas; por ejemplo, cada neurona es representada por una unidad de procesamiento que recibe información o señales de otras neuronas, para entregar una sola salida hacia el exterior o hacia otra neurona. A los ligamentos que existen entre las neuronas se le denomina sinapsis, ella permite a la red neuronal aprender con el tiempo. Ésta es representada mediante algún peso, es decir, un valor numérico, el cual varía dependiendo del conocimiento adquirido. Las redes neuronales artificiales deben realizar ciertas funciones para comportarse como una red neuronal biológica, como son poder aprender con base en la experiencia, para después generalizar el conocimiento nuevo y el ya adquirido permitiendo extraer características principales del conocimiento. A continuación, se muestra una breve descripción de estas etapas:

- **Aprendizaje**: En esta etapa la RNA adquiere conocimiento del exterior, permitiendo entrenarla, ajustando los pesos sinápticos dependiendo del cocimiento adquirido. Esta fase es vital, porque el conocimiento de entrenamiento que se presente a la RNA, define el comportamiento que tendrá al obtener nuevo conocimiento.
- **Generalización**: Esta etapa hace referencia al hecho de que las redes neuronales por su naturaleza, son capaces de tener un margen para clasificar o reconocer nuevo conocimiento adquirido, a pesar de que éste muestre pequeñas variaciones respecto a los patrones que se le dieron en la etapa de aprendizaje.
- **Abstracción**: Las RNA deben ser capaces de abstraer aquellas características en común que presenten los patrones de entrada.

Como se mencionaba anteriormente las RNA hacen uso de unidades de procesamiento para representar a las neuronas, existen conexiones entre neuronas (sinapsis) que tiene un cierto valor numérico y representan el aprendizaje adquirido. Las neuronas pueden recibir varias entradas, pero únicamente tener una salida.

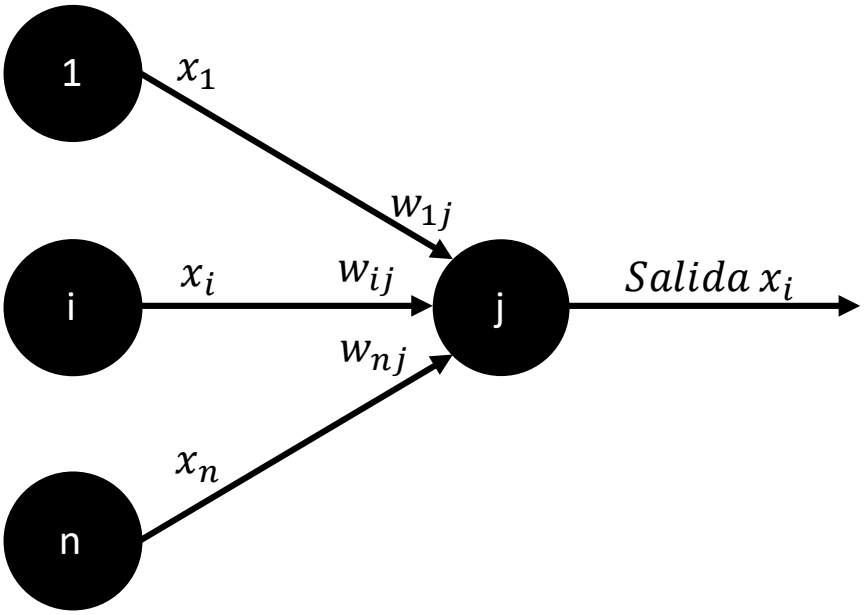


Figura 2.3. Esquema de una neurona artificial.

En primera instancia, se observa en la figura 2.3 que las neuronas 1 a n envían una serie de señales $(x_1, \dots, x_i, \dots, x_n)$ a la neurona j los pesos sinápticos $(w_{1j}, \dots, w_{ij}, \dots, w_{nj})$ sirven como señal excitadora para que la neurona j se active, realice su procesamiento y genere una salida o únicamente se inhiba. El valor total de entrada a la neurona j, depende del conjunto de valores de entrada que reciba a partir de las conexiones sinápticas; generalmente se hace una suma de los valores de entrada, multiplicados por el peso correspondiente a la conexión, esto es:

$$j_{entrada} = \sum x_{ij} w_{ij}$$

Existen diferentes formas de modelar una RNA como son:

- Perceptrón simple

  Es una estructura bastante sencilla, en la que se tiene un conjunto de neuronas de entrada que envían impulsos a una única neurona de salida, siendo ésta capaz de clasificar al objeto que tiene en su entrada. Es útil para poder reconocer patrones, pero únicamente resuelve aquellos problemas que son linealmente separables. La configuración se ilustra en la figura 2.4.

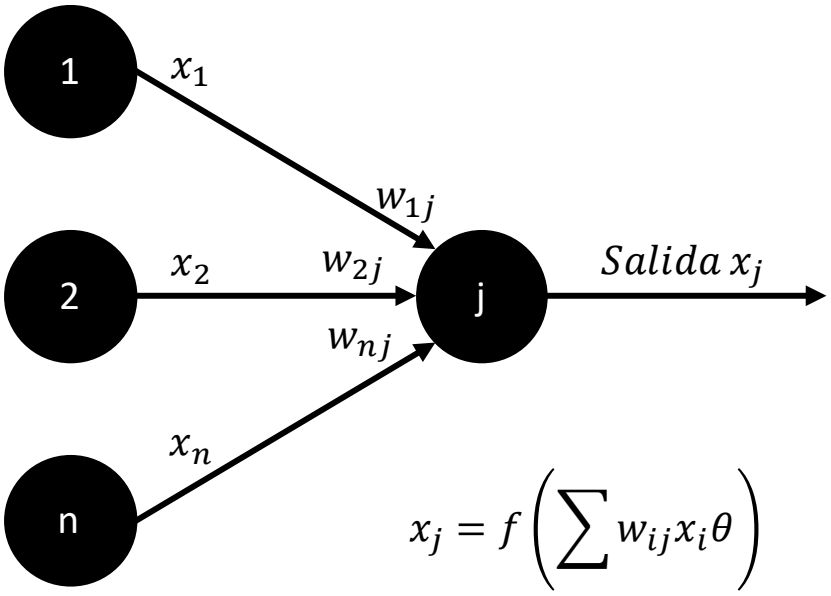


Figura 2.4. Arquitectura de un perceptrón.

La red neuronal de entrada va a realizar la suma de las entradas multiplicadas por los pesos asociados al enlace y le resta cierto umbral (Θ), este resultado pasa a una función de tipo escalón unitario; si el valor de la suma es 1, entonces el patrón pertenece a la clasificación A y si la suma es -1 el patrón pertenece a la clasificación B, podemos notar que este modelo solo permite tener dos clasificaciones de un patrón.

- Perceptrón multicapa

  Principalmente se basa en la utilización de varias capas de neuronas en lugar de una como el perceptrón simple. Además, la función de activación del perceptrón simple, que es una señal escalón, se cambia por una señal del tipo sigmoidal, que es una función diferenciable y no lineal, por lo que podrá resolver aquellos problemas que no sean linealmente separables. Su configuración se muestra en la figura 2.5.

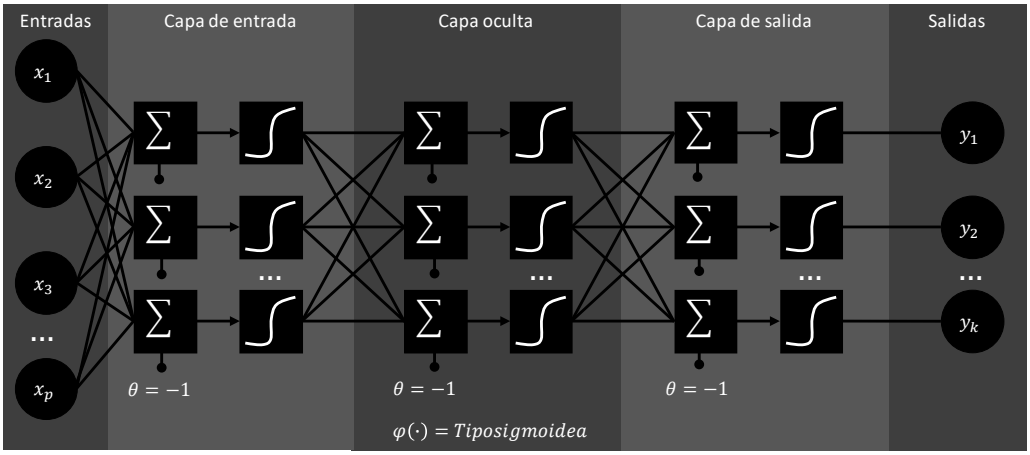


Figura 2.5. Esquema de un perceptrón multicapa.

Cabe destacar que ahora no se tiene un conjunto de neuronas a la entrada que envían sus señales a una única neurona de salida, sino que están enlazadas con una capa oculta de neuronas, que tiene tantas capas de neuronas como se desee. Se le llama capa oculta porque estas neuronas no interactúan con el exterior, además vemos que no se tiene una única neurona sino un conjunto de neuronas de salida. Como se mencionó anteriormente, las neuronas clasifican a los objetos a partir de una función sigmoidal para poder resolver problemas que no sean linealmente separables. La entrada de cada neurona se calcula de la misma forma que en el perceptrón simple, que es realizar la suma de las entradas multiplicadas por los pesos asociados al enlace y le resta cierto umbral (Θ).

Una de las más grandes aplicaciones es la clasificación de patrones, que se puede utilizar para el reconocimiento digital de imágenes.

Árboles de decisión

Un árbol de decisión es una técnica en la cual se toma una serie de decisiones de acuerdo a las características de un evento u objeto para llegar a un resultado final. Este resultado puede ser una interpretación, una descripción o una clasificación.

También permite representar una serie de sucesos que se derivan al tomar una decisión. Se basa en el uso de resultados y probabilidades asociadas.

Para construir un árbol de decisión, es necesario contar con una serie de datos de entrenamiento que definan las características del evento u objeto con sus posibles resultados. A mayor cantidad de datos de entrenamiento que muestren las diferentes variaciones en sus características, habrá mayor precisión en los resultados.

Permite la generalización, es decir, al crear el árbol es posible obtener un resultado a partir de las características asociadas al evento u objeto, aunque estas no se encuentren en los datos de entrenamiento siempre y cuando las variaciones con respecto a estos datos no sean grandes.

Gráficamente está compuesto por:

- **Nodos de decisión**: Indican las decisiones que deben tomarse para llegar a un resultado.
- **Rama**: Muestran las diferentes rutas que pueden tomarse a partir de una decisión para llegar a otras decisiones o a un resultado.
- **Probabilidad**: Indica una prueba aleatoria para tomar o no una cierta decisión o resultado.

Sistema de cómputo embebido

Un sistema de cómputo embebido (SCE) es aquel conjunto se software y hardware que desempeña tareas específicas para un solo propósito, debe contar con las capacidades de procesamiento necesarias para desempeñar dicha tarea y ser simple por razones de costo además de cumplir con las siguientes características:

- Procesador capaz de soportar las tareas a realizar en el tiempo estimado.
- Cantidad de memoria principal suficiente para los programas que ejecutará.
- Contar con interfaces de entrada y salida que se adapten a la implementación.
- Almacenamiento suficiente para alojar al sistema operativo en caso de existir, los programas y considerar espacio extra para almacenar datos o bien expansiones futuras.
- Debe consumir poca energía.

Además, un SCE puede contar con lo siguiente:

- Un sistema operativo a la medida.
- Multiprocesamiento para mejorar el tiempo de respuesta.
- Interfaz de red, en caso de requerir comunicación con otros equipos.

Un SCE es flexible y escalable a diferencia de un circuito integrado que realice la misma tarea, ya que existe la posibilidad de cambiar las funciones del SCE modificando el software que lo compone.

Un SCE cuenta con los siguientes elementos:

- Microprocesador.

  Encargado de realizar operaciones aritméticas y lógicas con el fin de ejecutar el programa que realizará las tareas del sistema, además de controlar los demás elementos de hardware.

- Memoria principal.

  Es aquella que almacena los procesos y datos utilizados por el microprocesador, normalmente son de tipo volátil, es decir, al quedarse sin energía pierde la información. Cuenta con capacidades de almacenamiento reducidas, únicamente aquellos sistemas que fueron diseñados para ejecutar un sistema operativo requieren memorias de mayor capacidad.

- Memoria secundaria.

  Es donde se alojan los programas que ejecutará el sistema, información útil para éstos y en su caso almacenar el sistema operativo. Es una memoria no volátil y su capacidad de almacenamiento es mayor al de la memoria principal.

- ROM-BIOS (Basic Input and Output System).

  Aquí se almacena el código necesario para inicializar el sistema.

- CMOS-RAM

  Almacena información de los componentes conectados al microprocesador, puede contener el reloj del sistema. En algunas ocasiones éste es reemplazado por un chip set.

- Chip Set.

  Es un controlador de buses, es decir, controla la comunicación entre el procesador y otros componentes, así como las interrupciones asociados a ellos. Suele estar integrado en el mismo encapsulado del microprocesador. Puede contar con elementos como son el reloj del sistema, temporizadores, etc.

Un sistema embebido contiene actuadores, sensores y el SCE que permiten desarrollar tareas específicas para resolver un único problema.

Sistemas operativos embebidos

Es un sistema operativo diseñado para ejecutar tareas específicas y que se implementa sobre un sistema embebido.

Es común usar un sistema operativo embebido (SOE) ya que permite ampliar las capacidades generales del sistema y realizar implementaciones con mayor sencillez, ya que este se encarga de administrar los recursos de hardware que son limitados, dirigir los procesos, gestionar información e implementar seguridad.

La mayoría de los SOE modernos contienen un núcleo capaz de manejar protocolos de red como TCP/IP, ethernet, entro otros; además de administrar periféricos con diversas interfaces como lo son USB, firewire, bluetooth, wi-fi, etc. Suelen utilizar muy poca memoria RAM y ROM, requieren bajas capacidades de procesamiento y deben ser tolerantes a fallas.

Hardware libre

Es aquel que persigue los mismos principios que el software libre, es decir:

- Libertad de uso.
- Libertad de compartir.
- Libertad de modificar.
- Libertad de distribución de nuevas versiones.

Sin embargo, se presenta una serie de inconvenientes, para poder compartir un elemento de hardware, es necesario volver a fabricarlo, lo que representa un costo económico y de tiempo; además, es necesario hacer pruebas para verificar el correcto funcionamiento. Otro problema que puede presentarse, es la disponibilidad de los componentes.

Al adquirir hardware libre se debe atender tres aspectos, el diagrama esquemático, el diseño del PCB (Printed Circuit Board – Plaqueta de Circuito Impreso) y la fabricación; hay que considerar que existen restricciones para el uso del software en el que se diseña el hardware, incluso del sistema operativo en el que se ejecuta, como son el uso de software libre o la necesidad de adquirir licencias.

Sistemas embebidos en tiempo real

En un sistema embebido en tiempo real (SETR) el tiempo de respuesta es menor al tiempo en el que la entrada presenta una variación, es decir, responde antes de que la entrada cambie de valor, lo que permite que la información tratada por el sistema sea siempre vigente.

Algunas de las características de estos sistemas son:

- Las acciones de repuesta deben ser correctas y ejecutarse en un intervalo de tiempo determinado.
- Responder antes de que se genere una interrupción por entrada al sistema.
- Los procesos que se ejecutan en el sistema tienen establecidos el tiempo y recursos a utilizar.
- No debe fallar y no debe degradarse la calidad de su servicio.
- Gestión de memoria poco exigente, usualmente los procesos residen en memoria principal.
- En caso de falla debe cumplir con actividades críticas y de más alta prioridad.
- Estabilidad.

Redes de comunicaciones

Una red de comunicaciones (RC), es un conjunto de dispositivos que se comunican entre sí para compartir información y recursos a través de un medio físico empleando un protocolo de comunicación. Las RC se utilizan con diferentes propósitos según el lugar donde se implementen, que puede ser en empresas, en hogares, en comercios, entre otros.

En las RC existe por lo menos un emisor y un receptor que se comunican a través de un medio, terrestre o aéreo, utilizando un conjunto de normas y procedimientos para la transmisión de información que gobiernan formatos, modos de acceso, secuencias temporales, etcétera.

Topologías de red

Es la forma en que se encuentran conectados los dispositivos de una red.

- Malla.

  Es aquella en donde cada dispositivo se encuentra conectado al resto, es posible transmitir información de un dispositivo a otro a través de diferentes caminos; es tolerable a fallas, pero a un costo elevado debido a que se requiere gran cantidad de infraestructura para su construcción.

- Anillo.

  Es aquella en donde cada dispositivo cuenta con una única conexión de entrada y otra de salida para conectarse al dispositivo siguiente, formando un circuito cerrado. La información se transmite de dispositivo a dispositivo llegando a su destino y continúa transmitiéndose hasta regresar al origen. En caso de falla de un dispositivo se ve afectado el circuito de comunicación, por lo que se deben tomar medidas específicas.

- Estrella.

  Es aquella donde cada dispositivo está conectado a un punto central que se encarga de controlar la comunicación en la red. En el caso de falla de un dispositivo la red no se ve afectada, pero si el punto central es el que falla, se pierde toda la comunicación.

- Árbol.

  Los dispositivos se encuentran conectados en forma jerárquica, es parecida a una serie de redes en estrella interconectadas. Existen dispositivos de enlace troncal que interconectan las redes en estrella. En caso de falla de un dispositivo, la red no se ve afectada, si falla un dispositivo de enlace troncal, los dispositivos interconectados a éste pierden comunicación.

- Bus.

  Es aquella donde los dispositivos se encuentran conectados a un único canal de comunicaciones que utilizan para comunicarse entre sí. Sólo un dispositivo puede transmitir datos a la vez, si un dispositivo falla la red no se ve afectada, si el medio de comunicación es el que falla se pierde la comunicación.

Clasificación según su cobertura

Las redes se pueden clasificar según su extensión física en las siguientes categorías:

- PAN (Red de área personal).

  Este tipo de redes sirven para interconectar los dispositivos que pertenecen a un solo individuo, tiene un corto alcance, normalmente se extiende a 10 metros.

- LAN (Red de área local).

  Es un conjunto de dispositivos interconectados que pertenecen a una misma organización u hogar en un área geográfica pequeña. Normalmente cuenta con una tasa de transferencia de datos entre 10 Mbps a 1 Gbps.

- MAN (Red de área metropolitana).

  Permiten la interconexión de redes LAN cercanas geográficamente como los es una ciudad. Ofrece la ventaja de que dispositivos remotos se comuniquen como si pertenecieran a una misma red de área local. La tasa de transferencia de datos suele ser mayor o igual a 155 Mbps.

- WAN (Red de área amplia).

  Similares a las redes MAN, pero su cobertura geográfica es del orden del tamaño de un país o de un continente. La tasa de trasferencia depende del costo de las conexiones.

Medios de transmisión

La transmisión de información puede realizarse a través de medios terrestres y aéreos, cada uno con sus propias características que se describen a continuación:

Terrestres o guiados

- Cable coaxial.

  Permite transportar información en forma de señales eléctricas de alta frecuencia, posee dos conductores, uno central elaborado de cobre encargado de transportar la información y otro exterior de forma tubular llamado malla que sirve como referencia a tierra; entre ellos se encuentra una capa aislante, todo el conjunto está protegido por una cubierta aislante.

  Es difícil de instalar debido a su rigidez, suele usarse en equipos de TV y centros donde se requieren altas tasas de transferencia de información. Solía usarse en la construcción de redes LAN, pero por su elevado costo fue reemplazado por otros medios.

  Posee una tasa de transferencia de 10 Mbps.

- Par trenzado.

  Es aquel que cuenta con uno o más pares de hilos de cobre aislados y entrelazados para anular interferencias externas y diafonía de los cables adyacentes. Es muy sensible al ruido lo que reduce su tasa de transferencia.

  Existen diferentes tipos de este cable según sus características, UTP y STP. Para el caso del UTP los pares trenzados se encuentran sin blindaje, es decir, sin una capa metálica conectada a tierra protectora de interferencias, al contrario del STP que si lo posee. Las tasas de transferencias suelen ser de 10 Mbps, 100 Mbps, 1 Gbps y 10 Gbps.

  Este tipo de cable es comúnmente utilizado en la construcción de redes LAN por su bajo costo y fácil manipulación.

- Fibra óptica.

  Consiste en un hilo muy fino de vidrio o materiales plásticos por el que se envían pulsos de luz provenientes de un láser o un led, para transmitir la información. A diferencia de los anteriores, este es inmune al ruido externo y permite muy altas tasas de transferencia que suele ser de 100 Gbps.

Aéreos

- Infrarrojo.

  Es un tipo de radiación electromagnética con una longitud de onda mayor que la de la luz visible, como consecuencia, no es capaz de atravesar objetos sólidos. Las conexiones pueden ser punto a punto, en donde los dispositivos tienen que verse directamente, o bien, difusa, donde la radiación se refleja en superficies que permiten hacerla incidir en todo el recinto y el dispositivo receptor no tiene que estar enfrentado con el emisor. Poseen una tasa de transferencia de aproximadamente 4 Mbps.

- Radiofrecuencia.

  Es un tipo de radiación electromagnética con una longitud de onda mayor al de la luz infrarroja, por lo tanto, es capaz de atravesar algunos objetos sólidos permitiendo su uso para la construcción de redes con una cobertura geográfica amplia. Poseen una tasa se trasferencia que oscila desde los 10 Mbps a los 1500 Mbps.

Estándares más comunes

802.3 (ethernet)

Define características de cableado y señalización a nivel físico. Permite la comunicación entre dispositivos a través de protocolo de acceso al medio compartido CSMA/CD (acceso múltiple al medio con detección de

onda portadora y detección de colisiones), es decir, los dispositivos determinan si el medio de transmisión y sus recursos se encuentran libres antes de transmitir. Su nombre proviene del concepto de éter.

Esta tecnología ha sido la más usada en construcción de redes LAN debido a que su instalación es muy sencilla y de bajo costo, la red es poco susceptible a fallas y su actualización es viable.

802.11 (Wi-Fi)

Define características de una red LAN inalámbrica. Permite la comunicación entre dispositivos a través del protocolo de acceso al medio compartido CSMA/CA (acceso múltiple con detección de portadora y prevención de colisiones). Si una red cumple con este estándar, se garantiza que cuenta con la certificación otorgada por la Wi-Fi Alliance garantizando la compatibilidad entre los dispositivos que la utilizan.

Con esta tecnología es posible crear redes de alta tasa de transferencia siempre y cuando los dispositivos que se comuniquen se encuentren suficientemente cerca, su alcance normalmente es de 100 metros, pero se ve disminuido al atravesar objetos sólidos.

Esta tecnología ha sido la más usada en construcción de redes LAN inalámbricas debido a que su instalación es muy sencilla y de bajo costo además de permitir a múltiples usuarios hacer uso de la red concurrentemente. Algunas de las desventajas de esta tecnología es su baja tasa de transferencia en comparación con ethernet y su posible vulnerabilidad a ataques.

802.15 (Bluetooth)

Define características de una red PAN inalámbrica, permite la transmisión de voz y datos entre dispositivos personales de manera sencilla.

Debido a que esta tecnología opera sobre la misma banda de frecuencias que Wi-Fi, el estándar 802.15 define características que permiten que ambas interactúen en el mismo medio.

802.16 (WiMAX)

Define características de una red MAN inalámbrica, permite ofrecer servicios a largas distancias que pueden llegar hasta los 50 kilómetros, especialmente en lugares donde la instalación de cableado resulta costosa. Permite la transmisión voz, video y datos (hasta 300 Mbps) en entornos difíciles, además ofrece mayor estabilidad y seguridad que los estándares anteriores.

Dispositivos de interconexión

Tarjeta de red

Es un dispositivo periférico que permite conectar un equipo a una red de comunicaciones mediante un protocolo definido por algún estándar, puede dar soporte de acceso a redes terrestres o inalámbricas. Tienen un identificador único definido por la IEEE llamado dirección MAC (control de acceso al medio).

Hub

Es un dispositivo que permite centralizar el cableado en una red de comunicaciones, ocupan la función de punto central, en una topología de estrella o dispositivo de enlace troncal en una topología de árbol, por tal motivo, puede decirse que es un dispositivo capaz de ampliar una red.

Su funcionamiento consiste en recuperar los datos que ingresan a uno de sus puertos y enviarlos a través de los demás, no es capaz de dirigir el tráfico en la red, se producen colisiones que impiden la fluidez del tráfico. Los hay en dos tipos, activos y pasivos. Los activos son aquellos conectados a una fuente de alimentación eléctrica permitiendo amplificar la señal de salida, los pasivos envían la señal de salida sin amplificar.

Switch

Es un dispositivo que permite conectar equipos a una red de comunicaciones para formar redes LAN, sus especificaciones técnicas siguen al estándar IEEE 802.3 y su uso se limita a redes que funcionan a través de medios terrestres. Generalmente las redes en topología de estrella lo ocupan como punto central y las de topología de árbol como dispositivo de enlace troncal.

Su funcionamiento se basa en el análisis de los datos de entrada en uno de sus puertos, filtrándolos y enviándolos únicamente a los puertos que permiten la llegada a su destino. La transferencia de datos es mayor a la del hub debido a que su actividad se realiza sin colisiones.

Punto de acceso

Similar al switch, permite formar una red LAN, pero de manera inalámbrica, además, este dispositivo puede ser usado como repetidor para ampliar la cobertura de la red. Sus especificaciones técnicas siguen al estándar IEEE 802.11.

Router

Es un dispositivo que permite encaminar paquetes de datos de una red a otra por el camino más adecuado, para ello, almacena los paquetes recibidos para procesar la información de origen y destino e identificar la ruta que seguirá para llegar a éste.

Una dirección IP es un número único que permite identificar a un dispositivo conectado a una red, una dirección IP puede repetirse siempre y cuando no pertenezca a la misma red. En el protocolo IPv4 este número se compone de cuatro grupos, cada uno del 0 al 255, separados por puntos. Cada paquete de datos cuenta con la dirección IP origen y destino.

El enrutamiento trata a los paquetes con sus respectivas direcciones IP, en función de la red a la que pertenecen, dirigiéndolos de acuerdo a un algoritmo de enrutamiento y a una tabla asociada.

Bases de datos espaciales

Una base de datos espacial es capaz de almacenar tanto información geográfica como datos alfanuméricos; son utilizados para el desarrollo de sistemas de información geográfica. Este tipo de bases requieren un sistema de referencia espacial para definir la localización y relación entre los elementos que las conforman.

Los elementos contenidos en las entidades que la forman se llaman objetos geográficos, tienen dos componentes, la componente descriptiva formada por atributos alfanuméricos, que describen al objeto, y una componente espacial que define su geometría con base a un sistema de referencia. Esto permite obtener información como forma, tamaño, localización y relaciones topológicas entre objetos.

Una relación topológica define la forma en que diferentes objetos espaciales comparten su geometría, se dan a partir de intersecciones, uniones, traslapes y adyacencias entre objetos.

Por lo general, este tipo de bases están formadas por un sistema manejador de bases de datos relacional, al que se le agrega una extensión espacial que permite la manipulación de objetos geográficos.

Tipos de datos

Se definen tipos de datos no existentes en bases de datos relacionales para representar la geometría de un objeto:

- Punto.

  Es un objeto adimensional (dimensión 0) que representa una ubicación única, formado por un par ordenado (x, y).

- Línea.

  Es un objeto unidimensional (dimensión 1) formado por una secuencia de puntos unidos por líneas rectas.

- Polígono.

  Es un objeto bidimensional (dimensión 2) representado por una o más líneas que forman un circuito cerrado.

- MultiPunto.

  Es una colección de puntos.

- MultiLínea.

  Es una colección de líneas

- MultiPolígono.

  Es una colección de polígonos.

- Colección Geométrica.

  Es una colección de puntos, líneas y polígonos.

Los puntos que forman a cada uno de los tipos de datos mencionados anteriormente pueden contar con dos dimensiones adicionales que son la elevación (z) y una medida (m) que refiere a un elemento descriptivo de esa ubicación como puede ser temperatura, población, índice de robo, etcétera.

Al conjunto de objetos geográficos con características en común y que representan elementos de la misma clase se le llama capa temática. Ejemplos de éstas son redes viales (ferrocarriles, carreteras, avenidas, calles), divisiones políticas (países, ciudades, colonias), hidrografía (ríos, lagos, mares), entro otros. Las capas temáticas se representan haciendo uso de los tipos de datos de la lista anterior.

Operaciones algebraicas

Son operaciones que se efectúan entre objetos espaciales que pertenecen a la misma o a diferente capa temática, se definen tanto en la componente descriptiva, como en la espacial, siendo las primeras similares a las operaciones aplicadas en bases de datos relacionales.

- Proyección descriptiva.

  Se define como un corte vertical a una tabla, el resultado de esta operación es una capa temática en donde el componente descriptivo, está formado por un subconjunto del componente descriptivo original, la componente espacial no sufre ningún cambio.

- Selección descriptiva.

  Se define como un corte horizontal a una tabla, el resultado de esta operación es una capa temática con aquellos objetos que cumplen con un predicado P en la componente descriptiva; la componente espacial no sufre ningún cambio.

- Unión descriptiva.

  Se define como la unión de dos capas temáticas, es decir, se obtiene como resultado una única capa temática que contiene los objetos involucrados en la operación, la componente espacial no sufre ningún cambio.

- Unión espacial.

  Se define como el resultado de sobreponer dos capas temáticas. Como resultado, genera una nueva capa temática en donde el componente descriptivo es el resultado de la combinación de los atributos descriptivos involucrados, para la componente espacial, se

genera una nueva geometría determinada por la intersección producida al sobreponer ambas componentes espaciales.

- Selección geométrica.

  Se define como el resultado de realizar una selección sobre la componente espacial de los objetos pertenecientes a una capa temática a través de algunas de las siguientes operaciones:

  - **Área de consulta**. Genera una nueva capa temática que incluye todos los objetos geográficos completos que coinciden, por lo menos, en algún punto del área de consulta.
  - **Área de extracción**. Genera una nueva capa temática que incluye únicamente la geometría de los objetos que intersectan con el área de extracción.
  - **Punto de consulta**. Esta operación contiene todos los objetos cuya geometría contiene un punto en común.

- Fusión.

  Se define como el resultado de realizar la unión geométrica de un conjunto de objetos pertenecientes a una misma capa temática bajo una condición determinada.

3. ANÁLISIS Y DISEÑO DEL SISTEMA

Reconocimiento de nivel de tráfico

En esta sección se desarrollará el módulo que permite reconocer el nivel de tráfico en una determinada zona, para ello es necesario realizar una captura fotográfica y someter la imagen resultante a un procesamiento que permita extraer características que la describan. Una vez que se extraigan las características de la imagen, es posible clasificarlas determinando el nivel de tráfico existente en la zona.

Procesamiento de imagen

Para poder realizar el reconocimiento de una imagen es necesario procesarla, aplicando filtros que permitan desechar elementos que no se desean reconocer. En la figura 3.1, se presentan dos muestras de imágenes que se desean reconocer de manera automática, la primera con un índice de tráfico alto y la segunda con un índice bajo.

Figura 3.1. Muestras de imágenes a reconocer.

Lo más conveniente es obtener el contorno de cada uno de los objetos contenidos en la imagen, siendo los más relevantes, los automóviles, ya que a través de ellos es posible obtener patrones característicos que describan cada tipo de imagen.

Escala de grises

Para obtener los contornos de los objetos, no es necesario conocer su color, por lo que esta característica se puede eliminar, esto se puede realizar aplicando un filtro de escala de grises.

Cada imagen consiste en un arreglo de pixeles, donde cada uno es representado por la combinación de los colores rojo, verde y azul de acuerdo al modelo de color RGB. La intensidad de cada uno de estos colores se define con un valor numérico binario de 8 bits, por lo que cada pixel se representa con 24 bits, 8 por cada uno de los colores.

Al transformar una imagen a escala de grises, cada pixel se representa con un único valor de 8 bits que indica qué tan oscuro o claro es, de acuerdo al promedio ponderado de los tres colores que tenía antes de la transformación.

El resultado de aplicar este tipo de filtro a la figura 3.1, se muestra en la figura 3.2.

Figura 3.2. Muestra de imágenes con el filtro de escala de grises aplicado.

Difuminado de la imagen

En las imágenes se observan gran cantidad de detalles que no se desean reconocer, para eliminarlos es conveniente obtener una imagen menos nítida. Para ello se deben eliminar las variaciones bruscas en la intensidad de color en la imagen, reemplazando el valor de cada pixel por la media de los pixeles que están a su alrededor, esto hace que las variaciones de color sean suavizadas. Los filtros utilizados para esta tarea son conocidos como filtros paso bajo, algunos ejemplos son.

- Filtro de la media
- Filtro de media ponderada
- Filtro de la mediana
- Filtros adaptativos
- Filtros gaussianos

En este caso se hizo uso de un filtro gaussiano; éste simula una distribución gaussiana en donde el pixel central tiene un valor máximo y los pixeles de alrededor van disminuyendo su valor conforme se van alejando, es decir, al recalcular el valor de un pixel en particular a través de los pixeles de su alrededor, los más cercanos tendrán mayor influencia en su valor final.

El resultado de aplicar este tipo de filtro a la figura 3.2, se ilustra en la figura 3.3.

Figura 3.3. Muestra de imágenes con un filtro Gaussiano.

Binarización

Para obtener los contornos de las imágenes no es necesario conocer la intensidad de cada pixel sino únicamente reconocer su forma, para ello cada pixel de la imagen se representa con un valor booleano facilitando la tarea de reconocimiento.

El método más sencillo para realizar este filtrado es a partir de un umbral establecido, si la intensidad del pixel supera este umbral se le otorga un valor, si es inferior se le otorga el otro valor.

Si este filtro se aplica a la figura 3.3, se obtiene el resultado de la figura 3.4.

Figura 3.4. Muestra de imágenes binarias haciendo uso de un umbral.

Como se observa en la figura 3.4 los objetos que se desean reconocer no se identifican debido a las variaciones de iluminación, por lo que es necesario emplear otro método de binarización.

Existen otros de métodos de binarización que utilizan filtros adaptativos, es decir, para calcular el umbral al momento de binarizar un pixel, se toma en cuenta la intensidad de los pixeles que se encuentran a su alrededor por lo que las variaciones de iluminación no le afectan y el objeto no debe perderse.

El método de binarización con un filtro adaptativo gaussiano, permite calcular el umbral de acuerdo a una distribución gaussiana, es decir, el umbral de cada pixel se calcula de acuerdo al valor de intensidad de los pixeles más cercanos.

Si se aplica este filtro a la figura 3.3, se obtendría el resultado mostrado en la figura 3.5. En ella se observa una mejora respecto a la figura 3.4 porque los objetos que se deseaban identificar se habían perdido.

Figura 3.5. Muestra de imágenes binarias haciendo uso de un umbral calculado con un filtro adaptivo gaussiano.

Detección de bordes

Para poder reconocer los objetos en las imágenes, es necesario identificar sus límites, para ello es necesario hacer uso de un algoritmo de detección de bordes. Algunos ejemplos son:

- *Roberts*
- *Sobel*
- *Laplaciano*
- *Canny*

En este caso se hizo uso del algoritmo de *Canny,* que se basa en tres criterios:

1. La detección de los bordes importantes sin detectar falsos bordes.
2. Los bordes se deben localizar en el borde real de la imagen.
3. Los bordes deben marcarse una sola vez.

Este algoritmo busca variaciones bruscas en la intensidad de la imagen, es decir, al no existir variaciones o encontrarse variaciones constantes en la intensidad de la imagen, no debe marcarse borde alguno, pero sí en el caso que existan cambios violentos.

El resultado de la identificación de bordes con el algoritmo de *Canny* aplicados a la figura 3.5 se muestran en la figura 3.6.

Figura 3.6. Bordes detectados en las imágenes de muestra por el medio del algoritmo de Canny.

Extracción de características

Al aplicar los filtros anteriores se obtiene una imagen simplificada en donde existen los bordes de los objetos a reconocer sin los detalles que no interesan. Se procede a realizar un análisis que permita extraer valores numéricos que describan la imagen. Se espera obtener valores que permitan diferenciar a las imágenes en donde existe tráfico y aquellas en donde no.

La extracción de estos valores numéricos se basa en la identificación de patrones característicos que no varían en las imágenes. Una de las opciones es contar el número de contornos que se encuentran en cada imagen, como se puede observar en la figura 3.6 hay un mayor número de contornos en la imagen con tráfico que en donde no lo hay.

Otro enfoque es extraer características que sean invariantes a la rotación y a la escala de los objetos a reconocer porque los automóviles presentes en la imagen pueden encontrarse en distintas posiciones.

Una de las propiedades numéricas en una imagen inmune a las variaciones de rotación y escala son los siete momentos invariantes a la traslación, rotación y escala propuestos por Ming-Kuei Hu. Se basan en los momentos geométricos que describen a una imagen, los cuales son parámetros que definen el tamaño, la orientación, la posición y la forma de una imagen.

Los momentos geométricos son:

- **Momentos geométricos simples**: Definen la forma de un objeto a partir de una función $f(x, y)$ en donde (x, y) son las coordenadas de cada pixel en la imagen y $f(x, y)$ es el valor asociado a cada pixel. En las imágenes en blanco y negro, el valor será 0 en caso de que el pixel sea distinto de negro y 1 si es negro.
- **Momentos centrales**: Permiten identificar a un objeto independientemente de su posición, su cálculo se basa en el centroide de los objetos, se localiza en un punto (x, y), en donde todo lo que se encuentra a la derecha o izquierda de la coordenada x tiene la misma masa al igual que lo que se encuentra arriba o debajo de la coordenada y.

- **Momentos centrales normalizados**: Permiten identificar objetos independientemente de su escala.
- **Momentos de orden 0**: Se definen como la suma de los pixeles distintos de blanco; en imágenes en blanco y negro calculan el área de los objetos.
- **Momentos de orden 1**: Son utilizados por los momentos centrales para obtener el centroide de los objetos.
- **Momentos de orden 2**: Permiten el reconocimiento de formas y son utilizados por los momentos centrales, la densidad del objeto se multiplica por distancias al cuadrado desde el centroide.
- **Momentos de orden 3**: Permiten calcular momentos invariantes a la rotación, traslación y escala que se apoyan en los momentos centrales normalizados de orden 2 y 3. A estos también se les conoce como "*Momentos invariantes de Hu*".

Al calcular los Momentos Invariantes de Hu para las imágenes en la figura 3.6 se obtienen los resultados mostrados en la tabla 3.1.

	M1	M2	M3	M4	M5	M6	M7
Con tráfico	0,0149	$1{,}9745e-05$	$8{,}3966e-08$	$1{,}7497e-08$	$1{,}3811e-16$	$-7{,}0148e-11$	$-6{,}5630e-16$
Sin tráfico	0,0234	0,0002	$4{,}2244e-07$	$2{,}9599e-07$	$-8{,}9419e-14$	$-2{,}8722e-09$	$5{,}4400e-14$

Tabla 3.1. Momentos Invariantes de Hu obtenidos para las imágenes de muestra procesadas.

Implementación del procesamiento y extracción de características de una imagen

Para realizar las tareas de procesamiento y lograr extraer las características numéricas que describen a las imágenes de tráfico es necesario hacer uso de una herramienta que facilite el proceso debido a la gran complejidad que implicaría desarrollarlo en un lenguaje de alto nivel.

Al buscar alternativas, la mejor opción encontrada fue la biblioteca *OpenCV*, ya que cuenta con una gran cantidad de funciones que permiten desarrollar aplicaciones de procesamiento de imágenes y visión computarizada de forma sencilla y rápida. Permite visualizar datos e imágenes, así como manipulación de vídeo.

Ventajas:

- Es de código abierto
- Es multiplataforma (*Microsoft Windows, MAC OS, Linux, Android, iOS*)
- Se puede hacer uso de ella a través de diferentes lenguajes de programación (*C, C++, Python, Java*)

Incorpora funciones que facilitan la captura de imágenes como lo es a través de una cámara o lectura de archivos, manipular estas imágenes y extraer características que la describan.

El código 3.1 permite realizar las tareas de filtrado necesarias para procesar la imagen y finalmente extraer los 7 momentos invariantes de Hu que la definen. El programa además muestra los 7 momentos de Hu en pantalla y cada una de las imágenes resultantes al aplicar un filtro.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import numpy
import cv2

captura = cv2.VideoCapture(0)

while(True):
    retval, imagen = captura.read()
    escala_grises = cv2.cvtColor(imagen, cv2.COLOR_BGR2GRAY)
    filtro_gausiano = cv2.GaussianBlur(escala_grises, (9,9), 2.5, 2.5)
    blanco_negro = cv2.adaptiveThreshold(filtro_gausiano, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
cv2.THRESH_BINARY, 11, 2)
    bordes = cv2.Canny(blanco_negro, 0, 1, 150)
    contorno puntos, topologia = cv2.findContours(bordes,cv2.RETR TREE,cv2.CHAIN APPROX SIMPLE)
    cv2.drawContours(imagen, contorno_puntos,-1,(0,255,0),-1)
    momentos = cv2.moments(bordes)
    momentos_hu = cv2.HuMoments(momentos)
    print momentos_hu
    cv2.imshow('1. Imagen en escala de grises', escala_grises)
    cv2.imshow('2. Imagen difuminada', filtro_gausiano)
    cv2.imshow('3. Imagen binaria', blanco_negro)
    cv2.imshow('4. Imagen con bordes', bordes)
    cv2.imshow('5. Imagen final', imagen)
        if cv2.waitKey(1) == ord('a'):
            break

captura.release()
cv2.destroyAllWindows()
```

Código 3.1. Extracción de características de una imagen.

Los valores utilizados en cada una de las funciones de filtrado fueron calculados empíricamente de acuerdo a los resultados más convenientes para reconocer los objetos deseados.

Clasificación

Se procede a utilizar un método de clasificación para identificar si la imagen que se analiza muestra un nivel de tráfico alto o bajo, haciendo uso de los siete momentos invariantes resultado de la extracción de características.

Para poder clasificar las imágenes de manera precisa, es necesario generar un gran número de muestras de imágenes con un alto nivel y otras con un bajo nivel de tráfico, procesarlas y analizar las características que se extraigan de cada una de ellas, para este caso los siete momentos de Hu.

Como se mencionó en el tema anterior, una de las técnicas de clasificación que existen son los árboles de decisión, se optó por utilizar dicha técnica debido a su alto nivel de precisión y al ahorro de recursos de cómputo una vez implementado.

Para construir el árbol de decisión es necesario obtener datos de entrenamiento, para ello debe tomarse una serie de muestras de las diferentes variaciones de tráfico en una vialidad. Estas muestras consisten en un conjunto de imágenes que serán procesadas para finalmente extraer las características numéricas que las definen.

Para obtener las imágenes de entrenamiento y sus respectivos valores característicos, se adaptó el código 3.1 al código 3.2 que permite tomar capturas automáticamente a través de la cámara, procesarlas y almacenar los 7 momentos invariantes de Hu que las describen en un archivo. También almacena la imagen tomada con los respectivos contornos identificados cada minuto guardando las referencias de tiempo.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import numpy
import cv2
import time

def imprime_momentos(momentos_hu):
    archivo.write(time.strftime("%H:%M:%S") + " ")
    for i in range(7):
        archivo.write(str(momentos_hu[i]) + " ")
    archivo.write("\n")

captura = cv2.VideoCapture(0)
archivo = open("entrenamiento/momentos_hu.txt", "a")

while(True):
    retval, imagen = captura.read()
    escala_grises = cv2.cvtColor(imagen, cv2.COLOR_BGR2GRAY)
    filtro_gausiano = cv2.GaussianBlur(escala_grises, (9,9), 2.5, 2.5)
    blanco_negro = cv2.adaptiveThreshold(filtro_gausiano, 255,
cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY, 11, 2)
    bordes = cv2.Canny(blanco_negro, 0, 1, 150)
    contorno_puntos, topologia =
cv2.findContours(bordes,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
    cv2.drawContours(imagen, contorno_puntos, -1, (0,255,0), -1)
    momentos = cv2.moments(bordes)
    momentos_hu = cv2.HuMoments(momentos)
    imprime_momentos(momentos_hu)
    retval = cv2.imwrite("entrenamiento/" + time.strftime("%H.%M.%S") + ".png", imagen)
    cv2.imshow('1. Imagen en escala de grises', escala_grises)
    cv2.imshow('2. Imagen difuminada', filtro_gausiano)
    cv2.imshow('3. Imagen binaria', blanco_negro)
    cv2.imshow('4. Imagen con bordes', bordes)
    cv2.imshow('5. Imagen final', imagen)
    time.sleep(60)
        if cv2.waitKey(1) == ord('a'):
            break

archivo.close()
captura.release()
cv2.destroyAllWindows()
```

Código 3.2. Creación de datos de entrenamiento.

Al ejecutar el programa anterior se obtienen 176 muestras, de las cuales 95 indican tráfico alto y 81 de tráfico bajo, se ilustra el resultado en la figura 3.7.

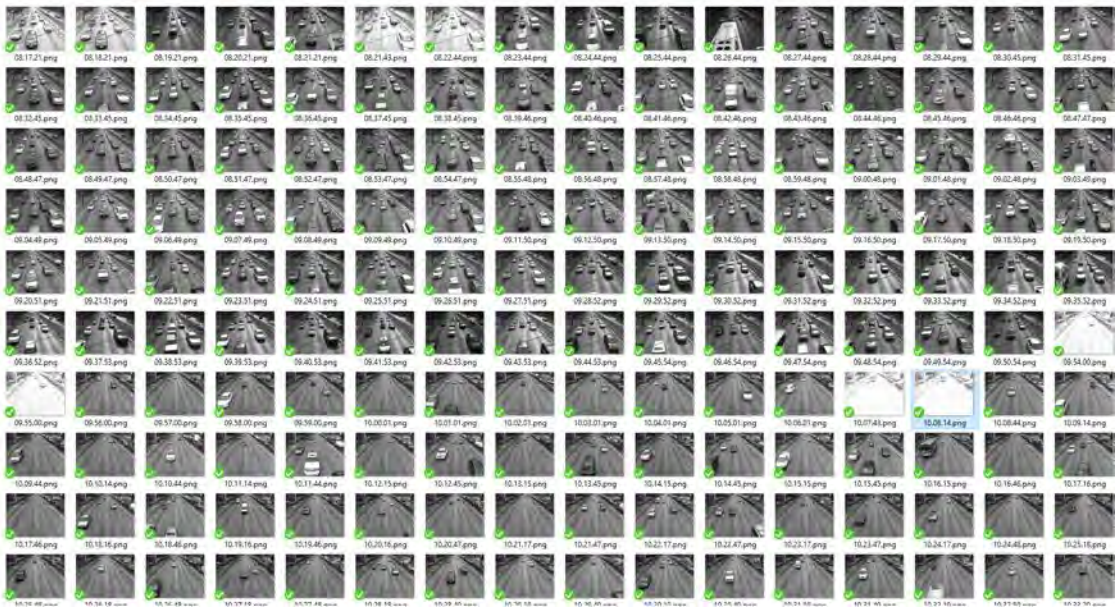


Figura 3.7. Algunas de las imágenes asociadas al proceso de entrenamiento.

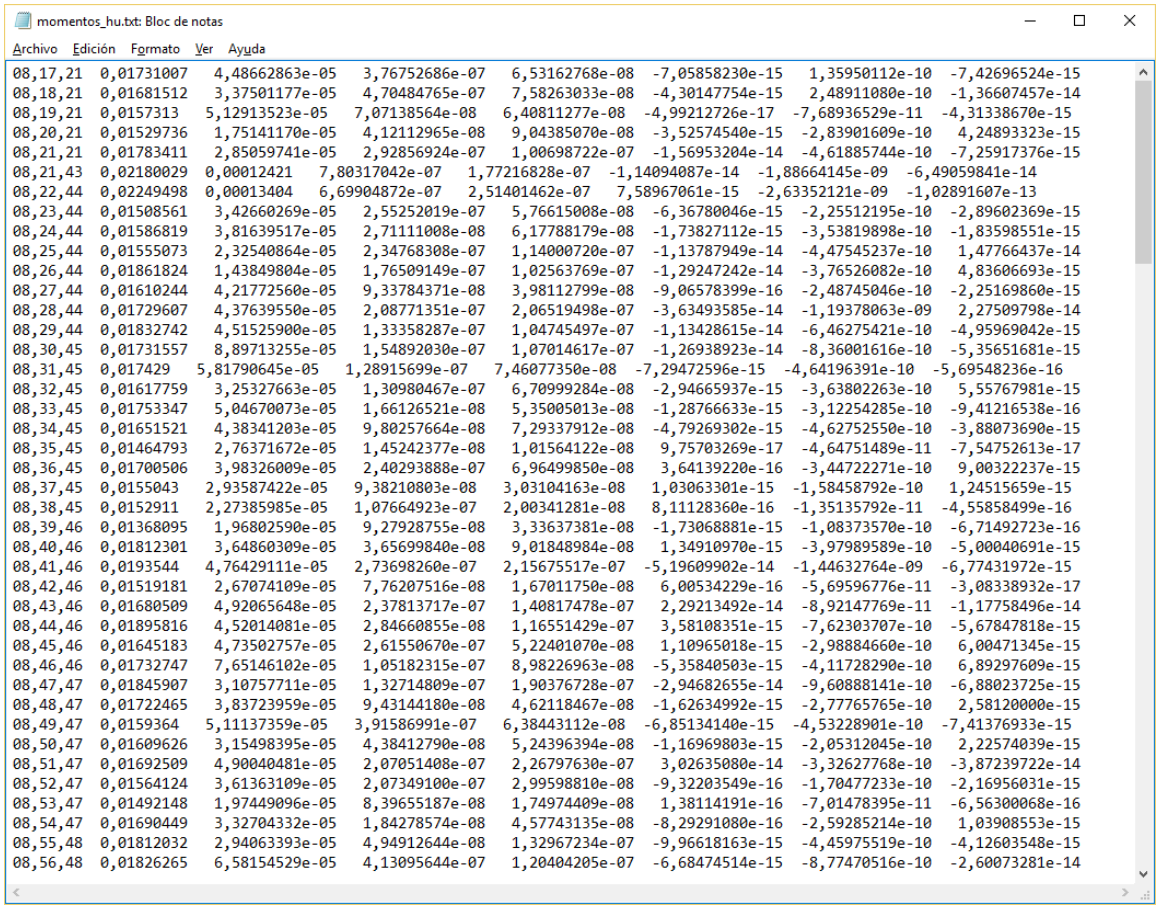


Figura 3.8. Muestra del archivo de texto con las características numéricas que definen a las imágenes.

Los valores generados por el programa de entrenamiento se muestran en la figura 3.8 y permiten la generación del árbol de decisión. Para esta tarea, se requiere hacer uso de una herramienta de análisis de datos debido a la complejidad. En este caso se optó por el uso de la herramienta *RapidMiner Studio Free 7.0.1* que permite realizar tareas de análisis y minería de datos a través de un entorno gráfico, cuenta con más de 500 operadores orientados al análisis de datos como son aquellos para realizar operaciones de entrada y salida, pre-procesamiento de datos y visualización. La herramienta se encuentra desarrollada en el lenguaje de programación Java lo que permite su uso en múltiples plataformas.

Para empezar, es necesario construir el archivo de entrada para *RapidMiner,* éste se realiza a partir de los datos de entrenamiento y se construye en un archivo de Microsoft Excel. La primera columna representa un identificador de las características asociadas a cada imagen, en las siguientes columnas se colocan cada uno de los 7 momentos asociados y finalmente se agrega una columna que sirve para identificar el nivel de tráfico asociado a los momentos de acuerdo a las imágenes, esta columna se llena según lo observado en cada imagen. La tabla de Microsoft Excel generada se ilustra en la figura 3.9.

	A	B	C	D	E	F	G	H	I
1	ID	M1	M2	M3	M4	M5	M6	M7	Nivel_tráfico
86	85	1,7254	4,7809	0,8383	25,3853	35,3813	-19,8588	10,9356	A
87	86	1,8597	4,0358	1,6728	10,4095	13,7359	-2,6476	-0,0727	A
88	87	1,7763	4,9548	3,1713	14,0199	-25,8352	-87,4354	14,3687	A
89	88	1,8299	3,5127	2,4031	8,5085	11,0283	-23,1619	-5,1381	A
90	89	1,6649	3,6351	2,2926	7,3180	-8,1362	-37,3259	-4,8631	A
91	90	1,6558	2,6989	1,8531	4,5688	3,4856	-12,5643	2,3501	A
92	91	1,7084	7,6620	2,1264	15,0238	-22,3539	-108,6736	-14,8790	A
93	92	1,5176	1,4120	0,3171	4,5858	-0,5603	-13,0663	1,6566	A
94	93	1,5029	2,6602	2,4124	3,2299	0,2122	-14,1969	2,8432	A
95	94	1,7125	5,5250	2,2370	4,7727	2,7575	-18,9245	-4,0886	A
96	95	1,8959	7,6644	3,8093	7,6686	-6,2234	-63,4201	-11,5350	A
97	96	3,7814	87,2310	46,0449	357,8466	11689,9044	6591,6565	-8622,1425	B
98	97	3,9071	91,0930	36,3550	242,1793	4880,7913	3120,0772	-5274,1414	B
99	98	2,3581	23,5190	5,5556	27,7632	-88,1445	-258,0375	64,1823	B
100	99	2,4061	22,2290	4,8294	28,0339	-100,9637	-394,0855	-21,1260	B
101	100	2,1584	15,8570	5,1963	21,9774	-36,6957	-111,1655	-64,5708	B
102	101	2,0589	17,5300	5,5968	27,0889	8,8150	47,6743	105,1077	B
103	102	2,3825	22,7110	5,1531	37,6675	-55,3230	-81,8860	156,4593	B
104	103	1,9253	8,0027	2,4959	9,6013	-14,7681	-52,4009	-1,6771	B
105	104	2,3413	21,7050	3,6781	21,0637	-56,6249	-294,2164	-15,1999	B
106	105	2,1900	21,5790	6,1277	37,3623	62,6414	158,0498	167,4373	B
107	106	2,0987	17,6330	3,3048	15,4603	-28,7157	-142,6415	19,9156	B
108	107	2,0238	13,7830	7,4578	35,6202	100,9287	235,0405	153,3579	B

Figura 3.9. Archivo de entrada para *RapidMiner* (base de conocimientos).

Una vez generado el archivo de entrada a la herramienta se procede a colocar una serie de operadores que permitirán la construcción del árbol de decisión (figura 3.10). En el operador Set Role se indica la columna que se desea predecir (nivel de tráfico) además de aquellas que servirán para obtener los resultados de la clasificación (los 7 momentos invariantes de Hu). El operador *X-Validation* permite estimar qué tan preciso es un modelo de datos; se basa en dos subprocesos uno de entrenamiento y otro de prueba, el modelo generado por el entrenamiento es utilizado por el subproceso de prueba y es capaz de estimar el porcentaje de precisión que tendrá en la práctica.

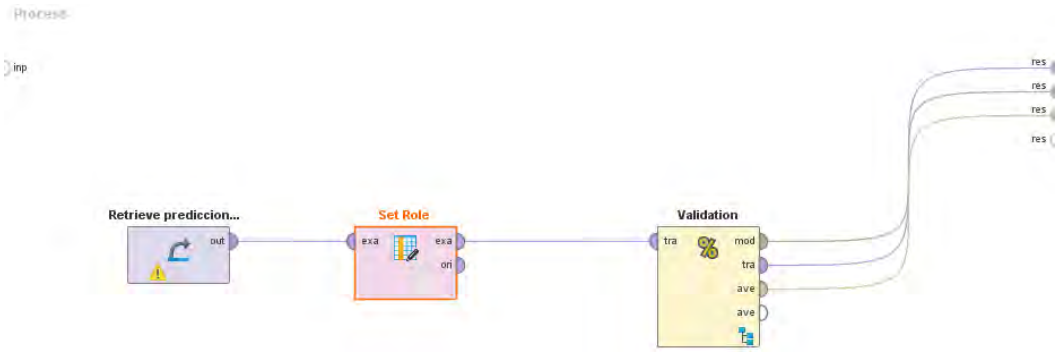


Figura 3.10. Operadores necesarios para la clasificación.

El operador *Decision Tree* permite generar un árbol de decisión, forma parte del subproceso de entrenamiento del operador *X-Validation* y cuenta con algunos parámetros que permiten especificar la

profundidad del árbol y la forma en que tratará la información que lo genera. El operador *Apply Model* toma el modelo generado por la fase de entrenamiento para ser utilizado por otro operador. Finalmente, el operador *Performance (Classification)* regresa una lista de valores de los criterios de rendimiento de la tarea de clasificación como son el error de clasificación, la precisión o el coeficiente de correlación. Estos operadores se muestran en la figura 3.11.

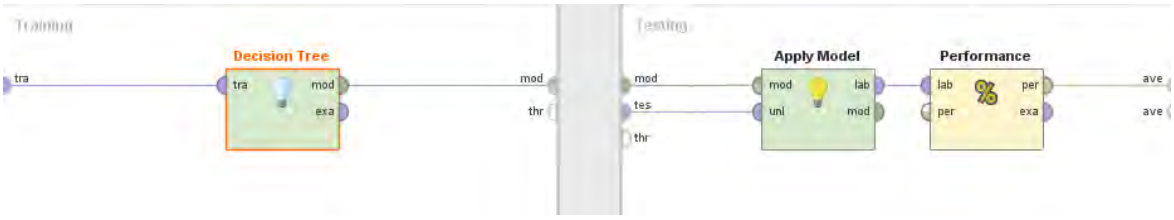


Figura 3.11. Operadores asociados a la fase de entrenamiento y prueba del operador *X-Validation.*

Una vez terminada la construcción del proceso de clasificación es posible observar los resultados que arroja la herramienta. En la figura 3.12 se observan los porcentajes de precisión de la tarea de clasificación. De las 91 muestras con nivel alto de tráfico, 87 fueron clasificadas correctamente por lo que el porcentaje de precisión es de 95.6%. De las 85 muestras con nivel bajo de tráfico, 77 fueron clasificadas correctamente por lo que el porcentaje de precisión es de 90.59%. Debido a los porcentajes anteriores, se concluye que la implementación del árbol de decisión para la tarea de clasificación es viable.

accuracy: 93.22% +/- 8.32% (mikro: 93.18%)

	true A	true B	class precision
pred. A	87	4	95.60%
pred. B	8	77	90.59%
class recall	91.58%	95.06%	

Figura 3.12. Resultados de la clasificación.

En las figuras 3.13 y 3.14 se muestra el árbol de decisión resultante del proceso de clasificación en la herramienta. De acuerdo al procesamiento de datos realizado por la herramienta, sólo tomó los momentos 2 y 3 ($M2\ y\ M3$); éstos son los únicos necesarios para realizar la tarea de clasificación, permiten obtener el árbol de decisión con el menor error de clasificación y mayor porcentaje de precisión. Los nodos hoja representados con una A indican que la decisión final es un nivel de tráfico alto mientras que los nodos hoja representados con una B indican un nivel bajo. Las barras de color rojo y azul representan la probabilidad de que el valor final de la decisión es acertado, en la descripción lógica estos valores se encuentran encerrados entre llaves.

```
M2 > 7.834: B {A=3, B=76}
M2 ≤ 7.834 & M3 > 5.480: B {A=1, B=3}
M2 ≤ 7.834 & M3 ≤ 5.480: A {A=91, B=2}
```

Figura 3.13. Representación lógica del árbol de decisión.

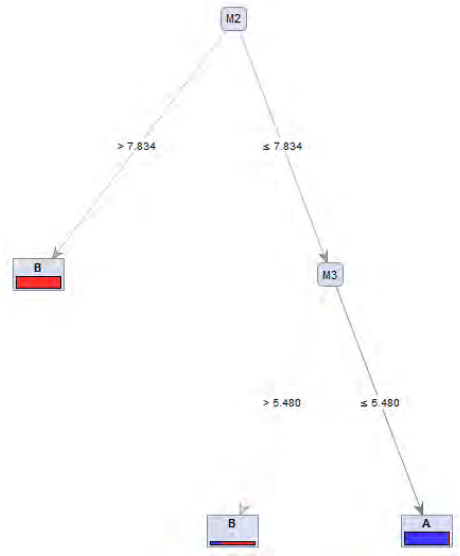


Figura 3.14. Representación gráfica del árbol de decisión.

Este árbol será el implementado en el programa encargado de realizar la tarea de clasificación que permita identificar si el nivel de tráfico que se encuentra en una vialidad es alto o bajo. El código 3.2, que sirvió para la etapa de entrenamiento es modificado para realizar el reconocimiento del nivel de tráfico a partir de una captura.

Implementación del reconocimiento del nivel de tráfico

Para la tarea de reconocimiento del nivel de tráfico a partir de una captura a una vialidad es necesario conjuntar los procesos descritos anteriormente, procesamiento, extracción de características y clasificación. Esta tarea se realiza diez veces para tener una decisión final confiable debido a que la cámara puede agregar ruido a las imágenes y también tomar en cuenta las probabilidades asociadas al árbol de decisión.

El código 3.3 muestra la implementación del reconocimiento de nivel de tráfico realizado a partir de los códigos expuestos antes.

```
#!/usr/bin/python

# -*- coding: utf-8 -*-

#Importación de bibliotecas
import numpy    #Biblioteca para manejo de arreglos y matrices
import cv2      #Biblioteca de opencv
import time     #Biblioteca para manejo de retardos
import random   #Biblioteca para manejo de funciones aleatorias

#Función de probabilidad
def probabilidad(probabilidad):
    if(random.uniform(0, 100) <= probabilidad):
        return True
    return False

#Se inicializa la camara 0
captura = cv2.VideoCapture(0)

#Ciclo infinito para detección de tráfico
while(True):
    i = 0
    estado = []
    ''' El siguiente ciclo permite tomar 10 capturas a través de la camara,
        procesarlas y obtener el nivel de tráfico clasificado en alto y bajo '''
    while(i < 10):
        #Lee la imagen captada por la camara
        retval, imagen = captura.read()
        #Transforma la imagen a escala de grises
        escala_grises = cv2.cvtColor(imagen, cv2.COLOR_BGR2GRAY)
        #Se aplica filtro gaussiano para difuminar la imagen
        filtro_gaussiano = cv2.GaussianBlur(escala_grises, (9,9), 2.5, 2.5)
        #Tranforma la imagen en blanco y negro (binario)
        #antiguo #retval, blanco_negro = cv2.threshold(filtro_gaussiano,127,255,cv2.THRESH_BINARY)
        blanco_negro = cv2.adaptiveThreshold(filtro_gaussiano, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY, 11, 2)
        #Se obtienen los bordes de la imagen haciendo uso del algoritmo de Canny
        bordes = cv2.Canny(blanco_negro, 0, 1, 150)
        #Se obtiene una lista de los puntos que conforman a cada uno de los contornos identificados
        contorno_puntos, topologia = cv2.findContours(bordes,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
        #Se obtiene la imagen que representa los contornos
        cv2.drawContours(imagen, contorno_puntos,-1,(0,255,0),-1)
        #Se calculan los momentos geométricos de tercer orden
        momentos = cv2.moments(bordes)
        #Se calculan los 7 momentos invariantes a la rotación, traslación y a la escala
        momentos_hu = cv2.HuMoments(momentos)
        momentos_hu = momentos_hu.tolist()
        #Se extraen los tres primeros momentos invariantes para realizar la clasificación
        m2 = momentos_hu[1][0] * 100000
        m3 = momentos_hu[2][0] * 10000000

        ''' Árbol de decisión para clasificar el nivel de tráfico según la imagen capturada,
            el valor se almacena en estado, 0 = bajo nivel y 1 = alto nivel '''
        if(m2 > 7.834):
            if(probabilidad(96.20)):
                estado.append(0)
            else:
                estado.append(1)
        else:
            if(m3 > 5.48):
                if(probabilidad(75.00)):
                    estado.append(0)
                else:
                    estado.append(1)
            else:
                if(probabilidad(97.85)):
                    estado.append(1)
                else:
                    estado.append(0)
```

```
        i = i + 1

    print(estado)

    alto = bajo = 0
    ''' De las 10 clasificaciones realizadas anteriormente, se identifica el
        numero de veces que el resultado fue nivel bajo o alto '''
    for i in estado:
        if(i == 0):
            bajo = bajo + 1
        else:
            alto = alto + 1

    ''' Aquel nivel de tráfico identificado un mayor número de veces se toma como
        resultado final de la clasificación '''
    if(bajo > alto):
        print("Se ha detectado poco tráfico")
    else:
        print("Se ha detectado mucho tráfico")

    #Muestra los resultados del procesamiento de cada imagen
    cv2.imshow('Captura1',escala_grises)
    cv2.imshow('Captura2',filtro_gaussiano)
    cv2.imshow('Captura3',blanco_negro)
    cv2.imshow('Captura4',bordes)
    cv2.imshow('Captura5',imagen)

    #Retardo para realizar la próxima identificación de tráfico
    time.sleep(1)

    #Permite terminar el programa al presionar la tecla a
    if cv2.waitKey(1) == ord('a'):
        break

#Se liberan los recursos utilizados
captura.release()
cv2.destroyAllWindows()
```

Código 3.3. Reconocimiento del nivel de tráfico.

Para obtener mejores resultados en el proceso de clasificación es necesario tomar un mayor número de muestras de entrenamiento en donde se tomen en cuenta las variaciones de iluminación que existen a lo largo del día. Para ello se puede hacer uso de un único árbol que se entrene con imágenes en diferentes condiciones de iluminación o bien, hacer uso de diferentes árboles de decisión dependiendo de la hora del día.

Es posible generar diversos árboles de entrenamiento según el entorno en el que se encuentre la vialidad, sin embargo, esto implica demasiado trabajo. Para solucionar este problema es posible ajustar la captura de la cámara para únicamente visualizar la vialidad.

Optimización de tráfico

La instalación de semáforos en una vialidad permite disminuir el tiempo de paso a los automóviles, este intervalo de tiempo debe ser perfectamente ajustado para no entorpecer el flujo automovilístico en una determinada vialidad. Los semáforos se encuentran ubicados en aquellos puntos en donde contribuyan a

mejorar la fluidez y la seguridad del tráfico, en caso contrario los semáforos suponen una problemática para la circulación automovilística.

Si dos o más vialidades siempre cuentan con un bajo nivel de tráfico no debe instalarse un semáforo, en caso de que una de las vialidades cuente con un mayor nivel de tráfico que las otras deberá asignarse un mayor intervalo de tiempo para mejorar el flujo automovilístico, restando tiempo a las vialidades con bajo nivel de tráfico. En caso de que varias vialidades tengan un alto nivel de tráfico, la asignación de tiempos para cruce de automóviles debe hacerse de acuerdo a la preferencia de la vialidad.

Introducción a la problemática

Dentro de las vialidades en una ciudad, los semáforos se encuentran localizados en aquellas zonas donde se presentan cruces entre vialidades, con base a un análisis del cruce identificando los posibles movimientos. Los movimientos que no pueden efectuarse de manera simultánea ya que entorpecen el tráfico y provocan problemas de seguridad, están en intersecciones conflictivas donde hay que instalar semáforos.

Cada semáforo debe estar ubicado en donde sea perfectamente visible por cualquier conductor junto con la señalización correspondiente que evite que el conductor dude acerca del movimiento que pretende realizar. En caso de tener luz roja, el conductor sabrá que debe detener su vehículo, para ello debe marcarse una línea en la vialidad que éste no debe rebasar. En ese punto el semáforo debe estar perfectamente visible y el vehículo no debe obstruir ni al tránsito peatonal ni a la circulación de la vialidad que se interseca.

Por motivos de seguridad y visibilidad cada uno de los semáforos ubicados en una vialidad, deben ser duplicados o incluso triplicados.

Existen intersecciones en donde alguna de las vialidades cuenta con dos sentidos de circulación, para estos casos es necesaria la instalación de un semáforo en cada uno de los sentidos como si se tratara de vialidades distintas. Estos semáforos deben trabajar de manera simultánea ya que ambos sentidos pueden circular sin conflicto, sin embargo, cuando un automóvil pretende incorporarse a la vialidad con la que está cruzando o bien al otro sentido de la misma vialidad, es necesario instalar un semáforo adicional al principal que permita realizar dicho movimiento.

Muchos movimientos de vuelta izquierda no requieren la instalación de un semáforo cuando la vialidad en contraflujo siempre presenta un bajo nivel de tráfico ya que la probabilidad de conflicto disminuye debido a que los vehículos en contraflujo tienen la posibilidad de ceder el paso aprovechando los huecos entre ellos. En este caso la instalación de un semáforo supondría un impacto negativo en el flujo automovilístico.

A veces las vialidades son demasiado estrechas y no es posible la instalación de un semáforo que permita el movimiento de vuelta izquierda ya que no hay espacio suficiente para que los vehículos esperen para realizar el movimiento requerido.

Existen vialidades en donde el movimiento de vuelta izquierda no está permitido, debido a que la vialidad en contraflujo siempre cuenta con un alto nivel de tráfico. Si los intervalos de tiempo de paso de cada uno de los semáforos en una intersección se ajustan de acuerdo al nivel de tráfico identificado en todo momento, es posible permitir el movimiento de vuelta izquierda en intersecciones donde no era posible, añadiendo el semáforo que permite realizar dicho movimiento.

Existen diferentes tipos de intersecciones viales según las características de las vialidades involucradas. A continuación, se revisarán los tipos más comunes.

Intersección en T, vialidades de un solo sentido

Este tipo de intersección está conformada por el cruce de dos vialidades que tienen un solo sentido de circulación. La intersección tiene una forma de T representada en la figura 3.15.

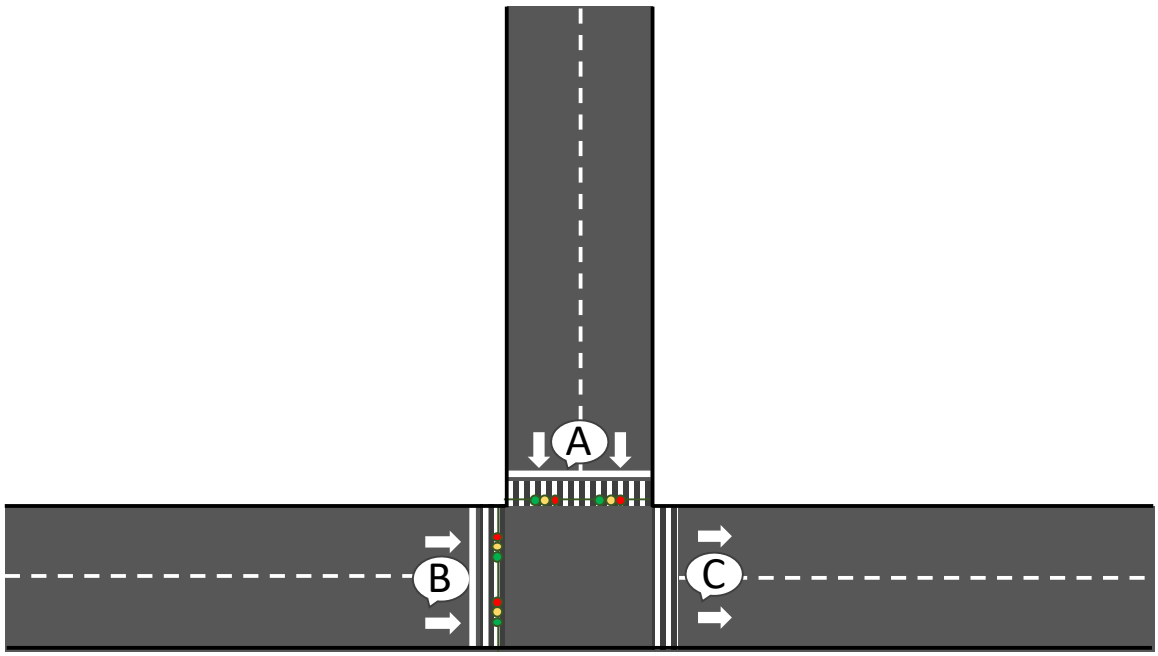


Figura 3.15. Intersección en T, vialidades en un solo sentido.

Para la figura 3.15, muestra la tabla 3.2 que indica los movimientos posibles a realizar en la intersección y los movimientos con los que interfiere de acuerdo a los puntos marcados en ella (A, B, C).

Movimiento	Posible	Movimientos interferidos
AB	No	-
AC	Sí	BC
BA	No	-
BC	Sí	AC
CA	No	-
CB	No	-

Tabla 3.2. Descripción de movimientos en la intersección en T, vialidades en un solo sentido.

Como se observa en la tabla anterior los únicos movimientos posibles son los que van de los puntos A-C y B-C, sin embargo, se interfieren el uno con el otro, por lo tanto, es necesario que estos movimientos se realicen en fases distintas por medio de semáforos que eviten conflictos. La cantidad de semáforos necesaria para evitar conflictos son dos, uno en cada vialidad, los semáforos no deberán operar al mismo tiempo, este comportamiento se muestra en la tabla 3.3.

Fase	Punto con semáforo en luz verde	Punto con semáforo en luz roja	Movimiento permitido
F0	A	B	AC
F1	B	A	BC

Tabla 3.3. Fases para los semáforos implementados en la intersección en T, vialidades en un solo sentido.

Intersección en T, una vialidad en un sentido y otra de dos sentidos

Este tipo de intersección está conformada por el cruce de dos vialidades, una de ellas tiene un solo sentido de circulación mientras la otra tiene dos. La intersección tiene una forma de T, se analizarán algunas de las posibilidades.

POSIBILIDAD I

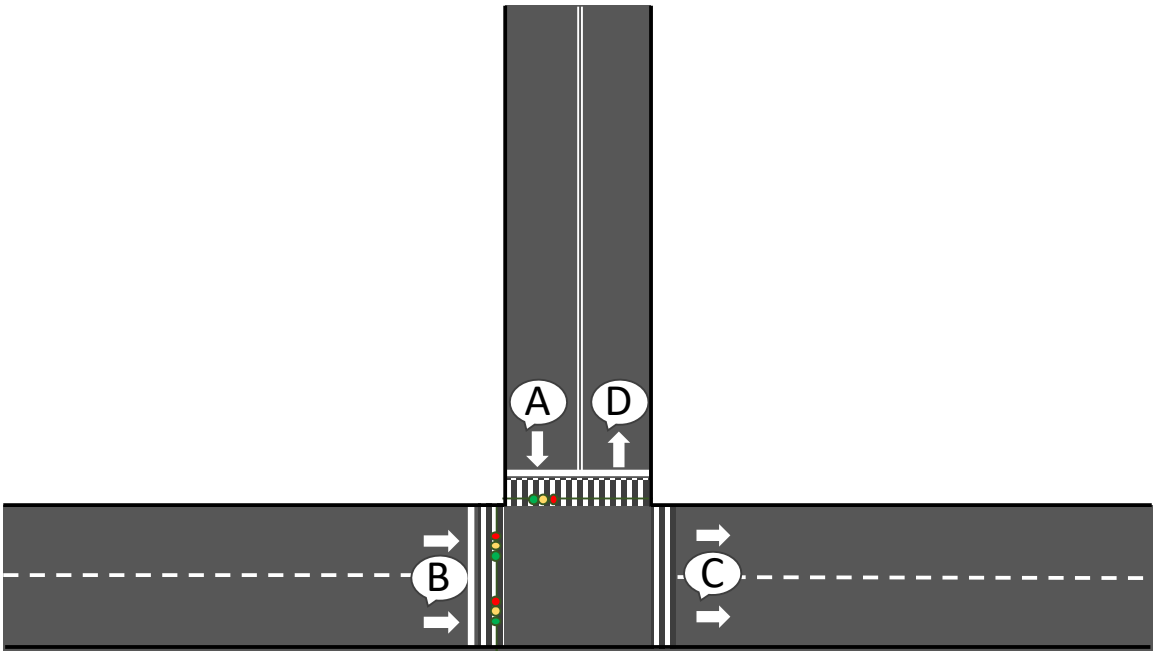


Figura 3.16. Intersección en T, una vialidad en un solo sentido y otra de dos sentidos (Posibilidad 1).

Para la figura 3.16, se muestra la tabla 3.4 que indica los movimientos posibles a realizar en la intersección y los movimientos con los que interfiere de acuerdo a los puntos marcados en ella (A, B, C, D).

Movimiento	Posible	Movimientos interferidos
AB	No	-
AC, AD	Sí	BC, BD
BA	No	-
BC, BD	Sí	AC, AD
CX	No	-
DX	No	-

Tabla 3.4. Descripción de movimientos en la intersección en T, una vialidad en un solo sentido y otra de dos sentidos (Posibilidad 1).

Algunos movimientos pueden no ser interferidos con una infraestructura más adecuada, por ejemplo, un camellón central que tenga un retorno puede permitir el movimiento A-D (vuelta en “U”) sin interferir el movimiento B - C.

A partir de la tabla anterior se observa claramente cuáles son aquellos movimientos que interfieren entre sí, por lo que es necesario que se realicen en diferentes fases colocando semáforos en los puntos de partida de los movimientos, para este caso los puntos A y B. Aquellos movimientos que no presentan interferencia entre sí pueden realizarse de manera simultánea pudiendo prescindir de un semáforo que lo entorpezca. Este comportamiento se muestra en la tabla 3.5.

Fase	Punto con semáforo en luz verde	Punto con semáforo en luz roja	Movimiento permitido
F0	A	B	AC y AD
F1	B	A	BC y BD

Tabla 3.5. Fases para los semáforos implementados en la intersección en T, una vialidad en un solo sentido y otra de dos sentidos (Posibilidad 1).

POSIBILIDAD II

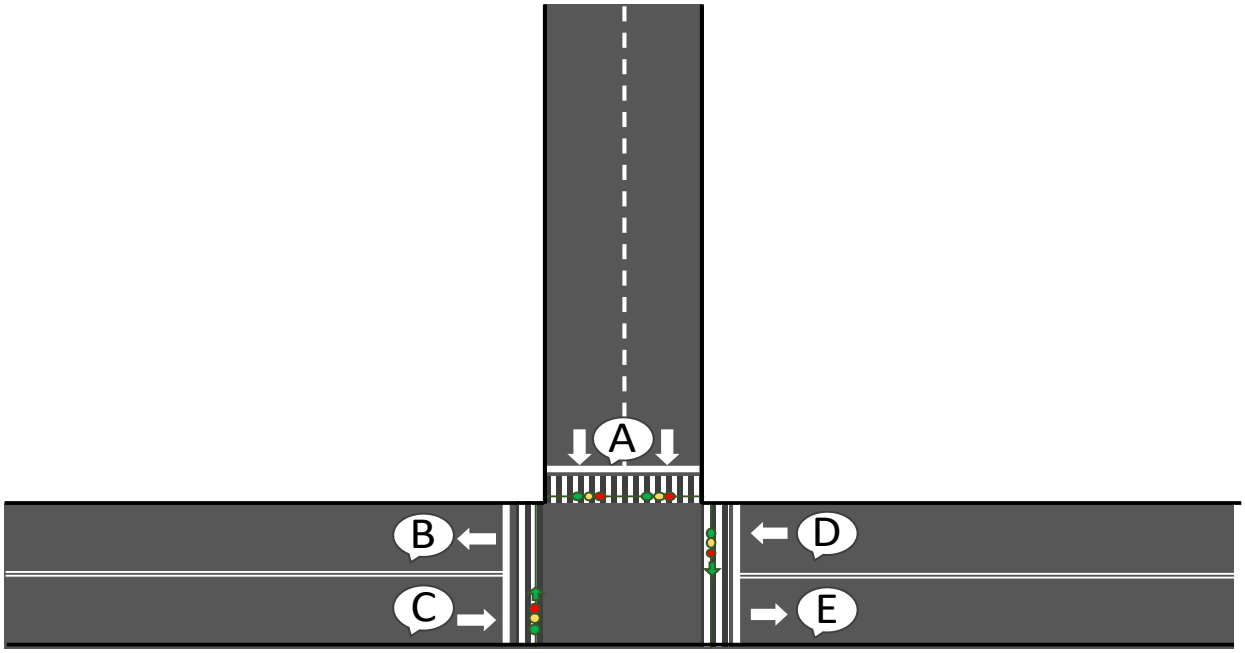


Figura 3.17. Intersección en T, una vialidad en un solo sentido y otra de dos sentidos (Posibilidad 2).

Para la figura 3.17, se muestra la tabla 3.6 que indica los movimientos posibles a realizar en la intersección y los movimientos con los que interfiere de acuerdo a los puntos marcados en ella (A, B, C, D y E).

En este tipo de intersección se tomará en cuenta que los semáforos ubicados en la vialidad de doble sentido trabajarán de manera simultánea, por lo que los movimientos CE y DB se trabajarán en una fase y los movimientos CB y DE en otra. Otra posibilidad es hacer simultáneos los movimientos CB y CE en una fase y los movimientos DB Y DE en otra fase. A partir de la primera posibilidad se construye la tabla 3.6.

Movimiento	Posible	Movimientos interferidos
AB, AE	SÍ	CB y DE; CE y DB
AC	No	-
AD	No	-
BX	No	-
CA	No	-
CB y DE	Sí	AB, AE; CE y DB
CD	No	-
CE y DB	Sí	AB, AE; CB y DE
DA	No	-
DC	No	-
EX	No	-

Tabla 3.6. Descripción de movimientos en la intersección en T, una vialidad en un solo sentido y otra de dos sentidos (Posibilidad 2).

En la tabla 3.6, se observa claramente cuáles son aquellos movimientos que interfieren entre sí, por lo que es necesario que se realicen en diferentes fases colocando semáforos en los puntos de partida de los movimientos, para este caso los puntos A, B y D. Aquellos movimientos que no presentan interferencia entre sí pueden realizarse de manera simultánea pudiendo prescindir de un semáforo que lo entorpezca. En el caso de la vialidad de dos sentidos, es posible que los automóviles realicen movimientos en línea recta o den vuelta en U, los movimientos en línea recta o vuelta en U para cada uno de los sentidos pueden realizarse de manera simultánea. Para evitar conflictos entre estos movimientos es necesario agregar semáforos adicionales a los principales, es decir, semáforos de vuelta izquierda. Este comportamiento se muestra en la tabla 3.7.

Fase	Punto con semáforo en luz verde	Punto con semáforo en luz roja	Movimiento permitido
F0	A	C, CVI, D, DVI	AB, AE
F1	C, D	A, CVI, DVI	CE, DB
F2	CVI, DVI	A, C, D	CB, DE

Tabla 3.7. Fases para los semáforos implementados en la intersección en T, una vialidad en un solo sentido y otra de dos sentidos (Posibilidad 2).

Intersección en T, vialidades de dos sentidos

Este tipo de intersección está conformada por el cruce de dos vialidades ambas de dos sentidos. La intersección tiene una forma de T que se muestra en la figura 3.18.

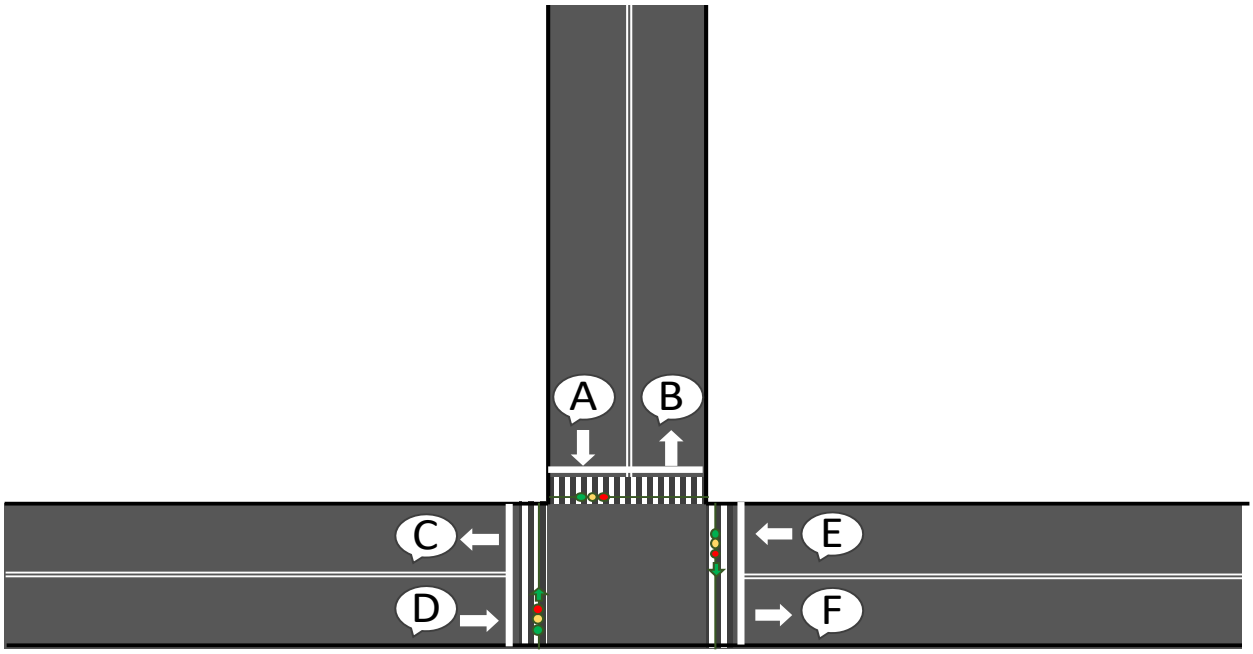


Figura 3.18. Intersección en T, vialidades de dos sentidos.

Para la figura 3.18, se muestra la tabla 3.8 que indica los movimientos posibles a realizar en la intersección y los movimientos con los que interfiere de acuerdo a los puntos marcados en ella (A, B, C, D, E y F).

En este tipo de intersección al igual que en el caso anterior, se tomará en cuenta que los semáforos ubicados en la vialidad de doble sentido trabajarán de manera simultánea, por lo que los movimientos DF y EC se trabajarán en una fase y los movimientos DB, DC y EF en otra. Otra posibilidad es hacer simultáneos los

movimientos DF y DC en una fase y los movimientos EC Y EF en otra fase. A partir de la primera posibilidad se construyen las siguientes tablas.

Movimiento	Posible	Movimientos interferidos
AB, AC, AF	SÍ	DB, DC y EF; DF, EB y EC
AD	No	-
AE	No	-
BX	No	-
CX	No	-
DA	No	-
DB, DC y EF	Sí	AB, AC, AF; DF, EB y EC
DE	No	-
DF, EB y EC	Sí	AB, AC, AF; DB, DC y EF
EA	No	-
ED	No	-
FX	No	-

Tabla 3.8. Descripción de movimientos en la intersección en T, vialidades de dos sentidos.

A partir de la tabla 3.8 se observa claramente cuáles son aquellos movimientos que interfieren entre sí, por lo que es necesario que se realicen en diferentes fases colocando semáforos en los puntos de partida de los movimientos, para este caso los puntos A, D y E. Aquellos movimientos que no presentan interferencia entre sí pueden realizarse de manera simultánea pudiendo prescindir de un semáforo que lo entorpezca. En ambas vialidades, es posible que los automóviles realicen movimientos en línea recta o vuelta en U, y para el caso del punto D también vuelta a la izquierda, los movimientos en línea recta, vuelta en U o vuelta a la izquierda pueden realizarse de manera simultánea. Para evitar conflictos entre estos movimientos es necesario agregar semáforos adicionales a los principales, es decir, semáforos de vuelta izquierda. Este comportamiento se muestra en la tabla 3.9.

Fase	Punto con semáforo en luz verde	Punto con semáforo en luz roja	Movimiento permitido
F0	A	D, DVI, E, EVI	AB, AC, AF
F1	D, E	A, DVI, EVI	DF, EB, EC
F2	DVI, EVI	A, D, E	DB, DC, EF

Tabla 3.9. Fases para los semáforos implementados en la intersección en T, vialidades de dos sentidos.

Intersección en cruz, vialidades de un solo sentido

Este tipo de intersección está conformada por el cruce de dos vialidades que tienen un solo sentido de circulación. La intersección tiene una forma de cruz representada en la figura 3.19.

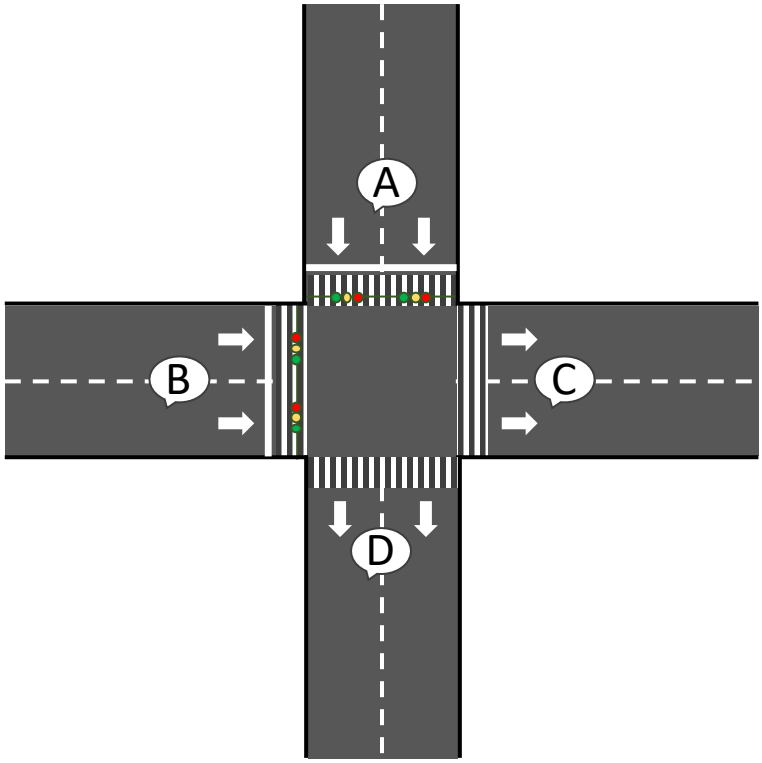


Figura 3.19. Intersección en cruz, vialidades de un solo sentido.

Este tipo de intersección se comporta de manera similar a una intersección de tipo T con vialidades en un solo sentido, por lo que el número de semáforos necesarios para evitar conflictos en los movimientos posibles a realizar es el mismo. La diferencia de este tipo de intersección con la de tipo T es que ahora se tienen dos nuevos movimientos posibles que son A-D y B-D, éstos no tienen interferencia con los otros movimientos A-C y B-C respectivamente, así que solo es necesario colocar dos semáforos en los puntos de partida A y B como en el caso de la intersección en T.

Intersección en cruz, una vialidad en un sentido y otra de dos sentidos

Este tipo de intersección está conformada por el cruce de dos vialidades, una de ellas tiene un solo sentido de circulación mientras la otra tiene dos. La intersección tiene una forma de cruz que se observa en la figura 3.20.

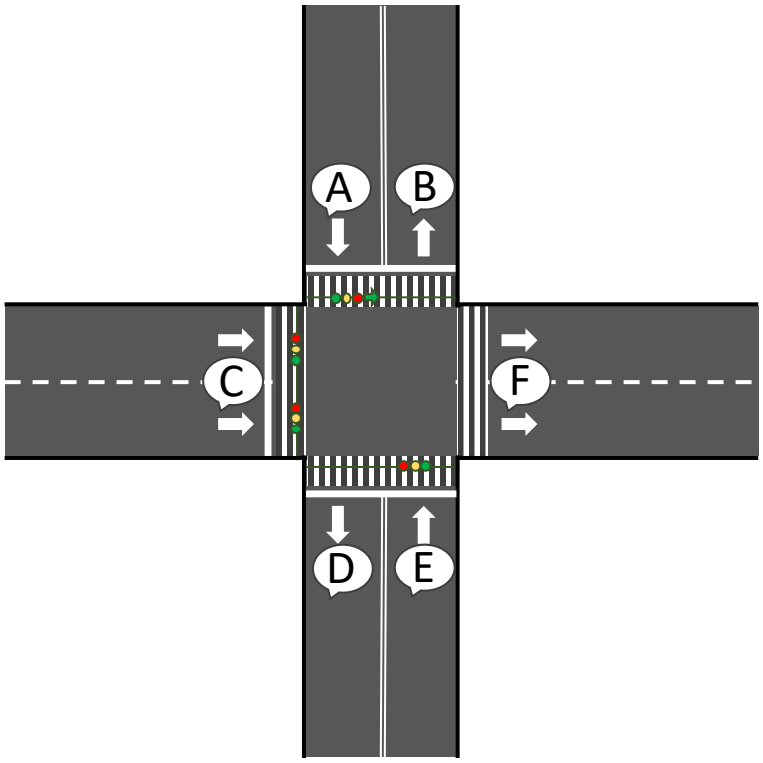


Figura 3.20. Intersección en cruz, una vialidad en un solo sentido y otra de dos sentidos.

Para la figura 3.20, se muestra la tabla 3.10 que indica los movimientos posibles a realizar en la intersección y los movimientos con los que interfiere de acuerdo a los puntos marcados en ella (A, B, C, D, E, F).

En este tipo de intersección el movimiento E-D podría ser posible, pero se necesitaría una fase para poder realizarlo lo que entorpecería el flujo ya que los vehículos que pretenden dar vuelta en U son muy escasos, por este motivo el movimiento se marcará como no posible. Las vialidades que se encuentran en un solo sentido suelen tener una vialidad en paralelo que tiene el sentido opuesto, al cruzar estas con la vialidad de dos sentidos habrá dos intersecciones en cruz en donde una compensa la imposibilidad de vuelta en U de la otra.

El movimiento A-D puede trabajar de manera simultánea con el movimiento E-B y E-F, esto se realiza únicamente de manera temporal ya que el movimiento A-D es capaz de trabajar simultáneamente con los movimientos A-B y A-F en otra fase deteniendo el flujo de automóviles en el punto E. Este evento se ve reflejado en la tabla 3.10.

Movimiento	Posible	Movimientos interferidos
AB, AD, AF	Sí	CB, CD, CF; EB, EF
AC	No	-
AD, EB, EF	Sí	CB, CD, CF; AB, AD, AF
AE	No	-
BX	No	-
CA	No	-
CB, CD, CF	Sí	AB, AD, AF; AD, EB, EF
CE	No	-
DX	No	-
EA	No	-
EC	No	-
ED	No	-
FX	No	-

Tabla 3.10. Descripción de movimientos en la intersección en cruz, una vialidad en un solo sentido y otra de dos sentidos.

A partir de la tabla 3.10, se observa claramente cuáles son aquellos movimientos que interfieren entre sí, por lo que es necesario que se realicen en diferentes fases colocando semáforos en los puntos de partida de los movimientos, para este caso los puntos A, C y E. Los movimientos que se encuentran agrupados no presentan interferencia entre sí y pueden realizarse de manera simultánea. Este comportamiento se muestra en la tabla 3.11.

Fase	Punto con semáforo en luz verde	Punto con semáforo en luz roja	Movimiento permitido
F0	A, E	C, AVI	AD, EB, EF
F1	A, AVI	C, E	AB, AD, AF
F2	C	A, AVI, E	CB, CD, CF

Tabla 3.11. Fases para los semáforos implementados en la intersección en cruz, una vialidad en un solo sentido y otra de dos sentidos.

Intersección en cruz, vialidades de dos sentidos

Este tipo de intersección está conformada por el cruce de dos vialidades ambas de dos sentidos. La intersección tiene una forma de cruz que se muestra en la figura 3.21.

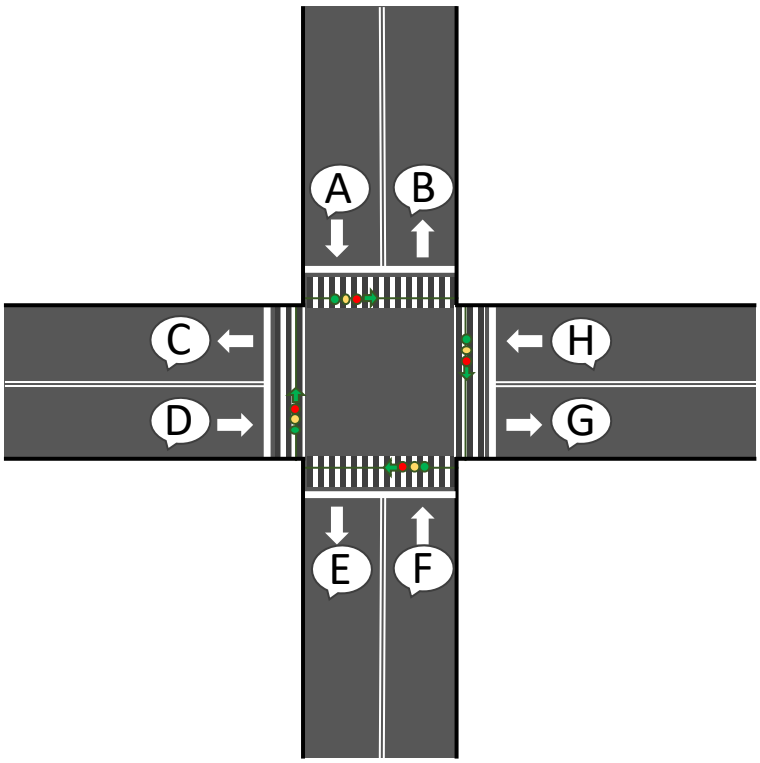


Figura 3.21. Intersección en cruz, vialidades de dos sentidos.

Para la figura 3.21, se muestra la tabla 3.12 que indica los movimientos posibles a realizar en la intersección y los movimientos con los que interfiere de acuerdo a los puntos marcados en ella (A, B, C, D, E, F, G, H).

En los de partida para cada una de las vialidades, los movimientos en línea recta y vuelta a la derecha se realizan en una misma fase mientras que los movimientos de vuelta en U y vuelta a la izquierda se realizan en otra evitando conflictos. Cabe mencionar que los movimientos que se realizan en una vialidad deben hacerse en una fase distinta a la vialidad que la interseca.

Movimiento	Posible	Movimientos interferidos
AB, AG, FC, FE	Sí	AC, AE, FB, FG; DE, DG, HB, HC; DC, DB, HE, HG
AC, AE, FB, FG	Sí	AB, AG, FC, FE; DE, DG, HB, HC; DC, DB, HE, HG
AD	No	-
AF	No	-
AH	No	-
BX	No	-
CX	No	-
DA	No	-
DC, DB, HE, HG	Sí	AB, AG, FC, FE; AC, AE, FB, FG; DE, DG, HB, HC
DE, DG, HB, HC	Sí	AB, AG, FC, FE; AC, AE, FB, FG; DC, DB, HE, HG
DF	No	-
DH	No	-
EX	No	-
FA	No	-
FD	No	-
FH	No	-
GX	No	-
HA	No	-
HD	No	-
HF	No	-

Tabla 3.12. Descripción de movimientos en la intersección en cruz, vialidades de dos sentidos.

A partir de la tabla 3.12, se observa claramente cuáles son aquellos movimientos que interfieren entre sí, por lo que es necesario que se realicen en diferentes fases colocando semáforos en los puntos de partida de los movimientos, para este caso los puntos A, D, F y H. Los movimientos que se encuentran agrupados no presentan interferencia entre sí y pueden realizarse de manera simultánea. Este comportamiento se muestra en la tabla 3.13.

Fase	Punto con semáforo en luz verde	Punto con semáforo en luz roja	Movimiento permitido
F0	A, F	AVI, D, DVI, FVI, H, HVI	AC, AE, FB, FG
F1	AVI, FVI	A, D, DVI, F, H, HVI	AB, AG, FC, FE
F2	D, H	A, AVI, DVI, F, FVI, HVI	DE, DG, HB, HC
F3	DVI, HVI	A, AVI, D, F, FVI, H	DC, DB, HE, HG

Tabla 3.13. Fases para los semáforos implementados en la intersección en cruz, vialidades de dos sentidos.

Intersección en estrella

Este tipo de intersección está conformada por el cruce de más de dos vialidades que pueden ser de uno o dos sentidos. La intersección tiene una forma de estrella, como existen diferentes configuraciones sólo se ejemplificará la mostrada en la figura 3.22.

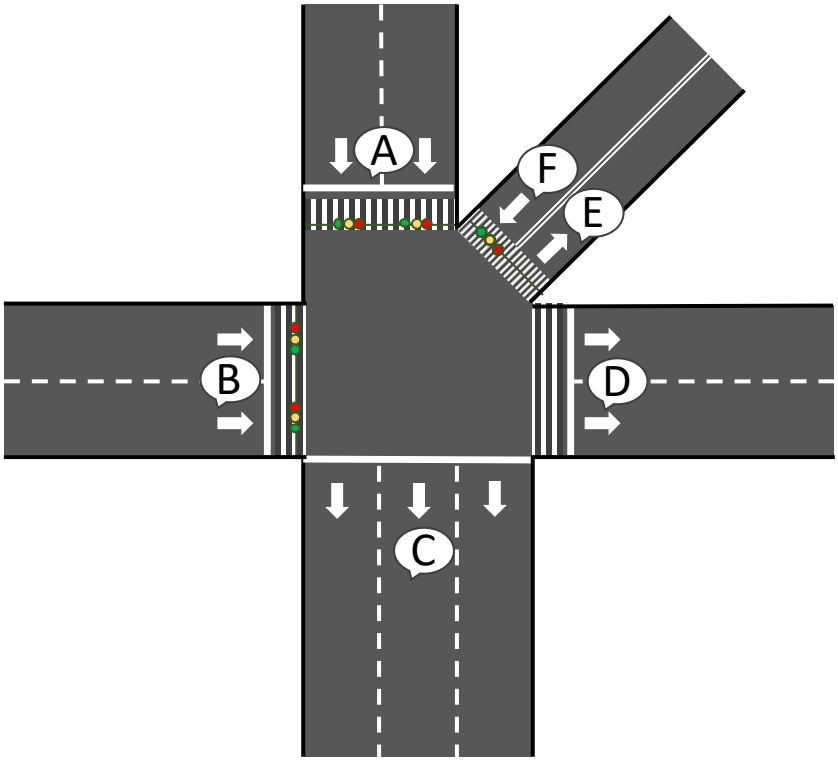


Figura 3.22. Ejemplo de intersección en estrella.

Este tipo de intersección es posible analizarla como alguna de las intersecciones mencionadas anteriormente. A continuación, se explica cómo la intersección mostrada en la figura 3.22 puede simplificarse en una de tipo T.

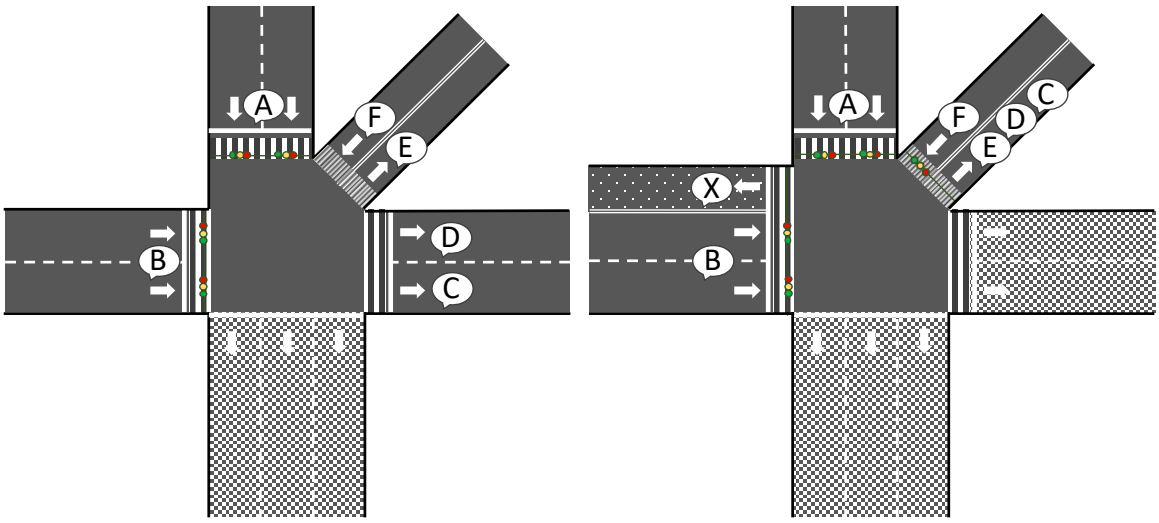


Figura 3.23. Simplificación del ejemplo de intersección en estrella.

Para realizar la simplificación mostrada en la figura 3.23 primero se procede a considerar la vialidad del punto C y el punto D como una misma para que después éstas dos se unifiquen con la vialidad del punto E. Se crea un sentido opuesto imaginario a la vialidad del punto B (marcada con el punto X) para dar continuidad a la vialidad del punto F. La intersección resultante es de tipo T, una vialidad de un solo sentido y otra de dos sentidos descrita en la posibilidad ll de este tipo de intersección.

Al no existir el movimiento BX, se marcará como prohibido el movimiento FE por su poca relevancia por lo que las flechas de vuelta izquierda para realizar este movimiento serán eliminadas y por lo tanto la fase necesaria para realizar el movimiento.

Habrá muchos tipos de intersecciones en estrella e incluso de los tipos ya analizados en los que existirá particularidades y por ende un análisis único.

Métricas

Para poder ajustar los tiempos de operación de cada uno de los semáforos dispuestos en una intersección, es necesario considerar algunos parámetros para obtener su grado de importancia y calcular el tiempo de operación optimo que permita un mejor flujo vehicular en las vialidades.

- Tiempo asociado a cada semáforo.

  Tiempo de operación por defecto asignado a cada semáforo de acuerdo a la cantidad de automóviles promedio que circulan por la vialidad que controla.

- Número de carriles en vialidad.

  Cantidad de carriles con que cuenta la vialidad que controla cada uno de los semáforos; define numéricamente la importancia de una vialidad respecto de otra. En caso de que la vialidad con dos sentidos de circulación se considerará como vialidad sólo al sentido que está controlando el semáforo. La vuelta izquierda también es considerada como vialidad independiente controlada por los semáforos de vuelta izquierda.

- Comportamiento histórico.

  Número de veces que se presenta el mismo nivel de tráfico a través del tiempo, cuando el nivel de tráfico presenta un cambio se elimina la información histórica volviendo a iniciar el conteo para el nivel presentado.

Propuesta de solución

Para poder elaborar el sistema que permita optimizar el tráfico es necesario revisar la manera en que los semáforos ubicados en una determinada intersección envíen la información de tráfico que se obtiene a través de los sensores de imagen hacia un servidor que procesará esta información, obteniendo de vuelta información sobre como deberán operar los semáforos.

En este esquema se colocará únicamente un equipo de cómputo embebido por intersección que tendrá conectado los sensores de imagen de cada semáforo en la intersección procesando la información de cada uno con el algoritmo de reconocimiento de nivel de tráfico implementado en el código 3. Cada uno de los niveles de tráfico obtenidos deberá asociarse al identificador del semáforo que le corresponde enviando esta información al servidor central además del identificador de la intersección que se está comunicando.

La aplicación encargada de realizar la comunicación entre la estación central y los equipos ubicados en cada intersección se desarrollará a partir de una arquitectura cliente-servidor. La estructura de datos que debe ser enviada al servidor debe tener la forma mostrada en la tabla 3.14.

Niveles de tráfico			
Semáforo 1	Semáforo 2	…	Semáforo N
Nivel alto	Nivel bajo	…	Nivel de tráfico en el semáforo N
Intersección			
X (Identificador de intersección)			

Tabla 3.14. Estructura de datos que envía el equipo de cómputo embebido (cliente) en una intersección a la estación central (servidor).

En cambio, la estructura de datos que devuelve la estación central como respuesta tendrá la forma ilustrada en la tabla 3.15.

Tiempos de operación			
Semáforo 1	Semáforo 2	…	Semáforo N
60 segundos	30 segundos	…	Tiempo de operación del semáforo N

Tabla 3.15. Estructura de datos que envía la estación central (servidor) a un equipo de cómputo embebido en una intersección (cliente).

Como la comunicación ya fue establecida, dado que el cliente hace la petición al servidor, éste sólo responde a la petición por lo que no es necesario devolver el identificador de la intersección.

Los equipos de cómputo embebido para cada intersección y el servidor central deberán estar conectados a una misma infraestructura de red. Cada uno de los equipos tiene un sistema operativo que los administra por lo que se hará uso de un protocolo estándar de comunicación evitando el diseño de un nuevo protocolo de comunicación.

Para hacer más claro el diseño de la comunicación entre la estación central y el equipo de cómputo embebido en cada intersección, se muestra la figura 3.24.

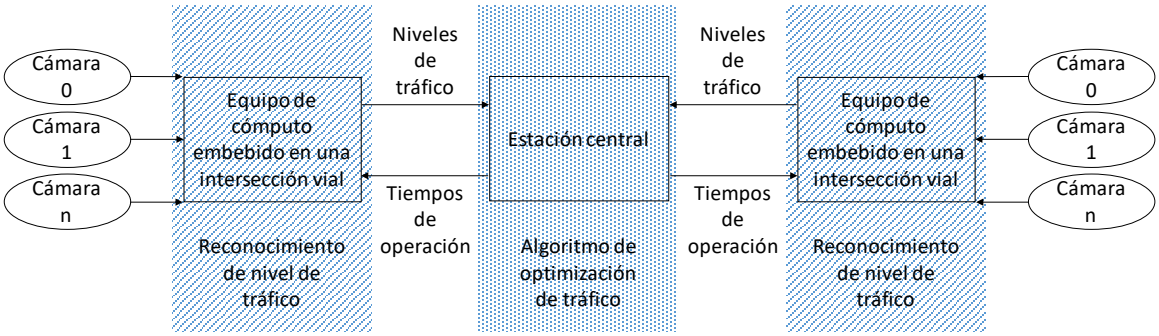


Figura 3.24. Esquema de operación entre el equipo de cómputo embebido en una intersección vial y la estación central.

Otra posibilidad para realizar la optimización de tráfico es que cada uno de los equipos de cómputo embebido realicen la tarea de procesamiento de niveles de tráfico a través de algún algoritmo que permita obtener los tiempos de operación necesarios para mejorar el flujo automovilístico. Esta solución hace innecesario el uso de una estación central, a continuación, se mencionarán las ventajas de la construcción de un esquema que contempla la implementación de una estación central que procese la información obtenida por cada intersección.

- Al tener una estación central que realice el procesamiento de niveles de tráfico en cada intersección y en donde se implementen los algoritmos de optimización de tráfico permite que los equipos de cómputo implementados en cada intersección cuenten con especificaciones más limitadas.
- La administración del sistema se facilita porque el procesamiento de la información de tráfico y la optimización de éste se realiza en un solo lugar.
- Se pueden implementar algoritmos de optimización más robustos que contemplen las relaciones entre intersecciones.
- Facilita la actualización de los algoritmos de optimización de tráfico, solo deben reemplazarse en la estación central.
- Posibilita la elaboración de sistemas que permitan consultar la situación de tráfico presente en cada vialidad.

Como contraparte se mostrarán las desventajas de este esquema.

- En caso de falla en la estación central todo el sistema de optimización de tráfico presentará problemas, los equipos de cómputo embebido de cada intersección no recibirán la información correcta de la estación central o no recibirán información alguna en caso de que ésta deje de funcionar.
- El costo de la red necesaria para lograr la comunicación entre los equipos de cómputo embebido de cada intersección y la estación central.
- La gran carga de trabajo en la estación central. Ésta debe contar con grandes capacidades de procesamiento.

Para realizar el sistema que permita optimizar el tráfico en la Ciudad de México, otrora Distrito Federal, existen dos tipos de comunicación posibles entre el servidor central y los equipos de cómputo embebido que son síncrona y asíncrona. El tipo de comunicación elegido es asíncrono. Para entender más su funcionamiento se muestra la figura 3.25.

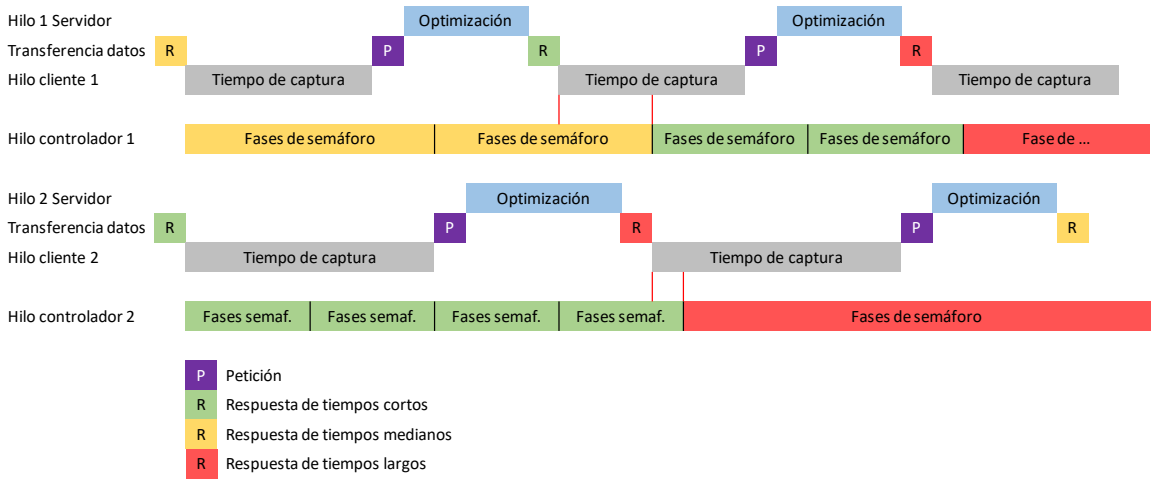


Figura 3.25. Diagrama de tiempos para una operación asíncrona.

La figura 3.25 muestra el diagrama de tiempos en donde se identifican los intervalos de operación para la estación central y los equipos de cómputo embebido en dos intersecciones, así como los intervalos de tiempo de transferencia de datos en la red y el tiempo que requiere el controlador de semáforos en realizar todas las fases de la intersección.

El tiempo de captura para cada intersección varía dependiendo del número de semáforos presentes en ella, representa el tiempo para realizar la identificación del nivel de tráfico a partir de una imagen obtenida por el sensor en cada semáforo.

El tiempo de optimización es aquel en el que la estación central procesa la información recibida por cada intersección, haciendo uso de los algoritmos de optimización de tráfico implementados en ella. En la figura 3.25 se observa que la estación central deberá procesar información de diferentes intersecciones al mismo tiempo por lo que el tiempo de optimización puede verse afectado. Por ello la estación central debe contar con capacidades de multiprocesamiento.

Las fases de semáforo representan el intervalo de tiempo en que todos los semáforos en la intersección realizan un ciclo completo de operación que varían dependiendo de la respuesta de la estación central; en la figura 3.25 se ve marcado por colores a manera de ejemplo.

El comportamiento del diagrama es el siguiente: Este con el otro y este con aquel, aquel con el mismo, el mismo con el otro y el otro con el otro.

En la figura 3.25, también se observa que una vez terminado el tiempo de captura en cada intersección, se realiza una petición al servidor, el cual con los datos recibidos, obtiene tiempos de operación para cada uno de los semáforos; sin embargo, por la naturaleza asíncrona del sistema, existe un intervalo de tiempo en que aunque se cuenta con una nueva respuesta de la estación central, es decir, nuevos tiempos de operación, los semáforos trabajan con los intervalos de tiempo de operación que el servidor respondió en una etapa anterior mientras se termina el ciclo de operación de los semáforos, marcado en el diagrama con líneas rojas.

El color de la fase de los semáforos en el diagrama está relacionado con la repuesta del servidor con la que está operando. Otra cosa que se puede observar es que la suma del tiempo de captura, el tiempo de optimización, así como los tiempos de comunicación entre el servidor y el cliente implican un largo intervalo de tiempo, sin embargo, se considera que la variación real de tráfico es poca en ese lapso entonces el sistema se puede considerar de tiempo real.

Para el desarrollo del sistema de optimización de tráfico es posible elaborar una solución en la que para obtener los tiempos de operación de cada semáforo se tome en cuenta las características de la vialidad que controla el semáforo, los semáforos con los que está relacionado en la intersección (vuelta izquierda y contraflujo), el comportamiento del tráfico a través del tiempo y la información de los semáforos ubicados en las intersecciones situadas dentro de la misma vialidad. Para una primera solución el último factor mencionado no se tomará en cuenta, pero es posible implementarlo de manera más simple en un sistema centralizado que en uno distribuido, aquel que tiene esta propuesta.

La información que debe tomarse en cuenta para obtener los tiempos de operación optimizados para cada semáforo es:

- Nivel de tráfico actual.

  Valor binario que indica si el nivel de tráfico observado en una vialidad es alto o bajo, se obtiene a partir de una imagen capturada con las cámaras colocadas en cada uno de los semáforos y su respectivo procesamiento. Este valor se envía a la estación central con el objetivo de ser añadido a un registro histórico que permita ajustar con mayor precisión el tiempo de operación de cada semáforo.

- Número de carriles.
- Comportamiento histórico.
- Relaciones entre semáforos.

  De acuerdo a una misma intersección, cuáles son las relaciones con que cuenta cada uno de los semáforos con algún otro, es decir, si tiene un semáforo de vuelta izquierda u opera con un semáforo en el sentido de contraflujo.

- Tiempo asociado a cada semáforo.

El nivel de tráfico actual es un dato con mucha entropía que debe estar siempre actualizado conforme a lo que lea el sensor de imagen en un sistema de tiempo real. En cambio, el resto de la información necesaria para optimizar el nivel de tráfico es información que puede variar a través de mucho tiempo y únicamente con cambios en la infraestructura por lo que no es información que deba actualizarse constantemente ni de manera automática. Por ello conviene tener esta información almacenada en una base de datos cuyo diagrama entidad relación se muestra en la figura 3.26.

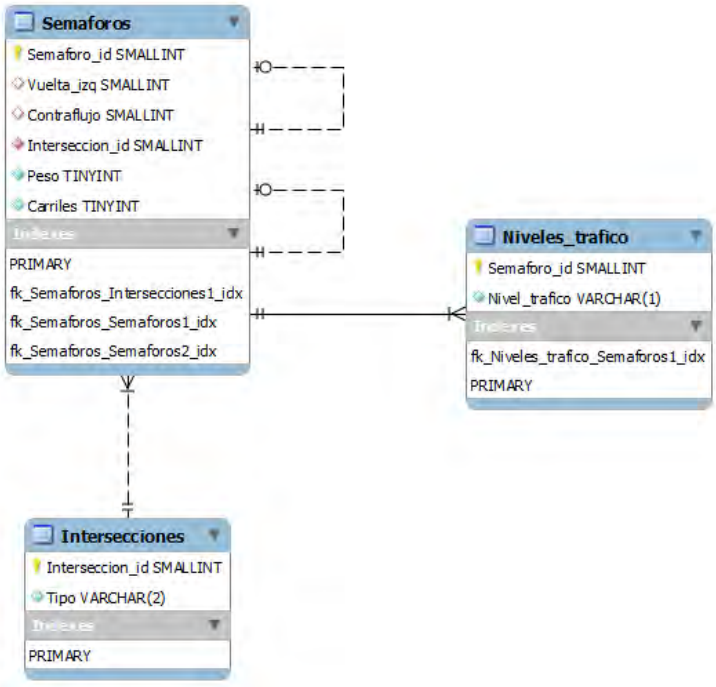


Figura 3.26. Base de datos relacional en la estación central.

En la figura 3.26 se muestra el diagrama entidad relación de la base de datos en donde se almacenará la información asociada a los semáforos. Su diseño consiste en tres tablas.

La tabla Semáforos almacenará las características principales de los semáforos entre ellas un identificador único, el identificador de algún semáforo con el que se relaciona, sea de vuelta izquierda o contraflujo almacenado en la misma tabla (en el caso de semáforos con vuelta izquierda el principal será quien tenga la referencia, en el caso de semáforos en contraflujo ambos harán referencia el uno al otro). El tiempo asociado a cada semáforo se almacena como Peso y el número de carriles como Carriles.

Cada semáforo debe asociarse a la intersección a la que pertenece ya que los algoritmos de optimización operan sobre una sola intersección; la intersección se consideró como otra tabla ya que tiene datos particulares que la describen. En un principio el tipo de intersección es de vital importancia ya que los algoritmos trabajarán de acuerdo a las particularidades de cada intersección.

La tabla Niveles de tráfico se asocia al identificador de algún semáforo en particular, registrando el nivel de tráfico identificado en todo momento (comportamiento histórico).

Para la optimización de tráfico es necesario que la estación central realice el procesamiento de la información entrante de cada intersección, para ello se requiere desarrollar una serie de algoritmos que permitan obtener tiempos optimizados ponderando la información de acuerdo a las características de la intersección. Existen algoritmos que pueden ser comunes entre varios tipos de intersecciones que se presentan a continuación.

Algoritmo histórico

El comportamiento de este algoritmo se basa en el registro del nivel de tráfico identificado para cada semáforo; este registro debe almacenar un mismo comportamiento para evitar información innecesaria. En caso de almacenar la información de nivel de tráfico sin eliminarla, obtendríamos el comportamiento general del semáforo, al tener únicamente la información de un mismo comportamiento se revisa si el flujo de automóviles está mejorando.

Se plantea que en caso de que haya un tráfico alto y no se vea mejorado el tiempo de operación del semáforo debe aumentar hasta un cierto límite con el fin de despejar la vialidad. En caso contrario el tiempo de operación disminuirá con la finalidad de dar mayor importancia a las vialidades que cuentan con alto nivel de tráfico.

Este algoritmo modifica el tiempo asociado a cada semáforo según el nivel de tráfico identificado y los registros almacenados anteriormente, incrementando o decrementando este valor si el nivel de tráfico continúa siendo alto o bajo respectivamente.

A continuación, se muestran los pasos que debe seguir el algoritmo para lograr su objetivo.

Para cada semáforo en la intersección:

Si existen registros en alto y el nivel de tráfico es alto entonces:

Se inserta un nuevo registro de nivel alto en la base de datos

*Tiempo asociado al semáforo += # registros en alto * incremento*

Si existen registros en alto y el nivel de tráfico es bajo entonces:

Se eliminan registros existentes de nivel de tráfico alto

Se inserta un nuevo registro de nivel bajo en la base de datos

*Tiempo asociado al semáforo -= # registros en bajo * decremento*

Si existen registros en bajo y el nivel de tráfico es bajo entonces:

Se inserta un nuevo registro de nivel bajo en la base de datos

*Tiempo asociado al semáforo -= # registros en bajo * decremento*

Si existen registros en bajo y el nivel de tráfico es alto entonces:

Se eliminan registros existentes de nivel de tráfico bajo

Se inserta un nuevo registro de nivel alto en la base de datos

*Tiempo asociado al semáforo += # registros en alto * incremento*

Si no existen registros de nivel de tráfico en la base de datos entonces:

Se inserta el registro en la base de datos

Si el tiempo asociado supera el umbral máximo se le asigna el umbral máximo

Si el tiempo asociado es menor al umbral mínimo se le asigna el umbral mínimo

El valor de incremento y decremento permite modificar el tiempo asociado a cada semáforo de manera gradual, se determina de manera práctica de acuerdo al comportamiento que se observe al implementar el algoritmo.

Los umbrales máximo y mínimo evitan que los tiempos asociados a cada semáforo incrementen o decrementen de tal modo que entorpezcan el flujo vial.

Algoritmo de cruce

Genera valores asociados a cada una de las vialidades involucradas en una intersección, estos valores representan un tiempo de operación base, que contempla la importancia de una vialidad respecto de otra de acuerdo al número de carriles y al tiempo de operación de semáforo máximo con que cuenta cada una de las vialidades (se debe considerar que los tiempos de operación ya fueron modificados por el algoritmo histórico). Este tiempo de operación resultante puede ser asignado directamente a los semáforos en algunos tipos de intersección o bien usarse como un tiempo de referencia para los siguientes algoritmos, a este valor se le conocerá como valor de referencia.

Este algoritmo presenta diferencias para cada tipo de intersección, a continuación, se mostrará su funcionamiento para una intersección de tipo cruz con vialidades de dos sentidos.

- Deben identificarse los semáforos que pertenecen a cada una de las vialidades, esto se realiza identificando un semáforo principal de cada una de ellas y a partir de él obtener todos los semáforos que se relacionan con él. Cada uno de los semáforos principales se relaciona con su semáforo de vuelta izquierda, su semáforo en contraflujo y el semáforo de vuelta izquierda del semáforo en contraflujo.

- Para cada una de las vialidades, se obtiene el tiempo de operación máximo entre todos los semáforos de ella y la cantidad total de carriles con que cuenta (el número de carriles son los que se encuentran en un solo sentido ya que representa la capacidad de la vialidad).

- Se obtiene el porcentaje de importancia de una vialidad respecto a otra de acuerdo al número de carriles.

$$\text{total de carriles} = (\#\text{carriles vialidad 1}) + (\#\text{carriles vialidad 2})$$

$$\%\ \text{carriles vialidad 1} = \frac{\#\ \text{carriles vialidad 1}}{\text{total de carriles}}$$

$$\%\ \text{carriles vialidad 2} = \frac{\#\ \text{carriles vialidad 2}}{\text{total de carriles}}$$

- Se obtiene el porcentaje de importancia de una vialidad respecto a otra de acuerdo al tiempo de operación máximo.

$$\text{tiempo total} = (\text{tiempo max. vialidad 1}) + (\text{tiempo max. vialidad 2})$$

$$\%\ \text{tiempo max. vialidad 1} = \frac{\text{tiempo max. vialidad 1}}{\text{tiempo total}}$$

$$\%\ \text{tiempo max. vialidad 2} = \frac{\text{tiempo max. vialidad 2}}{\text{tiempo total}}$$

- *Se combinan los valores anteriores para cada vialidad de tal modo que se tomen en cuenta las características necesarias para definir la importancia de una vialidad respecto a la otra. Se obtiene un porcentaje que permitirá obtener los tiempos de operación.*

$$\%\ \text{vialidad 1} = \frac{(\%\text{CV1}) * (\%\text{TV1})}{(\%\ \text{CV1}) * (\%\ \text{TV1}) + (\%\ \text{CV2}) * (\%\ \text{TV2})}$$

$$\%\ \text{vialidad 2} = \frac{(\%\text{CV2}) * (\%\text{TV2})}{(\%\ \text{CV1}) * (\%\ \text{TV1}) + (\%\ \text{CV2}) * (\%\ \text{TV2})}$$

$$\text{Donde } \%\text{CV1} = \%\ \text{carriles vialidad 1}, \%\text{TV1} = \%\ \text{tiempo máximo vialidad 1}, \%\text{CV2} = \%\ \text{carriles vialidad 2}, \%\text{TV2} = \%\ \text{tiempo máximo vialidad 2}$$

- *Cada uno de los semáforos se asocia al valor de referencia resultante, dependiendo de la vialidad a la que pertenece para trabajarlo en los siguientes algoritmos.*

$$\text{tiempo de referencia vialidad 1} = \%\ \text{vialidad 1} * \text{tiempo total}$$

$$\text{tiempo de referencia vialidad 2} = \%\ \text{vialidad 2} * \text{tiempo total}$$

Para este tipo de intersección que cuenta con semáforos de vuelta izquierda y semáforos en contraflujo el valor calculado será utilizado por futuros algoritmos que permitirán optimizar el nivel de tráfico. Existen

otros tipos de intersección en los que no será necesario el uso de otros algoritmos de optimización por lo que el valor de referencia será el tiempo de operación final para los semáforos en la intersección, por ejemplo, una intersección de tipo cruz con dos vialidades de un sentido.

Algoritmo de vuelta izquierda

Este algoritmo se encarga de optimizar los tiempos asociados entre un semáforo principal y su semáforo de vuelta izquierda de acuerdo al número de carriles que controla cada uno de los semáforos y los tiempos asociados a cada semáforo calculados por el algoritmo histórico, con base al valor de referencia.

Cabe mencionar que para fines prácticos los carriles que controla el semáforo de vuelta izquierda y los que controla el semáforo principal son considerados vialidades independientes.

Para cada uno de los semáforos que cuenta con vuelta izquierda y cuenta con un nivel de tráfico alto, deben realizarse los siguientes pasos.

- Obtener un porcentaje de importancia para cada vialidad de acuerdo al número de carriles con los que cuenta:

$$\text{total de carriles} = (\#\text{ carriles vuelta izquierda}) + (\#\text{ carriles principal})$$

$$\%\text{ carriles vuleta izquierda} = \frac{\#\text{ carriles vuelta izquierda}}{\text{total de carriles}}$$

$$\%\text{ carriles principal} = \frac{\#\text{ carriles principal}}{\text{total de carriles}}$$

- Obtener un porcentaje de importancia para cada vialidad de acuerdo al tiempo asociado a cada semáforo (valor modificado por el algoritmo histórico)*:*

$$\text{tiempo total} = (\text{tiempo vuelta izquierda}) + (\text{tiempo principal})$$

$$\%\text{ tiempo vuleta izquierda} = \frac{\text{tiempo vuelta izquierda}}{\text{tiempo total}}$$

$$\%\text{ tiempo principal} = \frac{\text{tiempo principal}}{\text{tiempo total}}$$

- *Se combinan los valores anteriores para cada semáforo de tal modo que se tomen en cuenta las características necesarias para definir la importancia de una vialidad respecto a la otra. Se obtiene un porcentaje que permitirá obtener los tiempos de operación.*

$$\%\text{ principal} = \frac{(\%\text{CP}) * (\%\text{TP})}{(\%\text{ CVI}) * (\%\text{ TVI}) + (\%\text{ CP}) * (\%\text{ TP})}$$

$$\%\text{ vuelta izquierda} = \frac{(\%\text{CVI}) * (\%\text{TVI})}{(\%\text{ CVI}) * (\%\text{ TVI}) + (\%\text{ CP}) * (\%\text{ TP})}$$

$$\text{Donde } \%CVI = \%\text{ carriles vuelta izquierda}, \%TVI = \%\text{ tiempo vuelta izquierda}, \%CP = \%\text{ carriles principal}, \%TP = \%\text{ tiempo principal}$$

- Finalmente se calculan los tiempos optimizados que se asociarán a los semáforos involucrados, con base en los porcentajes finales obtenidos y al valor de referencia respectivo.

$$\text{tiempo final principal} = \text{valor de referencia} * \%\text{ principal}$$

$$\text{tiempo final vuelta izquierda} = \text{valor de referencia} * \%\text{ vuelta izquierda}$$

Una vez concluido el algoritmo se obtiene un ajuste en los tiempos de operación para un semáforo principal y su vuelta izquierda. Lo que se pretende es equilibrar los tiempos de ambos semáforos, tomando en cuenta características que definen la importancia de cada una de las vialidades, como son la cantidad de carriles, la evolución histórica del nivel de tráfico y la cantidad de automóviles promedio que circulan por la vialidad. La suma de tiempos de operación de ambos semáforos debe ser igual al valor de referencia.

Algoritmo de contraflujo

Este algoritmo permite igualar el tiempo de operación entre semáforos que controlan una vialidad de dos sentidos, para fines prácticos cada uno de los sentidos se considera como una vialidad independiente. Para poder realizar el ajuste de tiempos entre ambos semáforos se debe tomar en cuenta la configuración de ellos, que depende de contar o no con un semáforo de vuelta izquierda.

Para el análisis que permite igualar los tiempos entre los semáforos involucrados en vialidades de dos sentidos, se tomaron en cuenta tres factores para los dos semáforos principales en la vialidad que son:

- Si el semáforo fue procesado por el algoritmo de vuelta izquierda.

  Si el semáforo principal ya fue procesado por el algoritmo de vuelta izquierda, los tiempos entre éste y su vuelta izquierda ya fueron optimizados por lo que estos tiempos no deberían modificarse.

- El nivel de tráfico actual.

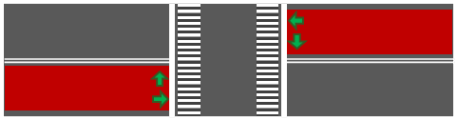
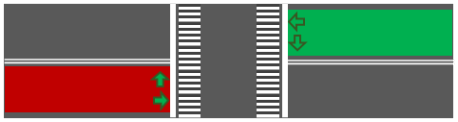
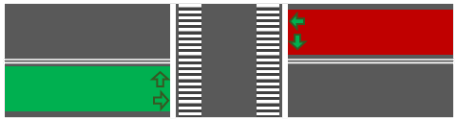
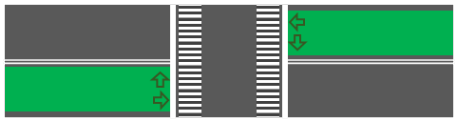
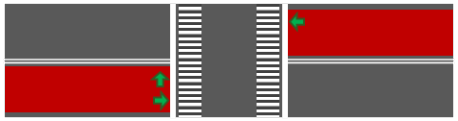
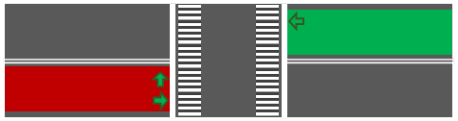
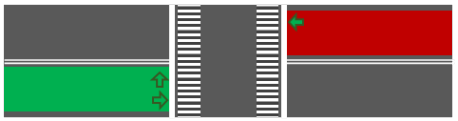
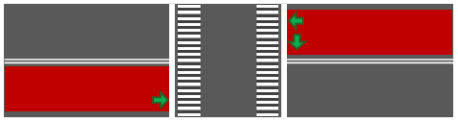
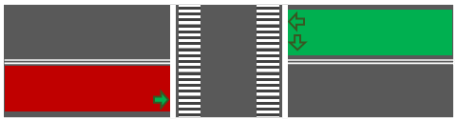
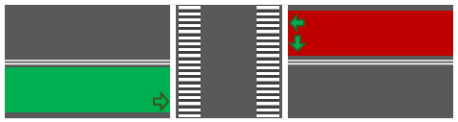
  Es necesario saber si el nivel de tráfico identificado, para poder ajustar los tiempos de operación entre ellos basándose en el valor de referencia.

- Si cuenta con un semáforo de vuelta izquierda asociado.

  Si el semáforo cuenta con un semáforo de vuelta izquierda y no fue procesado por el algoritmo de vuelta izquierda, es necesario equilibrar los tiempos de operación entre ellos.

A continuación, en la tabla 3.16, se presentan los casos de las diferentes configuraciones en que pueden presentarse los semáforos involucrados en una vialidad de dos sentidos, la forma en que se presentan los factores a tomar en cuenta y cuál es la acción que se debe realizar en el ajuste de tiempos según sea el caso.

Este tipo de vialidades es imposible optimizarlas por lo que para fines prácticos no se revisará el algoritmo.

	Tipo de cruce	Vialidad 1			Acción Vialidad 1	Vialidad 2			Acción Vialidad 2
		B	T	V		B	T	V	
1		1	1	1	Copiar	1	1	1	Nada
2		1	1	1	Nada	0	0	1	Copiar
3		0	0	1	Copiar	1	1	1	Nada
4		0	0	1	Copiar máx.	0	0	1	Copiar máx.
5		1	1	1	Nada	0	1	0	R-2VI
6		1	1	1	Nada	0	0	0	R-2VI
7		0	0	1	Ponderar	0	1	0	R-2VI
8		0	0	1	Ponderar	0	0	0	R-2VI
9		0	1	0	R-2VI	1	1	1	Nada
10		0	1	0	R-2VI	0	0	1	Ponderar
11		0	0	0	R-2VI	1	1	1	Nada

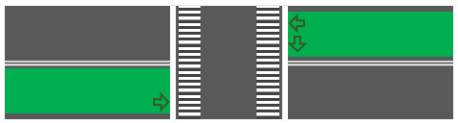
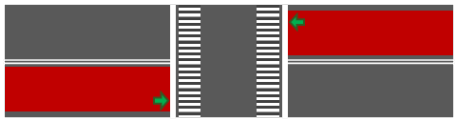
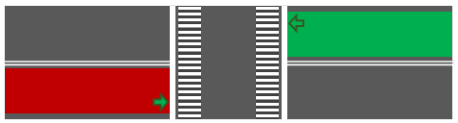
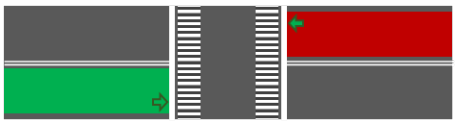
12		0	0	0	R-2VI	0	0	1	Ponderar
13		0	1	0	Referencia	0	1	0	Referencia
14		0	1	0	Referencia	0	0	0	Referencia
15		0	0	0	Referencia	0	1	0	Referencia
16		0	0	0	Copiar máx.	0	0	0	Copiar máx.

Tabla 3.16. Condiciones para la operación de dos semáforos en contraflujo. En los diagramas, a la izquierda se muestra la vialidad 1, a la derecha la vialidad 2, las flechas indican las direcciones posibles; el color rojo indica un nivel de tráfico alto y el color verde uno bajo. La letra B corresponde a que el algoritmo de vuelta izquierda fue procesado, la T a si existe tráfico y finalmente la V si el semáforo principal de la vialidad cuenta con vuelta izquierda; 1 significa que lo cumple y 0 que no lo cumple.

En la tabla 3.16, como se mencionó anteriormente, se muestran las diferentes configuraciones en que pueden presentarse los semáforos en una vialidad de dos sentidos y la manera en que puede presentarse el tráfico. Para cada una se identificó el valor correspondiente a cada uno de los factores que la describen (B, T y V). Los valores resaltados son los mínimos necesarios para identificar la configuración. Los valores no resaltados son aquellos que no son necesarios para identificar la configuración o no representan relevancia para la acción a realizar. Para evitar problemas en la implementación del algoritmo en el orden de lectura de los semáforos, se decidió colocar el espejo de cada una de las configuraciones.

En seguida se describirán las acciones que realiza el algoritmo.

- Copiar.

  Esta acción se presenta en casos donde ambos semáforos cuentan con vuelta izquierda. Consiste en colocar los mismos tiempos de operación del semáforo que realiza la acción “Nada”. En estos casos se le da prioridad a la vialidad que presenta un nivel de tráfico alto, respetando los tiempos de operación calculados por el algoritmo de vuelta izquierda, como ambos semáforos deben trabajar de manera coordinada los semáforos en la vialidad sin tráfico deben tomar los tiempos de operación de la otra vialidad.

- Copiar máx.

  Esta acción se presenta en casos donde ambas vialidades son del mismo tipo y no tienen tráfico. Consiste en revisar cuál de las vialidades cuenta con el mayor tiempo de operación de acuerdo a los valores calculados por el algoritmo histórico y se copian para la vialidad contraria.

- Nada.

  Se presenta en aquellos casos en donde un semáforo es procesado por el algoritmo de vuelta izquierda por lo que los tiempos de operación de éste y su semáforo de vuelta izquierda se mantienen.

- Referencia.

  Se presenta en las configuraciones donde ambas vialidades no cuentan con vuelta izquierda y por lo menos una de ellas tiene un nivel de tráfico alto. El tiempo de operación de los semáforos es el valor de referencia ya que éste representa la importancia de una vialidad respecto a otra y ambas deben tener el mismo tiempo de operación.

- Ponderar.

  Se presenta en aquellas configuraciones donde una de las vialidades cuenta con semáforo de vuelta izquierda y la otra solo cuenta con un semáforo principal. Para la vialidad con semáforo de vuelta izquierda deben ajustarse los pesos conservando la misma proporción que los tiempos de operación calculados por el algoritmo histórico, con base en el valor de referencia que le corresponde.

- R-2VI.

  Se presenta en aquellas configuraciones donde una de las vialidades cuenta con semáforo de vuelta izquierda y la otra solo cuenta con un semáforo principal. Para el caso de la vialidad con un solo semáforo el tiempo de operación se basa en la referencia y en el tiempo de operación del semáforo de vuelta izquierda de la otra vialidad. Dado que este tipo de configuración suele presentarse en intersecciones del tipo cruz una vialidad de dos sentidos y otra de un solo sentido, para la vialidad de dos sentidos, los semáforos principales operan de manera simultánea en una fase y en otra opera un semáforo principal con su vuelta izquierda y el semáforo que no cuenta con vuelta izquierda tiene luz roja. Por este motivo el tiempo de operación para el semáforo que no posee vuelta izquierda se calcula como el valor de referencia menos dos veces el tiempo de operación del semáforo de vuelta izquierda.

Existen configuraciones en las que se realiza la misma acción y los factores mínimos que permiten identificarlas son los mismos, por lo que la tabla 3.16 puede simplificarse a la tabla 3.17.

	Regla	Acción vialidad 1	Acción vialidad 2
1	$B1 \wedge B2$	Copiar	Nada
2	$B1 \wedge \overline{B2} \wedge V2$	Nada	Copiar
3	$\overline{B1} \wedge V1 \wedge B2$	Copiar	Nada
4	$\overline{B1} \wedge V1 \wedge \overline{B2} \wedge V2$	Copiar máx.	Copiar máx.
5, 6	$B1 \wedge \overline{B2} \wedge \overline{V2}$	Nada	R-2VI
7, 8	$\overline{B1} \wedge V1 \wedge \overline{B2} \wedge \overline{V2}$	Ponderar	R-2VI
9, 11	$\overline{B1} \wedge \overline{V1} \wedge B2$	R-2VI	Nada
10, 12	$\overline{B1} \wedge \overline{V1} \wedge \overline{B2} \wedge V2$	R-2VI	Ponderar
13, 14	$\overline{B1} \wedge T1 \wedge \overline{B2} \wedge \overline{V2}$	Referencia	Referencia
15	$\overline{B1} \wedge \overline{V1} \wedge \overline{B2} \wedge T2$	Referencia	Referencia
16	$\overline{B1} \wedge \overline{V1} \wedge \overline{B2} \wedge \overline{V2}$	Copiar máx.	Copiar máx.

Tabla 3.17. Reducción de configuraciones que presentan características similares.

Como se observa en la tabla 3.17, de las dieciséis configuraciones con que se contaba en un inicio (tabla 3.16), quedaron sólo once, además de eliminar aquellos factores que no son relevantes para identificar la configuración con su acción. El orden en que se deben presentar las reglas para realizar la acción correspondiente es importante para evitar que una regla se evalúe de manera errónea debido a los factores que fueron eliminados. Por ejemplo, si la regla 16 estuviera antes que la 15 una condición que debería haber sido evaluada como regla 15 se evaluaría como 16, dado que la regla 16 no toma en cuenta el factor de tráfico para la vialidad 2 coincidiendo el resto de los factores por lo que las acciones serían erróneas.

Las reglas de la tabla 3.17 son las que se utilizarán para implementar el algoritmo de contraflujo, primero se deben extraer los factores que describen a cada uno de los semáforos principales en una vialidad de dos sentidos y realizar la acción de la regla que le corresponde.

Al finalizar el algoritmo se obtiene un ajuste en los tiempos de operación entre semáforos que se localizan en vialidades de dos sentidos. Los semáforos que trabajan en una misma fase deben operar simultáneamente por lo que el tiempo de operación debe ser igual. Este algoritmo únicamente se encarga de sincronizar los tiempos de operación de los semáforos involucrados sacrificando en ocasiones la optimización lo que puede mejorarse a futuro con un análisis más profundo del funcionamiento.

4. CONSTRUCCIÓN E INSTALACIÓN DEL SISTEMA

Instalación y ubicación del equipo de cómputo de cada intersección

Según estadísticas de la Secretaría de Movilidad de la Ciudad de México (2016), contamos con 3 mil 076 intersecciones semaforizadas, en donde se colocará un equipo de cómputo embebido en donde se conectarán cada una de las cámaras que revisan el nivel de tráfico de cada semáforo. Este equipo de cómputo es el encargado de realizar las tareas de procesamiento de imagen y enviar la información de nivel de tráfico en la intersección a la estación central. Finalmente, aquí es en donde se recibirán los resultados de los algoritmos de optimización, es decir, los tiempos de operación para cada uno de los semáforos en la intersección; estos tiempos son enviados al controlador de cada semáforo.

Una de las tareas principales del equipo de cómputo embebido es el procesamiento de imagen; como se mencionó anteriormente se usará la herramienta OpenCV con el apoyo del lenguaje de programación Python. Otra tarea que debe realizar es el envío de información de niveles de tráfico a la estación central para posteriormente ser enviada a un puerto de salida que se conecta al controlador de semáforos.

Realizando una simulación en un equipo de cómputo embebido con un procesador de un solo núcleo a 1 GHz con 512 MB de memoria principal y 16 GB de memoria secundaria (memoria flash), el procesamiento de una imagen en promedio demora alrededor de 5 segundos. Una intersección de tipo cruz, una vialidad de dos sentidos y otra de un solo sentido, cuenta con cuatro semáforos. Considerando, que para cada uno de los semáforos se deben tomar diez muestras, el tiempo total de procesamiento necesario para la intersección es de 3 minutos con 20 segundos. Este resultado es adecuado ya que las intersecciones enviarán información a la estación central aproximadamente cada 5 minutos debido a que el nivel de tráfico no presenta drásticas variaciones en pequeños intervalos de tiempo por lo que al tratamiento de la información en este equipo se le puede considerar de tiempo real.

Por lo anterior, este equipo debe contar con las siguientes especificaciones mínimas:

- Procesador de un solo núcleo a 1 GHz.
- Memoria principal de 512 MB.
- Interfaz de red.
- Tantos puertos USB como cámaras en la intersección donde se instalará.
- Puerto de salida que permita conectar al controlador de semáforos.
- Puerto de vídeo y periféricos para tareas de gestión y mantenimiento.
- Sistema operativo que permita administrar el hardware y software necesario.

Un equipo de cómputo embebido comercial y que además es de hardware libre es *LinkSprite pcDuino3* que posee las siguientes especificaciones técnicas.

- Procesador de doble núcleo a 1 GHz de frecuencia de reloj, arquitectura ARM.
- Memoria principal de 1 GB.
- Memoria secundaria de 4 GB Flash.
- Soporte para los sistemas operativos Ubuntu 12.04 o Android 4.2 (*Jelly Bean*).
- Interfaz Ethernet 10/100 Mbps.
- Soporte para lenguajes de programación C, C++, Java y Python.
- Interfaces UART, ADC, PWM, GPIO, I2C y SPI.
- Un puerto USB.

Cabe aclarar que esta tarjeta cuenta con únicamente un puerto USB, pero el protocolo USB permite formar una red de dispositivos con este puerto de comunicación mediante HUB pudiendo conectar 127 dispositivos en un solo puerto raíz. Dado a que la alimentación de la tarjeta está pensada para alimentarse a sí misma y dispositivos de bajo consumo conectados a ella, es necesario que el HUB USB cuente con alimentación propia que suministrará a los dispositivos conectados a él.

También puede usarse la tarjeta *pcDuino1* que tiene sólo un núcleo de procesador, 2 GB en memoria secundaria y dos puertos USB como diferencias, se sigue ajustando a los requisitos.

Este equipo debe localizarse en una caja metálica conectada a tierra y debe tener varias divisiones para alojar:

- El equipo de cómputo embebido.
- El HUB USB con tantos puertos como cámaras en la intersección donde se instalará.
- El controlador de semáforos.
- Fuente de poder de 5V para alimentar los equipos anteriores.
- El equipo de red.
- Módulo de potencia para alimentar las luces de cada semáforo.

La caja debe evitar la entrada del agua de la lluvia y debe de tener rendijas que permitan la entrada y salida de aire para mantener ventilado a los equipos en su interior. Las rendijas de ventilación deben de contar con filtro de aire para evitar la entrada de partículas a su interior. Adicionalmente en el interior de la caja debe haber un ventilador que funcione cuando los equipos estén cerca de su temperatura máxima de operación. Las medidas de la caja pueden depender del tamaño de los equipos en su interior, se define especialmente por la fuente de poder, el HUB y el módulo de potencia. Debe tener un espacio dedicado a la entrada y distribución del cableado (cable de red, alimentación, cámaras y luces de semáforo).

Instalación de la estación central

La estación central es un equipo de cómputo de alto rendimiento en donde se implementarán los algoritmos de optimización de tráfico, así como la base de datos. Esta estación recibirá como entrada la información de tráfico de los semáforos ubicados en cada intersección, recupera información de las características de la intersección y sus semáforos para poder optimizar el tráfico mediante los algoritmos que se le implementen para dicha tarea, finalmente devuelve los tiempos de operación optimizados para cada uno de los semáforos en una intersección.

En la Ciudad de México existen 3 mil 076 intersecciones semaforizadas por lo que la estación central deberá atender 3 mil 076 peticiones en un pequeño intervalo de tiempo, alrededor de 5 minutos.

Realizando una simulación, con un tipo de intersección común (intersección en cruz, una vialidad de un sentido y otra de dos sentidos) para 3070 peticiones, una estación de trabajo con un disco duro de 7200 revoluciones por minuto con 32 MB de memoria cache, procesador de 4 núcleos a 2.4 GHz en frecuencia de reloj y 12 GB de memoria principal, se obtienen todas las respuestas en 8 minutos con 20 segundos ocupando 119.11 KB de memoria principal. Se notó que, en promedio, el servicio del manejador de base de datos, ocupó 5.1% de la capacidad del procesador. Dado a que las transacciones con la base de datos son varias, el disco duro, o bien, la unidad de estado sólido, debe tener un mejor rendimiento.

Por lo anterior y considerando futuras mejoras al proceso de optimización de tráfico, la estación central debe tener mínimo las siguientes características.

- Un procesador de 4 núcleos a 2.4 GHz de frecuencia de reloj.
- Memoria principal de 4 GB (una porción es usada por el sistema operativo).
- Memoria secundaria disco duro de 10000 rpm con 64 MB de cache.

Un equipo de cómputo comercial que puede prestarse para dar servicio como estación central es un servidor de alto rendimiento *HPE ProLiant DL20 Gen9* (procesador E3-1240v5, memoria principal 8 GB-U, controlador de almacenamiento H240) con las siguientes especificaciones.

- Un procesador de 4 núcleos (8 subprocesos) a 3.5 GHz de frecuencia de reloj.
- Memoria principal de 8 GB.
- Memoria secundaria disco duro SSF de 1 TB de almacenamiento.
- Adaptador Ethernet 332i de 1 Gbps de dos puertos.
- Factor de forma para bastidor (1U).

La estación central debe ser instalada en un bastidor junto con los equipos de alimentación y el equipo de red. Todo ello debe estar ubicado en un cuarto que cumpla con determinadas características que garanticen

el correcto funcionamiento de los equipos. A continuación, se enlista las características con que debe contar el cuarto.

- Paredes de concreto con pintura anti flama de color blanco para garantizar una correcta iluminación.
- Lámparas fluorescentes de luz blanca colocadas en el techo que proporcionen una iluminación de 540 lux sobre un metro del piso, el balastro debe estar afuera de la habitación.
- La altura de la habitación efectiva debe ser de 2.44 metro (entre el piso falso y el techo falso) con una puerta con seguro de 1 metro de ancho y 2 metros de altura. El área de la habitación deberá ser de 2 metros en cada lado.
- Debe tener un piso y techo falso en donde se colocará el cableado eléctrico y de datos, así como la instalación de ductos de aire acondicionado y equipo antincendios (estos deben ir en otro cuarto).
- Equipo de aire acondicionado de 9000 BTU con control automático parar operar a una temperatura mínima de 18 °C y un máximo de 24 °C, además de control de humedad para operar entre 30 y 55 % de humedad, Deberá tener filtros de aire para asegurar la menor cantidad de contaminantes en él, debe tomar el aire caliente con ductos en el techo falso y suplirlo a través del piso falso.
- La instalación eléctrica debe ser independiente con una capacidad de 5000 watts de potencia (900 watts ocupados por la estación central y el restante para los demás equipos de red). La instalación debe contar con conexión a tierra física fuera del edificio, el bastidor debe conectarse a esta tierra.
- El equipo antiincendios consiste en tanques de dióxido de carbono que liberan el contenido al detectar humo por medio de sensores. Los tubos se instalarán en el techo falso conectados a los tanques de dióxido de carbono con el dispositivo respectivo para su liberación automática.

Para la comunicación entre la estación central y los equipos de cómputo en cada intersección existen varias posibilidades, por la gran extensión de la ciudad lo más recomendable es usar una red ya existente. Es posible instalar una red de área metropolitana o bien contratar a un ISP (proveedor de servicios de internet) para realizar la interconexión, dado a la poca cantidad de equipos en el cuarto donde se encuentra la estación central es posible instalar los equipos de la entrada de servicios del ISP o bien de la red de área metropolitana en el mismo bastidor.

Implementación de la comunicación entre los equipos de cómputo y la estación central

Como se había planteado anteriormente la comunicación se realizará con base en una arquitectura cliente-servidor. Una manera práctica de implementarlo es a través de una plataforma web en la que el cliente envía su información al servidor en protocolo HTTP y quedando en espera de la respuesta de éste; cuando el servidor termina de procesar los datos recibidos envía los resultados al cliente.

La ventaja de este modelo es que el protocolo HTTP define los elementos necesarios para lograr una comunicación entre dos equipos, por lo que no es necesario crear los elementos necesarios para implementar la comunicación.

Una plataforma web puede implementarse en la mayoría de las redes de datos siempre y cuando los equipos involucrados tengan una dirección IP y rutas para acceder uno al otro. Para implementar la red de datos entre los equipos de cómputo embebido y la estación central existen diferentes opciones en donde las más viables son la construcción de una red metropolitana sea terrestre o aérea, o bien contratar a un proveedor de servicios de internet, ambas opciones tienen sus ventajas y desventajas que se mencionarán a continuación.

En caso de construir una red privada metropolitana se tiene la ventaja de que la red sería privada y quien realice el sistema puede darle el uso que desee. Por otro lado, la construcción de la red implica un amplio estudio ya que tendría cobertura en toda la ciudad lo que implica altos costos de implementación, puede generar conflictos a la hora de implementarse, debe ser persistente a fallas por lo que la topología recomendada sería malla lo que la haría aún más costosa; el mantenimiento y gestión de la red son responsabilidad de quien la implemente.

Si se opta por contratar a un proveedor de servicios de internet, se tiene que llegar a un convenio para que preste el servicio, lo que probablemente involucraría una renta y por tanto un alto costo a largo plazo. No obstante internet es una red que tiene cobertura en toda la Ciudad de México, su infraestructura es bastante robusta por lo que si alguna sección de la red llegase a fallar el resto continuaría funcionando de manera correcta. Una de las grandes ventajas de esta opción es que la administración y mantenimiento de la red se realizan por parte del proveedor. Los equipos de cómputo en cada intersección se pueden conectar a la estación central con una conexión de poca tasa de transferencia a internet, la cantidad de datos que serán transmitidos y recibidos será poca. La estación central debe conectarse a internet con una alta tasa de transferencia debido a que recibirá y transmitirá datos a los equipos de cómputo embebido en todas las intersecciones, también debe tener una dirección de IP fija que debe ser conocida por todos los clientes para realizar la conexión y de esta manera la estación pueda prestar servicio web.

En caso de que se elija contratar a un proveedor de internet, este debe proveer de un encaminador para cada uno de los equipos. En caso de los equipos de cómputo en cada intersección el dispositivo se alojará en la misma caja metálica donde se ubicará el equipo de cómputo embebido. Para la estación central el dispositivo puede ubicarse en la entrada de servicios del edificio en donde se encuentra o bien en el mismo cuarto del servidor.

Desarrollo e implementación del software en lo equipos de cómputo

La estación central por su arquitectura i386 amd64 es capaz de ejecutar una gran cantidad de sistemas operativos. Una opción recomendable a utilizar es *Debian* ya que es uno de los sistemas operativos más utilizados en plataformas web ya que se dispone de las últimas versiones de los paquetes a utilizar, es muy estable, su actualización es sencilla y además es software libre. Dado a que *Debian* no tiene una cronología

para lanzar sus actualizaciones, éstas son lanzadas solamente cuando se ha terminado una versión estable, por ende, la versión publicada al día 2 de abril de 2016 (8.4) es recomendada para instalarse en la estación central con arquitectura amd64.

El programa en el sistema de cómputo embebido debe realizar una petición a la estación central, la cual debe estar preparada para poder responder a la solicitud, por ello en ella debe existir una aplicación de servidor web que atienda las solicitudes de los clientes y les devuelva una respuesta. Un programa de servidor web muy usado es Apache el cual tiene como ventajas ser de código abierto, multiplataforma y muy robusto.

Por otro lado, la base de datos que contendrá la información de nivel de tráfico de manera histórica y características de cada semáforo debe ser implementada en un *RDBMS;* una opción viable es el manejador de bases de datos *MySQL* debido a su fácil configuración e integración a ambientes web, es multiplataforma, de código abierto y es uno de los más utilizados.

Para implementar los algoritmos de optimización de tráfico en la estación central, es necesario contemplar un lenguaje de programación que permita hacer uso del manejador de base de datos ya mencionado y además responder a las solicitudes del servidor web. Se optó por el uso del lenguaje de programación PHP ya que está optimizado para el desarrollo de aplicaciones web integrándose fácilmente con las herramientas ya mencionadas además de contar con funcionalidades muy robustas y estables para la solución de diversos problemas.

El equipo de cómputo embebido posee una arquitectura ARM por lo que la gama de sistemas operativos soportados se reduce de manera significativa, sin embargo, para el dispositivo elegido el fabricante distribuye dos sistemas operativos distintos optimizados para funcionar con ella, de los cuales se elige el sistema operativo Ubuntu 12.04 LTS, la versión distribuida por el fabricante tiene funcionalidad limitada respecto a la distribución original, sin embargo, continúa teniendo soporte para ejecutar programas escritos en Python y la biblioteca OpenCV.

Sistema de cómputo embebido

Para que el equipo de cómputo embebido sea capaz de realizar las tareas de procesamiento y extracción de características de imágenes, identificación de nivel de tráfico y envío de información obtenida a través de las cámaras a la estación central, es necesario instalar el siguiente software en ella.

Instalación del sistema operativo; Lubuntu 12.04 LTS

La tarjeta de cómputo embebido pcDuino3 viene con el sistema operativo *Lubuntu 12.04 LTS* instalado por defecto en la memoria interna del dispositivo, que es de 4 GB. La memoria con la que cuenta el equipo es suficiente para instalar el software requerido, sin embargo, se encuentra muy cerca del límite (3.7 GB) por lo que en caso de requerir mayor cantidad de memoria para instalar software adicional o actualizar el software existente, es necesario instalar el sistema operativo en una memoria flash externa por medio de la ranura SD. El fabricante provee imágenes del sistema operativo optimizado para la tarjeta en su página

web oficial, bastará descargar la imagen del sistema operativo y quemarla en la tarjeta SD por medio de los métodos descritos por el fabricante en el manual que tiene este fin.

El sistema operativo debe configurase cuando inicia por primera vez, algunos parámetros importantes a configurar son:

- **Resolución de la pantalla**: Se configura de acuerdo a la resolución del monitor que se utilizará para la instalación y mantenimiento del equipo de cómputo embebido.
- **Contraseña**: Se establece una contraseña para el usuario por defecto del sistema (Ubuntu) con privilegios de administrador. Para configurar la contraseña de root debe establecerse en la terminal una vez finalizada la configuración con el comando passwd.
- **Teclado**: Se configura el teclado de acuerdo a la distribución de teclado que se usará para configurar y dar mantenimiento al sistema. También debe configurarse la región.
- **Arranque**: Para el ahorro de recursos se debe configurar de tal modo que el escritorio gráfico no inicie y operar el equipo de cómputo embebido desde la línea de comandos. Cabe señalar que el programa se debe modificar para que no muestre elementos gráficos, es decir, eliminar las líneas con la instrucción cv2.imshow.
- **Actualizar**: Se recomienda realizar la actualización de todos los paquetes.

Una vez realizada la configuración inicial del sistema operativo, es necesario actualizarlo y mejorarlo para el correcto funcionamiento de la paquetería necesaria para las tareas que debe realizar el cómputo embebido, esto se realiza a través de los siguientes comandos: apt-get update y apt-get upgrade.

Instalación del ambiente; OpenCV 2.4.9 y complementos

Se debe realizar la instalación de la biblioteca de desarrollo de aplicaciones de visión artificial OpenCV en su versión 2.4.9 para Linux/Mac, entre otros complementos requeridos para ejecutar el sistema de reconocimiento de tráfico.

1. Es necesario ejecutar los siguientes comandos para evitar problemas al momento de la instalación.
    a. apt-get install libcv2.3 libcvaux2.3 libhighgui2.3
    b. apt-get install libcv-dev libcvaux-dev libhighgui-dev
2. Descargar e instalar el compilador.
    a. apt-get install build-essential cmake pkg-config libpng12-0 libpng12-dev libpng++-dev libpng3 libgtk2.0-dev pkg-config git
3. Instalar *GStreamer*, necesario para realizar aplicaciones audiovisuales.
    a. apt-get install libgstreamer0.10-0 libgstreamer0.10-dev gstreamer0.10-tools gstreamer0.10-plugins-base libgstreamer-plugins-base0.10-dev gstreamer0.10-plugins-good gstreamer0.10-plugins-ugly gstreamer0.10-plugins-bad

4. Descargar OpenCV 2.4.9.
    a. wget http://downloads.sourceforge.net/project/opencvlibrary/opencv-unix/2.4.9/opencv-2.4.9.zip?r=http%3A%2F%2Fopencv.org%2Fdownloads.html&ts=1464017653&use_mirror=iweb
5. Extraer el contenido del archivo descargado anteriormente.
    a. unzip opencv-2.4.9.zip (es posible que wget haya asignado otro nombre)
6. Realizar una carpeta en donde se compilará el código de OpenCV y acceder a ella.
    a. cd opencv-2.4.9
    b. mkdir release
    c. cd release/
7. Compilar la biblioteca OpenCV y se desactivar la compilación de los ejemplos para acelerar el proceso.
    a. cmake -D CMAKE_BUILD_TYPE=RELEASE -D BUILD_EXAMPLES=OFF ..
    b. make
    c. make install
    d. ldconfig
8. Instalar OpenCV para Python.
    a. apt-get install python-opencv
9. Finalmente instalar Numpy necesario para trabajar los arreglos de OpenCV en Python
    a. apt-get install python-numpy

No es necesaria la instalación del intérprete de Python, dado que el sistema operativo que provee el fabricante, ya lo tiene implementado.

El sistema de cómputo embebido, por defecto inicia sesión como el usuario *root*, por lo que se pueden ejecutar los comandos anteriores sin problema, no obstante, si se inicia sesión con algún otro usuario, es necesario anteponer la instrucción sudo en la mayoría de los comandos exceptuando, wget, cd y unzip.

Implementación del sistema de identificación de nivel de tráfico

Como se mencionó anteriormente, el equipo de cómputo embebido es el encargado de controlar los sensores de imagen con que cuentan los semáforos en una intersección, captura una imagen de la vialidad que controla cada uno de los semáforos, procesa y extrae características de la imagen que sirven para identificar el nivel de tráfico en la vialidad y finalmente estructura la información obtenida enviándola a la estación central. Esta información es el identificador de la intersección y el de cada semáforo con el nivel de tráfico reconocido.

La identificación de nivel de tráfico debe realizarse en cada una de las vialidades que controla un semáforo por lo que el algoritmo dedicado a ello debe ejecutarse tantas veces como semáforos haya en una intersección.

Cada una de las intersecciones cuenta con sus propias características que son almacenadas en la base de datos, sin embargo, el cómputo embebido debe conocer en qué intersección se encuentra y cuáles son los semáforos en ella. En el momento en el que el equipo de cómputo embebido se comunica con la estación central envía estos datos para que esta consulte las características y ejecute los algoritmos correspondientes para optimizar el tráfico.

Los datos de intersección y sus semáforos varían para cada uno de los equipos de cómputo embebido por lo que es conveniente que estos se encuentren almacenados en un archivo de texto plano que procesará el algoritmo de reconocimiento de tráfico. Cada uno de los semáforos tiene un sensor de imagen asociado que será conectado al cómputo embebido en diferentes puertos, por ello conviene que el mismo archivo de texto que contiene la lista de identificadores de semáforos, se coloque el puerto donde está conectado el sensor que le corresponde. A continuación, se detalla la estructura del archivo para una cierta intersección. Dicha intersección, se ilustra en la figura 4.1.

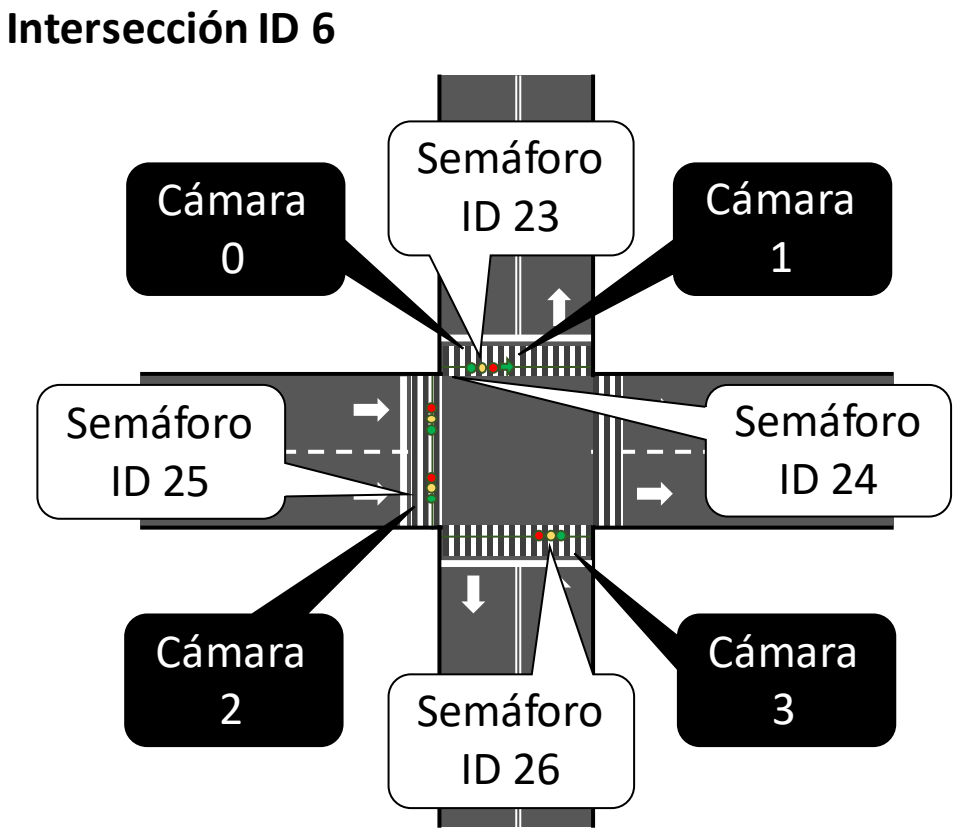


Figura 4.1. Intersección de vialidades mostrando el identificador de semáforo y el puerto donde se conecta su sensor de imagen.

Estructura del archivo		Contenido del archivo	
Intersección ID 6	-	6	0
Semáforo ID 23	Cámara 0	23	0
Semáforo ID 24	Cámara 1	24	1
Semáforo ID 25	Cámara 2	25	2
Semáforo ID 26	Cámara 3	26	3

Tabla 4.1. Archivo de texto plano que refleja la configuración de la intersección de la figura 4.1.

Para la tabla 4.1, que refleja la configuración de la intersección de la figura 4.1, en la primera columna se indica el identificador de la intersección y de los semáforos en ella, en la segunda columna se indica el puerto donde se encuentra conectado el sensor de imagen que le corresponde a cada semáforo, para el caso del identificador de la intersección el valor que se encuentra a su derecha puede ser cualquiera ya que no tiene un sensor de imagen asociado, pero es necesario para completar la matriz. Una vez conocida la estructura, en el contenido del archivo sólo se encuentran los identificadores correspondientes para facilitar su lectura, la separación de las columnas debe ser un espacio en blanco.

Una vez establecido el archivo de texto con los datos de la intersección en donde se encuentra el equipo de cómputo embebido, debe realizarse su lectura y obtener los datos de cada uno de los sensores para conjuntar el identificador de semáforo con el nivel de tráfico detectado, así como el número de intersección que se comunica, en una estructura de datos que será enviada a la estación central para su procesamiento.

Como se está trabajando en una plataforma web, una forma práctica de enviar la estructura de datos (mostrada en la tabla 3.14) es en formato URL por método POST para lo cual en el lenguaje en donde se implementó el algoritmo (Python) se tiene una solución sencilla, este proceso se muestra en desarrollo del código 4.1.

```
import urllib, urllib2 #Bibliotecas para manejo HTTP

import json     #Biblioteca para trabajo con formato JSON
from StringIO import StringIO #Biblioteca para manejo de cadenas desde archivo

#Se lee el archivo donde se encuentran los datos de la intersección

archivo_interseccion = open('interseccion.txt', 'r')
datos_interseccion = archivo_interseccion.read()
archivo_interseccion.close()

#Se convierte en un arreglo bidimensional los datos del archivo
datos_interseccion = numpy.genfromtxt(StringIO(datos_interseccion))

#Se declara un diccionario que tendrá los datos que se enviarán a la la estación central
datos_estacion_central = {}
```

```
#Ciclo infinito para detección de tráfico
while(True):

    #Se realiza la lectura del arreglo
    for i in range(0, len(datos_interseccion)):
        #La primera fila del arreglo contiene el identicador de la intersección
        if i == 0:
            id_interseccion = int(datos_interseccion[i, 0])
            #Se agrega el identicador de la intersección al diccionario
            datos_estacion_central['interseccion'] = id_interseccion
            continue

        #Las demás filas del arreglo contiene el identificador de semáforo y el puerto de cámara asociado
        id_semaforo = int(datos_interseccion[i, 0])
        camara = int(datos_interseccion[i, 1])
        #Se reconoce el tráfico para el sensor de imagen que le corresponde al semáforo
        nivel_trafico = reconocimiento_trafico(camara)
        ''' Se agrega al diccionario el nivel de tráfico identificado para un semáforo
            niveles_trafico[id_semaforo] = nivel_trafico'''
        datos_estacion_central['niveles_trafico[' + str(id_semaforo) + ']'] = nivel_trafico
        #print "Semaforo " + str(id_semaforo) + " tiene un nivel de trafico = " + str(nivel_trafico)

    #Se codifica el diccionario en formato URL
    datos = urllib.urlencode(datos_estacion_central)
    #Se hace la petición a la estación central y se guarda su respuesta
    repuesta = urllib2.urlopen("http://192.168.3.52:8080/RSI/RSI.php", datos)
    #Se decodifica la respuesta de la estación central que fue enviada en formato JSON
    tiempos = json.loads(repuesta.read())

    #print tiempos

    #Permite terminar el programa al presionar la tecla a
    if cv2.waitKey(1) == ord('a'):
        break
```

Código 4.1. Obtención del nivel de tráfico para una intersección; envió de información y obtención de respuesta de la estación central.

Como la estación central devuelve los tiempos de operación de cada semáforo, éstos se reciben y se procesa el formato para que quede finalmente en un arreglo llamado tiempos. En este caso, la dirección de IP que le corresponde a la estación central y a la cual el cómputo embebido debe conectarse a través de una red de datos es *http://192.168.3.52:8080/RSI/RSI.php*, sin embargo, variará en la implementación final. Todo ello debe realizarse en un ciclo infinito para la permanente operación del sistema.

La función "*reconocimiento_trafico*" se basa en el código 3.3, para ello, se le deben realizar las siguientes modificaciones:

- Eliminar las salidas de texto.
- Eliminar el ciclo infinito que envuelve a todo el algoritmo.
- Eliminar la salida del ciclo por medio de una entrada por teclado.
- El parámetro "*camara*" corresponde al puerto donde se conecta el sensor de imagen, por lo que éste debe ser el parámetro que necesita la función "*cv2.VideoCapture*" al llamarse.
- Devolver el nivel de tráfico identificado por el sensor de manera numérica: "0" corresponde a un nivel de tráfico bajo y "1" a un nivel alto.

La estación central devuelve los tiempos de operación en formato JSON que corresponde a la estructura presentada en la tabla 3.15. La estación central, antes de enviar los datos, los tiene almacenados en forma de arreglo, los codifica en formato JSON. El formato JSON consiste en una cadena de texto que representa una cierta estructura de datos con su contenido para hacerla portable. Este formato es práctico, porque al decodificarse en el cómputo embebido, se puede obtener nuevamente una estructura de datos equivalente a la que existía en la estación central, de esta manera, se obtienen los tiempos de operación para cada semáforo en un arreglo y poderles dar algún uso.

Estación Central

Instalación del sistema operativo; Debian 8.4.0

Para la instalación del sistema operativo, debe obtenerse una imagen desde su página oficial, configurar el arranque del servidor para iniciar a partir de esa imagen. La instalación consiste en un asistente, sólo se destacarán los puntos importantes del proceso de instalación:

- El país debe corresponder a México y configurar la zona horaria en la zona central (zona horaria correspondiente a la Ciudad de México).
- Debe configurarse la contraseña para el súper usuario y otro usuario con menor grado de privilegios.
- Utilizar el particionado de disco manual, crear dos particiones, una para el sistema operativo con punto de inicio "/" y tabla de partición *ext4* y otra que se recomienda que tenga el mismo tamaño de la memoria RAM que se usará como intercambio.
- El gestor de paquetes debe configurarse con un servidor geográficamente cerca y sin uso de proxy.
- En la selección de programas sólo debe seleccionarse la opción de "Utilidades estándar del sistema", el resto de las opciones deberán permanecer inactivas para evitar la instalación de paquetes innecesarios y acelerar el proceso de instalación. Por otro lado, se evita instalar la interfaz gráfica, liberando recursos del sistema.
- Debe de instalarse el cargador de arranque GRUB.

Instalación del ambiente; Apache 2.4.10, PHP 2.4.10, MySQL 5.5.49, servidor FTP y SSH

Para poder implementar el sistema de optimización de tráfico en la estación central se debe configurar el servidor web, el manejador de base de datos, el intérprete y un medio de acceso remoto que consiste en configurar el servidor como SSH y FTP.

Para instalar el ambiente, es necesario que un usuario cuente con privilegios de administrador, para ello conviene iniciar como súper usuario por medio del comando su.

Posteriormente se procede a la instalación de los componentes que forman el ambiente por medio de los siguientes comandos:

- apt-get update. Permite actualizar el sistema operativo.
- apt-get install apache2. Instala el servidor web apache
- apt-get install php5. Instala el intérprete de PHP.
- /etc/init.d/apache2 restart. Se reinicia el servicio del servidor web.
- apt-get install php5-mysql. Instala la extensión para manejar bases de datos MySQL en PHP.
- /etc/init.d/apache2 restart. Se reinicia el servicio del servidor web.
- apt-get install mysql-server. Instala el manejador de bases de datos MySQL, se pedirá una contraseña para el usuario *root*.
- apt-get install openssh-server. Instala el servidor de SSH para acceso remoto.
- apt-get install ftpd. Instala el servidor de FTP para transferencia remota de archivos.

El puerto por defecto que tiene configurado el servidor apache es el 80 pero se recomienda que sea cambiado a otro puerto ya que el servidor web no prestará servicio comercial sino para un propósito en específico. Otro aspecto importante es configurar al servidor web por medio del archivo *httpd.conf*, en donde se debe establecer la directriz *Allow* con las direcciones IP de los equipos de cómputo embebido para evitar accesos no deseados a través de una petición web del servidor así como establecer el valor de *MaxClients* con el número de intersecciones en la Ciudad.

Por otro lado, es conveniente configurar los usuarios que se pueden conectar al servidor de manera remota por SSH del archivo de configuración *sshd_config*, estableciendo además parámetros importantes como el puerto, las direcciones IP permitidas, entre otros. También se recomienda configurar al servidor FTP, en especial para los usuarios del sistema que pueden transferir archivos. Esto se hace configurando el archivo *ftpusers*.

Para que las configuraciones mencionadas anteriormente surtan efecto, es necesario reiniciar los servicios correspondientes.

Se debe verificar que el motor que utilice el manejador de bases de datos sea InnoDB ya que éste es el motor capaz de manejar restricciones por llaves foráneas. Para ello debe accederse al manejador por medio del comando mysql –u [nombre_usuario] –p, una vez hecho esto el manejador pedirá la contraseña del usuario y al introducirla correctamente se tiene acceso al manejador.

Una vez dentro del manejador se ejecuta el comando show engines; para revisar que el motor InnoDB se encuentre por defecto, es conveniente deshabilitar el autocommit del manejador para evitar comprometer una transacción sin antes verificar su correcto funcionamiento y de esta manera poder ejecutar el comando rollback; y eliminar transacciones no deseadas. Se realiza con el comando SET autocommit = 0;. Ahora cada que se quiera comprometer una transacción es necesario ejecutar el comando commit;.

Implementación de la base de datos

La implementación de la base de datos se debe realizar por medio del lenguaje declarativo SQL, se debe acceder al manejador y se procede a la creación de la base de datos que contendrá las estructuras necesarias para almacenar los datos relacionados al sistema de optimización de tráfico, realizar la creación, conexión con la base de datos y crear las tablas definidas en el diagrama entidad-relación de la figura 3.26. Debe ejecutarse el script mostrado en el código 4.2.

```
CREATE DATABASE [nombre_base];

CONNECT [nombre_base]

CREATE  TABLE IF NOT EXISTS `Intersecciones` (
  `Interseccion_id` SMALLINT UNSIGNED NOT NULL ,
  `Tipo` VARCHAR(5) NOT NULL ,
  PRIMARY KEY (`Interseccion_id`) )
ENGINE = InnoDB;

CREATE  TABLE IF NOT EXISTS `Semaforos` (
  `Semaforo_id` SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT ,
  `Vuelta_izq` SMALLINT UNSIGNED NULL ,
  `Contraflujo` SMALLINT UNSIGNED NULL ,
  `Interseccion_id` SMALLINT UNSIGNED NOT NULL ,
  `Peso` TINYINT UNSIGNED NOT NULL ,
  `Carriles` TINYINT UNSIGNED NOT NULL ,
  PRIMARY KEY (`Semaforo_id`) ,
  INDEX `fk_Semaforos_Intersecciones1_idx` (`Interseccion_id` ASC) ,
  INDEX `fk_Semaforos_Semaforos1_idx` (`Vuelta_izq` ASC) ,
  INDEX `fk_Semaforos_Semaforos2_idx` (`Contraflujo` ASC) ,
  CONSTRAINT `fk_Semaforos_Intersecciones1`
    FOREIGN KEY (`Interseccion_id` )
    REFERENCES `Intersecciones` (`Interseccion_id` ),
  CONSTRAINT `fk_Semaforos_Semaforos1`
    FOREIGN KEY (`Vuelta_izq` )
    REFERENCES `Semaforos` (`Semaforo_id` ),
  CONSTRAINT `fk_Semaforos_Semaforos2`
```

```
  FOREIGN KEY (`Contraflujo` )
  REFERENCES `Semaforos` (`Semaforo_id` )
)
ENGINE = InnoDB;

CREATE  TABLE IF NOT EXISTS `Niveles_trafico` (
 `Semaforo_id` SMALLINT UNSIGNED NOT NULL ,
 `Nivel_trafico` VARCHAR(1) NOT NULL ,
 INDEX `fk_Niveles_trafico_Semaforos1_idx` (`Semaforo_id` ASC) ,
 CONSTRAINT `fk_Niveles_trafico_Semaforos1`
  FOREIGN KEY (`Semaforo_id` )
  REFERENCES `Semaforos` (`Semaforo_id` )
)
ENGINE = InnoDB;
USE [nombre_base];
```

Código 4.2. Script que crea la base de datos y su estructura.

Una vez creado el esquema para almacenar la información asociada a los semáforos y a las intersecciones se procede a poblar la base de datos de acuerdo a la configuración vial del mapa de la Ciudad de México.

A manera de ejemplo, se muestra cómo deben almacenarse los datos relacionados a dos intersecciones ilustrados en la figura 4.2.

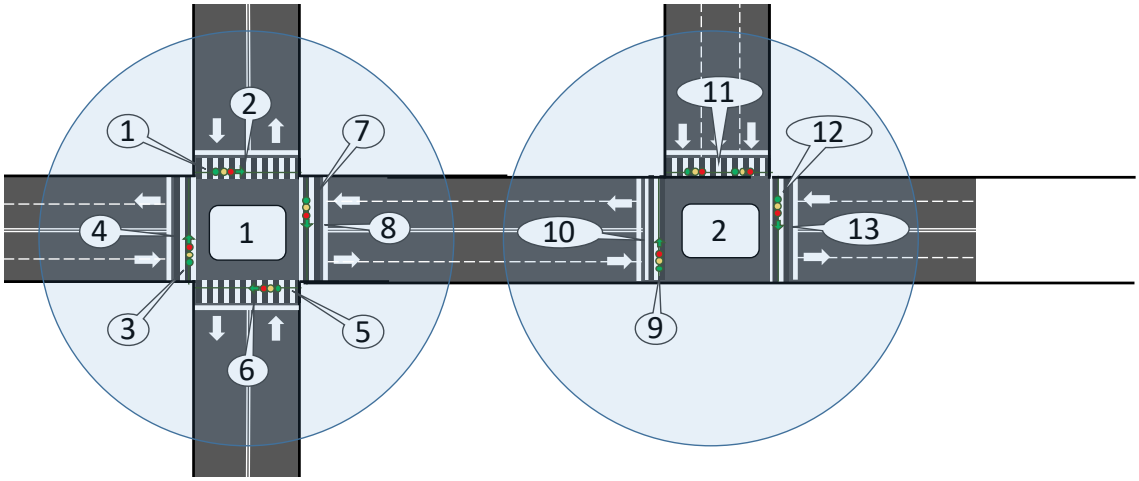


Figura 4.2. Dos intersecciones viales con identificadores. Los identificadores de los semáforos están dentro de óvalos y los identificadores de las intersecciones dentro de cuadros.

Para la configuración vial mostrada en la figura 4.2, se indicará la forma en que deben almacenarse los datos en la base de datos (tabla 4.2) además del script correspondiente (código 4.3).

INTERSECCIONES	
Intersección_id	Tipo
1	C22
2	T21

SEMAFOROS					
Semaforo_id	**Vuelta_izq**	**Contraflujo**	**Interseccion_id**	**Peso**	**Carriles**
1	2	5	1	20	1
2	-	-	1	10	1
3	4	7	1	30	2
4	-	-	1	10	1
5	6	1	1	20	1
6	-	-	1	10	1
7	8	3	1	30	2
8	-	-	1	10	1
9	10	12	2	30	2
10	-	-	2	10	1
11	-	-	2	40	3
12	13	9	2	30	2
13	-	-	2	10	1

Tabla 4.2. Diccionarios de datos que representan los datos que deberán ser almacenados en las tablas *intersecciones* y *semaforos* de acuerdo a la figura 4.2.

```
SET FOREIGN_KEY_CHECKS=0;
INSERT INTO INTERSECCIONES VALUES(1, 'C22');
INSERT INTO INTERSECCIONES VALUES(2, 'T21');
INSERT INTO SEMAFOROS VALUES (1, 2, 5, 1, 2, 1);
INSERT INTO SEMAFOROS VALUES (2, NULL, NULL, 1, 1, 1);
INSERT INTO SEMAFOROS VALUES (3, 4, 7, 1, 3, 2);
INSERT INTO SEMAFOROS VALUES (4, NULL, NULL, 1, 1, 1);
INSERT INTO SEMAFOROS VALUES (5, 6, 1, 1, 2, 1);
INSERT INTO SEMAFOROS VALUES (6, NULL, NULL, 1, 1, 1);
INSERT INTO SEMAFOROS VALUES (7, 8, 3, 1, 3, 2);
INSERT INTO SEMAFOROS VALUES (8, NULL, NULL, 1, 1, 1);
```

```
INSERT INTO SEMAFOROS VALUES (9, 10, 12, 2, 3, 2);
INSERT INTO SEMAFOROS VALUES (10, NULL, NULL, 2, 1, 1);
INSERT INTO SEMAFOROS VALUES (11, NULL, NULL, 2, 4, 3);
INSERT INTO SEMAFOROS VALUES (12, 13, 9, 2, 3, 2);
INSERT INTO SEMAFOROS VALUES (13, NULL, NULL, 2, 1, 1);
SET FOREIGN_KEY_CHECKS=1;
COMMIT;
```

Código 4.3. Script para población de la base de datos de acuerdo a la figura 4.2.

Cabe mencionar que los registros de la tabla *niveles_trafico* se crearan de manera automática por el sistema de optimización de tráfico. Otro aspecto que debe considerarse al momento de poblar la base de datos es deshabilitar las restricciones como lo son las llaves foráneas, debido a que los semáforos en contraflujo hacen referencia el uno con el otro por lo que es imposible insertar un semáforo en contraflujo al tener habilitada la llave foránea. Una vez finalizada la inserción de datos es conveniente habilitar nuevamente las restricciones para las llaves foráneas evitando así contar posteriormente con información inconsistente.

Para los scripts anteriores se recomienda integrarlos en un único archivo de texto con extensión sql, es posible transferir este archivo a la estación central a través del protocolo FTP implementado en la misma. Una vez realizado lo anterior, debe ejecutarse el comando SOURCE [directorio]; para cargar el script que crea la estructura y puebla la base de datos.

Implementación del sistema de optimización de tráfico

El sistema de optimización de tráfico consta principalmente de algoritmos dedicados a esta tarea que como se había mencionado anteriormente se implementan en el lenguaje de programación PHP. Por la complejidad de la estructura del sistema, una buena manera de desarrollarlo es a través del paradigma de programación orientada a objetos.

Como existen diferentes tipos de intersección y para cada una de ellas se realiza un procedimiento diferente de optimización de acuerdo a su estructura física, es conveniente construir una clase para cada una de ellas. Cuando una intersección se comunica con la estación central, se recibe su identificador, pudiendo consultar en la base de datos el tipo de intersección al que pertenece. Una vez consultado es necesario cargar la clase correspondiente a la intersección. El flujo general para cargar la clase correspondiente y construir un objeto que optimice el tráfico para una cierta intersección, es el siguiente:

- Hacer lectura de todos los archivos que están en la carpeta que almacena los códigos fuentes de todas las clases.
- Cargar los archivos por medio de la construcción del lenguaje *require*.
- Consultar en la base de datos el tipo de intersección y se almacena en una variable. La consulta se realiza a partir el identificador que se obtiene al realizar la comunicación.

- Por medio de la capacidad del lenguaje “funciones variables” se usa la variable anterior para construir el objeto que optimiza el tráfico para la intersección.
- Se optimiza el tráfico.
- Se envía por la salida estándar el arreglo de tiempos optimizados en formato JSON como respuesta del servidor.

Lo anterior se realiza en el momento en que el equipo de cómputo embebido hace una petición a la estación central a través del método POST. Ésta recibe datos por medio del servicio de servidor web y los envía al intérprete el cual los almacena en una variable global por lo que pueden ser consultados en cualquier momento durante la ejecución. Posteriormente, cuando el servidor envía datos por su salida estándar son recibidos por el equipo de cómputo embebido.

En el flujo anterior se hace instancia de dos objetos, el primero que tiene la funcionalidad de realizar transacciones con la base de datos y el segundo que corresponde al tipo de intersección para optimizar el tráfico.

GESTIÓN DE BASE DATOS

Para realizar las tareas de consulta y gestión de información en la base de datos se creó una clase que permite realizar dichas tareas. Al instanciar al objeto correspondiente a esta clase, se configura la conexión del intérprete con la base de datos mediante la clase *mysqli,* para lo que es necesario indicar el servidor donde se encuentra la base de datos, nombre y contraseña de un usuario con privilegios para gestionar la base de datos, además del nombre de la misma.

El objeto tiene la capacidad de realizar las funcionalidades mostradas en la tabla 4.3.

Nombre de la función	Tipo de acceso	Entradas	Salidas
consulta_tipo_interseccion	Público	Identificador de la intersección	Tipo de intersección, falso en caso de error
obtener_datos_semaforos	Público	Identificador de la intersección y niveles de tráfico en cada semáforo de la intersección	Identificador de semáforo, número de registros de nivel de tráfico alto, número de registros de nivel de tráfico bajo, identificador de semáforo de vuelta izquierda, identificador de semáforo de contraflujo, tiempo de operación por defecto, número de carriles, falso en caso de error
insertar_historico	Público	Identificador y nivel de tráfico del semáforo	Falso en caso de error
eliminar_historico	Público	Identificador del semáforo	Falso en caso de error

Tabla 4.3. Métodos encargados de gestionar la base de datos.

Ahora que se han mencionado en la tabla 4.3 los métodos que permiten la gestión de la base de datos, se describirá el modo de operación de cada uno de ellos.

- *consulta_tipo_interseccion*. A partir de un identificador de intersección, consulta en la base de datos el tipo de intersección al que pertenece.
- *obtener_datos_semaforos*, Extrae de la base de datos las características asociadas a cada uno de los semáforos en la intersección que se comunica y las almacena en un arreglo asociativo.

  El nivel de tráfico identificado en cada semáforo por el sensor de imagen es agregado al arreglo de características extraídas de la base de datos. La consulta usada por el método depende de si ya existen registros en la tabla *niveles_trafico* (registro histórico), en caso de no existir registros coloca el valor de False al número de registros de nivel de tráfico alto y bajo; si ya existen registros en dicha tabla, se procede a contar el número de registros de nivel de tráfico alto o bajo según sea el caso, se realiza la junta entre esta tabla y la tabla Semáforos para obtener la totalidad de características.

  Una vez obtenido el objeto con datos que devuelve la consulta realizada, se procede a procesar la información para estructurarla de la siguiente forma:

Arreglo
[
    Id semáforo 1 => Arreglo
    [
        Nivel de tráfico
        Registros de niveles alto
        Registros de niveles bajo
        Semáforo de vuelta izquierda
        Semáforo contraflujo
        Tiempo de operación por defecto
        Número de carriles
    ]
    …
    Id semáforo n => Arreglo
    [
        Nivel de tráfico
        Registros de niveles alto
        Registros de niveles bajo
        Semáforo de vuelta izquierda
        Semáforo contraflujo
        Tiempo de operación por defecto
        Número de carriles
    ]

]

Como puede observarse se obtiene toda la información necesaria para conocer el contexto físico de la intersección, lo cual permitirá a los algoritmos de optimización realizar su trabajo. Para efectos de la codificación, los parámetros anteriores son llamados de la siguiente manera:

Nivel de tráfico: *Nivel_trafico*.

Registros de niveles alto: *Alto*.

Registros de niveles bajo: *Bajo*.

Semáforo de vuelta izquierda: *Vuelta_izq*.

Semáforo contraflujo: *Contraflujo*.

Tiempo de operación por defecto: *Peso*.

Número de carriles: *Carriles*.

Lo anterior es el nombre del índice que permite referenciar; por ejemplo, si se quiere obtener el número de carriles en un semáforo con identificador 7, se haría de la siguiente manera:

```
$this->semaforos[7][Carriles];
```

Y si se deseara obtener el número de carriles del semáforo de vuelta izquierda de este mismo semáforo:

```
$this->semaforos[$this->semaforos[7][Vuelta_izq][Carriles];
```

Esto funciona si se considera que el resultado de este método se almacena en un atributo llamado *semaforos*.

- *insertar_historico*. Se utiliza en el algoritmo histórico, permite insertar registros en la tabla *niveles_trafico;* si se identificó un nivel de tráfico alto o bajo se almacena en la tabla un 1 ó 0 respectivamente y se asocia con su semáforo por medio del identificador.
- *eliminar_historico*. Se utiliza en el algoritmo histórico, borra todos los registros existentes en la tabla *niveles_trafico* para un determinado semáforo.

ALGORITMOS DE OPTIMIZACIÓN DE TRÁFICO

Los algoritmos comunes de optimización de tráfico se colocan en una clase (*algoritmos_base*), a su constructor se le pasará el número de intersección, el arreglo que contiene el nivel de tráfico identificado en cada semáforo, el objeto que gestiona la base de datos (ese objeto se creó con anterioridad a partir de la clase descrita anteriormente), los umbrales máximos y mínimo, así como el incremento con los que trabaja el algoritmo histórico.

En el constructor a partir de los parámetros que se le pasan para crear el objeto, se definen los atributos que éste tendrá, sin embargo, los datos de identificador de intersección y el arreglo de niveles de tráfico por semáforo, son usados para poder extraer las características de los semáforos en la intersección que se está trabajando, conociendo así su configuración física, el método *obtener_datos_semaforos* del objeto que gestiona la base de datos nos sirve para tal propósito.

La clase contiene los siguientes métodos mostrados en la tabla 4.4.

Nombre de la función	Tipo de acceso	Entradas	Salidas
vuelta_izquierda	Protegido	Sin entradas	Sin salidas
contraflujo	Protegido	Sin entradas	Sin salidas
historico	Protegido	Sin entradas	Sin salidas
enviar_tiempos	Protegido	Sin entradas	Tiempos de operación por salida estándar
calcula_referencia	Privado	Identificador de semáforo	Sin salidas
calculo_pesos_contraflujo	Privado	Marca de trabajado por *vuelta izquierda*, tráfico actual e identificador de semáforo de vuelta izquierda para los semáforos en contraflujo. Índice del semáforo actual y su contraflujo, referencias de los pesos del semáforo actual y su contraflujo.	Sin salidas

Tabla 4.4. Métodos generales de optimización de tráfico.

A continuación, se describen los métodos anteriores de la tabla 4.4:

- *vuelta_izquierda*. Calcula tiempos de operación asociados a los semáforos principales que cuentan con una vuelta izquierda. El tiempo total de la vialidad es repartido proporcionalmente de acuerdo al número de carriles, entre el semáforo principal y la vuelta izquierda. Este método se basa en el algoritmo de vuelta izquierda descrito en la propuesta de solución.
- *contraflujo* y *calculo_pesos_contraflujo*. Calcula tiempos de operación asociados a los semáforos ubicados en vialidades de dos sentidos, se encarga de ajustar los tiempos de operación de los semáforos de tal modo que los semáforos ubicados en cada sentido cuenten con el mismo valor de tiempo total, es decir, les proporciona el mismo tiempo de operación a los semáforos ubicados en cada uno de los sentidos de la misma vialidad. Estos métodos se basan en el algoritmo de contraflujo descrito en la propuesta de solución.
- *historico*. Calcula el tiempo de operación de un semáforo de acuerdo a los niveles de tráfico identificados a lo largo del tiempo. En caso de haber un cambio en el nivel de tráfico, elimina los registros anteriores, si el nivel de tráfico es el mismo, se incrementa o decrementa el tiempo asociado a cada semáforo y se registra en la base de datos. Los valores de incremento/decremento, así como los umbrales máximo y mínimo de los tiempos, son determinados en el constructor al

crear el objeto. Este método usa al objeto de gestión de base de datos creado con anterioridad. Se basa en el algoritmo histórico descrito en la propuesta de solución.

- *enviar_tiempos*. Toma el arreglo que contiene las características de cada semáforo en la intersección enviando los tiempos de operación por la salida estándar en formato JSON con el objeto de ser recibidos por el equipo de cómputo embebido en la intersección. Cabe señalar que los métodos de optimización, ya han modificado estos tiempos por lo que se envían optimizados.
- *calcula_referencia*. Asocia un valor de referencia para calcular el tiempo de operación para un semáforo en un determinado método en la clase *algoritmos_base*.

La forma en que se optimiza el tráfico en una intersección varia de su tipo, es decir, de su configuración física. Para ello cada uno de los tipos de intersección cuenta con una clase particular encargada de optimizar el tráfico, estas clases heredan los métodos, atributos y el constructor de la clase *algoritmos_base*, haciendo uso de sus métodos dependiendo de la configuración de la intersección, por ejemplo, una intersección puede hacer uso del algoritmo *vuelta_izquierda* dependiendo de si esta cuenta con semáforos con vuelta izquierda o no.

Cada una de las clases que optimizan tráfico para una determinada intersección, tiene un algoritmo en particular de *cruce* que depende de su configuración física. Todas las clases de intersección, cuentan con los métodos ilustrados en la tabla 4.5, aunque su funcionalidad varía de una a otra:

Nombre de la función	Tipo de acceso	Entradas	Salidas
optimizar_trafico	Publico	Sin entradas	Sin salidas
cruce	Privado	Sin entradas	Sin salidas
identifica_vialidad	Privado	Sin entradas	Sin salidas

Tabla 4.5. Métodos de optimización de tráfico en una intersección

La funcionalidad de los métodos descritos en la tabla 4.5 es la siguiente:

- *optimizar_trafico*. Se invoca una vez creado el objeto basado en esta clase. Realiza la ejecución de los algoritmos base, así como el algoritmo de cruce según sea necesario de acuerdo al siguiente orden:

    1. Algoritmo *histórico* (se ejecuta para todas las intersecciones).
    2. Algoritmo *cruce* (se ejecuta para todas las intersecciones, varía de acuerdo al tipo de intersección).
    3. Algoritmo *vuelto_izquierda* (se ejecuta para intersecciones con semáforos de vuelta izquierda).
    4. Algoritmo *contraflujo* (se ejecuta para intersecciones con vialidades de doble sentido).
    5. Algoritmo *enviar_tiempos* (se ejecuta para todas las intersecciones).

- *cruce*. Calcula los valores de referencia asociados a las vialidades en una intersección de acuerdo al número de carriles con que cuenta, el tiempo de operación máximo de los semáforos en cada vialidad y su configuración física. Estos valores pueden ser utilizados por los algoritmos *vuelta_izquierda* y *contraflujo* o bien ser asignados directamente al tiempo de operación final de un semáforo. Estas referencias son grabadas en un arreglo, el índice de cada una es el identificador del semáforo correspondiente; el método *calcula_referencia* usa este arreglo para obtenerla al ejecutar un método de la clase *algoritmos_base* que la use.

- *identifica_vialidad*. De acuerdo al tipo de intersección, este método clasifica a los semáforos en distintos arreglos de acuerdo a la vialidad a la que corresponden. Esta clasificación es necesaria ya que es la característica geográfica que toma en cuenta el método de *cruce*.

Cuando se crea un objeto con base en la clase de una intersección en particular, solamente el método *optimizar_trafico* es público, por lo que es el único que puede ser llamado de manera externa, éste en su funcionalidad hará uso de las clases restantes que se pueden encontrar en la misma clase correspondiente a la intersección, o bien, en la clase *algoritmos_base.*

A manera de ejemplo se especifica el flujo de ejecución de la optimización de tráfico para una intersección en particular, en este caso se eligió una intersección de tipo cruz con vialidades de dos sentidos:

1. Se cargan todos los archivos correspondientes a los algoritmos de optimización.
2. Se crea el objeto encargado de gestionar la base de datos.
3. Del objeto anterior se ejecuta el método *consulta_tipo_interseccion* y se obtiene "*C22*" correspondiente al tipo de intersección.
4. Se crea un objeto a partir de la clase C22 que hereda de la clase *algoritmos_base*. Al crear el objeto, el constructor necesita los tiempos de operación actuales y el objeto para gestionar base de datos, así como los parámetros que usará el método *historico*. Se crea el arreglo que contiene las características de todos los semáforos en la intersección (*semaforos*).
5. Se ejecuta el método *optimizar_trafico* que se encarga de optimizar el tráfico para esta intersección.
6. El método anterior llama al método *historico* que ajusta los tiempos de operación en el arreglo *semáforos* de acuerdo a los registros de nivel de tráfico identificados con anterioridad e inserta el nuevo nivel identificado.
7. El método *optimizar_trafico* llama al método *cruce* de la clase *C22* que a su vez ejecuta el método *identifica_vialidad* de la misma clase.
8. El método *identifica_vialidad* busca un semáforo principal, a partir de él se obtienen los semáforos restantes de la misma vialidad. Los datos de todos ellos se asignan al arreglo *vialidad1*. Posteriormente se busca un semáforo principal perteneciente a la otra vialidad y a partir de él se obtienen los semáforos restantes de la misma. Los datos de todos ellos se asignan al arreglo *vialidad2*. Todo lo anterior se basa en las características presentes en el arreglo *semaforos*.

9. El método *cruce* usa los arreglos del paso anterior para calcular los valores de referencia asociados a cada uno de los semáforos dependiendo de la vialidad en la que se encuentren. Lo anterior se realiza con base en la importancia de las vialidades involucradas tomando en cuenta el tiempo de operación máximo y el número de carriles para cada vialidad. Finalmente, estas referencias se guardan en el arreglo *referencias*.

10. El método *optimizar_trafico* llama al método *vuelta_izquierda* que calcula tiempos de operación asociados a los semáforos principales que cuentan con una vuelta izquierda. El tiempo total de la vialidad es repartido proporcionalmente de acuerdo al número de carriles, entre el semáforo principal y la vuelta izquierda. Para ello se toma en cuenta si el semáforo analizado tiene identificador de semáforo de vuelta izquierda almacenado en el arreglo *semaforos* para realizar o no la acción (los semáforos de vuelta izquierda no tienen referencia a un semáforo principal, es necesario encontrar al semáforo principal que le corresponde para realizar el reajuste de tiempos), mismo en el que se reajustan los tiempos de operación calculados. Este método llama a *calcula_referencia* para obtener el valor de referencia con el que se trabajará. Al finalizar todas las operaciones, se almacenan los identificadores de los semáforos trabajados en el arreglo *banderas_vuelta_izquierda* que servirá para el método *contraflujo*.

11. El método *optimizar_trafico* llama al método *contraflujo* para calcular los tiempos de operación de los semáforos en contraflujo y su vuelta izquierda dentro de una misma vialidad proporcionándoles el mismo tiempo de operación de acuerdo a distintos criterios. Los cálculos se ejecutan si el semáforo no ha sido procesado por este método llamando al método *calculo_pesos_contraflujo* que contiene como tal las acciones a realizar de acuerdo a los tres factores mencionados en la propuesta de solución. Al finalizar los cálculos, los identificadores de los semáforos trabajados se agregan al arreglo *banderas_contraflujo* ya que los semáforos en contraflujo (al ser ambos semáforos principales) se referencian mutuamente y se ejecutarían los cálculos dos veces si esto no se hace. Los tiempos de operación se reajustan en el arreglo *semaforos*.

12. El método *calculo_pesos_contraflujo* transforma los valores que se le pasan por parámetro en valores booleanos que son los factores que se toman en cuenta para realizar una acción de acuerdo a la tabla 3.17 por medio de condiciones.

13. El método *optimizar_trafico* llama al método *enviar_tiempos* que toma el arreglo *semáforos* para extraer de él los tiempos de operación de cada uno de los semáforos en la intersección. El arreglo se envía por salida estándar en formato JSON, cada índice contiene el identificador del semáforo y su contenido es el tiempo de operación.

Gráficamente, para el ejemplo anterior, la relación de las clases, sus métodos, atributos y su nivel de acceso se muestra la figura 4.3.

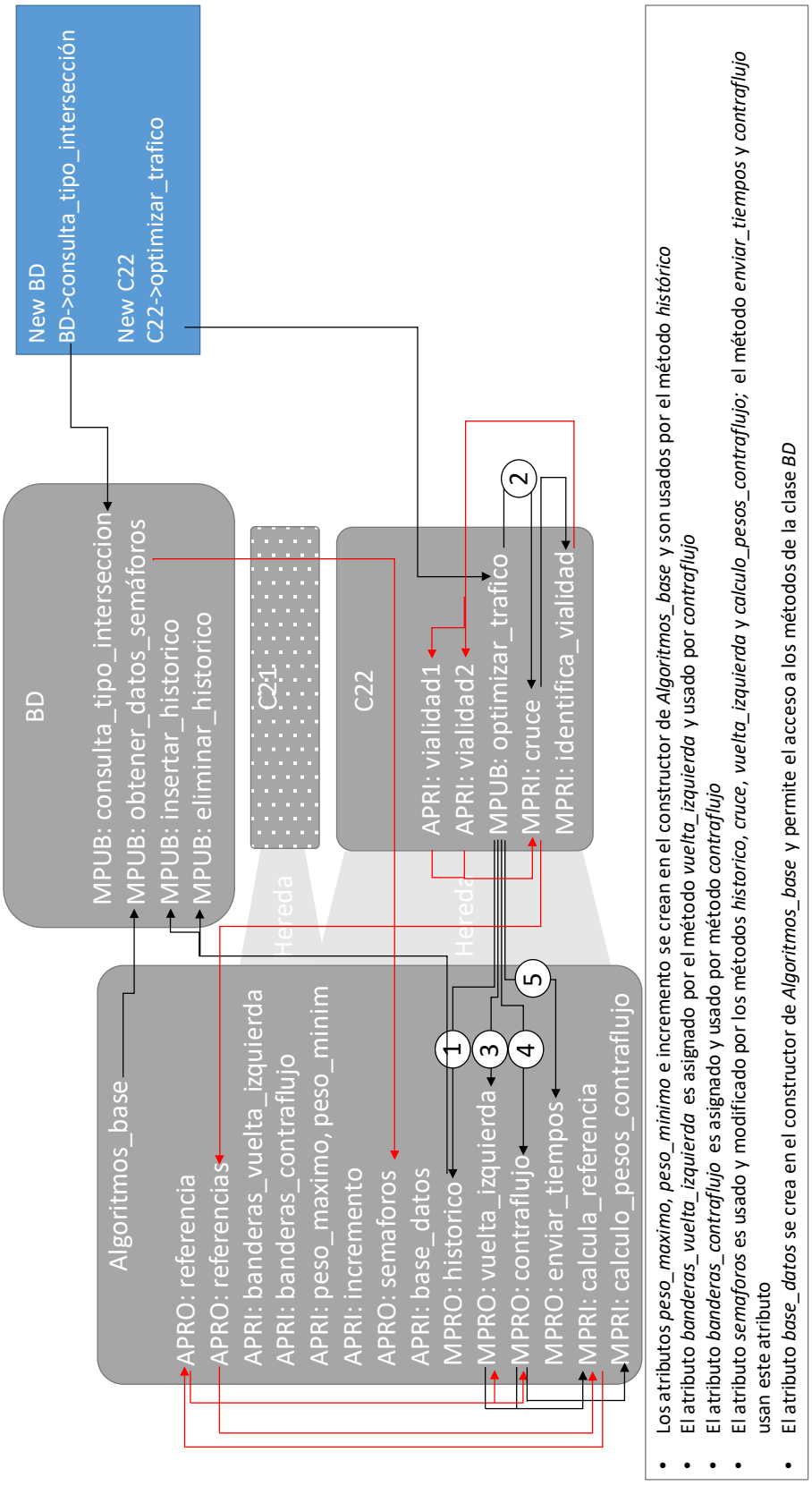


Figura 4.3. Relación de clases, sus métodos y atributos del sistema implementado en la estación central.

Para la figura 4.3, las líneas en color negro representan la relación entre métodos y las flechas indican qué método manda a llamar a cuál, por ejemplo, si se tiene Método A -> Método B, significa que el Método A manda a llamar al Método B. En caso de que una línea negra tenga como referencia el nombre de una clase, se hace referencia al constructor. Recordar que el constructor de las clases de los tipos de intersección es realmente el de *Algoritmos_base* ya que estas heredan de ella. Se tienen más clases para cada tipo de intersección, pero solo se hace uso de la clase para el tipo *C22*, que corresponde al tipo de intersección que se comunica a la estación central. Para el caso del método optimizar tráfico, las líneas tienen un orden que denota el orden de ejecución de los métodos marcado por un número dentro de un círculo.

Las líneas en color rojo representan la relación entre métodos y atributos de alguna clase y las flechas indican si los atributos son asignados o leídos por algún método, es decir, si la flecha sale de un método y llega a un atributo significa que se asigna, si la flecha sale de un atributo y llega a un método significa que es leído.

El cuadro azul representa el flujo principal del programa en donde la palabra new indica que se crea un objeto a partir de cierta clase, para después hacer uso de sus métodos públicos. Nótese que las clases pertenecientes a las intersecciones (a manera de ejemplo solo se muestran los tipos *C21* y *C22)* tienen una relación con la clase *Algoritmos_base,* que significa que éstas heredan sus atributos y métodos con acceso protegidos. Muchas relaciones que deberían haber sido indicadas por líneas se indican en la acotación ubicada en el pie del diagrama para simplificarlo.

A continuación, se describen las siglas que anteponen los métodos y atributos mostrados en el diagrama.

- **APUB**: Atributo con acceso público.
- **APRO**: Atributo con acceso protegido.
- **APRI**: Atributo con acceso privado.
- **MPUB**: Método con acceso público.
- **MPRO**: Método con acceso protegido.
- **MPRI**: Método con acceso privado.

Al momento de realizar la carga de todas las clases al inicio del programa, es importante cargar la clase *algoritmos_base* en un inicio, si esto no se realiza habrá problemas al cargar la clase de una intersección ya que éstas heredan los métodos y atributos de la clase *algoritmos_base*.

Una vez implementado todo lo anteriormente descrito en el lenguaje de programación PHP, deben transferirse todo el árbol de directorios con cada uno de los scripts a la estación central por medio del protocolo FTP. Un usuario con privilegios de administrador, deberá localizar los archivos en la carpeta */var/www/http/* para que puedan ser ejecutados por el intérprete de PHP a través de solicitudes por medio del servidor web. Para que esto sea posible, deben modificarse los permisos de acceso a esta carpeta por

medio del comando chmod de manera recursiva en *0755* (el propietario tiene privilegios de lectura, ejecución y escritura, el grupo de usuario y el acceso público puede leer y ejecutar el archivo).

Instalación de cámaras en los semáforos

El sensor de imagen necesario para realizar el reconocimiento del nivel de tráfico en una vialidad controlada por un semáforo es una cámara que debe contar con las siguientes características:

- Contar con interfaz USB.
- Sensor de imagen CMOS.
- Resolución mínima VGA.
- Protección ultravioleta.
- Resistencia total ante el polvo.
- Resistencia ante chorros de agua de gran presión desde cualquier dirección.
- Lente con autoenfoque.

Es recomendable que el sensor de imagen cuente con visión nocturna por medio de luz infrarroja, aunque no es totalmente necesario ya que se realizará una etapa de entrenamiento que permitirá clasificar el nivel de tráfico en condiciones de poca luz.

La cámara debe colocarse de tal modo que tenga una visión completa y clara de la vialidad, evitando obstáculos como pueden ser árboles, anuncios u otros objetos. Para ello la cámara debe colocarse en una estructura metálica resistente al viento en forma de arco, permitiendo que la cámara sea colocada en el centro de la vialidad. Cabe señalar que la cámara no se debe encontrar cerca del semáforo ya que cuando este se encuentra en luz roja, tienden a agruparse una cantidad de automóviles lo que provocaría que el sensor de imagen detecte una gran densidad de automóviles pudiendo obtener un resultado equivocado. Para ello el arco en donde irá colocada la cámara deberá estar a una cierta distancia alejado del semáforo dependiente de la vialidad, donde casi no se detecten automóviles en su campo de visión cuando el nivel de tráfico sea bajo y la luz del semáforo sea roja, obteniendo resultados más acertados.

Al encontrase la cámara a una distancia considerable de la caja en donde se ubica el equipo de cómputo embebido, el protocolo USB puede no soportarla por lo que se puede recurrir a alguna de las siguientes soluciones:

- Instalar repetidores USB activos.
- Transmitir el protocolo USB en cable UTP por medio de un convertidor.
- Convertir el tipo de medio.
- Usar otra tecnología de comunicación.

Una buena opción, es transmitir el protocolo USB en cable UTP ya que existe gran variedad de productos comerciales en el mercado con esta capacidad, en promedio soportan hasta 100 metros de distancia y necesitan una alimentación a 5 Volts en donde se encuentra el destino de la conexión, ya que la naturaleza de la corriente continua del protocolo USB, hace que la señal se degrade en una trayectoria larga. La corriente necesaria depende del consumo de la cámara.

Control de luces en los semáforos

Los semáforos tienen un ciclo de operación, cada etapa del ciclo maneja diferentes estados. Los estados que maneja un semáforo clásicamente son tres, la luz verde que indica a los vehículos poder continuar su marcha, la luz ámbar indica a los vehículos detenerse o en caso de no ser posible, cruzar con precaución; la luz roja indica a los vehículos detener su marcha para ceder el paso a peatones u a otra vialidad. Algunos semáforos tienen un estado extra, la luz verde en estado intermitente, indica estar a punto de cambiar al estado de luz ámbar.

De acuerdo a lo establecido en el reglamento de tránsito que entró en vigor en diciembre de 2015, muchos semáforos han sido modificados para tener una fase exclusiva de cruce de peatones.

Los semáforos cuentan con dos modos de operación, diurno y nocturno, excepto en aquellos localizados en las vialidades más importantes. Además del funcionamiento automático con el que cuenta cada semáforo, se tiene un modo manual, que consiste en un botón que al ser presionado efectúa un cambio de fase en los semáforos de una intersección vial.

Equipo controlador de semáforos

Una vez que la estación central envía los tiempos de operación optimizados a los equipos de cómputo embebido en cada intersección, estas los reciben y necesitan ser reflejados en los semáforos. Como se había mencionado los equipos de cómputo embebido sugeridos tienen interfaces GPIO (entradas y salidas de propósito general) que pueden ser usadas para controlar las luces en los semáforos e incluso para mejoras en la funcionalidad a futuro.

Para ello es necesario un equipo controlador que reciba la información del equipo de cómputo embebido, la mantenga almacenada y con ello realice todas las fases para los semáforos en la intersección. El equipo controlador debe prevenir que no se modifiquen los tiempos de operación con los que trabajan los semáforos mientras no se hayan terminado todas las fases de operación, esto se debe a que es posible que el equipo de cómputo embebido envíe información nueva al equipo controlador mientras se lleva a cabo este proceso.

Para poder realizar lo anterior, el equipo controlador debe de tener dos buffers de almacenamiento. El primero recibe la información que envía el equipo de cómputo embebido en cualquier momento y el segundo mantiene los tiempos de operación mientras se ejecutan las fases para un ciclo de operación en la intersección. Una vez que finalizan las fases, el contenido del segundo buffer se actualiza con el contenido del primero.

A grandes rasgos, se sugiere el siguiente flujo:

1. Recibir los tiempos de operación y almacenarlos en una estructura de datos dentro del primer buffer. Éste estará listo para recibir nueva información en cualquier momento.
2. Copiar los tiempos de operación del primer buffer al segundo.
3. Ejecutar las fases de operación de los semáforos con base en los tiempos almacenados en el segundo buffer.
4. Ejecutar a partir del paso 2.

El equipo de cómputo embebido por medio de uno de sus pines, encenderá una bandera para indicar al equipo controlador que se transmitirá información, para que éste otro se prepare para recibir datos. Por medio de otro pin, se enviará la información sobre los tiempos de operación de manera serial en una estructura y protocolo predefinidos para ambos equipos. Adicionalmente debe de existir una conexión de referencia, se recomienda que sea con base en tierra física.

Por otro lado, se recomienda la existencia de una memoria permanente que contenga la información de los tiempos de operación por defecto en caso de fallas en el equipo de cómputo embebido, en la red o en la estación central.

Debe tener almacenado la tabla de fases con los semáforos según el tipo de intersección, posteriormente, se complementará con la información de tiempos de operación para finalmente llevar a cabo el control de los semáforos.

En cada una de las fases, el equipo controlador debe de tener integrado el funcionamiento de un semáforo, es decir, el orden de las tres luces, duración de cada una, intermitencias y relaciones con los demás semáforos. Por ejemplo, cuando un semáforo esté a punto de entrar en operación con luz verde, los que dejan de estarlos, deben anticipar la intermitencia de la verde y luego la luz ámbar y después la luz roja, dejar un tiempo de seguridad y poner luz verde al semáforo que operará (aquel que permitirá el flujo de automóviles), ese tiempo es el mismo que determina la estación central.

El equipo de cómputo embebido, bien puede realizar las funciones del equipo controlador, y usar los pines que tiene para controlar directamente las luces, no obstante, debido a las limitaciones de hardware del equipo de cómputo embebido y a las tareas de reconocimiento tráfico que debe realizar, su respuesta al controlar las luces puede tener retardos, lo que provocaría que el sistema no se comporte como de tiempo real para este propósito. Para evitar este problema, se recomienda que el equipo controlador, no sea el de cómputo embebido, pudiendo ser simplemente un microcontrolador.

Debido a la cantidad limitada de pines con los que cuenta el equipo controlador y la complejidad de su expansión, es posible enviar el estado de cada una de las luces de manera serial en lugar de paralela, en un tiempo muy corto, y almacenarlos en registros de corrimiento universales anidados en función de la cantidad de luces a controlar. Una vez completados los datos de las luces en los registros, se activa su salida para que esta sea reflejada finalmente en las luces de los semáforos.

Dependiendo de la naturaleza de las luces de los semáforos en una intersección, que pueden ser de corriente alterna o directa en luz incandescente o bien, de corriente alterna en una matriz de diodos emisores de luz; se debe de construir un módulo de potencia realizado con transistores o triacs dependiendo del tipo de corriente que manejan las luces, pudiendo tener una derivación de alimentación ajustada de la fuente de corriente continua donde se conectan los demás equipos de la intersección, o bien tener alimentación de corriente alterna proveniente de la toma general de energía eléctrica provista para los semáforos, posiblemente ajustada por un transformador. Es importante colocar optoacopladores entre los pines de salida del equipo controlador y el módulo de potencia para evitar posibles daños a todos los equipos.

Finalmente, se recomienda instalar un equipo de energía de respaldo que alimente por lo menos al equipo controlador y al módulo de potencia con el fin de hacer más robusto el sistema ante posibles fallos.

El envío de la información depende del protocolo que se cree para comunicar el equipo de cómputo embebido y el equipo controlador manteniendo siempre la relación entre semáforos y tiempos de operación. En la implementación del programa que reconoce tráfico para el equipo de cómputo embebido (código 4.1), al final, existe un arreglo llamado *tiempos* que es la respuesta de la estación central con la optimización deseada y tiene que enviarse al equipo controlador.

La tarjeta *pcDuino*, tiene la capacidad de manejar sus puertos GPIO desde un programa elaborado en C o en Python por medio de las bibliotecas que tiene integradas en el sistema operativo.

Al final del código 4.1, después de obtener el arreglo de tiempos como respuesta del servidor, se necesitan enviar al equipo controlador por medio de los pines GPIO que tiene el equipo de cómputo embebido. Explotando la capacidad de la tarjeta *pcDuino*, la forma de usar uno de sus pines como salida se muestra en el código 4.4.

```
#!/usr/bin/env python

##La biblioteca OS permite el manejo del hardware espacial de la tarjeta pdDuino.
import time, os

''' Los pines GPIO se manejan leyendo y escribiendo en los archivos asociados a cada uno.
    Estos se encuentran implementados en la distribución del sistema que proporciona el fabricante de la tarjeta.'''
##Se crean accesos a las rutas de los archivos que permiten manejar los pines.
GPIO_MODE_PATH = os.path.normpath('/sys/devices/virtual/misc/gpio/mode/')
GPIO_PIN_PATH =os.path.normpath('/sys/devices/virtual/misc/gpio/pin/')
GPIO_FILENAME ="gpio"

##Los siguientes arreglos almacenarán los punteros de los archivos.
pinMode = []
pinData = []

##Los siguientes arreglos equivalen al valor para las acciones y configuraciones de los pines.
HIGH = "1"
LOW =  "0"
INPUT = "0"
OUTPUT = "1"
INPUT_PU = "8"
```

```
##Se asocia cada uno de los pines con el archivo que lo maneja.
for i in range(0,18):
    pinMode.append(os.path.join(GPIO_MODE_PATH, 'gpio' + str(i)))
    pinData.append(os.path.join(GPIO_PIN_PATH, 'gpio' + str(i)))

##Todos los pines se establecen como salida.
for pin in pinMode:
    file = open(pin, 'r+')  ##El archivo se abre en modo lectura y escritura.
    file.write(OUTPUT)      ##Se configura el pin como salida escribiendo.
    file.close()            ##Se cierra el archivo.

##Todos los pines tendrán como salida 'bajo'.
for pin in pinData:
    file = open(pin, 'r+')
    file.write(LOW)
    file.close()

''' Lo anterior corresponde a la configuración y debe localizarse en una función en el programa principal.
    Lo siguiente permite encender y apagar el pin 13 cada segundo'''
while(1):
    file = open(pinData[13], 'r+')
    file.write(HIGH)
    file.close()
    time.sleep(1)
    file = open(pinData[13], 'r+')
    file.write(LOW)
    file.close()
    time.sleep(1)
```

Código 4.4. Script en Python para manejar el pin 13 de la tarjeta *pcDuino* como salida haciéndolo parpadear.

Cabe señalar que los pines de la tarjeta son de muy baja corriente y funcionan a 3.3 Volts; para manejar de manera segura los puertos, es necesario que en el circuito donde está el equipo controlador se implemente un buffer alimentado a esa misma diferencia de potencial, por lo que se recomienda que sea de tecnología CMOS. De esta manera se protegen los puertos del equipo de cómputo embebido y además se pueden realizar distintas pruebas con diodos emisores de luz durante el desarrollo del equipo controlador.

Lo anterior es una forma de implementar la comunicación, pero si esta comunicación se realiza asíncronamente puede presentar el problema de que no se responda al tiempo que se desea, debido a que el sistema operativo está funcionando pudiendo consumir ciclos de reloj del procesador en otras actividades de mayor prioridad (misma razón por lo que no se recomienda que el equipo de cómputo embebido sea el equipo controlador) por lo que se recomienda que la comunicación sea síncrona, teniendo un pin que marque la referencia de tiempo en el que se envían los datos y puedan llegar de manera segura al equipo controlador.

La tarjeta tiene protocolos de comunicación en hardware, como lo es RS232, I2C y SPI, en el cual lo único que depende del procesador es enviar la información, el componente de hardware se encargará de hacerlo según el protocolo en los tiempos establecidos.

El protocolo RS232 puede ser la mejor manera de hacerlo, dado a que está implementado tanto en el cómputo embebido como en la mayoría de los microcontroladores (que será el equipo controlador). Una vez obtenidos los tiempos de operación optimizados en el flujo del programa, bastará mandar por el puerto serie los tiempos de operación asociados con su semáforo en forma de cadena al equipo controlador, éste podrá interpretarla y realizar su trabajo.

El código 4.5, muestra la forma de acceder al puerto serie RS232 en la tarjeta *pcDuino* en Python.

```
#!/usr/bin/env python

##Se carga la biblioteca para el manejo del puerto serie. Ya no es necesario tener la biblioteca time para ejecutar retardos en el procesador.
import serial

##Se configura el puerto, la configuración debe corresponder con la del equipo controlador.
puerto_serie = serial.Serial('/dev/ttyS1', 115200, timeout = 10)

##Se manda la información.
puerto_serie.write("S1=34;S2=57.2")

##Se espera a recibir una respuesta(instrucción opcional).
equipo_controlador = puerto_serie.read()

##Se debe cerrar el puerto al terminar el programa.
puerto_serie.close()
```

Código 4.5. Script en Python que permite comunicarse con el puerto serie de la tarjeta *pcDuino* en protocolo RS232.

El método anterior es más seguro porque el tiempo que maneja el protocolo se controla por el hardware asociado al puerto en la tarjeta, liberando al procesador de esa carga. Otra ventaja es la facilidad de implementación, no se necesita crear un protocolo de comunicación, lo único necesario es crear el formato para enviar la información al equipo controlador.

Es importante que los puertos tengan la misma configuración tanto en el equipo de cómputo embebido como en el equipo controlador. Es necesario que exista un conversor conectado al puerto serie del equipo de cómputo embebido que provea la diferencia de potencial y corriente que marca el protocolo RS232, ya que a la salida de la tarjeta la diferencia de potencial se basa en 3.3 Volts para *alto* y 0 Volts para *bajo*. El conversor debe de funcionar a esa diferencia de potencial, del lado del equipo controlador debe colocarse el mismo conversor pero que tenga como salida la tensión de funcionamiento del microcontrolador para lograr la comunicación. Estos conversores son circuitos integrados y suelen ir acompañados de capacitores electrolíticos.

5. ETAPA FINAL

Pruebas funcionales

Para demostrar que el ajuste de tiempos sea el adecuado, se presentará un ejemplo del funcionamiento de los algoritmos de optimización de tráfico para la intersección que se muestra en la figura 5.1.

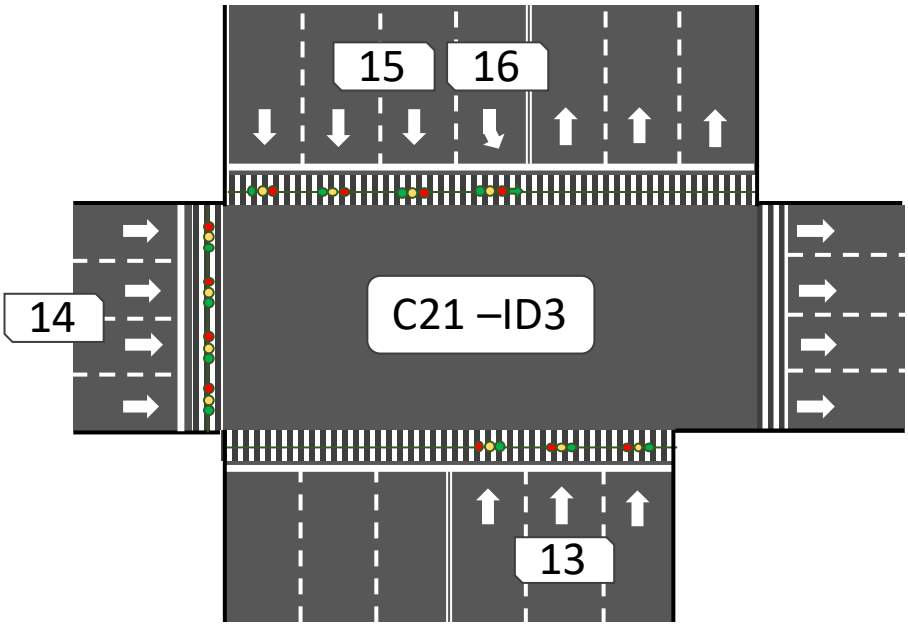


Figura 5.1. Vialidad de tipo cruz, una vialidad de dos sentidos y otra de un solo sentido.

A continuación, se describen las características de los semáforos que conforman a la intersección. En la implementación del sistema, esta información se encuentra almacenada en la base de datos.

- *Semáforo con ID 13: tiene 3 carriles, 50 segundos de tiempo de operación, 6 registros de nivel de tráfico alto, no se relaciona con un semáforo de vuelta izquierda y se relaciona con un semáforo en contraflujo cuyo ID es 15.*
- *Semáforo con ID 14: tiene 4 carriles, 60 segundos de tiempo de operación, 1 registro de nivel de tráfico alto, no se relaciona con un semáforo de vuelta izquierda y no se relaciona con un semáforo en contraflujo.*
- *Semáforo con ID 15: tiene 3 carriles, 70 segundos de tiempo de operación, 6 registros de nivel de tráfico alto, se relaciona con un semáforo de vuelta izquierda cuyo ID es 16 y se relaciona con un semáforo en contraflujo cuyo ID es 13.*

- *Semáforo con ID 16: tiene 1 carril, 20 segundos de tiempo de operación, 6 registros de nivel de tráfico bajo, no se relaciona con un semáforo de vuelta izquierda y no se relaciona con un semáforo en contraflujo.*

Se supondrá que los niveles de tráfico identificados en cada uno de los semáforos en la intersección son los siguientes: Semáforo con ID 13: Nivel de tráfico Alto; Semáforo con ID 14: Nivel de tráfico Alto; Semáforo con ID 15: Nivel de tráfico Alto; Semáforo con ID 16: Nivel de tráfico Bajo.

Los niveles de tráfico identificados por cada semáforo en la intersección, así como los registros de nivel de tráfico en la base de datos se deben a que en la vialidad que controlan los semáforos con ID 13 y 14 existe un embotellamiento mientras el resto de las vialidades se encuentran con un buen flujo automovilístico, aunque la vialidad que controla el semáforo 14 a pesar de tener un buen flujo automovilístico ocasionalmente presenta una alta densidad de vehículos.

Con base en el nivel de tráfico identificado en las vialidades que controlan cada uno de los semáforos, el equipo de cómputo embebido ubicado en la intersección con ID 3 envía la siguiente información en forma de cadena a la estación central:

niveles_trafico%5B13%5D=1&niveles_trafico%5B14%5D=1&niveles_trafico%5B15%5D=1&niveles_trafico%5B16%5D=0&interseccion=3.

Una vez que la estación central recibe la información del equipo de cómputo en la intersección procede a extraer las características de la intersección; con estos datos y los recibidos por el equipo de cómputo embebido se realiza una estructura de datos con la que trabajarán los algoritmos de optimización de tráfico.

Arreglo

```
[
    Id semáforo 13 => Arreglo
    [
        Nivel de tráfico: Alto
        Registros de niveles alto: 6
        Registros de niveles bajo: 0
        Semáforo de vuelta izquierda: Nulo
        Semáforo contraflujo: 15
        Tiempo de operación por defecto: 50
        Número de carriles: 3
    ]
    Id semáforo 14 => Arreglo
    [
        Nivel de tráfico: Alto
        Registros de niveles alto: 1
        Registros de niveles bajo: 0
```

Semáforo de vuelta izquierda: Nulo
Semáforo contraflujo: Nulo
Tiempo de operación por defecto: 60
Número de carriles: 4

]

Id semáforo 15 => Arreglo

[

Nivel de tráfico: Alto
Registros de niveles alto: 6
Registros de niveles bajo: 0
Semáforo de vuelta izquierda: 16
Semáforo contraflujo: 13
Tiempo de operación por defecto: 70
Número de carriles: 3

]

Id semáforo 16 => Arreglo

[

Nivel de tráfico: Bajo
Registros de niveles alto: 0
Registros de niveles bajo: 6
Semáforo de vuelta izquierda: Nulo
Semáforo contraflujo: Nulo
Tiempo de operación por defecto: 20
Número de carriles: 1

]

]

A partir de los datos anteriores la estación central ejecuta los algoritmos de optimización. El algoritmo *historico* es el primero en ejecutarse con los siguientes parámetros: umbral máximo de 600 segundos, umbral mínimo de 20 segundos, incremento o decremento de 10 segundos. Se obtienen los siguientes resultados:

- **Semáforo 13**: 110 segundos.
- **Semáforo 14**: 70 segundos.
- **Semáforo 15**: 130 segundos.
- **Semáforo 16**: 20 segundos.

A la postre el algoritmo *cruce* se ejecuta con base en los tiempos anteriores obteniendo el siguiente ajuste:

- **Semáforo 13**: 110 segundos.
- **Semáforo 14**: 76 segundos.
- **Semáforo 15**: 130 segundos.
- **Semáforo 16**: 20 segundos.

La vialidad horizontal al no contar con semáforo en contraflujo ni de vuelta izquierda no es procesado por los algoritmos correspondientes por lo que el algoritmo de *cruce* es el encargado de calcular su tiempo de operación final en lugar de asignar un valor de referencia. Para los semáforos 14, 15 y 16 por sus características deben ser procesados por los algoritmos *vuelta_izquierda* y *contraflujo*, el algoritmo *cruce* se encarga de calcular el valor de referencia correspondiente a estos semáforos que es de 124 segundos.

Posteriormente se ejecuta el algoritmo *vuelta_izquierda* que solamente afecta a los semáforos 15 y 16, quedando los tiempos de la siguiente forma:

- **Semáforo 13**: 110 segundos.
- **Semáforo 14**: 76 segundos.
- **Semáforo 15**: 118 segundos.
- **Semáforo 16**: 6 segundos.

Finalmente se ejecuta el algoritmo *contraflujo,* con lo que se obtienen los tiempos finales de operación para cada uno de los semáforos en la intersección 3, quedando de la siguiente forma:

- **Semáforo 13**: 112 segundos.
- **Semáforo 14**: 76 segundos.
- **Semáforo 15**: 118 segundos.
- **Semáforo 16**: 6 segundos.

De acuerdo a los tiempos de operación finales se observan los siguientes resultados:

- El semáforo 13 contaba con un tiempo de operación de 50 segundos antes de ejecutar los algoritmos de optimización, después de ser procesado el tiempo de operación final es de 112 segundos. Su tiempo de operación aumentó un 124%.
- El semáforo 14 contaba con un tiempo de operación de 60 segundos antes de ejecutar los algoritmos de optimización, después de ser procesado el tiempo de operación final es de 76 segundos. Su tiempo de operación aumentó un 27%.
- El semáforo 15 contaba con un tiempo de operación de 70 segundos antes de ejecutar los algoritmos de optimización, después de ser procesado el tiempo de operación final es de 118 segundos. Su tiempo de operación aumentó un 69%.

- El semáforo 16 contaba con un tiempo de operación de 20 segundos antes de ejecutar los algoritmos de optimización, después de ser procesado el tiempo de operación final es de 6 segundos. Su tiempo de operación disminuyó un 70%.

Los ajustes finales en los tiempos de operación corresponden a los tiempos optimizados para cada semáforo, es decir, son aquellos que permiten liberar el flujo automovilístico en las vialidades que controlan los semáforos. En los porcentajes anteriores es notable que el semáforo que identificó un mayor número de veces un nivel de tráfico alto (semáforo 13) es aquel que aumentó en mayor porcentaje su tiempo de operación en luz verde, permitiendo liberar el tráfico en esa vialidad. En el caso del semáforo en donde no se identificó un nivel de tráfico alto (semáforo 16) su tiempo de operación en luz verde disminuyó un 70% permitiendo utilizar ese tiempo para otra vialidad con un nivel de tráfico alto. Por otro lado, el semáforo 14 que no presenta gran cantidad de tráfico (determinado por el algoritmo histórico) aumentó su tiempo de operación en luz verde un 27% siendo este porcentaje inferior al del semáforo 13, utilizando ese tiempo para darle mayor prioridad.

Los tiempos de operación optimizados son enviados al equipo de cómputo embebido en la intersección 3 en formato JSON, de la siguiente forma:

{"13":111.7305458768836,"14":76.190476190476,"15":117.77003484321,"16": 6.0394889663182}.

El equipo de cómputo embebido se encargará de enviar estos tiempos de operación al controlador de luces quien finalmente controlará el cambio de luces en cada semáforo según los tiempos calculados por la estación central.

CONCLUSIONES

Para lograr desarrollar la red de semáforos inteligentes capaz de mejorar el flujo automovilístico en la Ciudad de México, se hizo uso de diversos conceptos y conocimientos en materia de cómputo, como son: procesamiento y reconocimiento digital de imágenes, minería de datos, métodos de inteligencia artificial, bases de datos, ingeniería de software, redes de datos, cómputo embebido, sistemas operativos, lenguajes de programación, arquitectura cliente-servidor, entre otros. Antes de iniciar con el desarrollo del proyecto fue necesario conjuntar todos los conceptos y conocimientos mencionados anteriormente ya que estos son el sustento de todo el proyecto.

En cuanto al reconocimiento de imágenes la tarea de identificación de nivel de tráfico en una vialidad fue bueno, sin embargo, para la implementación del proyecto es necesario realizar una etapa de entrenamiento más robusta que contemple la captura de imágenes en diversas condiciones de iluminación y climáticas que pueden generar problemas en el reconocimiento, los parámetros de filtrado de imágenes deben ser ajustados de acuerdo a las condiciones de cada vialidad para obtener una mejor identificación del nivel de tráfico. Las tareas de procesamiento y reconocimiento de imágenes al realizarse por intersección en cada uno de los equipos de cómputo embebido distribuyen las tareas en general que por su naturaleza son exhaustivas.

Para lograr identificar el nivel de tráfico en cualquier condición del entorno, una mejora al sistema sería que el equipo de cómputo embebido consulte el estado de clima en la zona en la que está trabajando y a partir de ello determinar el árbol de decisión que debe utilizar para su tarea de reconocimiento.

El análisis realizado, arduo y extenso, del modo de operación de los semáforos en distintos tipos de intersecciones permitió determinar cuáles eran los posibles algoritmos de optimización de tráfico que deben existir y la forma en que tienen que operar, ya que cada tipo de intersección presente en las vialidades de la Ciudad de México tiene características particulares que deben ser atendidas por los algoritmos.

Una parte esencial para el desarrollo de la red de semáforos fue la identificación del software y hardware correctos, que se ajusten a las tareas a realizar de manera óptima para obtener un sistema de tiempo real al menor costo posible. Para ello es necesario contar con conocimientos acerca de las tecnologías existentes que permiten realizar las tareas del sistema, así como normas para determinar las condiciones físicas en las que deben operar. Tanto el software como equipo de cómputo embebido son de uso libre, permitiendo destinarlos a cualquier propósito sin tener que adquirir costosas licencias.

Dentro del alcance del proyecto, se hicieron propuestas de diferentes maneras en que se puede implementar la red de datos que comunique a los equipos de cómputo embebido en cada intersección con la estación central. Este análisis debe realizarse previo a la implementación ya que el costo puede ser elevado debido a la permanencia del proyecto, la gran extensión geográfica en la que debe operar y el gran número de equipos que se comunican provocando gran cantidad de tráfico en la red de datos. Incluso cabe

la posibilidad de la existencia de una red metropolitana en la ciudad que pertenezca al estado y pueda atender las necesidades del sistema.

Una vez hecho el análisis para el desarrollo del sistema de optimización de tráfico se procedió a realizar la implementación de los algoritmos de optimización de tráfico y configuración del ambiente. En esta etapa se definió de manera precisa cuáles tareas deben ser realizadas por la estación central y cuales por los equipos de cómputo en cada intersección y las plataformas en las que funcionan. Los sistemas operativos, las herramientas de software utilizadas, las plataformas y las tecnologías, simplificaron en gran medida el trabajo permitiéndose enfocar más en el objetivo del proyecto.

Debido al gran número de componentes que conforman al sistema como son los equipos de cómputo, las cámaras, las luces en los semáforos, las distintas piezas que conforman los algoritmos, la base de datos, la estación central, entre otros. Fue necesario realizar un continuo análisis estructural y de flujo de información debido a que ninguno de estos componentes trabaja de manera independiente y es necesario comprender la forma en que se relacionan con los demás elementos que conforman el sistema.

El reconocimiento de nivel de tráfico a través de sensores de imagen se logró de manera satisfactoria. Para verificar el funcionamiento de esta tarea se realizaron pruebas reales en una vialidad en condiciones ambientales y de iluminación favorables.

Los tiempos obtenidos parecen ser prometedores, sin embargo, para el alcance de este proyecto no es posible realizar una prueba real o simulada debido a la alta complejidad que esto implica. Los resultados reales se obtendrán hasta el momento de implementar el sistema y verificar que realmente se mejore el flujo automovilístico en las vialidades de la ciudad. En caso de no lograrse, en primer lugar, deben ajustarse los parámetros con los que opera el sistema o bien realizar mejoras a los algoritmos de optimización. No obstante, cualquier modificación que se deba realizar se haría editando el software ubicado en la estación central.

La implementación realizada es escalable y su mantenimiento puede ser sencillo aplicando una serie de mejoras como:

- Mejora al algoritmo de optimización mediante sincronización de los semáforos en una vialidad y tomando decisiones con base al contexto. Es necesaria la construcción de una base de datos espacial que permita establecer las relaciones topológicas entre las vialidades de la ciudad con base en la base de datos relacional que se desarrolló en este proyecto.
- Hacer que la estación central cuente con una etapa de aprendizaje donde almacene las decisiones que lograron obtener mejores resultados y discernir para optimizar aún más los tiempos de operación.
- Hacer pública la información, representando en un mapa la información almacenada en la base de datos espacial. Esta información puede ser usada también por una aplicación móvil que acceda a los datos de la estación central y cree rutas óptimas para los usuarios.

- Actualizar el software de los equipos de cómputo embebido, monitorearlos y realizar la etapa de entrenamiento desde un equipo centralizado dedicado a ello, así como realizar acciones como son instrucciones de encendido y apagado.
- Encriptar la información que se transfiere.
- Tomar medidas de ahorro de energía desde el equipo controlador como apagar los semáforos en caso de tráfico muy bajo y ajustar la intensidad de sus luces de acuerdo a luz ambiental.
- Instalar una estación central de respaldo que entre en operación si la estación central principal falla.

La red de semáforos inteligentes pretende mejorar la infraestructura vial existente en la Ciudad de México, permitiendo mejorar la calidad de vida de los ciudadanos y reducir problemas ambientales. Cabe mencionar que hace falta de la participación de los usuarios de las vialidades como no estacionarse en vías principales, que el transporte público sólo haga paradas en lugares establecidos en un tiempo pertinente, no rebasar la línea en caso de visualizar que no es posible avanzar sin estorbar a la vialidad que se cruza…, en pocas palabras, que la ciudadanía tenga una mejor educación vial ya que sin esto el proyecto no tendrá el impacto deseado.

Ciudad Universitaria, Coyoacán.
Ciudad de México a las 12 horas del 9 de junio de 2016.

MMXVI

REFERENCIAS

1. Introducción

Artículos

FIMEVIC (Fideicomiso para el mejoramiento de las vías de comunicación del Distrito Federal). (24 de noviembre de 2015). *Diagnóstico de la movilidad de las personas en la Ciudad de México*. Recuperado de http://www.fimevic.df.gob.mx/problemas/1diagnostico.htm.

Atracción 360. (24 de noviembre de 2015). *Caos vial frena competitividad en el DF*. Recuperado de http://www.excelsior.com.mx/comunidad/2013/05/05/897506.

Lagunas I. (21 de julio de 2014). En el DF se sufre más. *ReporteIndigo*, pp 31.

Hernández M. (24 de abril de 2013). México, 2º país de AL con más muertes por contaminación. *Gráfico*, pp 9.

Facmed. (10 de diciembre de 2015). *Contaminación del aire*. Recuperado de http://www.facmed.unam.mx/deptos/salud/periodico/29contamin/.

Sagols L. (2010). *La ética ambiental frente a la sobrepoblación,* pp. 1-16. Universidad de Salamanca, Salamanca, Castilla y León, España. http://turing.iimas.unam.mx/~cgg/teach/Pamplona/05-SOTL.pdf.

2. Antecedentes

Libros

Dra. Ruiz Abellón M.C. (2015). *Introducción a los Árboles de Decisión*. Departamento de matemática aplicada y estadística, Universidad Politécnica de Cartagena. Cartagena, España.

García García, P.P. (2013). *Reconocimiento de imágenes utilizando redes neuronales artificiales*. Tesis de Fin de Máster en Ingeniería informática para la industria no publicada, Universidad Complutense de Madrid, Madrid, España.

González I., González J., y Gómez Arribas F. (2003). *Hardware libre: Clasificación y desarrollo de hardware reconfigurable en entornos GNU/Linux*. Universidad Autónoma de Madrid, Madrid, España.

Ing. Bisaro M., e Ing. Danizio E. (2015). *Diseño y Configuración de Redes de Computadoras: Redes Ethernet – IEEE 802.3*. Laboratorio de redes, Universidad Tecnológica Nacional.

Ing. Hernández Perales J.A. (2014). *Árbol de decisión*. Instituto de electrónica y computación, Universidad Tecnológica de la Mixteca. Oaxaca, México.

Artículos

AXIS Communications. (16 de febrero de 2016). *Camera elements*. Recuperado de http://www.axis.com/global/es/learning/web-articles/technical-guide-to-network-video/lens-elements.

Carlos Villagómez. (16 de febrero de 2016). *El sensor de imágenes*. Recuperado de http://es.ccm.net/faq/7691-el-sensor-de-imagenes.

Definición.DE. (16 de febrero de 2016). *Definición de cable UTP*. Recuperado de http://definicion.de/cable-utp/.

Galiana Llinares A.N. (17 de febrero de 2016). *Sistemas embebidos,* pp. 1-30. Recuperado de http://server-die.alc.upv.es/asignaturas/PAEEES/2005-06/A07%20-%20Sistemas%20Embebidos.pdf.

Ibersystems Solutions, S.L. (17 de febrero de 2016). *Redes WiMAX*. Recuperado de http://www.ibersystems.es/redes-wimax/.

Ingeniatic. (22 de febrero de 2016). *Sensor de imagen*. Recuperado de http://ingeniatic.euitt.upm.es/index.php/tecnologias/item/586-sensor-de-imagen.

Kioskea. (22 de febrero de 2016). *802.11i / WPA2*. Recuperado de http://es.ccm.net/contents/788-802-11i-wpa2.

Kioskea. (22 de febrero de 2016). *El concepto de red*. Recuperado de http://es.ccm.net/contents/252-el-concepto-de-red.

Kioskea. (22 de febrero de 2016). *Equipos de red - El concentrador*. Recuperado de http://es.ccm.net/contents/292-equipos-de-red-el-concentrador.

Kioskea. (22 de febrero de 2016). *Redes - Concentrador (hub), conmutador (switch) y router*. Recuperado de http://es.ccm.net/faq/656-redes-concentrador-hub-conmutador-switch-y-router#2-el-switch.

Kioskea. (22 de febrero de 2016). *Tipos de redes*. Recuperado de http://es.ccm.net/contents/257-tipos-de-redes.

Kioskea. (22 de febrero de 2016). *Topología de red*. Recuperado de http://es.ccm.net/contents/256-topologia-de-red.

Kioskea. (22 de febrero de 2016). *Transmisión de datos: Cableado*. Recuperado de http://es.ccm.net/contents/685-transmision-de-datos-cableado.

Microsoft. (25 de febrero de 2016). *Datos espaciales (SQL Server)*. Recuperado de https://msdn.microsoft.com/es-es/library/bb933790(v=sql.120).aspx.

Pastor J. (25 de febrero de 2016). *Esta línea de fibra de 255 Tbps es capaz de albergar todo el tráfico de Internet por sí sola*. Recuperado de http://www.xataka.com/otros/esta-linea-de-fibra-de-255-tbps-es-capaz-de-albergar-todo-el-trafico-de-internet-por-si-sola.

Ramcir_cjm. (25 de febrero de 2016). *Cable coaxial*. Recuperado de http://ramcir_cjm.tripod.com/Mvg.htm.

Soperativos.wikispaces. (25 de febrero de 2016). *Sistemas operativos de tiempo real o embebidos*. Recuperado de https://soperativos.wikispaces.com/10.+SISTEMAS+OPERATIVOS+DE+TIEMPO+REAL+O+EMBEBIDOS.

Top fotografía. (25 de febrero de 2016). *Sensor cmos*. Recuperado de http://www.topfotografia.net/Fotografia/teoria-de-la-fotografia/sensor-cmos/sensor-cmos.html.

Torunozeydi. (25 de febrero de 2016). *¿Qué es un árbol de decisión?.* Recuperado de https://unitorunozeydiio.files.wordpress.com/2011/03/quc3a9-es-un-c3a1rbol-de-decisic3b3n.pdf.

Tsusistemas. (25 de febrero de 2016). *Infrarrojo (IRDA)*. Recuperado de http://tsusistemas.tripod.com/julia1.html.

Uservers. (25 de febrero de 2016). *¿Qué es una dirección IP? Información de las direcciones IP, asignación y uso*. Recuperado de http://web.uservers.net/ayuda/soluciones/dominios/que-es-una-direccion-ip_NTk.html.

Usuario huehueh de deralaja. (25 de febrero de 2016). *Ondas y comunicaciones – Satélites, WiMAX, Wi-Fi, Li-Fi y Bluetooth*. Recuperado de https://deralaja.wordpress.com/2015/08/14/ondas-y-comunicaciones-satelites-wimax-wi-fi-li-fi-y-bluetooth/.

Vega Huerta H., Cortez Vásquez A., Huayna A.M., Alarcón Loayza L., y Romero Naupari P. (2009). *Reconocimiento de patrones mediante redes neuronales*. Revista de ingeniería de sistemas e informática, 6(2), 17-26.

Wikipedia. (26 de febrero de 2016). *Router*. Recuperado de https://es.wikipedia.org/wiki/Router.

Wikipedia. (26 de febrero de 2016). *Sistema de tiempo real*. Recuperado de https://es.wikipedia.org/wiki/Sistema_de_tiempo_real.

3. Análisis y diseño del sistema

Libros

Aldás S. (2009). Normas para instalación de semáforos. Ecuador.

Carlos Alberola López (2004). Tratamiento de la información I. ETSI Telecomunicación Campus Miguel Delibes, Valladolid, España.

Furfaro A. (2010). Manejo de bibliotecas OpenCV. Departamento de Computación. Facultad de ciencias exactas y naturales, Universidad de Buenos Aires, Argentina.

García García, P.P. (2013). Reconocimiento de imágenes utilizando redes neuronales artificiales. Tesis de Fin de Máster en Ingeniería informática para la industria no publicada, Universidad Complutense de Madrid, Madrid, España.

Artículos

AskUbuntu. (7 de marzo de 2016). How to install numpy and scipy for python?. Recuperado de http://askubuntu.com/questions/359254/how-to-install-numpy-and-scipy-for-python.

Bolaño Asenjo D., Corpas Martos J.J., Fernández Martínez O.D., Gutiérrez segura D. (7 de marzo de 2016). Momentos. Recuperado de http://slideplayer.es/slide/2346495/.

Debian. (9 de marzo de 2016 2016). Información sobre la versión de Debian “jessie”. Recuperado de https://www.debian.org/releases/stable/.

Construcción e instalación del sistema

Libros

González, R.C., Wintz, P. (1996). Procesamiento digital de imágenes (pp. 89-269). Estados Unidos: Addison-Wesley.

Marcos Martín. (2002). Descriptores de imagen. ETSI Telecomunicación Campus Miguel Delibes, Valladolid, España.

Ministerio de trasportes y telecomunicaciones. (2011). Actualización capítulo 4: Semáforos. Gobierno de Chile, Chile.

SIGMUR. (21 de abril de 2006). Técnicas de filtrado. Universidad de Murcia, Murcia, España.

Valverde Rebaza J. (2007). Detección de bordes mediante el algoritmo de Canny. Universidad Nacional de Trujillo, Trujillo, Perú.

Artículos

Garrón G. (29 de marzo de 2016). CentOS o Debian, para un servidor Web. Recuperado de https://www.garron.me/es/gnu-linux/centos-vs-debian.html.

LinkSprite. (7 de abril de 2016). Instalación de OpenCV en pcDuino. Recuperado de http://cnlearn.linksprite.com/?p=4093#.V0MY1I-cGUn.

OpenCV. (20 de octubre de 2015). Welcome to opencv documentation!. Recuperado de http://docs.opencv.org/2.4/

Secretaría de transportes y vialidad. (21 de abril de 2016). Estadísticas. Recuperado de http://data.semovi.cdmx.gob.mx/wb/stv/estadisticas.html

Stackoverflow. (16 de mayo de 2016). ImportError: numpy.core.multiarray failed to import. Recuperado de http://stackoverflow.com/questions/20518632/importerror-numpy-core-multiarray-failed-to-import.

Stackoverflow. (16 de mayo de 2016). Installing OpenCV for Python on Ubuntu, getting ImportError: No module named cv2.cv. Recuperado de http://stackoverflow.com/questions/25215102/installing-opencv-for-python-on-ubuntu-getting-importerror-no-module-named-cv2.

Unidad operativa de control de tránsito. (18 de mayo de 2016). Especificaciones técnicas para la instalación de semáforos. Recuperado de http://www.uoct.cl/wpcontent/files_mf/esp_tecnicas_instalacion_sem57.pdf.

Wikipedia. (21 de mayo de 2016). Hypertext Transfer Protocol. Recuperado de https://es.wikipedia.org/wiki/Hypertext_Transfer_Protocol.

Wikipedia. (21 de mayo de 2016). Servidor web. Recuperado de https://es.wikipedia.org/wiki/Servidor_web#1.1_Socket_a_direcci.C3.B3n_DNS.