DESARROLLO DE UN SISTEMA INTELIGENTE DE CONTROL DE TRÁFICO CON SOFTWARE DE CÓDIGO ABIERTO EN SISTEMAS EMBEBIDOS

TESIS

PARA OBTENER EL GRADO DE

MAESTRO EN
SISTEMAS INTELIGENTES MULTIMEDIA

PRESENTA

ING. JOSÉ ANDRÉS LAMEGO CASTRO

ZAPOPAN, JALISCO, JUNIO 2017

El Marqués, Querétaro a 18 de Abril de 2017.

Dr. Miguel González Valadez
Director de posgrado

Los abajo firmantes, miembros del Comité Tutorial del alumno Ing. José Andrés Lamego Castro, una vez revisada la Tesis o tesina titulada: " DESARROLLO DE UN SISTEMA INTELIGENTE DE CONTROL DE TRÁFICO CON SOFTWARE DE CÓDIGO ABIERTO EN SISTEMAS EMBEBIDOS", autorizamos que el citado trabajo sea presentado por el alumno para la revisión del mismo con el fin de alcanzar el grado de Maestro en Sistemas Inteligentes Multimedia durante el Examen de Titulación correspondiente.

Y para que así conste se firma la presente a los 18 días de Abril del año 2017.

Dr. Rogelio Álvarez Vargas
Asesor Académico

CIATEQ

23 de mayo de 2017

Respetables miembros del Jurado

Me ha tocado el honor de haber sido designado Revisor del trabajo titulado **"DESARROLLO DE UN SISTEMA INTELIGENTE DE CONTROL DE TRÁFICO CON SOFTWARE DE CÓDIGO ABIERTO EN SISTEMAS EMBEBIDOS"** del Ingeniero **JOSÉ ANDRÉS LAMEGO CASTRO.**

Después de haber leído detalladamente el trabajo que me fue entregado, he tenido la oportunidad de intercambiar información con el sustentante y como resultado de estas acciones he concluido que: El trabajo cumple con los requisitos suficientes para obtener el grado, por lo que no tengo ningún inconveniente en emitir esta carta de aprobación, a fin de que pueda seguir con sus trámites correspondientes para su titulación.

El trabajo tiene los siguientes aspectos positivos:

1.- Excelente correlación entre el desarrollo de sus hipótesis con sus resultados.
2.- En su redacción menciona la fuente de todo trabajo no original.

El trabajo tiene las siguientes oportunidades de mejora:

1.- Que mencione algunos aspectos para realizar un trabajo a futuro.

Haciendo un análisis crítico del trabajo y balanceando lo positivo y las oportunidades de mejora, considero **RECOMENDAR** al Jurado que le otorgue el Grado de Maestro en Sistemas Inteligentes Multimedia, al Ingeniero **JOSÉ ANDRÉS LAMEGO CASTRO,** por lo que acepto se imprima el trabajo de tesis.

No obstante, lo anterior, le solicitaría al sustentante me responda las siguientes preguntas:

En la sección 1.7 menciona que se colocan detectores en la ruta de paso de los vehículos y posteriormente menciona algunos sensores por lo que surge la siguiente pregunta:

1.- ¿Qué detector seleccionaría? Y ¿Por qué?

En la sección 1.9 comenta que, además de lo anterior, el SICT propuesto presenta ventajas adicionales sobre los sistemas tradicionales existentes, por lo que surge la siguiente pregunta:

1.- ¿Cuáles son los sistemas tradicionales existentes?

Le agradecería al Honorable Jurado tenga en consideración la propuesta de otorgar el Grado que pongo a su consideración.

Atentamente

Dr. Nery Delgadillo Checa

I. TITULO

Desarrollo de un Sistema Inteligente de Control de Tráfico con software de código abierto en sistemas embebidos.

II. RESUMEN

En esta investigación se busca desarrollar un Sistema Inteligente de Control de Tráfico (SICT) basado en software de código abierto sobre plataformas de sistemas embebidos. El sistema detectará por medio de cámaras de video a los vehículos y peatones que se desplacen por los caminos vehiculares y aceras que concurran en un crucero para determinar el patrón de cambio de luces de los semáforos en el crucero que logre una combinación de tiempos de espera reducidos, disminución de tráfico ocasionado por cambios de luces ineficientes en horas de alto aforo vehicular y un entorno de paso seguro para los peatones.

El objetivo primario de esta investigación es comprobar la factibilidad técnica de un sistema de esta naturaleza, donde se pone especial énfasis en los conceptos de re-uso de código y librerías, que son la base de la filosofía de la comunidad de software libre. Se propone, además, el objetivo secundario de que la experiencia obtenida en el desarrollo del modelo de trabajo del proyecto sirva como guía para los posibles trabajos de perfeccionamiento de este sistema que también son propuestos como parte de la investigación, además de servir de guía e inspiración para el diseño de otros dispositivos basados en sistemas embebidos que se puedan beneficiar de este modelo de desarrollo.

III. CONTENIDO

IV. LISTA DE FIGURAS

V. GLOSARIO

- **Bit**. Unidad básica de información en computación y en comunicaciones digitales. Puedo tener sólo uno de dos valores y puede, por lo tanto, ser implementado físicamente por medio de un dispositivo de dos estados. Dichos valores son comúnmente representados como 0 o 1.
- **Byte**. Unidad de información digital que usualmente corresponde a ocho bits, debido a que es el número de bits usados para codificar un carácter de texto individual en muchas arquitecturas de computadoras.
- **Indicador**. Señal luminosa, visual o auditiva que informa del estado de derecho de vía. En un semáforo pueden ser luces de colores con o sin gráficos (por ejemplo, flechas o íconos de peatones), mientras que en un paso peatonal pueden ser señales auditivas (por ejemplo, pitidos sucesivos).
- **Peatón**. Persona que transita en la vía pública sin vehículo.
- **Tráfico**. Movimiento o tránsito de personas por cualquier medio de transporte.
- **Vehículo**. Medio de transporte terrestre que funciona a base de motor o cualquier otra forma de propulsión, destinado a la transportación de personas o cosas.
- **Vialidad primaria**. Aquella vía principal para el movimiento de grandes volúmenes de tránsito entre las áreas que forman parte del sistema de red vial en un centro de población.
- **Vialidad secundaria**. Son las que permiten el movimiento del tránsito entre áreas o partes de la ciudad y que dan servicio directo a las vías primarias.
- **Vía rápida**. Vialidad que permite la libre circulación de vehículos con intersecciones a desnivel con otras vías de circulación.

1. INTRODUCCIÓN

1.1. ANTECEDENTES

En las sociedades modernas, el automóvil es un ingrediente primordial en la búsqueda de entender la dinámica de movilidad y la cultura urbana en general. Se trata de un bien material que ha sobrepasado su papel como herramienta de trabajo y medio de transporte, convirtiéndose además en un símbolo de estatus económico, capacidad tecnológica y otras múltiples percepciones ideológicas que en conjunto convierten la posesión de un automóvil en una necesidad mucho mayor que la de sólo servir como transporte de personas o mercancías. Esta demanda, claramente fomentada por los fabricantes de los vehículos, incentiva un mercado que es motor de la economía de las principales naciones del mundo. Por ejemplo, un estimado de 16 millones de vehículos nuevos fue comercializado en el año 2015 tan sólo en los Estados Unidos de América [1].

Como resultado directo del entorno socio-económico mencionado, desde la segunda mitad del siglo XX y hasta nuestros días, las políticas de desarrollo urbano y movilidad de la gran mayoría de las ciudades modernas han estado enfocadas en facilitar el entono legal, financiero y la infraestructura necesarios para el crecimiento del parque vehicular. Desafortunadamente, el incremento en el número de vehículos circulando, al no contar con el correspondiente desarrollo en vías de comunicación, combustibles de baja toxicidad, educación vial, etc., genera efectos adversos de diversa índole, incluyendo polución debida a las emisiones vehiculares y riesgo al circular para peatones y ciclistas entre otras. Dicho incremento en el número de vehículos produce una reducción gradual y progresiva en la velocidad promedio del tránsito hasta llegar en los peores casos a largos periodos de espera y se manifiesta principalmente en los cruceros de 2 o más caminos, debido principalmente a la obvia necesidad de los autos de circular en determinado momento por la misma área de paso. Este fenómeno de aglomeración de vehículos es conocido como congestión o tráfico vehicular. Durante un periodo de congestión vehicular, los automóviles se desplazan a baja velocidad, generalmente bajo condiciones de altas revoluciones de motor, lo que resulta en mayores concentraciones de contaminantes por las emisiones de los vehículos, así como una deteriorada experiencia de viaje para los ocupantes, pérdida de productividad laboral debida a los

mayores tiempos de traslado requeridos y muchos otros efectos perjudiciales tanto para el medio ambiente como para la población en general.

Los sistemas de control de tránsito en cruceros de mayor uso actualmente son los semáforos con indicadores luminosos, cuyos orígenes se remontan a finales del siglo XVIII y que en su versión moderna más usual presentan un cambio de luces automático basado en temporizador programable manualmente mediante un tablero de control único enlazado por cable físico [2]. Existen también versiones de semáforos con cambio de luces sincronizado, principalmente sobre caminos comunes para permitir mayores periodos de movimiento y otras versiones donde el cambio de luces es controlado de manera remota por una unidad de mando central, donde la secuencia de cambio de luces puede ser calculada por computadoras que usan datos como el flujo vehicular estimado en ciertos caminos a determinados horarios. La naturaleza y características del software utilizado para el cálculo de las secuencias de cambios de luces en este tipo de semáforos están dictados por el hardware empleado en la unidad de mando central mencionado, que puede ser un sistema comercial diseñado expresamente para este fin que incluya el software necesario, o puede ser un sistema adaptado a partir de computadoras de uso general.

Debido a las implicaciones de seguridad y afectaciones tanto ambientales como socio-económicas inherentes, los sistemas de control de tráfico están regulados en sus características físicas y de funcionamiento por entidades gubernamentales de algunos países. En los Estados Unidos de Norteamérica, la Asociación Nacional de Fabricantes Eléctricos (NEMA por sus siglas en inglés) define los estándares de conectores y rangos e intervalos de operación [3]. Los estándares definidos por la NEMA sirven como modelo para la regulación en la mayoría de países del hemisferio occidental.

En México la Norma Oficial Mexicana NMX-J-425/1-1981 regula las características de los sistemas de señalización luminosa para tránsito urbano y su capítulo 1 está dedicado a los semáforos [4].

1.2. DEFINICIÓN DEL PROBLEMA

El control de paso en caminos concurrentes es indispensable en un sistema vial urbano para permitir el tránsito seguro de vehículos y peatones. Cuando el sistema vial presenta el fenómeno de alta carga vehicular, ésta condición es agravada por el periodo de espera que deben experimentar los vehículos en determinados cruceros, necesario para permitir el paso alternado en diversas direcciones. Este periodo de espera, cuando está controlado por indicadores luminosos (semáforos), es generalmente ineficiente debido entre otras causas a:

- Mantener detenidos los vehículos en una de las vías mientras no hay vehículos circulando en otra.
- Mantener tiempos de paso similares para caminos con bajo y alto flujo de autos.
- Detener todos los vehículos para permitir el paso de peatones aun cuando no los haya.
- El cambio de las condiciones del tránsito de vehículos y personas durante diferentes horas y días de la semana.

Los semáforos controlados por temporizador convencional presentan un ritmo de cambio de luces que permite un flujo de vehículos y personas adecuado en mayor o menor grado para las condiciones de tránsito que el programador determina como de mayor importancia, por ejemplo, se puede buscar un mayor flujo en uno de los caminos, debido a que generalmente presente un mayor número de autos o personas. Sin embargo, la programación más usual destina un tiempo de paso equivalente para cada uno de los caminos concurrentes en el crucero, con el fin de permitir un promedio de tiempos de espera similar para todos los caminos en el crucero sin necesidad de reprogramar el temporizador cuando las condiciones de carga vehicular cambien.

El uso de los sistemas de control computarizado remoto está menos extendido, debido principalmente a consideraciones del elevado costo económico generado por los requerimientos de infraestructura especializada para la implementación y de personal especializado para su mantenimiento.

La capacidad máxima de desahogo de carga vehicular de un camino está determinada por las características físicas inherentes del mismo y ningún sistema de control de tráfico puede incrementar el flujo vehicular más allá de dicha capacidad, sin

embargo, es necesario optimizar el control para maximizar el aprovechamiento de esa capacidad de desahogo.

Por todo lo anterior, se requiere un sistema de control del cambio de luces de los semáforos que:

- Detecte continuamente la cantidad de los vehículos y peatones circulando en cada sentido de los caminos concurrentes en un crucero y adopte un patrón de cambio de luces adecuado para cada condición de tránsito.
- No mantenga autos o peatones detenidos en un sentido del camino cuando ningún otro auto o peatón esté cruzando en otro sentido o en un camino concurrente.
- Tenga un costo de implementación y mantenimiento competitivo que permita adoptarlo en comunidades de bajos recursos económicos.

1.3. JUSTIFICACIÓN

Para resolver el problema de detección continua de vehículos y peatones en movimiento, así como sus velocidades, en este trabajo de investigación se propone una solución basada en reconocimiento de imágenes empleando las librerías de visión computarizada de código abierto OpenCV [5], desarrollada sobre un sistema operativo Linux para sistemas embebidos desarrollado a su vez con herramientas del proyecto de código abierto para desarrollo de distribuciones Linux conocido como Proyecto Yocto [6].

El empleo del sistema operativo y las librerías de código abierto permitirá contar con herramientas desarrolladas y mantenidas por miles de programadores en todo el mundo, sin incurrir en grandes costos de licenciamiento.

En conjunto, el software de código abierto ofrece a los usuarios del SICT, tanto entidades privadas como instituciones públicas, no solo ventajas económicas inmediatas debidas a su naturaleza inherentemente gratuita, sino que permite que las implementaciones iniciales y el mantenimiento futuro del sistema desarrollado puedan ser realizadas por personal con conocimientos técnicos de dominio generalizado.

1.4. OBJETIVOS

1.4.1. Objetivo general

Diseñar un Sistema Inteligente de Control de Tráfico (SICT) consistente en un dispositivo de control computarizado embebido que emplee 2 cámaras de video para detectar en tiempo real vehículos y peatones desplazándose por 2 caminos vehiculares y aceras que concurran en un crucero y que, con base en la presencia o ausencia de dichos vehículos o peatones, determine el intervalo de tiempo entre cada cambio de luces de los semáforos en dicho crucero que permita optimizar el flujo de autos y peatones mediante una combinación de la minimización de paros de circulación innecesarias, mayores periodos de paso en el camino definido como de mayor prioridad y cambios de luces requeridos por la presencia de peatones.

1.4.2. Objetivos particulares

1. Diseñar un SICT capaz de procesar video procedente de 2 cámaras de video, que controle 2 semáforos convencionales para tráfico de vehículos en un crucero de 2 caminos.
2. Desarrollar una aplicación de software para el SICT que detecte los objetos en movimiento dentro del rango de visión de las cámaras.
3. La aplicación deberá calcular y comandar el patrón de cambio de luces en los semáforos óptimo para minimizar los tiempos de espera de los vehículos y peatones con base en las condiciones de cantidad de vehículos y peatones, o comandar el cambio de luces con base en patrones predeterminados.
4. La aplicación podrá cambiar a un patrón de cambio de luces por intervalos de tiempo predeterminados en caso de falla o configuración manual.

1.5. HIPÓTESIS

1. Es posible utilizar algoritmos de reconocimiento de imagen con software de código abierto para el desarrollo de un Sistema Inteligente de Control de Tráfico (SICT) basado en plataformas de sistemas embebidos con microprocesadores de arquitectura x86 de Intel, que funcione controlando los semáforos en un crucero de manera adecuada a las condiciones de tránsito detectadas de manera continua.
2. El uso del SICT propuesto permitirá minimizar el tiempo promedio de espera de los vehículos y peatones para pasar por el crucero regulado por semáforos bajo condiciones variables de flujo de vehículos y peatones.

2. MARCO TEÓRICO

2.1. TEORÍA GENERAL DEL FUNCIONAMIENTO DE LOS SEMÁFOROS

2.1.1. Implementación actual de semáforos para control de tráfico vehicular

Existen intersecciones en calles y avenidas donde se justifica la instalación de un semáforo para control de tráfico vehicular por medio de un estudio de condiciones de la intersección o crucero. La obligatoriedad y características de este estudio están determinados por la legislación vigente en cada sitio. Las causales más comunes mostradas en dicho estudio son, entre otras [7]:

- Volumen vehicular elevado durante 8 horas diarias.
- Volumen vehicular elevado durante 4 horas diarias.
- Volumen vehicular máximo en algún momento del día.
- Volumen de tránsito peatonal.
- Cruce escolar.
- Punto intermedio en un sistema coordinado de señales.
- Incidencia de accidentes.
- Intersección próxima a un crucero ferroviario.

En el estudio de condiciones del crucero puede también estar incluida una evaluación para determinar el tipo de control más adecuado para el semáforo siendo los más usuales los siguientes [7]:

2.1.1.1. Control manual.

Este tipo de control depende completamente de la manipulación directa de un operario humano para realizar los cambios de señalización. Usualmente todos los tipos de semáforos incluyen la capacidad de deshabilitar un modo automático pre-determinado y elegir el modo manual.

2.1.1.2. Control automático por tiempo predeterminado.

Este tipo de control se basa en ciclos de cambio de modo de señalización alternados secuencialmente usando un temporizador. No toma en cuenta el volumen de tráfico ni las condiciones ambientales.

Al usar este tipo de control, el semáforo otorga el derecho de paso en una intersección o crucero de acuerdo a un patrón de cambio pre-determinado. La secuencia de derecho de paso (fases) y la duración del intervalo entre cada cambio de señal es fijo, y esta duración puede ser elegida con base en patrones de tráfico históricos para el tipo de camino en cuestión (vialidad primaria, secundaria, etc.), por flujo vehicular esperado o de manera arbitraria en la localidad en que se ubique el semáforo.

2.1.1.2.1. Ventajas del control automático por tiempo pre-determinado

- La inherente simplicidad del equipo requiere procedimientos de servicio y mantenimiento relativamente sencillos.
- Se pueden coordinar varios semáforos para proveer un flujo continuo de vehículos a cierta velocidad por determinada ruta, permitiendo un control positivo de velocidad sobre dicha ruta.
- Los intervalos de tiempo pueden ser fácilmente ajustados en el sitio de implementación.
- Bajo ciertas condiciones puede ser programado para afrontar condiciones extremas de tráfico (horas pico).

2.1.1.2.2. Desventajas del control automático por tiempo pre-determinado

- No toma en cuenta o reacciona a fluctuaciones de corto plazo en el tráfico.
- Puede causar retrasos excesivos a peatones y vehículos durante periodos de tráfico ligero.

2.1.1.3. Control automático por actuadores o sensores.

En este tipo de control los cambios de señalización toman en cuenta el volumen de tráfico vehicular y/o peatonal al recibir señales de sensores de presencia manuales y/o automáticos. Generalmente se basa en un modo básico de cambio por tiempo determinado que puede ser modificado en tiempo real de acuerdo a las señales de los sensores de presencia o tráfico (tipo *semi-accionado*), aunque puede funcionar enteramente con base en las señales del conjunto de sensores y actuadores,

adecuando en tiempo real todas las funciones del sistema (tipo *de accionado completo*).

El rango completo de capacidades de control depende del tipo de equipo empleado y de los requerimientos operacionales.

2.1.1.3.1. Ventajas del control automático por actuadores o sensores.

- Usualmente reduce los retrasos, cuando el ajuste de patrón de cambios es adecuado.
- Adaptable a fluctuaciones en el corto plazo en el flujo del tráfico.
- Puede maximizar la capacidad de flujo de la vialidad al ajustar continuamente la duración de los periodos de derecho de paso (tiempo con luz verde).
- Puede permitir modos de paso continuo bajo condiciones de tráfico escaso, al cambiar a un modo de luz amarilla intermitente con el fin de evitar retrasos.
- Especialmente efectivo en intersecciones de fases múltiples.

2.1.1.3.2. Desventajas del control automático por actuadores o sensores

- El costo de este tipo de instalación es sustancialmente mayor que el de una instalación de tiempo pre-determinado.
- Los controladores y sensores son mucho más complejos que los controladores por tiempo pre-determinado, incrementando los requerimientos y el costo de la capacitación para el personal de mantenimiento y servicio.
- La instalación de los sensores y detectores es costosa y se requiere además un adecuado mantenimiento para asegurar una operación adecuada.

2.1.2. Teoría de señalización de los semáforos

2.1.2.1. Fase del semáforo

Una fase de señal de tráfico, o división, es la parte del ciclo dado a un movimiento individual, o a una combinación de movimientos no-conflictivos durante uno o más intervalos. Puede entenderse como un sentido o ruta de una vialidad, o la combinación

de sentidos que no interfieren entre sí. En la figura 1, se muestran las dos fases típicas de un crucero de 2 vialidades de 2 sentidos c/u (7):

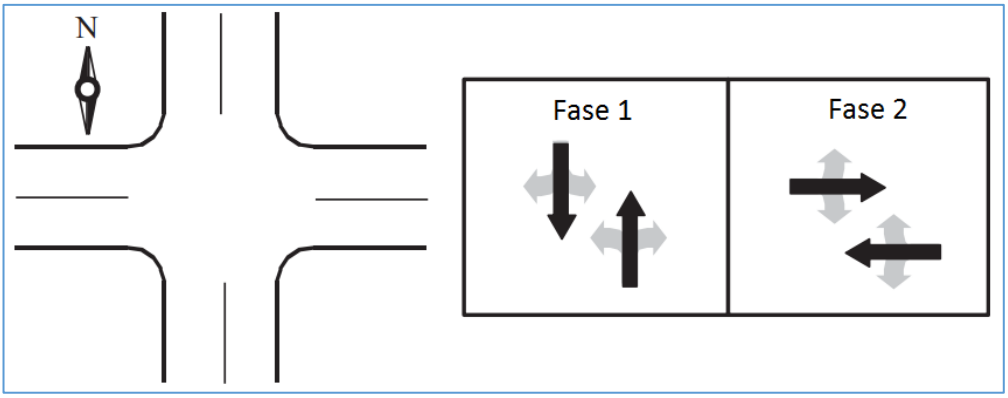

Figura 1: Fases típicas de un crucero de 2 vialidades de 2 sentidos c/u

En una fase del semáforo, los indicadores (luces de color o indicadores gráficos) se mantienen sin cambio. El orden pre-determinado de las fases es la secuencia de operación del semáforo. Este orden es fijo y se programa en el controlador por tiempo pre-determinado o en el controlador por actuadores, según sea el caso.

En la figura 2, se muestran ocho intervalos de cambio de los indicadores. Nótese que los intervalos 4 y 8 incluyen únicamente periodos en rojo. La suma de la división de fase 1 más la división de fase 2 es la duración del ciclo [7]:

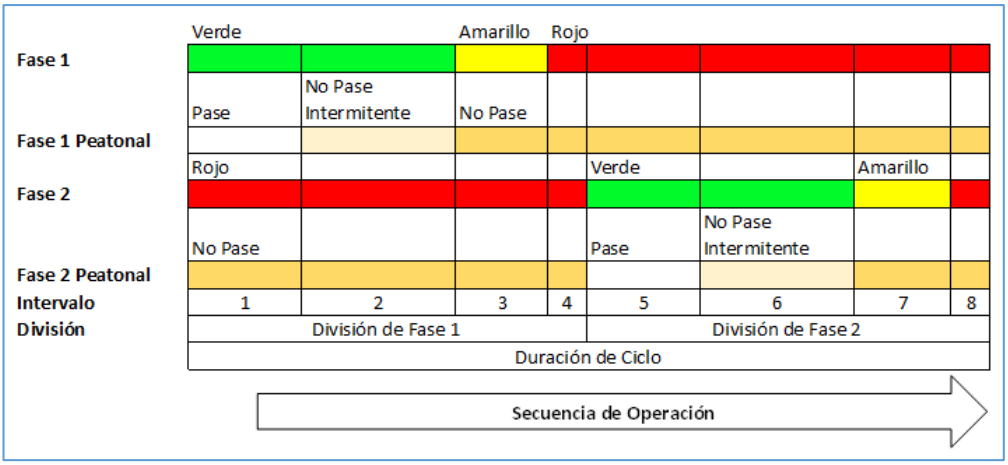

Figura 2: Intervalos de cambio de indicadores

2.1.2.2. Intervalo mínimo en Verde

Un intervalo mínimo en verde es la duración en tiempo mínima en que una fase se encontrará con los indicadores en Verde o Pase. Este intervalo debe ser tal que permita que lo vehículos detenidos (o peatones, en su caso) puedan ponerse en movimiento y

cruzar parcialmente la intersección antes de que se presente el intervalo Amarillo o preventivo. Debe permitir también que los vehículos que se aproximan sean detectados por los sensores, en el caso de sistemas con sensores o actuadores.

2.1.2.3. Intervalo máximo en Verde

Establece el límite máximo al que el intervalo en Verde puede extenderse en una fase. Usualmente existen diversos límites máximos, para responder a condiciones de presencia de tráfico en una fase que se encuentra en Rojo, por una solicitud manual de cambio de fase, o por un límite de tiempo pre-establecido.

2.1.2.4. Solicitudes y Llamadas

La señal de un sensor del sistema se conoce como llamada vehicular o peatonal, dependiendo del tipo de sensor que la origina, y puede significar para el controlador una solicitud de algún servicio o cambio de modo. En ausencia de una llamada nueva, una unidad de control de semáforo permanecerá en la fase de servicio en que se encuentre. Una llamada puede forzar al controlador a cambiar a alguna fase en particular.

Solicitud a Mínimo. En ausencia de cualquier llamada vehicular, esta solicitud se activa y provoca que el controlador implemente el intervalo de tiempo mínimo para la fase actual.

Solicitud a Máximo. En ausencia de llamadas en cualquiera de las fases contrarias, las llamadas en la fase actual pueden provocar que el controlador implemente el intervalo de tiempo máximo para la fase actual.

Solicitud Peatonal. La llamada peatonal produce esta solicitud al controlador, para proporcionar un cambio a fase Verde o Pase.

2.2. INVESTIGACIONES O DESARROLLOS TECNOLÓGICOS MÁS RECIENTES.

2.2.1. Detección de vehículos

Una de las ventajas de semáforos por actuadores o sensores es la capacidad de ajustar los parámetros de tiempo con base en la cantidad real de tráfico vehicular. Debido a que la cantidad de tráfico varía durante las horas y días de la semana, se colocan detectores en la ruta de paso de los vehículos.

En este tipo de control de semáforo, la operación del semáforo depende en gran medida de la correcta operación de los sensores.

2.2.1.1. Detección de presencia

Un detector de presencia vehicular tiene la capacidad de detectar que un vehículo, ya sea estático o en movimiento, ha ingresado a su área de detección. La señal de detección genera una solicitud en el sistema, para cambiar a una fase Verde, o para extender el intervalo de ésta. Las solicitudes generadas se pueden configurar en alguno de los siguientes modos, o en combinaciones de ellos:

- **Solicitud y Extensión**. Los vehículos pueden generar una solicitud de derecho de paso en cualquier momento.
- **Sólo Solicitud**. Los vehículos generan solicitudes de paso sólo durante la fase en Rojo.
- **Solicitud con Retraso**. Los vehículos generan solicitudes de paso sólo después de transcurrido un tiempo de retraso pre-determinado.
- **Solicitud con Retraso y Extensión Inmediata**. Los vehículos generan una solicitud de paso sólo después de transcurrido un tiempo de retraso pre-determinado, EXCEPTO cuando la solicitud proviene de un detector ubicado en la fase en Verde. En este caso, la solicitud se genera de inmediato.

2.2.1.2. Detección por pulsos

La detección en modo de pulsos es un modo de operación en la que el detector produce como señal un pulso corto al ocurrir una detección. Puede configurarse en alguno de los siguientes modos:

- **Sólo Extensión**. Los vehículos pueden generar una solicitud de paso sólo durante la fase en Verde.
- **Muestreo**. Los vehículos pueden generar una solicitud de paso en cualquier momento, pero ésta se considera sólo para efectos estadísticos, no para el control de tráfico.

2.2.1.3. Detección Preferente de Vehículos de Emergencia (PVE)

Un detector PVE es un dispositivo que solicita del controlador de semáforo el paso preferencial en la fase en que ha detectado un vehículo de emergencia con los códigos de emergencia activados. Los dispositivos PVE más usuales son de los siguientes tipos:

- **Óptico**. Detecta pulsaciones de luces estroboscópicas a frecuencias específicas. Puede detectar información codificada en la pulsación luminosa.
- **Sónico**. Detecta las sirenas de vehículos de emergencia aproximándose por medio de micrófonos direccionales.

La detección PVE tiene prioridad sobre la operación normal del controlador de semáforo.

2.2.2. Tipos de sensores de presencia vehicular

2.2.2.1. Inductivo

El sensor inductivo es el tipo de detector más común. Consta de una bobina de alambre embebida en el asfalto o pavimento por la que pasa una pequeña corriente eléctrica. Cuando un cuerpo metálico de suficiente masa pasa sobre la bobina, un sensor detecta un cambio en la inductancia del conjunto y se genera la señal de presencia. La velocidad del objeto se determina entonces al medir la diferencia entre las señales desde el sensor de entrada y el de salida de cada nodo o conjunto de medición. Al encontrarse el sensor relativamente protegido de los elementos, este tipo de sensor tiene la ventaja de requerir poco mantenimiento y ofrecer un bajo índice de fallas por desgaste, sin embargo su principal desventaja es que se requiere un conjunto de medición individual por carril de paso y se pueden reportar lecturas erróneas para vehículos que no circulan por la línea de paso esperada. La figura 3 muestra un ejemplo del conductor inductivo y su colocación.

Figura 3: Sensor de conductor inductivo

2.2.2.2. Magnetómetro

Un magnetómetro mide la diferencia en el nivel del campo magnético terrestre al verse afectado por el paso de un vehículo sobre el sensor. Este tipo de sensor tiene la ventaja de encontrarse embebido en el camino, lo que ofrece cierto grado de protección de los elementos. Sin embrago una desventaja de los sistemas de detección por magnetómetro es que requiere un sistema de lectura y procesamiento de señal más complejo que otros, lo que incrementa su costo de implementación. La figura 4 muestra un ejemplo del sensor de magnetómetro y su colocación en asfalto.

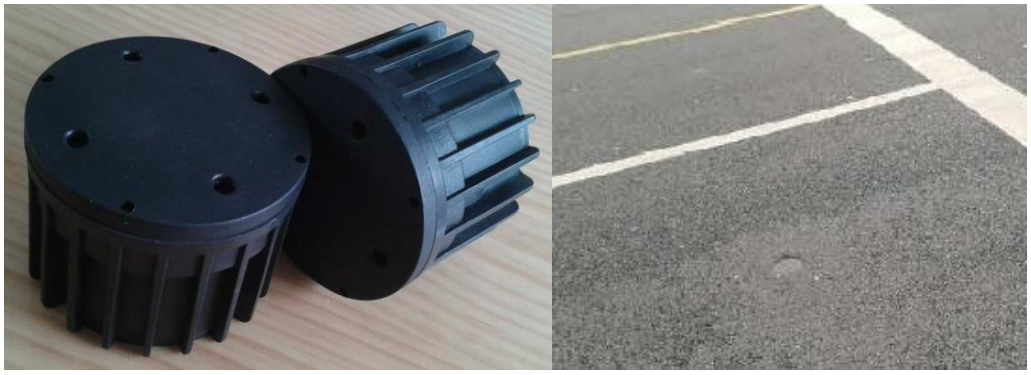

Figura 4: Sensor de magnetómetro

2.2.2.3. Microondas

Un detector de radar de microondas detecta el paso de un vehículo a través del campo de energía de microondas que el mismo detector genera. La ventaja de este sistema es la posibilidad de ser implementado sin mayores afectaciones en los caminos existentes, ya que los sensores pueden ser colocados en postes existentes a lo largo del camino, sin

embargo, la principal desventaja es su mayor costo con respecto a otros sistemas. La figura 5 muestra un ejemplo de un sensor vehicular por microondas.

Figura 5: Sensor vehicular por microondas

2.2.2.4. Ultrasónico

Similar al radar de microondas, el detector ultrasónico registra el paso de un vehículo a través del campo ultrasónico que el mismo detector genera. Al igual que el de microondas, este sistema tiene la ventaja de poder ser colocado en postes existentes a lo largo del camino, sin trabajos adicionales en el camino mismo, pero también tiene la desventaja de requerir un sistema de control relativamente más complejo y que puede ser más sensible a los elementos ambientales. La figura 6 muestra un ejemplo de un sensor vehicular ultrasónico de la marca Sumitomo Electric [8].

Figura 6: Sensor vehicular ultrasónico

2.2.2.5. Video

Un detector de video detecta una imagen reconocible como un vehículo, o el movimiento de objetos que pueden ser catalogados como vehículos de acuerdo a su tamaño, velocidad, ubicación o una combinación de las anteriores. Los sistemas de detección de video emplean algoritmos de procesamiento de imágenes que

generalmente combinan la detección de objetos en movimiento así como la presencia de objetos nuevos en el campo visual. Los sistemas más avanzados pueden incluso ofrecer la detección de objetos específicos e incluso detección visual. Evidentemente, la mayor necesidad de poder de cómputo requerido para estas prestaciones significa que estos sistemas pueden tener un costo elevado, lo que puede ser un inconveniente mayor para su implementación en sistemas de caminos con múltiples cruceros. La figura 7 muestra un ejemplo de un detector vehicular de video colocado sobre un semáforo y una simulación de la imagen procesada que puede presentar.

Figura 7: Detector vehicular de video

2.3. TEORÍA DE PROCESAMIENTO DE IMÁGENES Y VIDEO DIGITAL.

2.3.1. El modelo de color RGB

RGB (por las siglas en inglés de los colores primarios rojo, verde y azul) y su subconjunto CMY (de las siglas en inglés de cian, magenta y amarillo), forman el modelo aditivo de color más básico y el cual es conocido desde hace mucho tiempo y que ha sido empleado en campos como el estudio de la percepción humana del color, impresión de documentos e imágenes y en la fotografía y el cine.

Este modelo de color está directamente relacionado con la fisiología del ojo humano, que presenta tres tipos principales de receptores de luz (las células conos) y que son particularmente sensibles a un rango específico de color. La combinación de las señales de los conos forma en el cerebro una representación del color al que se visualiza. Esta es la base de la teoría de interpretación del color Young-Helmholtz, propuesta por Thomas Young y Hermann Helmholtz en el siglo XIX [9].

2.3.2. Procesamiento de imágenes digitales

La posibilidad de representar múltiples colores en el cerebro humano a partir de los colores primarios mencionados ha sido empleada también en el procesamiento de imágenes digitales. Una imagen puede ser representada en dos dimensiones (2D) por medio de una matriz de puntos conteniendo cada uno la información del nivel de intensidad en que cada color primario está presente (representación numérica) en dicho punto. A la concentración de puntos por unidad de área de imagen se conoce como resolución. Cada uno de esos puntos a su vez puede estar representado con una cantidad preestablecida de información en bits, siendo los más usuales 8, 16, 24 y 32 bits, de modo que, por ejemplo, en un espacio de color RGB de 24 bits cada punto de una imagen estará representado por 24 bits de datos

Es posible tratar las matrices de puntos como matrices matemáticas y realizar operaciones de álgebra lineal con ellas, como adición, multiplicación escalar y transposición, entre otras. Al reinterpretar cada matriz como una imagen en 2D se puede observar que cada operación sobre la matriz original produce una imagen resultante con un cambio específico y repetible. De este modo es como se pueden desarrollar algoritmos (o funciones) para modificar una imagen representada por una matriz de puntos RGB.

2.3.3. Funciones de procesamiento de imagen digital

2.3.3.1. Cambio de espacio de color.

Esta función convierte una imagen de un espacio de color a otro. Para el caso de una imagen del espacio RGB representada con 24 bits de datos, los primeros 8 bits de información de cada punto corresponderán a la intensidad del componente rojo, los siguientes al componente verde y los últimos ocho al componente azul de la imagen, así que un cambio de espacio de color puede realizarse por medio de una función que realice una operación de producto entre la matriz de datos de la imagen original y la matriz de datos que defina la equivalencia de valores entre el espacio de origen y el de destino. Estas matrices de equivalencias se encuentran definidas en todas las aplicaciones de procesamiento de imágenes [10]. Por ejemplo, para cambiar de un espacio RGB a uno Gray (monocromático), se realiza:

$$RGB[A]\ a\ Gray: Y \leftarrow 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B$$

Donde A es el arreglo de valores correspondientes a la imagen en 2D. R, G y B son los componentes de cada color en el espacio RGB y Y es el componente de intensidad de luminosidad en el espacio monocromático Gray.

2.3.3.2. Difuminación Gaussiana

Esta función difumina una imagen aplicando un filtro gaussiano, y consiste en promediar cada valor de la matriz de datos de la imagen original con los valores de los puntos vecinos usando un peso o grado de importancia determinado (kernel Gaussiano). Se debe utilizar un procedimiento adicional de extrapolación para procesar los datos correspondientes a los bordes de la imagen, donde no existen puntos vecinos [10].

2.3.3.3. Umbral o delimitado (*Threshold*)

Esta función se usa para obtener una imagen binaria a partir de una imagen de escala de grises o también puede usarse para reducir "ruido", es decir, filtrar valores demasiado pequeños o demasiado grandes de una matriz de datos de imagen. Existen varios tipos de delimitado:

- **Binario**. Si el valor del punto es mayor que un valor pre-definido (*thresh*), se cambiará a un valor determinado (*maxval*), de otro modo será cero:

$$dst(x,y) = \begin{cases} maxval\ if\ src(x,y) > thresh \\ 0\ otherwise \end{cases}$$

- **Binario invertido**. Si el valor del punto es mayor que un valor pre-definido (*thresh*), se cambiará a cero, de otro modo será cambiado a un valor determinado (*maxval*):

$$dst(x,y) = \begin{cases} 0\ if\ src(x,y) > thresh \\ maxval\ otherwise \end{cases}$$

- **Truncado**. Si el valor del punto es mayor que un valor pre-definido (*thresh*), se cambiará a un valor determinado (*threshold*), de otro modo permanecerá sin cambio:

$$dst(x,y) = \begin{cases} threshold\ if\ src(x,y) > thresh \\ src(x,y)\ otherwise \end{cases}$$

- **A cero**. Si el valor del punto es mayor que un valor pre-definido (*thresh*), permanecerá sin cambio. De otro modo, se cambiará a cero:

$$dst(x,y) = \begin{cases} src(x,y)\ if\ src(x,y) > thresh \\ 0\ otherwise \end{cases}$$

- **A cero invertido**. Si el valor del punto es mayor que un valor pre-definido (*thresh*) se cambiará a cero. De otro modo, permanecerá sin cambio:

$$dst(x,y) = \begin{cases} 0 \; if \; src(x,y) > thresh \\ src(x,y) otherwise \end{cases}$$

2.3.3.4. Dilatación morfológica

Esta función dilata una imagen usando un elemento estructural específico (*element*). El incremento en dimensión obtenido es útil para rellenar hoyos dentro de la región:

$$dst(x,y) = \max_{(x',y'):element(x',y')\neq 0} src(x+x', y+y')$$

2.3.3.5. Detección de contornos

Esta función encuentra los contornos de una imagen binaria por alguno de los siguientes modos de detección:

- **Externa**. Detecta únicamente los contornos exteriores, asignando una jerarquía negativa para todos los niveles de contornos interiores.
- **Lista**. Detecta todos los contornos sin establecer ninguna relación jerárquica.
- **Borde compuesto**. Detecta todos los contornos y los organiza en una jerarquía bi-nivel. Al nivel superior, están los bordes exteriores de los componentes. Al segundo nivel, están los bordes de los agujeros.
- **Árbol**. Detecta todos los bordes y construye una jerarquía de bordes anidados.

Se emplea, además, alguno de los siguientes métodos de aproximación de contorno:

- **Nulo**. Se almacenan absolutamente todos los puntos del contorno, es decir, para cualquier par de puntos subsecuentes del borde, estos serán vecinos horizontales, verticales o diagonales.
- **Simple**. Comprime los segmentos horizontales, diagonales y verticales, dejando únicamente sus puntos finales. Por ejemplo, un contorno cuadrangular se codifica con 4 puntos.
- **Compensado**. Compensación opcional por desplazamiento del punto.

2.3.3.6. Área de contorno

Esta función calcula el área dentro de un contorno por medio de la integración en sentido anti-horario del perímetro de la figura (*teorema de Green*) [11].

2.3.3.7. Contorno rectangular

Esta función calcula el contorno rectangular vertical mínimo que contiene todos los puntos proporcionados como parámetros de entrada.

2.3.3.8. Rectángulo

Esta función dibuja un contorno rectangular o un rectángulo relleno en la imagen destino usando dos puntos proporcionados como vértices opuestos de su perímetro.

2.3.3.9. Texto

Esta función dibuja una cadena de texto en la imagen destino.

2.3.3.10. Remodelar

Esta función cambia la forma de una matriz de datos. La matriz destino tendrá los mismos datos que la original, pero diferente número de canales, de filas, o de ambos.

2.3.3.11. Arreglo

Esta función crea una matriz de datos a partir de un objeto compatible con el formato.

2.3.4. Detección de movimiento usando el método de la substracción del fondo.

La detección de movimiento en videos digitales se enfoca principalmente en detectar movimiento de ciertos objetos del mundo real, tales como autos, peatones o ciclistas en videos provenientes de archivos o de cámaras [12]. Existen diversos métodos de detección, siendo uno de los más usuales el de substracción del fondo. Este método consiste básicamente en los siguientes pasos:

- Un video se segmenta en escenas o cuadros de imagen fija.
- Se comparan dos escenas sucesivas para determinar los cambios relativos.
- Los puntos que no presentan ningún cambio entre escenas se consideran parte del fondo y no son tomados en cuenta en los cálculos siguientes. Los puntos que presentan algún cambio se consideran de interés y son agrupados de acuerdo a su adyacencia a otros puntos de interés.

- Si el conjunto de puntos de interés es menor a un tamaño pre-determinado, se ignora para los siguientes cálculos. Si el conjunto es mayor al tamaño pre-determinado, se considera un objeto en movimiento.

2.3.5. Librería OpenCV

OpenCV es una librería de visión computarizada y aprendizaje de máquina de código abierto de amplio uso tanto en la industria como en la academia, ya que cuenta con una licencia tipo Berkeley Software Distribution (BSD) que permite el uso con fines comerciales [13].

Esta librería tiene más de 2500 algoritmos optimizados, que incluyen tanto un exhaustivo conjunto de algoritmos clásicos como de algoritmos de vanguardia para visión computarizada y aprendizaje de máquina. Estos algoritmos pueden usarse para detectar y reconocer caras, identificar objetos, clasificar acciones humanas en videos, seguir movimientos de cámara, seguir objetos en movimiento, extraer modelos tridimensionales de objetos, producir nubes de puntos tridimensionales desde cámaras estereoscópicas, combinar imágenes para producir imágenes de alta resolución de escenas enteras, encontrar imágenes similares en una base datos de imagen, remover ojos rojos de fotografías tomadas con flash, seguir el movimiento de los ojos, reconocer escenarios y establecer marcadores para superponerlos como realidad aumentada, etc. OpenCV tiene una comunidad de usuarios de más de 47'000 personas y un estimado de más de 7 millones de descargas.

Junto con reconocidas compañías como Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda o Toyota que emplean la librería, existen muchas compañías de reciente creación que también hacen un uso extensivo de ella, como Applied Minds, VideoSurf o Zeitera. El rango de aplicaciones de OpenCV comprende desde la combinación de imágenes de calles, detección de intrusos en videos de vigilancia en Israel, monitorización de equipo minero en China, ayudar a robots a navegar y recoger objetos en Willow Garage, detectar accidentes de ahogamiento en piscinas en Europa, reproducir arte interactivo en España y Nueva York, buscar desechos en autopistas en Turquía, inspeccionar etiquetas de productos alrededor del mundo, hasta la detección de rostros en Japón.

Tiene interfaces en C++, C, Python, Java y Matlab y soporta los sistemas operativos Windows, Linux, Android y MacOS. OpenCV se enfoca principalmente en aplicaciones

de visión en tiempo real y aprovecha las instrucciones de procesador MMX y SSE cuando están disponibles [14].

2.4. RELACIÓN DEL SICT CON LOS DISPOSITIVOS DE CONTROL EXISTENTES.

El SICT está concebido como un perfeccionamiento del método de control automático de semáforos por tiempo determinado (detallado en el capítulo 2.1.1.2 de este documento), combinado con las ventajas económicas y operacionales de los sistemas embebidos que utilizan software de código abierto.

El SICT busca cumplir con los requerimientos básicos de un sistema de control de semáforo por video, principalmente:

- Detección de vehículos en movimiento en el camino monitorizado.
- Priorización de derecho de paso a la vialidad con mayor número de vehículos circulando.
- Capacidad de contar con diversos modos de operación con base en horario de operación, condiciones de tráfico o de emergencia.

Además de lo anterior, el SICT propuesto presenta ventajas adicionales sobre el sistema de control por tiempo determinado mencionado, entre otras:

- Elevada adaptabilidad a recursos disponibles, ya que pueden emplearse instalaciones existentes tanto para la colocación física como para la alimentación de potencia, ya que los sistemas embebidos propuestos tienen amplios rangos de operación, reducido tamaño y son fácilmente configurables para diversos tipos de sensores, luminarias, localización física, etc.
- Reducido costo, estimado en un 10% del costo de un controlador automático por tiempo pre-determinado.
- Posibilidad de incrementar o modificar las características de operación a través de la reprogramación del software.
- Facilidad de mantenimiento y servicio debido al uso de una plataforma de software de uso extendido entre la comunidad de desarrolladores.

El concepto de desarrollo básico propuesto es el siguiente:

- Contar con una plataforma de hardware que funcione con la distribución de Linux Poky del Proyecto Yocto.
- La plataforma de hardware contará con las conexiones de alimentación de potencia necesarias, así como salidas para controlar las luminarias del o los

semáforos controlados. El control de las luminarias no incluirá alimentación de potencia a las luminarias, que contarán con un circuito independiente para este propósito, aislado de la alimentación del SICT.

- Se podrá conectar al sistema SICT una o más cámaras de video compatibles con el protocolo UVC [15].
- El SICT contará con un modo de operación básico que emule al de un sistema de control por tiempo predeterminado.
- El SICT contará con un modo de operación avanzado donde analizará el video proveniente de las cámaras conectadas y realizará ajustes en tiempo real para minimizar los tiempos de espera innecesarios.
- El procesamiento del video y el control del semáforo se llevarán a cabo utilizando un programa principal en lenguaje Python que incluya además la librería OpenCV, considerando el método de detección por substracción del fondo.
- El SICT tendrá opcionalmente un medio de transmisión remota de información con fines de análisis, almacenamiento o diagnóstico.

3. PROCEDIMIENTO DE INVESTIGACIÓN

3.1. DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO FUNCIONAL DE SICT

Con el objetivo primordial de verificar de forma empírica la validez de la hipótesis planteada al inicio de este proyecto, a saber:

- Es posible utilizar algoritmos de reconocimiento de imagen con software de código abierto para el desarrollo de un Sistema Inteligente de Control de Tráfico (SICT) basado en plataformas de sistemas embebidos que funcione controlando los semáforos en un crucero de manera adecuada a las condiciones de tránsito detectadas de manera continua.
- El uso del SICT propuesto permitirá minimizar el tiempo promedio de espera de los vehículos y peatones para pasar por el crucero regulado por semáforos bajo condiciones variables de flujo de vehículos y peatones.

Se procedió a diseñar e implementar un prototipo funcional de un SICT. El requisito principal para este prototipo es que debe contar con la totalidad de funcionalidad y capacidad descritas para el SICT, dejando como opcionales sólo las características físicas definidas por la normatividad aplicable para semáforos públicos.

3.1.1. Diseño de la estructura física

Tomando como principal factor de decisión el costo y la disponibilidad de materiales de fabricación, se eligió implementar el prototipo con una escala aproximada de 1:4 con respecto a un semáforo convencional actual, empleando una estructura de alambre de acero con forma de prisma rectangular de 45 cm de alto, 20 cm de ancho y 20 cm de fondo. Las seis luminarias empleadas (2 rojas, 2 verdes y 2 amarillas) tienen 6 cm de diámetro y funcionan con focos automotrices incandescentes de 12 Volts. La estructura soporta además de las seis luminarias, a la tarjeta de desarrollo Edison, un repetidor de puertos USB, dos cámaras web marca Logitech modelo C170, una antena WiFi, una placa de relevadores electromagnéticos y una fuente de poder dual de 12-5 Volts.
La estructura física final se muestra en la figura 8:

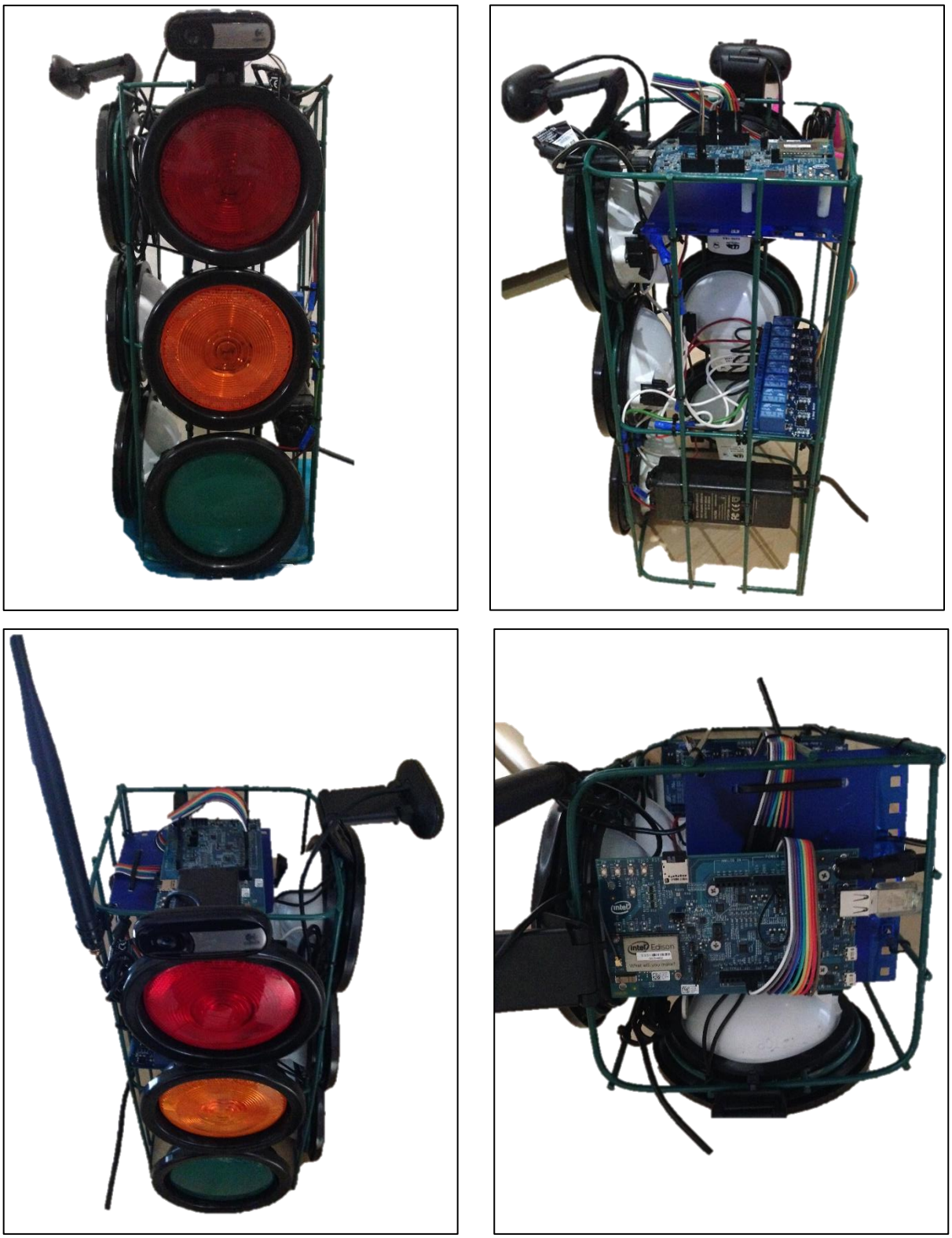

Figura 8: Estructura física del prototipo de SICT

3.1.2. Circuito eléctrico.

La plataforma de desarrollo Intel Edison funciona con una alimentación de 7 a 12 Volts con una potencia máxima de aproximadamente 1 Watt, haciendo inviable la alimentación directa de voltaje para las luminarias del SICT. Por esto, se emplea un sistema de relevadores electromecánicos que permiten controlar los actuadores con la señal de bajo amperaje de la tarjeta Edison, al tiempo que se manejan luminarias de 18 Watts conectadas a un circuito externo de 12 Volts con su propia fuente de poder. Es de hacer notar que este diseño de circuitos separados permite emplear luminarias de corriente alterna hasta de 220 Volts con una fuente de alimentación adecuada, con base en la especificación de los relevadores. El diagrama eléctrico básico se muestra en la figura 9:

3.1.2.1. Diagrama eléctrico básico

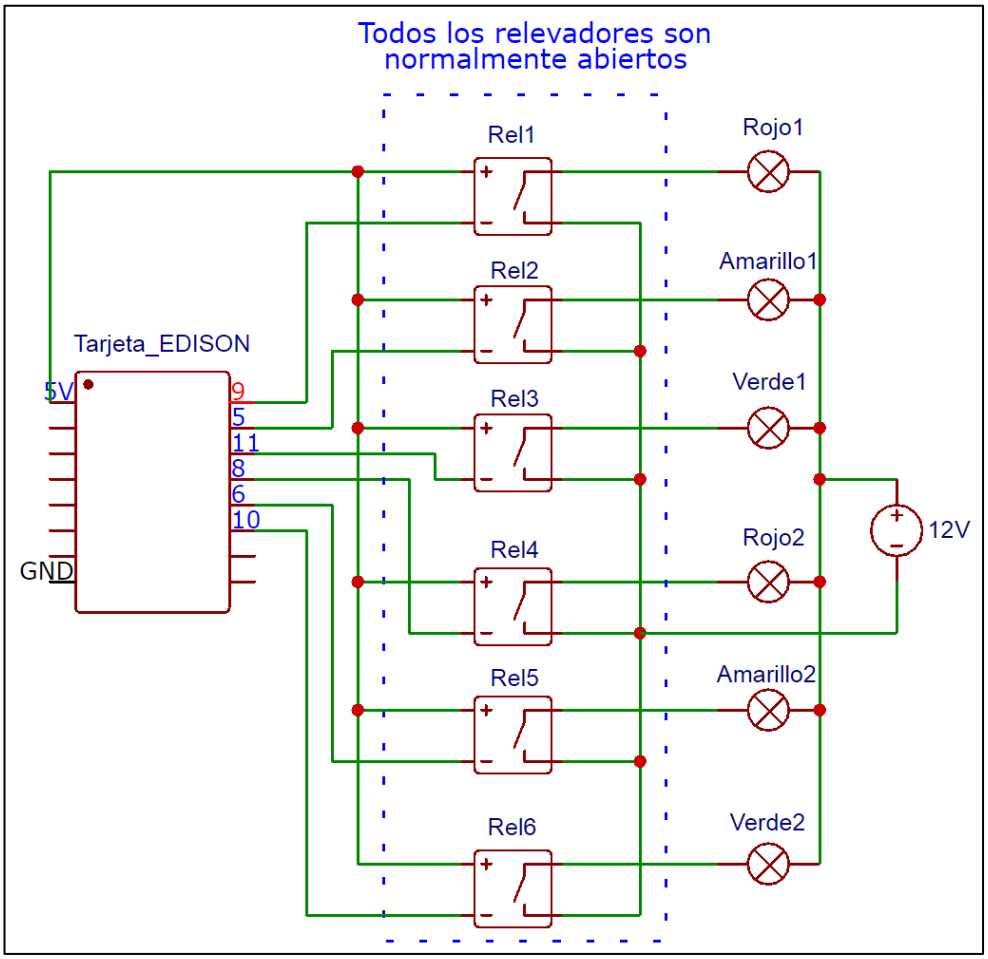

Figura 9: Diagrama eléctrico básico del prototipo de SICT

3.1.3. Componentes periféricos

Como complemento al circuito de control de luces, se requiere conectar a la tarjeta Edison la alimentación desde la fuente poder dual (o en su defecto, una fuente de poder dedicada), un replicador de puertos USB, dos cámaras web, una antena para WiFi y opcionalmente, un cable USB para comunicación con una computadora. La ubicación de las conexiones se muestra en la figura 10:

3.1.3.1. Conexión de componentes periféricos

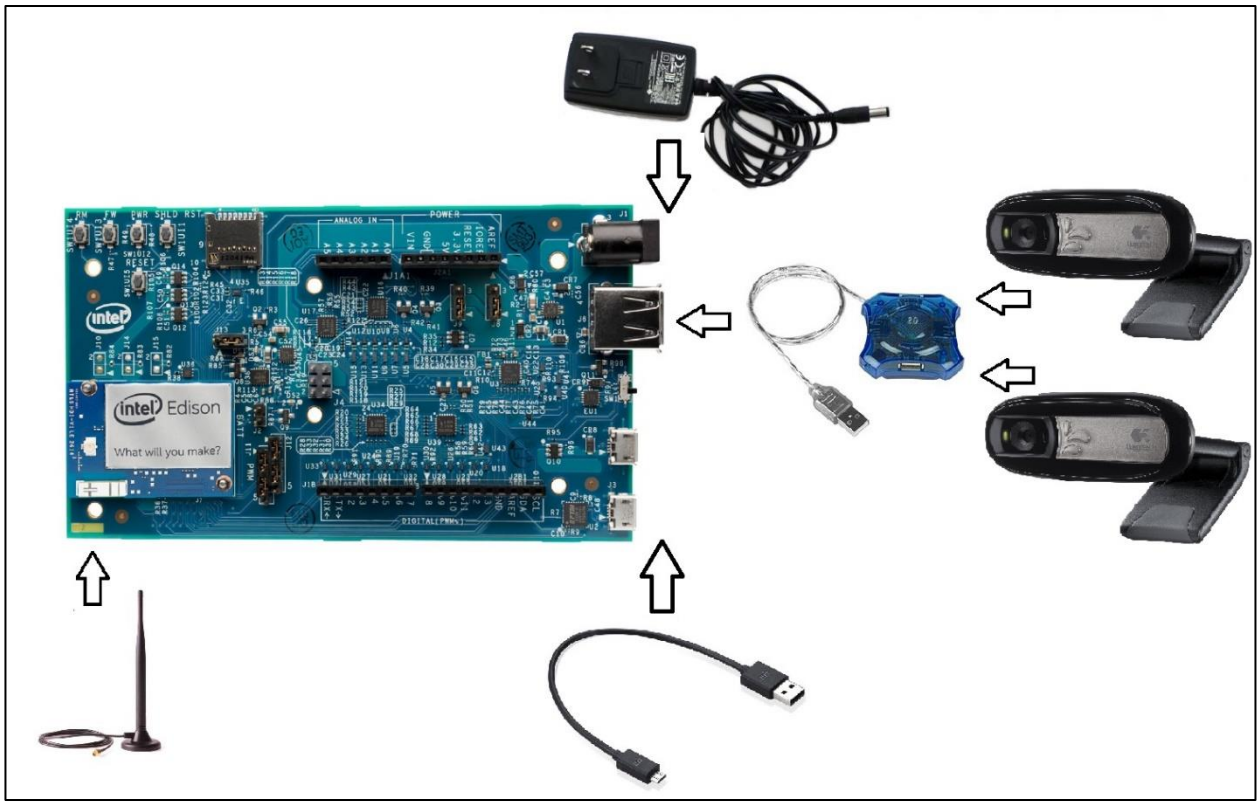

Figura 10: Componentes periféricos del prototipo de SICT

3.2. IMPLEMENTACIÓN DEL SICT

Siguiendo la premisa del uso de software de código abierto planteada en la hipótesis del proyecto, se elige basar el funcionamiento del SICT en un sistema operativo Linux Embebido generado usando las herramientas del proyecto Yocto [6] que incluye las librerías para manejo del lenguaje de programación Python [16], las librerías de manejo de imagen OpenCV [14], la librería de interfaz de puertos I/O *mraa* [17] y los programas propios del SICT descritos en el apartado "*Código*" de este documento. Una abstracción gráfica de este modelo se presenta en la figura 11:

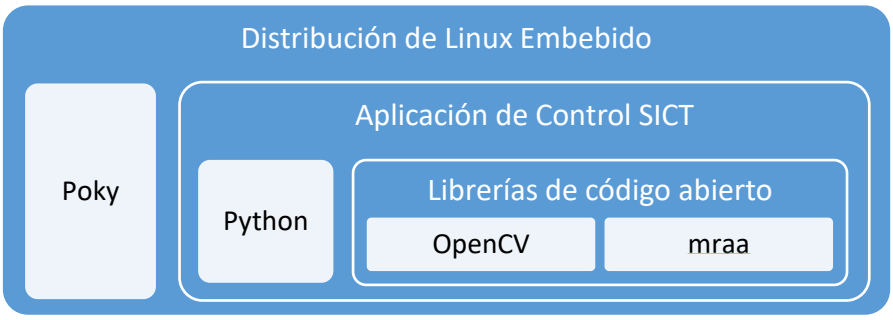

Figura 11: Modelo de implementación del SICT

3.2.1. Desarrollo de aplicación SICT

La aplicación principal del SICT está escrita en lenguaje Python y su funcionamiento es el siguiente:

- Configura el hardware del SICT (puertos de control, canales comunicación de video y de transmisión de http).
- Recibe la imagen de video desde las cámaras del sistema y la procesa para detectar movimiento y presencia de objetos.
- Genera llamados a la rutina de cambio de luces únicamente cuando detecta la presencia de vehículos en el camino que se encuentra detenido por la señal de luz roja.
- Crea una imagen procesada por cada uno de los caminos monitorizados que incluye la imagen capturada por las cámaras, un mensaje de texto sobre el estado de detección de vehículos, un mensaje de texto con el tiempo y fecha actuales, un mensaje con el tiempo restante para el siguiente cambio de luces y un conjunto de luces de color que replican el estado actual de las luces del semáforo.

- Genera un servidor http donde se publica un documento en formato *html* que incluye la imagen procesada generada por el SICT para ser consumida por los clientes conectados a la misma red que el SICT.

3.2.1.1. Diagrama de flujo del SICT

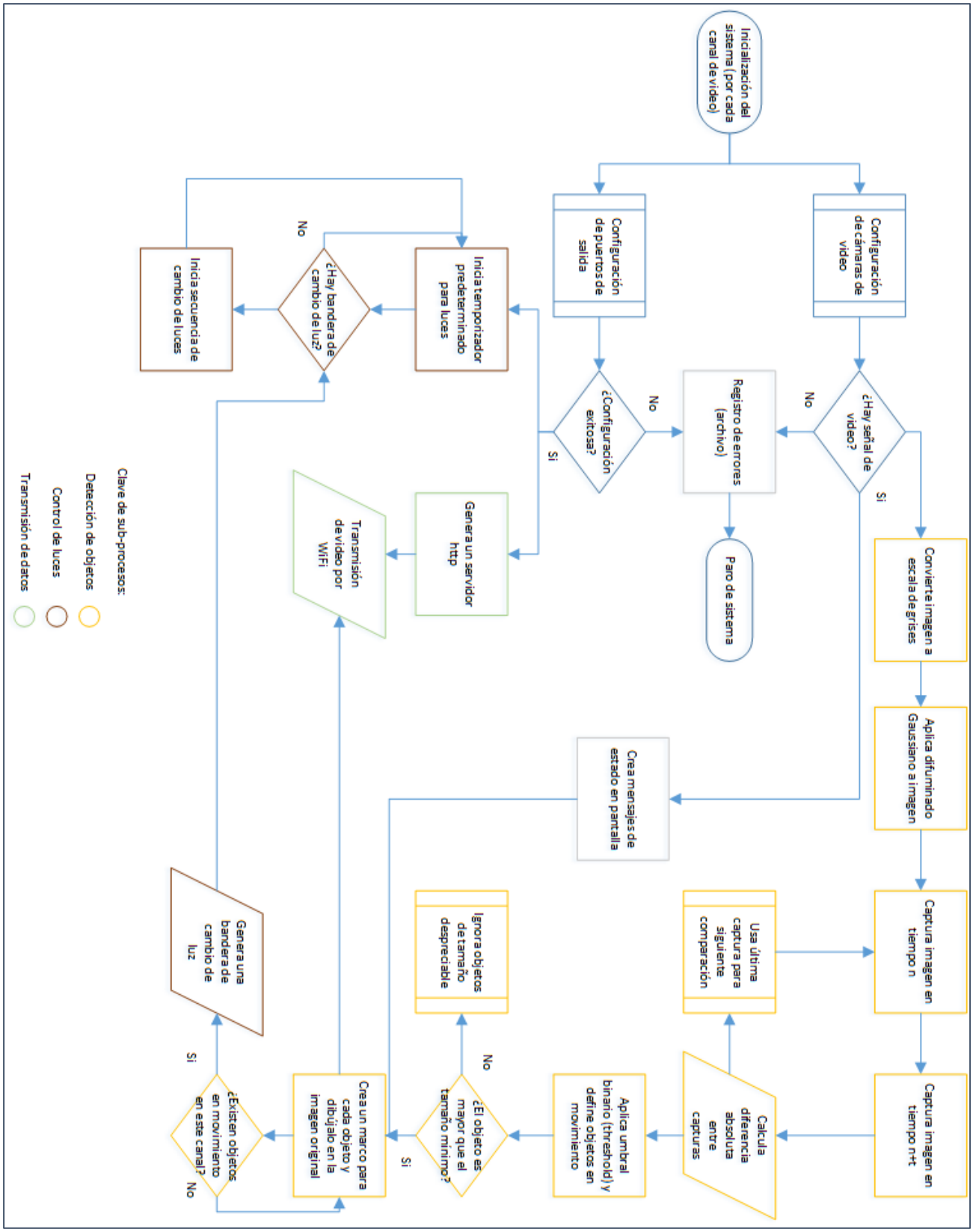

Figura 12: Diagrama de flujo del SICT

3.2.1.2. Código

Se desarrollaron tres archivos de código propios del SICT en lenguaje Python:

- **sict.py** – Programa de control principal. Incluye las siguientes funciones:
  - *gpio_setup* – Define los puertos digitales de la tarjeta Edison que serán usados y los configura como salidas para comandar los relevadores.
  - *turn_off_all_lights* – Debido a que los puertos digitales son "*pull-down*", para evitar estados de encendido indeseados se comanda un estado de energía alto para todos los puertos.
  - *cameras_setup* – Detecta el número de video cámaras disponibles en el sistema y configura el tamaño y saturación de la imagen a capturar.
  - *light_change* – Contiene el comando de cambio secuencial de luces de verde a amarillo y a rojo, alternado entre ambos sentidos del camino.
  - *light_circles* – Genera un conjunto de luces de color que simulan el estado actual de las luces del semáforo y las añade al video enviado por el sistema.
  - *cam_handler* – Contiene los algoritmos de lectura de imagen de video, detección de movimiento, creación de un marco alrededor de los objetos detectados, generación del mensaje de objetos detectados y fecha del sistema, generación del mensaje de tiempo restante para cambio de luces, y de generación de un documento *html* que incluye el video editado final.
  - *main* – Llama secuencialmente al resto de funciones del programa y genera un servidor http
- **set_server_ip.py** – Detecta la dirección IP del sistema que deberá ser usada para la conexión de los clientes.
- **sync_time.py** – Sincroniza el tiempo del SICT con el del *cluster* de servidores *ntp* de internet.

El código de estos programas se muestra en el ***Anexo A: Código*** de este documento.

3.2.2. Creación de imagen de Linux Embebido

El sistema operativo del SICT es una distribución de Linux Embebido creada con las herramientas del proyecto Yocto [6], que emplean un modelo de creación de imágenes basado en capas de software, comenzando con el *kernel* de Linux, los controladores de hardware específicos de la plataforma empleada (Intel Edison en este caso) y siguiendo

con la adición de capas específicas como librerías de lenguajes de programación o de aplicaciones particulares. Una representación gráfica del modelo seguido en el proyecto Yocto [6] se presenta en la figura 13:

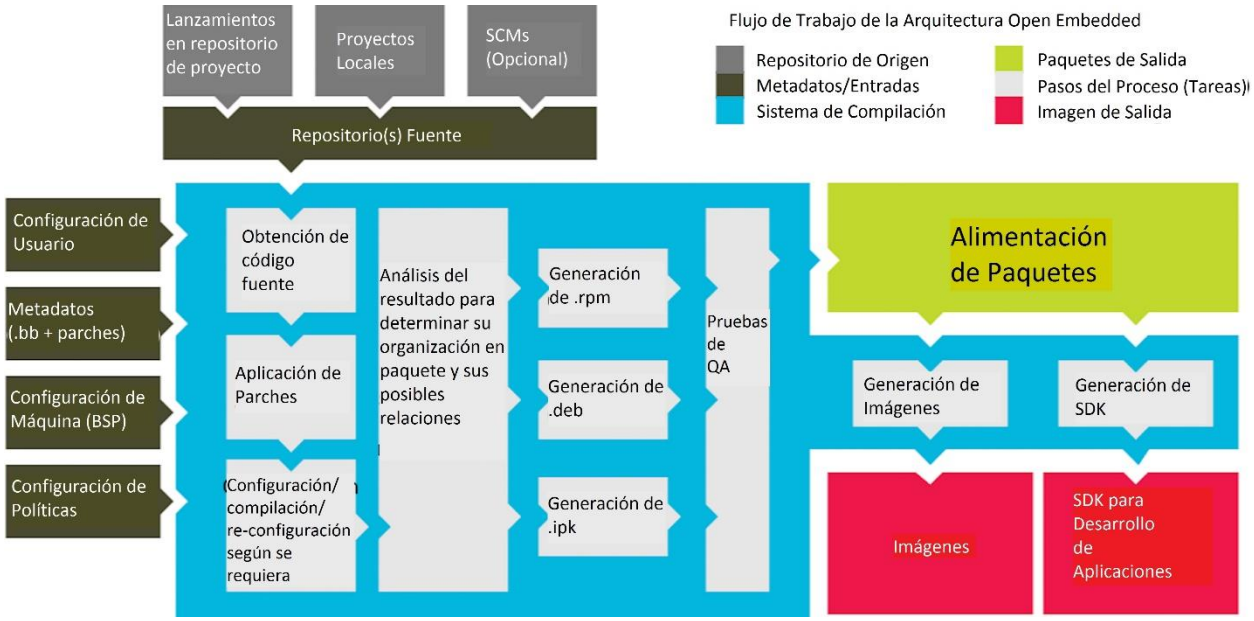

Figura 13: Modelo del Proyecto Yocto

3.3. ENFOQUE METODOLÓGICO

Para la verificación de la validez de la hipótesis del presente proyecto, se infiere que el ecosistema en el cual debe realizarse dicha verificación incluye una población de programadores de software con conocimiento de desarrollo de sistemas embebidos, quienes pueden evaluar el grado de éxito en la creación del SICT, así como de una población de conductores de vehículos y peatones quienes pueden evaluar el grado de éxito en la funcionalidad ofrecida por el sistema. Ambas poblaciones no son excluyentes entre sí, por lo que una puede incluir a la otra.

Por lo anterior, se eligió emplear un enfoque cualitativo de los resultados de la investigación, constituidos por la evaluación personal de los miembros de diferentes poblaciones de estudio sobre el desempeño del STCI, lo que permitió determinar el grado de certidumbre en la afirmación propuesta.

3.4. DISEÑO DE EXPERIMENTO DEL SICT

El proyecto de SICT fue publicado como un proyecto de código abierto en la plataforma de almacenaje, compartimiento y desarrollo de software *github* [18] con el nombre de *Intelligent Traffic Control* [19] para poner a disposición de la comunidad de desarrolladores nacional e internacional el código desarrollado.

El prototipo funcional de SICT fue presentado de manera presencial a diferentes poblaciones para su evaluación cualitativa sobre el desempeño del sistema en los siguientes eventos:

- Exposición "*Open House*" del **Centro de Tecnología de Código Abierto de Intel** (**OTC**). Abril 22, 2016.
- Exposición permanente del **Laboratorio para Pequeñas y Medianas empresas** (PYMES) del Centro de Diseño Guadalajara (GDC) de Intel. Mayo, 2016.
- Evento de inicio del Programa de Capacitación "*Internet de las Cosas*" en **México Innovación y Desarrollo** (**MIND**) [20]. Agosto 18, 2016.
- Conferencia de **Profesionales de Software** (**SWPC**) de Intel. Octubre 6, 2016.

3.5. POBLACIÓN ESTUDIADA

A continuación se describen las características relevantes de la población asistente a cada uno de los eventos donde se presentó el SICT:

- *Open House de OTC* – Comunidad de desarrolladores de software basado en código abierto. Incluye profesionales de sistemas operativos embebidos, de Android y de Linux.
- *Laboratorio PYMES de GDC* – Comunidad de estudiantes de Ingeniería en Computación y similares, Comunidad de entusiastas del Internet de las Cosas "*makers*", emprendedores del sector privado y representantes del sector público de gobierno.
- *Programa de Capacitación en MIND* – Comunidad de empresarios mexicanos con interés en implementar soluciones de bajo costo basadas en el "Internet de las cosas".
- *Conferencia SWPC* – Comunidad de profesionales del desarrollo de software para múltiples plataformas, incluye desarrolladores para plataformas basadas en Windows, en Linux y en MacOS.

- *Usuarios de Github* – Comunidad de desarrolladores de software global que emplean tanto software de código abierto como propietario.

4. RESULTADOS.

4.1. DEMOSTRACIÓN DEL FUNCIONAMIENTO DEL SICT

Con el fin de obtener una evaluación cualitativa de los asistentes a los eventos donde se presentó el prototipo del SICT, se configuró el sistema de la siguiente manera:

- El SICT detectó la presencia de peatones en dos caminos perpendiculares entre sí. Las luces del semáforo no tuvieron un periodo de tiempo predeterminado entre cada cambio de luces.
- Al detectar la presencia de uno o más peatones en alguno de los caminos, se generaba una solicitud de cambio de luz para el camino en cuestión.
- Cuando se detectaba presencia de peatones en ambos caminos, se seguía un patrón de cambios por intervalo de tiempo.
- El SICT transmitió en tiempo real el video capturado con ambas cámaras por red inalámbrica WiFi, incluyendo en la imagen el tiempo y fecha actuales, el estado de detección de peatones y círculos de color que replicaban el estado de las luces en cada sentido del camino.

4.1.1. Prueba de simulación de semáforos

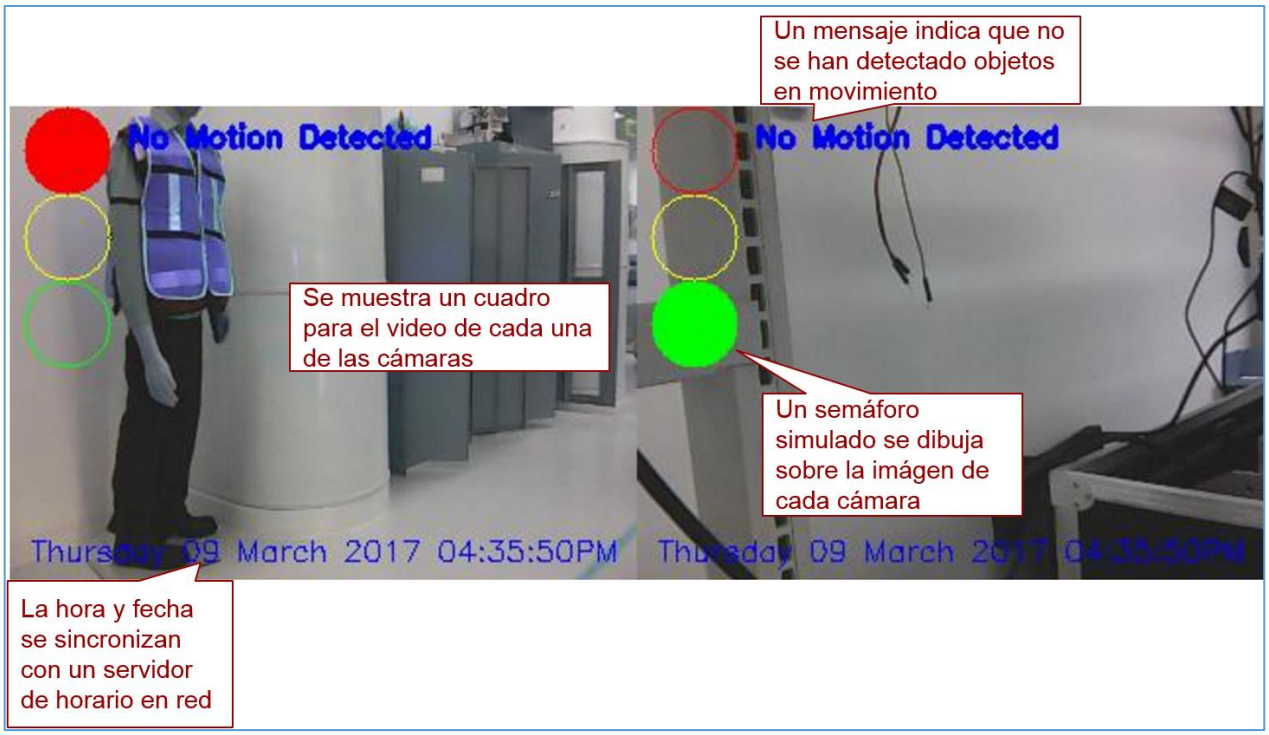

Figura 14: Prueba de simulación de semáforos

4.1.2. Prueba de detección de objetos en movimiento

Figura 15: Prueba de detección de objetos en movimiento

4.1.3. Prueba de detección de objetos múltiples

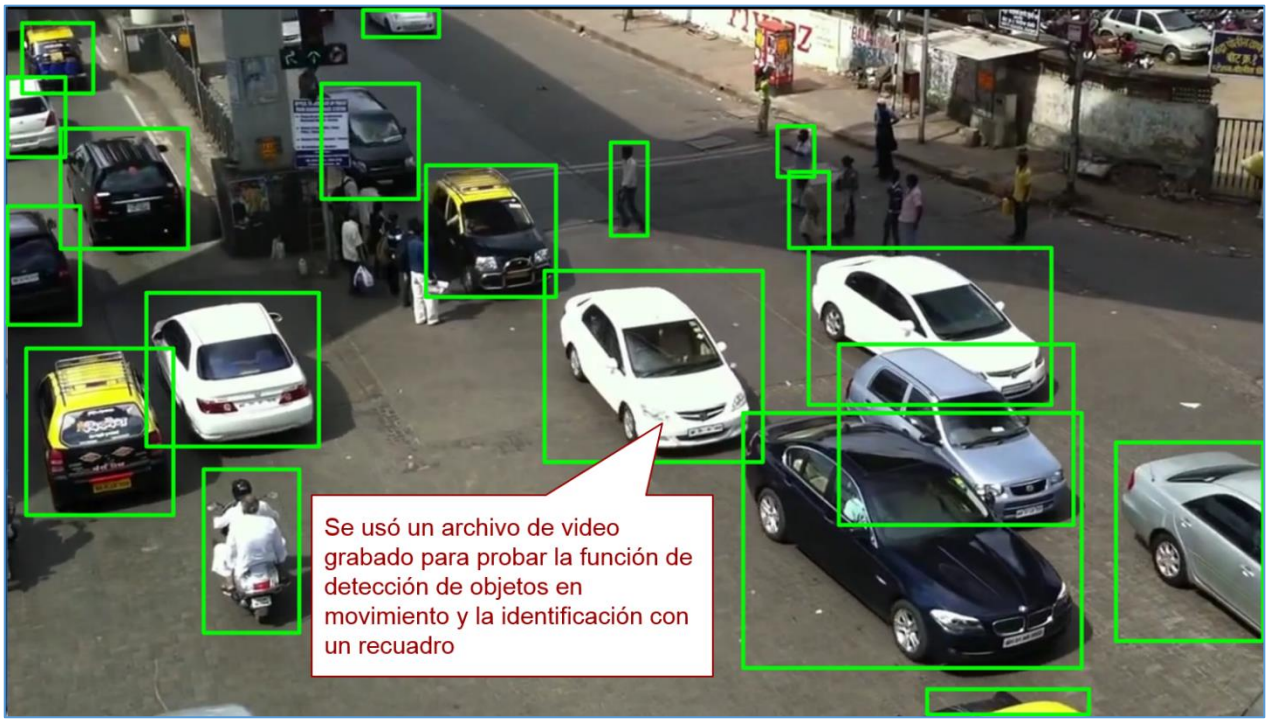

Figura 16: Prueba de detección de objetos múltiples

4.1.4. Presentación del SICT en el evento "Open House" del Centro de Tecnología de Código Abierto (OTC)

El Centro de Tecnología de Código Abierto (OTC por sus siglas en inglés) es el corazón de los esfuerzos de desarrollo de código abierto en Intel y tiene la misión de entregar software de código abierto e innovaciones tecnológicas con calidad de producto final, además que estar a cargo de asistir a todas las organizaciones de Intel que también trabajan en código abierto. El *Open House de OTC* es un evento anual en el que cada equipo de trabajo de OTC muestra a todas las unidades de negocio de Intel las áreas en las que se desempeña así como el nivel de experiencia con el que cuenta, con el fin de buscar oportunidades de colaboración y de fomentar la creatividad entre los empleados de Intel.

El SICT fue elegido para ser presentado en el área de demostración del equipo del Proyecto Yocto, como una muestra de los resultados prácticos que pueden obtenerse al emplear las herramientas del proyecto y para fomentar una discusión sobre otros posibles usos de dichas herramientas así como de las formas de colaboración de la comunidad de software de código abierto.

En el anexo C de este documento se incluyen imágenes de esta presentación en las ilustraciones número 17 y 18.

4.1.5. Inclusión del SICT a la exposición permanente del Laboratorio para Pequeñas y Medianas empresas (PYMES) del Centro de Diseño Guadalajara de Intel

El objetivo de este laboratorio es apoyar el desarrollo y aceleración de proyectos innovadores de empresas Pequeñas o Medianas (PYMES). Se busca que las PYMES tengan la oportunidad de transformar sus ideas en prototipos y en productos innovadores así como la interacción entre la Academia y la Industria por medio de charlas de tendencias tecnológicas que son ofrecidas periódicamente en el auditorio del Centro de Diseño Guadalajara de Intel.

El laboratorio para PYMES está equipado para el desarrollo y validación de prototipos y productos. Cuenta con equipo de pruebas como osciloscopios y analizadores lógicos, herramientas de desarrollo de software en arquitecturas de núcleos múltiples, herramientas de desarrollo de software para servidores, PC, tabletas y teléfonos

inteligentes, herramientas para desarrollo de sistemas embebidos en arquitecturas de núcleos múltiples, así como un área de oficinas para el grupo de trabajo de la PYME.
El SICT fue elegido para ser incluido en la exposición permanente del laboratorio, donde se incluyen prototipos funcionales de productos elaborados con tecnología de Internet de las cosas para diferentes áreas de la industria.

4.1.6. Presentación del SICT en evento de inicio del Programa de Capacitación Internet de las Cosas en MIND

El Programa de Capacitación Internet de las Cosas busca acercar a todos los sectores industriales al desarrollo y aplicación del internet de las cosas, a que conozcan a detalle su desarrollo y aplicación e identifiquen sus componentes para generar ideas que puedan llevarse a una PC, dentro de su segmento de mercado, así como las herramientas que les ayudarán a ahorrar costos e incrementar ventas. Está dirigido a representantes del departamento de Innovación, IT, Ingeniería, Ventas y Marketing de las empresas [20].
El SICT fue elegido para ser mostrado en el evento inicio de este programa, donde asistieron empresarios y desarrolladores de la industria del calzado, de la moda y de la manufactura que están interesados en implementar el internet de las cosas en sus respectivas empresas, o que ya lo están haciendo y buscan incrementar su conocimiento al respecto.

4.1.7. Presentación del artículo "Early prototyping an Intelligent Traffic Control System using Yocto Project and OpenCV" en la Conferencia de Profesionales de Software (SWPC) de Intel

La conferencia de Profesionales de Software de Intel (SWPC por sus siglas en inglés) es un evento anual que se lleva a cabo de manera simultánea en más de 10 sitios alrededor del mundo (15 sitios en 2016) y que tiene como objetivo poner a disposición de todos los empleados de Intel y de sus subsidiarias, soluciones probadas basadas en principios de desarrollo de software de calidad y sus resultados más recientes, a través de presentaciones, posters, demostraciones y talleres de aplicación. En esta conferencia los asistentes tienen la oportunidad de constatar cómo otras unidades de negocio de Intel usan herramientas y metodologías de calidad para impactar de manera positiva la entrega de productos de software.

El artículo en inglés "*Early prototyping an Intelligent Traffic Control System using Yocto Project and OpenCV*" (Implementación temprana de un Prototipo de Sistema Inteligente de Control de Tráfico usando el Proyecto Yocto y OpenCV), donde se muestra el proceso de diseño e implementación del SICT, fue seleccionado para ser presentado en la SWPC con el fin de mostrar a la audiencia las ventajas, desventajas y requerimientos del empleo de las herramientas de desarrollo del Proyecto Yocto y de las librerías de visión de computadora de código abierto OpenCV en el desarrollo en un corto tiempo de un prototipo de sistema embebido inteligente. Además de la presentación del artículo, seguida de una sesión de preguntas y respuestas, se llevó a cabo una demostración funcional del prototipo del SICT en un área especialmente dedicada a tal fin. La demostración fue reconocida como "*Best Demo*" (Mejor demostración) del evento con base en el número de votos de los asistentes.

El anexo C de este documento incluye la figura 20 con una imagen de la presentación del SICT en el SWPC 2017.

4.2. ANÁLISIS DE RESULTADOS

En todos los casos, el SICT presentó el funcionamiento esperado, siendo la respuesta y comentarios de los asistentes a los eventos descritos la materia del análisis de resultados posterior.

Un factor común a todas las audiencias estudiadas, fue la aceptación por el desempeño del SICT, coincidiendo la mayoría en que visualizaban beneficios inmediatos en el manejo del tráfico si el SICT se implementara a gran escala en un área metropolitana como la de la ciudad de Guadalajara.

También fue notoria la sorpresa al percibir la sencillez del sistema necesario para realizar el reconocimiento de imágenes, junto con la brevedad del código asociado. Muchos espectadores coincidieron en su intención de descargar (ellos mismos o por personal a su cargo) el código del SICT en línea para estudiar las funciones utilizadas y tomarlo como base para futuros proyectos.

Las características específicas de la respuesta de cada población se describen a continuación:

4.2.1. Comentarios de los visitantes al Laboratorio para Pequeñas y Medianas empresas (PYMES) del Centro de Diseño Guadalajara de Intel

El principal interés mostrado por los visitantes al laboratorio de PYMES ha sido por la posibilidad de incluir el aprendizaje de las herramientas y procedimientos del proyecto Yocto en sus respectivas áreas de trabajo, ya que la portabilidad que se puede lograr con este modelo les permite usar diferentes plataformas de hardware, de Intel y de otros proveedores, para sus implementaciones.

Los estudiantes se muestran fuertemente motivados a desarrollar aplicaciones que hagan uso del procesamiento de imagen por computadora al ver la sencillez del código necesario en el SICT y la posibilidad de programar en otros lenguajes. Las elevadas prestaciones de la librería OpenCV, junto con el hecho de que se trata de software de código abierto, les ha causado también un gran interés por profundizar en su uso y aprendizaje.

4.2.2. Comentarios de los asistentes al evento " Open House" del Centro de Tecnología de Código Abierto (OTC)

Los asistentes a este evento fueron en su mayoría empleados de Intel Guadalajara, para quienes fue una agradable sorpresa descubrir el elevado desempeño de la tarjeta de desarrollo Edison, especialmente al saber que todas las funciones del prototipo se realizan por completo en la tarjeta.

Varios de los espectadores se mostraron interesados en colaborar en el proyecto del SICT a través del repositorio de código *github*, especialmente en las posibles modificaciones al SICT para emplearlo en funciones alternas, como de control de espacios de estacionamiento.

4.2.3. Comentarios de los asistentes al Evento de Inicio del Programa de Capacitación de MIND

La asistencia a este evento en general estuvo compuesta por empresarios y sus colaboradores de las áreas de diseño. Su principal interés fue la posible utilización del procesamiento de imagen por computadora (compute visión) en sus ámbitos productivos, por ejemplo en el reconocimiento de formas geométricas para ser aprovechado en el control de calidad de la cadena de producción.

Una inquietud recurrente fue respecto a la protección de la propiedad intelectual cuando se emplea software de código abierto, por lo que la mención de la posibilidad de generar código propietario basado en software de código abierto fue recibida con bastante interés también.

4.2.4. Comentarios de los asistentes a la SWPC:

La SWPC está dirigida principalmente a desarrolladores, por lo que las preguntas sobre el SICT estuvieron principalmente dirigidas al algoritmo de procesamiento de imagen: Cuáles fueron los criterios usados para su diseño, la dificultad de implementación, la portabilidad del código y su desempeño con menores recursos de hardware, entre otras. Fue también notorio el interés en emplear las funciones de procesamiento de imagen en proyectos diversos, por lo que también se cuestionó sobre las posibles mejores prácticas que el autor considera que pueden ser benéficas en otros proyectos similares.

4.3. RETOS DEL PROYECTO

4.3.1. Elección e implementación de cámara de video

4.3.1.1. Módulo OV7670

Durante la fase de diseño del sistema se buscó un modelo de cámara de video que brindara una combinación eficiente de calidad de video, velocidad de transferencia, requerimiento de potencia y precio.

La primera elección fue el modelo OV7670 de la marca OmniVision [21]. Este módulo de cámara de video de 640x480 pixeles incluye un módulo de memoria intermedia del tipo "Primero en entrar, primero en salir" (FIFO, por sus siglas en inglés) que reduce el requerimiento de procesamiento de imagen en el microprocesador del sistema, y que tenía el costo más bajo entre las opciones de cámaras disponibles al momento de la investigación.

La implementación del módulo OV7670 requiere incluir una librería para configurar la cámara y controlar la señal de video. Existen diversas variantes disponibles para varias plataformas de hardware, sin embargo, las pruebas con la plataforma elegida para el SICT resultaron fallidas debido a que ninguna de las librerías mencionadas fue diseñada originalmente para ser compatible con microprocesadores de arquitectura x86, resultando en una señal de video deficiente, cuando no inexistente para la mayoría de las pruebas. La búsqueda de una librería funcional continuó por más de sesenta días, sin éxito; esto hizo patente la necesidad de desarrollar una librería específica para la plataforma de hardware del SICT, si se deseaba emplear la cámara OV7670, lo cual incrementaría considerablemente el tiempo de desarrollo del sistema, además de estar completamente fuera del objetivo principal del proyecto. Por todo lo anterior, se decidió abandonar este modelo de cámara como opción para el SICT.

4.3.1.2. Webcam C170

Como consecuencia del fracaso en el uso de un módulo de cámara de video conectado a los puertos digitales de la tarjeta de desarrollo Intel Edison, se procedió a evaluar la cámara tipo webcam USB modelo Logitech C170 de 1024 x 768 pixeles.

El primer factor de selección de este tipo de cámara fue la compatibilidad del puerto USB: La plataforma de desarrollo del SICT cuenta con un puerto USB versión 2.0, para el que existen librerías de control totalmente funcionales.

El segundo factor de decisión fue la velocidad de transferencia de datos. A pesar de tratarse de un puerto de comunicación serial, el USB 2.0 permite una velocidad de transferencia máxima de 60 megabytes por segundo (60 MBps), que, si bien no es ideal para transmisión de video, sí permite transmitir la imagen de dos cámaras de video de manera simultánea en la plataforma de desarrollo tipo Arduino del SICT [15]. Cabe hacer mención que la tarjeta de Desarrollo Intel Edison cuenta con un puerto de comunicación USB versión 3.0, que permite velocidades de transferencia 10 veces mayores que la versión anterior, sin embargo, la plataforma tipo Arduino sobre la que está montada la Edison sólo cuenta con el puerto 2.0. Esta limitante obligó a configurar la resolución de las imágenes capturadas a un tamaño de 320 x 240 pixeles, para asegurar un flujo continuo de información.

Finalmente, la relación costo-beneficio del modelo C170, que ofrece una resolución de imagen máxima de 1024 x 768 pixeles y su compatibilidad con el estándar UVC requerido por la librería OpenCV [10], fueron determinantes para elegirlo como la elección final para el prototipo de SICT.

4.3.2. Observación del alcance fijado en los objetivos del proyecto como acotamiento para el proyecto en general.

Durante todo el proceso de desarrollo del SICT fue recurrente la tendencia a desviarse de los alcances fijados en los objetivos del proyecto al buscar mejor las prestaciones del sistema. Por ejemplo, la detección de vehículos en movimiento puede extenderse a diferenciar el tamaño, la forma, la dirección o la velocidad en que dichos vehículos se mueven. La detección de objetos en las imágenes obtenidas de las cámaras también puede incluir detectar formas específicas, para diferenciar vehículos de peatones, o personas de animales, o también es posible detectar la presencia de ciertas imágenes predeterminadas, como escudos en vehículos o símbolos de unidades de emergencia. Estas variantes seguramente permitirían extender la funcionalidad del SICT y obtener aún mayores beneficios de su uso, sin embargo, durante la exploración de cada una de dichas mejoras quedó claro el aumento exponencial del tiempo requerido para el desarrollo del algoritmo de control, lo que, si se hubieran incluido dichas mejoras, hubiera extendido la duración del proyecto más allá del tiempo establecido como límite para este trabajo. Por todo esto, se decidió apegarse al objetivo básico de proporcionar un prototipo funcional del sistema, y proponer las mejoras mencionadas en el sitio de

compartición de código Github [19], con el fin de que miembros de la comunidad de desarrolladores de código abierto participen en su implementación.

CONCLUSIONES

Desde el inicio de las pruebas funcionales quedó claro para el autor que la viabilidad de la implementación del SICT empleando la plataforma de hardware elegida y software de código abierto es incuestionable. El prototipo de SICT ha sido demostrado en varios eventos durante los cuales su funcionalidad y ventajas han sido constatadas tanto por desarrolladores como por público en general. El funcionamiento del prototipo cubre la expectativa de contar con un control de patrón de cambios acorde a la presencia de vehículos o peatones en los caminos controlados y logra así reducir los tiempos de espera innecesarios. Se cumple también, el requerimiento de contar con un modo de cambio de luces por tiempo determinado que puede ser modificado por medio de la definición de variables en la aplicación de control scti.py (incluida en el anexo A de este documento).

La combinación de software de código abierto y componentes de uso común permite la creación de prototipos funcionales en un corto tiempo, lo que beneficia directamente a los equipos de desarrollo, quienes pueden enfocar sus esfuerzos en el desarrollo y mejora de sus algoritmos de control y procesamiento de datos, aprovechando el uso de sistemas operativos y librerías existentes para el manejo de las capas de manejo de hardware y de comunicaciones, por ejemplo.

La organización del código de aplicación principal en bloques funcionales permite ofrecer diferentes grados de funcionalidad simplemente con la adición o supresión de sub-bloques. La publicación del código en repositorios públicos permite no sólo la reutilización de estos bloques en otro proyectos similares (disponibles en los repositorios de la comunidad de desarrolladores de software de código abierto) sino además la participación de dicha comunidad en la mejora y posible evolución de este proyecto.

Este proyecto fue concebido desde su inicio como la base para desarrollos futuros de funciones y mejoras, complementándose unas a otras. Dichas posibles mejoras se enlistan en el repositorio de código del proyecto SICT en el sitio de compartición Github, donde se invita a la comunidad de desarrolladores de código libre tanto a hacer uso del código

del proyecto como a contribuir en el desarrollo de las nuevas funciones del SICT y proponer nuevas. Entre otras, estas nuevas funciones incluyen:

Crear una capa de aplicación SICT para el proyecto Yocto. Actualmente el código de la aplicación debe ser copiado de forma manual a una tarjeta de desarrollo Intel Edison que cuente con un sistema operativo creado con las herramientas del proyecto Yocto. Al contar con una capa de aplicación compatible, será posible generar imágenes de sistema operativo con la aplicación del SICT incluida y además permitir un control de versiones para la misma.

Identificar Regiones de Interés (RDI). Al definir las secciones del campo visuales que corresponden a áreas de actividad definidas (por ejemplo, aceras peatonales y arroyo vehicular), será posible optimizar los recursos del SICT (al discriminar áreas sin interés al analizar imágenes) y por lo tanto reducir el tiempo de respuesta en general del sistema.

Reconocimiento de objetos o formas. La implementación de un algoritmo de reconocimiento de objetos o de formas permitirá diferenciar los objetos detectados e ignorar algunos y enfocarse en otros, por ejemplo, autos y peatones, animales pequeños y humanos, etc.

Detección automática de hardware. Al detectar de forma automática la plataforma de hardware sobre la que funciona la aplicación el SICT, permitirá incluir diferentes configuraciones de puertos de comunicación GPIO para diferentes plataformas de hardware pero usando el mismo código de aplicación y elegir la versión adecuada en cada caso. Por ejemplo, el código de aplicación SICT tal y como está presentado en el anexo A de este documento, puede ser usado en tarjetas de desarrollo Intel Edison y si se configuran adecuadamente los puertos de comunicación GPIO puede ser usado en tarjetas de desarrollo de modelo Minnowboard Max o en tarjetas modelo Intel Galileo.

Sincronización de nodos múltiples. Al establecer comunicación bidireccional con otros dispositivos SICT se podrán sincronizar los cambios de luces a lo largo de un camino así como extender la transmisión de datos más allá del rango de alcance de uno sólo de los dispositivos.

Finalmente, la principal aportación del autor al proyecto puede resumirse en la conjunción del conocimiento técnico requerido para diseñar el sistema, implementar el algoritmo de control más económico en términos de requerimientos de poder de cómputo, y el dominio de las características y requerimientos del modelo de desarrollo

de software de código abierto, todo lo cual permite comprobar de manera teórica y práctica la factibilidad tanto técnica como económica del empleo de sistemas embebidos de arquitectura abierta y software de código abierto para el desarrollo de sistemas inteligentes multimedia.

BIBLIOGRAFÍA Y REFERENCIAS

1. **Hirsh, Evan, y otros.** 2015 Auto Trends. *Strategy&.* [En línea] 12 de 10 de 2015. http://www.strategyand.pwc.com/perspectives/2015-auto-trends.

2. **Ament, Phil.** Traffic Light. *The Great Idea Finder.* [En línea] 13 de Octubre de 2015. http://www.ideafinder.com/history/inventions/trafficlight.htm.

3. **The Association of Electrical Equipment and Medical Imaging Manufacturers.** *The Association of Electrical Equipment and Medical Imaging Manufacturers.* [En línea] 15 de Octubre de 2015. http://www.nema.org/pages/default.aspx.

4. Secretaría de Comercio y Fomento Industrial. Catálogo de Normas Mexicanas. *Dirección General de Normas.* [En línea] 12 de Julio de 1981. http://www.economia-nmx.gob.mx/normas/nmx/1981/nmx-j-425-1-1981.pdf.

5. OpenCV Foundation. OpenCV. *OpenCV.* [En línea] 20 de Octubre de 2015. www.opencv.org.

6. Yocto project, A Linux Foundation Collaborative Project. Yocto Project. *Yocto Project.* [En línea] 11 de October de 2015. http://www.yoctoproject.org.

7. Minnesota Department of Transportation. Traffic Signals 101 Course Book. *Minnesota Department of Transportation. Traffic Engineering.* [En línea] February de 2015.

8. Sumitomo Electric Industries, Ltd. Sumitomo Electric's ITS | Ultrasonic Vehicle Detector. [En línea] Sumitomo Electric Industries, Ltd., 2015. http://global-sei.com/its/products/uvd.html.

9. *On the Theory of Light and Colours.* Young, Thomas. s.l. : Phil. Trans. R. Soc. Lond., 1802. The Bakerian Lecture. págs. 12-48.

10. OpenCV Dev Team. OpenCv 2.4.12.0 Documentation. *OpenCV.* [En línea] 9 de April de 2016. http://docs.opencv.org/2.4/index.html.

11. Glasgow, Larry A. *Applied Mathematics for Science and Engineering.* New Jersey : John Wiley & Sons, 2014. ISBN-10: 1-118-74992-8.

12. Oscar Deniz Suarez, et al. *OpenCv Essentials.* Birmingham : Packt Publishing Ltd., 2014. ISN-13: 978-1-78398-425-1.

13. Open Source Initiative. The BSD 3-Clause License. *Open Source Initiative.* [En línea] 9 de April de 2016. https://opensource.org/licenses/BSD-3-Clause.

14. OpenCV Foundation. About OpenCV. *OpenCV.* [En línea] 9 de April de 2016. [Citado el:] http://www.opencv.org/about.html.

15. USB.org. USB Device Class Specifications. *Universal Serial Bus.* [En línea] July de 2011. http://www.usb.org/developers/docs/devclass_docs/USB_Video_Class_1_5.zip.

16. Python Software Foundation. Welcome to Python.org. *python.org.* [En línea] 13 de October de 2016. www.python.org.

17. Intel. mraa. *mraa.* [En línea] Intel, 13 de October de 2016. www.iotdk.intel.com/docs/master/mraa.

18. Github Inc. github. *github.com.* [En línea] 13 de October de 2016. ww.github.com.

19. Lamego, Jose. Intelligent Traffic Control. *Intelligent Traffic Control Repository.* [En línea] 13 de October de 2016. https://github.com/joselamego/IntelligentTrafficControl.

20. MIND. MIND Mexico. *MIND Mexico.* [En línea] Agosto de 2016. www.mindmexico.com.

21. Omnivision Technologies. Van Ooijen Technische Informatica. *Van Ooijen Technische Informatica.* [En línea] 8 de July de 2005. http://www.voti.nl/docs/OV7670.pdf.

22. City of Mesa Government. City of mesa. [En línea] September de 2015. mesaaz.gov/home/showdocument?id=11499.

ANEXO A: CÓDIGO

sict.py

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

#   LightControl. Traffic light control using object detection.
#   Copyright (C) 2016  Jose Lamego <jose.a.lamego@intel.com>
#
#   This program is free software: you can redistribute it and/or modify
#   it under the terms of the GNU General Public License as published by
#   the Free Software Foundation, either version 3 of the License, or
#   (at your option) any later version.
#
#   This program is distributed in the hope that it will be useful,
#   but WITHOUT ANY WARRANTY; without even the implied warranty of
#   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#   GNU General Public License for more details.
#
#   You should have received a copy of the GNU General Public License
#   along with this program.  If not, see <http://www.gnu.org/licenses/>.

# ************* Requirements **********************
# Host system must have Python 2.7.x installed and
# the following packages must be available:
# - os
# - threading
# - imutils
# - subprocess
# - sys
# - BaseHTTPServer
# - Image
# - StringIO
# - time
# - numpy
# - datetime
# - ntplib (for time-zone synchronization)
#

# ******* import the necessary packages ***********
import argparse
from datetime import datetime
import time
import cv2
import numpy as np
import threading
import logging
import mraa
import Image
import StringIO
from BaseHTTPServer import BaseHTTPRequestHandler,HTTPServer
from os import listdir
import sync_time
import set_server_ip

logging.basicConfig(filename='log', format='%(aSICTme)s %(message)s',
datefmt='%m/%d/%Y %I:%M:%S %p', level=logging.DEBUG)
```

```
# *************** Variables ************************

# dilate kernel size
dilate_kernel = 11

# Defaults
def_Thresh = 30
def_ksize = 5
def_minArea = 500

# Semaphore settings
moving_line = 1
light_text = "GREEN"
green_color = (0, 255, 0)
red_color = (255, 0, 0)
yellow_color = (255, 255, 0)
no_color = (0, 0, 0)
color_1 = green_color
color_2 = no_color
color_3 = no_color
no_motion_text = "No Motion Detected"
motion_text = "Moving object detected"
change_requested = 0
thick_1_a = -1
thick_1_b = 1
thick_1_c = 1
thick_2_a = 1
thick_2_b = 1
thick_2_c = -1

# Camera settings
cameras = []
camera_width = 320
camera_hight = 240
camera_saturation = 0.2

frame_1 = None
frame_2 = None
first_frame_1 = None
first_frame_2 = None

# *************** light groups *********************

# For "N" number of lights, lights Tuples ('Name',gpio#)
# must be supplied in following order:
# "Red1, Yellow1, Green1,...,RedN, YellowN, GreenN"
#
lights_tuples = [ ('redLight1',9),('yellowLight1',5),('greenLight1',11),\
          ('redLight2',8),('yellowLight2',6),('greenLight2',10) ]
lights = []

# ******************* point array *******************
# The following point arrays must be defined accordingly to the actual
road
pts_1 = np.array([[299,0],[0,479],[639,479],[329,0]], np.int32)
pts_2 = np.array([[299,0],[0,479],[639,479],[329,0]], np.int32)

# ******* Time intervales in seconds ***************
# lap_period_sec - Interval between each light change
```

```
lap_period_sec = 10
# yellow_period_sec - Time for the yellow light to be On before Red
yellow_period_sec = 2
# lap_to_go - Remaining time before a light-change event
lap_to_go = lap_period_sec
# last_second - Time when previous cycle ran
last_second = datetime.now().second

def gpio_setup():
    """
    gpio_setup: define gpio pins and set them as output
    """
    global lights
    n = len(lights_tuples)
    # Configure lights as outputs and turn them OFF
    try:
        for l in range(0, n):
            local_light = lights_tuples[l][0]
            local_light = mraa.Gpio(lights_tuples[l][1])
            lights.append(local_light)
            local_light.dir(mraa.DIR_OUT)
            local_light.write(1)
            logging.info('GPIO ' + str(lights_tuples[l][1]) + ' configured for ' +
lights_tuples[l][0] + '.')
        lights[2].write(0)
        lights[3].write(0)
    except:
        logging.error('Cannot configure GPIO for ' + lights_tuples[l][0] + '.')

def turn_off_all_lights():
    global lights
    n = len(lights_tuples)

    try:
        for l in range(0, n):
            lights[l].write(1)
    except:
        logging.error('Cannot turn off all lights.')

def cameras_setup():
    global available_cameras
    try:
        available_cameras = 0
        devs = listdir('/dev/')
        for dev in devs:
            if 'video' in dev:
                available_cameras += 1
        logging.info('Detected ' + str(available_cameras) + ' available
cameras in system.')
    except:
        logging.error('Cannot determine number of available cameras in
system.')

    global cameras

    try:
        for n in range(0, available_cameras):
            thisCam = cv2.VideoCapture(n)
            cameras.append(thisCam)
            time.sleep(0.25)
```

```
            thisCam.set(cv2.cv.CV_CAP_PROP_FRAME_WIDTH,
camera_width);
            thisCam.set(cv2.cv.CV_CAP_PROP_FRAME_HEIGHT,
camera_hight);
            thisCam.set(cv2.cv.CV_CAP_PROP_SATURATION,
camera_saturation);
            logging.info('Configured camera' + str(n+1) + ' size to ' + \
                        str(camera_width) + 'x' + str(camera_hight) + \
                        ' and ' + str(camera_saturation) + ' saturation.')
    except:
        logging.error('Cannot configure camera' + str(n+1) + '
size/saturation.')

class light_change(object):
    """light_change class
    The run()method will be started and
    it will run in the background.
    """
    def __init__(self, interval=yellow_period_sec):
        """ Constructor
        :type interval: int
        :param interval: Change interval in seconds
        """
        self.interval = interval

        global thread
        thread_light_change = threading.Thread(target=self.run, args=())
        thread_light_change.daemon = False
        thread_light_change.start()

    def run(self):
        """Method to change to yellow, then red light"""
        global moving_line
        global change_requested
        global thick_1_a, thick_1_b, thick_1_c, thick_2_a, thick_2_b,
thick_2_c
        global lap_to_go
        global lights
        line = moving_line
        if line == 1:
            thick_1_a = 1
            lights[2].write(1)
            thick_1_b = -1
            lights[1].write(0)
            time.sleep(self.interval)
            thick_1_b = 1
            lights[1].write(1)
            thick_1_c = -1
            lights[0].write(0)
            thick_2_a = -1
            lights[5].write(0)
            thick_2_b = 1
            lights[4].write(1)
            thick_2_c = 1
            lights[3].write(1)
            line = 2
        else:
            thick_2_a = 1
            lights[5].write(1)
            thick_2_b = -1
```

```
            lights[4].write(0)
            time.sleep(self.interval)
            thick_2_b = 1
            lights[4].write(1)
            thick_2_c = -1
            lights[3].write(0)
            thick_1_a = -1
            lights[2].write(0)
            thick_1_b = 1
            lights[1].write(1)
            thick_1_c = 1
            lights[0].write(1)
            line = 1

        moving_line = line
        change_requested = 0
        lap_to_go = lap_period_sec

def nothing(x):
    """
    Do nothing
    """
    pass

def make_odd(number):
    """
    Return the input number if its odd,
    or return input number + 1 if its even or zero.
    """
    if number == 0:
        number += 1
    if number % 2 ==0:
        number += -1
    return number

def light_circles():
    """
    Draw filled circles to simulate semaphore light.
    """
    global frame_1, frame_2
    x_circle_center = int(camera_width/11)
    y_circle_center = int(camera_hight/9)
    circle_radius = int(camera_width*.07)
    cv2.circle(frame_1, (x_circle_center, (circle_radius+(4*circle_radius))),
circle_radius, green_color, thickness=thick_1_a)
    cv2.circle(frame_1, (x_circle_center, (circle_radius+(2*circle_radius))),
circle_radius, yellow_color, thickness=thick_1_b)
    cv2.circle(frame_1, (x_circle_center, circle_radius), circle_radius,
red_color, thickness=thick_1_c)
    cv2.circle(frame_2, (x_circle_center, (circle_radius+(4*circle_radius))),
circle_radius, green_color, thickness=thick_2_a)
    cv2.circle(frame_2, (x_circle_center, (circle_radius+(2*circle_radius))),
circle_radius, yellow_color, thickness=thick_2_b)
    cv2.circle(frame_2, (x_circle_center, circle_radius), circle_radius,
red_color, thickness=thick_2_c)

def road_lines():
    """
    Draw road lines to define valid detection areas.
    """
```

```
    cv2.polylines(frame_1,[pts_1],True,yellow_color)
    cv2.polylines(frame_2,[pts_2],True,yellow_color)

class cam_handler(BaseHTTPRequestHandler):
    def do_GET(self):
        if self.path.endswith('.mjpg'):
            self.send_response(200)
            self.send_header('Content-type','multipart/x-mixed-replace;
boundary=--jpgboundary')
            self.end_headers()

            # loop over the frames of the video
            while True:
                try:
                    global first_frame_1, first_frame_2
                    global frame_1, frame_2
                    # grab the current frame and initialize the "Moving object"
                    # detection message
                    (grabbed_1, frame_1) = cameras[0].read()
                    (grabbed_2, frame_2) = cameras[1].read()
                    text_1 = no_motion_text
                    text_2 = no_motion_text

                    # if one of the frames could not be grabbed, then we have
reached the end
                    # of the video
                    if not (grabbed_1 and grabbed_2):
                        break

                    # Draw three circles to simulate a semaphore
                    light_circles()

                    # resize the frame, convert it to grayscale, and blur it
                    ksize = make_odd(def_ksize)
                    gray_1 = cv2.cvtColor(frame_1, cv2.COLOR_BGR2GRAY)
                    gray_2 = cv2.cvtColor(frame_2, cv2.COLOR_BGR2GRAY)
                    gray_1 = cv2.GaussianBlur(gray_1, (ksize, ksize), 0)
                    gray_2 = cv2.GaussianBlur(gray_2, (ksize, ksize), 0)

                    # if the first frame is None, initialize it
                    if first_frame_1 is None:
                        first_frame_1 = gray_1
                        first_frame_2 = gray_2
                        continue

                    # compute the absolute difference between the current
frame and
                    # first frame
                    Thresh = def_Thresh
                    frame_delta_1 = cv2.absdiff(first_frame_1, gray_1)
                    frame_delta_2 = cv2.absdiff(first_frame_2, gray_2)
                    thresh1 = cv2.threshold(frame_delta_1, Thresh, 255,
cv2.THRESH_BINARY)[1]
                    thresh2 = cv2.threshold(frame_delta_2, Thresh, 255,
cv2.THRESH_BINARY)[1]
                    # use current frame for next iteration comparisson
                    first_frame_1 = gray_1
                    first_frame_2 = gray_2
```

```
                # dilate the thresholded image to fill in holes, then find
contours
                # on thresholded image
                thresh1 = cv2.dilate(thresh1, np.ones((dilate_kernel
,dilate_kernel)),\
                 iterations=2)
                thresh2 = cv2.dilate(thresh2, np.ones((dilate_kernel
,dilate_kernel)),\
                 iterations=2)
                (cnts1, _) = cv2.findContours(thresh1.copy(),
cv2.RETR_EXTERNAL,\
                 cv2.CHAIN_APPROX_SIMPLE)
                (cnts2, _) = cv2.findContours(thresh2.copy(),
cv2.RETR_EXTERNAL,\
                 cv2.CHAIN_APPROX_SIMPLE)

                # loop over the contours
                for c1 in cnts1:

                    # if the contour is too small, ignore it
                    if cv2.contourArea(c1) < args["min_area"]:
                        continue

                    # compute the bounding box for the contour, draw it on
the frame,
                    # and update the text
                    (x, y, w, h) = cv2.boundingRect(c1)
                    cv2.rectangle(frame_1, (x, y), (x + w, y + h), (0, 255, 0), 2)
                    text_1 = motion_text

                for c2 in cnts2:

                    # if the contour is too small, ignore it
                    if cv2.contourArea(c2) < args["min_area"]:
                        continue

                    # compute the bounding box for the contour, draw it on
the frame,
                    # and update the text
                    (x, y, w, h) = cv2.boundingRect(c2)
                    cv2.rectangle(frame_2, (x, y), (x + w, y + h), (0, 255, 0), 2)
                    text_2 = motion_text

                # draw the motion-detection message on the frame
                cv2.putText(frame_1, "{}".format(text_1), (60, 20),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 2)
                cv2.putText(frame_2, "{}".format(text_2), (60, 20),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 2)

                # Draw the timesatmp on the frame
                cv2.putText(frame_1, datetime.now().strftime("%A %d %B %Y
%I:%M:%S%p"),\
                 (10, frame_1.shape[0] - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
(0, 0, 255), 1)
                cv2.putText(frame_2, datetime.now().strftime("%A %d %B %Y
%I:%M:%S%p"),\
                 (10, frame_2.shape[0] - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
(0, 0, 255), 1)

                # Draw the remaining time before next light-change
```

```
            # and triger the light-change when lap_period_sec is
completed
            # only once per second
            global lap_to_go
            global last_second
            this_second = datetime.now().second
            if not last_second == this_second:
                global change_requested
                if lap_to_go <= 0:
                    light_change()
                    lap_to_go = lap_period_sec
                if moving_line == 1:
                    if change_requested == 0:
                        if text_2 == no_motion_text:
                            if lap_to_go <= lap_period_sec:
                                lap_to_go +=1
                        else:
                            change_requested = 1
                else:
                    if change_requested == 0:
                        if text_1 == no_motion_text:
                            if lap_to_go <= lap_period_sec:
                                lap_to_go +=1
                        else:
                            change_requested = 1
                lap_to_go -= 1

            if moving_line == 1:
                if lap_to_go < lap_period_sec:
                    cv2.putText(frame_1, "{}".format(lap_to_go), (80, 65),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, yellow_color, 2)
            else:
                if lap_to_go < lap_period_sec:
                    cv2.putText(frame_2, "{}".format(lap_to_go), (80, 65),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, yellow_color, 2)

            width_1=len(frame_1[0,:])
            heigth1=len(frame_1[:,0])
            total_bytes = width_1 * heigth1 * 6

            total_array = bytearray(total_bytes)
            byte_array = np.array(total_array)
            merged_frame = byte_array.reshape(heigth1, (width_1*2), 3)
            merged_frame[0:heigth1, 0:width_1]=frame_1
            merged_frame[0:heigth1, width_1:(width_1*2)]=frame_2

            jpg = Image.fromarray(merged_frame)
            tmp_file = StringIO.StringIO()
            jpg.save(tmp_file,'JPEG')
            self.wfile.write("--jpgboundary")
            self.send_header('Content-type','image/jpeg')
            self.send_header('Content-length',str(tmp_file.len))
            self.end_headers()
            jpg.save(self.wfile,'JPEG')
            time.sleep(0.05)

            last_second = this_second
        except KeyboardInterrupt:
            break
    return
```

```
        if self.path.endswith('.html'):
            self.send_response(200)
            self.send_header('Content-type','text/html')
            self.end_headers()
            self.wfile.write('<html><head></head><body>')
            mjpeg_source="http://%s:8080/cam.mjpg" %serverIp
            print "I: cam.mjpeg sourced from %s" %mjpeg_source
            self.wfile.write('<img src=%s>' %mjpeg_source)
            self.wfile.write('</body></html>')
            return

def main():
    # construct the argument parser and parse the arguments
    ap = argparse.ArgumentParser(description="Traffic light control using
object detection.")
    ap.add_argument("-a", "--min-area", type=int, default=def_minArea,
help="minimum area size")
    global args
    args = vars(ap.parse_args())

    # ********* System setup **************************
    gpio_setup()

    sync_time.run()
    cameras_setup()
    global serverIp
    serverIp = set_server_ip.run()

    global frame_1
    global frame_2

    # initialize the first frame in the video stream
    first_frame_1 = None
    first_frame_2 = None

    try:
        server = HTTPServer(('',8080),cam_handler)
        print "I: Server started"
        server.serve_forever()
    except KeyboardInterrupt:
        cameras[0].release()
        cameras[1].release()
        server.socket.close()
        turn_off_all_lights()

if __name__ == '__main__':
    main()
```

set_server_ip.py

```
import logging
from subprocess import check_output
logging.basicConfig(filename='log', format='%(aSlCTme)s %(message)s',
                    datefmt='%m/%d/%Y %I:%M:%S %p', level=logging.DEBUG)
def run():
    try:
        server_ip = check_output(
            "ifconfig wlan0 | awk \'/t addr:/{gsub(/.*:/,\"\",$2);print$2}\'",
            shell=True)
        server_ip = server_ip.replace("\n", "")
```

```
        logging.info('Server IP defined as: ' + str(server_ip))
        return server_ip
    except:
        logging.warning('Server IP cannot be defined.')
```

sync_time.py

```
import logging
import ntplib
from time import strftime, localtime
from os import system
logging.basicConfig(
    filename='log', format='%(aSICTme)s %(message)s',
    datefmt='%m/%d/%Y %I:%M:%S %p', level=logging.DEBUG)

def run():
    try:
        client = ntplib.NTPClient()
        response = client.request('north-america.pool.ntp.org')
        system('date ' +
              strftime('%m%d%H%M%Y.%S', localtime(response.tx_time)))
        logging.info(
            'Time synchronized with north-america.pool.ntp.org server.')
    except:
        logging.warning('Could not synchronize local time with ntp server.')
```

ANEXO B: CONSTANCIAS DE EVENTOS

Conferencia de Profesionales de Software (SWPC) de Intel

intel SOFTWARE PROFESSIONALS CONFERENCE 2016

Software Empowering Intel

Jose A. Lamego

"Early prototyping an Intelligent Traffic Control System using Yocto Project and OpenCV"

In recognition for being one of the few shortlisted presenters at **SWPC**

Signed: Douglas Fisher

Date: October 6th 2016, Zapopan, Jalisco, México

Douglas Fisher
Senior Vice President and General Manager
Software and Services Group

Figura 17: Reconocimiento de la Conferencia SWPC

Marzo 6 2017

A quien corresponda

Hacemos constar que el prototipo "Sistema Inteligente de Control de Tráfico" (SICT) desarrollado por el Ingeniero José Andrés Lamego Castro fue presentado durante la edición 2016 del "Open House" del Centro de Tecnología de Código Abierto (OTC) de Intel Guadalajara, donde sirvió como muestra de implementación de las herramientas del proyecto Yocto.
Extendemos la presente para que pueda servir para los propósitos administrativos necesarios para la realización de su posgrado.

Sin más por el momento quedo a sus órdenes.

Cesar Lara
Software Engineering Manager
OTC site manager
Cesar.lara@intel.com
+523316452189

Intel Corporation
Av.Del bosque #1001
Mexico, Jalisco,
CP.45019

Figura 18: Constancia del "Open House" de OTC

Marzo 13, 2017

A quien corresponda:

Hacemos constar que el prototipo "Sistema Inteligente de Control de Tráfico" (SICT) desarrollado por el Ingeniero José Andrés Lamego Castro se encuentra incluido como parte de la exposición permanente del Laboratorio para Pequeñas y Medianas empresas (PyMES) del Centro de Diseño Intel Guadalajara desde Abril de 2016 con el objetivo de mostrar las capacidades de un sistema basado en software de código abierto empleando tecnología de Intel, así como de servir de modelo e inspiración para el desarrollo de sistemas similares.

Extiendo la presente para manifestar la satisfacción con el desempeño del prototipo SICT y para que pueda servir para los propósitos administrativos necesarios para la realización de su posgrado.

Abraham Arce M.

Abraham Arce Moreno
Innovation Technical Manager
Intel
Avenida del Bosque #1001, Colonia El Bajío
Zapopan, JAL 45019
abraham.arce.moreno@intel.com
+52 3331492168
+52 1 3331492791

Figura 19: Constancia del Laboratorio PyMES

Evento de inicio del Programa de Capacitación Internet de las Cosas en MIND

Marzo 13, 2017

A quien corresponda:

Hacemos constar que el prototipo "Sistema Inteligente de Control de Tráfico" (SICT) desarrollado por el Ingeniero José Andrés Lamego Castro fue presentado durante el evento de inicio del Programa de Capacitación Internet de las Cosas en las instalaciones de Mexico Innovacion y Diseño (MIND), donde se detalló el principio de funcionamiento del SICT, los requerimientos generales para el desarrollo de un sistema similar, y referencias técnicas para los interesados en incorporar proyectos de software de código abierto en sus respectivas industrias. El prototipo captó la atención de los asistentes solicitando a Intel posteriormente que abordaramos en los entrenamientos temas propuestos por el prototipo, empezando por cómo participar en comunidades de código abierto, Linux Embebido, el subsistema de Cámara y finalmente OpenCV como librería de visión por computadora.

Extendemos la presente para que pueda servir para los propósitos administrativos necesarios para la realización de su posgrado.

Abraham Arce Moreno
Innovation Technical Manager
Intel
Avenida del Bosque #1001, Colonia El Bajío
Zapopan, JAL 45019
abraham.arce.moreno@intel.com
+52 3331492168
+52 1 3331492791

Figura 20: Constancia del Programa de Capacitación Internet de las Cosas

ANEXO C: FOTOGRAFÍAS DE EVENTOS DE PRESENTACIÓN DEL SICT

Stand del SICT en el OpenHouse de OTC

Figura 21: Presentación del SICT en el OpenHouse de OTC

Figura 22: Presentación del SICT en el OpenHouse de OTC

Figura 23: Presentación del SICT en la conferencia SWPC