UNIVERSIDAD SANTO TOMÁS
PRIMER CLAUSTRO UNIVERSITARIO DE COLOMBIA

DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO ADAS CON COMUNICACIÓN IOT (XBEE) Y RECOLECCIÓN DE IMÁGENES PARA DETECCIÓN DE SEUDOACCIDENTES

Paula Natalia Rojas Machuca

Cristian Andrés Salas Rodríguez

UNIVERSIDAD SANTO TOMÁS

FACULTAD DE INGENIERÍA ELECTRÓNICA

TUNJA, BOYACÁ

2020

UNIVERSIDAD SANTO TOMÁS
PRIMER CLAUSTRO UNIVERSITARIO DE COLOMBIA

DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO ADAS CON COMUNICACIÓN IOT (XBEE) Y RECOLECCIÓN DE IMÁGENES PARA DETECCIÓN DE SEUDOACCIDENTES

Paula Natalia Rojas Machuca

Cristian Andrés Salas Rodríguez

Trabajo de grado para optar por el título de

INGENIERO ELECTRÓNICO

Director

Ing. Cesar Mauricio Galarza Bogotá

Codirector

Ing. Luis Fernando Castellanos Guarín

UNIVERSIDAD SANTO TOMÁS

FACULTAD DE INGENIERÍA ELECTRÓNICA

TUNJA, BOYACÁ

2020

Los autores se hacen responsables de las ideas expresadas en este documento con la intención de exonerar a la Universidad Santo Tomás Seccional Tunja y a la Facultad de Ingeniería Electrónico de cualquier responsabilidad.

Tunja, 2020

NOTA DE ACEPTACIÓN

_____

_____

_____

_____

_____

_____

Firma del Jurado

_____

Firma del Jurado

_____

Firma del Director

_____

Firma del Codirector

Tunja, noviembre 2020

DEDICATORIA

A Dios principalmente por haberme dado la vida y permitirnos haber llegado hasta la culminación de este trabajo, a mis padres Jackeline y Henry por su amor y esfuerzo para darme estudio, a mis hermanos Laura y Mauricio por brindarme su apoyo en todas mis decisiones y finalmente pero no menos importantes a mis sobrinos por darme alegría para continuar cada día.

Paula Natalia Rojas Machuca

Dedico este trabajo a mi madre y mi tía que siempre han estado conmigo brindándome su amor, a mis hermanos por brindarme su sabiduría y finalmente a mi novia que siempre me ha brindado su consejo y apoyo en las decisiones, ayudándome a seguir cada día.

Cristian Andrés Salas Rodríguez

AGRADECIMIENTOS

En primer lugar, queremos agradecer a Dios por guiarnos siempre en la realización de nuestros proyectos, darnos fortaleza, salud y vida para terminar este trabajo.

Gracias a nuestros padres: Jackeline - Henry, y, Bertha, por siempre estar para nosotros, llenarnos de amor y comprensión, y por darnos la oportunidad de realizar nuestros estudios superiores.

De igual manera agradecemos a nuestro director de tesis el Ing. Cesar Mauricio Galarza Bogotá por la ayuda recibida para el desarrollo de este proyecto, y a nuestro codirector el Ing. Luis Fernando Castellanos Guarín por las tutorías brindadas.

Agradecemos a nuestros profesores quienes nos guiaron en el aprendizaje, las enseñanzas y los consejos ofrecidos durante toda la carrera.

A la Universidad Santo Tomas Seccional Tunja por tener a nuestro alcance, los conocimientos y las herramientas necesarias para la realización del proyecto.

Para finalizar agradecemos a nuestros familiares y amigos por la compañía, apoyo y buenos deseos en alcanzar este logro.

Tabla de Contenidos

Lista de Figuras

Lista de Tablas

ABSTRAC

This work presents the Design and Implementation of an ADAS Prototype with IoT Communication (XBee) and Image Collection for Pseudo-Accident Detection, which is composed of a Raspberry-type processing unit, Inertial sensor, GPS Module, camera and a communication system. type XBee. With the purpose of acquiring and storing images that allow in a next stage to identify the type of vehicle that is in the back of the car and what speed it is carrying. To meet the objective, an algorithm was implemented which allows us to process the data obtained from the inertial sensor (acceleration, magnetometer and gyroscope) to determine the speed in the three axes, the direction in which it is moving and the rotation data. of the vehicle that, added to a filter, provides a backup in case of loss of the GPS signal (tunnels, poor reception, etc.). The processing, sending of data, image capture and visualization of the graphical interface are carried out simultaneously so that the processing is done efficiently, in the same way the base station will be able to receive the data and display it correctly. Finally, the capture and storage of the images, the data of the GPS module and inertial sensor will be saved in a local database and in the base station that communicates through the XBee modules.

RESUMEN

En este trabajo se presenta el Diseño e Implementación de un Prototipo ADAS con Comunicación IoT (XBee) y Recolección de Imágenes para Detección de Seudoaccidentes que está compuesto por una unidad de procesamiento tipo Raspberry, sensor Inercial, Modulo GPS, cámara y un sistema de comunicación tipo XBee. Con el propósito de realizar la adquisición y almacenamiento de imágenes que permitan en una siguiente etapa identificar el tipo de vehículo que está en la parte posterior del carro y que velocidad lleva este. Para cumplir con el objetivo se implementó un algoritmo el cual nos permite el procesamiento de los datos obtenidos del sensor inercial (aceleración, magnetómetro y giroscopio) para determinar la velocidad en los tres ejes, el sentido en que se está moviendo y los datos de rotación del vehículo que sumado a un filtro brinda un respaldo en caso de pérdida de la señal de GPS (túneles, mala recepción, etc.). El procesamiento, envió de datos, captura de las imágenes y visualización de la interfaz gráfica, se realizan en simultaneo para que el procesamiento se haga de manera eficaz, de igual manera la estación base podrá recibir los datos y visualizarlos correctamente. Finalmente, la captura y almacenamiento de las imágenes, los datos del moduló GPS y sensor inercial se guardaran en una base de datos local y en la estación base que se comunica por medio de los módulos XBee.

GLOSARIO

ADAS: Sistema Avanzado de Asistencia al Conductor, es un mecanismo que permite mejorar la seguridad del vehículo en el momento de la conducción y poder minimizar los riesgos de sufrir un accidente o colisión.

Accidente: Suceso no planeado y no deseado que provoca un daño, lesión u otra incidencia negativa sobre un objeto o sujeto.

Acelerómetro: Instrumento destinado a medir aceleraciones.

Airbag: También conocido como bolsa de aire es un elemento de seguridad pasiva del vehículo.

Colisión: Encuentro violento de dos o más cuerpos, de los cuales al menos uno está en movimiento.

Dispositivos: Aparato o mecanismo que desarrolla determinadas acciones.

Fotograma: Imagen cinematográfica considerada aisladamente.

Giroscopio: Dispositivo mecánico que sirve para medir, mantener o cambiar la orientación en el espacio de algún aparato o vehículo.

Hardware: Conjunto de elementos físicos o materiales que constituyen una computadora o un sistema informático.

Infraestructura vial: todo conjunto de elementos que permite el desplazamiento de vehículos en forma confortable y desde un punto a otro.

Magnetómetro: Dispositivo que sirve para cuantificar en fuerza o dirección la señal magnética de una muestra.

Medición inercial: Dispositivo electrónico que mide e informa acerca de velocidad, orientación y fuerzas gravitacionales de un aparato, usando una combinación de acelerómetros, giroscopios y magnetómetros.

Módulos: Estructura o bloque de piezas que, forma parte de un sistema y suele estar conectado de alguna manera con el resto de los componentes.

Monocular: De un solo ojo.

Pretensores: Dispositivo que, en caso de un choque frontal, compensa el alargamiento inevitable de los cinturones bajo la acción del cuerpo.

Prototipo: Primer ejemplar que se fabrica de una figura, un invento u otra cosa y que sirve de modelo para fabricar otras iguales, o molde original con el que se fabrica.

Resolución: Menor variación que se puede detectar en la cantidad que se está midiendo. Numero de pixeles que puede mostrar una imagen.

Satélite: Objeto que ha sido puesto en órbita intencionalmente.

Seudoaccidente: Posible accidente.

Siniestro vial: Evento generalmente involuntario, generado al menos por un vehículo en movimiento que causa daños a personas o bienes involucrados.

Sistema de frenos antibloqueo: Dispositivo utilizado en automóviles, aviones y en modelos avanzados de motocicletas que hace variar la fuerza de frenado para evitar que los neumáticos resbalen sobre el suelo en el proceso de frenado.

Sistema embebido: Es un sistema de computación diseñado para realizar una o algunas pocas funciones dedicadas, frecuentemente en un sistema de computación en tiempo real.

Software: Conjunto de programas y rutinas que permiten a la computadora realizar determinadas tareas.

I. INTRODUCCIÓN

El proyecto de investigación Diseño e Implementación de un Prototipo ADAS con Comunicación IoT (XBee) y Recolección de Imágenes para Detección de Seudoaccidentes, se realiza con el fin de dar solución a una problemática que afecta a los conductores, peatones y demás sistemas de movilidad que se pueden encontrar en las vías de Colombia, debido a que se presentan accidentes de tránsito, causados en su mayoría por los siguientes motivos: exceso de velocidad, no respetar normas y señales de tránsito o no mantener la distancia segura. Se implementó el prototipo con elementos como una Raspberry, un GPS, un sensor inercial, cámara y un XBee para la detección de posibles seudoaccidentes.

El trabajo se desarrolló por fases utilizando un modelo de cascada el cual presenta un orden riguroso donde el inicio de cada fase debe esperar a la finalización de la fase anterior; también se manejó una metodología mixta por el diseño e implementación de un hardware y una metodología cuantitativa respecto a los datos obtenidos de los sensores.

Como objetivo el proyecto propone el diseño e implementación de un prototipo ADAS que permita detectar seudoaccidentes y por medio de un algoritmo predecir la ruta del vehículo cuando el GPS pierda señal, con una interfaz gráfica para que el usuario pueda mirar las gráficas de los sensores, se guarda en bases de datos los valores de los sensores y se genera un archivo kml con el cual se puede monitorear la ruta realizada por el automóvil.

Se espera que mediante la implementación adecuada de este proyecto, los inconvenientes que se presenten en la movilidad de la ciudad de Tunja puedan ser minimizados y a la vez, los índices de accidentalidad se reduzcan considerablemente para bien de toda la comunidad.

II. JUSTIFICACIÓN

El estado colombiano es permisivo con los fabricantes de vehículos al no exigirle la implementación (lo cual aumentaría los precios) de sistemas básicos de seguridad desde fábrica, lo cual se relaciona directamente con la tasa de accidentalidad. Como lo muestra el informe de siniestros viales de 2019 del Observatorio Nacional de Seguridad Vial [1]: Hipótesis de causa probable. En este informe se describe que, de los 180.373 accidentes de tránsito en el 2019, 33.429 fueron por desobediencia a las señales de tránsito, 27.963 por no mantener la distancia de seguridad y 2.824 ocurrieron por exceso de velocidad. Según la ANSV (Agencia Nacional de Seguridad Vial) [2], para 2019 en Colombia fallecieron 6.634 personas en accidentes de tránsito, cifra que aumento a la del 2018 que fue de 6.476.

Con base en lo anterior, se evidencia la necesidad de diseñar un sistema de asistencia al conductor que sea adaptable a todo tipo de automóvil y con un precio accesible, con la intención de disminuir el índice de accidentalidad y mortalidad en las vías por imprudencia de los conductores. Las respectivas pruebas del prototipo se realizarán en la ciudad de Tunja con vehículos de prueba, pero con la posibilidad de una implementación en etapas posteriores en la ciudad de Bucaramanga debido a que la infraestructura vial está en un estado óptimo y la señalización de tránsito de la vía está ubicada satelitalmente de tal forma que es fácil de monitorear.

De esta manera se tiene previsto empezar el desarrollo de un prototipo de hardware ADAS teniendo en cuenta las causas más significativas de los accidentes viales, para esto en el diseño del sistema se requiere utilizar un GPS el cual permite determinar la ubicación del vehículo, un sensor inercial para tener la orientación del vehículo como también calcular la velocidad de desplazamiento de este, una cámara ubicada en la parte posterior del vehículo (con la cual se quiere determinar la distancia a la cual está el vehículo que esté detrás; esta parte no se ejecuta en este

proyecto pero es el enfoque del diseño de este hardware) y un módulo IoT (XBee) para él envió remoto y almacenamiento de los datos recolectados.

III. PLANTEAMIENTO DEL PROBLEMA

1. Formulación de preguntas

- ¿Qué variable o alerta se pueden controlar o generar para prevenir accidentes?
- ¿Qué sistemas ADAS existen en el mercado que se puedan adquirir fácilmente y a bajo costo?
- ¿Qué ventajas y desventajas tiene la implementación de un prototipo ADAS para la prevención de accidentes?
- ¿Cuántos accidentes automovilísticos se presentan anualmente en las vías del país?

2. Definición del problema

El parque automotor registrado en el RUNT (Registro Único Nacional de Transito) para junio del 2020 es de 15.671.961, del cual el 41% son vehículos [3], y las 10 referencias que más se venden en Colombia [4] tiene una baja calificación en seguridad para los conductores y pasajeros según el Programa de Evaluación de Autos Nuevos para América y el Caribe (Latín NCAP) [5], esta asociación se dedica a realizar pruebas para evaluar la seguridad de los modelos de automóviles disponibles en el mercado y les da una clasificación entre 0 y 5 estrellas donde 0 es la calificación más baja en seguridad y 5 se considera un vehículo seguro [5].

Las 10 referencias mencionadas son el 47% del nuevo parque automotor y tienen calificaciones muy bajas (0 a 3) en cuanto a seguridad (Tabla 1) para los conductores y pasajeros, adicionalmente el Ministerio de transporte en el Anuario Estadístico Transporte en Cifras - Estadísticas 2018 [6] el 21.4% del parque automotor activo fue adquirido antes del 2014. Y que el 28.2% fueron matriculados antes del 2002, según este mismo documento en el 2018 se presentaron 181.374 accidentes en carretera de los cuales se registraron 6.476 muertes.

REFERENCIA	# VENTAS EN COLOMBIA 2018-08 - 2019-08	%	CLASIFICACIÓN SEGÚN NCAP (0-5)
KIA PICANTO	14295	4,5%	0
RENAULT SANDERO	14133	4,4%	1
RENAULT LOGAN	13855	4,3%	1
RENAULT DUSTER	11252	3,5%	4
RENAULT STEPWAY	11585	3,6%	2
CHEVROLET SPARK	14767	4,7%	0
MAZDA 2	9018	2,8%	2
CHEVROLET ONIX	7914	2,4%	0
CHEVROLET BEAT	8930	2,8%	0
RENAULT KWID	4370	1,3%	3
Total de vehículos vendidos 2019	**316712**		

Tabla 1: Referencia de los 10 vehículos más vendidos y su clasificación NCAP. Fuente: Autores.

De acuerdo con el reporte de la Asociación Colombiana de Vehículos Automotores (ANDEMOS), ninguno de los vehículos mencionados cuenta con un sistema básico de seguridad (Airbag frontal para el conductor, Airbags laterales para el cuerpo, Pretensores del cinturón delantero del pasajero, Sistema de frenos antibloqueo, Airbag frontal para el pasajero delantero, Airbag para la rodilla del conductor, Sistema de aviso de cinturones, Airbags laterales para la cabeza, Pretensores del cinturón delantero, ISOFIX- Sistema de sujeción para sillas de seguridad para niños) presentado un alto riesgo de pérdidas humanas y económicas en accidentes de tránsito [4].

3. Delimitación del problema

- Delimitación Geográfica: Se tiene previsto que el proyecto se implemente en la ciudad de Tunja dado que esta ciudad cuenta con vías apropiadas, pero con una proyección a futuro en etapas siguientes a la ciudad de Bucaramanga debido a que esta ciudad tiene una de las mejores infraestructuras viales del país, contando también con la ubicación satelital de cada una de las señales de tránsito, cámaras, reductores de velocidad entre otros.
- Delimitación Temporal: El tiempo de desarrollo del proyecto está provisto para un año y dos meses, iniciando en agosto del 2019 y finalizando en octubre del 2020.
- Delimitación de la Población: La implementación del proyecto se llevará a cabo en algunos vehículos de la ciudad de Tunja para probar el desempeño del prototipo.
- Delimitación Temática: El proyecto consta del diseño del hardware para un prototipo de asistencia al conductor el cual alerte sobre posibles accidentes (seudoaccidentes) o factores que puedan causar un accidente, con el objetivo de mejorar la movilidad en la ciudad de Tunja y disminuir los índices de accidentalidad en las vías de esta ciudad.

IV. OBJETIVOS

1. Objetivo General

Diseñar e implementar el hardware para un prototipo de sistema de asistencia al conductor (ADAS), compuesto por una unidad de procesamiento tipo Raspberry, un sensor Inercial, GPS, Cámara y un sistema de comunicación tipo XBee (IoT). Para la adquisición de imágenes con un punto de referencia que permitan en una siguiente etapa identificar el tipo de vehículo que está detrás y la velocidad que este lleva.

2. Objetivos Específicos

- Analizar el estado del arte de sistemas ADAS, embebidos y tecnologías IoT.
- Diseñar e implementar un prototipo de hardware de ADAS con un sistema embebido tipo Raspberry, sensores (Inercial, GPS, cámara) y módulos IoT (XBee) para comunicación. Con una interfaz visual que permita ver los datos de los sensores.
- Implementar el prototipo de hardware de ADAS diseñado en un vehículo.
- Diseñar e implementar una unidad de recepción de los datos generados por el sistema ADAS con comunicación XBee. En esta unidad se debe visualizar los datos de los sensores y el recorrido del vehículo.
- Realizar pruebas de validación del prototipo de hardware ADAS, diseñado e implementado en el vehículo. Que permite la adquisición y almacenamiento de imágenes obtenidas de la cámara y la información proveniente de los diferentes sensores.

V. MARCO TEÓRICO

1. Marco histórico

Para la concepción del marco histórico se realizará una breve descripción acerca de las partes más importantes de este proyecto:

Global Position System (GPS)

Los inicios del funcionamiento de GPS nacen desde 1957. Cuando la Unión Soviética monitoreaba el satélite Sputnik I gracias al efecto Dopply se pensó que de la misma forma se podía conocer la posición del observador [7].

El primer sistema de navegación creado fue el TRANSIT, este permitía determinar la posición del observador conociendo la posición del satélite, la creación del mismo inicio en 1958 y se declaró un dispositivo operacional en 1964 tras el lanzamiento de 10 satélites [8]. Estimulados por el buen funcionamiento del sistema TRANSIT, el departamento de defensa de los Estados Unidos inicio los estudios de un sistema más avanzado, así fue como en 1973 se comenzó a trabajar en NAVSTAR-GPS (Figura 1) este sistema proporciona a los usuarios que dispongan de dispositivos receptores la información de su ubicación con una alta precisión en cualquier parte del mundo y a cualquier hora, este sistema permite un número ilimitado de usuarios [9], la implementación definitiva fue en 1992 dando origen a la mayoría de sistemas GPS que existen en la actualidad.

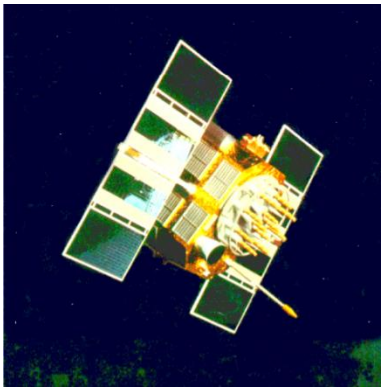

Figura 1: Satélite del sistema NAVSTAR_GPS. Fuente: [8].

Sensor Inercial

El desarrollo de los primeros sensores inerciales tuvo lugar con la invención del giroscopio en el siglo XIX por León Foucault, que al intentar demostrar la rotación de la tierra logro este hallazgo [10].

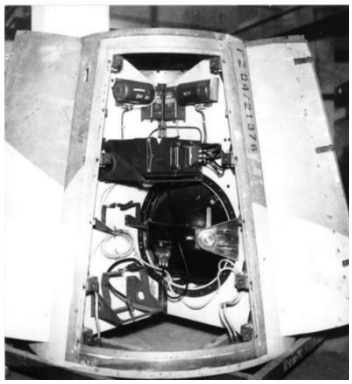

Figura 2: Diseños para misiles v-2 finales de 1941. Fuente: [10].

En la segunda guerra mundial se emplearon los primeros giróscopos con acelerómetros los cuales tenían como objetivo ayudar a despegar, volar y llegar al objetivo a los misiles balísticos V-2 (Figura 2) sin la intervención humana [11]. Después de la terminación de la segunda guerra mundial el desarrollo de los dispositivos tuvo un rápido desarrollo hasta 1965 que se patento a idea del INS (Inertial Navigation System) “strapdown”. Este sistema está fijado al cuerpo (objeto), sin embargo, este necesitaba una gran capacidad de cómputo y sensores precisos por las vibraciones del cuerpo, la capacidad de cómputo se pudo suplir hasta los años 70.

XBee

La empresa creadora de los dispositivos XBee fue fundada en 1985 como Digiboard y se hizo pública en 1989 como digi international, dando inicio a los primeros dispositivos enfocados al (IoT) [12], Los primeros módulos fueron introducidos en 2005 y se basaron en el estándar 802.15.4-2003 diseñado para una comunicación punto a punto y comunicación en topología de estrella.

Raspberry

La idea surgió en 2006 pero fue lanzado hasta febrero de 2012 por un grupo de la Universidad de Cambridge con el objetivo de fomentar las ciencias de la computación.

Los primero diseños se basaban en el microprocesador, el primer modelo en ser lanzado fue Raspberry pi 1 model A (Figura 3) esta tenía 26 conectores GPIO, salida de video vía HDMI y video RCA, un conector Jack de 3,5 milímetros, un puerto USB y un micro USB para la alimentación [13], luego de este se realizó una revisión del mismo adicionándole unos 14 pines GPIO, 3 puertos USB y un puerto ethernet.

Figura 3: Raspberry Pi 1 Modelo A+. Fuente: [13].

Luego salió la Raspberry pi 2 modelo B que entre sus mejoras cuenta con un CPU 900MHz quad-core ARM Cortex-A7 y 1 GB RAM, sin embargo, las mayoras mejoras se hicieron con la Raspberry pi 3 model B+, donde se utilizó CPU broadcom BCM2837B0, Cortex-a53(ARMV8)64-bit@ 1.4Ghz, 1 GB RAM LPDDR2 SDRAM y se le agrego conexión inalámbrica 2.4Ghz y 5Ghz IEEE 802.11.b/g/ac, bluetooth 4.2 y 4 puertos USB 2.0.

La última versión Raspberry pi 4 cuenta con un CPU BCM2711, Quad core Cortex-A72 (ARM V8) 64-bit 1.5Ghz en tres modelos disponibles (1GB,2GB o 4 GB) LPDDR4-3200 SDRAM, 2 puertos 2.0,2 puertos 3.0 adicionalmente cuenta con dos puertos micro HDMI que soportan 60 fps, y con conexión inalámbrica de 2.4 GHz y 5.0 GHz IEEE 802.11ac, Bluetooth 5.0, BLE. Figura 4.

Figura 4: Raspberry Pi 4 Modelo B. Fuente: [14].

Cámara

La primera cámara de la historia fue desarrollada por Joseph Nicephonre Niepce haciendo uso del principio de funcionamiento de cámara oscura (instrumento óptico que permite obtener una proyección plana de una imagen sobre una superficie formando la imagen en la película o en sensor digital) [15], en el cual necesito 8 horas de exposición a la luz en una plancha de peltre recubierta de betún de Judea, donde las partes afectadas por la luz se volvían insolubles, después se bañaba en un disolvente de aceite, separando las partes del barniz afectadas por la luz [16] generando imágenes como la mostrada en la Figura 5.

Figura 5: Primera fotografía de la historia 1825. Fuente: [17].

En 1988 gracias a la empresa kodak salió al mercado la primera cámara para aficionados marcando un antes y después en la fotografía, la cual utilizaba un rollo para realizar hasta 100 fotografías. A partir de ese momento se generaron muchos avances en la tecnología de adquisición de imágenes hasta llegar a las cámaras digitales actuales, por ejemplo, cámara GoPro Hero 8 que cuenta con una resolución de 13 MP y la capacidad de grabar a 4k 60 fotogramas por segundo [18].

Las cámaras no solo se han utilizado como elemento de entretenimiento; se han dado distintas aplicaciones en el ámbito científico aprovechando su capacidad de capturar momentos en tiempo real. La cámara para Raspberry PI 1080p HD y set 2 piezas luz led 3W IR que nos permite capturar momentos en una resolución de 1080 p y el sensor de 5 mp acompañado del set de 2 luz led IR que nos permite tomar imágenes en la oscuridad [19]. Figura 6.

Figura 6: Cámara Raspberry pi 5Mp con led IR. Fuente: [20].

ADAS

El concepto ADAS (Sistemas avanzados de asistencia al conductor) surge con los primeros esfuerzos por desarrollar dispositivos que dieran solución a los problemas causados por los automóviles y el tránsito de los mismo, las primeras ideas surgen partir de proyectos denominados PROTEHUS, IVHS y SSVS [21]. Donde los dispositivos se basaban en una infraestructura inteligente, logística inteligente y que contarían con visión artificial, con una proyección de 20 a 30 años partir del primer pronunciamiento de esta idea en 1989. A partir de ese momento iniciaron

los proyectos encaminados a dar solución a problemáticas causado por el tránsito de los automóviles. Figura 7.

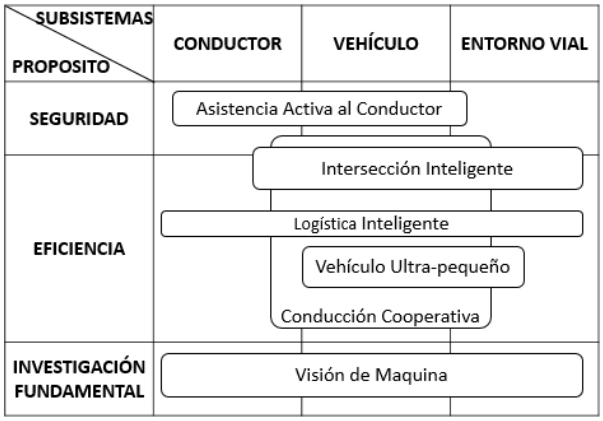

Figura 7: Soluciones del sistema SSVS. Fuente: [21].

En 1994 se iniciaron los esfuerzos por lograr un sistema de visión artificial con análisis de imágenes consecutivas monoculares en escenas de tráfico [22], donde lo que se buscaba era un sistema de visión de apoyo al conductor, sin embargo, con este sistema de apoyo solo se obtenía los tipos de vehículos que se encuentran en la vía, pero no los riesgos de colisión que se presentaban en la misma, en años posteriores se lograron sistemas de asistencia al conductor que funcionaban por medio de arreglos de dos cámaras para generar un arreglo con la máxima robustez en el análisis de escenas en 3D [23].

Todos los sistemas que se desarrollan como sistema ADAS deben ser capaces de detectar obstáculos con alta confiabilidad incluso en condiciones de mal tiempo y poca visibilidad, en 1998 se inició el desarrollo de sistema que hacía uso de sensores tipo radar [24] que permitían al usuario un control inteligente de velocidad de crucero y una alerta de colisión. Figura 8.

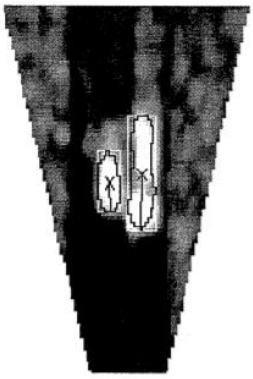

Figura 8: Vehículo detectado con Sistema de radar. Fuente: [24].

En la actualidad se quieren implementar sistemas ADAS que reconozcan los vehículos, pero haciendo uso de tecnologías que tienen mayor velocidad de procesamiento, un reducido tamaño y precios accesibles.

2. Marco conceptual

En esta sección se van a tratar los conceptos necesarios para dar a conocer el proyecto que se va a realizar:

- **Accidente de tránsito:** Evento que se genera normalmente de manera involuntaria que genera un vehículo en movimiento, el mismo puede que cause daños a bienes y/o a personas este normalmente afecta la circulación en las zonas de influencia del hecho.
- **Conjunto óptico:** Luces de servicio, direccionales, pilotos de freno, delimitadoras y reversa que prestan información a los conductores.
- **Señales luminosas de peligro:** Señales visibles en la noche que emiten su propia luz, dentro del espectro visible por el ojo humano en la noche (roja amarilla o blanco).
- **Transformación de vehículo:** Procedimiento físico y mecánico mediante el cual un vehículo automotor puede ser modificado con el fin de cumplir una función diferente o mejorar su funcionamiento, higiene o seguridad [25].

- **Sistemas embebidos:** Son sistemas de computación que fueron creados con un diseño y una arquitectura especifica según sean los requisitos del sistema a controlar (lavadora, computadores de carros, horno microondas) para el cual fue diseñado. Estos sistemas cuentan con microprocesador, microcontrolador memoria interna memoria externa, la mayoría de componentes del mismo se encuentran dentro una misma placa base y no se pueden modificar sus características como en los computadores de escritorio.

Figura 9: Sistema embebido. Fuente: [26].

La mayoría de los sistemas embebidos (Figura 9) deben reaccionar a estímulos del dispositivo a controlar (u operador) en tiempo real, pues estos normalmente se encuentran conectados a ambientes físicos a través de sensores y actuadores [27].

3. Marco teórico

ADAS: Los sistemas avanzados de asistencia al conductor (ADAS) se define como un sistema inteligente que puede mejorar la seguridad vial en términos de prevención de choques, mitigación, protección de la gravedad del choques y fases posteriores al choque, en términos más generales son sistemas de apoyo al conductor que están destinados a brindar una mayor seguridad vial y en otros casos mayor comodidad [28].

GPS: Llamado regularmente GPS, aunque, su nombre original es NAVSTAR-GPS1 es un sistema que permite la ubicación de un objeto haciendo uso de satélites, donde se puede alcanzar una precisión de hasta centímetros, pero lo habitual es de metros.

El funcionamiento del GPS se basa en un principio de triangulación para calcular la ubicación mediante una red de satélites con trayectorias sincronizadas alrededor de la tierra, exactamente a 20.000km de la superficie. Este calcula una ecuación de una esfera alrededor de cada satélite que hace parte de la triangulación, la intersección generada por cada una de las esferas determina la posición real del objeto. Razón por la cual entre más satélites participen en la triangulación mayor será la precisión [29].

MPU-9250: El MPU es un dispositivo de medición inercial con 9 ejes (3 acelerometro,3 de giroscopio y 3 de magnetómetro), este dispositivo permite medir la aceleración, teniendo en cuenta que a pesar que no exista movimiento, el acelerómetro estará midiendo la aceleración de la gravedad, además con el acelerómetro podemos obtener más medidas de manera indirecta como la velocidad y la posición; datos que nos van a permitir dar le veracidad a otros datos (GPS), los datos del giroscopio nos reconocerá a que dirección está apuntando el vehículo y el magnetómetro nos permitirá saber en dónde se ubica el polo norte magnético para mejorar el sentido de orientación del vehículo. [30]

XBee: Los módulos XBee son dispositivos que proporcionan comunicación por medio inalámbrico para la conexión entre dispositivos estos módulos utilizan el protocolo de red llamado IEEE 802.15,4 para crear redes punto a multipunto o para redes punto a punto, el diseño está enfocado en aplicaciones que necesitan alto tráfico de datos, bajas latencias [31].

Raspberry: La empresa Raspberry Pi Foundation inicio el desarrollo de un ordenador de placa reducida de bajo coste con el objetivo de estimular la enseñanza y el desarrollo de proyectos electrónicos de bajo costo, pero de gran complejidad.

La Raspberry pi 4 model B cuenta con un CPU BCM2711, Quad core Cortex-A72 (ARM V8) 64-bit 1.5Ghz en tres modelos disponibles (1GB,2GB o 4 GB) LPDDR4-3200 SDRAM, 2 puertos 2.0,2 puertos 3.0 adicionalmente cuenta con dos puertos micro HDMI que soportan 60 fps, y con conexión inalámbrica de 2.4 GHz y 5.0 GHz IEEE 802.11ac, Bluetooth 5.0, BLE.

4. Marco jurídico

Normatividad del trasporte en Colombia: Como organismo encargado de definir formular y regular las políticas de transporte tránsito y su infraestructura se encuentra el ministerio del trasporte donde como mayor herramienta tiene el Código Nacional de Tránsito Terrestre.

Dentro de las normativas que se han generado en la ley colombiana relacionada con equipos ADAS solo se han hecho acercamientos respecto a sistemas de freno (ABS), bolsas de aire y sistemas de retención de cabeza o apoyacabezas, sin embargo, cabe recalcar que con normativas como estas que fomentan el uso de dispositivos de seguridad y apoyo al conductor.

RESOLUCIÓN 3752 DE 2015

(Octubre 6)

ART. 1º—**Objeto**. Proteger la vida e integridad de las personas mediante la exigencia de elementos de seguridad activa y pasiva para uso en vehículos automotores y en remolques y semirremolques nacionales e importados que se comercialicen en el país en lo pertinente.

ART. 9º—**Reglamentación**. El Ministerio de Transporte expedirá el reglamento técnico de seguridad activa y pasiva dentro de los doce (12) meses siguientes a la expedición de la presente resolución. En caso de no contar con el reglamento técnico en el plazo señalado, al entrar en

vigencia la presente resolución se aceptarán los dispositivos señalados en las normas técnicas internacionales [32].

VI. METODOLOGÍA

La investigación está enmarcada en un tipo de investigación mixta con enfoque holístico, el diseño de la investigación es de campo, transeccional y multivariado, puesto que se busca diseñar e implementar un prototipo hardware de una plataforma de asistencia al conductor. Aunque se puede considerar también de carácter cuantitativo y descriptivo, con base a la naturaleza de los datos recolectados que provienen de sensores estandarizados, validados, confiables y al análisis de los datos recolectados que permiten determinar la velocidad del vehículo, distancia entre vehículos (en una futura etapa) y el modo de conducción.

El desarrollo de la metodología utiliza un modelo de cascada, es el enfoque metodológico que ordena rigurosamente las fases (Figura 10) del proceso para el desarrollo del proyecto, de tal forma que el inicio de cada fase debe esperar a la finalización de la fase anterior. Las fases están relacionadas con los objetivos propuestos que corresponden a diseñar e implementar un hardware para un prototipo de sistema de asistencia al conductor (ADAS), compuesto por un sensor Inercial, GPS, Cámara y un sistema de comunicación IoT (XBee).

Figura 10: Fases del diseño metodológico del proyecto. Fuente: Autores.

1. Fases para el desarrollo del proyecto

Fase1

- Revisión del estado del arte en sistemas embebidos y tecnologías IoT: Se hizo un estudio de los diferentes sistemas avanzados de asistencia al conductor que existen actualmente y como estos han ayudado en el mejoramiento de la movilidad, de igual forma se investigó sobre las diferentes tecnologías IoT que están funcionado hasta la actualidad.

Fase2

- Identificar las variables necesarias para el diseño del prototipo: Esto es necesario para la busque de los sensores que sean los más aptos para la implementación del prototipo.
- Determinar que sensores son los mejores a utilizar: Como las variables a medir ya se definieron ahora se tienen que seleccionar los sensores que mejor funcionen para la adquisición de los datos.
- Cotización de los elementos: Para empezar a realizar el prototipo es necesario determinar el costo de los elementos a utilizar y la disponibilidad de estos.

Fase3

- Adquirir los datos del sensor inercial y del GPS: Se verifica que el protocolo de comunicación I2C esté funcionando y se pueda adquirir los datos que el sensor inercial recolecta. Los datos de del GPS se adquieren por medio de comunicación serial y se comprueba que estos estén llegando correctamente y sean datos verdaderos de la localización.
- Calibrar los sensores: Para el funcionamiento correcto de los sensores se tiene que realizar una calibración y así tener una adquisición de datos buena.

Fase4

- Realizar pruebas de comunicación con los módulos XBee: Se realiza para comprobar el buen funcionamiento de la comunicación como el alcance y exposición a las diferentes interferencias que se puedan presentar.
- Implementación de un algoritmo para eliminar errores: Este algoritmo se implementará con los datos obtenidos del GPS y el sensor inercial para centrar la ruta y hacer una

predicción de esta en caso de que falle la señal del GPS, es debido afirmar que el algoritmo a utilizar se tomara de trabajos anteriores de la facultad especificando los méritos correspondientes.

Fase5

- Incorporar la adquisición de datos de los sensores y la comunicación con XBee: Con la integración de los datos con la comunicación ya es posible empezar a realizar pruebas del prototipo.
- Adquisición de imágenes con la cámara: Obtención de las capturas de la cámara cada cierto tiempo.

Fase6

- Elaborar una caja segura para la implementación del prototipo: Esta permite la adecuación del sistema para que quede fijo en el vehículo.
- Implementación del prototipo de hardware de ADAS con el sensor inercial, GPS, cámara y el módulo XBee en el vehículo: Se elige el mejor lugar de la estructura del vehículo en la cual el prototipo se pueda ubicar sin molestar a los usuarios.
- Comprobar la adquisición de datos del prototipo: Determinar si los datos se están recibiendo y transmitiendo de manera correcta y verídica.

Fase7

- Realizar pruebas con el prototipo de hardware de ADAS implementado en el vehículo: Con las pruebas ya se tiene una visión del comportamiento del sistema a los factores ambientales a los cuales va a estar expuesto.
- Analizar los datos obtenidos de las pruebas del prototipo: Se hace el análisis de los datos y la comparación de estos con una ruta definida para determinar que tan bien se está haciendo la adquisición de datos.
- Generar estadísticas del conductor: Se realizan graficas con los datos de velocidad, aceleración y la ruta para mirar que tan bien funciona el prototipo de hardware de ADAS.

Fase8

- Elaboración del documento final y publicación de resultados: Entrega final del trabajo, el prototipo de hardware de ADAS y los resultados obtenidos de las pruebas y análisis.

2. Hipótesis

Con este proyecto se espera que en su implementación se reduzca el índice de accidentalidad por causa de altas velocidades e irresponsabilidad de los conductores ayudando así a la movilización en la zona urbana de la ciudad de Tunja y en la carretera, también se quiere generar un sentido de conciencia en los conductores para que puedan mejorar su forma de conducción y respetar las normas de tránsito.

VII. RESULTADOS

1. Diseño del prototipo ADAS

Se realizó el diseño del sistema con los elementos necesarios para el desarrollo del prototipo de asistencia al conductor (ADAS), adecuado y de bajo costo, se puede ver el diagrama en la siguiente Figura 11.

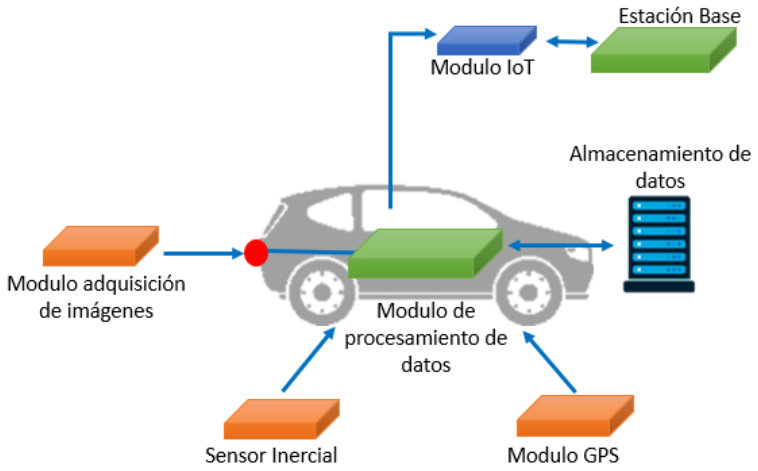

Figura 11: Diseño del prototipo ADAS. Fuente: Autores.

2. Selección de los sensores y módulos

Se quiere un prototipo económico y funcional por esto se necesita elegir las mejores opciones de sensores y módulos para la implementación del diseño.

2.1. Raspberry

Para el sistema de procesamiento se decidió utilizar una Raspberry pi 4 b debido a que estos son computadores de bajo costo y tienen las características necesarias para el prototipo (Tabla 2).

	Procesador	GPU	Memoria	Conectividad inalámbrica	Conectividad de red	Puertos
Raspberry pi 4B	quad-core Cortex-A72	VideoCore VI 500 MHz	4 GB	Wi-Fi 2,4GHz/5GHz, Bluetooth 5.0, BLE	Gigabit Ethernet	GPIO 40 pines, 2 Micro HDMI, 2 USB 2.0, 2 USB 3.0, CSI, DSI, Micro SD, USB-C (Alimentación)

Tabla 2: Características de la Raspberry pi 4 b. Fuente: Autores.

También dispone de comunicación serial e I2C en los puertos GPIO, además cuenta con dos puertos de alimentación de 5v, dos puertos de 3.3v y ocho puertos a tierra (Figura 12).

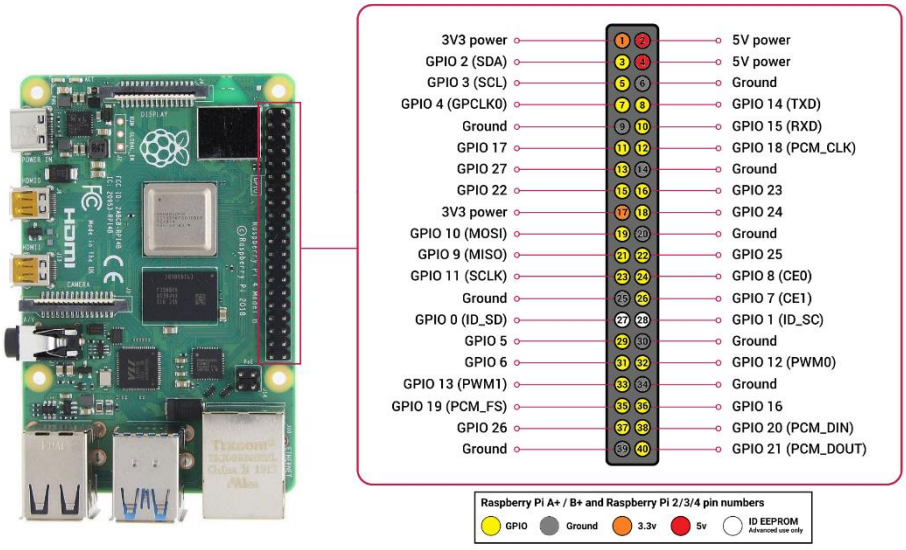

Figura 12: Raspberry pi 4 b y distribución de pines GPIO. Fuente: Autores.

2.2. Modulo GPS

El modulo GPS NEO 6M (Figura 13) se seleccionó ya que este cuenta con comunicación serial permite el acceso a los datos por medio de los pines GPIO Rx y Tx de la Raspberry, este módulo funciona con voltajes de alimentación de entre 3-5v (Tabla 3) por esta razón se puede alimentar con cualquiera de las dos salidas de alimentación de la Raspberry de 3.3v o 5v.

	Comunicación	Voltaje de alimentación	Baudrate	Sistema de coordenadas	Sensibilidad captura	Sensibilidad rastreo	Max. altura medible	Max. Velocidad	Exactitud
GPS NEO -6M	Serial	3.3-5 VDC	9600	WGS-84	-148dBm	-161dBm	18000	515 m/s	1micro seg.

Tabla 3: Características GPS NEO 6M. Fuente: Autores

Figura 13: GPS NEO 6M. Fuente:[33].

Al módulo se le incorpora una antena SMA (Figura 14) con frecuencia de 1575.42 MHz (Tabla 4) para reducir el tiempo y mejorar la señal de recepción del GPS.

	Frecuencia	Ganancia	Impedancia	Ancho de banda
Antena GPS SMA	1575.42 MHz	27 dB	50 Ω	5 MHz

Tabla 4: Características Antena adicional GPS SMA. Fuente: Autores.

Figura 14: Antena GPS SMA. Fuente [34].

2.3.Sensor Inercial

El sensor MPU 9250 (Figura 15) se eligió porque es de los pocos que tienen 9 ejes que incluyen 3 ejes de un giroscopio, 3 ejes de un acelerómetro y 3 ejes de un magnetómetro (Tabla 5), los cuales son importantes para la implementación del prototipo debido a que son necesarios para determinar velocidad, dirección y sentido.

	Rango giroscopio	**Rango acelerómetro**	**Rango magnetómetro**	**Comunicación**	**Alimentación**
MPU 9250	±250, ±500, ±1000 y ±2000 °/seg	±2g, ±4g, ±8g 7 y ±16g	±4800 µT	I2C	3-5 V

Tabla 5:Características MPU 9250. Fuente: Autores.

Figura 15: Sensor MPU 9250. Fuente: [35].

2.4.Cámara

Debido a que a futuro se quiere implementar un reconocimiento a distancia por medio de imágenes se decidió escoger una cámara para implementarla en el prototipo y la seleccionada fue una cámara infrarroja de 5MP (Figura 16) ya que esta cuenta con visión nocturna (Tabla 6).

	Sensor	**Lente**	**Apertura**	**Longitud Focal**	**FOV D**	**Mejor Resolución**
Cámara infrarroja 5MP	OV5647 5MP 2592x1944 píxeles	1/4 5M	1.8	1.85mm	70°	1080

Tabla 6: Características cámara infrarrojas 5MP. Fuente: Autores.

Figura 16: Cámara infrarroja para Raspberry de 5MP. Fuente: [36].

2.5.Modulo comunicación IoT

Ya que se piensa tener una estación base para recibir los datos y monitorear el prototipo es necesario seleccionar un módulo de comunicación IoT por esto se decidió utilizar el XBee pro S3B (Figura 17) ya que este trabaja a una frecuencia de 900MHz (Tabla 7) permitiendo un alcance de transmisión y recepción de 600m en línea de vista.

	Frecuencia	**Velocidad de datos**	**Sensibilidad**	**Interfaces seriales**	**Tipo de antena**	**Voltaje de alimentación**	**Corriente recepción**	**Corriente transmisión**	**Alcance**
XBee pro S3B	902 MHz - 928MHz	10 kbps	-110 dBm	UART	RP-SMA	3 - 3.6 V	80 mA	210 mA	Interior o zona urbana hasta 600m

Tabla 7: Características XBee pro S3B. Fuente: Autores.

Figura 17: XBee pro S3B. Fuente: [37].

Adicionalmente se seleccionó una antena de conector RP-SMA (Figura 18) que trabaja a una frecuencia de 2.4GHz (Tabla 8) para mejorar el alcance del XBee y así aprovechar al máximo la comunicación.

	Frecuencia	Ganancia	Impedancia	Alcancé XBee pro S3B
Antena RP-SMA	2.4 Ghz	5 dBi	50 Ω	Aprox. 15 Km

Tabla 8: Características antena adicional RP-SMA. Fuente: Autores.

Figura 18: Antena RP-SMA para XBee. Fuente: [38].

El sistema XBee permite realizar conexiones con cuatro topologías de red, las cuales son:

Pair (Par): Es la topología más sencilla de implementar con dos nodos, uno debe ser el coordinador y el otro puede ser dispositivo final (End device) o router [39].

Star (Estrella): Esta topología tiene al coordinador en el centro de la red y los dispositivos finales (End device) se conectan en círculo, los mensajes tienen que pasar por el coordinador, pero los dispositivos finales (End device) no tienen comunicación directa entre ellos [39].

Cluster Tree (Árbol): En esta topología el coordinador se ubica en la raíz del árbol, los routers se encargan de formar las ramas y transmitir los mensajes a los dispositivos finales (End device) [40].

Mersh (Malla): Es la topología con más ventajas y la más utilizada por los usuarios de XBee, esta cuenta con nodos router y coordinador, y es una configuración no jerárquica permitiendo la comunicación de todos los elementos que la conforman facilitando que si en algún momento falla la comunicación se puedan rehacer los enlaces y continuar con la comunicación [39].

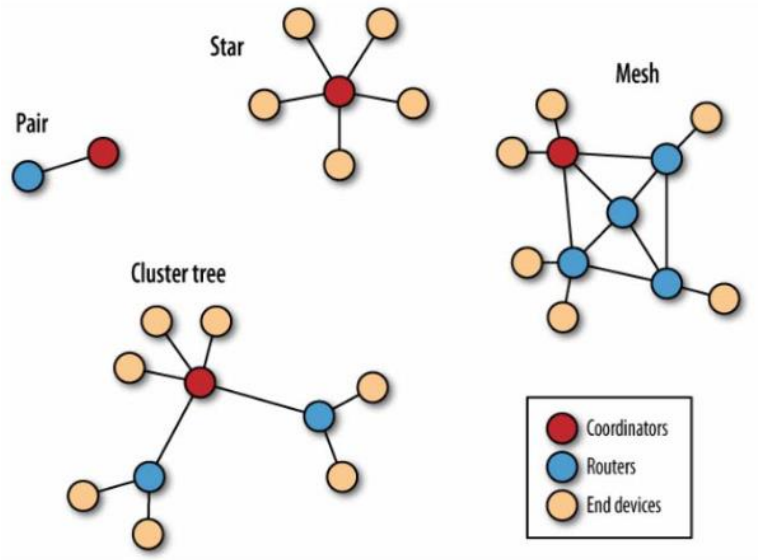

Figura 19: Topologias de res XBee. Fuente: [39].

3. Adquisición de datos

Ya teniendo seleccionados los mejores módulos y sensores para la implementación del prototipo se procede a configurar la Raspberry pi 4 con el sistema operativo compatible en este caso Raspbian y se descargan las actualizaciones, libreas y demás complementos para la adecuada adquisición de los datos.

Se conectan los módulos y sensores a la Raspberry pi 4 b como se muestra en la Figura 19 además se acopla el módulo de la cámara en el puerto de la Raspberry específico para este, el XBee se conecta al puerto USB, teniendo así todos los módulos y sensores alimentados y listos para la adquisición de datos.

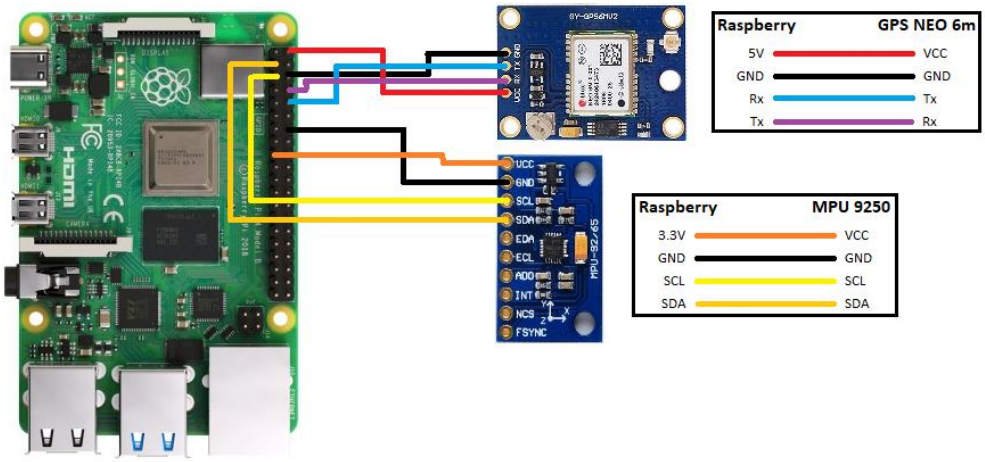

Figura 20: Conexión de los sensores y módulos en la Raspberry pi 4 b. Fuente: Autores.

3.1. Módulo GPS NEO 6M

El módulo GPS funciona con comunicación serial por esto es necesario descargar la librería piserial de Python la cual permite leer los datos del puerto serial, esto se realiza con un código en Python el cual se puede ver en la siguiente Figura 20.

```
import serial #Libreria piserial

#Definir puerto serial de lectura
ser=serial.Serial('/dev/ttyS0',baudrate=9600,timeout=0.5)

while(True):

    #Leer datos del puerto serial
    data=ser.readline().decode('utf-8')
    print(data) #Imprimir trama de datos del GPS
```

Figura 21: Código en Python que permite la lectura de los datos del Módulo GPS NEO-6M. Fuente: Autores.

Ya con el código implementado en Python el GPS envía una trama de datos que corresponden al formato estándar NMEA 0138 como se observa en la Figura 21.

```
$GPGSV,4,4,15,32,13,143,16,46,26,266,,51,50,262,*4C

$GPGLL,0532.63185,N,07321.24855,W,014743.00,A,A*7A

$GPRMC,014744.00,A,0532.63188,N,07321.24863,W,0.156,,221020,,,A*6D

$GPVTG,,T,,M,0.156,N,0.288,K,A*23

$GPGGA,014744.00,0532.63188,N,07321.24863,W,1,07,1.17,2651.4,M,2.4,M,,*42

$GPGSA,A,3,21,16,27,11,32,03,22,,,,,,2.49,1.17,2.20*09
```

Figura 22: Trama de datos NMEA que genera el Modulo GPS NEO-6M. Fuente: Autores.

3.2. Sensor MPU 9250

El sensor MPU 9250 cuenta con comunicación I2C por lo tanto se necesita de la librería FaBo9Axis_MPU9250 la cual recibe los datos del sensor y permite separarlos en los 3 ejes de Acelerómetro, 3 ejes del Giroscopio y 3 ejes del Magnetómetro, el código utilizado con la librería se puede ver en la Figura 22.

```python
import FaBo9Axis_MPU9250 #Libreria MPU-9250
import time #Libreria time

#Definir la entrada de los datos del MPU-9250
mpu9250 = FaBo9Axis_MPU9250.MPU9250()

while True:

    #Leer datos del Acelerometro
    accel = mpu9250.readAccel()
    print('Acelerometro', accel)
    #Leer datos del Giroscopio
    gyro = mpu9250.readGyro()
    print('Giroscopio', gyro)
    #Leer datos del magnetometro
    mag = mpu9250.readMagnet()
    print('Magnetometro', mag)

    time.sleep(1)
```

Figura 23: Código en Python para lectura de los datos del sensor inercial MPU 9250. Fuente: Autores.

Los datos que el sensor envía se pueden ver en la Figura 23 donde se muestran los ejes X, Y y Z del Acelerómetro, los ejes X, Y y Z del Giroscopio y los ejes X, Y y Z del Magnetómetro.

```
Acelerometro {'x': -0.004, 'y': -0.009, 'z': -0.913}
Giroscopio {'x': 0.107, 'y': -0.504, 'z': 0.008}
Magnetometro {'x': -38.839, 'y': -19.483, 'z': -118.93}
Acelerometro {'x': -0.002, 'y': -0.01, 'z': -0.91}
Giroscopio {'x': 0.168, 'y': -0.465, 'z': 0.099}
Magnetometro {'x': -39.558, 'y': -18.04, 'z': -117.543}
Acelerometro {'x': -0.006, 'y': -0.016, 'z': -0.909}
Giroscopio {'x': 0.191, 'y': -0.557, 'z': 0.046}
Magnetometro {'x': -38.659, 'y': -21.106, 'z': -117.716}
```

Figura 24: Datos generados por el sensor MPU 9250. Fuente: Autores.

3.3. Cámara Infrarroja 5MP 1080 con soporte para visión nocturna

Para el uso de la cámara es necesario instalar la librería OpenCv, además de la librería numpy, estas librerías permiten que se puedan visualizar los frames que captura la cámara y convertirlos en video o en capturas de imagen, pero como prueba inicial solo se captura el video para verificar el funcionamiento de la cámara con el siguiente código que se muestra en la Figura 24.

```python
import numpy as np #Libreria numpy
import cv2 #Libreria OpenCV

#Definir camara
cap = cv2.VideoCapture(0)

while(True):

    # Captura frame
    ret, frame = cap.read()

    #Mostrar video
    cv2.imshow('Camara',frame)
    #Definir tecla para parar la camara
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

#Cierra la camara
cap.release()
cv2.destroyAllWindows()
```

Figura 25: Código en Python con la librería OpenCv para capturar imagen de la cámara infrarrojas. Fuente: Autores.

El video proporcionado por la cámara se puede ver en la Figura 25 este se encuentra en la máxima resolución de 1080 p.

Figura 26: Video captado por la cámara infrarroja con resolución de 1080p. Fuente: Autores.

3.4. Módulo XBee pro S3B

El módulo XBee se conecta por medio de comunicación serial por lo tanto al igual que el GPS necesita de la librería piserial para realizar la comunicación con el otro XBee, para comprobar el funcionamiento de la comunicación se implementó un código (Figura 26) donde se pueden leer y enviar datos al otro modulo.

```
import serial #Libreria piseria

#Definir puerto serial del Xbee
ser=serial.Serial('/dev/ttyUSB0',9600,timeout=1)

while True:

    #Leer los datos que recibe el Xbee
    data=ser.readline().decode('utf-8')
    print(f"Recibido: {data}") #Imprimir datos

    #Envia datos tipo String al Xbee
    env=input('Enviar:')
    ser.write(env.encode())
```

Figura 27: Código en Python para realizar la comunicación inalámbrica ente los dos modulos XBee pro S3B. Fuente: Autores.

Para comprobar la conexión se conectó uno de los XBee al computador y por medio del programa XCTU se realizó la conexión a la Raspberry donde se envían y se reciben mensajes como se ve en la Figura 27.

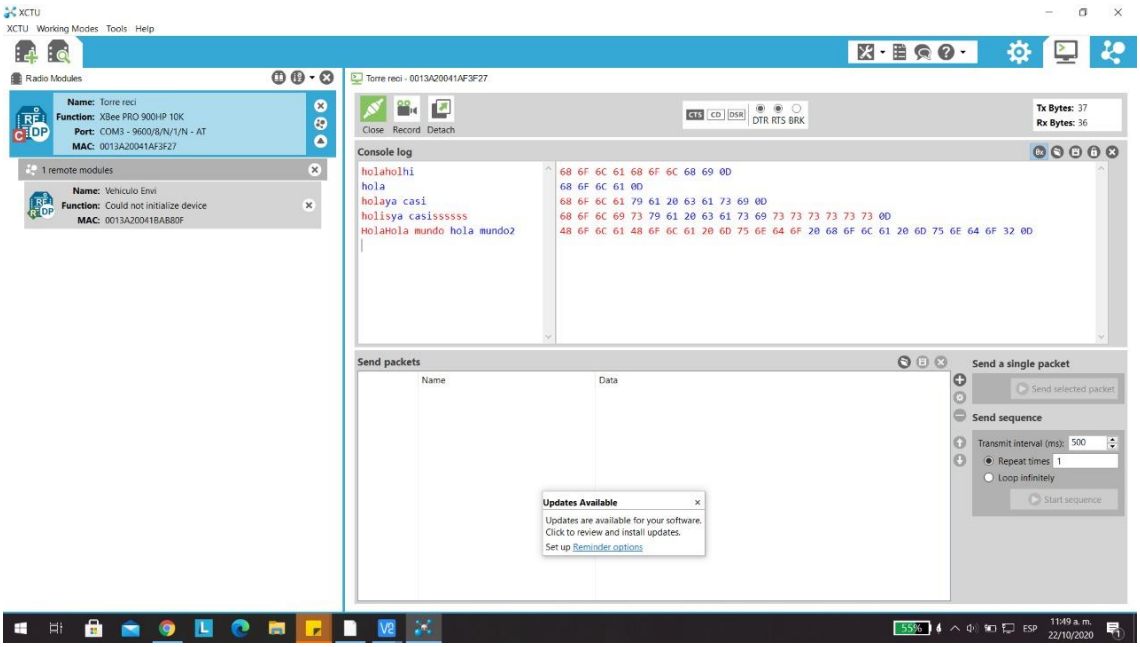

Figura 28: Programa XCTU con el cual se realizaron pruebas de la conexión inalámbrica entre los XBee pro S3B. Fuente: Autores.

En la Figura 28 se pueden ver los datos enviados y recibidos en la Raspberry mediante el código en Python.

```
Enviar:Hola mundo
Recibido:
Enviar:
Recibido:  hola mundo2
```

Figura 29: Mensajes enviados entre los dos XBee pro S3B. Fuente: Autores.

4. Calibración y configuración

Para el funcionamiento correcto de algunos sensores y módulos en la implementación del proyecto es necesario realizar una calibración o configuración adecuada para sacar el mejor provecho de estos elementos.

4.1. Calibración MPU 9250

Según [41] el sensor MPU 9250 ya viene calibrado de fábrica, pero a la hora de usarlo este puede presentar algunos leves cambios en las medidas (Figura 29) por esto es recomendable realizar una calibración.

Como primera medida se realiza la calibración del Acelerómetro ya que este debe tener medidas en sus diferentes ejes de 0g en el eje X, 0g en el eje Y y -1g en el eje Z, debido a esto se recomienda tomar una muestra de mínimo 1000 datos de cada eje dejando el sensor MPU 9250 en una superficie plana y calcular el promedio de estos datos, después restarle este valor al dato que nos entrega el eje X del acelerómetro [41].

$$Valor\ real\ aceleración = Valor\ medido - factor\ de\ corrección$$

El eje Y del acelerómetro para la calibración requiere igual de una cantidad mínima de 1000 datos a los cuales se les calcula el promedio y se le resta al valor leído por el eje X del sensor MPU 9250 [41].

$$Valor\ real\ aceleración = Valor\ medido - factor\ de\ corrección$$

Para terminar, se calibra el eje Z del acelerómetro, pero este como esta en dirección de la gravedad tiene que dar el valor alrededor de -1g [41].

$$Valor\ rea\ aceleración\ z = Valor\ real - (facto\ de\ correxión + 1)$$

Ya teniendo los tres ejes del acelerómetro calibrado se puede ver en la Figura 30 que los valores de los 3 ejes ya están calibrados, pero aún tienen ruido.

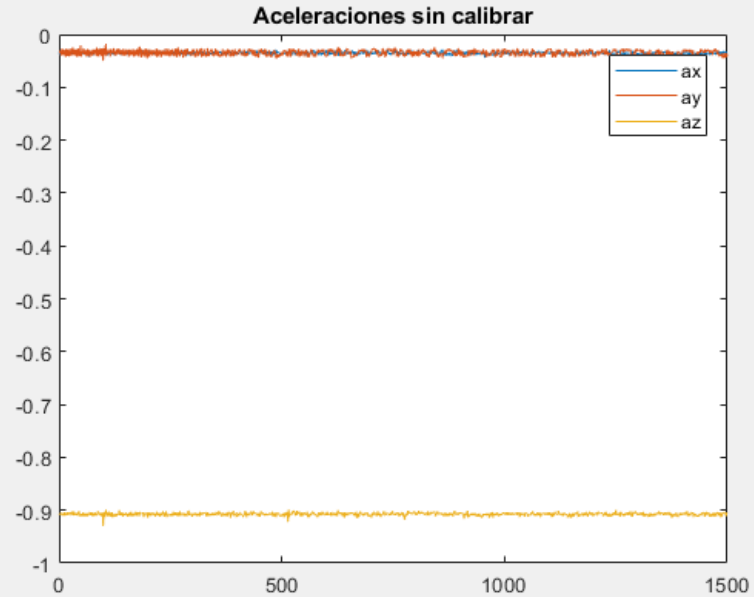

Figura 30: Grafica de los 3 ejes del acelerómetro sin calibrar. Fuente: Autores.

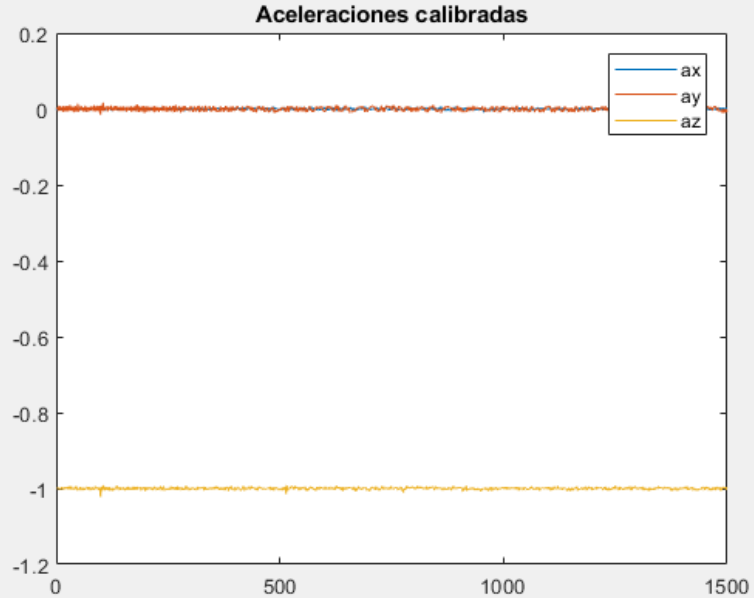

Figura 31: Grafica de los 3 ejes del acelerómetro calibrados. Fuente: Autores.

La calibración de los 3 ejes del giroscopio es similar a la de los ejes del acelerómetro, primero se toma mínimo 1000 datos, después se le saca el promedio y este se le resta al valor medido por el giroscopio [41].

$$Valor\ real\ Giroscopio = Valor\ real - (facto\ de\ correxión + 1)$$

En la gráfica de la Figura 31 se puede ver como son los valores del giroscopio sin calibra y compararlos con la Figura 32 donde se muestra la gráfica de los valores del giroscopio calibrados, pero con ruido.

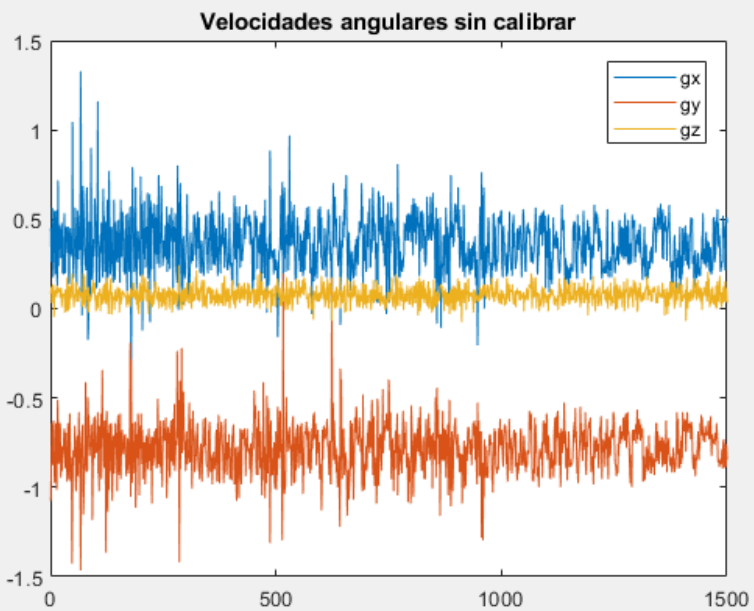

Figura 32: Grafica de los 3 ejes del giroscopio sin calibrar. Fuente: Autores.

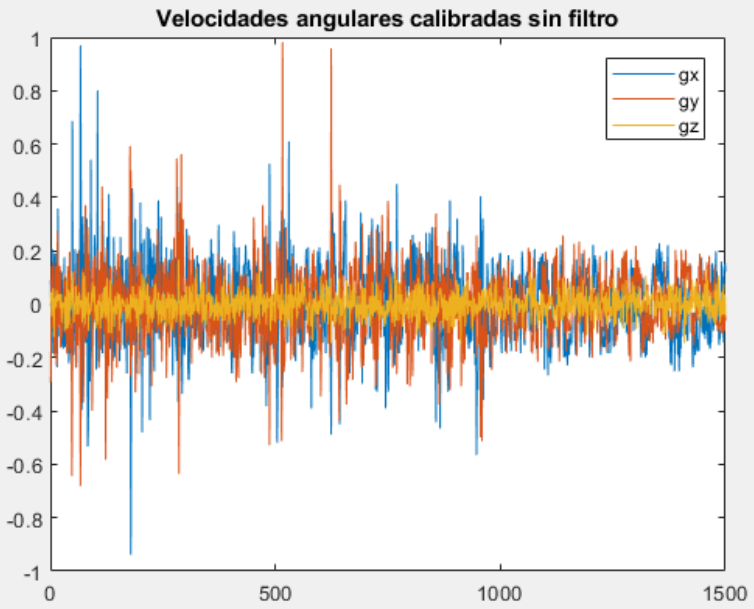

Figura 33: Grafica de los 3 ejes del giroscopio calibrados. Fuente: Autores.

El acelerómetro y el giroscopio del sensor MPU 9250 después de la calibración respectiva sigue mostrando ruido en sus lecturas como se muestra en la Figura 31 y Figura 33 respectivamente,

debido a esto es necesario implementar un filtro pasa bajos el cual se encargue de disminuir el ruido y así prevenir fallos en los cálculos del algoritmo de predicción de ruta. El filtro a implementar es un filtro pasa bajos modificado (Figura 34) debido a que este permite que el filtrado de los datos sea más rápido, característica importante dado que se necesita un procesamiento de datos eficaz, por esta razón se descarta implementar un filtro pasa bajos normal ya que este es lento [41] [42].

```
if (Value < CompareValue - NP | Value > CompareValue + NP)
{
  FilteredValue = Value;
}
else
{
  if (Value < CompareValue - NPS | Value > CompareValue + NPS)
  {
    FilteredValue = (Value + PreviousCorrect)/2;
  }
  else
  {
    FilteredValue = (1 - NPL)*PreviousCorrect + NPL*Value;
  }
}
```

Figura 34: Filtro pasa bajos modificado para reducir ruido en el acelerómetro y el giroscopio del MPU9250. Fuente: [42].

En la Figura 35 se puede observar cómo quedan los datos del acelerómetro calibrados y filtrados, de igual manera en la Figura 36 se ven los datos del giroscopio calibrados y filtrados.

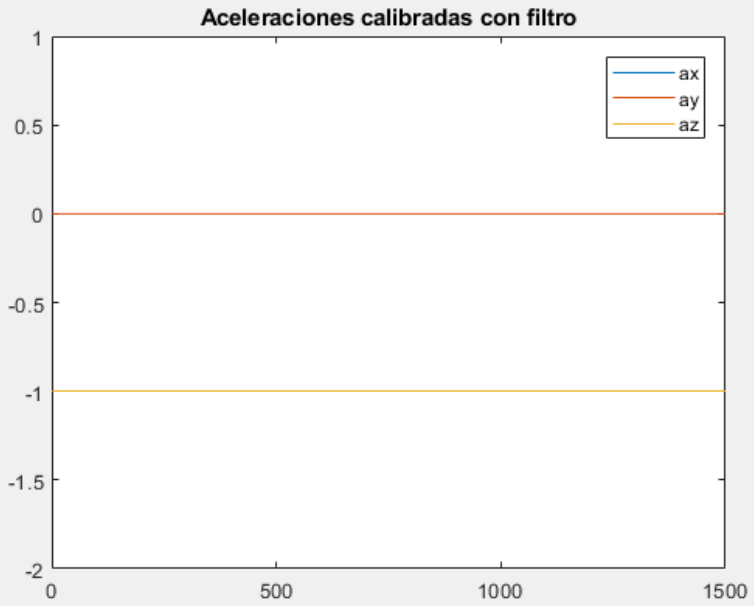

Figura 35: Grafica de los 3 ejes del acelerómetro calibrados y filtrados. Fuente: Autores.

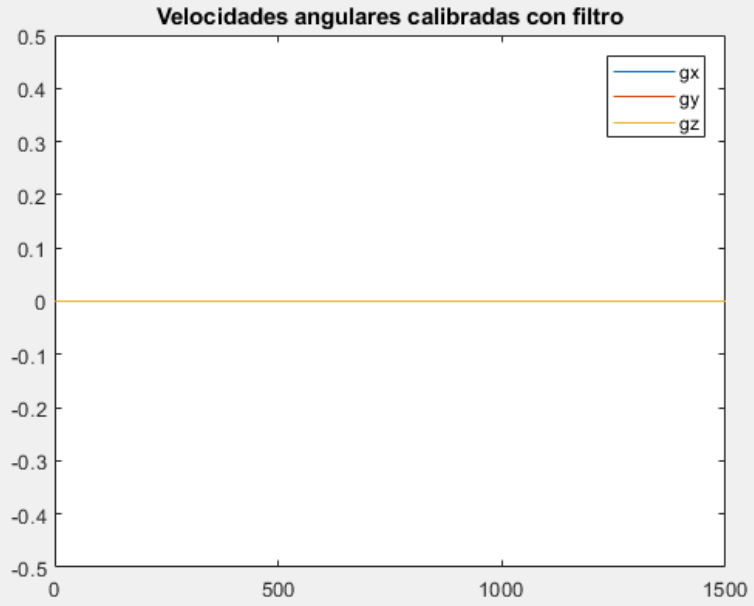

Figura 36: Grafica de los 3 ejes del giroscopio calibrados y filtrados. Fuente: Autores.

La calibración de magnetómetro es un poco más compleja para esto hay que tomar un mínimo de 1000 muestra de los valores que nos da el eje X y a la vez 1000 datos de valores que nos dé en eje Y, para tomar estas muestras hay que dar vuelta al módulo alrededor del eje X realizando una circunferencia (Figura 37). Ya teniendo la gráfica circular se compensan los valores del eje X y del eje Y del magnetómetro para centrar la circunferencia en el origen (Figura 38) teniendo así la

calibración del magnetómetro [41] y finalizando la calibración completa del sensor inercial MPU 9250.

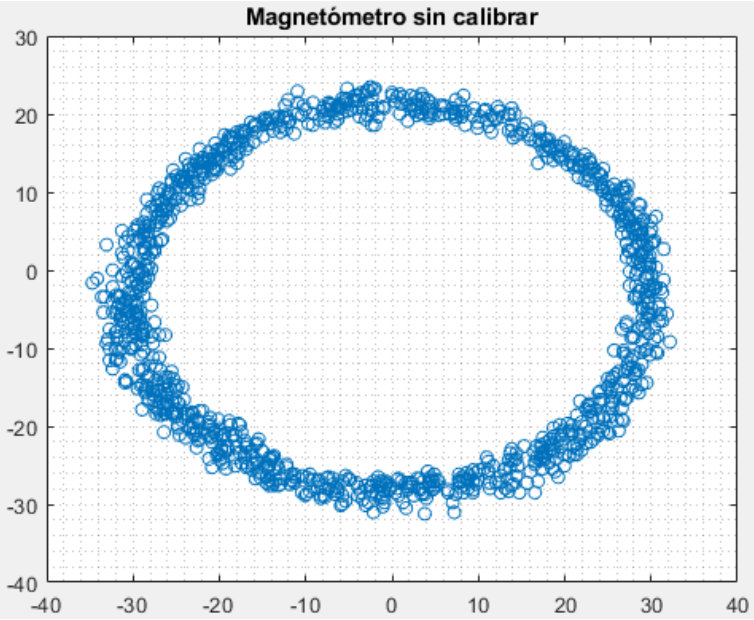

Figura 37: Grafica del magnetómetro sin calibrar. Fuente: Autores.

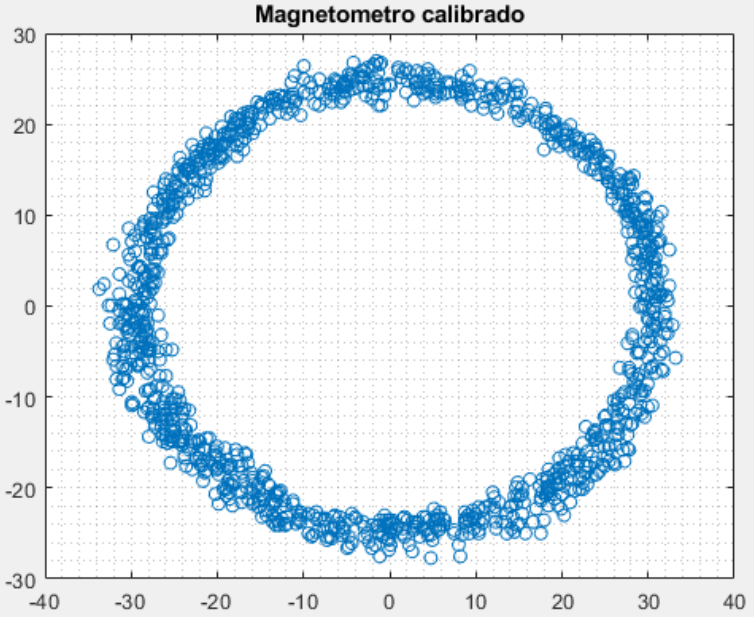

Figura 38: Grafica del magnetómetro centrado en el origen (calibrado). Fuente: Autores.

4.2. Configuración XBee

Para la configuración inicial de los dispositivos XBee se utilizó el software XCTU que permite modificar los parámetros más importantes de los módulos XBee Pro S3B a través de una interfaz gráfica fácil de usar, en la configuración de los módulos se utilizaran dos configuraciones muy diferentes, una para el modulo que va ubicado en la estación base y otra para el modulo que se ubica en sistema ADAS del vehículo.

CM Chanel Mask: Este ítem permite seleccionar los canales que se habilitan, donde cada bit corresponde a una frecuencia definida.

ID Network Id: El network id es el identificador de la red, este permite que los id que coinciden en los módulos se puedan comunicar entre sí.

PL Tx Power Level: Modifica la potencia de transmisión en 5 niveles diferentes de acuerdo a la Tabla 9.

Indicador	Potencia (dBm)
Highest[4]	24
Hight[3]	21
Medium[3]	18
Low[1]	15
Lowest[0]	7

Tabla 9: Tabla de equivalencias de potencia XBee. Fuente: Autores.

CU Routing /messaging mode: Modifica el tipo de enrutamiento y como distribuirá los mensajes en la red.

SH serial number high: Son los 32 bits más significativos de la dirección mac, que se asigna de forma exclusiva a cada módulo, este usualmente se utiliza para asignar comunicaciones punto a punto.

Sl Serial Number Low: Son los 32 bits menos significativos de la dirección mac, que se asigna de forma exclusiva a cada módulo.

DH Destination Addres High: Modifica la dirección con los 32 bits más significativos de la dirección mac del módulo de destino en la comunicación.

DL Distination Addres Low: Modifica la dirección con los 32 bits menos significativos de la dirección mac del módulo de destino en la comunicación.

NI Nodo Indentifier: Es el nombre que se le quiere agregar al nodo para facilitar su identificación dentro de una red.

ItemTo	Modulo1	Modulo2
CM Chanel Mask	FFFFFFFFFFF7FFFF	FFFFFFFFFFF7FFFF
ID Network Id	7FFF	7FFF
PL Tx Power Level	Highest	Highest
CU Routing /messaging mode	Non-Routing Cordinator	Standard Router
SH serial number high	13A200	13A200
Sl Serial Number Low	41AF3F27	41BAB80F
DH Destination Addres High	0	0
DL Distination Addres Low	FFFF	FFFF
NI Nodo Indentifier	Torre Reci	Vehiculo Envi

Tabla 10: Resumen de configuración de módulos XBee. Fuente: Autores.

5. Implementación del prototipo ADAS

Con los sensores y módulos seleccionados de acuerdo a la necesidad del prototipo se procedió a realizar un modelo del prototipo de asistente al conductor con los sensores elegidos (Hardware) y el Software necesarios para la implementación, basándonos en el modelo que se planteó en la Figura 11 se diseñó de nuevo el sistema, pero con la implementación de los elementos adecuados como se ve en la Figura 39.

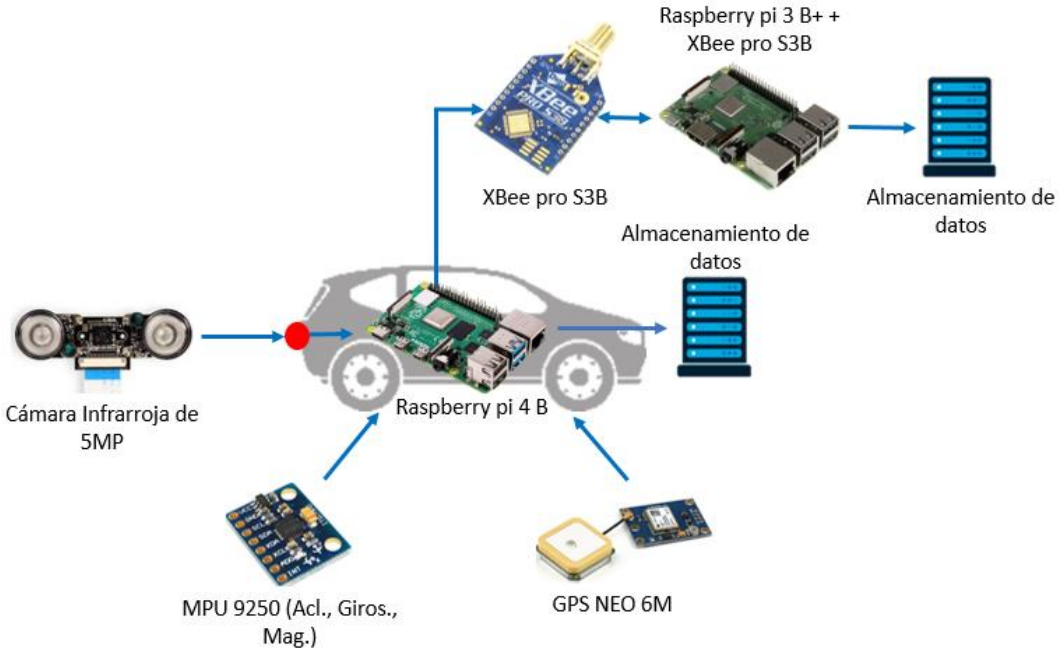

Figura 39: Diseño del sistema de asistencia al conductor ADAS. Fuente: Autores.

5.1. Hardware

El prototipo final cuenta con la conexión de todos los sensores y módulos escogidos para el óptimo funcionamiento de este, además se diseñó un soporte en el cual se ubica la Raspberry pi 4 b, el sensor MPU 9250 con una base ajustable en el eje y, el GEPS NEO 6M con la antena para mejorar la recepción de los datos, el XBee pro S3B y la cámara infrarroja de 5 MP como se ve en la Figura 40. El soporte del prototipo puede ser mejorado en el futuro.

Figura 40: El soporte del prototipo. Fuente: Autores.

5.2.Software

Se planteó un diagrama general donde se describe el proceso que se quiere llevar a cabo para el desarrollo del software teniendo en cuenta la recepción de los datos, el procesamiento de estos, el resultado final y la transmisión a la estación base este diagrama se muestra en la Figura 41.

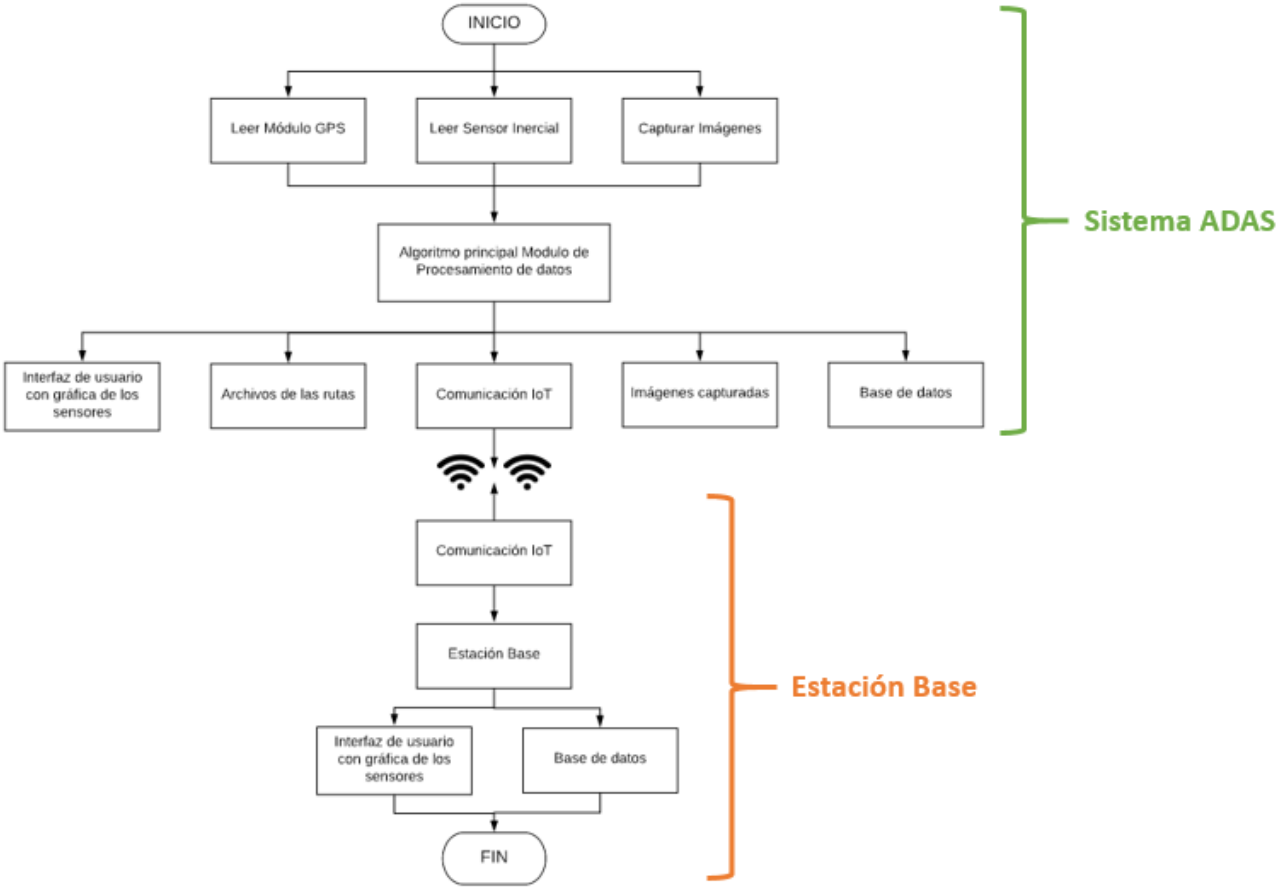

Figura 41: Diagrama general de software. Fuente: Autores.

Teniendo una idea de lo que se requiere para el desarrollo del software se decide también implementar este por medio de hilos para tener un procesamiento simultaneo de todos los datos necesarios.

Primero se hace un llamado de todas las librerías necesarias para este código, se configura la interfaz gráfica donde se verán las gráficas del sensor inercial que muestran la Aceleración en los tres ejes, la Velocidad en los tres ejes y la orientación del vehículo, y se definen algunas variables (Figura 42).

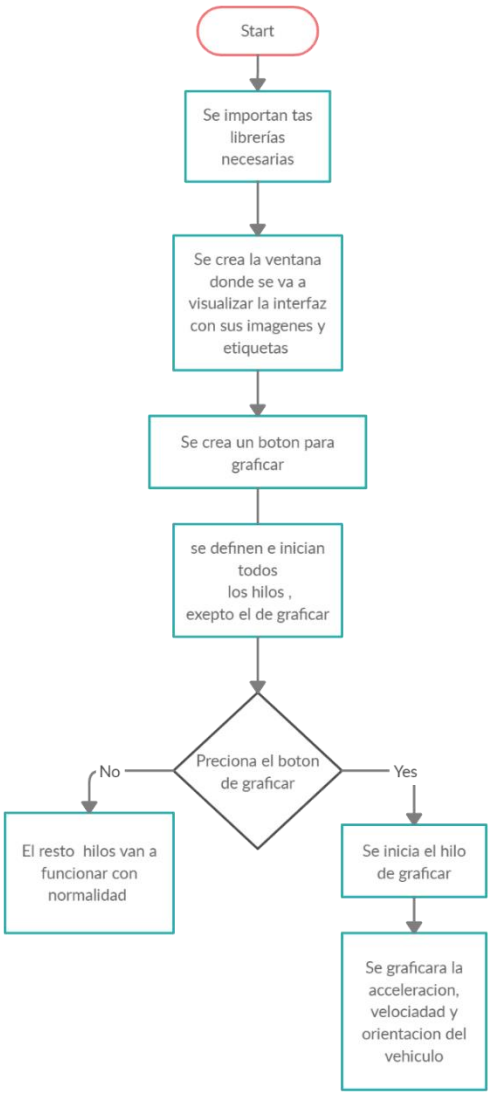

Figura 42: Diagrama de flujo de interfaz. Fuente: Autores.

Ahora se empiezan a definir los hilos donde el primer hilo guarda algunas variables globales para utilizar en hilos más adelante además de dar acceso a los puertos GPIO, configurar la entrada serial del GPS y leer los datos que el GPS genera (Figura 43).

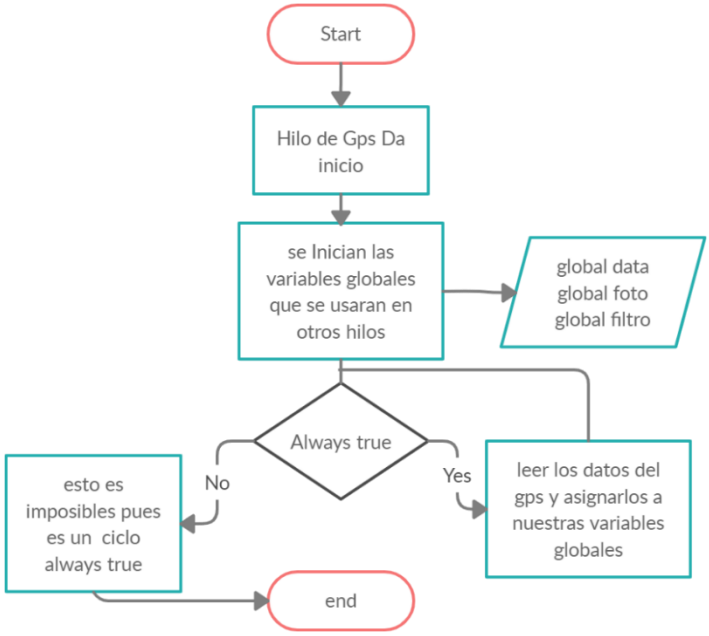

Figura 43: Hilo de GPS. Fuente: Autores.

Como se aprecia en la Figura 44 el siguiente hilo denominado filtro es el que realiza todo el proceso de predicción de ruta cuando el GPS pierde cobertura, se comienza definiendo las distintas variables a utilizar en el filtro para el sensor inercial, variables de los ángulos y variables auxiliares de integración, se coloca una condición para que el algoritmo no se ejecute hasta obtener un valor de referencia inicial, cumpliendo con la condición anterior ya se puede empezar a realizar el algoritmo donde se recogen los datos del GPS necesarios, además de leer los datos del sensor inercial agregando los valores de calibración de este sensor los cuales cambian dependiendo de cada sensor MPU 9250 por lo tanto se cambiaron los datos por los correspondientes al sensor utilizado en el prototipo implementado, se continuó con la adaptación del algoritmo de predicción de ruta [41], el prototipo guarda la ruta del vehículo en archivos kml para poder revisarlos al terminar el viaje y finalmente se envían los datos de los sensores a la Estación base.

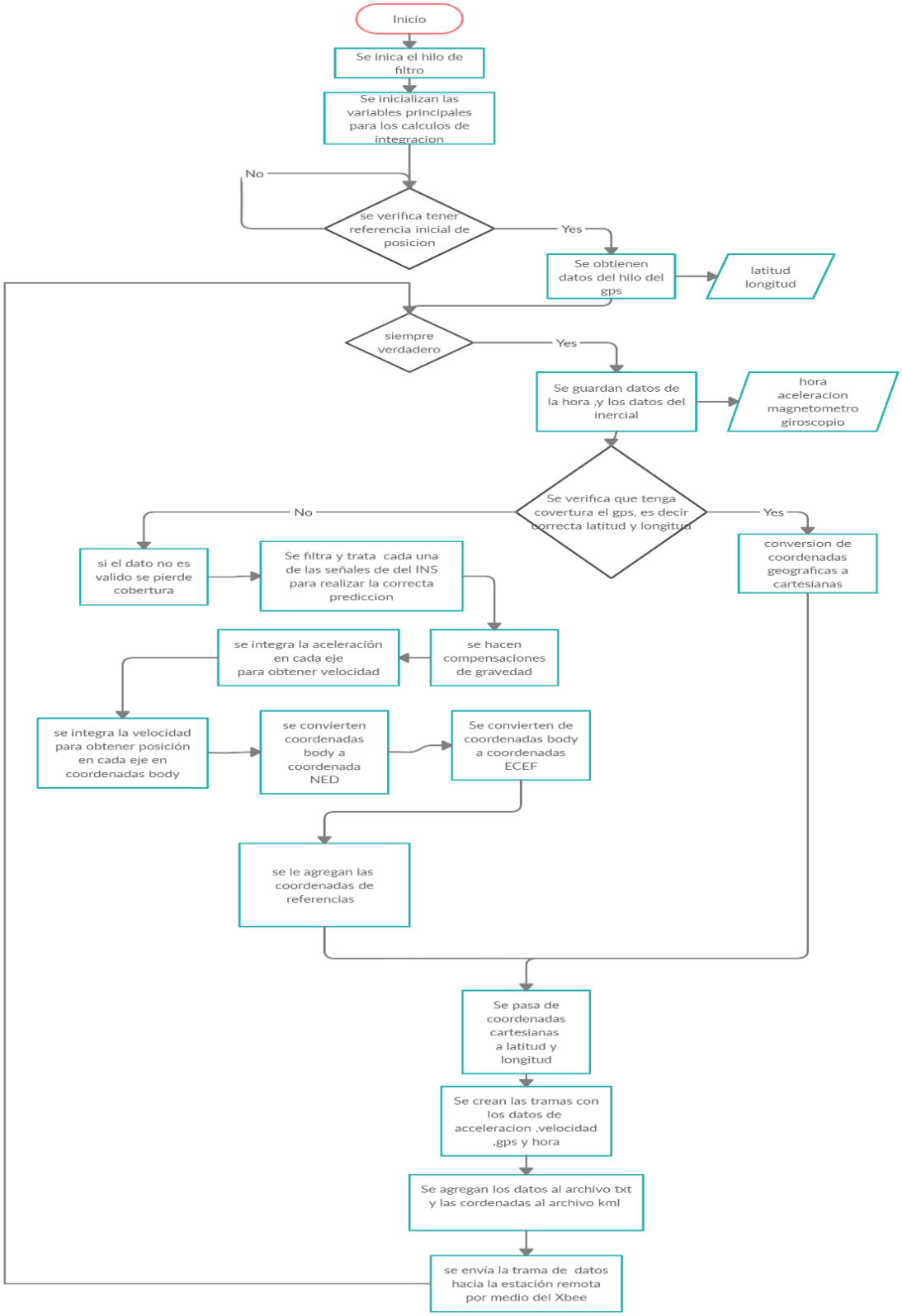

Figura 44: Hilo del filtro. Fuente: Autores.

El siguiente hilo llamado fotos permite la captura de imágenes en una resolución de 300x300 pixeles, además guarda 10 imágenes por segundo, la recolección de imágenes se utilizará en mejoras futuras para determinar el tipo de vehículo que viene en la parte posterior del carro y la distancia a la que se encuentra (Figura 45).

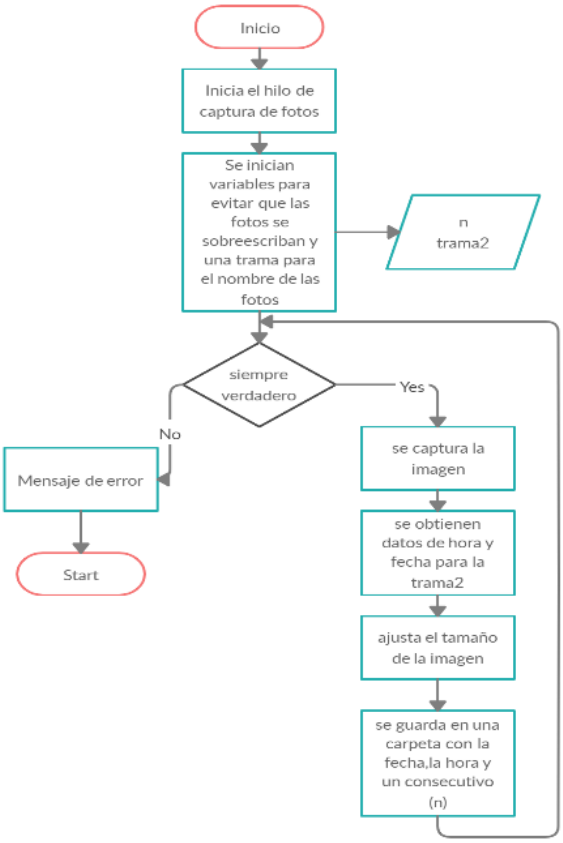

Figura 45: Hilo captura de fotos. Fuente: Autores.

Por último, el hilo denominado graficas con el cual se generan las gráficas de Velocidad en los 3 ejes, Aceleración en los 3 ejes y orientación del vehículo, estas graficas se pueden determinar con los datos que entrega el sensor MPU 9250 calibrado (Figura 46).

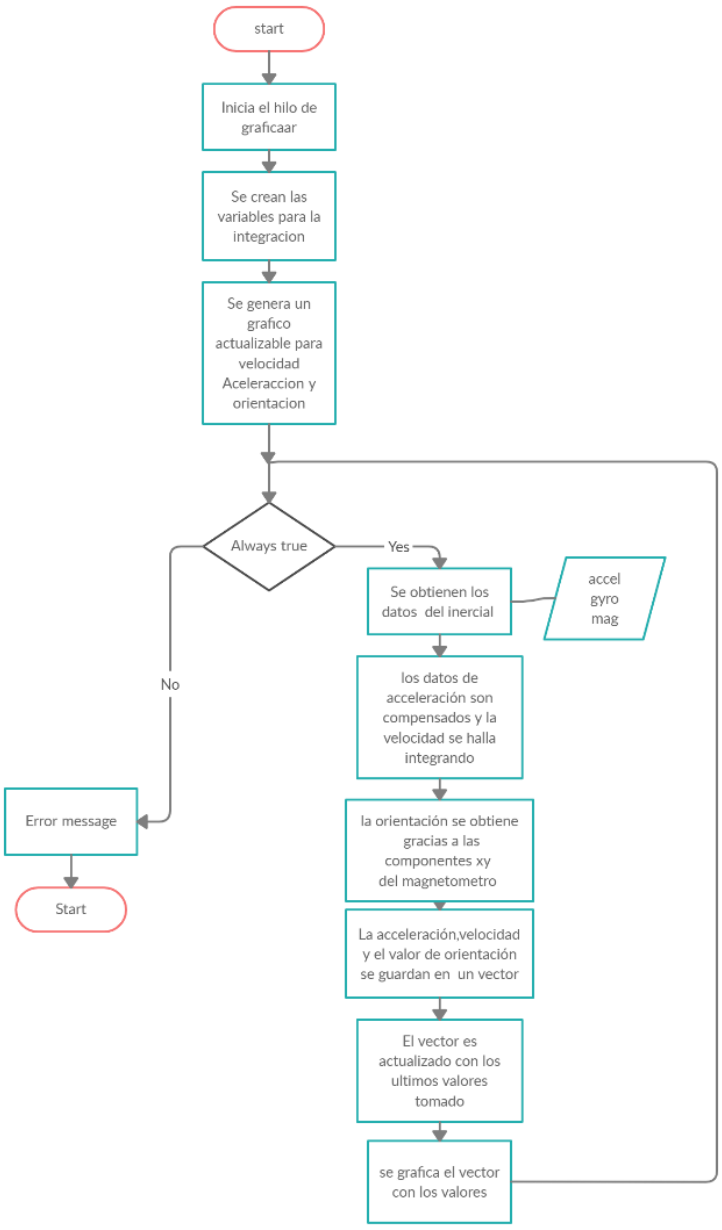

Figura 46:Hilo de gráficas. Fuente: Autores.

6. Estación base

Se quiere realizar un monitoreo del prototipo por esto se definió que se implementaría una estación base la cual está compuesta por una Raspberry pi 3 B+ y un módulo XBee pro S3B como se observa en la Figura 47 la estación base y el prototipo ADAS se comunican mediante los módulos XBee, permitiendo él envió de los datos de los sensores.

Figura 47: Módulo XBee de estación base. Fuente: Autores.

Para la recepción de los datos enviados por el prototipo ADAS se desarrolla un software que se maneja por hilos el cual permite la recolección de los datos de los sensores y de la ruta que este realiza, este código permite separar estos datos, guardar la ruta del vehículo y mostrar las gráficas de Aceleración en 3 ejes, Velocidad en 3 ejes y la orientación del vehículo en la Figura 48 se puede visualizar el diagrama de flujo de este software.

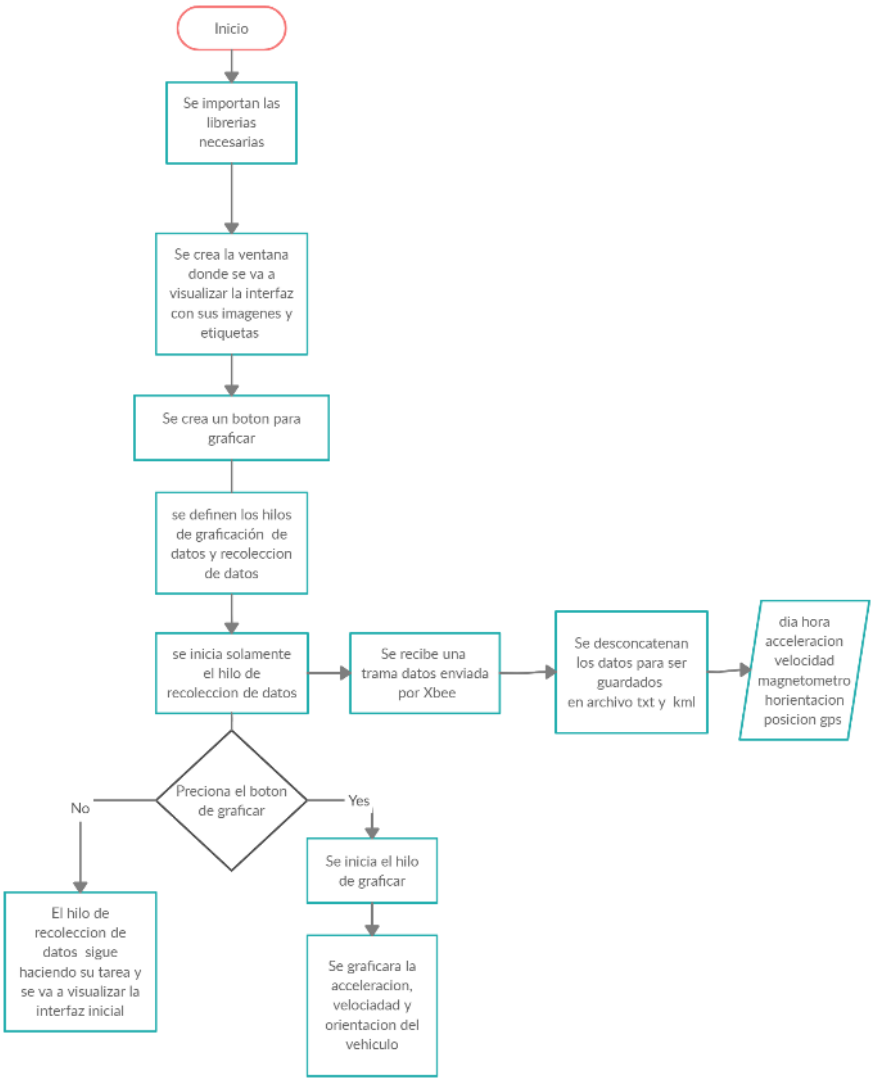

Figura 48: Funcionamiento de la estación base. Fuente: Autores.

7. Pruebas

Las pruebas se realizaron en la ciudad de Tunja en tres diferentes sitios dada la emergencia, donde cada sitio está determinado principalmente por el lugar donde se colocó la estación base para la recepción de los datos.

Universidad Santo Tomas Campus de la ciudad de Tunja

Estas primeras pruebas que se realizaron alrededor del campus universitario, donde el sitio de la antena eran los laboratorios de electrónica (Figura 49), la realización de estas primeras pruebas era para verificación de que los datos generados por el GPS y la resolución del mismo fueran útiles para el proyecto.

Figura 49: Universidad Santo Tomas. Fuente: Autores.

Los primeros datos tomados por el dispositivo presentaron muchos errores iniciales, pues los algoritmos del software solo contaban con la toma de datos y guardado de los mismo, pero no un filtro que evitara errores por perdidas de señal en el GPS como se ve en la Figura 50.

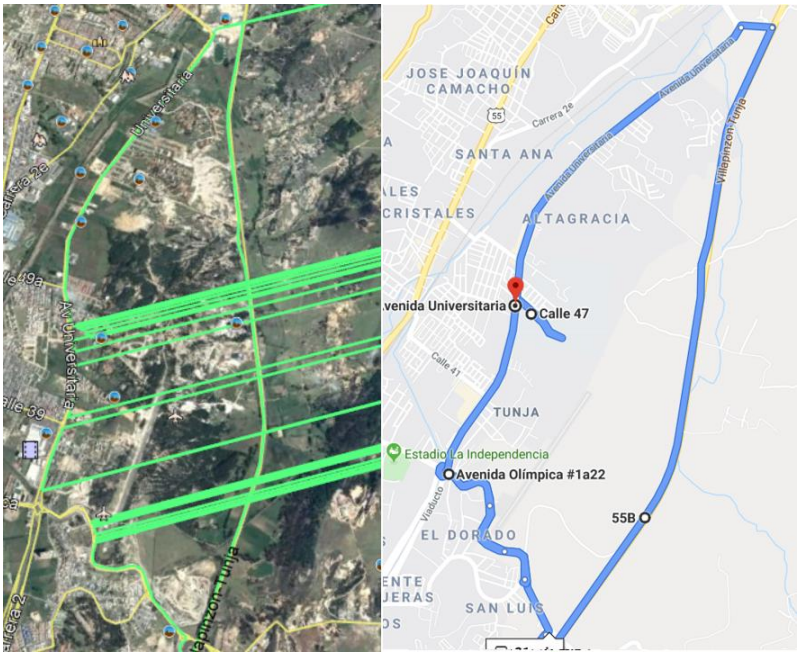

Figura 50: GPS sin filtro (izquierda) y GPS de dispositivo Android (derecha). Fuente: Autores.

Después de agregar el algoritmo de filtrado de la señal y el sistema de respaldo en caso de pérdida de la señal haciendo uso del inercial se obtienen datos muchos más fiables pues las pérdidas de señal en el GPS, no generaban una pérdida de puntos en la localización y la ruta que se generaba era la misma respecto a la generada por el dispositivo Android (Figura 51).

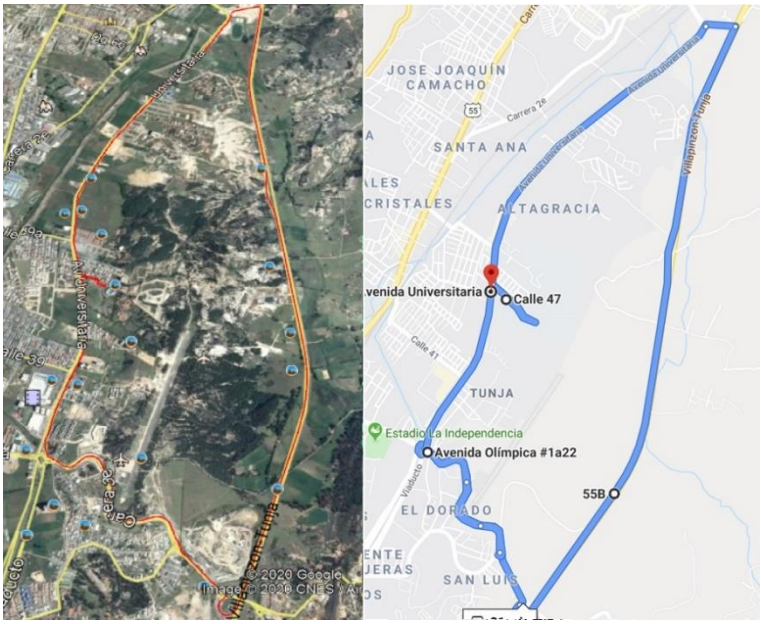

Figura 51: GPS con filtro(izquierda) y GPS Android (derecha). Fuente: Autores.

Panadería Fratellos

El siguiente grupo de datos se realizó alrededor del centro comercial Unicentro de la ciudad de Tunja, donde se tomó como punto central y lugar de la antena el edifico Fratellos Figura (52), por su disponibilidad para colocar la antena.

Figura 52: Lugar de la antena Segunda toma de datos. Fuente: Google Maps.

Para esta segunda toma de datos se quería evaluar la recepción de XBee y la configuración más adecuada del mismo para lograr la mejor recepción del XBee y así mismo los requerimientos del sistema para el correcto funcionamiento del mismo, en estas pruebas se evidencio un error (Figura 53), donde después de varias pruebas se evidencio que era cuestión de la potencia entregada a la Raspberry lo que generaba este error y adicional se generaron más algoritmos dentro del código que permitían que volviera a funcionar el dispositivo en caso de errores generados por errores de este tipo.

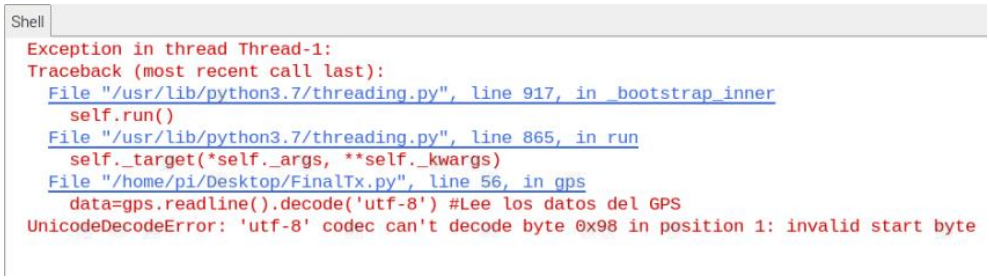

Figura 53: Error generado por falta de potencia en la alimentación. Fuente: Autores.

En estas pruebas el dispositivo ya generaba archivos kml y txt con los datos obtenidos del GPS, sensor inercial MPU 9250 (aceleración, giroscopio y magnetómetro) Figura 54.

latitud	longitud	Fecha	Hora	acceleracion_x	acceleracion_y	acceleracio_z	giro_x	giro_y	giro_z	magnetometro_x	magnetometro_y	magnetometro_z
5.5511	-73.344796	2020-09-12	20:08:34	-0.054	-0.1131.017	0.214	-0.305	0.664	22.761	-1.618	29.891	
5.551099	-73.344796	2020-09-12	20:08:35	-0.055	-0.117	1.018	0.526	-0.328	0.854	21.328	-1.618	28.163
5.551099	-73.344796	2020-09-12	20:08:36	-0.053	-0.115	1.019	0.793	-0.336	0.877	22.582	-1.798	28.682
5.551099	-73.344796	2020-09-12	20:08:37	-0.059	-0.109	1.013	0.694	-0.237	0.694	22.582	-3.237	29.718
5.551099	-73.344796	2020-09-12	20:08:38	-0.055	-0.12	1.012	0.687	-0.221	0.778	22.761	-1.618	27.818
5.551099	-73.344797	2020-09-12	20:08:39	-0.053	-0.118	1.014	-0.053	-0.313	0.526	22.582	-1.438	27.991
5.551098	-73.344796	2020-09-12	20:08:40	-0.054	-0.104	1.019	1.305	-0.214	0.641	22.582	-1.798	30.064
5.551098	-73.344797	2020-09-12	20:08:41	-0.055	-0.104	1.016	-1.16	-0.71	0.916	22.761	-1.618	30.237
5.551098	-73.344797	2020-09-12	20:08:42	-0.038	-0.168	1.028	-6.935	-2.602	7.896	23.658	-2.517	29.027
5.551098	-73.344796	2020-09-12	20:08:43	-0.034	-0.148	1.025	-3.448	-1.831	0.885	24.554	-0.899	28.509
5.551097	-73.344796	2020-09-12	20:08:44	-0.127	-0.005	0.937	55.054	8.469	-18.646	24.374	0.36	29.027
5.551097	-73.344796	2020-09-12	20:08:45	-0.08	-0.108	0.994	0.572	0.778	1.282	35.307	-3.057	41.986
5.551097	-73.344797	2020-09-12	20:08:46	-0.074	-0.078	1.016	-1.564	0.793	0.229	35.128	-1.438	43.195
5.551097	-73.344797	2020-09-12	20:08:47	-0.07	0.003	1.005	6.989	0.458	2.693	36.562	-2.517	45.269
5.551097	-73.344798	2020-09-12	20:08:48	-0.014	0.011	1.024	8.209	0.168	1.915	35.666	-1.259	45.096
5.551102	-73.344797	2020-09-12	20:08:49	-0.059	0.006	1.022	-1.053	-0.458	4.181	35.307	-2.338	45.787
5.551105	-73.344795	2020-09-12	20:08:50	-0.024	0.051	1.023	-6.584	-13.542	9.88	37.099	-0.18	45.441
5.551106	-73.344792	2020-09-12	20:08:51	-0.106	-0.039	1.021	-6.737	-4.898	-4.639	37.637	2.517	45.269
5.551106	-73.344791	2020-09-12	20:08:52	-0.068	-0.044	0.939	-8.247	-2.251	-9.75	38.712	0.719	42.504
5.551106	-73.344789	2020-09-12	20:08:53	-0.141	-0.152	1.099	-12.398	-5.196	-4.135	34.411	-3.956	44.923
5.551107	-73.344788	2020-09-12	20:08:54	-0.089	-0.147	1.065	-9.392	-3.693	8.507	33.336	-2.877	41.467
5.551112	-73.344785	2020-09-12	20:08:55	0.039	-0.112	0.995	7.904	5.516	12.367	33.515	-5.214	40.604
5.551116	-73.344777	2020-09-12	20:08:56	-0.028	-0.054	1.081	7.072	4.089	-12.276	37.637	1.798	40.085
5.551122	-73.344771	2020-09-12	20:08:57	-0.1	-0.118	0.882	-10.483	-2.007	-6.256	33.515	-1.978	40.604
5.551127	-73.344766	2020-09-12	20:08:58	-0.096	-0.142	1.141	6.531	-16.624	10.872	33.873	-3.776	42.331
5.551135	-73.34476	2020-09-12	20:08:59	-0.062	-0.1	1.023	11.063	0.084	11.482	33.873	-1.978	40.949
5.551144	-73.344755	2020-09-12	20:09:00	-0.039	-0.129	0.889	-4.829	-3.494	-0.9	34.053	-2.517	40.776
5.551152	-73.344751	2020-09-12	20:09:01	-0.044	-0.155	1.014	-5.722	1.93	-2.815	35.486	-1.079	40.431
5.551159	-73.344747	2020-09-12	20:09:02	-0.085	-0.291	1.104	-5.836	10.414	-10.757	36.741	-0.899	41.295
5.551165	-73.344742	2020-09-12	20:09:03	-0.074	-0.131	0.996	13.931	4.112	-21.362	33.873	-2.338	38.185
5.551169	-73.344739	2020-09-12	20:09:04	-0.094	-0.142	1.053	-8.308	-5.646	-3.189	30.289	-5.574	39.567
5.551171	-73.344738	2020-09-12	20:09:05	-0.009	-0.11	1.013	7.362	3.899	-4.227	26.167	-6.833	38.357
5.551175	-73.344736	2020-09-12	20:09:06	-0.117	-0.213	1.008	-18.745	-11.566	7.385	27.959	-6.114	39.394
5.551178	-73.344734	2020-09-12	20:09:07	-0.156	-0.188	1.04	-11.398	-12.154	21.706	32.977	-4.315	38.703

Figura 54: Datos generados por el sensor inercial MPU 9250 y GPS guardados en el archivo txt. Fuente: Autores.

Los archivos obtenidos en formato kml se pueden visualizar en programas como Google Earth o en aplicaciones online como GPS visualizer, lo cual permite visualizar los archivos generados en la misma Raspberry (Figura 55) u otros dispositivos de ser necesario.

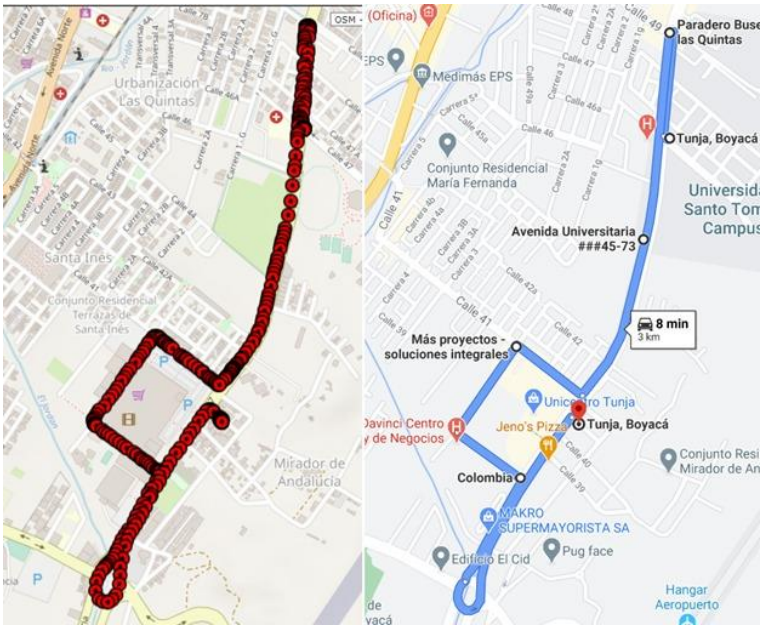

Figura 55: Archivo kml de la ruta visualizada online (Izquierda), Ruta GPS Android (Derecha). Fuente: Autores.

Estadio de la Independencia

Para definir el punto de ubicación de la estación base se decidió por un punto más alto esto mejoraría la recepción de la señal, razón por la cual se optó por colocar la antena en un edificio a una altura de 20 metros y deshabilitar el limitador de corriente que tiene los puertos USB del Raspberry permitiendo entregar mayor potencia al módulo XBee.

En esta última etapa de prueba se puede visualizar la interfaz gráfica de usuario (Figura 56) con las respectivas graficas de velocidad, aceleración y orientación del vehículo (Figura 57), al mismo tiempo que se generan la ruta en un archivo kml (Figura 58) y el archivo txt donde se guardan todos los datos de los sensores y módulos (Figura 59) en el dispositivo implementado en el vehículo como también en la estación base, en este recorrido el sistema implementado en el vehículo no tiene pérdida de puntos (Figura 58).

Figura 56: Interfaz gráfica de usuario. Fuente: Autores.

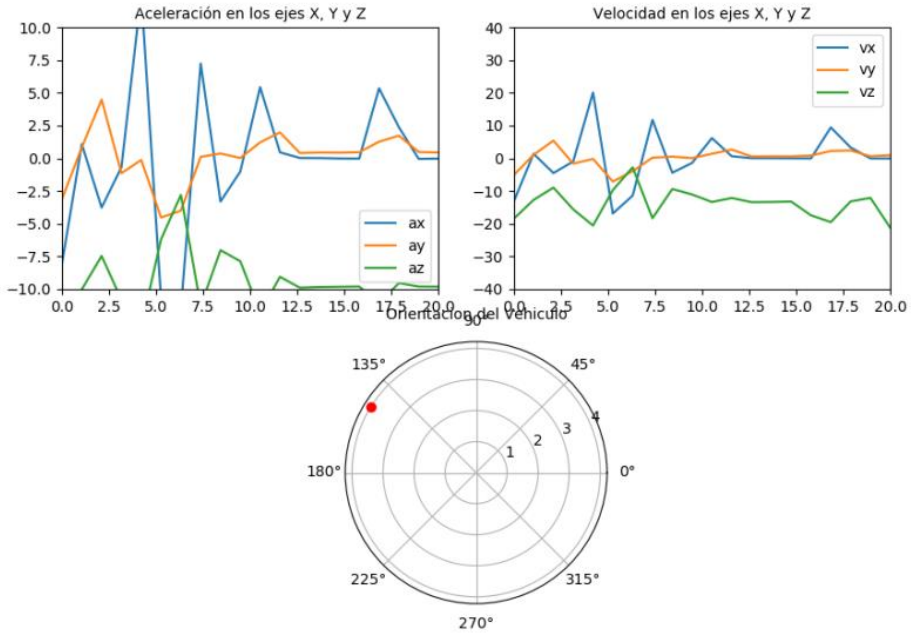

Figura 57: Graficas de Aceleración, Velocidad y Orientación del vehículo. Fuente: Autores.

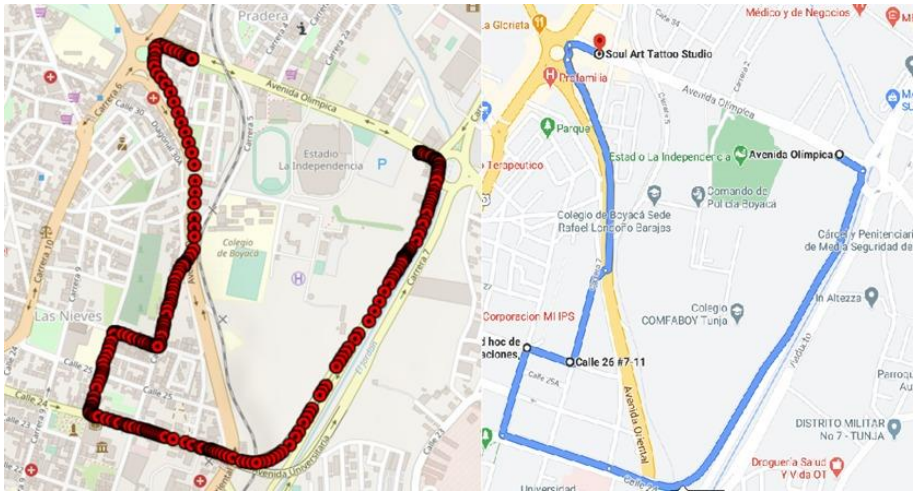

Figura 58: Ruta en archivo kml generada por el prototipo en la última prueba y visualizada online (Izquierda), Ruta GPS Android (Derecha). Fuente Autores.

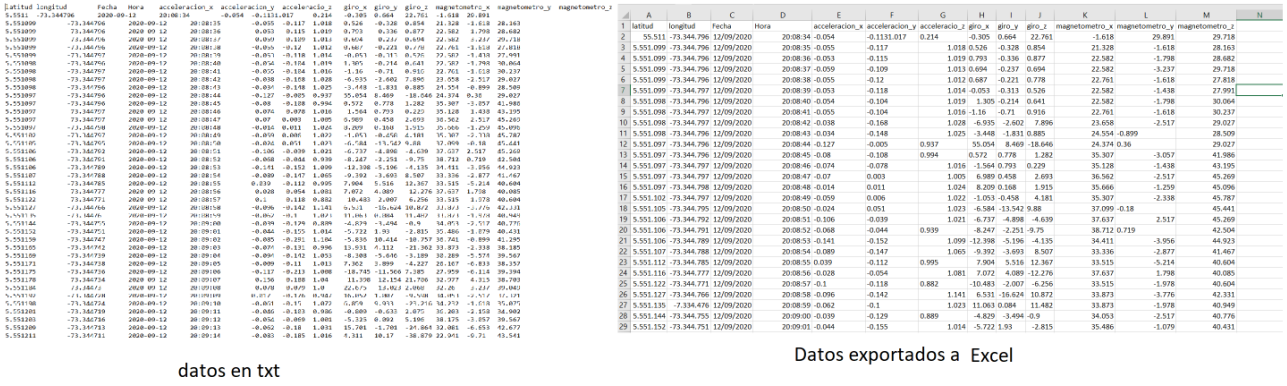

Figura 59: Datos obtenidos de los sensores guardados en el archivo txt con posibilidad de exportación a Excel. Fuente: Autores.

Además de los datos visualizados anteriores también se hace la recolección de imágenes por medio de la cámara infrarroja con una resolución de 300x300 pixeles (Figura 60), la recolección de estas se hace de 10 imágenes por segundo guardándose en una carpeta para revisarlas al terminar el recorrido.

Figura 60: Fotos capturadas por el prototipo con resolución de 300x300 pixeles. Fuente: Autores.

En las siguientes figuras se muestran las rutas tomadas por la estación base en las dos últimas ubicaciones donde se realizaron las pruebas, correspondientes a las pruebas realizadas en la Ruta de Fratellos (Figura 61) y la Ruta del Estadio de la Independencia (Figura 62).

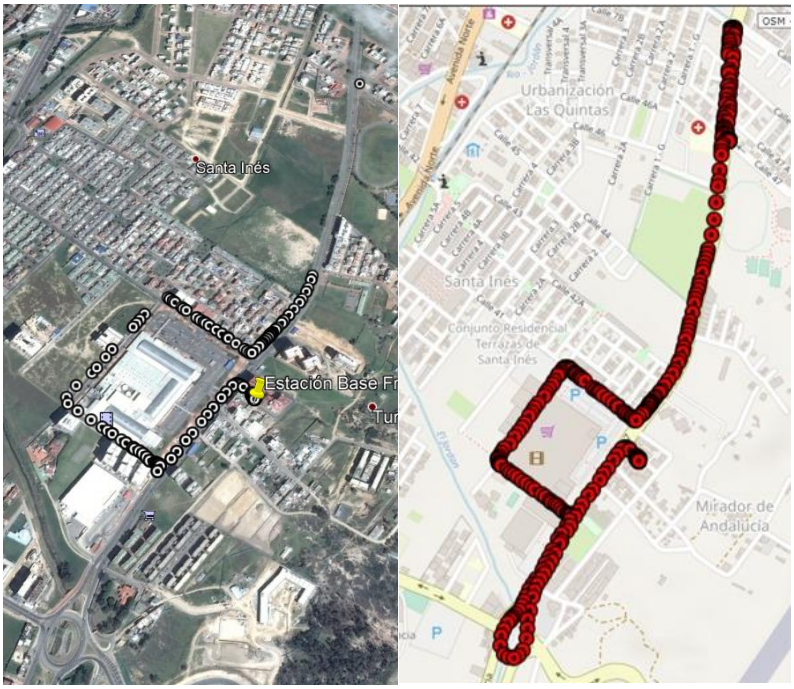

Figura 61: Ruta generada por la Estación Base ubicada en Fratelos (Izquierda) y Ruta generada por el sistema implementado en el vehículo (Derecha). Fuente: Autores.

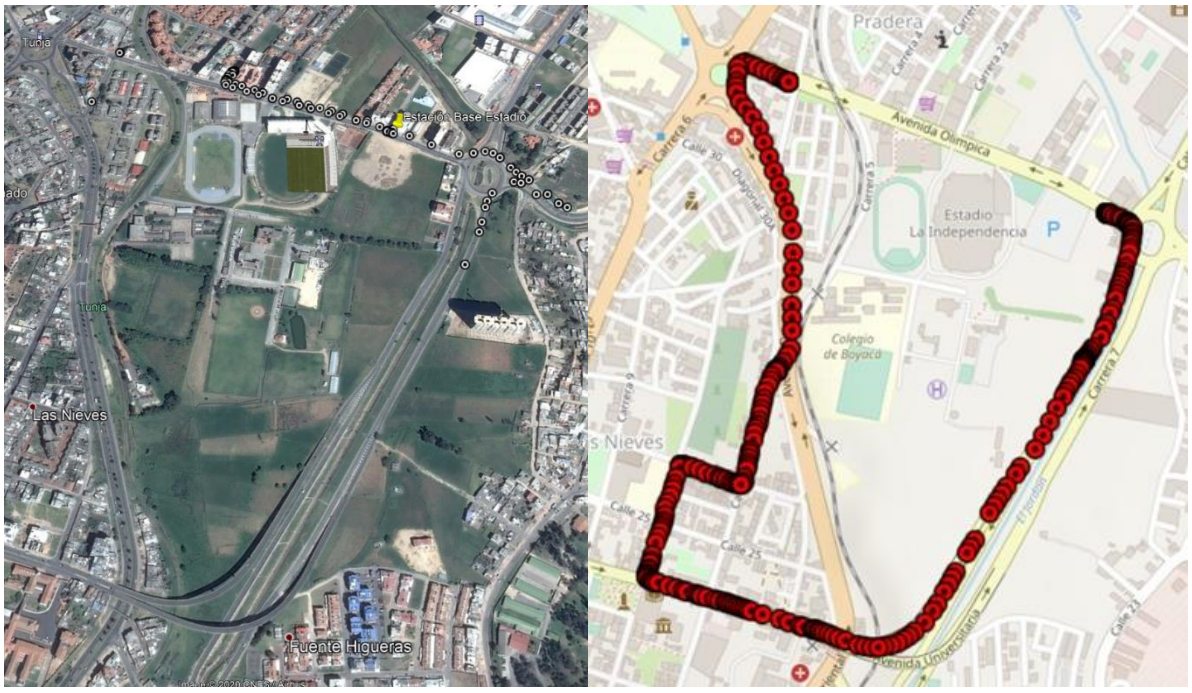

Figura 62: Ruta generada por la Estación Base ubicada cerca al Estadio de la Independencia (Izquierda) y Ruta generada por el sistema implementado en el vehículo (Derecha). Fuente: Autores.

Como se observa en la Figura 61 y en la Figura 62 los datos que recibe la Estación Base en comparación a los del sistema implementado en el vehículo son muy pocos y no alcanzan a

describir la ruta completa de las pruebas realizadas, en la Talaba 11 se puede ver una comparación de los datos del sistema implementado en el vehículo con los datos que lee la estación base mostrando la diferencia entre las dos recolecciones.

	Ruta Estadio	**Ruta Fratellos**
Datos recolectados en el sistema ADAS del Vehiculó	955	752
Datos recolectados en la Estación Base respectiva	130	293
Datos no recibidos por la Estación Base respectiva	825	459

Tabla 11: Datos recolectados en las rutas y en la estación base. Fuente: Autores.

Con los datos de la Tabla 11 podemos determinar la Tasa de Error la cual con la siguiente formula se determina:

$$BER = \frac{\#\ de\ bits\ con\ error}{\#\ total\ de\ bits\ enviados}$$

Donde # de bits con error para el prototipo correspondería al número de datos que no recibió la Estación Base y el # total de bits enviados se refiere a los datos enviados por el sistema implementado en el vehículo.

Para la Estación Base ubicada en la Ruta del Estadio la Independencia la Tasa de Error correspondería a un 86% como se ve en la siguiente ecuación:

$$BER = \frac{825}{955} = 0.864\ \acute{O}\ 86\%$$

Y para la Estación Base ubicada en la Ruta de Fratellos la Tasa de Error es de 61% como se expresa en la siguiente ecuación:

$$BER = \frac{459}{752} = 0.61\ \acute{O}\ 61\%$$

VIII. CONCLUSIONES

El desarrollo del prototipo muestra que es posible implementar un sistema de asistencia al conductor, de fácil acceso y económico dado que sistemas de esta clase en el mercado internacional pueden llegar a costar entre $5.000 y $15.000 dólares; este prototipo cuando se implemente tendrá un costo de $800.000 a $900.000 pesos colombianos y será adaptable a cualquier tipo de vehículo.

El prototipo de asistencia al conductor ADAS para detección de seudoaccidentes, se desarrolló e implementó cumpliendo con los requisitos establecidos, se utilizó un módulo GPS, un sensor inercial, una cámara infrarroja y un módulo de comunicación IoT XBee, elementos que son de fácil acceso en el mercado y de bajo costo, además la interfaz gráfica de usuario la cual permite visualizar graficas de velocidad, aceleración y orientación del vehículo.

Con el análisis del estado del arte de los diferentes sistemas ADAS, embebidos y tecnologías IoT se pudo llegar a comprender como funcionan este tipo de asistentes al conductor y cuáles eran los mejores sensores, sistemas embebidos y tecnologías IoT que facilitan la implementación y mejora de estos sistemas ADAS.

La unidad de recepción o estación base se implementó con comunicación XBee permitiendo la recepción de los datos provenientes del prototipo ADAS e igual que el sistema implementado en el vehículo, esta unidad cuenta con una interfaz gráfica de usuario que muestra las gráficas de velocidad, aceleración y orientación del vehículo.

Las pruebas realizadas con el prototipo ADAS implementado en el vehículo muestran resultados positivos para el sistema debido a que se tiene una adquisición de datos de todos los sensores sin perdidas, los datos obtenidos del inercial son de gran ayuda para las gráficas que se muestran en las interfaz de usuario y se recolectan una buena cantidad de imágenes con la cámara teniendo 10 tomas por segundo, todos estos datos de almacenan de forma correcta en la memoria de la Raspberry para su acceso en el momento sea necesario.

El desarrollo del prototipo de asistencia al conductor para detección de seudoaccidentes, tiene como propósito servir de herramienta para mejorar la seguridad de conductores y poder reducir los índices de accidentalidad en las vías de Colombia, de sistemas de este tipo pueden depender gran cantidad de vidas humanas.

Los módulos de comunicación XBee se escogieron debido a que se pueden realizar enlaces de hasta 300m ubicando la estación base en una zona alta y que esta tenga línea de vista con el prototipo implementado en el vehículo para tener una mejor recepción de datos.

El algoritmo de predicción de ruta es de gran ayuda para el sistema debido a que por túneles o fallas en la conexión con los satélites se puede llegar a presentar pérdida de cobertura, el algoritmo predice la ruta y se puede continuar generando el archivo el cual guarda la ruta realizada por el vehículo.

IX. RECOMENDACIONES

Es recomendable, en modificaciones futuras, mejorar el diseño de la carcasa del sistema por uno que sea hermético para evitar la mala manipulación de los elementos que compone el sistema además de incluir un sistema que avise al coordinador si el sistema lo intentan abrir.

Para modificaciones futuras también se tiene previsto agregar alertas de exceso de velocidad en el vidrio frontal del vehículo, además de un indicador de proximidad en el vidrio posterior que alerte al vehículo que se encuentre detrás que tan cerca esta.

Se sugiere para tener una mejor transmisión de datos, colocar la estación base en un lugar alto que también tenga línea de vista con el sistema implementado en el vehículo, además es recomendable poner replicadores a un radio de 300 metros de la estación base, esto debido a que en las pruebas realizada y los datos que recolectó la Estación Base realizando medidas desde la aplicación Google Earth en las dos últimas ubicaciones de prueba correspondientes a la ruta de Fratellos y la ruta del Estadio de la Independencia (Figura 63 y Figura 64), los datos que la estación alcanza a leer llegan hasta una distancia promedio de 312 metros. La distancia para colocar los repetidores depende también del entorno y puede ser que en otros entornos la distancia entre replicadores pueda variar, para ese caso se recomienda primero realizar un análisis de distribución de replicadores.

Figura 63: Distancias tomadas en la ruta generada por la Estación Base de Fratellos en los tres puntos hasta los cuales se recibieron datos. Fuente: Autores.

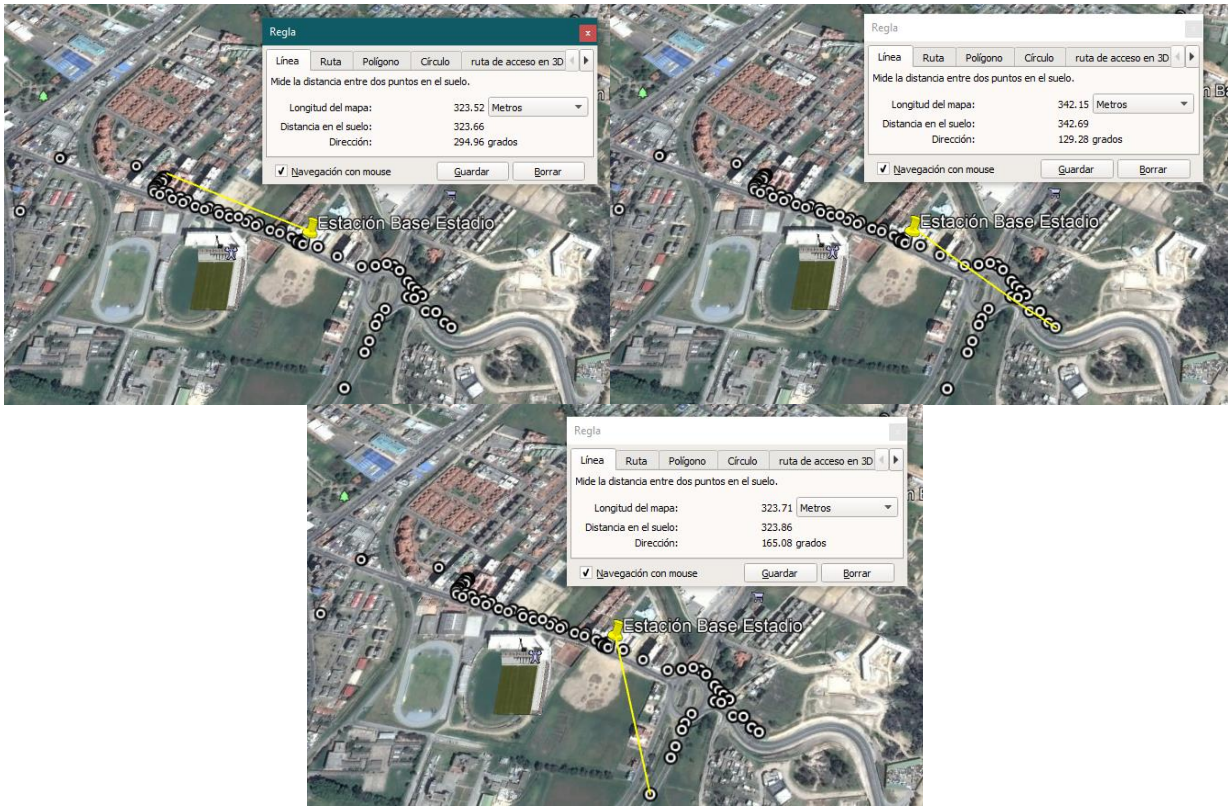

Figura 64: Distancias tomadas en la ruta generada por la Estación Base del Estadio de la independencia en los tres puntos hasta los cuales se recibieron datos. Fuente. Autores.

BIBLIOGRAFÍA

[1] Agencia Nacional de Seguridad Vial – Observatorio Nacional de Seguridad Vial. *Siniestros de transito 2019*. Obtenido de: https://ansv.gov.co/observatorio/index2b89.html?op=Contenidos&sec=76&page=79.

[2] Agencia Nacional de Seguridad Vial – Observatorio Nacional de Seguridad Vial. *Cifras definitivas. Comparativos 2018 2019 fatalidades*. Obtenido de: https://ansv.gov.co/observatorio/index0978.html?op=Contenidos&sec=76&page=80.

[3] Parque automotor registrado en RUNT julio de 2020. https://www.runt.com.co/runt-en-cifras/parque-automotor?field_fecha_de_la_norma_value%5Bvalue%5D%5Byear%5D=2020.

[4] www.andemos.org. (2020). *Informe Vehículos 2020-07*. Obtenido de: http://www.andemos.org/wp-content/uploads/2020/08/Informe-Vehiculos-2020-07.pdf.

[5] https://www.latinncap.com/es/resultados.

[6] República de Colombia - Ministerio de Transporte. (26 de 03 de 2019). *Transporte en Cifras - Estadísticas 2018*.

[7] Tamara Giménez Rodríguez María Elena Ros Bernabéu (2009) *Gravitación y Astrofísica* [Archivo PDF]. ESPAÑA. Recuperado de https://webs.um.es/bussons/GPSresumen_TamaraElena.pdf.

[8] Ing. Julio A. Capdevila (2018) *El Sistema de Posicionamiento Global GPS – Ansenuza* [Archivo pdf]. ARGENTINA. Recuperado de https://ansenuza.unc.edu.ar/comunidades/bitstream/handle/11086.1/1258/El%20Sistema%20de%20Posicionamiento%20Global.pdf?sequence=1&isAllowed=y.

[9] Álvarez Cárdenas, Edison Teodoro (2008). *Importancia de los cálculos astronómicos para la navegación* [archivo pdf]. Universidad Austral de Chile. recuperado de http://cybertesis.uach.cl/tesis/uach/2008/bmfcia473i/doc/bmfcia473i.pdf.

[10] *Sistemas de Navegación Inercial* [archivo pdf] Universidad de Sevilla. Ingeniería aeroespacial. Recuperado de http://www.aero.us.es/fna/files/T8FNA_print.pdf.

[11] Ing. Nelson Javier Rodríguez (2014) *Modelamiento Físico Matemático de un Cuadroptero* [archivo pdf] Colombia. Escuela de suboficiales FAC. Recuperado de file:///C:/Users/Cristian/Downloads/447-Texto%20del%20art%C3%ADculo-1643-1-10-20151103.pdf.

[12] Alfred. *Historia de los módulos XBee*. Recuperado de https://www.timetoast.com/timelines/historia-de-los-modulos-xbee.

[13] raspeberry.org official page https://www.raspberrypi.org/products/raspberry-pi-1-model-a-plus/.

[14] raspeberry.org official page https://www.raspberrypi.org/products/raspberry-pi-4-model-b/.

[15] Torretavira (2015) *História de las cámaras oscuras* [archivo pdf]. Recuperado de https://www.torretavira.com/wp-content/uploads/2015/09/camaras_oscuras.pdf.

[16] Martine Soria y Alfredo Romero. *EL HONRADO INVENTOR DE LA FOTOGRAFÍA* [archivo pdf]. Recuperado dehttp://cultura.dpz.es/ficheros/documentos/culturatjd/copy_of_z1_niepce.pdf.

[17] La primera fotografía de la historia (2014) https://expansion.mx/especiales/2014/05/01/la-primera-fotografia-de-la-historia.

[18] https://gopro.com/en/co/shop/hero8-black/tech-specs?pid=CHDHX-801-master.

[19] https://ferretronica.com/collections/raspberry-pi/products/camara-para-raspberry-pi-1080p-hd-video-webcam-generica.

[20] https://www.robotshop.com/us/es/kit-camara-odseven-para-raspberry-pi-3-b-5mp-c-version-ajuste-focal-nocturno.html.

[21] S. Tsugawa, T. Saito and A. Hosaka, *"Super smart vehicle system: AVCS related systems for the future"* Proceedings of the Intelligent Vehicles `92 Symposium, Detroit, MI, USA, 1992, pp. 132-137.

doi: 10.1109/IVS.1992.252245 URL: http://ieeexplore.ieee.org.crai-ustadigital.usantotomas.edu.co/stamp/stamp.jsp?tp=&arnumber=252245&isnumber=6442.

[22] D. Wetzel, H. Niemann and S. Richter, *"An image sequence analysis system for driver assistance"*, Proceedings of ANZIIS '94 - Australian New Zealnd Intelligent Information Systems Conference, Brisbane, Queensland, Australia, 1994, pp. 150-154.

doi: 10.1109/ANZIIS.1994.396931 URL: http://ieeexplore.ieee.org.crai-ustadigital.usantotomas.edu.co/stamp/stamp.jsp?tp=&arnumber=396931&isnumber=8985.

[23] C. Stiller, W. Pochmuller and B. Hurtgen, *"Stereo vision in driver assistance systems"*, Proceedings of Conference on Intelligent Transportation Systems, Boston, MA, USA, 1997, pp. 888-893.

doi: 10.1109/ITSC.1997.660591 URL: http://ieeexplore.ieee.org.crai-ustadigital.usantotomas.edu.co/stamp/stamp.jsp?tp=&arnumber=660591&isnumber=14437.

[24] G. Wanielik, N. Appenrodt, H. Neef, R. Schneider and J. Wenger, *"Polarimetric millimeter wave imaging radar and traffic scene interpretation"*, IEE Colloquium on Automotive Radar and Navigation Techniques (Ref. No. 1998/230), London, UK, 1998, pp. 4/1-4/7.
doi: 10.1049/ic:19980190 URL: http://ieeexplore.ieee.org.crai-ustadigital.usantotomas.edu.co/stamp/stamp.jsp?tp=&arnumber=668223&isnumber=14729.

[25] LEY 769 DE 2002 CODIGO NACIONAL DE TRANSITO

[26] https://licuadodeletras5.wordpress.com/los-sistemas-embebidos-y-su-importancia-en-la-actualidad/.

[27] Controladores industriales de diseño de alto nivel http://www.ieec.uned.es/investigacion/Dipseil/PAC/archivos/Informacion_de_referencia_ISE5_3_1.pdf.

[28] https://ec.europa.eu/transport/road_safety/sites/roadsafety/files/ersosynthesis2016-adas15_en.pdf.

[29] SISTEMA DE POSICIONAMIENTO GLOBAL (GPS) https://webs.um.es/bussons/GPSresumen_TamaraElena.pdf.

[30] MPU-9250 Nine-Axis https://www.invensense.com/products/motion-tracking/9-axis/mpu-9250/.

[31] Blog Electrónica Diseño (2015). *Wireless Mesh Networking ZigBee vs. DigiMesh*. Recuperado de http://hqwertyuiop.blogspot.com/2015/10/wireless-mesh-networking-zigbee-vs.html.

[32] http://legal.legis.com.co/document/Index?obra=legcol&document=legcol_56c4f1c1d33448718d7b6ae2ed0bfde.

[33] https://naylampmechatronics.com/sensores-posicion-inerciales-gps/106-modulo-gps-neo-6m-u-blox.html.

[34] https://ferretronica.com/products/antena-gps-activa-rango-de-frecuencia-1575-42-mhz-27-db?_pos=2&_sid=8f2a33826&_ss=r.

[35] https://naylampmechatronics.com/sensores-posicion-inerciales-gps/214-modulo-mpu9250-acelerometro-giroscopio-magnetometro-i2c.html.

[36] https://cdtechnologia.net/raspberry-pi/665-camara-con-infrarrojo-para-raspberry-pi-5mp-ir.html.

[37] https://xbee.cl/producto/xbee-pro-900hp-s3b-point2multipoint-900mhz-250mw-rpsma-10kbps/.

[38] https://www.dynamoelectronics.com/tienda/antena-dipolo-24ghz-5dbi-rp-sma/.

[39] A. Fortuño. *Desarrollo e implementación de una red de sensores Zigbee mediante el dispositivo Xbee de Digi*, Universitat Rovira i Virgili, 2012. URL: http://deeea.urv.cat/public/PROPOSTES/pub/pdf/1957pub.pdf.

[40] https://www.ecured.cu/Topolog%C3%ADas_de_red_ZigBee.

[41] G. S. Olivar. *Trabajo de Tesis: Diseño e implementación de un prototipo de posicionamiento híbrido GPS/GNSS + INS con comunicación IoT*, Morelos. Universidad Politécnica del Estado de Morelos, 2020.

[42] C. Treffers, L. van Wietmarschen. *Position and orientation determination of a probe with use of the IMU MPU9250 and a ATmega328 microcontroller*. TUDelf, 2016.

ANEXOS

Código prototipo ADAS

```
import FaBo9Axis_MPU9250#libreria del inercial
import pynmea2 # libreria de gps
import time#libreria de timer
import serial#comunicacion serial de gps y xbee
import RPi.GPIO as gpio # activacion de los pines gpio
import sys#
import simplekml# para la cracion de archivos kml
import math# para ecuaciones matematicas
import threading# para el uso de hilos
import numpy as np #generar las ecuaciones que definen la grafica
import matplotlib.pyplot as plt # generar las graficas
from collections import deque
import cv2#libreria de opencv para la camara
from tkinter import * #Libreria para la interfaz grafica

raiz = Tk()
raiz.title('ADAS Tx')
raiz.config(bg='#000000')

miframe=Frame(raiz)
miframe.pack(fill='both', expand='1')
miframe.config(bg='#FFFACD')

logo=PhotoImage(file='LogoUstaTunja.png')
Label(miframe, image=logo, bd=0).grid(row=0, column=0)

fac="""FACULTAD DE INGENIERÍA ELECTRÓNICA"""
Label(miframe, text=fac, justify='center', fg='#000000', bg='#FFFACD', font=('Arial', 16)).grid(row=1, column=0)

titulo="""DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO ADAS
CON COMUNICACIÓN IOT (XBEE) Y
RECOLECCIÓN DE IMÁGENES PARA DETECCIÓN DE SEUDOACCIDENTES"""
Label(miframe, text=titulo, justify='center', fg='#000000', bg='#FFFACD', font=('Arial', 24), padx=10, pady=10).grid(row=2, column=0)

nombres="""ESTUDIANTES: Paula Rojas, Cristian Salas  TUTORES: Ing. Cesar Galarza, Ing. Luis Castellanos"""
Label(miframe, text=nombres, justify='center', fg='#000000', bg='#FFFACD', font=('Arial', 12)).grid(row=3, column=0)

mpu9250 = FaBo9Axis_MPU9250.MPU9250()
cap = cv2.VideoCapture(0)
kml=simplekml.Kml() #Crea el archivo kml
ser=serial.Serial('/dev/ttyUSB0',9600,timeout=0.5)# configuracion xbee

data='Buscando GPS...'   #definir variables string
datafoto='Foto'
datafiltro='Filtro'

def gps():
    global data           #variables globales
    global datafoto
    global datafiltro
    gpio.setmode(gpio.BCM) #Da acceso a los puertos GPIO
    gps=serial.Serial('/dev/ttyS0',baudrate=9600,timeout=0.5) #configuracion serial gps


    while True:
        data=gps.readline().decode('utf-8') #Lee los datos del GPS
        datafoto=data
        datafiltro=data
```

```
def filtro():
    #Variables para filtros de los sensores
    comparevalueax=0
    previuscorrectax=0
    NPax=0.2
    NPSax=0.15
    NPLax=0.46

    comparevalueay=0
    previuscorrectay=0
    NPay=0.3
    NPSay=0.2250
    NPLay=0.46

    comparevalueaz=-1
    previuscorrectaz=-1
    NPaz=0.3
    NPSaz=0.2250
    NPLaz=0.46

    comparevaluegx=0
    previuscorrectgx=0
    NPgx=30
    NPSgx=30
    NPLgx=0

    comparevaluegy=0
    previuscorrectgy=0
    NPgy=30
    NPSgy=30
    NPLgy=0

    comparevaluegz=0
    previuscorrectgz=0
    NPgz=30
    NPSgz=30
    NPLgz=0

    alpha=0.85                #Variable alpha para el filtro complementario
    roll=0.1                  #Variables de angulos en 0.1 para no dividir entre 0
    pitch=0.1
    yaw=0.1

    #Variables auxiliares para el calculo de la integración
    nax0=0
    nvx0=0
    nauxvx0=0
    nay0=0
    nvy0=0
    nauxvy0=0
    naz0=0
    nvz0=0
    nauxvz0=0
    nnvx0=0
    dx0=0
    auxdx0=0
    nnvy0=0
    dy0=0
    auxdy0=0
    nnvz0=0
```

```
nauxvz0=0
nnvx0=0
dx0=0
auxdx0=0
nnvy0=0
dy0=0
auxdy0=0
nnvz0=0
dz0=0
auxdz0=0

valueax=0
valueay=0
valueaz=0
valuegx=0
valuegy=0
valuegz=0

latval=0

nrvx=0
nrvy=0
nrvz=0

while latval==0:                            #Ciclo para que el algoritmo no se ejecute hasta tenr una referencia inicial
    if (datafiltro[0:6] == '$GPRMC'):
        msg=pynmea2.parse(datafiltro)
        latval=float(msg.latitude)
        longval=float(msg.longitude)

t1=time.time()

while True:

        if (datafiltro[0:6])=='$GPRMC':
            msg=pynmea2.parse((datafiltro)) #lee los datos del GPS en la linea GPGGA
            #Datos GPS
            latval=float(msg.latitude) #Latitud
            longval=float(msg.longitude) #Longitud
            fecha=msg.datestamp #Fecha
            hora=msg.timestamp #Hora
            hour=int(str(hora)[0:2])-5 #Pasar a UTC-5 hora local Colombia
            hora=str(hour)+str(hora)[2:]

            tramatxt='/home/pi/Desktop/FicheroTx'+str(fecha)+'.txt'
            archivo=open(tramatxt,'a')

            #Datos inarcial MPU9265 calibrado
            accel=mpu9250.readAccel() #Acelerometro(x,y,z)
            ax = accel['x'] - 0.724660
            ay = accel['y'] + 0.315037
            az = accel['z'] - 0.07775

            gyro=mpu9250.readGyro() #Giroscopio(x,y,z)
            gx = gyro['x'] + 1.348519
            gy = gyro['y'] + 1.147669
            gz = gyro['z'] - 0.056487
```

```
gyro=mpu9250.readGyro() #Giroscopio(x,y,z)
gx = gyro['x'] + 1.348519
gy = gyro['y'] + 1.147669
gz = gyro['z'] - 0.056487

mag=mpu9250.readMagnet() #Magnetometro(x,y,z)
mx = mag['x'] - 18
my = mag['y'] - 16
mz = mag['z']

t2=time.time()
dT=t2-t1                        #Obtencion del tiempo entre muestras
t1=time.time()

#Creación de tramas
trama="GPSor(Lat: "+str(latval)+'\tLon: '+str(longval)+'\tFecha: '+str(fecha)+'\tHora: '+str(hora)+')\tINER
        (Acel: x='+str(ax)+'\ty='+str(ay)+'\tz='+str(az)+'\tGiro: x='+str(gx)+'\ty='+str(gy)+'\tz='+str(gz)+
        '\tMag: x='+str(mx)+'\ty='+str(my)+'\tz='+str(mz)+")\n"
print(trama)
archivo.write(trama)
archivo.close()

h=2800                          #Definición de la altura promedio de Tunja para la conversión de coordenadas
f=1/298.257                     #Variables necesarias para la conversion de coordenadas geograficas a cartesianas
a=6378140
b=(1-f)*a
e=math.sqrt(1-((b**2)/(a**2)))

if latval!=0 or longval!=0:  #Si la posicion es valida, se realiza la conversion de coordenadas geograficas a cartesianas
    rlat=math.radians(latval)
    rlon=math.radians(longval)
    N=a/math.sqrt(1-((e**2)*(math.sin(rlon))**2))
    x=(N+h)*math.cos(rlat)*math.cos(rlon)
    y=(N+h)*math.cos(rlat)*math.sin(rlon)
    z=(N*(1-e**2)+h)*math.sin(rlat)

    roll=0.1                      #Se regresan todas las variables a su valor inicial
    pitch=0.1
    yaw=0.1

    nax0=0
    nvx0=0
    nauxvx0=0zz
    nay0=0
    nvy0=0
    nauxvy0=0
    naz0=0
    nvz0=0
    nauxvz0=0
    nnvx0=0
    dx0=0
    auxdx0=0
    nnvy0=0
    dy0=0
    auxdy0=0
    nnvz0=0
    dz0=0
    auxdz0=0
```

```
    valueax=0
    valueay=0
    valueaz=0
    valuegx=0
    valuegy=0
    valuegz=0
    nrvx=0
    nrvy=0
    nrvz=0

else:     #Si la posicion no es valida (Se pierde cobertura), se ejecuta el algoritmo
    valueax=ax              #Filtros para las mediciones del INS
    if (valueax < comparevalueax-NPax or valueax > comparevalueax+NPax):
        filteredvalueax=valueax
    else:
        if(valueax < comparevalueax-NPSax or valueax > comparevalueax+NPSax):
            filteredvalueax=(valueax + previuscorrectax)/2
        else:
            filteredvalueax=(1-NPLax)*previuscorrectax + NPLax*previuscorrectax
        comparevalueax=filteredvalueax
        previuscorrectax=filteredvalueax
        rax=filteredvalueax

        raxan=rax/16438         #Se divide la medicion entre el LSB del acelerometro para una mejor precision

        valueay=ay
        if (valueay < comparevalueay-NPay or valueay > comparevalueay+NPay):
            filteredvalueay=valueay
        else:
            if(valueay < comparevalueay-NPSay or valueay > comparevalueay+NPSay):
                filteredvalueay=(valueay + previuscorrectay)/2
            else:
                filteredvalueay=(1-NPLay)*previuscorrectay + NPLay*previuscorrectay
        comparevalueay=filteredvalueay
        previuscorrectay=filteredvalueay
        ray=filteredvalueay

        rayan=ray/16438

        valueaz=az
        if (valueaz < comparevalueaz-NPaz or valueaz > comparevalueaz+NPaz):
            filteredvalueaz=valueaz
        else:
            if(valueaz < comparevalueaz-NPSaz or valueaz > comparevalueaz+NPSaz):
                filteredvalueaz=(valueaz + previuscorrectaz)/2
            else:
                filteredvalueaz=(1-NPLaz)*previuscorrectaz + NPLaz*previuscorrectaz
        comparevalueaz=filteredvalueaz
        previuscorrectaz=filteredvalueaz
        raz=filteredvalueaz

        razan=raz/16438
        if (valuegx < comparevaluegx-NPgx or valuegx > comparevaluegx+NPgx):
            filteredvaluegx=valuegx
        else:
            if(valuegx < comparevaluegx-NPSgx or valuegx > comparevaluegx+NPSgx):
                filteredvaluegx=(valuegx + previuscorrectgx)/2
            else:
                filteredvaluegx=(1-NPLgx)*previuscorrectgx + NPLgx*previuscorrectgx
```

```
rgx=filteredvaluegx/131       #Se divide la medicion entre el LSB del giroscopio para una mejor precision
valuegy=gy
if (valuegy < comparevaluegy-NPgy or valuegy > comparevaluegy+NPgy):
    filteredvaluegy=valuegy
else:
    if(valuegy < comparevaluegy-NPSgy or valuegy > comparevaluegy+NPSgy):
        filteredvaluegy=(valuegy + previuscorrectgy)/2
    else:
        filteredvaluegy=(1-NPLgy)*previuscorrectgy + NPLgy*previuscorrectgy
rgy=filteredvaluegy/131

valuegz=gz
if (valuegz < comparevaluegz-NPgz or valuegz > comparevaluegz+NPgz):
    filteredvaluegz=valuegz
else:
    if(valuegz < comparevaluegz-NPSgz or valuegz > comparevaluegz+NPSgz):
        filteredvaluegz=(valuegz + previuscorrectgz)/2
    else:
        filteredvaluegz=(1-NPLgz)*previuscorrectgz + NPLgz*previuscorrectgz
rgz=filteredvaluegz/131

aroll=math.atan(rayan/math.sqrt((raxan**2)+(razan**2)))       #Calculo de roll
apitch=math.atan(-raxan/math.sqrt((rayan**2)+(razan**2)))     #Calculo de pitch
roll=alpha*(rgx+roll)-(1-alpha)*aroll*180/math.pi         #filtro complementario para el calculo de los angulos
pitch=alpha*(rgy+pitch)-(1-alpha)*apitch*180/math.pi
rroll=math.radians(roll)                        #conversion de grados a radianes de los angulos
rpitch=math.radians(pitch)

nmagx=mx*math.cos(rpitch)+mz*math.sin(rpitch)       #Calculo del angulo yaw
nmagy=mx*math.sin(rroll)*math.sin(rpitch)+my*math.cos(rroll)-mz*math.sin(rroll)*math.cos(rpitch)
yaw=90-math.atan(nmagy/nmagx)*180/math.pi
ryaw=math.radians(yaw)                          #conversion de grados a radianes del angulo yaw

nax=rax*9.81                                   #compensaciones de gravedad para las mediciones de aceleracion
nay=ray*9.81
naz=raz*9.81

alphain=(2*math.pi*0.7*dT)/(2*math.pi*0.7*dT+1)  #calculo de alpha para la integracion de la aceleracion

nvx=nvx0+0.5*dT*(nax+nax0)                      #obtencion de velocidades (primer integracion)
nauxvx=(1-alphain)*nauxvx0+alphain*nvx0
nauxvx0=nauxvx
nrvx=nvx-nauxvx
nvx0=nvx
nax0=nax

nvy=nvy0+0.5*dT*(nay+nay0)
nauxvy=(1-alphain)*nauxvy0+alphain*nvy0
nauxvy0=nauxvy
nrvy=nvy-nauxvy
nvy0=nvy
nay0=nay

nvz=nvz0+0.5*dT*(naz+naz0)
nauxvz=(1-alphain)*nauxvz0+alphain*nvz0
nauxvz0=nauxvz
nrvz=nvz-nauxvz
nvz0=nvz
naz0=naz
```

```
        dx=dx0+0.5*dT*(nrvx+nnvx0)                      #Obtension del desplazamiento (Segunda integración)
        auxdx=(1-alphain)*auxdx0+alphain*dx0
        auxdx0=auxdx
        rdx=dx-auxdx
        dx0=dx
        nnvx0=nrvx

        dy=dy0+0.5*dT*(nrvy+nnvy0)
        auxdy=(1-alphain)*auxdy0+alphain*dy0
        auxdy0=auxdy
        rdy=dy-auxdy
        dy0=dy
        nnvy0=nrvy

        dz=dz0+0.5*dT*(nrvz+nnvz0)
        auxdz=(1-alphain)*auxdz0+alphain*dz0
        auxdz0=auxdz
        rdz=dz-auxdz
        dz0=dz
        nnvz0=nrvz

                        #Conversion del desplazamiento en coordenadas body a coordenadas NED
        xn=math.cos(rpitch)*math.cos(ryaw)*rdx+(math.cos(ryaw)*math.sin(rpitch)*math.sin(rroll)-math.sin(ryaw)
                *math.cos(rroll))*rdy+(math.cos(ryaw)*math.sin(rpitch)*math.cos(rroll)+math.sin(ryaw)
                *math.sin(rroll))*rdz
        yn=math.cos(rpitch)*math.sin(ryaw)*rdx+(math.cos(ryaw)*math.cos(rroll)+math.sin(ryaw)*math.sin(rpitch)
                *math.sin(rroll))*rdy+(math.sin(ryaw)*math.sin(rpitch)*math.cos(rroll)-math.cos(ryaw)
                *math.sin(rroll))*rdz
        zn=-math.sin(rpitch)*rdx+math.cos(rpitch)*math.sin(rroll)*rdy+math.cos(rpitch)*math.cos(rroll)*rdz

            #Conversion del desplazamiento en coordenadas NED a coordenadas ECEF (Con la suma de la referencia inicial)
        x=-math.sin(rnlat)*math.cos(rnlon)*xn-math.sin(rnlon)*yn+(-math.cos(rnlat)*math.cos(rnlon))*zn+x
        y=-math.sin(rnlat)*math.sin(rnlon)*xn+math.cos(rnlon)*yn+(-math.cos(rnlat)*math.sin(rnlon))*zn+y
        z=math.cos(rnlat)*xn-math.sin(rnlat)*zn+z

        #Conversion de coordenandas ECEF a LLA(Latitud, longitud, altura)
nlon=math.degrees(math.atan(y/x))
P=math.sqrt((x**2) +(y**2))
error=0.00001
nlatan=0
nlatnew=math.degrees(math.atan((z/P)*(1/(1-((a*(e**2))/math.sqrt((P**2)+(z**2)))))))
errorcal=nlatnew-nlatan

while errorcal>error:
    nlatan=nlatnew
    nlatanrad=math.radians(nlatan)
    nN=a/math.sqrt(1-((e**2)*((math.sin(nlatanrad))**2)))
    nalt=(((P/((math.cos(nlatanrad))))-nN)-22000)*10-70
    nlatnew=math.degrees(math.atan((z/P)*(1/(1-(e**2)*(nN/(nN+nalt))))))
    errorcal=nlatnew-nlatan
nlat=nlatnew
rnlon=math.radians(nlon)     #Conversion de latitud y longitud a radianes (Calculada por el GPS o por el algoritmo)
rnlat=math.radians(nlat)
nlon=round(nlon,10)          #Redondeo de latitud y longitud a 10 decimales (Para envio correcto de trama)
nlat=round(nlat,10)
tramakml='FicheroTx_'+str(fecha)+'.kml'
coo=kml.newpoint(coords=[(nlon,nlat)]) #coordenadas para el kml
coo.style.iconstyle.icon.href= 'http://maps.google.com/mapfiles/kml/shapes/placemark_circle_highlight.png'#icono de puntos rojos
```

```
                trama1="GPSmo(Lat: "+str(nlat)+'\tLon: '+str(nlon)+'\tFecha: '+str(fecha)+'\tHora: '+str(hora)+')\t
                        INER(Acel: x='+str(ax)+'\ty='+str(ay)+'\tz='+str(az)+'\tGiro: x='+str(gx)+'\ty='+str(gy)+'\
                        tz='+str(gz)+'\tMag: x='+str(mx)+'\ty='+str(my)+'\tz='+str(mz)+")\n"
                print(trama1)
                tramatxt='/home/pi/Desktop/FicheroTx'+str(fecha)+'.txt'
                archivo=open(tramatxt,'a')
                archivo.write(trama1)
                archivo.close()
                #Envia la trama de datos al Xbee Rx
                tramalong='lon,'+str(nlon)+"\n"
                ser.write(str.encode((tramalong)))
                tramalat='lat,'+str(nlat)+"\n"
                ser.write(str.encode((tramalat)))
                tramafe='fec,'+str(fecha)+'\n'
                ser.write(str.encode((tramafe)))
                tramadato="dat,"+"Lat: "+str(nlat)+"\tLon: "+str(nlon)+"\tFecha: "+str(fecha)+"\tHora: "+str(hora)+
                        '\tAcel: x='+str(ax)+'\ty='+str(ay)+'\tz='+str(az)+'\tGiro: x='+str(gx)+'\ty='+str(gy)+'\tz='+str(gz)+
                        '\tMag: x='+str(mx)+'\ty='+str(my)+'\tz='+str(mz)+ "OK""\n"
                ser.write(str.encode(tramadato))
                #data2=ser.readline().decode()# Leer el puerto serial
                #mensaje5a = data2.find("A1")
                #if mensaje5a !=-1:
                #    print('Dato recibido')
                #else:
                #    coo=kml.newpoint(coords=[(nlon,nlat)]) #coordenadas para el kml
                #    coo.style.iconstyle.icon.href= 'http://maps.google.com/mapfiles/kml/shapes/placemark_circle.png'#icono de puntos rojos
                #    print('perdida de cobertura')
                #kml.save(tramakml)


def capfotos():
    n=0
    trama2='Captura foto'
    while True:
        ret, frame = cap.read()
        if (datafoto[0:6])=='$GPRMC':
            msg=pynmea2.parse((datafoto)) #lee los datos del GPS en la linea GPRMC
            fecha=msg.datestamp #Fecha
            hora=msg.timestamp #Hora
            hour=int(str(hora)[0:2])-5 #Pasar a UTC-5 hora local Colombia
            hora=str(hour)+str(hora)[2:]
            trama2=str(fecha)+' '+str(hora) #Trama para la imagen

        n=n+1 #Contador para nombrar las fotos
        if n==200:
            n=0
        time.sleep(0.05)

        framere=cv2.resize(frame, (300, 300), interpolation=cv2.INTER_AREA) #Se hace un reajuste del tamaño de las images a 300x300
        nombre= str(trama2)+'_'+str(n)+".jpg"  #Nombre de las imagenes

        carpeta= '/home/pi/Desktop/Fotos/'+nombre #carpeta donde se guarda
        cv2.imwrite(carpeta, framere)   #Guardar imagen
    cap.release() #Comando para cerrar el puerto de la camara

def graficas():

    #Variables auxiliares para el calculo de la integración
    nax0=0
    nvx0=0
```

```
nauxvx0=0
nay0=0
nvy0=0
nauxvy0=0
naz0=0
nvz0=0
nauxvz0=0
nnvx0=0

t1=time.time()

x=np.linspace(0,20,20) #Generar los datos en x
#fig, (acx,vcx)=plt.subplots(2,figsize=(7,5)) #Crear ventana para mostrar las graficas
fig = plt.figure(figsize=(10,7))
acx=plt.subplot(2,2,1)
vcx=plt.subplot(2,2,2)
plr=plt.subplot(2,1,2, projection='polar')



onda=deque([0]*20,maxlen=20) #Genera vector de 0 a 20
onda1=deque([0]*20,maxlen=20) #Genera vector de 0 a 20
onda2=deque([0]*20,maxlen=20) #Genera vector de 0 a 20
linea,=acx.plot(x, onda, label='ax') #Genera vector de 0 a 20
linea1,=acx.plot(x, onda1, label='ay') #Graficar la aceleracion en y
linea2,=acx.plot(x, onda2, label='az') #Graficar la aceleracion en z
acx.set_title('Aceleración en los ejes X, Y y Z',fontsize=10) #Se le da un Titulo a la grafica de Aceleración
acx.legend() #Grafica el cuadro de leyendas

onda3=deque([0]*20,maxlen=20) #Genera vector de 0 a 20
onda4=deque([0]*20,maxlen=20) #Genera vector de 0 a 20
onda5=deque([0]*20,maxlen=20) #Genera vector de 0 a 20
linea3,=vcx.plot(x, onda3, label='vx') #Graficar la velocidad en x
linea4,=vcx.plot(x, onda4, label='vy') #Graficar la velocidad en y
linea5,=vcx.plot(x, onda5, label='vz') #Graficar la velocidad en z
vcx.set_title('Velocidad en los ejes X, Y y Z',fontsize=10) #Se le da un Titulo a la grafica de Velocidad
vcx.legend() #Grafica el cuadro de leyendas

onda6=deque([0]*1,maxlen=1)  #Genera vector de 0  a 1
linea6,=plr.plot(onda6, 4, 'ro') #Graficar la orientacion en el polar
plr.set_title('Orientacion del Vehiculo',fontsize=10) #Se le da un Titulo a la grafica de Aceleración

while True:
    accel = mpu9250.readAccel() #Acelerometro(x,y,z)
    ax = accel['x'] + 0.035113333333333
    ay = accel['y'] + 0.0347386666666666
    az = accel['z'] - 0.0925453333333376

    gyro=mpu9250.readGyro() #Giroscopio(x,y,z)
    gx = gyro['x'] - 0.358664000000001
    gy = gyro['y'] + 0.783637999999999
    gz = gyro['z'] - 0.0767860000000003

    mag=mpu9250.readMagnet() #Magnetometro(x,y,z)
    mx = mag['x']
    my = mag['y']+5
    mz = mag['z']

    t2=time.time()
```

```
dT=t2-t1                        #Obtencion del tiempo entre muestras
t1=time.time()

nax0=0                    #Se regresan todas las variables a su valor inicial
nvx0=0
nauxvx0=0
nay0=0
nvy0=0
nauxvy0=0
naz0=0
nvz0=0
nauxvz0=0
nnvx0=0

nax=ax*9.81                                  #compensaciones de gravedad para las mediciones de aceleracion
nay=ay*9.81
naz=az*9.81

alphain=(2*math.pi*0.7*dT)/(2*math.pi*0.7*dT+1)  #calculo de alpha para la integracion de la aceleracion

nvx=nvx0+0.5*dT*(nax+nax0)                       #obtencion de velocidades (primer integracion)
nauxvx=(1-alphain)*nauxvx0+alphain*nvx0
nauxvx0=nauxvx
nrvx=nvx-nauxvx
nvx0=nvx
nax0=nax

nvy=nvy0+0.5*dT*(nay+nay0)
nauxvy=(1-alphain)*nauxvy0+alphain*nvy0
nauxvy0=nauxvy
nrvy=nvy-nauxvy
nvy0=nvy
nay0=nay

nvz=nvz0+0.5*dT*(naz+naz0)
nauxvz=(1-alphain)*nauxvz0+alphain*nvz0
nauxvz0=nauxvz
nrvz=nvz-nauxvz
nvz0=nvz
naz0=naz

heading=math.atan2(my,mx)
if(heading<0.0):
    heading=heading+(2*math.pi)
if(heading>(2*math.pi)):
    heading=heading-(2*math.pi)
heading=heading*180/math.pi
heading=heading*(math.pi/180)

onda.append(nax) #Se le da reescribe el vector con valores de ax
onda1.append(nay) #Se le da reescribe el vector con valores de ay
onda2.append(naz) #Se le da reescribe el vector con valores de az
linea.set_ydata(onda) #Genera la linea de grafica para ax
linea1.set_ydata(onda1) #Genera la linea de grafica para ay
linea2.set_ydata(onda2) #Genera la linea de grafica para az
acx.set_ylim(-10,10) #Limite del eje y
acx.set_xlim(0,20) #Limite del eje x
plt.pause(0.1) #Tiempo de espera para graficar
```

```
        onda3.append(nrvx) #Se le da reescribe el vector con valores de nrvx
        onda4.append(nrvy) #Se le da reescribe el vector con valores de nrvy
        onda5.append(nrvz) #Se le da reescribe el vector con valores de nrvz
        linea3.set_ydata(onda3) #Genera la linea de grafica para nrvx
        linea4.set_ydata(onda4) #Genera la linea de grafica para nrvy
        linea5.set_ydata(onda5) #Genera la linea de grafica para nrvz
        vcx.set_ylim(-40,40) #Limite del eje y
        vcx.set_xlim(0,20) #Limite del eje x
        plt.pause(0.1) #Tiempo de espera para graficar

        onda6.append(heading)
        linea6.set_xdata(onda6)
        plt.pause(0.1)

        tramaax='iax,'+str(ax)+'\n'
        ser.write(str.encode((tramaax)))
        tramaay='iay,'+str(ay)+'\n'
        ser.write(str.encode((tramaay)))
        tramaaz='iaz,'+str(az)+'\n'
        ser.write(str.encode((tramaaz)))
        tramamx='imx,'+str(mx)+'\n'
        ser.write(str.encode((tramamx)))
        tramamy='imy,'+str(my)+'\n'
        ser.write(str.encode((tramamy)))
        tramavx='vlx,'+str(nrvx)+'\n'
        ser.write(str.encode((tramavx)))
        tramavy='vly,'+str(nrvy)+'\n'
        ser.write(str.encode((tramavy)))
        tramavz='vlz,'+str(nrvz)+'\n'
        ser.write(str.encode((tramavz)))


higps=threading.Thread(target=gps) #Crea el hilo de gps
hicapfotos=threading.Thread(target=capfotos) #Crea el hilo para capturar las fotos
hifiltro=threading.Thread(target=filtro) #Crea el hilo del anlgoritmo de predicción
#higraficas=threading.Thread(target=graficas) #Crea el hilo para las graficas
higps.start() #Inicia hilo de gps
hicapfotos.start() #Inicia hilo para capturar las fotos
hifiltro.start() #Inicia hilo del anlgoritmo de predicción
#higraficas.start() #Inicia hilo para las graficas

Button(miframe, text='Iniciar Gráficas', bg='#B0E0E6', activebackground='#5F9EA0', font=('Arial', 20), justify='center',
       pady=10, command=graficas).grid(row=4, column=0)

raiz.mainloop()
```

Código Estación Base

```
import serial # Importa la librearia para utilizar comunicacion serial
import simplekml # Importa la libreria para la cracion de los archivos kml(google earth)
from tkinter import * # Importa la libreria para crear la interfaz gráfica
import time#libreria de timer
import math# para ecuaciones matematicas
import threading# para el uso de hilos
import numpy as np #generar las ecuaciones que definen la grafica
import matplotlib.pyplot as plt # generar las graficas
from collections import deque

raiz = Tk()
raiz.title('ADAS Rx')
raiz.config(bg='#000000')

miframe = Frame(raiz)
miframe.pack(fill='both', expand='1')
miframe.config(bg='#FFFACD')

logo=PhotoImage(file='LogoUstaTunja.png')
Label(miframe, image=logo, bd=0).grid(row=0, column=0)

fac="""FACULTAD DE INGENIERÍA ELECTRÓNICA"""
Label(miframe, text=fac, justify='center', fg='#000000', bg='#FFFACD', font=('Arial', 16)).grid(row=1, column=0)

titulo="""DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO ADAS
CON COMUNICACIÓN IOT (XBEE) Y
RECOLECCIÓN DE IMÁGENES PARA DETECCIÓN DE SEUDOACCIDENTES"""
Label(miframe, text=titulo, justify='center', fg='#000000', bg='#FFFACD', font=('Arial', 24), padx=10, pady=10).grid(row=2, column=0)

nombres="""ESTUDIANTES: Paula Rojas, Cristian Salas  TUTORES: Ing. Cesar Galarza, Ing. Luis Castellanos"""
Label(miframe, text=nombres, justify='center', fg='#000000', bg='#FFFACD', font=('Arial', 12)).grid(row=3, column=0)

#archivo=open('/home/pi/Desktop/FicheroRx(1).txt','w')#abre el archivo txt para guarar los datos
ser=serial.Serial('/dev/ttyUSB0',9600,timeout=0.5) #Configuracion y creacion del puerto serial
print ("Buscando recepcion del Xbee...") # Mensaje de recepcion

kml=simplekml.Kml()#Se genera el archivo kml
lon=0 #Inician las variables de las cordenadas
lat=0
ax=0
ay=0
az=0
vx=0
vy=0
vz=0
mx=0
my=0


def datos():
    global lon
    global lat
    global ax
    global ay
    global az
    global vx
    global vy
    global vz
    global mx
```

```
global my

while True:
    data=ser.readline().decode()# Leer el puerto serial

    if (data[0:3])=='lon':# Se obtiene la longitud de la trama de datos
        lon=float(str(data)[4:])
        print('Longitud: ',lon)

    if (data[0:3])=='lat':# Se obtiene la latidud de la trama de datos
        lat=float(str(data)[4:])
        print('Latitud: ',lat)

    if (data[0:3])=='fec':# Se obtiene la fecha de la trama de datos
        fecha=str(data)[4:]
        print('Fecha: ',fecha)

    if (data[0:3])=='iax':# Se obtiene la aceleración en x de la trama de datos
        ax=float(str(data)[4:])
        print('ax: ',ax)

    if (data[0:3])=='iay':# Se obtiene la aceleración en y de la trama de datos
        ay=float(str(data)[4:])
        print('ay: ',ay)

    if (data[0:3])=='iaz':# Se obtiene la aceleración en z de la trama de datos
        az=float(str(data)[4:])
        print('az: ',az)

    if (data[0:3])=='vlx':# Se obtiene la velocidad en x de la trama de datos
        print('vx: ',data[4:])
        vx=float(str(data)[4:])
        print(vx)

    if (data[0:3])=='vly':# Se obtiene la velocidad en y de la trama de datos
        vy=float(str(data)[4:])
        print('vy: ',vy)

    if (data[0:3])=='vlz':# Se obtiene la velocidad en z de la trama de datos
        vz=float(str(data)[4:])
        print('vz: ',vz)

    if (data[0:3])=='imx':# Se obtiene dato del magnetometro en x de la trama de datos
        mx=float(str(data)[4:])
        print('mx: ',mx)

    if (data[0:3])=='imy':# Se obtiene dato del magnetometro en y de la trama de datos
        my=float(str(data)[4:])
        print('my: ',my)

    if lon!=0 and lat!=0:# Se verifica el correcto valor de la longitud y latidud
        coo=kml.newpoint(coords=[(lon,lat)]) #Se guardan las coordenadas con un punto
        coo.style.iconstyle.icon.href= 'http://maps.google.com/mapfiles/kml/shapes/placemark_circle.png'#Genera los puntos en el mapa
        kml.save('FicheroRx(1).kml') #se guarda el archivo kml

    if (data[0:3])=='dat': #Se obtiene los datos de la segunda trama para los archivos de la hora
        print('Datos: ',data[4:])
        archivo=open('/home/pi/Desktop/FicheroRx(1).txt','a')#Se abre el archivo txt donde se guara la ora la fecha y las coordenadas
        archivo.write(data[4:])
```

```
def graficas():
    x=np.linspace(0,20,20) #Generar los datos en x
    fig, (acx,vcx)=plt.subplots(2,figsize=(7,5)) #Crear ventana para mostrar las graficas
    onda=deque([0]*20,maxlen=20) #Genera vector de 0 a 20
    onda1=deque([0]*20,maxlen=20) #Genera vector de 0 a 20
    onda2=deque([0]*20,maxlen=20) #Genera vector de 0 a 20
    linea,=acx.plot(x, onda, label='ax') #Genera vector de 0 a 20
    linea1,=acx.plot(x, onda1, label='ay') #Graficar la aceleracion en y
    linea2,=acx.plot(x, onda2, label='az') #Graficar la aceleracion en z
    acx.set_title('Aceleración en los ejes X, Y y Z',fontsize=10) #Se le da un Titulo a la grafica de Aceleración
    acx.legend() #Grafica el cuadro de leyendas

    onda3=deque([0]*20,maxlen=20) #Genera vector de 0 a 20
    onda4=deque([0]*20,maxlen=20) #Genera vector de 0 a 20
    onda5=deque([0]*20,maxlen=20) #Genera vector de 0 a 20
    linea3,=vcx.plot(x, onda3, label='vx') #Graficar la velocidad en x
    linea4,=vcx.plot(x, onda4, label='vy') #Graficar la velocidad en y
    linea5,=vcx.plot(x, onda5, label='vz') #Graficar la velocidad en z
    vcx.set_title('Velocidad en los ejes X, Y y Z',fontsize=10) #Se le da un Titulo a la grafica de Velocidad
    vcx.legend() #Grafica el cuadro de leyendas

    while True:
        onda.append(ax) #Se le da reescribe el vector con valores de ax
        onda1.append(ay) #Se le da reescribe el vector con valores de ay
        onda2.append(az) #Se le da reescribe el vector con valores de az
        linea.set_ydata(onda) #Genera la linea de grafica para ax
        linea1.set_ydata(onda1) #Genera la linea de grafica para ay
        linea2.set_ydata(onda2) #Genera la linea de grafica para az
        acx.set_ylim(-3,3) #Limite del eje y
        acx.set_xlim(0,20) #Limite del eje x
        plt.pause(0.1) #Tiempo de espera para graficar

        onda3.append(vx) #Se le da reescribe el vector con valores de nrvx
        onda4.append(vy) #Se le da reescribe el vector con valores de nrvy
        onda5.append(vz) #Se le da reescribe el vector con valores de nrvz
        linea3.set_ydata(onda3) #Genera la linea de grafica para nrvx
        linea4.set_ydata(onda4) #Genera la linea de grafica para nrvy
        linea5.set_ydata(onda5) #Genera la linea de grafica para nrvz
        vcx.set_ylim(-3,3) #Limite del eje y
        vcx.set_xlim(0,20) #Limite del eje x
        plt.pause(0.1) #Tiempo de espera para graficar

hidatos=threading.Thread(target=datos)
#higraficas=threading.Thread(target=graficas)
hidatos.start()
#higraficas.start()

Button(miframe, text='Iniciar Gráficas', bg='#B0E0E6', activebackground='#5F9EA0', font=('Arial', 20), justify='center', pady=10,
       command=graficas).grid(row=4, column=0)

raiz.mainloop()
```

Manual Técnico y de Usuario

UNIVERSIDAD SANTO TOMAS
PRIMER CLAUSTRO UNIVERSITARIO DE COLOMBIA

MANUAL TECNICO Y DE USAUARIO

DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO ADAS CON COMUNICACIÓN IOT (XBEE) Y RECOLECCIÓN DE IMÁGENES PARA DETECCIÓN DE SEUDOACCIDENTES

Paula Natalia Rojas Machuca

Cristian Andrés Salas Rodríguez

UNIVERSIDAD SANTO TOMÁS

FACULTAD DE INGENIERÍA ELECTRÓNICA

TUNJA, BOYACÁ 2020

Tabla de Contenidos

Lista de Figuras

Lista de Tablas

I. PRESENTACION

Este manual presenta todo lo referente al uso, cuidado e información acerca del Prototipo ADAS con Comunicación IoT (XBee) y Recolección de Imágenes para Detección de Seudoaccidentes el cual es una herramienta tecnológica de ayuda que puede ser implementado en vehículos de cualquier clase para disminuir los accidentes de tránsito en las vías de Colombia.

II. INTRODUCCIÓN

El Prototipo ADAS con Comunicación IoT (XBee) y Recolección de Imágenes para Detección de Seudoaccidentes es un sistema que permite al conductor mirar estadísticas y graficas de la velocidad y aceleración del vehículo en el cual se implemente, se puede ver la orientación del vehículo, además genera un archivo kml con la ruta realizada por el vehículo la cual se puede visualizar mediante Softwares de uso libre online como Google Earth, Google Maps, GPS Visualizer, entre otros programas que lean archivos de este tipo; y también guarda capturas de imágenes que toma con la cámara ubicada en la parte posterior del vehículo.

El manual está diseñado para guiar a los usuarios de vehículos particulares y públicos que deseen implementar el prototipo y puedan realizar el procedimiento de instalación, uso del software y mantenimiento de este en caso de fallas o daños, además se presentan las configuraciones a realizar para el correcto funcionamiento del sistema.

III. OBJETIVO DE LA CARTILLA

Informar a los usuarios acerca del Prototipo ADAS con Comunicación IoT (XBee) y Recolección de Imágenes para Detección de Seudoaccidentes, el uso adecuado del sistema con una breve descripción de instalación en el vehículo así también el modo de empleo del software y la visualización de los diferentes archivos que muestra este programa.

IV. RECOMENDACIONES DE SEGURIDAD

- Lea el presente manual antes de iniciar el proceso de captura de los datos.

- Verifique el correcto funcionamiento del equipo antes de poner en marcha el vehículo.

- En caso de errores detener el vehículo en lugar seguro para verificación de los mismos.

- Cerciórese de la correcta alimentación del dispositivo de acuerdo al diagrama de conexiones y los requerimientos de la fuente.

V. ESPECIFICACIONES TECNICAS

Alimentación	**Dispositivo Local: 5 Vdc a 3 A con entrada tipo C** **Dispositivo remoto: 5 Vdc a 3 A con entrada tipo Mini A**
Dimensiones prototipo	**Dispositivo Local:** 130mmx70mmx150mmx **Dispositivo remoto:** 35mmx67mmx95mm
Grado de protección	**Dispositivo Local:** IP10 **Dispositivo remoto:** IP10
Temperatura de operación	**Dispositivo Local:** 0 a 75° **Dispositivo remoto:** 0 a 75°

Tabla 1: Tabla de especificaciones Técnicas.

VI. DESCRIPCION DEL SISTEMA

El Prototipo ADAS con comunicación IoT (XBee) y recolección de imágenes para detección de seudoaccidentes es un sistema que consta de dos dispositivos: un dispositivo local que guarda y envía datos (fecha, hora, latitud, longitud, aceleración, velocidad, giroscopio, magnetómetro) por medio de comunicación IoT (XBee) a su vez para su visualización al instante, grafica los más relevantes(aceleración, velocidad y orientación) en una interfaz de usuario y guarda las imágenes tomadas por el dispositivo solo de forma local, todo esto funcionando de manera conjunta gracias a una Raspberry 4 modelo B 4 gb.

El segundo dispositivo remoto recibe los datos enviados por el dispositivo local y los guarda en una base de datos en archivos kml y archivos txt y así mismo grafica los datos más relevantes en una interfaz de usuario (aceleración, velocidad y orientación), todo esto funcionando de manera conjunta por una Raspberry Pi 3 Model B+.

VII. DIAGRAMA DE CONEXIONES

El diagrama de conexiones del Prototipo ADAS con comunicación IoT (XBee) y recolección de imágenes para detección de seudoaccidentes consta de dos dispositivos remoto y local

REMOTO

El dispositivo remoto se compone de:

- Raspberry Pi 3 Model B+.
- XBee Pro s3b.
- Ventilador.
- Monitor para visualización opcional.

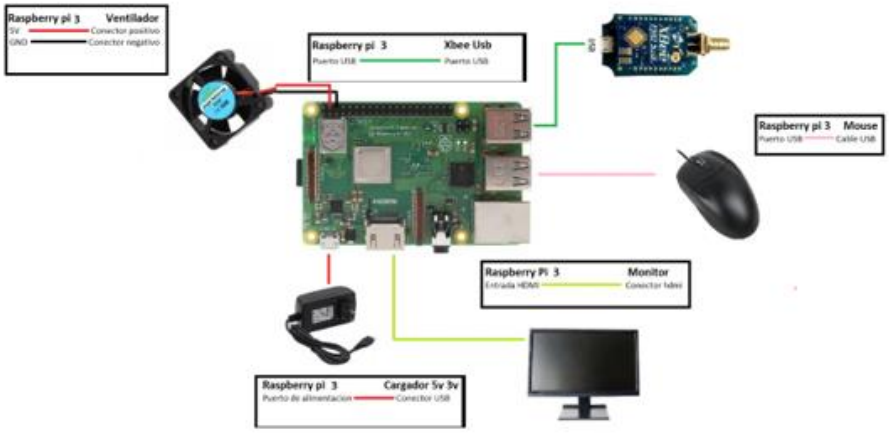

Figura 1: Diagrama de conexión estación remota.

Figura 2: Dispositivo remoto.

LOCAL

El dispositivo local se compone de:

- Raspberry 4, modelo B 4 gb.
- XBee pro s3b.
- Cámara infrarroja para Raspberry de 5 MP.
- Sensor MPU 9250.
- GPS NEO 6m.
- Ventilador 5V.

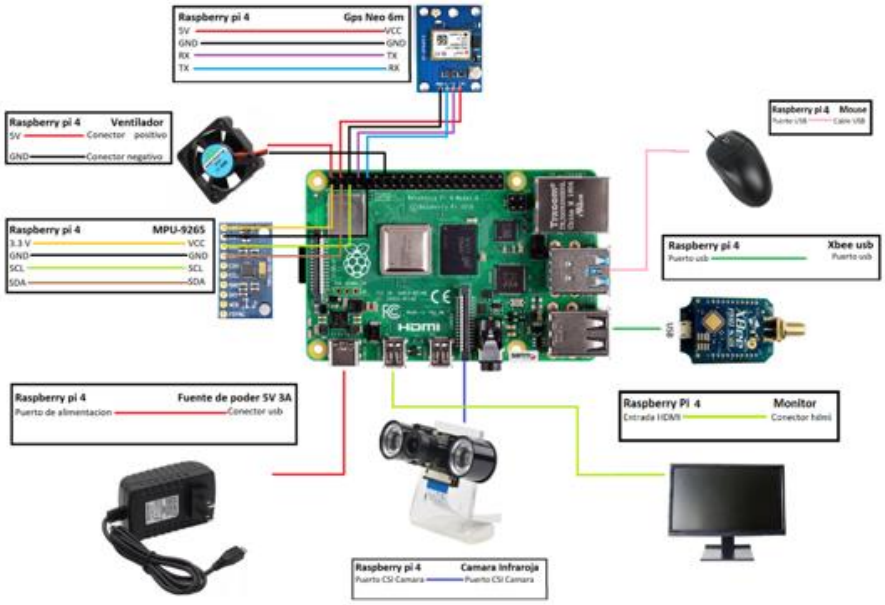

Figura 3: Diagrama de conexión para configuración inicial dispositivo local.

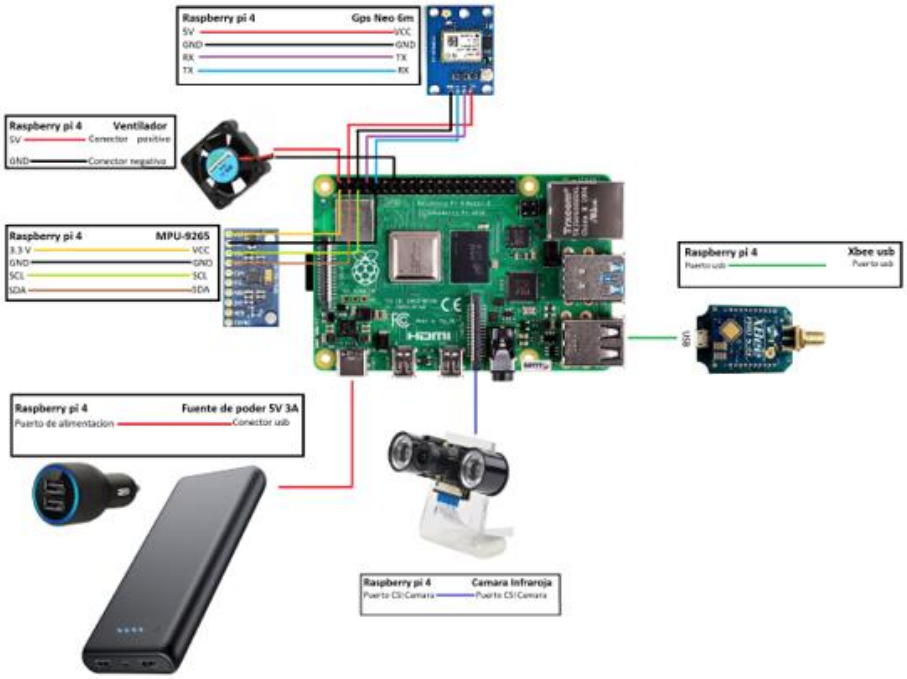

Figura 4: Conexiones del dispositivo local en funcionamiento.

Figura 5 Dispositivo local

VIII. PROCEDIMIENTO PARA FUNCIONAMIENTO

REMOTO

a) Encienda el dispositivo remoto Figura 2, de acuerdo al diagrama de conexiones Figura 1.

b) Después de encendido el dispositivo le debe mostrar la siguiente pantalla.

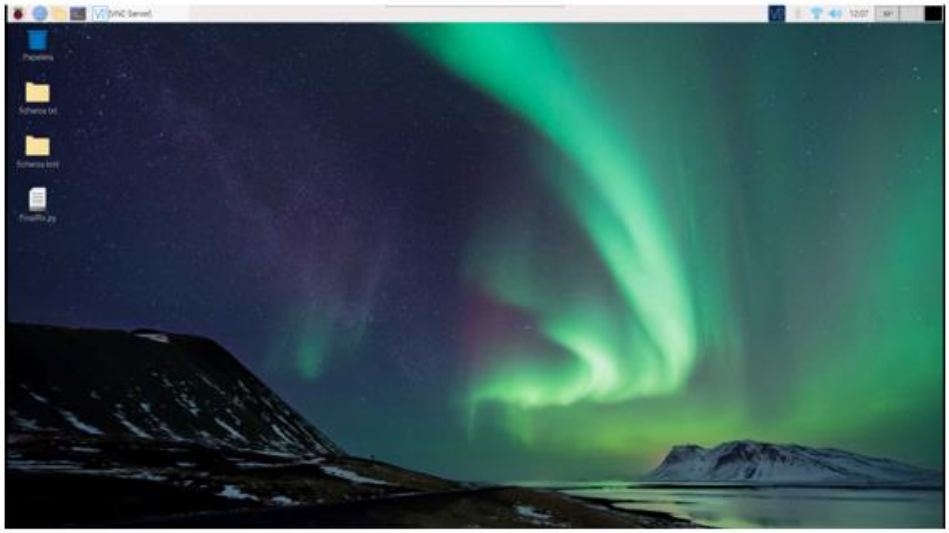

Figura 6: Escritorio del dispositivo remoto.

c) Inicie el archivo ADAS.py, con esto iniciara el programa Thonny, para iniciar nuestra aplicación hacer clic en el botón run.

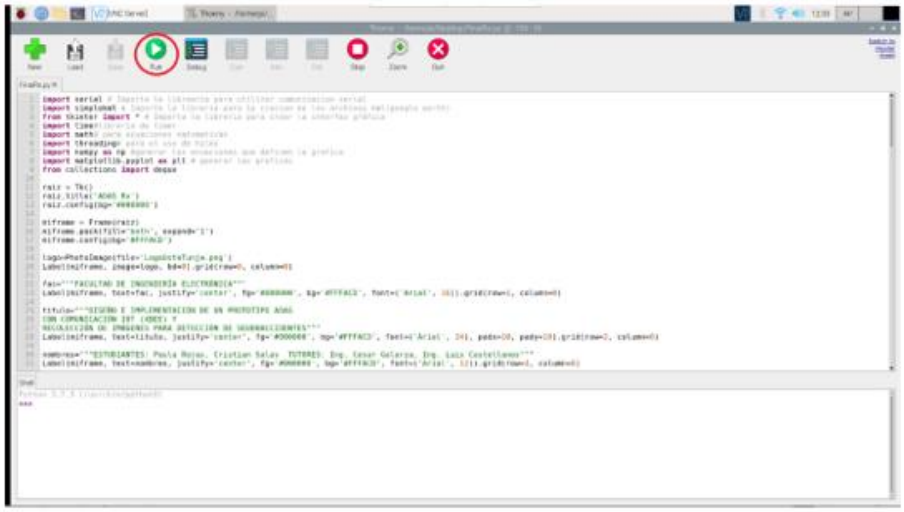

Figura 7: Código del dispositivo remoto.

d) A partir de este momento iniciara el programa de recepción de datos del dispositivo remoto y se comenzaran a guardar los datos en la carpeta ficheros txt los archivos .txt y en la carpeta ficheros kml los archivos .kml ubicadas en el escritorio Figura 6.

Figura 8: Interfaz de Usuario dispositivo remoto.

e) Para poder iniciar el proceso de graficado de los datos se hace clic en Iniciar Graficas.

Figura 9: Graficas en el dispositivo remoto.

f) Cuando se haya finalizado el proceso de toma de datos se hace clic en stop y esto detendrá el proceso de trazado y creación de archivos .kml y .txt.

g) Para la revisión de los datos generados en archivos .txt se ingresa a la carpeta de ficheros txt, donde se encontrarán los archivos organizados por la fecha en que se tomaron los datos Figura 10.

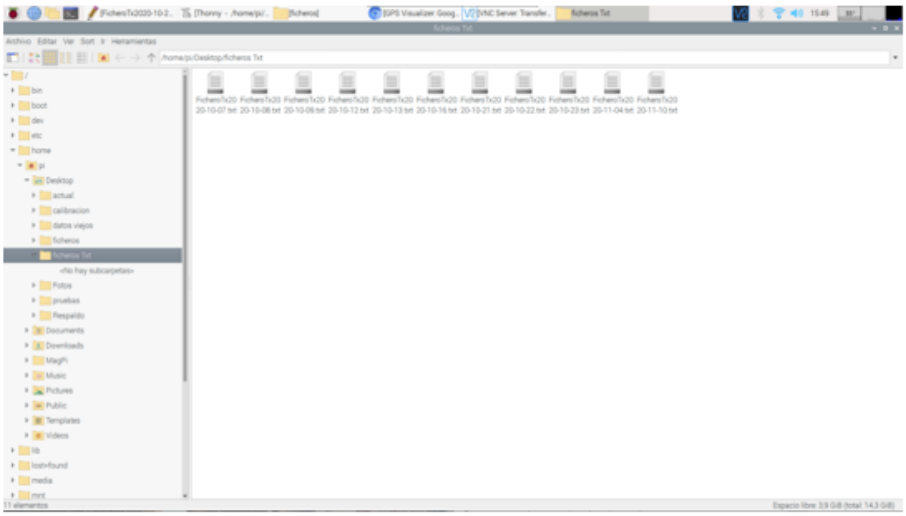

Figura 10: Carpeta Ficheros txt donde se ubican los archivos .txt ordenados por fecha.

h) Si se quiere revisar alguno de los archivos basta con dar doble clic al archivo deseado.

latitud	longitud	Fecha	Hora	acceleracion_x	acceleracion_y	acceleracio_z	giro_x	giro_y	giro_z	magnetometro_x	magnetometro_y	magnetometro_z
5.5511	-73.344796	2019-09-12	20:08:34	-0.054	-0.1131.017	0.214	-0.305	0.664	22.761	-1.618	29.891	
5.551099	-73.344796	2019-09-12	20:08:35	-0.055	-0.117	1.018	0.526	-0.328	0.854	21.328	-1.618	28.163
5.551099	-73.344796	2019-09-12	20:08:36	-0.053	-0.115	1.019	0.793	-0.336	0.877	22.582	-1.798	28.682
5.551099	-73.344796	2019-09-12	20:08:37	-0.059	-0.109	1.013	0.694	-0.237	0.694	22.582	-3.237	29.718
5.551099	-73.344796	2019-09-12	20:08:38	-0.055	-0.12	1.012	0.687	-0.221	0.778	22.761	-1.618	27.818
5.551099	-73.344797	2019-09-12	20:08:39	-0.053	-0.118	1.014	-0.053	-0.313	0.526	22.582	-1.438	27.991

Figura 11: Datos generados en un archivo .txt.

i) Para la revisión de los datos generados en los archivos .kml se ingresa a la carpeta ficheros kml, donde encontrara los ficheros organizados por la fecha en que se tomaron los datos Figura 12.

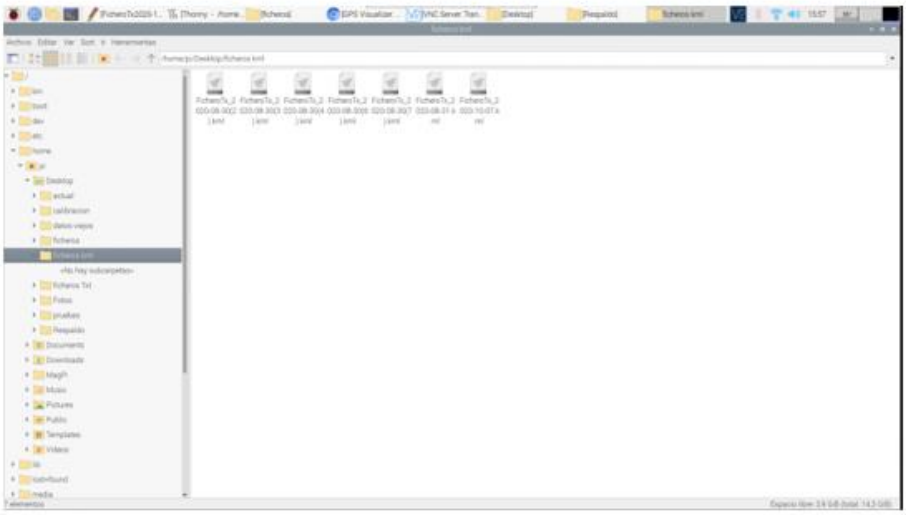

Figura 12: Carpeta Ficheros kml donde se ubican los archivos .kml ordenados por fecha.

j) Para la visualización de estos archivos se ingresa a la página https://www.gpsvisualizer.com/ Figura 13, debe dar clic en seleccionar archivo y se busca el archivo .kml que quiera visualizar Figura 14.

Figura 13: Visualizador online de archivos kml.

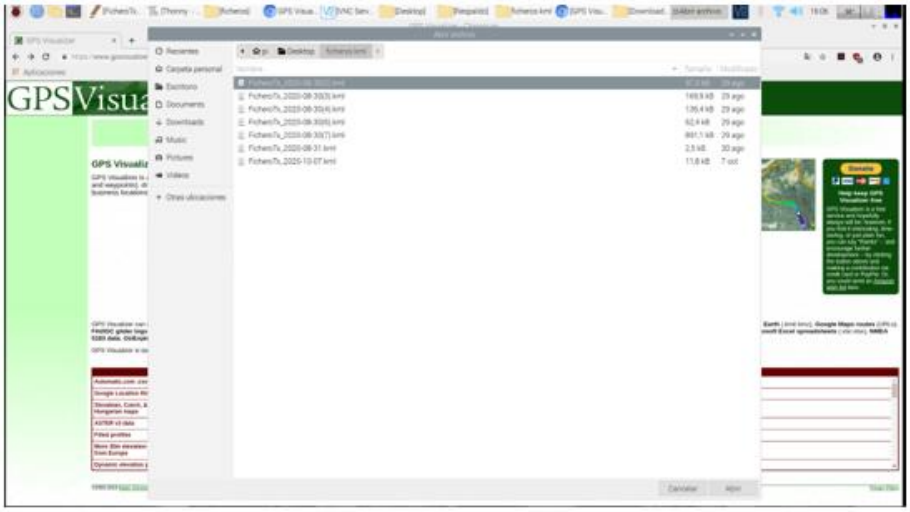

Figura 14: Selección del archivo kml que desea visualizar.

k) A continuación se cargara nuestro archivo seleccionado.

l) El formato de salida es Google Maps, luesgo se hace clic en map it y se podrá visualizar la ruta.

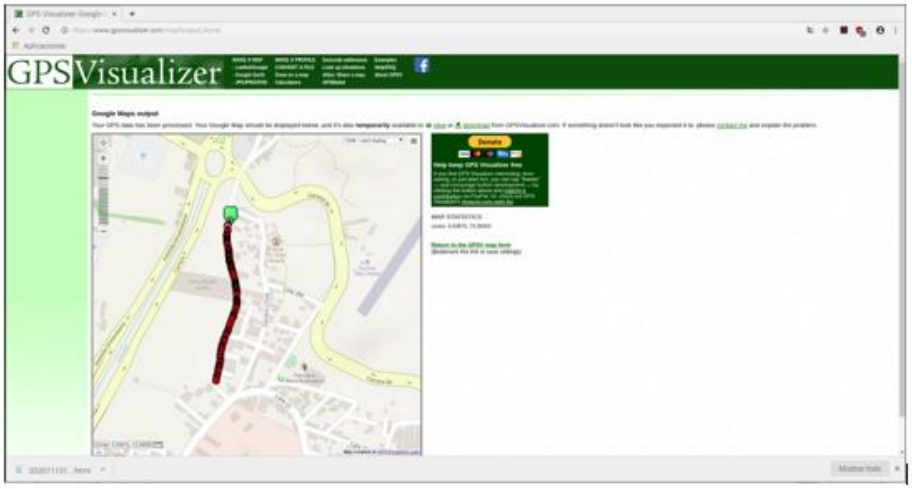

Figura 15: Ruta visualizada en GPS Visualizer.

LOCAL

a) Esta configuración inicial se realiza para poder visualizar los datos en una pantalla y dar inicio al programa en el vehículo, sin la necesidad de una pantalla.

b) Para poder visualizar los datos y las gráficas es necesario conectar el dispositivo con el diagrama de conexión para la configuración inicial Figura 3.

c) Después de encendido el dispositivo, le debe mostrar la siguiente pantalla Figura 16.

Figura 16: Escritorio del dispositivo local.

d) Primero utilizando un dispositivo Android, se crea una red wifi para que el dispositivo se conecte a este y queden en la misma red Figura 17.

Figura 17: Dispositivo conectado a la red (Izquierda) y red WiFi Android (Derecha).

e) A continuación, inicie el programa VNC y haga clic en otras formas de conexión.

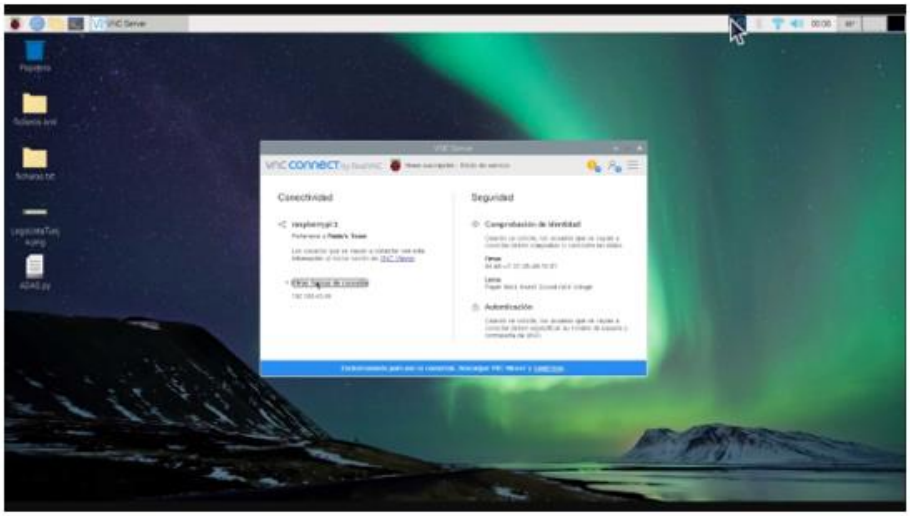

Figura 18: Programa VNC en el dispositivo local.

f) Descargue la aplicación VNC en su dispositivo Android con el que creo la red wifi y agregue un nuevo dispositivo con el botón +.

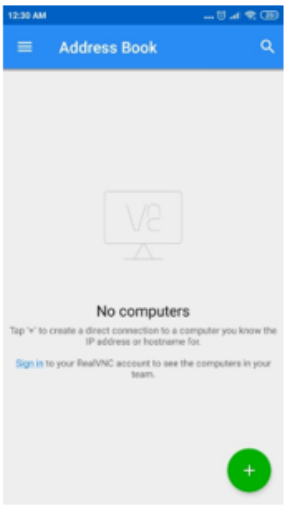

Figura 19: Aplicación VNC para celular.

g) Agrega los datos dados por el programa en el dispositivo local al ingresar en otras formas de conexión y presiona crear y conectar

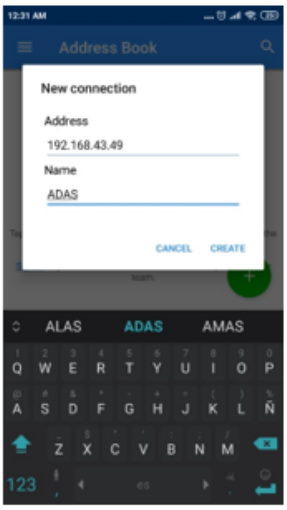

Figura 20: Ingreso de datos del dispositivo remoto.

h) Ahora pedirá un usuario y una contraseña:

- Usuario: pi.
- Contraseña: raspberry.

i) Se marcar la casilla Remember password y Continue.

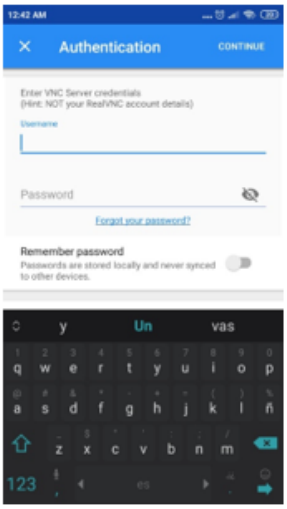

Figura 21: Autentificación para controlar el dispositivo local.

j) A continuación, se apaga el dispositivo local y se procede a realizar el diagrama de conexión en funcionamiento Figura 4 e instalación en el vehículo.

k) A partir de este punto se siguen los pasos del dispositivo remoto desde el punto b), utilizando como pantalla nuestro dispositivo Android.

Artículo V Encuentro Ciencia, Mujer y Tecnología 2019

Propuesta de Investigación

DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO ADAS CON COMUNICACIÓN IOT (XBEE) Y CON TRATAMIENTO DE IMÁGENES PARA DETECCIÓN DE SEUDOACCIDENTES

Rojas Machuca, Paula N[1]. Salas Rodríguez, Cristian A[2]. Galarza Bogotá, Cesar M[3]. Castellanos Guarin, Luis F[4].
Paula.rojas@usantoto.edu.co, Cristian.salas@usantoto.edu.co, Cesar.galarza@usantoto.edu.co, Luis.castellanosg@usantoto.edu.co

[1,2] Universidad Santo Tomás / Estudiante, Comunicaciones (IoT), Facultad de Ingeniería Electrónica, Tunja, Colombia
[3] Universidad Santo Tomás / Docente, Comunicaciones (IoT), Facultad de Ingeniería Electrónica, Tunja, Colombia
[4] Universidad Santo Tomás / Docente, Comunicaciones (IoT), Facultad de Ingeniería de Sistemas, Tunja, Colombia

INTRODUCCIÓN: Planteamiento del problema y justificación: El parque automotor registrado en el RUNT para octubre del 2018 es de 14.342.623, del cual 42% son vehículos de los cuales las 10 referencias más vendidas en Colombia tienen una baja calificación en seguridad para los conductores y pasajeros según el Latin NCAP. El Ministerio de transporte en el Anuario Estadístico Transporte en Cifras - Estadísticas 2017 el 28.1% del parque automotor activo fue adquirido antes del 2013. El 14.7% fueron matriculados antes del 2002. Como lo muestra el informe de componentes descriptivos y explicativos de la accidentalidad vial en Colombia 2019: incidencia del factor humano. Este informe describe que, de las 3.688 muertes ocasionadas con accidentes de tráfico en el 2012, 1.314 ocurrieron por exceso de velocidad, desobediencia a las señales de tránsito y no mantener la distancia de seguridad. Según la ANSV, para 2017 en Colombia fallecieron 6.479 personas en accidentes de tránsito, cifra similar a la del 2016. Con base en lo anterior, se evidencia la necesidad de crear herramientas tecnológicas que ayuden a reducir la accidentalidad por imprudencias de los conductores, adaptables a todo tipo de automóvil y precios accesibles. **Objetivo general:** Diseñar e implementar el hardware para un prototipo de sistema de asistencia al conductor (ADAS), compuesto por un sensor inercial, GPS, cámara y un sistema de comunicación IoT (xbee). **Objetivos específicos:** **(a)** Analizar el estado del arte en un sistemas embebidos y tecnologías IoT. **(b)** Diseñar un prototipo ADAS con un sistema embebido y sensores (inercial, GPS, cámara y módulo IoT para comunicación). **(c)** Realizar pruebas con el prototipo ADAS implementado en el vehículo. **(d)** Implementar el prototipo ADAS en un vehículo.

RESULTADOS Esperados: Como resultado de este proyecto se obtendrá un prototipo de sistema ADAS compuesto por hardware especializado con un módulo IoT. Se analizarán los resultados obtenidos de las pruebas de campo realizadas en conjunto con la secretaria de tránsito de Bucaramanga, con el propósito de identificar posibles errores o fallas al implementar el sistema ADAS. Los datos recolectados por el sistema pueden permitir realizar análisis de los modos de conducción y del estado de la vía, pero un estudio profundo se realizará en una segunda etapa.

REFERENCIAS:

[1] Devapriya, W., Babu, C. N., & Srihari, T. (2015). Advance Driver Assistance System (ADAS) - Speed bump detection. *IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*. India. [2] República de Colombia - Ministerio de Transporte. (26 de 03 de 2019). *Transporte Cifras20170828*. Obtenido de https://www.mintransporte.gov.co/loader.php?lServicio=Tools2&lTipo=descargas&lFuncion=descargar&idFile=16036 [3] www.andemos.org. (2018). *Informe Vehiculos 2018-10* [4] Jung, S., Lee, U., Jung, J., & Shim, D. H. (2016). Real-time Traffic Sign Recognition system with deep convolutional neural network. 13th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI). Xi'an, China [5] Lashkov, I., Smirnov, A., Kashevnik, A., & Parfenov, V. (s.f.). Ontology-Based Approach and Implementation of ADAS System for Mobile Device Use While Driving. En Communications in Computer and Information Science (págs. 117-131). International Conference on Knowledge Engineering and the Semantic Web

Artículo Congreso Internacional de Ingenierías 2019; II Congreso Internacional en Diseño, Fabricación y Nuevos Materiales CIDIFAM; V Congreso Internacional de Instrumentación Control y Telecomunicaciones CIICT

Diseño e implementación de un prototipo ADAS con comunicación IoT (Xbee) y con tratamiento de Imágenes para detección de seudoaccidentes

Design and implementation of an ADAS prototype with IoT communication (Xbee) and image treatment for pseudo-accident detection

Projeto e implementação de um protótipo ADAS com comunicação IoT (Xbee) e tratamento de imagem para detecção de pseudo-acidente

Paula Natalia Rojas Machuca[a], Cristian Andrés Salas Rodríguez, Cesar Mauricio Galarza Bogotá, Luis Fernando Castellanos Guarín

[a] Facultad de ingeniería Electrónica, *Comunicaciones (IoT),* Grupo *de Investigación y Desarrollo de Nuevas Tecnologías (GIDINT),* Universidad Santo Tomas de Tunja, paula.rojas@usantoto.edu.co
[b] Facultad de Ingeniería Electrónica, *Comunicaciones (IoT),* Grupo *de Investigación y Desarrollo de Nuevas Tecnologías (GIDINT),* Universidad Santo Tomas de Tunja, cristian.salas@usantoto.edu.co
[c] Facultad de Ingeniería Electrónica, *Comunicaciones (IoT),* Grupo *de Investigación y Desarrollo de Nuevas Tecnologías (GIDINT),* Universidad Santo Tomas de Tunja, cesar.galarza@usantoto.edu.co
[d] Facultad de Ingeniería Sistemas, *Comunicaciones (IoT),* Grupo *de Investigación y Desarrollo de Nuevas Tecnologías (GIDINT),* Universidad Santo Tomas de Tunja, luis.castellanosg@usantoto.edu.co

Resumen— En este artículo se encuentra la información pertinente al prototipo de sistema avanzado de asistencia al conductor (ADAS) respecto a la metodología con la que se quiere diseñar e implementar este en la ciudad de Bucaramanga. Por medio del diseño de un hardware para un prototipo de un sistema de asistencia al conductor el cual está compuesto por un sensor inercial, un GPS, cámara y un sistema de comunicación IoT (Xbee), la investigación que se desarrolla es de tipo aplicado dado que se quiere diseñar el prototipo, pero también se puede decir que es una investigación de carácter cuantitativo y descriptivo por la recolección de datos que proviene de sensores estandarizados, validados y confiables. Los resultados obtenidos hasta el momento son recorridos mostrados por medio de archivos kml donde se observa una buena toma de datos tanto en el prototipo del automóvil como en un servidor remoto.

***Palabras clave*— ADAS-GPS-sensor inercial--datos.**

***Abstract*— This article contains information relevant to the prototype of the advanced driver assistance system (ADAS) regarding the methodology with which it is intended to design and implement this in the city of Bucaramanga.**

By means of the design of a hardware for a prototype of a driver assistance system which is composed of an inertial sensor, a GPS, camera and an IoT (Xbee) communication system, the research carried out is of a given applied type. that you want to design the prototype, but you can also say that it is a quantitative and descriptive investigation for the collection of data that comes from standardized, validated and reliable sensors. The results obtained so far are routes shown by means of kml files where a good data collection is observed both in the car prototype and in a remote server.

***Keywords*— ADAS-GPS-inertial sensor-data.**

***Resumo*— Este artigo contém informações relevantes para o protótipo do sistema avançado de assistência ao motorista (ADAS) sobre a metodologia com a qual se destina a projetar e implementar isso na cidade de Bucaramanga.**

Por meio do projeto de um hardware para um protótipo de um sistema de assistência ao motorista composto por um sensor inercial, um GPS, uma câmera e um sistema de comunicação IoT (Xbee), a pesquisa realizada é de um determinado tipo aplicado. que você deseja projetar o protótipo, mas também é possível dizer que é uma investigação quantitativa e descritiva para a coleta de dados provenientes de sensores padronizados, validados e confiáveis. Os resultados obtidos até o momento são rotas mostradas por meio de arquivos kml onde uma boa coleta de dados é observada no protótipo do carro e em um servidor remoto.

***Palavras chave*— ADAS-GPS-sensor inercial -dados.**

I. Introducción

El parque automotor registrado en el RUNT para octubre del 2018 es de 14.342.623, del cual 42% son vehículos. Las 10 referencias más vendidas en Colombia tienen una baja calificación en seguridad para los conductores y pasajeros según el Latin NCAP [3]. Como lo muestra el informe de componentes descriptivos y explicativos de la accidentalidad vial en Colombia 2019: incidencia del factor humano. De las 3.688 muertes ocasionadas con accidentes de tráfico en el 2012, 1.314 ocurrieron por exceso de velocidad, desobediencia a las señales de tránsito y no mantener la distancia de seguridad. Según la ANSV, para 2017 en Colombia fallecieron 6.479 personas en accidentes de tránsito, cifra similar a la del 2016 [2]. Con base en lo anterior, se evidencia la necesidad de crear herramientas tecnológicas que ayuden a reducir la accidentalidad por imprudencias de los conductores, adaptables a todo tipo de automóvil y precios accesibles.

En la industria automotriz existe gran variedad de Sistemas Avanzados de Asistencia al Conductor (ADAS) por sus siglas en inglés y su implementación inicio en Europa, donde varios fabricantes de automóviles e institutos de investigación iniciaron la iniciativa Prometheus, alrededor de 1986 (Daimler AG, 2019) [6]. Los sistemas ADAS están enfocados a asistir a los conductores, dependiendo de sus propósitos, estos sistemas pueden tener diferentes arquitecturas y maneras de comunicarse con el conductor (de forma activa o pasiva). Ejemplo, pueden usar múltiples sensores tanto externos como internos), que actúan como fuentes de información, tener distintos conjuntos de indicadores o influenciar a otros sistemas de a bordo cuyo propósito es prevenir y disminuir la cantidad y gravedad de los accidentes de tránsito. Este tipo de sistemas son implementados en vehículos en su mayoría automáticos de alta gama de fabricantes como Nissan, BMW, MERCEDEZ BENZ, AUDI, TESLA entre otras [4].

En este trabajo se muestra el diseño inicial de un prototipo de un sistema de asistencia al conductor (ADAS). El cual se basa en la identificación de posibles seudoaccidentes, con respecto a la distancia existente entre el vehículo A y un vehículo B; que se aproxima al vehículo A por la parte posterior, utilizando un sensor inercial, un GPS, una cámara para una mejora en implementaciones futuras y un sistema de comunicación IoT por medio de un módulo Xbee. Las pruebas en campo se quieren realizar en la ciudad de Bucaramanga debido a que esta ciudad tiene un seguimiento estricto y bien detallado de las diferentes señales de tránsito de toda la ciudad, las cuales están monitoreadas por la secretaria de transporte de esta ciudad por satélite.

II. Materiales y métodos

Materiales

Los materiales utilizados para el desarrollo de este trabajo de investigación son los siguientes.

A. Raspberry Pi 3 B+:

La empresa Raspberry Pi Foundation inicio el desarrollo de un ordenador de placa reducida de bajo coste con el objetivo de estimular la enseñanza y el desarrollo de proyectos electrónicos de bajo costo, pero de gran complejidad. El desarrollo de esta inicio con la primera versión Raspeberry modelo A que contaba con un procesador Broadcom BCM2835, single core a 700 MHz una ram de 256 y una gráfica Broadcom Video core IV, esta utilizaba una fuente de alimentación de 5 voltios y 5 Amperios en común con el resto de versiones. El modelo a usar es el Raspberry pi 3 B+(Figura 1), este cuenta con un procesador ARMv8,quad-core a 1.4GHz ,una memoria ram de 1 Giga y una gráfica video Core IV, como se sabe que el proyecto va a contar con un tratamiento de imágenes es necesario un buen procesador que se ajuste a las necesidades del procesamiento características suplen la necesidad

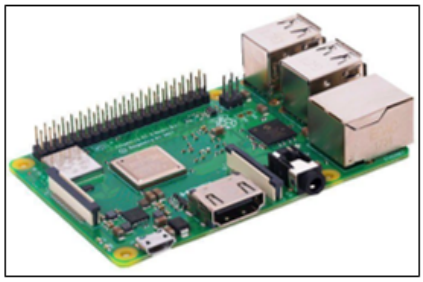

Figure 1 Raspberry Pi 3 B+

B. MPU-9250:

El mpu es un dispositivo de medición inercial con 9 ejes (3 acelerometro,3 de giroscopio y 3 de magnetómetro),este dispositivo nos va a permitir medir la aceleración ,teniendo en cuenta que a pesar que no exista movimiento, el acelerómetro estará censando la aceleración de la gravedad, además que con el acelerómetro podemos obtener más medidas de manera indirecta como la velocidad y la posición datos que nos van a permitir dar le veracidad a otros datos(GPS) ,los datos del giroscopio nos reconocerá a que dirección está apuntando el vehículo y el magnetómetro nos permitirá saber en dónde se ubica el polo norte magnético para mejorar el sentido de orientación del vehículo, otra característica dada la necesidad del proyecto es su bajo precio que ronda alrededor de 7,29 $ dólares ,que diferencia de otras referencias que cuestan lo mismo, pero solo tienen 6 ejes.

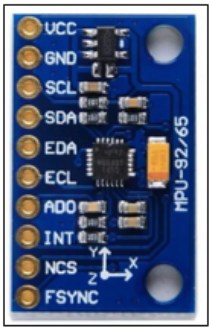

Figure 2 MPU-9250

C. NEO-6M:

Permite saber la geolocalización exacta de donde se encuentra el dispositivo, también nos proporciona una hora y una fecha en sistema de coordenadas WGS-84

Este módulo (figura 3) enviara los datos a las reaspbery|por medio de comunicación serial con un baudrate de 9600 y permite una alimentación de 3,5 a 5 voltios

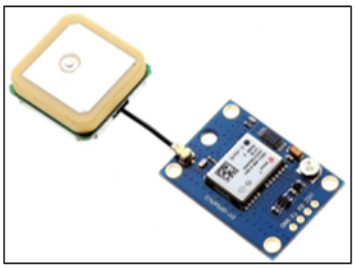

Figure 3 Modulo NEO-6M

D. Xbee PRO S3B:

Los xbee son modulos embebidos que funcionan por comunicación serial a radio frecuencia (o usb a Radio frecuencia), este además cuenta con un software de prueba y configuración; diseñados para tramos de largo alcance, el XSC-XBee es ideal para soluciones donde la penetración de RF y la distancia de transmisión absoluta ,esto nos a permitir realizar la conexión entre el dispositivo del vehículo, y el dispositivo previo antes de subirse a una internet sin la necesidad generar ningún gasto adicional ,pues no es necesario pagar ninguna suscripción a una ISP

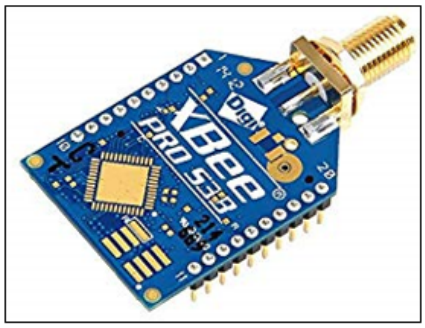

Figure 4 Xbee PRO S3B

E. Cámara para raspberry 1080p HD:

Esta cámara puede captar de imágenes estáticas de 2592 x 1944 píxeles, y también soporta la grabación de vídeo de 1080 p @ 30 fps, 720 p @ 60 fps y 640 x 480 p 60/90, cuenta con un 2pcs infrarrojos LED, esto nos permitirá captar imágenes la noche

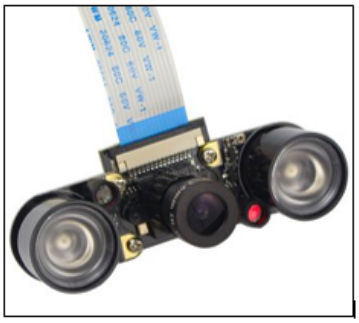

Figure 5 Camara para Raspberry 3

Metodología

La investigación que se desarrolla es de tipo aplicado, puesto que se busca diseñar un prototipo de una plataforma de asistencia al conductor. Aunque se puede considerar también de carácter cuantitativo y descriptivo, con base a la naturaleza de los datos recolectados, que provienen sensores estandarizados, validados, confiables y al análisis de los datos recolectados que permiten determinar la distancia entre vehículos y el modo de conducción.

El desarrollo de la metodología de esta investigación se divide en fases las cuales se muestran en la figura 6 que tienen relación con los objetivos de esta propuesta que corresponde a diseñar e implementar hardware para un prototipo de sistema de asistencia al conductor (ADAS), compuesto por un sensor Inercial, GPS, Cámara y un sistema de comunicación IoT (xbee).

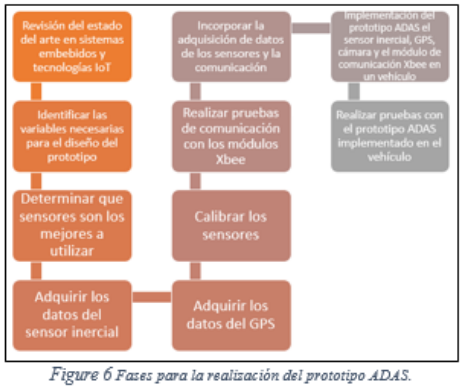

Figure 6 Fases para la realización del prototipo ADAS.

- Revisión del estado del arte en sistemas embebidos y tecnologías IoT: Se hizo un estudio de los diferentes sistemas avanzados de asistencia al conductor que existen actualmente y como estos han ayudado en el mejoramiento de la movilidad, de igual forma se investigó sobre las diferentes tecnologías IoT que están funcionado hasta la actualidad.
- Identificar las variables necesarias para el diseño del prototipo: Esto es necesario para la busque de los sensores que sean los más aptos para la implementación del prototipo.
- Determinar que sensores son los mejores a utilizar: Como las variables a medir ya se definieron ahora se tienen que seleccionar los sensores que mejor nos funcionen para la adquisición de los datos.
- Adquirir los datos del sensor inercial: Se verifica que el protocolo de comunicación I2C esté funcionando y se pueda adquirir los datos que el sensor inercial recolecta.

- Adquirir los datos del GPS: Los datos de del GPS se adquieren por medio de comunicación serial y se comprueba que estos estén llegando correctamente y sean datos verdaderos de la localización.

- Calibrar los sensores: Para el funcionamiento correcto de los sensores se tiene que realizar una calibración y así tener una adquisición de datos buena.

- Realizar pruebas de comunicación con los módulos Xbee: Se realiza para comprobar el buen funcionamiento de la comunicación como el alcance y exposición a las diferentes interferencias que se puedan presentar.

- Incorporar la adquisición de datos de los sensores y la comunicación: Con la integración de los datos con la comunicación ya es posible empezar a realizar pruebas del prototipo.

- Implementación del prototipo ADAS el sensor inercial, GPS, cámara y el módulo de comunicación Xbee en un vehículo: Se elige el mejor lugar de la estructura del vehículo en la cual el prototipo se pueda ubicar sin molestar a los usuarios.

- Realizar pruebas con el prototipo ADAS implementado en el vehículo: Con las pruebas ya se tiene una visión del comportamiento del sistema a los factores ambientales a los cuales va a estar expuesto.

Para la implementación final del prototipo ADAS se tiene como disposición de los elementos a utilizar el modelo diseñado que se muestra en la siguiente figura 7

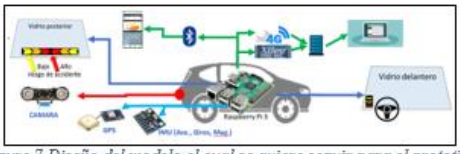

Figure 7 Diseño del modelo el cual se quiere seguir para el prototipo ADAS.

III. Resultados

A. *Resultados Esperados:*

Como resultado de este proyecto se obtendrá un prototipo de sistema ADAS compuesto por hardware especializado con un módulo IoT. Se analizarán los resultados obtenidos de las pruebas de campo realizadas en conjunto con la secretaria de tránsito de Bucaramanga, con el propósito de identificar posibles errores o fallas al implementar el sistema ADAS. Los datos recolectados por el sistema pueden permitir realizar análisis de los modos de conducción y del estado de la vía, pero un estudio profundo se realizará en una segunda etapa.

B. *Resultados Parciales:*

- Los resultados parciales de este proyecto se han obtenido gracias al desarrollo del prototipo de sistema ADAS compuesto por un módulo IoT (Xbee) para la comunicación, un GPS para localización, un sensor inercial para datos estadísticos y una cámara para medir las distancias de los carros en la parte posterior.

- La primera toma de datos se realizó en la ciudad de Tunja con una ruta predeterminada para poder comparar los datos generados por el google maps y los datos generados por el dispositivo GPS, adicional se tomarán los datos del sensor inercial y todos estos se enviaron a una base remota por medio del Xbee PRO S3B

Figure 8 Ruta que se generó con el dispositivo del vehículo

Figure 9 Ruta generada en google maps vehiculo

Latitud	Longitud	Fecha	Hora	Acel x	Acel y	Acel z	Giro x	Giro y	Giro z	Mag x	Mag y	Mag z
5,33012	-73,3447756	12/09/2019	20:08:34	-0,0254	-0,0215	1,0257	0,214	-0,305	0,664	22,761	-1,618	-73,855
5,331098	-73,3447756	12/09/2019	20:08:40	-0,054	-0,104	1,059	1,305	-0,214	0,641	22,582	-1,798	30,064
5,331098	-73,3447757	12/09/2019	20:08:41	-0,053	-0,104	1,056	1,26	-0,75	0,916	22,761	-1,618	30,237
5,331098	-73,3447757	12/09/2019	20:08:42	-0,038	-0,048	1,028	-0,305	-2,802	7,896	23,658	-2,517	29,623
5,331098	-73,3447756	12/09/2019	20:08:43	-0,034	-0,148	1,025	-3,448	-1,855	0,885	24,554	-0,899	28,908
5,331097	-73,3447756	12/09/2019	20:08:44	-0,117	-0,089	0,957	15,054	8,466	-18,646	24,374	0,36	29,027
5,331097	-73,3447756	12/09/2019	20:08:45	-0,08	-0,038	0,994	0,572	0,778	1,282	35,307	0,257	41,986
5,331097	-73,3447757	12/09/2019	20:08:47	-0,07	0,003	1,005	6,989	0,458	2,693	36,562	-2,517	45,268
5,331106	-73,3447752	12/09/2019	20:08:51	-0,104	-0,039	1,021	-6,757	-4,898	-4,639	37,637	2,517	45,269
5,331107	-73,3447788	12/09/2019	20:08:54	-0,089	-0,147	1,065	-9,192	-3,693	8,307	33,236	-2,877	41,467
5,331112	-73,344771	12/09/2019	20:08:57	-0,1	-0,118	0,882	10,483	-2,007	-6,256	33,015	-1,978	40,668
5,331152	-73,3447753	12/09/2019	20:09:01	-0,044	-0,155	1,014	-5,722	1,93	-2,853	35,486	-1,079	40,433
5,331159	-73,3447747	12/09/2019	20:09:02	-0,085	-0,255	1,104	5,836	10,454	-10,757	36,742	-0,899	42,299
5,331165	-73,3447742	12/09/2019	20:09:03	-0,074	-0,216	0,996	13,931	4,212	-21,362	33,875	-2,338	38,589
5,331169	-73,344739	12/09/2019	20:09:04	-0,094	-0,242	1,053	8,308	5,646	1,589	36,289	-0,539	39,563
5,331175	-73,3447738	12/09/2019	20:09:05	-0,009	-0,13	1,053	7,262	3,899	-4,227	36,247	-4,853	38,357
5,331175	-73,3447736	12/09/2019	20:09:06	-0,217	-0,253	1,008	18,745	11,566	7,385	27,958	-6,214	39,294

Figure 10 Tabla de datos resumen

Datos recibidos en la estacion remota

Figure 11 Datos de ruta obtenidos en la estación remota

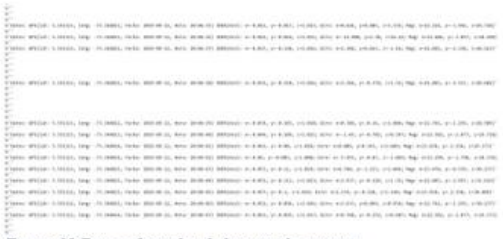

Figure 12 Datos obtenidos de la estación remota

IV. Discusión

Los datos recolectados han ayudado a determinar qué tipo de limitaciones puede tener nuestro dispositivo y que tipo de soluciones se tienen que generar para mejorar la fidelidad de los datos

Con relación al sistema Gps este arroga coordenadas que concuerdan con las esperadas para este proyecto, ya que estas cuentan con una alta precisión y gran similitud con los datos generados en Google Maps , por otra parte el sensor inercial necesita un manejo de datos más riguroso para poder corroborar los mismos de manera correcta, dicho de otra manera los datos del sensor inercial por el momento no satisfacen los requerimientos para esta investigación y es necesario incrementar los esfuerzos en un mejor tratamiento de los mismos ,igualmente los datos que han sido enviados a la central remota no han llegado en su totalidad dando, a entender que el alcance del módulo xbee tiene muy poca cobertura ,sin embargo son datos que no estarían de acuerdo con las características del mismo, y es importante tomar más medidas para comprobar que no sea un mal funcionamiento o una mal configuración del módulo

V. Agradecimientos

Los estudiantes P. Rojas y A. Salas (Colombia) agradecen a los tutores C. Galarza y L. Castellanos (Colombia) las ayudas brindadas para las correcciones correspondientes que se realizaron en la redacción de este articulo como en las tutorías brindadas para la elaboración del diseño del prototipo ADAS.

VI. Conclusiones

- La cantidad de datos obtenidos de la estación remota es mucho menor a la del dispositivo local, razón por la cual se recomienda garantizar la recepción de los mismo para así cumplir con los objetivos de la investigación.
- El rango de cobertura del módulo xbee según los datos obtenidos no es el esperado, pero, se deben tomar más medidas con diferentes configuraciones para rectificar el correcto funcionamiento
- Los datos obtenidos por el sensor inercial serán a analizados posteriormente para rectificar la correcta elección del sensor inercial.
- Los datos generados por el gps del dispositivo que se encuentra en el vehículo son los esperados de acuerdo a los datos de google maps
- Para las siguientes etapas del proyecto es necesario la verificación de cada uno de los procesos que hacen parte del mismo como un todo.

Referencias

[1] Devapriya, W., Babu, C. N., & Srihari, T. (2015). Advance Driver Assistance System (ADAS) - Speed bump detection. IEEE International Conference on Computational Intelligence and Computing Research (ICCIC). India.

[2] República de Colombia - Ministerio de Transporte. (26 de 03 de 2019). Transporte Cifras20170828. Obtenido de https://www.mintransporte.gov.co/loader.php?lServicio=Tools2&lTipo=descargas&lFuncion=descargar&idFile=16036

[3] www.andemos.org. (2018). Informe Vehiculos 2018-10

[4] Jung, S., Lee, U., Jung, J., & Shim, D. H. (2016). Real-time Traffic Sign Recognition system with deep convolutional neural network. 13th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI). Xi'an, China

[5] Lashkov, I., Smirnov, A., Kashevnik, A., & Parfenov, V. (s.f.). Ontology-Based Approach and Implementation of ADAS System for Mobile Device Use While Driving. En Communications in Computer and Information Science (págs. 117-131). International Conference on Knowledge Engineering and the Semantic Web.

[6] Daimler AG. (26 de 03 de 2019). he PROMETHEUS project launched in 1986: Pioneering autonomous driving. Obtenido de https://media.daimler.com/marsMediaSite/en/instance/ko/The-PROMETHEUS-project-launched-in-1986-Pioneering-autonomous-driving.xhtml?oid=13744534.

[7] S. Sarala, D. Sharath Yadav and A. Ansari, "Emotionally Adaptive Driver Voice Alert System for Advanced Driver Assistance System (ADAS) Applications," 2018 International Conference on Smart Systems and Inventive Technology (ICSSIT), Tirunelveli, India, 2018, pp. 509-512.

[8] A. Simić, O. Kocić, M. Z. Bjelica and M. Milošević, "Driver monitoring algorithm for advanced driver assistance systems," 2016 24th Telecommunications Forum (TELFOR), Belgrade, 2016, pp. 1-4

[9] S. M. Iranmanesh, H. Nourkhiz Mahjoub, H. Kazemi and Y. P. Fallah, "An Adaptive Forward Collision Warning Framework Design Based on Driver Distraction," in IEEE Transactions on Intelligent Transportation Systems, vol. 19, no. 12, pp. 3925-3934, Dec. 2018.

AUTORES

Paula Natalia Rojas Machuca, nació en Duitama el 3 de enero de 1997 en el Hospital Regional, sus padres son Henry Rojas Velandia y Luz Jackeline Machuca Perico, es la menor de tres hijos. Comenzó sus estudios de primaria en el año 2003 culminándolos en el año 2007 en el Colegio Nacionalizado la Presentación, se graduó de la educación secundaria en el año 2013 de esta misma institución. En el año 2015 inicio sus estudios de educación superior en la Universidad Santo Tomas seccional Tunja en la carrera de Ingeniería Electrónica. Actualmente cursa decimo semestre de su carrera.

Cristian Andrés Salas Rodríguez, nació en la cuidad de Tunja el 24 de agosto de 1991 en hospital San Rafael, se vinculó al colegio de Boyacá para dar inicio a su educación primaria, luego finalizo sus estudios en el mismo con especialización en Matemáticas; inicio sus estudios universitarios en Universidad Santo Tomas en la carrera de Ingeniería Electrónica, actualmente está cursando el décimo semestre

Cesar M. Galarza B. Docente de la Facultad de ingeniería Electrónica de la Universidad Santo Tomas seccional Tunja. Ingeniero electrónico graduado de la Escuela Colombiana de Ingeniería Julio Garavito. (ECI) (2007). Maestría en ingeniería electrónica e informática; de la Universidad de los Andes (UNIANDES) (2010). Doctorado en ingeniería electrónica por la Universidad Politécnica de Cataluña (UPC) (2018). Desarrollador Asociado Certificado de LabVIEW (2015-2017). Tiene experiencia en investigación en diferentes proyectos interdisciplinarios. Entre los proyectos en los que participó se encuentran: AUV GUANAY II (UPC - SARTI), Sistema de detección de minería (SDM) fase 1, fase 2, fase 3 y Prototipos SDM (INDUMIL – UniAndes) y Repulsión de un Plaga específica (Lepidoptera Copitarsia decolora) (ECI -BanRep).

Luis Fernando Castellanos Guarín Docente de la Facultad de ingeniería de sistema de la Universidad Santo Tomas seccional Tunja, Ingeniero de Sistemas con Énfasis en Telecomunicaciones, Especialista en tecnologías avanzadas para el desarrollo de software, Magister en Gestión, Aplicación y Desarrollo de Software, con experiencia en implementación y gestión de proyectos software (Diseño de Arquitectura, base de datos y SQA) para empresas como TeleBucaramanga, UNAB y INVIAS. Con experticia académica en desarrollo de aplicaciones web, gestión y auditoria de Base de datos, desarrollo e implementación de algoritmos de inteligencia artificial enfocado en Deep Learning para uso en PLN y visión por computadora.

Artículo Andescon 2020 Technology and Innovation for Andean Industry

Using Raspberry For Inertial And Gps Acquisition With Image Storage For An ADAS Prototype Using Xbee Communication

Paula Rojas
Faculty of Electronic Engineering
Universidad Santo Tomás
Tunja, Colombia
paula.tojas@usantoto.edu.co

Cristian Salas
Faculty of Electronic Engineering
Universidad Santo Tomás
Tunja, Colombia
cristian.salas@usantoto.edu.co

Cesar Galarza
Faculty of Electronic Engineering
Universidad Santo Tomás
Tunja, Colombia
cesar.galarza@usantoto.edu.co

Luis Castellanos
Faculty of Systems Engineering
Universidad Santo Tomás
Tunja, Colombia
luis.castellanosg@usantoto.edu.co

***Abstract*—This article contains relevant information for the prototype of the advanced driver assistance system (ADAS) in terms of the methodology with which it is intended to design and implement it in the city of Tunja. Through the design of a hardware for a prototype of a driver assistance system composed of an inertial sensor, a GPS, a camera and an IoT communication system (XBee), the research carried out is of an applied type determined because it wants to design the prototype, but it can also be said that it is a quantitative and descriptive investigation for data collection that comes from standardized, validated and reliable sensors. The results obtained so far are test routes where it is evidenced that the prototype can predict the route when the GPS signal is lost with the respective graphs of speed and acceleration of the vehicle and the transmission of these data to the base station.**

***Keywords*—ADAS, GPS, inertial sensor, data, Xbee, kml**

I. Introduction

The automotive fleet registered in the RUNT (Registro Único Nacional de Transito) for August 2019 is 15,033,565, of which 41% are vehicles [1]. The 10 best-selling references in Colombia [2] have a low safety rating for drivers and passengers according to the Latin NCAP (Programa de Evaluación de Vehículos Nuevos para América Latina y el Caribe) [3]. As shown in the 2018 road accidents report from the National Road Safety Observatory [4]: Probable cause hypothesis. This report describes that, of the 181,374 traffic accidents in 2018, 35,422 were due to disobedience to traffic signals, 28,515 were due to not maintaining a safe distance and 3,016 occurred due to speeding. According to the ANSV (Agencia Nacional de Seguridad Vial) [5], for 2018 in Colombia 6,850 people died in traffic accidents, a figure that increased to that of 2017, which was 6,719. Based on the above, there is evidence of the need to create technological tools that help reduce the accident rate due to recklessness of drivers, adaptable to all types of cars and at affordable prices.

There is a wide variety of Advanced Driver Assistance Systems (ADAS) in the automotive industry and their implementation began in Europe, where several automobile manufacturers and research institutes started the Prometheus initiative, around 1986 [6]. ADAS systems are focused on assisting drivers, depending on their purposes, these systems can have different architectures and ways of communicating with the driver (actively or passively). These types of systems are implemented in vehicles, mostly automatic high-end manufacturers such as NISSAN, BMW, MERCEDEZ BENZ, AUDI, TESLA among others [7].

This work shows the initial design of a prototype of a driver assistance system (ADAS). Which is based on the identification of possible pseudo-accidents, with respect to the distance between vehicle A and vehicle B; approaching vehicle A from the rear, using an inertial sensor, a GPS, a camera for improvement in future implementations, and an IoT communication system using an Xbee module.

II. Materials

The materials used for the development of this research work are as follows.

A. Raspberry Pi 4 Model B

The model use is the Raspberry Pi 4 B (Fig. 1), it has a Cortex-A72 processor, quad-core at 1.5GHz, a 4Giga RAM memory and a VideoCore IV graphics [8], as it is known that the project will have treatment images later you need a good processor that meets these processing needs.

Fig. 1. Raspberry Pi 4 B

B. MPU-9250

The MPU (Fig. 2) is an inertial measurement device with 9 axes, gyroscope (X-, Y-, Z) with scales in ranges of ±250°/Sec, ±500°/Sec, ±1000°/Sec and ±2000°/Sec, 3-axis accelerometer with scales of ±2g, ±4g, ±8g, and ±16g, and magnetometer with a range of ±4800μT [9], this device will allow us to measure acceleration, taking into account that even if there is no movement, the accelerometer will record the acceleration of gravity, in addition to the accelerometer we will be able to obtain more indirect measurements, such as speed and position, data that will allow us to give truth to other data (GPS), the data from the gyroscope will recognize us in which direction it is pointing the vehicle and the

978-1-7281-9365-6/20/$31.00 ©2020 IEEE

magnetometer allow us to know where the magnetic north pole is to improve the vehicle's sense of orientation.

Fig. 2. MPU-9250

C. NEO-6M and GPS Antenna

It allows to know the exact geolocation of where the device is, it also provides us with a time and date in the WGS-84 coordinate system [10].

This module (Fig. 3) will send the data to the raspberry through serial communication with a speed of 9600 bauds and allows a supply of 3.5 to 5 volts, has a capture sensitivity of -148dBm, tracking sensitivity of -161dBm and an accuracy of 1µSec [10].

Fig. 3. GPS NEO-6M

To increase the power of the signal NEO-6M, a GPS antenna was used (Fig. 4) with a Frequency range of 1575.42 MHz with a gain of 27dB [11].

Fig. 4. GPS Antenna

D. XBee PRO S3B

The XBee (Fig. 5) are embedded modules that work by serial communication at radio frequency (or usb to radio frequency), designed for long-range sections, the XSC-XBee is ideal for solutions where RF penetration and absolute transmission distance, this to allow the connection between the device of the vehicle and the previous device before uploading to the Internet without generating any additional expense, it works with a frequency of 900MHz with a data rate of 20kbps, sensitivity of -110dBm and an internal range or 600m urban areas [12], connecting a long RP-SMA Duck type antenna with a frequency range of 24GHz and gain of 5dBi [13] you can have an outdoor range of up to 9.6km.

Fig. 5. XBee PRO S3B

E. Infrared Camera for Raspberry Pi 5MP 1080p with Support for Night Vision

This camera (Fig. 6) can capture 2592 x 1944 pixel still images, and it also supports 1080p @ 30fps, 720p @ 60fps and 640x480p 60/90 video recording, it has a 2pcs infrared LED [14], this we will allow you to capture images at night.

Fig. 6. Infrared Camera for Raspberry Pi

III. Methodology

The research carried out is applied rate, since it seeks to design a prototype of a platform to assist the driver. Although it can also be considered quantitative and descriptive, based on the nature of the data collected, which come from standardized, validated, reliable sensors and the analysis of the collected data that allow determining the distance between vehicles and driving mode.

The development of the methodology of this research is divided into phases which are shown in Fig. 7 that are related to the objectives of this proposal, which corresponds to design and to implement hardware for a prototype driver assistance system (ADAS), composed by an Inertial sensor, GPS, Camera and an IoT communication system (Xbee).

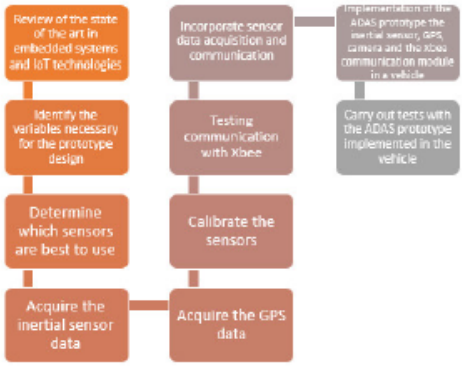

Fig. 7. Phases for the realization of the ADAS prototype.

For the final implementation of the ADAS prototype, the layout of the elements to be used is the designed model shown in the following Fig. 8.

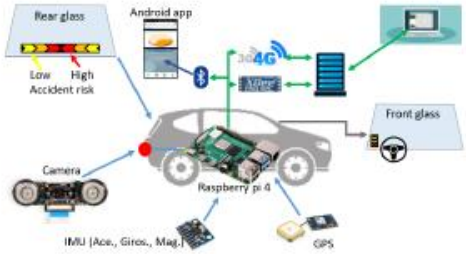

Fig. 8. Design of the model which you want to follow for the ADAS prototype.

The layout used for testing of the prototype can be viewed in Fig.9 where the position of the sensors and the camera are evident, this prototype was implemented on the back of the vehicle.

Fig. 9. Design used in the prototype tests.

IV. Results

A. Software operation

In the realization of the device the Python programming language was used making the use of threads to perform multiple processes at the same time and to be able to obtain the images in time (Fig. 10-13).

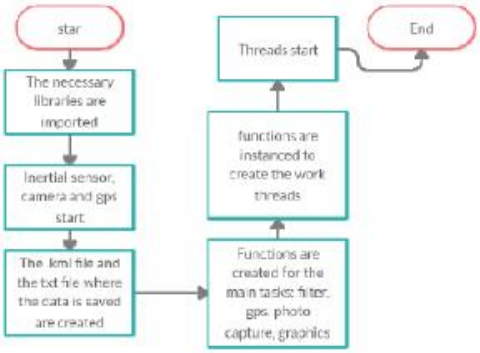

Fig. 10. General operation.

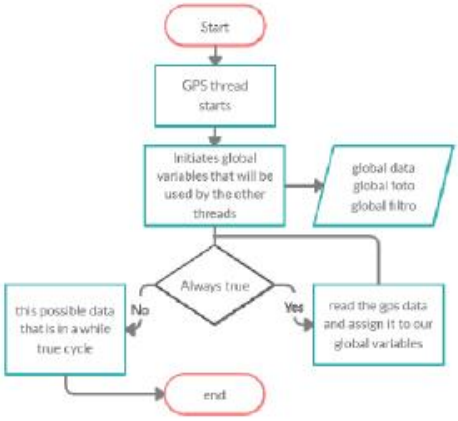

Fig. 11. GPS thread.

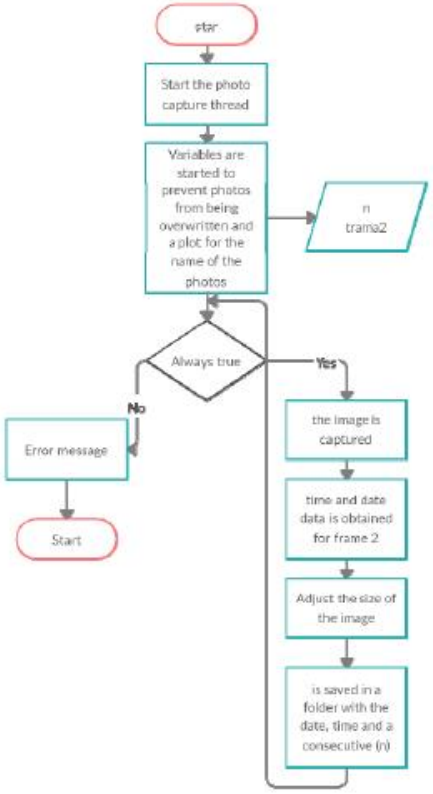

Fig. 12. Photo capture.

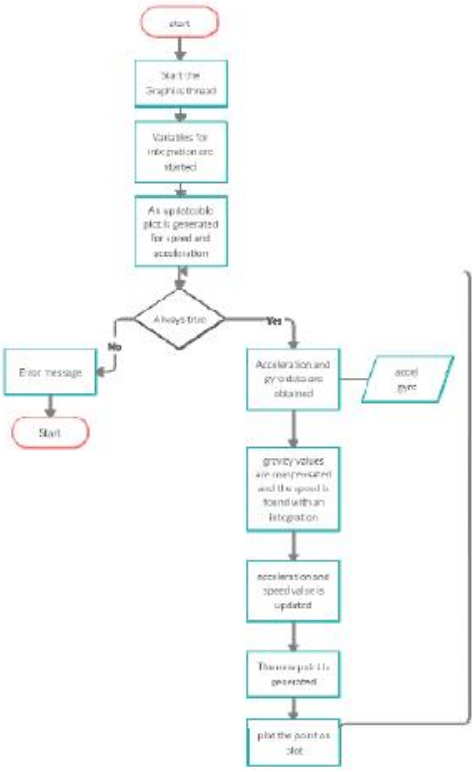

Fig. 13. Graphics of speed and acceleration.

B. *Partial results*

The partial results of this project have been obtained thanks to the development of the prototype ADAS system consisting of an IoT (Xbee) module for communication, a GPS for location, an inertial sensor for statistical data and a camera to measure the distances of the cars in the back.

The first data collection was carried out in the city of Tunja with a predetermined route to be able to compare the data generated by the google maps and the data generated by the GPS device, additionally the data from the inertial sensor will be taken and all of these were sent to a remote base via the Xbee PRO S3B.

The data collected at the remote station show a great difference with respect to the data sent from the vehicle, since when comparing the data, a loss of 35% of the data is perceived.

Data collected at the remote station shows less loss at the sites closest to the remote station.

Fig. 14. Route that was generated with the vehicle device.

Fig. 15. Route generated in google maps.

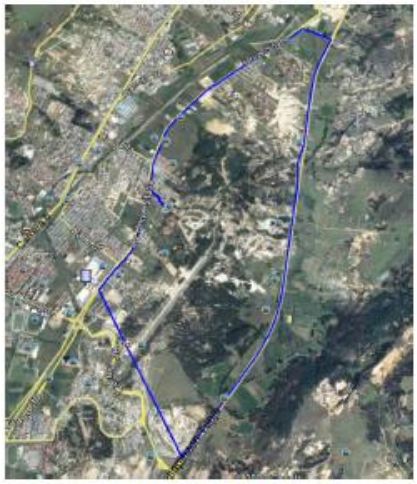

Fig. 16. Route data obtained at the remote station.

Fig. 17. Test with vehicle close to remote station.

Fig. 18. Test with vehicle far from remote station.

V. Conclusions

The prototype shows that it is possible to implement an affordable and easily accessible driver assistance system. In relation to the GPS module, it gives coordinates that agree with those expected for this project, since these have high precision and great similarity with the data generated in Google Maps, on the other hand, the inertial sensor needs a more rigorous data handling to be able to corroborate them correctly, so after performing the correct calibration and filtering the data of the inertial sensor, the prediction of the route can be achieved if the GPS loses coverage, in other words, the data of the inertial sensor satisfies the requirements For this research, the IoT communication module used in the project, in this case the XBee module, it can be concluded that for good data reception the transmitting device must be in line of sight to the base station, in some tests carried out in different places From the location of the base station, it was possible to notice that the coverage was minimal for the different buildings, also the transmission of The data was affected by the speed at which the vehicle travels since at a higher speed the reception of data can be lost and when the speed is reduced, adequate data is received again. On the other hand, if the Raspberry is not connected to a power supply of more than 3A and 5V, it presents problems with the reading of some sensors.

Acknowledgment

We thank our project managers Eng. Cesar Mauricio Galarza Bogotá and Eng. Luis Fernando Castellanos for their collaboration in the development of this prototype.

References

[1] Parque automotor registrado en RUNT, octubre de 2019. [Online] Available: https://www.runt.com.co/runt-en-cifras/parque-automotor

[2] ANDEMOS, "informe vehículos 2019-08", accessed august 2019. [Online] Available: www.andemos.org.

[3] LATINNCAP, "resultados", accessed august 2019. [Online] Available: www.latinncap.com/es/resultados

[4] Agencia Nacional de Seguridad Vial – Observatorio Nacional de Seguridad Vial. Siniestros Siales 2018. [Online] Available: https://ansv.gov.co/observatorio/index4714.html?op=contenidos&sec=59&page=54.

[5] Agencia Nacional de Seguridad Vial – Observatorio Nacional de Seguridad Vial. Cifras definitivas de enero – diciembre 2018. [Online] Available: https://ansv.gov.co/observatorio/index4b78.html?op=contenidos&sec=59

[6] Daimler AG. (26 de 03 de 2019). he PROMETHEUS project launched in 1986: Pioneering autonomous driving. Obtained: https://media.daimler.com/marsMediaSite/en/instance/ko/The-PROMETHEUS-project-launched-in-1986-Pioneering-autonomous-driving.xhtml?oid=13744534

[7] Jung, S., Lee, U., Jung, J., & Shim, D. H. (2016). Real-time Traffic Sign Recognition system with deep convolutional neural network. 13th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI). Xi'an, China

[8] Xataka, "Raspberry Pi 4 Model B, análisis", accessed august 2020. [Online] Available: https://www.xataka.com/ordenadores/raspberry-pi-4-analisis-caracteristicas-precio-especificaciones

[9] NAYLAMP MECHATRONICS, "Módulo MPU9250: Acelerómetro, Giroscopio, Magnetómetro I2C", accessed august 2020. [Online] Available: https://naylampmechatronics.com/sensores-posicion-inerciales-gps/214-modulo-mpu9250-acelerometro-giroscopio-magnetometro-i2c.html

[10] Electronilab, "Módulo GPS Ublox Neo-6M v2 con memoria EEPROM", accessed august 2020. [Online] Available: https://electronilab.co/tienda/modulo-gps-ublox-neo-6m-v2-con-memoria-eeprom/

[11] FERRETRÓNICA, "Antena GPS Activa Rango de Frecuencia 1575.42 MHz 27 dB", accessed august 2020. [Online] Available: https://ferretronica.com/products/antena-gps-activa-rango-de-frecuencia-1575-42-mhz-27-db?_pos=4&_sid=d9eb7aac6&_ss=r

[12] DIGI, " XBee PRO S3B XSC RPSMA", accessed august 2020. [Online] Available: https://xbee.cl/xbee-pro-s3b-xsc-rpsma/

[13] MCI electronics, "2.4GHz Antena Larga RP-SMA Tipo Duck (WiFi y Xbee)", accessed august 2020. [Online] Available: https://www.mcielectronics.cl/shop/product/2-4ghz-antena-larga-rp-sma-tipo-duck-wifi-y-xbee-9642

[14] HUBOT, "Cámara de Infrarrojos para Visión Nocturna Raspberry 5MP 1080p (SKU 516G1)", accessed august 2020. [Online] Available: https://hubot.cl/product/camara-vision-nocturna-raspberry-pi-3-5mp-1080p-sku-516/

Cotización

COTIZACIÓN

DYNAMO Electronics SAS.			
Producto	**Características**	**Precio**	**Link**
Raspberry pi 3 B+	• CPU + GPU: Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz • RAM: 1GB LPDDR2 SDRAM • Wi-Fi + Bluetooth: 2.4GHz y 5GHz IEEE 802.11.b/g/n/ac, Bluetooth • 4.2, BLE • Ethernet: Gigabit Ethernet sobre USB 2.0 (300 Mbps) • GPIO de 40 pines • HDMI • 4 puertos USB 2.0 • Puerto CSI y DSI para conectar una cámara y una pantalla táctil • Salida de audio estéreo y vídeo compuesto • Micro-SD	$175.000	https://www.dynamoelectronics.com/raspberry-accesorios-y-kits/1177-raspberry-pi-3-b-mas.html
Memoria microSD 32Gb Clase 10	• Capacidad 32Gb • Clase 10 • Velocidad 45MB/s Lectura • Tipo: SDHC • Tamaño: uSD, incluye adaptador SD	$66.000	https://www.dynamoelectronics.com/almacenamiento-de-datos-memorias-discos-duros-etc/1027-memoria-sd-16gb-sistema-operativo-raspberry.html
Cámara infrarroja LinkSprite a color serial	• VGA/QVGA/160*120 resolución • Suporta captura JPEG desde puerto serial • Baud rate 38400 • Voltaje de alimentación 3.3V or 5V • Corriente: 80-100mA • Pin cerca de C03 es salida AV (salida análoga)	$199.000	https://www.dynamoelectronics.com/colores-y-camaras/130-camara-infrarroja-cmos-640x480-bn.html
Mini IMU 9 DOF V5 (Acelerómetro, Magnetómetro y giróscopo)	• Numero de ejes: 9 • Rotación en eje X, Y y Z, aceleración en ejes X, Y y Z y magnetómetro X, Y y Z • Voltaje de alimentación: 2,5V-5,5V • Tamaño: 20 mm × 13 mm × 3 mm • Rangos de medidas: ○ Gyro: ±125, ±245, ±500, ±1000, or ±2000°/s	$52.950	https://www.dynamoelectronics.com/posicion-acel-imus-gps-encoder/322-imu-9-dof-v2-acelerometro-magnetometro-y-giroscopo.html

	○ Acelerómetro: ±2, ±4, ±8, or ±16 g ○ Magnetómetro: ±4, ±8, ±12, or ±16 gauss		
Adaptador para raspberry 5V 2A (conector MicroB)	• Voltaje entrada 100-240Vac • Corriente: 2A (a los 2A el voltaje es de 4,5V) • Certificaciones FCC/CE	$21.750	https://www.dynamoelectronics.com/baterias-y-adaptadores/25-adaptador-para-raspberry-5v-2a-conector-microb.html
Disipador para raspberry	• Cantidad: 3 • Material: aluminio	$3.000	https://www.dynamoelectronics.com/raspberry-accesorios-y-kits/759-disipador-para-raspberry.html
Xbee antena cable 100m XB24-AWI-001	• Voltaje 3.3V @ 50mA • Data Rate 250kbps • Potencia salida 1mW (+0dBm) • Rango 300ft (100m) • Antena Integrada tipo cable • Certificación FCC • Puertos de entrada 6 10-bit ADC • Puertos de propósito general 8 digitales • Encriptación 128-bit • Posibilidad de configuración con AT or API command set	$101.474	https://www.dynamoelectronics.com/xbee/179-xbee-antena-cable-100m.html?search_query=xbee&results=21
GPS NEO-6M en board	• Datos de GPS por Serial UART TTL a 9600bps • Compatible con varios controladores de vuelo para drones que usan gps por ejemplo APM2.0 y APM2.5 • Voltaje: 3.3 a 5VDC • Dimensiones: 36×26mm.	$38.000	https://www.dynamoelectronics.com/linea-economica/1368-gps-neo-6m-en-board.html

I + D Electrónica			
Producto	**Características**	**Precio**	**Link**
Tarjeta Raspberry Pi 3 B+ con Memoria 16GB	• Tarjeta Raspberry Pi 3 B+ • Memoria SD 16GB con OS Pre-cargado	$207.000	https://www.didacticaselectronicas.com/index.php/sistemas-de-desarrollo/raspberry-detail
Cámara de Visión Nocturna Raspberry Pi	• 5 millones de pixeles (5 Megapíxeles) • Módulo fotosensible: OV5647 • Apertura (f): 1.8 • Distancia focal: 3.6mm (ajustable) • Tamaño del CCD: 1/4 pulgada • Sensor best PIX: 1080p • Diagonal: 75.5 grados • Alimentación externa: 3.3V • Puede ser usada en una posición fija • 4 agujeros de fijación • Dimensiones: 25mmx24mm • Con cable conector 15PIN FFC	$147.000	https://www.didacticaselectronicas.com/index.php/sistemas-de-desarrollo/raspberry/accesorios-raspberry/camara-de-vision-nocturna-raspberry-pi-detail
Módulo IMU GY-89	• Sensores: LSM303, L3GD20 y BMP180. • 3 ejes de Acelerómetro. Sensibilidad: ±2,4,6,8,16g. • 3 ejes de Giroscopio. Sensibilidad: ± 245,500, 2000°/s. • 3 ejes de Magnetómetro. Sensibilidad: ±2,4,8,12 Gauss. • Barómetro: 300 a 1100 hPa. • Lectura del barómetro sólo por I2C. • Lectura de 16-bits por eje • Dimensiones: 22x17mm.	$44.000	https://www.didacticaselectronicas.com/index.php/sensores/barometro,-presion,-acelerometro,-giroscopio,-magnetrometro-detail
Adaptador Raspberry Pi 3 - 5V,2.5A	• Voltaje de entrada: 100-240V, 50/60 Hz • Voltaje de salida: 5V, 2500mA	$14.000	https://www.didacticaselectronicas.com/index.php/fuentes-adaptadores/adaptadores/adaptador-raspberry-pi-3-raspberry-pi3-fuente-voltaje-poder-5v-detail
Kit de disipadores para Raspberry pi de aluminio	• Ideal para refrigerar su Raspberry Pi si se encuentra en un lugar sin corriente de aire.	$1.400	https://www.didacticaselectronicas.com/index.php/ele

	• Mantener su Raspberry fresca le ayudará al rendimiento y reducir el riesgo de un fallo de hardware. • Los disipadores de calor están diseñados para encajar en todos los modelos Raspberry Pi (2/3).		mentos-electromecanicos/disipadores/disipadores-para-raspberry-refrigeraci%C3%B3n-detail
Módulo XBee 2.4Ghz Serie1	• Antena: cable • Sensibilidad: -92dBm • Potencia de transmisión: 1mW (+0 dBm) • Voltaje de operación: 2,8V - 3,4V	$104.400	https://www.didacticaselectronicas.com/index.php/comunicaciones/zigbee/modulos/modulo-xbee-24ghz-serie1-zigbee-inal%C3%A1mbrico-wireless-digi-detail
Modulo GPS UART NEO-6M (B)	• Chip NEO-6M u-blox • Interfaz IPX, para conectar diferentes antenas activas • Nivel TTL compatible con sistemas 3.3V/5V • Tasa de comunicación: 9600 baud, configurable via u-center • Pines verticales	$66.100	https://www.didacticaselectronicas.com/index.php/comunicaciones/gps/modulo-gps-uart-neo-6m-b-localizaci%C3%B3n,-gps,neo-6m-waveshare-detail

SIGMA ELECTRÓNICA.			
Producto	**Características**	**Precio**	**Link**
Raspberry pi 3 B+	• Procesador Broadcom BCM2837B0, Cortex-A53 • Memoria 1GB LPDDR2 SDRAM • Conectividad dual de 2.4GHz WLAN IEEE 802.11.b/g/n/ac y Bluetooth 4.2 • Gigabit Ethernet sobre USB2.0 de hasta 300Mbps • Puertos USB 2.0×4 • Header de 40 pines GPIO • Puerto de video HDMI y salida estéreo de 4 polos audio y video compuesto. • Puerto dedicado MIPI DSI para Display y MIPI CSI para cámara. • Decodificador Multimedia H.264, MPEG-4 decode (1080p30); H.264 encode • (1080p30); OpenGL ES 1.1, 2.0 graphics. • Conector MicroSD para cargar sistema operativo y almacenamiento masivo. • Requiere Fuente de energía de 5V/2.5A via conector microUSB. • Temperatura de operación 0-50°C.	$154.700	https://www.sigmaelectronica.net/producto/rpi3-b/
MICROSD 16GB	• Capacidad: 16GB • Rendimiento: 80MB/seg (lectura) y 10MB/seg (escritura), UHS-I Velocidad Clase 1 (U1) • Dimensiones (microSD): 11mm x 15mm x 1mm • Dimensiones (con adaptador SD): 24mm x 32mm x 2.1mm • Formato: FAT32 (SDHC 16GB) • Temperaturas de funcionamiento: -25 a 85 °C • Temperaturas de almacenamiento: -40 a 85 °C • Voltaje: 3.3V	$19.040	https://www.sigmaelectronica.net/producto/microsd-16gb/
SEN0184	• Cámara: 5 Megapíxeles Módulo fotosensible: OV5647 Parámetros de la cámara: Tamaño del CCD: 1/4 pulgadas Apertura (f): 1.8 Distancia focal: 3,6 mm (ajustable)	$180.880	https://www.sigmaelectronica.net/producto/sen0184/

	Diagonal: 75,7 grados Sensor mejor PIX: 1080p 4 orificios para tornillos pueden ser utilizados en una posición fija 3.3V fuentes de alimentación externas Soporta acceso a la luz infrarroja o luz de relleno Dimensiones: 25 mm x 24 mm		
POLOLU-2468	• Dimensiones: 0,8 «x 0,5» x 0,1 «(20 mm x 13 mm x 3 mm) • Peso sin pines del cabezal: 0,7 g (0,02 oz) de • Tensión de funcionamiento: 2,5 V a 5,5 V • Corriente de alimentación: 6 mA • Formato de salida (I ² C): • Gyro: una lectura de 16 bits por eje • Acelerómetro: una lectura de 16 bits por eje • Magnetómetro: una lectura de 16 bits por eje • Rango de sensibilidad: • Gyro: ± 245, 500 ± o ± 2,000 ° / s • Acelerómetro: ± 2, ± 4, ± 6, ± 8, o ± 16 g • Magnetómetro: ± 2, ± 4, ± 8, o ± 12 gauss	$74.950	https://www.sigmaelectronica.net/producto/pololu-2468/
FUENTE 5V2.5A	• Adaptador AC/DC con terminal de conexión tipo Micro-USB. • Voltaje entrado: 100~240VAC. • Voltaje Salida: 5VDC. • Corriente de salida: 2,5 Amperios. • Longitud del cable: 110cm.	$19.040	https://www.sigmaelectronica.net/producto/fuente-5v2-5a/
DISIPADOR RPI	• 1 x 14mm(L) x 14mm(W) x 5mm(H) Disipador for Soc • 1 x 9mm(L) x 9mm(W) x 4mm(H) Dispador para el Regulador de voltaje • 1 x 9mm(L) x 9mm(W) x 4mm(H) Disipador para LAN9512	$2.142	https://www.sigmaelectronica.net/producto/disipador-rpi/
XB24-AWI-001	• Tiene un alcance aproximado de 30 metros en interiores y de 100 metros en exteriores con línea de vista. • La potencia de transmisión es de 1mW (0dB) y la sensibilidad del receptor es de -92dB.	$119.000	https://www.sigmaelectronica.net/producto/xb24-awi-001/

	• Consumo de corriente: 45mA en transmisión, 50mA en recepción. • Funcionan a 2.4 Ghz con una rata máxima de 250Kbps. El módulo trae la antena incorporada.		
TARJETA L86	• Voltaje de operación: 3V hasta 4.3v. Se recomienda usar el L86 con 3.3v. • Dimensiones: 43mm largo x 30mm ancho (lado conexión) x 18mm Alto. • La tarjeta tiene dos hoyos de montaje en las esquinas, situados a 3mm de los lados. El diámetro de los hoyos es de 2mm	$53.550	https://www.sigmaelectronica.net/producto/tarjeta-l86/