Desarrollo de un sistema inalámbrico para la detección de vehículos cercanos para una empresa maderera

Development of a wireless system to detect nearby vehicles for a wood company

Fernando Marciaga[1] & Euclides Samaniego[2*]

[1]Licenciatura en Ingeniería de Sistemas y Computación – Facultad de Ingeniería en Sistemas Computacionales – Universidad Tecnológica de Panamá

[2]Grupo de Investigación en Inteligencia Computacional GIICOM – Facultad de Ingeniería en Sistemas Computacionales – Universidad Tecnológica de Panamá

Resumen Se describe en este trabajo el desarrollo de un sistema que se comunica inalámbricamente con otro igual, el cual será colocado en vehículos o camiones utilizados en la industria maderera y de esta forma mantener la seguridad de los mismos, los conductores y de la carga que contengan. Se da una descripción de los antecedentes y detalles de la empresa a la cual está destinado el sistema, el problema por el cual se desea desarrollar el sistema y los objetivos a alcanzar. Se describe los diferentes módulos utilizados para el desarrollo del sistema, desde el CPU que se desea utilizar por sus características, el módulo de comunicación inalámbrica, el módulo de posicionamiento satelital y la alimentación eléctrica del sistema. Posteriormente detallamos las diferentes herramientas utilizadas para el desarrollo del sistema, desde el lenguaje de programación, software de configuraciones, y configuraciones básicas realizadas al sistema operativo. Por ultimo presentamos varias pruebas de funcionamiento al sistema final, para así lograr entender mejor su modo de operación. Con este sistema se desea mantener alerta a los operadores de los vehículos de carga pesada, manteniendo así mayor seguridad y confianza al momento de transitar por estrechos caminos. Y evitando de este modo cualquier tipo de accidente y pérdida de tiempo al momento de transportar cargas.

Palabras claves RaspberryPi, ZigBee, seguridad.

Abstract The development of a system that communicates wirelessly with other equally, which will be placed on vehicles or trucks used in the wood industry and thus maintain the security of the same, drivers and cargo containing described in this work. A description of the background and details of the company to which the system is intended occurs, the problem you want to develop the system and goals to achieve. The different modules used for system development, from the CPU to be used by its features, the wireless communication module, the module satellite positioning and power supply system is described. Later we detail the different tools used to develop the system, from the programming language, software configurations, and basic settings made to the operating system. Finally we present several test runs at the end system to achieve better understanding of their mode of operation. With this system you want to keep alert operators of heavy vehicles, thus maintaining greater security and confidence when travel on narrow roads. And thus avoiding any kind of accident and loss of time when carrying loads.

Keywords RaspberryPi, ZigBee, security.

* Corresponding author: euclides.samaniego@utp.ac.pa

1. Introducción

El riesgo de accidentes vehiculares debido a la poca visibilidad del entorno, ya sea por efectos climatológicos o por las características físicas del lugar (curvas, callejones, vegetación, etc.), es un peligro siempre existente en las zonas de trabajo donde transitan muchos vehículos, siendo necesaria una solución que advierta a los conductores sobre la presencia de vehículos en su cercanía.

El prototipo en desarrollo pretende mostrar alertas visuales y audibles a los conductores de la cercanía de otros vehículos en sus alrededores, siempre y cuando dichos vehículos también posean instalado el dispositivo en cuestión, abriendo camino a la posibilidad de lograr una zona de trabajo más segura al hacer que todos los vehículos dentro de las instalaciones cuenten con dispositivos de detección, reduciendo así las probabilidades de colisión.

1.1 Objetivos generales

- Desarrollar un sistema que trabajará de manera inalámbrica para la detección de vehículos cercanos para una empresa maderera.

1.2 Objetivos específicos

- Conceptualizar la solución de la problemática.
- Identificar los diferentes módulos para el desarrollo del sistema.
- Establecer ubicación y definir la adaptabilidad del sistema en un vehículo
- Selección de Herramientas necesarias para el desarrollo del sistema.
- Desarrollar el código de programación que se encargara de manejar dicho sistema.
- Desarrollar el modelo para la administración del sistema, generar reportes y mantenimiento.
- Realizar diferentes pruebas de la funcionalidad del sistema.

2. Metodología

2.1 *Hardware* del sistema

2.1.1 CPU

Para el desarrollo de este sistema se ha escogido la RaspberryPi, la cual es uno de los productos más populares para estos fines, tanto por su atractivo precio como por las enormes opciones que trae consigo.

Una de sus principales ventajas radica en que debido a su pequeño tamaño se le puede conectar periféricos de automatización tales como sensores y cámaras que, a pesar de ser piezas simples de tecnología, requieren de complejos controladores de *software* para funcionar.

Sistema operativo: para la RaspberryPi en la actualidad se han creado muchas distribuciones, obviamente basadas en Linux. Entre las más mencionadas y que se encuentran en la web de Raspberry.org están: RASPBIAN, UBUNTU MATE, OSMC, PINET, SNAPPY UBUNTU CORE, OPENELEC, RISC OS.

Para el desarrollo del sistema se utilizará la distribución RASPBIAN. RASPBIAN es una distribución libre de GNU Linux basado en Debian 7.0 o Debian Wheezy optimizado para las capacidades y alcances de los módulos RaspberryPi, su lanzamiento oficial fue en junio de 2012 y junto con el sistema operativo provee software preinstalado para facilitar el uso del mismo. Ver figura 1 y cuadro 1.

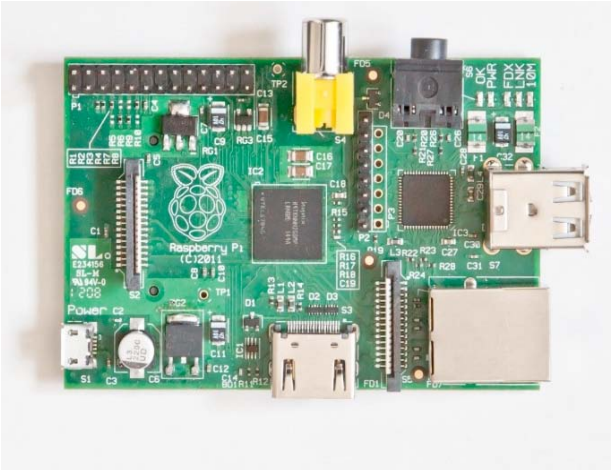

Figura 1. Raspberry Pi 1, modelo 1 B.

Cuadro 1. Especificaciones referentes al modelo B de RaspberryPi

Descripción	RaspberryPi 1 Modelo B
SoC	Broadcom BCM2835 (CPU + GPU + DSP + SDRAM + puerto USB)
CPU	ARM 1176JZF-S a 700 MHz (familia ARM11)
GPU	Broadcom VideoCore IV
Juego de Instrucciones	RISC 32 bits
Memoria	512 MiB SDRAM (Compartido con GPU)
Almacenamiento	SD/MMC/SDIO
Puertos USB 2.0	2
Entradas de Video	MIPI CSI para módulo de cámara RPF
Salidas de Video	RCA, HDMI, Interfaz DSI para LCD
Salida de Audio	Conector 3.5mm., HDMI
Red	10/100 Ethernet RJ45
Periféricos de Bajo Nivel	8xGPIO, SPI, I2C, UART
Consumo	700mA, 3.5 W
Fuente de Alimentación	5V por Micro USB o GPIO Header
Sistemas Operativos	GNU Llinux Debian (Raspbian), Fedora (Pidora), Arch Linux ARM, Slackware Linux, RISC OS
Dimensiones	85.60mm x 53.98 mm

2.1.2 Módulo inalámbrico

Para el desarrollo de la comunicación inalámbrica del sistema se ha utilizado un dispositivo llamado XBee.

Los módulos XBee son soluciones integradas que brindan un medio inalámbrico para la interconexión y comunicación entre dispositivos.

Estos módulos utilizan el protocolo de red llamado IEEE 802.15.4 para crear redes FAST POINT-TO-MULTIPOINT (punto a multipunto); o para redes PEER-TO-PEER (punto a punto). Ver figura 2.

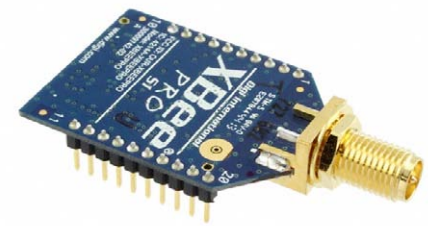

Figura 2. Xbee Pro S1.

Características técnicas: las comunicaciones Zigbee se realizan en la banda libre de 2.4GHz. A diferencia de bluetooth, este protocolo no utiliza FHSS (*Frequency hooping*), sino que realiza las comunicaciones a través de una única frecuencia, es decir, de un canal. Normalmente puede escogerse un canal de entre 16 posibles. Ver cuadro 2.

Cuadro 2: Especificaciones Técnicas XBee PRO S1

Especificaciones	XBee PRO S1
Funciones	
Tasa de datos	250 Kbps
Rango en interiores	300 ft. (90 m)
Rango en exteriores	1 milla (1600 m)
Potencia de transmisión	63mW (18 dBm)
Sensibilidad del receptor	-100 dBm (1% de tasa de error de paquetes)
Tasa de comunicación serial	1200 bps a 250 Kbps
Requerimientos de energía	
Voltaje	2.8 – 3.4 Vdc
Corriente al transmitir	250mA (@ 3.3 Vdc), antena RPSMA: 340mA (@3.3 Vdc)
Corriente al recibir	55mA (@ 3.3 Vdc)
Corriente al apagar	< 10 uA @ 25 ºC
Generales	
Interfaz de datos serial	3.3 Vdc CMOS UART
Banda de frecuencia	ISM 2.4 GHz
Inmunidad a interferencia	DSSS (Direct Sequence Spread Spread Spectrum)
Dimensiones	2.438cm x 3.294cm
Temperatura soportada	-40 a 85 ºC
Redes y Seguridad	
Encriptación	128 bits
ID's y canales	PAN ID (personal Area Network), 64-bit IEEE MAC, 16 canales

Funcionamiento: El módulo requiere una alimentación desde 2.8 a 3.4 V, la conexión a tierra y las líneas de transmisión de datos por medio del UART (TXD y RXD) para comunicarse con un micro controlador, o directamente a un puerto serial utilizando algún conversor adecuado para los niveles de voltaje.

Los módulos Xbee, pueden operar en los siguientes 5 modos:

- Modo Recibir/Transmitir.
- Modo de bajo consumo.
- Modo de comandos.
- Modo Transparente.
- Modo API.

Y cuando no se encuentra en ninguno de estos modos, se encontrará en el modo Idle.

2.1.3 Módulo de posicionamiento satelital

Un localizador GPS es un dispositivo capaz de rastrear a personas, vehículos u objetos a través de coordenadas geográficas (longitud y latitud) en tiempo real y gracias a una red constituida por 24 satélites orbitando a 21.000 km sobre nuestras cabezas con trayectorias sincronizadas para cubrir toda la superficie del globo.

El sistema almacenará su ubicación mediante un receptor GPS, y de este modo enviará dicha posición, de tal modo que se pueda ubicar si se acerca algún vehículo de frente o detrás, o ya sea que este detenido en el recorrido.

2.1.4 Módulo de alerta audiovisual

El funcionamiento del sistema es alertar cuando se acercan o se tienen cerca otros vehículos, se deberá tener un dispositivo que nos muestre dicha alerta, como se mencionó la RaspberryPi cuenta con salidas de audio y video por lo que empleamos una pantalla y un par de bocinas para mostrar y activar las alertas cuando se den. De esta manera será tanto visible como auditiva la forma de alertar al conductor del vehículo.

3. Desarrollo del sistema

3.1 Funcionamiento del sistema

3.1.1 Definición

El concepto básico de operación consiste en que todos los vehículos cuenten con dispositivos idénticos, a los cuales se les configura la información correspondiente al vehículo en donde fue instalado (modelo, matrícula, posición, etc.)

La comunicación se realiza de manera inalámbrica y se pretende cubrir una zona de por lo menos 50 m a la redonda en campo abierto.

Cada dispositivo es capaz de transmitir su señal y recibir la señal de los otros dispositivos que entren en su zona de cobertura.

3.1.2 Conexión de módulos

Conexión de módulo inalámbrico: la conexión del módulo inalámbrico se da mediante el puerto serie, pero en sí el modulo no se puede conectar directamente a la CPU.

Por lo cual se utilizará una placa llama XBee Explorer, mediante esta placa y un cable tipo USB a mini-USB podremos conectar el Xbee al CPU.

Conexión de reloj en tiempo real: se había descrito que la CPU no mantiene la fecha actualizada por lo cual es necesario utilizar un reloj en tiempo real (RTC), para dicho motivo, en el siguiente esquema podemos observar cómo debemos conectar un reloj en tiempo real a través de los GPIO de la RaspberryPi 1 (CPU).

Conexión de módulo de posicionamiento satelital: se desea que el sistema indique la ubicación de los vehículos cercanos, para esto se utilizara un módulo GPS.

Para ello conectaremos el módulo GPS a la RaspberryPi 1 (CPU) a través del GPIO.

El pin TX va conectado al pin RX del módulo GPS y el pin RX de va conectado al pin TX del módulo GPS. Alimentaremos el módulo desde los pines GPIO de 3.3V y GND de la RaspberryPi 1 (CPU).

Conexión de módulo audiovisual: las alertas mostradas al operador del vehículo, por el sistema se darán a través de un pequeño monitor LCD, y también contará con alertas auditivas por lo cual serán conectadas también un par de bocinas para lograr dicho fin. La conexión de video de la pantalla se hace a través del conector RCA de salida de video de la RaspberryPi y el sonido conectado al Jack de audio. La alimentación de la pantalla es de 12 VDC la cual será suministrada directamente desde el vehículo.

3.2 Alimentación eléctrica del sistema

Básicamente el sistema dependerá de la alimentación proveniente del vehículo, directamente será alimentado el sistema audiovisual, y también una batería para mantener el sistema encendido aún estando apagado el vehículo. Ver figura 3.

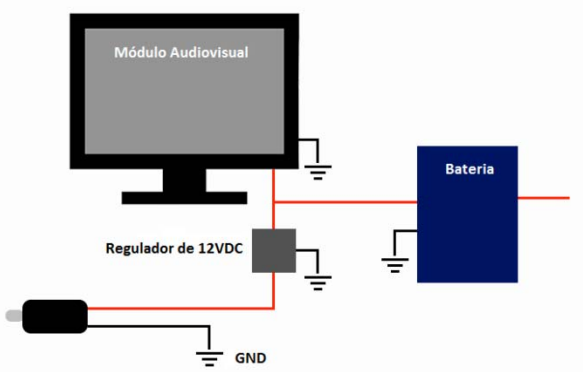

Figura 3. Diagrama de alimentación eléctrica del sistema.

4. Herramientas de desarrollo del sistema

4.1 *Software*

4.1.1 Python

Se trata de un lenguaje interpretado o de *script*, con tipado dinámico, fuertemente tipado, multiplataforma y orientado a objetos. Un lenguaje interpretado o de *script* es aquel que se ejecuta utilizando un programa intermedio llamado intérprete, en lugar de compilar el código a lenguaje máquina que pueda comprender y ejecutar directamente una computadora (lenguajes compilados). La característica de tipado dinámico se refiere a que no es necesario declarar el tipo de dato que va a contener una determinada variable, sino que su tipo se determinará en tiempo de ejecución según el tipo del valor al que se asigne, y el tipo de esta variable puede cambiar si se le asigna un valor de otro tipo.

Es fuertemente tipado ya que, no se permite tratar a una variable como si fuera de un tipo distinto al que tiene, es necesario convertir de forma explícita dicha variable al nuevo tipo previamente. El intérprete de Python está disponible en multitud de plataformas (UNIX, Solaris, Linux, DOS, Windows, OS/2, Mac OS, etc.) por lo que si no utilizamos librerías específicas de cada plataforma nuestro programa podrá correr en todos estos sistemas sin grandes cambios. La orientación a objetos es un paradigma de programación en el que los conceptos del mundo real relevantes para nuestro problema se trasladan a clases y objetos en nuestro programa. La ejecución del programa consiste en una serie de interacciones entre los objetos. Python también permite la programación imperativa, programación funcional y programación orientada a aspectos.

En el cuadro a continuación se describen las librerías utilizadas y sus características.

Cuadro 3. Librerías utilizadas y sus características

Librería	Características
Pygame	Es un conjunto de módulos de Python diseñados para la escritura de juegos. Pygame añade funcionalidad de la excelente biblioteca SDL. Esto le permite crear juegos con todas las funciones y programas multimedia en el lenguaje Python. Aunque Python posee varias librerías para el manejo de multimedia y gráficos. Pygame será la librería utilizada para el desarrollo de la parte visual o grafica del sistema a desarrollar, ya que es de fácil manejo y comprensión en lo que se refiere a desarrollo y puede trabajar bajo cualquier Sistema Operativo, mejorando de esta manera la adaptabilidad de esta parte del sistema.
Pyserial	Es una librería de Python que permite comunicarse a través de comunicaciones por serial (RS-232). Esta librería será utilizada para manejar las comunicaciones de los módulos con la CPU, ya que la funcionalidad de la misma es crear una conexión a través de un puerto serie.
Xbee API	Para lograr la comunicación a través del módulo inalámbrico XBee a través de Python se utilizará esta librería creada exclusivamente para el manejo de dicho módulo. Ya que estos módulos se comunican a través del puerto serie, esta librería depende de la librería Pyserial, explicada en el punto anterior.
Python-GPS	Esta librería la utilizaremos para poder leer los datos recogidos por el módulo de posicionamiento satelital, a través de nuestro programa y así definir nuestra ubicación y poder enviarlas a los sistemas que se encuentren cerca en ese momento.
Python-MySQLdb	Librería de Python utilizada para el manejo de base de datos MySQL. Será utilizado el servidor de base de datos MySQL para almacenar los diferentes datos de lectura que obtenga el sistema, ya que Python posee mencionada librería para obtener un fácil manejo al momento de acceder, actualizar y guardar los datos directamente desde el programa.

Rpi.GPIO	Librería de Python utilizada para controlar el GPIO de la RaspberryPi (CPU). El sistema tendrá opciones de saber cuándo se enciende o apaga el vehículo de manera que pueda saber, lo que debe de hacer en cada caso, para eso utilizaremos los GPIO de la CPU.

4.1.2 Plataforma de desarrollo

Eclipse es una plataforma de desarrollo, diseñada para ser extendida de forma indefinida a través de *plug-ins*. Fue concebida desde sus orígenes para convertirse en una plataforma de integración de herramientas de desarrollo.

No tiene en mente un lenguaje específico, sino que es un IDE genérico, aunque goza de mucha popularidad entre la comunidad de desarrolladores del lenguaje Java usando el *plug-in* JDT que viene incluido en la distribución estándar del IDE.

IDE: hablamos sobre eclipse el cual será la plataforma, y mencionamos el lenguaje de programación Python, de tal modo que necesitaremos un IDE el cual nos permita utilizar el lenguaje de programación a través de la plataforma de desarrollo. Para esto utilizaremos el IDE llamado PyDev.

PyDev es un *plug-in* para Eclipse que permite utilizar este IDE multiplataforma para programar en Python. Cuenta con autocompletado de código (con información sobre cada elemento), resaltado de sintaxis, un depurador gráfico, resaltado de errores, explorador de clases, formateo del código, refactorización, etc.

4.1.3 MySQL

Como se mencionó antes, será utilizada una librería para el manejo del servidor de base de datos MySQL a través de Python. Pero para poder manejar dicho servidor de base de datos, este debe existir en la CPU.

MySQL es un sistema de gestión de base de datos relacional (RDBMS) de código abierto, multi-hilo y multi-usuario, basado en lenguaje de consulta estructurado (SQL).

MySQL se ejecuta en prácticamente todas las plataformas, incluyendo *Linux, UNIX* y *Windows*.

4.1.4 XCTU

A través de este *software* podremos manejar las distintas configuraciones del módulo de comunicación inalámbrica, que utilizaremos en el sistema a desarrollar.

XCTU es una aplicación multiplataforma gratuito diseñado para permitir a los desarrolladores interactuar con módulos Xbee (Módulo de comunicación inalámbrica) a través de una interfaz gráfica fácil de usar. Incluye nuevas herramientas que hacen que sea fácil de configurar y probar los módulos Xbee.

4.2. Configuraciones del sistema operativo

4.2.1 Instalación de paquetes necesarios

Ya que el Sistema Operativo utilizado está basado en Linux, se detallará brevemente los paquetes necesarios para el funcionamiento básico del sistema.

Todos los paquetes se instalarán bajo de línea de comandos, utilizando la terminal de Linux.

Instalar Python: sudo apt-get install python;

Instalar Pygame: sudo apt-get install python-pygame;

Instalar MySQLdb: sudo apt-get install python-mysqldb;

Instalar Pyserial: sudo apt-get install python-serial;

Instalar GPIO: sudo apt-get install python-rpi.gpio;

Instalar Xbee-API: sudo apt-get install python-pip; sudo pip install xbee;

Instalar GPSD y Python-GPS: sudo apt-get install gpsd gpsd-clients; sudo apt-get install python-gps;

Instalar server de MySQL: sudo apt-get install mysql-server mysql-client;

Instalar gestor de auto inicio de sesión: sudo apt-get install mingetty.

4.2.2 Configuraciones básicas

A continuación, veremos ciertas configuraciones que debemos hacer a la RaspberryPi (CPU), para que el sistema pueda funcionar de la mejor manera y sin intervención del operador del vehículo.

Configuración de auto inicio de sesión y aplicación

Para poder hacer un auto inicio de sesión, utilizaremos el gestor *mingetty,* instalado previamente.

Editar archivo

/etc/inittab
Buscar la línea
1:2345:respawn:/sbin/getty 38400 tty1
Reemplazar por
1:2345:respawn:/sbin/mingetty --autologin <user-name> --noclear tty1
Para auto ejecutar la aplicación principal del Sistema, haremos lo siguiente:
Editar el archivo
/root/.bashrc
Agregar al final
cd /App
python Nombre_de_App.py
Configuración de módulo de reloj en tiempo real
Añadir al archivo /boot/config.txt
dtparam=i2c_arm=on
Añadir al archivo */etc/rc.local*
modprobe i2c-bcm2708
echo ds1307 0x68 > /sys/class/i2c-adapter/i2c 1/new_device
modprobe rtc-ds1307
hwclock –s
Configuración de GPSD
Editar archivo /etc/default/gpsd
START_DAEMON=”true”
GPSD_OPTIONS=”/dev/ttyACM0”
DEVICES=””
USBAUTO=”true”
GPSD_SOCKET=”/var/run/gpsd.sock”
Configuración de video
Editar archivo /*boot/config.txt*
sdtv_mode = 0
disable_overscan = 1

4.3 Implementación del sistema

4.3.1 Codificación de la programación

Se presentará a continuación un detalle de diferentes usos de las librerías y funciones de Python utilizadas para el desarrollo del sistema, de manera que puede lograrse un entendimiento básico de cómo se logra comunicar el sistema a través del *software* con cada módulo utilizado.

Conexión con módulo de comunicación inalámbrica
Importamos las librerías necesarias
Import serial
From xbee import Xbee
Iniciamos la conexión serial
Port = serial.Serial(PortSerial, Baudrate)
xbee = Xbee(Port)
Conexión con módulo de posicionamiento satelital
Importamos las librerías
*from gps import **
Iniciamos la recepción de datos GPS
gpsd = gps(mode=WATCH_ENABLE)
Obtener la latitud y longitud
gpsd.fix.latitude
gpsd.fix.longitude
Conexión a servidor de base de datos MySQL
Importamos librerías
Import MySQLdb
Abrir conexión con base de datos
DataBase = MySQLdb.connect(Host, User, Pass, DBName)
Función de implementación gráfica
Importar librerías
import pygame
Iniciar clases de librería gráfica
pygame.init()
Creación de una ventana
Screen = pygame.display.set_mode((Ancho, Alto))
Cargar una imagen
Image = pygame.image.load(Nombre_Imagen)
Cargar un tipo de fuente
Font = pygame.font.Font(Archivo_Fuente,Tamaño)
Usar fuente cargada
Texto = Font.render(Mensaje, Modo, Color)
Dibujar un cuadro en pantalla
pygame.draw.rect(Area, Color,((Posicion_x,Posicion_y), Tamaño))
Cargar un archivo de sonido
pygame.mixer.music.load(Nombre_de_archivo)
Reproducir un sonido
pygme.mixer.music.play(Cantidad_de_veces_a_reproducir)

4.4. Modos de estado del sistema en funcionamiento

Ver figuras 4, 5 y 6.

Figura 4. Estado del sistema sin alerta.

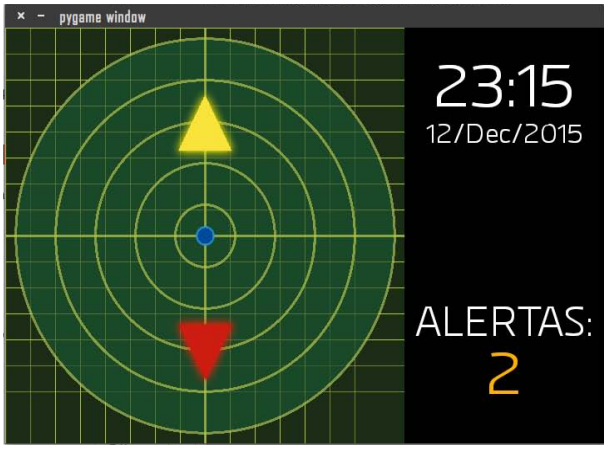

Figura 5. Estado del sistema con alerta de vehículo detenido.

Figura 6. Estado del sistema con alerta de vehículo en movimiento.

5. Conclusiones

El uso de módulos inalámbricos que trabajan a una frecuencia de 2.4GHz, nos brinda muchas ventajas puesto que además de ser una banda libre en todo el mundo, es de alta gama por lo cual es posible que no pueda ser interferida por algunas otras señales.

El uso de una mini CPU como la RaspberryPi facilita el uso de muchas herramientas de desarrollo, ya que permite trabajar sobre un sistema operativo, y ahorrar espacio y costos.

El uso del lenguaje de programación Python, nos da esa ventaja en el desarrollo de la programación del sistema, puesto que posee las bibliotecas necesarias para trabajar con los diferentes módulos utilizados en nuestro sistema, y además posee una gran interacción con el sistema operativo utilizado.

AGRADECIMIENTOS

A Dios por la fortaleza y sabiduría brindada, durante el proceso y la culminación de este trabajo, como también a mis padres y familiares por todo el apoyo brindado.

Al apoyo brindado por parte de mi profesor asesor, Dr. Euclides Samaniego, y demás profesores los cuales me mostraron ese camino hacia la meta, con cada clase impartida.

REFERENCIAS

[1] Mark Lutz, “Aprender a programar con Python 4th Edition”, O’Reilly Media, 2009.

[2] Rumbos Salomon, Rafael Eduardo, EL GRAN LIBRO DE DEBIAN GNU/LINUX, MARCOMBO, S.A.

[3] Alejandro Cobo López, “Guía Raspberry Pi”, (cc) 2013 – 2015.

[4] RaspberryPi User Guide, 2013, URL: http://www.cs.unca.edu/~bruce/Fall14/360/RPiUsersGuide.pdf

[5] Digi International Inc, 2014, URL: http://ftp1.digi.com/support/documentation/90000982_S.pdf

[6] Oracle Corporation, 2014, URL: http://dev.mysql.com/usingmysql/get_started.html