UNIVERSIDAD CATÓLICA ANDRÉS BELLO

FACULTAD DE INGENIERÍA

ESCUELA DE INGENIERÍA INFORMÁTICA

SISTEMA DE CONTROL INALÁMBRICO PARA VEHÍCULOS QUE ACCEDEN A UN PEAJE.

TRABAJO ESPECIAL DE GRADO

Presentado ante la

UNIVERSIDAD CATÓLICA ANDRÉS BELLO

Como parte de los requisitos para optar al título de

INGENIERO EN INFORMÁTICA

REALIZADO POR Horacio Baquero
Marcos Salazar

PROFESOR GUIA Biagio Cante

FECHA 24 de septiembre de 2007

UNIVERSIDAD CATÓLICA ANDRÉS BELLO

FACULTAD DE INGENIERÍA

ESCUELA DE INGENIERÍA INFORMÁTICA

SISTEMA DE CONTROL INALÁMBRICO PARA VEHÍCULOS QUE ACCEDEN A UN PEAJE.

Este Jurado; una vez realizado el examen del presente trabajo ha evaluado su contenido con el resultado: 20 puntos, Mención Publicación

J U R A D O E X A M I N A D O R

Firma: Nombre: BIAGIO CANTE

Firma: Nombre: JORGE GARCÍA

Firma: Nombre: HECTOR GUTIERREZ

REALIZADO POR	Horacio Baquero Marcos Salazar
PROFESOR GUIA	Biagio Cante
FECHA	24 de Septiembre de 2007.

CONTENIDO

1. Introducción 1
1.1. Introducción 1
1.1. Planteamiento del Problema 2
1.2. Objetivo General 3
1.3. Objetivos Específicos 3
1.4. Limitaciones 5
1.5. Alcance 5
1.6. Justificación 5
2. Marco teórico 7
2.1. Hardware 7
2.1.1. Microcontroladores 7
2.1.1.1. PIC 16F84 8
2.1.2. EIA-232: Puerto Serial 9
2.1.3. LM78XX 10
2.1.4. Tecnología TTL 10
2.1.5. MAX-232 11
2.1.6. Radiofrecuencias y Antenas 11
2.1.7. Dispositivos de Transmisión 14
2.1.7.1. Sunrom STR-433 14
2.1.7.1.1. Visión General 14
2.1.7.1.2. Distribución de Pines 15
2.1.7.1.3 Dibujo Mecánico 15
2.1.7.1.4. Operación 16
2.1.7.1.4.1. Detección AM Super Regenerativa 16
2.1.7.1.4.2. Data Slicer 16
2.1.7.1.4.3. Suministro de Poder 17
2.1.7.1.4.4. Antena 17
2.1.7.2. Sunrom STT-433 18
2.1.7.2.1. Visión General 18
2.1.7.2.2. Distribución de Pines 19
2.1.7.2.3. Dibujo Mecánico 19
2.1.7.2.4. Operación 19
2.1.7.2.4.1. Teoría 19
2.1.7.2.4.2. Tasa de Transmisión 20
2.1.7.2.4.3. Oscilador Estabilizador por Onda Acústica de Superficie (SAW) 20
2.1.8 Puerto Paralelo 20
2.2.Software 22
2.2.1. IC-Prog 22
2.2.1.1. Mecanismos de Seguridad 22
2.2.1.1.1. Encriptación Asimétrica 22
2.1.1.1.1. RSA 23
2.2.1.1.2. Funciones Criptográficas Resumen (Hash) 24
2.2.1.1.1.2. SHA-1 24
2.2.2. Redes Privadas Virtuales 25
2.2.3. Firewall(Cortafuegos) 27

2.2.4. Invocación Remota de Métodos (RMI) 28
2.2.5. MySQL 29
2.2.6. JDBC 30
3. Marco Metodológico 32
3.1. XP:eXtreme Programing 32
3.1.1. Planificación 32
3.1.2. Diseño 33
3.1.3. Desarrollo 33
3.1.4. Pruebas 33
3.2. Justificación de la Metodología 34
3.3. Plan de Desarrollo 34
4. Desarrollo 35
4.1. 1ra Iteración. Diseño de Hardware y Software 35
4.1.1. Planificación 35
4.1.2. Diseño 38
4.1.2.1. Dispositivos de Transmisión 38
4.1.2.2. Dispósitivos de Recepción 40
4.1.3. Desarrolllo 41
4.1.4. Pruebas 42
4.2. 2da Iteración. Diseño y Desarrollo de Clases de Conexión y Base de Datos 43
4.2.1. Planificación 43
4.2.2. Diseño 43
4.2.3. Desarrollo 46
4.2.4. Pruebas 46
4.3. 3ra Iteración. Integración de Código. Implementación VPN 47
4.3.1. Planificación 47
4.3.2. Diseño 49
4.3.3. Desarrollo 49
4.3.4. Pruebas 49
4.4. 4ta Iteración. Interfaz Gráfica. Testigo de Paso. Impresión de Ticket. Bitácora de Flujo Vehicular. Cifrado. 49
4.4.1. Planificación 50
4.4.2. Diseño 50
4.4.3. Desarrollo 53
4.4.4. Pruebas 54
4.5. 5ta Iteración. Diseño y Desarrollo Aplicación Administrador 54
4.5.1. Planificación 54
4.5.2. Diseño 55
4.5.3. Desarrollo 57
4.5.4. Pruebas 58
5. Resultados 59
6. Conclusiones y Recomendaciones 63
6.1. Conclusiones. 63
6.2. Recomendaciones. 65
7. Referencias Bibliográficas 67
8. Glosario de Términos 69
9. Apéndice 76
Apéndice A. Información Adicional. 77

9.A.1. Java. 77
9.A.2. Manejador de Bases de Datos. 79
Apéndice B. Circuitos. 82
Apéndice C. Diagramas de Secuencias. 83
Apéndice D. Pseudocódigo del Microcontrolador PIC. 86
Apéndice E. Fotos de Circuitos. 87
Apéndice F. Iteraciones. 90
Apéndice G. Planificación de Pruebas Unitarias. 91
Apéndice H. Ejemplos de Archivos. 92
Apéndice I. Planificación de Desarrollo del Proyecto. 93

ÍNDICE DE FIGURAS

Figura 2.1. Esquema de PIC16F84. 9
Figura 2.2. Puerto Serial. DB-25. DB-9. 9
Figura 2.3. LM7805. 10
Figura 2.4. Esquema de MAX232. 11
Figura 2.5. Espectro Electromagnético. 11
Figura 2.6. Dirección de la señal. 13
Figura 2.7. Velocidad de Propagación. 13
Figura 2.8. Longitud de Onda. 13
Figura 2.9. Dispositivo de Recepción. STR-433. 14
Figura 2.10. Distribución de Pines. STR-433. 15
Figura 2.11. Dibujo Mecánico. STR-433. 15
Figura 2.12. Dispositivo de Recepción. STT-433. 18
Figura 2.13. Distribución de Pines. STT-433. 19
Figura 2.14. Dibujo Mecánico STT-433. 19
Figura 2.15 Puerto Paralelo 20
Figura 2.16. Interfaz RMI. 28
Figura 4.1. Diagrama del Sistema. 35
Figura 4.2. Diagrama Distancias Fisicas 36
Figura 4.3. Circuito para Transmisión Inalámbrica. 38
Figura 4.5. Pseudocódigo para Programación del PIC. 39
Figura 4.6. Regulador de Voltaje 12V a 5V. 39
Figura 4.7. Regulador de Voltaje 9V a 5V. 40
Figura 4.8. Circuito para Recepción Inalámbrica. 40
Figura 4.9. Regulador de Voltaje 110 V a 5V. 41
Figura 4.10. Diagrama de Clases para Lectura de interfaz RS-232 41
Figura 4.11. Estructura de la trama transmitida. 41
Figura 4.12. Modelo Entidad-Relación. Base de Datos central. 44
Figura 4.13. Primer Diagrama de Clases para el servidor. 45
Figura 4.15. Primer Diagrama de Clases para la Aplicación Completa. 48
Figura 4.16. Formato de Ticket. 51
Figura 4.17. Segundo Diagrama de Clases para la Aplicación Completa 52
Figura 4.19. Circuito de prueba para el puerto paralelo. 54
Figura 4.20. Base de Datos para administradores. 55

Figura 4.21. Diagrama de Clases de la aplicación de administración. 56
Figura 9.B.1 Circuito programador para Micrcontrolador PIC 84
Figura 9.C.1 Diagrama de Secuencias para el primer diagrama de clases (servidor) 85
Figura 9.C.2 Diagrama de Secuencias para aplicación Cliente 86
Figura 9.C.3 Diagrama de Secuencias para aplicación de administración 87
Figura 9.D.1 Pseudocódigo 88
Figura 9.E.1 Circuito Transformador. 89
Figura 9.E.2. Regulador de Voltaje para el Vehículo. 89
Figura 9.E.3. Dispositivo Receptor (Empaquetado). 90
Figura 9.E.4. Dispositivo Receptor (Abierto). 90
Figura 9.E.5. Dispositivo Emisor (Empaquetado). 91
Figura 9.E.6. Dispositivo Emisor (Abierto). 91
Figura 9.G.1. Archivo de Movimientos del Administrador (Log). 92
Figura 9.H.2. Archivo de Registro de Tránsito. 92
Figura 9.I.1. Planificación de Desarrollo del Proyecto. 93

ÍNDICE DE TABLAS

Tabla 2.1. Equivalencia de Señales RS-232 9
Tabla 2.2. Espectro Radioeléctrico. 12
Tabla 2.3. Propiedades del STR-433. 15
Tabla 2.4. Descripción de Pines. STR-433. 15
Tabla 2.5. Propiedades del STT-433. 18
Tabla 2.6. Descripción de Pines. STT-433. 19
Tabla 2.7. Descripción de Pines Puerto Paralelo. 21
Tabla 3.1. Fases de Desarrollo. 34
Tabla 4.1. Cuadro Nacional de Atribución de Bandas de Frecuencias. (Extracto) 37
Tabla 4.2. Conexiones de Pines con Transmisor STT-433. 38
Tabla 9.F.1. Iteraciones. 91
Tabla 9.G.1. Planificación de Pruebas Unitarias. 90

SINÓPSIS

El presente trabajo de grado se propone realizar un sistema automatizado de peajes, soportado por dispositivos de radio frecuencia y microcontroladores en la parte de hardware para transmitir información cifrada desde el vehículo hacia la caseta típicamente de pago. El principio es detectar si un vehículo ha sido hurtado, y en caso de que haya sido así, detenerlo en el lugar. La tecnología actual de microcontroladores PIC combinada con transmisores de radio frecuencia a 433 MHz permite elaborar circuitos apropiados para condiciones donde el costo y la confiabilidad son preocupaciones primarias, aportando un margen de error tolerable y sencillo de manejar. Además, un marco orientado a objetos basado en una plataforma Java, resulta en una aplicación segura, portable y de fácil mantenimiento. Se optó por una interfaz de Invocación Remota de Métodos (ofrecida por Java) para elaborar la comunicación entre la caseta del peaje y una base de datos remota implementada en MySQL. Adicionalmente, se toman en cuenta m de seguridad para la transmisión y el manejo de la data involucrada, debido a la calidad privada de ella.

La metodología que se eligió para el desarrollo del trabajo de grado es conocida como XP (*eXtreme Programing*). Ella permite tener pequeñas aplicaciones consecutivas hasta llegar a la versión final y posee entre sus puntos clave la elaboración de pruebas constantes que garantizan de cierta manera un producto final completo. La estructura de este documento está descrita de la siguiente manera:

- Capítulo I. Planteamiento del problema. Objetivos generales y específicos. Limitaciones, alcance y justificación de él trabajo aquí propuesto.
- Capítulo II. Fundamentos teóricos.
- Capítulo III. Metodología de desarrollo.
- Capítulo IV. Detalla el desarrollo de cada objetivo por cada iteración en la metodología antes nombrada.
- Capítulo V. Resultados obtenidos en cada una de las etapas.
- Capítulo VI. Conclusiones y Recomendaciones.
- Capítulo VII. Bibliografía utilizada.
- Capítulo VIII. Glosario de Términos.
- Capítulo IX. Apéndice.

1.1 Introducción

En nuestro país, según los registros que se manejan en la página de web oficial del Cuerpo de Investigaciones Científicas, Penales y Criminalísticas (www.cicpc.gov.ve) y para el mes de Julio de 2007, se han recuperado aproximadamente 4,686 vehículos en el territorio nacional. Aunque la información referente a la cantidad de vehículos robados es "[...] clasificada que no forma parte del dominio público", uno solo puede especular sobre los números que se manejan internamente. Sin embargo, de lo que sí se puede estar seguro es que por cada robo/hurto se genera un gasto personal de compensación por la pérdida (transporte público, un nuevo vehículo) que impacta negativamente sobre el bolsillo de la víctima. Adicionalmente, ésta se ve a sí misma irremediablemente separada de su automóvil, pues la confianza en las autoridades se reduce por cada día que pasa. Para la persona común sólo se percibe que no se está haciendo nada que resulte efectivo con respecto a la seguridad de los bienes.

Debido a la velocidad con la que responde, la seguridad de automóviles basada en localización por satélite parece una propuesta eficaz. No obstante, los altos costos de ella y lo relativamente nuevo de la tecnología impiden que se afiance como la solución ideal. Venezuela, además, utiliza solo 2% de su presupuesto nacional para atacar el problema, cuando en países donde el índice de seguridad es tolerable, se utiliza entre el 8% y 10% del presupuesto. Es así como una propuesta basada en puntos de control poco costosos, soportado por un sistema de comunicación de ondas de radio de corto alcance, parece responder tanto a las limitantes del entorno como a los requerimientos de seguridad del país. Nótese la naturaleza estructural y no coyuntural del problema. Es necesario enfrentarlo con un sistema íntegro que limite el espectro de acción de la delincuencia y que a la vez responda para la prevención de ella, sin dejar de lado la capacidad de actualización y mejora con el tiempo.

De esta manera, se presenta un preámbulo al fin último de este trabajo especial de grado: responder a una necesidad del país mediante una solución tecnológica efectiva en su justificación, planeamiento y desarrollo, basada en comunicaciones inalámbricas de alcance regulado; y soportada por la premisa de que cada vehículo tiene un serial único que lo identifica y diferencia.

1.2 Planteamiento del problema

Entre el siglo XX y el XXI llevamos ya más de 60 años siendo testigos de la llamada tercera revolución industrial. La tecnología digital se ha vinculado de tal forma al ser humano que hoy es difícil imaginar el día a día sin ella; y aunque en algún momento fue poco más que un tema de investigación, hoy podemos ver a los negocios, la salud, la política e inclusive la seguridad haciendo uso de sus atributos. Las ventajas son cuantiosas y abrumadoras, y la sociedad ha buscado la manera de sacarle provecho; a veces no de la mejor forma, pero siempre asegurándose de que no pase de manera desapercibida.

Venezuela, por su condición de país en vías de crecimiento, no siempre se ha mostrado capaz de ir al ritmo vertiginoso de ese avance tecnológico. Entre otras cosas, la inflación, el bajo ingreso per cápita y la base industrial poco desarrollada, más que indicadores, se pueden interpretar como resultados del progreso histórico del país; hoy en gran medida, la tecnología que manejamos es de origen extranjero.

Por otro lado, la sociedad se encuentra sumida en una escalada de crimen y violencia casi sin precedentes, y aunque no es nuestra intención iniciar una diatriba sobre política interior, es importante establecer la trascendencia e impacto que tiene la delincuencia sobre el día a día del venezolano, y de ahí luego el fin de este proyecto. En primera instancia, el secuestro y el homicidio son parte de una realidad trágica que coloca al país en el tope –o cerca- de los *rankings* mundiales de inseguridad. Aún cuando esto sale del enfoque que queremos establecer, la verdad es que tales números son los que se comportan como catalizador eventual para la adopción de medidas a largo plazo que respondan a las necesidades de seguridad. Los planes y políticas consecuentes usualmente buscan atajar las dificultades de manera íntegra. No se ataca el robo de autos únicamente, sino al *secuestro express* que acontece luego, y a la estafa de la venta del vehículo, por ejemplo. Que el hurto y los crímenes contra la propiedad no estén tan arraigados a la vida cotidiana en comparación con otros países, no significa tampoco que no pese caprichosamente sobre el estrés del individuo y le hagan ulteriormente gastar cuantiosas sumas de dinero en dispositivos de contrarresto.

Gran cantidad de reportes describen autos robados en el interior que resultan encontrados en Caracas y viceversa; sea bajo condiciones de desmantelamiento posterior o para uso particular, se entiende un recorrido importante de distancia entre el

punto de robo y el lugar de recuperación. Aunque poco se puede hacer contra lo primero, la naturaleza del sistema propuesta reside en la integridad de la información y el no repudio de la que se maneja (mismos principios básicos de una comunicación segura). Cualquier detalle que no parezca estar acorde con la manejada por las autoridades, puede resultar en una señal de alerta. Ya quedará de parte de ellos hacer las experticias subsiguientes. Recordemos las ideas de Kevin Mitnick con respecto a la ingeniería social; según él, el hueco de seguridad más elemental de un sistema estaba en las personas que lo usan.

Finalmente, el robo de vehículos y el uso de estos implican un rango de acción limitado por las vías de tránsito que estén disponibles. Podría parecer abrumador en primer lugar, pero ofrece una oportunidad de acción. Un peaje propone beneficios inherentes al cobro de tributos; es un punto de control constituido sobre importantes vías de transporte donde cada vehículo debe reducir su velocidad para cumplir con el pago de la tarifa. Parece oportuno aprovechar la obligación de los transeúntes con el estado y verificar el aspecto legal de los vehículos que manejan; en la llegada al lugar de control, sus datos de identificación nacional (serial de motor, chasis o número de bastidor) e información física (color, placa, marca, año) serían cotejados con respecto a una base de datos que reúna qué automóviles cuentan con la condición de estar solicitados en el territorio nacional. Un sistema automatizado es la propuesta para ello.

1.3 Objetivo General

Diseñar e implementar un sistema de control inalámbrico para el acceso de los vehículos a un peaje.

1.4 Objetivos Específicos

- Recaudar información desde la fuente original (peaje) para establecer niveles de flujo vehicular y puntualizar datos sobre la infraestructura física y tecnológica ahí existente.
- Definir estructuras de datos necesarios para la correcta verificación del vehículo.
- Estudiar el sistema de información actualmente implementado en el registro automotor nacional.
- Estudiar el espectro radioeléctrico establecido por CONATEL para evitar interferencias con otros dispositivos de comunicación.

- Evaluar los distintos tipos de transmisores y receptores que existen en el mercado cuyas características encajen con las necesarias para el desarrollo del sistema.
- Evaluar los distintos tipos de microcontroladores que existen en el mercado cuyas características encajen con las necesarias para el desarrollo del sistema.
- Definir e implementar políticas de seguridad a aplicarse sobre los dispositivos de transmisión y recepción.
- Desarrollar un mecanismo de conexión entre el módulo de recepción de datos y la PC preparada para manejarlos.
- Desarrollar la base de datos donde se registrará el tránsito local; su función será sostener la información para comparar luego con la base de datos remota, con los parámetros de la existente en el registro automotor nacional actual.
- Establecer la conexión entre la base de datos local y la remota.
- Desarrollar una interfaz gráfica para el usuario y el administrador del sistema, con la posibilidad de manejar los diferentes dispositivos que se ven involucrados en todo el proceso, exceptuando los de transmisión y recepción.
- Instalar y configurar los equipos de transmisión en los diferentes vehículos, luego de estudiar las condiciones para lograr funcionalidad de ellos.
- Instalar y configurar el equipo de recepción luego de estudiar las condiciones que logren funcionalidad de él y de la interacción con los dispositivos de transmisión.
- Programar los microcontroladores para que se comporten como una interfaz entre el sistema de transmisión y recepción, la PC local y el dispositivo de subida y bajada de la barra de paso.
- Desarrollar el módulo para la impresión de tickets en la PC local donde se presentará la información capturada desde el módulo de recepción.
- Realizar las pruebas de verificación de funcionalidad de todo el sistema instalado.

1.5 Limitaciones.

- Para el uso de la información y desarrollo de la aplicación global se utilizará Java como lenguaje de programación.
- La impresión de tickets de comprobación será hecha a través de impresoras típicas de puerto serial (o paralelo en su defecto).
- Será probado para 3 automóviles y un solo puesto de control.

1.6 Alcance.

- La simulación del sistema será realizada bajo condiciones fidedignas a la realidad, tomando en cuenta la infraestructura física de un peaje y usando vehículos automotores, sensores de proximidad, receptores y demás factores relacionados con el sistema.
- Para efectos de la simulación de la base de datos remota y el terminal se utilizarán dos(2) Laptops, además del mecanismo de control de la barra de paso y los sensores correspondientes; todos ubicados en el área del estacionamiento de la UCAB.

1.7 Justificación.

En un país agobiado por la inseguridad, resulta menester la adopción de medidas que busquen frenar los mecanismos de actuación de la delincuencia y además busque prevenirla. A fin de obtener resultados definitivos a medianos plazos, estas soluciones deben atacar el problema desde tantos frentes sea posible; naturalmente, las tecnologías de información se asoman como ventajas importantes que deben ser aprovechadas. El sistema propuesto puede servir en primer lugar como un mecanismo reactivo, donde en el instante que el sistema verifica alguna irregularidad, el carro y su(s) ocupante(s) son detenidos de inmediato. Pero luego de ello, puede convertirse en un mecanismo de intimidación –y prevención: el conocimiento de la existencia del sistema puede persuadir al delincuente de mantenerse lejos de cualquier peaje del país y restringir su área de movimiento.

Por otro lado, el sistema además muestra gran potencial para no solo servir como un coadyuvante para la seguridad sino además de plataforma para otras aplicaciones. Futuras mejoras podrían incluir un sistema de rastreo, prepago de peaje y estadísticas o inclusive en la optimización del tráfico en las vías de comunicación donde

sea utilizado. No solo así, puede ser de utilidad para aplicaciones privadas de menor escala donde el control de vehículos sea necesario. Aplicaciones para la administración de estacionamientos privados, control en urbanizaciones e inclusive manejo en las importaciones de vehículos aparecen como propensas para la implementación.

Marco Teórico

2.1 HARDWARE

2.1.1. Microcontroladores

Un microcontrolador es un tipo de microprocesador enfatizado hacia la autosuficiencia y efectividad de costo que integra en su interior las tres unidades funcionales de un computador (Unidad Aritmético-Lógica, Unidad de Control y Registros) y elementos adicionales como una Memoria de Acceso Aleatorio. Son de propósito específico, es decir, son programados para realizar una tarea única.

Generalmente consta de:

- CPU o procesador. Es el cerebro del sistema que procesa todos los datos que viajan a lo largo del bus.
- Memorias. Está formada por una no volátil (*ROM, EEPROM, FLASH*) donde se almacenan los programas y una volátil (RAM) donde se almacenan los datos.
- Reloj principal. Normalmente todos los microcontroladores tienen incorporados circuitos osciladores para el funcionamiento de éstos.
- Puertos E/S (Entrada/Salida). Soportan las líneas que comunican al microcontrolador con los periféricos externos.
- Perro guardián o *Watchdog*. Contador que resetea al microcontrolador cada vez que rebosa. Sirve para evitar fallos de funcionamiento, por lo que hay que inicializarlo periódicamente antes de que rebose.
- Protección ante fallo de alimentación o *Browout*. Circuito que resetea al microcontrolador cuando la tensión de alimentación baja de un cierto límite.
- Temporizadores. Para controlar periodos de tiempo.
- Convertidores A/D y D/A. (Analógico/Digital y Digital/Analógico)
- Comparadores analógicos.
- Moduladores de anchura de impulsos.
- Puertos de comunicación. Tanto serie como paralelo.
- Control de interrupciones.

Debido a que los microcontroladores sólo incluyen las características específicas para una tarea, su coste es relativamente bajo. Un microcontrolador típico realiza

funciones de manipulación de instrucciones, posee E/S de acceso fácil y directo, y un proceso de interrupciones rápido y eficiente. Además reducen de manera notable los costes de diseño. Hay gran variedad de microcontroladores. Dependiendo de la potencia y características que se necesiten, se pueden elegir microcontroladores de 4, 8, 16 ó 32 bits. Existen versiones especializadas que incluyen módulos para comunicaciones, teclados, procesamiento de señales, procesamiento de video, y otras tareas.

2.1.1.1 PIC 16F84A

El PIC16F84A es un microcontrolador de la familia PIC producida por *Microchip Technology*. Históricamente fue el primero con un algoritmo de programación serial y memoria EEPROM que lo hizo propenso al uso amateur. Además de esto, solo se necesitaba de un programador sencillo para codificar, borrar y reprogramar el chip. Eventualmente, se habría de convertir en uno de los más populares de los ofrecidos por Microchip y el más prominente de su serie de 14 bits. Aunque hoy Microchip ha avanzado suficiente en su línea de productos como para considerar otros microcontroladores (PIC16F628 por ejemplo), la realidad es que el 16F84A no parece estar listo para desaparecer.

Sus características se resumen a continuación:

- Características de alto desempeño RISC:
  - Memoria de programa de 1024 palabras.
  - 68 Bytes de RAM de datos.
  - 64 Bytes de Memoria EEPROM.
  - Velocidad de Operación: 20 MHz por entrada de reloj y 200 ns por ciclo de instrucción.
  - Bytes de datos de 8 bits.
  - Palabras de instrucciones de 14 bits.
  - 35 Instrucciones de una sola palabra.
  - 15 Registros Especiales de Funciones de Hardware.
- Características Periféricas:
  - 13 Pins de E/S con dirección de control individual.
  - 25 mA máximos por pin en modo sumidero.
  - 25 mA máximos por pin en modo fuente.

- Tecnología Mejorada por CMOS[1] para memoria Flash/EEPROM:
- Tecnología de bajo poder y alta velocidad.
- Diseño estático completo.
- Rango Operacional de Voltaje: Comercial e Industrial 2.0 V a 5.5 V.
- Bajo consumo de poder: < 2 mA a 5V y 4 Mhz.

Las ventajas residen en la sencillez de su de manejo, programación y la gran cantidad de documentación que existe, así como también se bajo costo. La Figura 2.1 muestra un diagrama de la distribución de los pines.[2]

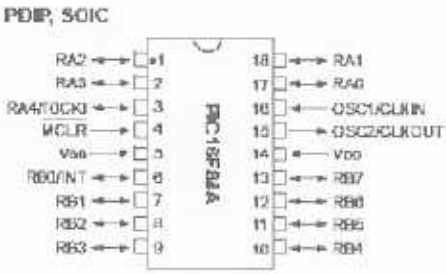

Figura 2.1. Esquema de PIC16F84A.
http://ww1.microchip.com/downloads/en/devicedoc/35007b.p

2.1.2. EIA-232: Puerto Serial

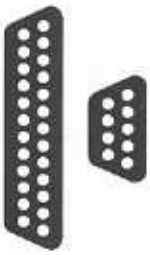

Figura 2.2.Puerto Serial. DB-25. DB-9.
http://www.machinetoolhelp.com/_Images/rs232-DB25-DB9.jpg.

EIA -232 (*Electronic Industries Association – 232*) se denomina a una interfaz de comunicación serial de datos binarios conocida hasta 1997 como RS-232 (*Recomended Standard – 232*, aunque hoy todavía se utiliza ese nombre) resumida en un protocolo donde se definen señales de voltaje específicos, temporización y función de señales, un protocolo de intercambio de información, y los conectores mecánicos que habiliten la comunicación.

La figura 2.2 muestra las modalidades en las que se puede conseguir dicha interfaz: Conectores DB-25 y DB-9 macho. Para realizar las conexiones entre dispositivos es recomendable utilizar cables de longitud de 15 metros o menos ya que el estándar estaba definido para distancias cortas, sin embargo, el límite está establecido en realidad por la capacitancia de los cables: cables de baja capacitancia permiten conexiones confiables a más de 50 metros.

Los datos binarios también están definidos de manera particular:

Señales entre +3V y +12 V	Cero (0) lógico.
Señales entre -3V y -12V	Uno (1) lógico.
Señales entre -3V y +3V	Valores de transición. Sin valor asignado.

Tabla 2.1. Equivalencia de Señales RS-232
Elaboración Propia.

[1] CMOS: Acrónimo para *Complementary Metal-Oxide Semiconductor.*
[2] PDIP, SOIC: Acrónimos para el empaquetado del circuito. *Dual-In-Line Plastic Packages. Small Outline Integrated Circuit.*

En cuanto a velocidad, el estándar recomienda un tope de 20,000 bps, pero debido a que la intención original era proveer un mecanismo de comunicación para Módems, es normal encontrar velocidades superiores (38,400, 57,600 bps) usando niveles de señales compatibles con el estándar.

2.1.3 LM78XX

Es necesario en todo circuito eléctrico, bien sea alimentado de corriente continua o corriente alterna, tener un dispositivo que se encargue de regular la diferencia de potencial para el buen funcionamiento del mismo.

Figura 2.3. LM7805.
http://www.neoteo.com/Portals/0/neoteo/1093full.jpg

Existe una amplia gama de reguladores de voltaje de salida fija, entre ellos se encuentra la familia de LM78XX; "XX" se refiere al voltaje de salida fija que provee dicho regulador, entre estos voltajes se pueden conseguir valores de: 5, 6, 8, 9, 10, 12, 15, 18 y 24V, para el presente trabajo de grado se eligió específicamente el LM7805 (Provee un voltaje de salida fija de 5V). Es importante mencionar que esta familia de reguladores poseen protección contra sobrecargas térmicas y contra cortocircuitos, en caso de producirse una temperatura de juntura mayor a 125° desconecta el flujo de energía.

2.1.4 Tecnología TTL[3]

TTL se refiere a la tecnología para diseñar y fabricar circuitos digitales integrados que utilizan compuertas lógicas consistiendo primordialmente de transistores bipolares. La característica distinguible de la familia básica TTL es que utilizan voltaje cercano a los +5V y utilizan una relativamente alta cantidad de corriente para manejar sus niveles lógicos. Voltaje menor a 1V equivale al cero (0) lógico, y por encima de 3.5V equivale al uno (1) lógico. Las señales TTL tienen una característica particular y es que la señal de entrada a una compuerta obtiene un valor de uno (1) lógico si se deja desconectada. Esto significa que el trabajo principal de una señal de entrada debe ser, cuando así se quiera, bajar la señal al cero (0) lógico. Más adelante nos daremos cuenta de la importancia y utilidad de la lógica Transistor a Transistor; además de la ventaja vinculada a estos niveles de voltaje que maneja.

---

[3] TTL: Acrónimo para *Transistor-transistor Logic.*

2.1.5 MAX232

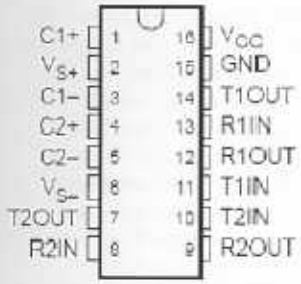

Figura 2.4. Esquema de MAX232.
http://www.ortodoxism.ro/datasheets/texasinstruments/max232.pdf

El MAX232 es un manejador/receptor dual que incluye un generador capacitivo de voltaje para niveles compatibles con el estándar EIA-232 a partir de una fuente única de 5V. Al usar las entradas del receptor (pines R1, R2) se transformar los voltajes típicos EAI-232 a niveles compatibles TTL/CMOS llegando a soportar hasta ±30V. Por otro lado, las entradas del manejador (C1, C2) transforman entradas TTL en valores compatibles con EAI-232. Aquí entonces, establecemos el vínculo entre la tecnología TTL y el estándar EAI-232: a través del MAX232, la comunicación entre un dispositivo que maneje dicha lógica y el puerto serial se hace posible.

2.1.6 Radiofrecuencias y Antenas

El término radiofrecuencia (*RF*), también denominado espectro de radiofrecuencia o RF, se refiere a una frecuencia de oscilación ubicada particularmente entre los 3MHz y 30GHz. Este rango se define por las limitaciones en tazas de vibración que la mayoría de los sistemas mecánicos pueden responder; la radiofrecuencia ocurre cuando se utilizan señales eléctricas originadas por corriente alterna, es decir, la oscilación en circuitos eléctricos. Dependiendo de esa rapidez de oscilación se puede decir que una señal RF cubre las siguientes bandas del medio, llamado *Espectro de Radio*:

Figura 2.5. Espectro Electromagnético.
http://es.wikipedia.org/wiki/Espectro_electromagn%C3%A9tico

Y luego más específicamente:

Nombre	Abreviatura Inglesa	Banda ITU	Frecuencias
-	-	-	Inferior a 3 Hz
Extremely low frequency. Frecuencia extremadamente baja.	ELF	1	3-30 Hz
Super low frequency. Frecuencia super baja.	SLF	2	30-300 Hz
Ultra low frequency. Frecuencia ultra baja.	ULF	3	300–3000 Hz
Very low frequency. Frecuencia muy baja.	VLF	4	3–30 kHz
Low frequency. Frecuencia baja.	LF	5	30–300 kHz
Medium frequency. Frecuencia mediana.	MF	6	300–3000 kHz
High frequency. Frecuencia alta.	HF	7	3–30 MHz
Very high frequency. Frecuncia muy alta.	VHF	8	30–300 MHz
Ultra high frequency. Frecuencia ultra alta.	UHF	9	300–3000 MHz
Super high frequency. Frecuencia super alta.	SHF	10	3-30 GHz
Extremely high frequency. Frecuencia extremadamente alta.	EHF	11	30-300 GHz
-	-	-	Sobre 300 GHz

Tabla 2.2. Espectro Radioeléctrico.
http://es.wikipedia.org/wiki/Radiofrecuencia

Para la emisión de estas ondas, se parte de los principios estudiados por James Clerk Maxwell y Heinrich Hertz. Según sus estudios, se encontró que una partícula se mueve aceleradamente cuando está eléctricamente cargada. Una corriente eléctrica establecida a lo largo de un cable resulta en partículas cargadas pero solo en el caso de que la corriente sea alterna; únicamente ahí es que estas experimentan un cambio de velocidad con respecto al tiempo. Supongamos luego que conectamos una varilla metálica a una fuente de corriente alterna (el tamaño de la varilla es también un factor importante, pero hablaremos de él más adelante). Los electrones, según los conceptos básicos duales de onda y materia estudiados por la mecánica cuántica, atraviesan de ida y vuelta la varilla. Cada vez que llega al extremo, se reflejan y siguen exactamente a la onda de ida. Esto resulta en el hecho de que solo 1% de la onda es irradiada; sin embargo, esta onda también rebota y regresa, para irradiar un 1% adicional. Este proceso se repite cientos de veces, hasta que la señal ha sido casi totalmente irradiada. Debido a que la onda reflejada encaja perfectamente con la onda de ida, no importa

cuántas veces oscilen, siempre estarán en la misma fase; es decir, no habrá diferencia de tiempo entre ellas. La varilla entonces adopta un nuevo nombre; el de *antena emisora*.

Una antena no tiene necesariamente que ser una varilla, puede tener otras características que influirán sobre la onda final emitida. Particularmente la longitud de la antena y la forma geométrica son de importancia, además de la relación entre la longitud de onda (λ) y la frecuencia (f) que establece que su producto es igual a la velocidad de propagación (v) como lo muestra la Figura 2.7.

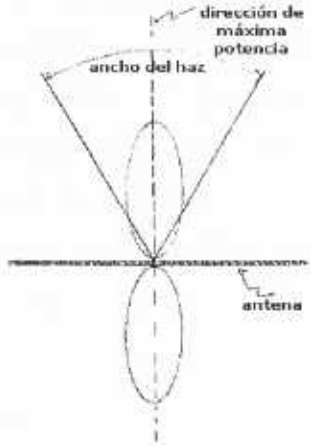

Figura 2.6. Dirección de la señal. http://omega.ilce.edu.mx:3000/sites/ciencia/volumen3/ciencia3/112/htm/sec_19.htm

Si sabemos además que la velocidad corresponde a la velocidad de la luz, podemos encontrar entonces que para determinada frecuencia, su longitud de onda se obtendrá por la fórmula mostrada en la Figura 2.8.

$$v = \lambda f$$

Figura 2.7. Velocidad de Propagación. Elaboración Propia.

$$\lambda = \frac{v}{f}$$

Figura 2.8. Longitud de Onda. Elaboración Propia.

Por otro lado, la potencia de la onda emitida es dependiente tanto de la longitud de onda, como de la longitud de la antena. Intuitivamente, parecería razonable sugerir que la mejor manera de transmitir es a través de una antena que cubra la completa longitud de onda en un ciclo, pero si decidiéramos que fuera así, una antena construida para una frecuencia de –por ejemplo- 54 MHz, necesitaría varilla de 2,7 metros. Es por ello que se utiliza, sobre todo para trabajos amateur, una antena de media onda. Incidentalmente, la potencia de emisión es máxima en esta situación, resultado del fenómeno de resonancia.[4]

Adicionalmente, una antena no emite la misma potencia en todas las direcciones; a lo largo de la antena (Figura 2.6) no hay de hecho emisión. No es así en la dirección perpendicular a la varilla donde de hecho se alcanza la potencia máxima; más aún, alrededor de la dirección perpendicular se forma un cono de transmisión; dentro de él, la emisión es apreciable e inteligible mientras que fuera, prácticamente no hay radiación. Este es el concepto básico de una *antena direccional*; la anchura del cono es la

[4] Más información en http://www.dxzone.com/catalog/Antennas/Theory/

medida del ancho del haz que se emite y es directamente proporcional a la frecuencia de la onda. Mientras mayor sea la frecuencia, mayor será el ancho del haz.

Una varilla a la que le es inducida corriente eléctrica por razón de una onda electromagnética se dice que tiene el nombre de *antena receptora*. Teniendo en cuenta que la frecuencia de esa corriente es la misma de la onda electromagnética y que las propiedades de esta nueva antena son las mismas de la antena emisora, se empiezan a tener los primeros indicios de comunicación inalámbrica. Vale destacar que, análogamente, una antena receptora absorbe la máxima potencia cuando su longitud es igual a la mitad de la longitud de onda de la onda incidente.

2.1.7 Dispositivos de Radiotransmisión

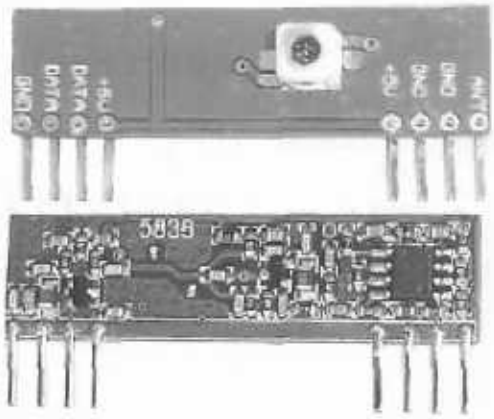

Figura 2.9. Dispositivo de Recepción STR-433.
http://www.sunrom.com/media/datasheet/STR-433.pdf

2.1.7.1 Sunrom STR-433

2.1.7.1.1 Visión General

El módulo de recepción compacto de radio frecuencia que lleva el nombre de STR-433 está sugerido para aplicaciones de telemetría y de control remoto. Además se sugiere también para aplicaciones donde el costo es un factor primario en la implementación, pero donde además se busca independencia de otros componentes. El módulo no necesita ninguno otro adicional, exceptuando de la antena y tiene propiedades de super-regeneración (la señal electrónica es amplificada por el mismo componente activo) que le permiten niveles excepcionales de sensibilidad.

La Tabla 2.3 resume las especificaciones de operación del dispositivo.

Parámetro	Símbolo	Mínimo	Típico	Máximo	Unidad
Voltaje de Operación	Vcc	4.5	5.0	5.5	VDC
Corriente de Operación	Icc	-	3.5	4.5	mA
Ancho de Banda de Recepción	BW rx	-	1.0	-	MHz
Frecuencia de Centro	Fc	-	433.92	-	MHz
Sensitividad	-	-	-105	-	dBm
Tasa Máxima de Datos	-	300	1k	3k	Kbits/s
Tiempo de Encendido	-	-	25	-	Ms
Temperatura de Operación	Top	-10	-	+60	°C

Tabla 2.3. Propiedades del STR-433
http://www.sunrom.com/media/datasheet/STR-433.pdf

2.1.7.1.2 Distribución de Pines

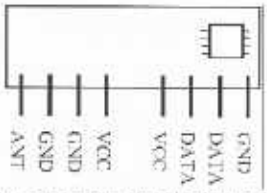

Figura 2.10. Distribución de Pines. STR-433.
http://www.sunrom.com/media/datasheet/STR-433.pdf

Nombre de PIN	Descripción
ANT	Entrada de Antena.
GND	Tierra del Receptor.
VCC(5V)	Están conectados eléctricamente y proveen voltaje de operación para el receptor. VCC puede ser aplicado a cualquiera de los 2 o ambos.
DATA	Salida de datos digital. Es capaz de conducir una carga TTL o CMOS. Es una salida compatible con CMOS.

Tabla 2.4. Descripción de Pines. STR-433.
http://www.sunrom.com/media/datasheet/STR-433.pdf

2.1.7.1.3 Dibujo Mecánico

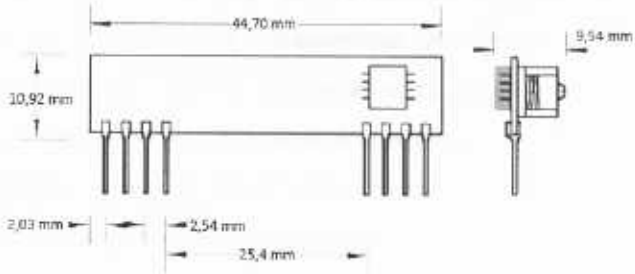

Figura 2.11. Dibujo Mecánico STR-433.
http://www.sunrom.com/media/datasheet/STR-433.pdf

2.1.7.1.4 Operación

2.1.7.1.4.1 Detección AM Super-Regenerativa

El STR-433 utiliza un detector AM super-regenerativo para desmodular la onda AM portadora. Un detector super-regenerativo es un estado de ganancia con retroalimentación mayor a la unidad. Esto resulta en oscilación que, con una constante de tiempo RC incluida en la etapa de ganancia, sería disminuida en el tiempo proporcional a esa constante. Eventualmente la oscilación se detiene y la carga actual de la etapa de ganancia disminuye, cargando al circuito RC y aumentando la ganancia, hasta que eventualmente la oscilación empieza de nuevo. De esta manera, la oscilación de la etapa de ganancia es prendida y apagada a una tasa definida por la constante RC. Esta tasa está elegida para ser super-audible (mayor a 20 KHz) pero mucho menor que la tasa principal de oscilación. La detección es lograda al medir el estado de ganancia actual del emisor. Cualquier señal de entrada RF a la frecuencia de la oscilación principal ayudará a que esta se reinicie. Si la amplitud de la radiofrecuencia de entrada disminuye, la oscilación principal estará prendida por mayor tiempo y la corriente del emisor será mayor. De esta manera, se puede detectar la señal banda-base al aplicar un filtro pasa bajo a la corriente del emisor. La corriente promedio del emisor no es muy linear como una función del nivel de la radiofrecuencia. Despliega una respuesta $\frac{1}{ln}$ debido a la naturaleza de crecimiento exponencial del disparo de la oscilación. La pendiente inclinada de un logaritmo cercano a cero resulta en alta sensibilidad para pequeñas señales de entrada.

2.1.7.1.4.2 Data Slicer

Para definir un Data Slicer es necesario primero recordar la estructura básica de una comunicación binaria y la codificación que puede tener. Generalmente, la información se dice está expresada, almacenada y transmitida de manera analógica o digital; cuando se habla de forma digital, la información es vista en un estado binario de abierto o cerrado, cero o uno, más o menos, prendido o apagados. Una señal digital puede estar codificada de distintas maneras (Manchester, Unipolar, NRZ, entre otras) pero cualquiera sea la forma, siempre hay una variación de voltaje para el tipo de valor que se esté leyendo. La señal digital siempre tendrá valores discretos y cuantificables, y manejará niveles de tensión eléctrica altos y bajos. Un Data Slicer es un detector de transiciones y una transición es el proceso de pasar entre esos niveles de tensión.

En el STR-433, el Data Slicer convierte la señal analógica banda base del detector super-regenerativo a una salida compatible CMOS/TTL. Debido a que el Data Slicer está vinculado al audio de salida mediante corriente alterna, hay una tasa mínima de información y a la vez se limitan las longitudes máximas y mínimas del pulso. Típicamente, la información es codificada en el lado de transmisión usando *Modulación de Anchura de* Pulso (PWM) o NRZ (*non-return-to zero*). La fuente más común de datos NRZ es de un UART[5] embebido en un microcontrolador; efectivamente la mayoría de las aplicaciones que utilizan codificación NRZ involucran microcontroladores y podemos verlas en circuitos integrados de control remoto, por ejemplo.

La información, cuando es enviada, se hace como una onda cuadrada de tasa constante. El ciclo de trabajo de esa onda cuadrada será generalmente o un 33% (para cero) o un 66%(para uno). En el STR-433, el Data Slicer está optimizado para utilizar información codificada PWN, aunque es capaz de trabaja con NRZ si ciertas reglas de codificación son seguidas.

2.1.7.1.4.3 Suministro de Poder

El STR-433 está diseñado para operar de una fuente de poder de 5V. Resulta sumamente importante que esta fuente sea silenciosa para disminuir el ruido radioeléctrico que genera.

Finalmente, el STR-433 está diseñado para operaciones de trabajo continuo. Es importante tener en consideración que desde el momento en que se induce energía en el receptor, puede tardar hasta 750 ms para que la información de salida (a través del pin *Data*) se haga válida.

2.1.7.1.4.4 Antena

El STR-433 está diseñado para manejar una gran gama de tipos de antenas, incluidas aquellas impresas directamente sobre una tabla de circuito y las simples de núcleo único de aproximadamente 17 cm para 120 metros de alcance, sabiendo que el desempeño de ellas varía. Cada vez que la línea sea más larga que ¼ de la longitud de onda de la frecuencia que carga, la antena debería ser una microtira[6] de 50 Ω.

---

[5] UART: Acrónimo para *Universal Asynchronous Receiver Transmitter*.

[6] Microtira: línea de transmisión donde se coloca sobre una cara de un plano dieléctrico una línea conductora, mientras del otro lado hay una conexión a tierra.

2.1.7.2 Sunrom STT-433

2.1.7.2.1 Visión General

La contrapartida del STR-433 es el STT-433. El diseño de bajo costo y empaquetado de tipo SIP[7] le permiten al STT-433 ser implementado en aplicaciones de alto volumen. Opera por fuentes entre 1,5V y 12V, lo que lo hace capaz de trabajar en aplicaciones energizadas por batería. El transmisor utiliza un oscilador estabilizado por *Onda Acústica de Superficie* (SAW) que asegura control preciso de frecuencias para el mejor desempeño de rango. La Tabla 2.5 resume las especificaciones del dispositivo.

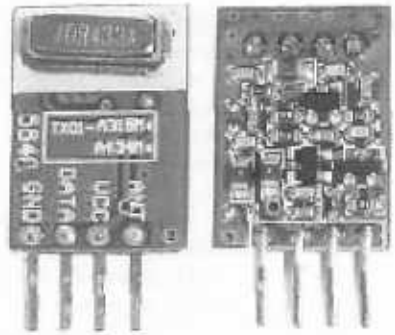

Figura 2.12. Dispositivo de Recepción. STT-433.
http://www.sunrom.com/media/datasheet/STT-433.pdf

Parámetro	Símbolo	Mínimo	Típico	Máximo	Unidad
Voltaje de Operación	Vcc	1.5	3.0	12	Voltios DC
Corriente de Operación para DATA = VCC	Icc	-	11 mA a 3V 59 mA a 5V	-	mA
Corriente de Operación para DATA = GND	Icc	-	100	-	µA
Precisión de Frecuencia	TOL Fc	-75	0	+75	KHz
Frecuencia Central	Fc	-	433	-	MHz
Poder de Salida de RF	-	-	4 dBM a 3V (2 mW) 16 dBM a 5V(39 mW)	-	dBm/ mW
Tasa de Transmisión	-	200	1K	3K	bps
Temperatura	-	-20	-	+60	°C
Retraso de encendido	-	-	20	-	ms
Alcance	-	-	30	120.47	m

Tabla 2.5. Propiedades del STT-433
http://www.sunrom.com/media/datasheet/STT-433.pdf

[7] SIP: Acrónimo para *System in Package*

2.1.7.2.2 Distribución de Pines

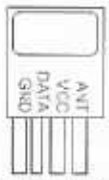

Figura 2.13. Distribución de Pines. STT-433. http://www.sunrom.com/media/datasheet/STT-433.pdf

Nombre de PIN	Descripción
ANT	Antena de Salida de 50 Ω. La impedancia del puerto de la antena afecta las emisiones armónicas y el poder de salida. Un filtro pasa bajo LC podría ser necesitado para filtrar las emisiones armónicas. Una antena puede ser simple de núcleo único de 17 cm o estar impresa en la tabla del circuito.
GND	Tierra del Transmisor.
VCC(5V)	Voltaje de operación del transmisor.
DATA	Salida de datos digital. Es compatible con CMOS y debería ser manejada con niveles de entrada CMOS.

Tabla 2.6. Descripción de Pines. STT-433. http://www.sunrom.com/media/datasheet/STT-433.pdf

2.1.7.2.3 Dibujo Mecánico.

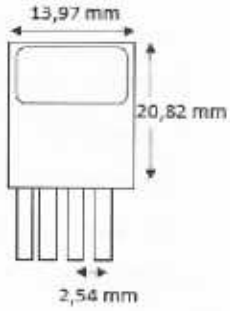

Figura 2.14. Dibujo Mecánico STT-433. http://www.sunrom.com/media/datasheet/STT-433.pdf

2.1.7.2.4 Operación

2.1.7.2.4.1 Teoría

La modulación OOK (*On Off Keying*) es una forma binaria de modulación por amplitud (AM). Cuando un cero (0) lógico está siendo enviado, el transmisor está de hecho apagado, suprimiendo por completo la onda portadora. En este estado, la corriente en él es menor a 1 mA. Cuando, en cambio, se está enviando un uno (1) lógico, el transmisor está totalmente encendido, así como la onda portadora. En este estado, el consumo de corriente en el módulo está en su tope (11 mA con 3V) El método de modulación OOK es usualmente el elegido cuando el consumo de poder y el costo son los factores primarios. Debido a que los transmisores por OOK no absorben poder cuando transmiten un cero (0), muestran significativamente un mejor consumo de poder en comparación con transmisores FSK[8]. La tasa de datos está limitada por el tiempo de

[8] FSK: Acrónimo para *Frequency-Shift Keying*.

prendido del oscilador, de aqui que este tiempo determina la tasa máxima de transmisión que puede alcanzar el transmisor.

2.1.7.2.4.2 Tasa de Transmisión

El tiempo de inicio del oscilador está en el orden de los 40 µs. Esto limita la tasa máxima de transmisión a 4.8 Kbit/s.

2.1.7.2.4.3 Oscilador Estabilizado por Onda Acústica de Superficie (SAW)

El transmisor es básicamente un oscilador LC de resistencia negativa cuya frecuencia central está severamente controlada por un resonador SAW. Los resonadores SAW son dispositivos de frecuencia fundamental que resuenan a una frecuencia mayor a la de los cristales.

2.1.8 Puerto Paralelo

El puerto paralelo fue la idea original de IBM para conectar la computadora a una impresora. Particularmente, querían las PCs fueran compatibles con las impresoras ofrecidas por Centronics, pero querían hacerlo sin que se usara la misma interfaz de Centronics (conector de 36 pines). Para ello crearon un cable especial que uniera un conector de 25 pines (DB-25) con un conector Centronics. Extrañamente, otros fabricantes adoptaron el mismo conector de Centronics para sus productos, de manera que el cable híbrido y el mecanismo de conexión de IBM se convirtieron en un estándar de facto para la industria. Hoy, se ha visto reemplazado por el puerto USB y está considerado como obsoleto para utilizarse en su función original. Sin embargo, es todavía popular entre estudiosos de la electrónica; a través de él se pueden controlar toda clase de circuitos para la automatización de procesos, adquisición de datos y demás actividades que demanden precisión.

Figura 2.15.Puerto Paralelo. http://www.machinetoolhelp.com/_images/rs232-DB25-DB9.jpg.

Todavía es posible encontrarlo en una computadora como un conector de 25 pines hembra teniendo cuidado de no confundirse con el conector serial (DB-25, macho) totalmente incompatible.

El puerto paralelo hoy fue estandarizado por la IEEE con el estándar 1284, hecho público en 1994, donde se establecen varios modos de trabajo además de la distribución de pines:

- Modo de Compatibilidad.
- *Modo Nibble.*

- Modo Byte.
- Modo EPP[9].
- Modo ECP[10].

La finalidad era diseñar nuevos manejadores que fueran compatibles entre si y a la misma vez con el puerto paralelo estándar. Los 3 primeros modos utilizan el hardware original, mientras los otros 2 requieren de hardware adicional que les permita correr a mayor velocidad manteniendo la compatibilidad original. El modo compatibilidad establece que solo se pueden enviar datos en una sola dirección a una velocidad tipica de 50KBps (tope 150+ KBps). Para recibir información, se debe hacer a través de los modos Nibble(4 bits de entrada) o Byte(8 bits de entrada).

Los pines por otro lado están distribuidos como los mostrados en la Tabla 2.7.

Número de Pin	Señal SPP	Dirección (E/S)	Registro
1	nStrobe	E/S	Control
2	Data 0	S	Data
3	Data 1	S	Data
4	Data 2	S	Data
5	Data 3	S	Data
6	Data 4	S	Data
7	Data 5	S	Data
8	Data 6	S	Data
9	Data 7	S	Data
10	nAck	E	Estado
11	Ocupado	E	Estado
12	Papel Acabado	E	Estado
13	Seleccionar	E	Estado
14	nAuto	E/S	Control
15	nError	E	Estado
16	nInicializar	E/S	Control
17	nSeleccionar Entrada	E/S	Control
18-25	Tierra	Tierra	-

Tabla 2.7. Descripción de Pines Puerto Paralelo.
http://www.beyondlogic.org/spp/parallel.htm

La letra "n" en la tabla anterior se utiliza para denotar que la señal es una baja activa. Por ejemplo, si ha ocurrido un error entonces el voltaje en la linea nError es bajo. Por el contrario, siempre hay un valor alto por esta linea, si la impresora está funcionando correctamente.

Los niveles de salida del puerto paralelo son normalmente niveles lógicos TTL. Se puede notar que para fines no relacionados con una impresora, el manejo de la información puede hacerse a través de los pines 2-9. 8 pines para manejar 8 bits.

[9] EPP: Acrónimo para *Enhanced Parallel Port*.
[10] ECP: Acrónimo para *Extended Capabilities Mode*.

2.2 SOFTWARE

2.2.1 IC-Prog

IC-Prog es un pequeño paquete que permite la programación de cualquier tipo de circuito serial integrado programable. Provee una interfaz de fácil uso y asi como también permite ver el código ensamblador del programa que se está por cargar en el PIC. Dispone en su núcleo de cinco espacios de memoria (*buffers*) para poder tener hasta cinco programas simultáneos. Dentro de una única ventana reúne el contenido a ser cargado, contenido de la memoria EEPROM y bits de configuración. Hay actualizaciones periódicas con funciones nuevas y problemas resueltos. Funciona bajo Windows 95 como Windows XP asi como en versiones intermedias.

2.2.1.1 Mecanismos de Seguridad

2.2.1.1.1 Encriptación Asimétrica.

El cifrado de clave pública (también conocido como *encriptación asimétrica*) es un esquema de encriptación basado en la premisa de que las claves para encriptar y desencriptar el texto, al contrario de cuando la encriptación es simétrica, no tienen que ser las mismas. El concepto fue propuesto por Whitfield Diffie y Martin Hellman en 1976 en la universidad de Stanford y se basa en un par de claves, una pública y una privada, donde la primera es conocida y la otra mantenida en secreto y donde un mensaje encriptado con la clave pública solo puede ser desencriptado con la privada. Además, existe una propiedad de autenticación; un mensaje encriptado con la clave privada puede ser verificado con la clave pública del emisor, comprobando que el mensaje es efectivamente original de quien dice ser el emisor (*firmas digitales*).

La esencia del algoritmo está en que, en una red de computadores, cada usuario perteneciente a ella emite una clave para compartir con otros y una clave para sí; todos los usuarios tienen acceso a un catálogo de claves públicas y todos tienen una única clave privada. Si se desea enviar un mensaje cifrado solo es necesario acudir al repositorio de claves públicas y utilizar la correspondiente al receptor para la encriptación. Hoy se encuentran entre los más populares, los algoritmos RSA, ElGamal, DSA, y los más recientes ECC (Criptografía por Curva Elíptica) cuya confianza todavía no alcanza a los anteriores.

2.2.1.1.1.1 RSA

RSA (*Rivest, Shamir, Adleman*) es un algoritmo descubierto en 1978 que significó el primer gran avance en el campo de la criptografía de clave pública. Se basa en el problema matemático de la factorización de un número entero en el producto de números primos y al que se refiere como el Problema RSA, cuyas raíces matemáticas van fuera del ámbito de este estudio. El algoritmo fue patentado por el Instituto Tecnológico de Massachusetts y aunque expiró el 21 de septiembre de 2000, RSA Security (hoy llamada RSA, *The Security Division of EMC*) lo liberó para el dominio público el 6 de septiembre de ese año.

Hoy, la compañía maneja un grupo de estándares para el uso de él llamados PKCS. En estos se establece, entre otros, el formato del algoritmo y esquemas de relleno para él. El esquema de relleno previene de una serie de ataques y clásicamente consiste en adjuntar caracteres al mensaje para poder encajar exactamente dentro del bloque de encriptado. Para RSA, la esencia del esquema de relleno está en hacer del algoritmo semánticamente seguro; esto es, dado 2 archivos de texto plano *similares*, la encriptación de ellos usando una misma clave pública resulta en textos indistinguibles entre ellos, inclusive si el atacante ha elegido los textos originales[11]. La encriptación tradicional RSA tiene naturaleza determinística (no hay componente pseudoaleatorio, el mismo texto cifrado es obtenido para varias ejecuciones del algoritmo al mismo mensaje plano con la misma clave pública); con un esquema de relleno, se transforma a uno de naturaleza probabilística (diferentes resultados para un mismo mensaje y misma clave, el mensaje será descifrado con cierta probabilidad).

RSA es considerado seguro y muy robusto después de 30 años desde su descubrimiento, aunque para ello se recomiendan claves mayores a 1024 bits y se cree suficiente para la mayoría de las aplicaciones. Esto mismo describe la desventaja inherente del uso de RSA; un algoritmo por clave pública en general es mucho más lento que uno simétrico.

2.2.1.1.2 Funciones Criptográficas Resumen (Hash)

Una función resumen (función *hash*, compendio de mensaje) es un método reproducible de transformación de data que a partir de un dominio típicamente grande, se obtiene un rango más pequeño. Matemáticamente, se define como la proyección de

[11] Conocido como Ataque de Texto Plano Elegido. *Chosen Plaintext Attack.*

conjunto con un número elevado de elementos (incluso infinitos), sobre un conjunto de tamaño fijo y mucho más pequeño que el anterior.

Particularmente, cuando se refiere a una función criptográfica se parte de que la función debe cumplir con ciertos principios.

- Dado un texto base, es fácil y rápido para una computadora calcular su número resumen[12].
- Para una misma función, todos los números resumen tienen el mismo tamaño independientemente del que sea el texto base.
- Es imposible reconstruir el texto base a partir del número resumen (Función en un solo sentido).
- Es imposible que dos textos base tengan el mismo número resumen (Resistente a colisiones).

Además, una función resumen debe ser parte del dominio público. Hoy son usadas con importancia en la verificación de integridad de archivos y la firma digital de mensajes y son de uso común particularmente los algoritmos MD5 y los pertenecientes a la familia SHA.

2.2.1.1.2.1 SHA-1[13]

La familia SHA de algoritmos, son 5 funciones hash diseñadas por la Agencia de Seguridad Nacional (*NSA*) de Estados Unidos y el Instituto Nacional de Estándares y Tecnología (*NIST*). Las especificaciones originales fueron publicadas en 1993 y hoy es recordado como la versión SHA-0, pero fueron rápidamente reemplazadas por una nueva revisión que se conoce hoy como SHA-1, liberada al dominio público en el RFC 3174. Actualmente, es extensivamente utilizado en mayor parte contra MD5 (aplicaciones como SSH, IPSec) debido a numerosos estudios de posibilidad de colisiones (archivos con mismo número resumen) encontradas sobre este. SHA-1 también se ha visto comprometido. Se dice que se puede conseguir una colisión en 2^{63} operaciones, en comparación con las 2^{80} de un ataque por fuerza bruta. Aunque la limitación de este ataque es directamente dependiente del poder de cómputo, lo coloca a nivel más teórico que práctico; la preocupación está en que puede dar paso a métodos más eficientes y eventualmente hacer del algoritmo poco seguro. De aquí que ya existen

[12] Número resumen: resultado obtenido de la función resumen sobre el texto base.

[13] SHA: acrónimo para *Secure Hash Algorithm*.

versiones mejoradas del algoritmo, que maneja números resumen más grandes. SHA-1 devuelve un número resumen de 160 bits (igual que SHA-0). Las nuevas versiones SHA-256 y SHA-512 manejan salidas de 256 y 512 bits cada una.

2.2.2. Redes Privadas Virtuales

Una red privada virtual es una red de data privada que hace uso de la infraestructura pública de telecomunicaciones mientras mantiene la privacidad, mediante uso de protocolos de entunelamiento y procedimientos de seguridad. Históricamente, resultan del potencial de las líneas alquiladas y de su alto costo. Cuando aparecieron las primeras redes de datos públicas, muchas compañías quisieron trasladar su tráfico de datos a la red pública sin renunciar a la seguridad de las redes privadas.

En una VPN, el envío de datos ocurre a través de un enlace privado punto-a-punto emulado entre 2 enrutadores, entre 2 servidores o entre un cliente y un servidor. Para el último caso, la computadora remota (con software para clientes VPN instalado) trabaja de la siguiente manera: cuando se desea hacer una conexión, el software VPN para clientes se conecta al servidor VPN usando un protocolo de entunelamiento que incluye en sí otros protocolos de autenticación y encriptación. Una vez que la computadora ha sido autenticada correctamente, una conexión segura entre ella y el servidor será establecida pues toda la información que transite entre ellos será encriptada y desencriptada por la parte que lo requiera. De esta manera, el túnel de red establecido entre ellos, a pesar de estar sobre una red pública, es considerado suficientemente seguro para que todas las demás computadoras detrás del servidor confíen en la que se encuentra del otro extremo de él. De hecho, el servidor VPN le asigna a la computadora remota un IP del dominio interno del servidor, de modo que todas las demás computadoras puedan comunicarse con ella utilizando esa dirección IP en el momento que lo deseen. Es trabajo del servidor traducir las direcciones real y asignada, para permitir que esa computadora remota, se vea como una local.

Se puede dar cuenta de que la fortaleza de una VPN reside en el protocolo de entunelamiento que utilice. Particularmente aparece uno de importancia:

- PPTP: definido por el Forum PPTP (Microsoft, Ascend, 3Com, ECI Telematics y U.S. Robotics). Ha estado disponible en sistemas operativos Windows 98 y Windows NT 4.0 pero fue criticado al haberse hecho disponible al escrutinio público por debilidades en la definición de claves, autenticación y algoritmos de encriptación. Microsoft revisó subsecuentemente el protocolo y corrigió las

fallas y hoy se espera que su uso se vaya reduciendo mientras IPSec se adopta en mayor medida. Utiliza claves generadas por MS-CHAP (*Microsoft - Challenge Handshake Authentication Protocol*), MS-CHAPv2 o EAP-TLS (*Extensible Authentication Protocol- Transport Layer Security*), y solo se pueden utilizar junto al algoritmo MPPE (*Microsoft Point-to-Point Encryption*). PPTP es considerado tan seguro como sea la palabra clave usada por el cliente. El principio de una palabra clave segura está en la imposibilidad de que sea una palabra contenida en una lista típicamente obtenida de un diccionario. Un ataque por diccionario se basa en tratar de encontrar la palabra clave en la lista, en lugar de revisar todas las alternativas posibles (ataque por fuerza bruta). Se recomienda que una palabra cumpla con las siguientes características:

- Utilizar caracteres no alfabéticos como signos de puntuación o números sin usar caracteres de control.
- Utilizar palabras no contenidas en diccionarios, como acrónimos o uniones entre vocales y consonantes que resulten en sonidos pronunciables.
- No utilizar el nombre de usuario de ninguna manera, ni siquiera a la inversa o con las letras entremezcladas.
- No utilizar un nombre real o parte de alguno.
- No utilizar información personal como fecha de nacimiento, número telefónico, cédula de identidad, etc.
- No utilizar menos de 8 caracteres.
- No utilizar nombres de libros, canciones, grupos de música, o cualquier tipo que pueda ser reconocido.
- No utilizar palabras claves demasiado fáciles o demasiado difíciles. Una fácil puede adivinarse en el momento de introducción por el teclado y una difícil de igual forma, al tener que escribirse lentamente.

Adicionalmente, se dice que una clave es tan segura como uno lo permita. Un sistema seguro no puede hacer retaliación contra las fallas de seguridad de sus propios usuarios. Se recomienda no compartir por ningún motivo y por ningún medio (hablado o escrito) una clave de acceso.

2.2.3 Firewall (Cortafuegos)

Un firewall es cualquier sistema (por hardware o software) que separe una computadora o una subred del resto de la red para protegerla de intrusiones externas que puedan suponer una amenaza para la seguridad de ellas. Provee protección del exterior a la vez que proporciona acceso hacia él. A continuación se usarán los términos *cortafuegos* y *firewall* indistintamente para referirse a lo mismo. Existen varios principios de las que debe partir un cortafuegos:

- Todo el tráfico desde el interior hacia el exterior y viceversa debe atravesar el cortafuego.
- Solo se permite el paso al tráfico autorizado, definido por una política de seguridad local. Se pueden utilizar diferentes cortafuegos con diferentes configuraciones implementadas.
- El cortafuego es inmune a la penetración, lo que implica que hay una relación de confianza entre él y el sistema operativo sobre el que es desplegado.

Se dice que la zona detrás de un firewall es el *perímetro de seguridad*, y que está separada por él de la región externa llamada zona de riesgo.

Un cortafuegos define un punto único de resistencia que mantiene a los usuarios fuera de la red protegida, prohíbe la entrada o salida de servicios potencialmente vulnerables y proporciona protección frente a distintos tipos de ataque de suplantación de IP (*spoofing*) y de enrutamiento. Esto usualmente deriva en errores por la falsa sensación de seguridad que proporciona; se espera que él la proporcione completamente y se descuida la seguridad interna de la red. Pero de esta misma manera, un firewall proporciona un punto de ubicación de sucesos relacionados con seguridad, útiles para implementar auditorías y alarmas ante ataques.

Existen varios tipos de cortafuegos dependiendo de donde está ocurriendo la comunicación, donde se le está interceptando y el estado que está siendo analizado. En primer lugar existen los llamados *routers* de filtrado de paquetes; van sobre la capa de Red del modelo OSI y se encarga de aplicar un conjunto de reglas basadas en la información contenida en un paquete de red a cada paquete IP para decidir si retransmitirlos o descartarlos. Si el filtrado está ocurriendo a través de reglas para el tráfico TCP (utilizando un directorio de conexiones salientes) donde se permitirá solo el tráfico a paquetes que encajen con cierto perfil, se dice que está usándose un *firewall de inspección de estado*.

Cuando ocurre que se interceptan todos los paquetes a nivel de aplicación, se dice que se tiene implementada una *Pasarela de nivel de aplicación*. Se bloquean todos los paquetes que en su nivel TCP/IP no cumplan con las características que las aplicaciones vinculadas definen. Más aún, se puede impedir que haya conexiones directas TCP de extremo a extremo y se establezcan en su lugar conexiones TCP internas separadas de la exterior pero vinculadas a ella; de este ser el caso se habla de una *Pasarela a nivel de circuito*.

Finalmente, existen los *Firewalls personales* que están diseñados para usuarios finales y tiene el fin de proteger solo la computadora sobre el que está instalado. La crítica de ellos reside principalmente en que es un servicio adicional (en lugar de reducir la carga de servicios, la aumenta) que consume recursos del sistema y puede ser víctima de un ataque al igual que la máquina, teniendo en cuenta además, que la relación con el sistema operativo lo hace vulnerable a las mismas fallas de él.

2.2.4 Invocación Remota de Métodos (RMI).

Se le llama RMI a una interfaz de programación exclusiva de Java equivalente en objetos a lo que se conoce como Llamada de Procedimientos Remotos (RPC) donde clases ubicadas en diferentes computadores pueden instanciarse e interactuar en una red distribuida. Debido a que usualmente el cliente y el servidor pueden residir en diferentes computadoras o procesos, es necesario establecer una relación entre los dos. RMI utiliza un programa de registros basado en red para controlar los objetos distribuidos llamado *RMIRegistry*; el objeto del servidor hace los métodos disponibles al vincularlos a un nombre en este programa, de manera que se termina comportando como un punto de gestión central de la invocación remota de métodos, sin atacar al problema de la invocación actual. *RMIRegistry* es un repositorio de nombres.

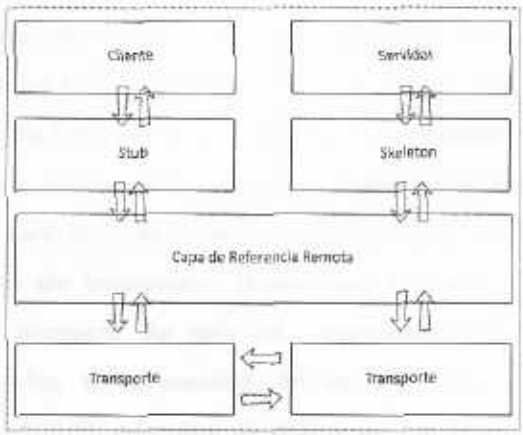

Figura 2.16. Interfaz RMI.
Elaboración Propia.

La interfaz funciona a través de 3 capas adicionales a los programas del cliente y del servidor. La primera de ellas, contiene a los objetos *Stub* y *Skeleton*. El primero está ubicado en la máquina del cliente y es utilizado por la máquina virtual de Java

para chequeo de tipos (Java es un lenguaje fuertemente tipado). Cuando un cliente invoca a un método del servidor, la petición pasa por el *Stub* y es luego direccionada al *Skeleton* en el servidor, que en respuesta llama al método apropiado en el objeto del Servidor. En otras palabras, el *Stub* actúa como un proxy para el *Skeleton* y el *Skeleton* es un proxy al método remoto necesitado. A partir de J2SE 5.0 (la versión actual es J2SE 6.0), se soporta la generación dinámica de clases *Stub* en el tiempo de ejecución, lo que hace innecesario el uso del compilador JRMI (*rmic*) para pregenerarlas para objetos remotos, aunque sigue siendo necesario para generar objetos *Stub* de objetos remotos que soportan clientes corriendo versiones anteriores. Esto significa además, que el archivo *Skeleton* no es generado a menos que se requiera (y debe hacerse expresamente) por la misma condición. Para información sobre Java, *véase Apéndice A.*

2.2.5 MySQL

Lanzado al mercado en 1995 (1998 para Windows) , MySQL es un manejador multiusuario de bases de datos relacional basado en SQL, componente de la compañía madre, MySQL AB, que hoy ofrece además de él, toda una serie de productos para servidores de bases de datos y herramientas de desarrollo. Originalmente, obtuvo apoyo como alternativa de código abierto a los sistemas propietarios de Oracle, IBM e Informix. Para aquella época pocos motores de bases de datos existían para pequeñas organizaciones o usuarios individuales, pero destacaba *Postgres;* sin embargo este requería de requerimientos similares a sus contrapartes comerciales y lo hacía sin la ventaja de poseer a SQL como el lenguaje de búsquedas. Usaba una variante del lenguaje QUEL llamado *PostQUEL*. Por un proyecto de tesis de doctorado, David Hughes en la Universidad Bond en Australia, en necesidad de un paquete que transformara peticiones SQL a lenguaje *PostQUEL* aparece el precursor de *MySQL*: *miniSQL* o *mSQL*.

El proyecto creció hasta un punto donde trabajar con *Postgres* (o cualquier sistema relacional grande) se volvió inadecuado. Solo se necesitaba soporte para pequeñas sentencias *Insert, Delete* y *Select;* pero el proyecto (de nombre Minerva) requería múltiples conexiones simultáneas a la base de datos. *Postgres* para soportar esto requería a su vez de múltiples instancias al servidor de la base de datos ejecutándose al mismo tiempo. En vista de estos problemas, Hughes reevaluó su decisión de usar *Postgres;* se dio cuenta de que solo tenía que agregar habilidades de almacenamiento y recuperación de datos y tendría al manejador que necesitaba.

Paralelamente, Michael Widenius trabajaba para la compañía sueca *TcX*, desarrollando aplicaciones web soportadas por un servidor interno de bases de datos llamado *UNIREG* que, desafortunadamente, era demasiado lento para la generación dinámica de páginas web. Widenius contacta a Hughes y, en una fusión entre ambos sistemas surge *MySQL 1.0* (aunque *mSQL* sigue estando disponible) sobre un nuevo modelo de negocio, el lanzamiento por Internet.

Actualmente *MySQL* es compatible virtualmente con todas las plataformas, incluidas Linux, Unix y Windows. Está disponible bajo 2 tipos distintos de licencia: GPL de código abierto y libre de cargo y a través de subscripción a la red *MySQL* para aplicaciones de negocio. Maneja totalmente multi-hilos (*hilos de kernel*) y provee interfaces de programación de aplicaciones (*APIs*) para numerosos lenguajes de programación, incluidos C, C++, Java, Perl, PHP y Python. Es utilizado en un amplio rango de aplicaciones de comercio electrónico, bases de datos web, *data warehousing* y aplicaciones distribuidas. Según MySQL AB, el producto tiene más de 6 millones de instalaciones activas en el mundo y su lista de clientes incluye Cisco, Google, NASA y Lufthansa entre otros. Soporta entre otras cosas, procedimientos almacenados, funciones, disparadores, cursores, vistas actualizables, soporte SSL, *selects* anidados, y sobre todo, cumplimiento de reglas *ACID*[14]. Para información sobre Manejadores de Bases de Datos, *véase Apéndice B*.

2.2.6 JDBC

Java Database Connectivity es un API que permite a programas de Java acceder a sistemas de manejo de bases de datos. Consiste de una serie de clases e interfaces escritas en Java a través de las cuales un programador puede escribir aplicaciones que conectan a bases de datos, envían sentencias SQL y procesan los resultados. Ofrece independencia pues a través de él, los usuarios pueden continuar el uso de sus manejadores de bases de datos y acceder la información en ellos fácilmente, inclusive si son distintos entre ellos. El potencial está en la posibilidad a través de él, de escribir una aplicación independiente de la plataforma y capaz de interactuar con varios manejadores.

El API consta de 2 partes, una para desarrolladores de aplicaciones y otra para desarrolladores de controladores de bajo nivel; la finalidad está en que cada manejador de bases de datos tenga un controlador compatible con el API, desarrollado

[14] ACID: acrónimo para *Atomicity, Consistency, Isolation, Durability.*

independiente pero certificado por Sun Microsystems. En su mayoría se consiguen de forma gratuita, van del lado de la aplicación cliente y son necesarios para convertir peticiones del programa en sentencias entendibles por el manejador.

3.1 XP: eXtreme Programming

El mercado actual cuenta con una variada gama de metodologías para el desarrollo de software entre las que se encuentran las denominadas *Metodologías Ligeras*. XP, parte de este subgrupo, se ha establecido como una de las más útiles para el desarrollo de software. Se le sugiere para proyectos de muy alto riesgo y con requerimientos dinámicos donde los clientes no saben lo que quieren y no se tiene un conocimiento claro de la tecnología a implementar. Esta metodología tiene la particularidad de obtener un código totalmente funcional en tiempos más cortos; establece que la comunicación del equipo de trabajo tiene que ser fluida y totalmente colaborativa, lo que garantiza reducción de costos, fecha de culminación del proyecto en los tiempos requeridos y, lo que es más importante, calidad de software.

El escenario propuesto por XP es el trabajo en equipo, donde la codificación se lleva a cabo por dos personas; mientras la persona que está frente al teclado asume un rol técnico y es encargado de la codificación *per se*, la segunda asume un rol estratégico, verificando el código y llenando huecos de lógica que pueden ser fácilmente ignorados por el codificador. Esta forma de trabajo garantiza código depurado de errores y reducción del tiempo de entrega. Además XP involucra en todas las etapas del desarrollo de la aplicación al cliente; este colabora con el equipo verificando que cada paso efectuado en la codificación de la aplicación siga fiel a la lógica del negocio, añadiendo cambios y detectando fallas antes de la entrega final del producto. XP se enfoca también en mantener cada iteración en un plano simple y claro desde el punto de vista del diseño, mantenimiento y codificación; si se cumple este esquema se asume que no se desperdiciará trabajo a mitad del desarrollo. Es sabido que muchas aplicaciones se pierden porque crecen en complejidad y se hacen imposibles de continuar al desviarse de los objetivos principales.

XP plantea cuatro fases de desarrollo: Planificación, diseño, desarrollo y pruebas. Cada una se describe a continuación.

3.1.1 Planificación

La planificación es llevada en primera parte por los clientes. Ellos están encargados de definir las reglas del negocio en lo que define XP como "Historias de Usuario". Estas tienen el mismo propósito que los ya familiares casos de uso, con la diferencia de que son desarrolladas por los clientes desde su perspectiva del negocio, manejando un lenguaje coloquial y estimando el riesgo de cada historia y el tiempo

esperado para cada implementación. Los desarrolladores en esta etapa se encargan de establecer tiempos reales actualizando el plan de desarrollo en función del alcance de cada historia. Las tareas se desglosan de la siguiente manera:

Cliente

- Definir requerimientos de software.
- Definir prioridades de desarrollo entre los requerimientos
- Fechas de culminación para cada requerimiento.

Desarrolladores

- Estimar el tiempo de desarrollo de cada requerimiento.
- Informar al cliente sobre riesgos del desarrollo de cada requerimiento.
- Establecer un árbol de prioridades según cada requerimiento a desarrollar.

3.1.2 Diseño

El diseño se basa en la búsqueda de una estrategia para mantener un desarrollo simple a lo largo del proyecto, esto se refiere a crear el menor número posible de clases y métodos. Es importante elegir patrones de diseño que permitan esta simplicidad y que sean versátiles al momento de añadir nuevos componentes.

3.1.3 Desarrollo

El desarrollo en XP se basa en la programación en parejas; la seguridad está en cierta medida garantizada debido a la simultánea toma de decisiones que hay durante la elaboración del código. Resulta dificultoso que dos personas se salten tareas por descuido o negligencia; el código, mientras está siendo escrito por una persona, está siendo revisado por la otra. Las estadísticas indican que la programación en parejas es realmente más eficiente. Si bien se sacrifica un poco de velocidad al comienzo, luego se compensa con el ahorro de tiempo para depuración posterior, cosa que no ocurre con la programación individual, por ejemplo. Es necesario mencionar que el cliente está integrado en esta fase ya que se encarga de verificar en cada paso en el desarrollo se cumpla con los requerimientos originales.

3.1.4 Pruebas

Todo código colocado en producción debe ser probado exhaustivamente, corrigiendo cualquier falla o falta en los requerimientos originales de la lógica del

negocio. El propósito principal de esta fase es elaborar casos de prueba antes de avanzar a la próxima versión de la aplicación.

3.2 Justificación de la metodología

Para el presente trabajo de grado se eligió la metodología XP porque se muestra versátil a la hora de añadir cambios durante el desarrollo de la aplicación; tiene como característica principal el desarrollo inmediato a pesar de no tener un manejo total de la tecnología a utilizar en el proyecto (para este caso en particular de trabajo de grado, el hardware). Permite elaborar versiones consecutivas funcionales verificadas por el equipo de desarrollo y por el cliente (el tutor) que permite avanzar con un margen mínimo de errores hacia la siguiente iteración y reducir a su vez el tiempo de desarrollo.

3.3 Plan de Desarrollo

El desarrollo del presente trabajo de grado consta de 5 iteraciones (Tabla 3.1). Es importante destacar que antes de obtener cada versión la fase de pruebas era revisada exhaustivamente por el tutor el cual indicaba que cambios deberían de realizarse para la próxima iteración.

Fase XP	Descripción	Producto
Planificación	• Definición historias de usuario • Selección de cada historia de usuario para cada entrega. • Se estableció el número de entregas de versiones en 5.	• Historias de usuario. • Diagrama de Gantt de la aplicación en general *(Véase Apéndice I)*.
Diseño	• Diseño de diagramas de clase para cada entrega. • Diseño de diagramas de secuencias para cada entrega.	• Diagramas de clase para cada entrega. • Diagramas de secuencias para cada entrega.
Desarrollo	• Desarrollo de código en parejas.	• Código fuente de la aplicación.
Pruebas	• Desarrollo de pruebas unitarias utilizando JUnit, herramienta de java para ejecutar pruebas automatizadas.	• Aplicación depurada de errores.

Tabla 3.1. Fases de Desarrollo.
Elaboración Propia.

4.1 1ra Iteración. Diseño de Hardware y Software

En esta fase de definió en primera instancia el tipo de hardware y software que se utilizaría en la implementación del presente trabajo de grado. La decisión se basó en costos, tecnología disponible y facilidad de implementación.

La Figura 4.1 muestra un diagrama de cómo debe ser el sistema final.

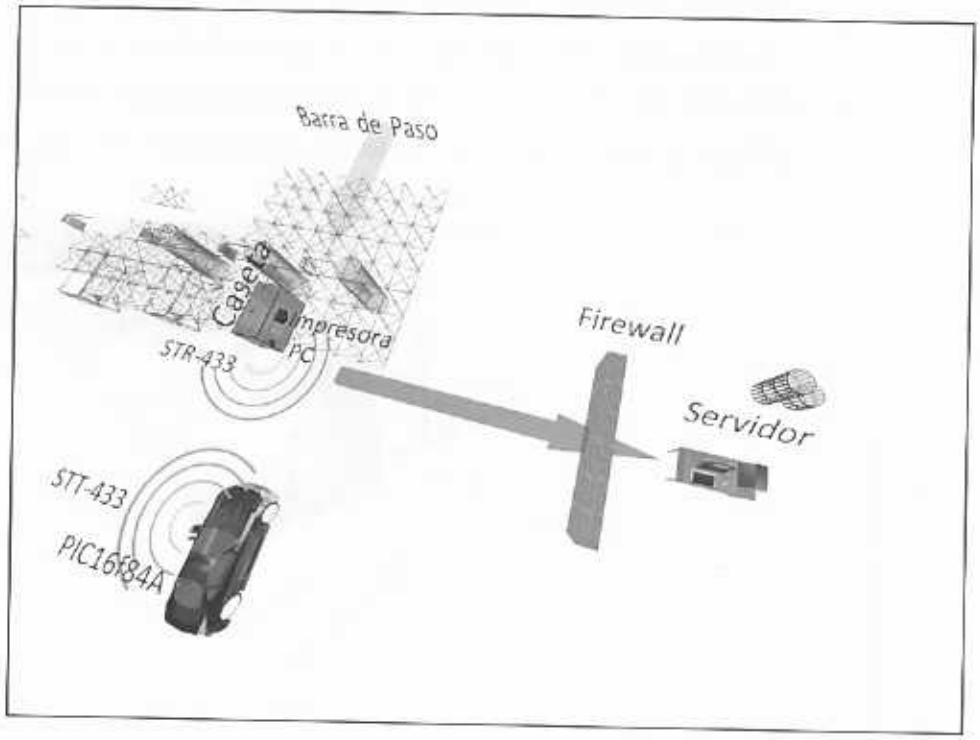

Figura 4.1. Diagrama del Sistema.
Elaboración Propia

4.1.1 Planificación

El ambiente de desarrollo se planteó para simular las condiciones reales de una caseta de peaje; fue necesario desplazarse hasta el ubicado en la autopista regional del centro en Caracas y tomar las medidas físicas consideradas importantes (Figura 4.2); ellas definen el entorno donde sería implementado el sistema y se deduce de ellas que el alcance de la señal irradiada debe ser de 2 metros. Las medidas del peaje fueron tomadas debido a que ellas resultan como punto de inicio para la toma de decisiones respecto a longitud de las antenas de recepción y emisión. Las distancias entre la caseta y el vehículo son indispensables para la correcta transmisión de los datos y para ver cuáles son los límites de la comunicación. Además de ello, se estiman medidas intravehiculares al momento de entrada al peaje, importante esta última medida para consideraciones posteriores de evasión de solapamiento de señales. Después de ello, se

levantó la información referente a las bandas disponibles en el espectro radioeléctrico venezolano en el organismo gubernamental CONATEL (Comisión Nacional de Telecomunicaciones), información de suma importancia debido a las restricciones que existen con respecto al uso de ciertas bandas de comunicación.

Una vez obtenida la información anterior fue necesario establecer una reunión con el tutor del presente trabajo de grado para definir los tipos de transmisores a utilizar, tomando la decisión de usar los dispositivos STR-433 y STT-433 (transmisor y receptor respectivamente). Ellos trabajan a una frecuencia de 433 MHz. La Tabla 4.1 muestra un extracto del Cuadro Nacional de Atribución de Bandas de Frecuencias.

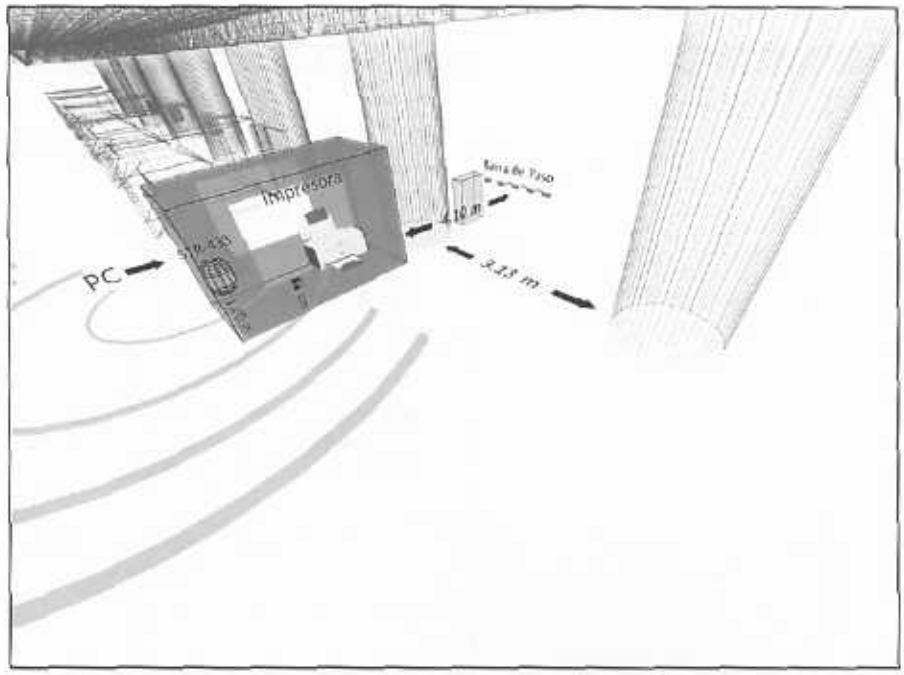

Figura 4.2. Diagrama con Medidas Físicas (Peaje Autopista Regional del Centro). Elaboración Propia.

El peaje mencionado cuenta con una infraestructura básica para el manejo tributario y el paso de vehículos, mas no alguna que pueda ser considerada de importancia para este trabajo especial de grado (aparte del hardware para el manejo de la barra de paso). Para este momento, se establece el tiempo de funcionamiento del peaje como constante; el flujo de vehículos a través de él transcurre ininterrumpidamente, lo que resuelve en diseñar posteriormente un sistema constantemente activo (en modo de escucha todo el tiempo) que, aunque reduce el tiempo de vida de los dispositivos electrónicos relacionados, no afecta el desempeño del tráfico.

La nota V2 establece que para la operación de estaciones de Radioaficionados

Banda	Atribución UIT Región 2	Atribución Venezuela	Nota
...			
420 – 430 MHz	FIJO MÓVIL salvo móvil aeronáutico Radiolocalización 5.269 5.270 5.271	FIJO MÓVIL salvo móvil aeronáutico	
430 – 432 MHz	RADIOLOCALIZACIÓN Aficionados 5.271 5.276 5.277 5.278 5.279	AFICIONADOS	V2
432 – 438 MHz	RADIOLOCALIZACIÓN Aficionados Exploración de la Tierra por satélite (activo) 5.279A 5.271 5.276 5.277 5.278 5.279 5.281 5.282		
438 – 440 MHz	RADIOLOCALIZACIÓN Aficionados 5.271 5.276 5.277 5.278 5.279		
...			

Tabla 4.1: Cuadro Nacional de Atribución de Bandas de Frecuencias. (Extracto)
Fuente: http://www.conatel.gob.ve/

y para el rango entre 430 y 440 MHz (UHF) con longitud de onda de 0.7 m, se pueden utilizar a los servicios de Aficionados y Aficionados por satélite, según lo referido en la *tabla 4.1*. Los servicios en letras mayúsculas (ejemplo: FIJO, RADIOLOCALIZACIÓN), se denominan *servicios primarios* mientras los que estén en minúsculas se denominan *servicios secundarios*. La diferencia entre ellos está en que los últimos, están sujetos a las siguientes reglas:

- *No deben causar interferencia perjudicial a las estaciones de un servicio primario a las que se les hayan asignado frecuencias con anterioridad o se les*

puedan asignar en el futuro;

- *No pueden reclamar protección contra interferencias perjudiciales causadas por estaciones de un servicio primario a las que se les hayan asignado frecuencias con anterioridad o se les puedan asignar en el futuro;*
- *Sólo tienen derecho a la protección contra interferencias perjudiciales causadas por estaciones del mismo servicio o de otros servicios secundarios a las que se les asignen frecuencias ulteriormente.*[1]

4.1.2 Diseño

4.1.2.1 Dispositivo de Transmisión

El diseño del transmisor consta de un PIC 16F84A, un transmisor STT-433, un cristal de cuarzo de 4mhz y dos condensadores de 22 pf, conectados de la manera dispuesta en la Figura 4.3.

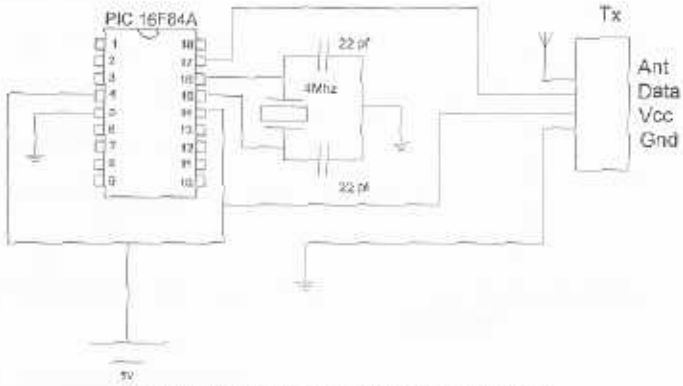

Figura 4.3. Circuito para Transmisión Inalámbrica.
Elaboración Propia.

Fue necesario utilizar 6 pines del PIC 16F84A, la conexión de dicho PIC con el transmisor STT-433 se describe en la Tabla 4.2.

No PIN (PIC 16F84A)	Conexión
Pin 4	Vcc.
Pin 5	Tierra.
Pin 14	Vcc.
Pin 15	Cristal de cuarzo – Condensadores.
Pin 16	Cristal de cuarzo – Condensadores.
Pin 17	Entrada de Data del Transmisor.

Tabla 4.2. Conexiones de Pines con Transmisor STT-433.
Elaboración Propia.

El cristal de cuarzo cumple con la función de suministrar al PIC 16F84A pulsos de reloj necesarios a una frecuencia determinada (4Mhz). Los condensadores cumplen la función de ofrecer voltaje continuo; suavizar la onda en caso de que el circuito tenga picos o bajas en el flujo de potencial. El PIC 16F84A tiene almacenado en su memoria la data referente al vehículo y está programado para que sea enviada por el PIN 17 (a través de impulsos compatibles TTL/CMOS) a su vez conectado al PIN de data del transmisor STT-433.

El PIC 16F84A fue programado (para el circuito para programarlo *véase Apéndice B*) con una rutina basada en el pseudocódigo mostrado en la Figura 4.5.

[1] Textual de *http://www.conatel.gob.ve/espectro.htm*

El módulo de transmisión será llevado por el vehículo. Por ello, necesita de alguna fuente que le proporcione los 5V para su funcionamiento. Debido a que, en general, los vehículos son alimentados por fuentes de 12V, se hizo necesario diseñar un circuito que regulara dicho voltaje y que lo redujera hasta 5V. Un modulo con esa función lo vemos en la Figura 4.6.

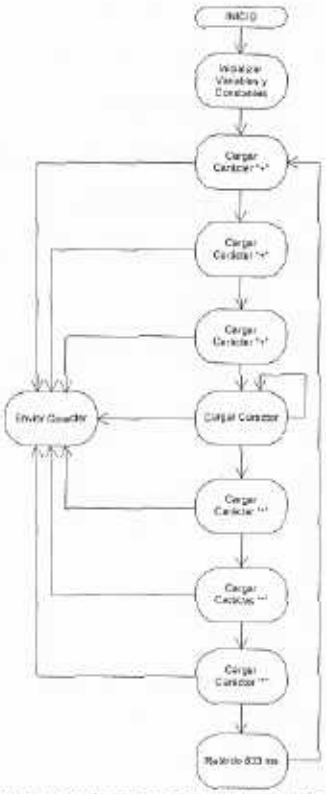

Figura 4.5. Pseudocódigo para Programación del PIC. Elaboración Propia.

Los vehículos tienen la particularidad de que cada vez que se aceleran, se produce en ellos una subida de potencial por la carga del alternador sobre la batería. Normalmente la diferencia de potencial entre el alternador y la batería del vehículo es de 14V cuando esta requiere de carga (12V si no es así). El fin de los condensadores es aplanar los picos y bajas de tensión que ocurran para ofrecer en todo momento una diferencia de potencial constante de 5V.

La Figura 2.8 muestra la fórmula necesaria para calcular la longitud de onda de una frecuencia determinada. Para las necesidades del sistema, se requiere un alcance máximo de 2 metros, lo que significa que cada carro solo tendrá ese alcance de irradiación de señal y deberá estar a esa distancia del receptor para que exista la comunicación. El cálculo que arroja esta fórmula se basa en el tamaño de la antena según la velocidad de la luz y la frecuencia utilizada para los componentes de transmisión. Según la Figura 2.8 queda dividir la velocidad de la luz (300.000 Km/seg) entre la frecuencia utilizada en los dispositivos de transmisión y recepción del presente trabajo de grado (433 MHz). Se obtiene que para la onda completa se necesita una antena de 0.6 metros que indica que ese es el tamaño de la antena para obtener la potencia máxima de irradiación para la onda completa. En el caso de los dispositivos de transmisión para el presente trabajo de grado, el alcance máximo con este tamaño de antena es de 120

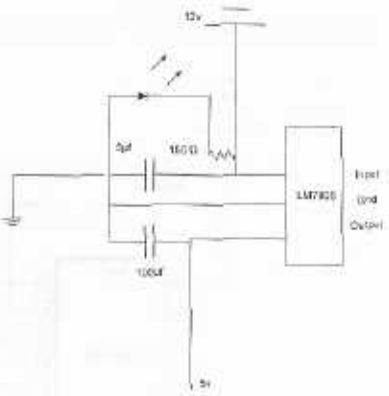

Figura 4.6. Regulador de Voltaje 12V a 5V. Elaboración Propia.

metros (*véase Tabla 2.5*) En realidad, solo son necesario dos metros de alcance, por lo que si la potencia máxima se logra con una antena de 60 centímetros y son 120 metros con este tamaño, para un alcance de dos metros se necesita una antena de 0,01 metros (1 centímetro). Este resultado se obtiene mediante una simple regla de tres.

Para las pruebas pertinentes a la transmisión de datos inalámbrica, el uso del vehículo en todo momento se convertía en una tarea engorrosa; se diseñó en su lugar, un circuito de regulación alternativo (Figura 4.7) que utiliza una batería de 9V (*batería PP3*) y del que se obtienen los 5V necesarios. El circuito anterior muestra un regulador LM7805; su función es reducir el potencial de 9v a 5v, la pila no tiene cambios abruptos de potencial por lo que no es necesario el uso de condensadores.

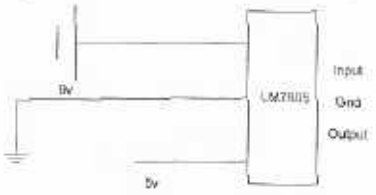

Figura 4.7. Regulador de Voltaje 9V a 5V. Elaboración Propia.

4.1.2.2 Dispositivo de Recepción

El módulo de recepción del presente trabajo de grado tiene la función de, además de recibir la señal enviada por el transmisor, transformar los valores de voltaje TTL 0 – 5 v, a voltajes RS232 +/-. Esto es posible gracias al convertidor MAX232. El módulo de recepción completo, consta de cuatro condensadores de 1 μF, un LED que sirve de testigo de funcionamiento del circuito y un puerto macho RS232. El diagrama completo se encuentra en la Figura 4.8.

A diferencia del módulo de transmisión que está ubicado en un vehículo, el módulo de recepción puede tener al alcance una toma de corriente de 110V. Éste se alimenta de 5V igualmente, de modo que resulta necesario construir un circuito transformador de corriente alterna a corriente continúa y reductor de voltaje. El diagrama de dicho diseño se muestra a continuación en la Figura

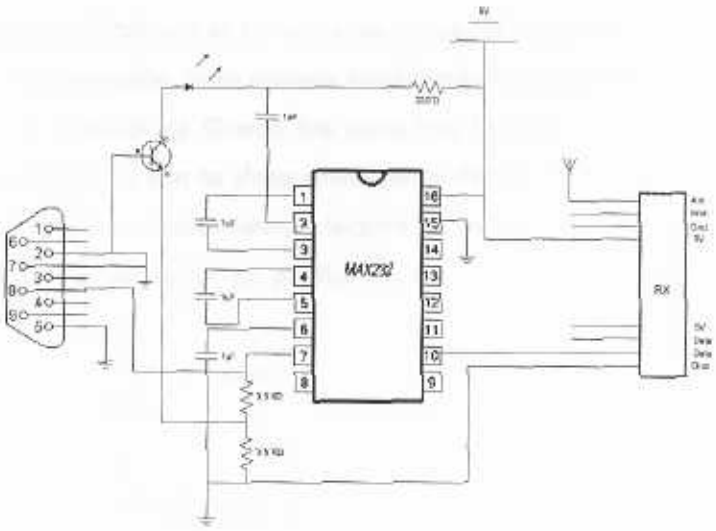

Figura 4.8. Circuito para Recepción Inalámbrica. Elaboración Propia.

4.9. Adicionalmente el módulo de recepción tiene la opción, al utilizar igualmente 5V, a través de los mismos medios dispuestos en las figuras 4.6 y 4.7.

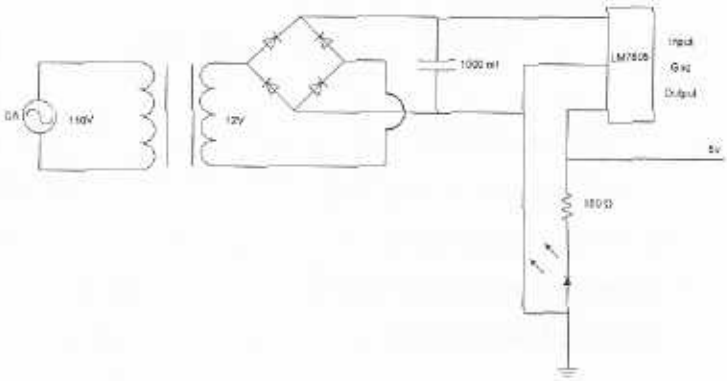

Figura 4.9. Regulador de Voltaje 110 V a 5V.
Elaboración Propia.

Para el diseño del software de recepción desde el puerto RS232 se planteó método en ella, se encargaría de estar escuchando por el puerto, de la captura de los datos que lleguen a él y del almacenamiento en buffer de ellos. Dicha clase se representa en el diagrama de clases de la Figura 4.10. La clase *DataIn* tiene un método cuya firma es *getSerialsFromArray(String[] array)*. Se encarga de capturar desde el puerto RS232 las ráfagas de bits que recibe el módulo receptor y almacenarlas en un campo de 100 Bytes. Esta longitud del arreglo es más que suficiente para almacenar varias versiones recibidas del serial de un vehículo.

DataIn
-serialArray : string
+getSerialsFromArray(String[] array)()

Figura 4.10. Diagrama de Clases para Lectura de interfaz RS-232.
Elaboración Propia.

4.1.3 Desarrollo

Una vez elaborado el diseño de los circuitos y el software, se procedió en primer lugar al ensamblado de los circuitos de transmisión, cuyo número establecido al principio del planteamiento del presente Trabajo Especial de Grado fue para tres (3) vehículos. Cada circuito contiene un PIC que se programa con la data procedente del serial real de tres vehículos distintos. Por motivos de fácil reconocimiento y lectura de la información desde el software se introdujo como parte del serial un prefijo de tres (3) asteriscos consecutivos ("***"), y un sufijo de tres (3) signos positivos ("+++"). Cada valor almacenado en el PIC queda de la

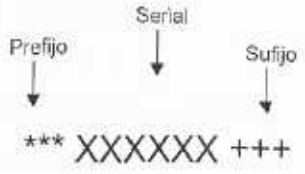

Figura 4.11. Estructura de la trama transmitida.
Elaboración propia.

forma mostrada en la Figura 4.11, si se toma como modelo un serial, por ejemplo, de seis letras "X" consecutivas.

Concluido el ensamblado de los circuitos se procedió a programar el software en Java. Al momento del desarrollar se optó por crear un arreglo de dos posiciones de longitud cien (100) Bytes en cada posición, longitud justificada anteriormente. El fin sería obtener un número considerable de seriales obtenidos desde el buffer, para al momento posterior de hacer la verificación, tener la certeza de que la data verificada es la correspondiente al vehículo en tránsito y que ella carece de errores.

La clase *DataIn* se basa en la captura de las ráfagas de bits provenientes del receptor en eventos, un evento, como su nombre lo indica, es un acontecimiento en pleno desarrollo; en este caso un evento es un byte de entrada en el buffer que proviene del puerto RS232, la aplicación siempre está en modo de oyente, atento a cualquier byte que provenga de dicho puerto.

4.1.4 Pruebas

Las pruebas realizadas para la parte de hardware se basaron, en primer lugar, en medir los voltajes de salida de los módulos de alimentación y que estos fuesen los establecidos en el diseño (5V). En el laboratorio, con el uso de 2 osciloscopios, se verificó la onda de entrada y salida de los módulos de transmisión; se verificaba que esta fuera la misma en ambos casos y que cualquier modificación que ocurriera en la oda de salida (velocidad, por ejemplo) se viera reflejada sobre la onda de entrada.

Para el software se elaboró una clase *JUnit* con dos métodos de prueba. El primer método (caso favorable en la ejecución de la aplicación) está encargado de pasar como parámetro ráfagas consecutivas de 7 bits al método *getSerialsFromArray* de la clase *DataIn*; el método setea una bandera cada vez que introduce un valor coherente en el arreglo de dos posiciones, este valor es devuelto al método de prueba y usado para la verificación de la ejecución de dicho método. Para el caso en que ocurra una falla en la obtención de la ráfaga de bits se elaboró el segundo método de prueba de la clase *JUnit*, que se encarga de pasar como parámetro caracteres basura (por ejemplo, carácter nulo para Java, '\u0000") y evaluar el comportamiento de la aplicación. En la ejecución de prueba se obtuvo el comportamiento esperado del método *getSerialsFromArray*: ignorar cada carácter basura pasado como parámetro.

4.2 2da Iteración. Diseño y Desarrollo de Clases de Conexión y Base de datos.

Una vez obtenida la ráfaga de bits enviada desde el módulo de transmisión, es necesario elaborar el sistema de comunicación entre la caseta del peaje y el servidor y establecer un mecanismo de comunicación entre el sistema de recepción y la base de datos que almacena la información concerniente a los vehiculos.

4.2.1 Planificación

El sistema de base de datos está basado en la información suministrada por personal del SETRA. La información sobre diseño e implementación de la base de datos de dicho instituto es de carácter clasificado. Para su estudio solo se pudo observar la aplicación que existe para los usuarios internos del organismo, los parámetros que recibe y los resultados que retorna. Ella se basa en la búsqueda de vehiculos por medio de dos filtros: *serial de carrocería* o *placa del vehículo*. De aquí se tomó la decisión de elaborar un diseño de base de datos relacional centralizada que constara de tres tablas usando la herramienta MySQL, respetando los estándares y buenas prácticas establecidas en el mercado actual. La razón fundamental de la decisión sobre la base de datos centralizada reside en la disminución de transacciones para agilizar la consulta del serial; así como los costos adicionales e innecesarios (recursos de cómputo) que implica el despliegue de una base de datos en la caseta.

Java ofrece mecanismos de implementación sencillos, en comparación con otros lenguajes de programación, al momento de conectar dos equipos remotos. Para el presente trabajo de grado se decidió adoptar la interfaz RMI (*Remote Method Invocation*) por sus ventajas en lo que a comunicación remota en arquitectura clientes/servidor se refiere y las facilidades de uso posteriores.

Debido a que se van a realizar las primeras conexiones entre dos computadoras, se hace la primera referencia al firewall de protección para el servidor. El firewall debe configurarse para permitir conexiones a través del puerto por defecto utilizado por el objeto a ser usado remotamente; es decir, *RMIRegistry* y el puerto 1099.

4.2.2 Diseño

El diseño de la base de datos consta de tres tablas. Dos tablas principales: *VEHICULO* y *PROPIETARIO* y una de vinculo llamada *PROPIETARIO_VEHÍCULO*. La primera define la estructura de la información referente al vehículo tal como su nombre lo indica mientras *PROPIETARIO* contiene información de personas que alguna vez hayan

poseído un vehículo. Para relacionar dichas tablas fue necesaria la tercera en la que se almacena cada cambio de dueño del vehículo a lo largo del tiempo. Es en esta tabla donde se encuentra el campo más crítico dentro del presente trabajo de grado: el estatus del vehículo. Dicho estatus indica si el vehículo está siendo solicitado por las autoridades o no. El diagrama entidad relación se muestra en la Figura 4.12. La búsqueda del vehículo siempre se ejecuta en base al dueño más reciente; es posible que exista más de un dueño de él a lo largo de su vida útil.

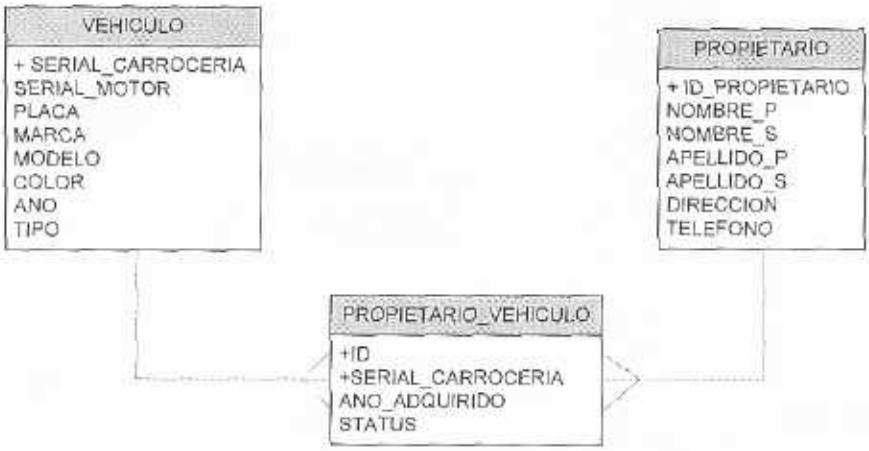

Figura 4.12. Modelo Entidad-Relación Base de Datos central.
Elaboración Propia.

Java tiene herramientas para establecer conexiones hacia bases de datos de fácil implementación y versátiles a la hora de decidir cambiar de manejador de base de datos. La decisión de utilizar la herramienta JDBC (*Java Database Connection*) se basa en dichos principios.

Por motivos de programación orientada a objetos, la consulta realizada en la base de datos debe ser almacenada en un objeto de dominio (*InfoTicket*) que contiene toda la información pertinente al vehículo; la forma de acceder a él es por medio de métodos de tipo get y set; recordando los principios básicos en programación orientada a objetos, a través de ellos se garantiza el encapsulamiento del objeto y se protege a la data de futuras alteraciones en tiempo de ejecución. La Figura 4.13 muestra el diagrama de clases de la aplicación actual junto al objeto de dominio con los métodos necesarios para setear y obtener la información. Para el esquema de funcionamiento del diagrama de clases *véase Apéndice C (Figura 9.C.1)*.

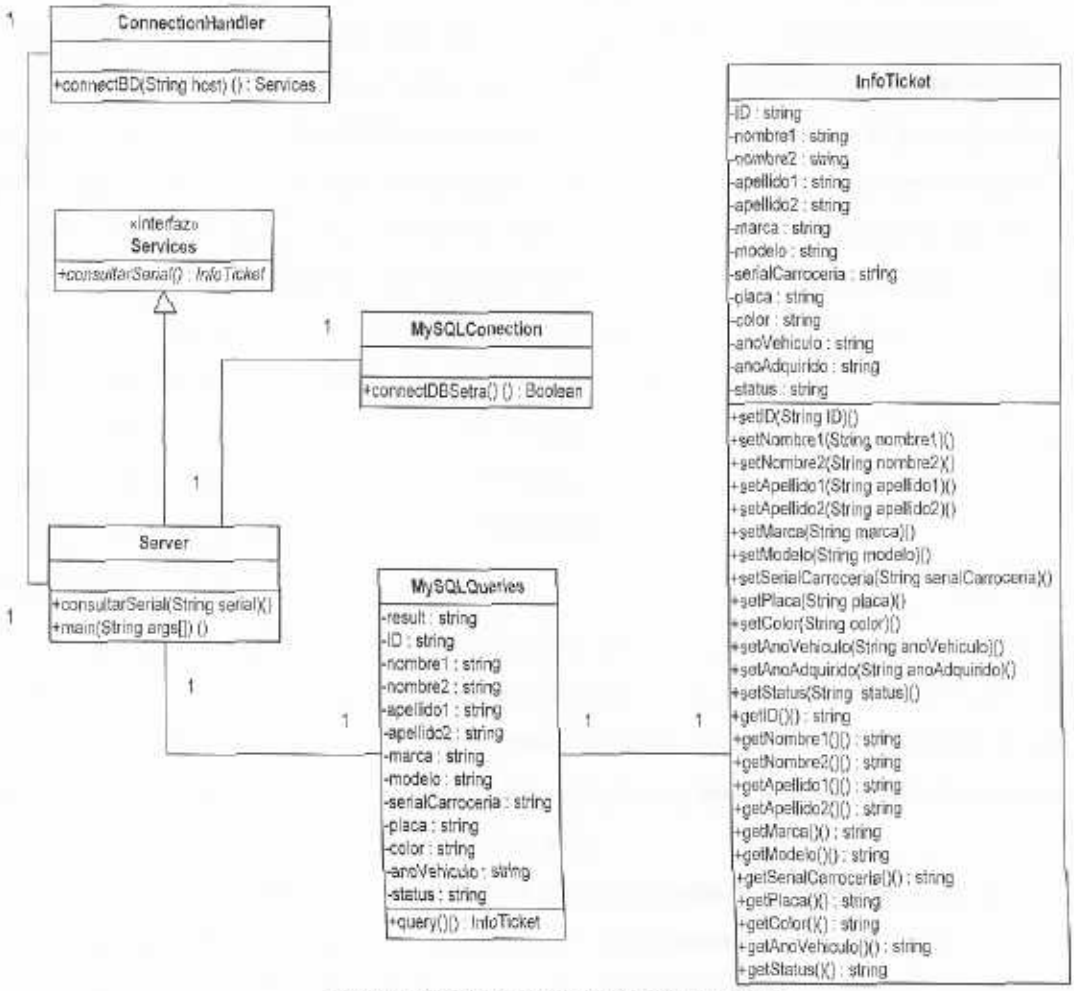

Figura 4.13. Primer Diagrama de Clases para el servidor.
Elaboración Propia.

La clase *ConnectionHandler* se encarga de establecer la conexión hacia el Servidor (Clase Server) mediante RMI. Es necesario destacar que estas dos clases en conjunto con la interfaz *Services* forman parte del mecanismo de RMI; una vez establecida la conexión entre el cliente y el servidor se invoca al método *connectBDSetra()* de la clase *MySQLConnection*, permitiéndose a través de JDBC la conexión a la Base de Datos. Obtenida ella, se invoca al método *query()* de la clase *MySQLQueries* que ejecuta la consulta a la base de datos y setea la información

obtenida en el objeto de dominio *InfoTicket* mediante un método *set*. Una vez seteado el objeto, se retorna.

4.2.3 Desarrollo

El desarrollo de la presente iteración se realizó en un tiempo corto ya que la fase de diseño y planificación fue pensada para este fin. El primer paso consistió en instalar el servicio de manejador de base de datos, utilizando el paquete de soluciones WAMP. La creación de las tres tablas necesarias para la consulta fue el paso posterior, para luego integrar la aplicación al objeto JDBC y configurarla para tener acceso al esquema de base de datos previamente creado. Una vez finalizada la parte de base de datos, se inició la implementación de RMI que terminó constando del desarrollo de dos clases: *ConnectionHandler* y *Server* además de la interfaz *Services*. Con RMI desarrollado y estableciendo comunicación entre los equipos remotos, se inicia el desarrollo de las clases, *MySQLQueries*, *MySQLConnection* e *InfoTicket*. Dichas clases se desarrollaron teniendo en cuenta la modularidad y propensión del sistema a futuros desarrollos e inclusión de nuevas funcionalidades.

4.2.4 Pruebas

Las pruebas se basaron en verificar en primer lugar que la base de datos es funcional, y permite según las tablas creadas introducir información en ellas correspondiente al tipo de dato. Es importante señalar que esta fue la primera prueba realizada ya que la conexión entre los dos equipos remotos y la conexión del servidor para ejecutar la consulta resulta implícita en ella.

Una vez concluido satisfactoriamente el desarrollo y la inserción en la base de datos se procedió a elaborar un test de *JUnit*. Este invoca al método *connectBD(String host)*, pasándole como parámetro la dirección IP del servidor. Además fue necesario setear un valor de prueba (un serial de carrocería) que representaba a un vehículo insertado en la base de datos. La prueba se ejecutó y el comportamiento de la aplicación fue el esperado: se capturó el objeto con toda la información concerniente al serial del vehículo introducido.

Para el caso de error en la presente iteración, se elaboraron dos pruebas unitarias más; en la primera se introduce un IP desconocido. En la segunda prueba se introduce un IP conocido pero un serial desconocido. En ambos casos se obtuvo el comportamiento esperado: la aplicación arrojó dos mensajes alertando que se produjo

un fallo en la ejecución normal de la aplicación, sin que esto significara el colapso de la misma. Para la planificación de las pruebas unitarias, *véase Apéndice G.*

4.3 3ra Iteración. Integración de código. Implementación de VPN.

Debido a la naturaleza de la información, es necesario que la que viaja por la red se haga de manera segura. Que la información de ida y subida llegue integra y confidencialmente mientras se asegura que las partes involucradas son las que tienen los permisos para ella. De aquí la decisión por implementar una VPN. Es necesario integrar el desarrollo previo para empezar a tener de este modo un producto en culminación. La integración empieza en el hardware de transmisión inalámbrica del serial de vehículos y termina en la ejecución de la consulta contra la base de datos, obteniendo el objeto de dominio que contiene dentro de sus atributos el estatus del vehículo.

4.3.1. Planificación

El objetivo principal de la presente iteración es acoplar los dos desarrollos anteriores de forma que se tenga el ciclo de transmisión y consulta finalizado. Es necesario antes del acoplamiento tener en cuenta que se debe garantizar la integridad de los datos mientras viajan por la red, la herramienta más indicada para dicha tarea es una conexión VPN, la creación de dicha conexión bajo el sistema operativo Windows (el cual fue utilizado para el desarrollo del presente trabajo de grado) no toma más de cinco minutos y garantiza la integridad de datos mencionada anteriormente.

Es necesario diseñar un mecanismo que se encargue de que cada vez que el objeto instanciado de la clase *DataIn* llene completamente las dos posiciones del arreglo mencionado en la iteración No. 1, se lleve a cabo una discriminación en la información contenida en él; que se garantice que la información capturada corresponde a un vehículo y está libre de errores.

La integración debe hacerse de la forma menos traumática para el código con la finalidad de conservar la integridad de la funcionalidad de la aplicación. En pocas palabras se debe utilizar el menor número de clases posibles para la integración.

El producto de esta iteración debe ser la respuesta del servidor según la información recibida desde el vehículo. La respuesta debe contener la información completa del vehículo además de la información de su dueño actual. En pocas palabras, el servidor debe devolver íntegramente el objeto *InfoTicket* con toda la información.

4.3.2 Diseño

Previamente al desarrollo del diseño de integración fue necesario definir el algoritmo de discriminación para la data obtenida desde el receptor. Como preámbulo se puede decir que se basa en comparar los seriales obtenidos asignándole a cada uno un valor de comparación (o votos) y seleccionando aleatoriamente uno de comparación. Aquel con el mayor valor de comparación será el usado para la consulta posterior en la base de datos. Para el pseudocódigo del algoritmo, *véase Apéndice C (Figura C.1)*

La integración de las dos iteraciones requiere la creación de una clase más. Se denominó *DataFixer* y resultó ser la encargada de implementar el pseudocódigo. Es necesario destacar que la clase DataIn se le cambió el nombre a *SerialListener*, para tener un nombre más descriptivo. A continuación en la Figura 4.15 se muestra el diagrama de clases luego de realizada la integración.

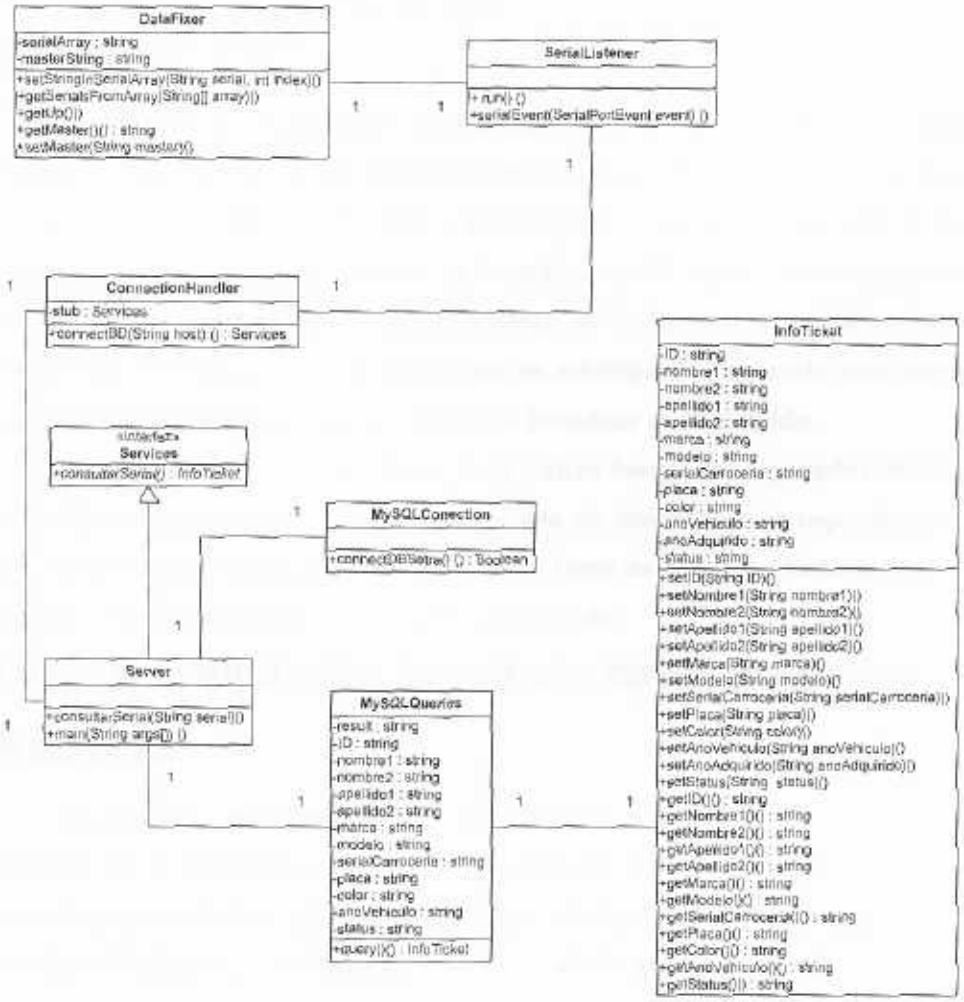

Figura 4.15. Primer Diagrama de Clases para la Aplicación Completa. Cliente y Servidor Integrados. Elaboración Propia.

4.3.3 Desarrollo

El desarrollo de la presente iteración se comenzó con la implementación del pseudocódigo anterior. La integración de la clase *SerialListener* solo se basó en invocar al método de la clase *DataFixer* que dispara la verificación de la ráfaga de bits que la clase *SerialListener* le pasa por parámetro.

La implementación de la conexión VPN es una opción que el sistema operativo Microsoft Windows trae por defecto; para crear una conexión VPN es necesario desde el lado del servidor elegir opción de creación de una nueva conexión de red de tipo VPN y asignarle una contraseña, esta conexión consta de un IP particular asignado mediante protocolo DHCP. En el lado cliente se crea una conexión del mismo tipo y se introduce la contraseña que se creó en el servidor, una vez realizados estos dos pasos la conexión se establece y para lograr que la aplicación se conecte mediante VPN se introduce el IP creado por la conexión VPN.

4.3.4 Pruebas

La fase de pruebas para la presente iteración se basa en el mismo modelo aplicado en las anteriores. Las mismas cuatro pruebas se aplican con la diferencia de que unas se incluyen dentro de otras y el código de la aplicación se une con el código de pruebas. En la primera prueba se introduce el IP del servidor y se verifica la conexión; establecida ella se procede a llenar el buffer con un serial existente en la base de datos. La segunda prueba se basa en establecer la conexión pero introducir un serial inválido. La tercera prueba se trata de introducir un IP inválido.

En cada caso los resultados de las pruebas fueron los esperados. En la primera se establece la conexión y se devuelve el objeto de dominio; en la segunda se arroja un mensaje de error al no conseguir el serial en la base de datos. La tercera prueba arroja una excepción al introducir el IP del servidor inválido.

4.4 4ta Iteración. Interfaz gráfica. Testigo de paso. Impresión de ticket. Bitácora de flujo vehicular. Cifrado.

Al llegar a esta iteración se tiene la conexión VPN, la base de datos, el hardware de transmisión – recepción y todos en un estado funcional; es necesario desarrollar una interfaz gráfica amigable, un testigo que indique la autorización del paso del vehículo y el mecanismo de impresión del ticket.

4.4.1 Planificación

La interfaz gráfica debe contener el mínimo de componentes funcionales. La función del usuario final –el que se encuentra en la caseta- es solo de observar si el vehículo está solicitado o no, y decidir consecuentemente entre permitir el paso o no hacerlo. Se decide no dejar como automática la restricción del paso debido al peligro que puede significar para la persona a cargo. El servidor debe emitir una señal de alerta en caso de que el estatus del vehículo sea *solicitado*, esta alerta (sus efectos y consecuencias) sale fuera del ámbito del proyecto. Luego de ahí, se deja a criterio del personal de la caseta permitir el paso o no. El vehículo de todas formas ya ha sido reportado.

El ticket impreso para el cliente debe contener un número de ticket secuencial de 8 dígitos, la hora y fecha de llegada al peaje, descripción del vehículo (placa, color, modelo, año), nombre y apellido del propietario, no es necesario imprimir el status del vehículo; el ticket es una constancia impresa de que el vehículo en un momento determinado pasó por el punto de control. Además de él, parece apropiado llevar una bitácora en la computadora donde se implemente la aplicación, siendo esta un archivo de texto en el cual se registra el paso de cada vehículo que establezca una conexión con la caseta. Esta bitácora contiene además de la información desplegada en el ticket, el estatus del vehículo y está conservada de forma encriptada para mantener su integridad y así mismo evitar que personas no autorizadas tengan acceso a él y lo modifiquen. En este servidor se debe desplegar una ventana en caso de que haya un vehículo solicitado.

4.4.2 Diseño

Para establecer una comunicación eficiente y brindar practicidad de uso, en la interfaz gráfica se establece que debe existir una ventana que invite al usuario a iniciar la aplicación introduciendo en un campo de texto el IP del servidor y un botón con la opción para iniciar la conexión. Una vez que se establece la conexión con el servidor se cierra la ventana. No le es posible a una persona ubicada en la caseta cerrar la conexión. Cada vez que un vehículo es verificado en la base de datos se despliega por pantalla una nueva ventana que muestra la información del vehículo (*InfoTicket*) y dos (2) botones con opciones de abrir la barra de paso y alertar autoridades, respectivamente. Ella permanece abierta mientras se ejecuta la impresión del ticket y debe cerrarse automáticamente luego de transcurridos 20 segundos. Tiempo necesario para que se

lleve a cabo la impresión del ticket y se ejecute el testigo de paso. La ejecución del testigo de paso reside en simplemente activar un pin del puerto paralelo LPT1 porque se considera que el hardware de funcionamiento de una barra de paso real, está ya desarrollado implementado. El principio de esta idea está en que ese hardware solo necesita recibir una señal que lo haga funcionar.

Para cumplir todos los pasos descritos anteriormente es necesario modificar el diagrama de clases planteado en el diseño de la iteración 3 para resultar en el que se despliega en la Figura 4.17.

Las clases nuevas creadas básicamente encargan de dibujar la interfaz gráfica. Tienen por nombre: *Connect*, *ProgressBar*, *InspectorWindowsy* y *TicketPrinter*. La clase *Connect* despliega una ventana de interfaz sencilla donde el usuario inserta el IP y solicita la conexión. La clase *ProgressBar* despliega una ventana con una barra de progreso que indica el avance de la ejecución de consulta del serial del vehículo. La clase InspectorWindows despliega una ventana con la información del objeto de dominio obtenido luego de efectuar la consulta del serial en la base de datos, y un botón con la opción de alertar a las autoridades y otro para abrir la barra de paso. *TicketPrinter* se encarga de transformar la información contenida en el objeto InfoTicket, enviarlo a la impresora. El formato del ticket se muestra en la Figura 4.16.

Figura 4.16. Formato de Ticket.
Elaboración propia.

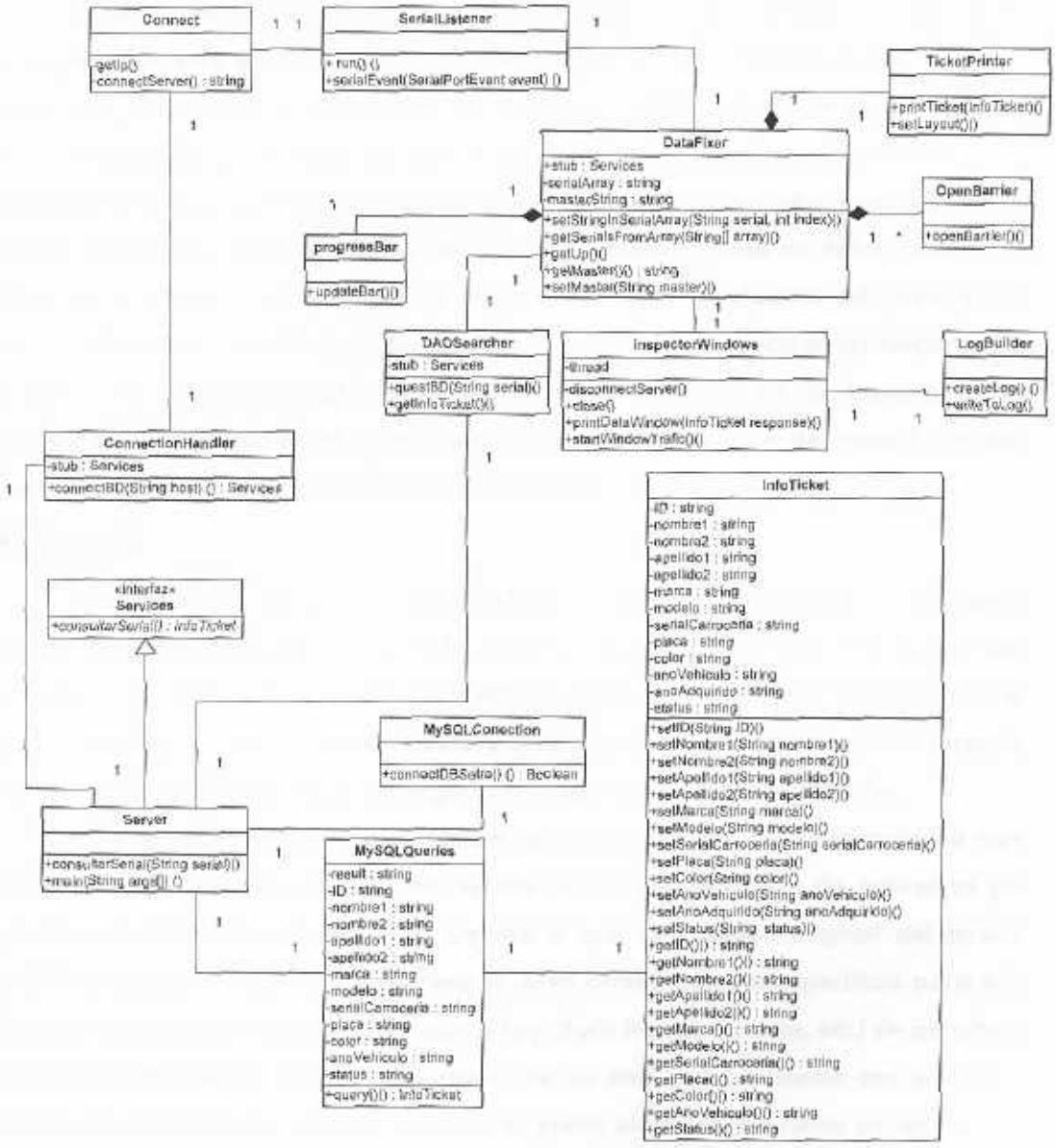

Figura 4.17. Segundo Diagrama de Clases para la Aplicación Completa. Cliente y Servidor Integrados. Elaboración Propia.

Existen dos clases que ejecutan lógica del negocio, una de ellas se encarga específicamente de disparar el evento que mediante puerto paralelo activa la barrera para permitir el paso del vehiculo. La segunda se encarga de crear el archivo de bitácora y de escribir en él. *Para el esquema de funcionamiento del diagrama de clases véase Apéndice C (Figura 9.C.2).*

El funcionamiento comienza cuando el usuario introduce el IP del servidor donde se encuentra la base de datos junto con la aplicación servidor. Una vez que se establece la conexión, se comienza con la lectura de la ráfaga de bits que provienen del receptor (clase *SerialListener*), al llenar el *buffer* con información sensata, se procede a la discriminación de los seriales para elegir el más coherente (DataFixer). Acto seguido se procede al llenado del objeto luego de la consulta desde la base de datos. Cuando se obtiene una respuesta del servidor se imprime el objeto de dominio *InfoTicket* en la ventana (clase *InspectorWindow*). Luego de ello se procede a ejecutar las funciones de impresión de ticket en papel (clase *TicketPrinter*), apertura de la barrera (clase *OpenBarrier*) e imprimir en el archivo de bitácora la información del vehículo en ese momento consultado. Todas ellas ejecutadas en ese orden.

4.4.3 Desarrollo

La construcción de una interfaz gráfica en Java resulta usualmente una tarea sencilla. Prácticamente no existen IDE's que no ofrezcan esas facilidades, ya sea mediante el kit *Swing* o el más antiguo *AWT*. El patrón de diseño *MVC* es contextual al sistema. El código en ellas solamente incluye el necesario para dibujarlas y la referencia a la clase que ejecuta la lógica del negocio, o la que controla dicha ejecución.

Java no es un lenguaje que maneje fácilmente puertos de comunicación a muy bajo nivel como si lo hacen otros. Esto sucede en el caso específico de activar un pin específico del puerto paralelo LPT1. Debido a que la intención original del puerto paralelo era para el manejo de impresoras, Java ofrece métodos específicos para ello (chequear si se acabó el papel, chequear si hay algún trabajo en cola, etc.) sin permitirlo tan bajo como el envío de un bit a través de un pin. Particularmente con el puerto paralelo, la limitación de Java con él, hizo necesario elaborar una rutina en lenguaje C invocada desde la aplicación principal que activara enviara un byte ("0000001") al puerto paralelo LPT1.

C, por ser un lenguaje a más bajo nivel que Java, se muestra capaz del manejo de puertos y de mucha facilidad de implementación, por lo que fue la mejor opción para la elaboración del código.

4.4.4 Pruebas

Las pruebas en esta fase fueron iguales a las aplicadas en la iteración anterior; los cambios de lógica de búsqueda fueron innecesarios, y los componentes añadidos solo fueron de despliegue de información y ejecución de rutinas

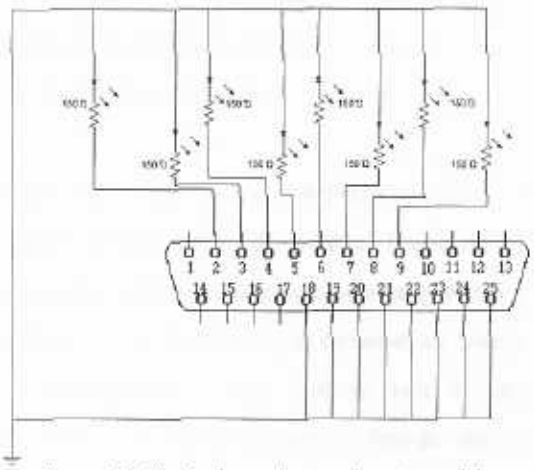

Figura 4.19. Circuito de prueba para el puerto paralelo. Elaboración Propia.

Las pruebas del programa C se basaron en construir un circuito con de 8 diodos LED conectados al puerto paralelo LPT1, cada diodo conectado a un pin de data de salida del puerto; para el presente trabajo de grado solo resulta necesario enviar una señal de encendido, es por ello que resulta la decisión de mandar un byte con el valor mencionado. En la Figura 4.19 se muestra el diseño del circuito utilizado para la prueba. Para la planificación de las pruebas unitarias, *véase Apéndice G*. Para las pruebas completas del sistema, se simuló en condiciones similares a las descritas en la 1era Iteración una serie de validaciones para 3 vehículos consecutivos, uno de ellos con el estatus de solicitado. El éxito de la aplicación hasta esta iteración se consigue en primer lugar al ver en la pantalla del peaje el status del vehículo como solicitado en caso de que lo esté, o no solicitado en el caso contrario, ejecutando el testigo de paso, registrando su llegada en el servidor e imprimiendo el ticket. En segundo lugar se observa la correcta ejecución de la aplicación al desplegarse una ventana en el servidor que notifica que existe un vehículo con status solicitado.

4.5 5ta Iteración. Diseño y Desarrollo de aplicación de administración

Es necesario crear una aplicación administrador que se encargue de gestionar ciertos aspectos de la aplicación bien sea para tener un control de la misma, o sencillamente le sirva al administrador como una forma de visualización general o de determinado tópico del negocio.

4.5.1 Planificación

Como se comentó en el párrafo anterior, es necesario desarrollar una aplicación subyacente a la principal encargada de la verificación constante de vehículos. Debe ser

capaz de manejar cierto número de personal autorizado, por lo que se hace necesaria una tabla nueva en donde se registre cada usuario autorizado –los llamados *administradores*. La aplicación solo permitirá acceso al personal que se encuentre almacenado en la tabla.

La aplicación de administración debe ser capaz de realizar respaldos automáticos de la base de datos que contiene la información de control (vehículos y propietarios); debe permitir realizar respaldos (y de cargarlos) cuando sea requerido y debe ser la encargada de crear una bitácora para el tráfico que accede al peaje guardando cada registro de manera segura (encriptada) pero que su vez le sea disponible al administrador; es decir, las bitácoras son solo visibles a través de la aplicación. Se debe ofrecer la opción de visualizar los archivos de bitácora para que el administrador elija cual desea examinar; se debe ofrecer la opción de crear un respaldo en cualquier momento por parte de un usuario administrador así como también poder elegir la fecha de respaldo automático; en cuanto a la gestión de usuarios es necesario ofrecer las opciones de creación de un nuevo usuario, modificación de clave de usuario actual, cerrar e iniciar sesión y eliminar usuario administrador; siempre a través de una interfaz amigable e intuitiva.

IV.5.2 Diseño

Figura 4.20.Base de Datos para Administradores. Elaboración Propia.

En primer lugar se debe diseñar la base de datos que contiene la información referente al administrador, para almacenar dicha información solo se requiere una tabla y se muestra en la Figura 4.20. La razón por la que se decide usar una base de datos en lugar de un método alternativo de rápido acceso (archivo de texto plano, XML) está en la posibilidad de crecimiento de la aplicación de administrador del peaje al fusionarse con una de administración de la base de datos de los vehículos; la asignación de privilegios para los usuarios, los niveles de responsabilidades que puedan tener y demás posibles mejoras.

El campo ID se refiere un número de control, el campo *USER_NAME* se refiere al nombre de usuario y es el atributo clave. El campo USER_WORD se refiere a la clave de acceso. Lo que se almacena en los 2 últimos, es un número resumen obtenido a través de la clase *Hasher*.

La Figura 4.21 muestra el diagrama de clases de la aplicación de administración. Para el esquema de funcionamiento del diagrama de clases *véase Apéndice C (Figura 9.C.3)*.

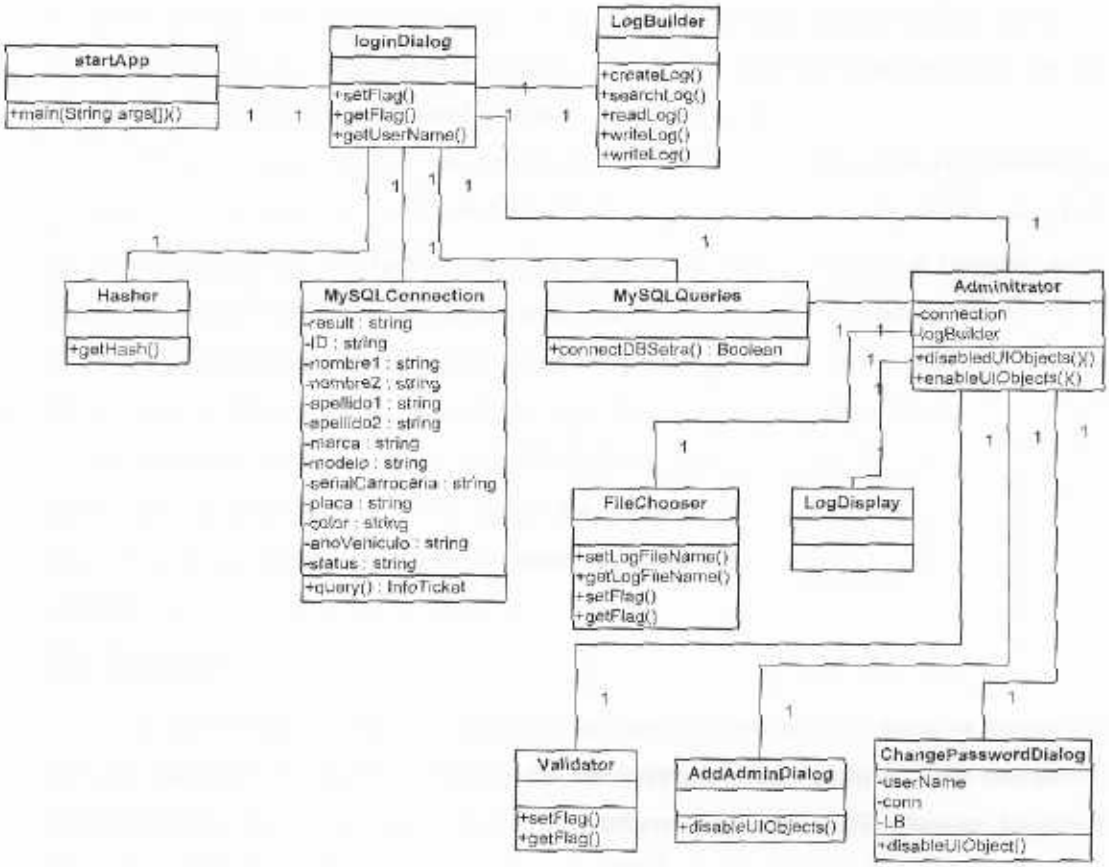

Figura 4.21. Diagrama de Clases de la aplicación de administración.
Elaboración Propia.

El funcionamiento de la aplicación comienza en una clase denominada *StartApp* que contiene un método *main* cuya labor es hacer nada más que hacer visible una ventana a través de la cual se inicia la sesión (*LoginDialog*). Esta muestra un par de campos para la validación de datos del administrador (nombre de usuario y contraseña); en el caso de que sea válido, se puede acceder libremente a las opciones de la aplicación posterior. Esta validación es a través de una clase *Hasher* que obtiene el número resumen de los campos introducidos y luego a través de *MySQLQueries* son comparados con los almacenados en la base de datos.

Entre las opciones que puede hacer uso el usuario se encuentra *cambiar password*; se le requiere introducir su clave antigua y dos veces la nueva para asegurar que el nuevo es efectivamente el deseado. Una opción trata de visualización de las bitácoras. Se despliega una lista de todas las bitácoras existentes, ordenadas por fechas de creación donde el usuario es capaz de seleccionar la que desea revisar para luego verla en detalle. Es importante destacar que dichas bitácoras son archivos de texto encriptados, desencriptadas en el momento de visualización.

Como es natural con las bases de datos, es necesario que regularmente se realice un respaldo de la contenida. El *backup* de la base de datos se realiza automáticamente en una fecha que determine el administrador, pero también está la opción de que el administrador la ordene, a su criterio para el día actual. Cada vez que el administrador elija cualquiera de las dos opciones anteriores se le solicitará su nombre de usuario y clave, para garantizar que la persona usando el programa en ese momento, es un administrador. Adicionalmente, para razones de análisis forense se crean un archivo *log* con la fecha de acceso a la aplicación por parte de administradores donde se registran todas las acciones que ellos ejecutan para un día determinado.

5.5.3. Desarrollo

El desarrollo se basó en elaborar las ventanas necesarias para la ejecución de las opciones que ofrece la aplicación de administración. Son en total siete ventanas, de las cuales cinco solo se encargan de mostrar información (Clases: *FileChooser*, *LogDisplay*, *Validator*, *AddAdminDialog*, *ChangePasswordDialog*). Las otras dos se encargan del despliegue de las opciones de la aplicación y de disparar la ejecución de la lógica del negocio.

No fue necesario añadir clases que realizarán conexión contra la base de datos ni una nueva clase de sentencias MySQL. Solo se le agregaron métodos a las ya existentes *MySQLConnection* y *MySQLQueries*.

El mecanismo de respaldo ocurre a través de una herramienta instalada por defecto en Windows (desde Windows 98). *Task Scheduler* le permite al usuario ejecutar tareas automáticas en una máquina cualquiera. Monitorea los parámetros (disparadores o *triggers*) elegidos para ejecutar la tarea y la ejecuta cuando ellos son cumplidos. Para este caso, se elaboró un *script* de Windows que ejecuta un programa llamado

5.1 Resultados

Podemos establecer en general que se obtuvieron los siguientes resultados: se logró construir un módulo de recepción y tres módulos de transmisión de datos, se desarrolló una aplicación capaz de capturar los datos y una verificarlos según lo contenido en una base de datos, y se logró desarrollar una aplicación de administración encargada de gestionar la estructura debajo de la aplicación primaria. Desarrollado todo paso a paso siguiendo la metodología elegida. Sin embargo, podemos revisarlos más específicamente según los objetivos específicos planteados.

- *Recaudar información desde la fuente original (peaje) para establecer niveles de flujo vehicular y puntualizar datos sobre la infraestructura física y tecnológica ahí existente.*

Para obtener una referencia del ambiente a ser implementado el sistema, se hizo necesario movilizarse hacia el peaje en funcionamiento más cercano a Caracas. Se tomaron medidas en el ubicado en la Autopista Regional del Centro, de distancias y alturas entre el vehículo y la caseta. Sobre tecnología, el peaje no cuenta con demasiada (solo una caja registradora por caseta). De ahí que lo sugerido en el presente trabajo de grado tendría que implementarse desde las raíces. El flujo vehicular establece para el manejo de este Trabajo Especial de Grado la importancia del sistema: tiene que estar en funcionamiento ininterrumpido.

- *Definir estructuras de datos necesarios para la correcta verificación del vehículo.*

Las estructuras de datos se basaron en los requerimientos necesarios del sistema entero; se crea un objeto de dominio (*InfoTicket*) donde se encapsula el resultado de cada consulta referente a vehículos y propietarios ejecutada sobre la base de datos. Una base de datos relacional céntrica –donde estará toda la información- se define para obtener en una búsqueda lo necesario para otorgar el permiso de paso. Archivos de texto plano posteriores registran cada vehículo que atraviesa el peaje.

- *Estudiar el sistema de información actualmente implementado en el registro automotor nacional.*

El objetivo se cumplió a pesar de la información es clasificada y no está disponible al escrutinio público. El único acceso se logró a través de la aplicación implementada en el puesto de trabajo de un usuario y por los parámetros de búsqueda que se requieren.

- *Estudiar el espectro radioeléctrico establecido por CONATEL para evitar interferencias con otros dispositivos de comunicación.*

La información se encuentra disponible al público en la página Web del organismo estatal CONATEL. Sin embargo, fue necesario dirigirse a las instalaciones para verificar que ella no está desactualizada, que no hay restricciones en el uso de ella y, más importante aún, que no las hay para el uso de la frecuencia.

- *Evaluar los distintos tipos de transmisores y receptores que existen en el mercado cuyas características encajen con las necesarias para el desarrollo del sistema.*

Según la información obtenida en CONATEL se estudió la utilización de transmisores que operen en la banda de 433 Mhz, de este objetivo resultó la decisión de usar los transmisores STT – 433 y STT – 433.

- *Definir e implementar políticas de seguridad a aplicarse sobre los dispositivos de transmisión y recepción.*

La seguridad del serial del vehículo se basa en el cifrado por clave pública de los datos. Los módulos son de fácil adquisición e inclusive de mayor facilidad de uso. La razón de la encriptación reside en el derecho a la privacidad del propietario del vehículo; los transmisores irradian constantemente la señal y solo son necesarios ligeros conocimientos de electrónica para incurrir en algún tipo de espionaje inalámbrico. La encriptación ocurre para evitar que los datos contenidos en el PIC16F84 del propietario se transformen en conocimiento de dominio público.

- *Desarrollar un mecanismo de conexión entre el módulo de recepción de datos y la PC preparada para manejarlos.*

El mecanismo fue diseñado, desarrollado y sometido a pruebas mediante el lenguaje de programación Java. Él provee las funcionalidades necesarias para proporcionar niveles de comunicación relacionados con el puerto serial de la máquina (libreria RX/TX).

- *Desarrollar la base de datos donde se registrará el tránsito local; su función será sostener la información para comparar luego con la base de datos remota, con los parámetros de la existente en el registro automotor nacional actual.*

Para este objetivo se desarrollo un repositorio de datos no relacional en el computador que actuaría como cliente. Cada vehículo que accede al peaje se registra en un archivo de texto plano, en cada registro se almacena la información concerniente

a la placa, serial de carrocería, hora y fecha de paso del vehículo por el andén del peaje. Se evitan de esta manera, problemas de replicación de datos y de ejecución de transacciones ubicando la base de datos relacional en el servidor.

- *Establecer la conexión entre la base de datos local y la remota.*

Debido al punto anterior no se elaboró la conexión, solo se establece la conexión necesaria para la verificación del serial y la respectiva respuesta del servidor a dicha consulta.

- *Desarrollar una interfaz gráfica para el usuario y el administrador del sistema, con la posibilidad de manejar los diferentes dispositivos que se ven involucrados en todo el proceso, exceptuando los de transmisión y recepción.*

La interfaz gráfica se desarrollo pensando en la facilidad de uso. Como se menciona a lo largo del tomo se desarrollaron dos aplicaciones, la aplicación verificadora y la aplicación de administración. La aplicación verificadora permite al usuario accionar el testigo de paso vehicular y emitir una alarma silenciosa hacia el servidor; la aplicación administrador permite llevar a cabo tareas de mantenimiento y control sobre las dos aplicaciones. Debe ser difícil para un usuario cualquiera acceder a la información, pero sencillo para un administrador con credenciales válidas.

- *Instalar y configurar los equipos de transmisión en los diferentes vehículos, luego de estudiar las condiciones para lograr funcionalidad de ellos.*

En primer lugar se diseñó, ensambló y se sometió a prueba a los circuitos necesarios para transmitir la data (módulo de transmisión, módulo de alimentación), una vez completado este paso se procedió a probarlos en el laboratorio y acto seguido luego de pasar las pruebas de laboratorio instarlos en los vehículos.

- *Instalar y configurar el equipo de recepción luego de estudiar las condiciones que logren funcionalidad de él y de la interacción con los dispositivos de transmisión.*

Se diseño, ensambló y se llevaron a cabo las pruebas necesarias para el módulo de recepción y el módulo de alimentación del mismo, una vez que se verificó en el laboratorio que el módulo recibe data se procede a verificar la recepción desde los módulos de transmisión.

- *Programar los microcontroladores para que se comporten como una interfaz entre el sistema de transmisión y recepción, la PC local y el dispositivo de subida y bajada de la barra de paso.*

Este objetivo se embebe una vez que se diseñan y se ensamblan los módulos de transmisión; cada módulo consta de un PIC donde se almacena el programa que permita la transmisión de data. El dispositivo para permitir el paso del vehículo se desarrollo mediante el lenguaje de programación C el cual haciendo uso del puerto paralelo dispara una señal que indica la apertura de la barrera.

- *Desarrollar el módulo para la impresión de tickets en la PC local donde se presentará la información capturada desde el módulo de recepción.*

El módulo se desarrolló mediante el lenguaje de programación Java usando una impresora doméstica corriente, la impresión se lleva a cabo luego de la verificación del vehículo independientemente del status de solicitado o no.

- *Realizar las pruebas de verificación de funcionalidad de todo el sistema instalado.*

Las pruebas de software se elaboraron mediante JUnit, herramienta provista por Java. Las pruebas de los módulos de recepción y transmisión se elaboraron mediante el uso de dispositivos de laboratorio tales como voltimetros y osciloscopios. Una vez pasadas todas las pruebas se hizo funcionar todo el sistema repetidas veces simulando distancias y condiciones reales para tres vehículos consecutivos, uno de ellos en condición de solicitado en la base de datos, las pruebas fueron completadas todas con éxito.

6.1 Conclusiones

Al inicio de cualquier desarrollo de software resulta apropiado hacer un buen levantamiento de requerimientos; ello se resume en nada menos que la base fundamental sobre el que se planea el diseño, ajustándolo a la realidad y a lo que se espera de ella, mientras se logra una visión más acertada sobre las tecnologías y arquitecturas posibles para el despliegue final. Que se ajusten coherentemente contra los objetivos y además se traduzcan en márgenes de error tolerables, fortalecidos por la certeza del ambiente de trabajo de la aplicación final, son ventajas importantes que darán sus frutos al final del día junto a los resultados inherentes al desarrollo. Para nuestro caso fue necesario definir las distancias entre el vehículo y la caseta del peaje, fue necesario documentarse en CONATEL sobre las frecuencias libres para uso y así evitar incurrir en multas o procedimientos legales engorrosos y tener idea sobre la banda que se utilizaría para realizar el posterior estudio en el mercado sobre los módulos que se adaptarían de la mejor forma. Además, estudiar en el SETRA el sistema de información utilizado resulto de importancia para las cuestiones posteriores del manejo de información.

La tecnología actualmente está al alcance de todos. Es increíblemente sencillo conseguir en Internet documentación sobre la elaboración de cualquier dispositivo en hardware y/o software para el uso libre. La mala intención, consideramos nosotros, no es extraña a la naturaleza humana. Así como es increíblemente fácil conseguir información de cualquier cosa en Internet, es de igual manera increíblemente fácil utilizarla para el beneficio propio a costillas de otros. Es factible que alguna persona, con el ímpetu suficiente y no demasiados recursos, se decida a elaborar algún mecanismo malicioso que busque perjudicar al sistema. Por ello es importante tener que implantar políticas de seguridad que garanticen la seguridad de los datos utilizados, incluida su confidencialidad si son considerados vitales y privados, mientras dificulten cualquier perjuicio que pudiera ser ocasionado. Hay que ser realista, los ataques contra los sistemas rara vez empeoran.

Las buenas prácticas de desarrollo de aplicaciones son consecuentes con una aplicación eficiente en cuanto a tiempos de respuesta y a nivel de código fuente. Obtener rutinas limpias y sencillas, resulta no pocas veces de la falta de una perspectiva clara en la definición de los requerimientos primarios, y por ello se hace tanto énfasis en

intentar tener esclarecer ese primer paso tanto como se pueda. Un ejemplo claro de esto está en la siguiente decisión: se puede implementar un circuito electrónico para el manejo de un motor eléctrico, o ingeniar una manera más natural –o familiar, en tal caso– que utilice rutinas en software y las interfaces de comunicación del computador para reducir, en primer lugar, el tiempo de desarrollo de la aplicación y en segundo, los costos finales del proyecto. Se está prescindiendo de la planificación y construcción en físico de los circuitos, pero al final depende desde qué perspectiva se sienta el desarrollador más cómodo, y de cuál sea la más eficiente.

Muchas aplicaciones fracasan por tener interfaces gráficas excesivas y cargadas de componentes que resultan abrumadores para el usuario final. Siempre se debe optar por ofrecer la forma más sencilla (en la medida que la aplicación lo permita) para la representación de las distintas funcionalidades que se ofrecen en cada ventana desplegada. Para el presente trabajo especial de grado se eligió una interfaz grafica minimalista donde el usuario pueda desempeñarse con comodidad pero permitirle disfrutar del completo potencial de la aplicación. De igual manera, se proporciona una visibilidad amigable, e inclusive intuitiva a la hora de obtener los resultados de las diversas transacciones efectuadas a lo largo de la ejecución de la aplicación.

El entorno de desarrollo es de vital importancia para codificar rápidamente. Java es un lenguaje de gran poder y versatilidad que brinda un alto número de librerías que facilitan el trabajo del desarrollador. Haciendo referencia a esas mismas prácticas, de la programación y patrones de diseño exitosos en el pasado reciente, puede obtenerse como producto final una aplicación de elevada calidad. A pesar de tener una herramienta poderosa de programación es necesaria la elección de una metodología que se adapte

Cada vez que finaliza una iteración (como lo hacen generalmente las metodologías en su geometría) es necesario someterlas a pruebas exhaustivas; esto se refiere a no solo probar con data factible, sino proponer un ambiente no favorable y condiciones en algunos casos perjudiciales, para observar el comportamiento de la aplicación y planificar su robustez. Toda aplicación final debe ser a prueba de fallos; ello es, debe resistir eventualidades desfavorables, mientras se actúa para su manejo y prosigue su proceso normal de ejecución. El éxito de un proyecto, de todas maneras, reside en la habilidad del desarrollador para hacerlo funcionar, lo haga funcionar bien y que lo haga en las condiciones particulares del ambiente de implementación.

6.2 Recomendaciones.

Todo sistema a pesar de ser elaborado siguiendo los mejores estándares de diseño y desarrollo, siempre tendrá módulos y componentes candidatos a mejoras manteniendo en una constante evolución el producto final. Las recomendaciones para el presente trabajo de grado se presentan a continuación:

- Para no limitar la distancia de transmisión del módulo ubicado en el vehículo se sugiere incrementar la antena e incorporar al sistema de recepción un sistema multicanal, dicho dispositivo es capaz de establecer canales de comunicación en la misma señal. Este dispositivo permite realizar la verificación más dinámica si se toma en cuenta que se pueden verificar varios vehículos de forma concurrente; alternativamente a este sistema multicanal se puede idear un protocolo electrónico en el cual el transmisor se active una vez que un dispositivo en la caseta le dé una señal para activarse. Así se evita la difusión en todo momento del serial del vehículo y se ofrece un nivel mayor de privacidad en la información, mientras a su vez aumenta el tiempo de vida del circuito.

- Se puede ampliar la aplicación administrador añadiéndole un módulo que mediante gráficos despliegue información almacenada en los archivos de bitácora que permita visualizar mejor estadísticas de flujo vehicular en un tiempo determinado.

- Por razones de practicidad se sugiere añadir al módulo de transmisión el módulo regulador de voltaje del mismo ya que para el presente trabajo de grado son módulos independientes.

- Implementar un mecanismo de VPN distinto al disponible por defecto en el sistema operativo Windows. Aún cuando las fallas han sido corregidas y PPTP es considerado seguro, puede ser recomendable utilizar opciones más actuales en el manejo de redes privadas virtuales, como lo son IPSec o SSL.

- Utilizar claves más larga para el encriptado de datos, ya que en el presente trabajo de grado se usaron claves de 512 bits de longitud, al ampliar esta clave se mejora la seguridad. La limitación de esto está en el proveedor del algoritmo de encriptación; de considerar claves de mayor longitud, su implementación en Java sería posible usando otro proveedor distinto al usado por defecto (llamado *SunJCE*).

- Para aplicaciones a mayor escala -varios peajes- se recomienda implementar un servidor local en cada uno de ellos, donde se almacene una base de datos que

contenga la información de los vehículos pertenecientes a la región pero que esté de igual manera conectado al servidor central para poder validar vehículos de otras regiones.

- Es posible implementar un mecanismo de validación separado de los peajes debido a que los transmisores están en constante funcionamiento. Para alcabalas móviles, por ejemplo, solo sería necesario tener acceso inalámbrico a Internet y a la misma clave privada de descifrado.
- Manejar la validación de las aplicaciones cliente y servidor utilizando certificados digitales.
- Separar la aplicación cortafuegos de la máquina servidor, para liberarla de consumo de recursos.
- Estudiar frecuencias de onda alternativas para el despliegue de la aplicación.
- Reducir el tamaño de los circuitos al menor posible.

[1] Microcontroller. *Wikipedia, The Free Encyclopedia*, [en línea].
Recuperado el 2 de Enero de 2007 de,
http://en.wikipedia.org/wiki/Microcontroller.
[2] Microcontroladores PIC de la Gama Media. *Universidad de Extremadura. Escuela de Informática Industrial*. Santiago Salamanca y David Muñoz.
Recuperado el 3 de Enero del 2007 de,
http://tauro.unex.es/vaguiti/INFORMATICA_INDUSTRIAL/bloqueIII/Cap1_PIC.pdf
[3] Microcontrolador. *Wikipedia, La enciclopedia libre*,[en línea].
Recuperado el 4 de Enero del 2007 de,
http://es.wikipedia.org/wiki/Microcontrolador
[4] Emisión de ondas electromagnéticas. Antenas, [en línea].
Recuperado el 8 de Diciembre del 2007 de,
http://omega.ilce.edu.mx:3000/sites/ciencia/volumen3/ciencia3/112/htm/sec_19.htm
[5] Java Overview. *Sun Microsystems*, [en línea].
Recuperado el 9 de Diciembre de 2006 de,
http://java.sun.com/javase/technologies/desktop/techoverview.jsp
[6] RMI. *Sun Microsystems*, [en línea]
Recuperado el 15 de Diciembre de 2006, de
http://java.sun.com/javase/technologies/core/basic/rmi/index.jsp
[7] Historia y fundamentos de Java, [en línea].
Recuperado el 10 de Diciembre de 2006 de,
http://pisuerga.inf.ubu.es/lsi/Invest/Java/Tuto/I_2.htm
[8] Ciclo de vida de eXtreme Programing, [en línea].
Recuperado el 7 de Enero de 2007 de,
http://oness.sourceforge.net/proyecto/html/ch05s02.html
[9] Antenna Basics, [en línea].
Recuperado el 8 de Diciembre de 2006 de,
http://web.telia.com/~u85920178/antennas/anten.htm
[10] How does a CB radio antenna work, how stuff works [en línea].
Recuperado el 10 de Diciembre de 2006 de,
http://electronics.howstuffworks.com/question490.htm
[11] 433 MHz RF Receiver STR 433 [en línea].
Recuperado el 11 de Diciembre de 2006 de,

http://www.sunrom.com/media/datasheet/STR-433.pdf

[12] The RS232 Standard, A tutorial with Signal names and definitions, [en línea].
Recuperado el 8 de Diciembre de 2006 de,
http://www.camiresearch.com/Data_Com_Basics/RS232_standard.html

[13] Layer two Tunneling protocol, [en línea].
Recuperado el 6 de Enero de 2007 de,
http://searchnetworking.techtarget.com/sDefinition/0,,sid7_gci493383,00.html

[14] IPSec Protocol Overview, [en línea]
Recuperado el 7 de Enero de 2007 de,
http://www.freesoft.org/CIE/Topics/141.htm

[15] Historia de los microcontroladores, [en línea]
Recuperado el 2 de Enero de 2007 de,
http://www.geocities.com/electrogera666/micro/historia.html

[16] Enciclopedia Hispánica, "Radiocomunicación". Enciclopædia Britannica Publishers, Inc. 1991-1992.

[17] Enciclopedia Encarta 2007. "Bases de Datos". Microsoft ® Encarta ® 2007. © 1993- 2006 Microsoft Corporation.

[18] Herbert, Nick. "Quantum Reality." Anchor Press. 1987.

[19] Stallings, William." Fundamentos de Seguridad en Redes. Aplicaciones y Estándares. " Pearson-Prentice Hall.2004.

[20] Elmasri, Ramez y Navathe, Shamkat. "Fundamentos de Sistemas de Bases de Datos." Pearson-Adisson Wesley. 2002.

[21] Tanenbaum, Andrew. "Redes de Computadoras". Pearson-Prentice Hall. 2003.

[22] Schildt, Herbert. "Java 2. Manual de Referencia."Osborne-McGraw-Hill. 2001.

[23] Serway, Raymond y Beichner, Robert. "Fisica para ciencias e ingeniería." McGraw-Hill. 2002.

[24] Cryptanalysis of SHA-1, [en línea].
Recuperado el 15 de enero de 2007 de,
http://www.schneier.com/blog/archives/2005/02/cryptanalysis_o.html

[25] Risks of Relying on Cryptography, [en línea].
Recuperado el 15 de enero de 2007 de,
http://www.schneier.com/essay-021.html

ACID: Acrónimo para *Atomicity, Consistency, Isolation, Durability*. Serie de propiedades que garantizan que una transacción en una base de datos es procesada confiablemente.

Administrador: Figura que según una serie de privilegios es capaz de gestionar ciertos aspectos de una aplicación.

AM: Acrónimo para *Amplitude Modulation*. En comunicaciones electrónicas, se basa en variar la fuerza de la onda transmitida en relación a la información enviada.

API: Acrónimo para *Application Programming Interface*. Código fuente que define cómo debe invocarse una determinada función de un programa desde una aplicación; también se dice que son las herramientas de programación para rutinas, protocolos y software.

AWT: Acrónimo para *Abstract Windows Toolkit*. Paquete nativo de Java que se encarga de construir interfaces gráficas donde se define todo lo referente a posición y tamaños de lo que puede ser dibujado en la pantalla del PC; siendo además, es capaz de responder a eventos.

Backup: Respaldo de información realizada en un momento determinado.

Benchmark: Anglicismo que puede ser interpretado como una técnica de medición de rendimientos según algún parámetro de referencia.

Bit: El término bit deriva de la frase dígito binario y se basa en una señal electrónica que puede ser encendida (1) ó apagada (0).

Buffer: Memoria intermedia entre un dispositivo y otro.

Bus: Conductor eléctrico que transfiere datos o corriente entre conductores del ordenador o entre ordenadores.

Byte: Un byte es la unidad fundamental de *datos* en los ordenadores personales, un byte son ocho bits contiguos.

C/C++: Lenguajes de programación de alto nivel (C es el predecesor de C++) que permiten elaborar desarrollos estructurados basados en funciones. C++ se diferencia de C por permitir el desarrollo orientado a objetos.

Cifrar/Descifrar: Encriptar/Desencriptar.

Código Fuente: Texto escrito siguiendo ciertas reglas utilizado como base para generar código ejecutable por la computadora.

Copyleft: Práctica de usar leyes de derechos de autor para remover restricciones en la distribución de copias y versiones modificadas de un trabajo para otros, requiriendo que las mismas libertades sean preservadas en versiones modificadas.

CMOS: Acrónimo para *Complementary Metal Oxide Semiconductor*. Familia lógica empleada en la fabricación de circuitos integrados, utilizados para guardar datos básicos del hardware y de configuración como información de discos duros, fecha y hora entre otros.

Clase: Agrupación de datos y de funciones que operan sobre dichos datos.

Cristal de Cuarzo: Dispositivo que se encarga de generar pulsos eléctricos a una frecuencia por propiedades piezoeléctricas; el cuarzo tiene como propiedad que al tallarlo generalmente en forma de lámina y aplicarle electricidad vibra actuando como estabilizador y regulador de frecuencias.

CONATEL: Acrónimo para Comisión Nacional de Telecomunicaciones. Organismo gubernamental Venezolano encargado de gestionar el espacio radioeléctrico del territorio nacional.

Compilador: Programa que se encarga de traducir un programa escrito en alto nivel hacia otro de más bajo nivel.

Datagrama: Fragmento de paquete que es enviado con suficiente información para que la red lo pueda encaminar hacia el receptor de manera independiente a los fragmentos restantes.

Diagrama de Clases: Esquema que representa las diferentes relaciones entre las Clases de una aplicación determinada.

Diagrama de Secuencia: Esquema que representa las diferentes Clases y la relación creada por los métodos invocados entre ellas.

EEPROM: Acrónimo de *Electrically-Erasable Programmable Read-Only Memory*. Tipo de memoria que puede ser programada, borrada y reprogramada eléctricamente. Usualmente están impresas dentro de microcontroladores, son capaces de retener los datos una vez que se apaga el equipo.

EIA-232/RS-232: Interfaz de comunicación usada por las computadoras. Puede trabajar en conexiones de una sola dirección o bidireccional, el tipo de conector a utilizar es el DB-25 o el DB-9.

Encriptar/Desenciptar: Procedimiento mediante el cual se logra el cifrado de una cadena de caracteres determinada. Desencriptar se refiere a la tarea inversa, obtener un código descifrado desde uno cifrado.

Enrutadores (*Routers*): Dispositivo electrónico que se encarga de distribuir paquetes entre los diferentes nodos de la red, interconectando redes completas o segmentos de estas.

Entidad-Relación: Diagrama que representa las distintas entidades y sus respectivas relaciones pertenecientes a una base de datos.

Entunelamiento: Proceso por el que se encapsulan los paquetes de datos que viajan dentro de una red.

Firmas Digitales: Son secuencias de bits que se obtienen mediante la aplicación de un algoritmo de cifrado asimétrico o de clave pública a un mensaje determinado y se adjunta al final de dicho mensaje como si fuese una firma autógrafa.

FLASH (Memoria): Memoria no volátil de computadoras que puede ser eléctricamente borrada y reprogramada. Es un tipo particular de memoria EEPROM borrada en bloques grandes.

FSK: Acrónimo para (*Frequency-Shift Keying*), es un tipo de modulación de frecuencia basada en valores binarios que varían entre valores determinados, la señal de modulación varia la frecuencia de la portadora, de modo que la señal modulada codifica la información asociándola a valores de frecuencia diferentes.

Get: Referente a obtener valores determinados.

GPL (Licencia General Públca GNU): licencia de software libre. Es la forma más popular y mejor conocida de el tipo de licencia *copyleft* fuerte que requiere que trabajos derivados de ella, estén disponibles bajo la misma licencia *copyleft*.

Hardware: Se refiere a los componentes físicos que componen a las computadoras.

Hilo (Hilo de Ejecución): Tarea en ejecución de un programa, usualmente simultánea y paralela a otros de igual condición.

IDE: Programa cuya función se basa en desarrollar código fuente de una lenguaje de programación determinado.

Instancia: Se dice que se crea una instancia cuando se crea una referencia a una Clase determinada.

IP: Acrónimo para *Internet Protocol*. Se basa en identificar de una manera lógica y jerárquica dispositivos (generalmente computadoras) conectados a dentro de una red que implemente dicho protocolo.

IP Spoofing: Creación de paquetes IP con direcciones IP falsificadas.

Java: Lenguaje de programación multiplataforma creado por Sun Microsystems.

JUnit: Conjunto de librerías que provee Java con las cuales se pueden crear pruebas para un determinado software.

Kernel: Núcleo del sistema operativo.

Librería: Conjunto de funcionalidades provistas por un lenguaje de programación.

L2PT: Acrónimo para *Layer 2 Tunneling Protocol*. Variación del protocolo IP donde se encapsula una trama L2PT en un paquete UDP.

Log(Archivo): Archivo que lleva la bitácora de algún evento determinado.

Logueado: Condición de un usuario dentro de una aplicación una vez que ha introducido su nombre de usuario y contraseña y resultaron ser ambos válidos para el acceso posterior.

LPT1: Puerto paralelo de las computadoras que sirve de interfaz para establecer comunicación hacia otros dispositivos externos.

Microtira: línea de transmisión donde se coloca sobre una cara de un plano dieléctrico una línea conductora, mientras del otro lado hay una conexión a tierra.

MVC: Acrónimo para *Model-View-Controller*. Patrón de diseño que separa el código fuente en tres componentes distintos: El *modelo*, donde se almacena toda la lógica del negocio; *el controlador*, donde se gestionan eventos tanto del controlador como de la vista y la *vista*, componente encargado de dibujar la interfaz gráfica.

Nibble: palabra de 4 bits.

NRZ: Acrónimo para *Non-Return- to-Zero*. Codificación digital basada en voltajes nunca con valores de cero en bits consecutivos de valor igual a uno. Mediante este tipo de codificación y asignando niveles de tensión a cada símbolo, se simplifica la tarea de codificar un mensaje.

Objeto: Abstracción de una unidad que representa un algo detalladamente, encapsulado ese algo mediante atributos.

Objeto de Dominio: Objeto cuyos atributos conservan la integridad unitaria mediante métodos get y set.

Oscilador LC: Es un tipo de oscilador que está formado por una bobina y un condensador conectado en paralelo, su funcionamiento se basa en almacenar energía en forma eléctrica en el condensador y en forma de campo magnético en la bobina.

Osciloscopios: Dispositivo electrónico de medición que se encarga de representar gráficamente las señales eléctricas y su variación en el tiempo.

OSI: Acrónimo para *Open System Interconnection*. Modelo (mas no una arquitectura) para las comunicaciones electrónicas que se basa en siete capas: Capa física, capa de enlace, capa de red, capa de transporte, capa de sesión, capa de presentación, capa de aplicación. Este modelo se creó para estandarizar las redes y crear compatibilidad entre ellas.

Paquete UDP: Acrónimo para *User Datagram Protocol*. Protocolo de comunicación basado en envío de datagramas IP en forma casi directa. UDP permite la transmisión de datagramas sin necesidad de confirmación de entrega o de recepción, uno de sus usos es la transmisión de video y sonido en tiempo real.

PC: Acrónimo para *Personal Computer*. Computador personal.

PKCS: Grupo de estándares de criptografía de clave pública. En total son 15 estándares creados por los laboratorios RSA.

PIC: Nombre de la familia de microcontroladores producida por *Microchip Technology*.

PIN: Se denomina PIN a cada conexión de un dispositivo electrónico.

Portable: Se refiere a una cualidad del software que tiene como característica permitir la ejecución del mismo en diversas plataformas.

Providencia Administrativa: Se refiere a las decisiones de los órganos de la administración pública nacional cuando no les corresponda la forma de decreto o resolución.

Proxy: Es un programa o dispositivo intermedio que realiza una acción de representación de otro.

Pseudocódigo: Representación escrita de un algoritmo en un lenguaje coloquial respetando reglas básicas de programación.

Query: Conjunto de condiciones y preguntas cuya utilidad es extraer información de las bases de datos.

RAM: Acrónimo para *Random Access Memory*. Utilizado normalmente como memoria temporal para almacenar resultados intermedios y datos similares no permanentes.

RISC: Acrónimo de *Reduced Instruction Set Computer*. Filosofía de diseño de CPU para computadoras que está a favor de conjuntos de instrucciones pequeños y simples que toman menor tiempo para ejecutarse.

Robusto: Cualidad del software que se basa en la ejecución a prueba de fallos del mismo.

Router: Dispositivo que determina la dirección que debe seguir información que viaja entre redes.

Script: Conjunto de instrucciones englobadas en un archivo usualmente de texto.

SHA (Secure Hash Algorithm): Familia de algoritmos diseñados por la Agencia de Seguridad Nacional (NSA) norteamericana a través de los cuales se obtiene un compendio de mensaje de tamaño fijo a partir de una secuencia de datos de entrada.

Sensibilidad: magnitud mínima de una señal de entrada para producir una señal de salida, habiéndose especificado una relación señal/ruido.

Servidor: Es una aplicación que realiza tareas en beneficio de otras aplicaciones llamadas clientes

Set: Asignar un valor.

Setear: Acción de asignar un valor a una variable determinada.

SETRA: Organismo gubernamental del estado Venezolano que se encarga de gestionar el parque automotor nacional.

SIP: Acrónimo para *System-In-Package*. Tecnología que encapsula circuitos integrados en un paquete dentro de un mismo módulo.

Software: Parte lógica de los dispositivos electrónicos basada en códigos que le indican al hardware que debe hacer.

SSL: Acrónimo para *Secure Socket Layer*. Protocolo de encriptado desarrollado por la empresa Nestcape que codifica datos enviados por Internet. Se basa en criptografía asimétrica y el uso de certificados digitales.

Swing: Evolución de AWT. Además de poseer todas las características de AWT Swing es capaz de operar en diferentes plataformas, punto no cubierto por AWT, los componentes de Swing soportan más características funcionales y gráficas.

Tabla: Abstracción lógica usaba en bases de datos que describe a los atributos de una entidad.

Test: Palabra usada para describir las pruebas unitarias implementadas en el presente trabajo de grado.

Tickets: Constancia impresa de paso o permiso de él por un punto determinado.

Tipado(Tipificado): Característica de un lenguaje de programación que se basa en no permitir violaciones de tipos de variables, por ejemplo una variable declarada como un entero no puede usarse como un arreglo.

TTL: Acrónimo para *Transistor-Transistor Logic*. Tecnología de construcción de circuitos electrónicos digitales, cuya tensión característica se halla entre los 4,75 V y los 5,25 V.

UART: Acrónimo para *Universal Asynchronous Receiver Transmitter*. Componente utilizado por ciertos sistemas digitales basados en microprocesador para convertir los datos a transmisión en paralelo, debido a que son enviados desde el CPU en serie. El fin último es establecer comunicación con otro sistema externo.

USB: Acrónimo para *Universal Serial Bus*. Bus externo que soporta tasas de transferencia de datos de 12 Mbps.

WAMP: Acrónimo para *Windows-Apache-MySQL-PHP*. Aplicación derivada de LAMP (*Linux* en lugar de *Windows*) que empaqueta los programas anteriores (menos el sistema operativo), en una sola aplicación de fácil instalación.

CAPÍTULO IX

APÉNDICE

APÉNDICE A. Información Adicional

9.A.1 Java

Java es un lenguaje de programación concebido originalmente por James Gosling, Patrick Naughton, Chris Warth, Ed Frank y Mike Sheridan en 1991. Tardó 18 meses en ser desarrollada su primera versión ("*Oak*"); diseñada en realidad para dispositivos electrónicos, microondas y para aquel entonces, la anacrónica televisión interactiva. El impulso inicial fue la propuesta de un lenguaje independiente de la plataforma; uno que trabajar con distintos procesadores, bajo distintos entornos y que a la vez fuera portable. Pero luego aparecería el factor definitorio del establecimiento de Java en la historia de los lenguajes de programación. Con la llegada de la World Wide Web, se establecía la importancia de aquel primer impulso: la Web necesitaba también de programas que fuesen portables.

La solución resultó ser lo que conocemos como la Máquina Virtual de Java, capaz de recibir código intermedio referido como *Java Bytecode* que a su vez es generado por un compilador de lenguaje Java; siendo aquí donde el programador tiene potestad de acción y decisión. Lo que SUN Microsystems sugiere concretamente, es tomar para ellos la responsabilidad de la compilación para las diferentes plataformas. Es labor de ellos desarrollar una máquina virtual (en código nativo) específica para cada sistema operativo, mientras solo queda del programador la elaboración de la aplicación final. Esta máquina virtual es parte de lo que se conoce como la Plataforma Java; una serie de clases y recursos, entre ellos el *Java Runtime Enviroment*, un compilador JIT, extensas librerías de código precompilado, y otras herramientas que proveen facilidades para el programador en el desarrollo de aplicaciones complejas. Java al final, cuenta con las siguientes propiedades:

- Simple. Java hereda gran parte de su sintaxis y características orientadas a objetos de C/C++, sin embargo, una gran cantidad de de conceptos originales de estos lenguajes se han eliminado o implementado de forma más clara o accesible. Java se hace fácil de aprender si se entienden los conceptos de la programación orientada a objetos y se hace fácil de dominar si se tiene experiencia programando.
- Orientado a objetos. Consecuencia de esa misma influencia de sus predecesores, Java adopta una aproximación pragmática altamente útil que incorpora los objetos y los tipos primitivos de forma funcional; para obtener un modelo, inclusive a veces criticado, sencillo, de fácil ampliación de alto rendimiento.

- Robusto. Debido a su naturaleza interpretada, Java ofrece ciertas propiedades que lo aventajan sobre otros lenguajes. En primer lugar, hay restricciones en áreas claves como lo son, por ejemplo, el manejo de memoria. El programador está libre de esta responsabilidad debido a que Java se encarga de la gestión de ella automáticamente. Luego, el manejo de excepciones orientada objetos permite que todos los errores de ejecución puedan (y deban) ser gestionadas por el programa. Es difícil que en Java aparezcan errores difíciles de localizar; el programa rara vez se comportará de manera no predecible a una condición particular.
- Programación multi-hilo. De acuerdo con los requerimientos del mundo real, Java permite la escritura de programas que hagan varias cosas a la vez. El programador debe preocuparse más del comportamiento específico del programa que del sistema multitareas que lo pueda estar rodeando.
- Arquitectura Neutral. El éxito que ha tenido Java a lo largo de la historia, es causa directa de esta misma propiedad. La portabilidad de código, además de su longevidad siempre fueron atributos cardinales en el desarrollo de la tecnología. La independencia del sistema operativo, los procesadores, recursos básicos y actualizaciones sobre ellos, libera a Java del mal funcionamiento. *Escribir una vez, ejecutar en cualquier sitio*, es la premisa fundamental de esta tecnología.
- Interpretado pero de alto rendimiento. Aunque Java es percibido como más lento y con requerimientos de memoria mayores a los de sus contrapartes nativas, la verdad es que su velocidad de ejecución ha mejorado desde la introducción de la compilación JIT y las optimizaciones de la máquina virtual que SUN Microsystems ha introducido con el tiempo. Actualmente el desempeño cuando ocurre la compilación JIT es menor al de lenguajes compilados pero no significativamente y en *benchmarks* de velocidad de ejecución, Java es comparable con C o C++y ha resultado inclusive superior.
- Distribuido. Para el momento de concepción original de Java ("Oak"), ya se tenían las primeras facilidades que permitían tener objetos simulados en computadores distintos permitiendo ejecutar procedimientos remotamente. Hoy este paquete de interfaces se conoce como RMI.

Java sin embargo ha sido criticado por suprimir ciertas características como la sobrecarga de operadores y la herencia múltiple para "proteger al programador de sí mismo", simplificar el

lenguaje y evitar errores. El tiempo decidirá si estas, entre otras críticas, resultan definitorias en el destino de Java como lenguaje de programación.

9.A.2 Manejador de Bases de Datos

Históricamente hablando, son ya casi 40 años desde que aparece el primero de una serie de estudios describiendo el Modelo Relacional de Datos por autoría del Dr. Edgar F. Codd como evolución lógica los modelos jerárquico y en red ingeniados 10 años antes . Codd los criticaba por confundir la descripción abstracta de la información con los mecanismos de acceso a ella; la razón de ello es que los primeros existían de acuerdo a la tecnología del momento. El modelo relacional tomó una perspectiva más teórica, argumentando que el software y el hardware habrían eventualmente de alcanzarlo. Se sabe que se han hecho contribuciones de gran tamaño a la investigación de las bases de datos y particularmente a las relacionales; pero hoy ninguna de ellas se considera tan fundamental o significativa como los trabajos originales de Codd.

Una base de datos es una colección de datos relacionados con las siguientes consideraciones:

- Representa aspectos del mundo real cuyos cambios son reflejados en la base de datos.
- Es una colección coherente de datos con significados inherentes. Datos aleatorios reunidos no constituyen una base de datos.
- Tiene un propósito específico. El diseño, construcción y relleno de datos de una base de datos se planifica para cumplir ese propósito.

Luego de ello, se establece que un sistema de bases de datos está distinguido de un enfoque por ficheros por las siguientes características:

- Existe un catálogo o diccionario de datos. Los datos no existen siendo libres de estructuras ni restricciones; ellas se encuentran en lo que se define como el *catálogo* del sistema. La información en él se denomina meta-datos y describe la estructura de la base de datos.
- Hay abstracción de datos. Hay una representación conceptual que le permite a los usuarios saber cómo existen los datos, sin entrar en detalles de almacenamiento ni la manera en que las operaciones son implementadas sobre ellos.
- Hay independencia entre programas, datos y operaciones. Un programa puede ser modificado sin que haya que hacer algún cambio sobre los otros módulos. Lo mismo para los demás. Esto es posible gracias a la propiedad de abstracción.

- Se permiten múltiples vistas a los usuarios. Una base de datos parte del principio de que hay muchos usuarios haciendo uso de ella; igualmente debe ser posible establecer una perspectiva particular de ella para estos.
- Se comparten los datos entre múltiples transacciones. Varios usuarios actuando simultáneamente sobre una base de datos requiere que haya algún mecanismo de control de concurrencia que permita que al actualizar un mismo dato, por ejemplo, esto se haga correctamente.

La serie de propiedades que garantizan la fiabilidad en el procesamiento de transacciones se conocen como ACID. Una transacción es cada operación lógica que ocurre dentro del contexto de la base de datos y por estas propiedades se dice que debe ser Atómica (o todas las tareas de la transacción son completadas o ninguna lo es), Consistente (la base de datos está en un estado legal antes y después de la transacción; no se pueden romper restricciones de integridad durante ella), Aislada (una transacción no puede ver datos de otra en ejecución) y Durables (una vez ejecutada la transacción, esta persiste y no puede ser deshecha).

Hay una serie de condiciones que un sistema de gestión de bases de datos puede ofrecer para ayudarles al administrador, diseñador y usuarios de ella en su uso:

- Control de redundancia. El problema con la redundancia está en la duplicación de trabajo (una actualización debe hacerse en tantos datos hayan redundantes) y la posibilidad de inconsistencias (una actualización ocurre sobre unos datos y no sobre otros) cuando nos referimos a la redundancia no controlada o planificada.
- Restricción de acceso. No todos los usuarios de una base de datos deberían ser capaces de ejecutar las mismas acciones sobre ella.
- Múltiples interfaces de usuario. Particularmente porque puede ocurrir que varios usuarios difieran en niveles de conocimiento. Puede ofrecerse una interfaz gráfica para unos, por ejemplo, y una por línea de comandos para otros.
- Representación de relaciones complejas entre los datos. Una base de datos puede contener multitud de datos que estén relacionados de variadas maneras entre sí. El sistema debe ser capaz de obtener y actualizar eficiente y rápidamente datos que estén relacionados entre sí.
- Restricciones de integridad. Una restricción de integridad sencilla, es el tipo de dato que puede haber en un campo, sea de tipo entero, o un carácter o un valor booleano.

La mayoría de los manejadores de bases de datos ofrecen alguna gama de restricciones de integridad y le permiten al usuario que defina las suyas.

- Suministro de Respaldo y Recuperación. Debido a la importancia de la data almacenada y la responsabilidad del manejador que pueda reposar sobre él, es importante que haya algún mecanismo de recuperación y de despliegue de copias de seguridad para compensar por los posibles fallos de software y hardware que ocurran en el futuro.

APÉNDICE B. Circuitos

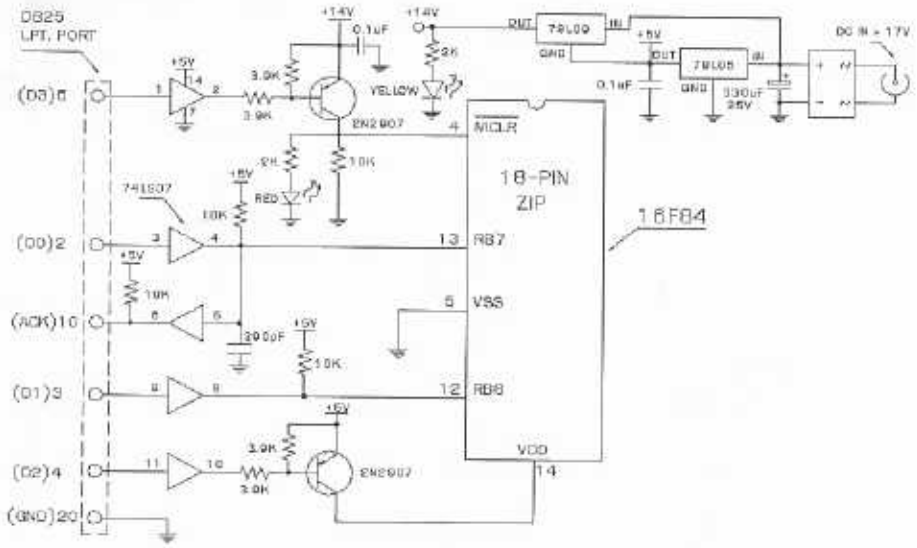

Figura 9.B.1. Circuito programador para microcontrolador PIC.
http://linuxgazette.net/issue79/sebastian.html

APÉNDICE C. Diagramas de Secuencias

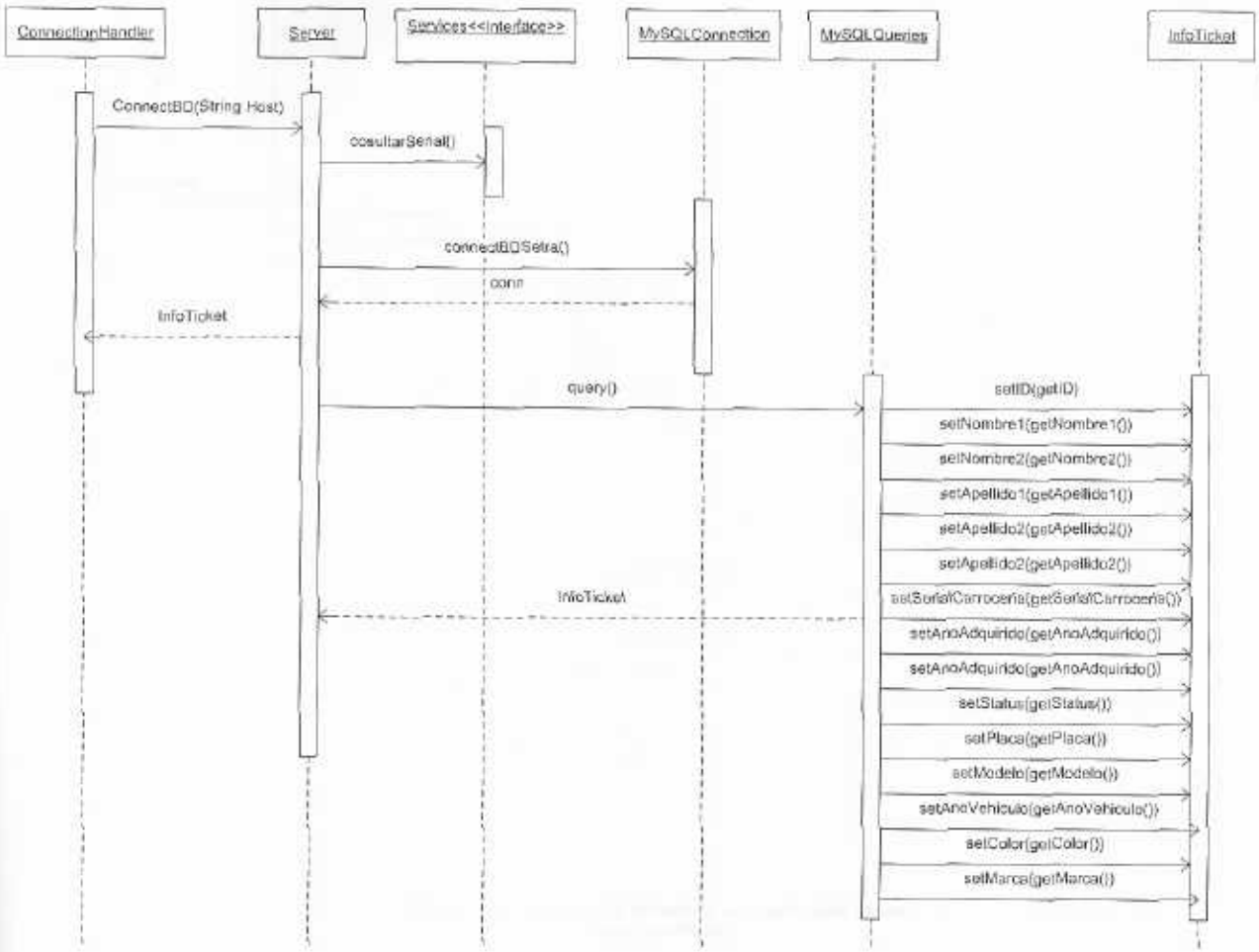

Figura 9.C.1. Diagrama de Secuencias para el Primer Diagrama de Clases (servidor).
Elaboración Propia.

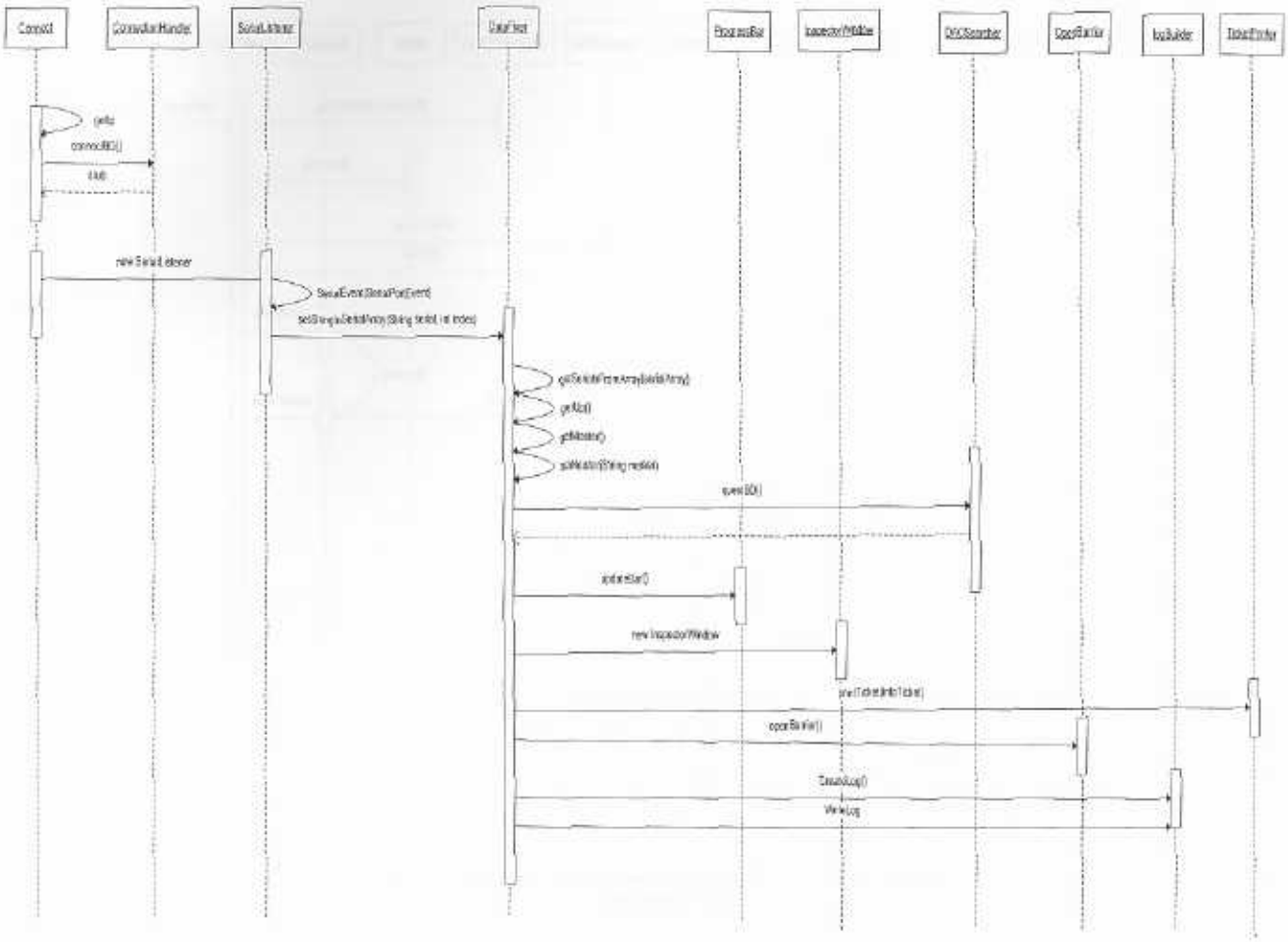

Figura 9.C.2. Diagrama de Secuencias para aplicación Cliente.
Elaboración Propia.

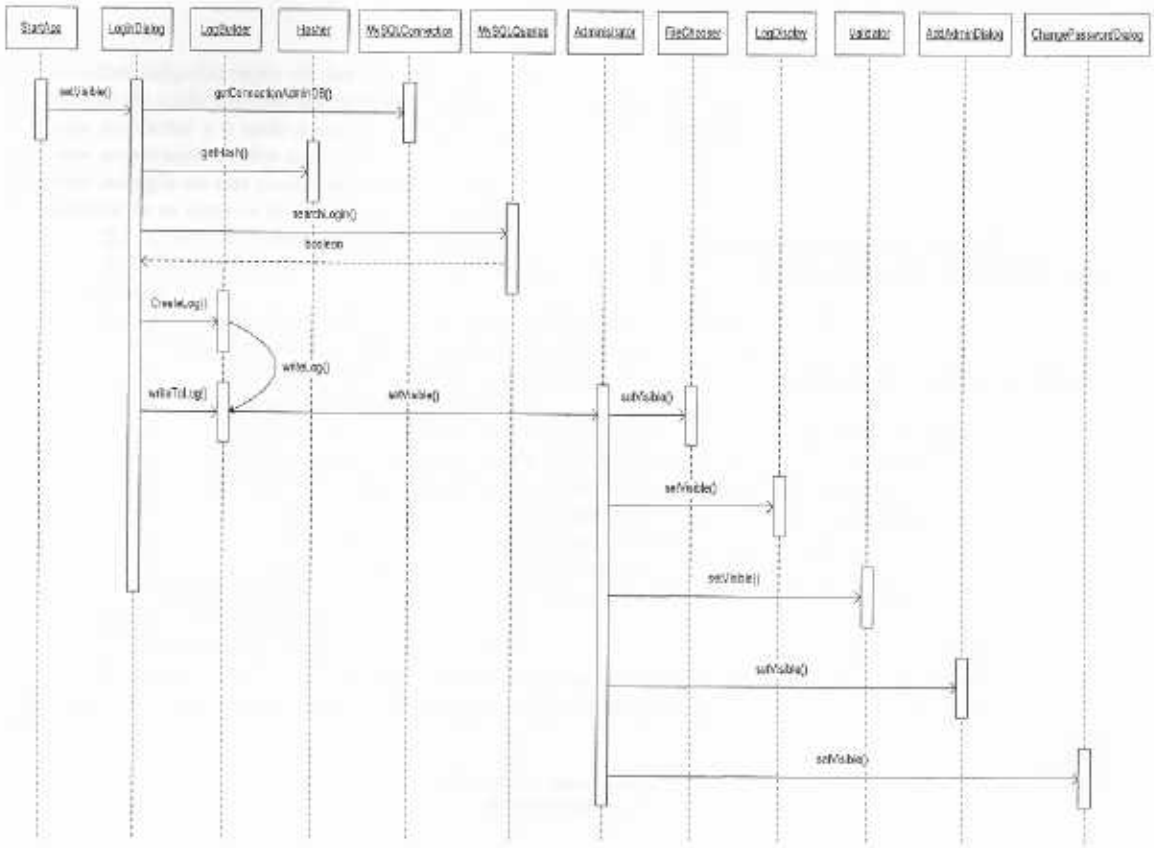

Figura 9.C.3 Diagrama de Secuencias de la aplicación de administración.
Elaboración Propia.

APÉNDICE D. Pseudocódigo del microcontrolador PIC

Método CorrectorDeCodigo(Arreglo de dos posiciones de seriales){

1. Obtener de cada posición del arreglo, los seriales sin el prefijo "***", ni el sufijo "+++".
2. *Crear un Vector y a cada posición del Vector asignarle el serial obtenido en el paso 1.*
3. *Crear un Arreglo de dos posiciones.*
4. *Crear Arreglo de dos posiciones de tipo entero.*
5. *Mientras no se alcance el tamaño del Vector, hacer:*
    - *5.1. Crear un Arreglo nuevo, cada elemento del Vector asignarlo a una posición del arreglo.*
    - *5.2. Obtener un número pseudoaleatorio, dicho número corresponde a la posición del arreglo creado en el paso 4.1.*
    - *5.3. Asignar el arreglo en esa posición a un String de comparación.*
    - *6. Desde 0 hasta el tamaño del arreglo creado en la posición 5.1 :*
        - 6.1 Comparar cada posición del arreglo con el String de comparación, sin son iguales incrementar el grado de importancia a uno.
        - *6.2 Si el número de la iteración es igual al tamaño del arreglo menos uno y el grado de importancia es menor al tamaño del vector entre dos, o, Si el número de la iteración es igual al tamaño del arreglo menos uno y el grado de importancia es igual a cero.*
            - *6.2.1. Obtener un nuevo número aleatorio y asignar como nuevo String de comparación al contenido en el arreglo del paso 5.1 en la posición aleatoria.*
            - 6.2.2. Comenzar de nuevo el punto 5.
    - *7. Asignar el String del arreglo del paso 6.2.1 al arreglo creado en el paso 3 en la posición correspondiente al índice de importancia.*
    - 8. Remover la primera posición del Vector.
    - *Fin mientras paso no se alcance el tamaño del Vector 5.*
9. *Comparar grados de importancia, el String q tenga mayor grado de importancia se selecciona como el serial del vehículo*

}

Figura 9.D.1. Pseudocódigo.
Elaboración propia.

APÉNDICE E. Fotos de Circuitos

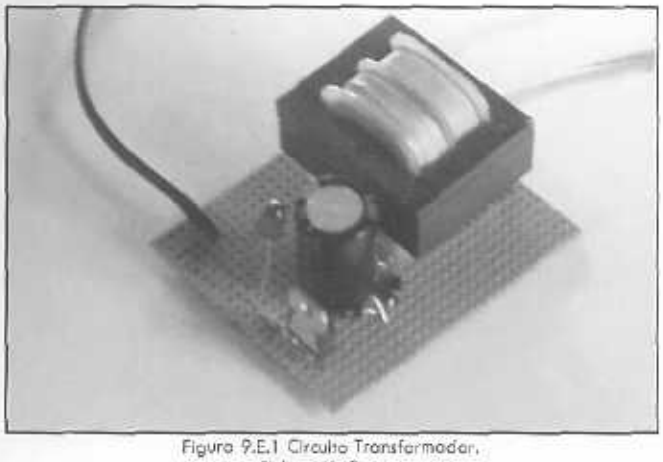

Figura 9.E.1 Circuito Transformador.
Elaboración Propia.

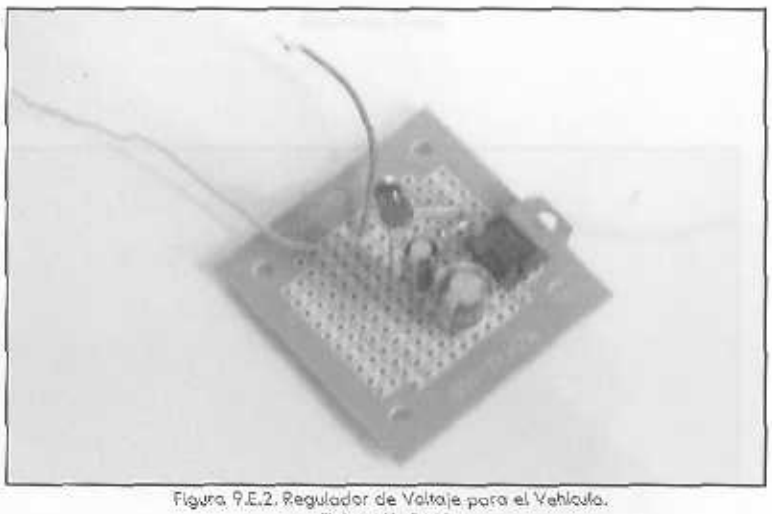

Figura 9.E.2. Regulador de Voltaje para el Vehículo.
Elaboración Propia.

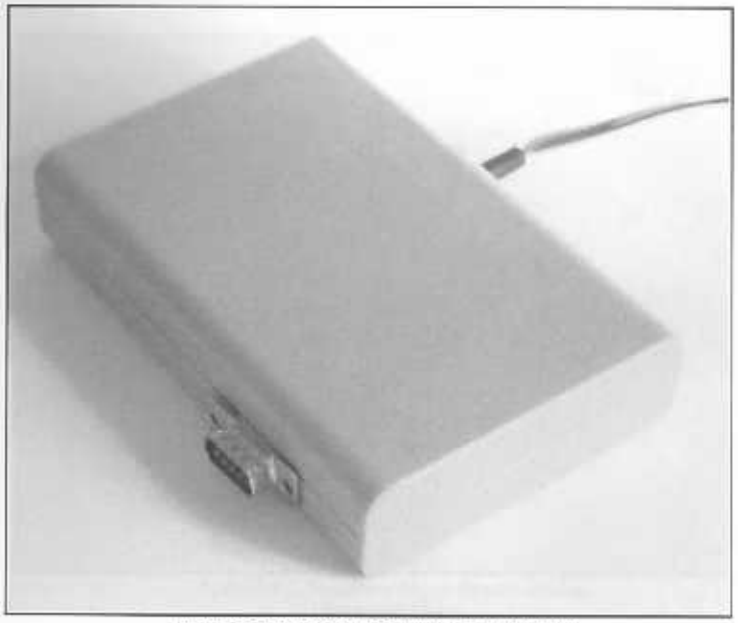

Figura 9.E.3. Dispositivo Receptor (Empaquetado).
Elaboración Propia.

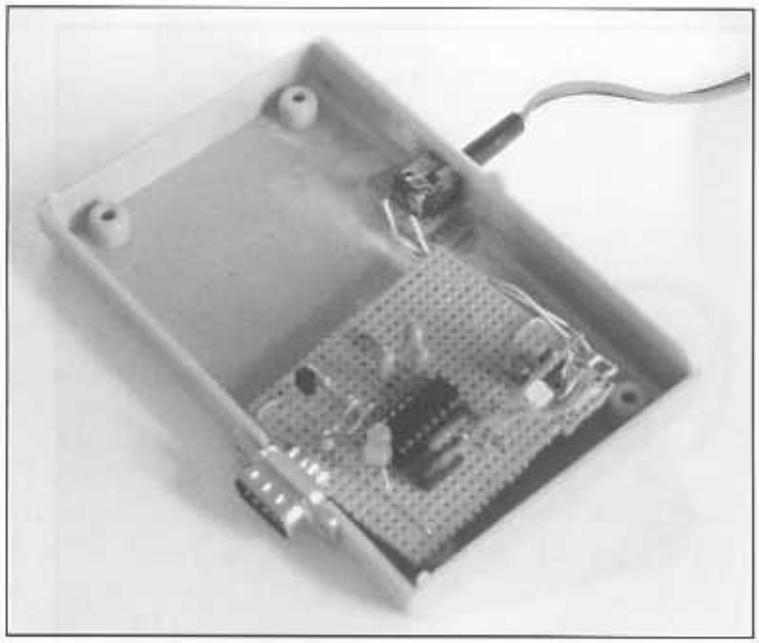

Figura 9.E.4. Dispositivo Receptor (Abierto).
Elaboración Propia.

APÉNDICE F. Iteraciones

x	Descripción	Producto
1era	Estudio en organismos CONATEL y SETRA. Diseño y desarrollo de Dispositivos de Transmisión y reguladores de Voltaje. Programación de PIC. Lecturas desde Puerto Serial.	Frecuencias disponibles para uso. Atributos para estructuras de datos. Dispositivos funcionales para Transmisión y Recepción. Interfaz para comunicación entre la computadora y el dispositivo de recepción.
2da	Desarrollo de Base de Datos y estructuras para conexión RMI.	Estructuras Lógicas de Almacenamiento. Clases, atributos y demás estructuras para conexión RMI.
3ra	Integración de Desarrollo Previo. Implementación VPN.	Sistema funcional para verificación. Conectividad segura a través de VPN.
4ta	Desarrollo de interfaz gráfica. Implementación de testigo de paso y mecanismo de impresión de tickets. Bitácora de Flujo Vehicular. Cifrado.	Sistema funcional de verificación completa (permiso de paso, interfaces gráficas e impresión de tickets) y registro de vehículos.
5ta	Desarrollo de Aplicación de Administración. Pruebas finales.	Aplicación de administración.

Tabla 9.F.1 Planificación de Pruebas Unitarias.
Elaboración Propia

APÉNDICE G. Planificación de Pruebas Unitarias

ID	Métodos Pruebas unitarias.	Pasos	PostCondición	Status
1	getSerialFromArray(byte[])	Crear una ráfaga de 7 bits concecutivas, y pasarsela como parámetro al método getSerialFromArray(byte[]) secuencialmente mediante un ciclo.	Se debe devolver un valor booleano positivo cada vez que se setea una ráfaga completa de 7 bits.	Desarrollada.
2	getSerialFromArray(byte[])	Crear ráfagas de 7 bits de caracteres basura y pasarselas como parámetro al método getSerialFromArray(byte[]) secuencialmente mediante un ciclo.	Se debe ignorar cada carácter basura que se introduce como parámetro.	Desarrollada.
3	connectBD(String)	Pasar como parámetro al método connectBD(String) el IP de la máquina servidor, luego pasar como parámetro ráfagas de 7 bits que representan un serial previamente introducido en la base de datos.	Se debe establecer la conexión y obtener el objeto de dominio con todos sus datos seteados.	Desarrollada.
4	connectBD(String)	Pasar como parámetro al método connectBD(String) un IP distinto al servidor de la apliación.	La conexión no se establece y nuevamente se le notifica al usuario que introduzca un IP válido.	Desarrollada.
5	connectBD(String)	Pasar como parámetro al método connectBD(String) el IP de la máquina servidor, luego pasar como parámetro ráfagas de 7 bits que no representan ningun serial almacenado en la base de datos.	Se establece la conexión y no se obtiene ninguna respuesta del servidor, manteniéndose la aplicación a la espera de un serial válido.	Desarrollada.
6	searchLogin(String,String)	Pasar como parámetro al metodo searchLogin(String,String) un login y password previamente introducido en la base de datos.	Se debe obtener un resultado booleano de valor true, indicando que la validación fue exitosa.	Desarrollada.
7	searchLogin(String,String)	Pasar como al método searchLogin(String,String) parámetro un login y password inválidos.	Se debe obtener un resultado booleano de valor false, indicando que la validación no fue exitosa.	Desarrollada.
8	changePassword(String,String)	Acceder al sistema con un login y password previamente introducido en la base de datos. Pasar como parámetro al método changePassword(String,String) un password igual al usuario administrador logueado en ese momento en la aplicación, y pasar como segundo parámetro un password nuevo.	Se debe obtener como resultado un valor booleano de valor true que indica que se realizó con éxito la operación.	Desarrollada.
9	changePassword(String,String)	Acceder al sistema con un login y password previamente introducido en la base de datos. Pasar como parámetro al método changePassword(String,String) un password distintol al usuario administrador logueado en ese momento en la aplicación, y pasar como segundo parámetro un password nuevo.	Se debe obtener como resultado un valor booleano de valor false que indica que la operación no se ejecutó.	Desarrollada.
10	deleteAdministrator(String)	Acceder al sistema con un login y password previamente introducido en la base de datos. Pasar como parámetro al método deleteAdministrator(String) un login previamente introducido en la base de datos y distinto al usuario administrador en ese momento logueado.	Se debe obtener como resultado un valor booleano de valor true que indica que se realizó con éxito la operación.	Desarrollada.

Tabla 9.G.1 Planificación de Pruebas Unitarias.
Elaboración Propia

APÉNDICE H. Ejemplos de Archivos

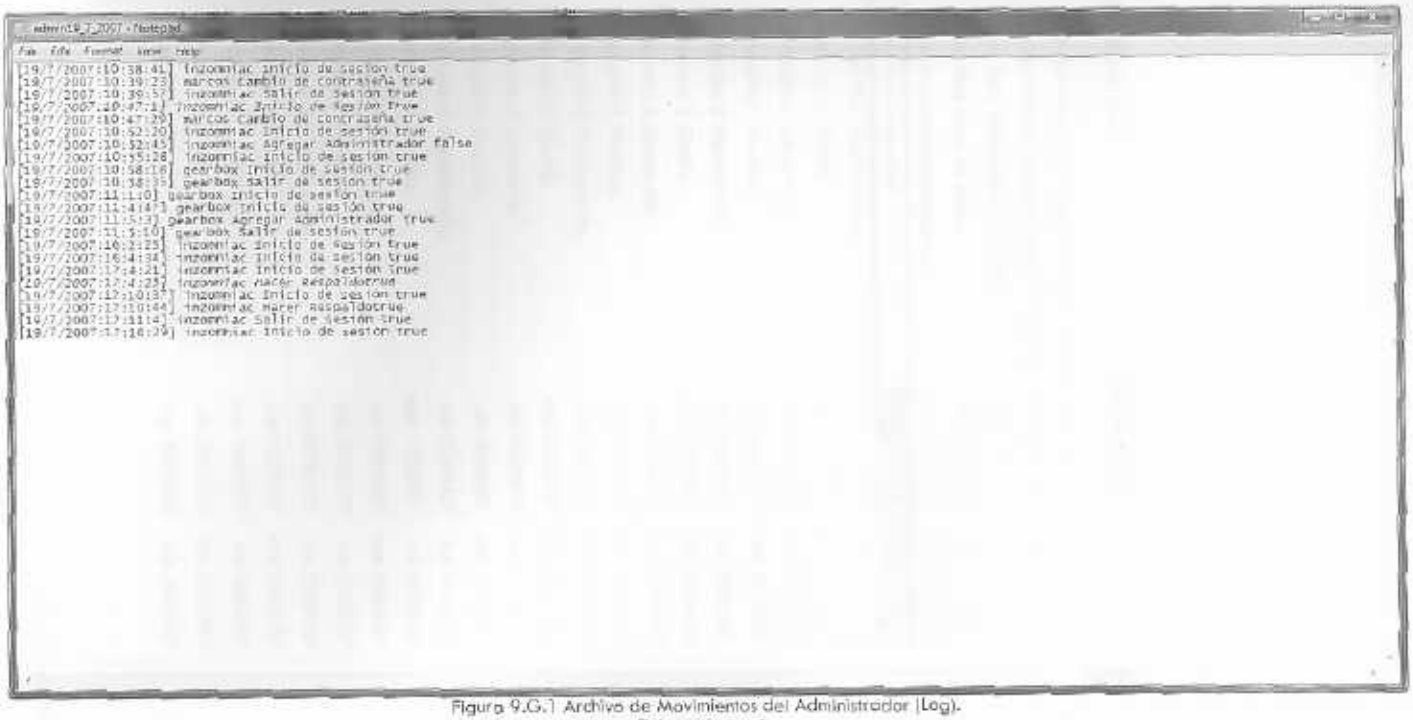

Figura 9.G.1 Archivo de Movimientos del Administrador (Log).
Elaboración Propia.

Figura 9.G.2 Archivo de Registro de Tránsito. (2 Entradas, Hora de entrada, Serial del Vehículo, Nombre y Apellido, Cédula de Identidad, Estatus)
Elaboración Propia.

APÉNDICE I. Planificación de Desarrollo del Proyecto

Diagrama de Gantt

	Nombre de tarea	Comienzo	Fin
1	Levantamiento de requerimientos SETRA	lun 20/11/06	mar 28/11/06
2	Levantamiento requerimientos CONATEL	mié 29/11/06	jue 07/12/06
3	Lev. Req. dispositivos TX y RX	vie 08/12/06	jue 28/12/06
4	Lev. Req base de datos	vie 29/12/06	lun 01/01/07
5	Lev Req Microcontroladores	mar 02/01/07	jue 04/01/07
6	Lev Req Seguridad	vie 05/01/07	mar 09/01/07
7	Iteración 1. Planificación.	mié 10/01/07	lun 29/01/07
8	Iteración 1. Diseño.	mar 30/01/07	vie 09/02/07
9	Iteración 1. Desarrollo.	lun 12/02/07	jue 08/03/07
10	Iteración 1. Pruebas	jue 08/03/07	vie 16/03/07
11	Iteración 2. Planificación.	vie 16/03/07	mar 27/03/07
12	Iteración 2. Diseño.	mié 28/03/07	mié 04/04/07
13	Iteración 2. Desarrollo.	jue 05/04/07	lun 16/04/07
14	Iteración 2. Pruebas.	lun 16/04/07	jue 19/04/07
15	Iteración 3. Planificación.	jue 19/04/07	mar 01/05/07
16	Iteración 3. Diseño.	mié 02/05/07	mar 08/05/07
17	Iteración 3. Desarrollo	mié 09/05/07	jue 17/05/07
18	Iteración 3. Prueba.	vie 18/05/07	mar 22/05/07
19	Iteración 4. Planificación.	mié 23/05/07	lun 11/06/07
20	Iteración 4. Diseño.	mar 12/06/07	vie 15/06/07
21	Iteración 4. Desarrollo	lun 18/06/07	mar 26/06/07
22	Iteración 4. Pruebas.	mié 27/06/07	vie 29/06/07
23	Iteración 5. Planificación.	lun 02/07/07	jue 12/07/07
24	Iteración 5. Diseño.	jue 12/07/07	mar 24/07/07
25	Iteración 5. Desarrollo.	mar 24/07/07	vie 24/08/07
26	Iteración 5. Pruebas.	vie 24/08/07	mié 05/09/07
27	Desarrollo Tono	lun 20/11/06	jue 20/09/07

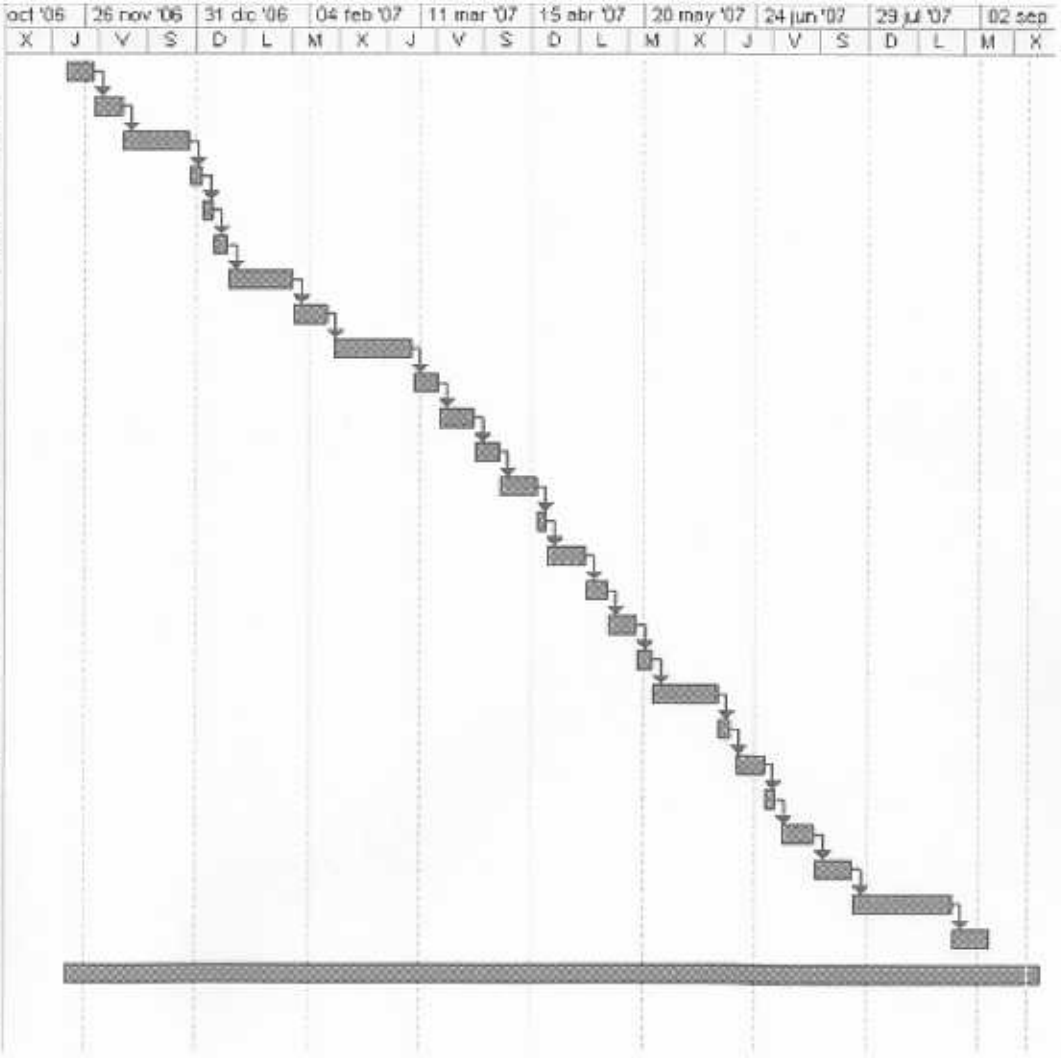

Figura 9.G.1. Planificación de Desarrollo del Proyecto.
Elaboración Propia.