TECNOLÓGICO DE MONTERREY®

INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY

CAMPUS CIUDAD DE MÉXICO

División de Ingeniería y Arquitectura

Departamento de Mecatrónica

PROYECTOS DE INGENIERÍA II E00881-02

Adquisición del valor de la velocidad en el momento de una colisión para compañías aseguradoras.

PROFESOR: Ing. José Vicente Quintanilla Bonifaz

Integrantes del equipo:

Christiane Parlange Uribe	971010
Fredy Merlín Luis	994509
Mario A. Sánchez Peralta	995391
Dulce Vela Rabadán	952329

ASESOR: Dr. Raúl Crespo Saucedo

Mayo, 2007

TECNOLÓGICO DE MONTERREY®

INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY
CAMPUS CIUDAD DE MÉXICO

Abril 2007
México, D.F.

Adquisición del valor de la velocidad en el momento de una colisión para compañías aseguradoras.

INTEGRANTES:

Christiane Parlange Uribe	971010
Fredy Martín Luis	994509
Mario A. Sánchez Peralta	995391
Dulce Vela Rabadán	952429

PROFESOR
Dr. Raúl Crespo Santiago

ASESOR
Ing. Jose Vicente Quintanilla

PROBLEMÁTICA:

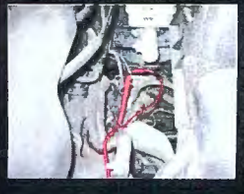
El exceso de velocidad es una de las causas más comunes en los accidentes automovilísticos. Para una compañía aseguradora, es de gran utilidad saber a qué velocidad conducía el asegurado antes de accidentarse para deslindarse de responsabilidades

OBJETIVOS:

- Desarrollar un prototipo que realice las siguientes funciones:
  - Tomar la velocidad de un automóvil en el momento en el que éste tiene una colisión.
  - Procesar la información y enviarla a un teléfono celular vía Bluetooth
  - Enviar esta información junto con el número de póliza vía SMS a un servidor.
  - Recuperar de la base de datos los datos generales del asegurado y del auto.

PRUEBAS Y RESULTADOS:

Adquisición Velocidad → Sensor velocidad



Detección impacto → acelerómetro

Comunicación Bluetooth (Sistema Embebido) → Embedded Blue

Recuperación datos → Base Datos MySQL

Interfaz gráfica → Java

METODOLOGÍA:

CONCLUSIONES

- Se logró el diseño e implementación de los siguientes módulos:
  - Adquisición Velocidad
  - Procesamiento de información
  - Comunicación vía Bluetooth al teléfono celular.
  - Comunicación con la Base de Datos vía SMS
    Recuperación de datos

INDICE

I. Introducción 4

1.1 Objetivos y Metas 5
1.2 Estado del Arte 6
1.3 Áreas de desarrollo 6

II. Problemática 7

III. Marco Teórico 9

3.1 Microprocesadores 9
3.2 Sensor de velocidad 11
3.3 EmbeddedBlue 500 13
3.4 LM7805 24
3.5 Acelerómetro 25
3.6 SMS 28

IV. Propuesta de Solución 37

V. Resultados y Pruebas 40

5.1 EmbeddedBlue Transceiver AppMod: Conectividad Bluetooth 40
5.2 Sensor velocidad 44
5.3 Convertidor Frecuencia-Voltaje 50
5.3.1 LM331 50
5.3.2 LM2907 51
5.4 Convertidor Analógico Digital 53

5.5 Sistema de procesamiento 54

5.5.1 Sistema mínimo 54

5.6 Acelerómetro 57

5.6.1 Teoría 'g' 60

5.6.2 ADXL278 63

5.7 Diseño de hardware 64

5.8 Programación 67

VI. Conclusiones y trabajo futuro 70

VII. Referencias 73

VIII. ANEXOS 74

8.1 Programación Ensamblador Proview 75

8.2 Programación JAVA 82

8.3 Financieros 103

I. INTRODUCCIÓN

El proyecto que a continuación se presenta, es el resultado de la implementación de tecnologías existentes (Tecnología inalámbrica, MEMS, teléfonos celulares, etc) para ayudar a las compañías aseguradoras a tomar una mejor decisión cuando algún cliente sufre un accidente automovilístico a causa del exceso de velocidad.

Nuestro proyecto consiste en ir midiendo constantemente (cada segundo) la velocidad a la que conduce el asegurado para que en caso de que se presente una colisión, nuestro sistema detecte la velocidad previa al impacto y sea almacenada en una memoria como también un historial sobre las velocidades previas a las que conduce el asegurado. Posteriormente el ajustador por medio de un celular con tecnología Bluetooth podrá acceder a la información almacenada en el sistema mínimo antes mencionado. Otra de las cualidades de nuestro sistema es que el ajustador agilizará el proceso de captura de información referente a la póliza, número de motor, entre otros datos. Esta información será descargada por medio de un dispositivo Bluetooth del sistema mínimo como también vía SMS (Short Message Service) desde la base de datos de la compañía aseguradora y se desplegará la información del cliente en la pantalla del teléfono celular del ajustador. Con esto se pretende, que el ajustador tenga mayor información para determinar si la causa del accidente se debió al exceso de velocidad y así poder determinar o deslindar responsabilidades a la compañía aseguradora.

Como consecuencia, nuestro proyecto está enfocado al mercado de las compañías aseguradoras, donde pretendemos que tenga un carácter preventivo de accidentes obligando al conductor a respetar la velocidad establecida en las vialidades.

1.1 Objetivos y Metas

El objetivo a cumplir en el desarrollo de este proyecto a lo largo de Proyectos de Ingeniería I y II es construir un prototipo que detecte la velocidad de un automóvil en el momento de una colisión. Para lograr esto, es necesario seguir los siguientes pasos a lo largo del desarrollo del proyecto:

- Identificar las tecnologías posibles para la solución específica del proyecto.

- Desarrollar, e investigar los bloques necesarios para poder crear un prototipo que cumpla con las características y funcionalidades desarrolladas a lo largo de este escrito.

- Comparar las posibles soluciones tecnológicas encontradas de acuerdo a la compatibilidad, precio, recursos necesarios y complejidad en la implementación.

- Investigar las técnicas de medición de velocidad en los automóviles así como el funcionamiento de los dispositivos necesarios para su implementación.

- Realizar pruebas de los dispositivos de comunicación elegidos para comprender su funcionamiento.

- Determinar el software de programación para dispositivos móviles y documentarse del mismo.

1.2 Estado del Arte

El proyecto se basa en la transmisión de información, con tecnología de conectividad inalámbrica, para rangos cortos de distancia y con alta velocidad en este caso tecnología Bluetooth.

1.3 Áreas de Desarrollo

1. Sistemas Digitales
2. Ingeniería automotriz
3. Programación de PDA's en Java
4. Microprocesadores
5. Electrónica
6. Electrónica de instrumentación

II. PROBLEMÁTICA

Una de las causas más comunes en los accidentes automovilísticos es el exceso de velocidad. Conocer la velocidad a la que conducía el cliente instantes antes de la colisión, es información muy valiosa para la compañía aseguradora, para poder determinar o deslindarse de responsabilidades, en caso de que el asegurado viaje a exceso de velocidad.

Determinar la culpabilidad del asegurado implica un ahorro significativo a las empresas aseguradoras, ya que el asegurado tendrá que pagar el deducible de su automóvil para hacer válida la garantía del seguro. Según un estudio realizado para la NHMRC (National Health and Medical Research Council) de la Universidad de Adelaide, el exceso de velocidad es un factor determinante en el 23% de los accidentes automovilísticos. Cada año más de 1.000 personas fallecen como consecuencia de llevar una velocidad inadecuada o excesiva.[1]

Estudios de la Universidad de Adelaide en Estados Unidos muestran que el riesgo de que un accidente cause una muerte o heridas se incrementa cuando hay un exceso de velocidad del establecido. Se calculó el riesgo relativo con respecto a velocidades superiores a los 60 km/hr:

[1] Kloeden CN, McLean AJ, Moore VM, and Ponte G (1997) Travelling Speed and the Risk of Crash Involvement, NHMRC Road Accident Research Unit, The University of Adelaide

Velocidad (km/hr)	Riesgo Relativo
60	1
65	2
70	4.16
75	10.60
80	31.81
85	56.55

Tabla 1: Velocidad vs. Riesgo Relativo [2]

Por ejemplo, un conductor viajando a 65 km/hr en una zona de 60 km/hr tiene el doble de posibilidad de sufrir un daño físico o un accidente mortal que aquel conductor que maneje a la velocidad establecida. Y a una velocidad de 70 km/hr, en una zona de 60 km/hr, el conductor tiene 4 veces más probabilidades de accidentarse.

[2] ibid

III. MARCO TEÓRICO

3.1 Microprocesador

Es un circuito electrónico que actúa como unidad central de proceso de una computadora, proporcionando el control de las operaciones de cálculo. Los microprocesadores también se utilizan en otros sistemas informáticos avanzados, como impresoras, automóviles o aviones.

El microprocesador es un tipo de circuito sumamente integrado. Los circuitos integrados, también conocidos como microchips o chips, son circuitos electrónicos complejos formados por componentes extremadamente pequeños ensamblados en una única pieza plana de poco espesor de un material conocido como semiconductor. Los microprocesadores modernos incorporan hasta 10 millones de transistores (que actúan como amplificadores electrónicos, osciladores o, más a menudo, como conmutadores), además de otros componentes como resistencias, diodos, condensadores y conexiones, todo ello en una superficie comparable a la de un sello postal.

Un microprocesador consta de varias secciones diferentes.

- La unidad aritmético-lógica (ALU) efectúa cálculos con números y toma decisiones lógicas.
- Los registros son zonas de memoria especiales para almacenar información temporalmente.
- La unidad de control decodifica los programas.
- Los buses transportan información digital a través del chip y de la computadora.
- La memoria local se emplea para los cómputos realizados en el mismo chip.

Los microprocesadores más complejos contienen a menudo otras secciones; por ejemplo, secciones de memoria especializada denominada memoria *cache*, que sirven para acelerar el acceso a los dispositivos externos de almacenamiento de datos. Los microprocesadores modernos funcionan con una anchura de bus de 64 bits (un bit es un dígito binario, una unidad de información que puede ser un uno o un cero): esto significa que pueden transmitirse simultáneamente 64 bits de datos.

Un cristal oscilador proporciona una señal de sincronización, o señal de reloj, para coordinar todas las actividades del microprocesador. La velocidad de reloj de los microprocesadores más avanzados es de unos 300 megahertz (MHz) —unos 300 millones de ciclos por segundo—, lo que permite ejecutar unos 1.000 millones de instrucciones cada segundo.

El microprocesador que utilizaremos es el AT89C51 de Atmel, el cual cuenta con 8kb de memoria ROM, 128kb de memoria RAM, 32 líneas de entrada y salida, 2 relojes o contadores de 16 bits cada uno, un puerto serial para transmitir y recibir, y 5 puertos de interrupción. El AT89C51 se alimenta con 5V y una corriente no mayor a 15mA.

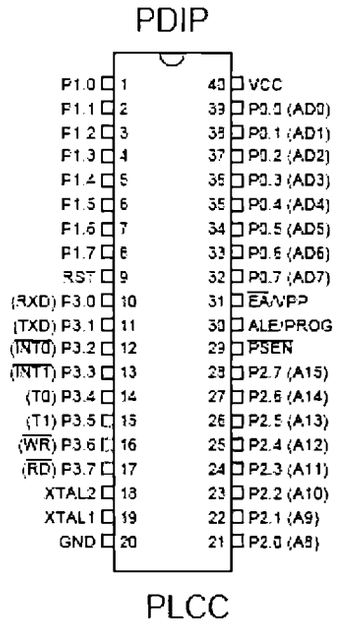


Figura 1[3]: Diagrama encapsulado

3.2 Sensor Velocidad

Es importante al implementar un sensor de velocidad, para fines automotrices, que las medidas se tomen sin tener ningún contacto mecánico con el eje de rotación. Para lograr estas mediciones, se utilizan varios métodos, pero el más común usado en la industria automotriz, es la medición magnética o algún fenómeno óptico.

En un sistema típico de medición de velocidad, la velocidad del vehículo es acoplada mecánicamente al sensor de velocidad por medio de un cable flexible, proveniente del eje de las llantas, que rota a una velocidad angular proporcional a la velocidad del automóvil. La señal en este cable genera una señal de pulsos como la que se muestra en la Figura 2, la cual se procesa por una computadora para obtener el valor digital de la velocidad.

[3] Hoja especificaciones Atmel: **http://www.atmel.com/dyn/resources/prod_documents/doc0265.pdf**

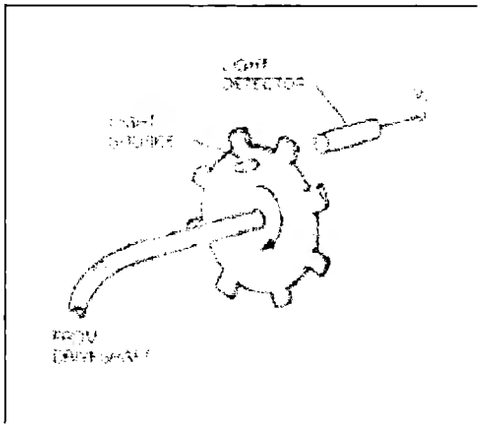


Figura 2: Sensor Óptico de Velocidad[4]

En el caso del sensor óptico, tenemos el cable flexible que lleva un disco dentado que gira entre una fuente y un detector de luz. Está alineado de tal manera que el disco interrumpe o deja pasar la luz de la fuente hacia el detector, dependiendo si la muesca se encuentra en la línea de vista entre ellos. El detector de luz produce un voltaje de salida cada vez que un pulso de luz de la fuente pase hacia el detector. El número de pulsos generados por segundo, es proporcional al número de muescas en el disco y a la velocidad del vehículo:

$$f = tNSK$$

Donde,

f- frecuencia en pulsos por segundo

N- número de muescas en el disco

S- Velocidad del vehículo

K- Constante de proporcionalidad encargada de linealizar la relación dada por la diferencia de tamaños en el disco y en las llantas.

Es importante mencionar que tanto los sensores ópticos como los magnéticos generan pulsos a la salida.

[4] Ribbens B. William, Understanding Automotive Electronics, pp. 268

Los pulsos de salida pasan por una compuerta de muestreo hacia un contador digital, como podemos ver en la Figura 3.

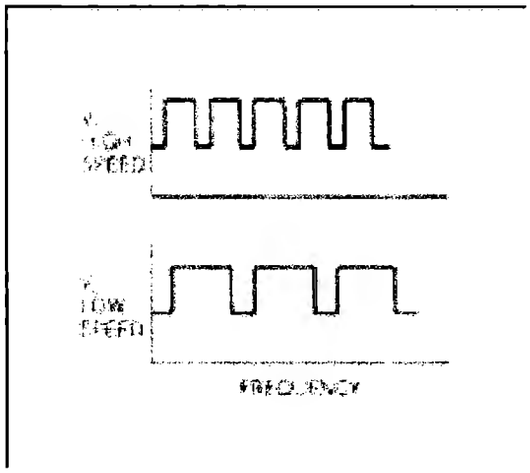


Figura 3: Variación en frecuencia con respecto al valor de velocidad[5]

La compuerta es un switch electrónico que ya sea que deje o no pasar los pulsos dependiendo si éste se encuentre abierto o cerrado. El intervalo de tiempo durante el cual la compuerta está cerrada, es precisamente controlado por la computadora. El contador digital cuenta el número de pulsos del detector de luz durante un tiempo t en el cual la compuerta está abierta. El número de pulsos P que cuenta el contador digital se puede expresar de la siguiente manera:

$$P = tNSK$$

Y este valor P, es proporcional a la velocidad del vehículo S. La señal eléctrica en el contador binario está en formato digital.

3.3 EmbeddedBlue 500

[5] ibid

El EmbeddedBlue 500 es un dispositivo de comunicación bluetooth el cual tiene las siguientes características:

- Frecuencia: 2.4 GHz FHSS (Frequency Hopping Spread Spectrum)
- Potencia Transmisión: 4dBm (max) Operación Clase 2.
- Rango de campo abierto: 300 ft.
- Bluetooth: Compatible con el standard v1.
- Sensibilidad Receptor a 0.1% BER: -85dBm
- Voltaje de alimentación de 5v a 12v
- Consumo de corriente 25mA transmitiendo a 9.6kbps, 30mA transmitiendo a 38.4kbps y 35mA transmitiendo a 115.2kbps.

El EmbeddedBlue 500 consta de 20 pines los cuales se describen en la siguiente tabla:

Pin	Parallax Pin	Function	Description	Usage
CN1 - 1	GND	GND	Ground	Required
CN1 - 2	GND	GND	Ground	Required
CN1 - 3	P0	TX	Serial Transmit line from eb500	Required
CN1 - 4	P1	RX	Serial Receive line to eb500	Required
CN1 - 5	P2	RTS	Request-to-Send on the serial port interface between the eb500 and the BASIC Stamp	Optional
CN1 - 6	P3	CTS	Clear-to-Send on the serial port interface between the eb500 and the BASIC Stamp	Optional
CN1 - 8	P5	Status	Bluetooth connection status (0 = not connected, 1 = connected)	Required
CN1 – 9	P6	Mode	Command/data mode toggle (0 = command, 1 = data)	Required
CN1 - 20	VCC	VCC	Power	Required

Tabla 2: Descripción pines entrada y salida[6]

[6] Hoja especificaciones EmbeddedBlue500

Es importante mencionar que para nuestra aplicación no todos los pines se utilizarán ya que algunos de estos como por ejemplo el CN1-5 y 6 se usan para el kit Basic Stamp.

A continuación se muestra la distribución de los pines en el EmbeddedBlue analizando el dispositivo de manera frontal.

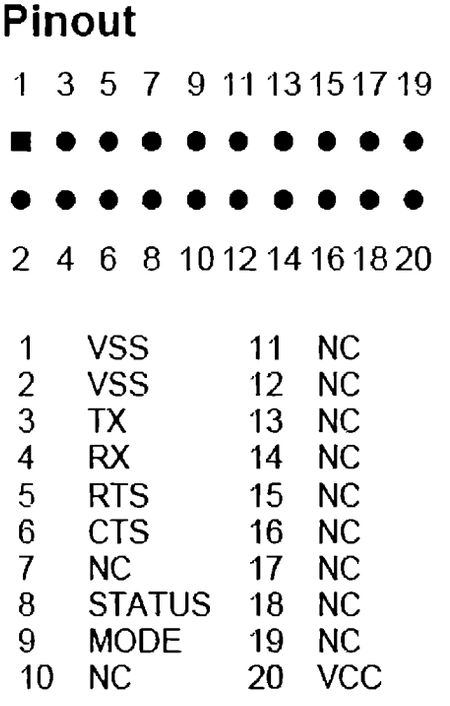


Figura 4[7]

Bluetooth es una tecnología de radio de corto alcance, que permite conectividad inalámbrica entre dispositivos remotos. Se diseñó pensando básicamente en tres objetivos: pequeño tamaño, mínimo consumo de energía y bajo precio.

[7] Hoja especificaciones EmbeddedBlue500

Bluetooth opera en la banda libre de radio ISM (Industrial, Scientific, Medical) a 2.4Ghz que incluye los rangos de frecuencia entre 902- 928MHz y 2.4- 2.484Ghz, que no requieren una licencia de operador otorgada por las autoridades reguladores de telecomunicaciones. Su máxima velocidad de transmisión de datos es de 1 Mbps. El rango de alcance Bluetooth depende de la potencia empleada en la transmisión. La mayor parte de los dispositivos que usan Bluetooth transmiten con una potencia nominal de salida de 0 dBm, lo que permite un alcance de aproximadamente10 metros en un ambiente libre de obstáculos. Debido a que la banda ISM está abierta a cualquiera, el sistema de radio Bluetooth deberá estar preparado para evitar las múltiples interferencias que se pudieran producir. Éstas pueden ser evitadas utilizando un sistema que busque una parte no utilizada del espectro o un sistema de salto de frecuencia. En este caso la técnica de salto de frecuencia es aplicada a una alta velocidad y una corta longitud de los paquetes (1600 saltos/segundo). Con este sistema se divide la banda de frecuencia en varios canales de salto, donde, los transceptores, durante la conexión van cambiando de uno a otro canal de salto de manera pseudo-aleatoria. Los paquetes de datos están protegidos por un esquema ARQ (repetición automática de consulta), en el cual los paquetes perdidos son automáticamente retransmitidos.

Bluetooth utiliza un sistema FH/TDD (salto de frecuencia/división de tiempo dúplex), en el que el canal queda dividido en intervalos de 625 μs, llamados slots, donde cada salto de frecuencia es ocupado por un slot. El orden de las frecuencias seleccionadas por el transmisor se toma de un grupo predeterminado impuesto por la secuencia de código. Cuando un dispositivo es capaz de enviar y recibir al mismo tiempo, esa capacidad bidireccional se denomina operación dúplex. Hay dos formas de ofrecer funcionamiento dúplex, la duplexación por división de frecuencia (FDD, Frequency Division Duplexing) y la duplexación por división de tiempo (TDD, Time division duplexing).

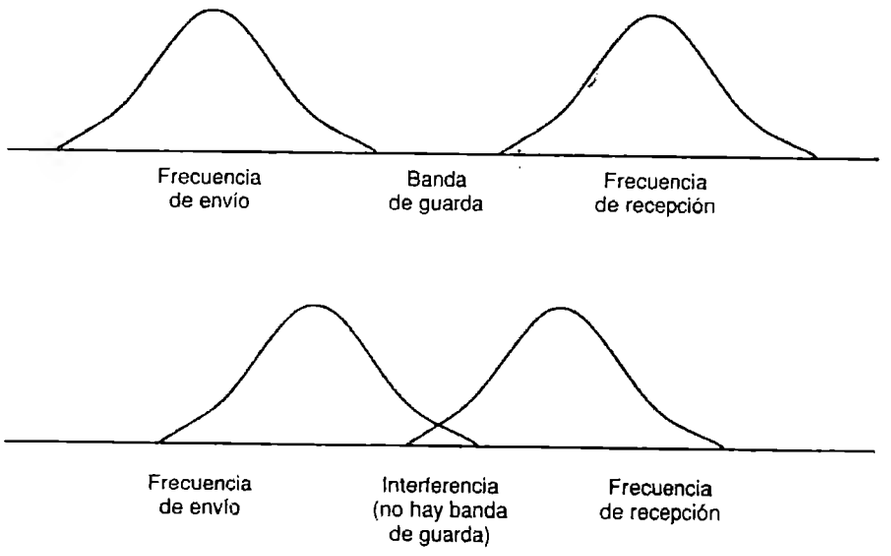


Figura 5: Bandas de frecuencias[8]

En la duplexación por división de frecuencia se utilizan dos frecuencias separadas para enviar y recibir. Una banda de guarda evita que las dos frecuencias se interfieran entre sí (arriba). Si no hubiera ninguna banda de guarda, existiría la posibilidad de interferencias (abajo).

[8] MULLER, Nathan J, Tecnología Bluetooth

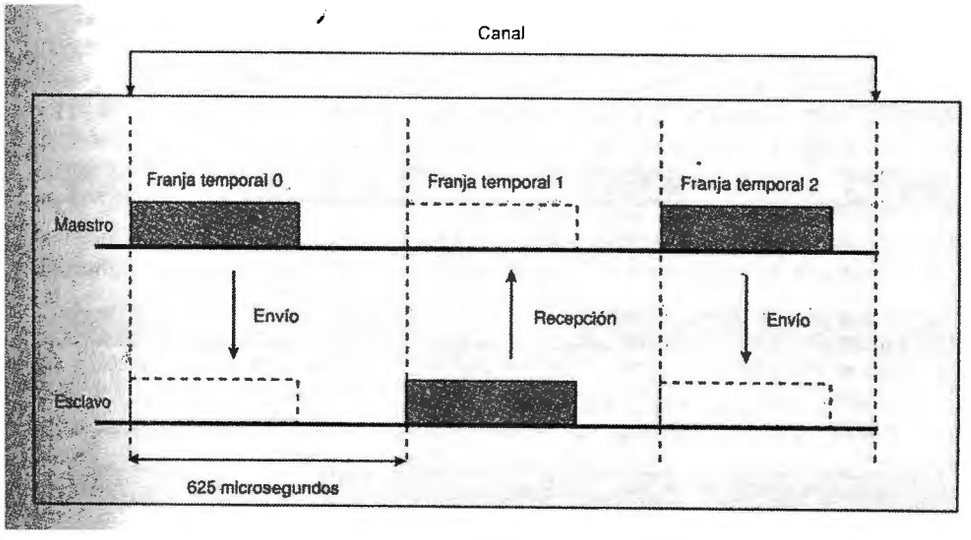


Figura 6: Paquetes en el tiempo[9]

Con el esquema TDD utilizado en la tecnología inalámbrica Bluetooth los paquetes se envían en franjas temporales de 625us de duración, entre las unidades maestra y esclava en una picored.

Dos o más unidades Bluetooth pueden compartir el mismo canal dentro de una picored (pequeña red formada por dos o más dispositivos que establecen automáticamente los terminales Bluetooth para comunicarse entre si y que comparten un canal en el mismo rango de cobertura), donde una unidad actúa como maestra, controlando el tráfico de datos en la picored que se genera entre las demás unidades, donde éstas actúan como esclavas, enviando y recibiendo señales hacia el maestro.

El salto de frecuencia del canal, está determinado por la secuencia de la señal, es decir, el orden en que llegan los saltos y por la fase de esta secuencia. En Bluetooth, la secuencia queda fijada por la identidad de la unidad maestra de la picored (un código único para cada equipo), y por su frecuencia de reloj.

[9] ibid

Además de los canales de datos, están habilitados tres canales de voz de 64 kbit/s por picored. Las conexiones son uno a uno con un rango máximo de diez metros, aunque utilizando amplificadores se puede llegar hasta los 100 metros, pero en este caso se introduce alguna distorsión. Los datos se pueden intercambiar a velocidades de hasta 1 Mbit/s. El protocolo bandabase que utiliza Bluetooth combina las técnicas de circuitos y paquetes para asegurar que los paquetes llegan en orden.

Programación Bluetooth en Java

Mientras que el hardware Bluetooth había avanzado mucho, hasta hace relativamente poco no había manera de desarrollar aplicaciones java Bluetooth – hasta que apareció JSR 82, que estandarizó la forma de desarrollar aplicaciones Bluetooth usando Java. Ésta esconde la complejidad del protocolo Bluetooth detrás de unos APIs (Application Programming Interface) que permiten centrarse en el desarrollo en vez de los detalles de bajo nivel del Bluetooth. Una API, es un conjunto de especificaciones de comunicación entre componentes software. Se trata del conjunto de llamadas al sistema que ofrecen acceso a los servicios del sistema desde los procesos y representa un método para conseguir abstracción en la programación, generalmente (aunque no necesariamente) entre los niveles o capas inferiores y los superiores del software.

Estos APIs para Bluetooth están orientados para dispositivos que cumplan las siguientes características:

- Al menos 512K de memoria libre (ROM y RAM) (las aplicaciones necesitan memoria adicional).
- Conectividad a la red inalámbrica Bluetooth.
- Que tengan una implementación del J2ME CLDC (Java 2 Platform Micro Edition Connected Limited Device Configuration).

El objetivo de ésta especificación era definir un API estándar abierto, no propietario que pudiera ser usado en todos los dispositivos que implementen J2ME. Por consiguiente fue diseñado usando los APIs J2ME y el entorno de trabajo CLDC/MIDP.

Los APIs JSR 82 son muy flexibles, ya que permiten trabajar tanto con aplicaciones nativas Bluetooth como con aplicaciones Java Bluetooth.

El API intenta ofrecer las siguientes capacidades:

- Registro de servicios.
- Descubrimiento de dispositivos y servicios.
- Establecer conexiones RFCOMM, L2CAP y OBEX entre dispositivos.
- Usar dichas conexiones para mandar y recibir datos (las comunicaciones de voz no están soportadas).
- Manejar y controlar las conexiones de comunicación.
- Ofrecer seguridad a dichas actividades.

Los APIs Java para Bluetooth definen dos paquetes que dependen del paquete CLDC **javax.microedition.io**:

- **javax.bluetooth**
- **javax.obex**

La aplicación de Bluetooth está dividida en cuatro partes:

- Inicialización de la pila.
- Descubrimiento de dispositivos y servicios.
- Manejo del dispositivo.
- Comunicación.

Los dispositivos Bluetooth que implementen este API pueden permitir que múltiples aplicaciones se estén ejecutando concurrentemente. El BCC **(Bluetooth Control Center)** previene que una aplicación pueda perjudicar a otra. El BCC es un conjunto de capacidades que permiten al usuario resolver peticiones conflictivas de aplicaciones definiendo unos valores específicos para ciertos parámetros de la pila Bluetooth.
Dado que los dispositivos inalámbricos son móviles, necesitan un mecanismo que permita encontrar, conectar, y obtener información sobre las características de dichos dispositivos.

Cualquier aplicación puede obtener una lista de dispositivos a los que es capaz de encontrar, usando ya sea el comando **startInquiry()** (no bloqueante) o **retrieveDevices()** (bloqueante). **startInquiry()** requiere que la aplicación tenga especificado un *listener,* el cual es notificado cuando un nuevo dispositivo es encontrado después de haber lanzado un proceso de búsqueda. Por otra parte, si la aplicación no quiere esperar a descubrir dispositivos (o a ser descubierta por otro dispositivo) para comenzar, puede utilizar **retrieveDevices()**, que devuelve una lista de dispositivos encontrados en una búsqueda previa o bien unos que ya conozca por defecto dichos dispositivos.
Para encontrar los dispositivos existen estas interfaces.

- **interface javax.bluetooth.DiscoveryListener**

  Esta interfaz permite a una aplicación especificar un evento en el *listener* que reaccione ante eventos de búsqueda. También se usa para encontrar dispositivos. El método **deviceDiscovered ()** se llama cada vez que se encuentra un dispositivo en un proceso de búsqueda. Cuando el proceso de búsqueda se ha completado o cancelado, se llama al método **inquiryCompleted()**. Este método recibe un argumento, que puede ser INQUIRY_COMPLETED, INQUIRY_ERROR o INQUIRY_TERMINATED, dependiendo de cada caso.

- **Interface javax.bluetooth.DiscoveryAgent**

Esta interfaz provee métodos para descubrir dispositivos y servicios. Para descubrir dispositivos, esta clase provee del método **startInquiry()** para poner al dispositivo en modo de búsqueda, y el método **retrieveDevices()** para obtener la información de dispositivos previamente encontrados. Además provee del método **cancelInquiry()** que permite cancelar una operación de búsqueda.

Para encontrar los servicios que nos da el dispositivo se utilizó la clase DiscoveryAgent que provee de métodos para buscar servicios en un dispositivo servidor Bluetooth e iniciar transacciones entre el dispositivo y el servicio. Este API no da soporte para buscar servicios que estén ubicados en el propio dispositivo.

Para descubrir los servicios disponibles en un dispositivo servidor, el cliente primero debe recuperar un objeto que encapsule funcionalidad SDAP (Service Discovery Application Profile) que es SDP (Service Discovery Profile)+GAP(Generic Access Profile). Este objeto es del tipo DiscoveryAgent, cuyo pseudocódigo viene dado por:

DiscoveryAgent da = LocalDevice.getLocalDevice().getDiscoveryAgent();
Algunas de las clases para encontrar servicios son:

- **class javax.bluetooth.UUID**

Esta clase encapsula números enteros sin signo que pueden ser de 16, 32 ó 128 bits de longitud.

Estos enteros se usan como un identificador universal cuyo valor representa un atributo del servicio, sólo los atributos de un servicio representados con UUID están representados en la Bluetooth SDP.

- **class javax.bluetooth.DataElement**

Esta clase contiene varios tipos de datos que un atributo de servicio Bluetooth puede usar. Algunos de estos son:

- String
- boolean
- UUID
- Enteros con signo y sin signo, de uno, dos, cuatro o seis bytes de longitud
- Secuencias de cualquiera de los tipos anteriores.

Esta clase además presenta una interfaz que permite construir y recuperar valores de un atributo de servicio.

- **class javax.bluetooth.DiscoveryAgent**

  Esta clase provee métodos para descubrir servicios y dispositivos.

- **interface javax.bluetooth.ServiceRecord**

  Este interfaz define el Service Record de Bluetooth, que contiene los pares (atributo ID, valor). El atributo ID es un entero sin signo de 16 bits, y el valor es de tipo DataElement. Además, este interfaz tiene un método **populateRecord()** para recuperar los atributos de servicio deseados (pasando como parámetro al método el ID del atributo deseado).

- **interface javax.bluetooth.DiscoveryListener**

Este interfaz permite a una aplicación especificar un *listener* que responda a un evento del tipo Service Discovery o Device Discovery. Cuando un nuevo servicio es descubierto, se llama al método **servicesDiscovered()**, y cuando la transacción ha sido completada o cancelada se llama a **serviceSearchCompleted()**.

Para usar un servicio en un dispositivo Bluetooth remoto, el dispositivo local debe comunicarse usando el mismo protocolo que el servicio remoto. Los APIs permiten usar RFCOMM, L2CAP u OBEX como protocolo de nivel superior

El Generic Connection Framework (GFC) del CLDC provee la conexión base para la implementación de protocolos de comunicación. CLDC provee de los siguientes métodos para abrir una conexión:

- Connection Connector.open(String name);

- Connection Connector.open(String name, int mode);
- Connection Connector.open(String name, int mode, boolean timeouts);

La implementación debe soportar abrir una conexión con una conexión URL servidora o con una conexión URL cliente, con el modo por defecto READ_WRITE.

3.4 LM7805

El LM7805 es un regulador de voltaje el cual te baja un voltaje no mayor a 20 V a uno de 5V. Este dispositivo consta de tres pines los cuales se muestran a continuación:

1. Input 2. GND 3. Output

Figura 7[10]

El LM7805 tiene las siguientes características:

- Voltaje de salida de 5V cuando el voltaje de entrada es de 7 a 20V.
- Corriente de salida de 5mA a 1A.

El diagrama a bloques del LM7805 es el siguiente:

[10] Hoja de especificaciones www.fairchild.com

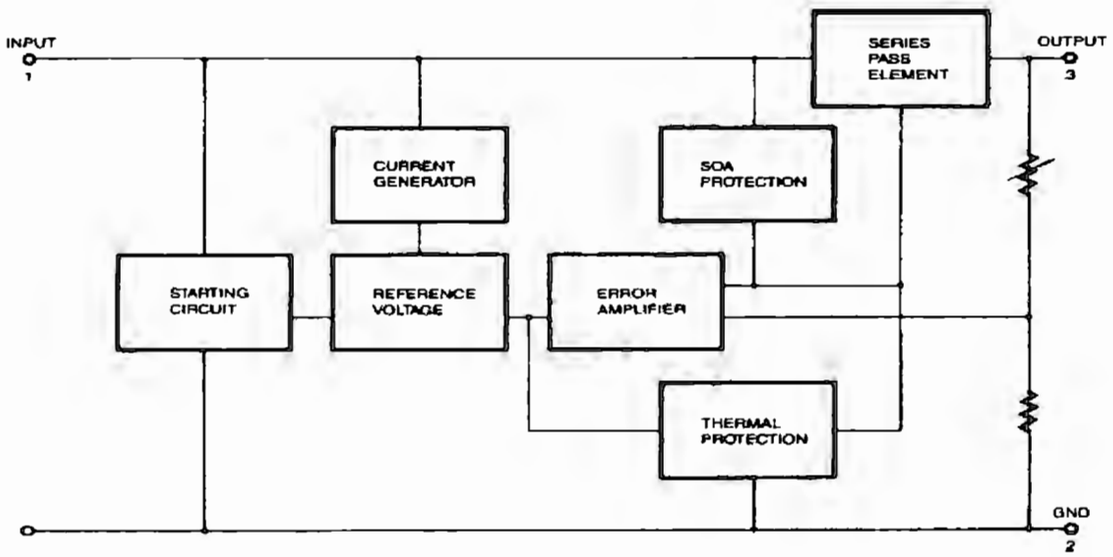


Figura 8[11]

3.5 Acelerómetro

Los acelerómetros son sensores e instrumentos para medir, desplegar y analizar tanto la aceleración como la vibración. Pueden ser elementos sencillos o transductores, o como un sistema o instrumento con funcionalidades como un contar con un display remoto, o almacenar información. Pueden tener desde uno hasta tres ejes de medición, y en los casos en los que se tengan ejes múltiples, generalmente éstos son ortogonales. Estos dispositivos tienen diferentes principios de operación. Los más comunes pueden ser piezoeléctricos, capacitivos, de resonancia, de inducción magnética, entre otros.

[11] ibid

Hay tres funcionalidades básicas a considerar cuando se selecciona un acelerómetro: rango de amplitud, rango de frecuencias en el que se trabaja, y las condiciones del ambiente (entre las cuales se considera la temperatura, el máximo impacto o vibración que pueda soportar).

Las opciones de salida eléctrica dependen del sistema que se utilice con el acelerómetro. Algunas salidas comunes son voltaje, corriente, o frecuencia. En cuanto a salidas digitales, se consideran las salidas paralelas y en serie. Otra opción es utilizar acelerómetros con una salida de cambio de estado de un switch o una alarma.

Todos los acelerómetros capacitivos tienen ciertos elementos de diseño en común. Incorporan una masa sísmica cuyo movimiento en respuesta a una vibración o "shock" se retrasa al "cascarón" del acelerómetro. El capacitor tiene dos placas, de las cuales, una está sujeta al marco exterior, (por lo tanto es estacionaria), y otra está sujeta al elemento sísmico, o masa inercial. El valor del capacitor, se calcula en función de la distancia entre las placas, que pueden variar con el movimiento de la masa sísmica.

Diseño del Elemento Sensor:

Una masa inercial, suspendida de cualquier capa del sensor, por medio de un sistema de vigas flexibles, presenta un movimiento rectilíneo al aplicar aceleración. La masa está conectada eléctricamente por medio de un circuito de medio puente de capacitancia variable. La capacitancia en un lado del circuito aumenta con la aceleración, mientras que en el otro lado decrece proporcionalmente, lo cual nos da una salida lineal. Un entre-hierro en el capacitor de 3.6 micrones, nos provee una sensibilidad de 0.003 pF/g.

La capa intermedia contiene la masa inercial y el sistema de suspensión. El sistema de suspensión es un vector de hasta 188 vigas en los niveles de las dos superficies principales de la masa inercial alrededor de la periferia rectangular en el marco de soporte. Si se

modifica la forma, el área, o el número de vigas, entre otras variables, se puede afectar la sensibilidad a la aceleración del sensor.

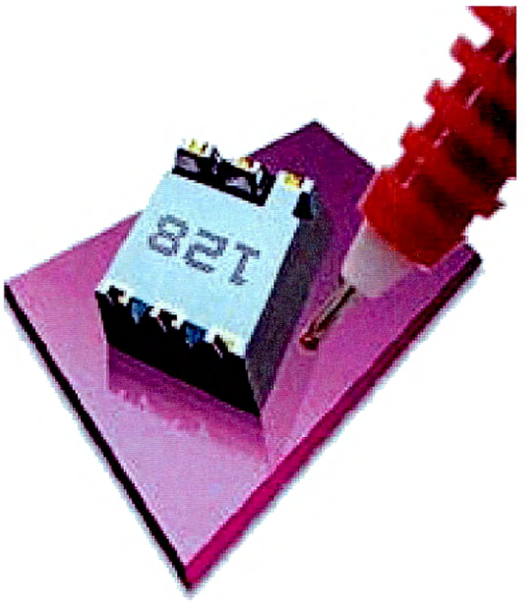


Figura 9: Comparación de dimensiones

Esta tabla nos muestra las características típicas de funcionamiento de los elementos variables capacitivos del sensor.

VC Sensor Element Performance Characteristics				
Full Scale Range	**±1 g**	**±10 g**	**±30 g**	**±150 g**
Sensitivity	0.8 pF/g	0.05 pF/g	0.016 pF/g	0.003 pF/g
Resonant Frequency	600 Hz	3,000 Hz	5,000 Hz	11,000 Hz
Damping Ratio	2.3	0.7	0.7	0.7
Temperature Change (-55°C to 125°C)	+0.08%/°C	+0.08%/°C	+0.08%/°C	+0.08%/°C
Nonlinearity BFSL	±0.2% F.S.O.	±0.2% F.S.O.	±0.2% F.S.O.	±0.2% F.S.O.
Overrange Stops	2 g min.	20 g min.	50 g min.	300 g min.

Tabla 3

3.6 SMS

El servicio corto de mensajes SMS

Este servicio fue desarrollado para el sistema GSM como parte de la fase dos de esta especificación, el servicio corto de mensajes o SMS por sus siglas en inglés, está basado en la capacidad de una terminal digital celular de mandar o recibir mensajes alfanuméricos. Los mensajes cortos pueden ser de hasta 140 bytes en tamaño, y enviados en pocos segundos donde exista la cobertura GSM. Más que un sistema común de mensajes, la entrega del mensaje es garantizada incluso cuándo la terminal celular no está disponible (cuando está apagado o fuera del área de cobertura). La red guardará el mensaje y lo entregará poco tiempo después de que la terminal celular anuncie su presencia en la red. El hecho que SMS (por medio de GSM) soporte roaming internacional con muy poco retardo lo hace muy adecuado para aplicaciones como mensajes, email, y notificaciones de correo de voz así como servicios de mensajes para múltiples usuarios. Hay dos tipos de SMS disponibles: transmisión celular y punto a punto. En la transmisión celular, un mensaje es transmitido a todas las estaciones móviles (MS) presentes en la célula que

tienen la capacidad de recibir mensajes cortos y tienen suscrito a este servicio particular de información. El servicio es en un solo sentido, y no hay confirmación de receptor se envía. Puede enviar hasta 97 caracteres de 7 bits ó 82 caracteres de 8 bits, típicamente usados para transmitir mensajes sobre condiciones de tráfico, pronósticos meteorológicos, estados de la bolsa y otras aplicaciones.

En los servicios punto a punto, los mensajes pueden ser enviados de un dispositivo móvil a otro o desde una PC a un dispositivo móvil y viceversa. Los mensajes son mantenidos y trasmitidos por un centro de SMS (SMSC- SMS Center). El SMSC es una versión electrónica de un servicio postal ordinario que guarda y después reenvía el mensaje cuando éste puede ser entregado. Cada red GSM debe soportar uno o más SMSCs para mantener y canalizar los mensajes. Cada SMSC checa, organiza, y manda el mensaje al operador. Este también recibe y pasa cualquier mensaje de confirmación a cualquier móvil GSM o red. Hay varias formas en las cuales un mensaje pude ser pedido, dependiendo de las interfaces que soporte el SMS de la red GSM. Los usuarios pueden llamar a la central de mensajes (ejemplo: un operador) o directamente crear el mensaje en el teclado de su dispositivo móvil.

La arquitectura de red GSM

La arquitectura puede ser dividida en tres componentes principales

- El suscriptor mantiene el MS (mobile station) nombrada la terminal GSM.
- El susbsistema de estación base controla el enlace de radio con el MS.
- El subsistema de red realiza la conmutación de llamadas y otras tareas de administración como autentificación.

La estación Móvil

La estación móvil y el subsistema de estación base se comunican a través de una interface aérea o enlace de radio. El MS consiste en la terminal física y contiene el transmisor de

radio, el display y el procesador de señales, y el módulo de identificación de suscriptor (SIM). El SIM provee al usuario la habilidad de accesar a sus servicios suscritos sin importar la ubicación de la terminal usada.

El subsistema de estación Base

El subsistema de estación base está compuesto por dos partes; el transmisor/receptor de la estación base (BTS) y el controlador de la estación base (BSC). El BSC es la conexión entre el MS y el centro de conmutación móvil (MSC). El BSC también se encarga de convertir el canal de voz de 13kb/s usado en un enlace de radio al canal estándar de 64kb/s usado por las redes públicas de conmutación telefónica (PSTN).

El subsistema de red

El MSC es el componente principal del subsistema de red. Se encarga de toda la funcionalidad necesaria para manejar suscriptores móviles como registro, autentificación, actualización de localización y ruteo a suscriptores con roaming. El MSC también actúa como una puerta de salida para el PSTN o ISDN, y provee la interface para el SMSC.

El protocolo de señalización GSM

El intercambio de mensajes de señalización respecto a movilidad, recursos de radio, y mantenimiento de conexión entre diferentes entidades de la red GSM se maneja por medio del protocolo en la Figura 10.

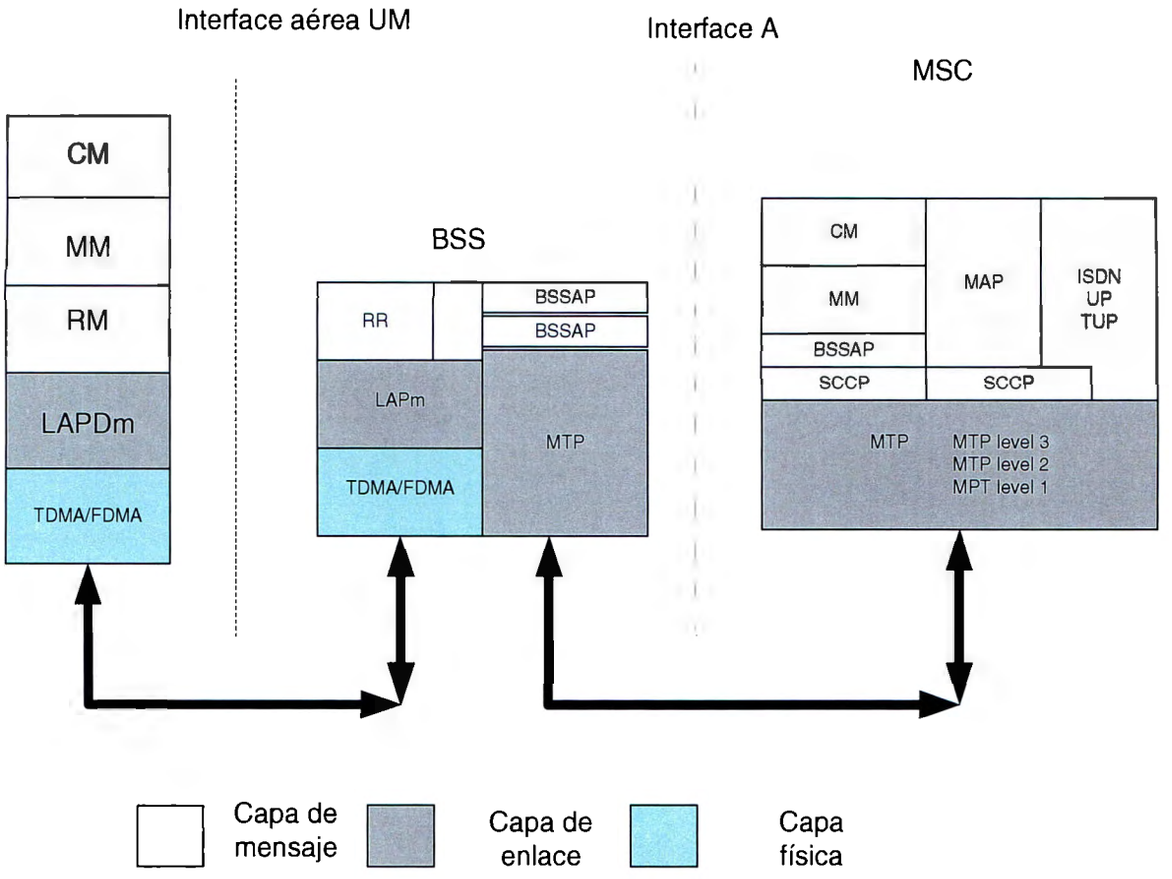


Figura 10[12]

La arquitectura consiste de tres capas; física, datos y mensaje. En el lado del MS, la capa de mensaje consiste en tres subcapas: administración de conexión (CM), administración de movilidad (MM), y administración de recursos (RR). La subcapa CM administra los servicios suplementarios relacionados con llamadas, SMS y servicios de soporte suplementarios de llamadas independientes. En el lado del MSC, la capa de mensajes se divide en cuatro subcapas. La parte de aplicación de la subestación del sistema base (BSSAP) del MSC provee las funciones de conmutación de canal, administración de recursos de radio y funciones de trabajo en Internet. La parte de transferencia de mensajes (MTP) y la parte de control de conexión de señalización son protocolos usados para

[12] PEERSMAN Guillame, CVETKOVIC Srba, "The Global System for Mobile Communications Short Message Service", IEEE Personal Communications, Junio 2000, pp. 17.

implementar la capa de enlace de datos y funciones de transferencia de capa 3 para mantener el control de llamada y administrar la movilidad de mensajes de señalización a través de la interface A. Los paquetes SCCP también son usados para mandar los mensajes para SMS. La señalización entre las diferentes entidades utiliza la ITU SS7 (International Telecommunication Union), utilizada ampliamente en ISDN y las redes públicas. SS7 es actualmente el único elemento en la estructura GSM capaz de conmutación de paquetes así como conmutación de circuitos. Se utiliza para transportar las señales de control y paquetes de mensajes cortos para SMS.

Implementación Práctica

SMS utiliza la señalización de canal SS7 para transmitir el paquete de datos, por lo que permite que se reciban mensajes de texto cuando el usuario está realizando una llamada de voz o datos. Un MS activo debe ser capaz de mandar y recibir mensajes cortos por la unidad de datos del protocolo de transporte (TPDU) en cualquier momento sin importar si hay una llamada de voz o datos en proceso. Se regresa una confirmación siempre al SMSC indicando si el MS ha recibido el mensaje corto o no. Una confirmación también será regresada al MS desde el SMSC indicando si el TPDU ha sido recibida exitosamente. El software entre el MS debe ser capaz de decodificar y guardar los mensajes.
La terminación móvil SMS (SMS-MT) es la habilidad de recibir un mensaje SMS de un SMSC y es más que obvio, mientras que la generación móvil SMS (SMS-MO) es la habilidad de mandar mensajes cortos a un SMSC.

La arquitectura básica de una red SMS

Cuando se canaliza un mensaje corto originado por un móvil, el SMSC reenvía el mensaje corto al SMS-SMSC. EL SMS-GMSC pregunta al HLR la información de ruteo y manda el mensaje corto al MSC apropiado. El MSC manda el mensaje corto al MS. Por otro lado cuando se canaliza una terminación móvil de mensaje corto, el MS encuentra el SMSC requerido de acuerdo a su titulo global. El SMSC identifica cada mensaje como único

agregándole una etiqueta en el campo SMS-DELIVER TP-SCTS . Es la responsabilidad del SMSC de asegurar que si dos mensajes cortos llegan al mismo tiempo las etiquetas únicas sean diferenciadas.

Datos de usuario (TP-UD)

El TP-UD es un campo utilizado para mandar el mensaje corto. Este puede guardar hasta 148 octetos de información para SMS punto a punto, junto con el encabezado dependiendo de la configuración del campo TP-UDHI. La cantidad de espacio tomado por el encabezado reduce la cantidad de información que el PDU puede mandar. El encabezado tiene al menos tres campos, el primer campo, el elemento de identificación de la información, es usado para identificar los mensajes cortos concatenados. La longitud de la información (IDL) es usada ara indicar la longitud del elemento de datos de información (IED) que sigue. Cada uno de estos elementos es de 1 octeto de longitud.

En los datos de usuario, el mensaje puede ser de 7 bits, 8 bits, o 16 bits. Usando el IEI permite mandar y recibir mensajes cortos concatenados. El campo de IED contiene la información necesaria para la entidad receptora para reensamblar los mensajes en el orden correcto y se codifica de la siguiente forma:

- Primer octeto: número de referencia del mensaje corto que identifica el mensaje en la misma transacción.
- Segundo octeto: especifica el número máximo de mensajes cortos en un mensaje corto concatenado, que no excederá 255.
- Tercer octeto: identifica el número de secuencia en un mensaje corto dentro de un mensaje concatenado.

La mínima longitud de un encabezado para un mensaje concatenado es de 8 octetos para 8bits y 16 bits y será de 8 octetos para datos de 7 bits; dejando 133 (140.7), 152(160-8), y 66 ((140-7)/2) caracteres para el mensaje corto. La máxima longitud del mensaje es por lo

tanto incrementada a 38.760(255x152), 33,915(255x133), o 16,830(255x66) dependiendo en el esquema de codificación de caracteres utilizado.

Consideraciones de ruteo de mensajes cortos

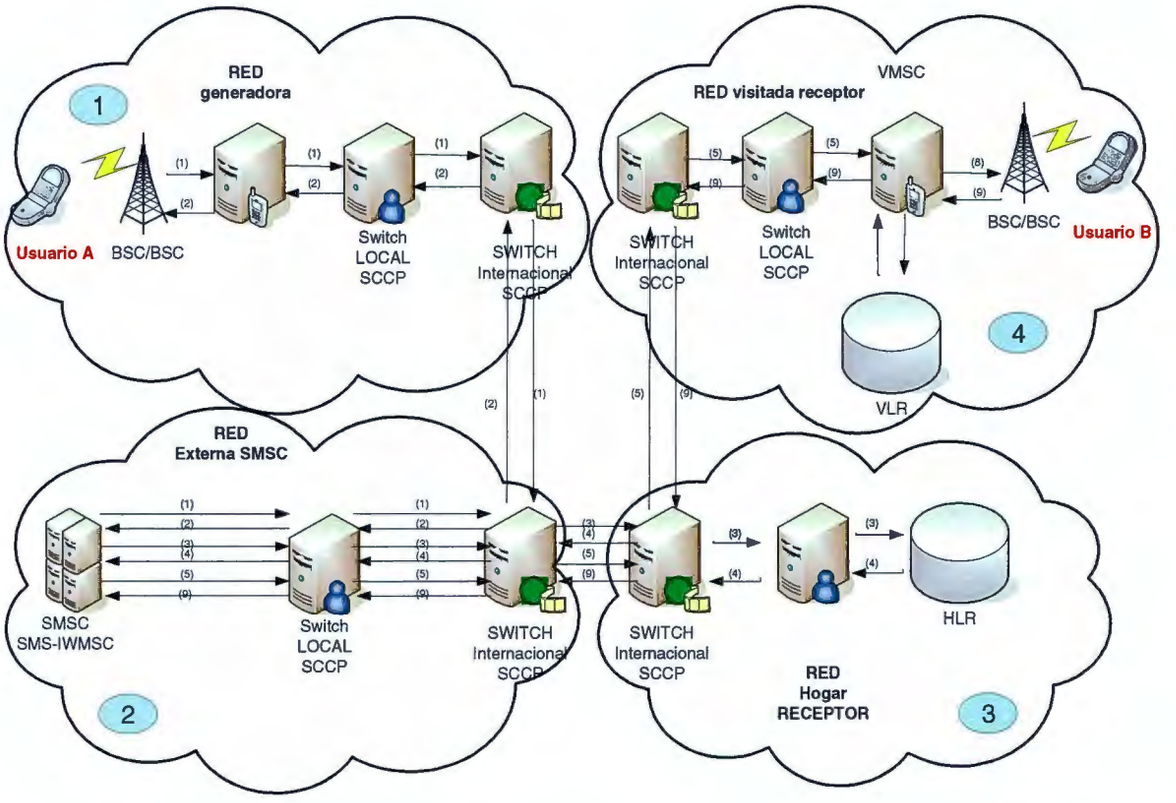


Figura 11

En la Figura 11, el usuario A en la red 1 manda un mensaje corto a un usuario B en la red 3 de roaming 4. El usuario A está usando el SMSC en la red 1 para hacer la petición del mensaje.

El celular local intercambia rutas del mensaje corto en un paquete SCCP de acuerdo con el titulo global del SMSC como se define en el plan de numeración E.164 [9]. El paquete SCCP es reenviado de intercambio hasta que encuentra el SMSC destino (1). La ruta debe se encontrarse en cada conmutador de SCCP a través de la ruta para que el mensaje alcance exitosamente el SCCP en la red 1.

Una vez que el paquete SCCP que tiene el mensaje llega a su SMSC destino, un mensaje de confirmación se manda de regreso al dispositivo móvil usando otro paquete SCCP (2).
Para entregar el mensaje corto al usuario B, el SMSC tiene que acezar la base de datos HLR en su red local. Una petición de paquete de localización SCCP, basado en el número del usuario móvil B, se manda por medio del SMSC (3).
Esta red internacional SCCP establece la ruta del paquete de localización SCCP al HLR apropiado. Cuando el HLR recibe la petición, este regresa la información de localización en otro paquete SCCP al SMSC (4).
El SMSC manda el mensaje al VMSC del usuario B, basado en la información recibida por el HLR (5). Finalmente este VMSC pregunta el VLR (6,7), y entrega el mensaje al usuario B (8). Una vez que se entrega el mensaje exitosamente un paquete de confirmación SCCP se manda de regreso al SMSC (9).

Protocolos para petición de mensajes cortos

El Instituto Europeo de Estándares de Telecomunicaciones (ETSI) especifica un protocolo para la petición de mensajes cortos como parte del estándar GSM. Esta especificación define tres protocolos de interface para transferir mensajes cortos SMS entre un MS y un equipo terminal (TE) vía una interface asíncrona.

Modo de texto

Este modo es un protocolo basado en caracteres que a su vez se basa en un set de comandos AT modificados para GSM. Este modo es adecuado para terminales no inteligentes o emuladores de terminales, y para aplicaciones construidas de software en estructuras de comandos como las definidas en ITU V25ter. Esta aplicación pasa el mensaje en texto plano al teléfono para construir el TPDU. Esto significa que el modo de texto ofrece poca funcionalidad. El modo de texto no soporta o automáticamente pasa mensajes de entrada a la aplicación (sólo notifica).

Modo PDU

El modo PDU es muy similar al modo de texto, excepto que deja a la aplicación la responsabilidad de construir el TPDU del mensaje corto. Este modo agrega la conveniencia del set de comandos AT la posibilidad de construir PDUs más sofisticados (por ejemplo permitir la transmisión de datos binarios, no únicamente caracteres).

IV. PROPUESTA DE SOLUCIÓN

Las partes básicas dentro del diseño de un auto son: motor, suspensión, transmisión, sistema de frenado, dirección, carrocería y sistema eléctrico. En la actualidad, distintas ramas de la electrónica como la instrumentación, control y en sí la electrónica son fundamentales para el buen desempeño de un auto.

Es por esto que, al ver los avances que los automóviles tienen en lo que respecta a la electrónica, decidimos que para adquirir la medición de la velocidad del automóvil utilizaremos la señal generada por éste, tomando la señal que se transmite al velocímetro. En los coches actuales se utiliza un sensor óptico de velocidad, el cual manda la señal al velocímetro. Éste se encuentra frente a una rueda perforada, que recibe el movimiento por medio de un cable flexible que lo toma de la caja de cambios.

El diodo led del sensor emite su luz sobre un fototransistor, cuando coincide cada ventana de la rueda perforada, lo cual hace que el fototransistor conduzca, y lleve ésta señal a un circuito de control, el cual maneja la aguja del velocímetro. La frecuencia de los impulsos generados varía con respecto a la velocidad y determina el movimiento de la aguja del velocímetro. En algunos casos, también se usa un sensor de efecto Hall también en el cable flexible para ésta función. La señal entregada es digital y con frecuencia proporcional a la velocidad. Por lo que tomaremos ésta señal en frecuencia del cable del sensor. Para poder manipularla, se usará un circuito convertidor de frecuencia a voltaje (F/V). Con la ayuda de un diodo, se recortará la señal para obtener un valor de voltaje de 5V (TTL). Este valor de voltaje analógico se acondicionará por medio de un Convertidor Analógico Digital (DAC) para poder utilizar esta señal y procesarla. Teniendo ésta señal digital, se utilizará como la entrada a un microprocesador 8051 que formará parte del sistema mínimo a desarrollar.

Se tenían algunas propuestas para poder detectar el momento justo de la colisión. Después de realizar algunas pruebas, se decidió el siguiente procedimiento:

La primera propuesta consta en tomar muestras de la señal de entrada (velocidad) cada segundo, y se irán almacenando en la memoria del microprocesador creando un historial. Con la ayuda del acelerómetro MEMS, se creará una señal de interrupción en el momento de un cambio brusco de velocidad, el cual el programa identificará como la colisión, y el valor registrado en ese momento, será la velocidad en el momento de la colisión.

Cuando el ajustador requiera adquirir los datos del historial, se conectará con el sistema mínimo mediante un teléfono celular por medio de la tecnología Bluetooth utilizando el EmbeddedBlue 500 para la transmisión y recepción de datos. Una vez obtenidos los datos, el software de interfaz se conectará con una base de datos remota donde descargará los datos del cliente.

Diagrama a bloques:

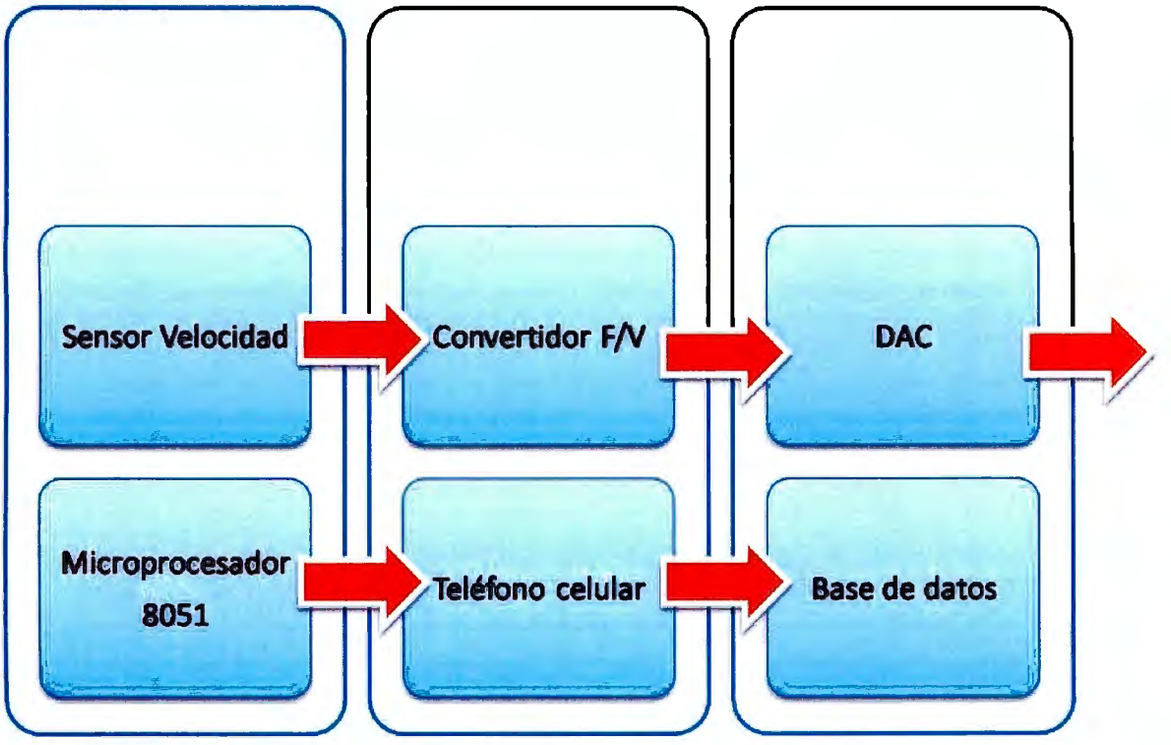


Figura 12: Diagrama a Bloques

V. RESULTADOS Y PRUEBAS

Con la propuesta de solución antes mencionada hemos conseguido nuestros resultados, que a continuación presentaremos.

5.1 EmbeddedBlue 500

Figura 13: Transceptor EmbeddedBlue

Realizamos la programación de este dispositivo, para poder transmitir hacia un teléfono celular. Con las pruebas pudimos transmitir información hacia el teléfono celular que para fines de éste proyecto, nos servirá para recibir y enviar los datos de velocidad hacia la base de datos.

Las pruebas realizadas a este dispositivo fueron para verificar su funcionamiento y entenderlo. Utilizamos la tarjeta BASIC STAMP 2 como interfaz con el dispositivo para programar unas rutinas de comunicación vía Bluetooth. La primera prueba fue

establecer una comunicación entre el dispositivo y una computadora con comunicación Bluetooth. Para realizar esto fue necesario emparejar el dispositivo con la computadora para que la misma lo reconociera.

Lo siguiente fue programar una rutina de lectura de puertos en el BASIC STAMP 2 para que una vez que se leen los datos se transmitan por medio de la tarjeta Bluetooth. Una vez realizada la rutina se observó la comunicación por medio del monitoreo del puerto en la computadora por medio de la Hyperterminal. La comunicación fue en tiempo real.

Lo siguiente fue establecer una comunicación desde la computadora al dispositivo Embedded Blue. Para esto programamos la rutina de lectura de serial en el Basic Stamp. Después de programar la rutina monitoreamos el puerto de comunicación serial ligado con Bluetooth desde la computadora para comenzar a trasmitir caracteres. En el Basic Stamp se leyeron y trasmitieron a otra computadora conectada por medio del puerto serial.

Una vez que logramos comunicar unidireccionalemente juntamos las rutinas para verificar la comunicación bidireccional. Esto se logró de la misma forma que la comunicación unidireccional por medio del monitoreo de puertos en las computadoras una conectada a la tarjeta Basic Stamp 2 por medio cable serial y la otra se comunicaba por medio de Bluetooth.

Posteriormente establecimos la comunicación con la tarjeta embedded blue con una Pocket PC con comunicación Bluetooth. La rutina de transmisión y recepción fue la misma únicamente se requirió previamente emparejar los dispositivos en el administrador de comunicación Bluetooth. Después se corrió la aplicación de monitoreo de puertos en la Pocket PC y se verificó que si existiera comunicación.

Para la parte de envió de datos desde Bluetooth hacia el dispositivo, investigamos acerca de varios lenguajes de programación para ver cual era el mas apropiado, para lo que nosotros necesitábamos, entre estos lenguajes encontramos algunos especiales para celulares, BREW y Symbian, sin embargo estos lenguajes a pesar de que se parecen a C++, eran lenguajes que no conocíamos, así que decidimos utilizar java, debido a que actualmente los celulares soportan este tipo de programación, en java utilizamos JSR-82 el cual es un API para la programación de dispositivos Bluetooth. Este depende de la configuración de CLDC (Connected Limited Device Configuration) de Java. De aquí utilizamos el paquete de javax.bluetooth. Este paquete provee la funcionalidad para la realización de búsquedas de dispositivos, servicios y comunicación mediante flujos de datos (streams) o arrays de bytes. Un cliente Bluetooth deberá realizar las siguientes operaciones para comunicarse con un servidor Bluetooth:

- Búsqueda de dispositivos
- Establecimiento de la conexión
- Comunicación

La creación de un servidor Bluetooth es más sencilla que la programación de un cliente ya que no necesitamos realizar ningún tipo de búsqueda. Los pasos que debe realizar un servidor Bluetooth son los siguientes:

- Crear una conexión servidora
- Especificar los atributos de servicio
- Abrir las conexiones cliente

Debido a que no tenemos una página de Internet o un servidor, nos comunicaremos por medio de mensajes (SMS), la idea es que un celular esté conectado a una computadora que va estar recibiendo el numero de póliza y el historial de velocidad, con estos datos se los enviara a la computadora para poder obtener de la base de datos la información necesaria

del asegurado y los enviara al celular para que se envíen por medio de Java SMS SDK al celular que tenga el ajustador.

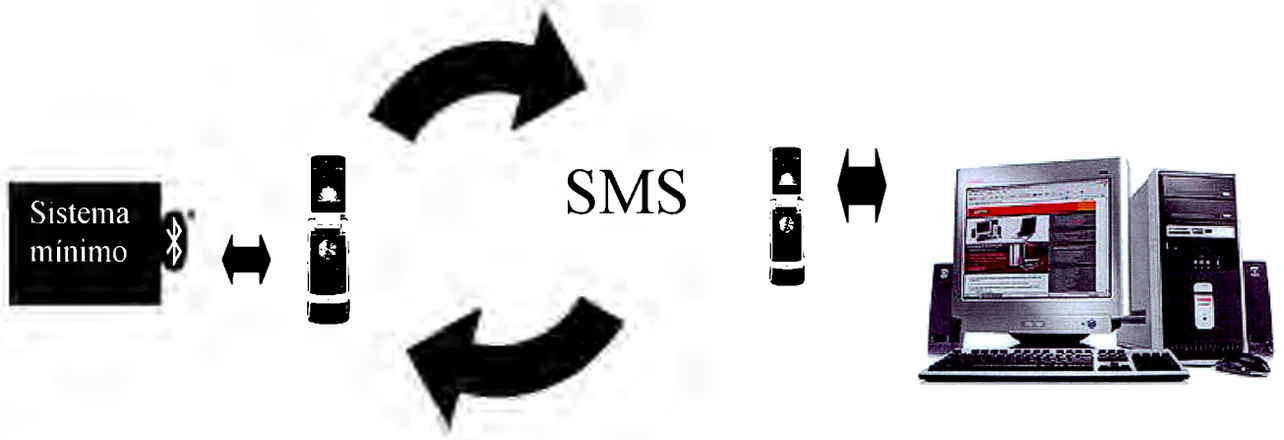


Figura 14

De manera general al instalar el programa (Anexo 1) en el primer celular este lo que hace es buscar los dispositivos que se encuentran dentro del alcance del celular. Después, el ajustador tiene que seleccionar cuál es el dispositivo al que se está conectado el sistema mínimo, éste le demandara la contraseña del ajustador, lo que permitirá que se pueda conectar con el sistema mínimo. Al establecer la conexión, el ajustador podrá recuperar del sistema mínimo el número de póliza, la velocidad de impacto y el historial de velocidades. Al recibir esta información, el celular la enviará hacia el servidor por medio de otro celular, por medio de un mensaje SMS. Este teléfono celular funcionado como servidor se conectará a la computadora de la base de datos por medio también de tecnología Bluetooth, y así se podrá regresar la información necesaria al ajustador tales como el nombre del asegurado, el modelo del auto, y número de accidentes.

5.2 Sensor de velocidad

Se han explorado ya muchas opciones y posibilidades para sensar la velocidad del automóvil. Hay dispositivos capaces de sensar la velocidad de un automóvil por medio de principios como Efecto Hall o Efecto Doppler, pero son muy caros, y se pudo encontrar un sensor dentro del mismo coche, manteniendo los costos de nuestro proyecto bajos, y de esta manera lograr que sea un proyecto rentable. Por lo que tomaremos la señal de velocidad del sensor de velocidad del automóvil y la procesaremos para su almacenamiento.

Como parte de las pruebas para obtener los valores de velocidad, realizamos mediciones tanto del sensor de velocidad como del dispositivo ABS en dos autos distintos. Para ambos casos, se hizo un barrido de velocidad cada 10 km/hr. Los resultados obtenidos se ven reflejados en las siguientes tablas y gráficas:

Sensor de Velocidad:

Velocidad (km/hr)	**Frecuencia (Hz)**
20	19
30	27
40	40
50	46
60	55
70	66.5
80	76
90	84.5
100	92
110	107
120	112
130	120
140	132
150	142
160	157

Tabla 4: Pruebas de Velocidad con sensor

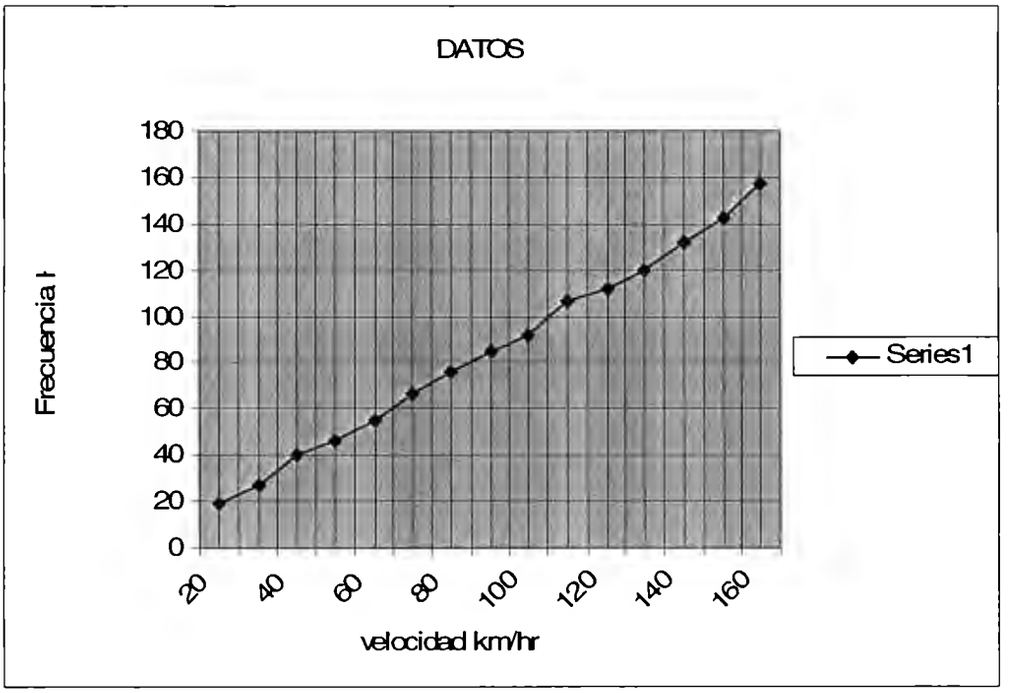


Figura 15: Gráfica velocidad vs. Frecuencia sensor velocidad.

Sensor ABS:

Velocidad (km/hr)	Frecuencia salida (Hz)
30	198
40	260
50	325
60	390
70	425
80	515
90	560
100	625
110	710
120	750
130	830
140	910
150	960

Tabla 5: Pruebas Velocidad con sensor ABS

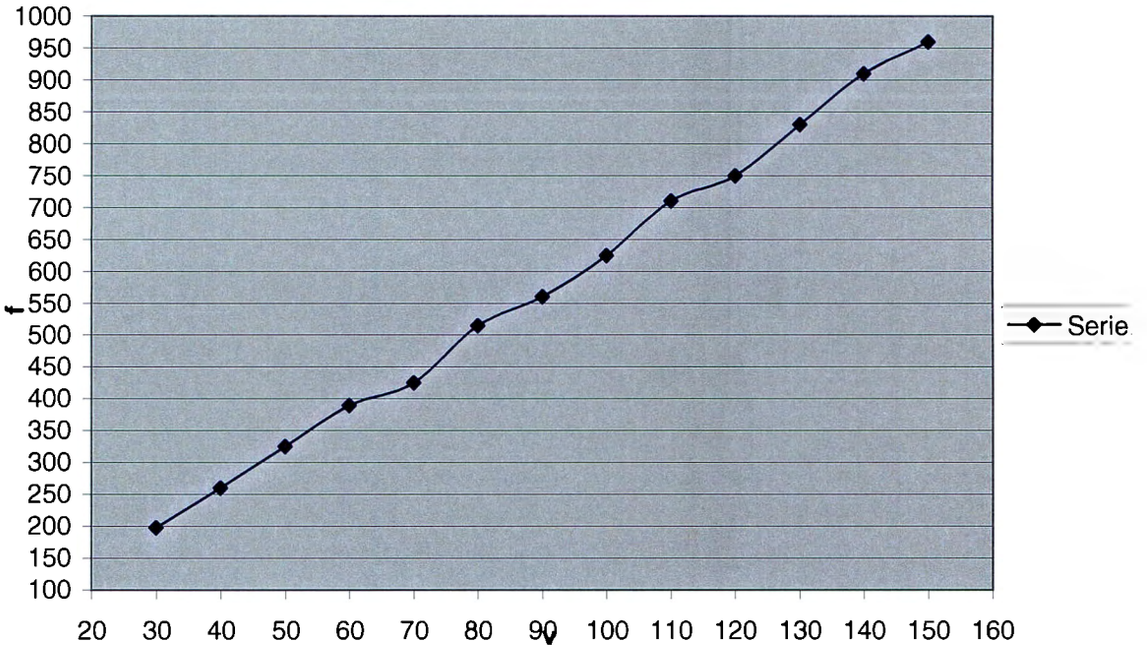


Figura 1: Gráfica velocidad vs. Frecuencia con sensor ABS

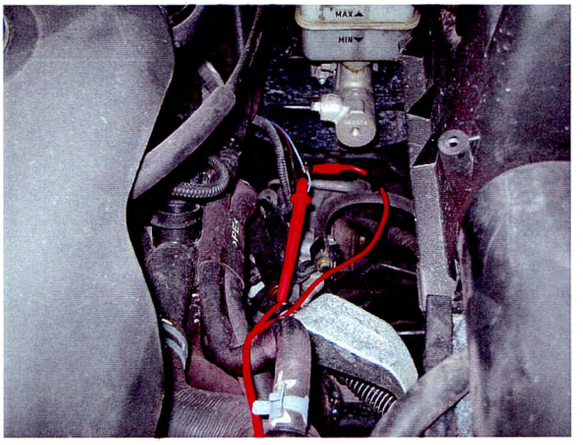


Figura 2: Conexión física con el sensor de velocidad del automóvil

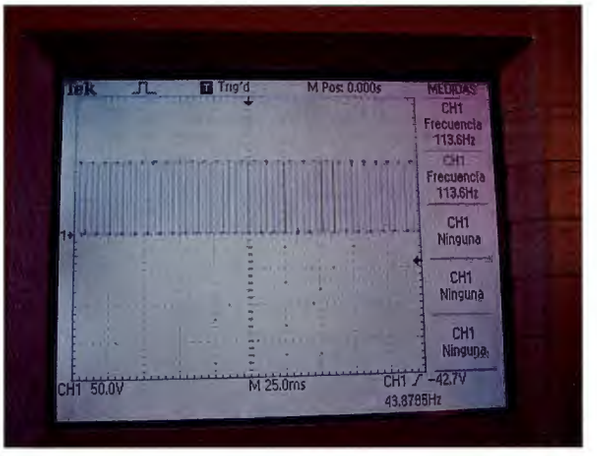


Figura 18: Visualización de la frecuencia en el osciloscopio

Figura 19: Visualización de la frecuencia en el multímetro

Figura 20: Aceleración del automóvil para adquirir los datos

Lo que pudimos observar con éstas gráficas en ambos casos, fue que el valor de frecuencia que detectamos es lineal con respecto a la velocidad. Pero en el caso del sensor de velocidad, se observa que también es directamente proporcional el valor de frecuencia al del voltaje.

5.3 Convertidor Frecuencia-Voltaje

Para esta etapa se tienen contemplados dos circuitos integrados, la elección dependerá de cual se logre calibrar mejor para nuestro propósito. Los circuitos son LM331 (propósito general) y LM2907 (de uso automotriz).

5.3.1 LM331

Este circuito integrado es capaz de convertir frecuencia en voltaje. Una señal de cierta frecuencia entra en el integrado y es diferenciada por una red C-R. La corriente promedio de salida del convertidor es directamente proporcional a la frecuencia esta determinada por la siguiente expresión.

$$I_{prom} = 1.1(R_t C_t) \times F$$

A la salida se utiliza una carga pasiva para generar un voltaje proporcional a la frecuencia, además esta carga esta acompañada de un capacitor que en conjunto funciona como un filtro para reducir el rizo. Para lograr mejorar la precisión se utilizó un filtro de dos polos a la salida agregando un amplificador operacional en conjunto con la red de filtrado.
Por lo tanto el voltaje de salida se encuentra en función de la frecuencia de entrada, la red C-R y las resistencia de Carga y ajuste.

$$Vsal = F_{IN} \times 2.09 \times \frac{R_L}{R_S} \times (R_t C_t)$$

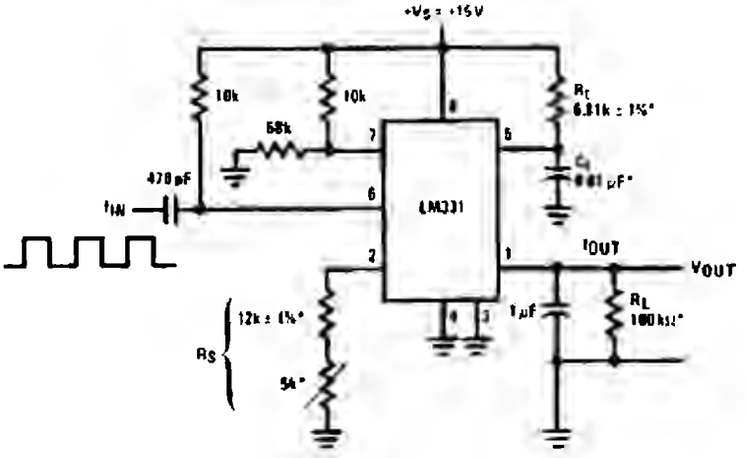


Figura 21: Configuración del LM331 para Frecuencia-Voltaje

5.3.2 LM2907

Es un convertidor monolítico de frecuencia a voltaje con un comparador/op amp de alta ganancia usado para operar relevadores, lámparas, u otra carga cuando la entrada de frecuencia alcance o sobrepase un cierto valor.

El voltaje de salida corresponde a la siguiente ecuación:

$$Vout = f_{IN} \times Vcc \times R_1 \times C_1$$

Para convertir la frecuencia a voltaje, se requiere de un capacitor temporal, una resistencia de salida, y un integrador o capacitor que sirva como filtro. Cuando la etapa de entrada cambia de estado (esto se debe a un cruce por cero o un voltaje diferencial en la entrada), el capacitor temporal se carga o descarga linealmente entre dos voltajes cuya diferencia es $V_{CC}/2$. Entonces en un medio ciclo de la entrada de frecuencia o en un tiempo igual a la mitad de la frecuencia de entrada, el cambio en la carga del capacitor es igual a $V_{CC}/2 \times C_1$. Entonces la cantidad promedio de corriente inyectado dentro o fuera del capacitor es:

$$\frac{\Delta Q}{T} = i_{C(PROM)} = C_1 \times \frac{V_{CC}}{2} \times (2F_{IN}) = V_{CC} \times F_{IN} \times C_1$$

El circuito de salida refleja esta corriente de manera muy precisa a la carga de la resistencia R_1, que está conectada a tierra, de manera que si los pulsos de corriente se integran con un capacitor que sirva de filtro, tenemos:

$$V_O = V_{CC} \times F_{IN} \times C_1 \times R_1 \times K$$

En donde K es la ganancia que generalmente vale 1. El valor de C_2 depende de la cantidad de voltaje de rizo permitido, y del tiempo de respuesta que requiramos.

$$V_{Rizo} = \frac{V_{CC}}{2} \times \frac{C_1}{C_2} \times \left(1 - \frac{V_{CC} \times F_{IN} \times C_1}{I_2}\right)$$

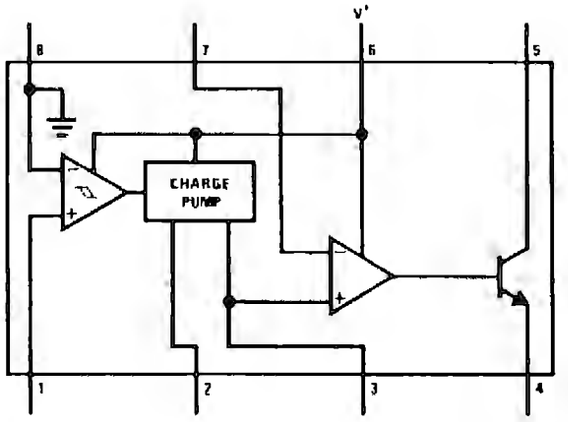


Figura 22: Bloques internos LM2907

5.4 Convertidor Analógico Digital

Debido a que obtuvimos una señal continua después de convertir la frecuencia en voltaje es necesario convertir la señal de la salida del convertidor frecuencia voltaje en una señal digital para su posterior procesamiento.

Para convertir una señal analógica en digital se llevan acabo dos procesos, estos son la cuantificación de la señal y su codificación. Cuantificar consiste en transformar la amplitud muestreada de una señal en un instante en una amplitud discreta tomada de un conjunto finito de amplitudes posibles. La cantidad de posibles valores depende de el número de bits que se vayan a utilizar, es decir para n bits existirán 2^n valores posibles. La codificación será la representación del valor de la señal que se ha asignado en la cuantificación por medio de un conjunto de símbolos binarios.

Todo este proceso se realiza dentro del convertidor Analógico-Digital. Es decir existirá un conjunto de bits, en este caso serán ocho bits, que representarán el nivel de voltaje a la entrada del convertidor. El tiempo de acceso a la información es de 135ns.

Para que se codifique linealmente la señal fue necesario ajustar la señal de entrada para que los niveles de voltaje correspondan a toda la escala de bits posible, es decir los 255 niveles.

5.5 Sistema de procesamiento

5.5.1 Sistema Mínimo

Para poder procesar las lecturas de velocidades, y tener un historial, se utilizará el microcontrolador 8051 de Intel. El programa deberá leer datos cada segundo y almacenarlos en la memoria del mismo. También utilizará el valor del acelerómetro como señal de interrupción para detectar el valor exacto de la velocidad de colisión.

El programa realizado para cumplir con estas funciones, es el siguiente. El compilador utilizado es el ProView32.

Se adquirirán los datos de velocidad previamente procesados mediante el convertidor de F/V y el convertidor analógico- digital, por un puerto de entrada del microprocesador cada segundo. El acelerómetro generará la interrupción externa al microprocesador indicando que debe de almacenar el último valor leído el cual corresponderá al valor de la última velocidad registrada al momento del accidente. Al mismo tiempo se desplegará por medio del display, la velocidad en todo momento para el conductor.

Cuando el ajustador requiera adquirir el historial de velocidades almacenado en nuestro sistema mínimo, éste enviará la contraseña correspondiente, se verificará si la información recibida es correcta. Si no lo es se enviará un mensaje al ajustador. De lo contrario se enviarán los datos anteriormente almacenados de forma serial.

Para poder probar el sistema mínimo en el automóvil en movimiento, necesitábamos alimentar nuestro sistema de alguna manera, por lo que tomamos la alimentación de 12V de la batería del automóvil, y la accesamos por la guantera del mismo. De ahí mismo, sacamos un cable con la señal del velocímetro para poderlo conectar al sistema mínimo y corroborar que funciona con el auto en movimiento.

También desarrollamos un regulador de voltaje para poder bajar los 12V que nos proporciona la batería a 5V que es el voltaje con el que alimentamos nuestro sistema mínimo. Esto lo podemos observar en la Figura 23.

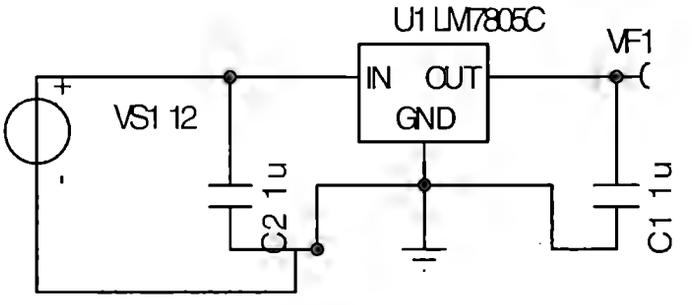


Figura 23: Regulador Voltaje

Para procesar la señal que nos entrega el velocímetro, que va de 0-12 V, se diseñó un arreglo de resistencias y diodos para convertir la señal a TTL (Figura 24).

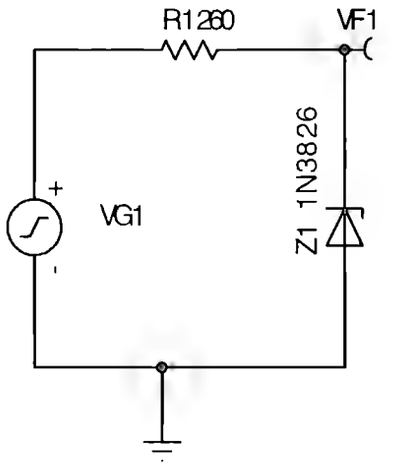


Figura 24: Arreglo para obtener señal TTL

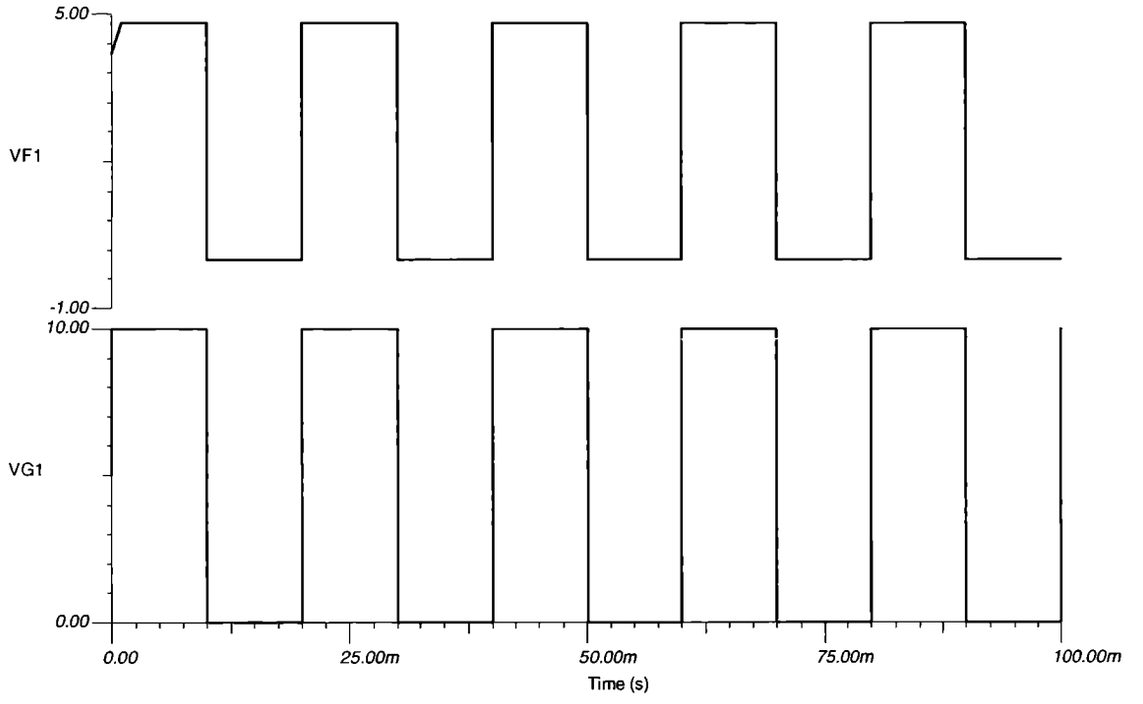


Figura 25:Salida TTL

Con estas modificaciones, pudimos probar que en efecto, el display de nuestro sistema mínimo coincide con el valor del velocímetro al estar en movimiento.

Para llevar un registro exacto del tiempo sin utilizar los timers del microcontrolador que pueden depender de que el circuito esté energizado y se pierdan los valores, utilizaremos un reloj externo el cual tendrá registro de manera digital del tiempo, y esta señal será enviada periódicamente al 8051. De esta manera, se sabrá en qué momento se produce el accidente, entonces estará relacionada la velocidad de la colisión con el tiempo en el que ocurre. Este reloj externo se considera dentro del trabajo a futuro para nuestro prototipo, ya que no se incluyó en el prototipo presentado. Esto también es de utilidad para las aseguradoras, para asegurarse de que el lugar en donde se encuentre el conductor al momento de llamar al ajustador, es en efecto el lugar en el cual se produce la colisión.

5.6 Acelerómetro

Un acelerómetro es un sensor que convierte la aceleración de movimiento o gravedad, en una señal eléctrica. Una de sus aplicaciones es en la industria automotriz para detectar choques, entregando a la salida ya sea una señal analógica o digital.

Figura 26: Acelerómetro[13]

Un acelerómetro puede servir como una medida de velocidad, distancia o fuerza. Lo puede lograr ya que si se integra la aceleración con respecto al tiempo, nos puede dar la velocidad a la que viaja un objeto. Y si integramos con respecto al tiempo, la velocidad, podemos obtener la distancia que recorre el mismo. Estas aplicaciones incluyen sensores para bolsas de aire, sistemas de localización de coches y control de elevación.

La resolución en estos dispositivos se refiere a la cantidad mínima de g's que puede medir. A continuación mostraremos una tabla de algunas medidas que se pueden considerar con la variable "g", como referencia.

[13] Analog Devices, What is an Accelerometer and when do you use one.

g's	Ejemplo
1	La aceleración causada por la gravedad de la Tierra sobre un objeto (por ejemplo, una computadora montada sobre un escritorio, experimenta una aceleración de 1g)
0-2	El rango que experimenta un humano al moverse, como al caminar.
5-30	La aceleración experimentada por un conductor en un accidente típico automovilístico.
100-2000	La aceleración que experimentaría una computadora (laptop) si se lanza desde un metro de altura hacia el piso (concreto).
10000	La aceleración que experimentaría una laptop si se disparara desde un cañón

Tabla 6: Referencias de valor 'g'.

Obtuvimos el acelerómetro ADXL278 de Analog Devices. Este dispositivo se utiliza para aplicaciones de la industria automotriz, como por ejemplo en el sistema de bolsas de aire. El rango de medición de éste es de ±35g. Para poder interpretar los datos que nos entrega el dispositivo, se sabe que éste nos entrega un voltaje de 2.5V al experimentar 0 g's. Este valor va cambiando con respecto a la aceleración que experimente, aumentando 38 mV por cada 'g' adicional.

Este dispositivo cuenta con 8 pines de conexión que se distribuyen de la siguiente manera:

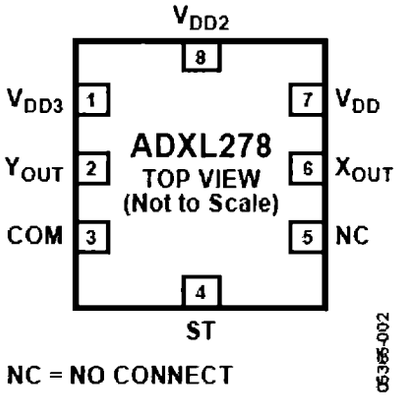


Figura 27: Diagrama conexión ADXL278

5.6.1 Teoría 'g'

Cuando cambia la velocidad de un objeto, se dice que se está acelerando. O sea que la aceleración es el cambio de la velocidad en el tiempo.

Generalmente, decimos que algo se acelera cuando un objeto tiene una velocidad ascendente. Por ejemplo, cuando leemos o vemos en la televisión la expresión "de cero a 100 en 6.7 seg", se refieren a que a un auto le toma 6.7 segundos para llegar a una velocidad de 100 km/hr, empezando desde un alto total. Se puede decir que esta es la manera en la que comúnmente se entiende el término 'aceleración', pero en Ingeniería y en Física, se refiere a mucho más que un incremento en la velocidad.

Una unidad que se utiliza comúnmente, es la aceleración debido a la gravedad. Debido a que estamos familiarizados con los efectos que puede tener la gravedad sobre nosotros y los objetos alrededor de nosotros, se buscó un estándar con el cual se pueda comparar la aceleración. Para ponerlo de manera práctica, podemos decir que todo se siente normal a 1g, doblemente pesado a 2g, y sin peso alguno a 0g. Generalmente el valor de g se expresa como 9.8 m/s^2, pero más adelante entraremos en detalle sobre su valor.

Para poner en perspectiva el peso en esta fuerza, pondremos un ejemplo. En un show de aviones, se anuncia que un piloto un giro el cual lo hacía experimentar 6g's. En otras palabras, el piloto pesaba 74 kg, entonces ahora siente que pesa 444 kg (6 veces su peso).

En este caso el piloto experimenta la fuerza de g en un largo período de tiempo, y en cambio en un accidente automovilístico, se lleva a cabo entre 100 y 150 ms.

Como hemos observado en la documentación, cuando se presenta una colisión a una alta velocidad, se experimentan aceleraciones de entre 25 y 50 g's. Por ejemplo, si una persona de 70 kg experimenta una aceleración de 50 g's durante un accidente, habrá una fuerza de 350 kg sobre su cuerpo. Es importante notar que el valor de g como hemos mencionado es realmente una aceleración, aunque para fines ilustrativos, se maneja como una fuerza.

Las bolsas de aire se implementaron por primera vez en 1970 por General Motors. Y hoy en día cada auto que se produce en Estados Unidos contiene bolsas de aire.

Los acelerómetros y otros sensores son usados comúnmente en los sistemas de bolsas de aire de los automóviles, los cuales, como en nuestro proyecto, serán los indicados para indicar si ha habido una desaceleración lo suficientemente fuerte como para activar las bolsas de aire tanto del conductor, como del pasajero. Por ejemplo, si hay una colisión con una barda, o alguna superficie así de sólida, habrá una rápida deceleración debido a que la velocidad del automóvil cambiará tan repentinamente. Para esto, la National Highway Traffic Safety Administration en Estados Unidos definió los requerimientos en los automóviles en 1998, indicando que la aceleración no puede excederse de 60 g's para activar las bolsas. En 1974, este valor estaba en los 36 g's. Para efectos de nuestro proyecto, evitando cualquier error, se tomó como criterio mínimo el valor de 30g's. En caso de que no haya sido un choque, el sistema seguirá reportando velocidades hasta que algún dispositivo móvil con tecnología Bluetooth y la clave de acceso necesaria, esta velocidad almacenada se enviará y se tomará como una velocidad a la que se produjo un

accidente. Estas lecturas pueden almacenarse el tiempo que sea necesario, dependerá de la memoria del diseño, pero se propone guardar las velocidades de un día.

Para poder medir los efectos fisiológicos en los humanos por efectos de aceleración, a continuación pondremos algunos ejemplos relacionados con el cuerpo humano.

a(g)	Evento
2.9	Estornudo
3.5	Toz
3.6	Empujones en una "multitud"
4.1	Manotazo en la espalda
8.1	Brincar de un escalón
10.1	Caerse de una silla
60	Límite de aceleración en el pecho durante un accidente automovilístico a 48 km/hr con bolsa de aire.
70-100	Colisión que mató a la Princesa Diana en 1997

También al considerar la aceleración, es importante entender como vimos anteriormente, se tiene que considerar la relación entre la ΔV y la aceleración. Ninguno de los dos sería necesario para poder cuantificar un potencial de lesión. Por ejemplo, un astronauta puede experimentar ΔV's de 29772.8 km/hr pero a lo largo de un extenso período de tiempo. Si utilizamos la relación

$$a=\Delta V/\Delta t$$

podemos calcular que un astronauta a esa misma velocidad pero en una hora, experimentará una fuerza de 0.23g. Un vehículo estacionario que sea golpeado por la parte trasera y acelerado a 8 km/hr en 0.12 segundos se expone a una aceleración de 1.9g. Pero también hay que considerar el movimiento de los cuerpos que experimenten esta fuerza. Por

ejemplo, el cuello presentará una aceleración hasta de 12g's debido a la flexibilidad que pueda presentar.

5.6.2 ADXL278

Hemos explorado diferentes tipos de acelerómetros, y nos decidimos por el modelo ADXL278 de Analog Devices, el cual es un circuito integrado

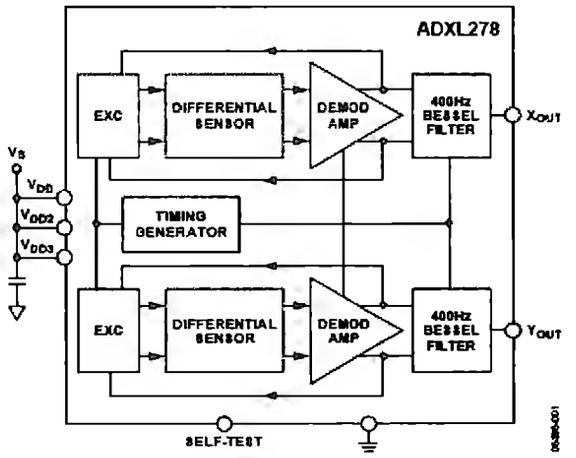


Figura 28: Diagrama a Bloques Acelerómetro

Este acelerómetro está formado por diferentes elementos o celdas capacitivas. Cada celda tiene placas fijas sujetas a un sustrato y placas sujetas al marco. Cualquier desplazamiento del marco, cambia la capacitancia diferencial la cual se mide en la circuitería.

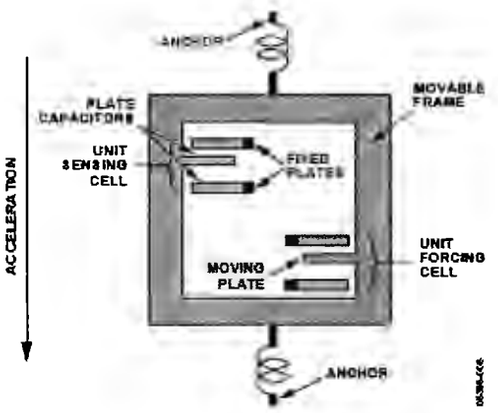


Figura 29: Diagrama simplificado del acelerómetro[14]

Las placas fijas funcionan gracias a una señal cuadrada de 200 KHz. La retroalimentación eléctrica ajusta las amplitudes de las ondas cuadradas logrando con esto, que la señal de a.c. en las placas movibles sea de 0. La señal de retroalimentación es linealmente proporcional a la aceleración aplicada.

5.7 Diseño de Hardware

Circuito del sensor de velocidad y codificación

[14] Analog Devices, ADXL278 Datasheet

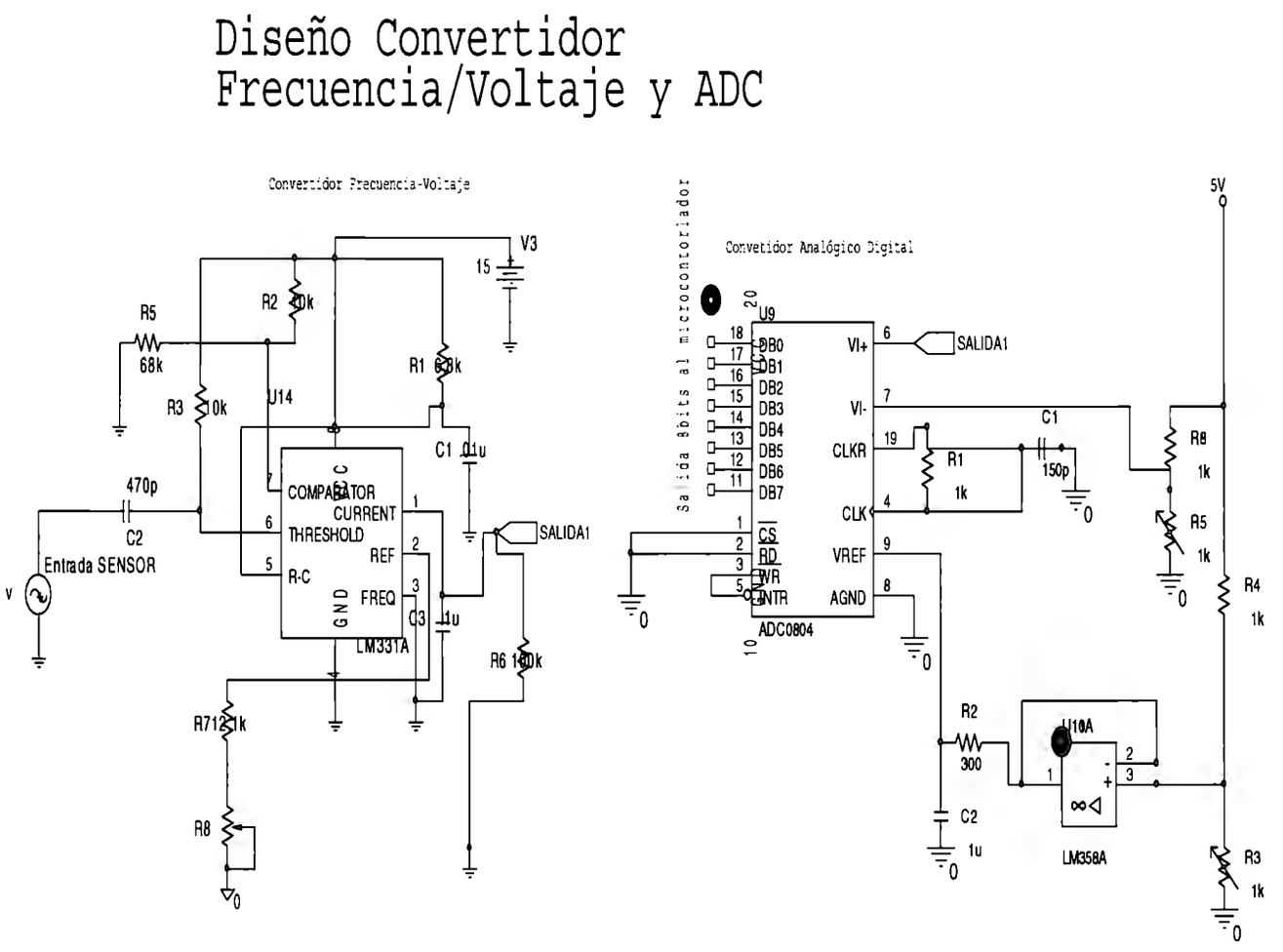


Figura 30: Diagrama de conexiones hardware

El diagrama de hardware del prototipo terminado es el que se muestra a continuación:

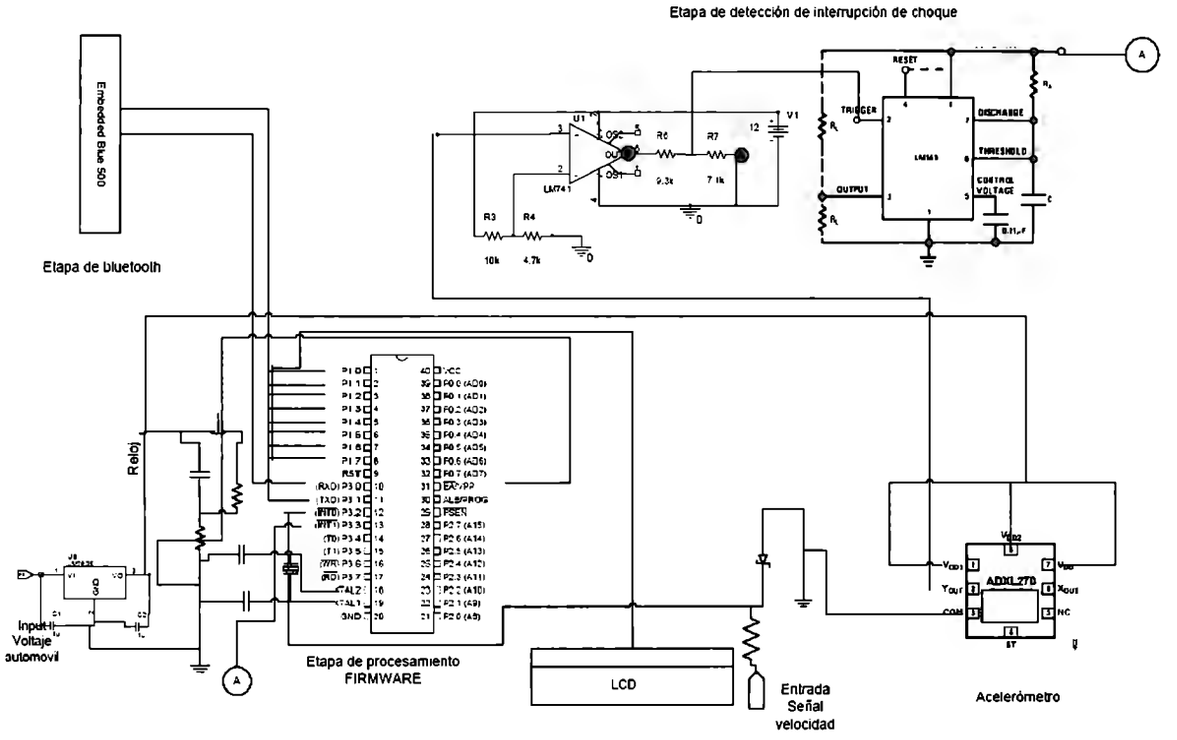


Figura 31: Diagrama Eléctrico Prototipo

5.8 Programación

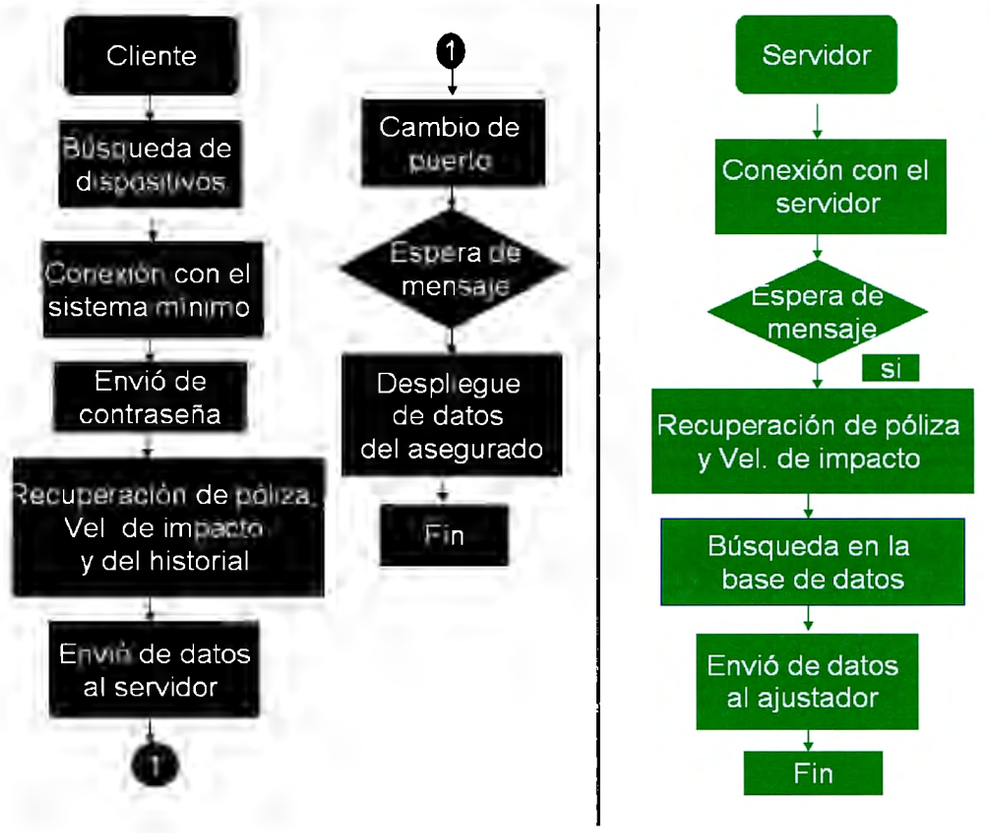
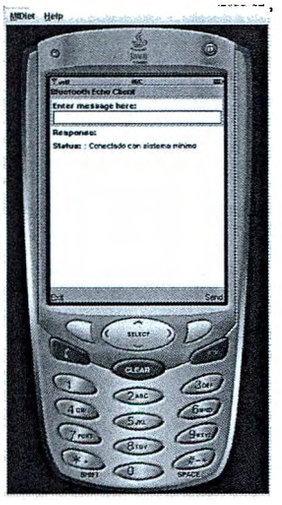


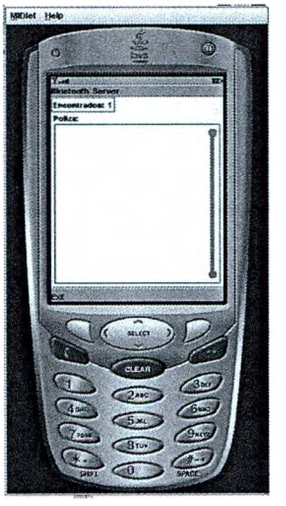
Figura 32: Diagrama de flujo software programación

En la Figura 32, podemos ver el diagrama de flujo de la programación del software. El programa lo dividimos en tres partes; la primera parte se basa en la comunicación del sistema mínimo y el celular. Con la implementación del programa logramos que el celular se conectara con el sistema mínimo por medio de la tecnología bluetooth. Lo primero que hace es buscar los dispositivos que están dentro de su alcance, el ajustador debe seleccionar la aplicación bluetooth que le corresponde al sistema mínimo, y después este, le demandará una contraseña al ajustador, para poderse conectar el sistema con el celular. Al momento en el que se establezca la conexión, el microcontrolador le enviará la póliza, la velocidad de impacto y el historial de velocidades, las cuales serán enviadas al servidor por medio de SMS. En este momento, entra la segunda parte de nuestro proyecto, en el cual el teléfono celular que recibe la información, se conectará al servidor y por medio de la póliza, buscará

en la base de datos la información del asegurado, y de esta manera, poder enviarlos al teléfono celular del ajustador La tercera parte consta de la programación del servidor y hacer la base de datos, sin embargo para la base de datos usamos MYSQL lo cual facilitó esta parte debido a que con este programa únicamente renombras los campos que necesites y llenas tu base de datos.

El código de programación se encuentra dentro de los anexos de este documento.





VI. Conclusiones y Trabajo a Futuro

Hemos cumplido con los objetivos planteados al inicio de este proyecto, y a la vez hemos logrado definir los pasos a seguir en la siguiente etapa del mismo.

A lo largo del proyecto, hemos cambiado el alcance, ya que en la etapa de factibilidad del mismo, se asumen muchas cosas, y realmente no se tiene un conocimiento profundo en cuanto a la operabilidad de los dispositivos, costos, ni eficiencia de los mismos. Igualmente, sobre la marcha fuimos definiendo el diseño del proyecto, comenzando por bloques pequeños, e incorporándolos al sistema completo.

Logramos hacer las pruebas que consideramos críticas en la elaboración del proyecto que fueron:

- Transmisión de datos por medio de la tecnología Bluetooth
- Obtención de la velocidad de un automóvil para ser procesada y manejada por el sistema mínimo.
- Realización del algoritmo para el sistema mínimo

Y por último, pudimos presentar un prototipo que realizaba las siguientes funciones:

- Obtener la velocidad de un automóvil en movimiento y monitorearla constantemente
- Recuperar la velocidad a la que un vehículo se impacta.
- Procesar y enviar esta información vía Bluetooth hacia un dispositivo celular.
- Crear un ambiente de interfaz gráfica con el cual el usuario pueda observar los datos enviados por el sistema mínimo que se encuentra en el automóvil.

- Enviar esta información vía SMS a otro dispositivo móvil que se conectará a una base de datos, regresando información adicional al ajustador referentes al asegurado y al auto.

Como trabajo futuro se tiene pensado implementar una base de tiempo para el sistema. Esto quiere decir que además de detectar la velocidad de colisión en caso de un accidente, se podrá determinar el momento exacto en que esto ha ocurrido. Para realizar esto será necesario contar con un circuito que tenga la capacidad de llevar un registro de tiempo. Se ha realizado una investigación de los posibles dispositivos y se determinó que el DS1305 es el que mejor satisface las necesidades del prototipo.

Este dispositivo funciona como un reloj auxiliar que llevará el registro de tiempo, cuenta con la capacidad de registrar desde hora exacta hasta días y meses. Para este circuito es necesario contar con otro oscilador a mayor frecuencia 32.768kHz para configurarlo y que dicho oscilador sea de alta precisión. El DS1305 debe ser configurado con la hora, día y mes exacto por medio de una interfase serial con el microcontrolador. El microcontrolador le mandará byte por byte escribiendo en las direcciones de memoria asignadas para cada campo de modo que el integrado quede configurado como reloj.

Una vez configurado el integrado este funcionará como reloj auxiliar, de modo que cuándo sea detectada la colisión en el sistema se hará una petición de lectura por medio de comunicación serial al reloj auxiliar, y se recuperará al igual que en la escritura campo por campo, es decir los bytes establecidos para los segundos, minutos, horas, días y mes.

El integrado a su vez cuenta con un sistema auxiliar de alimentación por medio de una batería auxiliar para que nunca pierda su calibración aún cuando se interrumpa su alimentación principal.

Otro punto importante para que el sistema de detección de velocidad sea menos vulnerable a alteraciones, será implementar un sistema de alimentación auxiliar o respaldo. Esto funcionará únicamente con un sistema de conmutación de alimentación, es decir, que si por alguna razón se interrumpiera la alimentación de la batería del automóvil el sistema conmutará su alimentación a una batería auxiliar independiente. De esta forma se resuelve el problema que en caso de que la colisión ocasione la desconexión del sistema de detección de velocidad o que el usuario desconecte el sistema se evite la perdida de la información adquirida.

También se tiene planeado implementar un sistema que limite la cantidad de corriente que ingresa al sistema de detección de velocidad que se encuentra principalmente alimentado de la batería del automóvil. Es necesario implementar esto debido a que las corrientes experimentadas dentro del sistema eléctrico del automóvil tienen picos muy altos cuando este es encendido, por lo tanto debe existir una etapa de "bypass" para que a pesar de estas variaciones el sistema mínimo de adquisición no sufra algún daño.

Se propone definir viabilidad en términos financieros del mismo, y realizar un estudio de mercado para buscar posibles inversionistas para poder comercializar nuestro prototipo. En los anexos se encuentra el desglose de precios del prototipo, al igual que los estados financieros supuestos, junto con la tasa de retorno.

VII. Referencias

- AGUILAR, Alejandro .Sistema eléctrico automotriz. México: Aconcagua Ediciones y Publicaciones, 1995. (reimpresión 1997)
- ALONSO, José Manuel. Circuitos eléctricos auxiliares. Madrid, España : Thomson, Paraninfo, 2002.
- CROUSE, William H. Equipo eléctrico y electrónico del automóvil. México, D.F. : Alfaomega, c1996
- FRANCO, Sergio. Design with operational amplifiers and analog integrated circuits . Boston : WCB/McGraw-Hill, c1998.
- KLOEDEN CN, McLean AJ, Moore VM, and Ponte G (1997) Travelling Speed and the Risk of Crash Involvement, NHMRC Road Accident Research Unit, The University of Adelaide
- MULLER, Nathan J, Tecnología Bluetooth, McGraw Hill, España, 2002.
- PEERSMAN Guillaume, CVETKOVIC Srba, IEEE Personal Communications, "The Global System for Mobile Communications Short Message Service" June 2000.
- RIBBENS, B. William, Understanding Automotive Electronics, USA, Newnes, 1998. pp. 187-294

Hojas de especificaciones:

- LM331
- LM2907
- ADC0804
- ADXL278

Páginas Web

- www.analogdevices.com
    - Hojas especificaciones ADXL278
- www.bluetooth.com
- www.analog.com
- http://www.parallax.com/detail.asp?product_id=30068
- www.alldatasheets.comhttp://www.nationaltech.net/faq.html
- http://answers.google.com/answers/threadview?id=523539
- http://www.nhtsa.dot.gov/cars/rules/rulings/AAirBagSNPRM/
- http://www.volpe.dot.gov/sdd/docs/2002/rail_cw_2002_6.pdf
- http://meas-spec.com/myMeas/download/pdf/english/application/MEAS%20G-levels.pdf
- http://hypertextbook.com/facts/2003/MichelleYee.shtml
- http://www.e-z.net/~ts/physics.htm
- http://www.chiroweb.com/archives/17/12/03.html
- http://hypertextbook.com/physics/mechanics/acceleration/
- http://today.java.net/pub/a/today/2004/07/27/bluetooth.html

ANEXOS

8.1 Programa Ensamblador PROVIEW32

```
BINARIO EQU 41H
BANDERA EQU 20H.0
EN   EQU P1.6
RW   EQU P1.5
RS   EQU P1.7
DAT  EQU P2

            ORG 0
                  JMP MAIN
            ORG 03H
              JMP EXT0
            ORG 13H
                  JMP EXT1
            ORG 1BH
              JMP TIMER1

            MAIN:
            CLR 20H.0
            MOV R1, #60H
            MOV TMOD, #00010001B

            MOV BINARIO, #0
   SETB IT0
            MOV IP, #00001101B
            MOV TH1, #03CH
            MOV TL1, #0B0H
         MOV R0, #20
            LCALL INIT_LCD
                        MOV A, #'V'
                        CALL WRITE_TEXT
                              MOV A, #'E'
                          CALL WRITE_TEXT
                          MOV A, #'L'
                          CALL WRITE_TEXT
                          MOV A, #'O'
                          CALL WRITE_TEXT
```

```
                    MOV A, #'C'
                    CALL WRITE_TEXT
                    MOV A, #'I'
                    CALL WRITE_TEXT
                    MOV A, #'D'
                    CALL WRITE_TEXT
                    MOV A, #'A'
                    CALL WRITE_TEXT
                    MOV A, #'D'
                    CALL WRITE_TEXT
                    MOV A, #':'
                    CALL WRITE_TEXT
                    MOV IE, #10011101B
          JMP $

;.............................
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
TIMER1:
MOV TH1, #03CH
MOV TL1, #0B0H
DJNZ R0, OTRA_VEZ
;;;CRITERIO;;;;;;;;;;;;
CALL CRITERIO
;;;;;;;;MODIFICACION;;;;;;;;;;;;;;
CJNE R1, #6AH, ALMACENA
MOV R1, #60H
ALMACENA:
MOV @R1, BINARIO
MOV A, R1
MOV R3,A
INC R1
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
MOV R0, #20
MOV 33H, BINARIO
MOV BINARIO, #0H
CALL BIN_BCD
CALL CONVIERTE
MOV A,30H
CALL SET_ADDRESS
CALL WRITE_TEXT
MOV A,31H
CALL WRITE_TEXT
MOV A, 32H
```

```
CALL WRITE_TEXT
OTRA_VEZ:
RETI
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
CRITERIO:
MOV A, BINARIO
CLR C
MOV R4, #210
SUBB A, R4
JC COMPARAR
MOV R4,#20
MOV A, BINARIO
SUBB A, R4
MOV BINARIO,A
JMP EXIT

COMPARAR:
MOV A, BINARIO
CLR C
MOV R4,#150
SUBB A,R4
JC COMPARAR2
MOV A, BINARIO
MOV R4,#15
SUBB A,R4
MOV BINARIO,A
JMP EXIT

COMPARAR2:
MOV A, BINARIO
CLR C
MOV R4,#90
SUBB A,R4
JC COMPARAR3
MOV A, BINARIO
MOV R4,#10
SUBB A,R4
MOV BINARIO,A
JMP EXIT

COMPARAR3:
CLR C
```

```
MOV A, BINARIO
MOV R4,#4
SUBB A,R4
MOV BINARIO, A
JMP EXIT
EXIT:CLR C
RET


;;;;;;;;EXTERNA1;;;;;;;;;;;;;;;
EXT1:
MOV IE, #10010000B
MOV SCON, #01000000B
MOV TMOD, #00100000B
MOV TH1, #0FDH
MOV TL1, #0FDH
SETB TR1
MOV A,#1011010B
MOV SBUF,A
JNB TI, $
CLR T1

MOV A,R3
MOV R1, A
MOV R3, #0
TX:
MOV A, @R1
MOV SBUF, A
JNB TI, $
DEC R1
;;;;
INC R3
;;;;
CLR TI
CJNE R1, #5FH, TX

;;;;;;;;;;NUEVO;;;;;;;;;;;;
MOV R1, #6AH
DNUEZ:
CJNE R3, #0AH, TX2
;;;MOV IE, #10011101B
;;MOV TH1, #03CH
```

```
;;;MOV TL1, #0B0H
;;MOV R0, #20
;;SETB TR1
RETI
TX2:
MOV A, @R1
MOV SBUF, A
JNB TI, $
DEC R1
CLR TI
INC R3
JMP DNUEZ

;;;;;;;;;;;;;EXTERNA0;;;;;;;;;;;;;;;;;;;
EXT0:
JB 20H.0,NOCORRETIMER
CPL 20H.0
SETB TR1
NOCORRETIMER:
INC BINARIO
RETI
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
CONVIERTE:
MOV A, 30H
ADD A, #30H
MOV 30H,A
MOV A, 31H
ADD A, #30H
MOV 31H,A
MOV A, 32H
ADD A, #30H
MOV 32H,A
RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
BIN_BCD:
MOV 30H,#0
MOV 31H,#0
MOV 32H,#0

OTRA:MOV A,#0
CJNE A,33H,SEGUIR
RET
```

```
SEGUIR:
INC 32H
MOV A,32H
CJNE A,#10,FINAL1
MOV 32H,#0
INC 31H
MOV A,31H
CJNE A,#10,FINAL1
MOV 31H,#0
INC 30H
MOV A,30H
CJNE A,#10,FINAL1
MOV 30H,#0

FINAL1:DEC 33H
JMP OTRA
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
INIT_LCD:                     ; Inicializa la pantalla LCD

        MOV TMOD, #00010001B    ; modo 2 timer uno y timer0 modo1

              CALL ESPERA_5MSEG

        CLR RS              ; RS y RW tienen que estar en cero
        CLR RW
        MOV P2, #38H
        SETB EN
        CLR  EN
        CALL ESPERA_5MSEG

        CLR RS                                  ;PRENDE EL DISPLAY
        CLR RW
        MOV P2,#0EH
        SETB EN
        CLR EN
        CALL ESPERA_5MSEG

        CALL CLEAR_LCD

                        CLR RS   ; Define modo de entrada
        CLR RW
```

```
        MOV P2, #06H
        SETB EN
        CLR EN
        CALL ESPERA_5MSEG
             RET

CLEAR_LCD:
                              CLR RS
        CLR RW
        MOV P2,#01B
        SETB EN
        CLR EN
        CALL ESPERA_5MSEG
        RET

SET_ADDRESS:
                              CLR RS
        CLR RW
                              MOV P2,#8AH
           SETB EN
         CLR EN
         CALL ESPERA_5MSEG
        RET

 WAIT_LCD:
        SETB EN
                              NOP
                              CLR EN
                              CLR RS
                              SETB RW
                              SETB EN
                              SETB P2.7
                              JB P2.7,$
                              CLR EN
                              SETB RS
                              CLR RW
                              RET

ESPERA_5MSEG:
        CLR TF0
        CLR TR0             ; paramos el timer 0
        MOV TH0,#0ECH       ;timer 0 en 5ms
```

```
        MOV TL0,#78H
        SETB TR0          ;arrancamos el timer 0
        JNB TF0,$
        CLR TF0
        CLR TR0           ; paramos el timer 0
        RET

WRITE_TEXT:

                          SETB  EN          ;Habilita el puerto
                          SETB  RS          ;RS = 1 -> Despliega el texto
                          MOV   DAT, A            ;Dato (letra) en el puerto 2
                          CLR   EN          ;Pulso para desplegar
                          CALL  WAIT_LCD    ;Espera a terminar la instrucción
                          RET
FINAL:
```

8.2 Programación JAVA

ProxyMidlet

```
package runnable;

import java.io.*;
import javax.bluetooth.RemoteDevice;
import javax.microedition.io.*;
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import java.util.Vector;
import javax.wireless.messaging.*;
import utils.*;

public class ProxyMidlet extends MIDlet implements CommandListener,
MessageListener{

   private Form form;
   private StringItem status;

   private Command connectCmd, exitCmd;

   private Display display;
```

```
private DeviceFinder deviceFinder;

private ChoiceGroup handlerList;
private List deviceList;

private Vector deviceVector;

private BluetoothLink bluetoothLink;
private String btAddress;
private String btData;


private MessageConnection msgConnection;
private String msgReceived;


public ProxyMidlet(){

   btAddress = "";
   btData = "";
   msgReceived = "";

   deviceList = new List("Device List", List.IMPLICIT);
   form = new Form("Proxy Online");

   deviceVector = new Vector();

   handlerList = new ChoiceGroup("Msg list", ChoiceGroup.EXCLUSIVE);
   status = new StringItem("status:", "done");
   form.append(handlerList);
   form.append(status);
   exitCmd = new Command("Exit", Command.EXIT, 1);
   form.addCommand(exitCmd);
   connectCmd = new Command("Connect", Command.SCREEN, 1);
   deviceList.addCommand(connectCmd);

   deviceList.setCommandListener(this);
   form.setCommandListener(this);

}
```

```
protected void startApp() {

   display = Display.getDisplay(this);

   showDevices();

   try {
      msgConnection = (MessageConnection)Connector.open("sms://:6001");
      // Register the listener for inbound messages.
      msgConnection.setMessageListener(this);

   }catch (IOException ioExc){
      System.out.println("Server connection could not be obtained");
      ioExc.printStackTrace();
   }
} // end of startApp()

protected void pauseApp() {}

protected void destroyApp(boolean unconditional){

   try {
      if (msgConnection != null) {
         msgConnection.setMessageListener(null);
         msgConnection.close();
      }
   }catch (IOException e) {
      //  Handle the exception...
      e.printStackTrace();
   }
   notifyDestroyed();
}


public void commandAction(Command c, Displayable d) {

   if (c == exitCmd)
      destroyApp(true);

   else if(c == connectCmd){
```

```
        if(deviceVector.size() > 0){

          btAddress = deviceList.getString(deviceList.getSelectedIndex());

          display.setCurrent(form);

        }else
          destroyApp(true);

      }

    } // end of commandAction()

    private void showDevices(){

      deviceFinder = new DeviceFinder();

      while(!deviceFinder.searchFinished()){}

      deviceVector = deviceFinder.getDeviceVector();
      deviceList.deleteAll();
      for(int i = 0; i < deviceVector.size(); i++){


deviceList.append(((RemoteDevice)deviceVector.elementAt(i)).getBluetoothAddress(),null
);
      }

      display.setCurrent(deviceList);
    }

    public void notifyIncomingMessage(MessageConnection messageConnection) {

      Message msg = null;
      // Try reading (maybe block for) a message
      try {
        msg = messageConnection.receive();
      }catch (Exception e) {
        // Handle reading errors
        System.out.println("processMessage.receive " + e);
      }
```

```
// Process the received message
if (msg instanceof TextMessage){

   TextMessage tmsg = (TextMessage)msg;

   msgReceived = tmsg.getPayloadText();

}else {

   // process received message
   if (msg instanceof BinaryMessage) {

      BinaryMessage bmsg = (BinaryMessage)msg;

      byte[] data = bmsg.getPayloadData();

      // Handle the binary message...
      msgReceived = data.toString();

   }
}

handlerList.append(msgReceived, null);

display.setCurrent(form);


bluetoothLink = new BluetoothLink(btAddress);
bluetoothLink.start();
while(!bluetoothLink.isOpen()){}

bluetoothLink.write(msgReceived);
btData = bluetoothLink.read();


try{

      TextMessage tmsg =
(TextMessage)msgConnection.newMessage(MessageConnection.TEXT_MESSAGE);
      tmsg.setAddress("sms://5539770726:6001");
      tmsg.setPayloadText(btData);
```

```
            msgConnection.send(tmsg);
        }catch (Exception exc){
            exc.printStackTrace();
        }
    }

}
```

DeviceFinder

```
package utils;


import java.io.*;
import javax.microedition.io.*;
import javax.bluetooth.*;
import java.util.Vector;

public class DeviceFinder implements DiscoveryListener{
    private String srchServiceName;   // the name of the desired service

    private DiscoveryAgent agent;

    // stores the remote devices found during the device search
    private Vector deviceVector;
    private boolean searchFinished;

    public DeviceFinder(){

        searchFinished = false;
        try {
            // get the discovery agent, by asking the local device
            LocalDevice local = LocalDevice.getLocalDevice();
            agent = local.getDiscoveryAgent();

            // initialize device search data structure

            setDeviceVector(new Vector());
            // start the searches: devices first, services later
```

```
            System.out.println("Searching for Devices...");
            agent.startInquiry(DiscoveryAgent.GIAC, this);   // non-blocking
        } catch (Exception e) {
            System.out.println("Search Error");
            e.printStackTrace();
        }

    }

// ------------------------- device search methods ---------------

  /* deviceDiscovered() and inquiryCompleted() are called
     automatically during the device search initiated by the
     DiscoveryAgent.startInquiry() call.
   */

  /* A matching device was found during the device search.
     Only store it if it's a PC or phone. */
    public void deviceDiscovered(RemoteDevice dev, DeviceClass cod){

        System.out.println("Device Name: " +  getDeviceName(dev) + ", BluetoothAddress:"
+ dev.getBluetoothAddress());

        int majorDC = cod.getMajorDeviceClass();
        int minorDC = cod.getMinorDeviceClass();   // not used in the code
        //System.out.println("Major Device Class: " + majorDC + "; Minor Device Class: " +
minorDC);

        // restrict matching device to PC or Phone
        //if ((majorDC == 0x0100) || (majorDC == 0x0200))
            getDeviceVector().addElement(dev);
        //else
            //System.out.println("Device not PC or phone, so rejected");
    }

/* Return the 'friendly' name of the device being examined,
     or "Device ??" */
    private String getDeviceName(RemoteDevice dev){

        String devName;

        try {
```

```
            devName = dev.getFriendlyName(false);  // false to reduce connections
        } catch (IOException e){
            devName = "Device ??";
        }

        return devName;
    }

// device search has finished; start the services search
    public void inquiryCompleted(int inqType){

        showInquiryCode(inqType);
        System.out.println("No. of Matching Devices: " + getDeviceVector().size());
        searchFinished = true;

    }


    private void showInquiryCode(int inqCode){
        if(inqCode == INQUIRY_COMPLETED)
            System.out.println("Device Search Completed");
        else if(inqCode == INQUIRY_TERMINATED)
            System.out.println("Device Search Terminated");
        else if(inqCode == INQUIRY_ERROR)
            System.out.println("Device Search Error");
        else
            System.out.println("Unknown Device Search Status: " + inqCode);
    }

    public void servicesDiscovered(int i, ServiceRecord[] serviceRecord) {
    }

    public void serviceSearchCompleted(int i, int i0) {
    }


    public boolean searchFinished(){

        return searchFinished;
    }
```

```
    public Vector getDeviceVector() {
        return deviceVector;
    }

    public void setDeviceVector(Vector deviceVector) {
        this.deviceVector = deviceVector;
    }

}
```

BluetoothLink

```
/*
 * BluetoothLink.java
 *
 * Created on 19 de marzo de 2007, 06:32 PM
 *
 * To change this template, choose Tools | Template Manager
 * and open the template in the editor.
 */

package utils;

import javax.bluetooth.*;
import javax.microedition.io.*;
import java.io.*;

public class BluetoothLink extends Thread {

    private StreamConnection conn;
    private InputStream in;
    private OutputStream out;

    private boolean isOpen;
    private String url;

    /** Creates a new instance of BluetoothLink */
    public BluetoothLink(String bluetoothAddress) {

        isOpen = false;
```

```
    url = "btspp://" + bluetoothAddress +
":1;master=false;encrypt=false;authenticate=false";

  }

  public boolean isOpen(){

    return isOpen;
  }

  public void run(){

    this.open();
  }

  public boolean open(){

    boolean success = false;

    try {

      conn = (StreamConnection) Connector.open(url);

      out = conn.openOutputStream();
      in = conn.openInputStream();
      success = true;
      isOpen = true;
    } catch (IOException ex) {
      ex.printStackTrace();
    }

    return success;
  }

  public boolean close(){

    boolean success = false;

    try {

      in.close();
      out.close();
```

```
      conn.close();
      success = true;
      isOpen = false;
    } catch (IOException ex) {
      ex.printStackTrace();
    }

    return success;
}

public String read(){

    byte[] data = null;
    try {
      int len = in.read();    // get the message length
      if (len <= 0) {
        System.out.println("Message Length Error");
        return null;
      }

      data = new byte[len];
      len = 0;
      // read the message, perhaps requiring several read() calls
      while (len != data.length) {
        int ch = in.read(data, len, data.length - len);
        if (ch == -1) {
          System.out.println("Message Read Error");
          return null;
        }
        len += ch;
      }
    } catch (IOException e) {
      System.out.println("readData(): " + e);
      return null;
    }


    return new String(data);  // convert byte[] to String
}

public boolean write(String msg){
```

```
        System.out.println("sendMessage: " + msg);
        try {
            out.write(msg.length());
            out.write(msg.getBytes());
            out.flush();
            return true;
        } catch (Exception e) {
            System.out.println("sendMessage(): " + e);
            return false;
        }

    }

    public int readByte(){

        int data = -1;

        try {

            data = in.read();
        } catch (IOException ex) {
            ex.printStackTrace();
        }

        return data;
    }

    public void writeByte(int data){

        try {

            out.write(data);
            out.flush();
        } catch (IOException ex) {
            ex.printStackTrace();
        }

    }


}
```

ClientMidlet

```
import java.io.*;
import javax.bluetooth.RemoteDevice;
import javax.microedition.io.*;
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import java.util.Vector;
import javax.wireless.messaging.*;
import utils.*;


public class ClientMidlet extends MIDlet implements CommandListener,
MessageListener{

   private Form loginForm, retrivedInfoForm, databaseInfoForm;
   private StringItem status, contract, impactSpeed, name, brand, year, modelNumber,
accidentNumber;
   private TextField pwd;
   private Command okCmd, connectCmd, reconnectCmd, sendCmd, exitCmd, returnCmd;

   private Display display;

   private DeviceFinder deviceFinder;

   private ChoiceGroup speedList;


   private List deviceList;

   private Vector deviceVector;

   private BluetoothLink bluetoothLink;

   private MessageConnection msgConnection;
   private String msgReceived;

   private String contractString;
   private String speedString;

   public ClientMidlet(){
```

```
msgReceived = null;

deviceList = new List("Device List", List.IMPLICIT);
loginForm = new Form("login");
retrivedInfoForm = new Form("Informacion");
databaseInfoForm = new Form("Database Informacion");


deviceVector = new Vector();

okCmd = new Command("Ok", Command.SCREEN, 1);
deviceList.addCommand(okCmd);


pwd = new TextField("Password: ", "0000", 25, TextField.ANY);
connectCmd = new Command("Connect", Command.SCREEN, 1);
loginForm.append(pwd);
loginForm.addCommand(connectCmd);


contract = new StringItem("Contract:", "");
impactSpeed = new StringItem("Impact speed:", "");
speedList = new ChoiceGroup("Speed list", ChoiceGroup.EXCLUSIVE);
status = new StringItem("status:", "done");
reconnectCmd = new Command("Reconnect", Command.SCREEN, 1);
sendCmd = new Command("Send", Command.SCREEN, 2);
retrivedInfoForm.append(contract);
retrivedInfoForm.append(impactSpeed);
retrivedInfoForm.append(speedList);
retrivedInfoForm.append(status);
retrivedInfoForm.addCommand(reconnectCmd);
retrivedInfoForm.addCommand(sendCmd);


name = new StringItem("Name:", "");
brand = new StringItem("Brand:", "");
year = new StringItem("Year:", "");
modelNumber = new StringItem("Model Number:", "");
accidentNumber = new StringItem("Accident Number:", "");
exitCmd = new Command("Exit", Command.EXIT, 1);
```

```
    returnCmd = new Command("Return", Command.SCREEN, 2);
    databaseInfoForm.append(name);
    databaseInfoForm.append(brand);
    databaseInfoForm.append(year);
    databaseInfoForm.append(modelNumber);
    databaseInfoForm.append(accidentNumber);
    databaseInfoForm.addCommand(exitCmd);
    databaseInfoForm.addCommand(returnCmd);



    loginForm.setCommandListener(this);
    deviceList.setCommandListener(this);
    retrivedInfoForm.setCommandListener(this);
    databaseInfoForm.setCommandListener(this);

}


protected void startApp() {

    display = Display.getDisplay(this);

    showDevices();

    try {
      msgConnection = (MessageConnection)Connector.open("sms://:6001");
      // Register the listener for inbound messages.
      msgConnection.setMessageListener(this);

    }catch (IOException ioExc){
      System.out.println("Server connection could not be obtained");
      ioExc.printStackTrace();
    }

} // end of startApp()

protected void pauseApp() {}

protected void destroyApp(boolean unconditional){

    try {
```

```
            if (msgConnection != null) {
                msgConnection.setMessageListener(null);
                msgConnection.close();
            }
        }catch (IOException e) {
            // Handle the exception...
            e.printStackTrace();
        }
        notifyDestroyed();
    }


    public void commandAction(Command c, Displayable d) {

        if (c == exitCmd)
            destroyApp(true);

        else if (c == sendCmd){

            status.setText("transfering data");

            try{

                TextMessage tmsg =
(TextMessage)msgConnection.newMessage(MessageConnection.TEXT_MESSAGE);
                tmsg.setAddress("sms://5539887905:6001");
                tmsg.setPayloadText(contractString + "," + speedString);
                msgConnection.send(tmsg);
            }catch (Exception exc){
                exc.printStackTrace();
            }


        }else if(c == okCmd){


            if(deviceVector.size() > 0){

                bluetoothLink = new
BluetoothLink(deviceList.getString(deviceList.getSelectedIndex()));
                bluetoothLink.start();
```

```
            display.setCurrent(loginForm);

        }else
            destroyApp(true);

    }else if(c == connectCmd){

        msgReceived = null;

        if(bluetoothLink.isOpen()){

            bluetoothLink.write(pwd.getString());

            contractString = "" + bluetoothLink.readByte();
            speedString = "" + bluetoothLink.readByte();

            contract.setText(contractString);
            impactSpeed.setText(speedString);

            for(int i = 0; i < 5; i++)
                speedList.append("" + bluetoothLink.readByte(), null);

            status.setText("Done");

        }else
            status.setText("Not connected");

        display.setCurrent(retrivedInfoForm);

    }else if(c == reconnectCmd){

        status.setText("Closing connection");

        bluetoothLink.close();

        showDevices();

    }else if(c == returnCmd){

        display.setCurrent(retrivedInfoForm);

    }
```

```
    } // end of commandAction()

    private void showDevices(){

        deviceList.deleteAll();
        display.setCurrent(deviceList);

        deviceFinder = new DeviceFinder();

        while(!deviceFinder.searchFinished()){}

        deviceVector = deviceFinder.getDeviceVector();
        for(int i = 0; i < deviceVector.size(); i++){


deviceList.append(((RemoteDevice)deviceVector.elementAt(i)).getBluetoothAddress(),null
);
        }


    }

    public void notifyIncomingMessage(MessageConnection messageConnection) {

        Message msg = null;
        //  Try reading (maybe block for) a message
        try {
            msg = messageConnection.receive();
        }catch (Exception e) {
            // Handle reading errors
            System.out.println("processMessage.receive " + e);
        }

        // Process the received message
        if (msg instanceof TextMessage){

            TextMessage tmsg = (TextMessage)msg;
            msgReceived = tmsg.getPayloadText();
        }else {
            // process received message
            if (msg instanceof BinaryMessage) {
```

```
            BinaryMessage bmsg = (BinaryMessage)msg;
            byte[] data = bmsg.getPayloadData();
            // Handle the binary message...
            msgReceived = data.toString();
        }
    }

    int stringIndex = 0;

    stringIndex = msgReceived.indexOf(",");
    name.setText(msgReceived.substring(0,stringIndex));
    msgReceived = msgReceived.substring(stringIndex+1, msgReceived.length());

    stringIndex = msgReceived.indexOf(",");
    brand.setText(msgReceived.substring(0,stringIndex));
    msgReceived = msgReceived.substring(stringIndex+1, msgReceived.length());

    stringIndex = msgReceived.indexOf(",");
    year.setText(msgReceived.substring(0,stringIndex));
    msgReceived = msgReceived.substring(stringIndex+1, msgReceived.length());

    stringIndex = msgReceived.indexOf(",");
    modelNumber.setText(msgReceived.substring(0,stringIndex));
    msgReceived = msgReceived.substring(stringIndex+1, msgReceived.length());

    stringIndex = msgReceived.indexOf(",");
    accidentNumber.setText(msgReceived.substring(0,stringIndex));


    status.setText("done");
    display.setCurrent(databaseInfoForm);

  }

}
```

8.3 Financieros

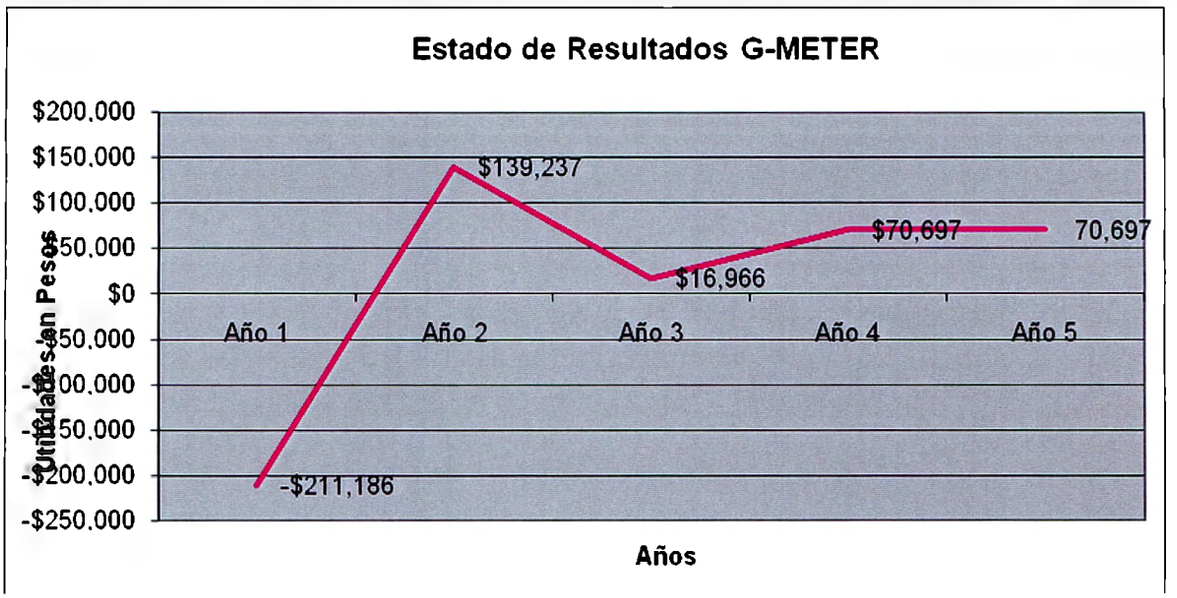
Costo Total:

Prototipo	**cto**
Costos variables	
Tarjeta bluetooth	1,200.00
Atmel 89c51	45.00
Cristal de cuarzo	13.00
LCD	115.00
Capacitores	10.00
Potenciometro	7.00
Placa fenólica	45.00
Resistencias	8.00
Bases	4.00
MC7805	6.00
Diodo zener	3.00
Push button	3.00
Base bluetooth	35.00
Cable wire rap	25.00
ADXL278 (acelerometro)	115.50
Total	**$1,634.50**
Total + Utilidad	**$3,269.00**

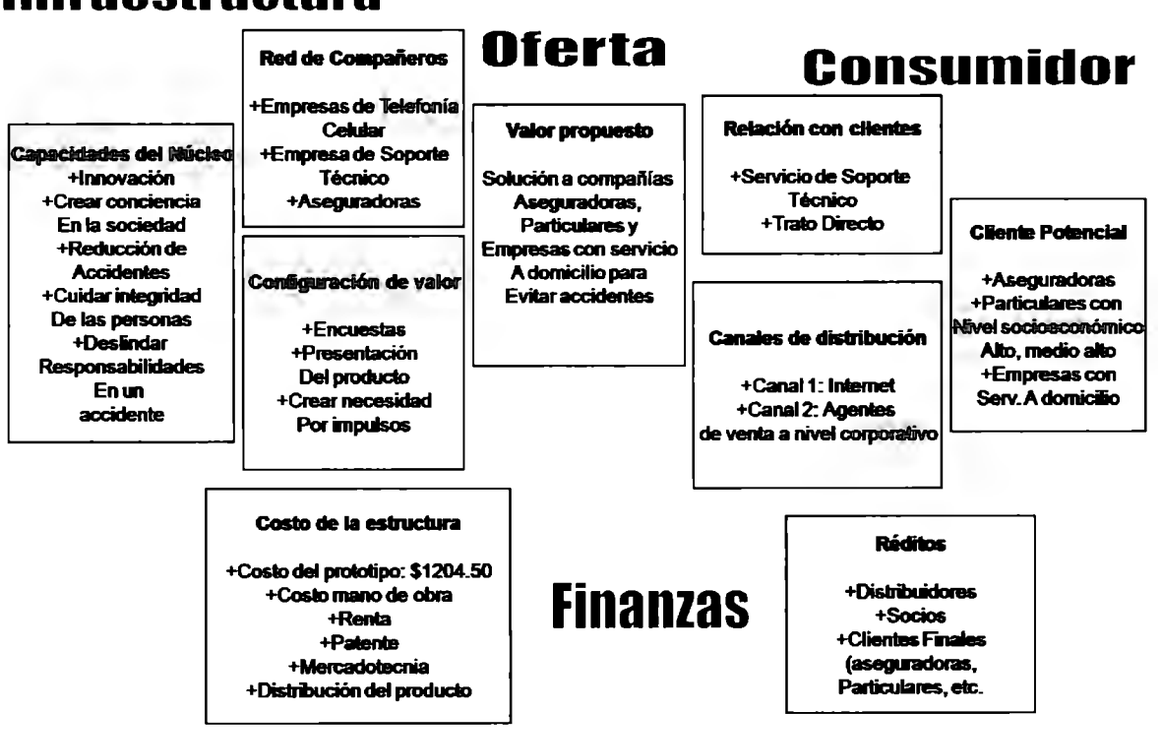
TIR:

Inversión	**Año 1**	**Año 2**	**Año 3**	**Año 4**	**Año 5**
$0	-$211,186	$139,237	$16,966	$70,697	70,697

VPN	TIR
$25	18%



Modelo de Negocios:



Planeación financiera de Strophic a 1 año

Del 22 de Abril del 2007 al 22 de Abril del 2012

		AÑO 1								AÑO 2				AÑO 3		AÑO 4		AÑO 5	
Meses/periodo		1 1	2 1	3 1	4 1	5 1	6 1	T3 3	T4 3	T1 3	T2 3	T3 3	T4 3	S1 6	S2 6	S1 6	S2 6	S1 6	S2 6
Producto 1 : G-meter	Unidades por mes	10	15	9	9	11	13	45	60	60	75	150	165	200	200	250	250	250	250
	$ 3,269	$ 32,690	$ 49,035	$ 29,421	$ 29,421	$ 35,959	$ 42,497	$ 147,105	$ 196,140	$ 196,140	$ 245,175	$ 490,350	$ 539,385	$ 653,800	$ 653,800	$ 817,250	$ 817,250	$ 817,250	$ 817,250
Total de Ingresos		$ 32,690	$ 49,035	$ 29,421	$ 29,421	$ 35,959	$ 42,497	$ 147,105	$ 196,140	$ 196,140	$ 245,175	$ 490,350	$ 539,385	$ 653,800	$ 653,800	$ 817,250	$ 817,250	$ 817,250	$ 817,250
Egresos																			
Recurso Humanos																			
Director General	Cant	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
(Ensamblador)	$ 1,000	$ 1,000	$ 1,000	$ 1,000	$ 1,000	$ 1,000	$ 1,000	$ 3,000	$ 3,000	$ 3,000	$ 3,000	$ 9,000	$ 9,000	$ 18,000	$ 18,000	$ 18,000	$ 18,000	$ 18,000	$ 18,000
Directores de Area	Cant	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
(Ensambladores)	$ 1,000	$ 5,000	$ 5,000	$ 5,000	$ 5,000	$ 5,000	$ 5,000	$ 15,000	$ 15,000	$ 15,000	$ 15,000	$ 45,000	$ 45,000	$ 90,000	$ 90,000	$ 90,000	$ 90,000	$ 90,000	$ 90,000
Trabajadores(Ensambladores)	Cant	-	-	-	-	-	-	-	-	-	-	1	1	1	1	2	2	2	2
	$ 2,500	$ -	$ -	$ -	$ -	$ -	$ -	$ -	$ -	$ -	$ -	$ 7,500	$ 7,500	$ 15,000	$ 15,000	$ 30,000	$ 30,000	$ 30,000	$ 30,000
	Empleados totales	6	6	6	6	6	6	6	6	6	6	7	7	7	7	8	8	8	8
IMSS, SAR, INFONAVIT	Sueldos totales	6,000	6,000	6,000	6,000	6,000	6,000	18,000	18,000	18,000	18,000	61,500	61,500	123,000	123,000	138,000	138,000	138,000	138,000
	35%	$ 2,100	$ 2,100	$ 2,100	$ 2,100	$ 2,100	$ 2,100	$ 6,300	$ 6,300	$ 6,300	$ 6,300	$ 21,525	$ 21,525	$ 43,050	$ 43,050	$ 48,300	$ 48,300	$ 48,300	$ 48,300
Total de RH		$ 8,100	$ 8,100	$ 8,100	$ 8,100	$ 8,100	$ 8,100	$ 24,300	$ 24,300	$ 24,300	$ 24,300	$ 83,025	$ 83,025	$ 166,050	$ 166,050	$ 186,300	$ 186,300	$ 186,300	$ 186,300
Comisiones de ventas	5%	$ 1,635	$ 2,452	$ 1,471	$ 1,471	$ 1,798	$ 2,125	$ 7,355	$ 9,807	$ 9,807	$ 12,259	$ 24,518	$ 26,969	$ 32,690	$ 32,690	$ 40,863	$ 40,863	$ 40,863	$ 40,863
Renta	$ 7,000	$ 7,000	$ 7,000	$ 7,000	$ 7,000	$ 7,000	$ 7,000	$ 21,000	$ 21,000	$ 21,000	$ 21,000	$ 21,000	$ 21,000	$ 42,000	$ 42,000	$ 42,000	$ 42,000	$ 42,000	$ 42,000
Teléfono	$ 400	$ 400	$ 400	$ 400	$ 400	$ 400	$ 400	$ 1,200	$ 1,200	$ 1,200	$ 1,200	$ 1,200	$ 1,200	$ 2,400	$ 2,400	$ 2,400	$ 2,400	$ 2,400	$ 2,400
Compra de Materiales del Proto	$ 1,635	$ 16,345	$ 24,518	$ 14,711	$ 14,711	$ 17,980	$ 21,249	$ 73,553	$ 98,070	$ 98,070	$ 122,588	$ 245,175	$ 269,693	$ 326,900	$ 326,900	$ 408,625	$ 408,625	$ 408,625	$ 408,625
Costos de Mercadotecnia	$ 30,000	$ 30,000	$ 30,000	$ 30,000	$ 30,000	$ 30,000	$ 30,000	$ 30,000	$ 30,000	$ 10,000	$ 10,000	$ 10,000	$ 10,000	$ 10,000	$ 10,000	$ 10,000	$ 10,000	$ 10,000	$ 10,000
Gastos extras	30%	$ 2,920	$ 3,166	$ 2,871	$ 2,871	$ 2,969	$ 3,067	$ 9,497	$ 10,232	$ 10,232	$ 10,968	$ 32,263	$ 32,998	$ 59,622	$ 59,622	$ 68,149	$ 68,149	$ 68,149	$ 68,149
Total de egresos		$96,400	$75,635	$64,553	$64,553	$68,247	$71,941	$166,904	$194,609	$174,609	$202,314	$417,180	$444,885	$639,662	$639,662	$758,336	$758,336	$758,336	$758,336
Utilidad antes de impuestos		- 33,710	- 26,600	- 35,132	- 35,132	- 32,288	- 29,444	- 19,799	1,531	21,531	42,861	73,170	94,500	14,138	14,138	58,914	58,914	58,914	58,914
Pago de impuestos	40%	-	-	-	-	-	-	-	612	8,612	17,144	29,268	37,800	5,655	5,655	23,566	23,566	23,566	23,566
Utilidad después de impuestos		- 33,710	- 26,600	- 35,132	- 35,132	- 32,288	- 29,444	- 19,799	919	12,919	25,717	43,902	56,700	8,483	8,483	35,348	35,348	35,348	35,348
Caja a fin de mes	$	- 33,710	- 60,310	- 95,441	- 130,573	- 162,861	- 192,305	- 212,104	- 210,573	- 189,043	- 146,181	- 73,012	21,488	35,626	49,764	108,678	167,592	226,506	285,419
	Inversión Inicial																		