ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE
TELECOMUNICACIÓN
UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Proyecto Fin de Carrera

Plataforma Para El Seguimiento Publicitario Basado En RFID

AUTOR: Jorge Martínez Rechina
DIRECTOR: Javier Vales Alonso
Octubre / 2014

Autor: Jorge Martínez Rechina

E-mail: j.martinezrechina@gmail.com

Director: Javier Vales Alonso

E-mail director: javier.vales@upct.es

Título PFC:

Resumen: Este proyecto consiste en la localización y seguimiento mediante etiquetas RFID pasivas, en la que información sigue la trayectoria y velocidad del objeto.

Titulación: Ingeniero Técnico de Telecomunicaciones Especialidad Telemática

Fecha: Octubre 2014

Índice

Capítulo 1: Introducción y objetivos

1.1 Introducción

La tecnología RFID está muy presente en nuestras vidas, desde el control de animales o mascotas al control de inventario o como medida antirrobo en los comercios. Es una tecnología barata, duradera y fiable. No necesita contacto visual por lo que ofrece mayor comodidad respecto a tecnologías usadas para los mismos fines.

La tecnología RFID ha sido utilizada ampliamente para la localización, monitorización y seguimiento en muchas aplicaciones para la vida real. Es utilizada para llevar controles de inventario, hacer seguimientos en cadenas industriales de manera eficiente, para identificar productos o animales, o para hacer seguimiento de vehículos.

Normalmente estas aplicaciones están basadas en tecnología RFID activa, pero cada vez es más utilizada la tecnología RFID pasiva por su bajo coste y su mantenimiento más simple.

En éste proyecto se pretende utilizar esta tecnología en un sistema publicitario basado en la presencia de personas, es decir, se desea mostrar información publicitaria acorde con los intereses del usuario mientras esté dentro del área de cobertura del sistema.

1.2 Resultados obtenidos

La localización y control mediante tecnología RFID es algo común en cadenas industriales para controlar el proceso de fabricación, el principal problema de esta tecnología usada para la localización, según los resultados obtenidos, es la precisión de la posición.

En espacios reducidos es posible obtener falsos resultados si las antenas están muy próximas ya que las etiquetas RFID pueden ser detectadas por varias antenas simultáneamente y dar resultados erróneos.

Es posible mejorar la posición utilizando RFID activos o modificando la potencia de la señal que emiten las antenas pero se perdería cobertura para detectar las etiquetas RFID. Con una parametrización correcta del lector RFID para los entornos que se quieran controlar es posible obtener una localización bastante exacta en varias dimensiones del objeto que se quiera controlar.

1.3 División del proyecto

1.4 Trabajos relacionados

Hay muchos estudios basados en sistemas de localización en interiores mediante RFID activo. Abordan temas como la precisión en la estimación de localización en entornos donde las potencias de las señales recibidas son similares. Normalmente la cobertura de un sistema de posicionamiento es inversamente proporcional a la precisión al detectar la posición.

L. M. Ni, Y. Liu, Y. C. Lau yA. P. Patil. “LANDMARC: Indoor Location Sensing Using Active RFID”. En Wireless Networks, 2004.

P. Bahl y V. N. Padmanabhan. “RADAR: An In-Building RF-based User Location and Tracking System”. En INFOCOM, 2000.

Se han llevado a cabo investigaciones sobre la tecnología RFID pasiva aunque muy pocas sobre localización y seguimiento.

J. Zhou y J. Shi. “RFID Localization Algorithms and Applications — A Review”. En Journal of Intelligent Manufacturing, Vol. 20, No. 6, pp. 695–707, 2009.

Capítulo 2: Tecnologías básicas utilizadas

2.1 RFID

2.1.1 Introducción

RFID son las siglas de Radio Frequency IDentification. Es un término genérico que suele describir un sistema que transmite la identidad (en la forma de un número de serie único) de un objeto o persona inalámbricamente, usando radio frecuencia. Se agrupan bajo la categoría general de tecnologías de identificación automáticas.

A diferencia de la tecnología de código de barras UPC, la tecnología RFID no requiere contacto o línea de visión para la comunicación. Los datos RFID pueden ser leídos a través del cuerpo humano, ropa o materiales no metálicos.

2.1.2 Componentes

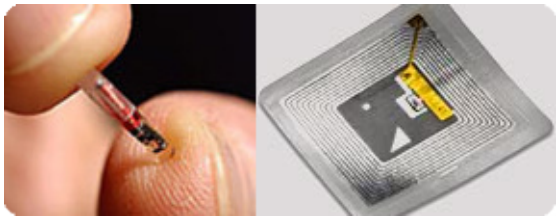
Etiquetas:

Una etiqueta RFID (RFID tag) es un microchip combinado con una antena en un paquete compacto, el paquete está estructurado para permitir que la etiqueta RFID sea adosado a algún objeto para su seguimiento.

La antena de la etiqueta recoge señales de un lector o escáner de RFID y luego devuelve la señal, normalmente con información adicional (como un número de serie único o alguna otra información personalizada).

Las etiquetas RFID pueden llegar a ser muy pequeños, desde el tamaño de un grano de arroz grande hasta el tamaño de un pequeño libro de bolsillo.

Hay tres tipos de etiquetas RFID, activas, pasivas y semi-pasivas que serán analizadas más adelante.



Antena / Lector:

El núcleo de los lectores RFID es, en esencia, un transcriptor de señales de radio que, al mismo tiempo, transmite y recibe señales de radio con la etiqueta RFID. Lo que ello nos indica es que un lector RFID tiene que hacer frente a una combinación de retos tecnológicos habituales de los sistemas de radio con otros retos no habituales, que son los típicos de la comunicación inalámbrica, pero bien conocidos por los técnicos en radares y expertos en comunicaciones pasivas, como es el caso de RFID.

Los lectores RFID, además de tener como características la exactitud, la eficiencia, la flexibilidad con un bajo ruido de radiación, deberán de tenerse muy en cuenta 6 Factores fundamentales y prácticos que nos ayudarán a escoger el lector RFID adecuado para nuestro trabajo:

- Sensibilidad. Deberá poder detectar señales procedentes de la etiqueta RFID de hasta -60 dBm de potencia, que es la mínima potencia que le puede llegar de una etiqueta RFID. Hoy, es posible detectar señales de hasta -115 dBm. Los buenos lectores RFID llegan a -80 dBm.
- Selectividad. Deberá poder seleccionar la señal procedente de la etiqueta RFID dentro de un vasto espectro de señales recibidas, algunas mucho más potentes que ella. Este aspecto resulta tan obvio como de vital importancia ya que las frecuencias RFID trabajan cerca de las frecuencias de telefonía y, si no se tiene en cuenta, pueden existir interferencias.
- Alcance Dinámico. Deberá de poder detectar y seleccionar señales procedentes, al mismo tiempo, de varias etiquetas RFID que estén a distancias diferentes, con lo que las potencias de emisión de las etiquetas pueden diferir en un factor mayor de 10.000 de diferencia.
- Trabajar bajo Normativas. En Europa, la normativa RFID permite operar entre 865,6-867,6 MHz de banda de frecuencia, con una potencia máxima del lector RFID de 2 W erp. En Europa la entidad reguladora es ETSI (European Telecommunications Standard Institute) con la normativa EN 302 208.

- Operatividad en entornos Densos de lectores RFID. Es una norma suplementaria, no obligatoria como una legislación, pero muy útil para poder soportar interferencias con otros lectores RFID. Para estar en conformidad con el estándar EPC Global Gen2 hace falta cumplir con esta norma.
- Inter-Operatividad multi-fabricante. Es una norma suplementaria, no obligatoria como una legislación, pero muy útil para poder trabajar con todo tipo de fabricantes de chips RFID y lectores RFID siendo intercambiables sus productos sin ningún problema. EPC Global tiene una certificación de inter-operatividad a disposición del mercado.

Middleware:

Middleware es la interfaz necesaria entre el lector y las bases de datos existentes y el software de gestión de información.

2.1.3 Tipos de RFID

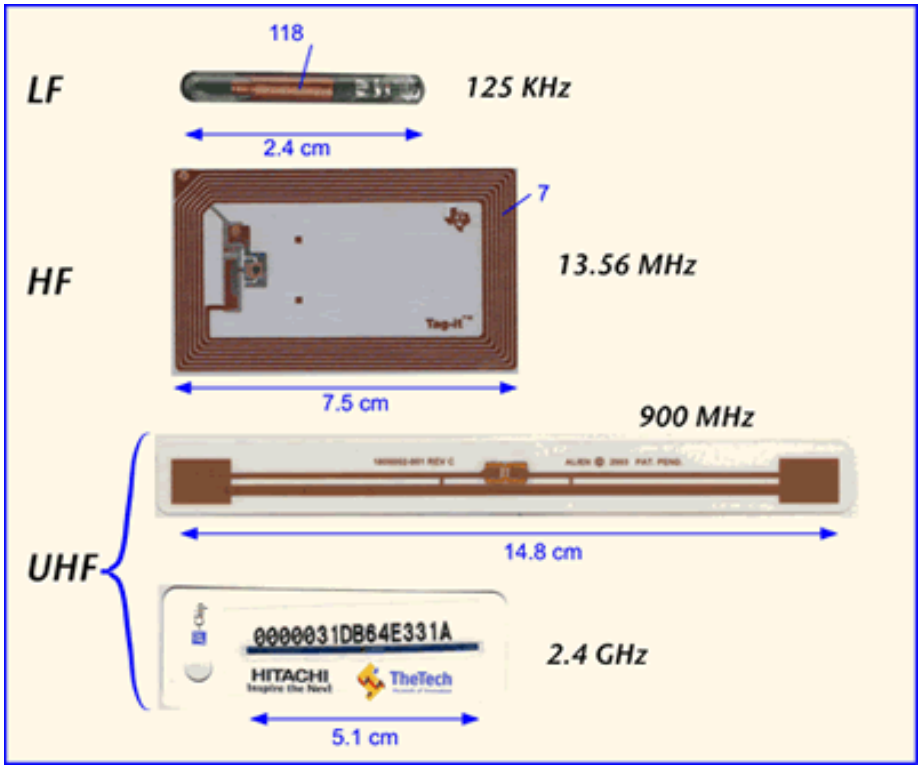
RFID pasivo y activo son diferentes tecnologías pero normalmente son evaluadas en conjunto. A pesar de que ambas utilizan la radio frecuencia para la comunicación entre la etiqueta y el lector, los medios de proporcionar la energía son

diferentes. RFID activo hace uso de una batería dentro de la etiqueta para proporcionar energía continua a los circuitos de radio frecuencia y de la etiqueta. RFID pasivo por el contrario, depende de la energía de radiofrecuencia transferidos del lector de etiquetas para la alimentación de la misma. Existen también un tercer tipo de etiqueta RFID, la etiqueta RFID semi-pasivo hace uso de una batería únicamente para alimentar el circuito, pero no para generar la frecuencia de emisión. La batería sirve para aumentar las propiedades y las características de la señal recibida.

RFID pasivo necesita señales fuertes del lector pero la fuerza de la señal rebotada de la etiqueta está en niveles bajos. RFID activos recibe señales de bajo nivel por etiqueta pero puede crear niveles de señal mayores hacia los lectores. Este tipo de RFID está siendo continuamente alimentado, ya sea dentro o fuera del área del lector. Las etiquetas activas consisten en sensores externos para el control de humedad, temperatura, movimiento así como otras condiciones.

Otra forma de clasificación de las etiquetas RFID viene dada por su frecuencia operativa, así, dentro del espectro de frecuencias estandarizadas para el RFID, nos encontramos una sustancial diferencia entre las etiquetas RFID que operan bajo cada una de ellas. Así, a medida que vamos aumentando de frecuencia en la que deseemos trabajar, las etiquetas RFID bajan significativamente de precio, llegando a unas diferencias de coste que pueden resultar significativas.

Aunque coloquialmente "a todo" se le llama RFID, y es cierto, las etiquetas que operan para cada frecuencia y los lectores RFID para leerlos, difieren en tecnología y prestaciones llegando a tener diferencias físicas (referente a las leyes de la física) completamente opuestas.



2.1.4 Funcionamiento RFID

El sistema RFID básico consiste en una antena, un transmisor-receptor y un transpondedor. La antena emite las señales de radio para proporcionar un medio de comunicación con el transpondedor (etiqueta RFID) y proporcionar la energía para comunicarse a los RFID pasivos.

Cuando una etiqueta RFID pasa a través del campo de la antena, se detecta la activación de la señal de la antena que "despierta" el chip RFID y transmite la información sobre su microchip para ser recogido por la antena.

2.2 Visual C# y Visual Studio

2.2.1 Introducción

C# (C Sharp) es parte de la plataforma .NET. C# es un lenguaje orientado a objetos simple, seguro, moderno, de alto rendimiento y con especial énfasis en internet y sus estándares (como XML). Es también la principal herramienta para programar en la plataforma .NET.

2.2.2 Visual C#

El lenguaje es muy sencillo, sigue el mismo patrón de los lenguajes de programación modernos. Incluye un amplio soporte de estructuras, componentes, programación orientada a objetos, manipulación de errores, recolección de basura, etc, que es construido sobre los principios de C++ y Java. Las clases son la base de los lenguajes de programación orientados a objetos, lo cual permite extender el lenguaje a un mejor modelo para solucionar problemas. C# contiene las herramientas para definir nuevas clases, sus métodos y propiedades, al igual que la sencilla habilidad para implementar encapsulación, herencia y polimorfismo, que son los tres pilares de la programación orientada a objetos. C# tiene un nuevo estilo de documentación XML que se incorpora a lo largo de la aplicación, lo que simplifica la documentación en línea de clases y métodos. C# soporta también interfaces, una forma de estipular los servicios requeridos de una clase. Las clases en C# pueden heredar de un padre pero puede implementar varias interfaces. C# también provee soporte para estructuras, un concepto el cual ha cambiado significativamente desde C++. Una estructura es un tipo restringido que no exige tanto del sistema operativo como una clase. Una estructura no puede heredar ni dar herencias de clases pero puede implementar una interfaz. C# provee características de componentes orientados, como propiedades, eventos y construcciones declaradas (también llamados atributos).

2.2.3 Plataforma .NET

La plataforma .NET es una plataforma de desarrollo de software con especial énfasis en el desarrollo rápido de aplicaciones, la independencia de lenguaje y la transparencia a través de redes.

La plataforma consta de las siguientes partes:

- Un conjunto de lenguajes de programación (C#, J#, JScript, C++ gestionado, Visual Básic.NET, y otros proyectos independientes).
- Un conjunto de herramientas de desarrollo
- Una librería de clases amplia y común para todos los lenguajes.
- Un sistema de ejecución de Lenguaje Común. (CLR).
- Un conjunto de servidores .NET

- Un conjunto de servicios .NET
- Dispositivos electrónicos con soporte .NET (PDA, teléfonos móviles, etc).

Todos los lenguajes que conformen con los estándares .NET, sin importar cual, podrán inter operar entre sí de forma totalmente transparente, las clases podrán ser heredadas entre unos lenguajes y otros, y se podrá disfrutar de polimorfismo entre lenguajes. Por ejemplo, si yo tengo una clase en C#, esta clase podrá ser heredada y utilizada en Visual Basic o JScript o cualquier lenguaje .NET. Todo esto es posible por medio de una de las características de .NET llamado Common Type System (CTS).

2.2.4 Visual Studio 2008

El entorno de desarrollo integrado (IDE) elegido para el proyecto es Microsoft Visual Studio 2008. Este IDE desarrollado por Microsoft para sistemas operativos Windows soporta varios lenguajes de programación, entre ellos Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET.

Las grandes ventajas de usar este entorno de desarrollo han sido la sencillez y claridad para crear un interfaz gráfico de usuario (GUI), así como la compatibilidad gracias a las librerías facilitadas por Alien Technology para la integración de sus lectores de etiquetas RFID. Además cuenta con un servidor SQL integrado para el uso de bases de datos.

2.3 Flash

2.3.1 Adobe Flash

Aplicación de edición multimedia desarrollado originalmente por Macromedia (ahora parte de Adobe) que utiliza principalmente gráficos vectoriales, pero también imágenes ráster, sonido, código de programa, flujo de vídeo y audio bidireccional para crear proyectos multimedia. Flash es el entorno desarrollador y Flash Player es la aplicación (la máquina virtual) utilizada para ejecutar los archivos generados con Flash.

Los proyectos multimedia pueden ser desde simples animaciones hasta complejas aplicaciones ya que, además de los gráficos, vídeos y sonidos, Flash

incorpora ActionScript, un completo lenguaje de programación que expande enormemente las posibilidades en los proyectos.

2.3.2 ActionScipt

ActionScript es el lenguaje de programación para crear scripts en Flash.

Podemos distinguir 2 tipos de scripts en un documento HTML. El primero es el que se ejecuta durante la descarga de la página en el navegador, existe la posibilidad de incluir contenido específico para aquellos navegadores que no incluyen scripts entre sus características. El segundo tipo son los ejecutados cada vez que un usuario (o el navegador) genera un "evento", por ejemplo la pulsación de un botón en un formulario.

Adobe ActionScript es el lenguaje de programación de la plataforma Adobe Flash. Originalmente desarrollado como una forma para que los desarrolladores programen en forma mas interactiva. La programación con ActionScript permite mucha más eficiencia en las aplicaciones de la plataforma Flash para construir animaciones de todo tipo, desde simples a complejas, ricas en datos e interfaces interactivas.

Una película de Flash está formada por una serie de fotogramas secuenciales llamada línea de tiempo, utilizando ActionScript en ciertos fotogramas clave podemos crear elementos interactivos o modificar los elementos de una película.

2.4 Alien Technology

Alien Technology fabrica etiquetas RFID en grandes cantidades y a un bajo coste. El alto rendimiento de la línea de etiquetas Squiggle desarrollado por Higgs3 está estableciendo un nuevo estándar de sensibilidad y de rendimiento de lectura para etiquetas de segunda generación. Los lectores ofrecen un rango mayor de lectura e interfaces de software que hace que sea más fácil la lectura de RFID en redes LAN o WAN.

Ofrece una amplia gama de lectores de RFID de segunda generación que son fáciles de implementar y administrar. El protocolo de lectura de Alien ofrece a los usuarios un conjunto de herramientas para implementar soluciones eficientes RFID. El

protocolo de lectura Alien es respaldado por las principales plataformas de software RFID como Microsoft BizTalk RFID, IBM WebSphere, Oat Systems u Oracle, así como SAP a través de middleware. Ofrece un kit de desarrollo bien documentado con librerías .NET y JAVA con interfaces personalizadas para controlar el lector.

En este caso se ha empleado un lector Alien ALR-9800 basado en la plataforma RFID para empresas de Alien Technology, diseñado para etiquetas conformes de clase 1 de segunda generación. Este lector establece nuevos estándares para los lectores RFID a través de su:

- Arquitectura empresarial escalable
- Mayor rendimiento de capacidad de lectura
- Una densidad superior de lectura

Dando como resultado un lector de nueva generación para la implementación eficiente de RFIS en una escala pequeña o grande con un nivel de rendimiento alto, un coste inicial y de mantenimiento bajos y acelerando el retorno de la inversión.

El ALR-9800 soporta todos los protocolos actuales con prácticamente ninguna latencia en el cambio de protocolo. Combina una alta sensibilidad con un bajo coste a través de su diseño avanzado de antenas multiestáticas, por lo que cada antena transmite y recibe al mismo tiempo utilizando la mitad del número de antenas requeridas que la mayoría de otros lectores.

Además es compatible con los modos individual, denso y múltiples lectores. El modo denso permite hacer un uso eficiente del espectro radioeléctrico disponible cuando un gran número de lectores operan en el mismo espacio. Cuenta con una capacidad de “escuchar” antes de “hablar” de banda ancha para evitar interferencias con lectores próximos y aprovecha su propósito general de capacidad de E/S para reducir el tiempo de encendido de cada lector y permitir así un mayor número de lectores coexistiendo en la misma instalación.

2.5 MySQL

MySQL es un gestor de base de datos sencillo de usar y increíblemente rápido. También es uno de los motores de base de datos más usados en Internet, la principal razón de esto es que es gratis para aplicaciones no comerciales.

Las características principales de MySQL son:

- Es un gestor de base de datos. Una base de datos es un conjunto de datos y un gestor de base de datos es una aplicación capaz de manejar este conjunto de datos de manera eficiente y cómoda.

- Es una base de datos relacional. Una base de datos relacional es un conjunto de datos que están almacenados en tablas entre las cuales se establecen unas relaciones para manejar los datos de una forma eficiente y segura. Para usar y gestionar una base de datos relacional se usa el lenguaje estándar de programación SQL.

- Es Open Source. El código fuente de MySQL se puede descargar y está accesible a cualquiera, por otra parte, usa la licencia GPL para aplicaciones no comerciales.

- Es una base de datos muy rápida, segura y fácil de usar. Gracias a la colaboración de muchos usuarios, la base de datos se ha ido mejorando optimizándose en velocidad. Por eso es una de las bases de datos más usadas en Internet.

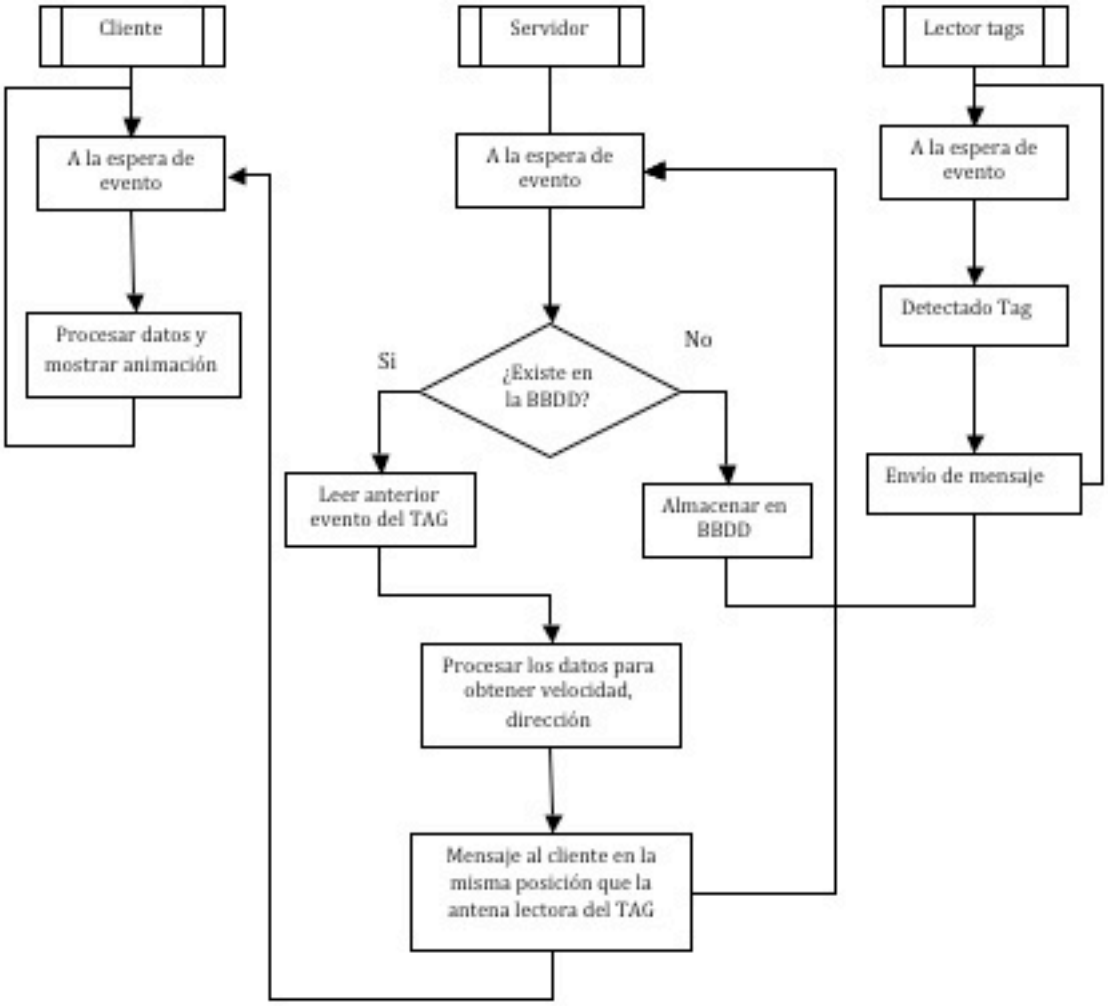
Capítulo 3: Arquitectura e implementación

La arquitectura del sistema para las pruebas del proyecto consiste en un PC actuando como servidor, un PC actuando como cliente, un lector de etiquetas RFID de Alien Technology modelo ALR 9800 al que están conectadas cuatro antenas.

El PC que actúa como servidor está conectado por red Ethernet al PC que actúa como cliente y por el puerto serie R232 al lector de etiquetas RFID. Se ejecuta una aplicación que está a la espera de recibir mensajes del lector RFID, tratar los datos y enviar un mensaje al cliente con los datos necesarios.

El PC que actúa como cliente ejecuta una aplicación Flash que se mantiene a la espera de recibir un mensaje del servidor con los datos necesarios para reproducir una animación.

El lector de etiquetas RFID está conectado al PC servidor y a 4 antenas. Las antenas están configuradas por parejas como emisora y receptora, y colocadas en las mismas posiciones que los PC clientes.


Cliente
A la espera de evento
Procesar datos y mostrar animación
Servidor
A la espera de evento
¿Existe en la BBDD?
Si
No
Leer anterior evento del TAG
Almacenar en BBDD
Procesar los datos para obtener velocidad, dirección
Mensaje al cliente en la misma posición que la antena lectora del TAG
Lector tags
A la espera de evento
Detectado Tag
Envio de mensaje

Servidor

La máquina utilizada como servidor es un ordenador personal de sobremesa conectada por Ethernet a las máquinas clientes y a su vez conectada con el lector de etiquetas RFID.

La aplicación que se ejecuta y que actúa como servidor consta de seis partes y se hace uso de las librerías suministradas por Alien Technology para el manejo de su lector RFID.

- Configuración del lector RFID.
- Conexión con el lector RFID.
- Conexión con el servidor SQL y la base de datos.
- Comunicación con el lector RFID.
- Cálculo de datos.
- Comunicación con las máquinas clientes y la transmisión de los datos calculados.

La aplicación se conecta con el lector de RFID pasándole a éste los parámetros para configurar el lector, como el puerto serie al que está conectado, la velocidad de transmisión, la secuencia de lecturas de las antenas, la forma de notificación de la lectura de las etiquetas y su formato.

Más tarde la aplicación lee el fichero donde se encuentran las direcciones de las máquinas clientes que mostrarán la animación del seguimiento de la publicidad. Crea los sockets UDP de todos los clientes almacenados en el fichero leído.

La aplicación conecta con el servidor SQL indicando la tabla de la base de datos que usará durante su ejecución. Ésta tabla contiene la información de las etiquetas que portan los usuarios asociados al identificador de cada etiqueta RFID.

Identificador de etiqueta	Nombre del sujeto	Texto asociado	Antena lectora	Dirección del sujeto	Velocidad del sujeto	Fecha y hora de la lectura

Los campos de la tabla, arriba indicados, en detalle son:

- Identificador de etiqueta: dado por el fabricante, es un número de serie asociado a cada etiqueta RFID.
- Nombre: el nombre de la persona portadora de la etiqueta RFID, este dato es previamente cargado en la tabla antes de la ejecución de la aplicación.
- Texto: texto que se muestra por pantalla de la máquina cliente, asociado, también, al identificador de la etiqueta es un dato cargado previamente en la tabla antes de la ejecución de la aplicación.
- Antena: número de la última antena que leyó la etiqueta, este dato lo proporciona el lector de RFID y se modifica cuando una etiqueta es leída.
- Dirección: la dirección hacia la que se mueve el sujeto con la etiqueta al que está asociado, los valores que puede tomar son positivo o negativo dependiendo de la secuencia de antenas que han leído la etiqueta, si es creciente (positiva), decreciente (negativa) o está detenido delante de uno de los clientes. Este dato depende de la forma que se ha montado el escenario de pruebas. Este dato es modificado cuando una etiqueta ha sido leída como mínimo dos veces para determinar si va en dirección positiva, negativa o parado.
- Velocidad: la velocidad estimada a la que ha pasado la etiqueta entre dos antenas lectoras, por defecto la velocidad inicial es de 1'03 metros por segundo que es la velocidad media de un humano andando.
- Tiempo: fecha y hora de la última lectura de esa etiqueta RFID por el lector, necesaria para calcular la velocidad del sujeto y el tiempo de vida de los datos.

Intenta conecta con el lector de etiquetas RFID y se mantiene a la espera de recibir mensajes del lector. Para ello utiliza varias funciones del programa, configura el lector de RFID, conecta con el lector RFID y después se queda a la espera de recibir mensajes que manejará por medio de eventos. En caso de fallar la conexión se procede al reintento de conectar controlado por un temporizador.

Cuando el lector RFID manda un mensaje, se trata el evento, el mensaje llega como una cadena de texto en donde se indica el identificador único de la etiqueta RFID, la antena lectora que ha detectado la etiqueta RFID, la hora en la que se detectó por primera vez la etiqueta y la hora que se detectó la última vez, además de lo almacenado en su memoria que en nuestro sistema carece de importancia. Los datos se comparan con los datos almacenados en la base de datos, los identificadores de las etiquetas deben estar en la base de datos previamente, y se comparan con los datos almacenados previamente para ver los anteriores eventos de esas etiquetas, si es la primera vez que se ha detectado esa etiqueta se almacenará en la base de datos la antena en la que ha sido leída y la velocidad y dirección estimadas. Si la antena que detecta la etiqueta RFID se encuentra en unos de los extremos y es la primera vez que se detecta esa etiqueta o se detecta en un tiempo considerablemente largo, se presupone que irá en dirección a la antena del otro extremo y a una velocidad estimada de 1’03 metros por segundo, que es la velocidad media de un ser humano andando.

Cuando ya ha sido detectada anteriormente la etiqueta RFID en un corto espacio de tiempo, se compara la antena que actualmente ha detectado la etiqueta con la antena que la detectó previamente y que está guardada en la base de datos. Como previamente la distancia entre las antenas o los PC clientes está fijada, sabemos la ordenación de las antenas y tenemos almacenada en la base de datos la hora en que la etiqueta fue detectada la última vez, junto con los datos del nuevo mensaje podemos calcular con un margen de error relativamente pequeño la velocidad a la que se mueve el portador de la etiqueta RFID y conoceremos su dirección.

La aplicación está preparada para detectar los cambios de velocidad y dirección del portador de la etiqueta, tanto si su velocidad es cero, es decir que se detiene delante de una de las antenas, como si sigue andando en sentido contrario. Aunque en nuestro sistema, al sólo disponer de cuatro antenas sólo se puede disponer de 2 clientes.

Al llegar al final del recorrido, es decir, al ser detectada la etiqueta en la última antena del recorrido después de haber pasado por la anterior, el sistema se actualiza reiniciando los datos de esa etiqueta RFID para que pueda volver a pasar en cualquier sentido en un futuro.

Después de hacer todas las operaciones para calcular la velocidad y dirección del portador de la etiqueta RFID y de haber actualizado la base de datos, la aplicación serializa los datos obtenidos tanto por el lector como de la tabla de la base de datos y manda un mensaje UDP a la dirección IP asociada al cliente que se encuentra en la misma posición que la antena que ha detectado la etiqueta RFID.

Hecho esto el servidor se queda a la espera de recibir un nuevo evento de lectura de etiqueta RFID y vuelve a repetir la operación.

Cliente.

La máquina utilizada como cliente es un ordenador personal de sobremesa conectada en red mediante Ethernet a la máquina servidor.

La función de la máquina cliente es la de mostrar una animación que debe moverse a la misma velocidad y en la misma dirección que el sujeto portador de la etiqueta RFID, para ello se ejecuta una aplicación flash que se conecta mediante socket al servidor.

Flash sólo tiene soporte para sockets TCP por lo que para conectarse con el servidor mediante un socket TCP es necesario el uso de una pasarela que convierta los datagramas UDP en un paquete de datos TCP. Para ello he optado por usar la aplicación "udp-tcp-bridge" de libre distribución.

Una vez la aplicación flash recibe los datos por el socket TCP se ejecuta la animación. Al no poder usar un motor físico con flash, he debido estimar la velocidad que llega en metros por segundo a una velocidad medida en píxeles por segundo por lo que la velocidad dependerá del tamaño de la pantalla y la resolución de ésta, y no será una velocidad real.

Los datos mostrados en la animación, además de los datos visuales como dirección y velocidad estimada, son el nombre y el texto asociado a la etiqueta. Aunque está preparado para poder reproducir cualquier animación que se desee, como la de anuncios informativos para lo que fue diseñado el sistema.

Lector RFID.

```
mReader.SerialPort = "COM1:";
mReader.HostBaudRate = 115200;
mReaderInfo.InterfaceType = ComInterface.enumSerial;
mReader.ReaderSettings = mReaderInfo;

try                 // extra precaution though it shouldn't throw exceptions
{
   mReader.InitOnCom(1, 115200);

   String result = mReader.ConnectEasy();
   Console.WriteLine("Resultado");
   Console.WriteLine(result);

   if (mReader.IsConnected)
   {
      // to make it faster and not to lose any tag
      Console.WriteLine("Conectado");
      mTimer.Stop();
```

```
mReader.AutoMode = "On";

mReader.AutoAction = "Acquire";

mReader.NotifyMode = "On";

mReader.NotifyHeader = "On";

mReader.NotifyTrigger = "ADD";

mReader.NotifyFormat = "Text";

mReader.NotifyAddress = "serial";

mReader.PersistTime = "10";

mReader.AntennaSequence = "0,1,2,3";
```

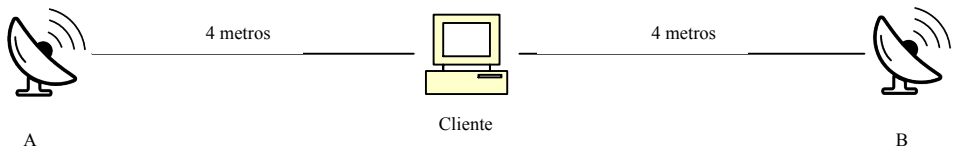
Capítulo 4: Resultados experimentales

Para la realización de los experimentos se tomaron en cuenta tres escenarios posibles.

Es posible variar la disposición de las antenas y su número para poder conseguir una mayor cobertura, mayor precisión o conseguir la localización en un entorno de dos dimensiones.

En este experimento, se quiso ubicar el tag RFID en un entorno de una sola dimensión y con una extensión reducida. Para ello se ubicaron las antenas por parejas a una distancia de unos 4 metros con la máquina cliente, encargada de la visualización de los resultados, ubicada en el centro.

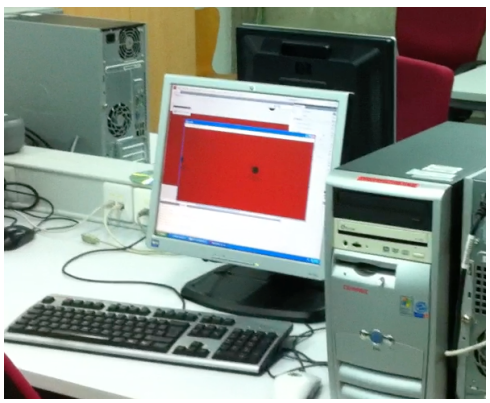
La velocidad de movimiento de las imágenes que representan las etiquetas RFID se tradujo de metros/segundo a píxeles/segundo, por lo que no es exacta y varía la parametrización del sistema dependiendo de la pantalla donde se visualiza la animación.



Escenario uno: pasar de un lado a otro

Para este escenario se trasladó un grupo de etiquetas RFID desde la posición de la antena A a la posición de la antena B sin detenerse.

En este escenario se comprobó que las imágenes se trasladaban en el monitor del cliente a una velocidad estimada similar a la del traslado de las etiquetas y en la misma dirección.



Escenario dos: detenerse enfrente de uno de los clientes

Para este escenario se trasladó un grupo de etiquetas RFID desde la posición de la antena A hasta la posición en la antena B deteniéndose en un punto intermedio del trayecto. Se comprobó que la animación mostraba las imágenes que representaban las etiquetas RFID deteniéndose en la pantalla antes de llegar al margen de la pantalla.

Escenario tres: darse la vuelta antes de llegar al otro cliente

Para este escenario se trasladó un grupo de etiquetas RFID desde la posición de la antena A hasta la posición de la antena B y cambiando de sentido antes de llegar a la posición de la antena B. Se comprobó que la animación mostraba un traslado de las imágenes que representan las etiquetas RFID se detenían en mitad de la animación y volvían al punto de partida.

Capítulo 5: Conclusiones y trabajos futuros

RFID se ha extendido en los últimos años a distintos ámbitos industriales y comerciales. Básicamente, la tecnología de RFID permite la identificación de elementos (tags) a una distancia de varios metros del dispositivo lector sin la necesidad de visión directa como es el caso de código de barras. Este proyecto explora la posibilidad de emplear RFID en un entorno publicitario. Partimos de la premisa de que los usuarios podrán ser geolocalizados a través de RFID y que podremos medir tiempo de permanencia en cada localización. Mediante esta información se pretende poder ofrecer publicidad y ofertas personalizadas a cada usuario. El proyecto ha consistido fundamentalmente en el desarrollo de esta plataforma y su verificación en pruebas esenciales de operatividad.

Capítulo 6: Bibliografía

- Alien Technology http://www.alientechnology.com/
- technovelgy http://www.technovelgy.com/
- Wikipedia http://en.wikipedia.org
- ISO http://www.iso.org/
- Microsoft
- Adobe

Código Servidor

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Timers;
using System.IO;
using System.Data.Common;

//Socket
using System.Net;
using System.Net.Sockets;

//RFID
using nsAlienRFID;

//SQLite
using System.Data.SQLite;

namespace Proyecto
{
   public partial class Form1 : Form
   {
      delegate void SetTextCallback(string text);
      private clsReader mReader;
      private ReaderInfo mReaderInfo;
      private NotifyInfo mNotifyInfo;
      private System.Timers.Timer mTimer;
      private byte[] buffer = new byte[255];
      private Socket s;
      private int puerto = 17777;
      private IPEndPoint ep,ep2;
```

```
//array de ip's, en este caso maximo de antenas 3
private string[] ip;
private string ruta = "c:\\Nubetags\\ip.txt";
private int indice = 0;
private String cadena, direccion;
private double velocidad; // m/s
private int max_antena = 4; //en este ejemplo el numero maximo de antenas es 4 (0-3)
//base de datos
private String db_antena, db_nombre, db_direccion, db_texto, db_tiempo;
private double db_velocidad;

public Form1()
{
   InitializeComponent();
}

private void conectar()
{
   Console.WriteLine("Conectando");
   mTimer.Stop();

   mReader = new clsReader();
   mReaderInfo = mReader.ReaderSettings;

   //mReader.ComTimeOutInterval = 4;

   mReader.SerialPort = "COM1:";
   mReader.HostBaudRate = 115200;
   mReaderInfo.InterfaceType = ComInterface.enumSerial;
   mReader.ReaderSettings = mReaderInfo;

   try      // extra precaution though it shouldn't throw exceptions
   {
      mReader.InitOnCom(1, 115200);

      String result = mReader.ConnectEasy();
      Console.WriteLine("Resultado");
      Console.WriteLine(result);
```

```
            if (mReader.IsConnected)
            {
               // to make it faster and not to lose any tag
               Console.WriteLine("Conectado");
               mTimer.Stop();

               mReader.AutoMode = "On";
               mReader.AutoAction = "Acquire";
               mReader.NotifyMode = "On";
               mReader.NotifyHeader = "On";
               mReader.NotifyTrigger = "ADD";
               mReader.NotifyFormat = "Text";
               mReader.NotifyAddress = "serial";
               mReader.PersistTime = "10";

               mReader.MessageReceived += new nsAlienRFID.clsReader.MessageReceivedEventHandler(
mReader_MessageReceived);
            }
            else
            {
               mTimer.Start();
            }
         }
         catch (Exception ex)
         {
            Console.WriteLine("Exception in btnConnect_Click(): " + ex.Message);
         }
         Console.WriteLine("Saliendo de conectar()");
      }

      private void Form1_Load(object sender, System.EventArgs e)
      {
         //Leer de un fichero las ips y ponerlo en un array
         ip = new string[max_antena];
         using(StreamReader sr = File.OpenText(ruta))
         {
            while ((cadena = sr.ReadLine()) != null)
            {
               ip[indice] = cadena;
```

```
            Console.WriteLine(ip[indice]);
            indice = indice + 1;
          }
        }
        //SOCKET
        try
        {
          ep = new IPEndPoint(IPAddress.Any,puerto);
          s = new Socket(AddressFamily.InterNetwork, SocketType.Dgram, ProtocolType.Udp);
          s.Bind(ep);
        }
        catch(Exception exc)
        {
          Console.WriteLine("Winsock error: " + exc.ToString());
        }
        //BBDD
        using (SQLiteConnection conn = new SQLiteConnection("Data Source=C:\\Nubetags\\tags.db3"
))
        {
          conn.Open();
          using ( SQLiteCommand comm = new SQLiteCommand() )
          {
            // Crear base de datos de usuarios
            comm.Connection = conn;
            comm.CommandText = "CREATE TABLE IF NOT EXISTS tags(tag TEXT UNIQUE, nombre TE
XT, texto TEXT, antena INTEGER, direccion TEXT, velocidad REAL, tiempo TEXT)";
            try
            {
              comm.ExecuteNonQuery();
            }
            catch ( SQLiteException exception )
            {
              Console.WriteLine ( "Failed : " + exception.Message );
            }

            // Recrear datos de partida de usuarios
            comm.CommandText = "DELETE FROM tags;INSERT INTO tags (tag, nombre, texto, antena,
 direccion, velocidad, tiempo) values ('E200 3411 B802 0117 6714 1375','Javier','Javier texto','-
1','0','0','-
```

```
1');INSERT INTO tags (tag, nombre, texto, antena, direccion, velocidad, tiempo) values ('E200 3411 B8
02 0117 6714 1378','María','María texto','-1','0','0','-
1');INSERT INTO tags (tag, nombre, texto, antena, direccion, velocidad, tiempo) values ('E200 3411 B8
02 0117 6714 1383','Bruce','Bruce texto','-1','0','0','-
1');INSERT INTO tags (tag, nombre, texto, antena, direccion, velocidad, tiempo) values ('E200 3411 B8
02 0117 6714 1376','Kevin','Kevin texto','-1','0','0','-
1');INSERT INTO tags (tag, nombre, texto, antena, direccion, velocidad, tiempo) values ('E200 3411 B8
02 0117 6714 1377','Jochen','Jochen texto','-1','0','0','-
1');INSERT INTO tags (tag, nombre, texto, antena, direccion, velocidad, tiempo) values ('E200 3411 B8
02 0117 6714 1305','Javier G','Javier G texto','-1','0','0','-
1');INSERT INTO tags (tag, nombre, texto, antena, direccion, velocidad, tiempo) values ('E200 3411 B8
02 0117 6714 1379','Raul','Raul texto','-1','0','0','-
1');INSERT INTO tags (tag, nombre, texto, antena, direccion, velocidad, tiempo) values ('E200 3411 B8
02 0117 6714 1380','Björn','Björn texto','-1','0','0','-
1');INSERT INTO tags (tag, nombre, texto, antena, direccion, velocidad, tiempo) values ('E200 3411 B8
02 0117 6714 1381','Chia-Chen','Chia-Chen texto','-1','0','0','-
1');INSERT INTO tags (tag, nombre, texto, antena, direccion, velocidad, tiempo) values ('E200 3411 B8
02 0117 6714 1382','Luca','Luca texto','-1','0','0','-
1');INSERT INTO tags (tag, nombre, texto, antena, direccion, velocidad, tiempo) values ('E200 3411 B8
02 0117 6714 1306','Rafael','Rafael texto','-1','0','0','-
1');INSERT INTO tags (tag, nombre, texto, antena, direccion, velocidad, tiempo) values ('E200 3411 B8
02 0117 6714 1374','Iker','Iker texto','-1','0','0','-1');";

            try
            {
               comm.ExecuteNonQuery();
            }
            catch (SQLiteException exception)
            {
               Console.WriteLine("Failed : " + exception.Message);
            }
         }
         conn.Close();
      }

      Console.WriteLine("Entrando en form1_Load");
      mTimer = new System.Timers.Timer((double)1000);
      mTimer.Elapsed += new ElapsedEventHandler(IntentoConexion);
      mTimer.AutoReset = true;
```

```
        mTimer.Start();

        this.KeyPreview = true;

        this.Focus();

        this.textBox1.Text = "Iniciando...\r\n";
    }

    private void IntentoConexion(object source, ElapsedEventArgs e)
    {
        conectar();
    }

    private void mReader_MessageReceived(string data)
    {
        int iTags;
        cadena = "";

        iTags = AlienUtils.ParseNotification(data, out mNotifyInfo);

        using (SQLiteConnection conn = new SQLiteConnection("Data Source=C:\\Nubetags\\tags.db3"
))
        {
            conn.Open();
            using (SQLiteCommand comm = new SQLiteCommand())
            {
                comm.Connection = conn;
                try
                {
                    using (SQLiteTransaction tran = conn.BeginTransaction())
                    {

                        if( true )
                        {
                            if (mNotifyInfo.TagList != null)
                            {

                                foreach (var value in mNotifyInfo.TagList)
```

```
                {
                    comm.CommandText = "SELECT * FROM tags WHERE tag = @myTestIndex LIMI
T 1";
                    DbParameter param = comm.CreateParameter();
                    param.Value = value.TagID;
                    param.ParameterName = "myTestIndex";
                    param.DbType = DbType.String;
                    comm.Parameters.Add(param);

                    using (SQLiteDataReader dr = comm.ExecuteReader())
                    {
                        while (dr.Read())
                        {
                            db_antena = dr.GetValue(3).ToString();
                            db_direccion = dr.GetValue(4).ToString();
                            db_velocidad = Double.Parse(dr.GetValue(5).ToString());
                            db_tiempo = dr.GetValue(6).ToString();
                            db_nombre = dr.GetValue(1).ToString();
                            db_texto = dr.GetValue(2).ToString();
                        }
                    }
                    velocidad = Calc_Velocidad(db_tiempo, value.LastSeenTime);
                    if (Double.Parse(db_antena) > value.Antenna || (Double.Parse(db_antena) == -
1 && value.Antenna == max_antena))
                    {
                        direccion = "negativa";
                        if (value.Antenna == 0)
                        {
                            //el tag ha terminado el recorrido, hay que reiniciarlo
                            comm.CommandText = "UPDATE tags SET antena='-
1', direccion='0', velocidad='0', tiempo='-1' WHERE tag='" + value.TagID + "'";

                        }
                        else
                        {
                            //el tag sigue en recorrido hacia antenas de numeracion mas baja
                            comm.CommandText = "UPDATE tags SET antena='" + value.Antenna.ToStri
ng() + "', direccion='" + direccion + "', velocidad='" + velocidad + "', tiempo='" + value.LastSeenTime + "
' WHERE tag='" + value.TagID + "'";
```

```
                    }
                }
                else if ((Double.Parse(db_antena) < value.Antenna && value.Antenna != max_ant
ena))
                {
                    direccion = "positiva";
                    if (value.Antenna == max_antena)
                    {
                        //el tag ha terminado el recorrido, hay que reiniciarlo
                        comm.CommandText = "UPDATE tags SET antena='-
1', direccion='0', velocidad='0', tiempo='-1' WHERE tag='" + value.TagID + "'";
                    }
                    else
                    {
                        //el tag sigue en recorrido hacia antenas de numeracion mas alta
                        comm.CommandText = "UPDATE tags SET antena='" + value.Antenna.ToStri
ng() + "', direccion='" + direccion + "', velocidad='" + velocidad + "', tiempo='" + value.LastSeenTime + "
' WHERE tag='" + value.TagID + "'";
                    }

                }
                else if (Double.Parse(db_antena) == value.Antenna)
                {

                }
                try
                {
                    comm.ExecuteNonQuery();
                }
                catch (SQLiteException exception)
                {
                    Console.WriteLine("Failed : " + exception.Message);
                }


                cadena = "&direccion=" + direccion + "&velocidad=" + velocidad.ToString() + "&n
ombre=" + db_nombre + "&texto=" + db_texto;
                buffer = Encoding.ASCII.GetBytes(cadena);
                //sendto(buffer)
```

```
                    try
                    {
                        ep2 = new IPEndPoint(IPAddress.Parse(ip[value.Antenna].ToString()), puerto);

                        s.SendTo(buffer, ep2);
                    }
                    catch (Exception exc)
                    {
                        Console.WriteLine(exc);
                    }
                }
            }
        }

            tran.Commit();
        }
    }
    catch (SQLiteException exception)
    {
        Console.WriteLine("Failed : " + exception.Message);
    }
  }
  conn.Close();
 }
}

private void SetText(string text)
{
  if (this.textBox1.InvokeRequired)
  {
    SetTextCallback d = new SetTextCallback(SetText);
    this.Invoke(d, new object[] { text });
  }
  else
  {
    this.textBox1.Text += text;
  }
}
```

```
    private void Form1_FormClosing(object sender, FormClosingEventArgs e)
    {
      if (mReader != null)
      {
        mReader.Dispose();
      }
    }

    private double Calc_Velocidad(String tiempo_db, String tiempo)
    {
      double t1,t2,resultado;
      string aux;
      string[] aux2;
      double espacio = 3; // por ejemplo
      if (String.Compare(tiempo_db,"-1")==0)
      {
        resultado = 1.028;
      }
      else
      {
        aux = tiempo_db.Remove(0, 11);
        aux2 = aux.Split(':');
        t1 = double.Parse(aux2[0]) * 3600 + double.Parse(aux2[1]) * 60 + double.Parse(aux2[2]);
        aux = tiempo.Remove(0, 11);
        aux2 = aux.Split(':');
        t2 = double.Parse(aux2[0]) * 3600 + double.Parse(aux2[1]) * 60 + double.Parse(aux2[2]);
        resultado = (t2 - t1)/ espacio;
      }
      return resultado;
    }

    private void button1_Click(object sender, EventArgs e)
    }
  }
}
```