ESCUELA POLITÉCNICA DEL EJÉRCITO

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

PROYECTO DE GRADO PARA LA OBTENCIÓN DEL TÍTULO EN INGENIERÍA

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA BASADO EN LA TECNOLOGÍA RFID PARA EL CONTROL DE INVENTARIO DE LA EMPRESA MILBOOTS

JAVIER MARCELO LARA GALARZA

SANGOLQUÍ – ECUADOR

2008

CERTIFICACIÓN

Certificamos que el presente proyecto de grado titulado "DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA BASADO EN LA TECNOLOGÍA RFID PARA EL CONTROL DE INVENTARIO DE LA EMPRESA MILBOOTS", fue realizado en su totalidad por el Sr. Javier Marcelo Lara Galarza, como requerimiento previo para la obtención del titulo de electrónica en automatización y control.

Ing. Víctor Proaño	Ing. Jaime Andrango
DIRECTOR	CODIRECTOR

AGRADECIMIENTO

Debo agradecer primeramente a Dios ya que de alguna forma me ayudó a adquirir los dispositivos para la realización de este proyecto a precios que estaban a mi alcance.

A mis padres que con su apoyo y comprensión me supieron ayudar a culminar con la tesis.

A mi director de tesis Ing. Víctor Proaño e Ing. David Andrade que me proporcionaron su tiempo y conocimientos.

A la compañía Alien technology que me proporcionaron la información necesaria para la realización del proyecto.

DEDICATORIA

Con mi más profundo agradecimiento a mi Dios y mis padres.

PRÓLOGO

El proyecto de grado "DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA BASADO EN LA TECNOLOGÍA RFID PARA EL CONTROL DE INVENTARIO DE LA EMPRESA MILBOOTS" tiene como objeto controlar el inventario de materia prima y producto terminado de la fábrica de botas de caucho MILBOOTS.

En la actualidad debido a la gran cantidad de materiales que una industria maneja para la fabricación, se hace necesario tener sistemas de control mas rápidos, eficientes y exactos para conocer qué posee la empresa, por eso se han desarrollado varios sistemas como es el código de barras.

La aparición de la tecnología RFID para el control de inventarios surge debido a la gran limitante que tienen los códigos de barras al no poder leer varios códigos al mismo tiempo, es decir necesariamente un operador tiene que manipular un lector para leer todos los códigos de barras, en cambio las etiquetas RFID pueden ser leídas por un lector, el cual simplemente tiene que estar ubicado estratégicamente para leer todos las etiquetas a su alcance, el cual es aproximadamente 3mts de alcance.

Por tanto en la actualidad empresas que están en la vanguardia de la tecnología como la gran cadena minorista WALMART han decidido reemplazar el código de barras por etiquetas inteligentes o más conocidas como etiquetas RFID.

Con el presente proyecto se pretende reemplazar el actual sistema de control de inventarios, el cual es manual, por el sistema con RFID, ahorrando a la empresa mucho dinero y tiempo en contabilizar manualmente la materia prima y producto terminado que la empresa dispone y evitando con esto muchos errores producto de la naturaleza humana imperfecta.

ÍNDICE

CAPÍTULO 1

MARCO TEÓRICO

1.1. Introducción

RFID (siglas de R*adio* F*requency* ID*entification*, en español **Identificación por radiofrecuencia**) es un sistema de almacenamiento y recuperación de datos remoto que usa dispositivos denominados **etiquetas**, **transpondedores (transmisor-receptor)** o **tags RFID**. El propósito fundamental de la tecnología RFID es transmitir la identidad de un objeto (similar a un número de serie único) mediante ondas de radio.

Una etiqueta RFID es un dispositivo pequeño, similar a una pegatina, que puede ser adherida o incorporada a un producto, animal o persona. Contiene antenas para permitirle recibir y responder a peticiones por radiofrecuencia desde un emisor-receptor RFID. Una de las ventajas del uso de radiofrecuencia (en lugar, por ejemplo, de código de barras) es que no se requiere visión directa entre emisor y receptor.

1.2. Historia

Las primeras patentes para dispositivos RFID fueron solicitadas en Estados Unidos, concretamente en Enero de 1973 cuando Mario W. Cardullo se presentó con una etiqueta RFID activa que portaba una memoria rescribible. El mismo año, Charles Walton recibió la patente para un sistema RFID pasivo que abría las puertas sin necesidad de llaves. Una tarjeta con un transponedor comunicaba una señal al lector de la puerta que cuando validaba la tarjeta desbloqueaba la cerradura.

El gobierno americano también trabajaba sobre esta tecnología en los años 70 y montó sistemas parecidos para el manejo de puertas en las centrales nucleares, cuyas puertas se abrían al paso de los camiones que portaban materiales para las mismas que iban equipados con un transpondedor. También se desarrolló un sistema para el control del ganado que había sido vacunado insertando bajo la piel de los animales una etiqueta RFID pasiva con la que se identificaba los animales que habían sido vacunados y los que no. Después han ido llegando mejoras en la capacidad de emisión y recepción, así como en la distancia, lo cual ha llevado a extender su uso en ámbitos tanto domésticos como de seguridad nacional, como sucede con el pasaporte expedido en la actualidad en los EEUU que lleva asociadas etiquetas RFID.

1.3. Arquitectura

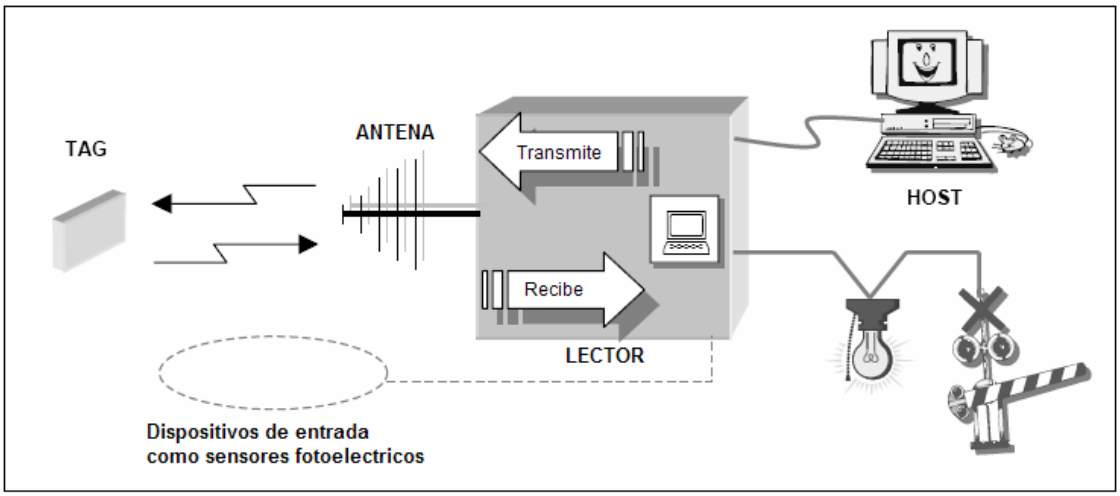


Fig. 1.1. Componentes de un sistema RFID

El modo de funcionamiento de los sistemas RFID es simple. La etiqueta RFID, que contiene los datos de identificación del objeto al que se encuentra adherido, genera una señal de radiofrecuencia con dichos datos. Esta señal puede ser captada por un lector RFID, el cual se encarga de leer la información y pasársela, en formato digital, a la aplicación específica que utiliza RFID.

Por tanto, un sistema RFID consta de los siguientes componentes:

- **Etiqueta RFID** o **transpondedor**: compuesta por una antena, un material encapsulado o chip y el PET, el cual es un material hecho de poliuretano transparente flexible el cual sostiene al chip y la antena. El propósito de la antena es permitirle al chip, el cual contiene la información, transmitir la información de identificación. Existen varios tipos de etiquetas. El chip posee una memoria interna con una capacidad que depende del modelo y varía de una decena a millares de bytes. Existen varios tipos de memoria:

Solo lectura: el código de identificación que contiene es único y es personalizado durante la fabricación de la etiqueta.

De lectura y escritura: la información de identificación puede ser modificada por el lector.

Anticolisión. Se trata de etiquetas especiales que permiten que un lector identifique varias al mismo tiempo (habitualmente las etiquetas deben entrar una a una en la zona de cobertura del lector).

- **Lector de RFID** o **transceptor**: compuesto por una antena, un transceptor (transmisor-receptor) y un decodificador. El lector envía periódicamente señales para ver si hay alguna etiqueta en sus inmediaciones. Cuando capta una señal de una etiqueta (la cual contiene la información de identificación de ésta), extrae la información y se la pasa al subsistema de procesamiento de datos.

1.4. Tipos de etiquetas RFID

Las etiquetas RFID pueden ser *activas*, *semipasivas* (también conocidas como *asistidas por batería*) o *pasivas*. Los tags pasivos no requieren ninguna fuente de alimentación interna (sólo se activan cuando un lector se encuentra cerca para suministrarles la energía necesaria). Los otros dos tipos necesitan alimentación, típicamente una pila pequeña.

El mercado con cada producto individualmente etiquetado con un Tag RFID en lugar de código de barras será comercialmente viable con volúmenes muy grandes de 10.000 millones de unidades al año, llevando el coste de producción a menos de 0,05$ cada Tag. La demanda actual de chips de circuitos integrados con RFID no está cerca de soportar ese

coste. Los analistas de las compañías independientes de investigación como Gartner and Forrester Research convienen en que un nivel de precio de menos de $0,05 (con un volumen de producción de 1.000 millones de unidades) sólo se puede lograr en unos 4 u 6 años, lo que limita los planes a corto plazo para una adopción extensa de las etiquetas RFID pasivas.

A pesar de las ventajas en cuanto al coste de las etiquetas pasivas con respecto a las activas son significativas, otros factores incluyendo exactitud, funcionamiento en ciertos ambientes como cerca del agua o metal, y confiabilidad hacen que el uso de etiquetas activas sea muy común hoy en día.

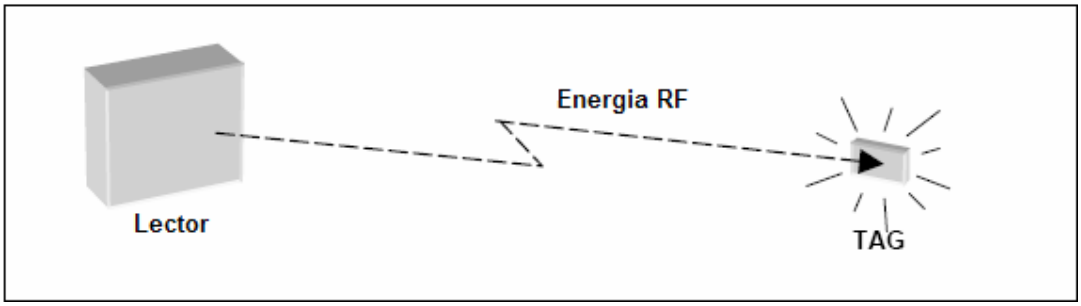


Fig. 1.2. Comunicación de un tag pasivo: el tag es energizado por la señal RF enviada por el lector

Tags pasivos. Los tags pasivos no poseen ningún tipo de alimentación. La señal que les llega de los lectores induce una corriente eléctrica mínima que basta para operar el circuito integrado CMOS del tag para generar y transmitir una respuesta. La mayoría de tags pasivos utiliza respuesta por reflejo o backscatter sobre la portadora recibida. Esto es, la antena ha de estar diseñada para obtener la energía necesaria y funcionar a la vez para transmitir la respuesta por reflejo.

Debido a las preocupaciones por la energía y el coste, la respuesta de una etiqueta pasiva RFID es necesariamente breve, normalmente apenas un número de identificación. La falta de una fuente de alimentación propia hace que el dispositivo pueda ser bastante pequeño: existen productos disponibles de forma comercial que pueden ser insertados bajo la piel. Las etiquetas pasivas, en la práctica tienen distancias de lectura que varían entre unos 10 centímetros (ISO 14443) hasta cerca de 6 metros (EPC e ISO 18000-6) dependiendo del tamaño de la antena, de la potencia y frecuencia en la que opera el lector.

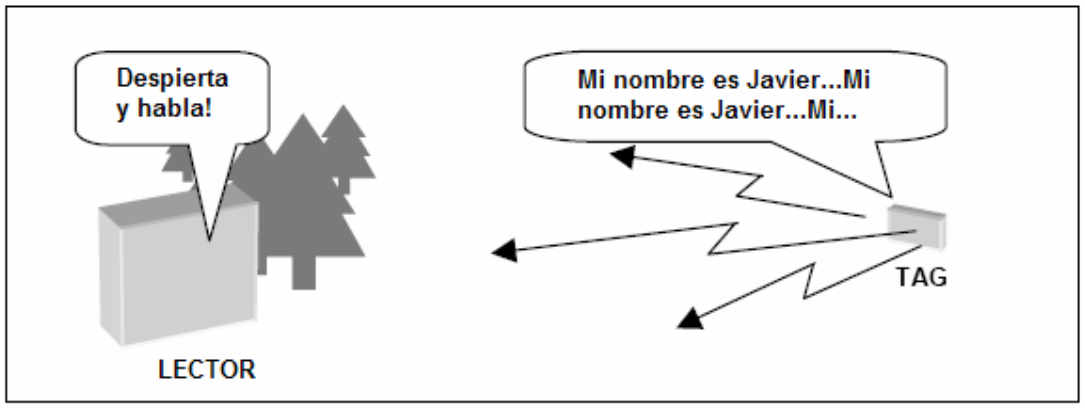


Fig. 1.3. Comunicación de un tag activo: el lector envía una señal al tag, éste responde usando energía de su propia batería

Tags activos. A diferencia de los tags pasivos, los activos poseen su propia fuente autónoma de energía, que utilizan para dar corriente a sus circuitos integrados y propagar su señal al lector. Gracias a su fuente de energía son capaces de transmitir señales más potentes que las de los tags pasivos, lo que les lleva a ser más eficientes en entornos dificultosos para la radiofrecuencia como en las inmediaciones de agua (incluyendo humanos y ganado, formados en su mayoría por agua), metal (contenedores, vehículos). También son efectivos a distancias mayores pudiendo generar respuestas claras a partir de recepciones débiles. Por el contrario, suelen ser más grandes, más caros, y su vida útil es en general mucho más corta debido a la batería.

Muchos tags activos tienen rangos efectivos de cientos de metros y una vida útil de sus baterías de hasta 10 años. Algunos de ellos integran sensores de registro de temperatura y otras variables que pueden usarse para monitorizar entornos de alimentación o productos farmacéuticos. Otros sensores asociados con RFID incluyen humedad, vibración, luz, radiación, temperatura, etc. Los tags, además de mucho más rango, tienen capacidades de almacenamiento mayores y la habilidad de guardar información adicional.

Tags semipasivos. Los tags semipasivos se parecen a los activos en que poseen una fuente de alimentación propia, aunque en este caso se utiliza principalmente para alimentar el microchip y no para transmitir la señal de respuesta. La energía contenida en la radiofrecuencia se refleja hacia el lector como en un tag pasivo. La batería puede permitir al circuito integrado de la etiqueta estar constantemente alimentado y eliminar la necesidad de diseñar una antena para recoger potencia de la señal entrante del lector. Las etiquetas

RFID semipasivas responden más rápidamente, por lo que son más fuertes en el ratio de lectura que las pasivas. Este tipo de tags tienen una fiabilidad comparable a la de los tags activos a la vez que pueden mantener el rango operativo de un tag pasivo. También sus baterías suelen durar más que los tags activos.

1.5. Clasificación

Los sistemas RFID se clasifican dependiendo del rango de frecuencias que usan. Existen cuatro tipos de sistemas:

Nombre	Frecuencia	Rango	Velocidad de lectura	Habilidad para leer en cercanías de agua o metal	Tamaño del Tag	Aplicaciones típicas
LF	125Khz	<0.5m	Lento	Mejor	Mas grande	Control de accesos
HF	13.56Mhz	<1m	Normal			Control de accesos, tarjetas inteligentes
UHF	860-930Mhz	<6m	Rápido			Control de rastreo de ítems, peajes electrónicos
Microonda	2.45-5.8GHz	<1m	Rápido	peor	Mas pequeño	Control de acceso para vehículos

Tabla 1.1. Clasificación de Tags por frecuencias.

1.6. Estandarización

Los estándares de RFID abordan cuatro áreas fundamentales:

- **Protocolo en el interfaz aéreo**: especifica el modo en el que etiquetas RFID y lectores se comunican mediante radiofrecuencia.
- **Contenido de los datos**: especifica el formato y semántica de los datos que se comunican entre etiquetas y lectores.

- **Certificación**: pruebas que los productos deben cumplir para garantizar que cumplen los estándares y pueden operar con otros dispositivos de distintos fabricantes.
- **Aplicaciones**: usos de los sistemas RFID.

Como en otras áreas tecnológicas, la estandarización en el campo de RFID se caracteriza por la existencia de varios grupos de especificaciones competidoras. Por una parte está ISO, y por otra Auto-ID Center (conocida desde octubre de 2003 como EPCglobal, *Electronic Product Code*). Ambas comparten el objetivo de conseguir etiquetas de bajo coste que operen en UHF; EPC es el más usado en América.

Los estándares EPC para etiquetas son de 4 clases:

- Clase 1: Tag simple, pasivo, con una memoria no volátil programable varias veces.
- Clase 2: Tag pasivo con más funcionalidades que clase 1, más memoria de usuario, control de acceso al Tag autentificado, y varias características adicionales.
- Clase 3: Tag semi-pasivo, fuente de energía integrada, circuito sensor integrado.
- Clase 4: Tag activo, comunicación Tag-a-Tag, comunicación activa.

Las clases no son ínteroperables y además son incompatibles con los estándares de ISO. Aunque EPCglobal está desarrollando una nueva generación de estándares EPC con el objetivo de conseguir interoperabilidad con los de ISO, aún está en discusión sobre el AFI (*Application Family Identifier*) de 8 bits.

El estándar **Clase 1 generación 2** de EPCglobal fue aprobado en diciembre de 2004, y es probable que llegue a formar la espina dorsal de los estándares en etiquetas RFID de ahora en adelante. EPC Gen2 es la abreviatura de "EPCglobal UHF Class 1 Generation 2".

Por su parte, ISO ha desarrollado estándares de RFID para la identificación automática y la gestión de objetos. Existen varios estándares relacionados, como ISO 10536, ISO 14443 e ISO 15693, pero la serie de estándares estrictamente relacionada con las RFID y las frecuencias empleadas en dichos sistemas es la serie 18000.

1.7. Regulación de frecuencias

No hay ninguna corporación pública global que gobierne las frecuencias usadas para RFID. En principio, cada país puede fijar sus propias reglas.
Las principales corporaciones que gobiernan la asignación de las frecuencias para RFID son:

- EE.UU.: FCC (*Federal Communications Commission*)
- Canadá: DOC (Departamento de la Comunicación)
- Europa: ERO, CEPT, ETSI y administraciones nacionales. Obsérvese que las administraciones nacionales tienen que ratificar el uso de una frecuencia específica antes de que pueda ser utilizada en ese país
- Japón: MPHPT (*Ministry of Public Management, Home Affairs, Post and Telecommunication*)
- China: Ministerio de la Industria de Información
- Australia: Autoridad Australiana de la Comunicación (*Australian Communication Authority*)
- Nueva Zelanda: Ministerio de desarrollo económico de Nueva Zelanda (*New Zealand Ministry of Economic Development.*
- Argentina: CNC (*Comisión Nacional de Comunicaciones*).
- Chile: Ministerio de Transportes y Telecomunicaciones.

Las etiquetas RFID de baja frecuencia (LF: 125 - 134 kHz y 140 - 148.5 kHz) y de alta frecuencia (HF: 13.56 MHz) se pueden utilizar de forma global sin necesidad de licencia. La frecuencia ultra alta (UHF: 868 - 928 MHz) no puede ser utilizada de forma global, ya que no hay un único estándar global. En Norteamérica, la frecuencia ultra alta se puede utilizar sin licencia para frecuencias UHF entre 908 - 928 MHz, pero hay restricciones en la energía de transmisión. En Europa esta frecuencia esta bajo consideración. Su uso sin licencia es sólo para el rango de 869.40 - 869.65 MHz, pero existen restricciones en la energía de transmisión. El estándar UHF norteamericano (908-928 MHz) no es aceptado en Francia ni Italia ya que interfiere con sus bandas militares. En China y Japón no hay regulación para el uso de la frecuencia UHF. Cada aplicación que use lectores de frecuencia UHF en estos países necesita de una licencia, que debe ser

solicitada a las autoridades locales. En Australia y Nueva Zelanda, el rango es de 918 - 926 MHz para uso sin licencia, pero hay restricciones en la energía de transmisión.

En el caso de Ecuador se rige a las normas de EPC (su representante en Ecuador es GS1, www.gs1ec.org) y el uso de la frecuencia es semejante a las normas FCC de Estados Unidos.

CAPÍTULO 2

HARDWARE

2.1. DESCRIPCIÓN DEL EQUIPO

2.1.1. Introducción

El lector RFID ALIEN 9800 está diseñado para leer y programar cualquier tag EPC de Clase 1 Generación 1 y Generación 2, y proporcionar reportes de eventos hacia una computadora, el método de comunicación puede ser mediante RS232 o a través de una red LAN con TCP/IP.

- El lector posee 1 puerto RS232 hembra (ver Fig. 2.2.)
- 1 puerto RJ45
- 1 conector con tornillos para entradas/salidas digitales
- 4 conectores tipo TNC polaridad reversa para las antenas

2.1.2. Requisitos

Para tener una interfase con el lector se requiere lo siguiente:

- Una PC con Windows 98 o mayor, y un puerto RS232 o conectividad ETHERNET
- Fuente de poder (incluido con el lector)
- 120 VAC 50-60Hz
- Software (librerías Alien para uso en Java o .Net)

2.1.3. Especificaciones

Las especificaciones técnicas del lector ALIEN 9800 se muestran en la tabla 2.1

Nombre	Lector RFID multipropósito Alien multi-port
Numero de modelo	ALR9800
Arquitectura	Lector Punto a multipunto, multistatic
Frecuencia de operación	902.75MHz - 927.25MHz
Canales de espera	50
Espaciado de canales	500KHz
Tiempo dwell de canales	<0.4 seconds
Transmisor RF	<30 dBm al final de un cable LMR-195
Método de modulación	Phase Reversal – Amplitude Shift Keying (PR-ASK)
Ancho de banda en 20db de modulación	< 400 KHz
Receptor RF	2 canales
Consumo de energía	45 Watts (120 VAC at 600 mA)
Interfaz de comunicación	RS-232 (DB-9 F), TCPI/IP (RJ-45)
Entradas/salidas	2 o 4 antenas coaxiales, 4 entradas/8salidas(colector abierto), puerto RS232, LAN, conector para la fuente
Dimensiones	(L) 9.0” (22.9 cm) x (A) 11” (28 cm) x (P) 2.22” (5.6 cm)
Peso	1.8 kg (4 lb)
Temperatura de operación	0°C to +50°C
Indicadores Led	Power, Link, Active, Ant0-3, CPU, Read, Sniff, Fault (red)
Software	Aplicaciones, códigos ejemplos, demo ejecutable (Alien Gateway)
Certificaciones	FCC Part 15; FCCID: P65ALR9800 IOC: 4370A-ALR9800

	9501101260000000155
Certificaciones de seguridad	cTUVus UL: 60950-1:3004 CAN/CSA: C22.2 No.60950-1-03

Tabla 2.1. Especificaciones del ALR9800

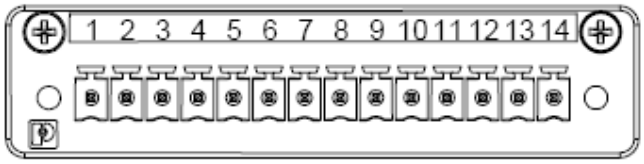


Figura 2.1 Conector con tornillos de las E/S digitales del lector

E/S conector (phoenix 14-pines)	
Pin 1	V+ (5-24VDC)
Pin 2	Salida 0 (opticalmente aislada)
Pin 3	Salida 1 “
Pin 4	Salida 2 “
Pin 5	Salida 3 “
Pin 6	Salida 4 “
Pin 7	Salida 5 “
Pin 8	Salida 6 “
Pin 9	Salida 7 “
Pin 10	Entrada 0 (opticalmente aislada, colector abierto, 10Vmax)
Pin 11	Entrada 1 “
Pin 12	Entrada 2 “
Pin 13	Entrada 3 “
Pin 14	V-

Tabla 2.2. Conector de las E/S digitales del lector

Para proteger las entradas con un voltaje mayor a 5V se debe colocar una resistencia de aproximadamente ¼ de vatio entre la fuente del sensor y la entrada, en la tabla 2.3 se muestra las resistencias para limitar la corriente según el voltaje de entrada.

Vin	R1
5 V	No
9 V	820 Ohms
12 V	1500 Ohms
24 V	3900 Ohms

Tabla 2.3 Valores de resistencias según voltaje de entrada

2.1.4. Panel

Panel de conexiones:

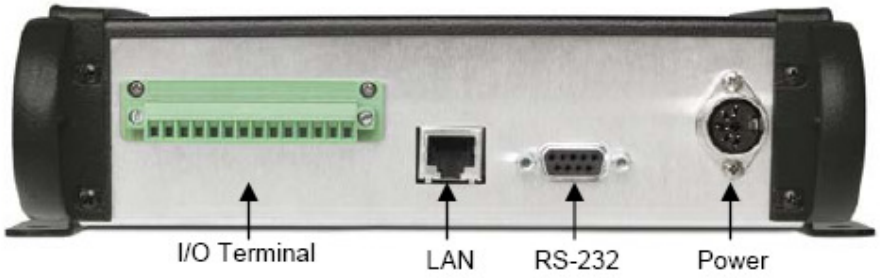


Fig 2.2 Panel de conexiones

Indicadores Led:

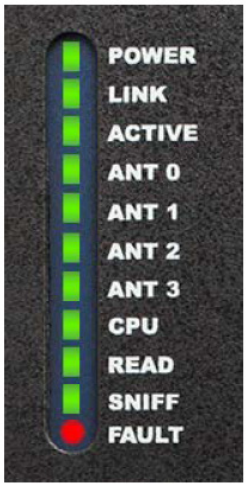


Fig 2.3 ALR 9800 Leds de diagnostico

POWER	Indica que el lector esta recibiendo energía
LINK	Indica que el lector está conectado a la red
ACTIVE	Indica que el lector está transmitiendo en la red
ANT 0-3	Indica que el lector está transmitiendo energía en la determinada antena
CPU	Indica que el CPU está corriendo correctamente
READ	Indica que el lector está recibiendo datos de un tag
SNIFF	Indica que un tag ha sido detectado, pero la energía puede no ser la suficiente para completar la transacción
FAULT	Indica un fallo

Tabla 2.4. Diodos indicadores (Leds) de diagnostico del lector ALR 9800 y su significado

2.1.5. Instalación para comunicación TCP/IP

2.1.5.1.Conectar cable Ethernet entre la PC o LAN y el lector

El lector puede estar configurado para mirar a un servidor DHCP (asignación dinámica de dirección IP) o usar las siguientes características:

- IP Adress: 192.168.1.100
- Mascara subnet: 255.255.255.0
- Gateway: 192.168.1.1

2.1.5.2.Conectar la fuente de poder al lector

2.1.5.3.Conectar los cables coaxiales a los puertos de las antenas (en pares)

Las antenas deben ser conectadas en pares, antena 0 y 1 o antenas 2 y 3, esto significa que mientras la antena 0 transmite, la antena 1 recibe o viceversa, de la misma manera funciona con las antenas 2 y 3.

2.1.5.4.Conectar el cable de la fuente de poder a la fuente de 120VAC

El diodo indicador de POWER se iluminará.

2.1.5.5.Verificar que la PC esté configurada con las mismas características de red que el lector

La PC debe tener la misma máscara subred (255.255.255.0) que el lector y tener una dirección IP estática (Ej. 192.168.1.101)

2.1.6. Sistema de operación

El lector ALR-9800 es controlado desde un software en una computadora host que se comunica con el lector usando el Alien Reader Protocol ASCII-based.

Se puede operar el lector desde cualquier aplicación, usando los ejemplos de código provistos por el fabricante en lenguajes de .Net y Java. En el capítulo 4 se describirá los comandos que se usan para controlar el lector.

2.2. DESCRIPCIÓN DE LAS ANTENAS

En el mercado existen varias antenas que se pueden usar para este lector, incluyendo las antenas que fabrica Alien (ALIEN-9610), pero debido al costo adicional que conlleva la compra de éstas se decidió fabricarlas.

2.2.1. Configuración de la antena

Otra variable que puede ayudar a leer Tags de difícil lectura es la elección de la antena de interrogación. Existen dos opciones básicas, cada una de la cuales tiene sus ventajas y desventajas en relación al desempeño.

- **Antena con polarización Lineal**: La polarización se refiere a la forma en que se transmite la RF. Polarización “lineal” quiere decir que el componente del campo eléctrico de una señal RF se propaga en un solo plano. El beneficio de la misma es un patrón de radiación más concentrado, que a menudo resulta en un rango de

lectura más grande. La desventaja es que muchas etiquetas RFID también están polarizadas linealmente, y ambos elementos deben orientarse hacia la misma dirección para lograr una comunicación óptica. Esto lleva a un proceso en el cual las cajas que se mueven en una cinta transportadora deben estar orientadas de una manera específica para poder ser leídas, en caso contrario pasan desapercibidas por el lector.

- **Antena con polarización circular**: Por otra parte, la polarización circular quiere decir que el componente del campo electrónico de una señal RF gira (en el sentido de las agujas del reloj o en sentido contrario) mientras se propaga alejándose de la antena de interrogación. La clara ventaja de esta opción es que la orientación de la etiqueta RFID ya no es un elemento importante. La desventaja es que la intensidad de la señal de la polarización circular es más dispersa que la de la polarización lineal, lo que limita el rango.

Ya que para el proyecto los tags siempre se van a colocar en una misma orientación, se usa **antenas lineales**.

2.2.2. Diseño de la antena

Existen varios tipos de antenas, pero debido a que en este caso se requiere maximizar el alcance, se usa una antena direccional en configuración lineal para que la energía concentre en una sola dirección, por lo tanto se diseñó una antena yagi.

La antena se diseñó en un software QY4 para diseño de antenas yagi, los datos que se ingresaron son los siguientes:

- Frecuencia central: 915MHz
- Número de directores: 4
- Diámetro del soporte: 12mm
- Diámetro del dipolo, reflector y directores: 7mm
- Impedancia de entrada: 50 Ω

El programa nos entrega la longitud de los directores, reflector y dipolo y sus respectivos espacios entre cada elemento:

		Dimensión (cm)
Reflector	R	16.4
Dipolo Doblado	DP	15
Director 1	D1	14.3
Director 2	D2	13.9
Director 3	D3	13.9
Director 4	D3	11.9

Tabla 2.5. Dimensiones de los elementos

		Distancia (cm)
Reflector - Dipolo	R-D	5.9
Dipolo - Director 1	DD1	3.8
Director 1- Director 2	DD2	5.2
Director 2- Director 3	DD3	6.4
Director 3- Director 4	DD4	8.5

Tabla 2.6. Distancias entre los elementos

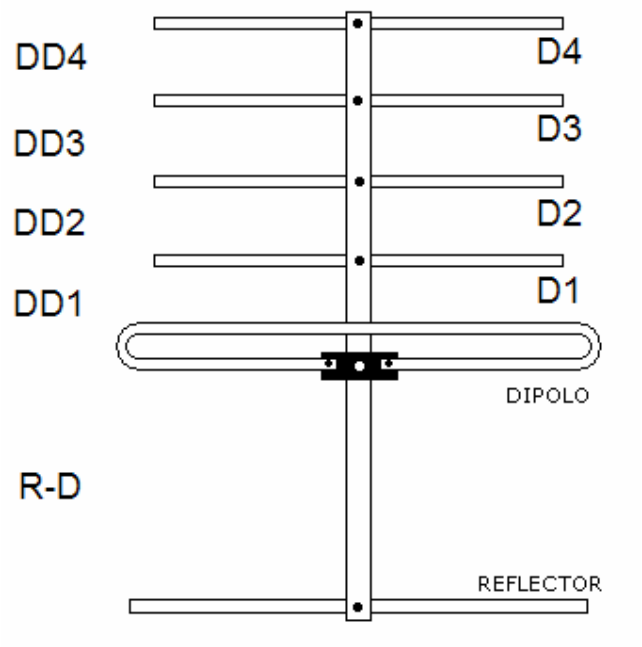


Fig. 2.4. Antena Yagi con 4 directores

2.2.3. Calculo de la longitud del cable

Ya que la longitud del cable debe ser un múltiplo entero de medias longitudes de onda se procede a calcular de la siguiente manera:

$$\lambda = \frac{c}{f} = \frac{3x10^8 m/s}{915x10^6 1/s} = 0.3279m$$

$$longitud\ cable = \#\frac{\lambda \cdot Fv}{2} = \frac{0.3279m \cdot 0.659}{2} = \#0.108m$$

= número entero

C = velocidad de la luz

Fv = factor de velocidad del cable RG8 y RG58

F = frecuencia

λ = longitud de onda

Entonces la longitud del cable (con coaxial RG58 o RG8) debe ser un múltiplo entero de 0.108m, así que una distancia prudente de la antena al lector es: # x 0.108 = 8x0.108 = **0.864m**

Nota: el lector usa conectores de tipo TNC polaridad reversa para conectarse con las antenas

2.3. DESCRIPCIÓN DEL CHIP A USAR

Ya que el sistema debe leer los tags a una distancia de hasta 2mts, se decidió usar chips en la frecuencia de los 900Mhz, ya que en esta frecuencia los tags logran distancias de lectura de hasta 3 mts; y como se mencionó en el capítulo 1, el chip que se está usando mayormente en el control de inventarios en las industrias que usan RFID es el **Clase 1 Gen2** que está en el rango de los 900Mhz, este estándar fue publicado por la empresa EPCglobal, la cual "conduce el desarrollo de la industria basada en las normas para el

código electrónico de producto (EPC) para apoyar el uso de identificación por radiofrecuencia (RFID)"

2.3.1. Introducción: Class 1 Generation 2 UHF Air Interface Protocol Standard (clase 1 generación 2 estándar de protocolo de comunicación aéreo)

Comúnmente conocido como el estándar "Gen 2", esta norma define los requisitos físicos y lógicos de un Tag pasivo con respuesta por reflejo (passive-backscatter), identificación por radio frecuencia (RFID) que funciona en la frecuencia de 860 MHz - 960 MHz.

Un interrogador (o lector RFID) envía información a uno o varios tags modulando una portadora RF usando **modulación por desplazamiento de amplitud de doble banda lateral** (double sideband amplitude shift keying DSB-ASK) o simple banda lateral SSB-ASK. Los tags reciben la energía de operación de la misma portadora RF.

Un interrogador recibe información de un tag trasmitiendo una portadora RF no modulada y escucha por alguna respuesta "backscattered" (reflejada o retrodispersión). Los tags comunican información por reflejo modulando la amplitud y/o fase de la portadora RF. La forma de comunicación entre interrogadores y Tags es "half-duplex", es decir que los tags no demodulan los comandos del interrogador mientras reflejan la señal.

2.3.2. Memoria del Tag

La memoria del tag es lógicamente separada en 4 bancos.

2.3.2.1. Memoria de reserva. Contiene los passwords (claves) para acceder y/o matar el tag. La clave para matar el tag esta en la dirección 00h a 1Fh (h = hexadecimal) la clave de acceso está almacenada en la dirección 20h-3Fh. El tag no ejecuta la operación de matar si la clave para matar es 0. La clave de acceso viene de fábrica con el valor 0**.** Si un tag tiene un

código de acceso diferente de 0, el interrogador requerirá enviar esta clave al Tag antes de recibir la información de la memoria de usuario. Si en los comandos que cambian la memoria del tag no se envía el correcto código de acceso, no se permitirá cambiar ninguna información contenida en el tag.

2.3.2.2.Memoria EPC. Memoria EPC. Contiene un CRC-16 (CRC cyclic redundany check) el CRC es usado por el tag para asegurar los comandos recibidos por el lector, y es usado por el lector para asegurar que se almacenó en el tag la información correcta, es decir el CRC es un espacio de memoria para asegurar que la información enviada y recibida está sin errores (parecido al bit de paridad que usa la comunicación RS232). Esta información se almacena en la dirección 10h-1Fh, y un código que identifica al tag en la dirección 20h

2.3.2.3.Memoria TID. (Tag IDentifier) Está en la dirección 00h-07h, contiene información del fabricante del tag, modelo y versión de Tag.

2.3.2.4.Memoria de Usuario. Permite al usuario guardar información

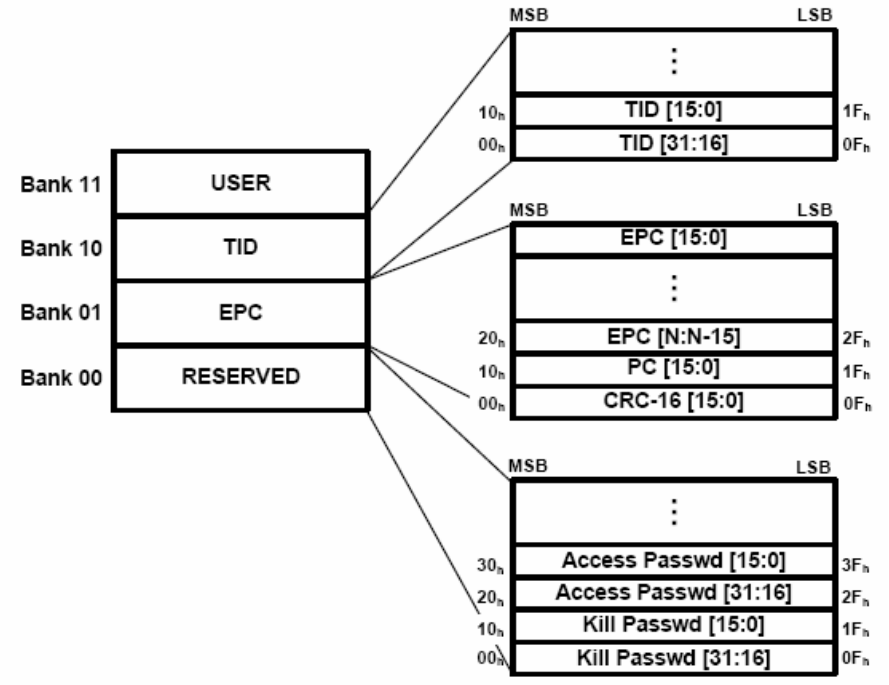


Fig 2.5. Mapa de memoria lógica

Los datos donde se almacena la información que usará el lector están en el **Bank 11_2** (o Banco 3) este contiene 96 bits que pueden ser programados por el lector con la información que se desee, la información contenida en los bancos 0 al 2 no puede ser leída directamente por el lector.

CAPÍTULO 3

DESCRIPCIÓN DEL FUNCIONAMIENTO

3.1. DESCRIPCIÓN DEL FUNCIONAMIENTO DE LA EMPRESA MILBOOTS

3.1.1. Introducción

La empresa MilBoots es una compañía que fabrica botas de caucho, y distribuye sus productos en todo el Ecuador y Colombia. La fábrica se encuentran en el parque industrial de la ciudad de Ambato (calle F y Av. IV lote 14)

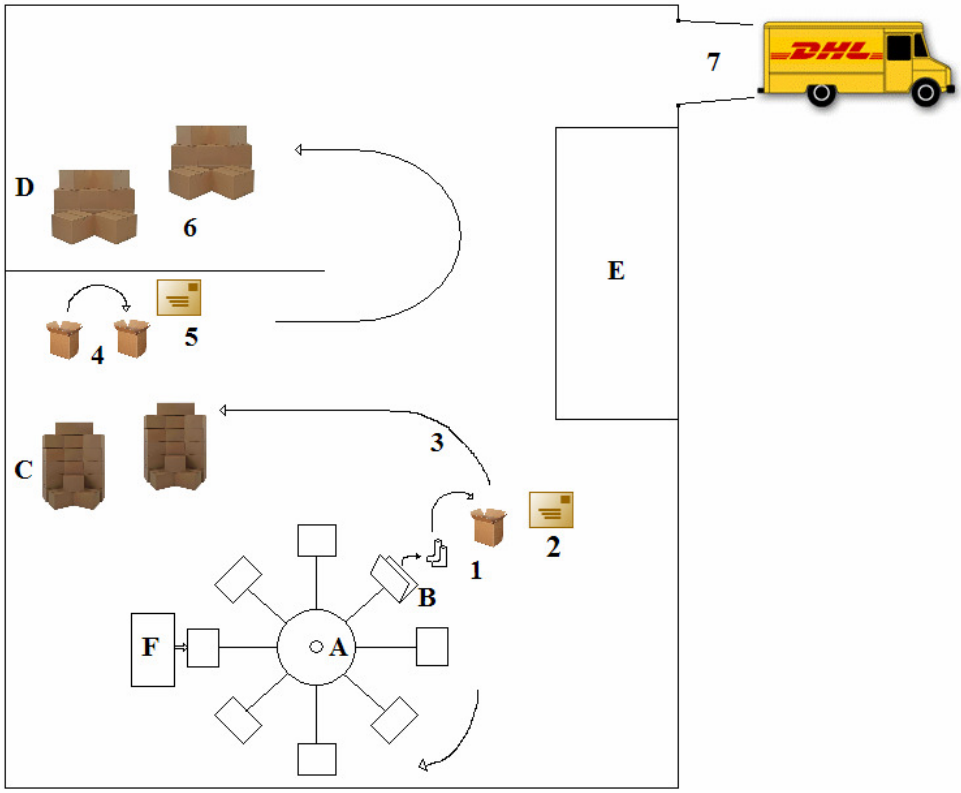


Fig. 3.1. Vista superior del galpón de la fábrica MilBoots

3.1.2. Descripción de partes:

A. El corazón de la fábrica es el sistema de inyección que prácticamente fabrica toda la bota, éste es un aparato que tiene un eje y en cada uno de sus brazos se encuentra una "estación" en la cual se encuentra el molde de la bota, conforme va girando la estación, pasa por procesos que van inyectando los materiales a la estación (ver Figura 3.1. F)
B. Esta es la estación, es similar a una caja, dentro de ésta se encuentra el molde de un par de botas, una vez que la estación ha girado 360° y regresa a la posición de inicio (justo donde se encuentra la letra "B" en la Figura 3.1.) los trabajadores abren la tapa y sacan las botas ya fabricadas.
C. Esta es la "Bodega" donde se almacena todas las botas que se han fabricado. Las botas se guardan en cartones, cada cartón lleva al menos 10 botas de las mismas características.
D. Se guarda las mismas botas de "bodega" pero en las cajas se encuentran las botas conforme a los pedidos que se deben fabricar. Por ejemplo: en un mismo cartón pueden ir 3 botas talla 36, 3 botas talla 37 y 4 botas talla 38.
E. Las oficinas.

3.1.3. Funcionamiento de la Empresa

La empresa se maneja de la siguiente manera:

1. Varios agentes vendedores viajan por todo el país ofreciendo los productos
2. Cada fin de semana regresan a la fábrica y entregan en las oficinas las notas de pedido.
3. Una vez que se ha recopilado todas las notas de pedido se procede a decidir lo que hay que fabricar: se resta los pedidos de lo que hay en bodega y la diferencia es lo que se debe fabricar, por ejemplo: si en bodega hay 600 botas, pero hay 1000 botas por entregar, entonces hay que fabricar 400 botas.
4. Una vez que la estación ha terminado el proceso de fabricación de la bota, los trabajadores (**bodegueros**) levantan la tapa de la estación y sacan del molde la bota ya fabricada y empacan las botas en cantidades de 10 por cartón. (Fig. 3.1. 1)
5. Colocan una etiqueta en el cartón indicando el tipo de producto que está dentro de éste (la información de la etiqueta se llena a mano) (Fig. 3.1. 2)

6. Los bodegueros llevan los cartones del producto terminado a **bodega** (Fig. 3.1.C) usando el montacargas.
7. Los **empacadores** de los pedidos sacan el producto que se empacó en los cartones y los reemplazan por el producto del pedido, por ejemplo, si un pedido tiene 5 botas talla 37 y 5 talla 38 el empacador saca de otro cartón las 5 botas de talla 38 y lo ponen con 5 botas de talla 37 en un solo cartón.
8. A cada cartón se le adhiere una etiqueta la cual contiene información del tipo de producto que lleva y a qué pedido pertenece (la información de la etiqueta se llena a mano)
9. Una vez empacado el producto, éste queda en espera en la **bodega de producto empacado** (fig. 3.1.D) hasta que llegue el camión para transportar los pedidos y entregar a los respectivos clientes en las diferentes zonas del país.

3.2. DESCRIPCIÓN DEL FUNCIONAMIENTO CON EL SISTEMA RFID DE CONTROL DE INVENTARIOS

3.2.1. Descripción de la base de datos

Antes de describir el funcionamiento, se procede a detallar las tablas de la base de datos que se usa para el sistema, el nombre de la base de datos es “milboots” dentro de ésta existen 15 tablas, en cada una de las cuales se guardan los distintos tipos de información como bodega, pedidos, etc., a continuación se detalla la lista de tablas, sus funciones y cada una de sus columnas.

Cada columna de una tabla puede albergar un solo tipo de dato, estos se detallan a continuación:

Tipo de dato	Mínimo/máximo valor que puede tomar	Uso de memoria
Bigint	$\pm 9x10E^{18}$	8 bytes
Int	±2’000’000’000	4 bytes
Smallint	±32’767	2 bytes
Tinyint	0-255	1 bytes
Bit	0 ó 1	1 bit

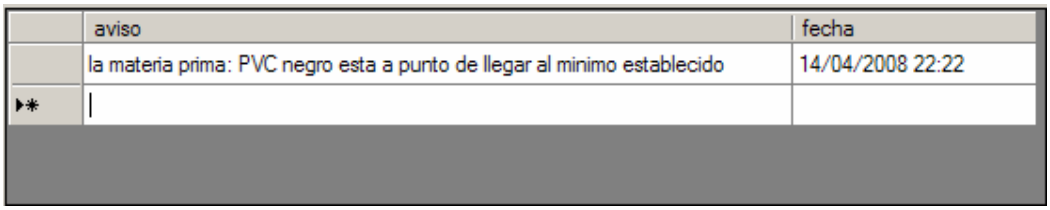
Float	$\pm 1{,}79 \times 10E^{308}$	8 bytes
Real	$\pm 3{,}4 \times 10E^{38}$	4 bytes
Datetime	Fecha Ej.: 10/04/2008 17:04:45	
Nvarchar(n)	Variable de tipo caracter, de tamaño variable, 'n' representa el Número de caracteres que puede tener.	2*n

Tabla 3.1. Tipos de datos en la base de datos

Nota:

El texto escrito entre paréntesis representa el tipo de datos que soporta esa columna.

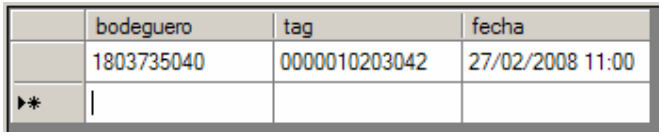
El prefijo dbo. Se usa para identificar las tablas.



	aviso	fecha
	la materia prima: PVC negro esta a punto de llegar al minimo establecido	14/04/2008 22:22
▶*		

Fig. 3.2. Tabla: dbo.avisos

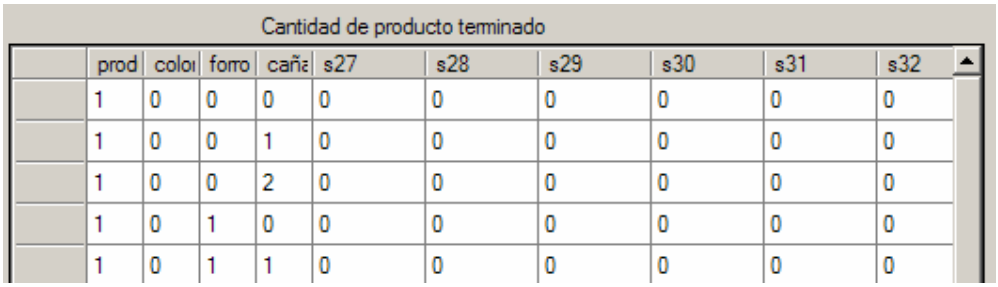
- Dbo.avisos, almacena los eventos relevantes que han sucedido tales como la falta de materia prima, etc., sus columnas son:
  - Aviso (nvarchar(150) es decir que es de tipo nvarchar con máximo 150 caracteres)
  - Fecha (datetime)



	bodeguero	tag	fecha
	1803735040	0000010203042	27/02/2008 11:00
▶*			

Fig. 3.3. Tabla: dbo.bodega

- Dbo.bodega, almacena el código de los tags que han ingresado a bodega a través del portal RFID
  - Bodeguero (nvarchar(12)) aquí se encuentra la cédula del trabajador que embodegó este tag
  - Tag (nvarchar(30)) representa la información que contiene un tag (ejm: 0000010203042D0A00000001) si un tag aparece en esta tabla quiere decir que se ha ingresado 1 cartón a bodega el cual tiene adherido un tag, el mismo que tiene ésta información.
  - Fecha (nvarchar(22)) fecha que se ingresó a bodega



Cantidad de producto terminado

	prod	colo	forro	caña	s27	s28	s29	s30	s31	s32
	1	0	0	0	0	0	0	0	0	0
	1	0	0	1	0	0	0	0	0	0
	1	0	0	2	0	0	0	0	0	0
	1	0	1	0	0	0	0	0	0	0
	1	0	1	1	0	0	0	0	0	0

Fig. 3.4. Tabla: dbo.bodegareal

- Dbo.bodegareal, almacena la cantidad de botas que hay en bodega, todavía sin empacar para los pedidos, toda la información se almacena en códigos numéricos
    - ID (int) es un número identificador para identificar con un único número a cada fila de la tabla (ésta columna está invisible)
    - Producto (tinyint) es el tipo de producto, en este caso solo fabrican botas por lo tanto el único número que aparece aquí es el 1
    - Color (tinyint) color de bota, ya que se fabrican 2 colores, aquí se puede tener el número 0 ó 1, el 0 representa el color NN (caña negra y suela negra) y el 1 es el NC (caña negra y suela crepé)
    - Forro (tinyint) existen 3 forros:

Código en la tabla SQL	abreviatura que usa la empresa	Lo que representa
0	D	Bota con media
1	L	Bota con malla
2	F	Bota sin forro

Tabla 3.2. Representación del código en la columna forro

    - Talla (tinyint) es la caña de la bota, es decir su altitud

Código en la tabla SQL	Código que usa la empresa	Lo que representa
0	S	La caña de la bota es pequeña
1	M	Mediana
2	X	larga

Tabla 3.3. Representación del código en la columna talla

- s27-s43 (int) en estas columnas se almacena la cantidad que existe de cada uno de estas botas.

ID	producto	color	forro	caña	S27	S28
0	1	0	0	0	0	100
1	1	0	0	1	0	0
2	1	0	0	2	0	0
3	1	0	1	0	0	0
4	1	0	1	1	50	0
5	1	0	1	2	0	0

Tabla 3.4. Ejemplo de la información almacenada en la tabla dbo.bodegareal

Esto quiere decir que en bodega hay 100 pares de botas del producto:1, color:0, forro:0, caña:0 de la talla 28 y hay 50 pares de botas del producto:1, color:0, forro:1, caña:1 de la talla 27, en términos reales: hay 100 pares de botas del producto:4x4, color: negro-negro, forro: medio, caña: small de la talla 28 y hay 50 pares de botas del producto:4x4, color: negro-negro, forro: malla, caña: medium de la talla 27.

El software sí admite nuevos tipos de producto, color, forro y caña; puede soportar hasta 255 valores en cada una de estas columnas.

- Dbo.configuracion, como el nombre lo indica aquí se almacena los datos de configuración del software
  - ind (tinyint) número de identificación
  - nproductos (tinyint) número de productos que se están fabricando, en la actualidad son sólo 1, máximo puede ser 255
  - ncolores (tinyint) número de colores que se están fabricando, en la actualidad son 2
  - nforros (tinyint) número de forros que se están fabricando, en la actualidad son 3
  - ntallas (tinyint) número de cañas que se están fabricando, en la actualidad son 3

- sizeminfab (tinyint) el size (o mas conocido como talla) más pequeño que se está fabricando, en la actualidad es la 37, mínimo puede ser 27
- sizemaxfab (tinyint) el size más grande que se está fabricando, en la actualidad es la 43, máximo puede ser 43
- nhoras (tinyint) número de horas que comprende una jornada de trabajo
- curva (tinyint) curva de producción, es decir cantidad de pares de botas que puede fabricar cada estación por hora
- nestaciones (tinyint) número de estaciones que están en funcionamiento
- nturnos (tinyint) número de jornadas laborales al día
- ultimosn (int) cada tag se programa con un número de serie único, ésta columna se usa para saber cual fue el último número de serie que se programó en un Tag.

	npedido	empacador	cliente	npares	ncajas
	1	1803735040	00034568	24	2
	2	1803735041	00034568	5	1
▶*					

Fig. 3.5. Tabla: dbo.empacado

- Dbo.empacado, se guarda información de los pedidos que ya fueron empacados y están listos para enviarse a los clientes.
  - npedido (bigint) número de pedido
  - empacador (nvarchar(15)) quién empacó el pedido
  - cliente (nvarchar(15)) RUC del cliente
  - ncajas (int) número de cajas de éste pedido
  - npares (int) número de pares de éste pedido

- dbo.iclientes, almacena información personal de los clientes
  - ruc (nvarchar(15))
  - nombre (nvarchar(15))
  - apellido (nvarchar(15))
  - empresa (nvarchar(15))
  - telefono (nvarchar(20))
  - fax (nvarchar(20))

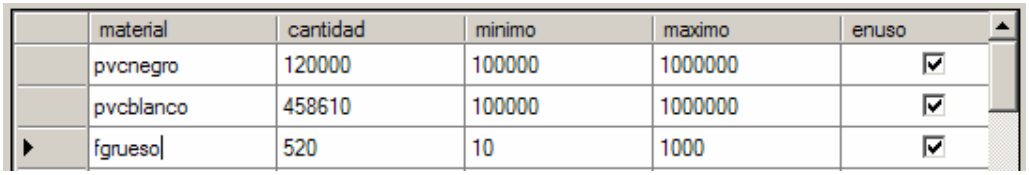
  - celular (nvarchar(20))
  - email (nvarchar(40))
  - gerente (nvarchar(20)) nombre del representante de este cliente
  - gtelefono (nvarchar(20)) teléfono del representante de este cliente
  - direccion (nvarchar(100))
  - ciudad (nvarchar(15))
  - sector (nvarchar(15))
  - pais (nvarchar(15))
  - dias (tinyint) aquí se guarda la cantidad de tiempo que se demora en llegar el pedido desde que sale de la fábrica hasta llegar al cliente
  - horas (tinyint)

- dbo.ipedidos, almacena la información de los pedidos que hayan hecho los clientes
  - npedido (bigint) número de pedido
  - vendedor (nvarchar(15)) cédula del agente vendedor
  - cliente (nvarchar(15)) RUC del cliente
  - fechap (datetime) fecha que se realizó el pedido
  - fechae (datetime) fecha que se debe entregar este pedido
  - observaciones (nvarchar(100))
  - posicion (tinyint) si posición =0 significa que éste pedido no ha sido empacado todavía, si es =1 éste pedido ya ha sido empacado.

- Dbo.ipedidost, almacena la cantidad y qué productos contienen todos pedido
  - npedidot (bigint) número de pedido
  - Producto (tinyint)
  - color (tinyint)
  - forro (tinyint)
  - talla (tinyint)
  - s27-s43 (smallint) size
  - posicion (tinyint) significa lo mismo que en la anterior tabla
  - fecha (datetime) fecha que se debe entregar este pedido

- dbo.ipersonal, almacena la información personal de los trabajadores de la empresa.
  - cedula (nvarchar(15))

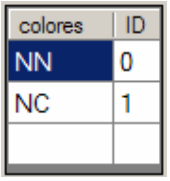
- nombre(nvarchar(15))
- apellido(nvarchar(15))
- telefono(nvarchar(20))
- celular (nvarchar(20))
- email (nvarchar(40))
- familiar(nvarchar(20)) nombre del familiar mas cercano
- ftelefono(nvarchar(20)) teléfono del familiar mas cercano
- direccion(nvarchar(100))
- sector (nvarchar(40)) sector en el que vive
- area (tinyint) área de trabajo, si =1 es empacador de producto terminado, si =2 empacador de pedidos



	material	cantidad	minimo	maximo	enuso
	pvcnegro	120000	100000	1000000	☑
	pvcblanco	458610	100000	1000000	☑
▶	fgrueso	520	10	1000	☑

Fig. 3.6. Tabla: dbo.stock

- dbo.stock, almacena la cantidad de materia prima que la empresa dispone
  - material (nvarchar(15)) se guarda el nombre de la materia prima
  - cantidad (float)
  - minimo (float) representa la cantidad mínima que la fábrica debe tener, si la "cantidad" es menor o igual a este valor, el programa da una alerta diciendo que se debe adquirir más de esta materia prima
  - maximo (float) cantidad máxima que la fábrica debe tener
  - enuso (bit) si es **True** significa que ésta materia prima sí se está usando en el proceso de fabricación, si es **False** el programa no toma en cuenta los mínimos de ésta materia prima.
  - ID (int) número identificador.



colores	ID
NN	0
NC	1

Fig. 3.7. Tabla: dbo.t_colores

- Dbo.t_colores
  - Colores (nvarchar(5)) nomenclatura o código que usa la empresa para identificar los colores, por ejemplo cuando se pone NN significa que es negro-negro, mientras que el software usa el número 0 para representar este color.
  - ID (tinyint) representación de ese color en la base de datos.
- Dbo.t_forros
  - forros (nvarchar(5))
  - ID (tinyint)
- Dbo.t_productos
  - Productos (nvarchar(5))
  - ID (tinyint)
- Dbo.t_tallas
  - tallas (nvarchar(5))
  - ID (tinyint)

	prod	colo	forro	caña	size	pvcnegro	pvcblanco	fgrueso
	1	0	0	0	37	20	5	0
	1	0	0	0	38	21	6	0

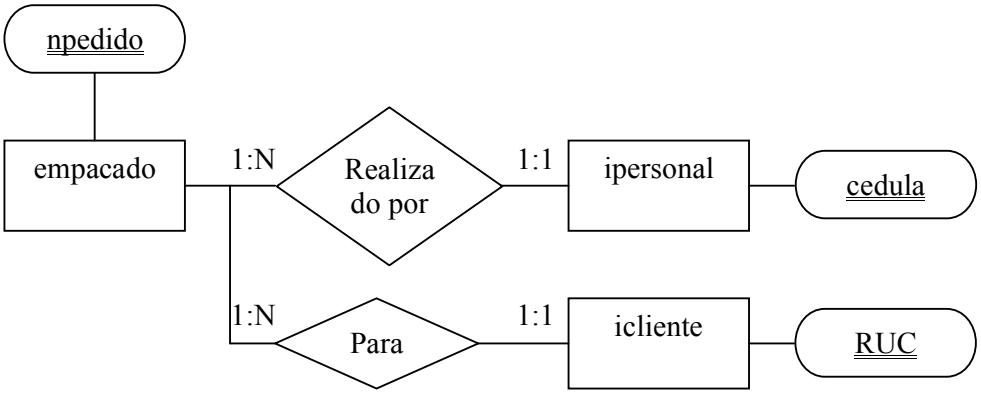
Fig. 3.8. Tabla: dbo.umprima

- Dbo.umprima, se usa para determinar la cantidad de materia prima usa cada tipo de bota
  - ID (bigint) número identificador de fila (invisible)
  - producto (tinyint)
  - color (tinyint)
  - forro (tinyint)
  - talla (tinyint)
  - size (tinyint)
  - pvcnegro (real)
  - pvcblanco (real)
  - fgrueso (real) forro grueso, conocido como malla
  - fdelgado (real) forro delgado, conocido como media
  - fundas (real)
  - cartones (real)

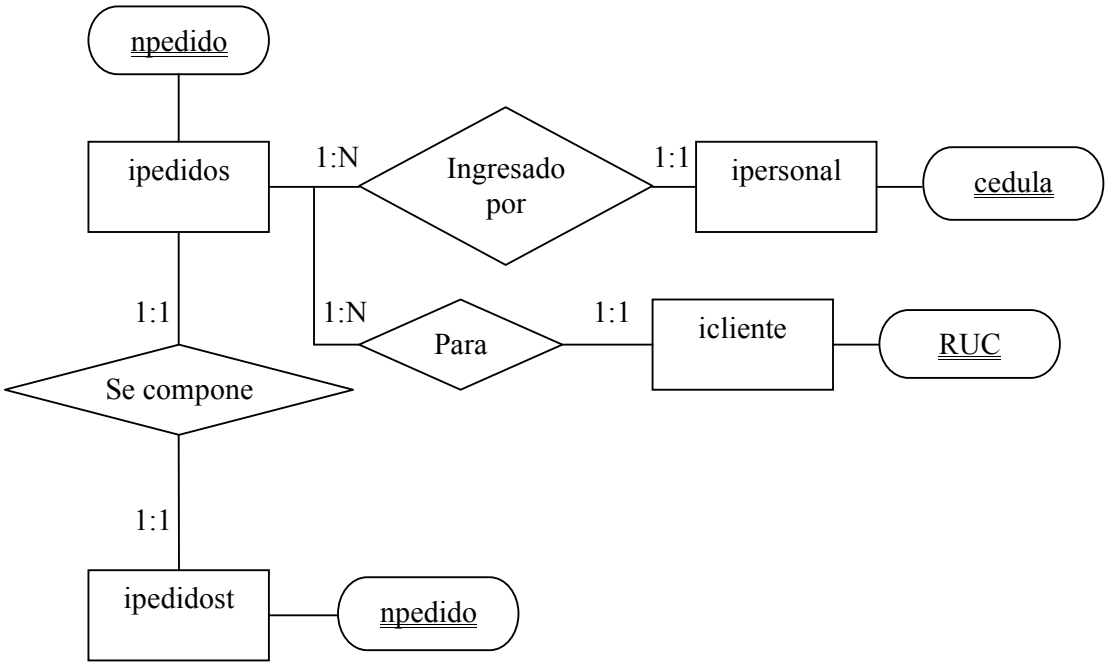
- grapas (real)
- cembalaje (real) cinta de embalaje
- material1-10 (real)

3.2.2. Diagramas Entidad-Relación

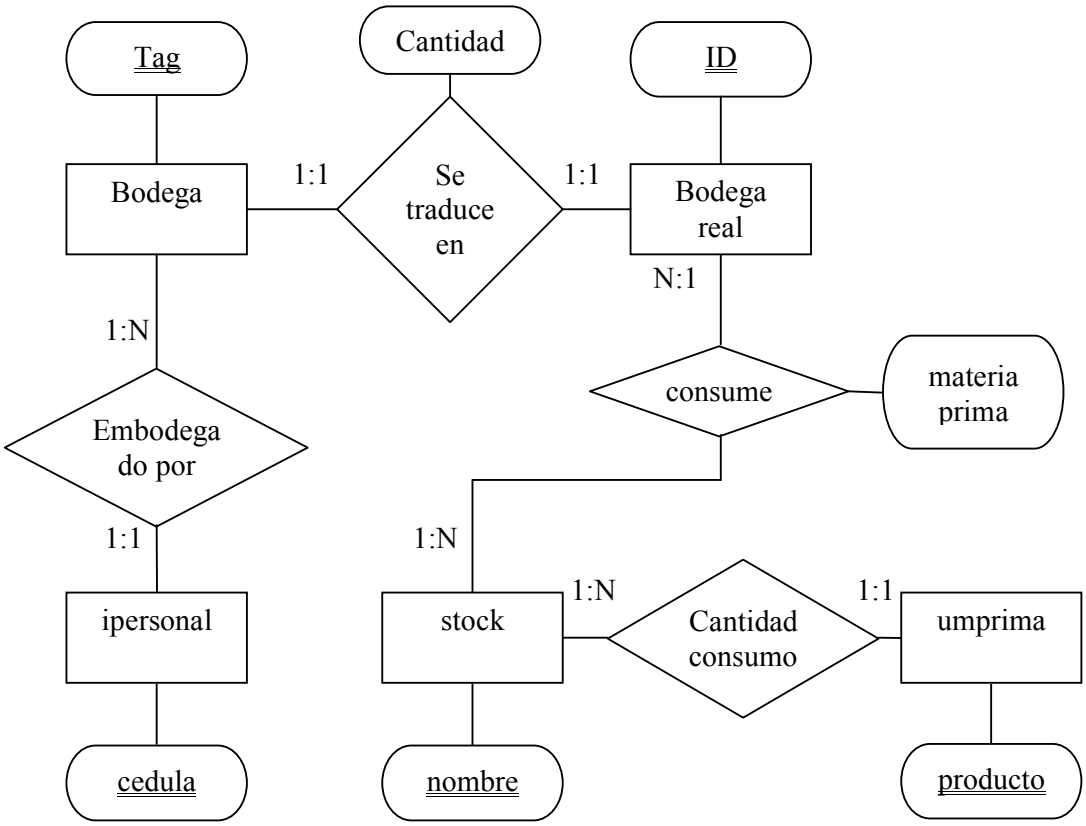
En la entidad o tabla **empacado** se guarda la información de los pedidos que han sido empacados y quien los empacó (la entidad **ipersonal** guarda una lista de todos los empleados de la empresa)



La entidad **ipedidost** posee la información de la cantidad de botas del pedido, en **ipedidos** se guarda información adicional acerca del mismo como el cliente y quien fue el agente vendedor que tomó nota del pedido.



En **bodega** se guarda el código de los Tags que se han pasado por el portal RFID y quién los embodegó, en **bodegareal** se traduce del Tag a cantidades reales de producto que se tiene; todo producto consume materia prima la cual se almacena en **stock**, y la materia prima que se consume depende del producto que se ingrese a bodega, la cantidad de materia prima que usa cada tipo de producto se guarda en la entidad **umprima** (uso de materia prima)



3.2.3. Descripción de los procesos ha realizar con el sistema RFID

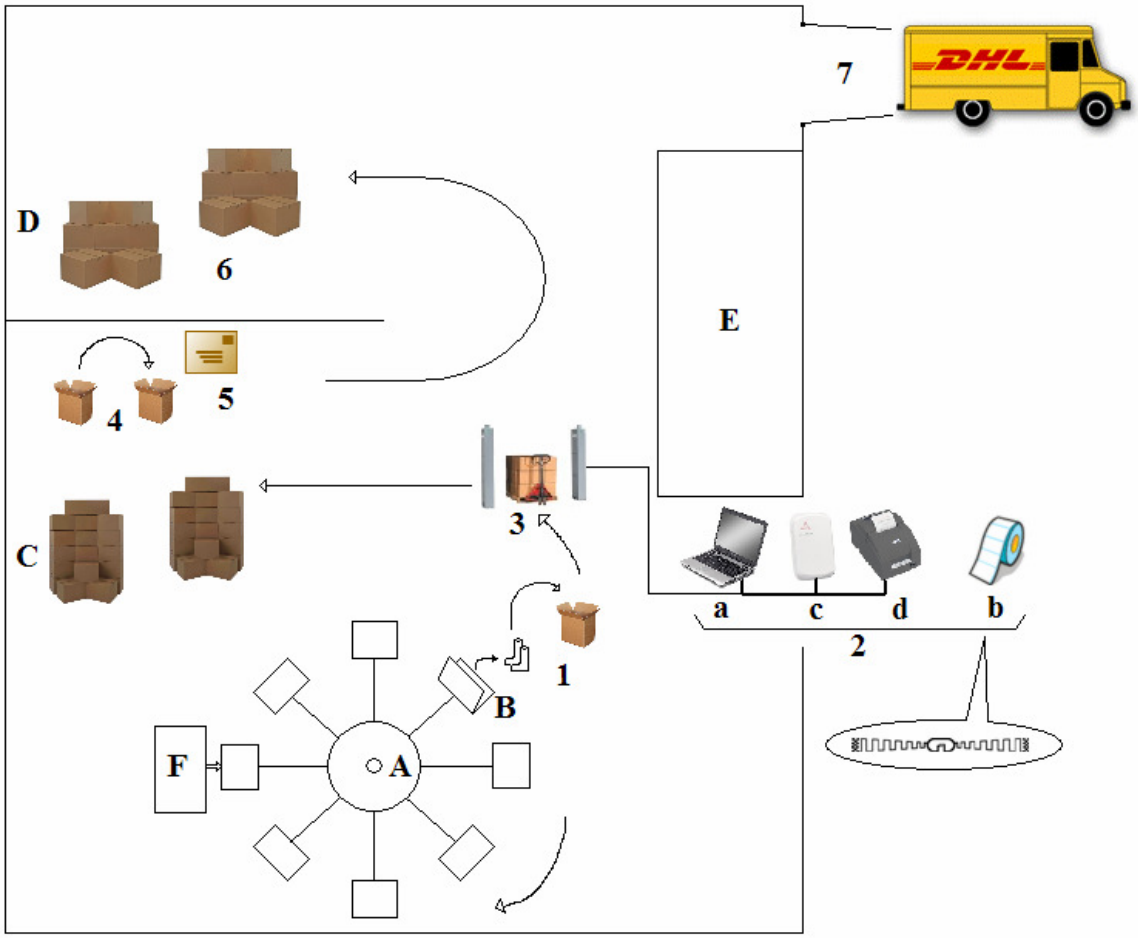


Fig. 3.9. Vista superior de la fábrica con el sistema RFID

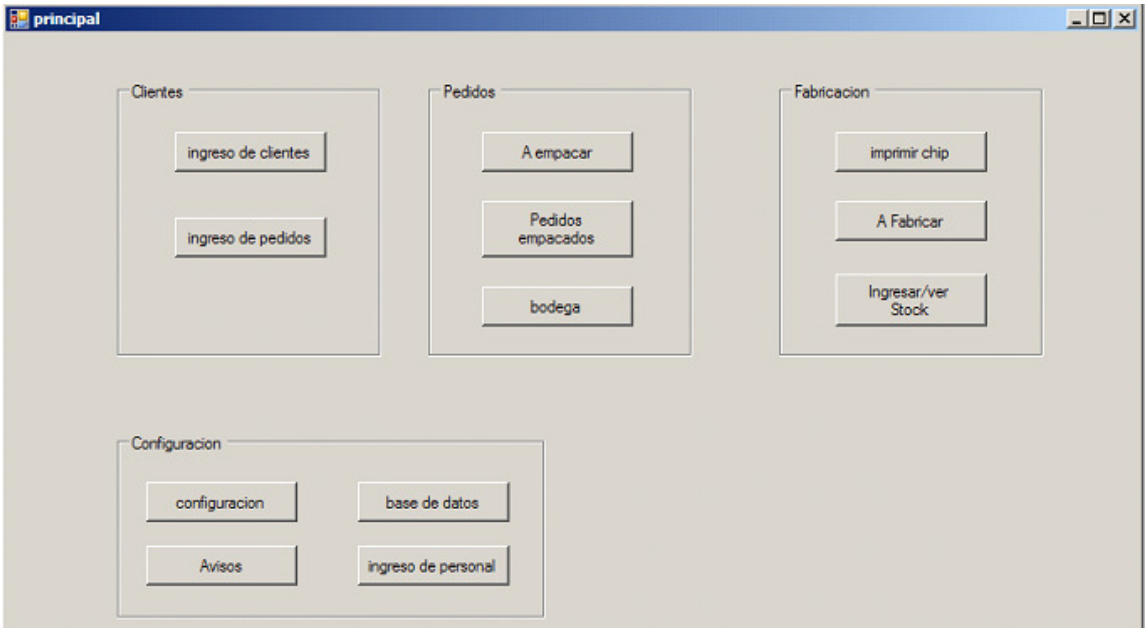


Fig. 3.10. Ventana principal del software

1. Varios agentes vendedores viajan por todo el país ofreciendo los productos
2. Cada fin de semana regresan a la fábrica e ingresan las notas de pedido en el software en la ventana de **ingreso de pedidos** (ver Fig. 3.10.)
3. Una vez que se ha recopilado todas las notas de pedido se procede a decidir lo que hay que fabricar: se resta los pedidos de lo que hay en bodega y la diferencia es lo que se debe fabricar, por ejemplo: si en bodega hay 600 botas, pero hay 1000 botas por entregar, entonces hay que fabricar 400 botas. El software hace esta resta automáticamente en la ventana de **a fabricar**.
4. Una vez que la estación ha terminado el proceso de fabricación de la bota, los trabajadores (**bodegueros**) levantan la tapa de la estación y sacan la bota ya fabricada y empacan las botas en cantidades de 10 por cartón (Fig.3.9. 1)
5. En lugar de llenar a mano una etiqueta informativa del producto que lleva el cartón ahora:
   a. Se dirigen a la computadora (fig. 3.9. 2.a) donde está instalado el software, y en la ventana (o mas conocido en términos de programación: formulario) de **imprimir chip** se escoge el tipo de producto y la cantidad de botas que contiene el cartón.
   b. Se arranca una etiqueta RFID (o también se conoce como inlay o tag) del rollo (fig. 3.9. 2.b)
   c. Acerca la etiqueta hacia la antena de escritura de tags (al menos 1mt.) (fig. 3.9. 2.c) y al presionar un botón se programa el tag.
   d. Y a su vez se imprime en una etiqueta con información de lo que lleva el cartón (en la impresora de la fig. 3.9. 2.d) la etiqueta se aprecia en la Fig. 3.11.

```
Producto: 4x4
Color: NEGRO-NEGRO
Forro: MALLA
Caña: SMALL
Talla: 37
Cantidad: 10
Fecha: 23/04/08
Tag: 000001000000250A 00000000
Bodeguero: 0092546940 – Diego Garcés
```

Fig. 3.11. Etiqueta de producto terminado

6. Los bodegueros llevan los cartones a bodega pasando a través del portal RFID (el bodeguero debe llevar consigo una tarjeta de identificación RFID en la cual está su número de cédula) el portal envía la información de los tags que leyó a la PC y **el software realiza las siguientes operaciones**:

   a. Agrega a la tabla **dbo.bodegareal**, lo que se ingresó a la bodega. Por ejm. Si ingresaron a bodega 5 cartones y en cada uno de ellos había 10 botas talla 27, entonces se almacena en **dbo.bodegareal** que ingresaron 50 botas talla 27, como se aprecia en la fig. 3.12.

	prod	colo	forro	caña	s27	s28	s29
	1	0	0	0	0	0	0
	1	0	0	1	50	0	0
	1	0	0	2	0	0	0
	1	0	1	0	0	0	0

Fig. 3.12. Tabla: dbo.bodegareal

   b. Ya que el programa sabe la cantidad de materia prima que usa cada bota (tabla **dbo.umprima** ver fig. 3.8.) se resta de la tabla **dbo.stock**, la cantidad de materia prima que se usó para todas las botas que se ingresaron. Por ejemplo, si se ingresaron 50 botas talla 27 y el software sabe que cada bota usa 200gr de PVCnegro, entonces resta de **dbo.stock** 200x50=**10000gr** de PVCnegro.

   c. Se almacena en la tabla: **dbo.bodega** los chips que leyó el lector así como el chip del trabajador, esto es para saber quien ingresó a bodega cada uno de los respectivos cartones. Por ejemplo. Como se ve en la fig. 3.13. el tag con información: 000001000000250A00… fue guardado en bodega por la persona con Nro. De cédula: 00925469…

	bodeguero	tag	fecha
▶	00925469...	000001000000250A00...	01/03/2008 20:37:34
	00925469...	000001000000250A00...	01/03/2008 20:37:30
	00925469...	000001000000250A00...	01/03/2008 20:37:33
	00925469...	000001000000250A00...	01/03/2008 20:37:28
	00925469...	000001000000250A00...	01/03/2008 20:37:30
	00925469...	000001000000250A00...	01/03/2008 20:37:34
	00925469...	000001000000260A00...	01/03/2008 20:37:29
	00925469...	000001000000270A00...	01/03/2008 20:37:38
	00925469...	000001000001250A00...	01/03/2008 20:37:39

Fig. 3.13. Tabla: dbo.bodega

7. El personal que empaca los pedidos, ingresa en la computadora (fig. 3.9. 2.a) a la ventana **A empacar** para saber qué pedidos están pendientes por empacar y cuales son los de mas urgencia, una vez que decide cual pedido va a empacar, presiona botón “Este pedido ha sido empacado” y el software realiza las siguientes operaciones:
   a. Resta de la tabla: **dbo.bodegareal** la cantidad de botas que corresponden a este pedido. Por ejemplo. Si en **dbo.bodegareal** habían 50 botas talla 27 y este pedido comprendía 30 botas talla 27, entonces ahora en **dbo.bodegareal** aparecerá 20 botas.
   b. Añade a la tabla: **empacado** el pedido que se empacó, indicando cual es la ciudad, cliente, número de cartones y número de botas.

8. Una vez empacado el producto, éste queda en espera en la **bodega de producto empacado** (fig. 3.9.D) hasta que llegue el camión para transportar los pedidos y entregar a los respectivos clientes en las diferentes zonas del país.

CAPÍTULO 4

DESCRIPCIÓN DEL SOFTWARE

4.1. INTRODUCCIÓN

El lector ALR9800 incluye varios programas y librerías en .Net y Java para que los desarrolladores de software puedan controlar el lector desde cualquier programa.
El presente proyecto se desarrolló en .Net en el lenguaje C#.

Para controlar el lector desde cualquier software en C# se debe añadir al proyecto una referencia llamada `AlienRFID1.dll`. La sección de código donde se escriben la librerías a usar en el programa, se escribe: `using nsAlienRFID;` y en la sección de código donde se programa se declara la clase: `public static clsReader mReader.` Una vez realizado esto se puede controlar el software. Por ejemplo si se desea que el lector programe los Tags a través de la antena 1 se escribe lo siguiente: `mReader.ProgAntenna = "1";`

4.2. DESCRIPCIÓN DE LOS COMANDOS QUE SE USAN EN EL SISTEMA

Nota: la frase "obtiene o pone" significa que esta función sirve para obtener información o para enviar información al lector.

- **Para obtener información**: `string ant_prog = mReader.ProgAntenna;` en la variable "`ant_prog`" se guardará `"1"` lo cual significa que el lector esta usando la antena "1" para programar los Tags.
- **Para enviar información al lector**: `mReader.ProgAntenna = "1";` esto significa que a partir de ahora el lector va a programar los Tags usando la antena "1"

A continuación se describe todos los comandos que se usaron para el desarrollo del sistema:

`result = mReader.TagList;` entrega en la variable tipo string "result" la lista de tags que el lector ha logrado captar, en forma de texto multilíneas.

`mReader.InitOnNetwork("192.168.1.100", 23);` inicializa la conexión con el lector de la dirección IP "192.168.1.100" a través del puerto 23.

`result = mReader.Connect();` se conecta con el lector, si la operación se completa exitosamente se asigna en la variable "result" la palabra "Connected"

`mReader.IsConnected;` devuelve en forma de boleado (boleano es una variable de tipo BIT es decir que solo toma 2 valores lógicos, "1" o "0" ó sino "True" o "False") el estado del lector, si esta conectado devuelve **True**.

`mReader.Login("alien", "password")` accesa al lector enviando el nombre de usuario "alien" y la clave "password" si no logra acceder, la función devuelve un False.

`mReader.AutoMode = "On";` coloca al lector en automode, es decir que el lector encuentra tags y los guarda automáticamente en una lista de tags.

`mReader.NetworkTimeout = "45000";` obtiene o pone el tiempo en segundos para cerrar una conexión cuando no ha habido una comunicación de datos con el lector, es decir que si no se ha intercambiado ninguna información entre lector y PC por más de 45000 segundos, el lector automáticamente se desconecta de la red LAN.

`mReader.KeepNetworkConnectionAlive = true;` obtiene o pone la propiedad que tiene el lector de mantener el lector conectado a la red todo el tiempo, es decir evita que se de el "networktimeout"

`mReader.AntennaSequence = "0,1";` obtiene o pone la secuencia y qué antenas se encenderán, en este ejemplo: primero se enciende la antena 0, luego la 1 y las antenas 2 y 3 no se van a encender (las 4 antenas que posee el lector se encienden solo una a la vez para evitar interferencias.)

`mReader.RFAttenuation = 0;` obtiene o pone la atenuación de la señal RF que emite el lector, incrementar el RFAttenuation en 10 disminuye la señal RF en 1dB

`mReader.RFLevel = 300;` obtiene o pone el nivel de potencia RF que emite el lector en dB/10 el valor máximo es 316.

`mReader.AutoStartTrigger = "0,1";` obtiene o pone el estado de las entradas para que el lector pase del estado pasivo al estado de trabajo, "0,1" es un número decimal que representa el estado en que las 8 entradas digitales del lector debe estar para que el lector pase al estado de trabajo, en este ejemplo $"0,1"_{10} = 0000\ 0001_2$ es decir que solo la primera entrada digital debe estar en "1" lógico para que el lector trabaje. Si en lugar de "0,1" estuviera "1,3" entonces las entradas digitales deben estar en 0001 0011, es decir que las entradas digitales "0" "1" y "4" deben estar en "1" lógico para que el lector se active.

`mReader.AutoStopTimer = "8000";` obtiene o pone el tiempo (en segundos) que el lector trabaja después de haber iniciado el trabajo.

`mReader.PersistTime = "5";` obtiene o pone el tiempo (en segundos) que la información de un tag permanece en el "taglist" interno del lector.

`mReader.NotifyAddress = mServers[0].NotificationHost;` obtiene o pone dónde el lector va a enviar las notificaciones periódicas que envía el lector.

`mReader.TagStreamAddress = mServers[1].NotificationHost;` obtiene o pone dónde el lector va a enviar el "tagstream" (lista de tags)

`mReader.IOStreamAddress = mServers[2].NotificationHost;` obtiene o pone dónde el lector va a enviar el "IOstream" (estado de las entradas y salidas discretas)

`mReader.TagListMillis = "On";` si está en "On", la fecha en la que el lector adquirió el tag lo pone también con milisegundos, ejemplo: `Tag:DEAD BEEF CAFE 8042, Date:2006/05/30 12:12:02.388`

`mReader.NotifyTime = "5";` obtiene o pone el tiempo (en segundos) en que se cierra un mensaje de notificación.

`mReader.NotifyFormat = "Text";` obtiene o pone el formato de la notificación.

`mReader.NotifyHeader = "Off";` obtiene o pone, si la notificación llevará o no encabezado.

`mReader.TagStreamFormat = "Terse";` obtiene o pone el formato del "tagStream" (lista de tags o TagList)

`mReader.IOStreamFormat = "Text";` obtiene o pone el formato con que el lector envía el estado de las entradas y salidas digitales del lector.

`mReader.IOStreamKeepAliveTime = "30";` obtiene o pone el tiempo en segundos que el lector mantiene abierto el socket TCP después de enviar la ultima notificación del estado de las entradas. Es decir que simplemente pasados los 30 segundos de no haber cambios en las entradas el lector, borra el contenido del "IOStream" interno del lector, es decir toda la información de los últimos cambios dados en las entradas del lector.

`mReader.TagStreamKeepAliveTime = "30";` obtiene o pone el tiempo en segundos que el lector mantiene abierto el socket TCP después de enviar la ultima notificación de la lista de tags.

`mReader.AutoWaitOutput = "-1";` obtiene o pone el estado de las salidas discretas del lector mientras éste se encuentra en estado de espera (estado de espera: ninguna de las antenas están activadas para recibir información de tags). Si es "-1" entonces no realiza ninguna operación en las salidas.

`mReader.AutoWorkOutput = "-1";` obtiene o pone el estado de las salidas digitales del lector mientras éste se encuentra en estado de trabajo, es decir que al menos una de las antenas están activadas para recibir información de tags. Si es "-1" entonces no realiza ninguna operación.

`mReader.AutoTrueOutput = "-1";` especifica el estado de las salidas digitales cuando el lector está en "AutoMode" modo automático.

`mReader.AutoFalseOutput = "-1";` especifica el estado de las salidas digitales cuando el lector no está en "AutoMode".

`mReader.Disconnect();` esta función hace que el lector se desconecte de la sesión de red a la cual está conectado.

Nota: todas estas funciones a excepción de la ultima mencionada, se ejecutan (tal como están escritas) al cargarse el software de control para configurar correctamente el lector.

4.3. DESCRIPCIÓN DEL SOFTWARE

Como se mencionó anteriormente el lenguaje de programación es C# .NET ya que el lector proporciona librerías en este lenguaje para usar el mismo.

La base de datos que su usa es SQL Server Express edition, ya que ésta viene incluida en el paquete de .NET express y es la más fácil de usar con el lenguaje C#. La diferencia entre .NET Express y .NET es que el Express es gratuito y la capacidad de la base de datos es máxima de 5 GB y no permite conexiones remotas. En el capítulo 3 se explicaron todas las tablas de la base de datos. Antes de dar una descripción del software desarrollado se explica la manera en que se usa la información en el tag.

4.3.1. Distribución de información en el Tag

Un tag se coloca en cada cartón que contiene el producto terminado (aproximadamente 10 botas por cartón) por lo tanto la información contenida en el chip se diseñó para que represente la cantidad y el tipo de material terminado que lleva cada cartón.

La memoria de usuario, donde se almacena toda la información para identificar al producto al cual el tag está adherido es de 96 bits, es decir 24 dígitos hexadecimales, la forma con la que se usa este espacio de memoria es la siguiente:

0000 01 02 03 04 2D 0A 00000000_{16}

Dígito: 23 0

- los dígitos del 20 al 23 MSB (es decir el 0000) se usa para identificar el tipo de información que lleva el tag.
    - Si es 0000 significa que éste es un tag para identificar un producto
    - Si es 0001 significa que éste es un tag para identificar a un empleado
- los dígitos del 18 al 19 MSB (es decir el 01) se usa para identificar el producto
- los dígitos del 16 al 17 MSB (es decir el 02) se usa para identificar el color del producto
- los dígitos del 14 al 15 MSB (es decir el 03) se usa para identificar el forro
- los dígitos del 12 al 13 MSB (es decir el 04) se usa para identificar la caña (o talla)
- los dígitos del 10 al 11 MSB (es decir el 2D) se usa para identificar la talla (o size)
- los dígitos del 8 al 9 MSB (es decir el 0A) se usa para identificar la cantidad de botas con este forro, caña y talla que lleva el cartón
- los dígitos del 0 al 7 MSB (es decir el 00000000) representa un número de serie que es único

4.3.2. Descripción del software desarrollado

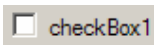
El software tiene varias ventanas (o formularios) para los diferentes propósitos que tiene el sistema, a continuación se detalla la lista de formularios y sus funciones:

- Principal, contiene todos los botones para alcanzar los distintos formularios del programa y para dar avisos visuales acerca de los procesos que se están realizando.
- iclientes, (ingreso de clientes) aquí se encuentra toda la información correspondiente a la lista de clientes que tiene la empresa.
- ipedidos, (ingreso de pedidos) los agentes vendedores ingresan aquí la hoja de pedido que cada cliente ha realizado.
- aempacar, éste formulario le indica a los empacadores de pedidos, cual es el pedido que se debe empacar.
- empacado, aquí se muestra una lista de los pedidos que ya han sido empacados y están listos para enviarse.

- bodega, se muestran 2 tablas, la una indica la cantidad de botas que la empresa ha fabricado (dbo.bodegareal) y que todavía no han sido empacadas para los pedidos y la otra indica el código de los tags y quién los embodegó (dbo.bodega)
- imchip, (imprimir chip) se usa para imprimir una etiqueta indicativa de lo que lleva cada cartón y programar ésta misma información en la etiqueta RFID.
- afabricar, indica la cantidad de botas que hay que fabricar, restando lo que hay en bodega de los pedidos.
- stock, aquí se muestra una lista de todas las materias primas que la empresa usa para fabricar las botas y la cantidad que resta de cada una de ellas.
- configuración, aquí se puede modificar datos como la cantidad de materia prima que usa cada bota, las tallas que la fábrica esta produciendo, etc.
- Bdatos, (base de datos) en este formulario se encuentran todas las tablas que tiene la base de datos, para poder modificar cualquier dato a conveniencia.
- avisos, esta ventana se despliega cada vez que hay un evento relevante, por ejemplo que un bodeguero ingresó botas a la bodega pero sin identificarse, o cuando cierta materia prima está a punto de acabarse y debe hacerse un nuevo pedido de ésta.
- ipersonal, (ingreso de personal) aquí se encuentra toda la información correspondiente a la lista de trabajadores que tiene la empresa, y tiene un botón para programar un carné RFID (en el cual se programa el número de cédula del trabajador)

Antes de describir el código fuente se muestra los diferencies prefijos para diferenciar los diferentes elementos de un formulario:

- button: b_xxxx (ejm: al botón borrar se le asigna el nombre: b_borrar) 

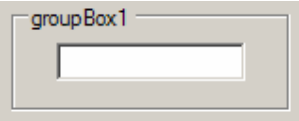
- textBox (lugar donde se ingresa texto): x_xxxx 
- checkBox (si está con un visto, el checkbox devuelve el valor “True”): ch_xxxx 

- label (etiqueta informativa): l_xxxx 

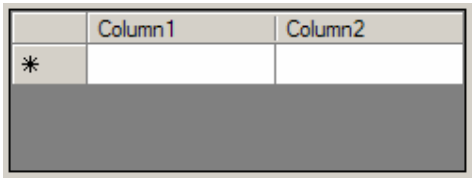
- numericUpDown (para ingresar valores numéricos): n_xxxx 

- comboBox (contiene una lista de elementos): c_xxxx 

- groupBox (contiene varios controles): groupBox1…#



- progressBar (indicador de barra, la barra azul crece o decrece según la programación): progressBar1…# 
- datagrid (semejante a una hoja de Excel, el cual tiene cierta cantidad de filas y columnas para ingresar datos) : dg_xxxx o xxxxDataGridView



A las tablas de la base de datos se las identifica con el prefijo dbo.xxx

El nombre del formulario tiene la extensión .cs: xxxx.cs

Cuando se mencione la palabra `#region xxxx` significa que éste es un espacio de código (parecido a una función) el cual tiene una etiqueta de nombre `xxxx`

A continuación se describe cada uno de los formularios del software; al ejecutarse el programa se abre la ventana principal.cs, y desde aquí se puede acceder a las demás ventanas del software. Primeramente se describe brevemente el formulario, luego en la figura subsiguiente se muestra la pantalla del formulario y luego se muestra el código fuente que contiene este formulario. No existe un código fuente único, cada formulario tiene su propio código fuente.

4.3.2.1. Principal.cs

Esta es la pantalla principal, que aparece cuando se ejecuta el programa, esta ventana siempre estará visible, y desde este formulario se accede al resto de las pantallas.

Tiene varios íconos para indicar si el software está conectado con el lector, si hay avisos por leer, y el número (fig. 4.1. “2”) indica el número de tags que se han leído en los últimos 30 segundos.

Pantalla:

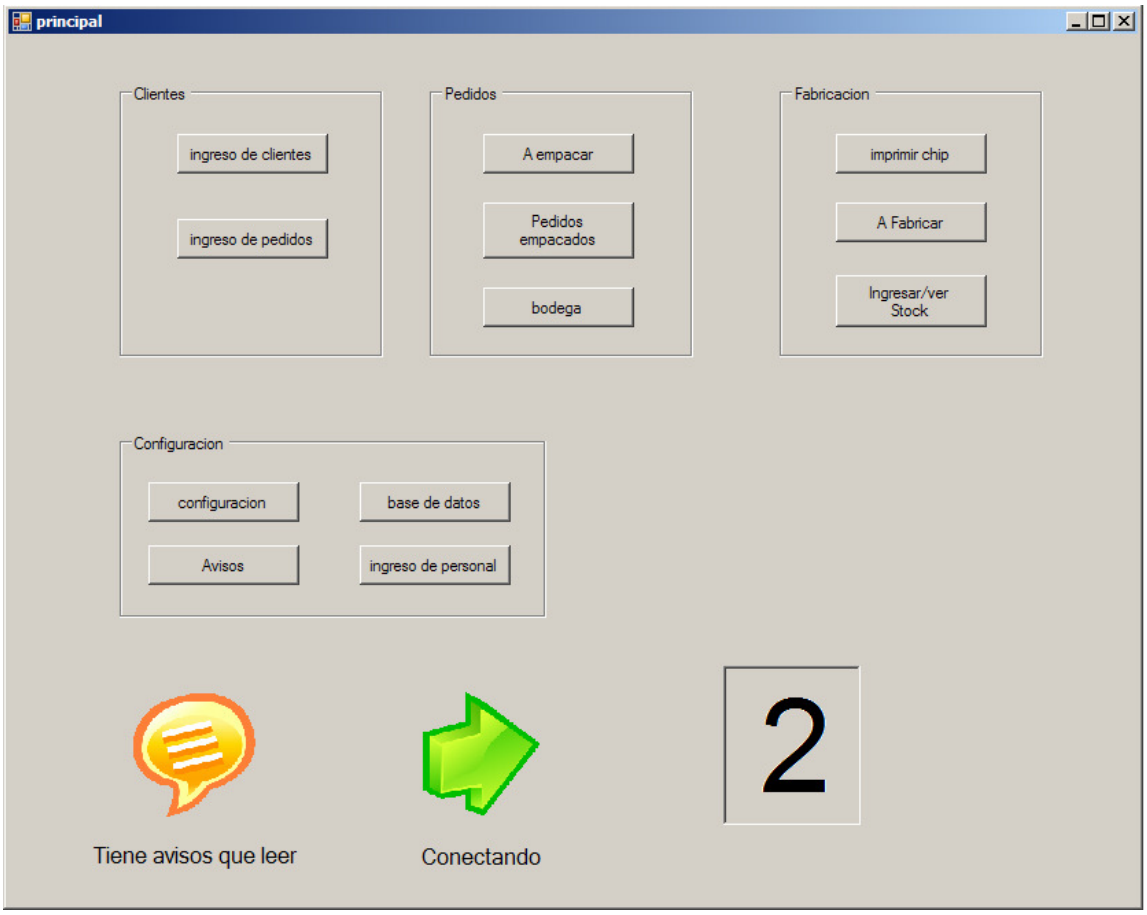


Fig. 4.1. Formulario: principal

Fig. 4.2. Este logo aparece cuando el software no se puede conectar con el lector

- Botón: “ingreso de clientes” abre la ventana iclientes.cs
- Botón “ingreso de pedidos” abre la ventana ipedidos.cs
- Botón “A empacar” abre la ventana aempacar.cs
- Botón “pedidos empacados” abre la ventana empacado.cs
- Botón “bodega” abre la ventana bodega.cs
- Botón “imprimir chip” abre la ventana imchip.cs
- Botón “A fabricar” abre la ventana afabricar.cs
- Botón “Ingresar/ver stock” abre la ventana stock.cs

- Botón “configuración” abre la ventana configuracion.cs
- Botón “Avisos” abre la ventana avisos.cs
- Botón “base de datos” abre la ventana bdmanual.cs
- Botón “ingreso de personal” abre la ventana ipersonal.cs

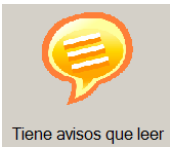


Fig. 4.3. Avisos

Fig. 4.4. Conectando con el lector

Fig. 4.5. Lector conectado

Fig. 4.6. Número de tags

La figura 4.3. Se hace visible sólo cuando hay datos dentro de la tabla dbo.avisos.

La figura 4.4. Se hace visible cuando el lector está tratando de establecer conexión con el lector.

La figura 4.5. Se hace visible cuando el lector está conectado al software

La figura 4.6. Es un textBox (nombre: x_items) el cual indica el número de tags que el lector ha captado en los últimos 30 segundos

Código Fuente:

- Al hacer clic en el botón “ingreso clientes” (b_iclientes) se ejecuta esta función:

```
private void b_iclientes_Click(object sender, EventArgs e)
{
      iclientes viclientes = new iclientes();
      viclientes.Show();
}
```

La cual crea la clase `viclientes` y usando esta clase se llama a la función `show()` para mostrar el formulario **iclientes.cs** en la pantalla. Lo mismo sucede para los demás 11 botones que abren a 11 formularios

- La función: `timer_conectar_Tick()` es un temporizador que se ejecuta cada 5 segundos, cada vez que se ejecuta, el código dentro de ésta función trata de conectarse con el lector, si al cabo de 10 ejecuciones de esta función el lector no se conecta, entonces hace visible la figura 4.2. en el formulario.

4.3.2.2. iclientes.cs

En este formulario se ingresan los datos personales de los clientes que tiene la empresa, dentro del groupbox **Tiempo en llegar la mercadería** se coloca el tiempo que se demora en llegar el pedido desde que sale de la fábrica hasta el cliente.

Pantalla:

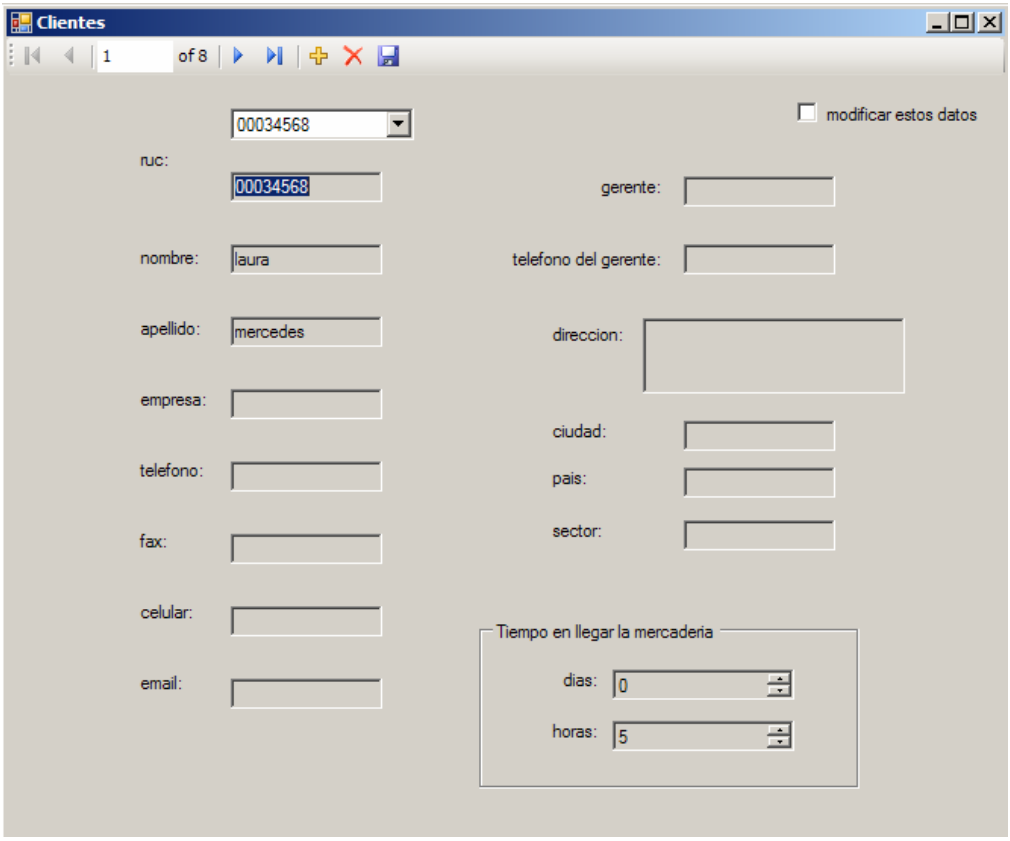


Fig. 4.7. Formulario: iclientes

Código Fuente:

Cada uno de los textBox tienen conexión directa con las columnas de la tabla dbo.iclientes, para hacer esto simplemente se hace lo siguiente:

- Añadimos una base de datos al software: en la barra de herramientas hago clic en data->add new datasources -> new connection -> busca la base de datos -> OK -> y clic en siguiente hasta finalizar.
- Luego de esto aparecerá una ventana al lado izquierdo del programa de desarrollo donde se muestran las bases de datos (ver fig. 4.8.), ahí se encuentra desplegado la base de datos milboots y sus respectivas tablas.
- Si se desea que toda una tabla aparezca en forma de datagrid sobre el formulario, simplemente se arrastra de esta ventana "Data Sources" la tabla que se desea hacia donde se encuentra el formulario.

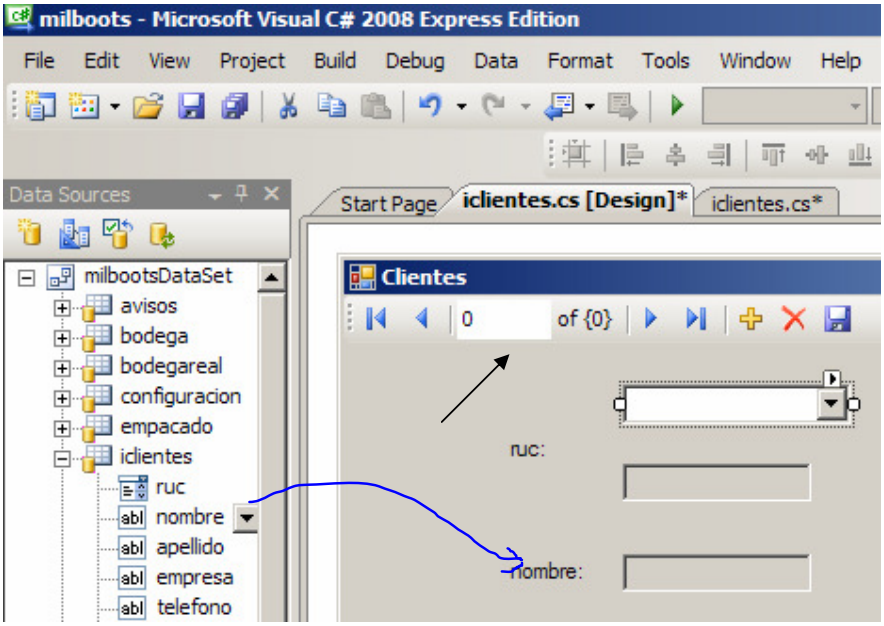


Fig. 4.8. Método para conectar un textbox con la columna de una tabla, en este caso la columna es: nombre

La flecha negra indica el "binding navigator" que es el menú de herramientas para desplazarse por los datos de una tabla, si se hace clic en el botón el dato mostrado en el textbox cambia al dato que se encuentra en la siguiente fila de la tabla, de está columna;

por ejemplo si el textbox es “nombre:” cada vez que se haga clic en este botón, el nombre mostrado en el textbox cambia.

El botón añade una fila más a la tabla.

El botón elimina la fila actualmente mostrada en el textbox, es decir si en el textbox del nombre está “Javier” se borra toda la información de Javier (ruc, dirección, etc.)

El botón guarda la información actualmente mostrada en los textBox.

4.3.2.3. ipedidos.cs

En este formulario se ingresan los pedidos que traen los agentes vendedores cada semana.

Para agregar un nuevo pedido se hace clic en el botón se llenan los datos del pedido y luego se guarda con el botón

Para borrar un pedido completo se escoge el pedido con el comboBox “Nro de pedido” se hace clic en y a continuación se hace clic en el botón

Pantalla:

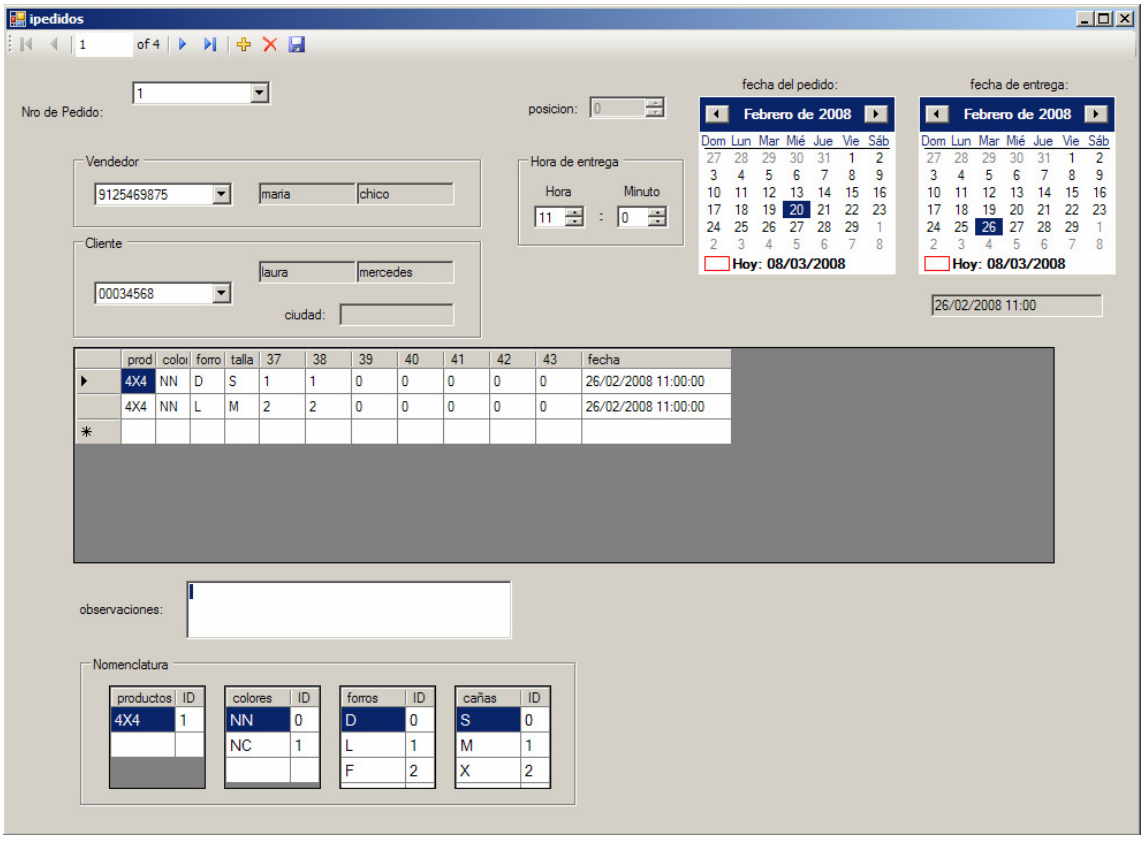


Fig. 4.9. Formulario: ipedidos

En el combobox “Nro de pedido” se escribe el número de pedido que se encuentra escrito en la hoja de pedido, este número es único.
En el groupbox “Vendedor” se escoge el vendedor que realizó este pedido
En el groupbox “Cliente” se escoge el cliente
El NumericUpDown “posición” indica la posición del pedido, si es 0 este pedido no ha sido empacado todavía, si es 1 ya ha sido empacado.
El groupbox “hora de entrega” se escribe la hora a la que el pedido debe llegar al cliente.
El monthcalendar “fecha del pedido” se coloca la fecha en la que se realizó el pedido
El monthcalendar “fecha de entrega” se coloca la fecha a la que se debe entregar el pedido.
El textbox debajo de éste, indica en forma de texto la fecha de entrega.

En el **datagrid** se coloca la cantidad y el tipo de botas que ha hecho el cliente, en la primera columna se coloca el tipo de producto, en la actualidad el único producto que se fabrica es la bota de caucho la cual tiene por nombre 4x4, en la columna color, forro y caña se coloca la información que corresponda según la nomenclatura que se encuentra en los datagrids que se muestran en la parte inferior.
En las columnas de las tallas (27-43) se coloca la cantidad de botas.

Nota: El pedido se guarda en dos tablas, dbo.ipedidos y dbo.ipedidost; en dbo.ipedidos se guarda la información del pedido, como el cliente, el vendedor, etc. y en dbo.ipedidost se guarda el tipo y la cantidad de producto que pidió el cliente, por ejm. Los datos que se encuentran en la figura 4.9. Se almacenan de la siguiente forma en las tablas:

	npedido	vendedor	cliente	fechap	fechae	observaciones	posicion
▶	1	9125469875	00034568	20/02/2008	26/02/2008 11:00		0
	2	9125469875	00034568	20/02/2008	27/02/2008 11:00		0
	3	9125469875	00034568	20/02/2008	28/02/2008 15:00		0
	4	9125469875	00034568	20/02/2008	27/02/2008 11:00		0
*							

Fig. 4.10. Tabla dbo.ipedidos, la primera fila muestra los datos de la figura 4.9. que es el pedido #1

	npedidot	producto	color	forro	talla	s37	s38	s39	s40	s41	posicion	fecha
▶	10	1	0	0	0	1	1	0	0	0	0	26/02/2008 11:00
	11	1	0	1	1	2	2	0	0	0	0	26/02/2008 11:00
	20	1	0	0	0	6	0	0	0	0	0	27/02/2008 11:00
	30	1	0	0	0	30	30	30	0	0	0	28/02/2008 15:00
	40	1	0	0	2	7	7	7	0	0	0	27/02/2008 11:00
*												

Fig. 4.11. Tabla dbo.ipedidost, las 2 primeras filas indican las botas del pedido #1

Como se aprecia en la figura 4.9. Hay dos filas en el datagrid las cuales pertenecen al pedido 1, para representar esto en la tabla dbo.ipedidost (figura 4.11.) la primera columna indica el número de pedido mas una terminación que va desde el 0 hasta el 9, en este caso tenemos el 10 y 11 que significa que pertenecen al pedido #1.

Código Fuente:

- La siguiente función se ejecuta cuando se hace clic en el botón  (guardar):

```
private void ipedidosBindingNavigatorSaveItem_Click(object sender,
EventArgs e)
```

La función realiza lo siguiente:

1) Si es que se ha hecho anteriormente clic en el botón  (borrar) se ejecuta la región `#region borra todos los pedidos de ipedidostData q pertenecen a este c_npedido` (líneas 35-62) la cual borra de las tablas dbo.ipedidos y dbo.ipedidost toda la información de éste pedido.
2) Si no se ha hecho clic en borrar entonces significa que los datos estén siendo añadidos o modificados. La `#region verifica si los datos son correctos` (líneas 66-151) verifica que los datos ingresados en el datagrid estén dentro de los parámetros, por ejm: si se ingresa en color las letras “NG” el programa verifica que dentro de los colores esté este parámetro, como (referirse a la figura 4.9.) se aprecia que en el datagrid del color solo hay el NN y el NC, entonces para este caso se muestra un mensaje: `"no se puede identificar el texto ingresado en alguna celda de producto, color, forro o caña"` ésta región también verifica que la cantidad de botas no sea mayor a la máxima que es 32000 debido al tamaño de la tabla que permite un número máximo de 16 bits, si algún número es mayor aparece el mensaje: `el pedido exede el valor maximo permitido que es 32000`

3) Si los datos son correctos, la `#region traduce tipos de productos a 1s y 0s` (líneas 153-218) traduce los datos del datagrid al formato usado en la computadora
4) Si los datos van a ser ingresados por primera vez, es decir que se ha hecho clic en el botón  , se ejecuta la porción de código (líneas 245-284) para guardar la información.
5) Si los datos de éste pedido van a ser modificados: verifica la cantidad de filas en el datagrid y compara con las filas que pertenecen a este pedido en la tabla dbo.ipedidost. (líneas 290-300)
   a) La `#region actualiza los datos ya existentes y añade los nuevos` (líneas 301-407) se ejecuta si es mayor o igual al número de filas que ya existen en la tabla dbo.ipedidost, ésta actualiza los datos con la función `ipedidostTableAdapter.Update` (líneas 316-369) y añade las nuevas filas que se encontraron con la función `ipedidostTableAdapter.Insert` (líneas 380-405)
   b) Si es menor la `#region actualiza los datos ya existentes y borra las filas que no se encontraron` (líneas 412-508), actualiza las filas que ya existen y las filas sobrantes en la tabla dbo.ipedidost son borradas con la función `ipedidostTableAdapter.Delete`

	npedidot	producto	color	forro	talla	s37	s38	s39	s40	s41	posicion	fecha
▶	10	1	0	0	0	1	1	0	0	0	0	26/02/2008 11:00
	11	1	0	1	1	2	2	0	0	0	0	26/02/2008 11:00
	20	1	0	0	0	6	0	0	0	0	0	27/02/2008 11:00
	30	1	0	0	0	30	30	30	0	0	0	28/02/2008 15:00
	40	1	0	0	2	7	7	7	0	0	0	27/02/2008 11:00
*												

a)

	prod	colo	forro	talla	37	38	39	40	41	42	43	fecha
▶	4X4	NN	D	S	1	1	0	0	0	0	0	26/02/2008 11:00:00
*												

b)

Fig. 4.12. a) Tabla dbo.ipedidost, b) datagrid en la ventana de ipedidos.cs

Como se aprecia en la figura 4.12 el b) tiene solo una fila mientras que el a) tiene dos, al momento de ejecutarse la función guardar, el programa va a ejecutar 5) b) (`#region actualiza los datos ya existentes y borra las filas que no se encontraron`) porque hay menor número de filas en el datagrid que en la tabla.

Hay que tener en cuenta que en la ventana de "pedidos" fig. 4.12.b) el tipo de producto está en letras y en a) está en números, como ya se ha explicado anteriormente, el programa traduce de letras a números para guardar en la base de datos

- Esta función se ejecuta al abrirse el formulario de "pedidos":

```
private void ipedidos_Load(object sender, EventArgs e)
```

Esta función pone en estado invisible los tamaños de botas que en la actualidad no se fabrican, por eso las columnas 27 a 36 no se observan en la Fig. 4.12.b) pero en el futuro tienen paneado fabricar desde la 27, para que se hagan visibles estas columnas se tendrá que acceder a la pagina de configuración.

- Esta función se ejecuta cuando cambia el texto del textbox donde se escribe el nuevo número de pedido:

```
private void t_npedido_TextChanged(object sender, EventArgs e)
```

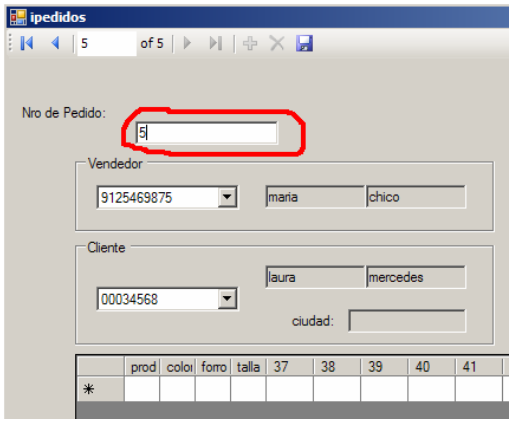


Fig. 4.13. Textbox: t_npedido

El código de ésta función hace o no visible el botón  dependiendo si el número de pedido que se ingresó es correcto (no es repetido y no contiene letras) si no tiene ningún problema, hace visible el botón para que se pueda guardar los datos.

4.3.2.4. aempacar.cs

Este formulario muestra qué pedidos se deben empacar; procedimiento:

1) Se escoge el pedido, sea este manual o automático

a) Manual: se escoge mediante el combobox que dice “por pedido:” el pedido a empacar
b) Automático: el programa automáticamente escoge el pedido que es mas próximo a empacar

2) Una vez escogido el pedido, se hace clic en el botón “escoger pedido” y los datos de este pedido se cargarán en el “dg_pedido” (datagrid 3)
3) El programa verifica que en bodega (bodegarealDataDrid – datagrid 4) haya la cantidad suficiente de botas para empacar éste pedido
4) si la cantidad de botas en bodega es mayor o igual a la del pedido, entonces se habilita el botón “este pedido ha sido empacado”
5) una vez que el empacador pone en respectivos cartones el pedido, entonces hace clic en el botón “este pedido ha sido empacado” y el software resta del “bodegaReal” (datagrid 4) la cantidad de botas de este pedido, es decir si en bodega habían 50 botas y este pedido requería 20 botas, entonces al hacer clic en el botón se resta 50-20=30, 30 botas quedarían en bodega.

Pantalla:

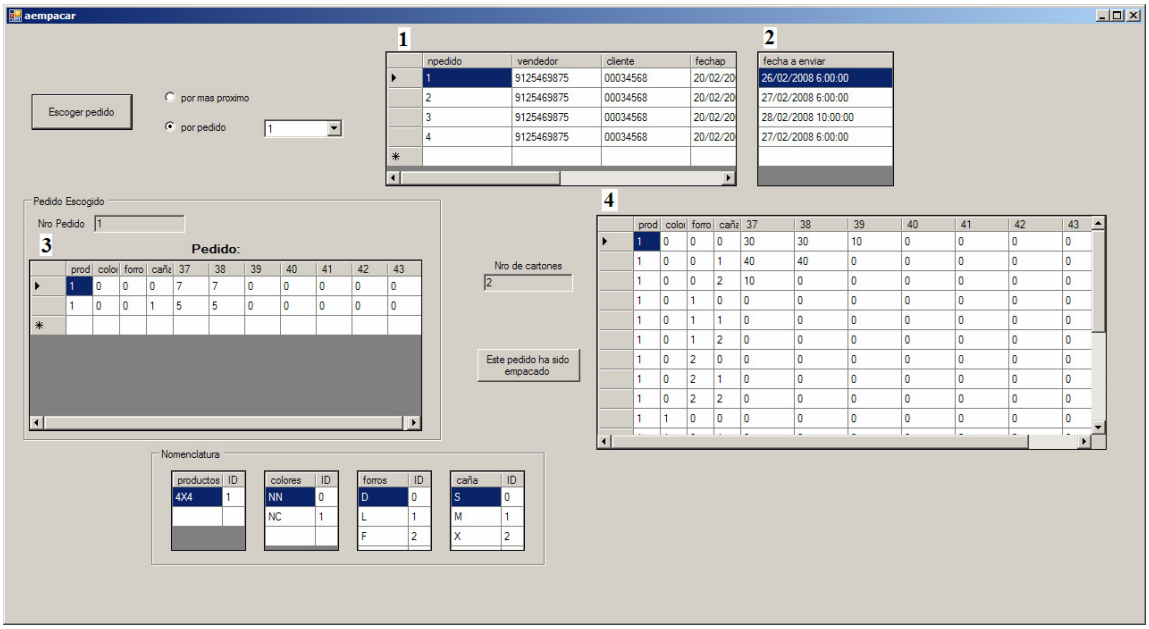


Fig. 4.14. Formulario: aempacar

- En el “ipedidosDataGridView” (datagrid 1) se muestra la tabla dbo.ipedidos, el cual contiene todos los pedidos.

- El “dg_fechas” (datagrid 2) muestra las fechas a las que los pedidos deben salir de bodega para llegar a tiempo al cliente, por ejm. Si un cliente es de Colombia el pedido demora en llegar 5 días, si el pedido lo realizó para el 26 de febrero, el pedido debe salir de bodega el 21 de febrero, ésta fecha es la que se muestra en este datagrid.
- En el “dg_pedido” (datagrid 3) se coloca la cantidad de botas que deben ser empacadas para este pedido
- En el “bodegaRealDataGridView” (datagrid 4) se coloca la cantidad de botas que la fábrica tiene en bodega

Código Fuente:

- Al cargarse el formulario “aempacar” se ejecuta esta función:

```
private void aempacar_Load(object sender, EventArgs e)
```

Dentro de ésta se realiza el proceso de llenar el “dg_fechas” (datagrid 2) con las fechas de los pedidos ya restadas el tiempo de transporte (líneas 80-105).

- Ésta función se ejecuta cuando se hace clic en el botón “escoger pedido”

```
private void b_escoger_Click(object sender, EventArgs e)
```

Al escoger el pedido la función llama a la subfunción `llena_mx_pedido();` para cargar en el “dg_pedido” (datagrid 3) la cantidad de botas que se requiere para este pedido.

Una vez que se cargó el pedido, se procede a verificar si en bodega hay la suficiente cantidad de botas para este pedido, para esto se llama a la subfunción `verifica_haya_nbodega();` (línea 207) si es que sí hay la cantidad de botas suficiente entonces se procede:

1) Calcular el número de pares de botas con la región `#region saca el npares de este pedido` (líneas 209-221)

1) También calcula el número de cajas que tendrá este pedido con la región `#region calcula el ncajas de este pedido` (líneas 223-228)
2) Y se habilita el botón "este pedido ha sido empacado" (b_pempacado)

- Cuando el empacador haya empacado el pedido entonces hace clic en el botón "este pedido ha sido empacado" y se ejecuta la siguiente función:

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

Este botón llama a la subfunción `pedido_completado();` ésta función se encuentra en las líneas 330-540 y hace lo siguiente:

1) mediante la `#region resta de bodega lo q sta en mx_pedido` resta de "bodegaRealDataGridView" la cantidad de botas que se usó para éste pedido
2) mediante la `#region pone en empacadoData` inserta en la tabla "dbo.empacado" una fila con la siguiente información: Nro de pedido, cliente, empacador, Número de pares y número de cartones.
3) La `#region pos=1 en ipedidosData` pone en la columna de "posición" de la tabla dbo.ipedidos el número 1, indicando con esto que éste pedido ha sido empacado.
4) La `#region pone pos=1 en ipedidostData` pone en la columna de "posición" de la tabla dbo.ipedidost el número 1.
5) Muestra un mensaje `MessageBox.Show("pedido completo");`
6) Llama a la función `c_npedidoItems();` para volver a poner en el checkbox "c_npedido" solamente los números de pedidos con posición =0

4.3.2.5. empacado.cs

En este formulario se guarda información acerca de los pedidos ya empacados y listos para despachar.

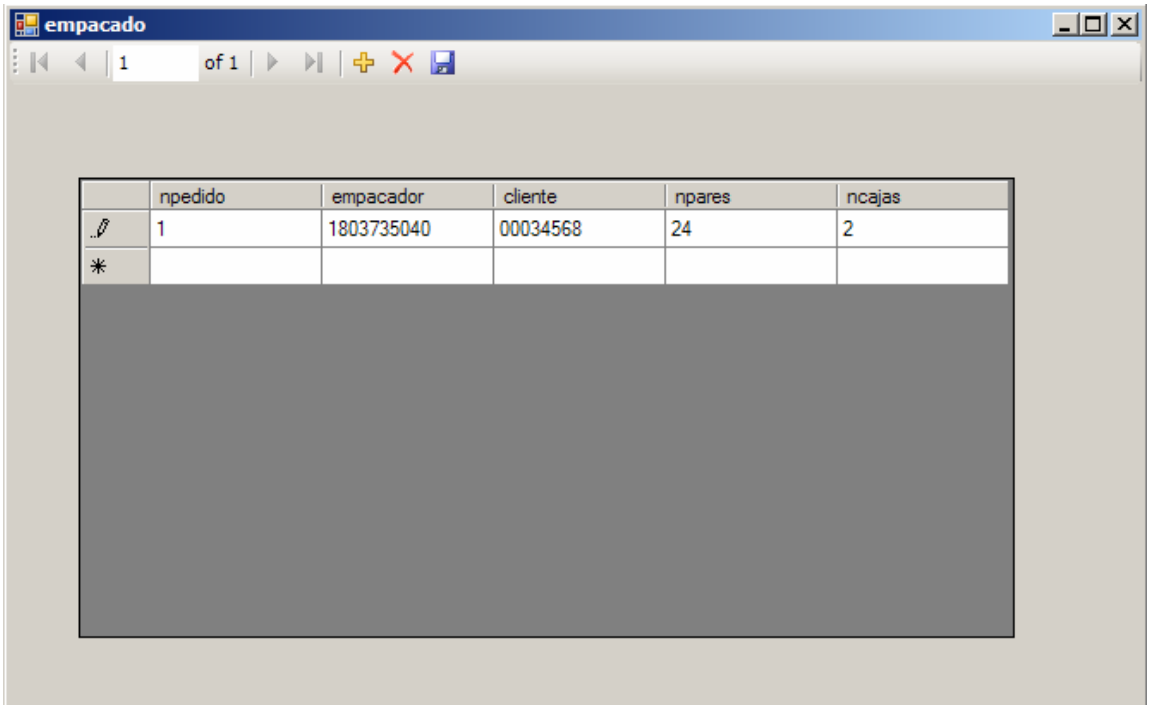


Fig. 4.15. Formulario: empacado

Como se aprecia en la figura 4.15. El pedido Nro. 1 ya está empacado:

- Fue empacado por el empleado con cédula 1803735040
- El RUC del cliente es 00034568
- Número de pares de éste pedido 24
- Y el número de cajas 2

En realidad éste formulario no tiene ningún código fuente, sólo tiene el datagrid "empacadoDataDridView" el cual se observa en la figura 4.15. El cual representa los datos de la tabla dbo.empacado

4.3.2.6. bodega.cs

En el formulario "bodega" básicamente se muestran 2 datagrids:

- el datagrid "bodegaRealDataGridView" de la izquierda inferior, muestra la cantidad de botas que existe en bodega, por ejm.: en bodega hay 50 botas de talla 27 y tipo: color:0, forro:0, caña:1.
- El datagrid "bodegaDataGridView" de la derecha indica los tags que han sido leídos por el lector y quién los ha embodegado, por ejm. El tag 000001000000250A... indica que ha sido empacado por el bodeguero de cédula de identidad nro. 0092549… la fecha 01/03/2008 20:37:34.

Pantalla:

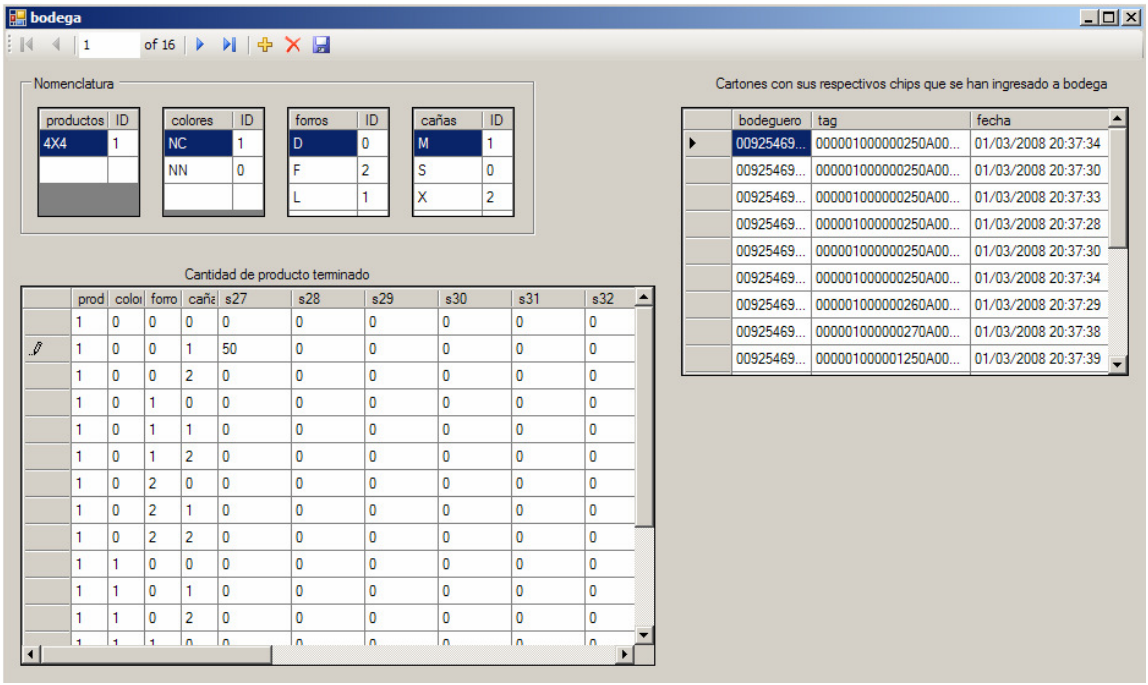


Fig. 4.16. Formulario: bodega

Los datagrids a la izquierda superior indican qué representa cada uno de éstos códigos, por ejm. Cuando dice que el **color** es 0, entonces significa que es el color NN es decir negro-negro.

No hay código fuente en este formulario.

4.3.2.7. imchip.cs

Este formulario se usa para programar el tag con la respectiva información del contenido del cartón al cual va estar adherido, por ejm. Los datos que se muestran en la Figura 4.17 pertenecen a un cartón el cual contiene 10 botas de color: NN (negro-negro), forro: D (media), caña: S (small), Talla: 37.

Una vez que se ha determinado el tipo de producto y la cantidad se hace clic en el botón "imprimir chip y etiqueta" para que el software programe un tag con ésta información e imprima una etiqueta indicativa con los respectivos datos.

Pantalla:

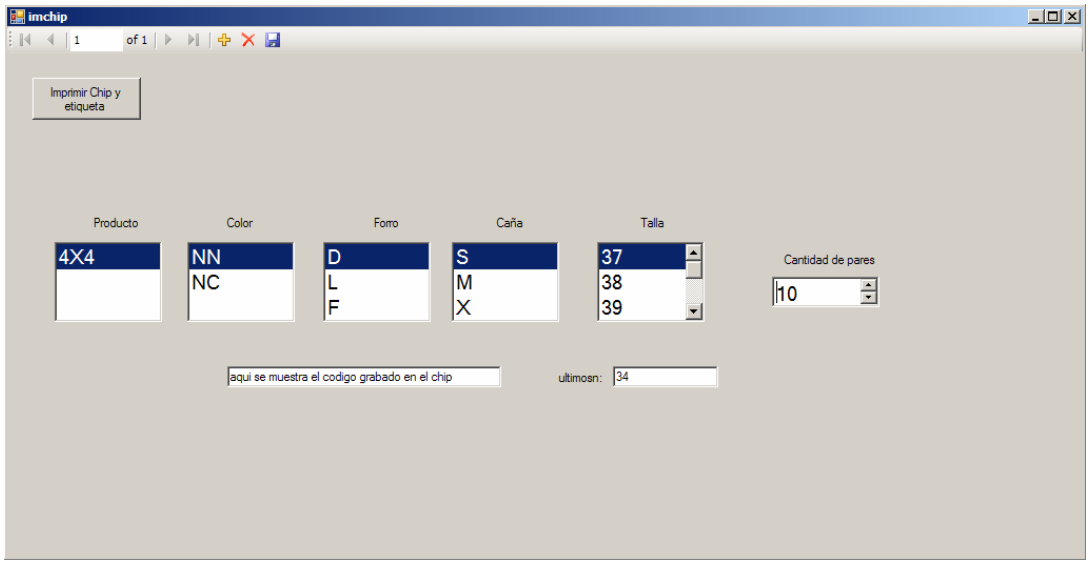


Fig. 4.17. Formulario: imchip

Código fuente:

- Esta función se ejecuta al hacer clic en el botón "imprimir chip y etiqueta":

```
private void b_imprimir_Click(object sender, EventArgs e)
```

Realiza los siguientes procesos:

1) La `#region ve cual es el ultimo serial number y aumentar en 1` (líneas 93-106) guarda en la variable "sn" el número de serie que llevará el tag

2) La `#region guarda en tag el valor segun los datos ingresados` (líneas 112-139) guarda en la variable "tag" el código según el tipo de bota escogida en formato hexadecimal, usando la función `convertir_ahex(int valor)` por ejm.:

nomenclatura	Equivalencia en el software	Equivalencia en hexadecimal para el Tag
Bota 4x4	1	01
Color (negro-negro): NN	0	00
Forro (malla): L	1	01
Caña (large): X	2	02
Talla: 37	37	25
Cantidad: 10	10	0A

Tabla 4.1. Traducción de formato usado por el software a código hexadecimal que lleva el Tag

Entonces la variable tag guardaría el valor: "000001000102250A"
Luego se suma el valor del `tag = tag + sn` que sería igual: "000001000102250A00000001"

3) A continuación la `#region pone los espacios q requiere el ProgramTag para programar` (líneas: 142-154) inserta dentro de la variable tag, espacios (" ") debido a que la función para programar el chip necesita que el valor a grabar esté con espacios cada 2 caracteres, es decir tag quedaría así: "00 00 01 00 01 02 25 0A 00 00 00 01"
4) Y la `#region programa chip` realiza el proceso de programar el chip (líneas 156-194):

Esta clase enciende las antenas 0,1 y 3:

```
tags2.mReader.AntennaSequence = "0,1,3";
```

Esta función guarda en el lector el dato que se va a programar:

```
tags2.mReader.ProgramID = tag;
```

Esta función programa el chip con la información que está dentro de la variable “tag”:

```
status = tags2.mReader.ProgramTag(tag);
```

Si no se puede programar debido un error en el dato “tag” o que ningún tag está dentro del área de cobertura de las antenas, entonces se produce un mensaje `"se ha intentado 10 veces y no puede programar el chip, verifique la conexion"`

Si se programa satisfactoriamente aparece el mensaje: `"el chip se ha programado satisfactoriamente"`

1) Y finalmente en la `#region imprimir papel` se imprime una etiqueta con la siguiente información:

```
producto: 4x4
color: NN
forro: L
caña: X
talla: 37
cantidad: 10
tag:
00 00 01 00 01 02 25 0A 00 00 00 01
Fecha impresión:
03/02/2008 17:40:35
```

4.3.2.8. afabricar.cs

Este formulario en resumen resta lo que hay en bodega de los pedidos y como resultado da la cantidad de botas que hay que fabricar, por ejm.: si en bodega se tiene 1000 botas y hay 1400 botas para entregar en los pedidos el resultado sería que se debe fabricar 400 botas.

Pantalla:

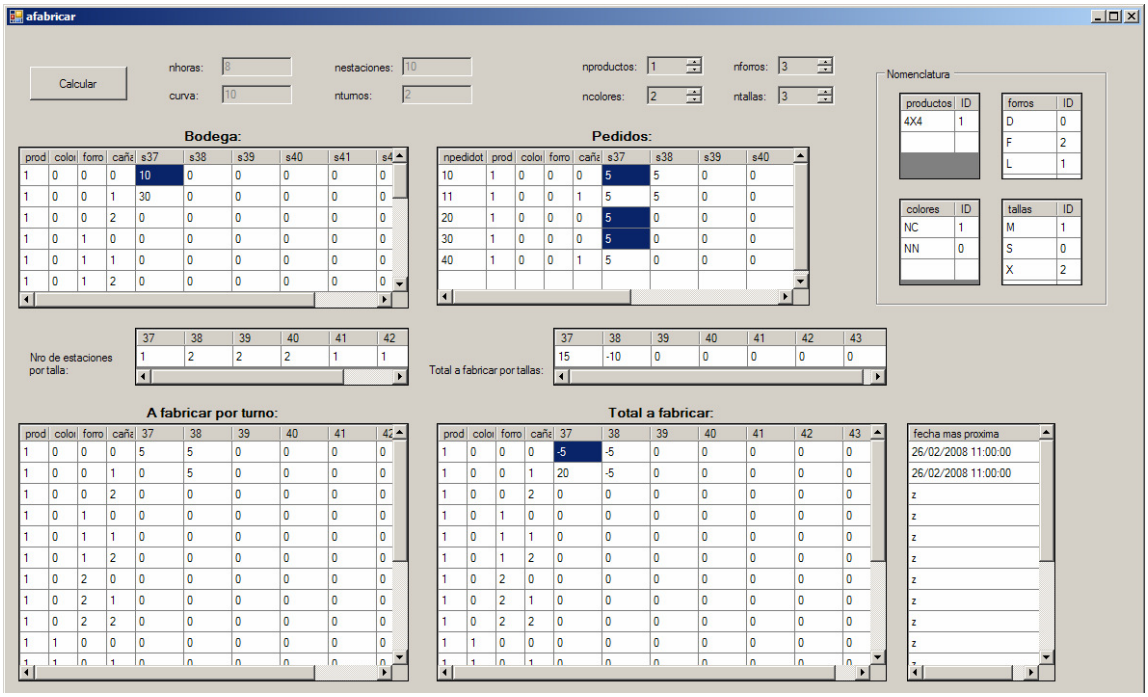


Fig. 4.18. Formulario: afabricar

- El datagrid "bodega" (bodegaRealDataGridView) guarda la cantidad de botas que hay en bodega.
- El datagrid "pedidos" (ipedidostDataGridView) guarda los pedidos.
- El datagrid "Total a fabricar por tallas" (dg_sumatoria) suma el total de botas que hay que fabricar según la talla sin importar el tipo de color, forro o caña que sean, esto sirve para determinar el número de estaciones que se asignará a cada bota, ya que cada estación solo puede fabricar un tipo de bota por turno, debido a que el cambio de molde de una talla a otra en la estación es un proceso demorado.
- El datagrid "Total a fabricar" (dg_diferencia) es el resultado de la resta entre bodega y pedidos, por ejm.: en bodega hay 10 botas color:0, forro:0, caña:0, talla:37(como se aprecia en la celda sombreada del datagrid "bodega" de la fig. 4.18.) y sumando el total de pedidos de este mismo tipo de bota (celdas sombreadas del datagrid "pedidos") da un total de 15 botas, por lo tanto 10-15 = **-5** que es la cantidad de botas que se deben fabricar y esto aparece en el datagrid "Total a fabricar" (en la celda sombreada)
- El datagrid "fecha mas próxima" (dg_diffecha) al lado derecho inferior, se muestra la fecha mas próxima que se encontró en pedidos, para éste tipo de bota, por ejm.:

el pedido nro.1 (fila 0) en el datagrid "pedidos", pedido nro. 2 (fila 2), pedido nro. 3 (fila 3) tienen botas de tipo color:0, forro:0, caña:0, talla:37 cada uno de estos pedidos son para entregar en diferentes fechas, pero la fecha que se coloca en la primera fila del dg_diffecha es la fecha que sea mas próxima de éstos 3 pedidos.

- El datagrid "Nro de estaciones por talla" (dg_estaciones) muestra como debería estar distribuido el número de estaciones por talla; mientras menor sea el valor que se encuentra en una talla del dg_sumatoria mayor estaciones debe estar destinado a esta talla ya que la cantidad de botas a fabricar es mayor, por ejm.: el valor en la talla 37 (del dg_sumatoria) es 15 y el valor para la talla 38 es -10, entonces se debe asignar mas estaciones a la talla 38; esto se demuestra observando el dg_estaciones: para la talla 37 se asignó solo 1 estación mientras para la talla 38 se asignó 2 estaciones.
- En el datagrid "A fabricar por turno" (dg_afabricar) se coloca sólo los valores del dg_diferencia que sean negativos y hasta llegar a la cantidad de botas máxima que un turno puede fabricar, este valor es la multiplicación del textbox "nhoras" x "curva" x "nestaciones", por ejm.
    - Si el "nhoras" (número de horas de la jornada laboral) es 8
    - La "curva" (curva de producción) es 10
    - El "nestaciones" (número de estaciones funcionando) es 10

  Entonces la cantidad de botas que cada turno puede fabricar es 800

Código fuente:

- Ésta función se ejecuta al hacer clic en el botón "Calcular" (b_calcular)

```
private void b_calcular_Click(object sender, EventArgs e)
```

Realiza los siguientes procesos.

Nota: la matriz **mx_bodega**, **mx_afabricar** y **mx_diferencia** son matrices que contienen la misma información que los datagrids **bodegaRealDataDridView**, **dg_afabricar** y **dg_diferencia** respectivamente.

1) la `#region llena mx_bodega <- bodegaRealData, y los tipos de productos en mx_afabricar` (líneas 83-100) llena la matrices mx_bodega con la misma

información de bodegaRealDataDridView, y el dg_afabricar se llena solo las columnas 0-3 es decir solo los tipos de productos.

2) la `#region llena el mx_diferencia y asigna el #de estaciones en el arreglo mx_nestaciones` (líneas 102-268) tiene los siguientes subprocesos:
   a) la `#region llenar_mx_diferencia mx_diferencia=bodegareal-ipedidost` (líneas 105-173) primero llena el mx_diferencia con la misma información de bodega (mx_bodega) y luego va restando todos los pedidos (ipedidostDataGridView) al final se obtiene la cantidad de botas que hay que fabricar; los pedidos que se restan sólo son los pedidos con posición=0. y también va colocando en el mx_diffechas las fechas mas próximas de los pedidos
   b) la `#region visualiza en dg_diferencia <- mx_diferencia` (líneas 175-184) pone en el datagrid "total a fabricar" (dg_diferencia) la misma información de la matriz mx_diferencia.
   c) La `#region visualiza dg_difffecha` (líneas 186-192) coloca en el dg_diffechas la misma información que ya se almacenó en mx_diffechas.
   d) La `#region suma lo que hay en mx_afabricar en el sumatoria` (líneas 194-214) suma en el mx_afabricar todas las botas que están en el mx_diferencia sin hacer diferencia entre qué tipos de botas son.
   e) la `#region asigna el # d staciones una vez q c tnga en un arreglo la sumatoria de los pedidos` (líneas 216-265) coloca en dg_estaciones el número de estaciones por talla, mientras menor sea el número en cierta talla del mx_sumatoria mas estaciones se asigna a esta talla.
3) La `#region saca en el mx_dirfecha el orden de las fechas (de mx_diffecha) desde la + proxima` (líneas 270-306) coloca desde la primera fila las fechas que sean mas próxima del mx_diffecha, esto se hace para saber qué tipo de producto es de mas urgencia para fabricar, es decir que en este datagrid se encuentran fechas en orden de tiempo desde la mas próxima a la mas lejana.
4) La `#region pone en mx_afabricar hasta q llege al limite maximo q s 800` (líneas 308-345) como el nombre mismo lo dice, simplemente coloca en mx_afabricar la misma información del mx_diferencia pero va contando el número de botas hasta que llegue al -800, y solo coloca los números negativos.
5) La `#region visualiza dg_afabricar` (líneas 347-357) simplemente visualiza los datos que se tienen en la matriz mx_afabricar colocándolos en el dg_afabricar.

4.3.2.9. stock.cs

Éste formulario indica la cantidad de materia prima que la fábrica dispone, conforme se agota la materia prima las barras indicadoras (progressBar) disminuyen su longitud y van cambiando su color al tono rojo, una vez que la materia prima llega a ser menor que el mínimo establecido por la empresa, entonces aparece una X al lado del material que se ha agotado. Como se puede observar en la fig. 4.19. el pvcnegro tiene 90,000 pero la cantidad mínima que la empresa debería tener en stock es 100,000 (ver datagrid al costado derecho fig. 4.19.) por lo tanto aparece una X.

Cuando la empresa ha comprado más materia prima debe llenar el textbox "Cantidad" (x_cantidad) con la cantidad que se ha adquirido y hacer clic en el botón "Llenar".

Pantalla:

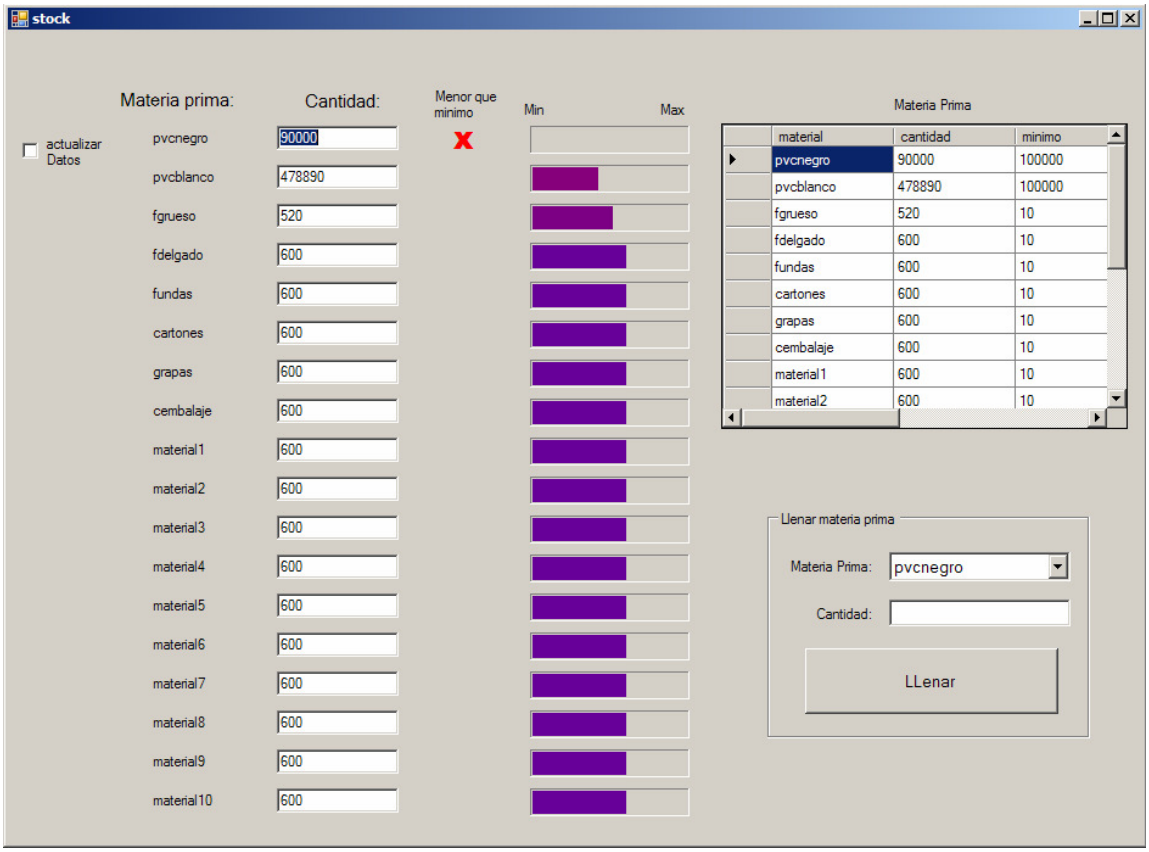


Fig. 4.19. Formulario: stock

Código fuente:

- Ésta función se ejecuta cuando se carga el formulario stock:

```
private void stock_Load(object sender, EventArgs e)
```

Tiene los siguientes procesos:

1) La `#region pone los minimos y maximos` (líneas 31-68) asigna en los progressbar el rango del mínimo y máximo valor que pueden mostrar, por ejm. Si se coloca como máximo 100 y mínimo 10, cuando el valor esté en 100 la barra estará en el máximo  y cuando el valor esté en 10 la barra estará en el mínimo 
2) La `#region pone los nombres` (líneas 71-90) simplemente pone en las etiquetas del formulario (labels) los nombres que aparecen en el datagrid, por ejm. Pvcnegro, pvcblanco, etc.

- El checkbox **actualizar** sirve para actualizar los datos que están en el datagrid y presentarlos en el formulario, al hacer clic se ejecuta la función:

```
private void ch_actualizar_CheckedChanged(object sender, EventArgs
e)
```

Ésta función usa otra función llamada `actualizar();` para actualizar los datos la cual se describe a continuación:

- `private void actualizar()`

1) La `#region pone los valores en textbox` (líneas 104-123) pone las cantidades actuales de materia prima que están en el datagrid (2da columna) y los coloca en los textbox (textBox1,2,3...)
2) La `#region pone los valores en progress bar` (líneas 125-235) pone las mismas cantidades pero en los progressbar.

- Ésta función se ejecuta al hacer clic en el botón “Llenar”:

```
private void b_llenar_Click(object sender, EventArgs e)
```

Esta función suma la cantidad colocada en el textbox “Cantidad” (x_cantidad) a la materia prima que se escogió en el combobox “Materia prima” (c_material)

4.3.2.10. configuracion.cs

En este formulario se encuentran los controles de configuración del sistema, que a continuación se detallan:

Pantalla:

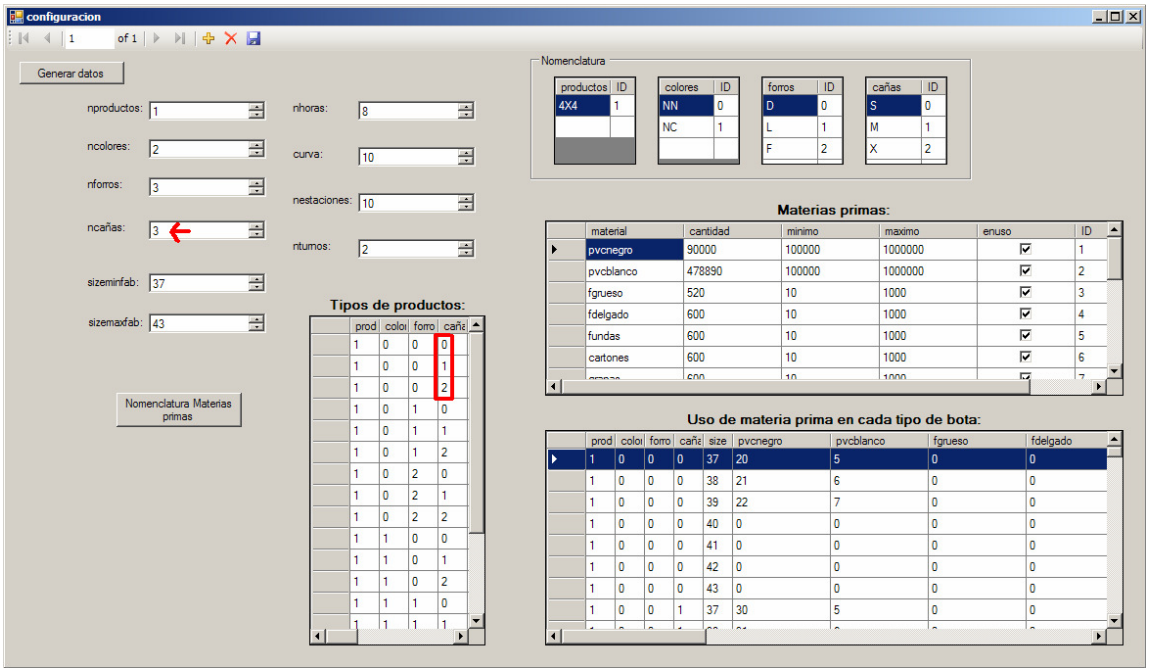


Fig. 4.20. Formulario: configuracion

Este formulario se usa para los siguientes propósitos:

- Determinar el número de:
  - productos (nproductosNumericUpDown)
  - colores (ncoloresNumericUpDown)
  - forros (nforrosNumericUpDown)
  - cañas (ncanasNumericUpDown)
  - menor talla que se fabrica (sizeminfabNumericUpDown)
  - máxima talla que se fabrica (sizemaxfabNumericUpDown)

Que se fabrican, con estos valores se genera la grilla de productos, como se ve en el datagrid "Tipos de productos", por ejm. Como el nro de cañas = 3 (mirar flecha roja fig. 4.20.) el software sólo genera 3 tipos de cañas como se ve en el datagrid, en la columna de cañas.

- Los controles NumericUpDown:
  - "nhoras" representa el número de horas de una jornada laboral
  - "curva" curva de producción, es el número de pares de botas que cada estación puede fabricar por hora
  - "nestaciones" número de estaciones que están en funcionamiento
  - "nturnos" número de jornadas laborables al día.

- En el datagrid "Materias primas" (stockDataGridView) se guarda los mínimos, máximos de cada materia prima y cuales materias primas están en uso.

- En el datagrid "Uso de materia prima de cada tipo de bota" se guarda la cantidad de materia prima que se usa para fabricar cada tipo de bota, por ejm. La bota (que está sombreada fig. 4.20.) de color:0, forro:0, caña:0 y talla: 37 contiene o usa 20gr. de pvcnegro y 5gr. De pvcblanco.

- El botón "Nomenclatura materias primas" abre otra ventana en la cual se asigna para cada numeración la nomenclatura que usa la empresa.

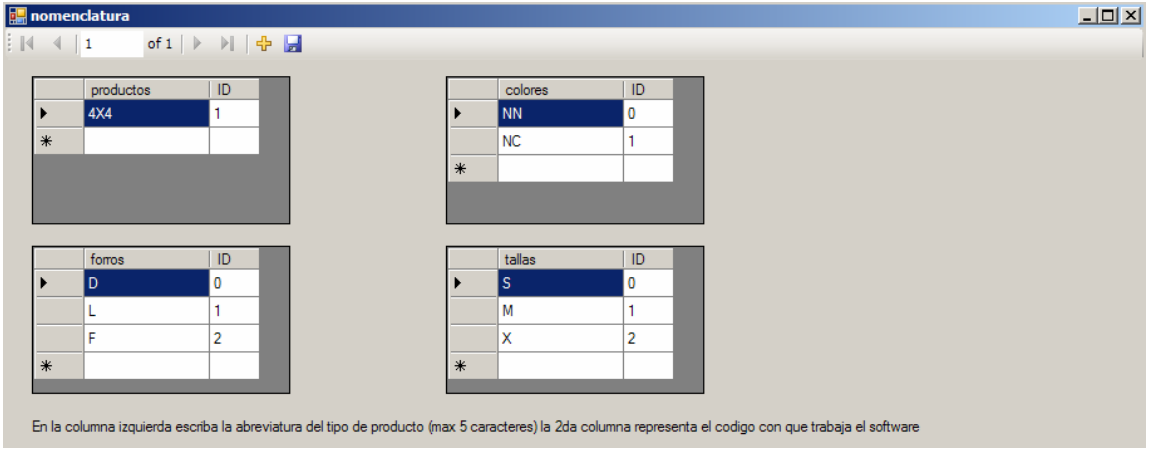


Fig. 4.21. Formulario: nomenclatura

En las columnas de la izquierda (ver fig. 4.21.) se escribe la abreviatura con la que la empresa trabaja para representar a los productos y la columna derecha representa el código con el cual el software trabaja, por Ej. La abreviatura NN que representa una bota de color negro-negro, en el software este tipo de bota se representa por el valor 0.

Código fuente:

- Ésta función se ejecuta cuando se hace clic en el botón "Generar datos" (b_guardar):

```
private void b_guardar_Click(object sender, EventArgs e)
```

Genera el datagrid "Tipos de productos" (fig. 4.20.) con todas las combinaciones de productos que se pueden dar, por ejm. Si el nproductos=1, ncolores=2, nforros=3, ncañas=3 y las tallas que se fabrican son de la 37 a la 43 que equivale a 7 tallas, el número total de productos diferentes que se pueden fabricar sería 1x2x3x3x7 = 126. A continuación se detalla los procesos dentro de ésta función:

1) la `#region umprima` (líneas 72-235) genera los datos de los tipos de productos en el umprimaDataGridView (fig. 4.20.)
   a) la `#region guarda el respaldo en mx_respaldo <- umprimaDataGridView` (líneas 78-92) guarda en un respaldo (en la matriz mx_respaldo) cualquier información que esté en el datagrid umprimaDataGridView.
   b) La `#region pone en mx_tproductos[x,y] los tipos de productos` (líneas 94-134) va poniendo en la matriz mx_tproductos los tipos de productos, por ejm. Ver tabla 4.2.

producto	color	forro	caña	talla	pvcnegro	….
1	0	0	0	37…43	0	
1	0	0	1	37…43		
1	0	0	2	37…43		
1	0	1	0	37…43		
1	0	1	1	37…43		

Tabla 4.2. Matriz mx_tproductos con los primeros 5 datos

Nota: En la tabla 4.2. La columna de talla 37…43 representa que en cada celda hay 7 filas, es decir que en total en la tabla hay 35 filas.

c) La `#region pone el mx_respaldo en mx_tproductos[x,y]` (líneas 136-160) vuelve a colocar el respaldo en la matriz mx_tproductos, por ejm. Si en el respaldo el producto:1,color:0,forro:0,caña:0, talla:37 en la columna de pvcnegro había un 20 entonces aquí también habrá un 20

producto	color	forro	caña	talla	pvcnegro	….
1	0	0	0	37	20	

Tabla 4.3. Primera fila del mx_tproductos

d) La `#region borra todo umprima` (líneas 162-191) borra toda la tabla dbo.umprima

e) La `#region guarda en umprima la nueva info` (líneas 193-192) usa la función `umprimaTableAdapter.Insert(…)` para insertar en la tabla la nueva información.

2) La `#region bodegaReal` (líneas 237-287) hace la misma operación que 1) pero con la tabla dbo.bodegareal.

4.3.2.11. avisos.cs

Avisos, este formulario se despliega cada vez que hay un evento relevante por ejemplo que un bodeguero ingresó botas a la bodega sin identificarse con su tarjeta RFID, o cuando cierta materia prima está a punto de acabarse y debe hacerse un nuevo pedido de ésta. Por ejm. Como se ve en la fig. 4.22 el mensaje está indicando que la materia prima "pvcnegro" está menor que el valor mínimo establecido por la empresa y que hay que comprar esta materia prima.

Pantalla:

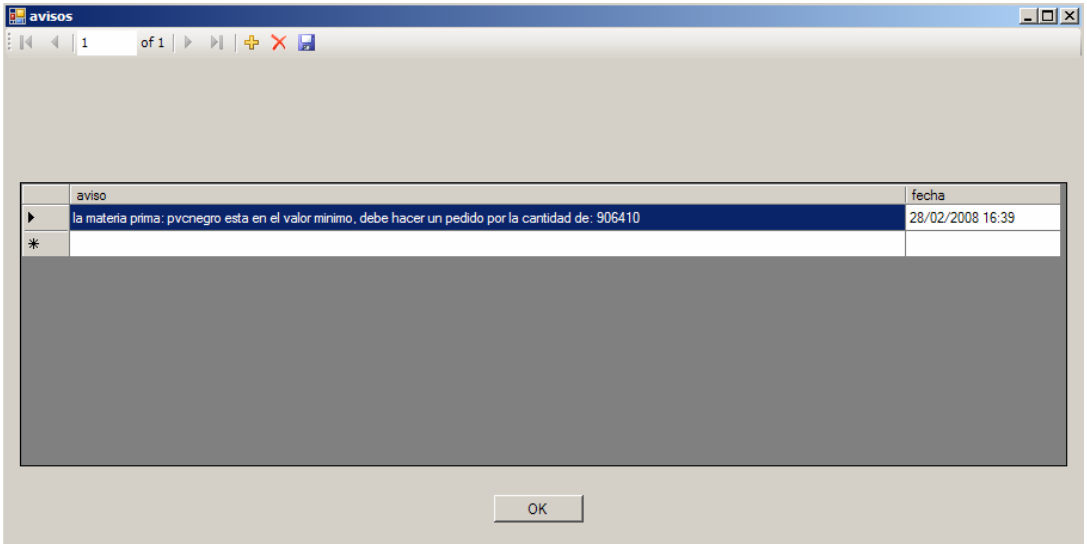


Fig. 4.22. Formulario: avisos

No hay código fuente en este formulario, sino solo en el formulario se muestra la tabla dbo.avisos

4.3.2.12. bdmanual.cs

En este formulario se encuentran las tablas:

- dbo.stock
- dbo.bodega
- dbo.ipedidos
- dbo.ipedidost

Estas tablas se encuentran en forma de datagrids para que la información de cada un de las tablas pueda modificarse de una forma manual.

Pantalla:

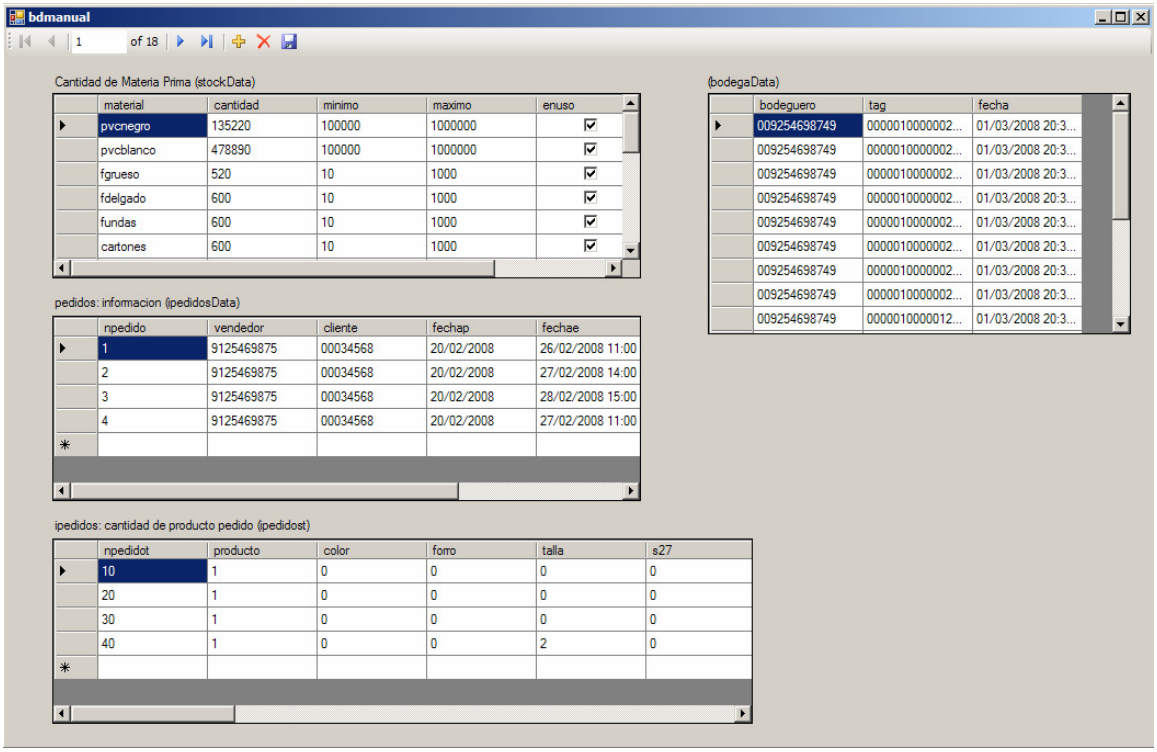


Fig. 4.23. Formulario: bdmanual

No existe código fuente en este formulario.

4.3.2.13. ipersonal.cs

Éste formulario almacena toda la información personal de cada uno de los trabajadores y además el groupbox "carnet inteligente" programa la tarjeta de identificación RFID que llevarán los empleados.

Pantalla:

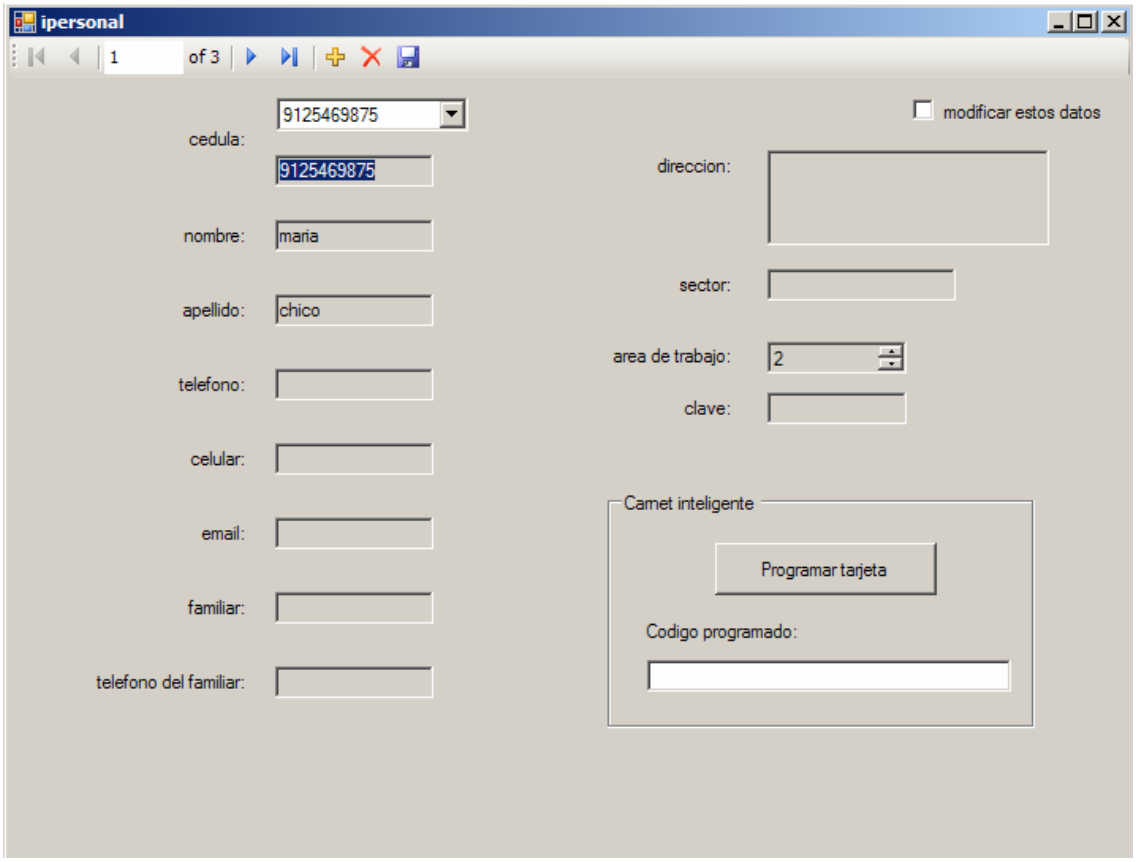


Fig. 4.24. Formulario: ipersonal

Código fuente:

- Ésta función se ejecuta cuando se hace clic en el botón "Programar tarjeta" (b_gchip):

```
private void b_gchip_Click(object sender, EventArgs e)
```

Se encarga de programar en una tarjeta RFID el número de cédula y el área de trabajo del trabajador, tiene los siguientes procesos:

1) La `#region pone en cedula y area los valores de los respectivos textbox` (líneas 41-49) pone en la variable "cedula" el texto que está en el textbox "cedula:" y pone en la variable "area" el área de trabajo del empleado pero convirtiendo el valor del textbox "area de trabajo:" en hexadecimal usando la función `ahex(int valor)`

2) Coloca en la variable “tag” el dato a programar en el chip aumentando los 0s para que tenga los 24 dígitos requeridos para programar el chip. (linea 51)
3) La `#region pone los espacios que requiere el ProgramTag para programar` (líneas 54-66) pone en la variable “tag” espacios ya que la función para programar el chip requiere que el dato esté en este formato para programar, por ejm. “tag” quedaría así: "00 01 0A 18 03 73 50 40 00 00 00” donde: “0001” representa que la información del tag representa a un trabajador, “0A” es el área de trabajo, “18 03 73 50 40” es Nro. de cédula del trabajador y “00 00 00” son de relleno.
4) La `#region programa chip` (líneas 70-92) programa el chip que esté al alcance de las antenas. Nota: debe asegurarse que solo un tag esté en el área ya que si hay mas de un el lector los programará a todos con la misma información.

4.3.2.14. tags2.cs

Al cargarse el programa se abre este formulario, pero se lo mantiene oculto, pero es el que realiza todas las operaciones al momento de leer los tags.

Estas operaciones son:

- Guarda en la tabla “bodega” todos los tags que captó el lector (Fig. 4.25-1)
- Guarda en la tabla “bodegaReal” la cantidad de botas que se ingresaron (traduciendo desde los Tags leídos) (Fig. 4.25-2)
- Descuenta de la tabla “stock” (Fig. 4.25-3) la cantidad de materia prima que se usó en las botas que pasaron, basándose en los datos de la tabla “umprima” (Fig. 4.25-4)
- Si alguna materia prima es inferior a la cantidad mínima que se debe tener (ver Fig. 4.25-3), entonces se guarda un mensaje en la tabla “avisos” (Fig. 4.25-5)

Pantalla:

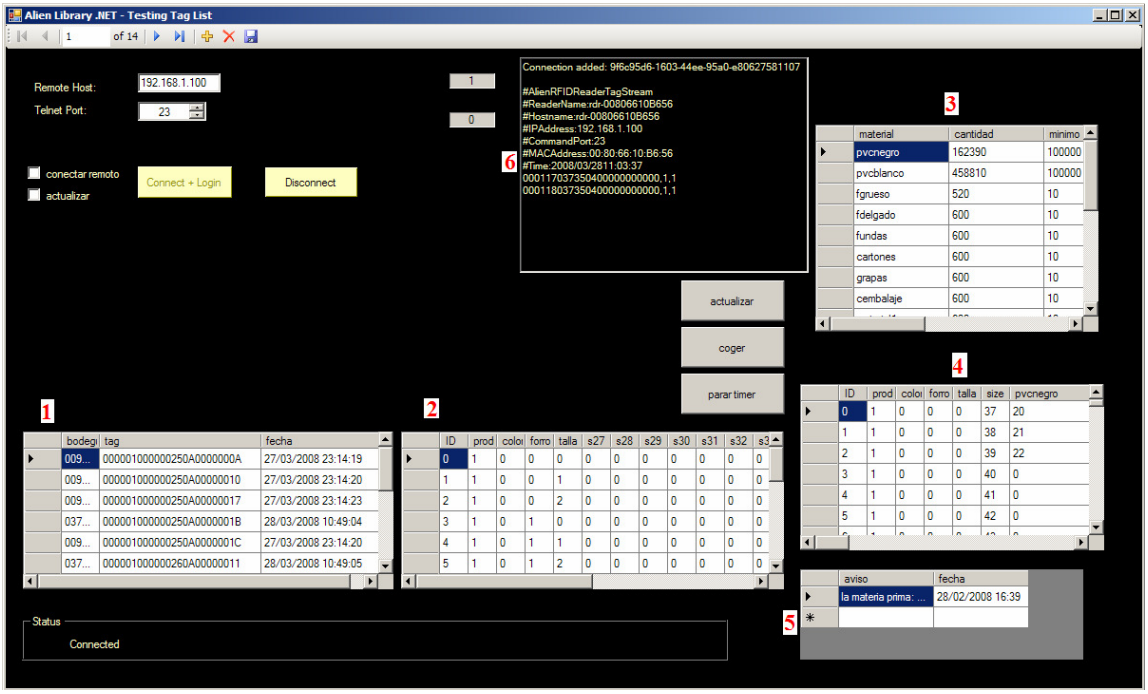


Fig. 4.25. Formulario: tags2

Código fuente:

- Al abrirse esta ventana se ejecuta la siguiente función, la cual trata de conectarse con el lector:

```
public void conectar()
```

Esta función realiza lo siguiente (líneas 99-175):

1) `mReader.InitOnNetwork("192.168.1.100", 23);` inicializa la conexión con el lector de la dirección IP "192.168.1.100" a través del puerto 23.
2) `result = mReader.Connect();` se conecta con el lector, si la operación se completa exitosamente se asigna en la variable "result" la palabra "Connected".
3) `mReader.IsConnected;` devuelve en forma de boleado el estado del lector, si esta conectado devuelve True.
4) `mReader.Login("alien", "password")` accede al lector enviando el nombre de usuario "alien" y la clave "password" si no logra acceder, la función devuelve un False.

5) La `#region configuracion` (líneas 133-167) pone las características que se desea el lector tenga, entre ellas son:
   a) `mReader.KeepNetworkConnectionAlive = true;` //mantiene el lector conectado a la LAN
   b) `mReader.TagListFormat = "Text";` // el formato de la lista de Tags
   c) `mReader.RFLevel = 300;` //el nivel de potencia RF que el lector envía hacia las antenas.

Según la configuración que se realizó, el lector envía todos los eventos en forma de texto al textBox "txtNotifications" como se ve en la Fig. 4.25-6 o mas ampliado en la Fig. 4.26

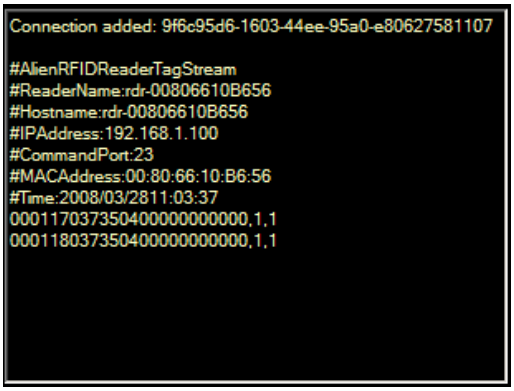


Fig. 4.26. Datos que llegan desde el lector

Estos son los datos que envía el lector hacia la PC, el texto: 00011703735040000…..,1,1 representa un Tag que captó el lector.

- Esta es la función principal, se ejecuta cada vez que llegue información del lector, realiza las siguientes operaciones:

  `factualizar();`

1) Verifica si el tag que se leyó es de tipo producto, es decir que los primeros 4 caracteres del tag debe ser: "0000"
   a) Si es Tag de producto, entonces lo inserta en la tabla "bodega" con la función:

   ```
   bodegaTableAdapter.Insert("bod",tag,
   Convert.ToString(DateTime.Now));
   ```

   b) La `#region inserta en bodegareal, resta de materia prima y verifica minimos` (líneas 206-446):
      i) Inserta en la tabla "bodegaReal" la cantidad de botas equivalentes a los tags que se leyeron (ver Fig. 4.25-2) por ejm. Si se captaron 5 tags, y cada tag indicaba que tenía 10 botas, entonces se guardan en "bodegaReal" 50 botas

ii) Resta de la tabla “stock” la cantidad de materia prima que se usó para fabricar las botas que se guardaron en “bodegaReal”. Cada bota usa cierta cantidad de materia prima dependiendo de su tipo (esta información se guarda en la tabla “umprima” (Fig. 4.25-4)) si todas las 50 botas que se ingresaron fueron: color: Negro-Negro, forro: malla, caña: small y talla: 37, este tipo de bota según la tabla “umprima” usa 20 gr. De PVCnegro, entonces se descontaría de la tabla “stock” (Fig. 4.25-3) 50botas x 20gr = 100 gr. De PVCnegro.

iii) Verifica que las cantidades existentes de materia prima (Fig. 4.25-3-columna:2) sean mayores a los mínimos establecidos por la empresa (Fig. 4.25-3-columna:3), si alguna de ellas son menores, entonces se guarda en la tabla “avisos” un mensaje que se debe comprar mas materia prima ya que está a punto de agotarse.

c) La `#region va contando los items que pasan en principal` (líneas 448-458) coloca en el textbox “x_items” del formulario “principal” (Fig. 4.6) la cantidad de tags que se han leído en los últimos 30 segundos.

2) Si el tag que se leyó pertenece a una credencial, es decir comienza con “0001” como en la Fig. 4.26. entonces la `#region guarda en credencial=tag` (líneas 468-473) guarda en la variable “credencial” el número de cédula de este Tag, por ejm. El Nro. De cédula que contiene el primer tag de la Fig. 4.26 es: 1703735040

3) La `#region pone en bodeguero el chip de la parsona q sta embodegando` (líneas 484-510) pone en la Tabla “bodega” columna “bodeguero” la “credencial” que captó el lector, para así saber qué persona guardó los cartones, por ejm. en la Fig. 4.25-1 el Tag: 000001000000250A0000000A fue guardado en bodega por la persona con Nro. De cédula: 009…

CAPÍTULO 5

ANÁLISIS COSTO-BENEFICIO

5.1. Costo equipos

A continuación se detalla los dispositivos que se usan para el sistema y sus precios.

Descripción	Precio incluye IVA	Precio incluido transporte
Lector ALIEN-9800	2,234.40	2,314.40
4 antenas UHF 900MHz (incluye cable)	100	100
Sensor de movimiento	17.92	17.92
software	1200	1200
Impresora etiquetas EPSON TMU-220	225	225
accesorios	50	50
	TOTAL	**3,602.32**

Tabla 5.1. Costo del proyecto

5.2. Análisis Costo-Beneficio

El costo inicial del proyecto es: ≈ $4000, el gasto mensual por el proyecto:

Etiquetas de papel	7.17
≈ 1400 inlays	173.60
Mantenimiento	10
TOTAL	**$190.77**

Tabla 5.2. Gastos mensuales del nuevo sistema

Datos para calcular el retorno de inversión:

- Costo por hora de trabajo: $1.25
- Número de cartones que se empacan en una jornada de 8 horas: 70
- Precio de cada inlay: $0.124

Los datos se calcularon para una sola jornada laboral de 8 horas al día y 26 días laborables al mes.

Nota: (') significa minutos, ('') significa segundos

En la tabla 5.3. Se aprecia la diferencia de tiempos que un bodeguero usa para poner un cartón con botas recientemente fabricadas, en bodega (proceso 1. 2. y 3. de la Fig. 3.9.)

Para embodegar un cartón:

Con el nuevo sistema		Actualmente	
20''	Programar tag e Imprimir etiqueta	Llenar datos en la etiqueta informativa que llevará cada cartón	1'
0	El sistema lo hace automáticamente	Agregar la botas que se ingresaron, la tabla **bodega**	1'
0	El sistema lo hace automáticamente	Actualizar datos de la materia prima que se usó **stock**	1'30''
20''	TOTAL	TOTAL	3'30''

Tabla 5.3. Diferencia de tiempos que se utiliza para embodegar un cartón

- Entonces el ahorro por cartón embodegado sería 3'30'' menos 0'20'' = **2'10''**, ya que se empacan 70 cartones en cada jornada laboral, el ahorro durante el día sería: 2'10''x70 = 152' = 2h32m, el pago por hora al trabajador es $1.25, entonces el ahorro por mes sería 2h22m x $1.25 x 26 días = **$82.30**

Para empacar 1 cartón de un pedido X:

Con el nuevo sistema		Actualmente	
3'	Poner las botas en los cartones	Poner las botas en los cartones	3'
0	El sistema lo hace automáticamente	Actualizar datos en la PC: restando de **bodega** las botas que	2'

		ya se empacaron y poniendo este pedido en la tabla: **empacado**	
3’	TOTAL	TOTAL	5’

Tabla 5.4. Diferencia de tiempos que se utiliza para empacar 1 cartón de un pedido cualquiera

- Entonces por cada cartón empacado para los pedidos sería de 5’00’’menos 3’00’’ el ahorro es de =**2’00’’**, ya que se empacan 70 cartones en cada jornada laboral, el ahorro durante el día sería: 2’x70 = 140’ = 2h20m, el pago por hora al trabajador es $1.25, entonces sería 2h20m x $1.25 x 26 días = **$75.8**3 de ahorro cada mes, que equivale al 38% del salario mínimo de un trabajador ($200)

- Dado a que todo el sistema actual es manual, se dan errores al momento de transferir la información de las hojas de pedidos a las etiquetas que van en cada cartón, este error es del 0,1% del total de botas producidas, es decir que de cada 1000 botas que se envían a los clientes, se envía 1 bota de mas, por lo tanto es al menos 1 bota al día que se está perdiendo, lo que equivale a **1 bota x $4 del costo de la bota x 20 días = $80** al mes.

- Con el nuevo sistema se tiene un **control mas preciso de la materia prima existente**, por lo tanto no es necesario tener almacenado grandes cantidades de materia prima sino solo la necesaria, esto quiere decir que al menos se ahorraría un 40% del espacio físico utilizado por las materias primas almacenadas y en términos de dinero deberíamos cuantificar según dicho espacio y el costo de arriendo que eso conllevaría el cual podría ser al menos de **$200** mensuales.

- Al no tener un **sistema adecuado que controle la materia prima existente o en stock** a fin de indique cuánto se tiene y cuándo se debe comprar; ocurren paros de producción y/o trabajo de empleados debido a falta de materia prima, haciendo que se pierdan horas-hombre en el proceso de producción cada mes y eso conlleva a una pérdida cuantitativa, que en términos monetarios sería alrededor de **$15** mensuales.

- **Tiempo gastado en cuantificar o en constatar la materia prima, el producto terminado o el empacado**, al menos 5 conteos al mes (cada conteo se efectúa con 1 hora-hombre) entonces serían 5 horas-hombre: **$6.25** mensuales

Por lo expuesto tendríamos el ahorro mensual como a continuación se detalla:

Descripción	Ahorro en términos monetarios
Eficiencia de bodeguero	82.30
Eficiencia de empaquetadores	75.83
Error de envíos	80.00
Espacio en bodega y amortización de materia prima	200.00
Paralización del trabajo de los empleados por falta de materia prima	15.00
Conteo de material	6.25
TOTAL	**$459.38**

Tabla 5.5. Ahorro total que traería el nuevo sistema al mes

5.2.1. Calculo del retorno de inversión en años

Es decir que **$459.38** (ahorro total, tabla 5.5.) - **$242.85** (gasto total por mantenimiento del nuevo sistema, tabla 5.2.) = **$216.53** que será el ingreso neto de la empresa al mes.
Como el costo del proyecto es ≈ $4000 entonces la inversión se retornaría en:

$$\frac{\$4000}{\$216.53\ al\ mes} = 18.5\ meses \approx 1.5\ años$$

5.2.2. Calculo del T.I.R (Tasa Interna de Retorno) y EL V.A.N. (Valor Actual Neto)

Costo inicial del proyecto = $4000
Ingresos netos de cada año = $216.53 X 12 meses = $2600

T.I.R. después de 10 años = 59%[1]

V.A.N. = $ 5856,05

Éstos cálculos sólo tomaron en cuenta datos mesurables, aparte de ellos hay varios beneficios que traería la instalación de este sistema de control, además se hizo el calculo sólo para una jornada de trabajo al día, en la actualidad la empresa trabaja con 2 y hasta 3 jornadas al día, esto equivaldría a que la cantidad de dinero ahorrado sería dos o tres veces mayor.

5.3. Beneficios de RFID

- No se necesita de personal que digite en la PC la cantidad de producto que ingresa o egresa de bodega, este proceso lo hace el lector de tags automáticamente.

- La diferencia de etiquetas RFID y códigos de barras
    - En RFID no se necesita de un operario que maneje un lector, sino que el portal automáticamente lee todos los tags a su alcance, y no necesita línea de vista, es decir que el tag puede estar colocado dentro del cartón y aún así leído sin ningún problema; en cambio el código de barras necesita de un operador que apunte el lector hacia los códigos.
    - En RFID el lector, lee al mismo tiempo todos los tags que estén a su alcance mientras que el código de barras solo puede leer de uno a la vez.
    - Información almacenada en la etiqueta puede ser actualizada a demanda
    - Gran capacidad de almacenaje de información
    - Mayor distancia de lectura
    - Gran precisión en la recuperación de datos (mayor que el código de barras)
    - Soporta condiciones de inclemencia como: suciedad, polvo, humedad, temperatura

- Con RFID se elimina errores humanos como puede suceder con el código de barras (leer un mismo código 2 veces) o con sistemas manuales de adquisición de datos.

[1] Estos cálculos se realizaron con la función de Excel del TIR y VNA

5.4. Beneficios del sistema automático de control de inventarios usando la tecnología RFID

- Inventario exacto y en tiempo real de: producto terminado, materia prima y producto empacado que tiene la empresa.

- Mucho menos dinero amortizado en grandes cantidades de materia prima almacenada, el software automáticamente le indica qué materia prima está por agotarse y debe comprar.

- Se disminuye recursos económicos en contrataciones de personal y se evitan errores humanos ya que toda la información está almacenada en el software:
    - Pedidos
    - Pedidos empacados
    - Materia prima
    - Material terminado
    - Clientes
    - Trabajadores

- Sin papeleos, sin pérdida de información.

- El software determina la cantidad de producto que se debe fabricar basándose en los pedidos y en lo que se tiene en bodega, y automáticamente determina cual es el producto de mas urgencia para fabricar.

- El trabajo del empaquetado será mucho mas eficiente, ya que el software automáticamente le indica qué pedido empaquetar ahorrando recursos.

- Aunque aumente el número de pedidos y el trabajo por hacer, no tendrá que recurrir a contratar mas personal ya que el sistema optimiza el trabajo de los empleados hasta en un 22%

- El precio de los tags seguirá disminuyendo (fuente: www.rfidjournal.com)

- No tendrá que construir una bodega adicional ya que el sistema minimiza el tiempo que el material terminado, material empacado y materia prima se mantienen en la fábrica.

CAPÍTULO 6

CONCLUSIONES Y RECOMENDACIONES

6.1. CONCLUSIONES

- La tecnología RFID ha traído muchas ventajas comparado a los sistemas con códigos de barras y sistemas manuales, ya que recolecta datos automáticamente sin requerir de un operario.

- El sistema fue muy aceptado por todos los trabajadores de la empresa, desde la secretaría hasta los bodegueros ya que el sistema facilita en mucho los procesos de actualización de datos.

- El sistema evita errores humanos de tipificación.

- El sistema es más flexible que los sistemas con códigos de barras ya que el tag es reprogramable.

- El uso de un sistema informático para almacenar información permite al personal recoger datos al instante y en tiempo real.

- El tag Clase 1 Generación 2 915MHz usado en el presente proyecto, resultó ser preciso para esta aplicación ya que tiene un rango de lectura suficiente y además no tiene errores de lectura.

- El costo-beneficio de este sistema trae un buen retorno de inversión para la empresa, tal es así que con la implementación del proyecto la compañía retornaría la inversión en un año y medio.

6.2. RECOMENDACIONES

- Al momento de pasar cartones con tags por el portal debe asegurarse que ningún objeto metálico o líquido se interponga entre el tag y las antenas, ya que estos cuerpos atenúan la señal del tag.

- Al momento de pasar cartones con tags por el portal observe que en el textbox "Ingresando productos a Bodega" aparezca la cantidad de cartones que se pasaron (ver Fig. 4.6.) para verificar que el lector captó todos los tags.

- Asegurarse de no llevar los cartones a mucha velocidad con el montacargas cuando atraviese por el portal.

- Evite el contacto con el chip del tag ya que es la parte más sensible de éste.

- Al momento de programar los tags cerciórese que solo haya 1 Tag al alcance de las antenas (al menos 1m alrededor), ya que si hay mas de uno, el lector programará a todos con exactamente la misma información.

BIBLIOGRAFÍA

http://www.epcglobalsp.org/tech/Readers/FINAL_Reader_Configuration_SPA.pdf, configuración de lectores y antenas RFID

http://yv5apf.com/FVelo.pdf, factores de velocidad de los cables más comunes

http://www.qsl.net/xe3rn/coaxiales.htm, cálculo para la longitud del cable de la antena

http://www.rfidconsultation.eu/docs/ficheiros/smith.pdf, normas y estándares RFID

http://www.supertel.gov.ec/entidades.htm, superintendencia de telecomunicaciones

http://www.epcglobalinc.org/standards/uhfc1g2/uhfc1g2_1_1_0-standard-20071017.pdf, el estándar Clase 1 Gen 2

INDICE DE FIGURAS

INDICE DE TABLAS

GLOSARIO

Base de datos.- Base de Datos es un conjunto exhaustivo no redundante de datos estructurados organizados independientemente de su utilización y su implementación en máquina accesibles en tiempo real y compatibles con usuarios concurrentes con necesidad de información diferente y no predicable en tiempo.

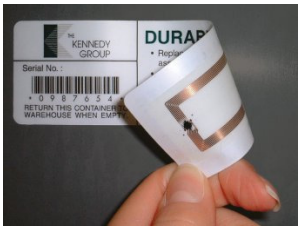
Tabla.- Un arreglo bidimensional compuesto por líneas y columnas. Cada línea representa una entidad que nosotros queremos memorizar en la base de datos. Las características de cada entidad están definidas por las columnas de las relaciones, que se llaman atributos. Entidades con características comunes, es decir descritas por el mismo conjunto de atributos, formarán parte de la misma relación.

Dbo.XXXX.- Representa una tabla de la base de datos la cual tiene el nombre XXXX

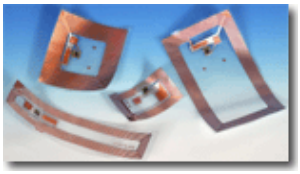
Antena.- Dispositivo capaz de emitir o recibir ondas de radio. Está constituida por un conjunto de conductores diseñados para radiar (transmitir) un campo electromagnético cuando se le aplica una fuerza electromotriz alterna.

RFID.- (Radio Frequency IDentification, en español Identificación por Radio Frecuencia) es una nueva tecnología para la identificación de objetos a distancia sin necesidad de contacto, ni siquiera visual. Se requiere lo que se conoce como etiqueta o tag RFID que consiste en un microchip que va adjunto a una antena de radio y que va a servir para identificar al elemento portador de la etiqueta.

Etiqueta Inteligente (Tag).- Elemento que tiene embebido el chip electrónico que contiene la información necesaria para identificar al producto. Esta información se transmite al exterior mediante una antena que también va incorporada en la etiqueta. Consta de un inlay (o chip) adherido a una etiqueta de papel que puede ser impresa.



Inlay.- Comprende el chip RFID, la antena (de aluminio, cobre o plata) y éstos van adheridos al (PET) capa de polietileno que sostiene al chip y la antena. Dos tipos de inlay "dry" sin adhesivo o "wet" que tiene una capa de adhesivo.



Lector (Reader).- Equipo que genera la energía necesaria para el funcionamiento del chip embebido en la etiqueta y que es capaz de recibir y entender (decodificar) la información contenida en la misma y también de escribir información sobre la etiqueta.

XXXX.cs.- Nombre de un formulario o ventana de un software desarrollado en C# .Net

#region XXXX.- Porción de código el cual tiene una etiqueta de nombre XXXX

Código Fuente.- Conjunto de líneas que conforman un bloque de texto, escrito según las reglas sintácticas de algún lenguaje de programación destinado a ser legible por humanos.

Formulario (ventana).- Donde se encuentra el diseño del formulario, aquí se hallan los botones, textbox, checkbox, etc. Que usa esta ventana.

Función.- Porción de código que tiene un nombre determinado y realiza una labor concreta.

Fecha de entrega:

Sangolquí a 29 de Mayo del 2008

_____

Ing. Víctor Proaño

Director de la carrera ingeniería en electrónica, automatización y control

_____

Javier Lara

C.I.: 1803735040

Autor