Software de control de un lector RFID UHF

PROYECTO DE FIN DE GRADO

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Escola Superior d'Enginyeries Industrial, Aeroespacial i Audiovisual de Terrassa

Grado en electrónica industrial y automatización

Autor: Marc Peña Pérez

Director: Raúl Fernández García

Junio 2019

Resumen:

El uso de la tecnología RFID cada vez está siendo más utilizada, y no solamente en un campo en concreto, ya que la variedad de industrias que utilizan este sistema es muy variada, pasando desde la medicina, farmacia y en definitiva el campo de la salud, hasta las empresas dedicadas a la gestión de grandes stocks o paquetería.

Este proyecto, describe el desarrollo de un software de control de tarjetas RFID UHF capaces de proporcionar datos desde un sensor incorporado en dichas tarjetas. En la primera parte de esta memoria, explicaremos el funcionamiento básico de la tecnología RFID, explicando el porqué es tan importante la frecuencia que utilizamos, ya que esta definirá el tipo de tarjeta y los estándares a los que está sujeta. Posteriormente, hablaremos de los protocolos de comunicación de dichas tarjetas con los lectores RFID, y los aplicaremos para el desarrollo de un software capaz de extraer la información que queremos. Mostrando la aplicación y el funcionamiento de esta misma.

Índice

Índice de figuras:

Motivaciones y objetivos:

En esta sociedad que vive en una era donde las tecnologías de la información son un pilar fundamental, donde conseguir información de manera rápida y efectiva, donde todo el mundo lleva encima dispositivos que se pueden comunicar mediante tecnologías de distancia, es básicamente una premisa que rige la mayoría de grandes marcas en todos los sectores.

Es en este entorno, donde la tecnología de identificación por radiofrecuencia (RFID), sirve para identificar diferentes objetos de forma automática, dando una identidad a cada una de estas “tags” y simplemente siendo adherida al objeto a identificar.

Los sistemas RFID han sido utilizados ampliamente por una gran parte de la sociedad, hecho que ha proporcionado un crecimiento de esta tecnología llevándola a un nivel superior donde ahora estas tarjetas que no necesitan batería ahora pueden llevar un sensor incorporado, a parte de augmentar su distancia de lectura eficaz.

El objetivo de este trabajo, es el desarrollo de un software que sea capaz de leer las diferentes tarjetas de UHF (ultra high frequency), i obtener los datos de su sensor. Para este proyecto no contábamos con una tarjeta con sensor incorporado, pero si con una que, aunque en este momento no lo lleve si que lo puede incorporar y por lo tanto tiene el espacio de memoria reservado para su utilización.

Mis motivaciones para la realización de este trabajo son el aprendizaje de métodos de las tecnologías de comunicación, en concreto RFID, y sus aplicaciones a nivel industrial que pueden llegar a tener en un futuro próximo.

Uno de los aspectos a tener en cuenta para la realización de una aplicación informática, es la usabilidad de esta, por lo que tendrá que ser un software sencillo de utilizar e intuitivo.

Sistemas RFID

Identificación por radiofrecuencia.

El RFID o identificación por radiofrecuencia, proviene de las siglas en inglés Radio Frecuency Identification. Esta tecnología se basa en sistemas de identificación de etiquetas si la necesidad de ningún tipo de cableado. Estas etiquetas se pueden situar en objetos, animales, personas o cualquier tipo de material. Esta tecnología permite la comunicación de dichas tarjetas con un ordenador mediante un lector RFID, el cual se comunica con las tarjetas mediante ondas electromagnéticas que llevan tanto la información como la energía necesaria para que estas tarjetas puedan funcionar.

Las tarjetas RFID tienen un código diferente cada una de ellas, y luego otro código que puede ser introducido por el usuario, de manera que puedas identificar con precisión aquello a lo que se haya enganchado una tarjeta. Esta tecnología, al estar adherida a un ordenador puede llevar a la automatización de los procesos de identificación.

El funcionamiento básico de estos sistemas se basa en ondas enviadas por el lector que estimula la antena de las tarjetas RFID, estas, reaccionan a las ondas y emiten una respuesta que el lector puede detectar y enviar al ordenador.

Sistemas RFID

Los sistemas RFID se encuentran dentro de los sistemas de radiofrecuencia como su nombre indica, es por este motivo, que las bandas de frecuencia para su funcionamiento están delimitadas para evitar interferir con otros sistemas como podrían ser la radio, la televisión, los teléfonos móviles.

Según el rango en el que operan estas tarjetas la clasificación es la siguiente:

- LW, Low frecuency: 30-300KHz
- HF, High frecuency: 3-30MHz
- UHF, Ultra high frecuency: 300MHz-3GHz

Cada una de ellas regulada por una normativa en concreta para su comunicación con los lectores.

Como explicaremos más adelante la energía que contienen las ondas de radiofrecuencia viene basada por la frecuencia de esta misma.

Low Frecuency

Al ser las tarjetas que funcionan con un rango de frecuencias más bajo, su rango de funcionamiento es también el más bajo, de unos 10 cm para su lectura efectiva.

Las aplicaciones en las cuales se usa esta tecnología, son aquellas que no necesitan una gran distancia para su lectura. Basada en esta tecnología se ha desarrollado la tecnología NFC, utilizada en los teléfonos móviles i las tarjetas de crédito para pagar sin contacto.

Estas tarjetas suelen ser tarjetas pasivas. (ver el apartado: tarjetas)

High Frecuency

Esta es la primera evolución de la tecnología RFID, al ver augmentada la frecuencia de funcionamiento, augmentamos el rango efectivo de lectura hasta 1m de distancia entre tarjeta i lector.

Esta tecnología, es más sensible a perturbaciones que su predecesora, aunque por el contrario la velocidad de transmisión de datos augmenta, por lo que se pueden enviar o recibir más mensajes por unidad de tiempo o mensajes más largos.

Ultra High Frecuency

Esta es la tecnología más nueva en este campo, en este caso el rango de lectura variara en función del tipo de tarjeta que estemos utilizando, el rango de lectura dependiendo del tipo puede llegar hasta los 200m de distancia. (ver apartado: tarjetas)

La regulación de este tipo de tarjetas viene dada por la norma EPCGlobal Gen 2.

Los rangos de aplicación de este tipo de tarjetas son muy variados, gestión de inventarios, control de productos farmacéuticos, estudios médicos, gestión de transporte e incluso el deporte de alto rendimiento.

Actualmente las tecnologías LF y HF están quedando en desuso por los beneficios de las UHF, las grandes distancias y las longitudes de los mensajes y a la vez porqué estas tarjetas pueden llevar incorporados sensores.

Elementos de los sistemas RFID

Los sistemas RFID constan de pocos elementos, básicamente se pueden dividir en:

- Tarjetas, etiquetas(tags).
- Lectores
- Controladores.

Tarjetas o Tags

Las tarjetas tienen la función de guardar el identificador de la tarjeta y la información con la que esta haya sido configurada. Para esto su memoria se divide en diferentes apartados. (ver apartado: Regulación)

Los tags de constituyen de los siguientes elementos:

- **Antena:** Es la encargada tanto de recibir y enviar la información como de alimentar el circuito de energía. A mayor tamaño de antena mayor rango de lectura. Dependiendo de la aplicación se utilizan un tipo de antena u otros.
- **Circuito integrado o microchip:** Es el encargado de procesar la señal i generar la respuesta para que puede ser leída por el lector.

Los chips utilizados en la tecnología UHF disponen de variedad de memorias dependiendo del modelo de chip y de la función que este desarrollará

- **Solo Lectura:** Solo contiene la información de identificación de la tarjeta. No puede ser modificada una vez fabricada la tarjeta.
- **Lectura y Escritura:** En este caso diferenciaremos dos tipos.
  - **WORM, o write one read multiple**, escribe una vez lee varias.
  - **MTP, o Multi-Time Programmable,** se puede programar varias veces con un lector accediendo a la memoria destinada para ello.

La clasificación según su autonomía se basa en tres grandes grupos:

- **Pasivas:** Son tarjetas que no disponen de ningún tipo de alimentación eléctrica más que su propia antena, esta recoge la energía de la señal induciendo una pequeña carga eléctrica en el circuito que le permita funcionar y generar una lectura de la señal y una respuesta. Por este motivo la distancia máxima de lectura será de 12 m. Al no ser necesaria batería, estos tags pueden ser fabricadas en una gran variedad de materiales, reduciendo su coste y permitiéndoles una

adaptabilidad al producto que se escoja para llevar este tag muy considerable.

Fig. 1

- **Semi pasivas:** Son tarjetas que llevan adherida una pequeña batería, esta batería, no es necesaria para el correcto funcionamiento del tag, simplemente augmenta el rango de lectura eficaz de nuestro dispositivo. Estas tarjetas simplemente entraran en funcionamiento cuando un lector requiera su información.
  Al ser simplemente un soporte para augmentar el rango de lectura, estas baterías pueden llegar a durar mucho tiempo ya que solo se activarán cuando sean necesarias.
  El rango de lectura eficaz augmenta hasta los 100m.

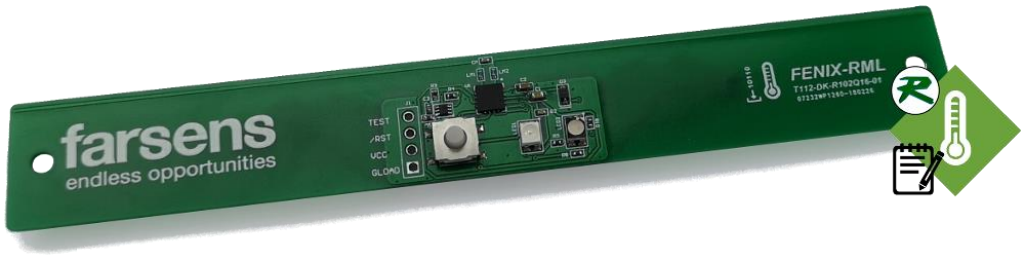


Fig. 2

- **Activas:** Estas son tags con batería incorporada, lo que les permite mayor autonomía y rango de lectura eficaz. A diferencia de la anterior, estos tags emiten de manera periódica una señal, sin ningún requerimiento de un lector para ello.

  Gracias a que esta batería alimenta tanto a la antena como al chip o circuito integrado su rango de lectura augmenta hasta los 200m y es más robusta frente a perturbaciones exteriores, lo que las hace ideales para trabajar a grandes distancias y en entornos más complicados.

A causa de su complejidad, estas tarjetas son las que tienen un mayor coste económico. Y si vida útil disminuye.

Fig. 3

Lector

El lector o también conocido como interrogador, es el dispositivo encargado de la comunicación entre el software y el tag. Los lectores se comunican para obtener la información del tag y en su defecto escribir la información que queramos siempre que sea posible.
El Lector consta de unas antenas que convierten la corriente eléctrica en ondas electromagnéticas con las que se comunica con los tags.

Las antenas más frecuentes son las polarizadas lineales i circulares. Dependiendo de la distancia a la que queramos llegar elegiremos una antena u otra.

Fig. 4

Funcionamiento de los sistemas RFID

Los sistemas de identificación por radiofrecuencia, constan de tres elementos básicos como hemos comentado previamente.

- Tag
- Lector
- Controlador

Cuando un objeto con tag RFID entra en el rango de lectura de un lector, y este esta emitiendo señal de forma continua para saber si hay algún tag disponible, el tag empieza a responder a la señal ofrecida por el lector. Una vez que el tag ha emitido una respuesta, el lector deriva esta información, por otro canal, sea mediante cable o no, a un controlador.

Dicho controlador, suele ser un ordenador con una base de datos y un programa especifico para conectar con el lector.

El ciclo del sistema es el que explicamos a continuación.

1- El ordenador pregunta al lector si encuentra algún tag disponible.
2- El lector lanza una señal interrogadora al aire
3- El tag, si lo hay, envía su identificación como respuesta.
4- El lector recibe la respuesta y la manda al controlador
5- El ordenador recibe los datos y los procesa.

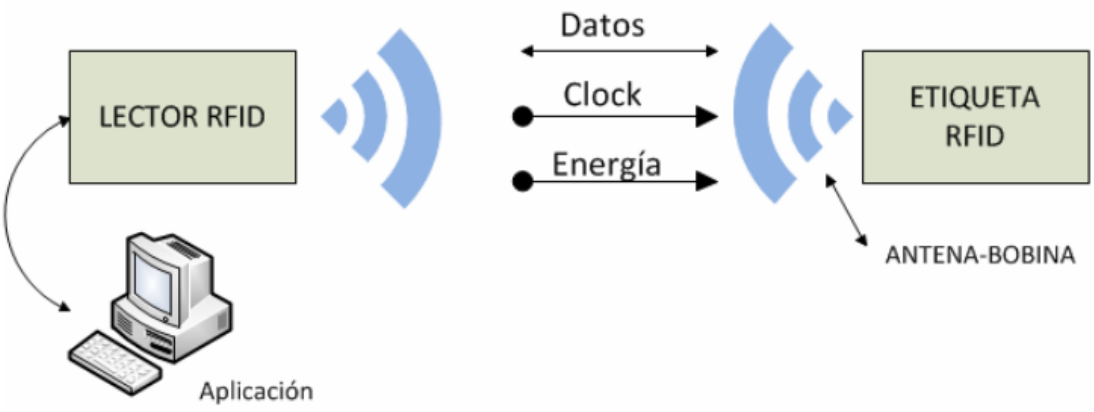


Fig. 5

Principió físico de funcionamiento

Los principios de funcionamiento de los tags RFID se basan en la física de campos electromagnéticos y la física de campos electromagnéticos. Mediante estos dos campos de la física podemos explicar el porqué de la transmisión de energía sin necesidad de ningún tipo de cableado y la distancia a la que podemos enviar o recibir esta señal.

En telecomunicaciones, se ordenan las ondas mediante un convenio internacional de frecuencias en función del objetivo a desarrollar por las ondas electromagnética, yendo estas desde los rayos gamma hasta las ondas de radio.

Clasificación de las ondas en telecomunicaciones

Sigla	Rango	Denominación	Empleo
VLF	10 kHz a 30 kHz	Muy baja frecuencia	Radio gran alcance
LF	30 kHz a 300 kHz	Baja frecuencia	Radio, navegación
MF	300 kHz a 3 MHz	Frecuencia media	Radio de onda media
HF	3 MHz a 30 MHz	Alta frecuencia	Radio de onda corta
VHF	30 MHz a 300 MHz	Muy alta frecuencia	TV, radio
UHF	300 MHz a 3 GHz	Ultra alta frecuencia	TV, radar, telefonía móvil, WLAN[2]
SHF	3 GHz a 30 GHz	Super alta frecuencia	Radar
EHF	30 GHz a 300 GHz	Extremadamente alta frecuencia	Radar

Fig.6

La radiación electromagnética que provee de energía nuestros tags, es una combinación de campos eléctricos y campos magnéticos que varían en función del tiempo.

Cuando un elemento de nuestro sistema, como es la antena, conduce una señal no continua, se genera una radiación electromagnética modulada en la misma frecuencia.

Esto provoca que, al modular la señal en la antena de nuestro sistema, modulamos a la misma vez la señal que está recibiendo nuestro tag.

La explicación teórica de la radiación electromagnética la podemos encontrar en dos puntos básicos.

- Las ecuaciones de Maxwell: Son un conjunto de cuatro leyes de la física que forma un conjunto, dichas leyes son, la ley de Gauss para el

campo eléctrico, la Ley de Gauss para el campo magnético, la ley de Faraday-Lenz y la ley de Ampere.

En dichas teorías encontramos la explicación al comportamiento de los campos generados por nuestro lector y en como estos se generan.

La explicación a la transmisión de la energía la encontramos en el principio de:

- Dualidad onda-corpúsculo: Este es un fenómeno cuántico que define que muchas partículas pueden exhibir comportamientos de ondas. Lo que lleva a la conclusión de que no hay diferencias entre partículas y ondas.

Es por este motivo, que, en el caso de la radiación electromagnética, las señales emitidas pueden considerarse como un flujo de fotones en vez de como simples ondas. Estos fotones, tienen una energía directamente proporcional a la frecuencia de la señal emitida. Esta relación la encontraos en la relación de Plank:

$$E = h\nu$$

Donde:

- h es la consante de Plank.
- V es la frecuencia de la señal
- E es la energía de los fotones.

Si de la misma manera tomamos en consideración la señal emitida como una onda:

$$c = \lambda\nu$$

Donde:

- c es la consante de la velocidad de la luz
- λ es la longitud de la onda
- v es la frecuencia de la onda.

De esta manera podemos determinar que a mayor longitud de onda, obtenemos una menor frecuencia y una menor energía.

Estándares y regulación

Ya que nuestro software tratará el caso de los tags UHF, nos centraremos en explicar la regulación a la que son sometidos dichos tags.

Estos estándares, afectan a la comunicación de los tags con los lectores, definiendo los archivos de memoria, y los protocolos de comunicación y de acceso a estos por parte del usuario.

La gestión de los estándares RFID UHF, recaen sobre diferentes estamentos:

- **ISO:** International Organization for Standarization.
- **IEC:** International Electotechnical Commosion.
- **ETSI:** European Telecomunications Standards Institute.
- **ITU:** International Communication Union.
- **FCC:** Federal Communication Comission
- **EPC GLobal**

Los sistemas RFID están ahora mismo bajo los efectos de la norma ISO-1800-6C, también conocido por EPC Class 1 Generation 2 o EPC Gen2. La Class, se refiere a que función desarrolla el tag, y la Generation se refiere a los aspectos físicos y lógicos de los tags.

Es conocida por el nombre de EPC, que és una empresa privada, porqué fue esta empressa quien propuso a ISO esta normativa, que finalmente fue aceptada y aprobada en 2006.

Anteriormente, se utilizaba para los tags UHF la ISO-1800-6b. Que ha quedado en desuso por la nueva normativa.

Clasificación tags EPC Gen2

Siguiendo con la norma las tarjetas se clasifican de class 0 a class 5.

- **Class 0:** Tag UHF pasivo de solo lectura.

- **Class1:** Tag UHF o HF WORM.

- **Class2:** Tag pasivo de lectura y escritura.

- **Class3:** Tag con batería y sensor.

- **Class4:** Tag activo de lectura y escritura capaz de comunicarse con otros tags.

- **Class5:** Class4 con capacidad de emitir energía a otros tags.

Memoria Tags Gen2

Según lo dispuesto en la norma, las tarjetas han de tener cuatro reservas de memoria diferentes.

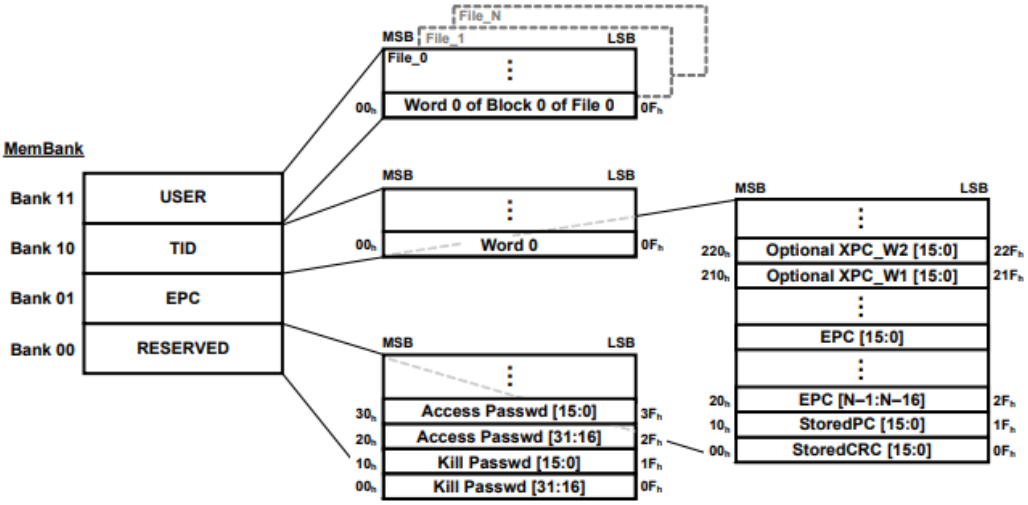


Fig. 7

1- **User:** Es la parte de memoria accesible para el usuario, si escribimos alguna información en un tag será en este apartado de memoria.
2- **TID:** Parte de información correspondiente al tag, no provee ninguna información del objeto al que va adherido. En este apartado encontraremos el fabricante, que tipo de tag es y que protocolos de comunicación utiliza.
3- **EPC:** Este es el apartado donde encontraremos la información del objeto adherido a nuestro tag. Dependiendo de la potencia del chip o del circuito integrado tendrá una longitud u otra, normalmente 96 bits. En este apartado encontraremos otra información de control como el CRC (Cyclic Redundancy Check), XPC (Extended Protocol Control) o el PC (Protocol Control).
4- **Reserved:** Contiene las contraseñas de habilitación e inhabilitación del tag, en caso de que no sean programadas, tendrán el valor por defecto.

Protocolo de comunicación

La comunicación entre el lector y el tag, está parametrizado en todo momento, incluso desde el apartado físico propio de los tags. En este apartado haremos un resumen de los protocolos que afecten a los diferentes dispositivos de un sistema RFID.

GS1 EPCglobal™: GS1 EPC™ Tag Data Standard

ISO/IEC 15961: Information technology — Radio frequency identification (RFID) for item management — Data protocol: application interface

ISO/IEC 15962: Information technology — Radio frequency identification (RFID) for item management — Data protocol: data encoding rules and logical memory functions

ISO/IEC 15963: Information technology — Radio frequency identification for item management — Unique identification for RF tags

ISO/IEC 18000-1: Information technology — Radio frequency identification for item management — Part 1: Reference architecture and definition of parameters to be standardized

ISO/IEC 18000-63: Information technology automatic identification and data capture techniques — Radio frequency identification for item management air interface — Part 63: Parameters for air interface communications at 860–960 MHz

ISO/IEC 19762: Information technology AIDC techniques – Harmonized vocabulary – Part 3: radio-frequency identification (RFID)

ISO/IEC 29167-1: Information technology — Automatic identification and data capture techniques — Part 1: Security services for RFID air interfaces

Distancia de lectura

Las distancias de lectura efectiva de los tags no son siempre estables, ya que este factor depende de muchos otros:

- Tipo de tag
- Potencia del lector
- Sensibilidad Tag
- Antena
- Frecuencia
- Orientación relativa del tag y del lector
- Entorno

Aun utilizando el lector de mayor potencia y un tag activo, es posible que, por la orientación de la antena, aun pudiendo leer el tag hasta 200m, a 20m tenga problemas para detectarlo siempre.

Es por este motivo que cuando queremos saber el alcance de nuestro sistema debemos tener en cuenta tanto los propios del sistema como tipología de tag y potencia del lector, como también del entorno en el que nos encontramos.

Entornos con grandes máquinas electrónicas que generen ruido electrónico a las frecuencias a las que estamos trabajando puede hacer nuestros tags invisibles.

Es importante sobre el entorno si hay elementos físicos entre el tag y el lector que puedan apantallar las señales generadas por estos imposibilitando la comunicación.

Por este motivo hoy en día cuando se hace uso de esta tecnología, se utilizan varios lectores en diferentes posiciones para una mejor recepción de las señales.

Aplicaciones hoy en día

Aun siendo una tecnología muy joven aun en desarrollo, debido a los bajos costes de la instalación de sus sistemas, como a la versatilidad de usos que se le puede dar, esta tecnología está presente en muchos lugares de nuestro entorno.

Como podemos observar en la web fe Farsens (proveedores del tag con el que hemos trabajado) podemos ver que los rangos de aplicación son muy variados.

- Monitorización de plantaciones agrícolas en invernaderos
- Monitorización de temperatura en productos perecederos
- Monitorización de temperatura en rotores de generadores / motores
- Monitorización de presión de ruedas (TPMS) en maquinaria pesada
- Monitorización del estado de estructuras (SHM) in construcción / edificios / infraestructuras
- Monitorización de puntos calientes de centros de transformación eléctrica
- Pago mediante teléfonos móviles o tarjetas
- Control de accesos
- Automoción
- Gestión de paquetería
- Gestión de material sanitario
- Monitorización de pacientes en hospitales

Como vemos los rangos de esta tecnología solo hacen que augmentar, estos son solo algunos casos en los que estas tecnologías ya se están aplicando, y con el IoT (internet of things) solo hará que augmentar, ya que al final todo está más conectado y la información de los productos cuotidianos cada vez nos es más accesible.

Fig. 8

Software de control UHF RFID

Introducción

En todas las aplicaciones que hemos comentado con anterioridad, y como hemos explicado en el apartado sobre el funcionamiento de los sistemas RFID, todos necesitan de un controlador que gestione i se comunique con el lector para obtener la información de los tags.

En este proyecto, nos vamos a centrar en la realización de un software que sea capaz de gestionar varios parámetros de entrada y salida de nuestro lector, y que acceda a la sección de memoria deseada para obtener la información de un sensor adjunto a nuestro tag, y lo grafique.

El diseño se realiza desde 0 y mediante las herramientas aportadas por el fabricante para poder interactuar con el lector de forma correcta.

El lector utilizado es el lector R1260L SLATE RFID UHF Desktop Reader, capaz de detectar los tags que cumplan con el EPC Class1 Gen2.

Caen RFID

CAEN RFID es una compañía líder en el sector de la identificación automática, basando su desarrollo en RAIN RFID (tags pasivos EPC Class1 Gen2)

Fue fundada en Italia en 2006, aunque sus actividades empezaron en 2003 como una división de CAEN SpA.

Caen Proporciona junto a sus hardware, de lectores como tarjetas, un pequeño programa con el que interactuar con el lector y las tarjetas desde un pc mediante un usb, como también una biblioteca de programación para poder realizar programas por el propio usuario y desarrollar nuevas funcionalidades.

R1260I

El lector que vamos a utilizar para desarrollar nuestro pequeño software es el R1260I de la familia easy2read de CAEN, incorpora una antena para aplicaciones de corto y medio alcance.

1. Especificaciones técnicas

Technical Specifications Table	
Frequency Range	– 865.6÷867.6 MHz (ETSI EN 302 208 v3.1.1) (Mod. R1260E, R1260EB) – 902÷928 MHz (FCC part 15.247) (Mod. R1260U, R1260UB) – 920.625÷924.375 MHz (SRRC RFID national standards)
RF Power	Programmable in 15 levels (1dB step) from 4dBm ERP to 18dBm ERP (from 2.5mW ERP to 67mW ERP)
Antenna	Integrated Circular Polarized Antenna
Number of Channels	– 4 channels (compliant to ETSI EN 302 208 v3.1.1) (Mod. R1260E, R1260EB) – 50 hopping channels (compliant to FCC part 15.247) (Mod. R1260U, R1260UB) – 16 hopping channels (compliant to SRRC RFID national standards) All subsets of FCC band are supported via FW upgrade
Standard Compliance	EPC C1 G2/ISO 18000-63
User Interface	Green LED: Power Blinking red LED: Tag detection Blinking yellow LED: USB communication activity Buzzer: user programmable event signalling
Connectivity	USB Type A plug connector Bus powered USB 2.0 device Must be connected to High-power Port (500 mA @ VBUS) It appears as USB serial port Virtual Com Port (VCP) drivers for Windows XP/Vista /Seven (7), Windows CE 4.2, Linux 2.40 and greater – Baudrate: up to 115.200kbps – Databits: 8 – Stopbits: 1 – Parity: none – Flow control: none
Dimensions	(W)297 x (L)205 x (H)15 mm^3 (11.7 x 8 x 0.6 $inch^3$)
Length of USB Cable	1.5 m
DC Power	5 VDC bus powered (USB) Max 400 mA
Operating Temperature	-10°C to +55°C
Weight	525 g

Fig. 9

En la imagen de la página superior, podemos obsrevar las especificaciones técnicas del lector que vamos a utilizar.

El lector es de la empresa CAENRFID y en nuestro caso disponemos del modelo SLAE RFID UHF.

Como podemos observar, trabaja en los rangos de frecuencia del UHF, es decir de ultra frecuencia.

Este Lector puede ser programado en varios niveles de potencia, como de la misma manera también se puede configurar diferentes canales de comunicación con los tags.

Podemos observar que la comunicación con el ordenador se realiza mediante un usb con un protocolo de puerto serie, el cual también puede ser configurado en diferentes niveles de baudios para conseguir la mejor configuración de lectura y de comunicación con el ordenador.

A su vez, se nos indica que el lector tiene solamente una antena circular las dimensiones están en la imagen anterior, así como las características visuales, de tamaño y de alimentación del lector.

En la pagina siguiente podemos ver como el patrón de radiación de la antena no és uniforme, lo que afecta directamente a la lectura de los tags dependiendo de su situación respecto el lector como de su orientación.

2. Patrón de radiación

The radiation patterns of Slate R1260U/UB are shown in the following figures.

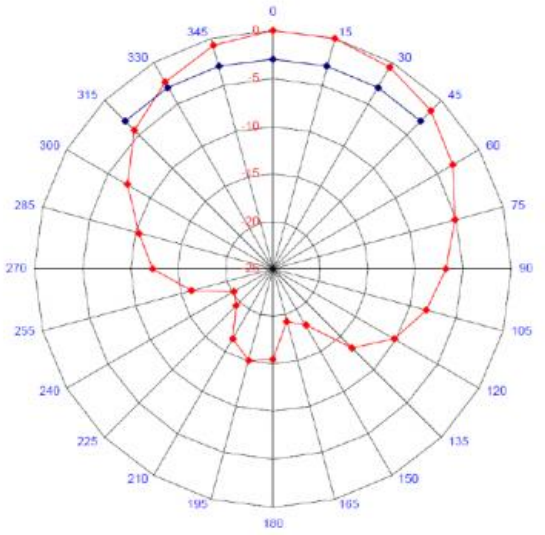
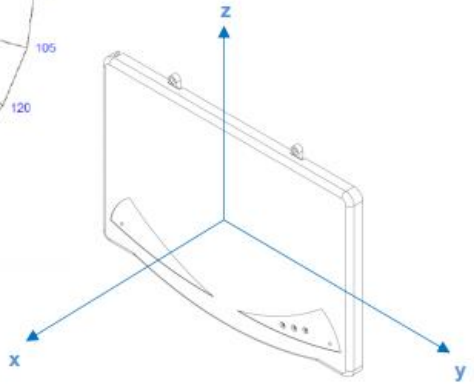


Fig. 5.1: Slate R1260U/UB Radiation pattern H plane



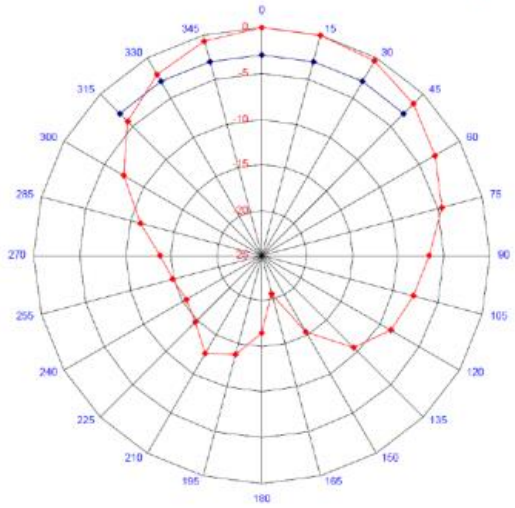


Fig. 5.2: Slate R1260U/UB Radiation pattern V plane

Fig. 10

API

El fabricante del lector utilizado, provee a aquellos que quiera programar un software de control del lector, una biblioteca en diferentes idiomas de programación, el SDK (software development kit). Estas bibliotecas permiten crear una aplicación API.

Los idiomas de programación de esta biblioteca son:

- C
- Java
- C# o Visual Basic.Net
- Android

Los componentes de las diferentes bibliotecas son los mismos para todos lenguajes.

Los archivos que explican que contienen las librerías son gratuitos y están expuestos en la web del fabricante. Estas librerías están en formato .chm, es decir archivos de ayuda de Windows xp.

En los archivos adjuntos de este proyecto añadiremos el archivo de explicación de la biblioteca en formato Word/pdf.

Al no existir ya un soporte para los sistemas xp, la conversión de formato para poder leer la información se ha realizado mediante la web, https://www.aconvert.com/ebook/chm-to-docx/.

De la misma manera proporciona dos archivos extra desde su web denominados CAEN RFID Reference manual y otro Caen RFID API_User Manual.

Esto provoca la perdida de información y la malformación del documento.

El proveedor a la vez proporciona los códigos máquina de comunicación necesaria entre el controlador y el lector en un archivo llamado easy2read_protocol.

Debido a que las bibliotecas básicas de c# no permiten la creación de gráficas en tiempo real, hemos necessitado una librería llamada KayChart, proveniente de una nuget (librerias creadas por usuarios) pública.

Tag

Para este proyecto, el tag que vamos a utilizar es el de la empresa FARSENS, el modelo SPIDER-R, un tag UHF con capacidad para soportar un sensor en su funcionamiento.

SPIDER-R

Como hemos mencionada anteriormente, este es un tag de ufh. Su funcionamiento puede ser tanto passivo, semi-passivo o activo, dependiendo de si se utiliza una batería en forma de pila para su alimentación y de la configuración que hagamos del tag.

El tag consta de una antena, un microprocesador UHF, 12 pines de conexionado externo, y un adaptador para insertar una pila en su parte posterior.

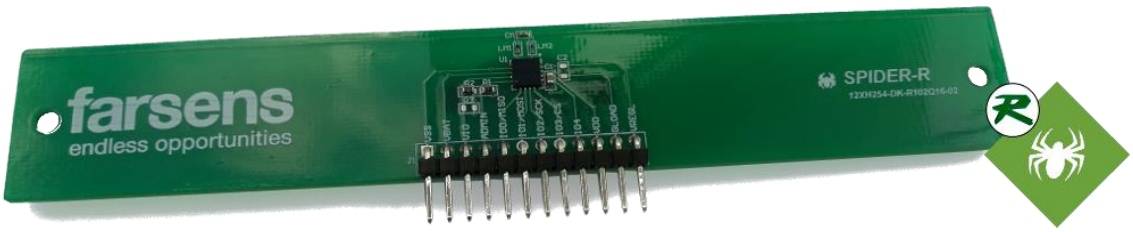


Fig. 11

La parte más importante de este tag, es su microprocesador, ya que es quien procesará la señal del lector y modulará una respuesta a la pregunta efectuada por él lector.

El chip utilizado en este Tag és el ROKY-100.

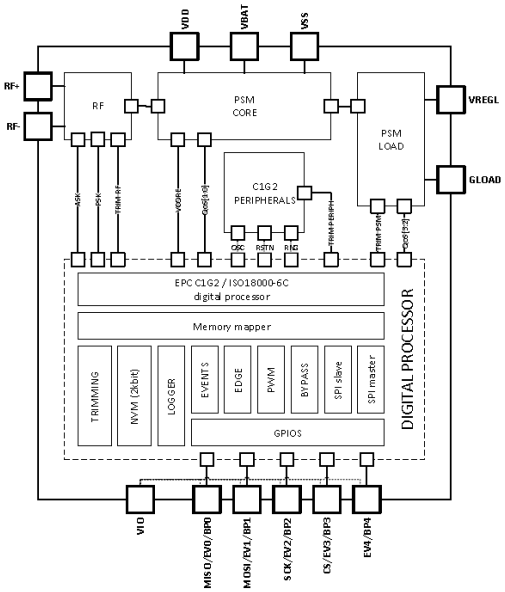


Fig. 12

En la imagen anterior podemos observar la distribución de pines del microcontrolador y su funcionamiento interno

El diagrama de pines es el siguiente:

PIN		TYPE	DESCRIPTION
NAME	NO.		
RF			
RF+	1	RF	Positive input of differential RF signal
RF-	4	RF	Negative input of differential RF signal
Power supply			
VDD	14	Power	Supply voltage of the tag
VSS	5	Power	Ground of the tag
VBAT	6	Power	Battery supply voltage
VREGL	16	Power	Regulated load supply output
GLOAD	15	Power	Switched ground of the load
Digital I/O			
VIO	7	Power	Power supply for the digital I/O pins
RFU	8	Input	Reserved for future use. Connect to VSS.
MISO/EV0/BP0	9	I/O	SPI master input line / Event pin 0 / Bypass pin 0
MOSI/EV1/BP1	10	I/O	SPI master output line / Event pin 1 / Bypass pin 1
SCK/EV2/BP2	11	I/O	SPI clock line / Event pin 2 / Bypass pin 2
CS/EV3/BP3	12	I/O	SPI chip select line / Event pin 3 / Bypass pin 3
EV4/BP4	13	I/O	Event pin 4 / Bypass pin 4

Fig. 13

El sistema de almacenamiento de memoria del microchip viene definido por el protocolo EPC-C1-G2.

Para la parte del sensor, este microchip necesita a activación de la tierra de la carga.

El sensor se coloca entre los pines VREGL y GLOAD

Elección de entrono de programación

El entorno de programación, es el programa o conjuntos de programas utilizados para desarrollar un software. En este entorno el único elemento indispensable es el compilador que traduce el lenguaje escrito por nosotros a código máquina para poder funcionar.

En un primer momento, intentamos realizar este proyecto mediante los programas Matlab y LabView, ya que en varias asignaturas de la carrera son softwares que se han ido utilizando y la idea era de utilizar conocimientos previamente adquiridos.

Al no existir una librería para estos sistemas de programación lo primero que intentamos era mandar las señales con código hexadecimal, tal y como el manual easy2read nos proponía.

Al realizar este paso, no conseguíamos obtener las respuestas esperadas, de hecho, mediante otros programas que se conectaban al mismo lector, sabíamos que habíamos conectado con el lector, però este no respondía a nuestras demandas puesto que necesita de unos pasos obligatorios antes de proceder a la lectura de los tags.

Lo siguiente que intentamos fue mediante la importación de las bibliotecas existentes en los otros idiomas, ya que estos dos entornos permiten hacer esta exportación de librerías.

Este hecho daba problemas entre como estos entornos pedían la información i en como la librería lo necesita para poder funcionar.

Habiendo probado los entornos de programación ya conocidos decidí utilizar el entorno de programación Visual Basic en lenguaje c#, ya que las funciones lógicas y las estructuras de los programas son parecidas al lenguaje c y c++.

Además, este lenguaje de programación cuenta con una gran comunidad activa en internet donde poder encontrar soluciones a los problemas que nos vamos encontrando con nuestro proyecto.

Diseño de la aplicación

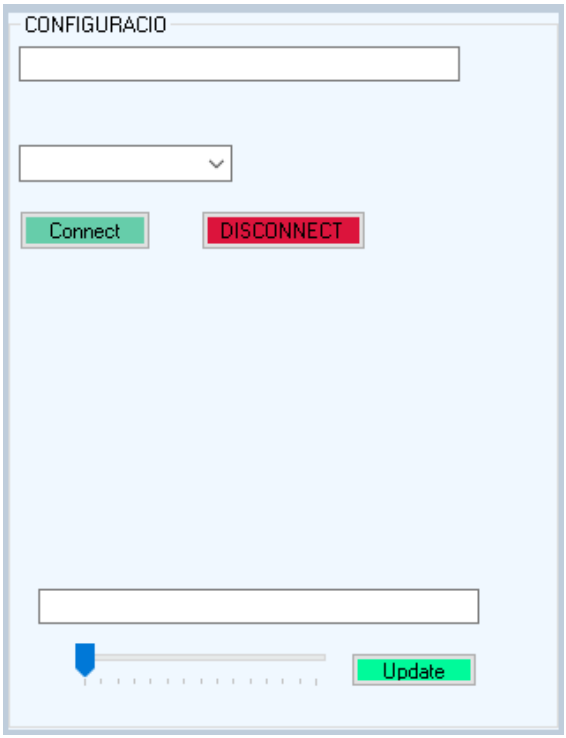
El diseño de nuestra aplicación será lo más sencilla e intuitiva que podamos hacer y se dividirá en dos secciones:

- Configuración y conexión.
- Lectura.

En la parte de conexión y configuración, podemos observar diferentes elementos que nos permiten la selección del puerto serie donde nuestro dispositivo está conectado, los botones de conexión y desconexión para poder evitar solapamientos de aplicaciones y la configuración de potencia de nuestro lector.

En la otra parte, podemos observar la gráfica que se formará a tiempo real, los métodos de lectura, tanto una sola lectura como múltiples lecturas, la información de cuantas veces hemos leído un tag, el valor de su sensor y finalmente el EPC del ultimo tag leído.

Conexión y configuración



En el apartado de conexión y configuración podemos observar los diferentes elementos de izquierda a derecha y de arriba abajo:

- Un cuadro de texto que te guía en lo primero que debes hacer.
- Menú desplegable donde podremos elegir de entre los puertos serie disponible aquel en el que se encuentre nuestro dispositivo.
- Botones para hacer la conexión y desconexión con nuestro lector.
- Cuadro de texto indicando donde podemos configurar la potencia.
- Rollbar para seleccionar potencia junto al botón para actualizarla.

Fig. 14

Obtención de datos

En la siguiente imagen podemos observar los diferentes elementos de nuestro sistema de obtención de datos, procederemos como en el apartado anterior de izquierda a derecha i de arriba abajo.

- Botón que habilita una única lectura del tag.
- Botón que habilita la lectura continua de los tags.
- Botón para para el loop de lectura continua.
- Gráfica que se auto actualiza en tiempo real.
- Cuadro de texto con el número de tags leídos.
- Cuadro de texto con el dato del sensor del tag.
- Cuadro de texto con parte del código EPC.

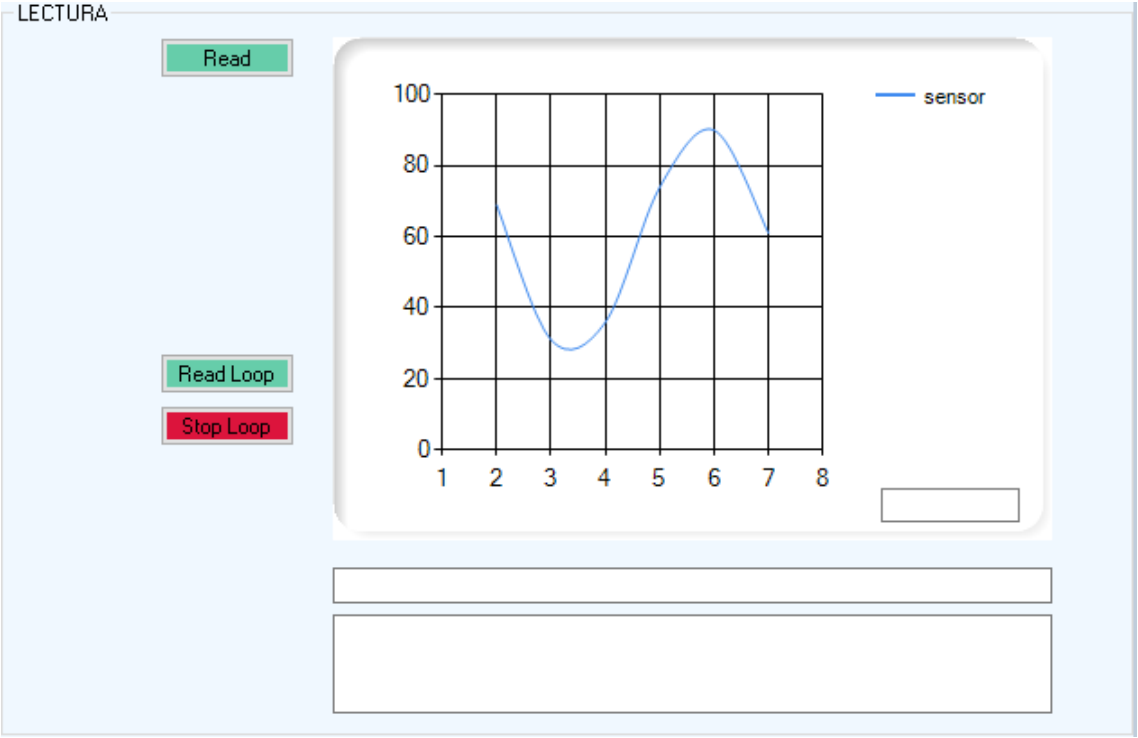


Fig. 15

Funciones utilizadas

Para cada uno de estos apartados, hemos tenido que utilizar el algún punto elementos de las diferentes bibliotecas utilizadas.

Ya sea de las bibliotecas propias del entorno de programación escogido o de las bibliotecas necesarias para el implemento de las lecturas de los tags como de su visualización.

Las bibliotecas utilizadas para este proyecto son las siguientes:

- System;
- System.Collections.Generic;
- System.ComponentModel;
- System.Data;
- System.Drawing;
- System.Linq;
- System.Text;
- System.Windows.Forms;
- com.caen.RFIDLibrary;
- System.Threading;
- System.Threading.Tasks;
- rtChart;
- System.Diagnostics;
- System.IO.Ports;

Funciones utilizadas en la parte de configuración y conexión

Primero debemos buscar en que puerto está conectado nuestro lector, para ello utilizamos la biblioteca System.IO.Ports.

Las instrucciones utilizadas son las siguientes:

```
string[] ports = SerialPort.GetPortNames();

foreach (string puerto in ports)
{
   Cbopuertos.Items.Add(puerto);
   string use = Cbopuertos.Text;
}
```

Esta función busca los puertos serie que pueden ser utilizados y los guarda en forma de array de strings, posteriormente guarda los valores en el desplegable.

Habiendo ya seleccionado el puerto a través del que trabajaremos, debemos realizar la conexión con nuestro lector.

```
private void BtnConnect_Click(object sender, EventArgs e)
      {
          if (b)
          {
              try
              {
                  string use = Cbopuertos.Text;
                  MyReader = new CAENRFIDReader();
                  Mydatat = new IDSTagData();
                  MyReader.Connect(CAENRFIDPort.CAENRFID_RS232, use);
                  textBox1.Text = "Connected";
                  //textBox4.Text = "Set Baude Rate";
                  textBox5.Text = "Set Power";
                  MySource = MyReader.GetSource("Source_0");
                  //MySource.SetQ_EPC_C1G2(3);

                  MySource.SetSession_EPC_C1G2(CAENRFIDLogicalSourceCons
                  tants.EPC_C1G2_SESSION_S1);
                  b = false;
              }
              catch (Exception ex)
              {
                  Console.WriteLine("Selecciona un port");
                  textBox1.Text = "Selecciona un port";

              }
          }
          else
          {
              textBox1.Text = "Already Connected";
          }
       }
```

Como podemos observar en este caso estamos dentro del callback del botón de conexión.

En la primera parte usamos el string previamente seleccionado en el desplegable. Seguidamente, utilizamos la biblioteca com.caen.RFIDLibrary, creamos las clases con la biblioteca e iniciamos la conexión.

Como podemos observar utilizamos las funciones “try & catch” para evitar que en caso de que no se pueda realizar la conexión por algún motivo el programa siga funcionando.

Para realizar las lecturas de los tags utilizamos el siguiente código:

```
String s = BitConverter.ToString(MyTags[i].GetId());
short x = MyTags[i].GetRSSI();
```

Con estas dos líneas de código realizamos la lectura del tag, la parte EPC,

```
double u= Convert.ToDouble(Mydatat.SensorValue);
```

Y con esta leemos el dato del sensor. La diferencia entre una solo lectura

y varias lecturas es simplemente la utilización de un bucle que repite la instrucción

Mostramos por pantalla los datos obtenidos. En este caso diferenciamos entre los cuadros de texto i la gráfica.

```
textBox3.Text = resultst;
```

Esta instrucción relaciona una de nuestras variables con un cuadro de texto.

Para la gráfica utilizaremos la biblioteca rtChart:

```
Task.Factory.StartNew(() =>
          {

             data.updateChart(updateWithdata, 1000);


           });
           PerformanceCounter dada = new PerformanceCounter();
           double updateWithdata()
           {
                return Convert.ToDouble(Mydatat.SensorValue);
           }
```

Debemos utilizar esta librería externa ya que las gráficas que dispone el Visual Studio, son gráficas estáticas, que no se actualizan a medida que se obtienen nuevos datos.

Funcionamiento de la aplicación

Esta aplicación ha sido diseñada para funcionar con un tag, no para hacer lecturas de varios tags a la vez ya que para hacer pruebas solo teníamos un tag con posibilidad de llevar un sensor incorporado.

En todo momento se ha intentado que, aunque las condiciones para implementar la llamada de un objeto no existan el programa no deje de funcionar i de una alerta al usuario.

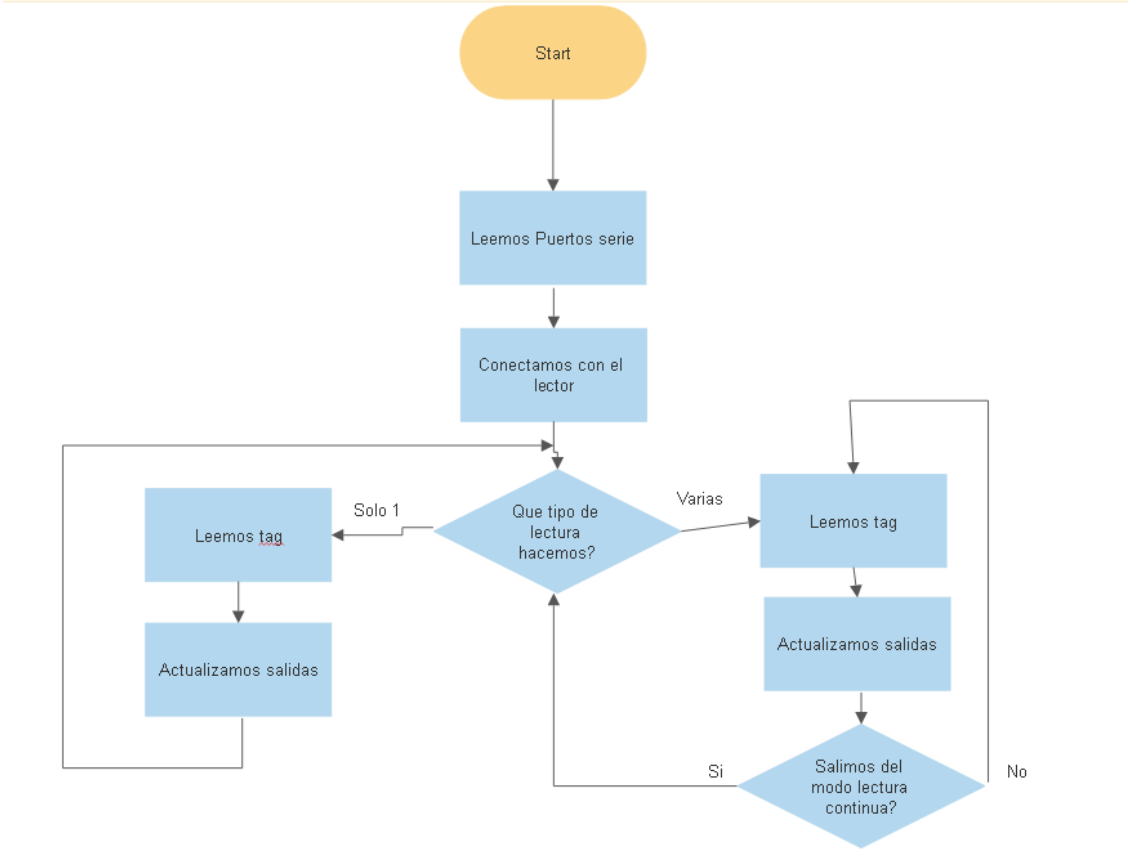


Fig. 16

Resultados

Los resultados obtenidos de la lectura de las tarjetas con nuestro software han variado dependiendo la tarjeta utilizada, una que es capaz de contener un sensor y una que no puede llevar sensor.

Con las dos tarjetas funcionando en modo pasivo la variación en la lectura de ambas tarjetas es considerable.

Es por este motivo por el que no hemos visto a modificar el código de nuestro programa para evitar una lectura excesivamente rápida, ya que provoca un error en las lecturas de forma repetitiva, e impida la correcta lectura de los tags.

Aun siendo una mejor que la otra en cuanto a rendimiento en la lectura del lector, ambas daban error como si no hubiese tarjetas preparadas para la lectura.

En cuanto a la lectura del EPC de las tarjetas hemos visto que las funciones implementadas en las bibliotecas del SDK proporcionado por CAEN, dan la información troceada y no completa, dando directamente el mensaje sin ningún tipo de encabezado en el mensaje.

Este hecho difiere de lo esperado, ya que esperábamos obtener el código completo. Es por este motivo que en el código de nuestro programa viene preparado para poder trocear mensajes hexadecimales y transformarlo en el método de lectura aceptado por los cuadros de texto que son los strings.

La tipología de código esperada en el EPC es la siguiente:

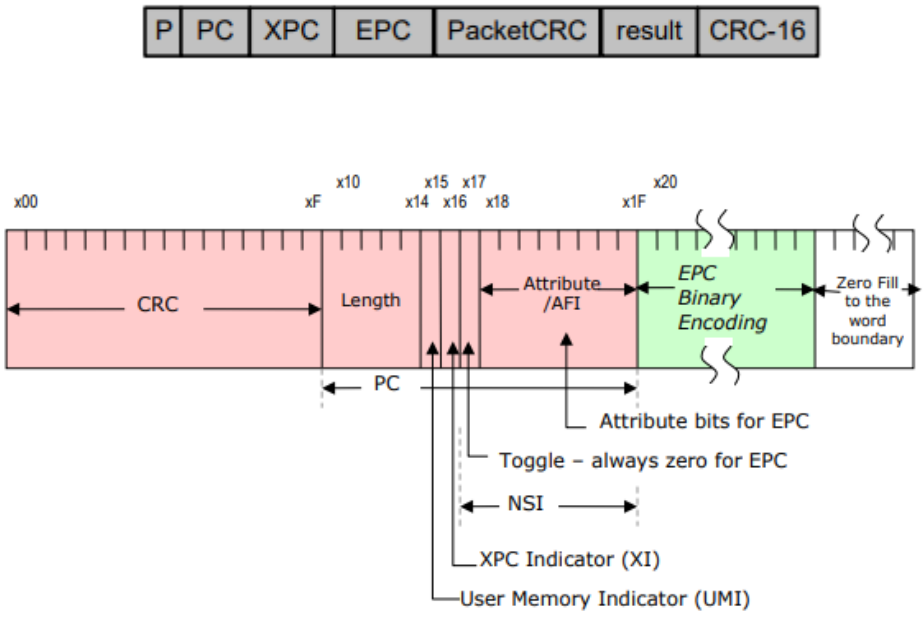


Fig. 17

En su lugar obteníamos simplemente el código hexadecimal EPC:

00112233445566778899AABBCCDDEE

En cuanto a los datos obtenidos por el sensor, este nos viene dado por un integer, al no tener ningún sensor incorporado en la tarjeta el valor obtenido és siempre 0.

Esto afecta directamente a la visualización de la gráfica ya que esta obtiene un valor 0 en todo momento que va por la parte baja de la gráfica y no se aprecia el valor.

Lectura de sensor

FARSENS TAG:

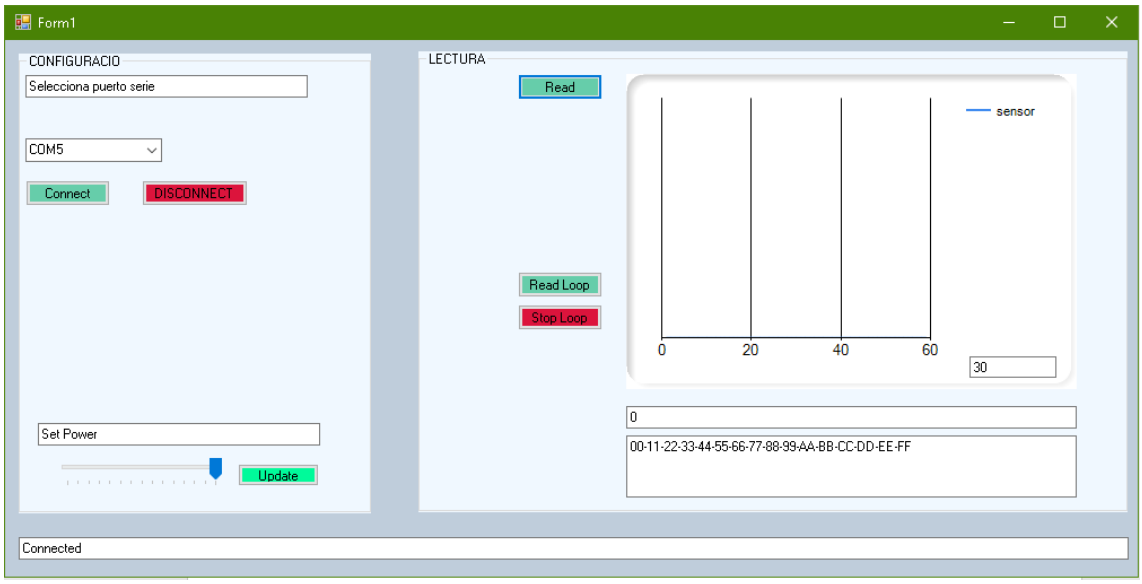


Fig. 18

Como podemos observar en esta imagen el sensor no envía ninguna señal y por configuración del lector este devuelve un 0.

Comparándolo con la imagen posterior, donde utilizamos un tag que no puede llevar sensor, el resultado és el mismo que en el anterior caso.

TAG SIN SENSOR:

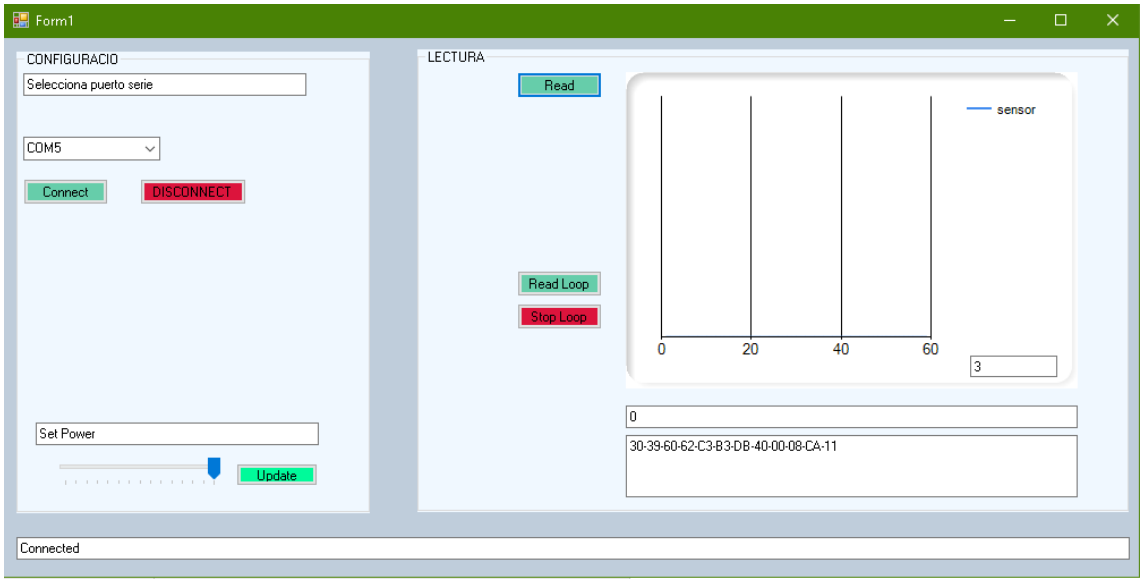


Fig. 19

Como hemos explicado anteriormente, podemos observar como en los dos casos, obtenemos el código EPC de las tarjetas.

Análisis de resultados

Como hemos podido observar, no hemos obtenido ningún resultado en la lectura del sensor incorporado en el tag de FARSENS. Eso es debido a que el lector utilizado está pensado para trabajar con el chip SL900A. Un chip para tarjetas RFID-UHF.

SL900A Block Diagram

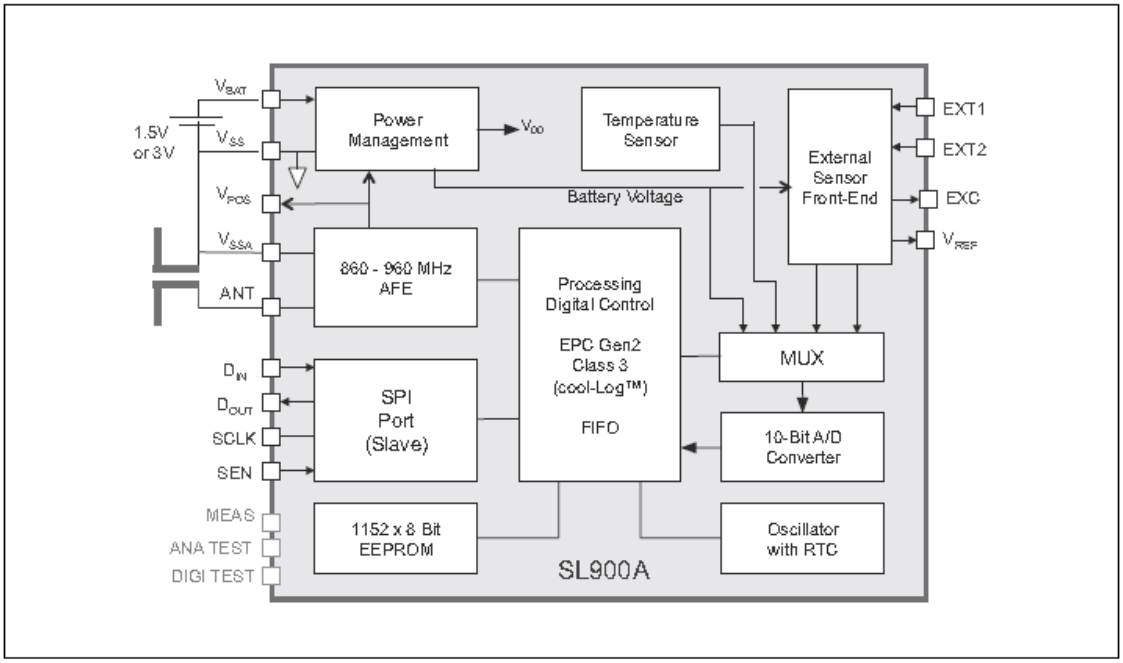


Fig. 20

Como podemos observar la estructura de este chip con el anterior es diferente. Aun siendo los dos chips para trabajar en tarjetas la cantidad de sensores que pueden llevar es diferente, uno contiene un sensor interno i el otro no e incluso uno contiene pines para la configuración analógica de ciertos aspectos y el otro no.

El lector Slate, está pensado para trabajar con tarjetas del mismo fabricante CEANRFID, sus bibliotecas para trabajar con el lector están pensadas para estos casos, no para trabajar con tarjetas de otros fabricantes o basados en chips diferentes al SL900A.

Para poder leer el sensor del chip Rocky-100, es primero necesario habilitar la entrada del micro GLOAD, mediante una comunicación regulada por el protocolo

EPC C1G2. Però esté lector no nos permitía salirnos del las herramientas que usa con sus chips.

Conclusiones y futuro

En este trabajo de fin de grado he realizado una pequeña aplicación que permite interactuar con lector RFID UHF, y a su vez que este comunique adecuadamente con las tarjetas UFH. A su vez esta aplicación permite obtener los datos de los sensores que estas tarjetas pueden llevar incorporados y acceder a su identificador y graficar los resultados de estos sensores.

Esto permite al obtener los datos de los sensores y ver que datos entrega la tarjeta, linealizar el sensor y obtener datos analógicos.

Esto permitiría por ejemplo tener una tarjeta en una habitación y saber la temperatura de esta en todo momento sin necesidad de baterías ni cableados.

Esta aplicación está pensada para seguir en su desarrollo, y no ser una aplicación finalizada. Los siguientes puntos de evolución deberían ser la implementación de arrays de estructuras para guardar la información de varias tarjetas a la vez. Y de esta manera hacer una aplicación de lectura multi tarjetas para simular un entorno controlado por tarjetas RFID por ejemplo.

Otra vía de ampliación o de continuación de este trabajo seria la de repetición del trabajo mediante un lector que permita una programación más libre y no restringida ya que como hemos podido comprobar en este trabajo aun existiendo un protocolo para la comunicación lector tarjeta, un software cerrado en el lector puede impedir trabajar con ciertas herramientas de ciertas tarjetas.

Intentamos la realización de este software mediante otros entornos de programación yendo a un nivel más bajo de programación però el lector no nos permitía trabajar con el mediante este uso. Aun habiendo conseguido conectarnos con el lector, este no nos emitía ninguna respuesta a los inputs que nosotros le ofrecíamos.

Un software y hardware como el mundo de Arduino permitirían la realización de software que permitiera trabajar con más tarjetas.

Durante la realización de este proyecto, he podido comprobar la importancia de realizar una documentación que pueda ser consultada fácilmente por las personas que tengan que trabajar con ella.

Bibliografía

1- EPC™ Radio-Frequency Identity Protocols Generation-2 UHF RFID Specification for RFID Air Interface Protocol for Communications at 860 MHz – 960 MHz Version 2.0.0 Ratified (Online) https://www.gs1.org/sites/default/files/docs/epc/uhfc1g2_2_0_0_standard_20131101.pdf

2- EPC Tag Data Standard (Online) https://www.gs1.org/sites/default/files/docs/epc/GS1_EPC_TDS_i1_10.pdf

3- Protocolos y estándares de UHF RFID (Online) http://trace-id.com/es/protocolos-estandares-uhf-rfid/

5- Empresa Farsens proveedora de tags UHF RFID (Online) http://www.farsens.com

6- Rocky-100 datasheet

7- SLA_900 datasheet

8- EPCglobal, “EPC Tag Data Standards” Differences 2010

9- “ISO-Internationall Organization for Standarization” (Online) https://www.iso.org/home.html

10- “Comunidad C#” https://es.stackoverflow.com/

11- Ayuda de Microsoft par Visual Basic (Online) https://dotnet.microsoft.com/

12- CAEN RFID API Reference Manual_rev_12

13- CAEN RFID API_UserMan_rev_05A

14- easy2read_protocol_Technical Information Manual rev19

Anexo 1

Código principal.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using com.caen.RFIDLibrary;
using System.Threading;
using System.Threading.Tasks;
using rtChart;
using System.Diagnostics;
using System.IO.Ports;

namespace provaRFID
{


    public partial class Form1 : Form
    {
        bool bandera = true;
        bool b = true;
        int Contador = 0;
        private CAENRFIDReader MyReader;
        CAENRFIDLogicalSource MySource;

        public IDSTagData Mydatat { get; private set; }

        public Form1()
        {
            InitializeComponent();
            textBox1.Text = "Disconnected";
            textBox4.Text = "Selecciona puerto serie";
            string[] ports = SerialPort.GetPortNames();
            Cbopuertos.Items.Clear();
            foreach (string puerto in ports)
            {
                Cbopuertos.Items.Add(puerto);
                string use = Cbopuertos.Text;
            }
            if(Cbopuertos.Items.Count>0)
            {

            }
            else
            {

            }
        }

        private void Form1_Load(object sender, EventArgs e)
        {


        }
```

```
        private void Button1_Click(object sender, EventArgs e)
        {

            try
            {
                // MySource.SetReadCycle(1);
                CAENRFIDTag[] MyTags = MySource.InventoryTag();
                if (MyTags.Length > 0)
                {
                    for (int i = 0; i < MyTags.Length; i++)
                    {
                        String s = BitConverter.ToString(MyTags[i].GetId());
                        short x = MyTags[i].GetRSSI();
                        Console.WriteLine(s + " --- " + MyTags[i].GetSource());
                        textBox2.Text = s;
                        int tam_s = s.Length;
                        //String s1 = s.Substring((tam_s - 11), 11);
                        //String s2 = s.Substring((tam_s - 11), 2);
                        //String s3 = s.Substring((tam_s - 8), 2);
                        //String s4 = s.Substring((tam_s - 5), 2);
                        //String s5 = s.Substring((tam_s - 2), 2);
                        double u= Convert.ToDouble(Mydatat.SensorValue);


                        //String St = s2 + s3 + s4 + s5;
                        //int result = int.Parse(St,
System.Globalization.NumberStyles.HexNumber);
                        string resultst = u.ToString();

                        textBox3.Text = resultst;
                        Contador++;
                        string c = Contador.ToString();
                        textBox6.Text = c;
                        Console.WriteLine(x);
                        kayChart data = new kayChart(chart1, 60);
                        data.serieName = "sensor";

                            Task.Factory.StartNew(() =>
                            {

                                 data.updateChart(updateWithdata, 1000);


                            });
                            PerformanceCounter dada = new PerformanceCounter();
                            double updateWithdata()
                            {
                                return Convert.ToDouble(Mydatat.SensorValue);
                            }

                        ;
                    }
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine("ERROR ----------------------");
            }

            Console.WriteLine("Press a key to end the program.");
        }
```

```
        private void Button2_Click(object sender, EventArgs e)
        {
            try
            {
                MyReader.Disconnect();
                textBox1.Text = "Disconnected";
                textBox5.Text = " ";
                b = true;
            }
            catch (Exception ex)
            {
                Console.WriteLine("ja estas desconectat");
                textBox1.Text = "ja estas desconectat";

            }
        }

        private void BtnConnect_Click(object sender, EventArgs e)
        {
            if (b)
            {
                try
                {
                    string use = Cbopuertos.Text;
                    MyReader = new CAENRFIDReader();
                    Mydatat = new IDSTagData();
                    MyReader.Connect(CAENRFIDPort.CAENRFID_RS232, use);
                    textBox1.Text = "Connected";
                    //textBox4.Text = "Set Baude Rate";
                    textBox5.Text = "Set Power";
                    MySource = MyReader.GetSource("Source_0");
                    //MySource.SetQ_EPC_C1G2(3);

MySource.SetSession_EPC_C1G2(CAENRFIDLogicalSourceConstants.EPC_C1G2_SESSION_S1);
                    b = false;
                }
                catch (Exception ex)
                {
                    Console.WriteLine("Selecciona un port");
                    textBox1.Text = "Selecciona un port";

                }
            }
            else
            {
                textBox1.Text = "Already Connected";
            }
        }

        private void ReadLoop_Click(object sender, EventArgs e)
        {

            while (bandera)
            {
                try
                {
                    CAENRFIDTag[] MyTags = MySource.InventoryTag();
                    if (MyTags.Length > 0)
                    {
                        for (int i = 0; i < MyTags.Length; i++)
```

```
                        {
                            String s = BitConverter.ToString(MyTags[i].GetId());
                            short x = MyTags[i].GetRSSI();
                            Console.WriteLine(s + " --- " +
MyTags[i].GetSource());
                            textBox2.Text = s;
                            int tam_s = s.Length;
                            //String s1 = s.Substring((tam_s - 11), 11);
                            //String s2 = s.Substring((tam_s - 11), 2);
                            //String s3 = s.Substring((tam_s - 8), 2);
                            //String s4 = s.Substring((tam_s - 5), 2);
                            //String s5 = s.Substring((tam_s - 2), 2);
                            double u = Convert.ToDouble(Mydatat.SensorValue);

                            //String St = s2 + s3 + s4 + s5;
                            //int result = int.Parse(St,
System.Globalization.NumberStyles.HexNumber);
                            string resultst = u.ToString();

                            textBox3.Text = resultst;
                            Contador++;
                            string c = Contador.ToString();
                            textBox6.Text = c;
                            Console.WriteLine(x);
                            kayChart data = new kayChart(chart1, 60);
                            data.serieName = "sensor";

                            Task.Factory.StartNew(() =>
                            {

                                data.updateChart(updateWithdata, 1000);


                            });
                            PerformanceCounter dada = new PerformanceCounter();
                            double updateWithdata()
                            {
                                return Convert.ToDouble(Mydatat.SensorValue);
                            }
                            Application.DoEvents();

                        }
                    }
                }

                catch (Exception ex)
                {
                    Console.WriteLine("ERROR ----------------------");

                }
                System.Threading.Thread.Sleep(500);
            }
            bandera = true;
        }

        private void Chart1_Click(object sender, EventArgs e)
        {

        }

        private void TextBox3_TextChanged(object sender, EventArgs e)
        {
```

```
}

private void TextBox2_TextChanged(object sender, EventArgs e)
{

}



private void TextBox5_TextChanged(object sender, EventArgs e)
{

}


private void TrackBar1_Scroll(object sender, EventArgs e)
{

}

private void TextBox6_TextChanged(object sender, EventArgs e)
{

}


private void Button4_Click(object sender, EventArgs e)
{
    try
    {
        int power;
        power = trackBar1.Value;
        MyReader.SetPower(power);
    }
    catch (Exception ex)
    {
        Console.WriteLine("Estas conectat?");
        textBox1.Text = "Estas conectat ?";

    }

}

private void Button5_Click(object sender, EventArgs err)
{
    bandera = false;
}

private void TextBox4_TextChanged(object sender, EventArgs e)
{

}

private void ComboBox1_SelectedIndexChanged(object sender, EventArgs e)
{

}

private void TextBox1_TextChanged(object sender, EventArgs e)
{
```

```
        }

        private void GroupBox1_Enter(object sender, EventArgs e)
        {

        }

        private void TextBox6_TextChanged_1(object sender, EventArgs e)
        {

        }
    }
}
```

Anexo 2

Código UI.

```
namespace provaRFID
{
    partial class Form1
    {
        /// <summary>
        /// Variable del diseñador necesaria.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Limpiar los recursos que se estén usando.
        /// </summary>
        /// <param name="disposing">true si los recursos administrados se deben desechar; false en caso contrario.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Código generado por el Diseñador de Windows Forms

        /// <summary>
        /// Método necesario para admitir el Diseñador. No se puede modificar
        /// el contenido de este método con el editor de código.
```

```
        /// </summary>
        private void InitializeComponent()
        {
            this.components = new System.ComponentModel.Container();
            System.Windows.Forms.GroupBox groupBox2;
            System.Windows.Forms.DataVisualization.Charting.ChartArea
chartArea1 = new
System.Windows.Forms.DataVisualization.Charting.ChartArea();
            System.Windows.Forms.DataVisualization.Charting.Legend legend1 =
new System.Windows.Forms.DataVisualization.Charting.Legend();
            System.Windows.Forms.DataVisualization.Charting.Series series1 =
new System.Windows.Forms.DataVisualization.Charting.Series();
            this.textBox6 = new System.Windows.Forms.TextBox();
            this.button5 = new System.Windows.Forms.Button();
            this.textBox3 = new System.Windows.Forms.TextBox();
            this.textBox2 = new System.Windows.Forms.TextBox();
            this.chart1 = new
System.Windows.Forms.DataVisualization.Charting.Chart();
            this.ReadLoop = new System.Windows.Forms.Button();
            this.button1 = new System.Windows.Forms.Button();
            this.button2 = new System.Windows.Forms.Button();
            this.textBox1 = new System.Windows.Forms.TextBox();
            this.btnConnect = new System.Windows.Forms.Button();
            this.trackBar1 = new System.Windows.Forms.TrackBar();
            this.textBox5 = new System.Windows.Forms.TextBox();
            this.button4 = new System.Windows.Forms.Button();
            this.Cbopuertos = new System.Windows.Forms.ComboBox();
            this.textBox4 = new System.Windows.Forms.TextBox();
            this.SpPuertos = new System.IO.Ports.SerialPort(this.components);
            this.groupBox1 = new System.Windows.Forms.GroupBox();
            groupBox2 = new System.Windows.Forms.GroupBox();
            groupBox2.SuspendLayout();
```

```
        ((System.ComponentModel.ISupportInitialize)(this.chart1)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.trackBar1)).BeginInit();
        this.groupBox1.SuspendLayout();
        this.SuspendLayout();
        //
        // groupBox2
        //
        groupBox2.BackColor = System.Drawing.Color.AliceBlue;
        groupBox2.Controls.Add(this.textBox6);
        groupBox2.Controls.Add(this.button5);
        groupBox2.Controls.Add(this.textBox3);
        groupBox2.Controls.Add(this.textBox2);
        groupBox2.Controls.Add(this.chart1);
        groupBox2.Controls.Add(this.ReadLoop);
        groupBox2.Controls.Add(this.button1);
        groupBox2.Location = new System.Drawing.Point(366, 10);
        groupBox2.Name = "groupBox2";
        groupBox2.Size = new System.Drawing.Size(628, 408);
        groupBox2.TabIndex = 1005;
        groupBox2.TabStop = false;
        groupBox2.Text = "LECTURA";
        //
        // textBox6
        //
        this.textBox6.Location = new System.Drawing.Point(488, 270);
        this.textBox6.Multiline = true;
        this.textBox6.Name = "textBox6";
        this.textBox6.Size = new System.Drawing.Size(77, 19);
        this.textBox6.TabIndex = 1002;
```

```
        this.textBox6.TextChanged += new
System.EventHandler(this.TextBox6_TextChanged_1);
        //
        // button5
        //
        this.button5.AllowDrop = true;
        this.button5.BackColor = System.Drawing.Color.Crimson;
        this.button5.CausesValidation = false;
        this.button5.FlatAppearance.MouseDownBackColor =
System.Drawing.Color.White;
        this.button5.Location = new System.Drawing.Point(88, 224);
        this.button5.Name = "button5";
        this.button5.Size = new System.Drawing.Size(75, 23);
        this.button5.TabIndex = 1001;
        this.button5.Text = "Stop Loop";
        this.button5.UseVisualStyleBackColor = false;
        this.button5.Click += new System.EventHandler(this.Button5_Click);
        //
        // textBox3
        //
        this.textBox3.Location = new System.Drawing.Point(184, 314);
        this.textBox3.Multiline = true;
        this.textBox3.Name = "textBox3";
        this.textBox3.Size = new System.Drawing.Size(399, 20);
        this.textBox3.TabIndex = 7;
        this.textBox3.TextChanged += new
System.EventHandler(this.TextBox3_TextChanged);
        //
        // textBox2
        //
        this.textBox2.Location = new System.Drawing.Point(184, 340);
        this.textBox2.Multiline = true;
```

```
            this.textBox2.Name = "textBox2";
            this.textBox2.Size = new System.Drawing.Size(399, 55);
            this.textBox2.TabIndex = 6;
            this.textBox2.TextChanged += new
System.EventHandler(this.TextBox2_TextChanged);
            //
            // chart1
            //
            this.chart1.BackgroundImageLayout =
System.Windows.Forms.ImageLayout.None;
            this.chart1.BackImageTransparentColor = System.Drawing.Color.White;
            this.chart1.BackSecondaryColor = System.Drawing.Color.White;
            this.chart1.BorderSkin.BackColor =
System.Drawing.Color.LightSteelBlue;
            this.chart1.BorderSkin.BackSecondaryColor =
System.Drawing.Color.White;
            this.chart1.BorderSkin.SkinStyle =
System.Windows.Forms.DataVisualization.Charting.BorderSkinStyle.Sunken;
            chartArea1.Name = "ChartArea1";
            this.chart1.ChartAreas.Add(chartArea1);
            legend1.Name = "Legend1";
            this.chart1.Legends.Add(legend1);
            this.chart1.Location = new System.Drawing.Point(184, 20);
            this.chart1.Name = "chart1";
            this.chart1.Padding = new System.Windows.Forms.Padding(1);
            series1.ChartArea = "ChartArea1";
            series1.ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Spline;
            series1.Legend = "Legend1";
            series1.Name = "sensor";
            this.chart1.Series.Add(series1);
            this.chart1.Size = new System.Drawing.Size(399, 279);
            this.chart1.TabIndex = 5;
```

```
            this.chart1.Text = "chart1";
            this.chart1.Click += new System.EventHandler(this.Chart1_Click);
            //
            // ReadLoop
            //
            this.ReadLoop.BackColor = System.Drawing.Color.MediumAquamarine;
            this.ReadLoop.Location = new System.Drawing.Point(88, 195);
            this.ReadLoop.Name = "ReadLoop";
            this.ReadLoop.Size = new System.Drawing.Size(75, 23);
            this.ReadLoop.TabIndex = 4;
            this.ReadLoop.Text = "Read Loop";
            this.ReadLoop.UseVisualStyleBackColor = false;
            this.ReadLoop.Click += new
System.EventHandler(this.ReadLoop_Click);
            //
            // button1
            //
            this.button1.BackColor = System.Drawing.Color.MediumAquamarine;
            this.button1.ForeColor = System.Drawing.SystemColors.ControlText;
            this.button1.Location = new System.Drawing.Point(88, 20);
            this.button1.Name = "button1";
            this.button1.Size = new System.Drawing.Size(75, 23);
            this.button1.TabIndex = 0;
            this.button1.Text = "Read";
            this.button1.UseVisualStyleBackColor = false;
            this.button1.Click += new System.EventHandler(this.Button1_Click);
            //
            // button2
            //
            this.button2.BackColor = System.Drawing.Color.Crimson;
            this.button2.Location = new System.Drawing.Point(109, 112);
```

```
            this.button2.Name = "button2";
            this.button2.Size = new System.Drawing.Size(94, 23);
            this.button2.TabIndex = 1;
            this.button2.Text = "DISCONNECT";
            this.button2.UseVisualStyleBackColor = false;
            this.button2.Click += new System.EventHandler(this.Button2_Click);
            //
            // textBox1
            //
            this.textBox1.Location = new System.Drawing.Point(12, 440);
            this.textBox1.Name = "textBox1";
            this.textBox1.ScrollBars =
System.Windows.Forms.ScrollBars.Horizontal;
            this.textBox1.Size = new System.Drawing.Size(983, 20);
            this.textBox1.TabIndex = 2;
            this.textBox1.TextChanged += new
System.EventHandler(this.TextBox1_TextChanged);
            //
            // btnConnect
            //
            this.btnConnect.BackColor =
System.Drawing.Color.MediumAquamarine;
            this.btnConnect.Location = new System.Drawing.Point(6, 112);
            this.btnConnect.Name = "btnConnect";
            this.btnConnect.Size = new System.Drawing.Size(75, 23);
            this.btnConnect.TabIndex = 3;
            this.btnConnect.Text = "Connect";
            this.btnConnect.UseVisualStyleBackColor = false;
            this.btnConnect.Click += new
System.EventHandler(this.BtnConnect_Click);
            //
            // trackBar1
```

```
            //
            this.trackBar1.Location = new System.Drawing.Point(30, 356);
            this.trackBar1.Maximum = 18;
            this.trackBar1.Minimum = 4;
            this.trackBar1.Name = "trackBar1";
            this.trackBar1.Size = new System.Drawing.Size(158, 45);
            this.trackBar1.TabIndex = 8;
            this.trackBar1.Value = 4;
            this.trackBar1.Scroll += new
System.EventHandler(this.TrackBar1_Scroll);
            //
            // textBox5
            //
            this.textBox5.Location = new System.Drawing.Point(17, 327);
            this.textBox5.Name = "textBox5";
            this.textBox5.Size = new System.Drawing.Size(250, 20);
            this.textBox5.TabIndex = 13;
            this.textBox5.TextChanged += new
System.EventHandler(this.TextBox5_TextChanged);
            //
            // button4
            //
            this.button4.BackColor = System.Drawing.Color.MediumSpringGreen;
            this.button4.Location = new System.Drawing.Point(194, 363);
            this.button4.Name = "button4";
            this.button4.Size = new System.Drawing.Size(72, 20);
            this.button4.TabIndex = 15;
            this.button4.Text = "Update";
            this.button4.UseVisualStyleBackColor = false;
            this.button4.Click += new System.EventHandler(this.Button4_Click);
            //
```

```
        // Cbopuertos
        //
        this.Cbopuertos.FormattingEnabled = true;
        this.Cbopuertos.Location = new System.Drawing.Point(6, 75);
        this.Cbopuertos.Name = "Cbopuertos";
        this.Cbopuertos.Size = new System.Drawing.Size(121, 21);
        this.Cbopuertos.TabIndex = 1002;
        this.Cbopuertos.SelectedIndexChanged += new
System.EventHandler(this.ComboBox1_SelectedIndexChanged);
        //
        // textBox4
        //
        this.textBox4.Location = new System.Drawing.Point(6, 19);
        this.textBox4.Name = "textBox4";
        this.textBox4.Size = new System.Drawing.Size(250, 20);
        this.textBox4.TabIndex = 1003;
        this.textBox4.TextChanged += new
System.EventHandler(this.TextBox4_TextChanged);
        //
        // groupBox1
        //
        this.groupBox1.BackColor = System.Drawing.Color.AliceBlue;
        this.groupBox1.Controls.Add(this.textBox4);
        this.groupBox1.Controls.Add(this.Cbopuertos);
        this.groupBox1.Controls.Add(this.trackBar1);
        this.groupBox1.Controls.Add(this.button4);
        this.groupBox1.Controls.Add(this.textBox5);
        this.groupBox1.Controls.Add(this.btnConnect);
        this.groupBox1.Controls.Add(this.button2);
        this.groupBox1.Location = new System.Drawing.Point(12, 12);
        this.groupBox1.Name = "groupBox1";
```

```
            this.groupBox1.Size = new System.Drawing.Size(312, 407);
            this.groupBox1.TabIndex = 1004;
            this.groupBox1.TabStop = false;
            this.groupBox1.Text = "CONFIGURACIO";
            this.groupBox1.Enter += new
System.EventHandler(this.GroupBox1_Enter);
            //
            // Form1
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.BackColor = System.Drawing.SystemColors.InactiveCaption;
            this.ClientSize = new System.Drawing.Size(1003, 475);
            this.Controls.Add(groupBox2);
            this.Controls.Add(this.groupBox1);
            this.Controls.Add(this.textBox1);
            this.Name = "Form1";
            this.Text = "Form1";
            this.Load += new System.EventHandler(this.Form1_Load);
            groupBox2.ResumeLayout(false);
            groupBox2.PerformLayout();
            ((System.ComponentModel.ISupportInitialize)(this.chart1)).EndInit();
            ((System.ComponentModel.ISupportInitialize)(this.trackBar1)).EndInit();
            this.groupBox1.ResumeLayout(false);
            this.groupBox1.PerformLayout();
            this.ResumeLayout(false);
            this.PerformLayout();

        }

        #endregion
```

```
        private System.Windows.Forms.Button button1;
        private System.Windows.Forms.Button button2;
        private System.Windows.Forms.TextBox textBox1;
        private System.Windows.Forms.Button btnConnect;
        private System.Windows.Forms.Button ReadLoop;
        private System.Windows.Forms.DataVisualization.Charting.Chart chart1;
        public System.Windows.Forms.TextBox textBox2;
        private System.Windows.Forms.TextBox textBox3;
        private System.Windows.Forms.TrackBar trackBar1;
        private System.Windows.Forms.TextBox textBox5;
        private System.Windows.Forms.Button button4;
        private System.Windows.Forms.Button button5;
        private System.Windows.Forms.ComboBox Cbopuertos;
        private System.Windows.Forms.TextBox textBox4;
        private System.IO.Ports.SerialPort SpPuertos;
        private System.Windows.Forms.GroupBox groupBox1;
        private System.Windows.Forms.TextBox textBox6;
    }
}
```